OMAP2420 Multimedia Processor

(Texas Instruments OMAP™ Family of Products)

Technical Reference Manual

Literature Number: SWPU107
May 2007

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI 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 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. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
Low Power Wireless	www.ti.com/lpw	Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

Preface

Read This First

About This Manual

This manual describes the OMAP2420 ES2.1.1 high-performance multimedia device.

How to Use This Manual

This document contains the following chapters:

Chapter 1: Introduction

This chapter introduces the basic functionality and essential components of the OMAP2420 high-performance multimedia device.

Chapter 2: Memory Mapping

This chapter describes memory mapping on the OMAP2420 multimedia device.

Chapter 3: MPU Subsystem

This chapter describes the features and functions of the OMAP2420 digital signal processor (DSP) microprocessor unit (MPU) subsystem and provides a programming model of the subsystem.

Chapter 4: DSP Subsystem

This chapter describes the features and integration of the OMAP2420 digital signal processor (DSP) subsystem and provides a programming model of the subsystem.

Chapter 5: Power, Reset, and Clock Management

This chapter discusses power reset and clock management (PRCM) on the OMAP2420 multimedia device.

Chapter 6: Internal Interconnect

This chapter describes the OMAP2420 device internal interconnect.

Chapter 7: Interprocessor Communication

This chapter discusses interprocessor communication (IPC) between the on-chip processor of the OMAP2420 device.

Chapter 8: System Control Module

This chapter describes the system control module (SCM) on the OMAP2420 multimedia device.

Chapter 9: Memory Management Unit

This chapter describes the memory management unit (MMU) devices in the OMAP2420 multimedia device.

Chapter 10: DMA

This chapter describes the generic DMA module on OMAP2420 that is used in the DSP subsystem (DDMA), camera subsystem (CamDMA), and system DMA (sDMA).

Chapter 11: Interrupt Controller

This chapter describes the interrupt controller (INTC) module used in the microprocessor unit (MPU) and digital signal processor (DSP) subsystems of the OMAP2420 device.

Chapter 12: Memory Subsystem

This chapter describes the memory subsytem on the OMAP2420 device, including the general-purpose memory controller (GPMC), SDRAM controller subsystem, and on-chip memory (OCM) subsystem.

Chapter 13: Enhanced Audio Controller

This chapter describes the enhanced audio controller (EAC) device in the OMAP2420 multimedia device.

Chapter 14: Camera Subsystem

This chapter describes the camera subsystem that provides the system interface and functionality to connect image sensor modules to the OMAP2420 device.

Chapter 15: Display Subsystem

This chapter describes the display subsystem of the OMAP2420 multimedia device.

Chapter 16: Timers

This chapter discusses several types of timers on the OMAP2420 device for use by system software.

Chapter 17: UART/IrDA/CIR

This chapter discusses the three UART devices on the OMAP2420 multimedia device.

Chapter 18: I^{2}C

This chapter describes the inter-integrated circuit I^{2}C modules on the OMAP2420 multimedia device.

Chapter 19: Multichannel SPI

This chapter describes the two multichannel SPI (McSPI) modules on the OMAP2420 device.

Chapter 20: HDQ/1-Wire

This chapter describes the features and functions of the HDQ/1-Wire module on the OMAP2420 device.

Chapter 21: Multichannel Buffered Serial Port

This chapter discusses the multichannel buffered serial port (McBSP) on the OMAP2420 device.

Chapter 22: Universal Serial Bus

This chapter describes the three universal serial bus (USB) ports and several varieties of USB functionality on the OMAP2420 device.

Chapter 23: General-Purpose Interface

This chapter discusses the four general-purpose input/output (GPIO) modules that form the general-purpose interface in the OMAP2420 device.

Chapter 24: Pinout Overview

This chapter provides an overview of the pinout on the OMAP2420 high-performance multimedia device.

Chapter 25: Device Booting

This chapter describes the high-level booting concepts on the OMAP2420 multimedia device and provides basic knowledge of booting on the device.

Chapter 26: Initialization

This chapter provides an overview of the requirements and process for initializing an OMAP2420 device and the boot ROM operational requirements and behavior expectations.

Appendix A

Application Notes References

Appendix B

Glossary

Appendix C

Index

Notational Conventions

This document uses the following conventions.

- Program listings, program examples, and interactive displays are shown in a `special typeface` similar to a typewriter's. Examples use a **`bold version`** of the special typeface for emphasis; interactive displays use a **`bold version`** of the special typeface to distinguish commands that you enter from items that the system displays (such as prompts, command output, error messages, etc.).

  Here is a sample program listing:

  ```
  0011  0005  0001         .field    1, 2
  0012  0005  0003         .field    3, 4
  0013  0005  0006         .field    6, 3
  0014  0006               .even
  ```

  Here is an example of a system prompt and a command that you might enter:

  ```
  C:  csr -a /user/ti/simuboard/utilities
  ```

- In syntax descriptions, the instruction, command, or directive is in a **bold typeface** font and parameters are in an *italic typeface*. Portions of a syntax that are in **bold** should be entered as shown; portions of a syntax that are in *italics* describe the type of information that should be entered. Here is an example of a directive syntax:

  .asect ”*section name*”**,** *address*

  .asect is the directive. This directive has two parameters, indicated by *section name* and *address*. When you use .asect, the first parameter must be an actual section name, enclosed in double quotes; the second parameter must be an address.

- Square brackets (**[** and **]**) identify an optional parameter. If you use an optional parameter, you specify the information within the brackets; you don't enter the brackets themselves. Here's an example of an instruction that has an optional parameter:

  LALK *16-bit constant [, shift]*

  The LALK instruction has two parameters. The first parameter, *16-bit constant*, is required. The second parameter, *shift*, is optional. As this syntax shows, if you use the optional second parameter, you must precede it with a comma.

  Square brackets are also used as part of the pathname specification for VMS pathnames; in this case, the brackets are actually part of the pathname (they are not optional).

- Braces ({ and }) indicate a list. The symbol **|** (read as *or*) separates items within the list. Here's an example of a list:

  ```
  { * | *+ | *- }
  ```

  This provides three choices: `*`, `*+`, or `*-`.

Unless the list is enclosed in square brackets, you must choose one item from the list.

- Some directives can have a varying number of parameters. For example, the .byte directive can have up to 100 parameters. The syntax for this directive is:

  .byte *value*$_1$ *[, ... , value*$_n$*]*

  This syntax shows that .byte must have at least one value parameter, but you have the option of supplying additional value parameters, separated by commas.

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.

Warning:

Recipient agrees to not knowingly export or re-export, directly or indirectly, any product or technical data (as defined by the U.S., EU and other Export Administration Regulations) including software, or any controlled product restricted by other applicable national regulations, received from Disclosing party under this Agreement, or any direct product of such technology, to any destination to which such export or re-export is restricted or prohibited by U.S. or other applicable laws, without obtaining prior authorization from U.S. Department of Commerce and other competent Government authorities to the extent required by those laws. This provision shall survive termination or expiration of this Agreement.

According to our best knowledge of the state and end-use of this product or technology, and in compliance with the export control regulations of dual-use goods in force in the origin and exporting countries, this technology is classified as follows:

–US ECCN: 5E002
–EU ECCN: 5E002

and may require export or re-export license for shipping it in compliance with the applicable regulations of certain countries.

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.

Trademarks

OMAP, TMS320C55x, C55x are trademarks of Texas Instruments Inc.

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

HDQ is a trademark of Benchmarq.

Windows is a trademark of Microsoft.

1-Wire is a trademark of National Semiconductor.

POWERVR

The PowerVR technology logo is a trademark of Imagination Technologies Limited.

Other trademarks and trade names are those of their respective owners.

If You Need Assistance. . .

If you want to. . .	**Do this. . .**
Request more information about Texas Instruments Digital Signal Processing (DSP) products	Call the CRC[†] 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[†] hotline: **(800) 336–5236**
Ask questions about product operation or report suspected problems	Call the DSP hotline: **(713) 274–2320**
Report mistakes in this document or any other TI documentation	Fill out and return the reader response card at the end of this book, or send your comments to: Texas Instruments Incorporated Technical Publications Manager, MS 702 P.O. Box 1443 Houston, Texas 77251–1443

[†] Texas Instruments Customer Response Center

Survey

Help us meet your expectations:

We are always looking at ways to develop our service and improve our quality to fit your needs.

So, please take a few minutes to complete our short questionnaire by

- ☐ Providing general suggestions.

Or

- ☐ Rating a document and describing its critical points.

http://www.ti.com/csdocsurvey (password: 123survey)

Thank you.

History

Version	Literature Number	Date	Notes
*	SWPU107	May 2007	1

Notes:

1) First release of open source version.

Contents

12 Memory Subsystem 12-1

Figures

Tables

Equations

Chapter 1

Introduction

This chapter introduces the features, supporting subsystems, and architecture of the OMAP2420 high-performance multimedia application device.

Topic **Page**

1.1 Overview

The OMAP2420 high-performance multimedia application device is based on the enhanced OMAP™ 2.0 architecture and is integrated on TI's advanced 90-nm process technology.

The OMAP2420 delivers low-power enhanced video imaging, audio, and graphics processing to support multiple high-quality multimedia features:

- Streaming video
- Video conferencing
- Video recording
- Audio
- High-resolution still-image capture (used in 2.5G and 3G wireless terminals and high-performance personal digital assistants [PDAs])

The following subsystems support the OMAP2420:

- Microprocessor unit (MPU) subsystem based on the ARM1136JF-S microprocessor
- Digital signal processor (DSP) subsystem based on the industry-leading TMS320C55x™ (C55x™) DSP, which supports a wide range of audio and video applications
- Imaging and video accelerator subsystem with an MPU and imaging accelerators to enable high-end imaging and video applications
- Camera subsystem 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 National Television System Committee (NTSC)/phase alternation line (PAL) video output.
- Level 3 (L3) and level 4 (L4) interconnects that provide high-bandwidth data transfer for multiple initiators to internal and external memory controllers and to on-chip peripherals

The OMAP2420 supports high-level operating systems, such as the following:

- Windows CE
- Symbian OS
- Linux
- Palm OS

Texas Instruments and multiple third parties offer a rich and comprehensive set of video, still picture, and audio code/decode (codec) devices.

The OMAP2420 also offers the following features:

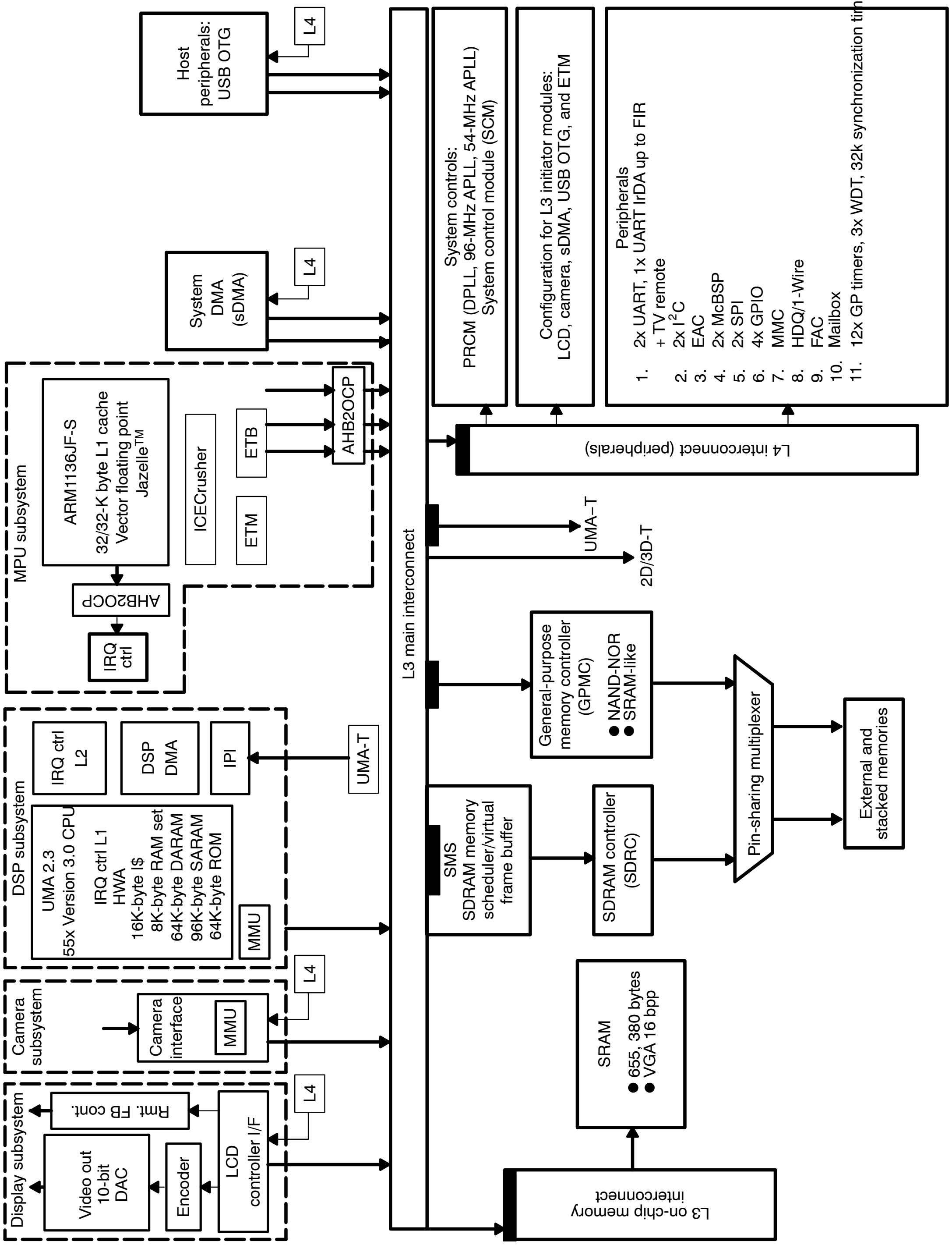
- Comprehensive power and clock-management scheme enabling high-performance, low-power operation, and ultralow-power standby features
- Connection to TI 2.5G or 3G modem chipsets for a complete terminal solution
- Core operating voltage range from low-voltage point (not supported before ES2.2) to high-voltage point nominal and an interface specified at 1.8 V nominal

1.2 OMAP2420 Description

The OMAP2420 is offered in the 325-ball 12 x 12 mm 0.5 mm pitch package.

Figure 1−1 is a block diagram of the OMAP2420.

Figure 1−1. OMAP2420 Block Diagram



1.2.1 MPU Subsystem (Based on ARM1136JF-S)

The MPU general-purpose processor consists of the following components:

- V6 instruction set architecture (ISA)
- Caches
  - 32K-byte instruction and 32K-byte data; 4-way set associative memory management units (MMUs)
  - 64-entry instruction and 64-entry data write buffer
- Vector floating-point processor
- Jazelle Java accelerator
- ICECrusher/end-of-transmission block embedded trace buffer (ETB)/embedded trace macrocell (ETM) for emulation and trace

1.2.2 TMS320C55x DSP Subsystem

The general-purpose fixed-point C55x DSP includes the following features:

- Revision 3.0 core with reduced power consumption and improved performance based on higher parallelism and new instructions
- 16K-byte 2-way set associative instruction cache
- Two 4K-byte RAM sets
- 64K-byte dual-access SRAM
- 96K-byte single-access SRAM
- 64K-byte ROM
- Multichannel direct memory access controller (dDMA):
  - 1 read port, 1 write port
  - 24 logical channels
  - 32 hardware requests
  - 128- x 64-bit FIFO depth
- Video decoder/encoder hardware accelerators

1.2.3 On-Chip Memory

The on-chip memory (OCM) configuration offers secure and nonsecure memory resources for general-purpose program and data storage:

- 96K-byte ROM
- 640K-byte single-access SRAM

1.2.4 External Memory Interfaces

The OMAP2420 includes the following external memory interfaces:

- General-purpose memory controller (GPMC)
  - NOR flash, NAND flash, SRAM, and PSRAM asynchronous and synchronous protocols
  - Flexible asynchronous protocol control for external ASIC or peripheral interfacing
  - 16-bit data, up to eight chip-selects, 128M-byte address bus, 1G-byte total address space
- SDRAM controller (SDRC)
  - SDRAM, DDR, mobile SDRAM, and mobile DDR
  - 16- or 32-bit data, two chip-selects, configurations up to 2G bits on each chip-select

1.2.5 DMA Controllers

DMA controllers use system DMA (sDMA) configured for memory-to-memory, memory-to-peripheral, and peripheral-to-memory transfers:

- 1 read port, 1 write port
- 32 logical channels
- 256 x 32 bits FIFO depth

1.2.6 Multimedia Accelerators

The OMAP2420 uses high-end imaging and video applications:

- Imaging and video accelerator
  - Video teleconferencing (VTC) + recording (2 encode + 1 decode)
  - H.263 video conferencing
  - Joint Photographic Experts Group (JPEG) format
    - Bayer red-blue-green (RGB) charge-coupled device (CCD) requiring full image pipeline
    - Luminance and chrominance (YUV) data (no image pipeline required)
- Camera interface
  - Up to 10-bit data in Bayer RGB and ITU-R BT.656 formats
  - CAN calibration protocol (CCP) standard
  - Integrated DMA and MMU for high-bandwidth data transfer

- Display controller
  - Color and monochrome displays up to 2048 x 2048 x 24 bits per pixel (bpp) resolution
  - Picture-in-picture (overlay), color-space conversion, rotation, resizing support
  - NTSC/PAL video encoder with integrated DAC output

1.2.7 Comprehensive Power Management

The OMAP2420 provides the following power-management features:

- Clock and reset generation and distribution
- Wake-up event management
- Low-voltage operation modes
- Advanced leakage-management techniques to achieve ultralow standby power

1.2.8 Peripherals

The OMAP2420 supports a comprehensive set of peripherals for flexible and high-speed interfacing and on-chip programming resources:

- Enhanced audio controller (EAC)

  Provides three programmable audio interfaces enabling low power, multiple audio source mixing, and sample rate conversion

- UART1/2

  Two general serial communication interfaces

- UART3

  UART + IrDA up to FIR + TV remote control interface

- McBSP1/2

  Two general-purpose multichannel buffered serial interfaces

- I^{2}C1/2

  Two master/slave I^{2}C standard interfaces

- HDQ/1-Wire

  Benchmark HDQ and Dallas Semiconductor 1-Wire protocol interfaces

- McSPI1/2

  Two multichannel serial-port-interface (SPI) controllers

- USB OTG

  Multiport USB 2.0 full-/half-speed USB host/client controllers

- MMC/SD

  Interface controller for MMC/SD/SDIO standards

- General-purpose (GP) timers

  12 GP timers

- Watchdog timers (WDTs)

  Four WDTs

- 32-kHz timer

  32-kHz clock timer

- Quad GPIO

  Four 32-bit general-purpose input/output (I/O) controllers

- Mailbox

  MPU/DSP interprocessor communications

- Control

  I/O multiplexing and chip-configuration control

1.2.9 Debug Support

Comprehensive and concurrent Joint Test Action Group (JTAG)-based multiprocessor debug support uses the TI XDS560, ARM Ltd. Multi-ICE/Real-ICE, or third parties, such as Lauterbach emulation tools.

The ARM Ltd. embedded trace macrocell (ETM), in conjunction with an embedded 4K-byte trace buffer, supports the MPU trace.

Chapter 2

Memory Mapping

This chapter describes memory mapping for the OMAP2420 multimedia device.

Topic **Page**

2.1 Introduction

The OMAP2420 microprocessor unit (MPU) can handle 4G bytes with its 32-bit address port. The system memory mapping provides two levels of granularity for space address allocation. The first level features 1G-byte granularity with four quarters (Q0 to Q3) of 1GB. At the second granularity level, system targets are mapped on 128M bytes in each 1GB quarter.

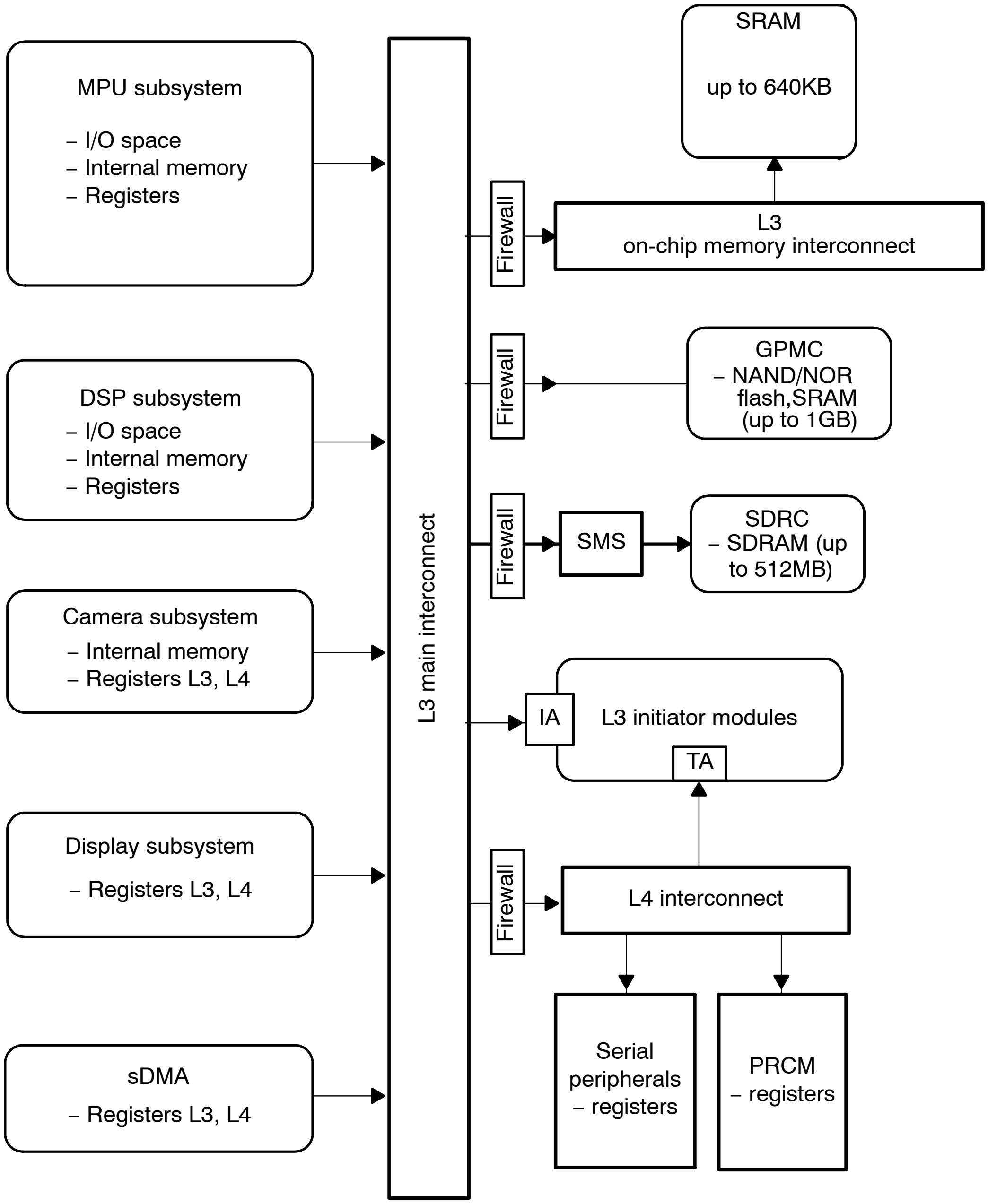
Memory map space is composed of memory space (GPMC, SDRC, etc.), register space (L3, L4 interconnect), and I/O space (DSP subsystem, etc.), which are shared among the OMAP2420 modules.

Each module/subsystem is connected to an interconnect port by a dedicated interface agent, which can be configured to tune the access, depending on the characteristics of each module.

Internal accesses are conditioned by firewalls defined in Chapter 6, *Internal Interconnect*, and in Table 6−7.

Figure 2−1 shows the memory type of each OMAP2420 module.

Figure 2−1. OMAP2420 Memory Organization and Type



2.2 Memory Mapping

2.2.1 Global Memory Space Mapping

This section provides a global view of OMAP2420 memory mapping. Some memory spaces are detailed below.

- ❑ Boot space

  The system boots in a 1M-byte boot space. This area is in either internal ROM or the general-purpose memory controller (GPMC) memory space.

  If the boot from internal ROM is selected, this 1M-byte memory space is redirected to the on-chip boot ROM memory address space (0x4000 0000 – 0x400F FFFF).

  If the boot from the GPMC is selected, this memory space is part of GPMC memory space. At reset, the 0x0000 0000 address is available on chip-select 0 (GPMC.nCS0) for a memory size of 16M bytes.

- ❑ GPMC space

  Eight chip-selects (GPMCn.CS0 to GPMCn.CS7) are available on the Q0 addressing space for the GPMC (GPMC for NOR/NAND flash, SRAM memories). These chip-selects have programmable size and programmable base addresses in the total memory space of Q0 (1GB).

  With the GPMC, software can access NAND-flash, NOR-flash, synchronous, and asynchronous protocol interfaces.

- ❑ SDRAM controller (SDRC) space

  Two chip-selects (SDRC.nCS0 and SDRC.nCS1) are available on the Q2 addressing space.

  The base address of chip-select 0 memory space (selected by the SDRC.nCS0 signal) is always mapped at 0x8000 0000. The base address of chip-select 1 memory space (selected by SDRC.nCS1) is programmable through the SDRC_CS_CFG register fields CS1STARTLOW and CS1STARTHIGH.

  According to the SDRC_CS_CFG register, the chip-select 1 base address is 0xA000 0000 (CS1STARTLOW = 0x0 and CS1STARTHIGH = 0x4) at reset and ranges from 0x8200 0000 (CS1STARTLOW = 0x1 and CS1STARTHIGH = 0x0) to 0xBFFF FFFF (CS1STARTLOW = 0x3 and CS1STARTHIGH = 0x7), according to the values programmed for CS1STARTLOW and CS1STARTHIGH.

  The SDRC-SMS virtual memory space is a different memory space used to access the memory through the rotation engine.

Table 2−1 describes the memory space mapping.

Table 2−1. OMAP2420 Memory Space Mapping

Device Name		Start Address (hex)	End Address (hex)	Size	Description
GPMC/ boot space—Q0		0x0000 0000	0x3FFF FFFF	1GB 1MB	8-/16-bit Ex/R/W See *Boot space* and *GPMC space*, above
Internal boot ROM	Reserved	0x4000 0000	0x400F FFFF	1MB	
Reserved		0x4010 0000	0x401F FFFF	1MB	
Internal SRAM	Reserved	0x4020 0000	0x4029 FFFF	640KB	
Reserved (SRAM)		0x402A 0000	0x402F FFFF	384KB	
Reserved		0x4030 0000	0x47FF FFFF	125MB	
L4-peripherals—Q1		0x4800 0000	0x4FFF FFFF	128MB	See Section 2.2.3.3, *L4 Peripherals Memory Space Mapping*.
Reserved		0x5000 0000	0x57FF FFFF	128MB	
DSP sub-system—Q1	dmemory	0x5800 0000	0x5802 7FFF	160KB	DARAM + SARAM
	Reserved	0x5802 8000	0x5803 FFFF	96KB	
	dIPI	0x5900 0000	0x5900 0FFF	4KB	
	dMMU	0x5A00 0000	0x5A00 0FFF	4KB	
	Reserved	0x5A00 1000	0x5BFF FFFF	~64MB	
Reserved	Reserved	0x5C00 0000	0x5FFF FFFF	64MB	
Reserved		0x6000 0000	0x67FF FFFF	128MB	
L3-control registers—Q1		0x6800 0000	0x6FFF FFFF	128MB	See Table 2−7.
SDRC−SMS—Q1		0x7000 0000	0x7FFF FFFF	256MB	SDRC−SMS virtual memory space
SRDC/ SMS—Q2	CS0	0x8000 0000	0x9FFF FFFF	512MB	SDRC−SMS SDRAM main memory space (SMS)
	CS1	X[1]	X[1]+0x01FF FFFF	512MB	
Not used—Q3		0xC000 0000	0xFFFF FFFF	1GB	Future use

Notes: 1) X = 0xA000 0000 @ reset, 0x8200 0000 ≤ X ≤ 0xBFFF FFFF

2.2.2 DSP Subsystem Memory Space Mapping

Digital signal processor (DSP) I/O memory space contains only I/O addresses.

DSP memory space includes data/program memory space.

DSP mapping contains a 128-KB I/O memory space and a 16-MB data/program memory space.

Table 2−2 shows DSP I/O memory mapping.

This memory area is accessible with I/O read/write specific instructions.

Also, the 128-KB I/O space placement in the 16-MB DSP memory space is programmable through an input/output mapping (IOMA) register. This allows the DSP software to access the I/O space through normal instructions. Thus, the I/O space is accessible with both mappings.

All the addresses in Table 2−2 and Table 2−3 are as seen from the DSP.

Table 2−2. DSP I/O Memory Mapping – 8-Bit Addressed (1)

DSP I/O Space	**Start Address (hex − byte aligned)**	**End Address (hex − byte aligned)**	**Size (KB)**	**Description**
DSP core configuration registers	0x0000	0x47FF	18	Accessible only with I/O accesses
DSP subsystem INTC	0x4800	0x4BFF	1	Internal memory
Reserved	0x4C00	0x4FFF	1	
DSP core prefetch buffer	0x5000	0x57FF	2	Internal memory
DSP global configuration registers	0x5800	0x5FFF	2	Internal memory
DSP DMA	0x6000	0x6FFF	4	Internal memory
DSP peripherals[1]	0x7000	0x1FFFE	100	External memory

Notes: 1) Up to 50 external peripherals are accessible with the DSP I/O space.

DSP peripherals are mapped on a 2-KB I/O space, but these spaces are mapped on a 4-KB space in the DSP memory space because of the minimal granularity of the memory management unit (MMU), which is 4 KB. This prevents mapping two peripherals in the same MMU space.

In the DSP memory space, 4KB are attributed for each peripheral, but only 2KB are used for the registers, and the other 2KB are reserved.

Table 2−3 shows DSP memory mapping for an IOMA base address value of 0xFC0000 (corresponding to an IOMA register value of 0x3E).

For this IOMA value, up to 34 peripherals can be mapped in the DSP mapping.

The IOMA value used in Table 2−3 is the base address of the overlaid I/O space on the DSP memory space. For directions for setting the IOMA address, see Section 4.4.9, *Defining Base Address of I/O Space*, in Chapter 4.

Table 2−3. DSP Subsystem Memory Space Mapping with IOMA = 0xFC0000[1]

DSP Memory Space (8-bit addresses)	**Start Address (hex – byte aligned)**	**End Address (hex – byte aligned)**	**Size (KB)**	**Description**
DARAM	0x00 0000	0x00 FFFF	64	DSP local memories
SARAM	0x01 0000	0x02 7FFF	96	DSP local memories
External memory	0x02 8000	0xFB FFFF		
Reserved	0xFC 0000	0xFC 8FFF	18	Beginning of remapped I/O area
DSP subsystem INTC configuration	0xFC 9000	0xFC 97FF	2	Remapped I/O

Notes: 1) Because of the IOMA value (0xFC 0000), only 34 peripherals are accessible with the DSP memory space.

Table 2–3. DSP Subsystem Memory Space Mapping with IOMA = 0xFC0000[1]

DSP Memory Space (8-bit addresses)	**Start Address (hex – byte aligned)**	**End Address (hex – byte aligned)**	**Size (KB)**	**Description**
Reserved	0xFC 9800	0xFC 9FFF	2	Remapped I/O
DSP core prefetch buffer	0xFC A000	0xFC A7FF	2	Remapped I/O
Reserved	0xFC A800	0xFC AFFF	2	
Global configuration registers	0xFC B000	0xFC B7FF	2	Remapped I/O
Reserved	0xFC B800	0xFC BFFF	2	
DMA configuration	0xFC C000	0xFC CFFF	4	Remapped I/O
Reserved	0xFC D000	0xFC DFFF	4	
External peripheral[1]	0xFC E000	0xFC E7FF	2	Remapped I/O of one peripheral
Reserved	0xFC E800	0xFC EFFF	2	
External peripheral[1]	0xFC F000	0xFC F7FF	2	Remapped I/O of one peripheral
Reserved	0xFC F800	0xFC FFFF	2	
…	…	…	…	…
External peripheral[1]	0xFE F000	0xFE F7FF	2	Remapped I/O of the 34th peripheral
Reserved	0xFE F800	0xFE FFFF	2	
Program/Data ROM	0xFF 0000	0xFF FFFF	64	

Notes: 1) Because of the IOMA value (0xFC 0000), only 34 peripherals are accessible with the DSP memory space.

Table 2−4 proposes the DSP memory mapping for an IOMA base address value of 0xF8 0000 (corresponding to an IOMA register value of 0x3F).

For this IOMA value, up to 50 peripherals can be mapped in the DSP mapping.

Table 2−4. *DSP Subsystem Memory Space Mapping with IOMA = 0xF80000 (1)*

DSP Memory Space (8-bit addresses)	Start address (hex – byte aligned)	End address (hex – byte aligned)	Size (KB)	Description
DARAM	0x00 0000	0x00 FFFF	64	DSP local memories
SARAM	0x01 0000	0x02 7FFF	96	DSP local memories
External memory	0x02 8000	0xF7 FFFF		
Reserved	0xF8 0000	0xF8 8FFF	18	Beginning of remapped I/O area
DSP subsystem INTC configuration	0xF8 9000	0xF8 97FF	2	Remapped I/O
Reserved	0xF8 9800	0xF8 9FFF	2	Remapped I/O
DSP core prefetch buffer	0xF8 A000	0xF8 A7FF	2	Remapped I/O
Reserved	0xF8 A800	0xF8 AFFF	2	
Global config registers	0xF8 B000	0xF8 B7FF	2	Remapped I/O
Reserved	0xF8 B800	0xF8 BFFF	2	
DMA configuration	0xF8 C000	0xF8 CFFF	4	Remapped I/O
Reserved	0xF8 D000	0xF8 DFFF	4	
External peripheral[1]	0xF8 E000	0xF8 E7FF	2	Remapped I/O of one peripheral
Reserved	0xF8 E800	0xF8 EFFF	2	
External peripheral[1]	0xF8 F000	0xF8 F7FF	2	Remapped I/O of one peripheral
Reserved	0xF8 F800	0xF8 FFFF	2	
…	…	…	…	…
External peripheral[1]	0xFB F000	0xFB F7FF	2	Remapped I/O of the 50th peripheral
Reserved	0xFB F800	0xF8B FFFF	2	
External memory	0xFC 0000	0xFE FFFF		
Program/data ROM	0xFF 0000	0xFF FFFF	64	

Notes: 1) Because of the IOMA value (0xF8 0000), up to 50 peripherals are accessible with the DSP memory space.

DSP physical address:

It is possible to access the registers by addressing memory space or I/O space (see Chapter 4, *DSP Subsystem)*. Physical addresses are byte addresses and must be divided by 2 to obtain the DSP half-word address.

2.2.2.1 Example

This section provides an example of how to change standard software mapping.

The MPU software controls the programming of different memory mappings. TI provides a software component to handle the Linux or Symbian Bridge. The example conforms to the standard programming in this software component.

The example uses the mappings in Tables 2−2 through 2−4. MMU programming must be coherent with the IOMA value, 0xFC 0000.

Table 2−5 lists the correspondence of addresses between L4 peripheral memory and DSP memory/I/O space (for an IOMA value of 0xFC 0000).

Only 34 peripherals can be mapped. See Chapter 4, *DSP Subsystem*.

Table 2−5. Address (in Bytes) Links Table Between DSP and L4 Peripherals (IOMA = 0xFC0000)

Device Name	**Start Address (System space)**	**Size**	**Start Address (DSP MEM space)**	**Start Address (DSP I/O space)**	**Size**
PRCM ❏ DPLL ❏ Clock manager ❏ Power manager	0x4800 8000	2KB	0xFCE000	0x7000	
Reserved	0x4800 A000				
Quad GPIO: GPIO1	0x4801 8000	4KB	0xFCF000	0x7800	2KB
GPIO2	0x4801 A000	4KB	0xFD0000	0x8000	2KB
GPIO3	0x4801 C000	4KB	0xFD1000	0x8800	2KB
GPIO4	0x4801 E000	4KB	0xFD2000	0x9000	2KB
WDTIMER3 (DSP)	0x4802 4000	4KB	0xFD3000	0x9800	2KB
Display subsystem	0x4805 0000	1KB	0xFD4000	0xA000	
Camera	0x4805 2000	1KB	0xFD5000	0xA800	
sDMA	0x4805 6000	4KB	0xFD6000	0xB000	2KB
Reserved	0x4805 8000	4KB	0xFD7000	0xB800	2KB
USB	0x4805 E000	4KB	0xFD8000	0xC000	2KB
McBSP1	0x4807 4000	4KB	0xFDA000	0xD000	2KB
McBSP2	0x4807 6000	4KB	0xFDB000	0xD800	2KB
GPTIMER5	0x4807 C000	4KB	0xFDC000	0xE000	2KB
GPTIMER6	0x4807 E000	4KB	0xFDD000	0xE800	2KB
GPTIMER7	0x4808 0000	4KB	0xFDE000	0xF000	2KB
GPTIMER8	0x4808 2000	4KB	0xFDF000	0xF800	2KB
EAC	0x4809 0000	4KB	0xFE0000	0x10000	2KB
FAC	0x4809 2000	4KB	0xFE1000	0x10800	2KB
Mailbox	0x4809 4000	4KB	0xFE2000	0x11000	2KB

Note: All MMU pages are configured to be 4KB pages for L4 peripherals.

2.2.3 L3 and L4 Interconnect Register Memory Mapping

The L3 interconnect is divided into two submodules:

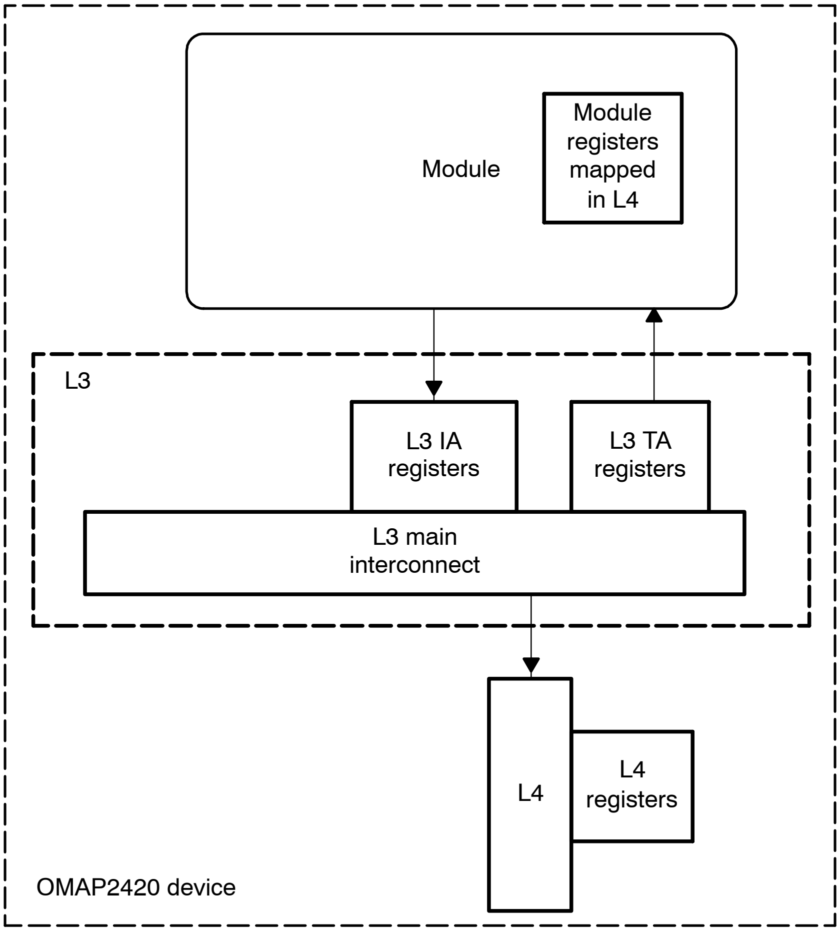
- LLRC or XB: Low-latency-response crossbar, which is a custom routing between modules, optimized for latency. It is a true crossbar structure, connecting all processors to the L3 memories: SDRAM, flash, on-chip L3 RAM, and ROM.
- SB: Silicon backplane, which is optimized for bandwidth allocation, including a 64-bit shared data path

The L4 interconnect is optimized to support the interconnection of a high number of peripherals.

The register mappings of L3 and L4 interconnects are provided in this section. These registers are configured by the software.

Figure 2−2 shows the repartition of the registers used for intermodule communication.

Figure 2−2. Register Localization



2.2.3.1 Control Register Mapping

L3 interconnect registers are mapped in a 128-MB space composed of SB-interconnect and crossbar-interconnect control registers, which allow the configuration of the firewalls and L3 interconnect parameters.

Communication Channels

Target and initiator modules are connected to the L3 interconnect, but not all initiators can communicate with all targets. According to the rights configured in the L3 interconnect registers, firewalls authorize or forbid communication between initiator and target. Table 2−6 shows the possible links between the OMAP2420 modules. For more information, see Chapter 6*, Internal Interconnect*.

Table 2−6. Functional Paths Between Initiator Module and Target Module of the L3 Interconnect

		L4		**L3 (LLRC)**			**L3 (SB)**
Initiator	**Initiator Ports**	**Target**	**L4 Peripheral Inter connect[1]**	**SMS**	**GPMC**	**OCM RAM and ROM**	**DSP Mem**
MPU	MPU RD	+[2]		+	+	+	+
	MPU WR	+		+	+	+	+
	MPU Inst	+		+	+	+	+
LCD	LCD			+	+	+	
Camera	Camera			+	+	+	
	MMU			+	+	+	
dDMA	dDMA RD	+		+	+	+	
	dDMA WR	+		+	+	+	
DSP	DSP data	+		+	+	+	+
	DSP instruction			+	+	+	+
	DSP MMU[3]			+	+	+	+
USB	USB			+	+	+	
sDMA	sDMA RD	+		+	+	+	+
	sDMA WR	+		+	+	+	+

Notes:

1) A functional data path always exists from an L4 initiator agent to any L4 target module. As a result, all L3 initiator modules for which a data path to L4 exists can access all L4 peripherals.

2) The two modules communicate. Cell contains a + when a functional path exists. Cell is blank when no functional path exists.

3) When an MMU miss occurs and the DSP MMU is the original initiator of the transfer, DSP MMU connectivity indicates the only possible paths. In other cases, transfer requests are initiated by other DSP subsystem initiator modules and pass through the DSP MMU for address remapping. In this case, the DSP MMU can remap accesses with respect to permission of the original initiator (dDMA RD, dDMA WR, DSP data, DSP instruction). The functional path to be considered is the path from the original initiator module to the target module without any consideration if it is remapped or not by the DSP MMU.

2.2.3.2 L3 Interconnect Registers

The L3 interconnect instance requires registers to steer and ensure communication between target and initiator modules.

Table 2−7 describes three types of registers included in the L3 interconnect:

- L3 SB registers: Contain information such as status error, configuration time-out, release code, and software reset (for target)
- L3 SB/XB registers (firewall): Used by the firewall to control access rights
- L3 crossbar registers (error login): Give information about the error source and error type

Table 2−7. L3 Control Register Mapping

Device Name	**Start Address (hex)**	**End Address (hex)**	**Size**	**Description**
Reserved	0x6800 0000	0x6800 03FF	1KB	
sDMA − R	0x6800 0400		512 bytes	Initiator − L3 SB registers
sDMA − W	0x6800 0600		512 bytes	Initiator − L3 SB registers
Display interface − controller module	0x6800 0800		512 bytes	Initiator − L3 SB registers
DSP subsystem	0x6800 0A00		512 bytes	Initiator − L3 SB registers
MPU subsystem	0x6800 0C00		512 bytes	Initiator − L3 SB registers
Reserved	0x6800 0E00		512 bytes	
USB	0x6800 1200		512 bytes	Initiator − L3 SB registers
Camera	0x6800 1400		512 bytes	Initiator − L3 SB registers
Reserved	0x6800 1600		512 bytes	
Reserved	0x6800 1C00		512 bytes	
Reserved	0x6800 2200		512 bytes	
L4	0x6800 2400		512 bytes	Target − L3 SB registers
Reserved	0x6800 2600		512 bytes	
DSP (firewall)	0x6800 2800		1KB	Target − L3 SB−TW
Reserved	0x6800 2C00		512 bytes	
DSP subsystem (dmemory, dMMU, dIPI)	0x6800 2E00		512 bytes	Target − L3 SB registers
Reserved	0x6800 3400		512 bytes	
Reserved	0x6800 3800		512 bytes	
Reserved	0x6800 3E00		512 bytes	
SMS	0x6800 4000		512 bytes	Target − L3 SB registers
OCM	0x6800 4200		512 bytes	Target − L3 SB registers
GPMC	0x6800 4400		512 bytes	Target − L3 SB registers
Reserved	0x6800 4600		5*512 bytes	
RAM (firewall)	0x6800 5000		1KB	Target − L3 XB – OCM
RAM (error login)	0x6800 5400		1KB	Target − L3 XB – OCM
ROM (firewall)	0x6800 5800		1KB	Target − L3 XB – OCM
ROM (error login)	0x6800 5C00		1KB	Target − L3 XB – OCM
GPMC (firewall)	0x6800 6000		1KB	Target − L3 XB
GPMC (error login)	0x6800 6400		1KB	Target − L3 XB
Reserved	0x6800 6800		2*512 bytes	
SMS (error login)	0x6800 6C00		1KB	Target − L3 XB
Reserved	0x6800 7000		4KB	
SMS registers	0x6800 8000		4KB	Target

Table 2−7. L3 Control Register Mapping (Continued)

Device Name	Start Address (hex)	End Address (hex)	Size	Description
SDRC registers	0x6800 9000		4KB	Target
GPMC registers	0x6800 A000		4KB	Target
Reserved	0x6800 B000	0x6FFF FFFF		

2.2.3.3 L4 Peripherals Memory Space Mapping

The L4 peripheral is mapped in a 128-MB space composed of L4 interconnect configuration registers and module registers.

The module registers are physically located in the modules and their memory space is about 4KB, although required space is often less than 2KB.

See Chapter 6, *Internal Interconnect.*

Table 2−8. L4 Peripheral Memory Space Mapping

Device Name	Start Address (hex)	End Address (hex)	Size	Description
OMAP2420 system control	0x4800 0000		4KB	Module
	0x4800 1000		4KB	L4 interconnect
Reserved	0x4800 2000		8KB	
32KTIMER	0x4800 4000		4KB	Module
	0x4800 5000		4KB	L4 interconnect
Reserved	0x4800 6000		8KB	
PRCM	0x4800 8000		2KB	Module region A (MPU, CORE, WKUP)
DPLL Clock manager Power manager	0x4800 8800		2KB	Module region B (DSP)
	0x4800 9000		4KB	L4 interconnect
Reserved	0x4800 A000		32KB	
Reserved	0x4801 2000		24KB	
Quad GPIO: GPIO1	0x4801 8000		4KB	GPIO1 module
	0x4801 9000		4KB	Quad GPIO top
GPIO2	0x4801 A000		4KB	GPIO2 module
	0x4801 B000		4KB	Quad GPIO L4 interconnect
GPIO3	0x4801 C000		4KB	GPIO3 module
	0x4801 D000		4KB	Reserved
GPIO4	0x4801 E000		4KB	GPIO4 module
	0x4801 F000		4KB	Reserved
Dual WDTIMER:	0x4802 0000		4KB	Reserved

Notes: 1) Reserved area for MPU interrupt module. Access to the MPU interrupt module registers at this memory space is conditional on suitable MPU CP15 initialization. See Chapter 3, *MPU Subsystem*, and Chapter 11, *Interrupt Controller*.

Table 2−8. L4 Peripheral Memory Space Mapping (Continued)

Device Name	**Start Address (hex)**	**End Address (hex)**	**Size**	**Description**
Reserved	0x4802 1000		12KB	Dual WDTIMER L4 interconnect
WDTIMER3(DSP)	0x4802 4000		4KB	Module
	0x4802 5000		4KB	L4 interconnect
Reserved	0x4802 6000		8KB	L4 interconnect
GPTIMER1	0x4802 8000		4KB	Module
	0x4802 9000		4KB	L4 interconnect
GPTIMER2	0x4802 A000		4KB	Module
	0x4802 B000		4KB	L4 interconnect
Reserved	0x4802 C000		80KB	
L4 configuration	0x4804 0000		2KB	Address protection (AP)
	0x4804 0800		2KB	Initiator port (IP)
	0x4804 1000		4KB	Link agent (LA)
Reserved	0x4804 2000		24KB	
ARM11ETB	0x4804 8000		8KB	Module
	0x4804 A000		4KB	L4 interconnect
Reserved	0x4804 B000		20KB	
Display subsystem	0x4805 0000		1KB	Display subsystem top
	0x4805 0400		1KB	Display controller (DISPC)
	0x4805 0800		1KB	Remote frame buffer interface (RFBI)
	0x4805 0C00		1KB	Video encoder (VENC)
	0x4805 1000		4KB	L4 interconnect
Camera	0x4805 2000		1KB	Camera top
	0x4805 2400		1KB	Camera core
	0x4805 2800		1KB	Camera DMA
	0x4805 2C00		1KB	Camera MMU
	0x4805 3000		4KB	L4 interconnect
Reserved	0x4805 4000		2*4KB	
sDMA	0x4805 6000		4KB	Module (L3)
	0x4805 7000		4KB	L4 interconnect
Reserved	0x4805 8000		20KB	L4 interconnect
Reserved	0x4805 D000		4KB	
USB	0x4805 E000		4KB	Module (L3)
	0x4805 F000		4KB	L4 interconnect

Notes: 1) Reserved area for MPU interrupt module. Access to the MPU interrupt module registers at this memory space is conditional on suitable MPU CP15 initialization. See Chapter 3, *MPU Subsystem*, and Chapter 11, *Interrupt Controller*.

Table 2−8. L4 Peripheral Memory Space Mapping (Continued)

Device Name	**Start Address (hex)**	**End Address (hex)**	**Size**	**Description**
Reserved	0x4806 0000		8KB	L4 interconnect
Reserved	0x4806 2000		8KB	L4 interconnect
Reserved	0x4806 4000		8KB	L4 interconnect
Reserved	0x4806 6000		8KB	L4 interconnect
Reserved	0x4806 8000		8KB	L4 interconnect
UART1	0x4806 A000		4KB	Module
	0x4806 B000		4KB	L4 interconnect
UART2	0x4806 C000		4KB	Module
	0x4806 D000		4KB	L4 interconnect
UART3 (with infrared)	0x4806 E000		4KB	Module
	0x4806 F000		4KB	L4 interconnect
I2C1	0x4807 0000		4KB	Module
	0x4807 1000		4KB	L4 interconnect
I2C2	0x4807 2000		4KB	Module
	0x4807 3000		4KB	L4 interconnect
McBSP1	0x4807 4000		4KB	Module
	0x4807 5000		4KB	L4 interconnect
McBSP2	0x4807 6000		4KB	Module
	0x4807 7000		4KB	L4 interconnect
GPTIMER3	0x4807 8000		4KB	Module
	0x4807 9000		4KB	L4 interconnect
GPTIMER4	0x4807 A000		4KB	Module
	0x4807 B000		4KB	L4 interconnect
GPTIMER5	0x4807 C000		4KB	Module
	0x4807 D000		4KB	L4 interconnect
GPTIMER6	0x4807 E000		4KB	Module
	0x4807 F000		4KB	L4 interconnect
GPTIMER7	0x4808 0000		4KB	Module
	0x4808 1000		4KB	L4 interconnect
GPTIMER8	0x4808 2000		4KB	Module
	0x4808 3000		4KB	L4 interconnect
GPTIMER9	0x4808 4000		4KB	Module
	0x4808 5000		4KB	L4 interconnect

Notes: 1) Reserved area for MPU interrupt module. Access to the MPU interrupt module registers at this memory space is conditional on suitable MPU CP15 initialization. See Chapter 3, *MPU Subsystem*, and Chapter 11, *Interrupt Controller*.

Table 2−8. L4 Peripheral Memory Space Mapping (Continued)

Device Name	**Start Address (hex)**	**End Address (hex)**	**Size**	**Description**
GPTIMER10	0x4808 6000		4KB	Module
	0x4808 7000		4KB	L4 interconnect
GPTIMER11	0x4808 8000		4KB	Module
	0x4808 9000		4KB	L4 interconnect
GPTIMER12	0x4808 A000		4KB	Module
	0x4808 B000		4KB	L4 interconnect
Reserved	0x4808 C000		16KB	
EAC	0x4809 0000		4KB	Module
	0x4809 1000		4KB	L4 interconnect
FAC	0x4809 2000		4KB	Module
	0x4809 3000		4KB	L4 interconnect
MAILBOX	0x4809 4000		4KB	Module
	0x4809 5000		4KB	L4 interconnect
Reserved	0x4809 6000		8KB	
SPI1	0x4809 8000		4KB	Module
	0x4809 9000		4KB	L4 interconnect
SPI2	0x4809 A000		4KB	Module
	0x4809 B000		4KB	L4 interconnect
MMC/SDIO	0x4809 C000		4KB	Module
	0x4809 D000		4KB	L4 interconnect
Reserved	0x4809 E000		80KB	L4 interconnect
HDQ 1-Wire	0x480B 2000		4KB	Module
	0x480B 3000		4KB	L4 interconnect
Reserved	0x480B 4000		296KB	
MPU interrupt[1]	0x480F E000		4KB	
Reserved	0x480F F000	0x4FFF FFFF	127MB	

Notes: 1) Reserved area for MPU interrupt module. Access to the MPU interrupt module registers at this memory space is conditional on suitable MPU CP15 initialization. See Chapter 3, *MPU Subsystem*, and Chapter 11, *Interrupt Controller*.

Chapter 3

MPU Subsystem

This chapter describes the features, integration, and functions of the OMAP2420 microprocessor unit (MPU) subsystem, and includes a programming model and a register summary.

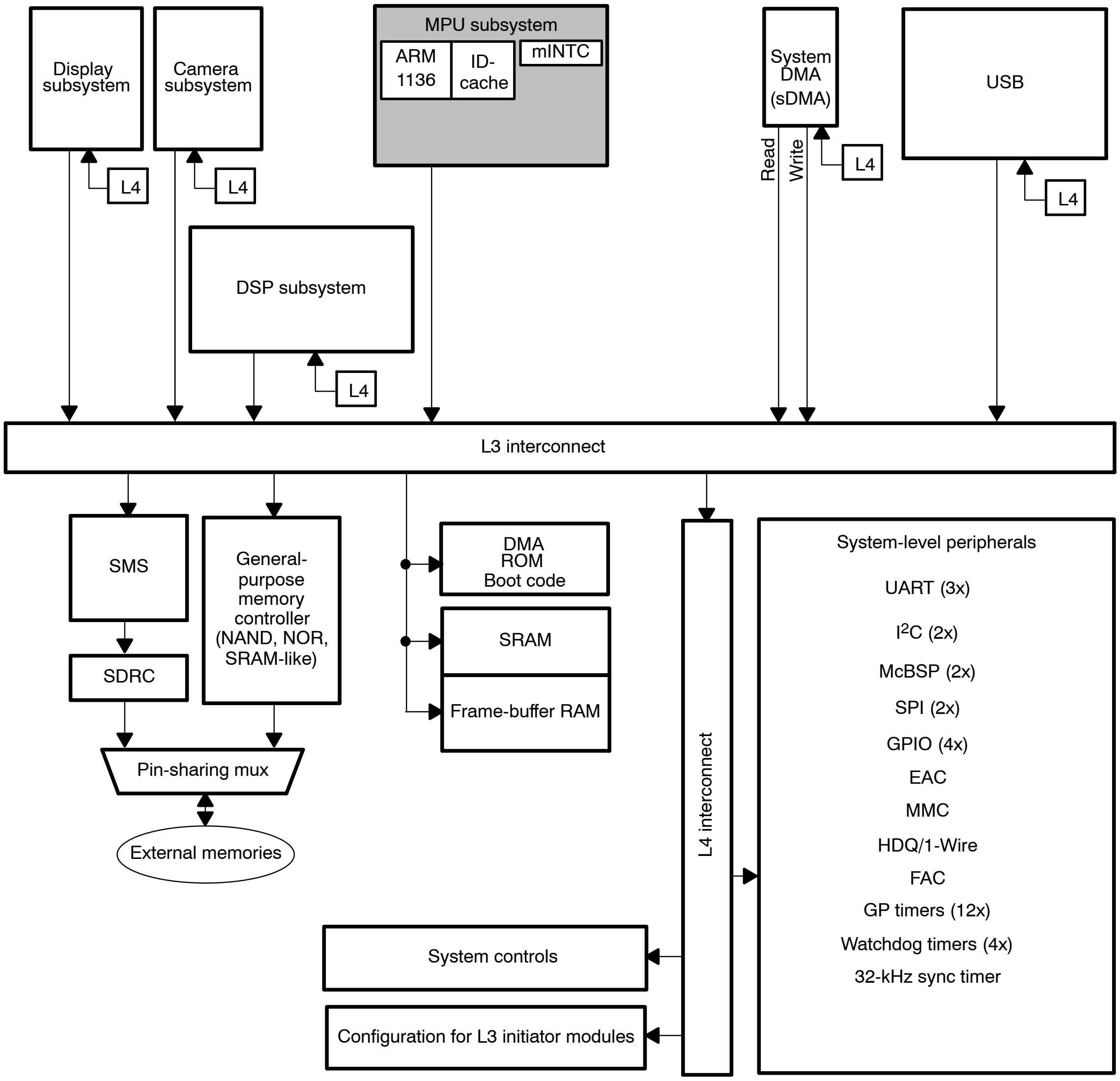
Topic **Page**

3.1 Microprocessor Subsystem Overview

The microprocessor unit (MPU) subsystem handles transactions for all OMAP2420 modules and external memory through the level 3 (L3) interconnect. The MPU subsystem handles transactions for external peripherals through the level 4 (L4) interconnect.

Figure 3−1 is a block diagram of the MPU subsystem.

Figure 3−1. MPU Subsystem



3.1.1 Main Features

The MPU is a soft macro that integrates the ARM1136 megacell with additional logic. The MPU includes an interrupt controller (INTC).

The MPU subsystem integrates the MPU core and the INTC (96 interrupt lines, synchronous) and includes three types of external interface:

- Power, reset, and clock-management (PRCM) module:
  - MPU functional clock, INTC functional clock, and enable signal for gating and division
  - Reset management for the MPU core
  - INTC clock and reset
  - Power control of the MPU subsystem power domains
- L3 interconnect

  Provides high-speed data and instruction access
- Peripherals

  The INTC allows a maximum of 96 interrupt lines for the MPU.

3.2 MPU Subsystem Integration

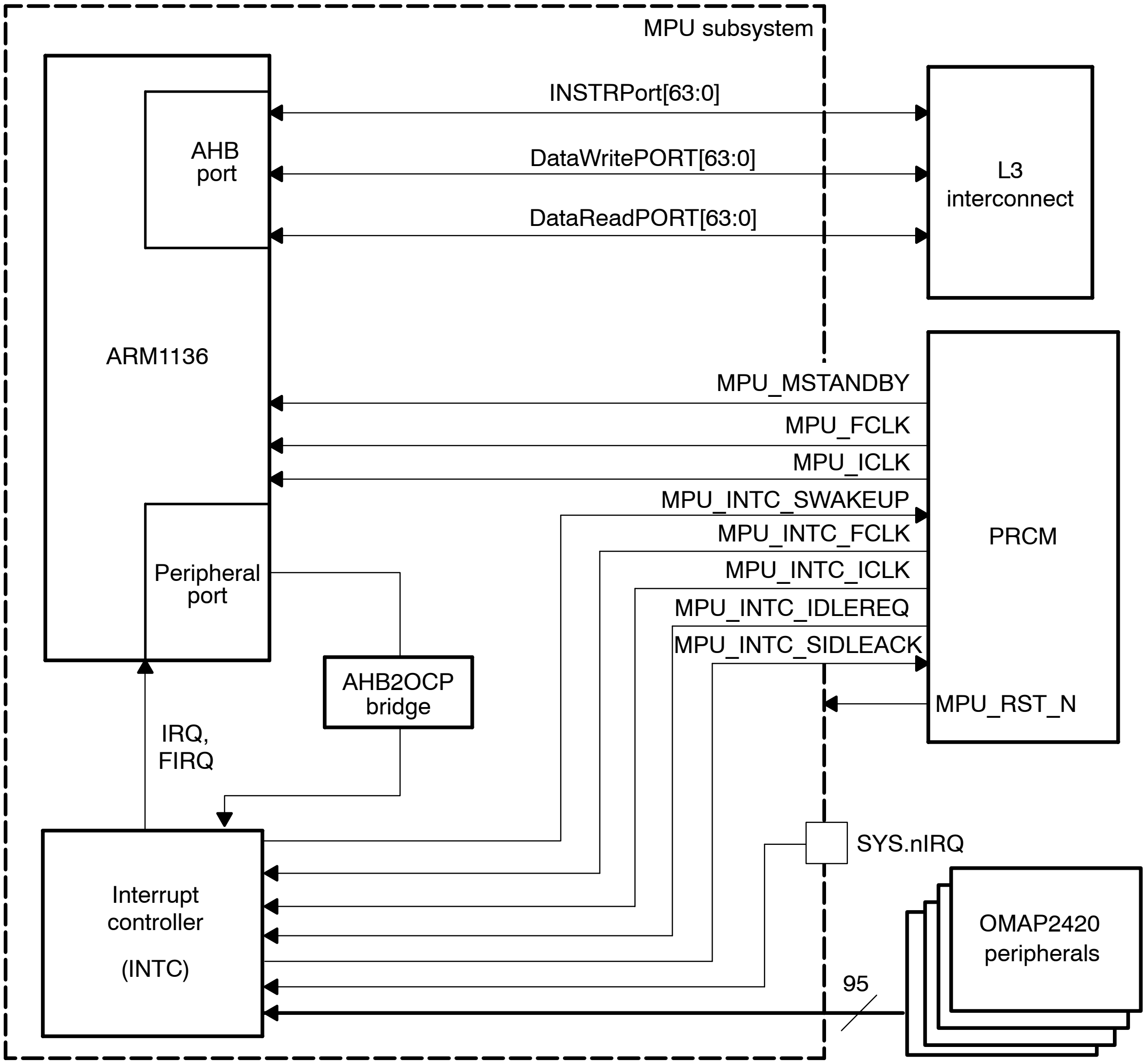
3.2.1 MPU Subsystem Description

The MPU subsystem integrates two submodules:

- ❑ MPU: provides high processing capability
- ❑ Interrupt control

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

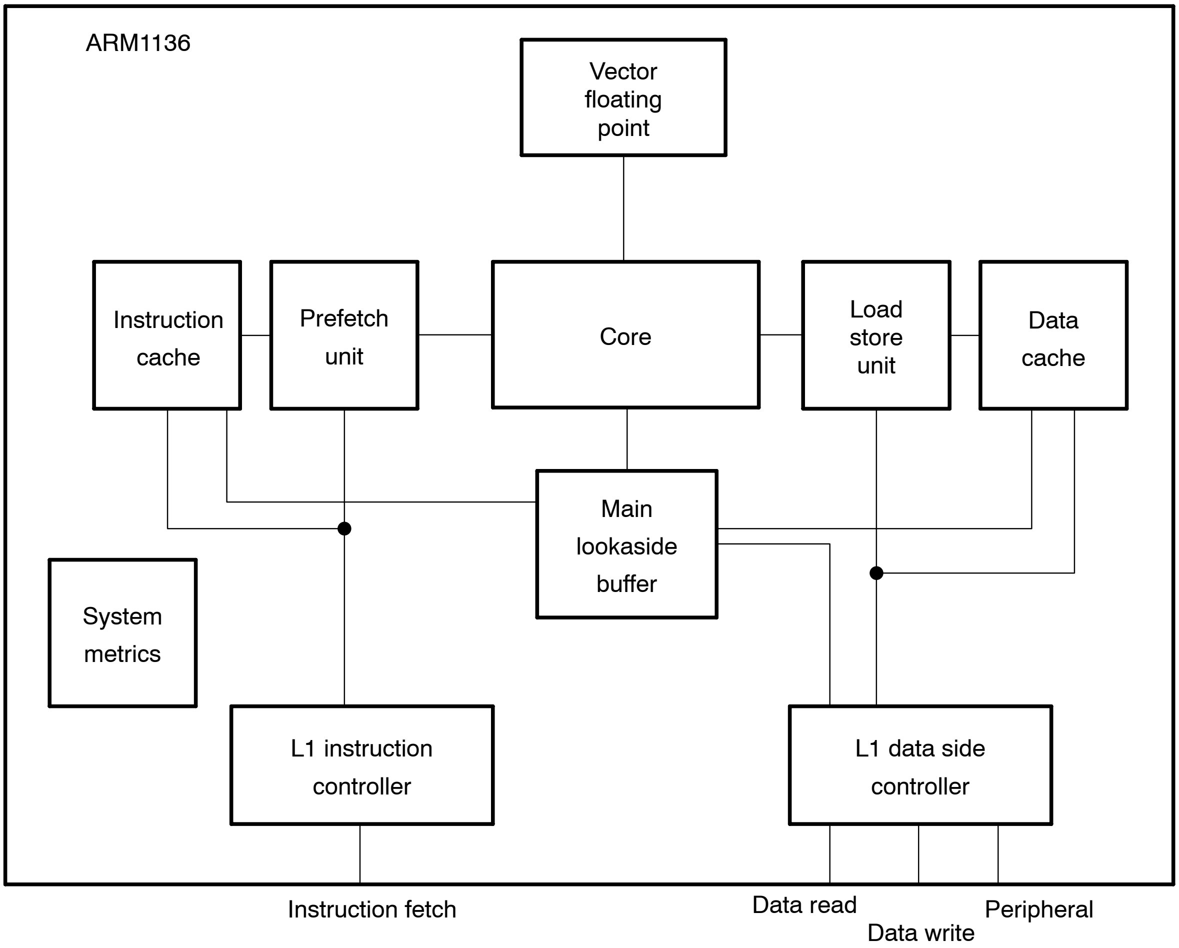
Figure 3−2. MPU Subsystem Overview



3.2.1.1 The ARM1136 Subsystem

Figure 3−3 shows the ARM1136JF-S modules implemented in the OMAP2420.

Figure 3−3. ARM1136 Functional Overview



ARM1136 Instruction, Data, and Private Peripheral Port

The ARM1136 provides three separate buses to access the physical memory space. The instruction and data (read, write) buses are connected to the L3 interconnect, which allows access to overall device resources (internal and external memory and internal peripherals), while the private peripheral port is dedicated to the MPU INTC connection. For details about accessing the private peripheral port, see Section 3.3, *MPU Subsystem Functional Description.*

Write-posted transactions have side effects that must be considered; write access to shared and nonshared devices or to the cacheable memory region is posted not only in the ARM1136 write buffer, but also at several stages of the L3 and L4 interconnect. This interconnect buffering is not visible to the ARM1136 memory-barrier instruction. A strongly ordered (nonbufferable/ posted-write transaction) memory region must be used if the posting effect is not desired, unless software synchronization is applied when a bufferable region is used.

It is assumed that the L4 memory space (common peripheral) is mapped to a strongly ordered region to avoid side effects (dual virtual mapping can also be used to access the posted sensitive L4 peripheral register through a strongly ordered region; the nonsensitive L4 peripheral register can be accessed through a shared device region to allow a higher-performance posted transaction).

MPU Subsystem Features

This section describes the primary functionalities of the ARM core in the MPU subsystem. The ARM1136JF-S processor is built around the ARM11 core in an ARMv6 implementation that runs the 32-bit ARM, 16-bit Thumb™, and 8-bit Jazelle™ instructions.

Note:

The ARM1136 is configured in little-endian mode for the OMAP2420.

Note:

The OMAP2420 does not implement the following ARM functions:

- ☐ ARM1136 DMA

  An external DMA in the OMAP2420 is accessible through the L3 interconnect (see Chapter 6 *Internal Interconnect*).

- ☐ Vectored interrupt controller (VIC) port

  The MPU subsystem implements an external INTC.

Table 3−1 lists ARM features implemented in the OMAP2420.

Table 3−1. Key Features of the MPU Core

Features	Comments
ARM version 6 ISA	Standard ARM instruction set plus Thumb, DSP, Jazelle Java accelerator, and media extensions Backward-compatible with previous ARM ISA versions
I-cache	32KB
I-cache way	4
D-cache	32KB
D-cache way	4
TLB	64 main TLB memory + 10 lockable entries (register-based)
Branch target address cache (BTAC)	128 entries
Memory-management unit (MMU)	Mapping sizes of 4KB, 64KB, 1MB, and 16MB

Table 3−1. Key Features of the MPU Core

Features	Comments
Vector floating point (VFP)	Present
Tightly coupled memory (TCRAM)	Not present
DMA	Not present
High-speed AMBA® bus level 2 interfaces supporting prioritized multiprocessor implementations	Advanced high−performance bus instruction (AHB), data read, data write connected to the L3 interconnect through AHB2OCP bridges AHB peripheral connected to INT AHB asynch mode not supported
Low interrupt latency	Present
VIC port	Not supported
External coprocessor	Not supported

For additional information, see http://www.ARM.com.

3.2.1.2 Clocks

At boot (the digital phase-locked loop [DPLL] in bypass mode), the MPU can run with the system clock at 12 MHz, 13 MHz, or 19.2 MHz.

Clock Configuration

For clock-configuration details, see Chapter 5, *Power, Reset, and Clock Management*.

Clock Names

The PRCM module provides two clocks to the MPU subsystem:

- ☐ MPU interface clock: Ensures correct communication between the MPU subsystem and the L3 interconnect
- ☐ MPU functional clock: Supplies the functional part of the MPU subsystem. Without the functional clock, the MPU subsystem is not operational.

Table 3−2 lists the ARM1136 functional clock and interface clock signals.

Table 3−2. ARM1136 Subsystem Clock Signals

Signal Name	Direction	Interface	Comments
MPU _FCLK	In	PRCM	Functional clock (PRCM module source). Supplies the ARM core and MPU logic.
MPU _ICLK	In	PRCM	Interface clock (PRCM module source). Supplies the MPU subsystem L3 interface and the L3 registers.

Reset Domain

The MPU reset domain is divided into two domains:

- ☐ One reset domain for the ARM and its logic
- ☐ One reset domain for the MPU INTC

Table 3−3 lists the ARM reset signal and ARM1136 reset domain.

Table 3−3. MPU Subsystem Reset Signal

Peripherals	**Signal Name**	**Reset Domain**
ARM1136	MPU_RST_N	MPU_RST

3.2.2 INTC

3.2.2.1 INTC Subsystem Features

The INTC includes the following features:

- ☐ Up to 96 levels of sensitive interrupt inputs, including one external interrupt (SYS.nIRQ)
- ☐ Individual programmable priority for each interrupt input
- ☐ Each interrupt can be steered to FIQ or IRQ.
- ☐ Independent priority sorting for FIQ and IRQ

Figure 3−4 shows the integration of the INTC in the MPU subsystem.

Table 3−4 lists other INTC features.

Figure 3−4. INTC Integrated in the MPU Subsystem

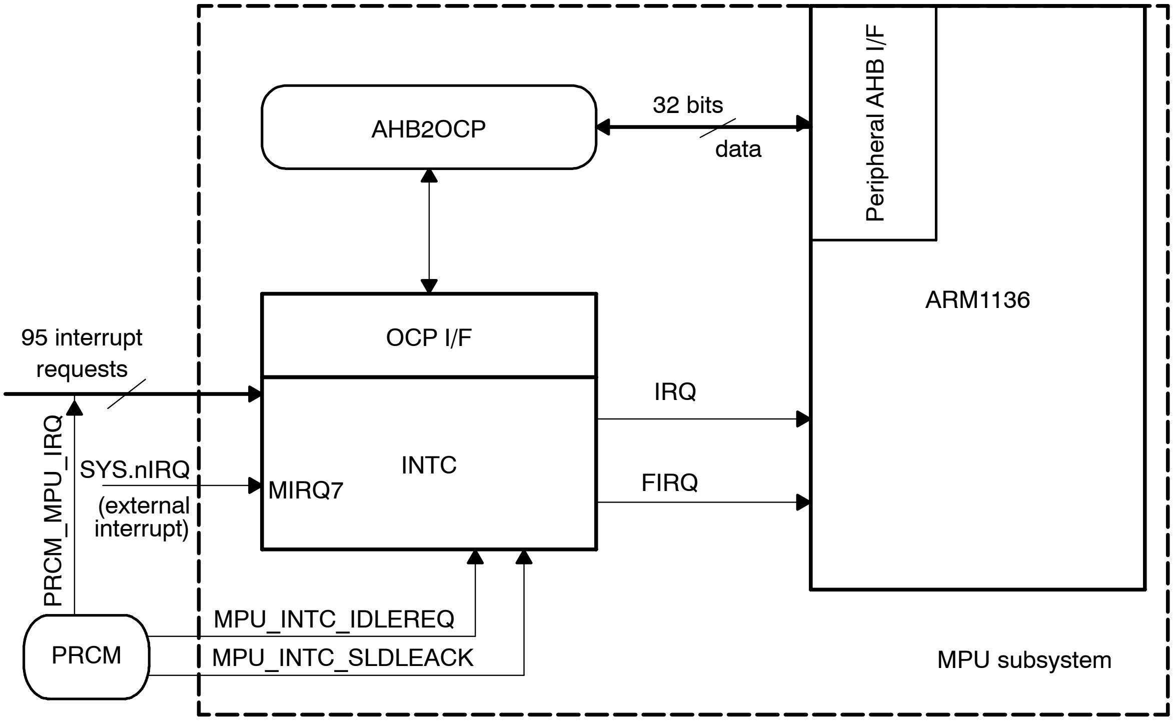


Table 3−4. INTC Generics in the MPU Subsystem

Parameter	Description	Value in MPU
NB_IT	Number of interrupt inputs	96
CODE_NB_IT	Number of interrupt number encodings	7
CODE_IT_SELECT	Number of sorting loops	3
CODE_PRIO	Number of interrupt priority encodings	6
SYNC_MODE	Relationship between L3 interconnect clock and functional clock 0: Asynchronous 1: Synchronous	1

3.2.2.2 External Interrupt SYS.nIRQ

The SYS.nIRQ pin is the only external interrupt in the MPU subsystem.

SYS.nIRQ is an optional external active-low interrupt that can be managed directly by the MPU INTC. Otherwise, this pin can be configured as a general-purpose input/output (GPIO). The SYS.nIRQ pin is mapped on interrupt line 7 of the MPU INTC.

3.2.2.3 Pin List and Pad Multiplexing with Other Functions

Although one interrupt is mapped as a direct external signal on the OMAP2420, additional external interrupt signals can be obtained using GPIO functions.

Table 3−5 describes the MPU subsystem multiplexing pin. The black cells indicate that the pin is not used with that mode, the gray cell indicates that the pin is used for that mode, and the white cell indicates that an alternate function is used for that mode.

Table 3−5. MPU Subsystem Multiplexing Pin

MPU Interface	Description	DIR	Ball	Multiplexing Modes			
				Mode0	Mode1	Mode 2	Mode3
SYS.nIRQ	External interrupt signal (active low)	I/O	W19	▨	■	■	GPIO.60

3.2.2.4 INTC Connectivity Attributes

Table 3−6 lists the signal names interacting with the INTC.

Table 3−6. Connectivity Attributes

Attributes	Values	Name	Mapping	Comments
Functional clock	MPU	INT_M_FCLK	MPU_INTC_ FCLK	Source is the PRCM module.
Interrupt request inputs	Up to 96	M_IRQ_[95:0]	See Section 3.2.2.7, *INTC Subsystem Mapping*	Inputs to the module; sources come from various modules.

Table 3−6. Connectivity Attributes (Continued)

Attributes	Values	Name	Mapping	Comments
Interrupt request outputs	2	INT_M_FIQ	ARM1136_FIQ	Fast interrupt
		INT_M_IRQ	ARM1136_IRQ	Normal interrupt
Wake source	1	INT_M_WAKE	MPU_INTC_ SWAKEUP	Destination is the PRCM module.

3.2.2.5 Clock Scheme

The MPU subsystem module can run at either the MPU clock rate or at half the MPU clock rate (as a function of the PRCM module setting).

The generic INTC used in the MPU subsystem has two clock input ports. However, the PRCM module provides only a single clock and a single synchronous clock enable. The clock enable generates the lower-frequency L3 interconnect clock from the higher-frequency functional clock.

As Table 3−7 shows, the INTC has a separate functional clock (MPU_INTC_FCLK) with an interface clock.

Table 3−7. INTC Subsystem Clock Signals

Signal Name	Direction	Interface	Comments
MPU_INTC_FCLK	In	PRCM	IRQ functional clock
MPU_INTC_ICLK	In	PRCM	IRQ functional clock gating/division

3.2.2.6 Reset Scheme

The MPU subsystem INTC belongs to a separate reset domain (CORE_RST). The CORE_RST signal resets the logic and the INTC.

3.2.2.7 INTC Subsystem Mapping

As Table 3−8 indicates, many interrupts are shared with other devices. The software checks that at least one interrupt is not masked.

Table 3−8. MPU Subsystem Interrupt Mapping

IRQ	Source	Description
M_IRQ_0	EMUINT	MPU emulation[1]
M_IRQ_1	COMMRX	MPU emulation[1]

Notes:
1) These interrupts are generated internally in the MPU subsystem.
2) The PRCM module uses this interrupt for four events:
   Transition in progress complete
   End of on or off period in the event generator
   Voltage transition complete
   Wake-up event
3) Shared interrupt also mapped on DSP level 2
4) Shared interrupt also mapped on DSP level 1
5) From the ARM1136 performance monitor control register (PMUIRQ)

Table 3−8. MPU Subsystem Interrupt Mapping (Continued)

IRQ	Source	Description
M_IRQ_2	COMMTX	MPU emulation[1]
M_IRQ_3	BENCH[5]	MPU emulation[1]
M_IRQ_4	Reserved	
M_IRQ_5	Reserved	
M_IRQ_6	Reserved	
M_IRQ_7	SYS.nIRQ	External interrupt (active low)
M_IRQ_8	Reserved	
M_IRQ_9	Reserved	
M_IRQ_10	L3_IRQ	L3 interconnect (transaction error)
M_IRQ_11	PRCM_MPU_IRQ	PRCM[2]
M_IRQ_12	SDMA_IRQ0	System DMA (sDMA) interrupt request 0
M_IRQ_13	SDMA_IRQ1	sDMA interrupt request 1
M_IRQ_14	SDMA_IRQ2	sDMA interrupt request 2
M_IRQ_15	SDMA_IRQ3	sDMA interrupt request 3
M_IRQ_16	Reserved	
M_IRQ_17	Reserved	
M_IRQ_18	Reserved	
M_IRQ_19	Reserved	
M_IRQ_20	GPMC_IRQ	General-purpose memory controller module
M_IRQ_21	Reserved	
M_IRQ_22	Reserved	
M_IRQ_23	EAC_IRQ	Audio controller [3]
M_IRQ_24	CAM_IRQ0	Camera interface interrupt request 0
M_IRQ_25	DSS_IRQ	Display subsystem module [3]
M_IRQ_26	MAIL_U0_MPU_IRQ	Mailbox user 0 interrupt request
M_IRQ_27	DSP_UMA_IRQ	DSP subsystem UMA core software interrupt
M_IRQ_28	DSP_MMU_IRQ	DSP subsystem MMU interrupt
M_IRQ_29	GPIO1_MPU_IRQ	GPIO module 1

Notes:

1) These interrupts are generated internally in the MPU subsystem.
2) The PRCM module uses this interrupt for four events:
   Transition in progress complete
   End of on or off period in the event generator
   Voltage transition complete
   Wake-up event
3) Shared interrupt also mapped on DSP level 2
4) Shared interrupt also mapped on DSP level 1
5) From the ARM1136 performance monitor control register (PMUIRQ)

Table 3−8. MPU Subsystem Interrupt Mapping (Continued)

IRQ	Source	Description
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	Reserved	
M_IRQ_34	MAIL_U3_MPU_IRQ	Mailbox user 3 interrupt request
M_IRQ_35	WDT3_IRQ	Watchdog timer module 3 overflow
M_IRQ_36	WDT4_IRQ	Watchdog timer module 4 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 [3, 4]
M_IRQ_42	GPT6_IRQ	General-purpose timer module 6 [3, 4]
M_IRQ_43	GPT7_IRQ	General-purpose timer module 7 [3]
M_IRQ_44	GPT8_IRQ	General-purpose timer module 8 [3]
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	GPT12_IRQ	General-purpose timer module 12
M_IRQ_49	Reserved	
M_IRQ_50	Reserved	
M_IRQ_51	Reserved	
M_IRQ_52	Reserved	
M_IRQ_53	Reserved	
M_IRQ_54	Reserved	
M_IRQ_55	Reserved	
M_IRQ_56	I2C1_IRQ	I^{2}C module 1
M_IRQ_57	I2C2_IRQ	I^{2}C module 2

Notes:
1) These interrupts are generated internally in the MPU subsystem.
2) The PRCM module uses this interrupt for four events:
   Transition in progress complete
   End of on or off period in the event generator
   Voltage transition complete
   Wake-up event
3) Shared interrupt also mapped on DSP level 2
4) Shared interrupt also mapped on DSP level 1
5) From the ARM1136 performance monitor control register (PMUIRQ)

Table 3−8. MPU Subsystem Interrupt Mapping (Continued)

IRQ	Source	Description
M_IRQ_58	HDQ_IRQ	HDQ/1−Wire
M_IRQ_59	MCBSP1_IRQ_TX	McBSP module 1 transmit [3]
M_IRQ_60	MCBSP1_IRQ_RX	McBSP module 1 receive [3]
M_IRQ_61	Reserved	
M_IRQ_62	MCBSP2_IRQ_TX	McBSP module 2 transmit [3]
M_IRQ_63	MCBSP2_IRQ_RX	McBSP module 2 receive [3]
M_IRQ_64	Reserved	
M_IRQ_65	SPI1_IRQ	McSPI module 1
M_IRQ_66	SPI2_IRQ	McSPI module 2
M_IRQ_67	Reserved	
M_IRQ_68	Reserved	
M_IRQ_69	Reserved	
M_IRQ_70	Reserved	
M_IRQ_71	Reserved	
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) [3]
M_IRQ_75	USB_IRQ_GEN	USB device general interrupt
M_IRQ_76	USB_IRQ_NISO	USB device non-ISO
M_IRQ_77	USB_IRQ_ISO	USB device ISO
M_IRQ_78	USB_IRQ_HGEN	USB host general interrupt
M_IRQ_79	USB_IRQ_HSOF	USB host start of frame
M_IRQ_80	USB_IRQ_OTG	USB OTG
M_IRQ_81	Reserved	
M_IRQ_82	Reserved	
M_IRQ_83	Reserved	
M_IRQ_84	Reserved	
M_IRQ_85	FAC_IRQ	FAC module

Notes:
1) These interrupts are generated internally in the MPU subsystem.
2) The PRCM module uses this interrupt for four events:
   Transition in progress complete
   End of on or off period in the event generator
   Voltage transition complete
   Wake-up event
3) Shared interrupt also mapped on DSP level 2
4) Shared interrupt also mapped on DSP level 1
5) From the ARM1136 performance monitor control register (PMUIRQ)

Table 3−8. MPU Subsystem Interrupt Mapping (Continued)

IRQ	Source	Description
M_IRQ_86	Reserved	
M_IRQ_87	Reserved	
M_IRQ_88	Reserved	
M_IRQ_89	Reserved	
M_IRQ_90	Reserved	
M_IRQ_91	Reserved	
M_IRQ_92	Reserved	
M_IRQ_93	Reserved	
M_IRQ_94	Reserved	
M_IRQ_95	Reserved	

Notes:

1) These interrupts are generated internally in the MPU subsystem.
2) The PRCM module uses this interrupt for four events:
   Transition in progress complete
   End of on or off period in the event generator
   Voltage transition complete
   Wake-up event
3) Shared interrupt also mapped on DSP level 2
4) Shared interrupt also mapped on DSP level 1
5) From the ARM1136 performance monitor control register (PMUIRQ)

3.3 MPU Subsystem Functional Description

3.3.1 Interrupts

The MPU INTC is connected on the MPU private peripheral port. The INTC runs at either the processor speed or at half the speed of the processor (for more information, see Chapter 11, *Interrupt Controller*).

The INTC prioritizes all service requests from system peripherals and generates either 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 more information, see Chapter 11, *Interrupt Controller.*

3.3.1.1 ARM INTC Mapping

Although the INTC is accessible at any physical address on the ARM11 private-peripheral interface (the INTC is aliased on any 2K-byte boundary), the physical base address 0x480F E000 must be used as the INTC base address. This base address corresponds to a 4K-byte unpopulated address region in the L4 (common peripheral area) physical memory space. Use of this reserved base address prevents the overlapping of physical addresses (between the INTC and another target), which can result in unexpected transaction redirection when both the memory-management unit (MMU) and the remap peripheral port register are active (see the mapping example in Table 3−9).

Table 3−9. L4 Peripheral INTC Memory Space Mapping

Device Name	Start Address (hex)	End Address (hex)	Size	Description
MPU subsystem INTC[1]	0x480F E000		4KB	

Notes: 1) Reserved area for MPU interrupt module. Access to the MPU interrupt module registers at this address space is conditional on suitable MPU CP15 initialization (L4-shared I/O).

Peripheral Port Memory Mapping

The peripheral interface is a bidirectional interface that services the INTC. The peripheral port can be addressed two ways:

- ❑ The INTC is mapped in the 1-MB L4 virtual region of shared I/O (data port) using the peripheral port relocation ARM1136 register. This implies that all accesses to empty space (4KB) reserved in L4 for INTC (see Table 3−9) are routed to the peripheral port.

  The CP15 register is used to remap the peripheral port when the MMU is off. To remap the INTC on the reserved shared I/O space (address 0x480F E000), use the following subroutine:

  ```
  INTH_BASE_ADDRESS EQU 0x480F E000

  INTH_4K_LENGTH EQU 0x3

  MapInterruptController
  ```

```
ldr r0, =INTH_BASE_ADDRESS ; Base address where to map
ldr r1, =INTH_4K_LENGTH ; Size =4k
orr r2,r0,r1
mcr p15,0,r2,c15,c2,4 ;Writing to CP15
```

- The INTC region is mapped in a nonshared region.

  Declare the nonshared attribute region in the MMU page (4KB minimum) table descriptor to access the ARM1136 peripheral port. Accesses to regions of memory marked as device and nonshared are routed to the peripheral interface.

  Even though the peripheral port and data port are separated space, do not map (overlap) at the same physical address on the two spaces (must appear as common space to the OS).

Note:

To ensure portability, the shared I/O space must be used for INTC mapping.

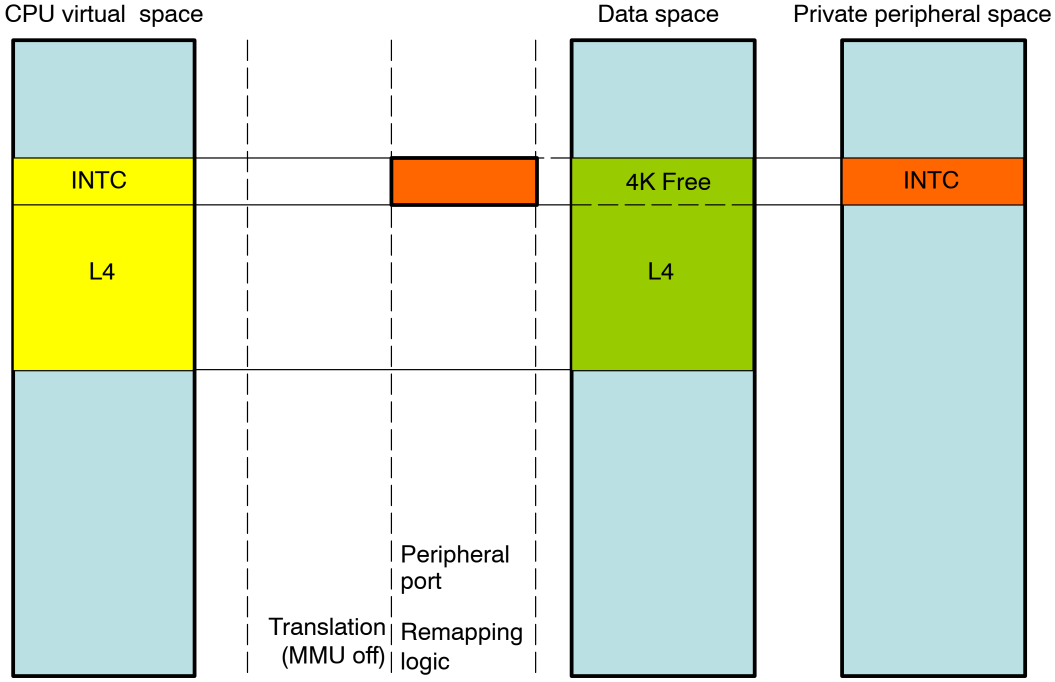
Figure 3−5 shows how the peripheral port memory remap register can be used to allow shared memory to access the peripheral port. In this case, the INTC is mapped to the same 1-MB section as the L4 shared I/O.

Accessing the INTC When MMU Is Off

The remap peripheral port register must be initialized with the remap physical base address 0x480F E000 on a 4K-byte region size (see the software routine example in Figure 3−5). Because no translation occurs (the MMU is off), the logical address used to access the INTC equals the physical address. It is recommended that the same software rules (as with the MMU on) be applied, to ensure ordering and central processing unit (CPU) pipeline synchronization for transactions directed to the INTC.

In Figure 3−5, the MMU is off. The relocation register is used to access the peripheral port.

Figure 3−5. Using Peripheral Port Memory Remap Register to Access Peripheral Port (MMU Off)



MMU is off. Relocation register is used to access peripheral port.

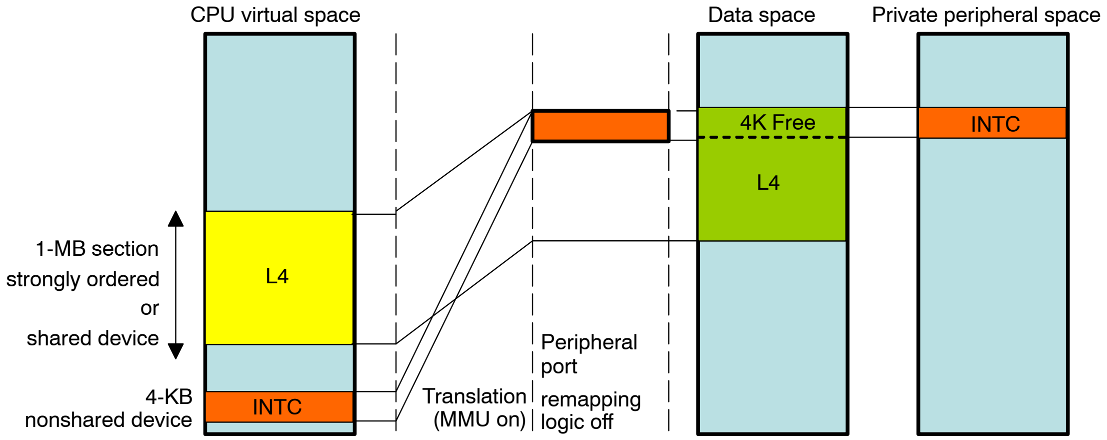
Accessing the INTC Through a Dedicated Nonshared Region When MMU Is On

Figure 3−6 shows the INTC mapped in a separate nonshared region. In this case, the nonshared attribute region is used to access the peripheral port.

A minimum 4K-byte region (any 4K-byte virtual region) must be declared a nonshared region with address translation to the reserved physical address 0x480F E000 4K-byte region. In that case, the remap peripheral-port register must be disabled (a zero-length region to be programmed). Software rules must be applied to ensure ordering and CPU pipeline synchronization for a transaction directed to the INTC.

In Figure 3−6, the MMU is on. The INTC is mapped in a dedicated 4K-byte nonshared device region. The relocation register is not needed (must be disabled).

Figure 3−6. Using Nonshared Attribute Region to Access Peripheral Port (MMU On)



Accessing the INTC Through the Common L4 Memory Region When MMU Is On

Figure 3−7 shows the INTC mapped in a separate shared region. In this case, the shared attribute region is used to access the peripheral port.

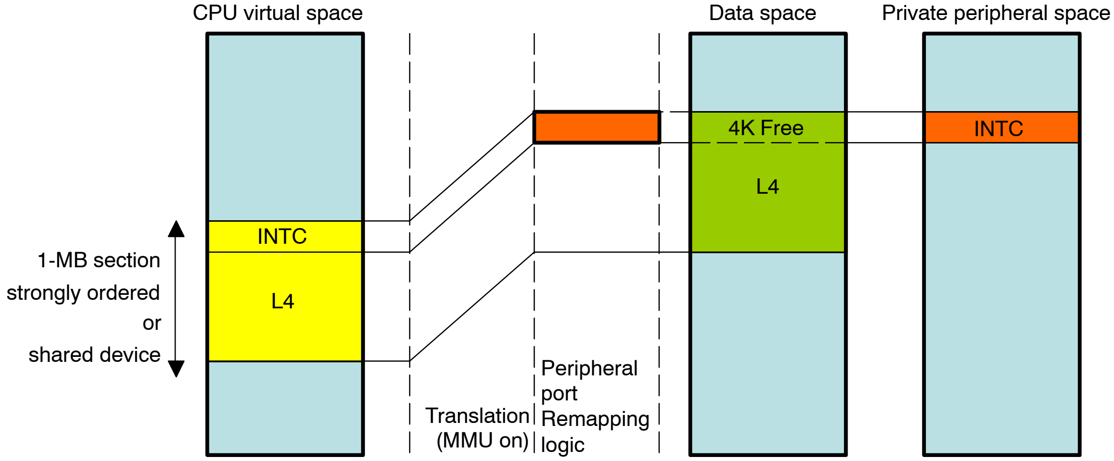
To prevent the use of a dedicated MMU translation lookaside buffer (TLB) entry, as required in the example in Figure 3−6 (a dedicated 4K-byte nonshared region must be present in the MMU TLB), it is possible to access the INTC through the L4 region if the remap peripheral-port register is enabled with the previous MMU-off setting. The L4 region can be mapped on a single 1M-byte MMU section, which results in a single MMU TLB entry use. Because this region must be declared as a strongly ordered region (transaction routed on the data port and not on the private peripheral port), the remap peripheral-port register must be enabled to redirect the INTC transaction to the peripheral port.

Note:

The shared-device attribute region can also be used for the L4 attribute region to allow a posting write to be issued to a peripheral (performance optimization). In that case, the software must account for the side effects of write-posting behavior (in the CPU write buffer and in the interconnect) and apply the appropriate strategy to control these effects (memory barrier, write followed by read). This usually applies to the software driver controlling the INTC and the interrupt service routine (ISR).

In Figure 3−7, the MMU is on. The INTC is mapped in the same 1M-byte section that L4 strongly ordered or in a shared device. The relocation register is used to access the peripheral port.

Figure 3−7. Using a Shared Attribute Region to Access Peripheral Port (MMU On)



3.3.1.2 ARM11 Private Peripheral Port

The ARM11 private peripheral interface is a dedicated bus separated from the common data and instruction buses. It is used to tightly connect peripherals that require low-latency access control, like the INTC. A different configuration allows the read and write transaction to be directed to the peripheral port rather than to the common data (read, write) port. A brief description of the programming is provided in the following paragraphs.

ARM11 Memory Region Attribute

The ARM11 memory space can be divided into multiple regions with different attributes (such as cacheable or noncacheable). The ARM11 MMU dynamically tags the transaction (read or write) with the attribute associated with the memory region (virtual address range) in which the transaction destination (virtual) address falls.

Accessing Private Peripheral Port with Nonshared Device Memory Attribute Region

Transactions issued to a nonshared device memory region are directed to the private peripheral port (as opposed to a strongly ordered shared device, which is directed to the common data port). This assumes that the ARM11 MMU is turned on and has the appropriate setting (page description). The transaction is directed on the peripheral port with the physical address that results from the initial virtual-address translation associated with the memory region.

Accessing Private Peripheral Port When the MMU Is Off

When the MMU is turned off, and the entire memory space is thus considered a strongly ordered memory region, the only way to direct a transaction to the private peripheral port is to initialize the remap peripheral-port register (ARM11 CP15 register) with a (physical) address range so that transactions are directed to the peripheral port. In this case, the physical address directed on the peripheral bus corresponds to the transaction virtual address, because no translation occurs (the MMU is off).

Remap Peripheral Register

If the remap register is enabled (the remapped region size is greater than 0), the redirection to the peripheral port of any physical matching address transaction is effective, whether the MMU is on or off (redirection occurs after the MMU translation because it uses the transaction physical address for the address compare). Thus, a transaction made to a device other than a nonshared device can be redirected to the peripheral port.

Nonshared (and Shared) Device Versus Strongly Ordered Transaction

Transactions to nonshared (and shared) device regions are bufferable (posted write), and their completions are loosely synchronized with CPU pipeline execution (whereas strongly ordered regions support nonbufferable transactions with tight CPU pipeline synchronization with respect to their completion). Therefore, a memory barrier must be used in certain circumstances to ensure completeness (write transaction) of a nonshared device transaction before further CPU execution can take place. An example is where a peripheral in a bufferable region is the source of an interrupt. When the interrupt has been serviced, the request must be removed before interrupts can be re−enabled. Ensure that the software driver in charge of the INTC and the ISR complies with the ordering and synchronization rules.

3.3.2 Power Management

3.3.2.1 Power Domains

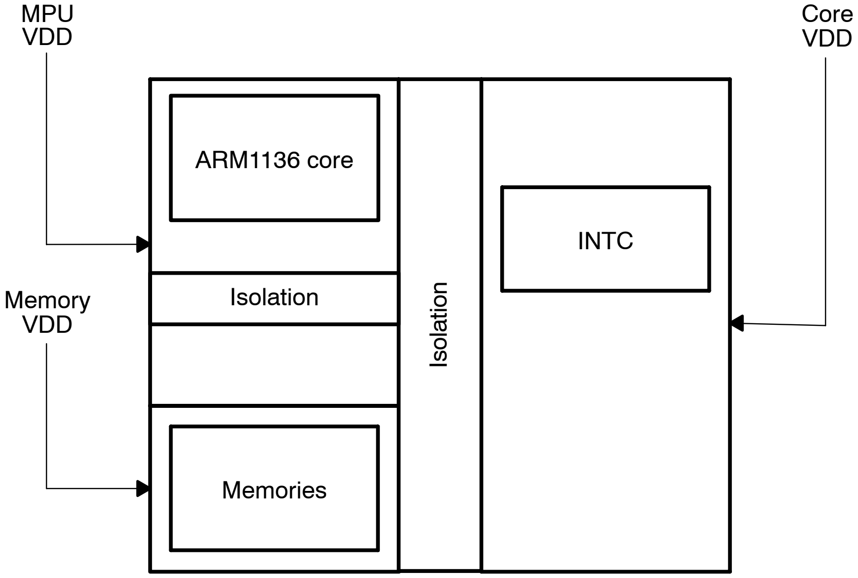
To reduce leakage, organize the MPU functional blocks in separate power domains to allow unused domains to be switched off (or be placed in a retention state) while keeping used domains active (see Figure 3−8).

ARM subchip power is split into two areas:

- MPU positive supply voltage (VDD): Covers the ARM11 core logic. An embedded power switch controlled by the PRCM module allows the ARM11 core power to be switched off while retaining memories in retention mode (see Section 3.3.2.2, *Power Modes*).
- Mem VDD: Covers the memories of the ARM subsystem.

The core VDD domain is based on the device core domain, which covers all logic in the MPU except the ARM subchip.

Figure 3−8. MPU Power Domains



3.3.2.2 Power Modes

Static power management requires action at both software and hardware levels. At the software level, the OMAP2420 defines four power states to enable power to be minimized, regardless of application requirements (see Table 3−10):

- ACTIVE
- INACTIVE
- RETENTION
- OFF

Table 3−10. Power Mode Definitions

OMAP2420 Power State	Required ARM1136 Power Mode	Definition
ACTIVE	Run mode	All clocks and powers on
INACTIVE	Standby mode	All clocks off, powers on, no memory retention
RETENTION	Dormant mode All retained	ARM1136 logic powered off and cache retained, all clocks off ARM1136 logic and cache retained (powered)
OFF	Shutdown mode	All clocks and powers off

- ACTIVE state

  The ACTIVE state is the normal application mode. All MPU clocks and power are on and all ARM1136 functions are available. Device voltage can be scaled from low-voltage point (not supported before ES2.2) to high-voltage point, depending on performance needs. This action applies simultaneously to all domains on the chip.

- INACTIVE state

  The INACTIVE state disables all MPU subsystem clocks while keeping power up, but there is no memory retention. Therefore, the MPU subsystem is not functional and no access can be performed.

  The INACTIVE state is different from the standby mode supported by ARM, Ltd. Standby mode turns off most but not all ARM11 subchip internal clocks. INACTIVE state, however, implies that the functional clock is off (gated by the PRCM module) for the entire MPU subsystem.

  To go to INACTIVE state, the application must perform an ARM1136 transition to standby mode to assert the MPU_MSTANDBY signal.

  To enter standby mode, execute the STANDBY WAIT_FOR_INTERRUPT CP15 operation. To ensure that the memory system is not affected by the entry into standby mode, the following operations are performed:

  - A DRAIN_WRITE_BUFFER operation ensures that all explicit memory accesses occur in program order before the STANDBY WAIT_FOR_INTERRUPT operation completes. This avoids any possible deadlocks caused when memory access triggers or enables an interrupt for which the core is waiting. This operation can require some TLB page table walks.
  - Any other memory accesses in progress when the STANDBY WAIT_FOR_INTERRUPT instruction is executed are completed normally. This ensures that the level 2 memory system does not experience disruption as a result of the WAIT_FOR_INTERRUPT.

- RETENTION state (ARM1136 dormant mode or all retained)

  When the MPU domain is in RETENTION state, there are two possible states:

- All clocks are off and both ARM logic and cache are retained (all retained).
- ARM logic is off and only caches are retained (dormant mode).

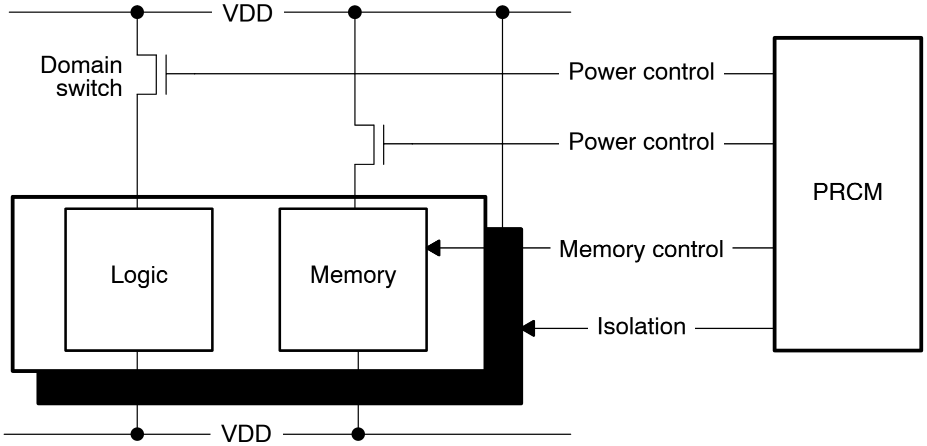
RETENTION state (ARM logic is off and caches are retained) for the MPU subsystem is equivalent to the dormant mode defined by ARM, Ltd. It is a low-power state from which it is possible to exit with a short latency. This feature allows power consumption to be reduced when the device is in low-power mode, retaining memory contents for a fast context restore. The MPU subsystem is not functional in RETENTION state. All clocks are gated.

☐ OFF state

In the OFF state, the entire device is powered down. All states, including cache, must be saved externally. The device is returned to ACTIVE state/ run mode by assertion of a reset. When all the states of the ARM1136JF-S processor are saved, it executes a WAIT_FOR_INTERRUPT instruction. The signal STANDBYWFI is asserted to indicate to the PRCM module that the processor can enter shutdown mode.

Figure 3−9 shows the MPU domain power supply.

Figure 3−9. MPU Domain Power Supply



To enter RETENTION state, the software must perform context saving (in the general-purpose registers or cache, for example) and carries out the SWFI (STANDBY WAIT FOR INTERRUPT) instruction. When the ARM1136 MPU_MSTANDBY external output signal is asserted, the power manager controller can decide (based on its setting) to operate the dormant mode clock gating.

In dormant mode, cache memory is retained (powered), and then core logic is powered off. In full retained mode, cache memory and logic are retained (powered).

The active interface between the PRCM module and the ARM1136 subsystem is then reduced to the following:

- MPU_MSTANDBY signal (output from ARM1136)
- ISOLATION signal (input): Activate isolation
- MEMORY control (input): Drive the memory in retention
- Clocks, reset

The ARM1136 core wakeup goes through a reset of the core or an unmasked interrupt and a proper restore of the context of the processor. The registers used for programming the different power modes are described in Section 3.5, *MPU Subsystem Registers*.

3.4 MPU Subsystem Programming Model

3.4.1 MPU Subsystem Initialization Sequence

See Chapter 26, *Initialization*.

3.4.2 Clock Control

The example in this section shows how to select the MPU clock.

MPU clock frequency is selectable by writing in the PRCM module CM_CLKSEL_MPU[4:0] register bits. MPU frequency can equal CORE_CLK/[1, 2, 4, 6, or 8].

Notes:

- CORE_CLK depends on DPLL programming (m and n values in CM_CLKSEL1_PLL) and on DPLL output selection (CM_CLKSEL2_PLL).
- For clock configuration settings, see Chapter 5, *Power, Reset, and Clock Management*.
- Any clock frequency change occurs only after validation in PRCM module register bit PRCM_CLKCFG_CTRL[0].

3.4.3 MPU Power Transition

The following sequence describes how to operate two different MPU power domain state transitions:

- Run mode to dormant mode
- Run mode to retention mode

3.4.3.1 Run Mode to Dormant Mode Transition

1) Set the BITS field to retention by setting the PM_PWSTCTRL_MPU[1:0] field to 01.

2) Set the PRCM module bit register PM_PWSTCTRL_MPU[2] to 0 so that logic goes off in retention mode.

3) Set the CM_CLKSTCTRL_MPU[0] bit to 0 to cut off the clocks.

4) Save context (MPU registers) in a retained area (either cache or PRCM module general-purpose registers).

5) Program the SWFI ARM1136 instruction. After executing this instruction, the MPU domain goes to standby mode and asserts the MPU_ MSTANDBY signal. At this stage, if there is no pending interrupt, the PRCM module stops the MPU clocks.

6) The MPU domain enters dormant mode (logic is off and cache memory is retained).

3.4.3.2 Active Mode to Full Retention Mode Transition

1) Program the BITS field to retention by setting the PM_PWSTCTRL_MPU[1:0] field to 01.

2) Set the PRCM module bit register PM_PWSTCTRL_MPU[2] to 1 so that logic continues to be retained.

3) Set the CM_CLKSTCTRL_MPU[0] bit to 0 to cut off the clocks.

4) Program the SWFI ARM1136 instruction. After executing this instruction, the MPU domain goes into standby mode and asserts the MPU_MSTANDBY signal. At this stage, if there is no pending interrupt, the PRCM module stops the MPU clocks.

5) The MPU domain enters full retention mode (logic and cache memory are retained).

3.4.4 Interrupt Programming Sequence

The INTC receives incoming interrupts and supports only level-sensitive interrupts. The following sequence allows management of an interrupt request (IRQn) in the MPU INTC:

Note:

The sequence is the same for IRQ and FIQ.

Step 1: Assign the IRQn priority in the MPU subsystem INTC register bits INTC_ILRn[7:2].

Step 2: In the MPU subsystem INTC register bit INTC_ILRn[0], identify the type of interrupt (FIQ or IRQ) to be generated by the INTC. The INTC asserts the IRQ/FIQ signal and begins priority calculation. When the highest-priority interrupt is known, the SIR_IRQ/SIR_FIQ register is updated to the current interrupt number.

Step 3: When the INTC recognizes the interrupt, it initiates the ISR.

Step 4: In the ISR, read the MPU subsystem SIR_IRQ/SIR_FIQ[6:0] INTC register bits through the peripheral AHB port to determine which interrupt line is the cause of the interrupt. Based on the content of SIR_IRQ, the host can execute code specific to that interrupt.

Step 5: Deassert the IRQ/FIQ outputs from the INTC by writing 1 to the MPU subsystem INTC register bit INTC_CONTROL[0] or INTC_CONTROL[1]. Setting this bit enables the INTC to process any other pending interrupt.

Step 6: Return to interrupted code, subs pc, r14_irq, #4 (usually handled by the compiler).

For more information, see Chapter 11, *Interrupt Controller*.

3.5 MPU Subsystem Registers

Table 3−11 shows all MPU subsystem registers and their physical addresses. For a detailed description, see Section 11.6, *Registers*, of Chapter 11, *Interrupt Control.*

The MPU subsystem registers are INTC registers (see Table 3−11). The base address value (BaseAddr) depends on the software (see Section 3.3.1, *Interrupts*).

Table 3−11. MPU Subsystem INTC Register Summary

Register Name	Type	Register Width (Bits)	Physical Address
INTC_REVISION	R	32	BaseAddr+0x0000
INTC_SYSCONFIG	RW	32	BaseAddr+0x0010
INTC_SYSSTATUS	R	32	BaseAddr+0x0014
INTC_SIR_IRQ	R	32	BaseAddr+0x0040
INTC_SIR_FIQ	R	32	BaseAddr+0x0044
INTC_CONTROL	RW	32	BaseAddr+0x0048
INTC_PROTECTION	RW	32	BaseAddr+0x004C
INTC_IDLE	RW	32	BaseAddr+0x0050
INTC_ITR0	R	32	BaseAddr+0x0080
INTC_MIR0	RW	32	BaseAddr+0x0084
INTC_MIR_CLEAR0	RW	32	BaseAddr+0x0088
INTC_MIR_SET0	RW	32	BaseAddr+0x008C
INTC_ISR_SET0	RW	32	BaseAddr+0x0090
INTC_ISR_CLEAR0	RW	32	BaseAddr+0x0094
INTC_PENDING_IRQ0	R	32	BaseAddr+0x0098
INTC_PENDING_FIQ0	R	32	BaseAddr+0x009C
INTC_ITR1	R	32	BaseAddr+0x00A0
INTC_MIR1	RW	32	BaseAddr+0x00A4
INTC_MIR_CLEAR1	RW	32	BaseAddr+0x00A8
INTC_MIR_SET1	RW	32	BaseAddr+0x00AC
INTC_ISR_SET1	RW	32	BaseAddr+0x00B0
INTC_ISR_CLEAR1	RW	32	BaseAddr+0x00B4
INTC_PENDING_IRQ1	R	32	BaseAddr+0x00B8
INTC_PENDING_FIQ1	R	32	BaseAddr+0x00BC
INTC_ITR2	R	32	BaseAddr+0x00C0
INTC_MIR2	RW	32	BaseAddr+0x00C4
INTC_MIR_CLEAR2	RW	32	BaseAddr+0x00C8
INTC_MIR_SET2	RW	32	BaseAddr+0x00CC

Table 3−11. MPU Subsystem INTC Register Summary (Continued)

Register Name	Type	Register Width (Bits)	Physical Address
INTC_ISR_SET2	RW	32	BaseAddr+0x00D0
INTC_ISR_CLEAR2	RW	32	BaseAddr+0x00D4
INTC_PENDING_IRQ2	R	32	BaseAddr+0x00D8
INTC_PENDING_FIQ2	R	32	BaseAddr+0x00DC
INTC_ILR0 − INTC_ILR31	RW	32	BaseAddr+0x0100− BaseAddr+0x017C
INTC_ILR32 – INTC_ILR63	RW	32	BaseAddr+0x0180− BaseAddr+0x01FC
INTC_ILR64 – INTC_ILR95	RW	32	BaseAddr+0x0200− BaseAddr+0x027C

3.5.1 MPU Register Descriptions

For detailed descriptions of the MPU registers, see Chapter 5, *Power, Resets, and Clock Management*, and Chapter 11, *Interrupt Controller*.

Chapter 4

DSP Subsystem

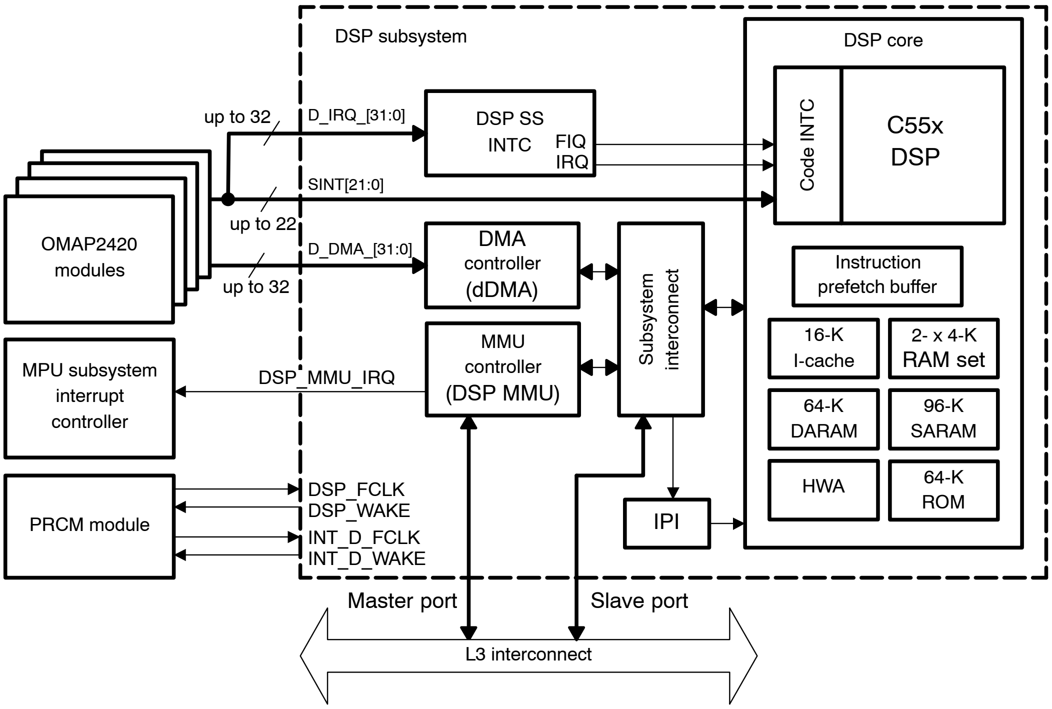
This chapter describes the features and integration of the OMAP2420 digital signal processor (DSP) subsystem and provides a programming model of the subsystem.

Topic **Page**

4.1 DSP Subsystem Overview

The OMAP2420 device integrates a DSP subsystem comprising the TMS320C55x DSP revision 3.0 core, video hardware coprocessor, dedicated direct memory access (DMA), memory-management unit (MMU), and interrupt controller modules. The DSP subsystem communicates with other OMAP2420 system modules through one master port and one slave port, both connected to the level 3 (L3) interconnect. Figure 4−1 is a block diagram of the DSP subsystem.

Figure 4−1. DSP Subsystem Top−Level Block Diagram



4.1.1 DSP Subsystem Key Features

The DSP subsystem has the following key features:

- Two levels of interrupt control
- Integrated memory subsystem (DSP local memories)
- Integrated MMU and DMA
- Dedicated interface (intrusive port interface) for remote host accesses to DSP subsystem memories (includes endianness conversion support)
- DSP subsystem interfaces:
  - 64-bit L3-interconnect transactions for I-cache refill and DMA, including multithreading between DSP and DMA accesses, and direct data access to external memories and peripherals
  - Page-based endianness for DSP external accesses

- Fully pipeline-intrusive port interface to ensure optimized bandwidth for host accesses to DSP memories

4.1.2 DSP Core Features

The DSP subsystem has the following core features:

- 16-bit fixed point
- Enhanced instruction caching ability:
  - 16K-byte 2−way set associative instruction cache
  - Two 4K-byte RAM sets
- Two multipliers and two arithmetic/logic units enable the following operations:
  - Two instructions per cycle
  - Up to two multiply-accumulates per cycle
- Video hardware accelerators:
  - Discrete cosine transformation (DCT) and inverse discrete cosine transformation (iDCT)
  - Pixel interpolation
  - Motion estimation

4.1.3 DSP RAM/ROM Subsystem Features

The DSP RAM/ROM subsystem has the following features:

- 64K-byte embedded dual-access SRAM (DARAM)
- 96K-byte embedded single-access SRAM (SARAM)
- 64K-byte embedded program-data ROM (PDROM)

4.1.4 Advanced DMA Controller

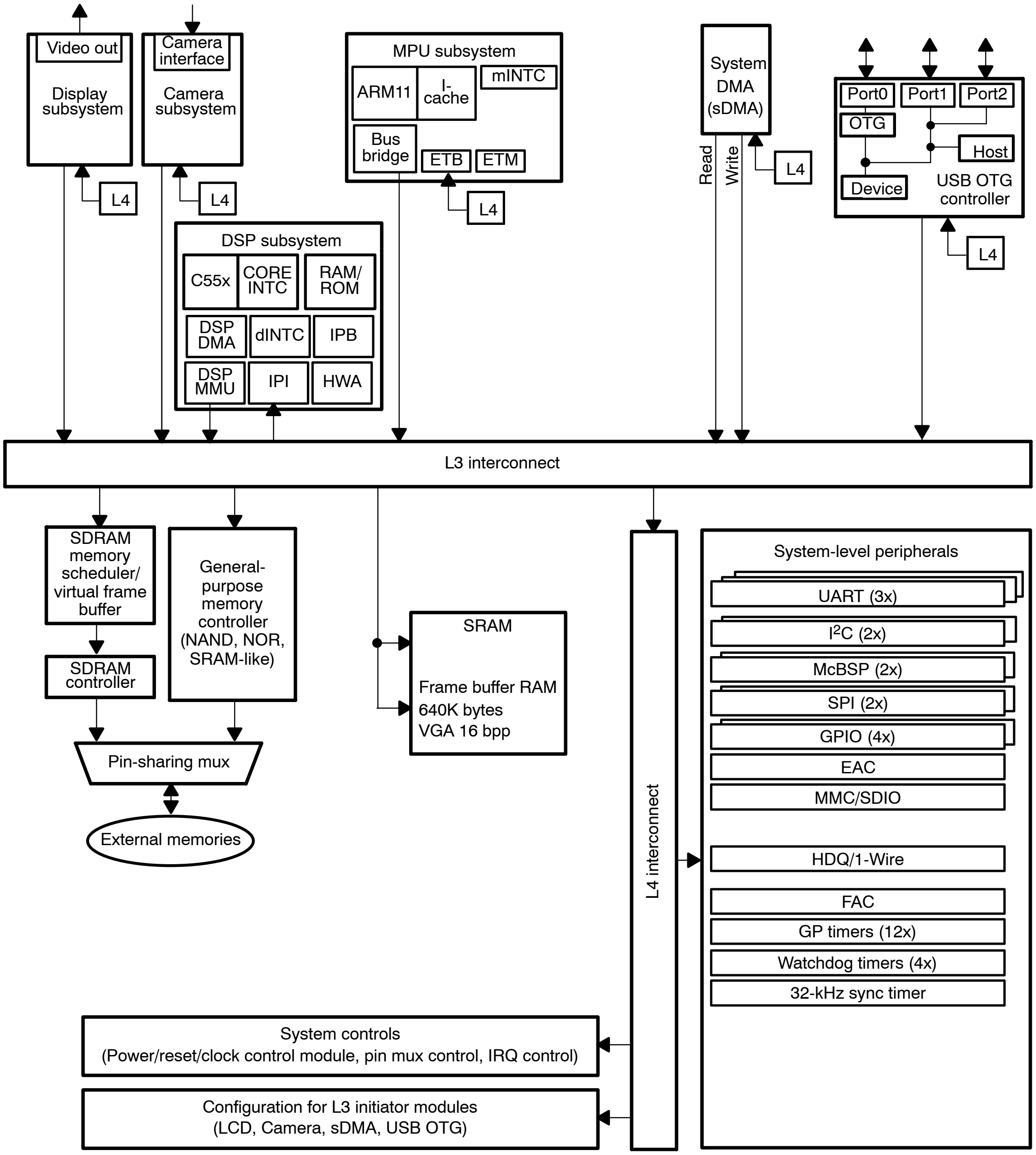
The advanced DMA controller has the following features:

- 24 logical DMA channels
- Multithreaded, fully pipelined, 2-port 64-bit data

4.2 DSP Subsystem Integration

Figure 4−2 shows DSP subsystem integration in the OMAP2420.

Figure 4−2. DSP Subsystem in the OMAP2420 Top Level



4.2.1 Power, Reset, and Clock Management

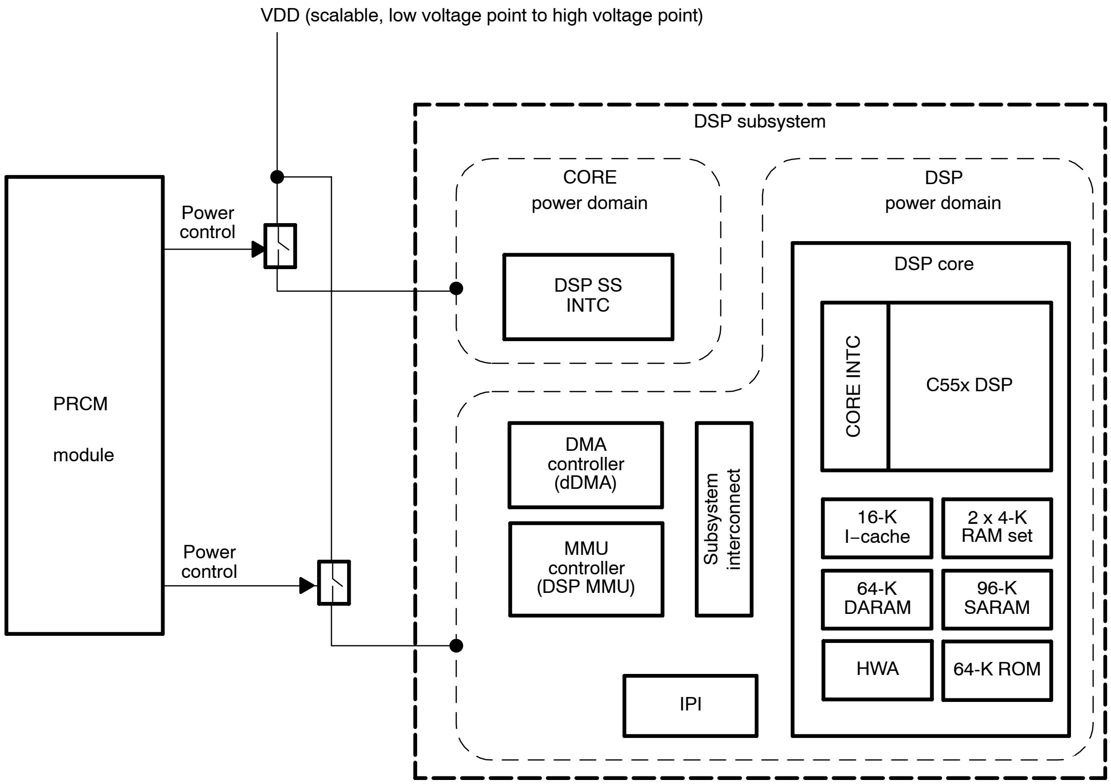
4.2.1.1 DSP Subsystem Power Domains

The DSP subsystem is split into two power domains:

☐ The DSP power domain
☐ The CORE power domain

Figure 4−3 shows the DSP power domains.

Figure 4−3. DSP Subsystem Power Domains



The DSP power domain can be powered off if the CORE power domain is powered on. The power, reset, and clock-management (PRCM) module ensures that the DSP power domain is always powered off when the CORE power domain is powered off.

The DSP power domain includes all the modules in the DSP subsystem except the DMA interrupt controller (dINTC).

The CORE power domain includes the dINTC from the DSP subsystem. The CORE power domain includes several other system-level components of the OMAP2420. The dINTC module is included in the CORE power domain to enable the powered-off DSP subsystem to be awakened after receiving an interrupt.

For a detailed description of power domain partitioning and control in the OMAP2420, see Chapter 5, *Power, Reset, and Clock Management*.

Before the DSP power domain can be powered off, the power, reset, and clock−management module (PRCM) in the OMAP2420 puts the dINTC into idle mode and turns off its clock. (If the INTC is enabled and has any pending interrupt requests to the DSP core, the DSP power domain is not powered off; thus, the dINTC must be disabled before powering off the DSP power domain.) Table 4−1 lists the DSP subsystem power domains.

Table 4−1. Power Domains for the DSP Subsystem

DSP Subsystem Module	Power Domain
DSP core	DSP
DSP DMA	
DSP MMU	
DSP_IPI	
DSP INTC	CORE

See Section 4.4, *Programming Model*, for more information about the programming model for DSP ubsystem power management.

4.2.1.2 DSP Subsystem Reset Domains

One or more reset domains are defined in each power domain in the OMAP2420 architecture. Each reset domain includes one or more modules, and independent control of these reset domains allows sequencing the release of resets to ensure a safe reset on the entire power domain.

DSP Subsystem Resets

In the DSP subsystem, there are three reset domains: two in the DSP power domain and one in the CORE power domain, as shown in Table 4−2.

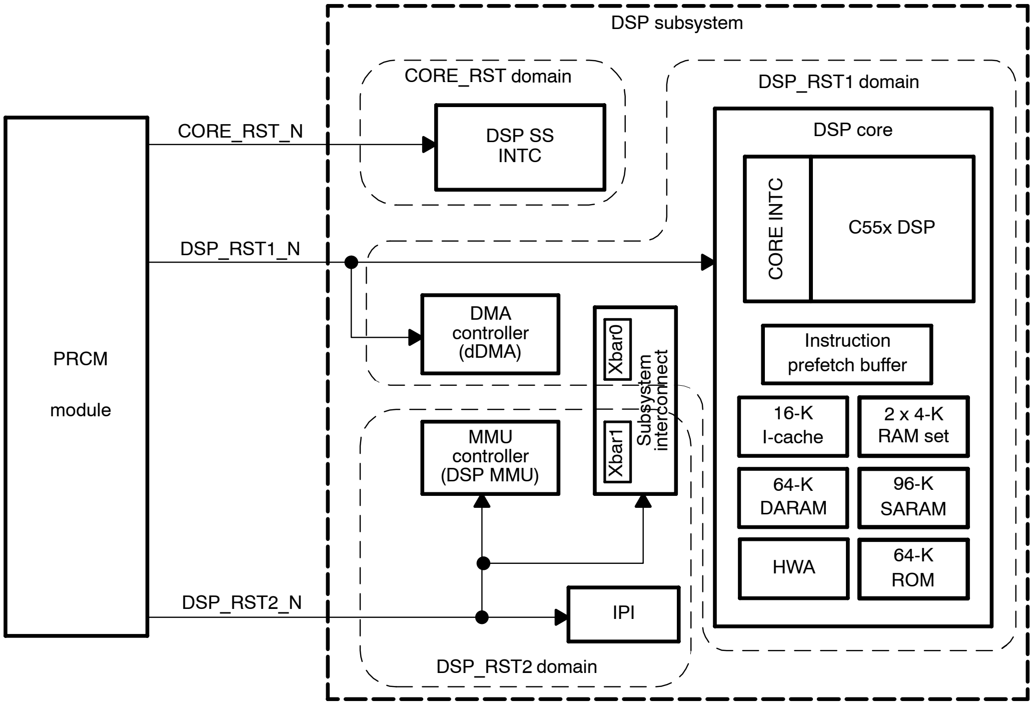
Table 4−2. Power Domains for DSP Subsystem

DSP Subsystem Module	Reset Domain	Reset Signal
DSP core	DSP1_RST	DSP_RST1_N
DSP DMA		
DSP MMU	DSP2_RST	DSP_RST2_N
DSP_IPI		
DSP subsystem INTC	CORE_RST	CORE_RST_N

As shown in Figure 4−4, there are three hardware reset signals at the boundary of the DSP subsystem:

- DSP_RST1_N is connected to the DSP core, subsystem interconnect crossbar 0, and DSP DMA modules.
- DSP_RST2_N is connected to the intelligent peripheral interface (IPI), MMU, and subsystem interconnect crossbar 1 modules.
- CORE_RST_NRES is connected to the DSP subsystem INTC module.

Figure 4−4. DSP Subsystem Resets



These reset signals are sourced from the PRCM module. The DSP_RST1_N signal maps to the RM_RSTCTL_DSP.RST1_DSP bit and the DSP_RST2_N signal maps to the RM_RSTCTL_DSP.RST2_DSP bit in the PRCM register RM_RSTCTRL_DSP. The DSP INTC reset signal, CORE_RST_N is reset with the CORE power domain, which is reset at the occurrence of either global cold or global warm reset events. (See Chapter 5, *Power, Reset, and Clock Management*, for more information about global reset sources.)

Some modules of the DSP subsystem can also be reset by software control.

Note:

Because the DSP DMA and core INTC modules are DSP private peripherals, software reset for them can be accessed only from software running on the DSP.

The input/output (I/O) map (IOMA) in Table 4−3 is the page index for accessing DSP I/O space addresses from DSP memory space. IOMA is programmed in the DSP IPI register, IPI_IOMAP.

Table 4−3. Software Resets in DSP SS

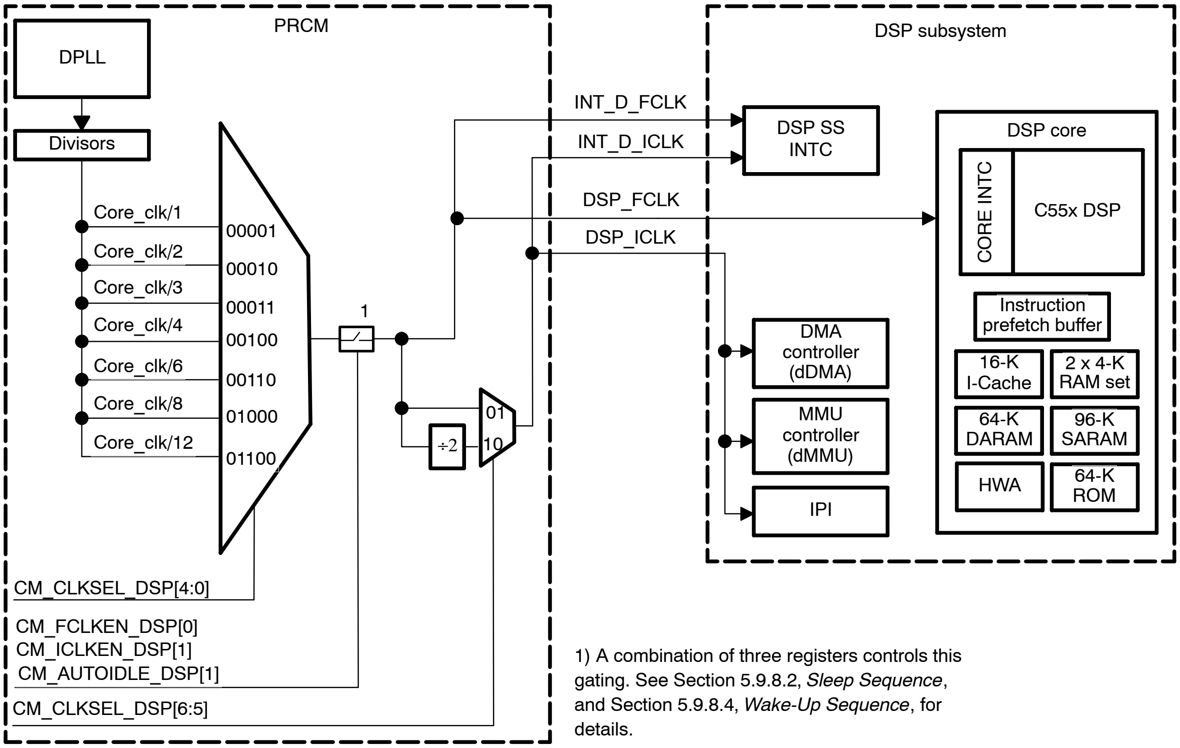
Register Name/ Bit Location	Physical Address	DSP MEM-Space Address	Field Name	Description
DSP MMU				
MMU_SYSCONFIG[1]	0x5A00 0010	n/a	SOFTRESET	DSP MMU software reset
DSP DMA				
DSP_DMA_SYSCONFIG[1]	N/A	IOMA+0xC02C	SOFTRESET	DSP DMA software reset
DSP INTC				
DSP_INTC_SYSCONFIG[1]	N/A	IOMA+0x9010	SOFTRESET	DSP core INTC software reset

See Section 4.4.9, *Defining Base Address of I/O Space*, for more information about the programming model for DSP subsystem reset management.

4.2.1.3 DSP Subsystem Clock Domains

The DSP subsystem contains three clock domains: one in the DSP INTC, one in the DSP core, and one for the DSP subsystem peripherals (DMA, IPI, and MMU). The PRCM module delivers a functional clock and an interface clock to the DSP INTC and the DSP core. A portion of the DSP subsystem (the DSP DMA, IPI, and MMU) can run at the same frequency as the DSP core, or at half the speed of the DSP core clock. Figure 4−5 shows the clock interfacing from the PRCM module to the DSP subsystem and the associated clock control registers in the PRCM module. The source of the DSP subsystem clocks is the PRCM core clock generated from the PRCM digital PLL.

Figure 4−5. DSP SS Clocks from PRCM



1) A combination of three registers controls this gating. See Section 5.9.8.2, *Sleep Sequence*, and Section 5.9.8.4, *Wake-Up Sequence*, for details.

Table 4−4 and Table 4−5 show the valid clock configurations for the DSP subsystem for normal and low system voltage settings. Only these specific clock configurations are valid for applications. These configurations are optimized for specific systems as identified below:

- Clock configuration 1: Optimized for 165-MHz SDRAM
- Clock configuration 2: Optimized for 100-MHz SDRAM
- Clock configuration 3: Optimized for 133-MHz SDRAM
- Clock configuration 4: Optimized for 133-MHz SDRAM and maximum MPU frequency

Note:

Frequencies listed in Table 4−4 and Table 4−5 are for reference only.

Clock configurations depend on core voltage and maximum clock frequencies. Values in this document may not be applicable to production devices.

CAUTION

Table 4−4. Valid DSP Subsystem, L3 Interconnect Clock Configurations for High-Voltage Operation

Clock Configuration	**PLL_ CLOCK (MHz)**	**DSP_ FCLK (MHz)**	**CM_CLK SEL_DSP [4:0] Setting**	**L3_CLK (MHz)**	**CM_ CLKSEL1_ CORE[4:0] Setting**	**DSP_ ICLK (MHz)**	**CM_ CLKSEL_ DSP [6:5] Setting**	**CM_ CLK SEL_ DSP[7]**	**Synchronizer Enabled**
1	660	220	00011b	165	00100b	110	10b	1	Yes
2	600	200	00011b	100	01000b	100	10b	0	No
3	532	177.33	00011b	133	01100b	88.66	10b	1	Yes
4	660	220	00011b	110	00110b	110	10b	0	No

Table 4−5. Valid DSP Subsystem, L3 Interconnect Clock Configurations for Low-Voltage Operation

Clock Configuration	**PLL_ CLOCK (MHz)**	**DSP_FCLK (MHz)**	**CM_CLK SEL_DSP [4:0] Setting**	**L3_CLK (MHz)**	**CM_ CLKSEL1_ CORE[4:0] Setting**	**DSP _ICLK (MHz)**	**CM_ CLKSEL_ DSP [6:5] Setting**	**CM_ CLK SEL_ DSP[7]**	**Synchronizer Enabled?**
1	330	110	00011b	82.5	01000b	55	10b	1	Yes
2	300	100	00011b	50	01100b	50	10b	0	No
3	266	88.665	00011b	66.5	01000b	44.33	10b	1	Yes
4	330	110	00011b	55	01100b	55	10b	0	No

The CM_CLKSEL_DSP[6:5] field controls the clock rate to the DSP DMA, MMU, and IPI (DSP_ICLK).

To control the clock synchronizer for the DSP subsystem, use PRCM register CM_CLKSEL_DSP[7]; 0 = disabled, 1 = enabled. For clock configurations 1 and 3, the system software must ensure that the synchronizer is enabled. For detailed information about valid OMAP2420 clock configurations and setting the digital PLL clock rate, see Chapter 5, *Power, Reset, and Clock Management*.

Section 4.4.5, *Clock Management*, provides more information about the programming model for DSP subsystem clock management.

4.2.2 DSP Subsystem Interrupts

To expand the interrupt capabilities of the DSP, the DSP subsystem includes two levels of interrupt control: one INTC integrated in the DSP core (formerly Level1 INTC) and one INTC at the DSP subsystem level (formerly Level2 INTC). Interrupt mapping from/to the DSP processor subsystem and core interrupt controllers is shown in Table 4−6 through Table 4−8. Selection of DSP subsystem INTC FIQ or IRQ generation to the DSP core from the 31 possible interrupts is also programmable using the FIQnIRQ field of register INTC_ILR[0:32].

The priority level for interrupts in the DSP subsystem INTC is also programmable through the INTC_ILR[0:32] registers; however, the INTC in the DSP core has fixed priority levels for all 28 interrupts, as shown in Table 4−6 and Table 4−7.

The DSP subsystem interrupt controller is in the CORE power domain. Asserting an unmasked interrupt to this interrupt controller wakes up the DSP subsystem and DSP core if either is idle and clocks are disabled from the PRCM module.

Table 4−6. Interrupt Names to DSP Subsystem INTC

Mapping	Source	Description
D_IRQ_0	Reserved	
D_IRQ_1	PRCM_DSP_IRQ	PRCM
D_IRQ_2	DDMA_IRQ0	DSP DMA interrupt request 0[1, 2]
D_IRQ_3	DDMA_IRQ1	DSP DMA interrupt request 1[1, 2]
D_IRQ_4	DDMA_IRQ2	DSP DMA interrupt request 2[1], [2]
D_IRQ_5	DDMA_IRQ3	DSP DMA interrupt request 3[1, 2]
D_IRQ_6	GPIO1_DSP_IRQ	GPIO module 1
D_IRQ_7	GPIO2_DSP_IRQ	GPIO module 2
D_IRQ_8	GPIO3_DSP_IRQ	GPIO module 3
D_IRQ_9	GPIO4_DSP_IRQ	GPIO module 4
D_IRQ_10	GPT5_IRQ	General-purpose timer module 5[1, 3]
D_IRQ_11	GPT6_IRQ	General-purpose timer module 6[1, 3]
D_IRQ_12	GPT7_IRQ	General-purpose timer module 73
D_IRQ_13	GPT8_IRQ	General-purpose timer module 8[3]
D_IRQ_14	MAIL_U1_DSP_IRQ	Mailbox user 1 interrupt request[1]
D_IRQ_15	EAC_IRQ	Audio controller1
D_IRQ_16	MCBSP1_IRQ_TX	MCBSP module 1 transmit[3]
D_IRQ_17	MCBSP1_IRQ_RX	MCBSP module 1 receive[3]
D_IRQ_18	Reserved	
D_IRQ_19	MCBSP2_IRQ_TX	MCBSP module 2 transmit[3]
D_IRQ_20	MCBSP2_IRQ_RX	MCBSP module 2 receive[3]
D_IRQ_21	Reserved	
D_IRQ_22	UART3_IRQ	UART module 3 (also infrared)[3]

1) Shared interrupt—also mapped on DSP core INTC

2) Interrupts generated internally in the DSP subsystem

3) Shared interrupt—also mapped on MPU

Table 4−6. Interrupt Names to DSP Subsystem INTC (Continued)

Mapping	Source	Description
D_IRQ_23	Reserved	
D_IRQ_24	Reserved	
D_IRQ_25	Reserved	
D_IRQ_26	Reserved	
D_IRQ_27	Reserved	
D_IRQ_28	Reserved	
D_IRQ_29	CAM_IRQ1	Camera module
D_IRQ_30	DSS_IRQ	Display subsystem module[3]
D_IRQ_31	Reserved	

1) Shared interrupt—also mapped on DSP core INTC

2) Interrupts generated internally in the DSP subsystem

3) Shared interrupt—also mapped on MPU

Table 4−7. DSP Core INTC Interrupt Mapping

Mapping	Source	Description	Priority
SINT0	RESET	DSP subsystem reset[1]	1
SINT1	Reserved		3
SINT2	DINT_FIQ	DSP subsystem INTC FIQ interrupt[2]	5
SINT3	DINT_IRQ	DSP subsystem INTC IRQ interrupt[2]	7
SINT4	Reserved		8
SINT5	MAIL_U1_DSP_IRQ	Mailbox user 1 interrupt request[3]	9
SINT6	Reserved		11
SINT7	Reserved		12
SINT8	Reserved		13
SINT9	DDMA_IRQ1	DSP DMA interrupt request 1[2, 3]	15
SINT10	Reserved		16
SINT11	Reserved		17
SINT12	Reserved		19
SINT13	Reserved		20

1) DSP reset by PRCM or DSP subsystem internal software reset

2) These interrupts are generated internally in the DSP subsystem.

3) Shared interrupt—also mapped on DSP subsystem INTCS

4) Shared interrupt—also mapped on MPU

Table 4−7. DSP Core INTC Interrupt Mapping (Continued)

Mapping	Source	Description	Priority
SINT14	Reserved		23
SINT15	Reserved		24
SINT16	Reserved		6
SINT17	Reserved		10
SINT18	DDMA_IRQ0	DSP DMA interrupt request 0[2, 3]	14
SINT19	Reserved		18
SINT20	DDMA_IRQ2	DSP DMA interrupt request 2[2, 3]	21
SINT21	DDMA_IRQ3	DSP DMA interrupt request 3[2, 3]	22
SINT22	GPT5_IRQ	General-purpose timer module 5[3, 4]	25
SINT23	GPT6_IRQ	General-purpose timer module 6[3, 4]	26
SINT24	Bus errors	Bus errors[2]	4
SINT25	DLOG	Emulation interrupt—DLOG[2]	27
SINT26	RTOS	Emulation interrupt—RTOS[2]	28
SINT27	Reserved		29
SINT28	Reserved		30
SINT29	Reserved		2
SINT30	Reserved		–
SINT31	Reserved		–

1) DSP reset by PRCM or DSP subsystem internal software reset
2) These interrupts are generated internally in the DSP subsystem.
3) Shared interrupt—also mapped on DSP subsystem INTCS
4) Shared interrupt—also mapped on MPU

CAUTION

Do not use DSP core interrupts for wake-up events. If an interrupt must wake up the DSP subsystem from idle mode, it must be connected to the DSP subsystem INTC.

Table 4−8 shows the DSP core INTC interrupts sorted by priority.

Table 4−8. DSP Core INTC Interrupts (Sorted by Priority)

Mapping	Source	Description	Priority
SINT0	RESET	DSP subsystem reset	1
SINT1	Reserved		3
SINT24	Bus errors	Bus errors	4
SINT2	DINT_FIQ	DSP subsystem INTC FIQ interrupt	5
SINT16	Reserved		6
SINT3	DINT_IRQ	DSP subsystem INTC IRQ interrupt	7
SINT4	Reserved		8
SINT5	MAIL_U1_DSP_IRQ	Mailbox user 1 interrupt request	9
SINT17	Reserved		10
SINT6	Reserved		11
SINT7	Reserved		12
SINT8	Reserved		13
SINT18	DDMA_IRQ0	DSP DMA interrupt request 0	14
SINT9	DDMA_IRQ1	DSP DMA interrupt request 1	15
SINT10	Reserved		16
SINT11	Reserved		17
SINT19	Reserved		18
SINT12	Reserved		19
SINT13	Reserved		20
SINT20	DDMA_IRQ2	DSP DMA interrupt request 2	21
SINT21	DDMA_IRQ3	DSP DMA interrupt request 3[1, 2]	22
SINT14	Reserved		23
SINT15	Reserved		24
SINT22	GPT5_IRQ	General-purpose timer module 5[1, 3]	25
SINT23	GPT6_IRQ	General-purpose timer module 6[3]	26
SINT25	Reserved		27
SINT26	Reserved		28
SINT27	Reserved		29

1) Shared interrupt—also mapped on DSP subsystem INTCS.

2) These interrupts are generated internally in the DSP subsystem.

3) Shared interrupt—also mapped on MPU

4.2.3 DSP Subsystem DMA Requests

Table 4−9 shows DMA request mapping to the DSP subsystem DMA controller.

Table 4−9. Interrupt Names to DSP Subsystem INTC

Mapping	Interrupt Name	Description
D_DMA_0	Reserved	
D_DMA_1	Reserved	
D_DMA_2	EAC.AC_DMA_RD	EAC audio codec port—read request[1]
D_DMA_3	EAC.AC_DMA_WR	EAC audio codec port—write request[1]
D_DMA_4	EAC.MD_UL_DMA_RD	EAC modem/voice port—uplink read request[1]
D_DMA_5	EAC.MD_UL_DMA_WR	EAC modem/voice port—uplink write request[1]
D_DMA_6	EAC.MD_DL_DMA_RD	EAC modem/voice port—downlink read request[1]
D_DMA_7	EAC.MD_DL_DMA_WR	EAC modem/voice port—downlink write request[1]
D_DMA_8	EAC.BT_UL_DMA_RD	EAC Bluetooth port—uplink read request[1]
D_DMA_9	EAC.BT_UL_DMA_WR	EAC Bluetooth—uplink write request[1]
D_DMA_10	EAC.BT_DL_DMA_RD	EAC Bluetooth—downlink read request[1]
D_DMA_11	EAC.BT_DL_DMA_WR	EAC Bluetooth port—downlink write request[1]
D_DMA_12	MCBSP1_DMA_TX	MCBSP module 1—transmit request[1]
D_DMA_13	MCBSP1_DMA_RX	MCBSP module 1—receive request[1]
D_DMA_14	MCBSP2_DMA_TX	MCBSP module 2—transmit request[1]
D_DMA_15	MCBSP2_DMA_RX	MCBSP module 2—receive request[1]
D_DMA_16	UART3_DMA_TX	UART module 3—transmit request[1]
D_DMA_17	UART3_DMA_RX	UART module 3—receive request[1]
D_DMA_18	Reserved	
D_DMA_19	Reserved	
D_DMA_20	Reserved	
D_DMA_21	Reserved	
D_DMA_22	Reserved	
D_DMA_23	Reserved	
D_DMA_24	Reserved	
D_DMA_25	Reserved	
D_DMA_26	Reserved	

1) Shared DMA request—also mapped on system DMA (sDMA)

Table 4−9. Interrupt Names to DSP Subsystem INTC (Continued)

Mapping	Interrupt Name	Description
D_DMA_27	Reserved	
D_DMA_28	Reserved	
D_DMA_29	Reserved	
D_DMA_30	Reserved	
D_DMA_31	Reserved	

1) Shared DMA request—also mapped on system DMA (sDMA)

4.2.4 L3 Interconnect

The DSP subsystem interfaces with the L3 interconnect through dedicated initiator agents (IAs) and target agents (TAs) that are part of the L3 interconnect. Each IA and TA provides status and configuration registers as listed in Tables 4−10 through 4−12. For a complete description of L3 interconnect agents and the programming model, see Chapter 6, *Internal Interconnect*.

Table 4−10. DPS SS L3 Interconnect Target Agent Registers

Register Name	Type[1]	Register Width (Bits)	Physical Address
TA.SBID	R	64	0x6800 2FF8
TA.SBTMCONFIG	R/W	64	0x6800 2FB8
TA.SBTMSTATE	R/W	64	0x6800 2F98
TA.SBTMERRLOG	R/W	64	0x6800 2F50
TA.SBTMERRLOGA	R	64	0x6800 2F48

1) R =Read, R/W = Read/write

Table 4−11. DSP SS L3 Interconnect Initiator Agent Registers

Register Name	Type	Register Width (Bits)	Physical Address
IA.SBID	R	64	0x6800 0BF8
IA.SBIMCONFIG	R/W	64	0x6800 0BA8
IA.SBIMSTATE	R/W	64	0x6800 0B90
IA.SBIMERRLOG	R/W	64	0x6800 0AB0
IA.SBIMERRLOGA	R	64	0x6800 0AA8

Table 4−12. DPS SS L3 Interconnect Firewall Registers

Register Name	Type	Register Width (Bits)	Physical Address
ERROR_LOG	R/W	64	0x6800 2820
REQ_INFO_PERMISSION0	R/W	64	0x6800 2848

Table 4−12. DPS SS L3 Interconnect Firewall Registers (Continued)

Register Name	Type	Register Width (Bits)	Physical Address
READ_PERMISSION0	R/W	64	0x6800 2850
WRITE_PERMISSION0	R/W	64	0x6800 2858
ADDR_MATCH1	R/W	64	0x6800 2860
REQ_INFO_PERMISSION1	R/W	64	0x6800 2868
READ_PERMISSION1	R/W	64	0x6800 2870
WRITE_PERMISSION1	R/W	64	0x6800 2878
ADDR_MATCH2	R/W	64	0x6800 2880
REQ_INFO_PERMISSION2	R/W	64	0x6800 2888
READ_PERMISSION2	R/W	64	0x6800 2890
WRITE_PERMISSION2	R/W	64	0x6800 2898
ADDR_MATCH3	R/W	64	0x6800 28A0
REQ_INFO_PERMISSION3	R/W	64	0x6800 28A8
READ_PERMISSION3	R/W	64	0x6800 28B0
WRITE_PERMISSION3	R/W	64	0x6800 28B8

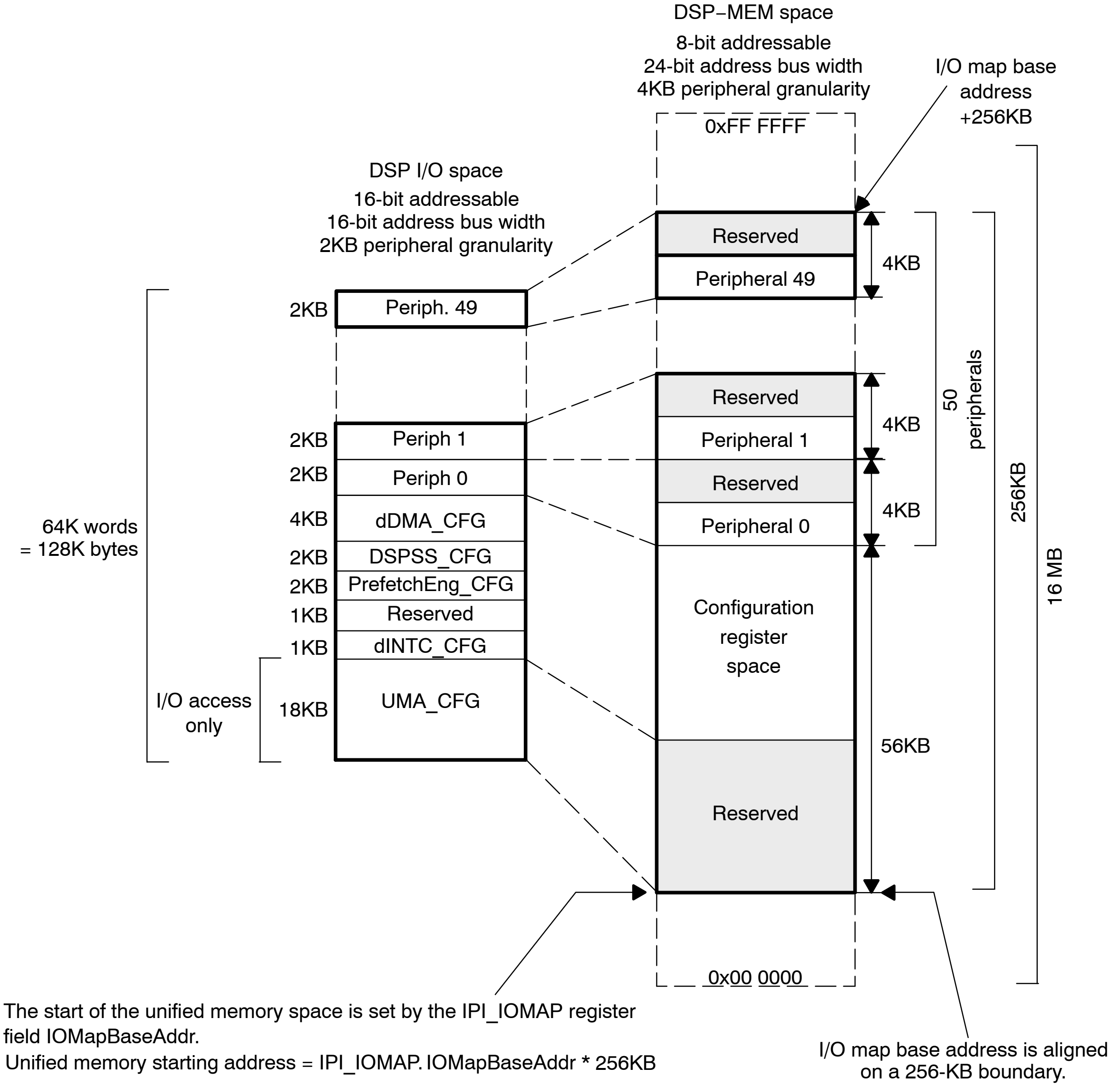
4.3 Functional Description

4.3.1 DSP Memory and I/O Space Overview

The C55x DSP core in the OMAP2420 provides a unified program, data, and I/O memory space.

Figure 4−6 shows how the I/O space is unified in the 16MB DSP memory space.

Figure 4−6. DSP Memory and I/O Space Overlay



The start of the unified memory space is set by the IPI_IOMAP register field IOMapBaseAddr.
Unified memory starting address = IPI_IOMAP. IOMapBaseAddr * 256KB

I/O map base address is aligned on a 256-KB boundary.

Program address space is used by the DSP CPU core to read instructions from memory, while data address space is used for general-purpose memory access. I/O address space is used for two-way communication with DSP peripherals. The DSP memory space is 16M bytes deep, while the I/O space is 128K bytes deep. Table 4−13 shows the DSP I/O space map, and Table 4−14 shows the DSP memory space and how the I/O space is unified in memory space.

Table 4−13. DSP I/O−Space Mapping (8-Bit Addressed)

DSP I/O Space (16-Bit Addresses)[1]		Range Name	Size (K Bytes)	Comment
0x0000	0x47FF	DSP core configuration registers	18	Accessible only with DSP I/O space accesses
0x4800	0x4BFF	DSP INTC	1	
0x4C00	0x4FFF	Reserved	1	
0x5000	0x57FF	DSP core prefetch buffer	2	Internal memory
0x5800	0x5FFF	DSP global configuration registers	2	
0x6000	0x6FFF	DSP DMA	4	
0x7000	0x1FFFE	DSP peripherals	100	External memory

1) I/O space addresses shown in Table 4−13 are byte-aligned. To calculate the appropriate equivalent DSP half-word (16-bit) address, divide the byte-aligned address by 2.

Table 4−14. DSP Memory Space Mapping

DSP Memory Space (Byte Addresses)		Range Name	Size (in K Bytes)	Comment	
0x000000	0x00FFFF	Dual-access RAM	64	DSP local memories	
0x010000	0x027FFF	Single-access RAM	96	DSP local memories	
0x028000	IOMA−1	External memory			
IOMA	IOMA+0x8FFF	Reserved	18	Beginning of remapped I/O area	
IOMA+0x9000	IOMA+0x97FF	DSP INTC configuration	1		
IOMA+0x9800	IOMA+0x9FFF	Reserved	3		
IOMA+0xA000	IOMA+0xA7FF	DSP core prefetch buffer configuration	2		
IOMA+0xA800	IOMA+0xAFFF	Reserved	2		
IOMA+0xB000	IOMA+0xB7FF	DSP core global configuration	2		
IOMA+0xB800	IOMA+0xBFFF	Reserved	2		
IOMA+0xC000	IOMA+0xCFFF	DSP DMA configuration	4		
IOMA+0xD000	IOMA+0xDFFF	Reserved	4		
IOMA+0x0*X*000	IOMA+0x0*X*7FF	External peripheral	2	X = {E…F}	50 external peripherals also accessible through I/O space
IOMA+0x0*X*800	IOMA+0x0*X*FFF	Reserved	2	X = {E…F}	

Table 4−14. DSP Memory Space Mapping (Continued)

DSP Memory Space (Byte Addresses)		Range Name	Size (in K Bytes)	Comment

IOMA+0x*ZY*000	IOMA+0x*ZY*7FF	External peripheral	2	Y = {0..F)
				Z = {1..3}

IOMA+0x*ZY*800	IOMA+0x*ZY*FFF	Reserved	2	Y = {0..F)
				Z = {1..3}

IOMA+0x40000	0xFEFFFF	External peripheral		
0xFF0000	0xFFFFFF	Program/data ROM	64	

IOMA in Table 4−14 represents the base address of the overlaid I/O space on the DSP memory space. For directions for setting the IOMA address, see Section 4.4.4, *Reset Management*.

The I/O page is 256K bytes; however, of this 256KB, only 128KB is usable (because peripherals are mapped to separate 4KB pages). The MPU can adjust the location of the I/O space overlay in DSP memory space by accessing the IPI_IOMAP register in the IPI and setting the I/O-map base address value.

Peripherals can be accessed either through the D-port of the DSP core (through the B, C, D bus of the DSP CPU for reads and E, F bus for writes) with pipeline protection provided by defined software semantics (macros) or by DSP CPU I/O semantics with pipeline protection provided in hardware.

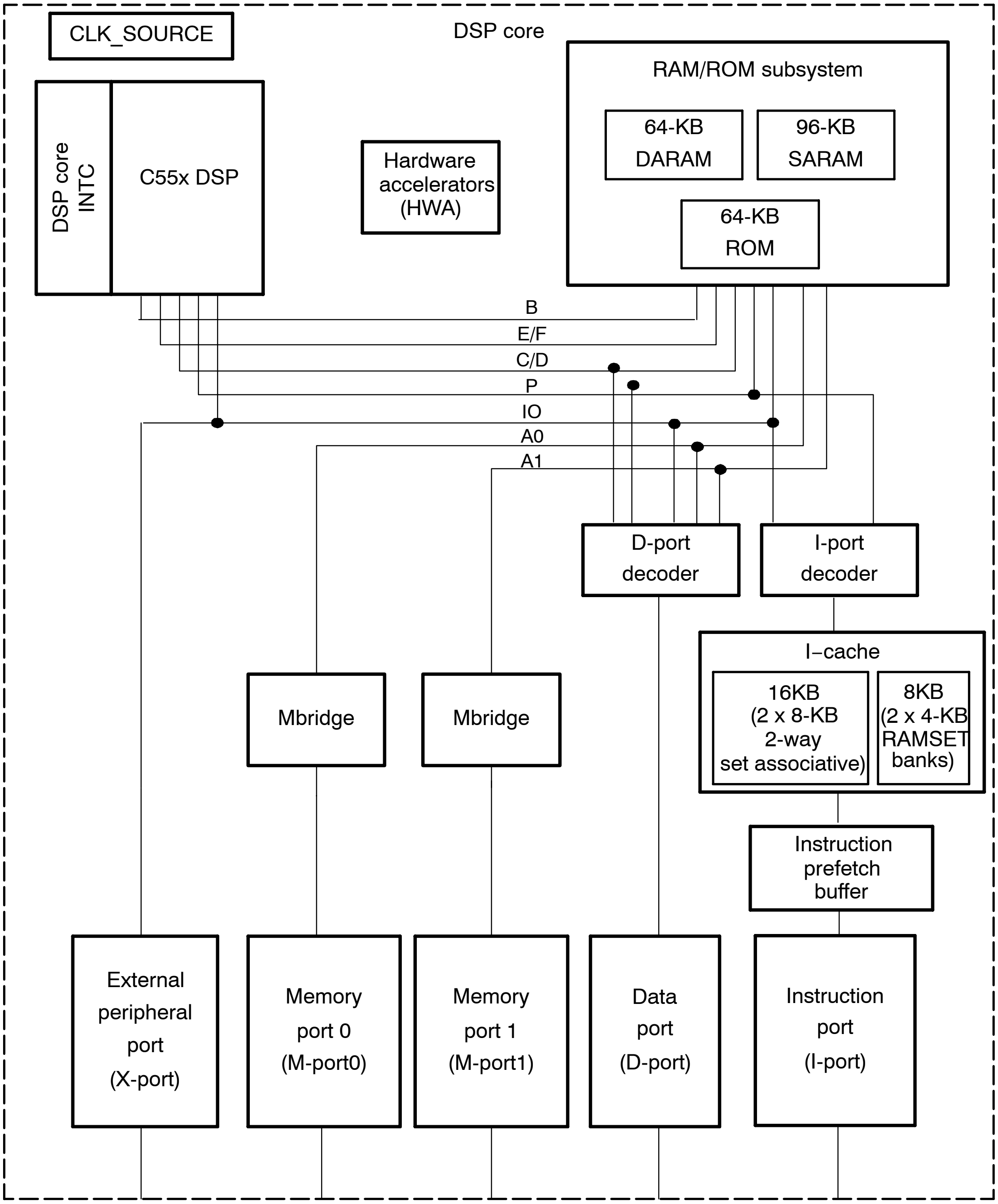
Software must use the DSP core D-port to access all peripherals in unified DSP memory space. DSP core internal configuration registers cannot be accessed through the D port.

4.3.2 DSP Core Overview

The DSP core used in the OMAP2420 architecture is a C55x DSP reusable platform. It is a high-performance low-power fixed-point digital signal processor. It has unified program/data memory architecture with dual MACs, 16MB of total addressable memory space, and 128KB of I/O space. It supports separate program/data interfaces for memory accesses and a peripheral interface for I/O accesses. Internally, there are hardware accelerators, real-time emulation logic, and interrupt control tightly coupled with power control logic.

Figure 4−7 is a block diagram of the DSP core.

Figure 4−7. DSP Core Functional Block Diagram



The C55x DSP accesses data through three 16-bit read-only ports (B, C, and D) and two write-only ports (E and F). The B port is used only to access DSP local memory. The C, D, E, and F ports are arbitrated to access local memories. The C, D, E, and F ports are also arbitrated to access regular memory through the D port (to the memory address space). D and E buses are arbitrated to access the peripheral port (X port) for access to the I/O address space.

The C55x DSP core supports a two-deep pipeline memory protocol. An internal stall/serialization protocol is used to indicate the request status of all memory elements, because many modules are connected to the same bus.

All DSP core modules are directly connected to the internal DSP I/O interface, decode their own I/O memory-mapped registers (MMRs). and support the full DSP I/O protocol. All internal DSP I/O space is zero wait-state.

The following sections describe OMAP2420 DSP core submodule interface features and functions.

4.3.3 DSP Core Interfaces

4.3.3.1 Instruction Port

The instruction port (I-port) has the following features:

- Master port
- 32-bit data bus width
- External interface clock of 1/Nx DSP clock operation N = {1,2}
- 24-bit address width
- Burst support

The I-port is the interface to synchronous system-level program memory. The I port can access all memory space not allocated as internal memory.

The I-port interfaces the I-cache burst protocol with the internal subsystem interconnect, but can also interface directly with the DSP program bus in DSP core configurations that do not include I-cache. When I-cache is disabled, requests go directly to the I-port. The I-port serializes its own DSP accesses with other memory elements, implements the external protocol state-machine, and drives the external bus protocol. The I-port also includes a prefetch engine that hides most of the consecutive miss latency on instruction fetches.

4.3.3.2 Data Master Port

The data master port (D-port) has the following features:

- Master port
- 32-bit data bus width
- External interface clock of 1/Nx DSP clock operation N = {1,2}
- 24-bit address bus width
- Fully pipelined arbitration

The D-port is the interface to synchronous system-level data memory. The D-port can access all memory space not allocated as internal memory.

The D-port arbitrates between DSP C-, D-, E-, F-, and D-port accesses. It serializes the pipelined DSP accesses with other memory elements, implements the external protocol state-machine, and drives the external peripheral bus protocol.

D-port write operations can be programmed to be posted or nonposted through the data port configuration register DCR.WPE bit. (See the DSP core DCR register description in Table 4−66.)

Write-posting control is used by software macros to provide I/O access through the D-port.

Note:

Write posting is not automatically disabled during a read-modify-write operation.

4.3.3.3 D-Port Write-Posting Registers

The DSP core features two 32-bit write-posting registers for the DSP E and F buses with software control (disabled on reset).

The D-port has two write-posting registers, which are freely associated with E and F bus writes. The write-posting registers are used to store the write address and data so that writes can be zero wait-state for the DSP.

A simultaneous read and write to a single D-port causes a read stall while the write is accessed.

4.3.3.4 I/O-Data Master Port

The I/O data master port (X-port) has the following features:

- Master port
- 32-bit data bus width
- External interface clock of 1/Nx DSP clock operation N = {1,2}
- 17-bit address bus width

The X-port is the interface to synchronous system-level peripherals. The X-port can access all I/O space not allocated or reserved for DSP core internal registers.

The X-port arbitrates between DSP I/O writes and reads and X-port accesses. It serializes pipelined DSP accesses with other memory elements, implements the external protocol state-machine, and drives the external peripheral bus protocol.

The X-port uses only write nonposting operations.

4.3.3.5 M0/M1 Ports

The M-ports have the following features:

- Slave port

- 32-bit data bus width (M-port 0)
- 64-bit data bus width (M-port 1)
- External interface clock of 1/Nx DSP clock operation N = {1,2}

The two M-ports are slave interfaces into the DSP core RAM/ROM subsystem. Each M-port can access the entire internal memory space.

4.3.3.6 DSP Core Interface Write Posting and Nonposting

The DSP core supports both posted and nonposted write operations for D-port bus transactions, and only nonposted write operations on the X-port. These terms and their functional differences are described in the following paragraphs.

Posted Write

The initiating core/host (DSP core D-port) is not aware of when the slave completes a write. (The initiator performs a write operation and releases the bus before receiving a write response from the slave.)

Nonposted Write

The initiating core/host (DSP core D- or X-ports) is aware of when the slave completes a write. (The initiator waits for a response from the slave before ending the write operation.)

Nonposted writes are a way to ensure locally that a write operation is effective.

The tradeoff between the two types of write operations is that of improved bus efficiency when using posted writes, at the expense of reliability and coherence in the system; and less efficiency but more reliability and coherence using nonposted writes. Certain DSP subsystem operations require the use of nonposted writes; see Section 4.4.1.3, *DSP Subsystem Active to Idle Mode Transition***.**

Selection of the write-posting mode for the D-port is set by the WPE bit in the D-port DCR register. The default setting for this register is 0 (write posting disabled).

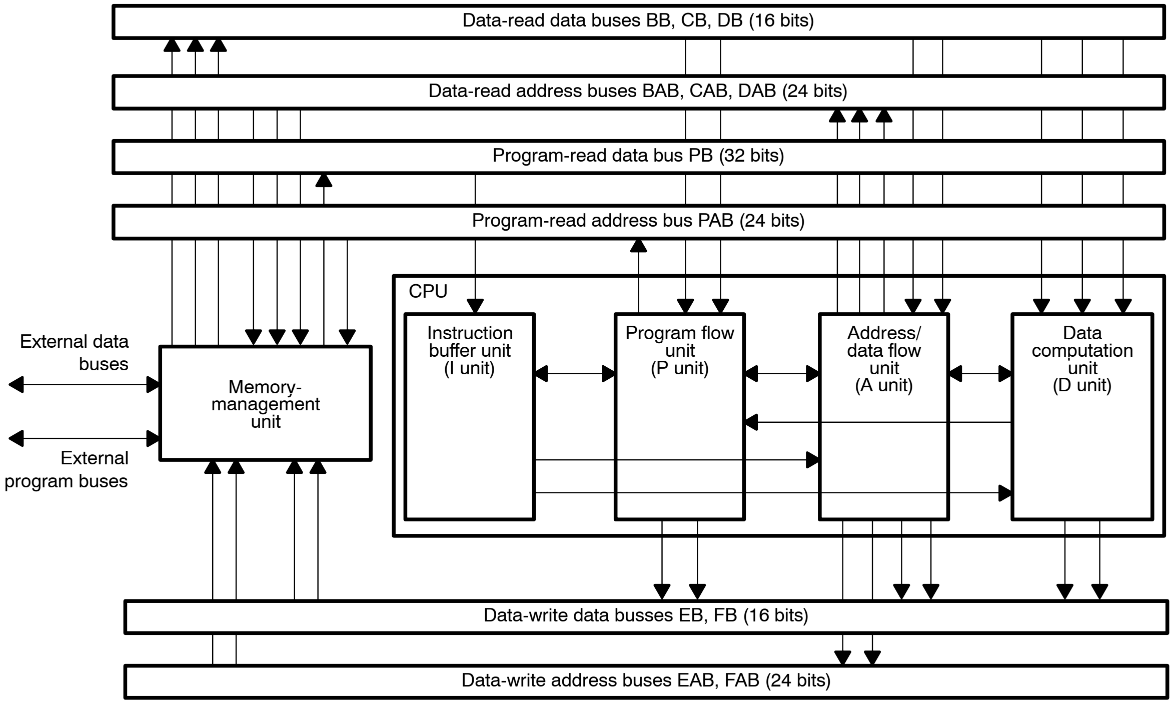
4.3.4 DSP CPU

The functional block diagram in Figure 4−8 shows the principal blocks and bus structure in the C55x DSP core.

The four main blocks are the instruction buffer unit, the program flow unit, the address data flow unit, and the data computation unit. The DSP architecture is built around one 32-bit program bus (P), one 32-bit data bus (B), four 16-bit data buses (C, D, E, and F) and six 24-bit address buses (P, B, C, D, E, and F). CPU program and data spaces share a 16M-byte addressable space. The addressable space also includes an I/O mapped area that uses the same E and D buses, but with separate qualifiers. This is the DSP I/O address space.

To avoid penalties caused by I-port latency, stalled program bus requests can be dismissed for speculative fetches, branches, and interrupts if the instruction buffer queue determines that the request is no longer required.

Figure 4−8. DSP CPU Core (C55x DSP) Block Diagram


Data-read data buses BB, CB, DB (16 bits)
Data-read address buses BAB, CAB, DAB (24 bits)
Program-read data bus PB (32 bits)
Program-read address bus PAB (24 bits)
CPU
External data buses
External program buses
Memory-management unit
Instruction buffer unit (I unit)
Program flow unit (P unit)
Address/ data flow unit (A unit)
Data computation unit (D unit)
Data-write data busses EB, FB (16 bits)
Data-write address buses EAB, FAB (24 bits)

4.3.5 DSP HWA

The OMAP2420 device contains several hardware-acceleration (HWA) modules to improve performance and reduce power consumption for certain computations relating to image and video processing. These coprocessors include the following video hardware accelerators:

- ☐ Discrete cosine transformation (DCT) and inverse discrete cosine transformation (iDCT): ID register value is 0x4E.
- ☐ Enhanced pixel interpolation (EPI): ID register value is 0xFA.
- ☐ Motion estimation calculation (ME): ID register value is 0x0B.

4.3.5.1 Strobe Register

The HWA is statically selected before it is used. A register is implemented (ID = 0xC1) to select which HWA to use. The strobe register is used to select a specific HWA. This selection scheme maintains compatibility with existing software. This strobe register is accessible by specific DSP instructions. It must be written with a value (for example, 0x00 with video HWAs) before using an HWA.

The selection of the hardware accelerator is validated when the HWA instruction strobe signal is activated and allows the decoding of the HWA instruction field. The scheme is shown in Table 4−15.

Table 4−15. Strobe Register

7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Selection When HWA Strobe is Active**
1	1	1	1	1	1	1	0	Select HWA 0 − EPI
1	1	1	1	1	1	0	1	Select HWA 1 − DCT
1	1	1	1	1	0	1	1	Select HWA 2 − ME
1	1	1	1	0	1	1	1	Select HWA 3
1	1	1	0	1	1	1	1	Select HWA 4
1	1	0	1	1	1	1	1	Select HWA 5
1	0	1	1	1	1	1	1	Select HWA 6
0	0	0	0	0	0	0	0	Select standard mode; old selection approach; legacy
1	1	1	1	1	1	1	1	All HWAs idled (optional)

The configurations in Table 4−15 are programmed using regular read and write copr() dataflow modes available in any HWA. For video accelerators, compatibility with the previous scheme (legacy) is ensured by programming 0x00 in this register. When another strobe pattern is written (0xFE, 0xFD, 0xFB, 0xF7, 0xEF, 0xDF, or 0xBF), the corresponding strobe select bit at 0 is connected to the associated HWA.

Bit 7 of the strobe register is also connected to all HWAs to switch between old and new selection mode. If the old selection mode is programmed, the fact that all individual strobes are at 0 allows automatic validation of the HWA interface strobe signal to all HWAs (this ensures complete hardware compatibility with already-designed modules).

Examples 4−1 and 4−2 show an HWA macro for HWA selection.

Example 4−1. Select standard mode (HWA remapper macro)

```
_Hwa_no_remap   .macro
AC0 = #0x4 ; hwremap address
AC1 = #0x0 ; 00000000
AC1 = copr(#0xff,AC0,AC1) ; write access
.endm
```

Example 4−2. Select HWA #1 − DCT

```
_Hwa_no_remap   .macro
AC0 = #0x4 ; hwremap address
AC1 = #0xfd ; 11111101
AC1 = copr(#0xff,AC0,AC1) ; write access
.endm
```

Note:

On C55x DSP HWA, for an EPI ID register value of 0xFA, a bus error interrupt can appear at the first call to EPI when the coprocessor instruction copr() is executed. The pixel interpolation control register (PICR) causes the bus error. If you ignore this error, everything continues to run correctly. To avoid this bus error, the PICR register must be initialized when the use of EPI is required. This is performed by writing the following routine between the HWA remapper macro and the first coprocessor instruction:

```
;routine which initializes PICR
  AC0  = #0x4                   ; address  of PICR
  AC1 = #0x0
  AC1=copr(#0x1f,AC0,AC1)      ; write #0 in PICR
```

The routine below allows reading ADREG for checking whether the EPI ID register value equals 0xFA:

```
;routine which read ID register
  AC0  = #0x2                   ; address  of ADREG
  AC1 = #0x1                    ; address of ID
  AC1=copr(#0x1f,AC0,AC1)      ; write #1 in ADREG
  AC0  = #0x0
  AC1=copr(#0x1e,AC0,AC1)      ; Reads ID
```

4.3.6 DSP Instruction Cache and Prefetch Buffer

The I-cache is a program cache between the DSP and external program memory. It interfaces directly between the DSP and I-port and supports the DSP program bus protocol. Both 2-way set associative and RAMSET cache memories are supported. When the I-cache is disabled, program requests are bypassed directly to the I-port.

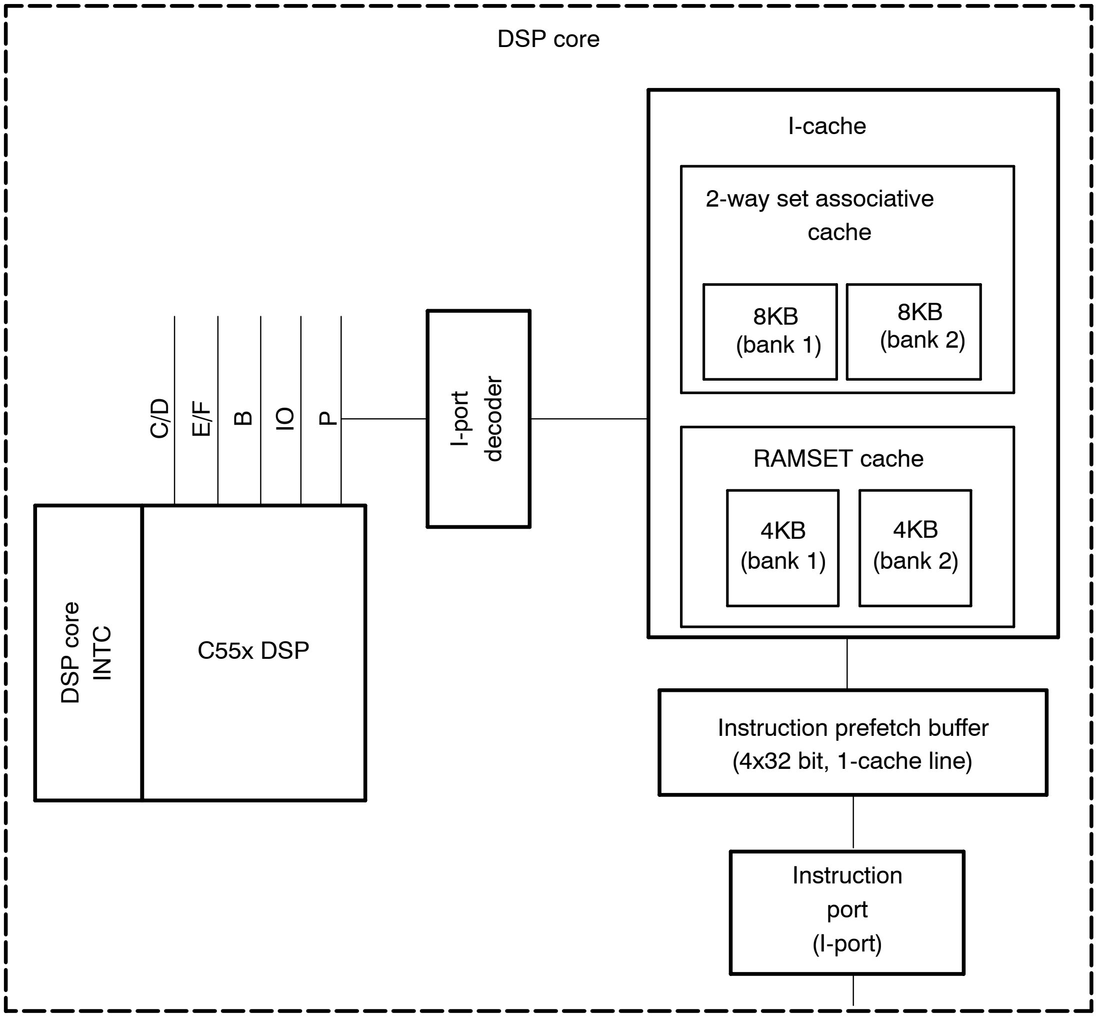
4.3.6.1 I-Cache Features

The I-cache has the following features:

- 2-way set-associative cache. Each way can store 8-KB data.
- 2 RAMSET banks. Each can store 4-KB data.
- The RAMSET banks can be configured into preloaded mode or filled-as-needed mode.
- A fetch request can hit in 2-way set-associative space or RAMSET space.
- If the I-cache has a hit, the data returns without causing DSP to stall.
- The line size of cache and RAMSET is 16 bytes. I-cache returns 4 bytes for each fetch request.
- A missed request is serviced by a 4x4-byte burst sequence to the I-port.
- The first request of the burst for a missed request is issued at the beginning of the missed line. Cache requests are streamed.
- I-cache can be locked while enabled. When locked, the contents in the I-cache and RAMSET do not change as the result of a line miss.
- The entire I-cache, or cache/RAMSET lines corresponding to a specific address, can be flushed while fetching.
- Memories are accessible by DSP I/O accesses when the ICTEST bit of the TMCR1 register is set.
- Supports emulation debug read and breakpoint/watchpoints
- Program dismiss capable

Figure 4−9 is a block diagram of the I−cache.

Figure 4−9. I-Cache Block Diagram



The I-cache is enabled by setting the CAEN bit of the ST3_55 register in the DSP core. When the I-cache is disabled, all program requests are bypassed through it.

The I-cache supports 16KB of 2-way set-associative, and 2 x 4 KB of RAMSET cache space. The default configuration is 16KB of 2-way set-associative cache space, with the RAMSET enabled.

The 2-way set-associative cache is always enabled when the I-cache is enabled. To enable RAMSET, the I-cache RAMSET control registers (ICRCR1/2) and the I-cache RAMSET tag registers (ICRTR1/2) must be configured. When RAMSET is enabled, a RAMSET hit has priority over 2-way set-associative cache space.

In RAMSET preload mode, I-cache automatically preloads and validates all the lines of a RAMSET before validating the RAMSET tag. In filled-as-needed mode, I-cache validates the tag of a RAMSET, but keeps all the lines in the RAMSET invalid. A RAMSET line is filled and validated only when DSP requests this line.

In all hits, the I-cache returns data without causing the DSP to stall. In the case of a miss, a burst of 4 data is sent to the I-port to be serviced. Data returned by the I-port is checked when a new request arrives, stored, and forwarded without stall.

I-cache contents are frozen when the CAFRZ bit of the ST3 register is set, and can be cleared by setting the CACLR bit of the ST3 register (qualified by the GF bit of the IGCR register).

If the I-cache is disabled, requests are bypassed through the I-cache and sent directly to the I-port. In this case, the I-cache buffers only requests to the I-port.

4.3.6.2 Cache Set-Associative Memory Organization

The 2-way set-associative cache configuration is organized as two 8K-byte parallel blocks of memory. The size of the I-cache RAM block and cache line determines the number of lines in the cache. Each 8K-byte memory block consists of 512 cache lines containing 16 bytes (4 words) of instructions from consecutive addresses. Control and status bits are associated with each cache line to support virtual-to-physical mapping and cache control/operation.

A 24-bit program virtual address is parsed into offset, line index, and tag sections, as shown in Figure 4−10. The line index determines which cache entry to test. The tag is used to check the cache contents against the requested instruction address. The offset field determines which bytes on the cache line to fetch when a cache hit occurs. Because the 2-way set-associative cache structure has two 8K-byte RAM blocks in parallel, two cache lines and two tags are associated with each line number defined by the index. When an instruction is written in the cache, its tag is also written.

Figure 4−10. Virtual Address Mapped to Cache Line (Set Associative Mode)

23 13	12 4	3 2
Tag	Index	Offset

4.3.6.3 RAMSET Memory Organization

RAMSET memory is a special type of cache memory. The 8K-byte memory block is divided into two 4K-byte sections that can be controlled independently. Because each RAMSET memory block is 4K bytes long and the line size is four words, there are 256 lines in each module. This makes the index field size 8 bits and the tag field size 12 bits. In RAMSET memory there is only a single tag for each 4K-byte block, as opposed to each line in regular cache memory. This organization allows dynamic remapping of 4K-byte memory blocks from external to internal memory (once a RAMSET memory block is filled).

Figure 4−11. Virtual Address Mapped to Cache Line (RAMSET Mode)

23 12	11 4	3 2
Tag	Index	Offset

4.3.6.4 Idling the I-Cache

The I-cache must be disabled before it is placed in idle mode. The I-cache enable/disable control is in the DSP core memory-mapped register ST3_55, at

DSP memory space word address 0x04. The I-cache is put in idle mode by setting the I-cache bit of the ICR register and executing the IDLE instruction.

4.3.6.5 DSP Core Instruction Prefetch Buffer

DSP CPU performance is degraded when the program executed is not in cache. In this case, the instructions must be fetched from memory that is external not just to the DSP subsystem, but usually also to the OMAP chip. When there is a cache miss, there can be a substantial delay between the CPU issuing a request and the delivery of the data. Furthermore, the CPU waits until delivery of the data before issuing further requests. The purpose of the instruction prefetch unit is to reduce this delay, by speculatively fetching program data from external memory. If the CPU subsequently requests this data, it is available with a much shorter delay.

The prefetch unit buffer holds one cache line organized as 4- x 32-bit memory. The instruction prefetch unit does not decode instructions, but simply fetches lines at the addresses directly after the last address requested by the CPU/I-cache.

The prefetch unit attempts to read data only from areas of memory (pages) that have already been read by the CPU/I-cache through the I-port. This ensures that the prefetch unit does not cause memory access errors.

4.3.7 DSP RAM/ROM Subsystem Overview

The DSP RAM/ROM subsystem (RSS) is the tightly-coupled memory of the DSP core. It consists of three memory channels:

- ☐ The DARAM channel (64K bytes) is a zero wait-state memory that supports up to two memory accesses per DSP CPU clock cycle into each RAM block. Accesses can be made from any internal data, program, or DMA bus. DARAM memory consists of eight blocks of 8K bytes each.
- ☐ The SARAM channel (96K bytes) is a zero wait-state memory that supports one memory access per DSP CPU clock cycle into each RAM block. This access can be a 32-bit value. Accesses can be made from any internal data, program, or DMA bus. SARAM memory consists of 12 blocks of 8K bytes each.
- ☐ The PDROM channel (64K bytes) is a single wait-state memory that supports one memory read in two DSP CPU clock cycles. This access can be a 32-bit value. Accesses can be made from any internal data read or program bus. The PDROM memory consists of one block of 64K bytes.

The RSS supports software control of DSP versus M-port arbitration. The DSP and M-port are guaranteed equal access priority to memories by setting the PRIODMA bit of the CONFIG register to 1. These accesses are interleaved between the DSP and the M-port. When either performs an access, the arbitration shifts to the other because it now has the highest priority. Memory arbitration is performed for each memory element (block) in the channel, not simply for each channel unit.

If a DSP memory read/write conflict occurs, the write access is stored in a buffer and the read is allowed. The write buffer becomes a pending write request. If there is a future read at the pending address, the data is read from the buffer instead of the memory. If there is another write, the write pending buffer is flushed. DSP write pending buffers are flushed before an M-port access.

4.3.7.1 RAM/ROM Subsystem Standby Mode

The SARAM/DARAM cores in the RAM subsystem and I-cache support an ultralow-leakage standby mode that is automatically invoked under the following conditions:

- When the DSP and the M-port are both idle, the memories in the RAM subsystem are put in standby.
- When the I-cache is idle, the SARAM memories associated with the RAMSET, set-associative, and tags are put in standby.

The MSEN bit in the IMSCR register (memory standby enable, bit 0, IMSCR I/O register, 0x03) allows a second level of control of whether memories enter standby mode. When set to 0, I-cache memories enter standby mode when the I-cache is idle, or SARAM/DARAM memories enter standby mode when the DSP and the M-port are both idle. When set to 1, memory standby state is disabled, regardless of idle conditions. At reset, MSEN is set to 0, enabling memory standby during idle.

4.3.8 DSP Core Memory Port Bridge

The DSP core memory bridge (M bridge) provides communication between DSP subsystem modules (initiator) and memories integrated in the DSP core. Because the DSP core clock frequency can be 2 times that of the other DSP subsystem peripherals, the M bridge handles clock resynchronization between these modules. The M bridge also supports burst packing/unpacking of request/response to convert from the 64-bit subsystem interconnect to the 32-bit memory data bus and vice versa, optimizing the bandwidth of the connection.

The M bridge also supports communication between an initiator and a target running at the same frequency. In this case, bandwidth on the 64-bit bus is divided by 2.

4.3.9 DSP DMA Overview

The DSP subsystem includes a DMA controller based on a generic module that is also used for the OMAP2420 system DMA (sDMA). The DSP DMA allows offloading of the task of managing memory transactions between system peripherals and DSP internal memory and thus frees CPU resources for other tasks. The DSP DMA has the following properties and capabilities:

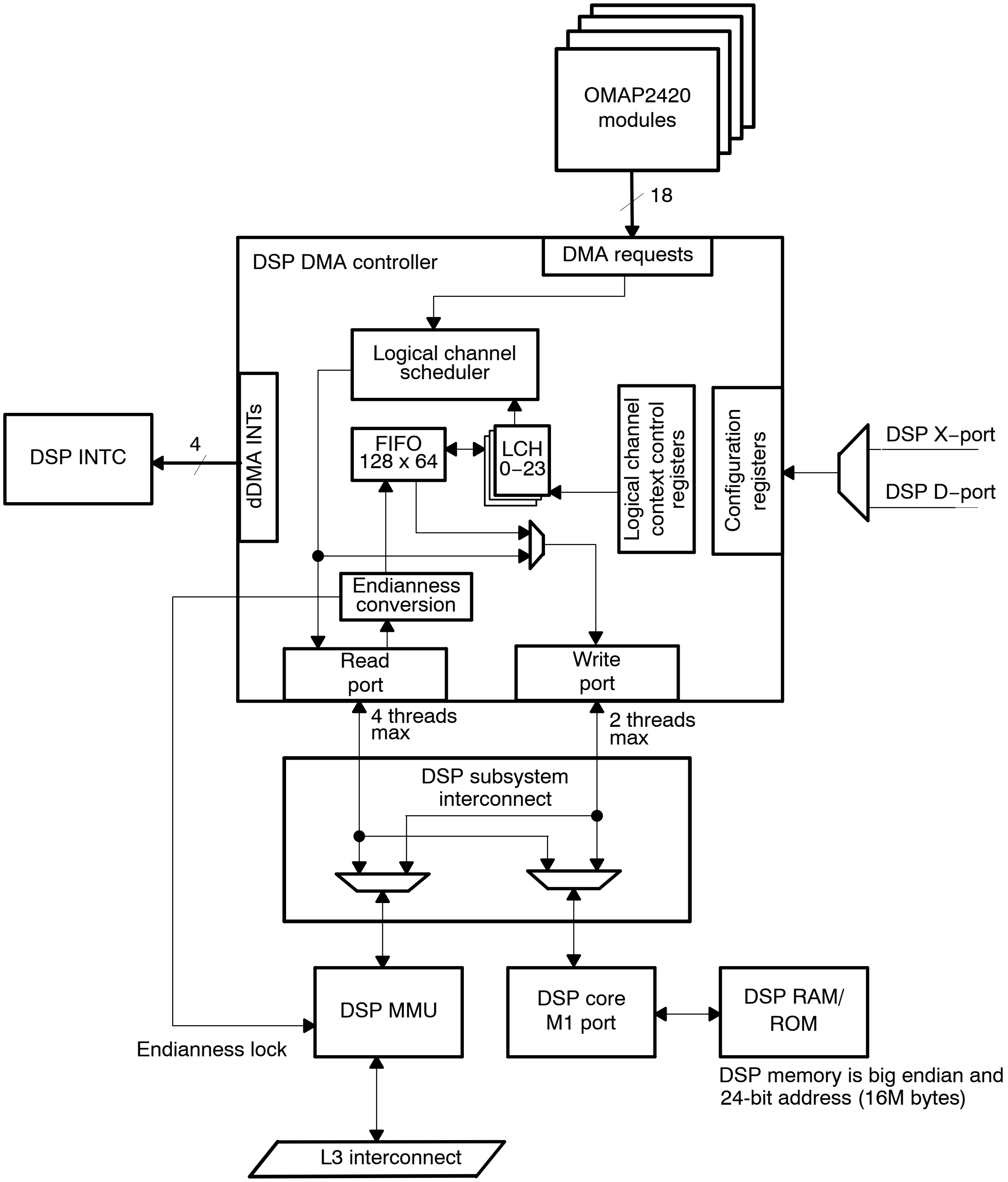
- Support of 24 logical channels
- 64-bit data width/24-bit addresses

- One dedicated write multithreaded port (2 threads)
- One dedicated read multithreaded port (4 threads)
- One configuration port (through DSP)
- 8-bit, 16-bit, and 32-bit data packing over a 64-bit bus
- Dynamic thread allocating policy (transaction-based)
- Nondestructive emulation reads to the configuration
- Endianness conversion and lock
- Independent read/write context allowing more efficient buffering, pipelining, and prefetch
- Support of packet transfer and end-of-packet transfer interrupt

DSP DMA transactions have one source (from which DMA reads) and one destination (to which DMA writes) per channel. The source can be internal memory or external memory, and the destination can be in internal memory or external memory (independent of source). Internal memory accesses go directly to DSP local SARAM/DARAM memories, while external memory accesses go through the DSP MMU. Internal DSP memory accesses must use big-endian format, so if conversion is required by external system memory or the CPU, it must be performed using MMU or DMA conversion logic.

Figure 4−12 shows the DMA controller integrated in the DSP subsystem. For information about the functionality of the DSP subsystem DMA module, including interrupts, power-saving modes, and register descriptions, see Chapter 10, *DMA*.

Figure 4−12. DSP Subsystem DMA Controller



The DSP DMA can signal to the DSP MMU that endianness conversion has been performed (or does not need to be performed) by locking the endianness. This effectively disables any further endianness conversion logic after the DMA in the DSP subsystem data path.

DSP DMA accesses to both internal and external memories must always use the lower 24 bits of the 32-bit address field and also must be ordered as big-endian. (This internal memory access occurs through the DSP M1 port.)

4.3.10 DSP MMU Overview

The DSP MMU communicates DSP accesses from the DSP core to the L3 interconnect, mapping the 16M bytes of the DSP virtual addresses to any place in the 4G-byte address space of the OMAP2420 (the 16M-byte mapping may not be in a single contiguous block of physical memory space). At reset, the MMU is disabled and the DSP external memory space is mapped to the first 16M bytes of system memory (the GPMC memory space, external flash, or SRAM memory.

The DSP MMU has two primary goals:

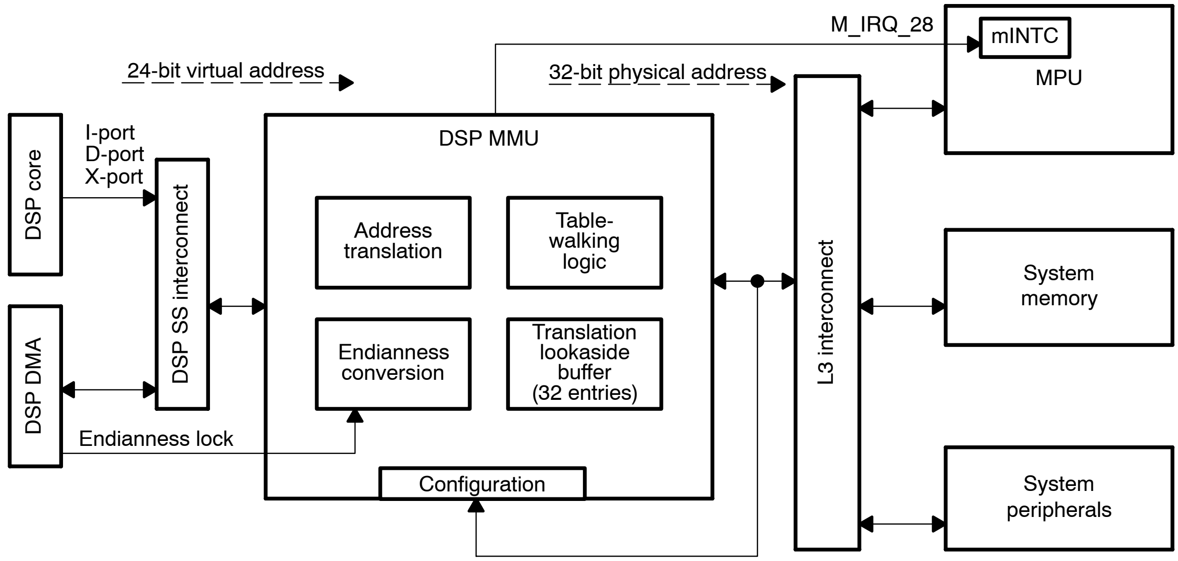
- Translate the DSP internal (virtual) addresses into OMAP (physical) addresses
- Perform endianness conversion when required

The DSP MMU has the following primary features:

- 32 entries/fully associative translation lookaside buffer (TLB)
- 1 interrupt line to the MPU
- 24-bit virtual addresses –> 32-bit physical addresses
- Endianness conversion handling
- 1 translation/cycle, 1-cycle pipeline
- ARM V6 architecture model compatible
- 4KB, 64KB pages, 1MB section, 16MB supersection
- Predefined (static), software-driven (interrupt-based), or table-driven (hardware table-walker) software translation strategies

Figure 4−13 shows the integration and functionality of the DSP MMU module.

Figure 4−13. DSP MMU Block Diagram



4.3.10.1 MMU VA to PA Translation

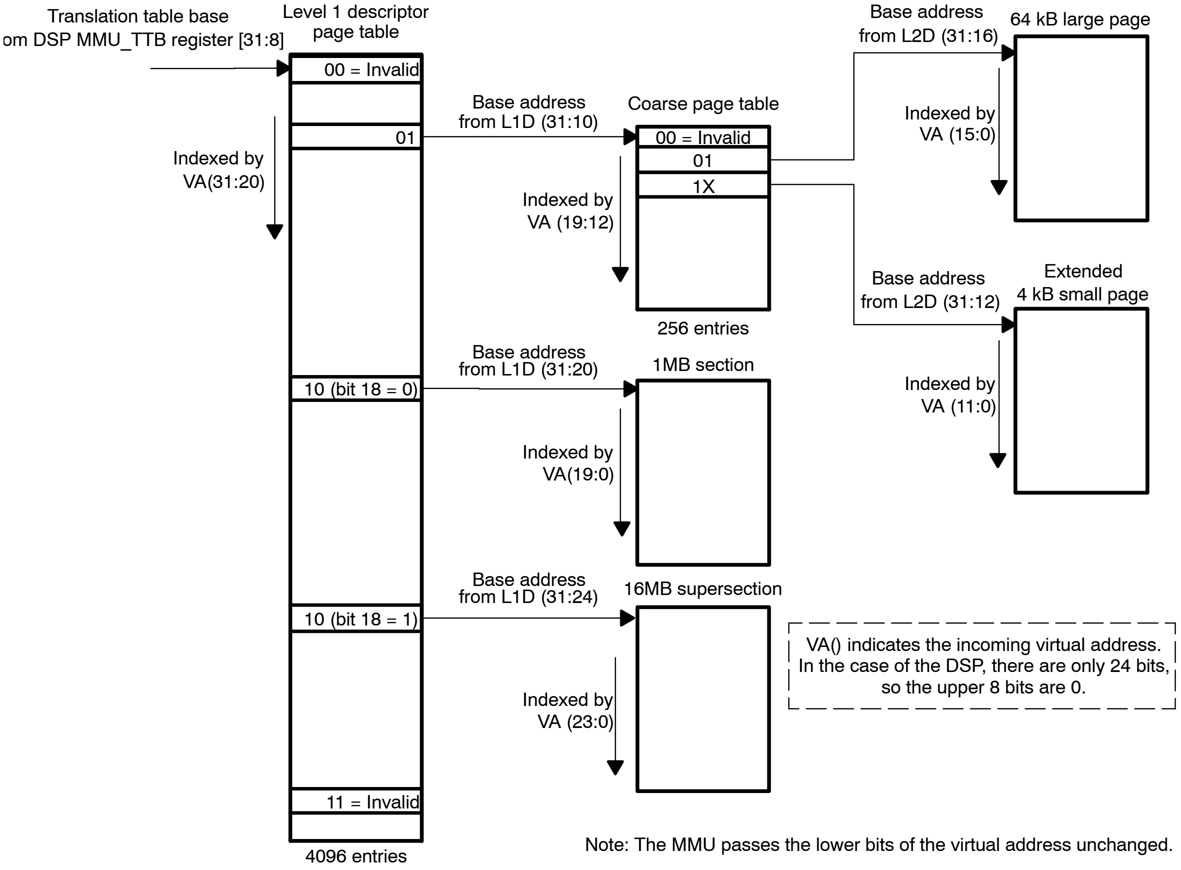
The DSP MMU translates the 24-bit DSP external addresses (0x028000 to IOMA-1, IOMA+0x00E000 to IOMA+0x040000, and IOMA+0x040000 to 0xFEFFFF) to physical addresses in the 32-bit MPU address space. Address translation is performed by a translation table structure (TTB) that maps the most-significant bits (MSBs) of the DSP byte address onto another set of MSBs of a 32-bit MPU byte address. The least-significant bits (LSBs) of the DSP-generated byte address are used as page/section indices in the address translations and are not altered when forming the new 32-bit physical address. The TTB translations are expedited by the TLB, which serves as a cache of recently used page translations. Address mapping can be programmed at the TTB level or by writing the TLB entries directly. The DSP MMU contains 32 TLB entries that can be configured to remap 4KB, 64KB, 1MB, or 16MB segments of memory.

The TLB can be managed statically or dynamically (through the use of interrupts) by the MPU OS, but the MMU also includes hardware table-walking logic that allows the MMU to autonomously traverse the page table on a TLB miss. On a TLB miss, the translation table-walking logic automatically retrieves the information from the translation table stored in physical memory and updates the TLB cache. Figure 4−14 describes the TTB structure in detail.

If the DSP MMU requires software intervention, the MPU services the event; DSP MMU service requests are signaled to the MPU with a dedicated interrupt, M_IRQ_28. The DSP MMU is programmed by the MPU. In general, the MMU is initialized at boot time, but it can also be reprogrammed dynamically. The MMU is programmed through its configuration port on the L3 interconnect. DSP MMU configuration registers are at MPU base address 0x5A00 0000.

For more information about the functionality of the DSP subsystem MMU module, including interrupts, register descriptions, and the programming model, see Chapter 9, *Memory Management Units*.

Figure 4−14. DSP MMU Translation Table Hierarchy


Translation table base
om DSP MMU_TTB register [31:8]
Level 1 descriptor page table
00 = Invalid
01
10 (bit 18 = 0)
10 (bit 18 = 1)
11 = Invalid
4096 entries
Indexed by VA(31:20)
Base address from L1D (31:10)
Coarse page table
00 = Invalid
01
1X
256 entries
Indexed by VA (19:12)
Base address from L1D (31:20)
1MB section
Indexed by VA(19:0)
Base address from L1D (31:24)
16MB supersection
Indexed by VA (23:0)
Base address from L2D (31:16)
64 kB large page
Indexed by VA (15:0)
Base address from L2D (31:12)
Extended 4 kB small page
Indexed by VA (11:0)
VA() indicates the incoming virtual address. In the case of the DSP, there are only 24 bits, so the upper 8 bits are 0.

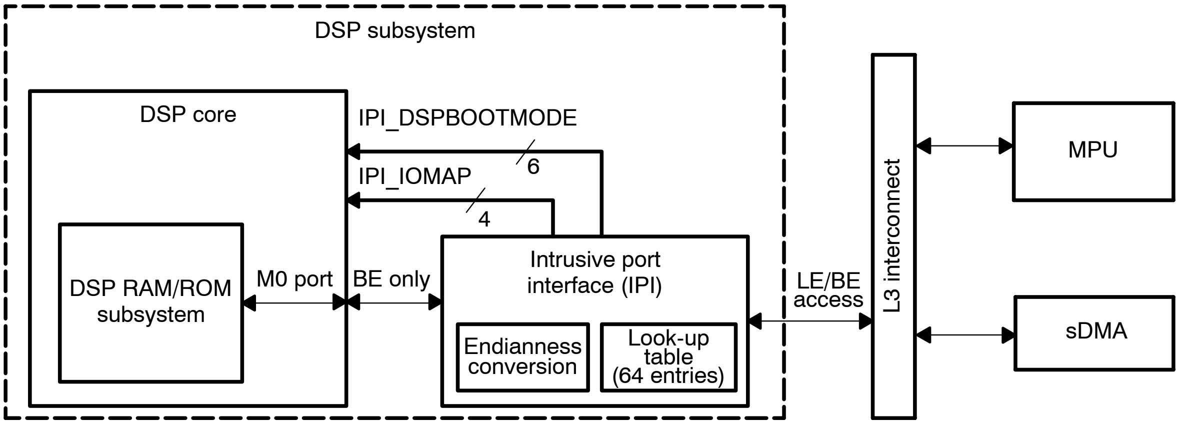
Note: The MMU passes the lower bits of the virtual address unchanged.

See Chapter 9, *Memory Management Units*, for more information about DSP MMU operation and the programming model.

4.3.11 DSP IPI Overview

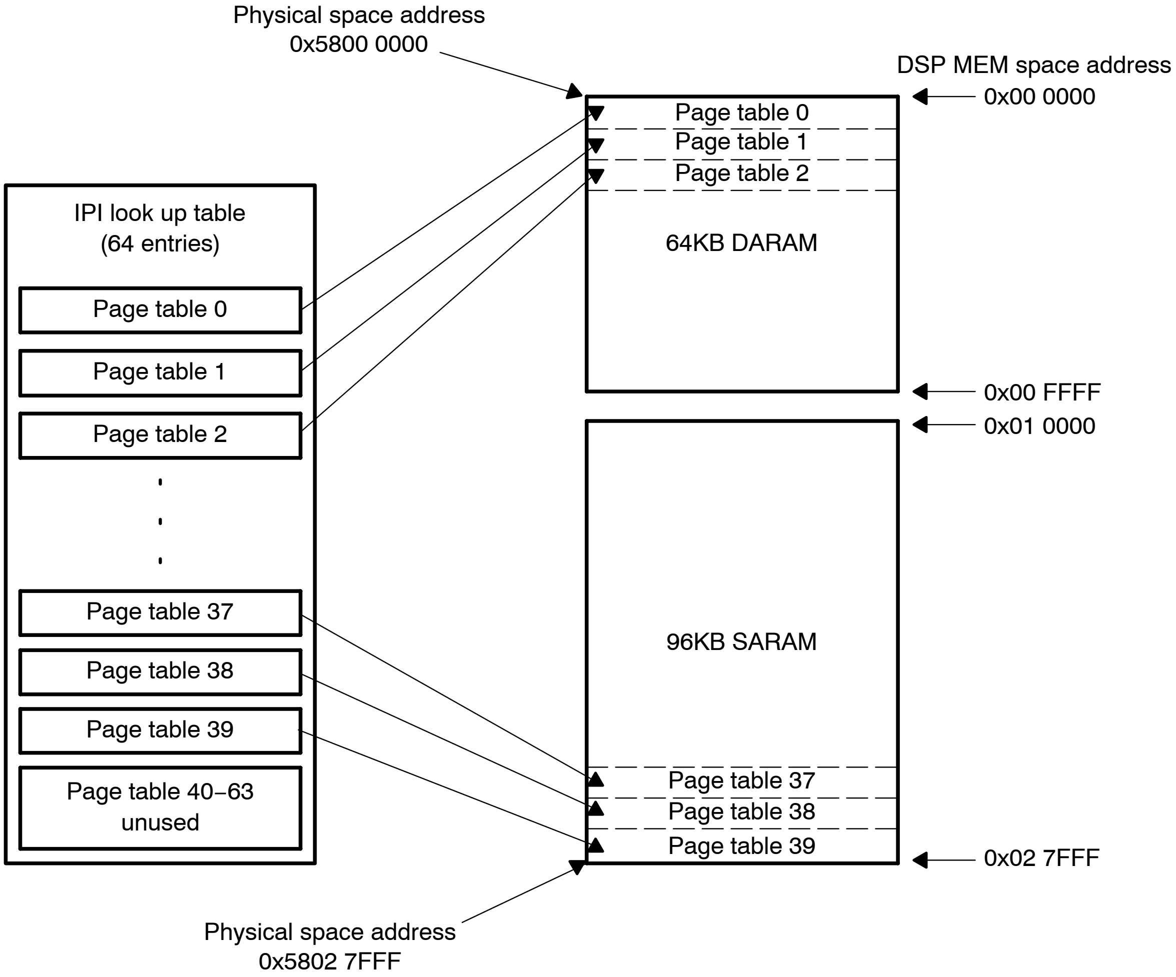
The DSP subsystem connects to the L3 system interconnect through the intrusive port interface (IPI). The IPI is the interface for the MPU and the sDMA controller to access internal DSP subsystem SARAM and DARAM memories, facilitating software downloads and data transfers. In the case of MPU little-endian configurations, the IPI performs the required endianness conversion from MPU little-endian to DSP big-endian. The IPI also communicates the selected DSP boot mode and DSP I/O-space base address to the DSP core through MPU programmable registers. Figure 4−15 shows how the IPI block interfaces with the DSP core and the L3 system interconnect.

Figure 4−15. Intrusive Port Interface Block Diagram



The internal memories of the DSP subsystem are all big-endian, but they can be externally accessed by little-endian or big-endian system accesses. The information useful for endianness conversion is available on a page-by-page basis. The SARAM/DARAM DSP memory space is divided into forty 4-KB pages by the IPI. Each page is assigned an element size attribute by MPU software, and this element size attribute is stored in a look-up table in the DSP IPI. When the MPU or sDMA accesses DSP internal memory (through physical address 0x5802 7FFF), the IPI uses the stored element size for the addressed page to correctly perform endianness conversion (if required). Because the DSP internal memories are always big-endian, the only page table attribute required is the element size. The IPI endianness conversion logic supports 8-bit, 16-bit, and 32-bit element sizes.

Figure 4−16. DSP IPI Look−Up Table Mapping



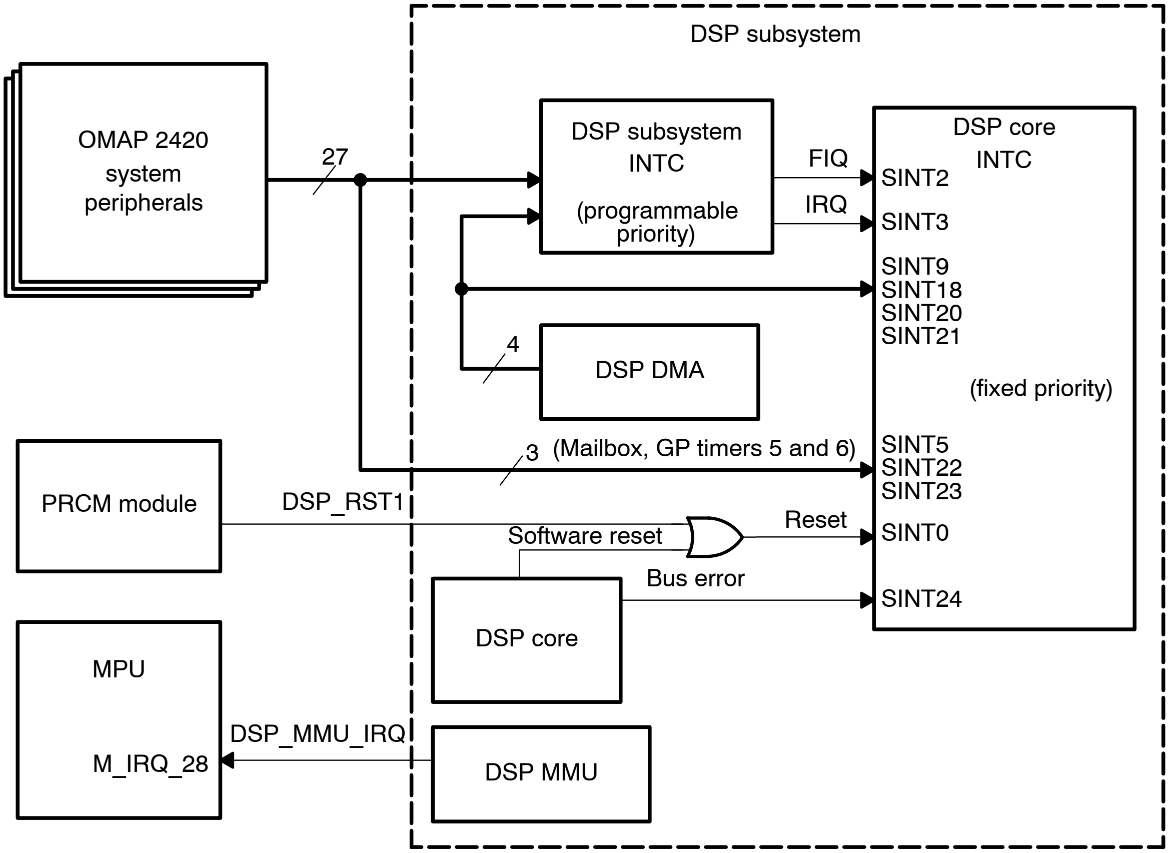
The IPI has visibility over 256KB of DSP local memory (64 pages), but it does not check for validity of access. In case of an access to unmapped memory, the access is propagated to the master port with or without endianness conversion, depending on the configuration of the IPI and on the endianness qualifiers of the request on the slave port, exactly as with a valid access. The existence of the accessed location is checked in the DSP core M-port and an error is reported.

DSP private peripherals; that is, DSP INTC, DSP DMA, are not accessible through the IPI.

4.3.12 DSP INTC Overview

The OMAP2420 DSP subsystem has two cascaded levels of interrupt control, one level integrated with the DSP core and the other integrated at the DSP subsystem level. Figure 4−17 shows how the two interrupt controllers connect internally and to the other system peripherals.

Figure 4−17. DSP Subsystem Interrupt Controllers



4.3.12.1 DSP Subsystem INTC

The DSP subsystem interrupt controller is based on a generic module instantiated in two other places in the OMAP2420 design (this same base INTC module is also used in the MPU). This INTC module supports up to 32 external interrupts; however, the DSP instantiation of this INTC uses only 31 of the 32 available interrupts. Each of the interrupts can be routed to one high-priority (FIQ) or one low-priority (IRQ) interrupt output. The interrupt input-to-FIQ/IRQ routing is programmable on a per-interrupt basis.

As shown in Figure 4−3, the DSP subsystem INTC is connected to a power domain that is separate from the rest of the DSP subsystem. This allows the majority of the DSP subsystem logic to be sent to a lower power state while keeping the INTC powered for monitoring potential DSP wake-up events. For more information about the functionality of the DSP subsystem INTC module, including interrupt processing, masking, power saving modes, and register descriptions, see Chapter 11, *Interrupt Controller*.

4.3.12.2 DSP Core INTC

The DSP core INTC supports nine asynchronous maskable interrupts, a bus error interrupt, a reset interrupt, and three emulation interrupts. All external interrupts are internally synchronized with the DSP core clock using two synchronizing registers. All are level−sensitive and must be explicitly cleared by software.

4.3.13 DSP Subsystem Endianness Overview

The DSP subsystem can convert the endianness of data transferred to/from DSP big-endian memory and little-endian-configured system memory. Both the DSP MMU and the DSP DMA modules are capable of performing endianness conversion on data associated with DSP write and read accesses to external system memory. Also, the DSP subsystem interface for direct MPU access to DSP memories, the IPI, can perform endianness conversion for applications where the MPU memory is configured as little-endian.

Figure 18 and Figure 19 show the concepts of big- and little-endianness used in this document.

Figure 4−18 shows an example of how byte-scalar data stored in memory is viewed by the DSP (big-endian) and the MPU (with external memory configured as little-endian).

Figure 4−18. Endianness and Memory Views for Byte Scalar Data

External memory (scalar = byte)			
D[31:24]	D[23:16]	D[15:8]	D[7:0]
A	B	C	D

DSP view of external memory

Byte 0 (MSB)	Byte 1	Byte 2	Byte 3 (LSB)
A	B	C	D

MPU view of memory

Byte 3 (MSB)	Byte 2	Byte 1	Byte 0 (LSB)
A	B	C	D

Figure 4−19 shows an example of how 16−bit scalar data in memory is viewed by the DSP and MPU.

Figure 4−19. Endianness and Memory Views for 16-Bit Scalar Data

External memory (scalar = 16 bit)	
D[31:16] (MSW)	D[15:0] (LSW)
AB	CD

MSW = Most-significant word
LSW = Least-significant word

DSP view of external memory

Word 0 (MSW)	Word 1 (LSW)
AB	CD

MPU view of memory

Word 1 (MSW)	Word 0 (LSW)
AB	CD

4.3.13.1 DSP DMA Endianness Conversion

The DSP DMA endianness conversion logic is fully programmable to support various combinations of per-channel source/destination endianness settings, and locked or unlocked endianness for the source or the destination. Destination and source endianness settings for each of the 24 DMA channels are programmable in the DMA4_CSDP register (offset 0x90 +[N*0x60], where N = channel number) in the DSP DMA register space. If the source and destination endianness settings for a given channel match, no endianness conversion is performed. If there is a mismatch, endianness conversion is applied on each transfer, big-endian to little-endian or vice versa, based on the underlying scalar type of the transfer.

4.3.13.2 DSP DMA Endianness Locking

When endianness conversion must be handled only by the DMA controller, the DMA allows endianness to be locked for either the DMA source or DMA destination interfaces (or both), thus informing other system-level modules (such as the DSP MMU) to lock the endianness; this effectively disables any further endianness conversion logic after the DMA. Endianness lock can be applied to the DMA channel source or destination (or both) by the DMA4_CSDP.SRC_ENDIANNESS_LOCK and DMA4_CSDP.DST_ENDIANNESS_LOCK register bit fields in the DSP DMA register space.

4.3.13.3 DSP MMU Endianness Conversion

The DSP subsystem MMU can convert the endianness of data transferred to/ from DSP big-endian memory and little-endian-configured system memory. DSP MMU endianness conversion is based on the properties for the page accessed (found in the MMU TLB cache entries) and underlying qualifiers with the data transferred. Page table properties, including endianness, element size, and mixed-data attributes, are set by system software running on the MPU. (See Chapter 9, *Memory Management Units*, for more information about setting MMU page table properties). When the DSP MMU is disabled, no endianness conversion is performed by the MMU.

The DSP DMA is one potential initiator to the DSP MMU, and the DMA module can also perform endianness conversion. If the DSP DMA endianness for a specific channel is not locked, the DSP MMU endianness settings override those of the DMA channel and perform any required endianness conversion, based solely on MMU page table data and incoming data qualifiers. If the endianness is locked by the DSP DMA, the MMU does not perform endianness conversion for those DSP DMA channel-initiated transfers.

4.3.13.4 MMU Endianness Conversion Examples

Table 4−17 through Table 4−30 list MMU endianness conversion result examples for DSP write and read operations to various possible configurations of system memory (through MMU TLB page tables).

For the sample results in Table 4−17 through Table 4−30, the data in external memory is assumed to match that shown in Table 4−16.

Table 4−16. Memory Values

External Memory			
31:24	23:16	15:8	7:0
A	B	C	D

For example, if the DSP performs a 32-bit read access to system memory address 0 (which holds data as shown in Table 4−16) and that section of memory is configured, according to its page table attributes, to be little-endian and have an element size of 2 bytes (as shown, for example, in Table 4−17), the DSP read returns value CDAB (where CD is MSB and A is LSB).

DSP MMU 32-Bit Access Endianness Examples

Table 4−17. MMU 32-Bit Access, Little-Endian System Memory

32-Bit Access to Address 0	DSP MMU Page Table Configuration			DSP Access Result	MPU Access Result
	Endianness	Element Size	Mixed Data		
	Little	1 byte	0	DCBA	ABCD
	Little	2 bytes	0	CDAB	ABCD
	Little	4 bytes	0	ABCD	ABCD
	Little	X	1	ABCD	ABCD

Table 4−18. MMU 32-Bit Access, Big-Endian System Memory

32-Bit Access to Address 0	DSP MMU Page Table Configuration			DSP Access Result	MPU Access Result
	Endianness	Element Size	Mixed Data		
	Big	1 byte	0	ABCD	ABCD
	Big	2 bytes	0	ABCD	ABCD
	Big	4 bytes	0	ABCD	ABCD
	Big	X	1	ABCD	ABCD

DSP MMU 16-Bit Access Endianness Examples

Table 4−19. MMU 16-Bit Access to Address 0, Little-Endian System Memory

16-Bit Access to Address 0	DSP MMU Page Table Configuration			DSP Access Result	MPU Access Result
	Endianness	Element Size	Mixed Data		
	Little	1 byte	0	DC	CD
	Little	2 bytes	0	CD	CD
	Little	4 bytes	0	AB	CD
	Little	X	1	CD	CD

Table 4−20. MMU 16-Bit Access to Address 2, Little-Endian System Memory

16-Bit Access to Address 2	DSP MMU Page Table Configuration			DSP Access Result	MPU Access Result
	Endianness	Element Size	Mixed Data		
	Little	1 byte	0	BA	AB
	Little	2 bytes	0	AB	AB
	Little	4 bytes	0	CD	AB
	Little	X	1	AB	AB

Table 4–21. MMU 16-Bit Access to Address 0, Big-Endian System Memory

16-Bit Access to Address 0	**DSP MMU Page Table Configuration**			**DSP Access Result**	**MPU Access Result**
	Endianness	**Element Size**	**Mixed Data**		
	Big	1 byte	0	AB	CD
	Big	2 bytes	0	AB	CD
	Big	4 bytes	0	AB	CD
	Big	X	1	AB	CD

Table 4–22. MMU 16-Bit Access to Address 2, Big-Endian System Memory

16-Bit Access to Address 2	**DSP MMU Page Table Configuration**			**DSP Access Result**	**MPU Access Result**
	Endianness	**Element Size**	**Mixed Data**		
	Big	1 byte	0	CD	AB
	Big	2 bytes	0	CD	AB
	Big	4 bytes	0	CD	AB
	Big	X	1	CD	AB

DSP MMU 8-Bit Access Endianness Examples

Table 4–23. MMU 8-Bit Access to Address 0, Little-Endian System Memory

8-Bit Access to Address 0	**DSP MMU Page Table Configuration**			**DSP Access Result**	**MPU Access Result**
	Endianness	**Element Size**	**Mixed Data**		
	Little	1 byte	0	D	D
	Little	2 bytes	0	C	D
	Little	4 bytes	0	A	D
	Little	X	1	D	D

Table 4−24. MMU 8-Bit Access to Address 1, Little-Endian System Memory

8-Bit Access to Address 1	DSP MMU Page Table Configuration			DSP Access Result	MPU Access Result
	Endianness	**Element Size**	**Mixed Data**		
	Little	1 byte	0	C	C
	Little	2 bytes	0	D	C
	Little	4 bytes	0	B	C
	Little	X	1	C	C

Table 4−25. MMU 8-Bit Access to Address 2, Little-Endian System Memory

8-Bit Access to Address 2	DSP MMU Page Table Configuration			DSP Access Result	MPU Access Result
	Endianness	**Element Size**	**Mixed Data**		
	Little	1 byte	0	B	B
	Little	2 bytes	0	A	B
	Little	4 bytes	0	C	B
	Little	X	1	B	B

Table 4−26. MMU 8-Bit Access to Address 3, Little-Endian System Memory

8-Bit Access to Address 3	DSP MMU Page Table Configuration			DSP Access Result	MPU Access Result
	Endianness	**Element Size**	**Mixed Data**		
	Little	1 byte	0	A	A
	Little	2 bytes	0	B	A
	Little	4 bytes	0	D	A
	Little	X	1	A	A

Table 4−27. MMU 8-Bit Access to Address 0, Big-Endian System Memory

8-Bit Access to Address 0	DSP MMU Page Table Configuration			DSP Access Result	MPU Access Result
	Endianness	**Element Size**	**Mixed Data**		
	Big	1 byte	0	A	D
	Big	2 bytes	0	A	D
	Big	4 bytes	0	A	D
	Big	X	1	A	D

Table 4−28. MMU 8-Bit Access to Address 1, Big-Endian System Memory

8-Bit Access to Address 1	DSP MMU Page Table Configuration			DSP Access Result	MPU Access Result
	Endianness	**Element Size**	**Mixed Data**		
	Big	1 byte	0	B	C
	Big	2 bytes	0	B	C
	Big	4 bytes	0	B	C
	Big	X	1	B	C

Table 4−29. MMU 8-Bit Access to Address 2, Big-Endian System Memory

8-Bit Access to Address 2	DSP MMU Page Table Configuration			DSP Access Result	MPU Access Result
	Endianness	**Element Size**	**Mixed Data**		
	Big	1 byte	0	C	B
	Big	2 bytes	0	C	B
	Big	4 bytes	0	C	B
	Big	X	1	C	B

Table 4−30. MMU 8-Bit Access to Address 3, Big-Endian System Memory

8-Bit Access to Address 3	DSP MMU Page Table Configuration			DSP Access Result	MPU Access Result
	Endianness	**Element Size**	**Mixed Data**		
	Big	1 byte	0	D	A
	Big	2 bytes	0	D	A
	Big	4 bytes	0	D	A
	Big	X	1	D	A

4.3.13.5 DMA and MMU Endianness Conversion Compatibility

DSP DMA endianness conversion operation is based solely on the DSP DMA channel parameters as programmed in the DMA4_CSDP register in the DSP DMA. DSP MMU endianness conversion is based on the MMU page table parameters for the system memory accessed and on the in-band data qualifiers to the MMU for individual transactions.

When the DSP DMA is used with the DSP MMU to do transfers between DSP internal memory and external memory, the DSP DMA supplies an in-band qualifier to the DSP MMU indicating the endianness of the data transferred.

The MMU compares this qualifier to the endianness setting in the corresponding page table; if the endianness settings do not match, the MMU performs endianness conversion; otherwise, no conversion occurs. However, when the DSP DMA locks the endianness, the MMU does not perform endianness conversion, regardless of the page table setting.

Thus, two use models are supported: one where the system MPU (ARM1136) sets the endianness through MMU page table settings, and one where the DSP CPU overrides those endianness settings when performing a DMA transfer.

4.4 Programming Model

4.4.1 Power Management

DSP subsystem power is controlled by the PRCM module in the OMAP2420. Control of the power up/down behavior of the DSP subsystem is accomplished through registers in the PRCM module. Table 4−31 shows the DSP power domain registers that control the power of the DSP subsystem.

> **Note:**
>
> The registers shown in Table 4−31, Table 4−33, Table 4−34, Table 4−36, and Table 4−37 are not in the DSP subsystem, but are in the PRCM register space.

4.4.1.1 Controlling the Power State of the DSP Power Domain

Table 4−31. DSP SS Power-Management Registers

Register Name	Physical Address	Reset Source[1]	Description
DSP SS Power-Management-Associated Registers			
PM_PWSTCTRL_DSP	0x4800 88E0	W	Controls power state transition of the DSP power domain
PM_PWSTST_DSP	0x4800 88E4	W	Stores status of power state transition of the DSP power domain

1) Reset source: W = Warm reset, C = Cold reset. For more information about the sources of warm and cold resets, see Chapter 5, *Power, Reset, and Clock Management*.

4.4.1.2 DSP Power Domain ACTIVE−to−OFF State Transition

The following sequence describes how to operate a DSP power domain state transition from the ACTIVE state to the OFF state:

Step 1: Program the POWERSTATE field to OFF in the PM_PWSTCTRL_DSP register.

Step 2: Place the DSP subsystem in idle mode (as described in Section 4.4.1.3, *DSP Subsystem ACTIVE-to-IDLE State Transition)*.

Step 3: The transition occurs when the DSP processor is in standby mode. This is achieved after the DSP exevutes its IDLE instruction and asserts its standby signal. At this stage, if there is no pending interrupt or any MPU access and if the AUTOSTATE_DSP bit in the CM_CLKSTCTRL_DSP register is set, the PRCM stops the DSP clock and disables power to the DSP power domain.

Step 4: The MPU can poll the INTRANSITION bit PM_PWSTST_DSP[20] to know when the transition state is complete. It can also poll the STANDBY_DSP and bit in the CM_IDLEST_DSP register to track standby modes.

There is no MPU software control of DSP memory power states. They are independently managed by the DSP software.

If the AUTOSTATE_DSP bit is cleared, the transition cannot occur until software sets the FORCESTATE bit PM_PWSTCTRL_DSP[18].

4.4.1.3 DSP Subsystem ACTIVE-to-IDLE State Transition

The only way for the DSP SS to go to IDLE state is by executing the IDLE instruction.

To put the DSP subsystem in IDLE state, the DSP software must perform the following procedure:

Step 1: The DSP software performs the following steps:

1) Set to active all bits of the DSP core ICR (idle control) register.

a) Save the current mask vector and mask all DSP core INTC maskable interrupts except FIQ and IRQ.

b) Program the dDMA standby mode to force standby or smart standby. If necessary, save the current dDMA channel-enable configuration and deactivate the dDMA channel-enable bit of all the channels. For more information, see Section 4.4.10, *DSP DMA*.

c) Set the power mode for the DSP subsystem power domain in the PRCM dedicated configuration registers. See Table 4−31.

d) (Optional; required only when going to OFF state) Save some or all of the memory-mapped registers with nonretention capability in a memory with retention capability or in a power domain that will not be powered off.

Step 2: Set these configurations using write-nonposted commands to ensure that configuration is complete before issuing the IDLE command.

Step 3: DSP issues the IDLE instruction.

Note:

It may take a long time for the dDMA to assert the Mstandby signal, depending on its configuration. Disabling all channels and configuring the dDMA in ForceStandby mode is a good general rule for a quick and safe DSP subsystem idle transition. For more information about DMA4 power management, see Chapter 5, *Power, Reset, and Clock Management*.

When the DSP DMA is put in standby and the DSP executes the IDLE instruction, the DSP subsystem is in standby mode and the PRCM module disables the clocks to the DSP subsystem (dependent on the PRCM register bit CM_CLKSTCTRL_DSP setting). At this point, the DSP subsystem can be released from standby by three events:

- An interrupt event to the DSP subsystem INTC
- Access to the DSP IPI port
- DMA request event on an enabled channel (PRCM clocks must be enabled for DMA request to cause a DSP subsystem wakeup)

Note:

A wakeup caused by a DSP subsystem interrupt is a permanent wakeup, whereas a wakeup caused by IPI access is temporary (the DSP subsystem returns to idle after all IPI transactions are complete; completion is determined by a time-out period between L3 transactions to the IPI).

When a nonmasked interrupt arrives on the DSP core INTC, the DSP goes into its interrupt service routine:

- Restore the saved dDMA channel-enable configuration.
- Restore from memory any saved memory-mapped registers with nonretention capability.
- Restore the interrupt mask vector.

4.4.1.4 DSP Core Interrupts During PRCM Clock Shutoff

During transitions to low-power states, all DSP core interrupts (other than FIQ and IRQ) must be cleared to ensure that the DSP CPU does not receive an interrupt while the PRCM module is asynchronously shutting off DSP clocks.

4.4.2 DSP Core Idle Control

The DSP core idle configuration register (ICR) is used to set idle modes for DSP core submodules. When the IDLE instruction is executed, the contents of the ICR become requests to the DSP core submodules to go to idle mode. The idle status register is a read-only register that shows the status of each submodule.

To prevent submodules from going to idle mode while idle requests are issued, a priority is followed when multiple DSP core submodules are idled. A bus error is generated if this priority is violated. (The submodule can still be idled, but if the DSP or DMA tries to access the resource, it hangs.) Also, idling of some core submodules depends on other domains being previously idled (and vice versa; for these same domains to be used, some other domains must be operational).

On the execution of the IDLE instruction, hardware idles the DSP core domain submodules in the order of priority shown in Table 4−32 (priority 1 is the first to be idled). However, software must ensure that the interdependent domain idles/disables are followed, as shown in column 3 of Table 4−32 (for example, software must ensure that the I-cache is disabled before it is idled). When modules come out of idle mode, the priority is reversed. The priority is also reversed when a DSP DMA access wakes up the DSP core, except that the priority 3 and 4 domains happen at the same time.

Once started, an idle sequence continues until complete. Once the ICR is configured and the IDLE instruction executed, the DSP core blocks access to the ICR register until all DSP core peripherals are idled.

For I/O accesses to modules that are in idle mode, the X-port generates a bus error if the INTERREN bit of the XCR register is set to 1.

Table 4−32. DSP Core Domain Idle Priority

DSP Core Submodule	Priority[1]	Requirements for Entering Idle Mode
DSP	1	None
DMA	1	None
I-cache	2	I-cache disabled
HWA	2	None
PERIPH	2	None
M-port	3	DMA
I-port	3	DSP and I-cache in idle
X-port	3	DMA and DSP in idle
D-port	3	DMA and DSP in idle

1) This is the priority by which the DSP core hardware idles DSP core internal modules after executing the IDLE instruction. For a submodule to be idled, the corresponding bit in the ICR register for each module must be set.

When a module in the DSP core is idle mode, it cannot receive commands.

If an attempt is made to put the I-cache in idle mode without its first being disabled, the attempt fails and a bus error is generated.

Placing the DPLL submodule in idle mode effectively shuts off source clock input to the DSP core.

If the DSP core is idle and an interrupt occurs to the DSP core INTC, the DSP core I-port, D-port, and X-ports are also forced out of idle mode. However, the ICR register remains unchanged so that the IDLE instruction can be executed after the interrupt is serviced without having to recover the contents of the ICR register.

The DSP RSS is requested to go to idle mode when the M-port and DSP are idle, depending on the setting of the MSENx bit field in the DSP core IMSCR register. The IMSSTR register indicates the status of the DSP subsystem internal memory.

4.4.2.1 M-Port Wakeup Through DMA Access

The DSP core can wake up the M-port domains of the DSP core to allow for data transfer or loading program code into internal memory. When the DSP

DMA makes an access to the M1 port, or the IPI accesses the M0 port, the DSP core brings the DSP RSS out of idle mode.

4.4.2.2 DMA Access to DSP Local Memories During DSP Subsystem Idle

When the DSP subsystem is in idle mode, the DSP DMA can access the DSP local memories. When a DMA request is detected, a wake-up signal is issued to the DSP core by the PRCM module, releasing the DSP internal memories from idle mode. On release from idle mode, the DSP subsystem accepts the command already present and services it to the DSP local memories, then returns to idle mode.

4.4.2.3 IPI Port Access to DSP Local Memories During DSP Subsystem Idle

When the DSP subsystem is in idle mode, the IPI port of the DSP subsystem can access the DSP local memories. When an IPI access is detected, a wake-up signal is issued to the DSP core, releasing the internal memories from idle mode. On release from idle mode, the DSP subsystem accepts the command already present and services it to the DSP local memories; if there are no new accesses, it returns to idle mode.

4.4.3 Wake-Up Management

Table 4−33 shows the wake-up management registers.

Table 4−33. Wake-Up Management Registers

Register Name	Physical Address	Reset Source[1]	Description
DSP Subsystem Wake-Up-Management-Associated Registers			
PM_WKEN_DSP	0x4800 88A0	W	Controls the wake-up behavior of the DSP IPI interface
PM_WKDEP_DSP	0x4800 88C8	W	Controls the dependency of the DSP power domain wakeup on other power domain wakeups

1) Reset source: W = Warm reset, C = Cold reset. For more information about the sources of warm and cold resets, see Chapter 5, *Power, Reset, and Clock Management*.

The PM_WKEN_DSP.EN_DSP_IPI bit controls the IPI port wake−up enable (0 = DSP IPI wakeup is disabled, 1 = DSP IPI wakeup is enabled).

The PRCM register PM_WKDEP_DSP controls the wake-up relationship of the DSP power domain to the MPU, CORE, and WKUP power domains. For more information, see Chapter 5, *Power, Reset, and Clock Management*

4.4.4 Reset Management

The DSP subsystem receives three reset signals from the PRCM module in the OMAP2420. Table 4−34 shows the DSP power domain registers that control the reset signals to the DSP subsystem. (These registers are not in the DSP subsystem, but in the PRCM register space.)

Table 4−34. DSP Subsystem Reset Management Registers

Register Name	Physical Address	Reset Source[1]	Description
Reset-Management-Associated Registers			

RM_RSTCTRL_DSP	0x4800 8850	W	Controls DSP power domain software reset behavior.
			RM_RSTCTRL_DSP.RST1_DSP = DSP_RST1_N signal.
			RM_RSTCTRL_DSP.RST2_DSP = DSP_RST2_N signal.

RM_RSTST_DSP	0x4800 8858	C	Stores the source of the most recent reset to the DSP power domain. (More than one cause can be reported for a reset.)

1) Reset source: W = Warm reset, C = Cold reset. For more information about the sources of warm and cold resets, see Chapter 5, *Power, Reset, and Clock Management*.

4.4.4.1 Application Reset Rules

The rules to assert the input reset signals are as follows:

Step 1: If CORE_RST_N is asserted, DSP_RST1_N must also be asserted.

Step 2: If DSP_RST2_N is asserted, DSP_RST1_N must also be asserted (and also CORE_RST_N, because of Step 1).

Step 3: The rules to deassert the various input reset signals are as follows:

- DSP_RST2_N must be deasserted before DSP_RST1_N is deasserted.
- CORE_RST_N must be deasserted before DSP_RST1_N is deasserted, to give access to a configuration port. There is no issue in deasserting CORE_RST_N when DSP_RST1_N is not also deasserted, because interrupts are masked after reset.

Table 4−35 is a summary of authorized reset configuration.

Table 4−35. Reset Assertion and Release Dependencies

	Assertion Dependencies	Deassertion Dependencies
DSP_RST1_N		CORE_RST_N and DSP_RST2_N deasserted
DSP_RST2_N	Assertion of DSP_RST1_N Assertion of CORE_RST_N	DSP_RST2_N must be deasserted before DSP_RST1_N.
CORE_RST_N	Assertion of DSP_RST1_N	

4.4.4.2 Accessing DSP Local Memories During Reset

An L3 initiator can access the DSP local memories when the DSP is under reset (DSP_RST1_N active). The DSP internal memories are automatically released from standby mode by the DSP core on an IPI port access to the DSP subsystem. (See Section 4.4.1.3, *DSP Subsystem Active to Idle Mode Transition*.)

The DSP_RST2_ signal must be deasserted before the DSP local memories are accessed through IPI.

4.4.4.3 Timing Constraints on DSP Subsystem Resets

The DSP_RST1_N must be released at least 10 DSP_ICLK cycles (where DSP_ICLK is a divided clock from DSP_FCLK) after the DSP_RST2_N is released. This must be guaranteed by the software.

The DSP_RST1_N must remain active for at least 4 DSP_ICLK cycles.

Accesses to the intrusive port of the DSP subsystem must not occur within 20 DSP_ICLK cycles before and after the release of the DSP_RST1_N signal.

Some modules can be reset independently by software by programming the soft reset bit *(*register bit SYSCONFIG[1] is the soft reset bit for modules that support the open−core protocol [OCP)] SYSCONFIG register) in the OCP configuration register. This bit has the same effect on the module logic as the hardware signal.

4.4.5 Clock Management

The DSP subsystem receives clocks from the PRCM module in the OMAP2420. PRCM registers are used to control the power-up/reset behavior of the DSP subsystem. Table 4−36 shows the DSP power domain registers that control the clocks to the DSP subsystem. (These registers are not in the DSP subsystem, but are in the PRCM register space.)

Table 4−36. DSP Subsystem Clock-Management Registers

Register Name	Physical Address	Reset Source[1]	Description
DSP Subsystem Clock-Management-Associated Registers			
CM_FCLKEN_DSP	0x4800 8800	W	Controls functional clock activity to the DSP power domain
CM_ICLKEN_DSP	0x4800 8810	W	Controls interface clock activity to the DSP power domain
CM_IDLEST_DSP	0x4800 8820	W	Allows checking status of the DSP IPI idle mode and DSP subsystem standby mode
CM_AUTOIDLE_DSP	0x4800 8830	W	Allows enabling/disabling auto management of the DSP IPI clock with the DSP domain power state transition
CM_CLKSEL_DSP	0x4800 8840	W	Controls selection of the DSP subsystem and DSP interface clock frequencies
CM_CLKSTCTRL_DSP	0x4800 8848	W	Controls DSP subsystem clock enabling during ACTIVE to INACTIVE power state transition

1) Reset source: W = Warm reset, C = Cold reset. For more information about the sources of warm and cold resets, see Chapter 5, *Power, Reset, and Clock Management*.

4.4.5.1 Controlling IPI Idle Mode

The DSP subsystem allows the IPI interface to enter a low-power mode during times of inactivity.

PRCM registers CM_ICLKEN_DSP[1] and CM_AUTOIDLE_DSP[1] work together on the idle conditions for the intrusive port interface of the DSP subsystem, as shown in Table 4−37.

Table 4−37. IPI Idle Mode Control

CM_ICLKEN_DSP[1] (physical address = 0x4800 8810)	CM_AUTOIDLE_DSP[1] (physical address = 0x4800 8830)	Description
0	X	Intrusive port is put in idle mode unconditionally.
1	0	Intrusive port is not put in idle mode (even when DSP core and DSP DMA are in idle mode).
1	1	Intrusive port is put in idle mode when DSP core and DSP DMA are put in idle mode.

The activity status of the IPI port can be checked by accessing the CM_IDLEST_DSP[1] field (physical address = 0x4800 8820) in the PRCM module.

4.4.5.2 DSP DMA Clock Gate Control

Writing 0 to the DSPSS_GCR.dDMAClkEnable disables the input clock signal of the DSP DMA to optimize power when the DSP DMA is not in use.

When disabling the DMA module, DSP software must check that there is no nonsynchronized enabled channel and that synchronized channels are not being serviced.

If there are synchronized channels, software must clear this bit for the DMA request event triggering the enabled DMA channels to be serviced. The DMA requests hangs waiting for service until this bit is cleared.

It is recommended to use this bit only when the DSP DMA has long periods of inactivity.

4.4.6 DSP MMU Programming Model

4.4.6.1 Configuring the DSP MMU

The DSP MMU must be configured while the DSP subsystem has no pending transactions. A good programming rule is to configure the DSP MMU (including TTB initialization) while the DSP and the dDMA are under reset.

For peripherals external to the DSP subsystem, the IPI register bits IPI_ IOMAP[5:0] (I/O map base address field) must be programmed in the same sequence and coherently with the DSP MMU.

Note:

It is possible to program the DSP MMU so that DSP local memory can be accessed by the DSP through the DSP MMU/L3/IPI, assuming that the firewall on the L3 connection to the intrusive port of the DSP subsystem is configured to enable such an access and that the MMU is first configured by the MPU (ARM1136) to ensure that there is an address translation for addresses of the MMU configuration registers. For details about how to configure the L3 firewalls, see Chapter 6, *Internal Interconnect.*

4.4.6.2 Enabling DSP MMU and Initializing the TTB

After reset, the TLB is empty and the MMU is disabled.

The MMU_LOCK register fields must be initialized by the MPU before enabling the DSP MMU. TLB entries can then be initialized as needed, starting from entry zero and incrementing the CurrentVictim field value in the MMU_LOCK register as each entry is written. The DSP MMU register bits MMU_LOCK[15:10] (base value field) can be set to lock the initialized TLB entries if the table walker is to be enabled.

If required, the TTB is then programmed and the table walker enabled.

The MMU can then be enabled (a single write to the CNTL_REG register), and the DSP can be released from reset.

If some TLB entries are not initialized (and the CurrentVictim counter has not reached the maximum MMU entry count (32 entries), the table walker fills the

TLB sequentially on misses until the CurrentVictim value reaches the maximum entry limit. The MMU random replacement algorithm is then activated for subsequent misses.

The replacement algorithm ensures approximately equal probability of replacing any TLB entry that is not locked. The entries used for successive replacements should be sufficiently uncorrelated to avoid repeated misses caused by coincidence with software repeat/loop lengths.

4.4.6.3 Lock Mechanism and the Current_Victim Counter

Any of the 32 TLB entries in the DSP MMU can be locked; however, a maximum of only 31 entries can be locked. The lock mechanism prevents an entry of the TLB from being replaced by another entry when a TLB miss occurs. If the BASEVALUE (BaseValue) field of the MMU_LOCK register is > 0, TLB entries from BaseValue – 1 to 0 are locked.

The CurrentVictim counter of the MMU_LOCK register specifies the location of the entry, which is loaded or replaced.

A TLB miss fault does not occur if the hardware table is enabled.

4.4.6.4 TLB Fault Handling

Two types of faults can occur:

- ❑ TLB miss with table walker disabled: No translation is found for the virtual address required. If the hardware table walker is disabled, a fault is generated.
- ❑ Translation fault: No translation is found for the virtual address required (TLB miss), and the hardware table walker is enabled, but no page table entry exists for the requested address.

When a fault occurs, an interrupt is generated. An interrupt status register (ISR) is then responsible for fault recovery. The DSP is stalled by the MMU while the fault is handled. For example, for a TLB miss, the ISR can load the missing entry from the page table.

The ISR can determine the cause of the abort interrupt by reading the MMU_IRQSTATUS register. The virtual address that caused the fault can be determined by reading the MMU_FAULT_AD register.

In the case of a TLB miss, the MMU continues to service the DSP request immediately after a valid TLB entry is written. In the case of a translation fault, the ISR releases the MMU by writing to the ISR. The MMU continues servicing the DSP request. The ISR can also terminate DSP operation by resetting the DSP and the MMU.

4.4.6.5 Initializing Locked TLB Entries

Follow this procedure to load a TLB manually.

Step 1: Load the virtual address, the preserved bit, the valid bit, and the section/page size in the MMU_CAM register.

Step 2: Load the physical address and the endianness/element_size bits in the MMU_RAM register.

Step 3: Update the CURRENTVICTIM field in the MMU_LOCK register (specifying the location of the entry to be loaded).

Step 4: Set the LDTLBITEM bit of the MMU_LD_TLB register to load these values.

The valid bit determines which MMU_CAM entries are used in address comparison.

4.4.6.6 Flushing TLB Entries

The entire TLB can be flushed at once by setting the GLOBALFLUSH bit in the MMU_GFLUSH register. TLB entries with a preserved bit set to 1 (bit *P* in the MMU_CAM register) are not flushed. The preserved bit should be used only on locked TLB entries, because it does not prevent replacement by the table-walking logic.

Regardless of the preserved bit setting, a specific TLB entry can be flushed by setting the FLUSHENTRY bit in the MMU_FLUSH_ENTRY register. The entry to be flushed is specified by the virtual address in the MMU_CAM register.

4.4.6.7 Table-Walking Logic

The 32 entries of the TLB may not be sufficient to store all the necessary translations for all the memory space to be accessed. In this situation, a DSP access generates a TLB miss when a virtual address with no matching translation is presented to the MMU. If the hardware table walker is disabled, this miss generates an interrupt to the MPU while the DSP is held in a stalled state. The MPU system software can update the TLB with the required translation and release the DSP. However, a more efficient option is to enable the table walker.

When the hardware table walker is enabled, it automatically fills the TLB when misses occur. When a memory access is made and there is no TLB entry for the virtual address, the table walker loads the missing entry from the page table stored in system memory.

The OMAP2420 system memory area must be initialized with the page descriptors (which are implementation-dependent and in line with the hardware table-walking logic design). The translation table base address, in the MMU_TTB register, must also be initialized with the (physical) address of this memory area before enabling the MMU.

When a TLB miss occurs, a level 1 descriptor is read based on the virtual address and the value of the TTB register. The read value gives information to the MMU about the page (page size, endianness, data type, and upper bits of the physical address). If required, a second-level descriptor is read with respect to the translation table hierarchy. In this case, the level 1 descriptor address field and the middle part of the virtual address give the address where this level 2 descriptor is read. When the value of this second-level descriptor is read, the MMU finally builds the TLB entry. The table-walking logic writes into one of the TLB lines (CAM or RAM) between BaseValue and 31.

For more information about the table translation hierarchy, see Chapter 9, *Memory Management Units*.

4.4.7 Intrusive Port Interface Programming Model

4.4.7.1 IPI Configuration

The IPI must be programmed when no access to the DARAM or SARAM is pending in the system. The software must ensure that no enabled program transfers from system DMAs or accesses from other CPUs to the DSP local memories occur during IPI programming. Programming the IPI during accesses to the DSP local memories through the DSP subsystem intrusive port has undefined effects.

After reset, the IPI propagates the requests on its slave port to its master port, without performing any conversion.

4.4.7.2 Enable/Disable IPI Endianness Conversion

If the IPI_ENABLE bit is clear, the ELEMENT_SIZE information is not taken from the look-up table. Instead, the data type used is constantly 32-bit. This means that an access on the slave port while the IPI_ENABLE bit is clear is not endianness-converted on the request path or on the response path.

Data are not altered by concurrent asynchronous enable/disable of the endianness conversion feature.

4.4.7.3 IPI Reset

At reset, the IPI ELEMENT_SIZE information is undefined (however, a read to the IPI_INDEX_REGISTER returns a constant 32-bit default value). The IPI_ENABLE bit field is cleared. For more information about slave port accesses while IPI_ENABLE is clear, see Section 4.1, *DSP Subsystem Overview.*

4.4.7.4 IPI Look−Up Table Initialization

It is highly recommended that the software update all the entries of the IPI table before issuing a request on the IPI data slave port. During look-up table initialization, the IPI_ENABLE bit must be clear (IPI disabled). Initializing the look-up table when IPI_ENABLE is set has undefined effects.

DSP IPI register IPI_INDEX. (PAGEINDEXVALUE field) is the index in the look-up table that points to memory attributes as defined in the IPI_ENTRY register, for the memory regions (pages) in the range:

Index[0] = 0x00000−0x00FFF

Index[1] = 0x01000−0x01FFF

....

Index[39] = 0x27000−0x27FFF

4.4.8 DSP Boot

Before the DSP can boot, the OMAP MPU must perform the following initialization sequence:

Step 1: The MPU sets the CM_FCLKEN_DSP.EN_DSP bit in the PRCM module.

Step 2: The MPU clears the RSTCTRL_DSP.RST2_DSP bit in the PRCM module to release the DSP MMU and IPI from reset. DSP is maintained under reset, because the PRCM.RSTCTRL_DSP.RST2_DSP bit value is kept at 1.

Step 3: The MPU configures the IPI look-up table with the based, element-size memory attributes. When the IPI look-up table is configured, the MPU sets the IPI_ENABLE.Enable bit.

Step 4: The MPU uploads code in DSP local memories, if required.

Step 5: The MPU programs the IPI_ DSPBOOTCFG.DSPBootMod bit field to select the boot mode for the DSP.

Step 6: The MPU programs the IPI_ IOMAP.IOMapBaseAddr bitfield, to set the base address of the I/O page in virtual DSP memory space. This must be done coherently with DSP MMU programming so that consecutive address conversions lead to the physical address of the correct I/O memory location.

Step 7: The MPU programs the DSP MMU translation tables in memory and updates the base address of those tables in the MMU_TTB.TTBAddress bit. It can then set the MMU_CNTL.MMUEnable and MMU_CNTL.TWLEnable bits.

Step 8: The MPU can protect the programming of the DSP MMU and IPI from erroneous accesses by configuring the firewall associated with the slave port of the DSP subsystem, in the L3 interconnect.

Step 9: The MPU must wait 20 cycles of the DSP subsystem clock (divided clock).

Step 10: The MPU releases the DSP from reset by writing 0 in the RSTCTRL_DSP.RST1_DSP bit.

Step 11: The DSP boots according to the value of the IPI_ DSPBOOTCFG.DSPBBootMod bit.

This sequence must be applied each time the DSP subsystem is powered off (after a cold reset of the device and after a DSP subsystem wakeup from OFF state).

When the IPI_DSPBOOTCFG.DSPBootMod bit equals 0, the boot address 0xFFFF00 is in external memory. The DSP MMU translation table must include an address translation for this virtual address, to the physical address that points to the boot code.

When the IPI_DSPBOOTCFG.DSPBootMod does not equal 0, the boot address 0xFFFF00 is in a ROM internal to the DSP subsystem. It is recommended that the boot-loader be placed at this address.

4.4.8.1 Selecting the Boot Mode of the DSP

The boot mode of the DSP is set by the DSP IPI register bits IPI_DSPBOOTCFG[3:0]. The value stored by the MPU at IPI_DSPBOOTCFG[2:0] sets the value of the DSP core register BOOTMOD[3:0] when the DSP core is released from reset.

The DSP core always starts at address 0xFFFF00h; however, the setting of DSP core register bits BOOTMOD[3:0] determines how this address gets mapped. If BOOTMOD[3:0] is set to 0000b at reset, the DSP core hardware maps 0xFFFF00h to external memory and ignores the bootloader in internal PDROM. Any other setting of DSPBOOTMOD[3:0] automatically maps 0xFFFF00h to the 64-KB internal PDROM and launches the bootloader stored there. The bootloader subsequently reads the BOOTMOD[3:0] value to determine which of several DSP CPU boot sequences to run, according to Table 4−38. Boot sequences and mapping on IPI_DSPBOOTCFG.DSPBootMod values are defined by ROM software.

Table 4−38. Boot Configuration Summary (per PDROM Software)

BOOTMOD[3:0]	Boot Process	Start (8-Bit Word) Address of CPU after Bootloader Program is Executed	Notes
0000	No boot	FFFF00h (reset vector from memory external to UMA v2.3)	Bootloader stored externally
0001	No boot download (pseudo-direct boot)	080000h (external)	Bootloader maps DSP CPU start execution to external memory, address 0x080000h[1]
0010	Idle state	Not applicable	In this mode, the code executed from internal PDROM sets the domains of the DSP core to idle mode. The DSP can be waked up from its idle mode by a new reset.
0011	16-bit external memory	User-defined (on-chip RAM)	Bootloader reads boot table from external memory at address 0x080000h, copies the program code to internal RAM, and on completion branches to the destination address of the first section copied. (External memory data bus width = 16 bits)[1]
0100	32-bit external memory	User-defined (on-chip RAM)	Bootloader reads boot table from external memory at address 0x080000h, copies the program code to internal RAM, and on completion branches to the destination address of the first section copied. (External memory data bus width = 32 bits)[1]

Table 4−38. Boot Configuration Summary (per PDROM Software) (Continued)

BOOTMOD[3:0]	Boot Process	Start (8-Bit Word) Address of CPU After Bootloader Program is Executed	Notes
0101	API boot 010000h	On-chip SARAM	MPU processor loads code and data directly into DSP memory while the DSP is in reset. The DSP executes the loaded code when released from reset. API boot does not use a boot table. The bootloader code runs from internal PDROM after the MPU loads the desired code and data and releases the DSP from reset and branches to word address 0x08000h at the beginning of the internal SARAM block.
Other values	No boot download	024000h (on-chip SARAM)	

1) The MPU must ensure that the DSP MMU is configured correctly so that when the DSP jumps to address 0x080000h, it is pointed to valid memory space.

4.4.9 Defining Base Address of I/O-Space

The base address of the I/O-space in memory space can be changed by the MPU software by writing in the IPI_IOMAP.IOMapBaseAddr field the 6-bit index of the I/O page in the 16M-byte memory address space (pages are 256KB large, yet only 128K bytes are used because the 2K-byte peripheral configuration address space is mapped on 4K-byte pages). (For information about address biasing, see Section 4.3.1, *DSP Memory and I/O Space Overview*.)

It is highly recommended to configure the IPI_IOMAP.IOMapBaseAddr field in the DSP MMU translation table setup sequence. Both DSP MMU and IPI modules can be released from reset independently of the DSP so that it is possible to configure both of them while the DSP is under reset. Moreover, the IOMap-Address change is effective only on release of the reset of the DSP.

Before changing the IPI_IOMAP.IOMapBaseAddr setting, MPU software must ensure that the DSP is not currently performing external accesses. It is highly recommended that this value be changed while both the DSP core and the DSP DMA are under reset.

The DSPSS_IOMAP.IOMapBaseAddr bitfield is updated with the IPI_IOMAP.IOMapBaseAddr bitfield the cycle after the release of the DSP from reset. The DSPSS_IOMAP.IOMapBaseAddr bit is the value used to decode addresses in the DSP subsystem interconnect. The DSPSS_IOMAP register is read-only and is private to the DSP.

Software must never write 0x00 to the IPI_IOMAP.IOMapBaseAddr bitfield, because base address 0x00 is reserved for accesses to local memories. Writing 0x00 to the IPI_IOMAP.IOMapBaseAddr has an undefined effect. All other values are possible, assuming that they are programmed in conjunction with and coherently with the memory-management unit configuration of the DSP subsystem.

The address resulting from the address biasing for I/O accesses is still in the DSP virtual address space and must be converted by the DSP MMU to a physical address in OMAP system space. Thus, the IPI_IOMAP.IOMap BaseAddr field must be configured coherently with the programming of the DSP MMU translation tables.

Note:

The IPI_IOMAP.IOMapBaseAddr value must be set statically by the MPU and known statically (at compile time) by DSP applications.

4.4.10 DSP DMA

The DSP DMA-generated address must be in the 16-MB range of the DSP memory space; that is, bits 31 to 23 of the generated address are always cleared. Programming the DSP DMA with transfers that generate addresses up to the 16M-byte range of the DSP space has undefined effects. The DSP software ensures that accesses are bounded in the DSP space 16M-byte range. No error or interrupt is generated in cases where an access within a DMA transfer goes beyond this bound.

The DSP DMA must be programmed as big-endian when accessing DSP local memories. Little-endian access to DSP local memories resulting from a DSP DMA channel transfer has undefined effects.

For more information about the DMA programming model, see Chapter 9, *DMA*.

4.4.11 DSP Core Instruction Cache/Prefetch Buffer Programming Model

4.4.11.1 Enabling/Disabling the Prefetch Engine

The instruction prefetch engine in the DSP core can be enabled/disabled by writing to the DSP core register IGCR (PrefetchEnable) bit.

Writing 1 to the IGCR[13] bit enables the prefetch functionality. If the I-cache is enabled, I-cache misses the result in advance fetch of the next line in memory. If the I-cache is disabled, the prefetch engine is disabled, as well.

Writing 0 to the IGCR[13] bit disables the prefetch functionality independently of the configuration of the I-cache.

Table 4−39 summarizes the possible configurations of the instruction caching mechanisms in the DSP subsystem.

Table 4−39. Prefetch Engine and I-Cache Configurations

DSP Core MMR Register ST3_55.*CAEN*	**I-Cache.PrefetchEnable**	**I-Cache**	**Prefetch Engine**
0	X	Disabled	Disabled
1	0	Enabled	Disabled
1	1	Enabled	Enabled

The instruction prefetch engine has packing capability when the prefetch engine is disabled.

At reset, I-cache.PrefetchEnable is 0 so that the prefetch engine is disabled.

4.4.11.2 Protecting Against Advance Fetch

While the instruction prefetch engine is hardware-protected against doing advance fetch across 4KB-page boundaries, the I-cache and the C55x DSP instruction buffer queue (IBQ) are not.

An instruction fetch of the DSP can result in an advance fetch of a maximum of 64 bytes in external memory. The worst case is when the instruction is a 16-bit GOTO instruction at the very last 2 bytes of a 4-KB page. In this case, the 64 bytes after this instruction are not executed by the DSP but are fetched in advance jointly by the IBQ and the I-cache. This implies an additional 16-byte fetch if the prefetch engine has 1 line, and an additional 16-byte fetch if the prefetch engine has 2 lines.

Instruction fetches can have undesired behavior in the following cases:

- ❑ Access to an I/O peripheral (for which read might have destructive side effects)
- ❑ Access to a secured memory region protected against reads from the DSP

The DSP code must be padded with NOPs on the 64 bytes after the last instruction if DSP code is in internal memory, and with 96 bytes after the last instruction if the code is in external memory.

Note:

The first 96 bytes of the I/O space consist of the following DSP core registers: ICR, ISR, IMSCR, IMSSTR, APISIZE, and BOOTMOD. Reads to these registers have no side effects; therefore, for programs with the last instruction close to the I/O space, there is no requirement to protect against advance fetch.

4.4.12 Accessing Peripherals

DSP subsystem access to certain system peripherals requires the constraint of regular memory semantics by disabling or managing specific performance optimizations in the DSP core and its memory hierarchy. These performance optimizations and their associated issues relative to DSP peripheral accesses are described in the following paragraphs along with suggested peripheral access methods.

4.4.12.1 Peripheral Access Optimization

Write Posting

A write to a DSP peripheral configuration register can cause the DSP to wait for completion before continuing execution. A good example is the write to clear an interrupt before the corresponding interrupt service routine acknowledges the reception of the interrupt to the DSP; this write is completed in the external interrupt source before the ISR acknowledge is executed (for the C55x DSP, clear of the local mask bit). For regular memory accesses, the DSP issues the write to the memory hierarchy without waiting for completion at the end target. This write is said to be buffered or posted.

The DSP subsystem, memory hierarchy, and end peripherals provide software controllability of this optimization so that the execution of the instruction after a write to such a peripheral is stalled until the write is complete. When write-posting optimization is globally disabled, the write is referred to (in the scope of this document) as a nonposted write. For more information, see Section 4.3.3.2, *Data Master Port*.

Data Internal Bypass

A read access to a peripheral configuration register can also have side effects. An example in the OMAP2420 is a read to one of the message registers of the mailbox module, which, when complete, can cause the mailbox to generate an interrupt to the message source that the mailbox is empty (or not full).

Generally, the DSP is stalled until the read data is back from the end peripheral; thus, there is no uncertainty about read access completion. However, in certain circumstances, such as a write followed by a read at the same address, in the instruction flow, the data read comes directly from the pipe stage (data internal bypass optimization), not from the end peripheral. Thus, the completion of the associated read access side effect, and possibly even the completion of the read operation itself, is not ensured when the DSP resumes execution.

The DSP subsystem, memory hierarchy, and end peripherals provide software controllability of this optimization, so that the execution of the instruction after a read to such a peripheral is stalled until the read is complete. Use of either of the two DSP peripheral access methods described in the subsequent subsections ensures that the read is complete before the C55x continues to the next instruction.

Write/Read Reordering

Because of potential system-level side effects of both reads and writes, correct ordering between reads and writes is required when accessing DSP peripherals, even to different addresses. Because of the DSP internal pipeline architecture, accesses to regular memory by the DSP can issue a read before a write, even if the write occurs first in the program execution (with the condition that there is no address collision). This is acceptable for regular memory, as no dependency is expected between the two operations. However, this is not acceptable for peripheral registers to which writes and/or reads can have system-level effects. Use of either of the two DSP peripheral access methods described in Section 4.4.12.2 ensures that read and write accesses are locked to the order of program execution.

4.4.12.2 DSP Peripheral Access Methods

The DSP subsystem offers two methods of generating guaranteed access to DSP peripherals:

- ☐ Dedicated peripheral address space (I/O space)
- ☐ Software macros to provide the same type of protected access to the unified address space (MEM space)

I/O Space Addressing

The C55x DSP core natively provides special addressing modes to protect accesses to DSP peripherals. When using these addressing modes, accesses are pipeline-protected and writes are not posted. Also, data internal bypass from posting buffers is not used, because writes do not go through the posting buffers.

I/O Software Macros

The READ_IO and WRITE_IO software macros, defined in subsection *READ_IO Macro* and subsection *WRITE_IO Macro,* respectively, allow access to DSP peripherals mapped to the unified DSP memory space. These macros allow protected access to peripherals mapped onto MEM space exactly as if they were mapped onto the I/O space, except that the protection is by software rather than in hardware. These macros flush the C55x DSP pipeline before and after execution of regular access (to ensure pipeline protection), ensure that interrupts are globally disabled during that process to make it atomic, and ensure that write posting is disabled before executing the regular write.

Note:

It is not required to use these I/O macros if the only purpose is to disable write posting. You can manually disable/enable write posting by clearing/setting the register bit DCR[7] (WPE bit) in the DSP core.

See Section 4.3.1, *DSP Memory and I/O Space Overview*, for an overview of DSP subsystem unified memory (program, data, and I/O memory spaces). Also see Section 4.4.9, *Defining Base Address of I/O Space*, for instructions for how to set the base address of the DSP I/O space mapped to DSP MEM space.

Note:

The DSP subsystem does not ensure strongly ordered accesses. All accesses made using I/O-space addressing are processed in the order dictated by the program execution flow.

Because reads through the DSP core D-port can be executed before a preceding write access (because of the architecture of the DSP instruction pipeline), the pipeline must be flushed before I/O space reads through the D-port.

Also, because posted writes through the DSP core D-port go through a store buffer, use a macro to ensure correct ordering and completion of the write.

READ_IO Macro

```
The following macro allows you to read IOs through the
D-port:

   ;  The macro modifies T0

   ;

READ_IO   .macro   Reg, Mem

   ; flush the pipeline to prevent the read to be executed
before preceding writes

   *port(#0x000E) = T0

   T0 = *port(#0x000E)

   ; Read from DPORT

   Reg = Mem

   .endm
```

WRITE_IO Macro

```
   ;  The following macro allows you to write to IOs
through the D-port

   ;  The macro modifies TC1 and TC2

   ;

WRITE_IO  .macro      Mem, Reg

   pshboth (XAR0)

   ; set INTM to 1 (mask interrupts) and get old value

   TC1 = bit(@ST1_L, #ST1_INTM), bit(@ST1_L, #ST1_INTM) =
1 || mmap()

   ; flush pipeline to make sure that the write is fin-
ished and that INTM is set

   *port(#0x000E) = AR0

   AR0 = *port(#0x000E)

   ; write posting is controlled by bit #7 of GCR register

AR0 = #7


; disable write posting and get old value of WPE bit

   .if$isdefed(”GCR_800”)

      ; 5510 design, GCR is mapped to @0x800

      TC2 = bit(*port(#0x800), AR0)

      bit(*port(#0x800), AR0) = #0

   .elseif $isdefed(”GCR_200”)
```

```
      ; 5502 design, GCR is mapped to @0x200
      TC2 = bit(*port(#0x200), AR0)
      bit(*port(#0x200), AR0) = #0
   .endif
   ; TWO cycles are needed for the write to WPE to be effective
   NOP
   NOP
   ; this pop followed by the next push is here
   ; to prevent the case where the memory access uses AR0
   XAR0 = popboth()
   ; write to DPORT
   Mem = Reg
   pshboth (XAR0)
   ; write posting is controlled by bit #7 of GCR register
   AR0 = #7
   ; restore write posting flag
   .if$isdefed(”GCR_800”)
      ; 5510 design, GCR is mapped to @0x800
      if (TC2)  execute(D_UNIT)
      bit(*port(#0x800), AR0) = #1
   .elseif $isdefed(”GCR_200”)
      ; 5502 design, GCR is mapped to @0x200
      if (TC2)  execute(D_UNIT)
      bit(*port(#0x200), AR0) = #1
   .endif
   ; restore INTM flag
if (!TC1)  execute(D_UNIT)
      bit(ST1, #ST1_INTM) = #0
   ; flush the pipeline to make sure that the write is finished
   *port(#0x000E) = AR0
   AR0 = *port(#0x000E)
   XAR0 = popboth()
   .endm
```

4.5 DSP Subsystem Registers

4.5.1 DSP Subsystem Register Map

Table 4−40. DSP Subsystem Register Map

Module Name	Base Address	Size	Comments
DSP core	0x0000[1]	18K bytes	DSP I/O space
dINTC	0xFC 9000	1K bytes	DSP memory space
DSP SS Global	0xFC B000[2, 3]	2K bytes	DSP memory space
dDMA	0xFC C000[2, 3]	4K bytes	DSP memory space
DSP MMU	0x5A00 0000[4]	64K bytes	OMAP2420 memory space (physical)
IPI	0x5900 0000	4K bytes	OMAP2420 memory space (physical)

1) DSP core configuration registers are accessible only from DSP I/O space memory

2) The DSP global configuration registers and INTC and DMA modules are private peripherals that can be accessed only by the DSP. (They are not visible to the L3/MPU.)

3) The base address for dINTC, dDMA, and DSP global configuration registers in this table is based on an IOMA address offset set to the default of (0x3F*256KB) = 0xFC 0000. For more information about setting IOMA, see Section 4.4.4, *Reset Management*.

4) The DSP MMU is a DSP public peripheral that can be accessed by both the DSP and the MPU.

4.5.2 Module Register Mapping Summary

Table 4−41. DSP INTC Register Summary (Private)

Register Name	Type	Register Width (Bits)	Virtual Address (DSP Memory Space, Byte Address)	Virtual Address (DSP I/O Space, Byte Address)
INTC_REVISION	R	32	0xFC 9000	0x4800
INTC_SYSCONFIG	RW	32	0xFC 9010	0x4810
INTC_SYSSTATUS	R	32	0xFC 9014	0x4814
INTC_SIR_IRQ	R	32	0xFC 9040	0x4840
INTC_SIR_FIQ	R	32	0xFC 9044	0x4844
INTC_CONTROL	RW	32	0xFC 9048	0x4848
INTC_PROTECTION	RW	32	0xFC 904C	0x484C
INTC_IDLE	RW	32	0xFC 9050	0x4850
INTC_ITR0	R	32	0xFC 9080	0x4880
INTC_MIR0	RW	32	0xFC 9084	0x4884
INTC_MIR_CLEAR0	RW	32	0xFC 9088	0x488C
INTC_MIR_SET0	RW	32	0xFC 908C	0x488C
INTC_ISR_SET0	RW	32	0xFC 9090	0x4890
INTC_ISR_CLEAR0	RW	32	0xFC 9094	0x4894

Table 4−41. DSP INTC Register Summary (Private) (Continued)

Register Name	Type	Register Width (Bits)	Virtual Address (DSP Memory Space, Byte Address)	Virtual Address (DSP I/O Space, Byte Address)
INTC_PENDING_IRQ0	R	32	0xFC 9098	0x4898
INTC_PENDING_FIQ0	R	32	0xFC 909C	0x489C
INTC_ILR0 – INTC_ILR31	RW	32	0xFC 9100– 0xFC 917C	0x4880 – 0x48BE

Table 4−42. DSP DMA Register Summary (Private)

Register Name	Type	Register Width (Bits)	Virtual Address (DSP Memory Space, Byte Address)	Virtual Address (DSP I/O Space, Byte Address)
DMA4_REVISION	R	32	0xFC C000	0x6000
DMA4_IRQSTATUS_L0	RW	32	0xFC C008	0x6008
DMA4_IRQSTATUS_L1	RW	32	0xFC C00C	0x600C
DMA4_IRQSTATUS_L2	RW	32	0xFC C010	0x6010
DMA4_IRQSTATUS_L3	RW	32	0xFC C014	0x6014
DMA4_IRQENABLE_L0	RW	32	0xFC C018	0x6018
DMA4_IRQENABLE_L1	RW	32	0xFC C01C	0x601C
DMA4_IRQENABLE_L2	RW	32	0xFC C020	0x6020
DMA4_IRQENABLE_L3	RW	32	0xFC C024	0x6024
DMA4_SYSSTATUS	R	32	0xFC C028	0x6028
DMA4_OCP_SYSCONFIG	RW	32	0xFC C02C	0x602C
DMA4_CAPS_0	R	32	0xFC C064	0x6064
DMA4_CAPS_2	R	32	0xFC C06C	0x606C
DMA4_CAPS_3	R	32	0xFC C070	0x6070
DMA4_CAPS_4	R	32	0xFC C074	0x6074
DMA4_GCR	RW	32	0xFC C078	0x6078
DMA4_CCR__0_23	RW	32	0xFC C080– 0xFC C920	0x6080 – 0x6920
DMA4_CLNK_CTRL__0_23	RW	32	0xFC C084– 0xFC C924	0x6084 – 0x6924
DMA4_CICR__0_23	RW	32	0xFC C088– 0xFC C928	0x60488– 0x6928
DMA4_CSR__0_23	RW	32	0xFC C08C– 0xFC C92C	0x608C– 0x692C
DMA4_CSDP__0_23	RW	32	0xFC C090– 0xFC C930	0x6090 – 0x6930

Table 4−42. DSP DMA Register Summary (Private) (Continued)

Register Name	Type	Register Width (Bits)	Virtual Address (DSP Memory Space, Byte Address)	Virtual Address (DSP I/O Space, Byte Address)
DMA4_CEN__0_23	RW	32	0xFC C094– 0xFC C934	0x6094 – 0x6934
DMA4_CFN__0_23	RW	32	0xFC C098– 0xFC C938	0x6098 – 0x6938
DMA4_CSSA__0_23	RW	32	0xFC C09C– 0xFC C93C	0x609C – 0x693C
DMA4_CDSA__0_23	RW	32	0xFC C0A0– 0xFC C940	0x60A0 – 0x6940
DMA4_CSEI__0_23	RW	32	0xFC C0A4– 0x00FC C944	0x60A4 – 0x6944
DMA4_CSFI__0_23	RW	32	0xFC C0A8– 0xFC C948	0x60A8 – 0x6948
DMA4_CDEI__0_23	RW	32	0xFC C0AC– 0xFC C94C	0x60AC – 0x694C
DMA4_CDFI__0_23	RW	32	0xFC C0B0– 0xFC C950	0x60B0 – 0x6950
DMA4_CSAC__0_23	R	32	0xFC C0B4– 0xFC C954	0x60B4 – 0x6954
DMA4_CDAC__0_23	R	32	0xFC C0B8– 0xFC C958	0x60B8 – 0x6958
DMA4_CCEN__0_23	R	32	0xFC C0BC– 0xFC C95C	0x60BC – 0x695C
DMA4_CCFN__0_23	R	32	0xFC C0C0– 0xFC C960	0x60C0 – 0x6960
DMA4_COLOR__0_23	RW	32	0xFC C0C4– 0xFC C964	0x60C4 – 0x6964

Table 4−43.DSP IPI Register Summary (Public)

Register Name	Type	Register Width (Bits)	Physical Address
IPI_REVISION	R	32	0x5900 0000
IPI_SYSCONFIG	RW	32	0x5900 0010
IPI_INDEX	RW	32	0x5900 0040
IPI_ENTRY	RW	32	0x5900 0044
IPI_ENABLE	RW	32	0x5900 0048
IPI_IOMAP	RW	32	0x5900 004C
IPI_DSPBOOTCFG	RW	32	0x5900 0050

Table 4−44. DSP MMU Register Summary (Public)

Register Name	Type	Register Width (Bits)	Physical Address
MMU_REVISION	R	32	0x5A00 0000
MMU_SYSCONFIG	RW	32	0x5A00 0010
MMU_SYSSTATUS	R	32	0x5A00 0014
MMU_IRQSTATUS	RW	32	0x5A00 0018
MMU_IRQENABLE	RW	32	0x5A00 001C
MMU_WALKING_ST	R	32	0x5A00 0040
MMU_CNTL	RW	32	0x5A00 0044
MMU_FAULT_AD	R	32	0x5A00 0048
MMU_TTB	RW	32	0x5A00 004C
MMU_LOCK	RW	32	0x5A00 0050
MMU_LD_TLB	RW	32	0x5A00 0054
MMU_CAM	RW	32	0x5A00 0058
MMU_RAM	RW	32	0x5A00 005C
MMU_GFLUSH	RW	32	0x5A00 0060
MMU_FLUSH_ENTRY	RW	32	0x5A00 0064
MMU_READ_CAM	R	32	0x5A00 0068
MMU_READ_RAM	R	32	0x5A00 006C
MMU_EMU_FAULT_AD	R	32	0x5A00 0070

Table 4−45. DSP Core Register Summary

Register Name	Type	Register Width (Bits)	Virtual Address (DSP I/O Space, Byte Address)
CMR	R	32	0x0000
ICR	RW	16	0x0002
ISR	R	16	0x0004
IMSCR	RW	16	0x0006
IMSSTR	R	16	0x0008
APISIZE	R	16	0x001C
BOOTMOD	RW	16	0x001E
XCR	RW	16	0x0200
XERR	R	16	0x0204
XPSA0	R	16	0x0206
XPSA1	R	16	0x0208
XPSA2	R	16	0x020A

Table 4−45. DSP Core Register Summary (Continued)

Register Name	Type	Register Width (Bits)	Virtual Address (DSP I/O Space, Byte Address)
DCR	RW	16	0x0400[1]
DERR	R	16	0x0404[1]
DPSA0	R	16	0x0406[1]
DPSA1	R	16	0x0408[1]
DPSA2	R	16	0x040A[1]
IERR	R	16	0x0604
IPSA0	R	16	0x0606
IPSA1	R	16	0x0608
TINT0	RW	16	0x0620
TINT1	RW	16	0x0622
TINT2	RW	16	0x0624
TINT3	RW	16	0x0626
SPSA0	R	16	0x0FF2
SPSA1	R	16	0x0FF4
TIDLE	RW	16	0x0FF6
CONFIG	RW	16	0x0FFA
TMCR0	RW	16	0x0FFC
TMCR1	RW	16	0x0FFE
IGCR	RW	16	0x2800
ICFLAR0	RW	16	0x2802
ICFLAR1	RW	16	0x2804
ICNWCR	RW	16	0x2806
ICSR	R	16	0x2808
ICRCR1	R	16	0x280A
ICRTR1	RW	16	0x280C
ICRCR2	R	16	0x280E
ICRTR2	RW	16	0x2810
INWMC	R	16	0x2812

1) These registers are double mapped for future compatibility.

Table 4−46. DSP Subsystem Global Configuration Registers Summary

Register Name	Type	Register Width (Bits)	Virtual Address (DSP Memory Space, Byte Address)	Virtual Address (DSP I/O Space, Byte Address)
DSPSS_IOMAP	R	32	0xFC B000	0x5800
DSPSS_GCR	R/W	32	0xFC B004	0x5804
DSPSS_IRQSTATUS	R/W	32	0xFC B010	0x5810
DSPSS_IRQENABLE	R/W	32	0xFC B014	0x5814

4.5.3 DSP Subsystem Register Descriptions

The detailed description of individual registers is given below for the DPS subsystem IPI, DSP core modules, and global DSP subsystem configuration registers. See Chapter 9, *Memory Management Units*, Chapter 6, *Internal Interconnect*, and Chapter 10, *DMA*, for details about the MMU, INTC, and DMA subsystem module registers, respectively.

4.5.3.1 IPI Register Description

Table 4−47. IPI_REVISION

Address Offset	0x00		
Physical Address	0x5900 0000	**Instance**	IPI1
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 Examples: 0x10 for 1.0, 0x21 for 2.1	R	

Table 4−48. IPI_SYSCONFIG

Address Offset	0x10		
Physical Address	0x5900 0010	**Instance**	IPI1
Description	This register controls the OCP interface 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																															AUTOIDLE

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x00000000

Table 4−48. IPI_SYSCONFIG (Continued)

Bits	Field Name	Description	Type	Reset
0	AUTOIDLE	Internal OCP clock-gating strategy 0x0: OCP clock is free-running 0x1: Automatic OCP clock-gating strategy is applied, based on OCP interface activity.	RW	0

Table 4−49. IPI_INDEX

Address Offset	0x40		
Physical Address	0x5900 0040	**Instance**	IPI1
Description	A write to this register updates the table index (pointer) used in subsequent programming accesses of the Element Size Attribute Table. A read to this register returns the current page index value and affects neither its value nor the value of the Element Size Attribute Table Entry register. The Index register value is automatically incremented after a read or a write access to the Element Size Attribute Table Entry register (post-increment support). Auto-increment occurs only if the least-significant byte of the Element Size Attribute Table Entry register is accessed.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																PAGEINDEXVALUE															

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x0000
15:0	PAGEINDEX VALUE	Current element size page index value	RW	0x0000

Table 4−50. IPI_ENTRY

Address Offset	0x44		
Physical Address	0x5900 0044	**Instance**	IPI1
Description	A write to this register updates the Element Size Attribute value of the Memory Page pointed to by the current Element Size Attribute Table Index register value. A read of this register returns the current Element Size Attribute value of the Memory Page pointed to by the current Element Size Attribute Table Index register value. The Element Size Attribute Table Index register value is automatically incremented after a read or a write access to this Entry register (post-increment support). Auto-increment occurs only if the least-significant byte of this register is accessed.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																														ELEMSIZEVALUE	

Bits	Field Name	Description	Type	Reset
31:2	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x00000000
1:0	ELEMSIZEVALUE	Element size value 0x0: 8-bit element size 0x1: 16-bit element size 0x2: 32-bit element size 0x3: Reserved	RW	0x2

Table 4−51. IPI_ENABLE

Address Offset	0x48		
Physical Address	0x5900 0048	**Instance**	IPI1
Description	Writing 1 to bit 0 of this register enables the IPI. Writing 0 to bit 0 of this register disables the IP, no endianness conversion is done by the IPI, and OCP requests are propagated from the slave port to the master port without change. A read to this register returns the current value of the enable 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																															ENABLE

Table 4−51. IPI_ENABLE (Continued)

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x00000000
0	ENABLE	Enable for endianness conversion	RW	0

Table 4−52. IPI_IOMAP

Address Offset	0x4C		
Physical address	0x5900 004C	**Instance**	IPI1
Description	This register defines the IOMap base address for conversion from a 16-bit DSP-I/O word address to a 24-bit DSP-MEM byte 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																										IOMAPBASEADDR					

Bits	Field Name	Description	Type	Reset
31:6	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x0000000
5:0	IOMAP BASEADDR	IOMap base address	RW	0x3F

Table 4−53. IPI_DSPBOOTCFG

Address Offset	0x50		
Physical address	0x5900 0050	**Instance**	IPI1
Description	This register defines the DSP configuration for boot.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																												DSPBOOTMODE			

Table 4−53. IPI_DSPBOOTCFG (Continued)

Bits	Field Name	Description	Type	Reset
31:4	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x0000000
3:0	DSPBOOTMOD	DSP boot mode 0000: DSP boots from external memory. All other values: DSP boots from local ROM.	RW	0x0

4.5.4 DSP Core Register Descriptions

Table 4−54. CMR

Address Offset	0x0000		
Physical address	0x0000 0002	**Instance**	UMA1
Description	Control 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																														BERR	Reserved

Bits	Field Name	Description	Type	Reset
31:2	Reserved	Write 0 for future compatibility. Read returns 0.	R	0x00000000
1	BERR	XPORT bus error 0x0: No bus error occurred. 0x1: DBUS or EBUS bit in XERR register is set.	R	0
0	Reserved	Reserved	R	0

Table 4−55. ICR

Address Offset	0x002		
Physical Address	0x0000 0002	**Instance**	UMA1
Description	Idle control register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						HWA	IPORT	MPORT	XPORT	DPORT	DPLL	PERIPH	ICACHE	DMA	DSP

Table 4−55. ICR (Continued)

Bits	Field Name	Description	Type	Reset
15:10	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x00
9	HWA	HWA idle 0x0: HWA = 0: Do not idle this module when the IDLE instruction is executed. 0x1: HWA = 1: Idle this module when the IDLE instruction is executed.	RW	0
8	IPORT	IPORT idle 0x0: IPORT = 0: Do not idle this module when the IDLE instruction is executed. 0x1: IPORT = 1: Idle this module when the IDLE instruction is executed.	RW	0
7	MPORT	MPORT idle 0x0: MPORT = 0: Do not idle this module when the IDLE instruction is executed. 0x1: MPORT = 1: Idle this module when the IDLE instruction is executed.	RW	0
6	XPORT	XPORT idle 0x0: XPORT = 0: Do not idle this module when the IDLE instruction is executed. 0x1: XPORT = 1: Idle this module when the IDLE instruction is executed.	RW	0
5	DPORT	DPORT idle 0x0: DPORT = 0: Do not idle this module when the IDLE instruction is executed. 0x1: DPORT = 1: Idle this module when the IDLE instruction is executed.	RW	0

Table 4−55. ICR (Continued)

Bits	Field Name	Description	Type	Reset
4	DPLL	DPLL idle. The DPLL bit must be set for DSP SS to signal PM that it is in standby. 0x0: DPLL = 0: Do not idle this module when the IDLE instruction is executed. 0x1: DPLL = 1: Idle this module (represented by the signal UMA_IDLE_DPLL_TR) when the IDLE instruction is executed.	RW	0
3	PERIPH	PERIPH idle. The PERIPH bit does not affect peripheral idle control, but still needs to be set to 1 to enable DSP SS standby. 0x0: PERIPH = 0: Do not idle this module when the IDLE instruction is executed. 0x1: PERIPH = 1: Idle this module (represented by the signal UMA_IDLE_PERIPH_TR) when the IDLE instruction is executed.	RW	0
2	ICACHE	ICACHE idle 0x0: ICACHE = 0: Do not idle this module when the IDLE instruction is executed. 0x1: ICACHE = 1: Idle this module when the IDLE instruction is executed.	RW	0
1	DMA	DMA idle 0x0: DMA = 0: Do not idle this module when the IDLE instruction is executed. 0x1: DMA = 1: Idle this module (represented by the signal UMA_IDLE_DMA_TR) when the IDLE instruction is executed.	RW	0
0	DSP	DSP idle 0x0: DSP = 0: Do not idle this module when the IDLE instruction is executed. 0x1: DSP = 1: Idle this module when the IDLE instruction is executed.	RW	0

Table 4–56. ISR

Address Offset	0x004		
Physical Address	0x0000 0004	**Instance**	UMA1
Description	Idle status register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						HWA	IPORT	MPORT	XPORT	DPORT	DPLL	PERIPH	ICACHE	DMA	DSP

Bits	Field Name	Description	Type	Reset
15:10	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x00
9	HWA	HWA idle status 0x0: HWA = 0: HWA is not idled. 0x1: HWA = 1: HWA is idled.	R	0
8	IPORT	IPORT idle status 0x0: IPORT = 0: IPORT is not idled. 0x1: IPORT = 1: IPORT is idled.	R	0
7	MPORT	MPORT idle status 0x0: MPORT = 0: MPORT is not idled. 0x1: MPORT = 1: MPORT is idled.	R	0
6	XPORT	XPORT idle status 0x0: XPORT = 0: XPORT is not idled. 0x1: XPORT = 1: XPORT is idled.	R	0

Table 4−56. ISR (Continued)

Bits	Field Name	Description	Type	Reset
5	DPORT	DPORT idle 0x0: DPORT is not idled. 0x1: DPORT is idled.	R	0
4	DPLL	DPLL idle status 0x0: DPLL = 0: DPLL is not idled. 0x1: DPLL = 1: DPLL is idled.	R	0
3	PERIPH	PERIPH idle status 0x0: PERIPH = 0: PERIPH is not idled. 0x1: PERIPH = 1: PERIPH is idled.	R	0
2	ICACHE	ICACHE idle status 0x0: ICACHE = 0: ICACHE is not idled. 0x1: ICACHE = 1: ICACHE is idled.	R	0
1	DMA	DMA idle status 0x0: DMA = 0: DMA is not idled. 0x1: DMA = 1: DMA is idled.	R	0
0	DSP	DSP idle status 0x0: DSP = 0: DSP is not idled. 0x1: DSP = 1: DSP is idled.	R	0

Table 4−57. IMSCR

Address Offset	0x006		
Physical Address	0x0000 0006	**Instance**	UMA1
Description	Idle memory standby control register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															MSEN

Table 4–57. IMSCR (Continued)

Bits	Field Name	Description	Type	Reset
15:1	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x0000
0	MSEN	Memory standby enable 0x0: MSEN = 0: Memory standby enabled during idle 0x1: MSEN = 1: Memory standby disabled during idle	RW	0

Table 4–58. IMSSTR

Address Offset	0x008		
Physical Address	0x0000 0008	**Instance**	UMA1
Description	Idle memory standby status register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															MSEN

Bits	Field Name	Description	Type	Reset
15:1	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x0000
0	MSEN	Memory standby enable 0x0: MSEN = 0: Memory standby enabled 0x1: MSEN = 1: Memory standby disabled	R	0

Table 4−59. APISIZE

Address Offset	0x01C		
Physical Address	0x0000 001C	**Instance**	UMA1
Description	Shared memory enable register (The value of GL_APISIZE_TA[15:0] when reset is lifted)		
Type	R		

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">APISIZE</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
15:0	APISIZE	The value of GL_APISIZE_TA[15:0] when reset is lifted	R	0x0000

Table 4−60. BOOTMOD

Address Offset	0x01E		
Physical Address	0x0000 001E	**Instance**	UMA1
Description	Boot mode register		
Type	RW		

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>SLPVSS_ENABLE</td><td>SLPVDD_DISABLE</td><td colspan="10">Reserved</td><td colspan="4">SIDLEMODE</td></tr>
</table>

Bits	Field Name	Description		Type	Reset
15	SLPVSS_ENABLE	Memory standby type enable		RW	0
		0x0:	SLPVSS_ENABLE = 0: Do not enable SLPZVDD during memory standby (default condition).		
		0x1:	SLPVSS_ENABLE = 1: Enable SLPZVSS during memory standby.		
14	SLPVDD_ DISABLE	Memory standby type enable		RW	0
		0x0:	SLPVDD_DISABLE = 0: Enable SLPZVDD during memory standby (default condition).		
		0x1:	SLPVDD_DISABLE = 1: Disable SLPVDD during memory standby.		
13:4	Reserved	Write 0 for future compatibility. Read returns 0.		RW	0x000
3:0	BOOTMOD	The value of GL_BOOT_MOD_TA[3:0] when reset is lifted.		R	0x0

Table 4−61.XCR

Address Offset	0x200		
Physical Address	0x0000 0200	**Instance**	UMA1
Description	XPORT configuration register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTERREN	Reserved								PRIORITY	Reserved					

Bits	Field Name	Description	Type	Reset
15	INTERREN	Internal I/O error enable 0x0: INTERREN = 0: Do not generate bus errors for internal Reserved registers. 0x1: INTERREN = 1: Generate bus errors for internal Reserved registers.	RW	0
14:7	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x00
6	PRIORITY	Priority 0x0: PRIORITY = 0: The DSP interfaces have the highest priority. 0x1: PRIORITY = 1: The slave port and DSP have interleaved priority.	R	0
5:0	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x00

Table 4−62. XERR

Address Offset	0x204		
Physical Address	0x0000 0204	**Instance**	UMA1
Description	XPORT bus error register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTERR	RESERVED		ERROR1	Reserved					SLAVE	RESERVED	EBUS	DBUS	RESERVED		

Bits	Field Name	Description	Type	Reset
15	INTERR	Internal I/O error 0x0: INTERR = 0: No error 0x1: INTERR = 1: Timeout error occurred for internal accesses.	R	0
14:13	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x0
12	ERROR1	ERROR1 error 0x0: ERROR1 = 0: No error 0x1: ERROR1 = 1: Error1 error	R	0
11:7	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x00
6	SLAVE	Slave error 0x0: SLAVE = 0: No error 0x1: SLAVE = 1: Error occurred from XPORT slave access.	R	0
5	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0
4	EBUS	EBUS error 0x0: EBUS = 0: No error 0x1: EBUS = 1: Error occurred from EBUS access.	R	0

Table 4−62. XERR (Continued)

Bits	Field Name	Description	Type	Reset
3	DBUS	DBUS error 0x0: DBUS = 0: No error 0x1: DBUS = 1: Error occurred from DBUS access.	R	0
2:0	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x0

Table 4−63. XPSA0

Address Offset	0x206		
Physical Address	0x0000 0206	**Instance**	UMA1
Description	XPORT PSA register 0 If the PSAEN bit of the TMCR0 register is set, this field records the master interface output states. This register is reset only when XPORT is reset.		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSA_15_0															

Bits	Field Name	Description	Type	Reset
15:0	PSA_15_0		R	0x0000

Table 4−64. XPSA1

Address Offset	0x208		
Physical Address	0x0000 0208	**Instance**	UMA1
Description	XPORT PSA register 1 If the PSAEN bit of the TMCR0 register is set, this field records the master interface output states. This register is reset only when XPORT is reset.		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSA_31_16															

Bits	Field Name	Description	Type	Reset
15:0	PSA_31_16		R	0x0000

Table 4−65. XPSA2

Address Offset	0x20A		
Physical Address	0x0000 020A	**Instance**	UMA1
Description	XPORT PSA register 2 If the PSAEN bit of the TMCR0 register is set, this field records the master interface output states. This register is reset only when XPORT is reset.		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								PSA_39_32							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x00
7:0	PSA_39_32		R	0x00

Table 4−66. DCR

Address Offset	0x400 or 0x1000		
Physical Address	0x0000 0400	**Instance**	UMA1
Description	DPORT configuration register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								WPE	PRIORITY	Reserved					

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x00
7	WPE	Write-posting enable 0x0: WPE = 0: Write posting disabled 0x1: WPE = 1: Write posting enabled	RW	0

Table 4−66. DCR (Continued)

Bits	Field Name	Description	Type	Reset
6	PRIORITY	Priority 0x0: PRIORITY = 0: The DSP interfaces have the highest priority. 0x1: PRIORITY = 1: The slave port and DSP have interleaved priority.	R	0
5:0	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x00

Table 4−67. DERR

Address Offset	0x404 or 0x1004		
Physical Address	0x0000 0404	**Instance**	UMA1
Description	DPORT bus error register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED			ERROR1	Reserved					SLAVE	FBUS	EBUS	DBUS	CBUS	BBUS	RESERVED

Bits	Field Name	Description	Type	Reset
15:13	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x0
12	ERROR1	ERROR1 error 0x0: ERROR1 = 0: No error 0x1: ERROR1 = 1: Error1 error	R	0
11:7	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x00
6	SLAVE	Slave error 0x0: SLAVE = 0: No error 0x1: SLAVE = 1: Error occurred from XPORT slave access.	R	0

Table 4−67. DERR (Continued)

Bits	Field Name	Description	Type	Reset
5	FBUS	FBUS error 0x0: FBUS = 0: No error 0x1: FBUS = 1: Error occurred from FBUS access.	R	0
4	EBUS	EBUS error 0x0: EBUS = 0: No error 0x1: EBUS = 1: Error occurred from EBUS access.	R	0
3	DBUS	DBUS error 0x0: DBUS = 0: No error 0x1: DBUS = 1: Error occurred from DBUS access.	R	0
2	CBUS	CBUS error 0x0: CBUS = 0: No error 0x1: CBUS = 1: Error occurred from CBUS access.	R	0
1	BBUS	BBUS error 0x0: BBUS = 0: No error 0x1: BBUS = 1: BBUS access has occurred through DPORT, which is illegal.	R	0
0	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0

Table 4−68. DPSA0

Address Offset	0x406 or 0x1006		
Physical Address	0x0000 0406	**Instance**	UMA1
Description	DPORT PSA register 0 If the PSAEN bit of the TMCR0 register is set, this field records the master interface output states. This register is reset only when DPORT is reset.		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSA_15_0															

Bits	Field Name	Description	Type	Reset
15:0	PSA_15_0		R	0x0000

Table 4−69. DPSA1

Address Offset	0x408 or 0x1008		
Physical Address	0x0000 0408	**Instance**	UMA1
Description	DPORT PSA register 1 If the PSAEN bit of the TMCR0 register is set, this field records the master interface output states. This register is reset only when DPORT is reset.		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSA_31_16															

Bits	Field Name	Description	Type	Reset
15:0	PSA_31_16		R	0x0000

Table 4−70. DPSA2

Address Offset	0x40A or 0x100A		
Physical Address	0x0000 040A	**Instance**	UMA1
Description	DPORT PSA register 2 If the PSAEN bit of the TMCR0 register is set, this field records the master interface output states. This register is reset only when DPORT is reset.		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								PSA_39_32							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x00
7:0	PSA_39_32		R	0x00

Table 4−71. IERR

Address Offset	0x604		
Physical Address	0x0000 0604	**Instance**	UMA1
Description	IPORT bus error register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED			ERROR1	Reserved											PBUS

Bits	Field Name	Description	Type	Reset
15:13	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x0
12	ERROR1	ERROR1 error 0x0: ERROR1 = 0: No error 0x1: ERROR1 = 1: Error1 error	R	0
11:1	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x000
0	PBUS	PBUS error 0x0: PBUS = 0: No error 0x1: PBUS = 1: Error occurred from PBUS access.	R	0

Table 4−72. IPSA0

Address Offset	0x606		
Physical Address	0x0000 0606	**Instance**	UMA1
Description	IPORT PSA register 0 If the PSAEN bit of the TMCR0 register is set, this field records the master interface output states. This register is reset only when IPORT is reset.		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSA_15_0															

Bits	Field Name	Description	Type	Reset
15:0	PSA_15_0		R	0x0000

Table 4−73. IPSA1

Address Offset	0x608		
Physical Address	0x0000 0608	**Instance**	UMA1
Description	IPORT PSA register 1 If the PSAEN bit of the TMCR0 register is set, this field records the master interface output states. This register is reset only when IPORT is reset.		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												PSA_19_16			

Bits	Field Name	Description	Type	Reset
15:4	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x000
3:0	PSA_19_16		R	0x0

Table 4−74. TINT0

Address Offset	0x620		
Physical Address	0x0000 0620	**Instance**	UMA1
Description	Test interrupt register 0		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT_15_0															

Bits	Field Name	Description	Type	Reset
15:0	INT_15_0	Interrupt feedback When COUNT is written with a nonzero value and counts down to zero, this field becomes the GL_INT_NA[15:0] input (when the FDBK bit of the TMCR0 register is set).	RW	0xFFFF

Table 4−75. TINT1

Address Offset	0x622		
Physical Address	0x0000 0622	**Instance**	UMA1
Description	Test interrupt register 1		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNT_6_0							BERR	NMI	RESET	INT_21_16					

Table 4−75. TINT1 (Continued)

Bits	Field Name	Description	Type	Reset
15:9	COUNT_6_0	Countdown to feedback	RW	0x00
8	BERR	Bus error feedback When COUNT is written with a nonzero value and counts down to zero, this field becomes the GL_BUSERR_NA input (when the FDBK bit of the TMCR0 register is set).	RW	1
7	NMI	NMI feedback When COUNT is written with a nonzero value and counts down to zero, this field becomes the GL_NMI_NA input (when the FDBK bit of the TMCR0 register is set).	RW	1
6	RESET	Reset feedback When COUNT is written with a nonzero value and counts down to zero, this field becomes the GL_RESET_NA input (when the FDBK bit of the TMCR0 register is set).	RW	1
5:0	INT_21_16	Interrupt feedback When COUNT is written with a nonzero value and counts down to zero, this field becomes the GL_INT_NA[21] input (when the FDBK bit of the TMCR0 register is set).	RW	0x3F

Table 4−76. TINT2

Address Offset	0x624		
Physical Address	0x0000 0624	**Instance**	UMA1
Description	Test interrupt register 2		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV_15_0															

Bits	Field Name	Description	Type	Reset
15:0	DIV_15_0	Feedback countdown divider Divided value of feedback counter	RW	0x0000

Table 4−77. TINT3

Address Offset	0x626		
Physical Address	0x0000 0626	**Instance**	UMA1
Description	Test interrupt register 3		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WRAP	ANDEN	Reserved					DIV_24_16								

Table 4−77. TINT3 (Continued)

Bits	Field Name	Description	Type	Reset
15	WRAP	Wrap enable When WRAPEN is set to 1, the countdown for feedback is continuous. When the feedback interrupt occurs, the process resumes.	RW	0
14	ANDEN	AND enable When ANDEN is set to 1, an interrupt occurs by combining the INT/NMP and BERR inputs with the normal interrupts when they take effect. This bit has no effect when FDBK is enabled.	RW	0
13:9	Reserved		RW	0x00
8:0	DIV_24_16	Feedback countdown divider Divided value of feedback counter	RW	0x000

Table 4−78. SPSA0

Address Offset	0xFF2		
Physical Address	0x0000 0FF2	**Instance**	UMA1
Description	SPORT PSA register 0 If the PSAEN bit of the TMCR0 register is set, this field records the master interface output states. This register is reset only when UMA core is reset.		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSA_15_0															

Bits	Field Name	Description	Type	Reset
15:0	PSA_15_0		R	0x0000

Table 4−79. SPSA1

Address Offset	0xFF4		
Physical Address	0x0000 0FF4	**Instance**	UMA1
Description	SPORT PSA register 1 If the PSAEN bit of the TMCR0 register is set, this field records the master interface output states. This register is reset only when the UMA core is reset.		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												PSA_19_16			

Bits	Field Name	Description	Type	Reset
15:4	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x000
3:0	PSA_19_16	Temporary data storage for DFT functional testing	R	0x0

Table 4–80. TIDLE

Address Offset	0xFF6		
Physical Address	0x0000 0FF6	**Instance**	UMA1
Description	Idle testing register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PORTFORCE	RSSFORCE	DSPFORCE	Reserved												

Bits	Field Name	Description	Type	Reset
15	PORTFORCE	PORT force out of idle 0x0: PORTFORCE = 0: Ports behave normally. 0x1: PORTFORCE = 1: Force all ports to run clocks (highest priority signal).	RW	0
14	RSSFORCE	RSS force out of idle 0x0: RSSFORCE = 0: This field behaves normally. 0x1: RSSFORCE = 1: Force this field to run clocks, including auto-idle (highest priority signal).	RW	0
13	DSPFORCE	DSP force out of idle 0x0: DSPFORCE = 0: This field behaves normally. 0x1: DSPFORCE = 1: Force this field to run clocks (highest priority signal).	RW	0
12:0	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x0000

Table 4–81. CONFIG

Address Offset	0x00FFA		
Physical Address	0x0000 0FFA	**Instance**	UMA1
Description	System configuration register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	Reserved			Reserved					Reserved	ICACHEPRES	PRIODMA	Reserved	Reserved		

Bits	Field Name	Description	Type	Reset
15	Reserved	Reserved. This bit must not be changed by software.	RW	1
14:12	Reserved	Reserved	R	0x0
11:7	Reserved	Reserved. This bit must be left at its reset value.	RW	0x04
6	Reserved	Reserved	R	0
5	ICACHEPRES	ICACHE present Instruction cache status 0x0: ICACHE is disabled. 0x1: ICACHE is enabled.	R	0
4	PRIODMA	DMA priority 0x0: DSP has higher priority than MPORT to access RSS. 0x1: MPORT and DSP have interleaved priority to access RSS.	RW	0
3:0	Reserved	Reserved. This bit must be left at its reset value.	RW	0x0

Table 4−82. TMCR0

Address Offset	0xFFC		
Physical Address	0x0000 0FFC	**Instance**	UMA1
Description	Test mode control register 0		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				TPORTEN	TESTBOOT	LDBYPASS	TESTDONE	TESTFAIL	TBOOTINIT	PSAEN	RESERVED	MMAP2	MMAP1	FDBKXD	FDBK

Bits	Field Name	Description	Type	Reset
15:12	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x0
11	TPORTEN	Value of TPORT_TPORTEN_TR signal	RW	0
10	TESTBOOT	Value of TPORT_TESTBOOT_TR signal	RW	0
9	LDBYPASS	Bypass synchronization logic inside TPORT 0x0: LDBYPASS=0 Bypass of TPORT synchronization logic is disabled. 0x1: LDBYPASS=1 Bypass of TPORT synchronization logic is enabled.	RW	0
8	TESTDONE	Indicates whether the test is done 0x0: TESTDONE=0 The test is not done. 0x1: TESTDONE=1 The test is done.	RW	0
7	TESTFAIL	Indicates whether the test failed 0x0: TESTFAIL=0 The test did not fail. 0x1: TESTFAIL=1 The test failed.	RW	0
6	TBOOTINIT	Enables test boot mode. External program fetches are sent to TPORT. Has higher priority than MPNMC. Has the same effect as MNNMC memory map, except that the external range is TPORT. To test real MPNMC functionality, TBOOTINIT must be inactive. 0x0: TBOOTINIT=0 TESTBOOT mode is disabled. 0x1: TBOOTINIT=1 TESTBOOT mode is enabled.	RW	0

Table 4−82. TMCR0 (Continued)

Bits	Field Name	Description	Type	Reset
5	PSAEN	Enables PSA registers 0x0: PSAEN=0 PSA registers update is disabled. 0x1: PSAEN=1 PSA registers update is enabled.	RW	0
4	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0
3	MMAP2	Enables MMAP2 memory map 0x0: MMAP2=0: MMAP2 is disabled. 0x1: MMAP2=1: MMAP2 is enabled.	RW	0
2	MMAP1	Enables MMAP1 memory map 0x0: MMAP1=0 MMAP1 is disabled. 0x1: MMAP1=1 MMAP1 is enabled.	RW	0
1	FDBKXD	Enables FDBK mode 0x0: FDBKXD=0: Feedback mode is disabled. 0x1: Feedback mode is enabled.	RW	0
0	FDBK	Enables global feedback control, which causes UMA input ports to be driven by FDBK logic, or to known states. 0x0: FDBK=0: UMA input ports not driven by FDBK logic (functional behavior). 0x1: FDBK=1: UMA input ports driven by FDBK logic.	RW	0

Table 4−83. TMCR1

Address Offset	0xFFE		
Physical Address	0x0000 0FFE	**Instance**	UMA1
Description	Test mode control register 1		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	HOMINIT	APISIZE	RESERVED	PDISMISS	WRESP	CLKRATIO		OCP	HMREQ	PWAKEUP	ICTEST	BOOTMOD			

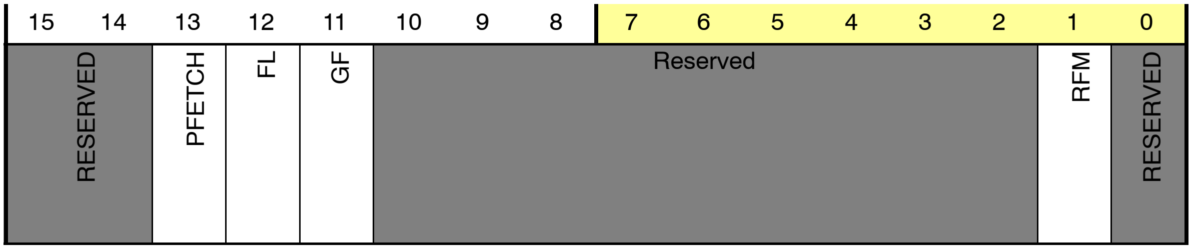
Bits	Field Name	Description	Type	Reset
15	Reserved		RW	0
14	HOMINIT	Feed back this field to all GL_HOMSAM_INIT_TA[3:0] signals.	RW	0
13	APISIZE	Feed back this field to all GL_APISIZE_TA[15:0] signals.	RW	0
12	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0
11	PDISMISS	Feed back this field to the GL_PDISMISS_ENABLE_TA signal.	RW	0
10	WRESP	Feed back this field to the GL_*_WRESP_TA signals	RW	0
9:8	CLKRATIO	Feed back this field to all QSEL_RATIO_TA[1:0] signals. 0x0: TESTDONE=0 The test is not done. 0x1: TESTDONE=1 The test is done.	RW	0x0
7	OCP	Feed back this field to GL_OCP_TA when feedback is enabled. 0x0: TESTFAIL=0 The test did not fail. 0x1: TESTFAIL=1 The test failed.	RW	0
6	HMREQ	Feed back this field to GL_HMREQ_TR when feedback is enabled. 0x0: TBOOTINIT=0 TESTBOOT mode is disabled. 0x1: TBOOTINIT=1 TESTBOOT mode is enabled.	RW	0

Table 4−83. TMCR1 (Continued)

Bits	Field Name	Description	Type	Reset
5	PWAKEUP	Feed back this field to GL_PORTWAKEUP_TA when feedback is enabled. 0x0: PSAEN=0 PSA registers update is disabled. 0x1: PSAEN=1 PSA registers update is enabled.	RW	0
4	ICTEST	Enables ICACHE memory remapping to I/O space	RW	0
3:0	BOOTMOD	Feed back this field to GL_BOOT_MOD_TA[3:0] when feedback is enabled.	RW	0x0

Table 4−84. IGCR

Address Offset	0x2800		
Physical Address	0x0000 2800	**Instance**	UMA1
Description	ICACHE global configuration register		
Type	RW		



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		PFETCH	FL	GF	Reserved									RFM	RESERVED

Bits	Field Name	Description	Type	Reset
15:14	Reserved	Write 0 for future compatibility. Read returns 0.	R	0x3
13	PFETCH	Prefetch enable 0x0: PE = 0: Enable UMA_ICACHE_PFETCH_TR bit = 0 (disable L2 prefetch). 0x1: PE = 1: Enable UMA_ICACHE_PFETCH_TR bit = 1 (enable L2 prefetch).	RW	0
12	FL	Flush line 0x0: FL = 0: Set to 0 when the line flush is complete. Flush line can only happen if ICACHE is enabled. 0x1: FL = 1: Line flush started when this bit is written to 1; no effect is written to 0.	R	0

Table 4−84. IGCR (Continued)

Bits	Field Name	Description	Type	Reset
11	GF	Global flush mode 0x0: GF = 0: ICACHE clear (CACLR of ST3 register) must take into account the FLUSH bits of the ICNWCR, ICRCR1, and ICRCR2 registers. 0x1: GF = 1: The entire ICACHE is flushed when the CACLR bit if the ST3 register is set to 1.	RW	0
10:2	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x18F
1	RFM	RAMSET fill mode 0x0: RFM = 0: Validate the corresponding RAMSET tag after a write or refill into a particular RAMSET once a line is filled. Line Valid Bits of the RAMSET remain 0 until a specific line is filled due to a fetch. Complete fill is not triggered by writing the tag. 0x1: RFM = 1: Preload all lines of the RAMSET before setting the Valid Bit of the corresponding RAMSET tag. Incoming requests do not have a valid compare and are sent to IPORT.	RW	1
0	Reserved	Write 0 for future compatibility. Read returns 0.	R	0

Table 4−85. ICFLAR0

Address Offset	0x2802		
Physical Address	0x0000 2802	**Instance**	UMA1
Description	ICACHE flush line address register 0		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_15_0															

Bits	Field Name	Description	Type	Reset
15:0	ADDR_15_0	Address If the FL bit of the IGCR register is written, the address value written to this register is flushed.	RW	0x0000

Table 4−86. ICFLAR1

Address Offset	0x2804		
Physical Address	0x0000 2804	**Instance**	UMA1
Description	ICACHE flush line address register 1		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								ADDR_23_16							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x00
7:0	ADDR_23_16	Address If the FL bit of the IGCR register is written, the address value written to this register is flushed.	RW	0x00

Table 4−87. ICNWCR

Address Offset	0x2806		
Physical Address	0x0000 2806	**Instance**	UMA1
Description	ICACHE N-way control register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														FLUSH	RESERVED

Bits	Field Name	Description	Type	Reset
15:2	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x0000
1	FLUSH	2-way flush mask 0x0: FL = 0: The 2-way is not flushed when the CACLR bit of the ST3 register and the GF bits of the IGCR registers are set to 1. 0x1: FL = 1: The 2-way is flushed when the CACLR bit of the ST3 register is set and the GF bits of the IGCR registers are set to 1.	RW	0
0	Reserved	Write 0 for future compatibility. Read returns 0.	RW	1

Table 4−88. ICSR

Address Offset	0x2808		
Physical Address	0x0000 2808	**Instance**	UMA1
Description	ICACHE status register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															BERR

Bits	Field Name	Description	Type	Reset
15:1	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x0000
0	BERR	Bus Error 0x0: BERR = 0: Set to 0 when it is read. 0x1: BERR = 1: Set to 1 if the ICRCR1 and ICRCR2 registers are configured to be the same. The DSP receives a bus error interrupt.	R	0

Table 4−89. ICRCR1

Address Offset	0x280A		
Physical Address	0x0000 280A	**Instance**	UMA1
Description	ICACHE RAMSET control register 1		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAGVALID	Reserved													FLUSH	ENABLE

Bits	Field Name	Description	Type	Reset
15	TAGVALID	TAG valid 0x0: TAGVALID = 0: Set to 0 when the corresponding ICRTR register is written. 0x1: TAGVALID = 1: When RFM of IGCR register is set, TAGVALID is set to 1 after all lines in the RAMSET are preloaded. The preload starts when the corresponding ICRTR is written.	R	0

Table 4−89. ICRCR1 (Continued)

Bits	Field Name	Description	Type	Reset
14:2	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x0000
1	FLUSH	RAMSET flush mask 0x0: FL = 0: The RAMSET is not flushed when the CACLR bit if the ST3 register is set to 1 and the GF bits of the IGCR registers are set to 1. 0x1: FL = 1: The RAMSET is flushed when the CACLR bit if the ST3 register is set to 1 and the GF bits of the IGCR registers are set to 1.	RW	0
0	ENABLE	RAMSET enable mask 0x0: ENABLE = 0: RAMSET is not enabled by CAEN bit of the ST3 register. 0x1: ENABLE = 1: RAMSET is enabled by CAEN bit of the ST3 register. This bit cannot be modified if the CAEN bit of ST3 register is already set to 1.	RW	1

Table 4−90. ICRTR1

Address Offset	0x280C		
Physical Address	0x0000 280C	**Instance**	UMA1
Description	ICACHE RAMSET TAG register 1		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				TAG											

Bits	Field Name	Description	Type	Reset
15:12	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x0
11:0	TAG	RAMSET tag TAG value for corresponding RAMSET	RW	0x000

Table 4–91. ICRCR2

Address Offset	0x280E		
Physical Address	0x0000 280E	**Instance**	UMA1
Description	ICACHE RAMSET control register 2		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAGVALID	Reserved													FLUSH	ENABLE

Bits	Field Name	Description	Type	Reset
15	TAGVALID	Tag valid 0x0: TAGVALID = 0: Set to 0 when the corresponding ICRTR register is written. 0x1: TAGVALID = 1: When RFM of IGCR register is set, TAGVALID is set to 1 after all lines in the RAMSET are preloaded. The preload starts when the corresponding ICRTR is written.	R	0
14:2	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x0000
1	FLUSH	RAMSET flush mask 0x0: FL = 0: The RAMSET is not flushed when the CACLR bit if the ST3 register is set to 1, and the GF bits of the IGCR registers are set to 1. 0x1: FL = 1: The RAMSET is flushed when the CACLR bit if the ST3 register is set to 1, and the GF bits of the IGCR registers are set to 1.	RW	0
0	ENABLE	RAMSET enable mask 0x0: ENABLE = 0: RAMSET is not enabled by the CAEN bit of the ST3 register. 0x1: ENABLE = 1: RAMSET is enabled by the CAEN bit of the ST3 register. This bit cannot be modified if the CAEN bit of ST3 register is already set to 1.	RW	1

Table 4−92. ICRTR2

Address Offset	0x2810		
Physical Address	0x0000 2810	**Instance**	UMA1
Description	ICACHE RAMSET TAG register 2		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				TAG											

Bits	Field Name	Description	Type	Reset
15:12	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x0
11:0	TAG	RAMSET TAG TAG value for corresponding RAMSET	RW	0x000

Table 4−93. INWMC

Address Offset	0x2812		
Physical Address	0x0000 2812	**Instance**	UMA1
Description	ICACHE N-way miss counter register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNT															

Bits	Field Name	Description	Type	Reset
15:0	COUNT	Count value Count value of 2-way misses	R	0x0000

4.5.5 DSP SS Global Configuration Register Descriptions

Table 4−94. DSPSS_IOMAP

Address Offset	0x00		
Physical Address	0x00005800	**Instance**	DSS1
Description	This register defines the IOMap base address for conversion from a 16-bit DSP-I/O word address to a 24-bit DSP-MEM byte address.		
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																										IOMAP BASEADDR					

Bits	Field Name	Description	Type	Reset
31:6	Reserved	Write 0 for future compatibility. Read returns 0.	R	0x0000000
5:0	IOMAP BASEADDR	IOMap base address	R	0x3F

Table 4−95. DSPSS_GCR

Address Offset	0x04		
Physical Address	0x00005804	**Instance**	DSS1
Description	This register enables cutting clocks at the DSP subsystem level to save power.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															DDMACLKENABLE

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x00000000
0	DDMACLK ENABLE	Enable the root clock of the dDMA.	RW	1

Table 4−96. DSPSS_IRQSTATUS

Address Offset	0x10		
Physical Address	0x00005810	**Instance**	DSS1
Description	The interrupt status register contains the status of out-of-band error reporting interrupts. Writing 1 to a bit resets this 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																															OUTBANDERRSTATUS

Table 4−96. DSPSS_IRQSTATUS (Continued)

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x00000000
0	OUTBANDERRSTATUS	Out-of-band error interrupt status bit	RW	0

Table 4−97. DSPSS_IRQENABLE

Address Offset	0x14		
Physical Address	0x00005814	**Instance**	DSS1
Description	The interrupt enable register enables masking/unmasking out-of-band reporting 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																															OUTBANDERRENABLE

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x00000000
0	OUTBANDERR ENABLE	Out-of-band error interrupt enable bit	RW	0

Chapter 5

Power, Reset, and Clock Management

This chapter describes, reset, and clock management (PRCM) in the OMAP2420 multimedia device.

Topic **Page**

5.1 Power, Reset, and Clock Manager Overview

5.1.1 PRCM Introduction

The OMAP2420 significantly reduces dynamic power consumption and static leakage current to extend end-product battery life. An enhanced power-management subsystem called the power, reset, and clock manager (PRCM) implements this reduction:

- Chip partitioning into five independent power domains
- Fine granular clock and reset control
- Voltage scaling support
- Memory retention support
- Centralized management of power, reset, and clock functions

The PRCM interfaces with every component of the OMAP device for clock, reset, and power management using L3 interconnect sideband power control signals.

The PRCM integrates enhanced features that allow the device to adapt energy consumption dynamically according to changing application needs and performance requirements. An innovative hardware architecture allows for a drastic reduction in leakage current.

The PRCM is composed of two main entities:

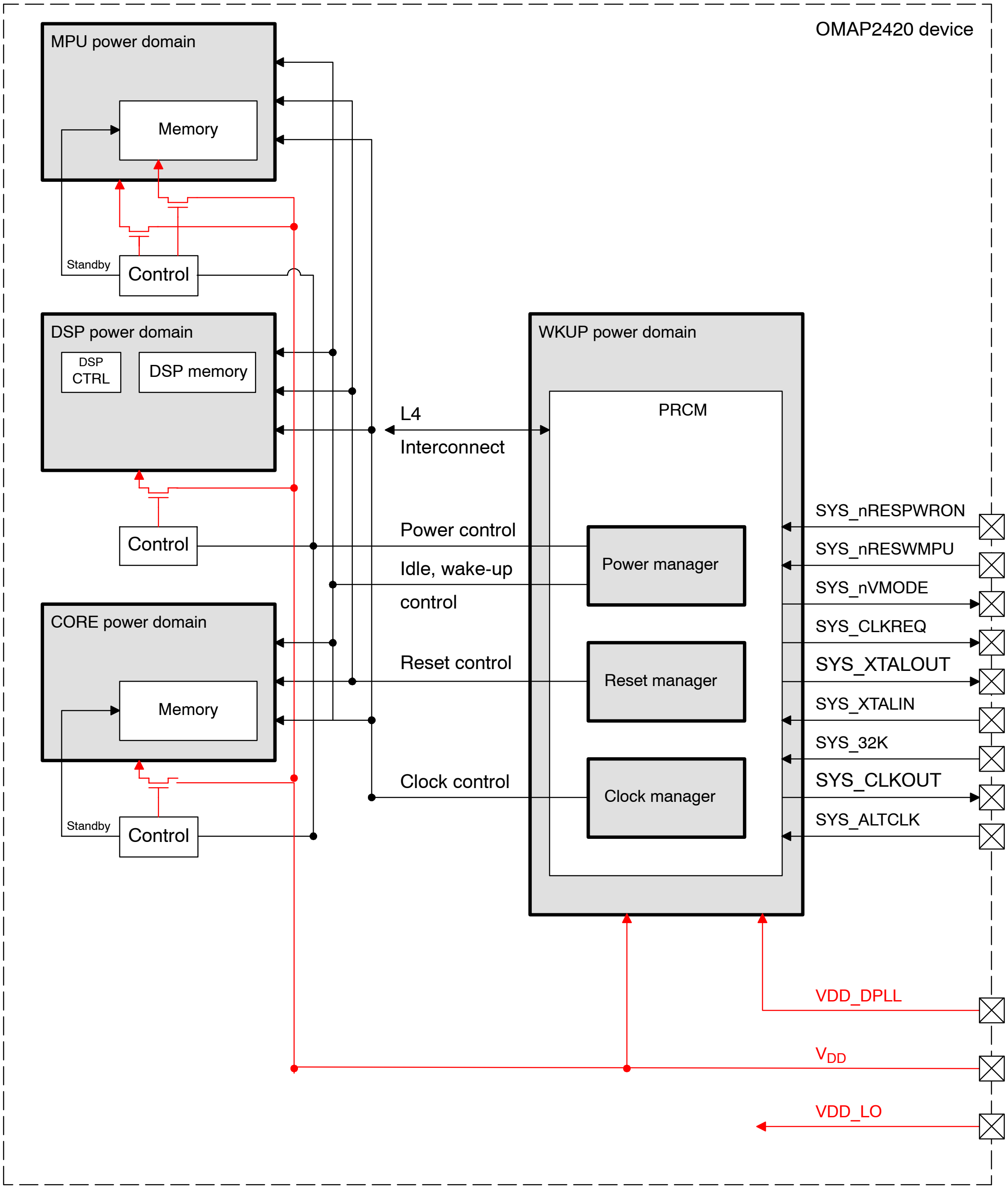
- Power/reset/wake-up manager

- Clock generation, distribution, and manager

The PRCM is fully configurable through its L4 interface port.

Figure 5−1 is an overview of the PRCM.

Figure 5−1. PRCM Overview



5.1.2 Power-Saving Techniques

The OMAP2420 architecture integrates three main power-management capabilities to reduce dynamic consumption and static leakage:

☐ Chip partitioning into five independent power domains
☐ Voltage scaling support (DVS)
☐ Memory retention support

5.1.2.1 Dynamic Power Savings

An efficient OMAP2420 idle scheme allows clocks to be activated and deactivated safely without complex software management. The PRCM controls the idle scheme:

- ❏ Efficient local clock autogating for each module
- ❏ Implementation of control signals between the PRCM and each module

The same voltage source supplies the five power domains and can be externally scaled from high-voltage point to low-voltage point. A companion power IC works with the OMAP2420 to implement an efficient communication protocol for controlling voltage supplied to the device.

5.1.2.2 Static Leakage Control

Internal RAM memory can be set in standby (retention) mode independently of the logic. In this state, memory is not operational, but content is retained with minimal leakage. This feature allows power consumption to be reduced when the device is in sleep modes while maintaining memory contents for a fast context restore. The memory standby feature is software-controllable.

Embedded V_{DD} switches are implemented for all power domains except those containing the PRCM. This allows the domains to be powered off, if not in use, to provide maximum power savings.

Note:

In retention mode, to protect against corrupting data, there is no access of internal memory.

5.1.3 PRCM Features

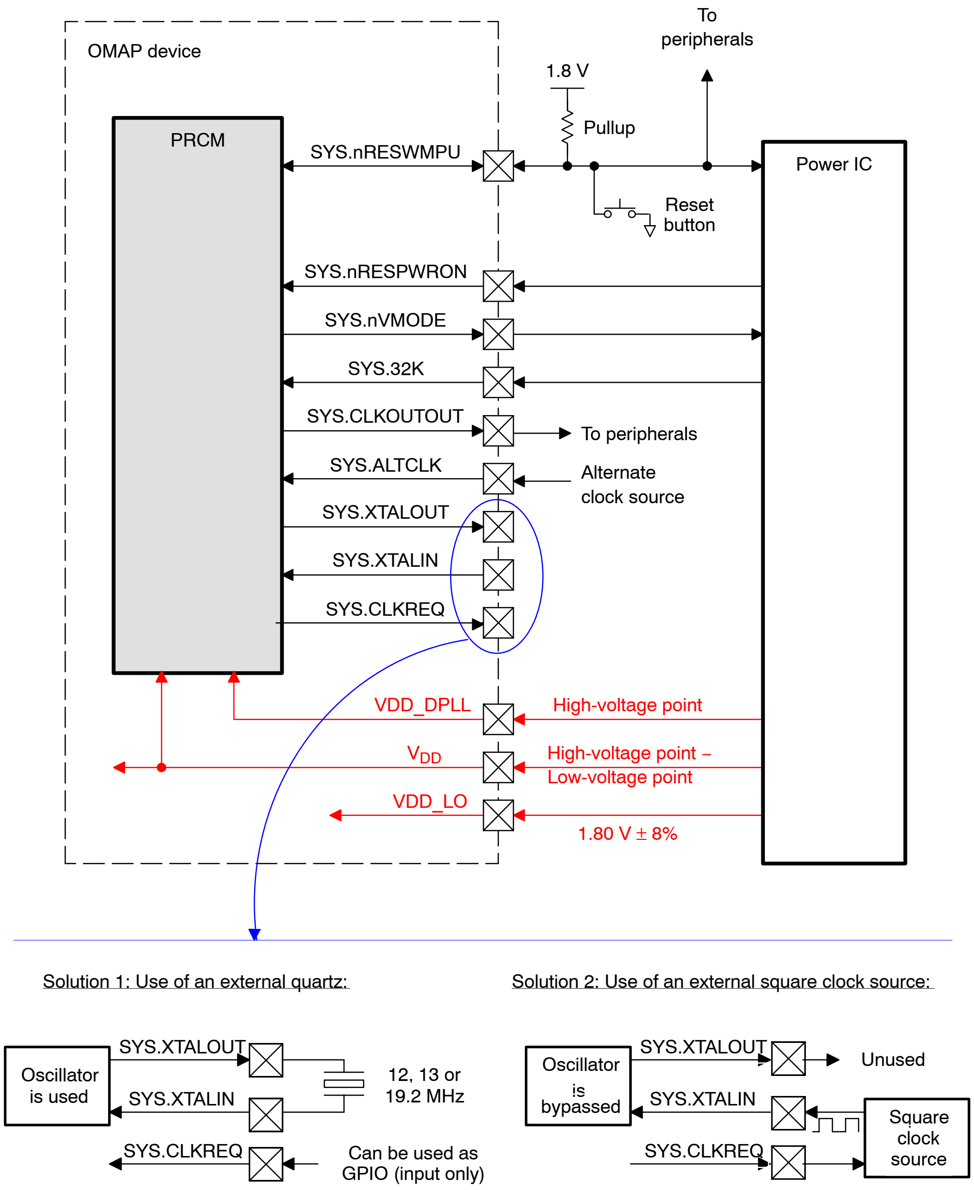
The PRCM includes the following features:

- ❏ Power-up sequence handling throughout the OMAP2420
- ❏ Clock and reset management and distribution with high granularity to device subsystems
- ❏ Control of external supply voltage regulation
- ❏ Management of five independent power domains (four with embedded V_{DD} switches)
- ❏ Power domains containing multiple clock domains for more granular power control
- ❏ Software and partial hardware control of domains
- ❏ Monitoring and handling of idle and wake-up events
- ❏ Debug features

5.2 Power, Reset, and Clock Manager Environment

To obtain the most efficient operation, the OMAP2420 uses a custom power IC device and external clock sources. Figure 5−2 shows the interface to a power IC device and external clock sources.

Figure 5−2. PRCM Functional External Interface (Detailed View)



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

5.2.1 Clock External Interface

All device clocks except the 32-kHz clock are derived from independent sources outside the power IC (see Figure 5−3 and Table 5−1).

Figure 5−3. External Clock Interface

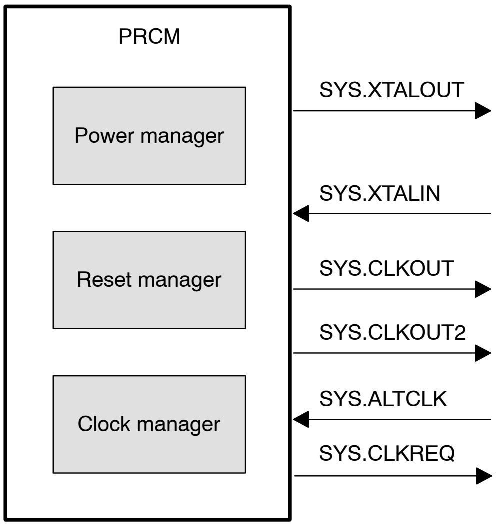


Table 5−1. Clock Interface Description

Signal Name	I/O	Description	Reset
SYS.XTALOUT	O	Crystal output	0
SYS.XTALIN	I	Crystal or square clock input	N/A
SYS.CLKOUT	O	Output clock	0
SYS.CLKOUT2	O	Output clock 2	0
SYS.ALTCLK	I	Alternate external clock input	N/A
SYS.CLKREQ	O	Clock request signal	1

The system clock source can be one of two options:

- CMOS digital clock that enters on the SYS.XTALIN pin
- Crystal oscillator clock managed by SYS.XTALIN and SYS.XTALOUT

Only one of the clock sources can be used at a time.

CAUTION

The SYS.CLKREQ pin is an OMAP2420 output used to turn the system clock switch on or off. It can be used as a GPIO if it is not used as a clock request.

An alternate clock input, SYS.ALTCLK, provides a precise clock for NTSC (54 MHz), USB (48 MHz), or any other frequency for the timers. If it is not used as a clock input, this pin can be configured as a GPIO.

5.2.2 Power IC External Interface

The power IC interfaces to external reset, 32-kHz clock, DVS, and clock-management signals (see Figure 5−4 and Table 5−2).

Figure 5−4. External Power IC Interface

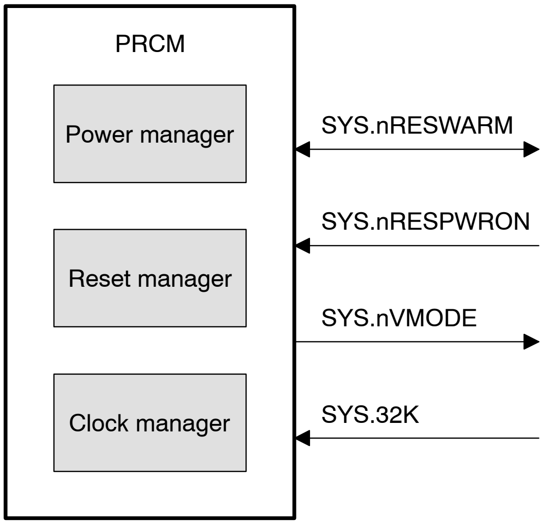


Table 5−2. Power IC Interface Description

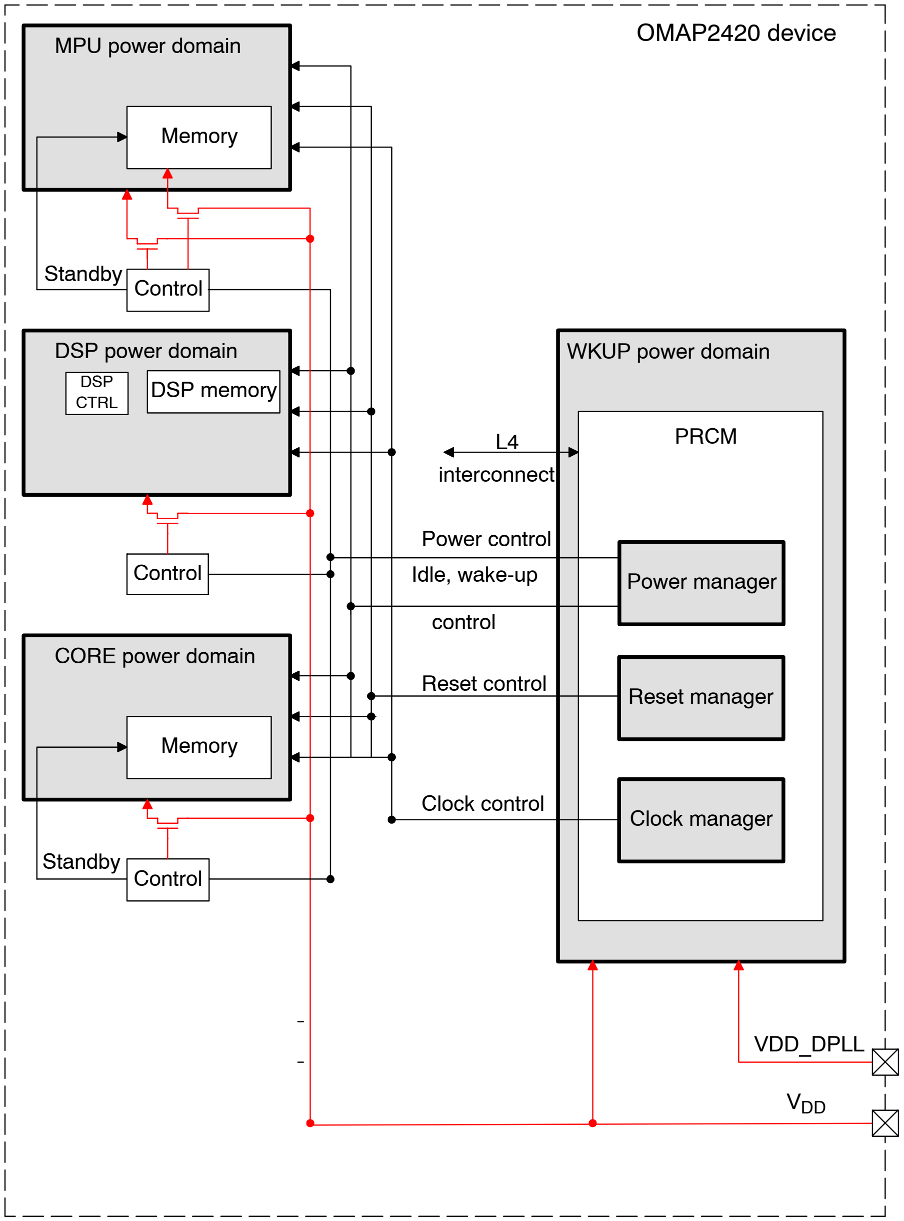
Signal Name	**I/O**	**Description**	**Reset**
SYS.nRESPWRON	I	Power-on reset	N/A
SYS.nRESWARM_IN	I	Warm-boot reset	N/A
SYS.nRESWARM_OUT	O	External peripheral reset	N/A
SYS.32K	I	32-kHz digital CMOS clock input	N/A
SYS.nVMODE	O	Core voltage indicator	1

5.3 Power, Reset, and Clock Manager Integration

5.3.1 PRCM Integration Overview

Figure 5–5 shows the PRCM integration.

Figure 5–5. PRCM Integration Overview

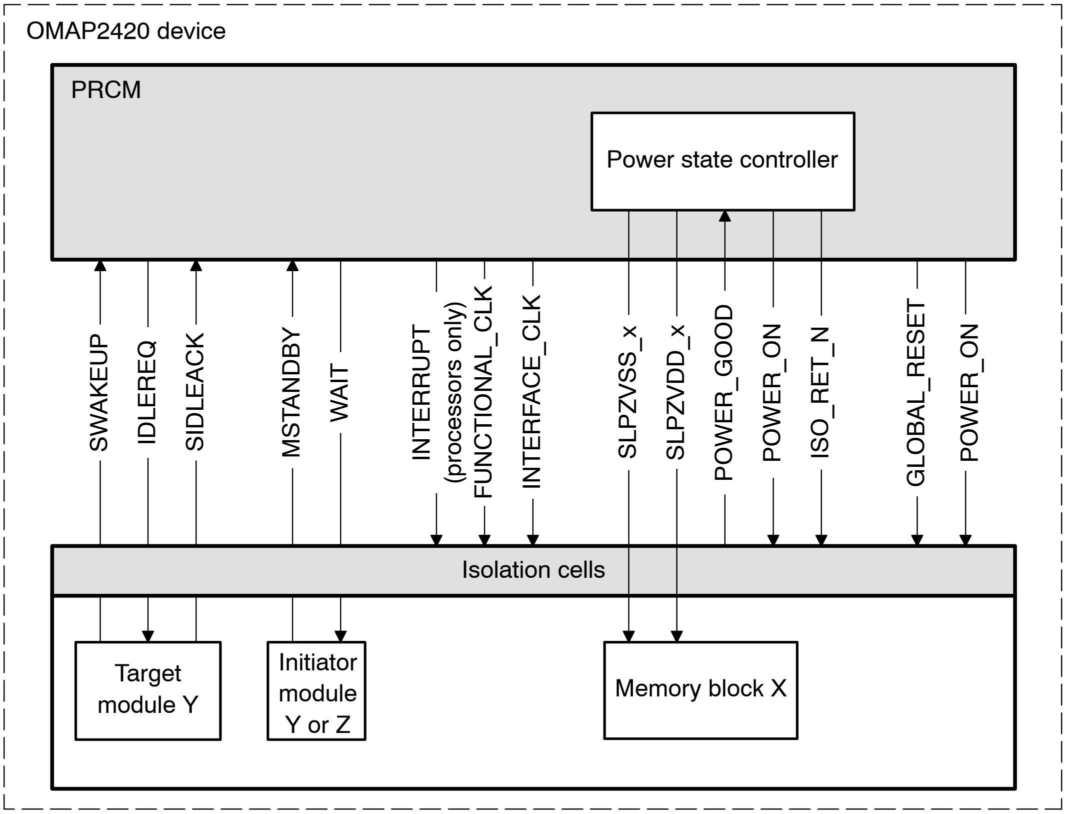


PRCM internal registers can be accessed for configuration and control through the L4 interconnect. The control interfaces from the PRCM go to the five power domains in the device for clock, reset, power, and idle/wake-up management.

5.3.2 PRCM to Module Interface

Figure 5−6 shows details of the control interface to a generic power domain.

Figure 5−6. PRCM Internal Module Interfaces (Detailed View)



5.3.3 Reset and Power-Management Scheme

Table 5−3 lists the hardware reset domains for modules in the OMAP2420. Table 5−4 lists the power domains for the modules.

Table 5−3. Hardware Reset Domains

PRCM Subsystem	Reset Domain
Power manager – Clock manager	WKUP_RST
Clock generator	WKUP_PWRON
DPLL	DPLL_SYSRST
APLL 54	N/A (analog)
APLL 96	N/A (analog)
Oscillator	N/A (analog)

Table 5−4. Power Domains

PRCM Subsystem	Power Domain
Power manager – Clock manager	WKUP
Clock generator	WKUP
DPLL	WKUP
APLL 54	Analog
APLL 96	Analog
Oscillator	Analog

5.3.4 Hardware Requests

Table 5−5 lists the interrupts.

Table 5−5. Interrupts

Name	Mapping	Comments
PRCM_MPU_IRQ	M_IRQ_11	Destination is MPU subsystem interrupt controller (MPU INTC).
PRCM_DSP_IRQ	D_L2_IRQ_1	Destination is DSP subsystem interrupt controller (DSP INTC).

5.3.5 Pin List and Pad Multiplexing

In Table 5−6, gray indicates that the mode is used for the corresponding pin, black indicates that the mode is not used for the corresponding pin, and white indicates an alternate function.

Table 5−6. Pin Multiplexing for PRCM Module

Function n	**Description**	**DIR**	**Ball**	**Alternate Functions**					
	Interface			**Mode0**	**Mode1**	**Mode2**	**Mode3**	**Mode4**	**Mode5**
SYS.32K	32 kHz clock input	I	Y17	▒	■	■	■	■	■
SYS.XTALIN	Main input clock. Oscillator input or LVCMOS at 19.2, 13, or 12 MHz.	IO	Y18	▒	■	■	■	■	■
SYS.XTALOUT	Oscillator output	O	V16	▒	■	■	■	■	■
SYS.ALTCLK	Alternate clock source selectable for GPTIMERs (maximum 54 MHz), USB (48 MHz), or NTSC/PAL (54 MHz).	I	N18	hdq.sio	usb2.tllse0	▒	gpio.101	■	■
SYS.CLKREQ	Request from OMAP2420 for system clock	O	AA17	▒	■	■	gpio.52	■	■
SYS.CLKOUT	Configurable output clock	O	W14	▒	■	■	gpio.123	■	■
SYS.CLKOUT2	Configurable output clock 2	O	Y11	▒	usb2.dat	▒	gpio.16	■	■
SYS.nRESPWRON	Power-on reset	I	Y14	▒	■	■	■	■	■
SYS.nRESWARM	Warm boot reset (open drain output)	IO	Y16	▒	■	■	■	■	■
SYS.nVMODE	Indicates voltage mode	O	Y20	▒	■	■	■	■	■

5.3.6 L4 Interconnect Interface

The PRCM has an interface with the L4 interconnect through a dedicated target agent (TA), which is part of the L4 interconnect, and provides status and configuration registers as listed in Table 5−7. By default, TA register values are functional, but they can be overwritten. For a complete description, see Chapter 6, *Internal Interconnect*.

Table 5−7. L4 Interconnect Registers

Register Name	Type	Register Width (Bits)	Physical Address
COMPONENT	R	32	0x4800 9000
AGENT_CONTROL	RW	32	0x4800 9020
AGENT_STATUS	R	32	0x4800 9028

5.4 Clock Manager Functional Description

The PRCM provides efficient clock distribution with an aggressive idle scheme. This reduces dynamic consumption and offers simple, unified clock software control.

5.4.1 Clock Generation

5.4.1.1 External Clock Sources

The OMAP2420 provides the external clock inputs/outputs shown in Table 5−8.

Table 5−8. OMAP2420 External Clock Inputs/Outputs

Clock Name	Description
SYS.XTALIN	XTAL or square input
SYS.XTALOUT	XTAL output
SYS.CLKOUT	Output clock
SYS.CLKOUT2	Output clock 2
SYS.32K	32-kHz clock input
SYS.ALTCLK	48-MHz or other up to 54-MHz clock input

The OMAP2420 requires two input clocks:

- A 32-kHz clock, used for low-frequency operations. It supplies the WKUP power domain for operations in lowest power mode. This clock is provided through the SYS.32K pin.
- A system clock (12, 13, or 19.2 MHz), which is the main source clock of the chip. It supplies the PLLs and several OMAP2420 modules. Table 5−9 shows ways to connect the system clock input.

Table 5−9. System Clock Input Configurations

Input Source	Mapping	Comment
Quartz	SYS.XTALIN AND SYS.XTALOUT	Requires use of internal oscillator
Square clock (1.8 CMOS signal)	SYS.XTALIN (SYS.XTALOUT unconnected)	Oscillator bypassed

An alternate clock, provided through the SYS.ALTCLK pin, can be used to supply internal peripherals and PLLs.

Two output clock signals, SYS.CLKOUT and SYS.CLKOUT2, are available through the SYS.CLKOUT and SYS.CLKOUT2 pins and can provide the clock listed in the following table.

Clock Name	SYS_CLK[1]	CORE_CLK[1]	54-MHz Clock[1]	96-MHz Clock[1]
SYS.CLKOUT	x	x	x	x
SYS.CLKOUT2	x	x	x	x

1) Because of pad characteristics, the clocks are not available directly; they must be divided.

WARNING

Because of the SYS.CLKOUT pad characteristic, the 96-MHz clock cannot be directly provided and the user must divide it. For the same reason, SYS_CLK and CORE_CLK (depending on their values) must be divided before being used with the SYS.CLKOUT pad.

SYS.CLKOUT and SYS.CLKOUT2 can be divided by 2, 4, 8, or 16. Off-state polarity is programmable.

Note:

SYS_CLKOUT belongs to the WKUP power domain. SYS_CLKOUT2 belongs to the CORE power domain.

5.4.1.2 *Internal Clock Sources Definition*

The PRCM clock generator provides the following internal clock sources, from which all functional and interface clocks are generated and distributed:

- FUNC_32_CLK: 32-kHz clock (from external input clock)
- SYS_CLK: 12-, 13-, or 19.2-MHz system clock
- WDT1_SYS_CLK: 12-, 13-, or 19.2-MHz clock
- CORE_CLK: According to the DPLL control settings, this clock can be DPLL_CLKOUTx2, DPLL_CLKOUTx2 divided by two, or 32 kHz. A programmable divider with division factors of 1 or 2 allows quick frequency switching between DPLL_CLKOUTx2 divided by 1 and DPLL_CLKOUTx2 divided by 2. This mechanism is used to support voltage scaling.

CAUTION

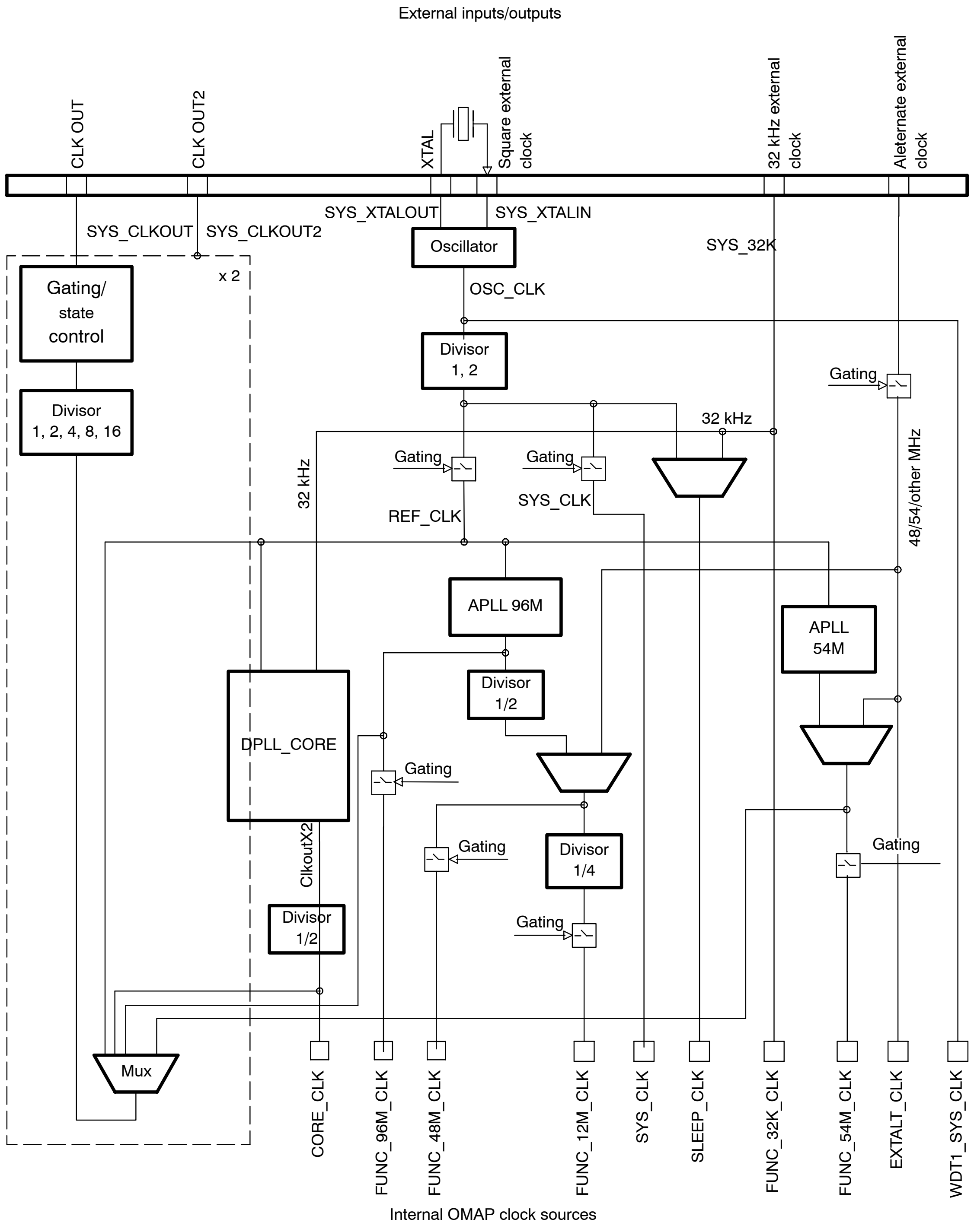
DPLL_CLKOUT x2 is the default configuration. After reset, CORE_CLK runs at a frequency of two times the value set for the DPLL. (See Section 5.9.13, *DPLL Programming*, for more information.)

- SLEEP_CLK: 32-kHz or 12-, 13-, or 19.2-MHz clock. Used only inside the PRCM to run the power-management logic. It runs on the system clock, unless the system clock source is turned off. Then it runs on the 32K input.
- FUNC_96M_CLK: 96-MHz clock (from APLL)
- FUNC_48M_CLK: 48-MHz clock (from APLL or external input clock)
- FUNC_12M_CLK: 12-MHz clock (from APLL or external input clock)
- FUNC_54M_CLK: 54-MHz clock (from APLL or external input clock), for TV out (OMAP2420 only)

- EXTALT_CLK: Any external input clock: 48, 54 MHz, or other (provided through SYS.ALTCLK). The maximum frequency is 54 MHz.

Figure 5−7 shows internal clock generation in relation to external clock sources.

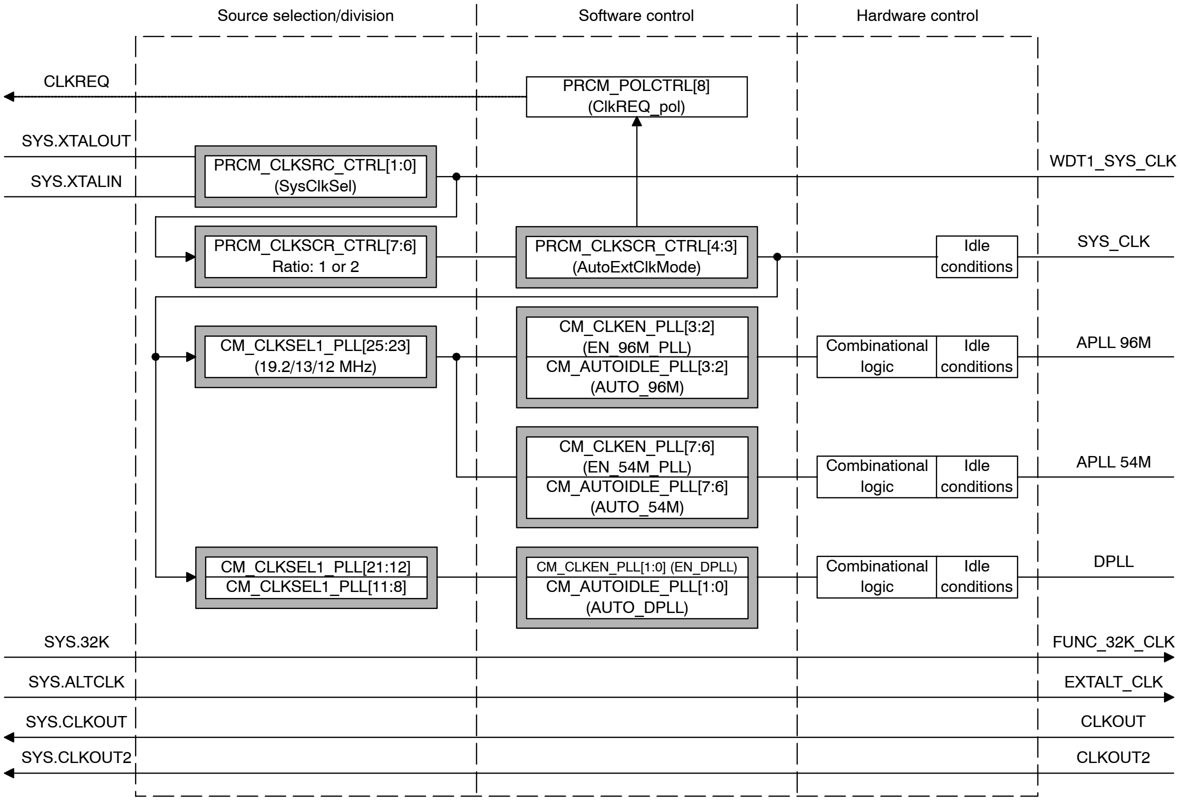
Figure 5−7. PRCM Clock Generator



5.4.1.3 Internal Clock Source Control

Control of internal clock sources depends primarily on PLL configuration and source selection. Figure 5−8 shows the first control level available, which corresponds to the external source selection and PLL settings.

Figure 5−8. Internal Clock Sources Control – Level 0



Notes:

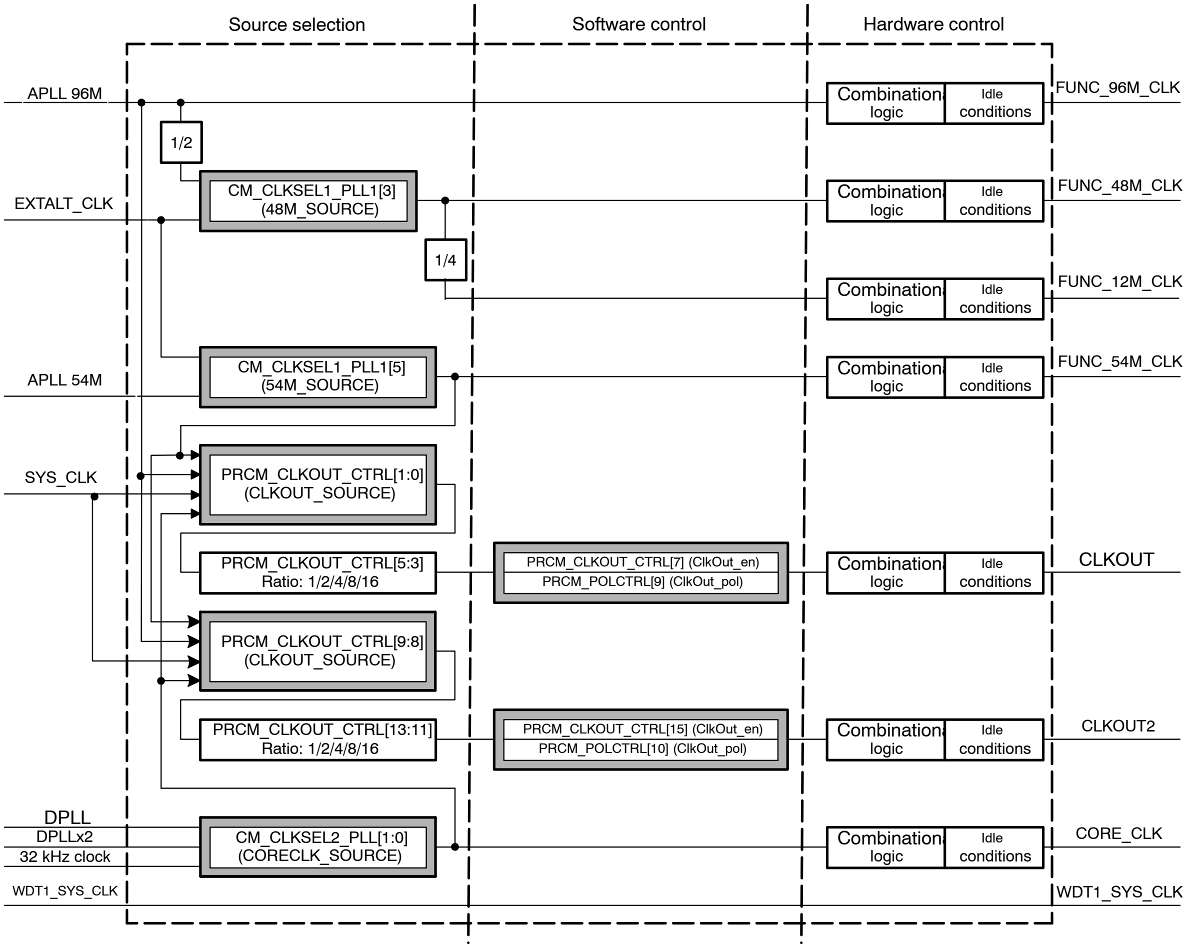
1) The available software control is shown with the PRCM register bit field that must be set to activate the functionality.
2) The Software Control column lists the PRCM bits used to enable and disable the clocks.
3) The *Hardware Control* column lists the cases in which additional conditions are required to effectively gate the clocks.
4) Boxes labeled Idle conditions indicate that the PRCM needs idle hardware acknowledge from the OMAP2420 modules before gating the clock (see Section 5.8, *Sleep and Wake-Up Processes*, for details).
5) Boxes labeled Combinational logic denote a logical combination between the programmed register fields.

Note:

Software does not control the FUNC_32K_CLK and EXTALT_CLK, each of which depends directly on the external sources connected to the inputs SYS.32K and SYS.ALTCLK.

Figure 5−9 shows how internal clock sources are generated from PLL output and/or external sources.

Figure 5−9. Internal Clock Sources Control – Level 1


Source selection
Software control
Hardware control
APLL 96M
1/2
CM_CLKSEL1_PLL1[3] (48M_SOURCE)
EXTALT_CLK
1/4
CM_CLKSEL1_PLL1[5] (54M_SOURCE)
APLL 54M
SYS_CLK
PRCM_CLKOUT_CTRL[1:0] (CLKOUT_SOURCE)
PRCM_CLKOUT_CTRL[5:3] Ratio: 1/2/4/8/16
PRCM_CLKOUT_CTRL[7] (ClkOut_en)
PRCM_POLCTRL[9] (ClkOut_pol)
PRCM_CLKOUT_CTRL[9:8] (CLKOUT_SOURCE)
PRCM_CLKOUT_CTRL[13:11] Ratio: 1/2/4/8/16
PRCM_CLKOUT_CTRL[15] (ClkOut_en)
PRCM_POLCTRL[10] (ClkOut_pol)
DPLL
DPLLx2
32 kHz clock
CM_CLKSEL2_PLL[1:0] (CORECLK_SOURCE)
WDT1_SYS_CLK
Combinational logic
Idle conditions
FUNC_96M_CLK
FUNC_48M_CLK
FUNC_12M_CLK
FUNC_54M_CLK
CLKOUT
CLKOUT2
CORE_CLK
WDT1_SYS_CLK

5.4.2 Clock Distribution

5.4.2.1 Clock Manager

The clock manager distributes the clock sources to all modules in the OMAP2420 and manages clock divisions, synchronization, and gating.

Clock Generation

The CORE_CLK signal is the source of all interface clocks and some functional clocks of the OMAP2420.

The clocks derived from CORE_CLK are fully balanced over the chip:

❑ CORE_CLK is provided to a single divider (14 ratios: 1, 2, 3, 4, 5, 6, 8, 9, 12, 16, 18, 24, 32, and 48). Each ratio outputs a clock. These outputs are gated when the division is not in use.

❑ For each module, submodule, or group of modules requiring an independent programmable clock, multiplexers allow selection between different ratios.

- A gate that can be controlled either automatically or by software follows each multiplexer.

FUNC_32_CLK, FUNC_96M_CLK, FUNC_48M_CLK, FUNC_12M_CLK, FUNC_54M_CLK, ALTEXT_CLK, and SYS_CLK clock signals are all unbalanced functional clocks used to generate functional clocks.

SYS_CLK supplies the GP timers.

Clock Gating

To reduce active current consumption, the clock manager features an automatic clock control, according to the following hierarchy:

- Clock-manager tree gating: A clock source shared by different modules is enabled when at least one module needs it and is disabled when all related modules do not require it.
- Power-domain clock tree gating: Ensures that clocks are not distributed to a shut down power domain
- Clock master/source gating: Clock sources are activated/stopped at different levels (DPLL, APLL, oscillator) according to the device modes.

This gating, managed by the PRCM, completes the local clock autogating managed by each module or subsystem.

Synchronizer Clocking

Synchronizers allow the MPU and DSP to run together at their maximum performance level, supporting the ratios 2/3 and 1/1.

The OMAP2420 uses a synchronizer implemented as follows:

- One synchronizer (2/3) between the DSP subsystem and the L3 interconnect

The PRCM clock manager controls the synchronizer by a fine management of phase signals and edge synchronization.

5.4.2.2 Clock Distribution

Internal clock sources generate the clock signals distributed in the OMAP2420. To reduce dynamic power consumption, the OMAP2420 is divided into six clock domains that the PRCM can control independently.

A clock domain includes several modules that must function at the same time. Each clock domain can contain several clocks.

- A power domain can contain one or more clock domains.
- A clock domain can belong to several power domains.

This division allows hierarchical power management that functions as follows:

- Local clock gating at the module or subsystem level
- Clock gating at the clock domain level (All modules that belong to this clock domain are idled and not clocked.)
- Power transition to either the retention or off state when all clock domains in a power domain are idled

The clock domains are defined as follows:

- L3 clock domain: CORE_L3_ICLK
- L4 clock domain: CORE_L4_ICLK, CORE_L4_USB_CLK, WU_L4_ICLK, FUNC_96M_CLK, FUNC_48M_CLK, FUNC_12M_CLK, WU_32K_CLK, CORE_32K_CLK, WDT1_SYS_CLK, SYNC_32K_CLK, WU_GPT1_CLK, and CORE_GPT[2..12]_CLK
- Display subsystem (DSS) clock domain: DSS_L4_ICLK, DSS_L3_ICLK, DSS_CLK1, DSS_CLK2, and DSS_54M_CLK
- MPU subsystem clock domain: MPU_FCLK, MPU_ICLK, INT_M_FCLK, and INT_M_ICLK
- DSP subsystem clock domain: DSP_FCLK, DSP_ICLK, INT_D_FCLK, and INT_D_ICLK

Clock domains are different from power domains. Figure 5−10 shows clock domain repartition in relation to the power domains.

Figure 5−10. Clock Domains Versus Power Domains

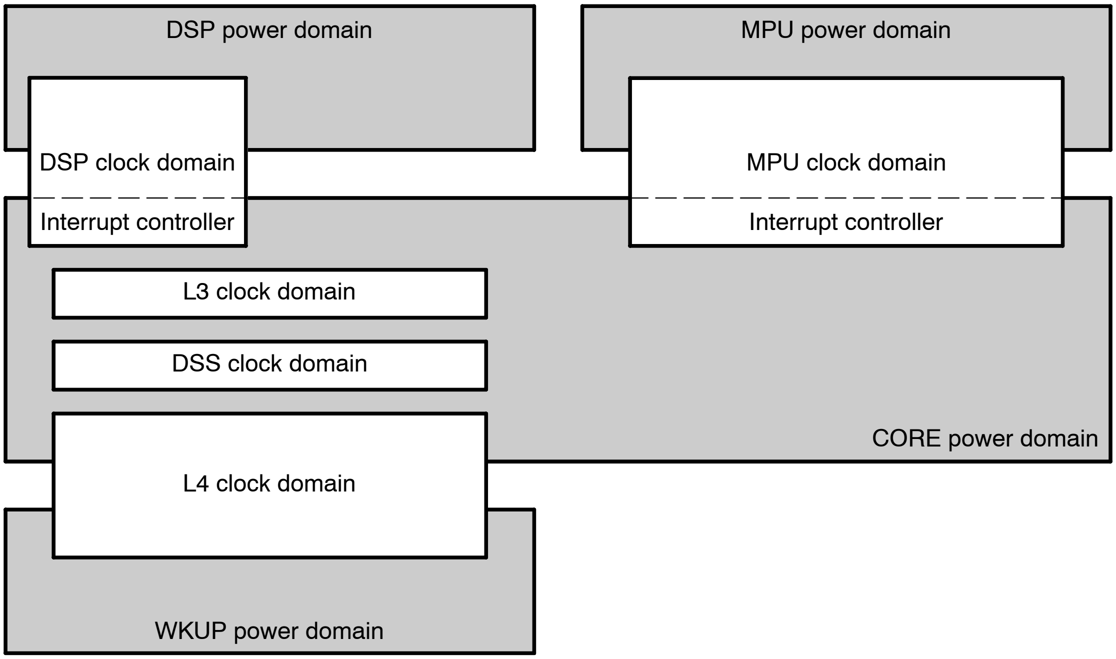


Figure 5−11 through Figure 5−13 show how PRCM clock outputs are distributed to the modules and subsystems.

Figure 5−11. PRCM Clock Signal Distribution – Interface Clocks

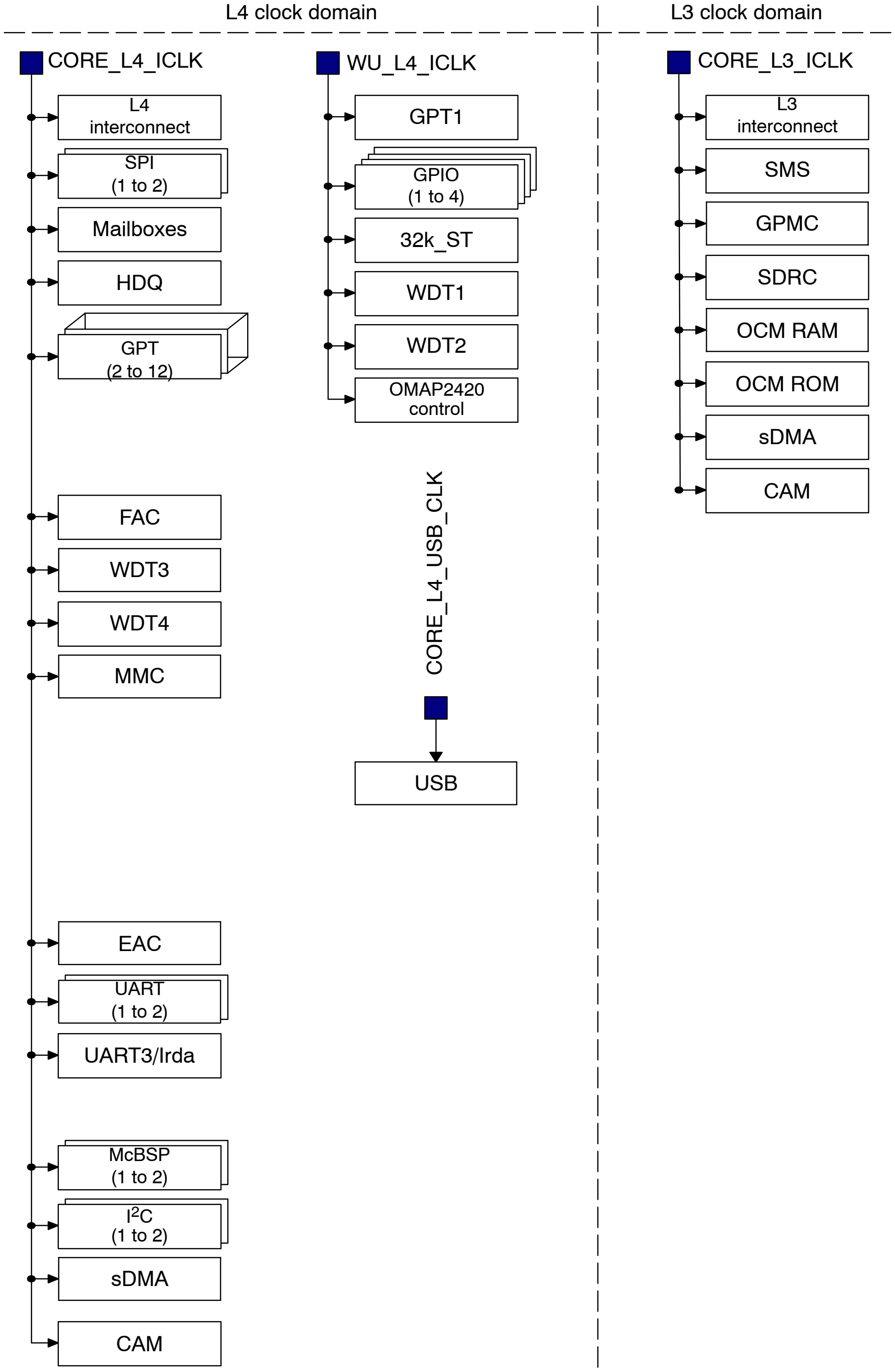


Figure 5−12. PRCM Clock Signal Distribution – Main Functional Clocks

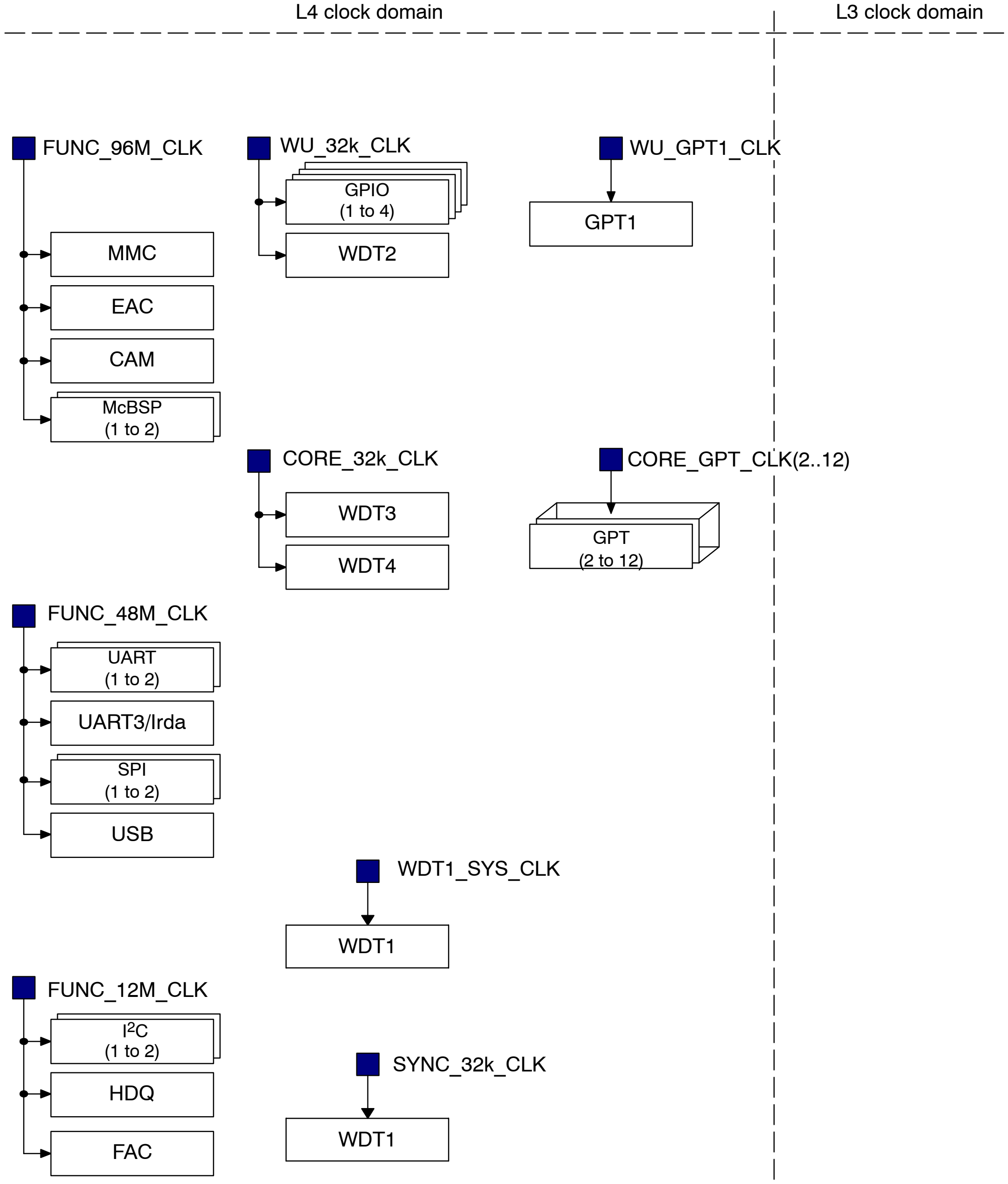
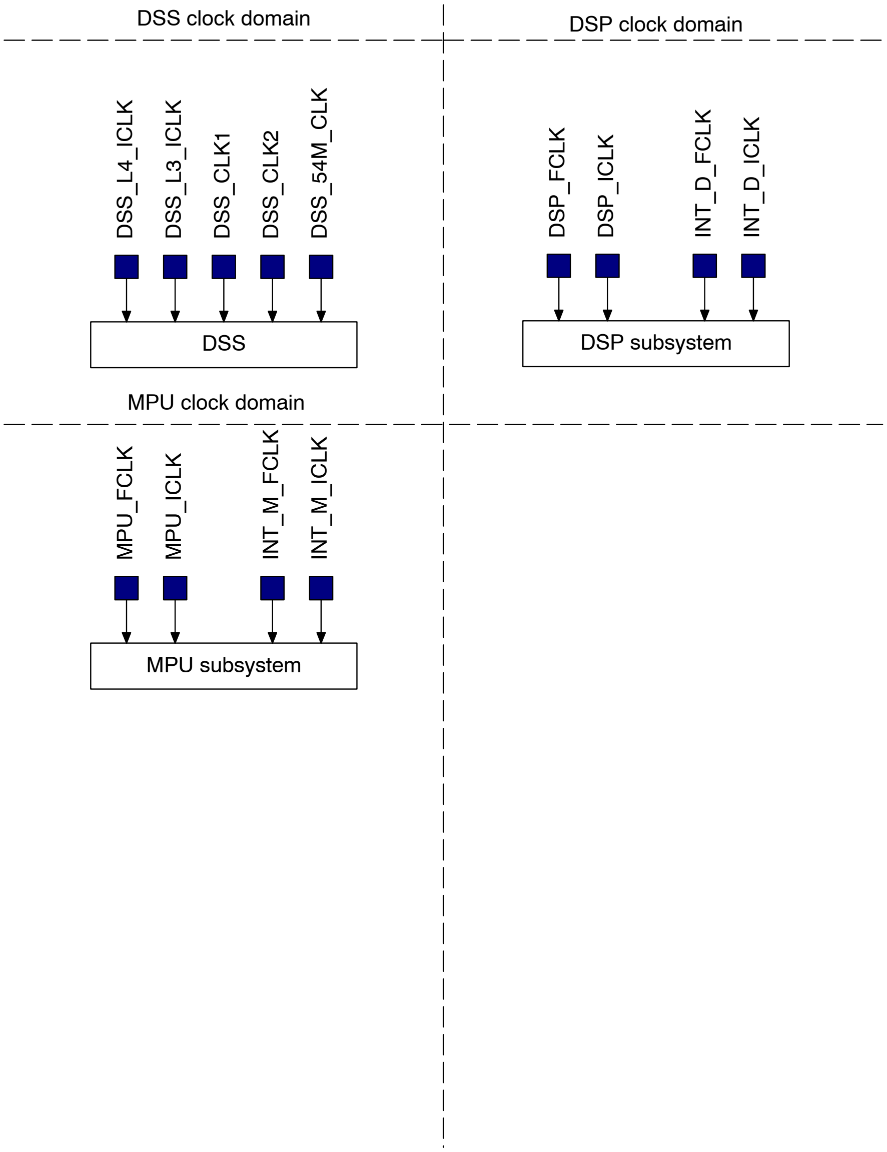


Figure 5−13. PRCM Clock Signal Distribution – Subsystem Clocks



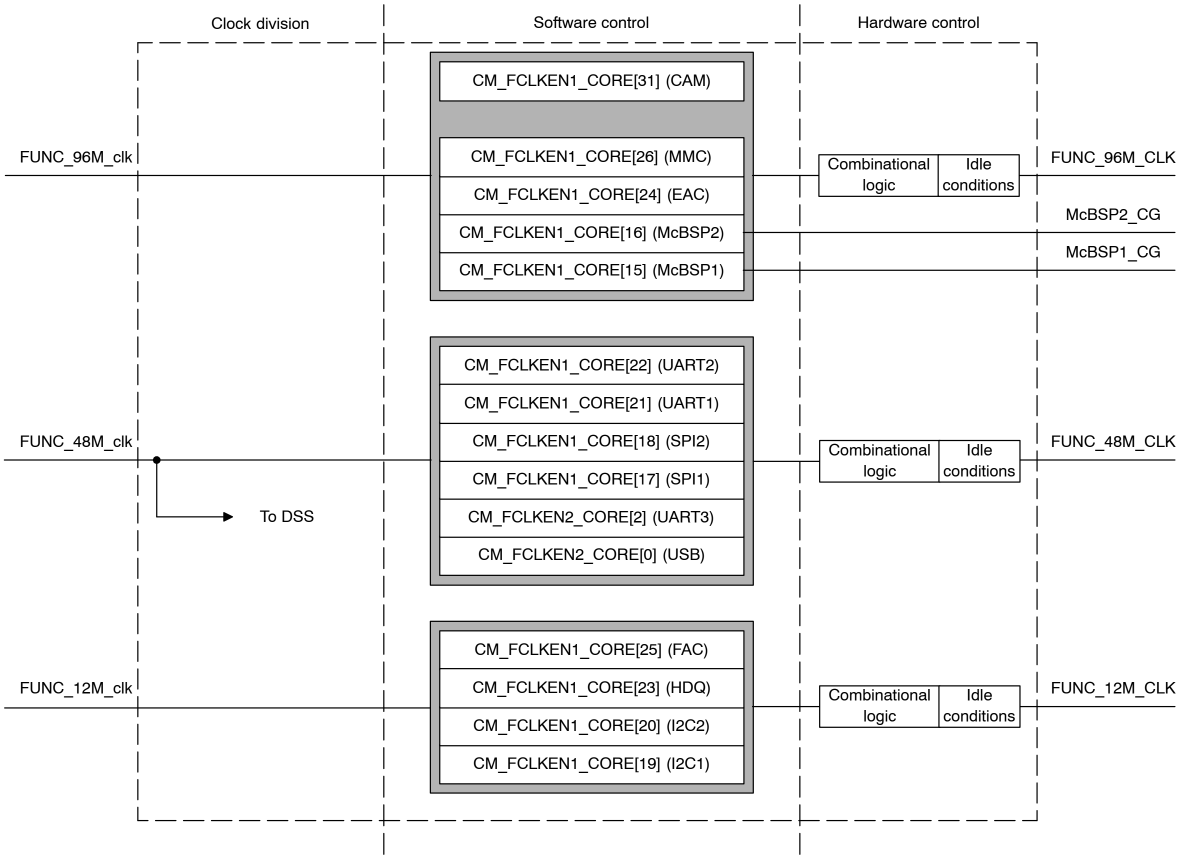
5.4.2.3 Clock Division, Distribution, and Control

This section describes how to control PRCM clock outputs. All clock signals are generated from the PRCM internal clock sources. You can use software to configure both internal sources and clock outputs.

The PRCM can cut a clock signal if all modules that use it when active do not need it (in idle or standby mode). The clock cannot be cut if at least one module or subsystem is using it.

Figure 5−14 through Figure 5−18 show the software control available for each clock output.

Figure 5−14. Clock Output Control – Functional Clocks 1


Clock division
Software control
Hardware control
FUNC_96M_clk
CM_FCLKEN1_CORE[31] (CAM)
CM_FCLKEN1_CORE[26] (MMC)
CM_FCLKEN1_CORE[24] (EAC)
CM_FCLKEN1_CORE[16] (McBSP2)
CM_FCLKEN1_CORE[15] (McBSP1)
Combinational logic
Idle conditions
FUNC_96M_CLK
McBSP2_CG
McBSP1_CG
FUNC_48M_clk
To DSS
CM_FCLKEN1_CORE[22] (UART2)
CM_FCLKEN1_CORE[21] (UART1)
CM_FCLKEN1_CORE[18] (SPI2)
CM_FCLKEN1_CORE[17] (SPI1)
CM_FCLKEN2_CORE[2] (UART3)
CM_FCLKEN2_CORE[0] (USB)
Combinational logic
Idle conditions
FUNC_48M_CLK
FUNC_12M_clk
CM_FCLKEN1_CORE[25] (FAC)
CM_FCLKEN1_CORE[23] (HDQ)
CM_FCLKEN1_CORE[20] (I2C2)
CM_FCLKEN1_CORE[19] (I2C1)
Combinational logic
Idle conditions
FUNC_12M_CLK

Notes:

1) The Software control column lists the PRCM register bits used to enable and disable the clocks.
2) The Hardware control column lists the cases in which gating the clocks effectively requires additional conditions.
3) Boxes labeled Idle conditions indicate that the PRCM needs an idle hardware acknowledge from the OMAP2420 modules before gating the clock (see Section 5.8, *Sleep and Wake-Up Processes*, for details).
4) Boxes labeled Combinational logic denote a logical combination between the programmed register fields.

   For example, FUNC_96M_CLK can be cut only if all related enable bits listed in the Software Control column are disabled (combinational logic conditions) and all related modules acknowledge the idle request (idle conditions).

Figure 5−15. Clock Output Control – Functional Clocks 2

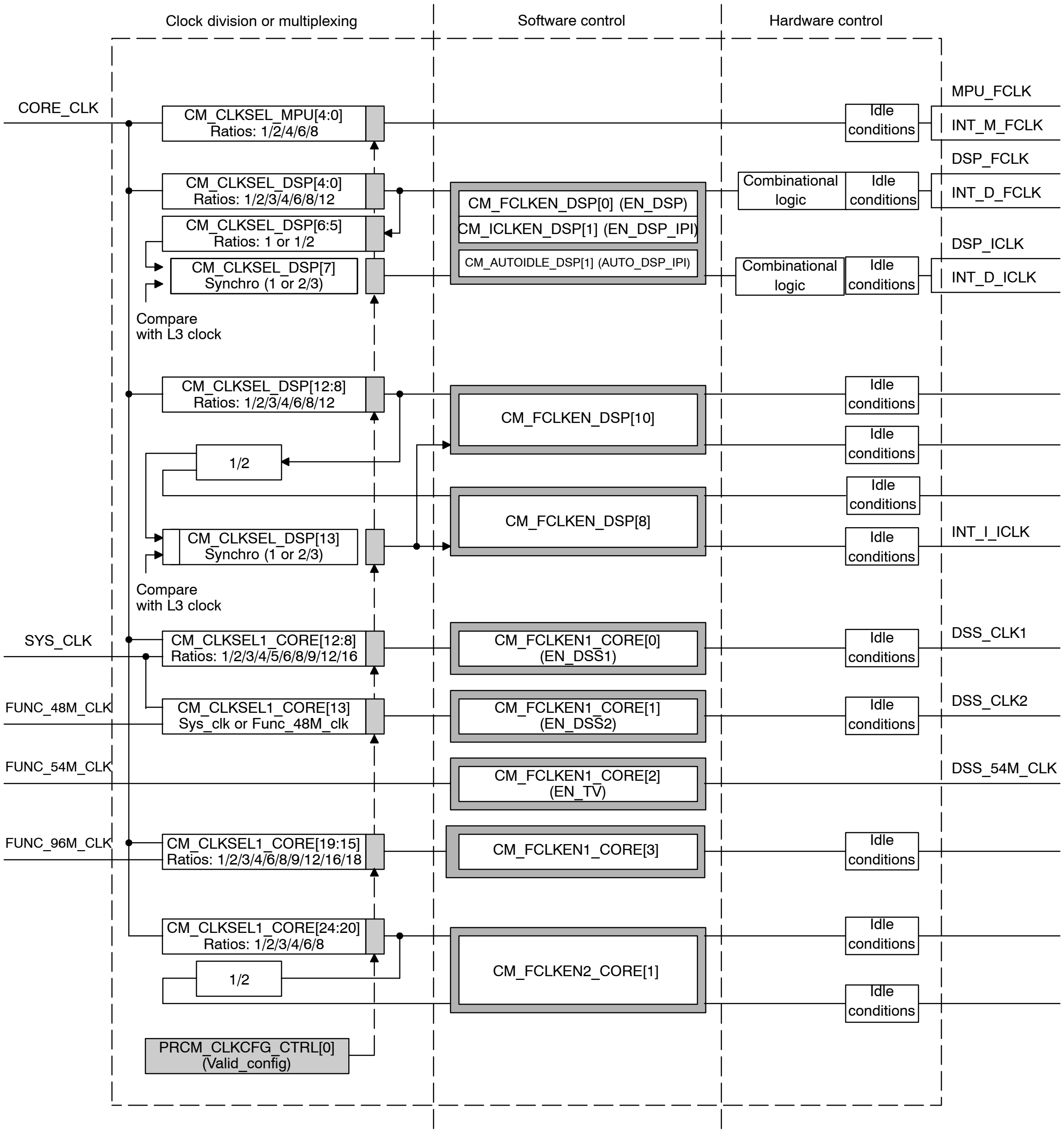


Figure 5−16. Clock Output Control – Functional Clocks 3

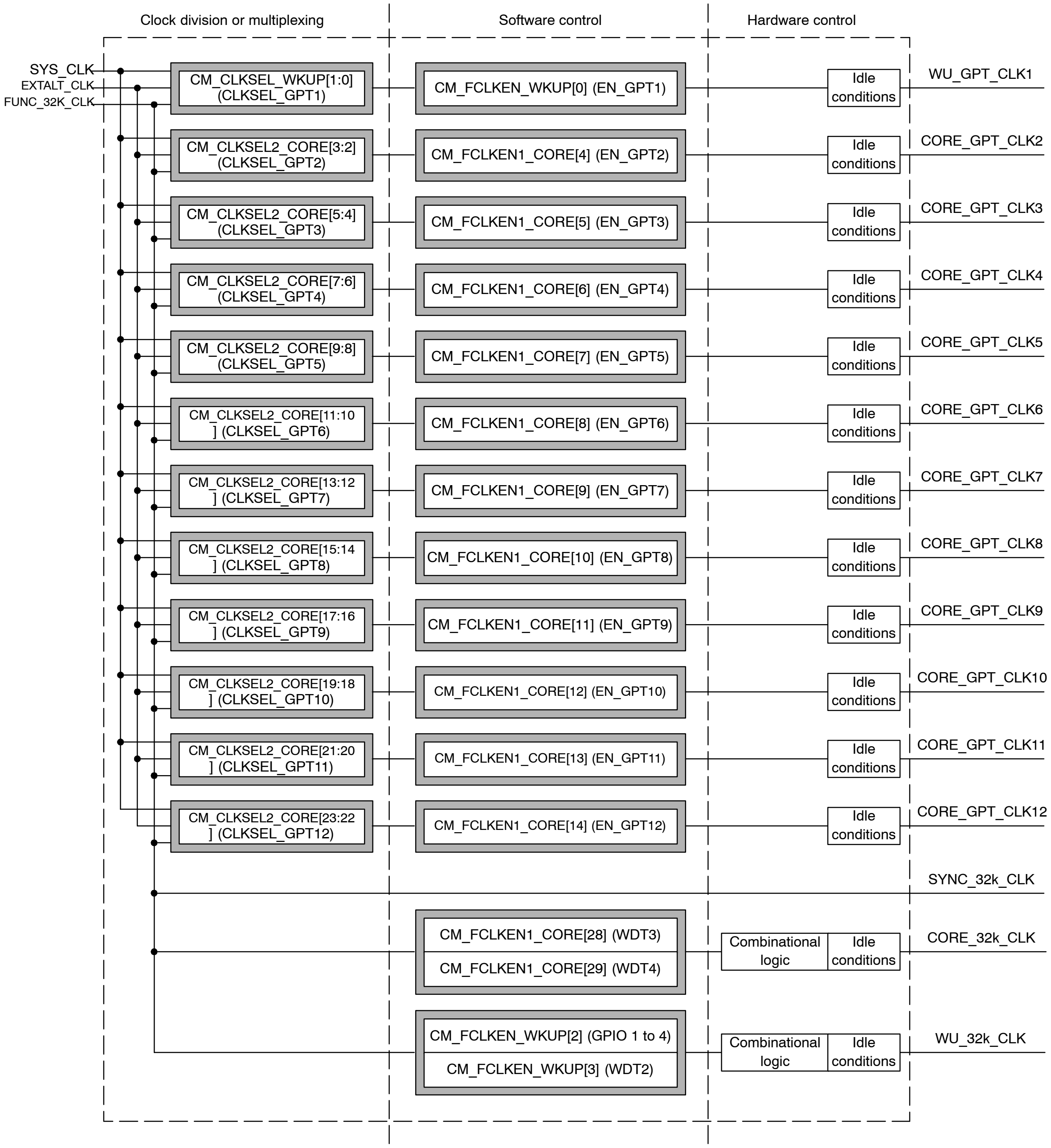

Clock division or multiplexing
Software control
Hardware control
SYS_CLK
EXTALT_CLK
FUNC_32K_CLK
CM_CLKSEL_WKUP[1:0] (CLKSEL_GPT1)
CM_CLKSEL2_CORE[3:2] (CLKSEL_GPT2)
CM_CLKSEL2_CORE[5:4] (CLKSEL_GPT3)
CM_CLKSEL2_CORE[7:6] (CLKSEL_GPT4)
CM_CLKSEL2_CORE[9:8] (CLKSEL_GPT5)
CM_CLKSEL2_CORE[11:10] (CLKSEL_GPT6)
CM_CLKSEL2_CORE[13:12] (CLKSEL_GPT7)
CM_CLKSEL2_CORE[15:14] (CLKSEL_GPT8)
CM_CLKSEL2_CORE[17:16] (CLKSEL_GPT9)
CM_CLKSEL2_CORE[19:18] (CLKSEL_GPT10)
CM_CLKSEL2_CORE[21:20] (CLKSEL_GPT11)
CM_CLKSEL2_CORE[23:22] (CLKSEL_GPT12)
CM_FCLKEN_WKUP[0] (EN_GPT1)
CM_FCLKEN1_CORE[4] (EN_GPT2)
CM_FCLKEN1_CORE[5] (EN_GPT3)
CM_FCLKEN1_CORE[6] (EN_GPT4)
CM_FCLKEN1_CORE[7] (EN_GPT5)
CM_FCLKEN1_CORE[8] (EN_GPT6)
CM_FCLKEN1_CORE[9] (EN_GPT7)
CM_FCLKEN1_CORE[10] (EN_GPT8)
CM_FCLKEN1_CORE[11] (EN_GPT9)
CM_FCLKEN1_CORE[12] (EN_GPT10)
CM_FCLKEN1_CORE[13] (EN_GPT11)
CM_FCLKEN1_CORE[14] (EN_GPT12)
CM_FCLKEN1_CORE[28] (WDT3)
CM_FCLKEN1_CORE[29] (WDT4)
CM_FCLKEN_WKUP[2] (GPIO 1 to 4)
CM_FCLKEN_WKUP[3] (WDT2)
Idle conditions
Combinational logic
WU_GPT_CLK1
CORE_GPT_CLK2
CORE_GPT_CLK3
CORE_GPT_CLK4
CORE_GPT_CLK5
CORE_GPT_CLK6
CORE_GPT_CLK7
CORE_GPT_CLK8
CORE_GPT_CLK9
CORE_GPT_CLK10
CORE_GPT_CLK11
CORE_GPT_CLK12
SYNC_32k_CLK
CORE_32k_CLK
WU_32k_CLK

Figure 5−17. Clock Output Control – Interface Clocks 1

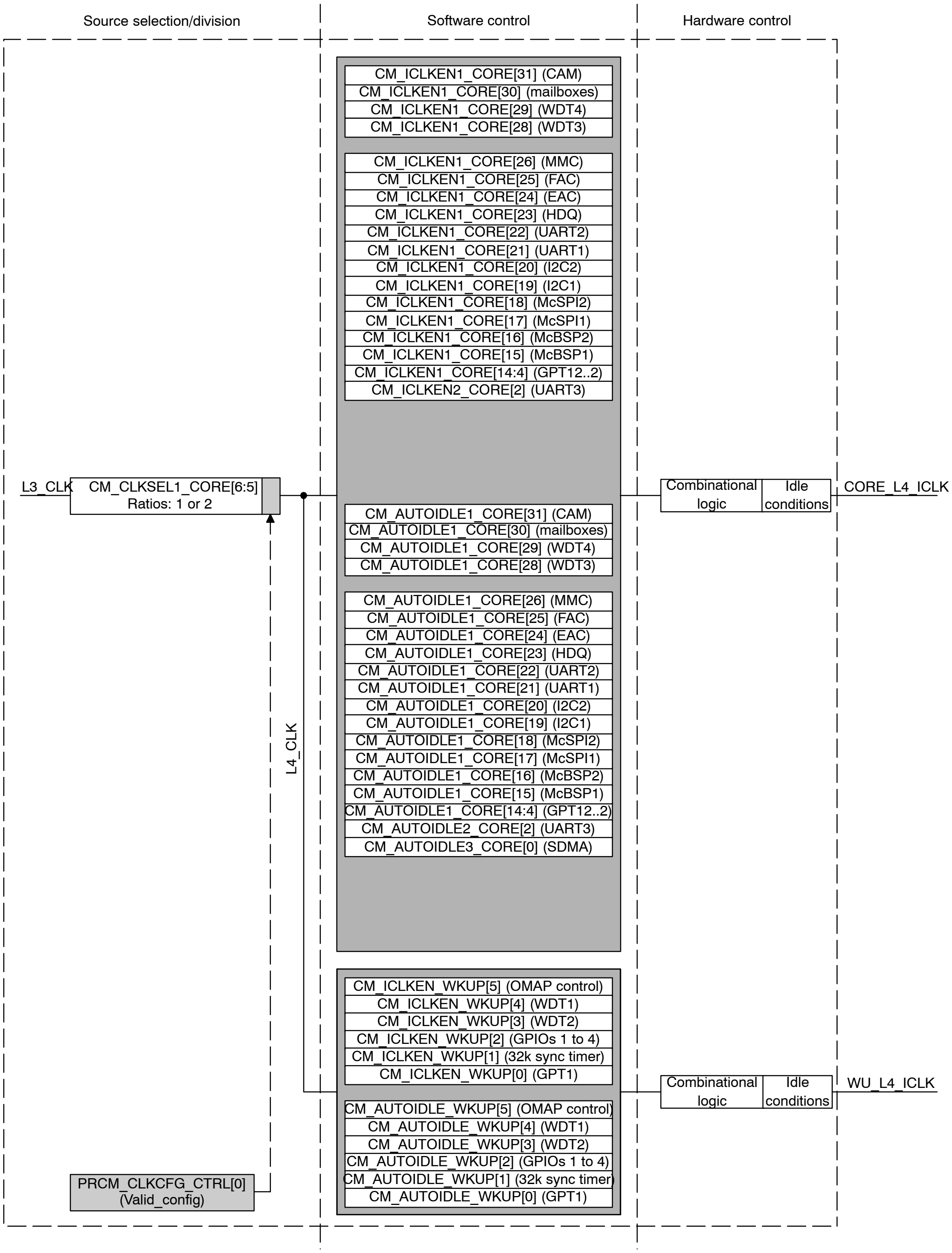
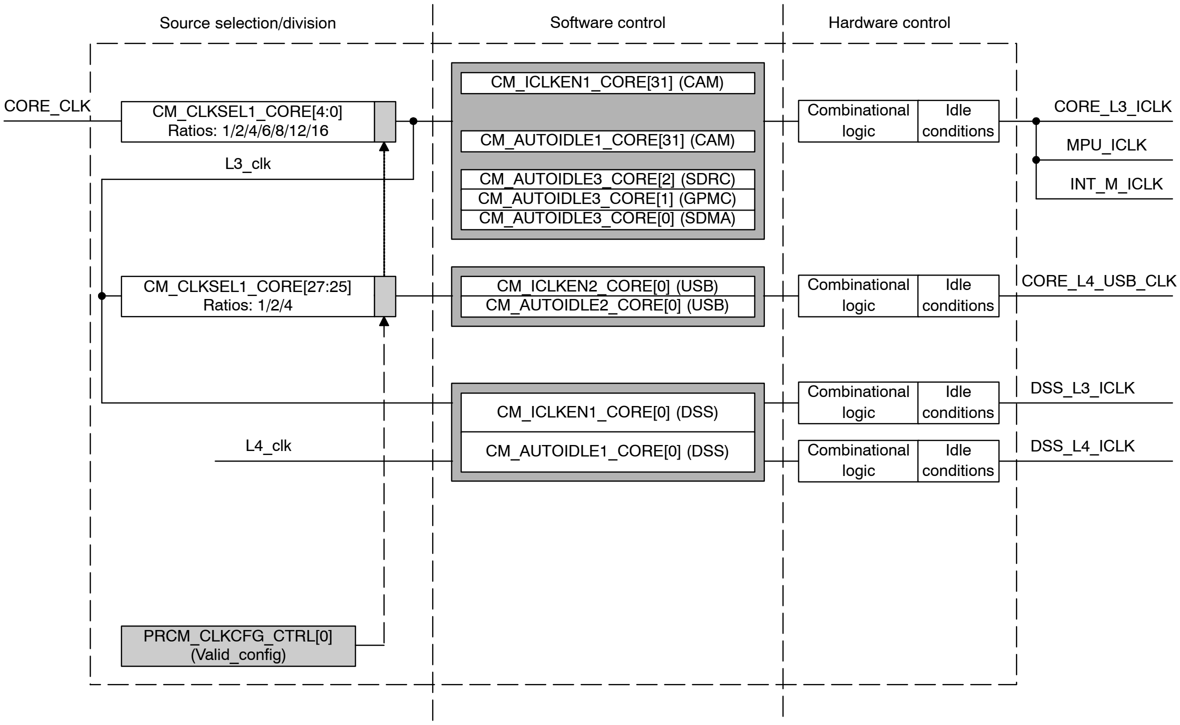

Source selection/division
Software control
Hardware control
CM_ICLKEN1_CORE[31] (CAM)
CM_ICLKEN1_CORE[30] (mailboxes)
CM_ICLKEN1_CORE[29] (WDT4)
CM_ICLKEN1_CORE[28] (WDT3)
CM_ICLKEN1_CORE[26] (MMC)
CM_ICLKEN1_CORE[25] (FAC)
CM_ICLKEN1_CORE[24] (EAC)
CM_ICLKEN1_CORE[23] (HDQ)
CM_ICLKEN1_CORE[22] (UART2)
CM_ICLKEN1_CORE[21] (UART1)
CM_ICLKEN1_CORE[20] (I2C2)
CM_ICLKEN1_CORE[19] (I2C1)
CM_ICLKEN1_CORE[18] (McSPI2)
CM_ICLKEN1_CORE[17] (McSPI1)
CM_ICLKEN1_CORE[16] (McBSP2)
CM_ICLKEN1_CORE[15] (McBSP1)
CM_ICLKEN1_CORE[14:4] (GPT12..2)
CM_ICLKEN2_CORE[2] (UART3)
L3_CLK
CM_CLKSEL1_CORE[6:5]
Ratios: 1 or 2
Combinational logic
Idle conditions
CORE_L4_ICLK
CM_AUTOIDLE1_CORE[31] (CAM)
CM_AUTOIDLE1_CORE[30] (mailboxes)
CM_AUTOIDLE1_CORE[29] (WDT4)
CM_AUTOIDLE1_CORE[28] (WDT3)
CM_AUTOIDLE1_CORE[26] (MMC)
CM_AUTOIDLE1_CORE[25] (FAC)
CM_AUTOIDLE1_CORE[24] (EAC)
CM_AUTOIDLE1_CORE[23] (HDQ)
CM_AUTOIDLE1_CORE[22] (UART2)
CM_AUTOIDLE1_CORE[21] (UART1)
CM_AUTOIDLE1_CORE[20] (I2C2)
CM_AUTOIDLE1_CORE[19] (I2C1)
CM_AUTOIDLE1_CORE[18] (McSPI2)
CM_AUTOIDLE1_CORE[17] (McSPI1)
CM_AUTOIDLE1_CORE[16] (McBSP2)
CM_AUTOIDLE1_CORE[15] (McBSP1)
CM_AUTOIDLE1_CORE[14:4] (GPT12..2)
CM_AUTOIDLE2_CORE[2] (UART3)
CM_AUTOIDLE3_CORE[0] (SDMA)
L4_CLK
CM_ICLKEN_WKUP[5] (OMAP control)
CM_ICLKEN_WKUP[4] (WDT1)
CM_ICLKEN_WKUP[3] (WDT2)
CM_ICLKEN_WKUP[2] (GPIOs 1 to 4)
CM_ICLKEN_WKUP[1] (32k sync timer)
CM_ICLKEN_WKUP[0] (GPT1)
Combinational logic
Idle conditions
WU_L4_ICLK
CM_AUTOIDLE_WKUP[5] (OMAP control)
CM_AUTOIDLE_WKUP[4] (WDT1)
CM_AUTOIDLE_WKUP[3] (WDT2)
CM_AUTOIDLE_WKUP[2] (GPIOs 1 to 4)
CM_AUTOIDLE_WKUP[1] (32k sync timer)
CM_AUTOIDLE_WKUP[0] (GPT1)
PRCM_CLKCFG_CTRL[0]
(Valid_config)

Figure 5−18. Clock Output Control – Interface Clocks 2



5.4.3 Clock Configurations

> **Clock configurations depend on core voltage and maximum clock frequencies. Values in this document may not be applicable to production devices.**
>
> CAUTION

5.4.3.1 Scalable Clocks

The OMAP2420 can support six clock configurations:

❑ Regular configurations

1) Optimized for 165-MHz SDRAM speed grade. The DPLL maximum output clock is 660 MHz, the core L3 interconnect clock is 165 MHz, and the MPU clock is 330 MHz.
2) Optimized for 100-MHz SDRAM speed grade. The DPLL maximum output clock is 600 MHz, the core L3 interconnect clock is 100 MHz, and the MPU clock is 300 MHz.
3) Optimized for 133-MHz SDRAM speed grade. The DPLL maximum output clock is 532 MHz, the core L3 interconnect clock is 133 MHz, and the MPU clock is 266 MHz.
4) 133-MHz-SDRAM speed grade and max MPU frequency. The DPLL maximum output clock is 660 MHz, the core L3 interconnect clock is 110 MHz, and the MPU clock is 330 MHz.

- Low-frequency and boot configurations

1) Boot clock. The DPLL clock, the core L3 interconnect clock, and the MPU clock frequency are a system clock frequency (12, 13 or 19.2 MHz).
2) 32-kHz frequency configuration. The DPLL clock is 32 kHz, the core L3 interconnect clock is 32 kHz, and the MPU clock is 32 kHz.

CAUTION

No other configuration is ensured to work. The software must set the correct clock configuration.

Table 5−10 through Table 5−14 give detailed views of the different configurations, and they deal with all modules supplied from the DPLL output. For each PRCM clock signal output, the frequency is indicated, as well as the ratio with regards to the DPLL frequency.

The following legend applies to the tables:

- A low-power mode is described for each of the six first configurations. Clock ratios are unchanged, but the DPLL output clock is set to DPLL_CLKOUTx2 divided by two. These low-power modes allow scaling down the chip voltage to low-voltage point, whereas high-voltage point is required in normal mode.
- A green box indicates that register programming is required, and provides the related fields. The Divider/Sync column provides the value to write in the registers.
- The MPU subsystem interrupt controller and DSP subsystem interrupt controller clocks are not listed. Their frequencies are automatically fixed when the related processor clocks are fixed.
- When a synchronizer is required between the interconnect and the DSP subsystem, the related column is marked activated.

CAUTION

The DPLL setting requires particular attention. The DPLL_MULT and DPLL_DIV bit fields (CM_CLKSEL1_PLL[21:12] and CM_CLKSEL1_PLL[12:8], respectively) allow selection of the DPLL frequency.

In the default configuration, the CORE_CLK frequency is the DPLL frequency x2. The low-power mode uses CORE_CLK in its DPLL frequency x1 configuration, which is set through the CORE_CLK_SRC bit field in the CM_CLKSEL2_PLL register (CM_CLKSEL2_PLL[1:0]). For DPLL programming details, see Section 5.9.13, *DPLL Programming*.

Table 5−10. Regular OMAP2420 Clock Configuration—SDRAM 165 MHz

			Configuration I SDRAM Speed Grade: 165 MHz			
			DSP synchronizer activated			
			Normal		**Low Power**	
Module	**Clocks–Sync**	**Control Bits**	**Divider/ Sync**	**Freq (MHz)**	**Divider/ Sync**	**Freq (MHz)**
DPLL	DPLL settings	CM_CLKSEL1_PLL[21:8]	M, N	330	M, N	330
	CORE_CLK	CM_CLKSEL2_PLL[1:0]	DPLL x2	660	DPLL	330
MPU subsystem	MPU_FCLK	CM_CLKSEL_MPU[4:0]	2	330	2	165
DSP subsystem	DSP_FCLK	CM_CLKSEL_DSP[4:0]	3	220	3	110
	DSP_ICLK	CM_CLKSEL_DSP[6:5]	6	110	6	55
	Synchronizer	CM_CLKSEL_DSP[7]	Activated	\	Activated	\
L3 interconnect, interface clocks	CORE_L3_ICLK	CM_CLKSEL1_CORE[4:0]	4	165	4	82.5
	MPU_ICLK					
	DSS_L3_CLK					
L4 interconnect, interface clocks	CORE_L4_ICLK	CM_CLKSEL1_CORE[6:5]	8	82.5	8	41.25
	WU_L4_ICLK					
	DSS_L4_ICLK					
	CORE_L4_USB_ CLK	CM_CLKSEL1_CORE [27:25]	16	41.25	16	20.63

Table 5−11. Regular OMAP2420 Clock Configuration–SDRAM 100 MHz

Configuration II
SDRAM Speed Grade: 100 MHz

All synchronizers bypassed

			Normal		Low Power	
Module	**Clocks–Sync**	**Control Bits**	**Divider/ Sync**	**Freq (MHz)**	**Divider/ Sync**	**Freq (MHz)**
DPLL	DPLL settings	CM_CLKSEL1_PLL[21:8]	M, N	300	M, N	300
	CORE_CLK	CM_CLKSEL2_PLL[1:0]	DPLL x2	600	DPLL	300
MPU subsystem	MPU_FCLK	CM_CLKSEL_MPU[4:0]	2	300	2	150
DSP subsystem	DSP_FCLK	CM_CLKSEL_DSP[4:0]	3	200	3	100
	DSP_ICLK	CM_CLKSEL_DSP[6:5]	6	100	6	50
	Synchronizer	CM_CLKSEL_DSP[7]	By-passed	\	By-passed	\
L3 interconnect, interface clocks	CORE_L3_ICLK	CM_CLKSEL1_CORE[4:0]	6	100	6	50
	MPU_ICLK					
	DSS_L3_CLK					
L4 interconnect, interface clocks	CORE_L4_ICLK	CM_CLKSEL1_CORE[6:5]	6	100	6	50
	WU_L4_ICLK					
	DSS_L4_ICLK					
	CORE_L4_USB_CLK	CM_CLKSEL1_CORE [27:25]	12	50	12	25

Table 5−12.Regular OMAP2420 Clock Configuration – SDRAM 133 MHz

Configuration III SDRAM Speed Grade: 133 MHz						
		DSP synchronizer activated				
			Normal		**Low Power**	
Module	**Clocks–Sync**	**Control Bits**	**Divider/ Sync**	**Freq (MHz)**	**Divider/ Sync**	**Freq (MHz)**
DPLL	DPLL settings	CM_CLKSEL1_PLL[21:8]	M, N	266	M, N	266
	CORE_CLK	CM_CLKSEL2_PLL[1:0]	DPLL x2	532	DPLL	266
MPU subsystem	MPU_FCLK	CM_CLKSEL_MPU[4:0]	2	266	2	133
DSP subsystem	DSP_FCLK	CM_CLKSEL_DSP[4:0]	3	177.3	3	88.67
	DSP_ICLK	CM_CLKSEL_DSP[6:5]	6	88.67	6	44.33
	Synchronizer	CM_CLKSEL_DSP[7]	Activated	\	Activated	\
L3 interconnect, interface clocks	CORE_L3_ICLK	CM_CLKSEL1_CORE[4:0]	4	133	4	66.5
	MPU_ICLK					
	DSS_L3_CLK					
L4 interconnect, interface clocks	CORE_L4_ICLK	CM_CLKSEL1_CORE[6:5]	8	66.5	8	33.25
	WU_L4_ICLK					
	DSS_L4_ICLK					
	CORE_L4_USB_CLK	CM_CLKSEL1_CORE [27:25]	16	33.25	16	16.63

Table 5−13. Regular OMAP2420 Clock Configuration —SDRAM 133 MHz (Max MPU)

Configuration IV
SDRAM Speed Grade: 133 MHz – Max MPU speed

DSP synchronizer bypassed

			Normal		**Low Power**	
Module	**Clocks–Sync**	**Control Bits**	**Divider/ Sync**	**Freq (MHz)**	**Divider/ Sync**	**Freq (MHz)**
DPLL	DPLL settings	CM_CLKSEL1_PLL[21:8]	M, N	330	M, N	330
	CORE_CLK	CM_CLKSEL2_PLL[1:0]	DPLL x2	660	DPLL	330
MPU subsystem	MPU_FCLK	CM_CLKSEL_MPU[4:0]	2	330	2	165
DSP subsystem	DSP_FCLK	CM_CLKSEL_DSP[4:0]	3	220	3	110
	DSP_ICLK	CM_CLKSEL_DSP[6:5]	6	110	6	55
	Synchronizer	CM_CLKSEL_DSP[7]	By-passed	\	By-passed	\
L3 interconnect, interface clocks	CORE_L3_ICLK	CM_CLKSEL1_CORE[4:0]	6	110	6	55
	MPU_ICLK					
	DSS_L3_CLK					
L4 interconnect, interface clocks	CORE_L4_ICLK	CM_CLKSEL1_CORE[6:5]	12	55	12	27.5
	WU_L4_ICLK					
	DSS_L4_ICLK					
	CORE_L4_USB_CLK	CM_CLKSEL1_CORE [27:25]	12	55	12	55

Table 5−14. Low-Frequency and Boot OMAP2420 Clock Configurations

Configuration VII and VIII Miscellaneous (Boot and 32 kHz)						
DSP synchronizer bypassed						
			VII		**VIII**	
Module	**Clocks–Sync**	**Control Bits**	**Divider/ Sync**	**Freq (MHz)**	**Divider/ Sync**	**Freq (kHz)**
DPLL	DPLL settings	CM_CLKSEL1_PLL[21:8]	M, N	19.2	M, N	32
	CORE_CLK	CM_CLKSEL2_PLL[1:0]	DPLL	19.2	DPLL	32
MPU subsystem	MPU_FCLK	CM_CLKSEL_MPU[4:0]	1	19.2	1	32
DSP subsystem	DSP_FCLK	CM_CLKSEL_DSP[4:0]	N/A	0	N/A	0
	DSP_ICLK	CM_CLKSEL_DSP[6:5]	N/A	0	N/A	0
	Synchronizer	CM_CLKSEL_DSP[7]	By-passed	\	By-passed	\
L3 interconnect, interface clocks	CORE_L3_ICLK	CM_CLKSEL1_CORE[4:0]	1	19.2	1	32
	MPU_ICLK					
	DSS_L3_CLK					
L4 interconnect, interface clocks	CORE_L4_ICLK	CM_CLKSEL1_CORE[6:5]	1	19.2	1	32
	WU_L4_ICLK					
	DSS_L4_ICLK					
	CORE_L4_USB_CLK	CM_CLKSEL1_CORE [27:25]	1	19.2	0	N/A

The DPLL clock frequency can be 12, 13, or 19.2 MHz in boot configuration (Configuration VII).

5.4.3.2 Peripheral Clocks

Table 5−15 and Table 5−16 list all peripheral clock requirements and top-level mapping.

All peripheral functional clocks remain unchanged when device voltage is scaled, because they are defined for low-voltage and high-voltage operations. Only the following modules have different behavior at low voltage:

- ❑ Camera subsystem: Module can support external CCP clock up to 104 MHz at low-voltage point (while it is up to 208 MHz at high-voltage point).
- ❑ Display subsystem: System can support CORE_CLK clock division on DSS_CLK1 up to 67.5 MHz at low-voltage point (while it is up to 133 MHz at high-voltage point). Frequency up to 48 MHz at low-voltage point on DSS_CLK2 is supported.
- ❑ 96-MHz APLL provides the 96-MHz, 48-MHz, and 12-MHz clocks.
- ❑ 54-MHz APLL provides the 54-MHz clock.
- ❑ 32-kHz clock is provided by external source.
- ❑ System clock: Provided by external source; can be 12 MHz, 13 MHz, or 19.2 MHz
- ❑ Nd (Display) is a CORE_CLK divider. When operating from CORE_CLK, the display subsystem is sensitive to voltage scaling (/ 2).
- ❑ Alternate external clock: Provided by an external source. It must not exceed 30 MHz for GP timers.

Table 5−15. OMAP2420 Peripheral Clock Requirements

		Functional Clock Requirements			
Module	**Domain**	**Clock 1**	**Clock 2**	**Clock 3**	**Clock 4**
MMC-SDIO	CORE	96 MHz			
EAC	CORE	96 MHz			
CAM	CORE	L3 clock	96 MHz		
McSPI 1	CORE	48 MHz			
McSPI 2	CORE	48 MHz			
UART 1	CORE	48 MHz			
UART 2	CORE	48 MHz			
UART 3 (IrDA)	CORE	48 MHz			
USB OTG	CORE	48 MHz			
Display subsystem	CORE	CORE_CLK/ Nd	48 MHz	System	54 MHz
I^{2}C 1	CORE	12 MHz			

Table 5−15. OMAP2420 Peripheral Clock Requirements (Continued)

		Functional Clock Requirements			
Module	**Domain**	**Clock 1**	**Clock 2**	**Clock 3**	**Clock 4**
I^{2}C 2	CORE	12 MHz			
HDQ	CORE	12 MHz			
FAC	CORE	12 MHz			
GPT 2	CORE	32 kHz	Alternate	System	
GPT 3	CORE	32 kHz	Alternate	System	
GPT 4	CORE	32 kHz	Alternate	System	
GPT 5	CORE	32 kHz	Alternate	System	
GPT 6	CORE	32 kHz	Alternate	System	
GPT 7	CORE	32 kHz	Alternate	System	
GPT 8	CORE	32 kHz	Alternate	System	
GPT 9	CORE	32 kHz	Alternate	System	
GPT 10	CORE	32 kHz	Alternate	System	
GPT 11	CORE	32 kHz	Alternate	System	
GPT 12	CORE	32 kHz	Alternate	System	
WDT 3	CORE	32 kHz			
WDT 4	CORE	32 kHz			
WDT 1	WKUP	System			
WDT 2	WKUP	32 kHz			
GPIO 1	WKUP	32 kHz			
GPIO 2	WKUP	32 kHz			
GPIO 3	WKUP	32 kHz			
GPIO 4	WKUP	32 kHz			
32K synchronous timer	WKUP	32 kHz			
GPT 1	WKUP	32 kHz	Alternate	System	
OMAP2420 control	WKUP	L4 clock			

Table 5−16. OMAP2420 Peripheral Clocks Mapping

Module	Domain	Functional Clock Mapping			Interface Clock Mapping
		Clock Input 1	**Clock Input 2**	**Clock Input 3**	
MMC-SDIO	CORE	FUNC_96MCLK			CORE_L4_ICLK
EAC	CORE	FUNC_96M_CLK			CORE_L4_ICLK
CAM	CORE	CORE_L3_ICLK	FUNC_96M_CLK		CORE_L3_ICLK
McSPI 1	CORE	FUNC_48M_CLK			CORE_L4_ICLK
McSPI 2	CORE	FUNC_48M_CLK			CORE_L4_ICLK
UART 1	CORE	FUNC_48M_CLK			CORE_L4_ICLK
UART 2	CORE	FUNC_48M_CLK			CORE_L4_ICLK
UART 3 (IrDA)	CORE	FUNC_48M_CLK			CORE_L4_ICLK
USB OTG	CORE	FUNC_48M_CLK			CORE_L4_USB_CLK
DSS	CORE	DSS_CLK1	DSS_CLK2	DSS_54M_CLK	DSS_L3_ICLK
I^2C 1	CORE	FUNC_12M_CLK			CORE_L4_ICLK
I^2C 2	CORE	FUNC_12M_CLK			CORE_L4_ICLK
HDQ	CORE	FUNC_12M_CLK			CORE_L4_ICLK
FAC	CORE	FUNC_12M_CLK			CORE_L4_ICLK
GPT 2	CORE	CORE_GPT2_CLK			CORE_L4_ICLK
GPT 3	CORE	CORE_GPT3_CLK			CORE_L4_ICLK
GPT 4	CORE	CORE_GPT4_CLK			CORE_L4_ICLK
GPT 5	CORE	CORE_GPT5_CLK			CORE_L4_ICLK
GPT 6	CORE	CORE_GPT6_CLK			CORE_L4_ICLK
GPT 7	CORE	CORE_GPT7_CLK			CORE_L4_ICLK
GPT 8	CORE	CORE_GPT8_CLK			CORE_L4_ICLK
GPT 9	CORE	CORE_GPT9_CLK			CORE_L4_ICLK
GPT 10	CORE	CORE_GPT10_CLK			CORE_L4_ICLK
GPT 11	CORE	CORE_GPT11_CLK			CORE_L4_ICLK
GPT 12	CORE	CORE_GPT12_CLK			CORE_L4_ICLK
WDT 3	CORE	CORE_32k_CLK			CORE_L4_ICLK
WDT 4	CORE	CORE_32k_CLK			CORE_L4_ICLK
WDT 1	WKUP	WDT1_sys_CLK			WU_L4_ICLK
WDT 2	WKUP	WU_32k_CLK			WU_L4_ICLK

Table 5−16. OMAP2420 Peripheral Clocks Mapping (Continued)

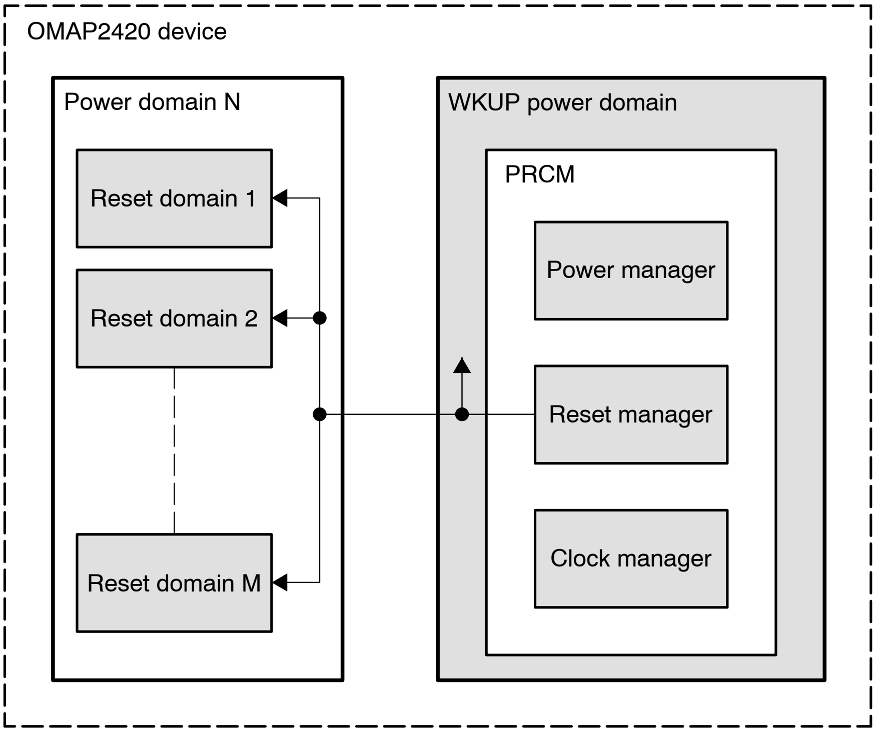
Module	Domain	Functional Clock Mapping			Interface Clock Mapping
		Clock Input 1	Clock Input 2	Clock Input 3	
GPIO 1	WKUP	WU_32k_CLK			WU_L4_ICLK
GPIO 2	WKUP	WU_32k_CLK			WU_L4_ICLK
GPIO 3	WKUP	WU_32k_CLK			WU_L4_ICLK
GPIO 4	WKUP	WU_32k_CLK			WU_L4_ICLK
32K synchronous timer	WKUP	Sync_32k_CLK			WU_L4_ICLK
GPT 1	WKUP	WU_GPT1_CLK			WU_L4_ICLK
OMAP2420 control	WKUP				WU_L4_ICLK

5.5 Reset Manager Functional Description

5.5.1 Reset Manager Overview

Figure 5−19 is an overview of the reset manager inside the PRCM.

Figure 5−19. Reset Manager Overview



The PRCM manages the reset of all modules in the OMAP2420. Resets are distributed throughout the five independent power domains. One or more reset domains are implemented inside each power domain.

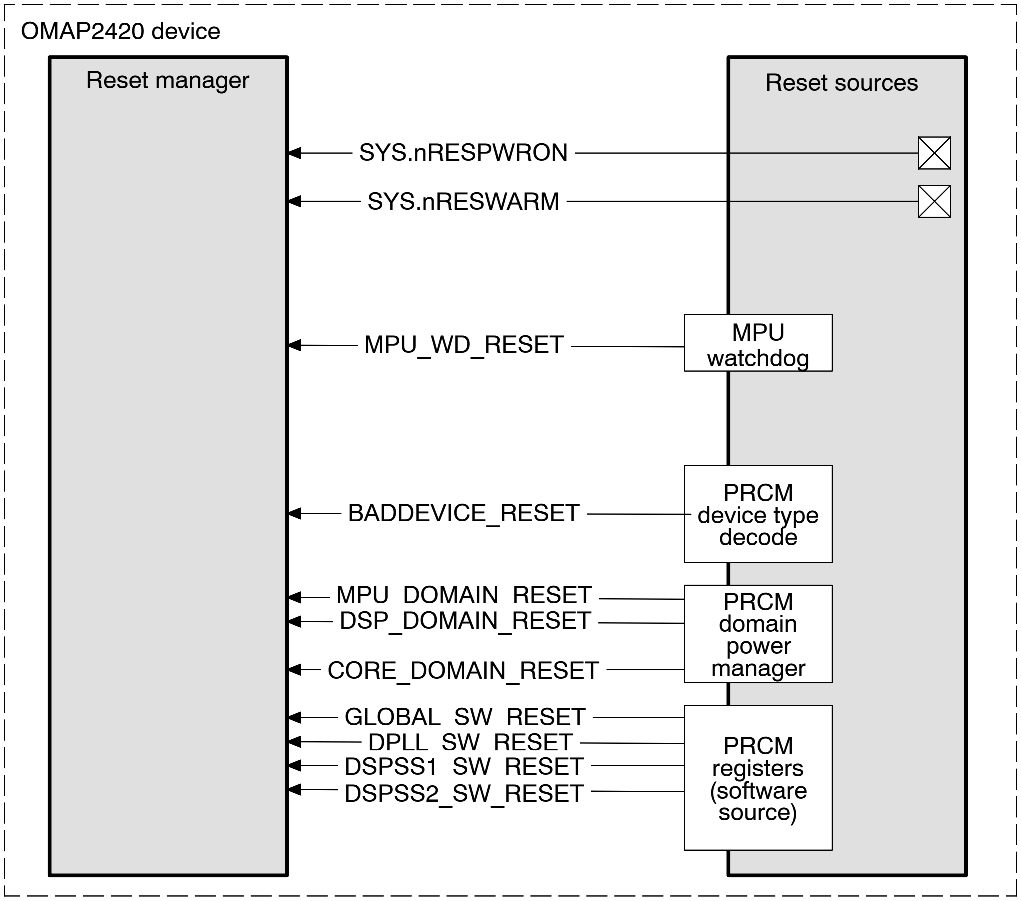
Each reset domain controls one or more modules in the power domain. Independent control of these reset domains allows the release of resets to be sequenced and ensures a safe reset of the entire power domain.

Resets are generated by several hardware sources or software control. Modules that can be reset by software have a soft reset bit implemented in the module SYSCONFIG configuration. Soft reset has exactly the same effect on module logic as has the hardware signal.

5.5.2 Reset Sources

Figure 5−20 shows an overview of the reset sources.

Figure 5−20. Reset Sources Overview



5.5.2.1 Definitions

Reset sources can be divided into two categories, depending on the scope of the reset response:

- Global reset: Affects the entire device
- Local reset: Affects a single power or reset domain

For a global reset, the PRCM executes the following sequence:

1) The PRCM applies a global reset to the device.

2) Most PRCM registers, including those that control power states, are set to a known state to ensure that all power domains are switched on.

3) Power domain reset is completed synchronously when clocks are activated.

Sources can also be categorized depending on when the reset occurs:

- Cold reset: Occurs only on device power-up
- Warm reset: Occurs when the device is in a normal operating state

Modules that behave differently in cold reset and warm reset also have a qualifier signal (POWER_ON) asserted on cold reset only. The reset and its qualifier reconstruct the cold reset and the warm reset signal inside modules that need it.

A global warm reset is applied to the entire device except for the following modules:

- SDRC
- 32-kHz synchronous timer
- OMAP2420 control
- DPLL
- Part of PRCM

5.5.2.2 Detailed Description

This section details the reset sources shown in Figure 5−20.

Global Reset Sources

Cold Reset:

- SYS.nRESPWRON signal (external input pin): The entire device is reset on power-up.

- BADDEVICE_RESET: If a device is identified as bad after reading eFuses, BADDEVICE_RESET is asserted continuously and applied to the entire device. A bad device determination is made if the device type does not match eFuse configuration.

Warm Reset:

- SYS.nRESWARM signal (external bidirectional pin): External hardware warm reset

- MPU_WD_RESET: MPU-watchdog overflow reset

- GLOBAL_SW_RESET: Global software reset

- DPLL_SW_RESET: DPLL software reset resets the DPLL and the device (acts as a global software reset, including the DPLL). Before applying this reset, the software must ensure that the SDRAM is set correctly in self-refresh mode.

Local Reset Sources

Cold Reset:

PM_DOMAIN_RESET (MPU, DSP, CORE): A power domain reset is asserted for any power domain transition from off to active state (or from dormant to active state for the MPU power domain).

Warm Reset:

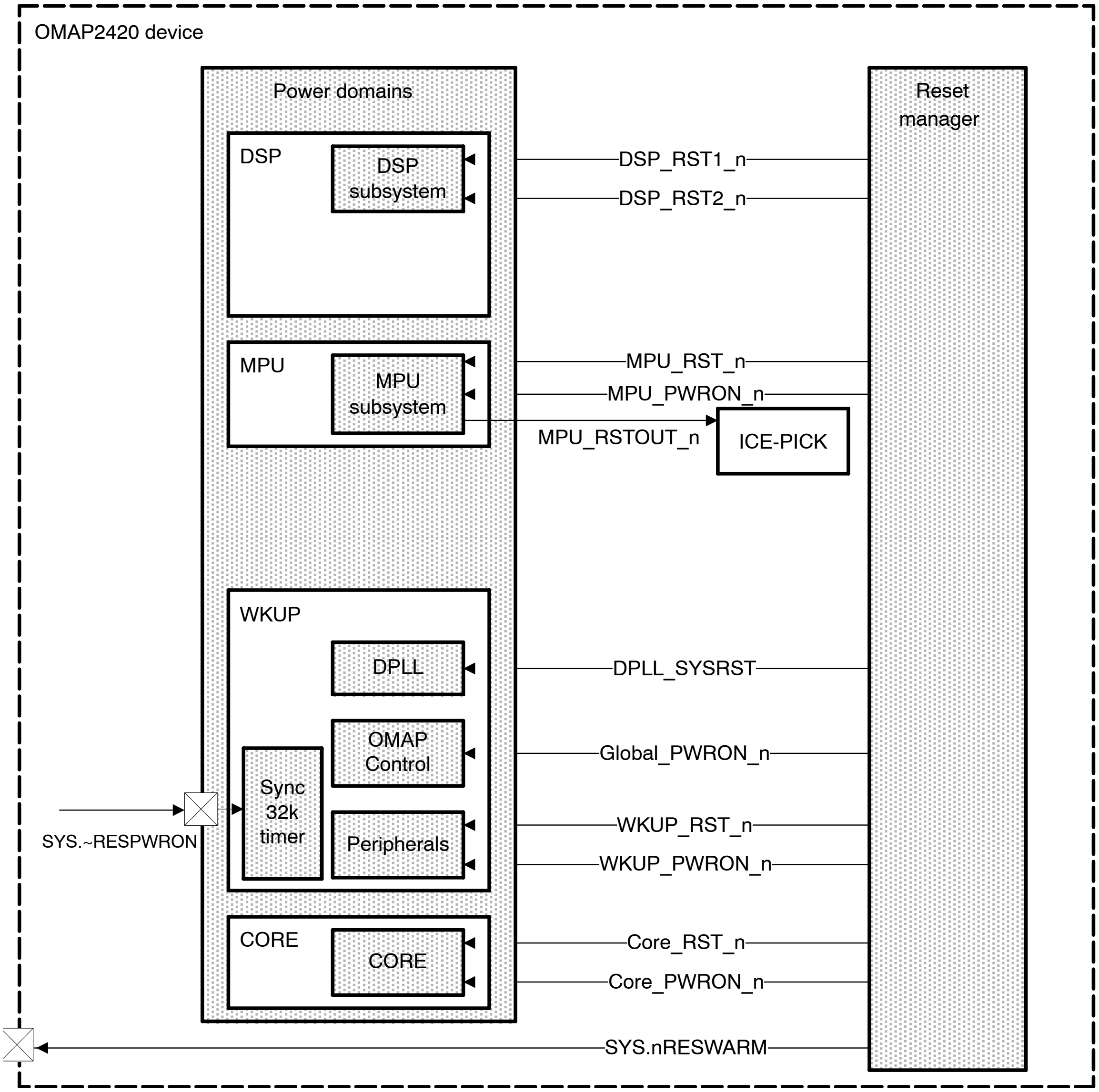
DOMAIN_SW_RESET: Inside a power domain, a module or group of modules can have an independent software reset signal.

5.5.3 Reset Distribution

Reset distribution is organized by power domain. Each power domain can have one or more reset domains controlled by internal reset signals. The reset signals are asserted for both cold and warm resets. A qualifying power-on signal is asserted only during cold reset.

Figure 5−21 is an overview of the reset distribution.

Figure 5−21. Reset Distribution Overview



5.5.3.1 MPU Power Domain

The MPU power domain contains the MPU subsystem and has one reset input and one power-on qualifier:

- MPU_RST_N resets the MPU.
- MPU_PWRON_N, active on device power-up only (cold reset), resets the ICE-crusher and the MPU debug port.

The MPU power domain also provides a reset output, MPU_RSTOUT_N, that is used by the ICE-crusher to reset the entire device (global warm reset).

5.5.3.2 CORE Power Domain

The CORE power domain uses one reset signal and one power-on qualifier signal:

- CORE_RST_N
- CORE_PWRON_N

The CORE_RST_N signal resets the following modules:

- SDMA
- SMS
- GPMC
- OCM ROM
- OCM RAM
- L3 interconnect
- L4 interconnect
- Interrupt controllers (MPU and DSP)
- Display subsystem
- Mailboxes
- CAM
- Peripherals: GPTs (2 to 12), McBSP1, McBSP2, MS-Pro, MMC, EAC, UART1, UART2, UART3 (IrDA), McSPI1, McSPI2, MSI2C1, MSI2C2, HDQ, FAC, USB OTG, WDT 3, and WDT 4

All modules can be independently reset by software using the SYSCONFIG register for each module.

The SDRAM controller is reset by the combined assertion of the CORE_RST_N signal and CORE_PWRON_N qualifier. According to the SRFRONRESET configuration bit, the SDRC behaves differently on cold and warm reset sources. For warm reset, SRFRONRESET, if active, places the attached SDRAM memories in self-refresh and does not execute a global reset. Software can independently reset the module using the SYCONFIG register.

5.5.3.3 DSP Power Domain

The DSP power domain contains the DSP subsystem.

The DSP subsystem has two reset inputs:

- DSP_RST1_N resets the DSP core and the DMA controller
- DSP_RST2_N resets the MMU, the LUT and the interconnect

Their release is controlled by software (RM_RSTCTRL_DSP register). The DSP DMA controller can also be independently reset by software using the SYSCONFIG register.

5.5.3.4 WKUP Power Domain

The WKUP power domain uses two reset signals and two power-on qualifier signals:

- WKUP_RST_N
- DPLL_SYSRST
- GLOBAL_PWRON_N
- WKUP_PWRON_N

The WKUP_RST_N signal resets the following modules:

- GPT1
- Watchdog timer 2
- GPIOs

The modules can also be independently reset by using each module's respective SYSCONFIG register.

The DPLL_SYSRST signal resets the DPLL module at power-up. In addition, the DPLL module can be independently reset by software (using the RST_DPLL bit in the RM_RSTCTRL_WKUP register). For safety, one device global software reset is generated simultaneously with a software reset of the DPLL.

The GLOBAL_PWRON_N signal resets the OMAP2420 at power-up only. Release occurs before the WKUP_PWRON_N signal.

For synchronization at the system level, the 32-kHz synchronization timer is reset directly by the SYS.nRESPWRON signal.

All or part of the PRCM (logic and registers) is reset according to the reset source (cold or warm).

Note:

The APLLs and oscillator are analog cells and, therefore, are not reset.

5.5.3.5 External Warm Reset Assertion

Any input pulse on SYS.nRESWARM causes a global warm reset.

Any global warm-reset source (including an external source) causes the SYS.nRESWARM reset signal to be driven out and maintained for as duration of at least [RSTIME1 + RSTIME2] (see Section 5.9, *Programming Model*, for details). This ensures that the OMAP2420 and its peripherals are correctly reset together.

5.5.4 Reset Summary

Table 5−17 summarizes reset sources for the power domains.

Table 5−17. Reset Summary

	Reset Sources										
Power Domain and Reset Signal	**Global Power-on Reset**		**Global Warm Reset**			**Local Cold Reset**			**Local Software Resets**		
	SYS_nRESPWRON	BADDEVICE_RESET	SYS_nRESWARM	MPU_WD_RESET	GLOBAL_SW_RESET	MPU_DOMAIN_RESET	DSP_DOMAIN_RESET	CORE_DOMAIN_RESET	DPSS_SW_RESET1	DPSS_SW_RESET2	DPLL_SW_RESET
MPU											
MPU_RST_N	●	●	●	●	●	●		●			●
MPU_PWRON_N	●	●				●		●			
MPU_RSTOUT											
DSP											
DSP_RST1_N	●	●	●	●	●		●	●	●		●
DSP_RST2_N	●	●	●	●	●		●	●		●	●
CORE											
CORE_RST_N	●	●	●	●	●			●			●
CORE_PWRON_N	●	●						●			

Table 5−17. Reset Summary (Continued)

	Reset Sources										
Power Domain and Reset Signal	**Global Power-on Reset**		**Global Warm Reset**			**Local Cold Reset**			**Local Software Resets**		
	SYS_nRESPWRON	BADDEVICE_RESET	SYS_nRESWARM	MPU_WD_RESET	GLOBAL_SW_RESET	MPU_DOMAIN_RESET	DSP_DOMAIN_RESET	CORE_DOMAIN_RESET	DPSS_SW_RESET1	DPSS_SW_RESET2	DPLL_SW_RESET
WKUP											
WKUP_RST_N	•	•	•	•	•						•
WKUP_PWRON_N	•	•									
GLOBAL_PWRON_N	•	•									
DPLL_SYS_RST	•	•									•
OMAP pad											
SYS_nRESWARM	•	•	•	•	•						•

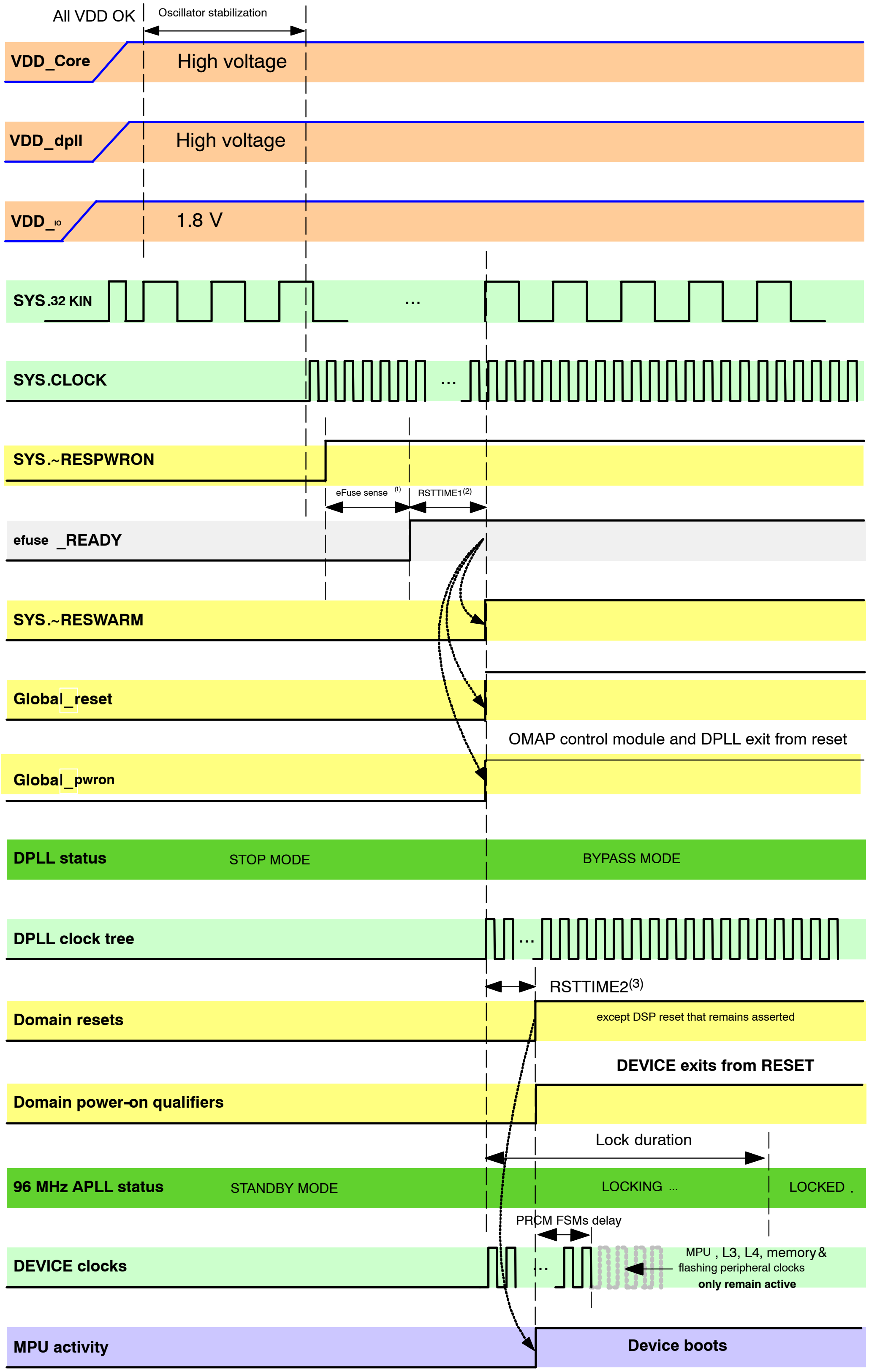
5.5.5 Reset Sequences

This section describes the reset process for the three types of reset:

- ❑ Power-on reset
- ❑ Global warm reset
- ❑ Power domain wake-up

5.5.5.1 Power-On Reset Sequences

Figure 5−22. Power-Up Reset Sequence



Notes:
1) eFuse sense duration is 83 µs (2.7 x 32-kHz clock cycles) when the system clock is running at 19.2 MHz and 134 µs (4.2 x 32-kHz clock cycles) when the system clock is running at 12 MHz.
2) RSSTIME1 maximum duration is the default value programmed in the RM_RSTTIME_WKUP register + 1 (3 x 32-kHz clock cycles).
3) RSSTIME2 maximum duration is the default value programmed in the RM_RSTTIME_WKUP register + 1 (17 x L4 clock cycles = 17 x system clock cycles at power up).

The power-up sequence behaves as follows:

1) While SYS.nRESPWRON is asserted:
   a) Driven as an output, SYS.nRESPWARM is asserted low.
   b) All domain resets are asserted.
   c) The oscillator is enabled.
   d) SYS_CLK is ON.
   e) The APLL 96-MHz is stopped (FUNC_96M_CLK, FUNC_48M_CLK, FUNC_12M_CLK, and FUNC_54M_CLK are off).
   f) The APLL 54 MHz is stopped.
   g) The DPLL is in off mode (CORE_CLK is off).
   h) All clocks derived from CORE_CLK are off.
2) SYS.nRESPWRON conditions of deassertion:
   a) V_{DD} core must be in regulation at high-voltage point.
   b) V_{DD} I/O must be in regulation at 1.8 V.
   c) Oscillator output clock must be stable.
   d) 32-kHz clock must be stable.
3) On deassertion of SYS.nRESPWRON:
   a) All domain resets are kept asserted.
   b) The oscillator is kept enabled.
   c) SYS_CLK is on.
   d) The APLL 96-MHz is stopped (FUNC_96M_CLK, FUNC_48M_CLK, FUNC_12M_CLK, and FUNC_54M_CLK are off).
   e) The APLL 54 MHz is stopped.
   f) The DPLL is in off mode (CORE_CLK is off).
4) When the eFuses sense is complete:
   a) Driven as an output, SYS.nRESWARM is released.
   b) All domain resets are kept asserted and all domain reset managers start the reset count RSTTIME2.
   c) The oscillator is kept enabled.
   d) SYS_CLK is ON.
   e) The APLL 96-MHz starts locking (used for UART3 and USB modules) based on the default input clock reference (19.2 MHz).

   f) The APLL 54 MHz is stopped.

   g) The DPLL is in bypass mode (CORE_CLK is running at SYS_CLK frequency).

   h) All clocks run at SYS_CLK frequency (dividers are 1 by default).

5) When all domain reset-manager reset counts expire:

   a) All domain resets are deasserted, except DSP resets.

   b) The oscillator is kept enabled.

   c) SYS_CLK is on.

   d) The APLL 96-MHz is on.

   e) The APLL 54 MHz is stopped.

   f) The DPLL is in bypass mode (CORE_CLK is running at SYS_CLK frequency).

   g) Clocks are stopped or kept running according to the PRCM setting:

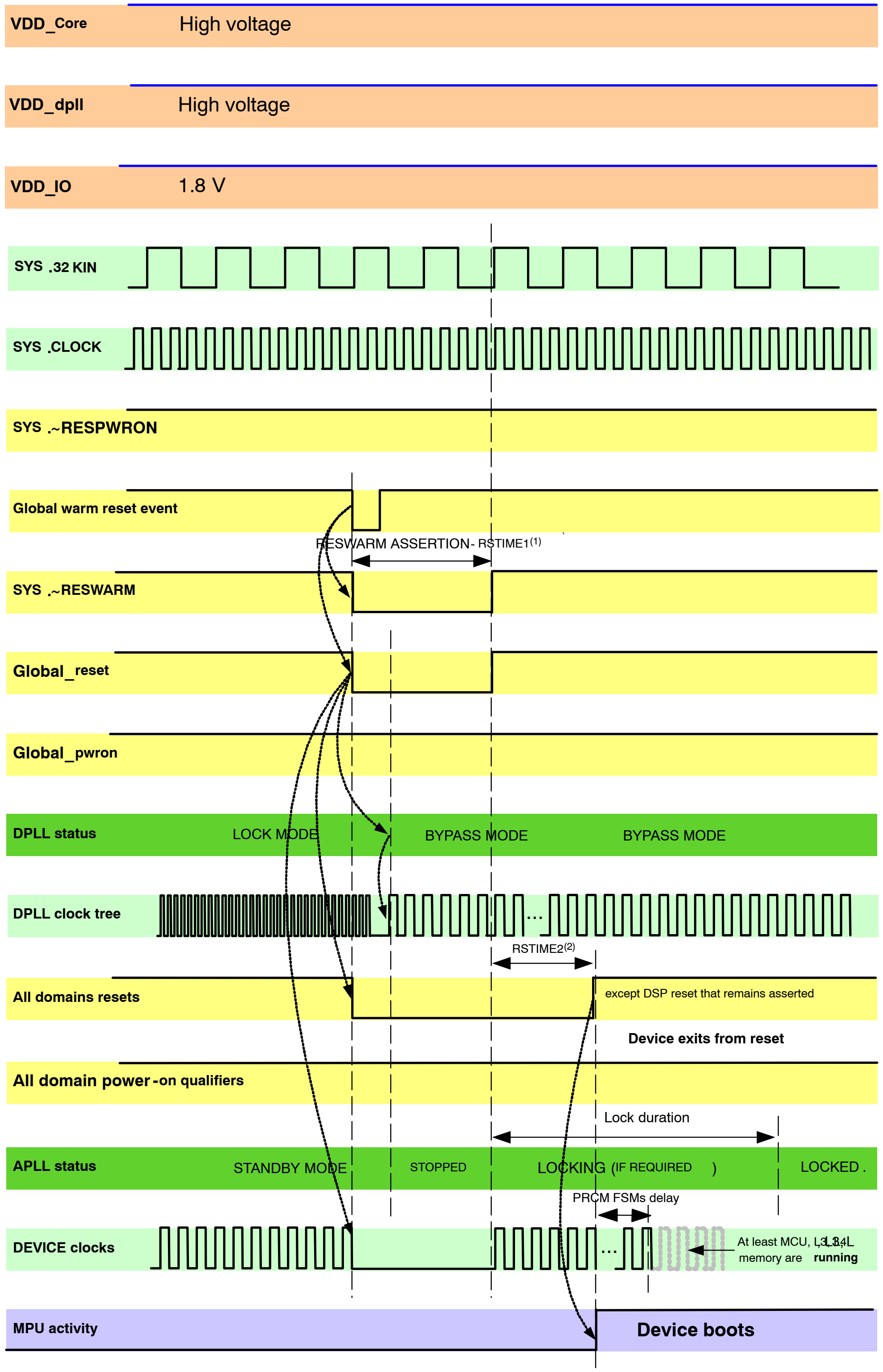
      - MCU, L3, L4 interconnect, and memory interface clock run.
      - Flashing peripheral clocks run.
      - All other clocks are off.

   h) The MCU boots.

5.5.5.2 Global Warm Reset Sequence

Figure 5−23 shows the warm reset sequence signaling.

Figure 5−23. Warm Reset Sequence



Notes:

1) RSSTIME1 duration is the value programmed in the RM_RSTTIME_WKUP register + 1 (the default value is 3 x 32-kHz clock cycles). There is one exception: if a transition to RET/OFF is initiated before reset assertion, the PRCM state-machine ended cleanly with the power state transition before applying the programmed RM_RSTTIME_WKUP value. This extra delay for RESWARM deassertion is not predictable, because it is highly dependent on the PRCM software setup and the exact PRCM state when reset is asserted.
2) RSSTIME2 maximum duration is the value programmed in the RM_RSTTIME_WKUP register + 1 (the default value is 17 x L4 clock cycles = 17 x system clock cycles after reset).

The reset sequence behaves as follows on an external warm or internal software reset:

- Operating condition assumptions:
  - V_{DD} core is either low-voltage point or high-voltage point.
  - V_{DD} I/O is in regulation at 1.8 V.
  - Oscillator output clock is stable.
  - 32-kHz clock is stable.
- Reset sequence:

1) While SYS.nRESPWARM or any global internal reset source (software or MPU watchdog) is asserted:
   a) The major part of the PRCM is reset.
   b) The device level remains or goes to voltage level L0. Ramp time must be lower than RSTTIME1.
   c) All domain resets are asserted; all power-on qualifiers are deasserted.
   d) The oscillator is kept enabled.
   e) SYS_CLK is on.
   f) APLLs are stopped.
   g) The DPLL is in bypass mode.
   h) All clocks derived from CORE_CLK are on.
   i) SDRAM is put in self-refresh mode.
2) On global reset source deassertion:
   a) The device reset manager starts the reset count RSTTIME1.
   b) All domain resets are kept asserted; all power-on qualifiers are deasserted.
   c) The DPLL is in bypass mode (CORE_CLK is system clock).
3) When the device manager reset count expires:
   a) All domain resets are kept asserted, all power-on qualifiers are deasserted, and all domain reset managers start the reset count RSTTIME2.
   b) APLLs are stopped or enabled, depending on the setting.
   c) The DPLL is in bypass mode (CORE_CLK runs at SYS_CLK).
   d) All clocks run at SYS_CLK frequency (dividers are 1 by default).
4) When all domain reset managers reset counts expire:
   a) All domain resets are deasserted, except DSP reset.
   b) The DPLL is in bypass mode (CORE_CLK runs at THE SYS_CLK frequency).

c) Clocks are stopped or kept running according to the PRCM setting:

  - MCU, L3, L4 interconnect, and memory interface clocks are running.
  - All other clocks are off.

d) The MCU boots.

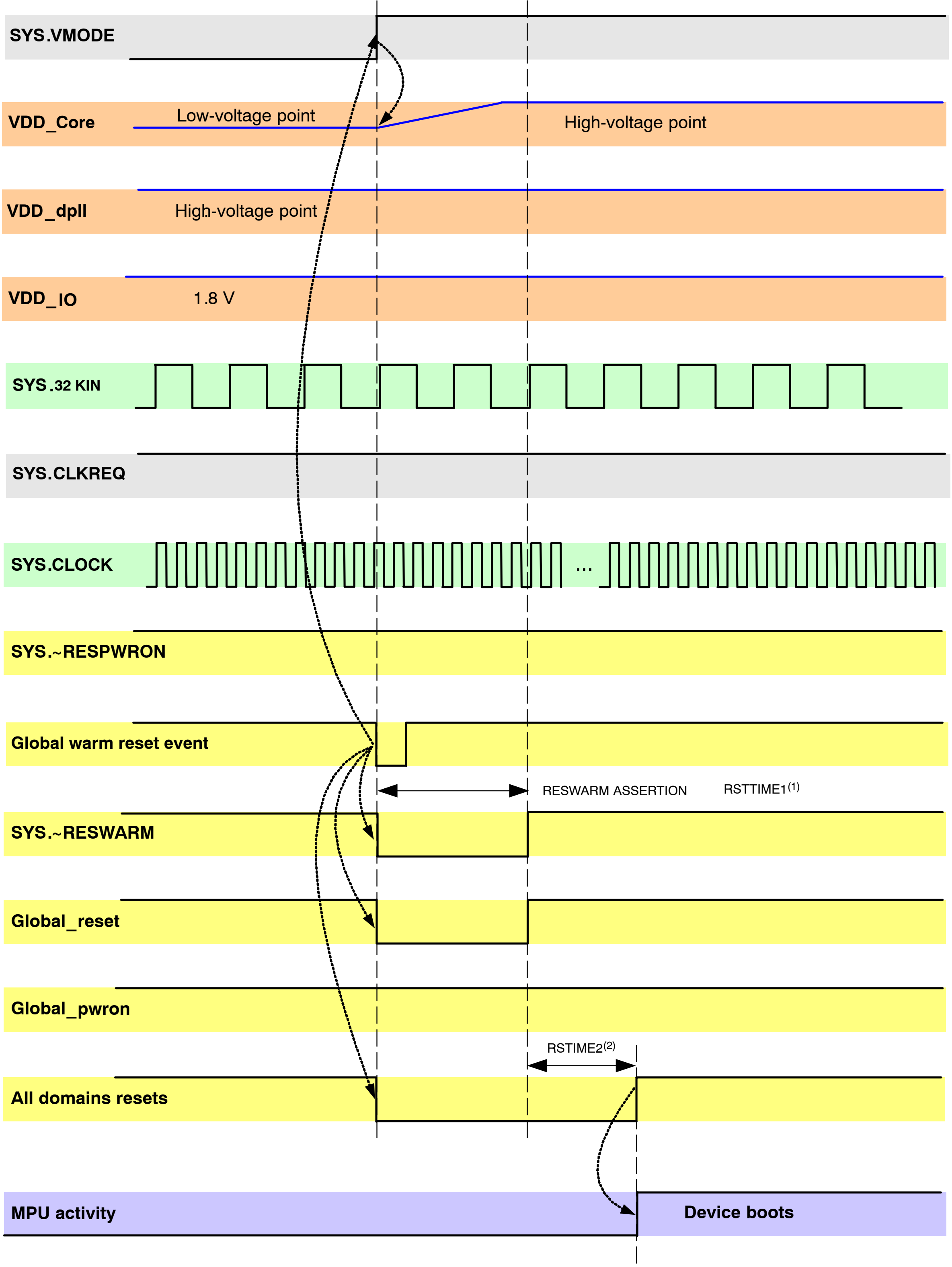
On warm reset, external control signals SYS.nCLKREQ and SYS.nVMODE behave as follows:

- SYS.nCLKREQ is asserted to request the external clock (when using the internal oscillator, this signal operates but is not useful and can be replaced by a GPIO).
- SYS.nVMODE is set to request the programmed volt level (the device can wake up at either low-voltage point or high-voltage point, depending on the software settings).
- The internal volt-setup counter is started to account for regulator ramp/stabilization time. The PRCM can also rely on the external POWER OK signal.
- If the internal oscillator is in use, it is started and the clock setup time is counted.
- When both voltage and system clock are stable, internal reset can be released and the device reboots.

Figure 5−24 describes a warm reset sequence when the device is operating at low voltage.

- Voltage is at low-voltage point. The external clock is active.
- On external reset assertion, the voltage goes back to high-voltage point.

Figure 5−24. Voltage Transition Induced by Any Global Warm Reset

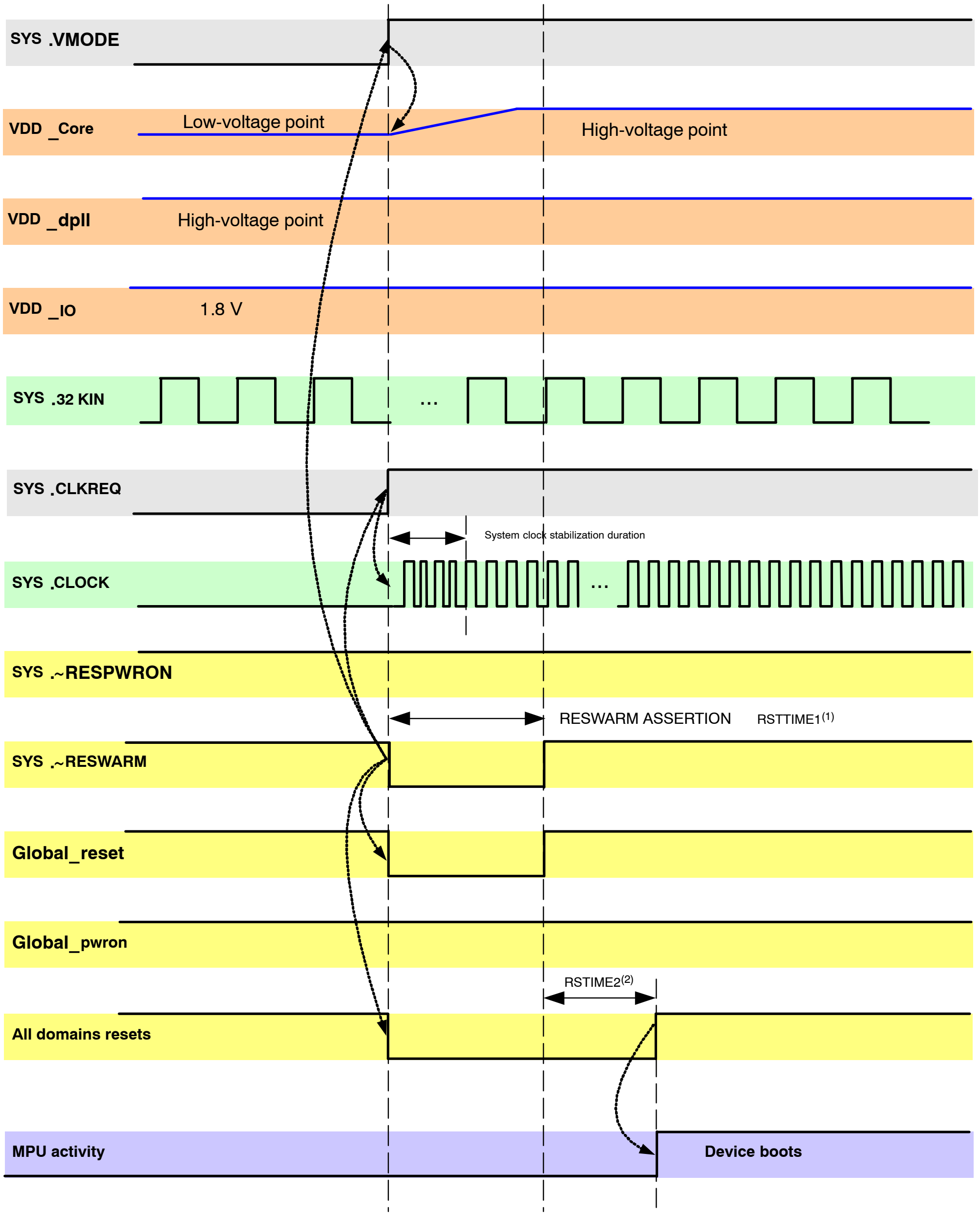


Notes:
1) RSSTIME1 maximum duration is the value programmed in the RM_RSTTIME_WKUP register + 1 (the default value is 3 x 32-kHz clock cycles). The voltage ramp-up time should be included if it is required that the MPU not start until voltage is stable.
2) RSSTIME2 maximum duration is the value programmed in the RM_RSTTIME_WKUP register + 1 (the default value is 17 x L4 clock cycles = 17 x system clock cycles after reset).

Figure 5−25 describes a wake-up sequence induced by an external warm reset:

- Device in retention or off mode. Voltage is at low-voltage point. The external clock is shut down.
- On external reset assertion, SYS_CLK is reactivated, voltage goes back to high-voltage point, and the device is awakened.

Figure 5−25. Device Wake-Up Sequence on External Warm Reset

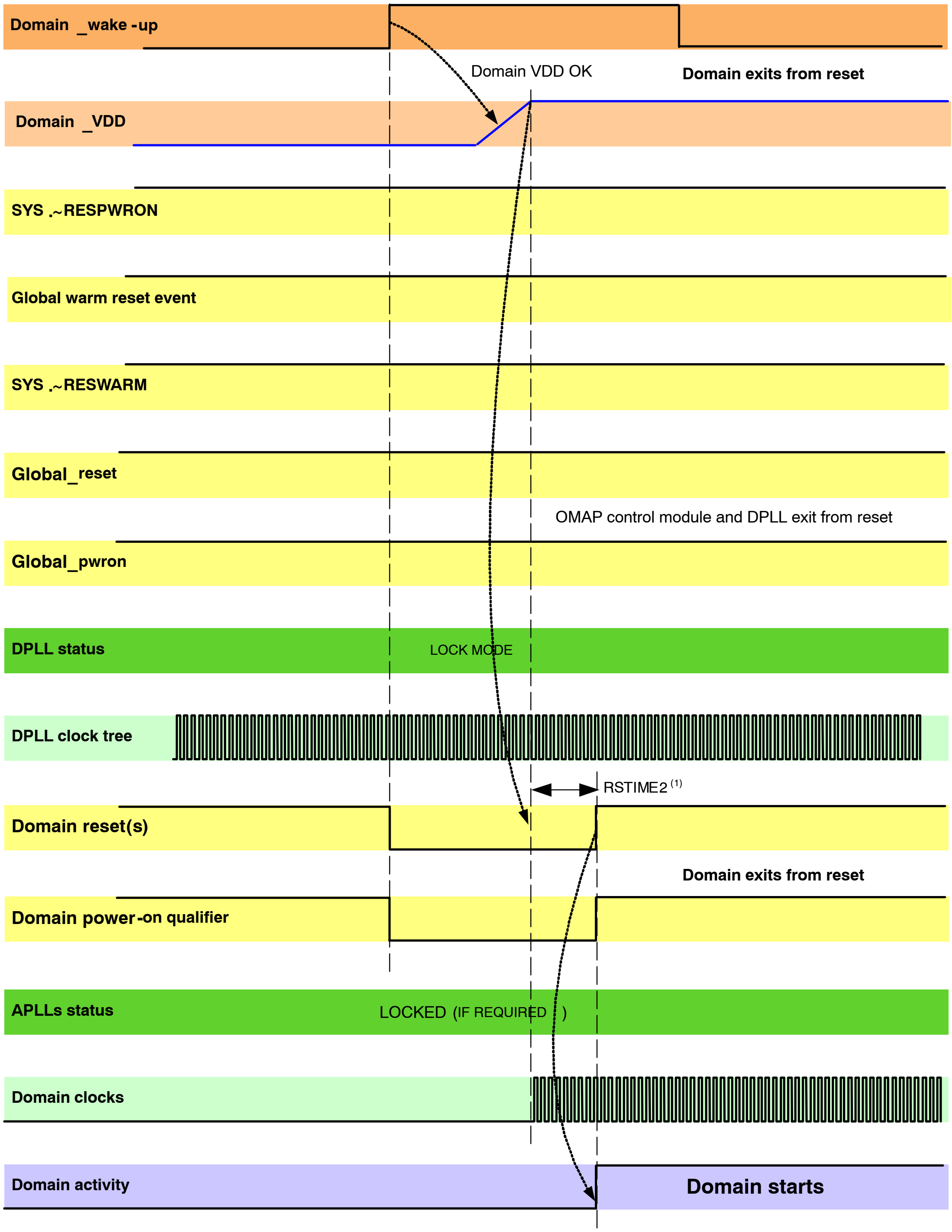


Notes:
1) RSSTIME1 maximum duration is the value programmed in the RM_RSTTIME_WKUP register + 1 (the default value is 3 x 32-kHz clock cycles.)
2) RSTTIME2 maximum duration is the value programmed in the RM_RSTTIME_WKUP register + 1 (the default value is 17 x L4 clock cycles = 17 x system clock cycles after reset.)

5.5.5.3 Power Domain Wake-Up Sequence

When a domain wake-up transition occurs (from off to active), a reset is applied on the domain. Figure 5−26 shows the power domain wake-up reset sequence signaling.

Figure 5−26. Power Domain Wake-Up Reset Sequence



Note: RSSTIME2 maximum duration is the value programmed in the RM_RSTTIME_WKUP register + 1 (the default value is 17 x L4 clock cycles).

Operating Condition Assumptions

- ☐ V_{DD} core is either low-voltage point or high-voltage point.
- ☐ V_{DD} I/O is in regulation at 1.8 V.
- ☐ Oscillator output clock is stable.
- ☐ 32-kHz clock is stable.

Reset Sequence

1) On DOMAIN_WAKEUP signal assertion:
    a) The PRCM applies a domain reset.
    b) All resets of the domain are asserted.
2) When the domain is on:
    a) The domain reset manager starts the reset count RSTTIME2.
    b) If required, clocks in the domain are activated.
3) When the domain reset manager reset count expires:
    a) If domain clocks are activated, domain reset is deasserted or not, depending on the domain:
        - MPU reset is always deasserted (then MPU boots).
        - Core reset is always deasserted.
        - Wake-up reset is always deasserted.
        - DSP resets can be deasserted (processors boot) or not depending on the RM_RSTCTRL_DSP register programming.

5.6 Power Management Functional Description

5.6.1 Overview

OMAP2420 architecture integrates advanced power-management capabilities. Three main features are implemented to reduce both dynamic consumption and static leakage:

- Device partitioning into five power domains, each controlled independently
- Memory retention
- Voltage scaling support

5.6.1.1 Device Partitioning

The OMAP2420 is segmented in five different power domains: MPU, DSP, CORE, and wake-up (WKUP). This allows switching off (or putting in retention state) unused domains while keeping others active. This implementation is based on internal switches that allow independent control for each power domain. The same voltage is supplied to the five power domains and can be externally scaled.

Table 5−18 through Table 5−21 list the OMAP2420 module distribution.

Table 5−18. MPU Power Domain Modules

MPU
MPU subsystem

Table 5−19. DSP Power Domain Modules

DSP
DSP subsystem

Table 5−20. Core Power Domain Modules

CORE	
L3 interconnect	L4 interconnect
MPU interrupt controller	HDQ −1WIRE
MPU SSM	UART1
MPU AHB2OCP	UART2
MPU ETB MGMT	UART3
DSP interrupt controller	
	SPI−1
sDMA	SPI−2
SMS	
SDRC	USB OTG
GPMC	
OCM ROM	
OCM RAM	DES/3DES
MAILBOXES	
MMC_SDIO	
STI/XTI	McBSP−1
EAC	McBSP−2
FAC	MSI2C−1
GPTIMER−2	MSI2C−2
GPTIMER−3	CAM interface
GPTIMER−4	Display subsystem
GPTIMER−5	WDTIMER3
GPTIMER−6	WDTIMER4
GPTIMER−7	
GPTIMER−8	
GPTIMER−9	
GPTIMER−10	
GPTIMER−11	
GPTIMER−12	

Table 5−21. WKUP Power Domain Modules

WKUP	
PRCM	OMAP2420 control module
(including oscillator, DPLL,	WDTIMER1
APLL (54, 96 MHz)	WDTIMER2
GPTIMER	
32 KHz synchronization timer	
GPIO−1	
GPIO−2	
GPIO−3	
GPIO−4	

5.6.1.2 SRAM Retention Mode

The retention state is a low-power state from which it is possible to exit with short latency because memory contents and logic states are maintained. This state is not functional. All clocks are gated. The standby feature is software-controllable.

MPU cache memories have their own switched supply that retains the cache contents while MPU logic is off (dormant mode).

The CORE and DSP domain memories are supplied through their domain power switch. They cannot retain data when the switch is open (off mode).

Table 5−22 summarizes memory and logic states for each power domain.

Table 5−22. Power Domain Features

Power Domain	Independent Power Control	Memory Retention	Logic Retention
MPU	Yes: Separate embedded switches for logic and memory	Yes: Memory can be kept in retention even if the MPU logic is off (dormant mode)	Yes: While power domain is supplied
DSP	Yes: Embedded switch	Yes: Only when logic is supplied	Yes: While power domain is supplied
CORE	Yes: Embedded switch	Yes: Only when logic is supplied	Yes: While power domain is supplied
WKUP	Yes: Always supplied (no embedded switch)	N/A	Yes: Domain is always supplied

5.6.1.3 Voltage Scaling

The PRCM ensures correct internal sequencing when dynamic voltage scaling is used. It also offers direct control to allow quick swapping between two programmed voltage levels (L0 and L1). Depending on the external power IC features, several voltage steps can be available between supported maximum and minimum voltages. The supported voltage range is from either low-voltage point to high-voltage point. All five power domains are supplied from the same voltage source.

When running at low-voltage point, all main clocks (processors, hardware accelerators, interconnect, memory controllers, etc.) must be divided by 2 while peripheral clocks remain unchanged.

For more information, see Section 5.9, *Programming Model*.

5.6.2 Power Domain Implementation

A power domain is a section of the device with independent and dedicated power rails. Modules with the same type of power use model (power mode) are grouped in the same power domain. Power modes are characterized by total device (chip) power consumption and wake-up latency. System power modes are defined (by software only) as any valid combination of domain power states, as opposed to power states, in which there are no hardware-predefined power modes.

5.6.2.1 Power States

Power states are associated with power domains. They are characterized by domain clock activity, power-supply voltage level, and state retention. Because of the independent control of domain power states, system software can select the correct power state for each domain to match current activity and performance requirements.

Definitions

The WKUP power domain is continuously active. The MPU, DSP, and CORE power domains can be in four different power states, as shown in Table 5−23.

Table 5−23. Supported Power States

States	Power Supply	Clocks	State Retained
Active	On (voltage scalable)	On	All
Inactive	On (voltage scalable)	Off	All
Retention	On (memory in retention)	Off	All
Off	Off	Off	None

The different power states are defined as follows:

- ❑ Active is a state in which the domain is functional. Main clocks are running. The device voltage can be scaled, according to the performance needs, from low-voltage point to high-voltage point. Voltage scaling applies simultaneously to all domains.
- ❑ Inactive is a state in which all domain clocks are stopped.
- ❑ Retention is a low-power state from which it is possible to exit with short latency because memory and logic contents are maintained. This state is not functional. All clocks are gated.
- ❑ Off is a non-functional state in which both power and clocks are shut down. A wake-up detection function is mandatory to exit from this state.

Table 5−24 summarizes the retention configurations that are possible in the OMAP2420.

Table 5−24. Retention Configurations

Memory State	Logic State	Domain Power State
Retention	Retention	Full retention
Off	Retention	Context retention
Retention	Off	MPU dormant mode (supported in MPU domain only)

To save power in low-activity mode, you can also put unused CORE-domain memory blocks in retention or off state while the CORE domain remains active.

The software must avoid performing any accesses to memory blocks in retention mode.

Because logic in the retention state must be powered, the main supply voltage cannot be shut down. To minimize leakage current, it is recommended to scale down the voltage to its minimum value when all domains are programmed in retention state.

The OMAP2420 hardware implementation has the following restrictions:

On state:

- When the MPU domain is on, the MPU memories (caches) are always on.
- When the DSP domain is on, DSP core memories are on or in retention (driven by DSP core); the remaining DSP subsystem memories are always on.
- When the core domain is on, the core domain memory can be on, in retention, or off.

Retention state:

- When the MPU domain is in retention, both MPU logic and cache are retained, or MPU logic is off and caches are retained (dormant mode).
- When the DSP domain is in retention, the DSP subsystem and memory are retained.
- When the CORE domain is in retention, both CORE logic and memory are retained, or CORE logic is retained and memory is off.

Off state:

- The CORE power domain can enter off state only when the MPU, and DSP power domains are in the off state.

Each power domain output signal at the interface between power domains is connected through an isolation latch cell. These cells ensure correct electrical isolation between the domains and an appropriate interface state at the boundary of domains.

Combinations

Table 5−25 provides some examples of power domain state combinations.

Table 5−25. Power Mode Scenarios

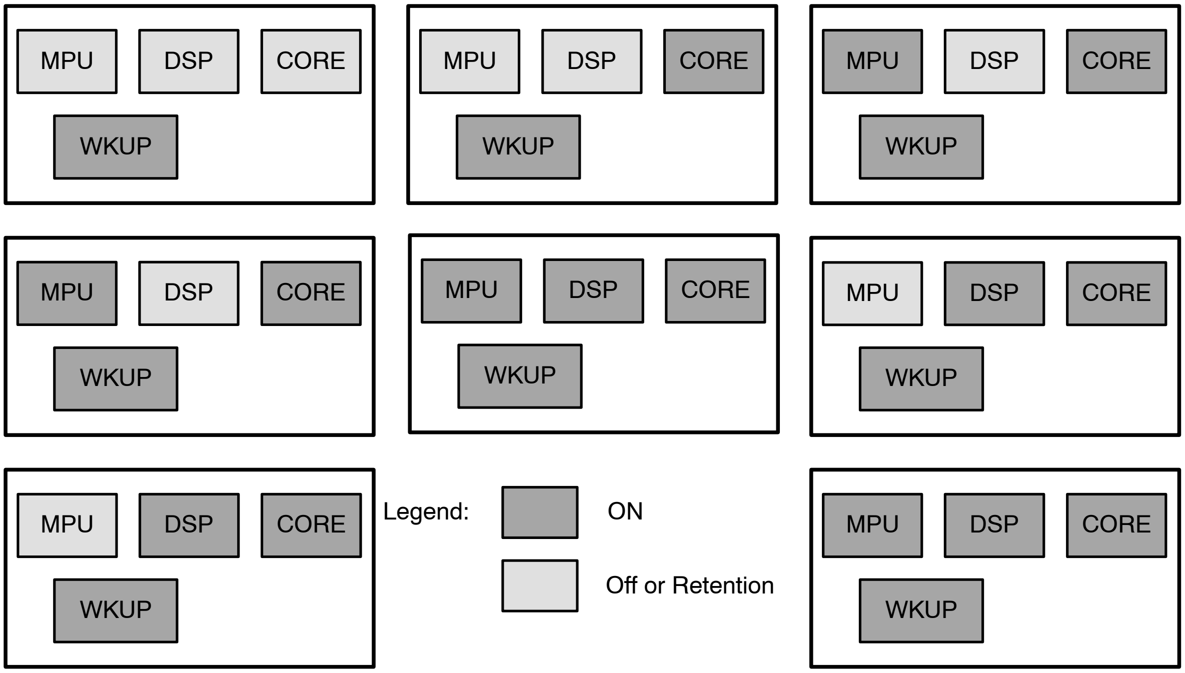
Modes	MPU Power Domain State	DSP Domain State	CORE Power Domain State	WKUP Power Domain State	Comment
Device on	Inactive	Inactive	Inactive	Active	No leakage management
Device Retention	Retention	Retention	Retention	Active	Chip is not operational but all its context is saved.
Standby	Retention	Off	Partially inactive (only the display subsystem is active)	Active	Typical chip standby mode (base for display refresh)
Sleep	Retention	Off	Retention	Active	Very low leakage mode with reduced latency restore
Off	Off	Off	Off	Active	Off mode: Lowest power consumption. This mode allows wakeup without external voltage ramp.

Figure 5−27 shows the different power state combinations that are possible with the OMAP2420.

Note:

The CORE power domain must be active when the MPU or DSP power domain is active.

Figure 5−27. Power State Combinations



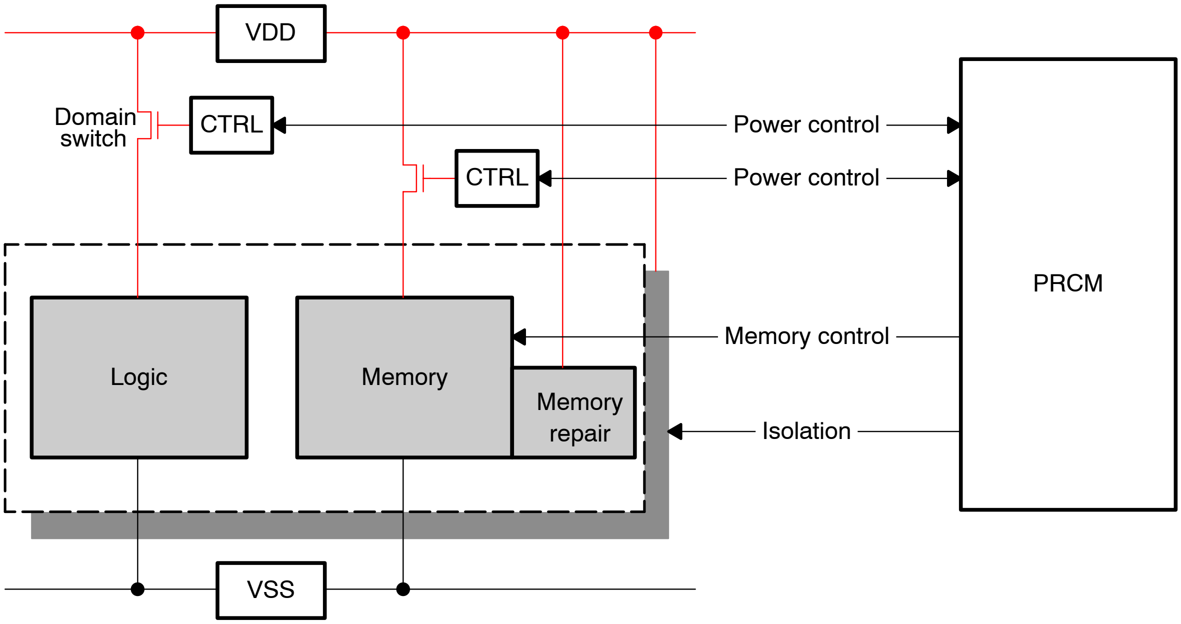
5.6.2.2 Power Supply Control

The following sections describe how power is supplied to memory and logic for each power domain.

MPU Power Supply Control

Memory is controlled independently from logic to support the MPU dormant mode (see Figure 5−28).

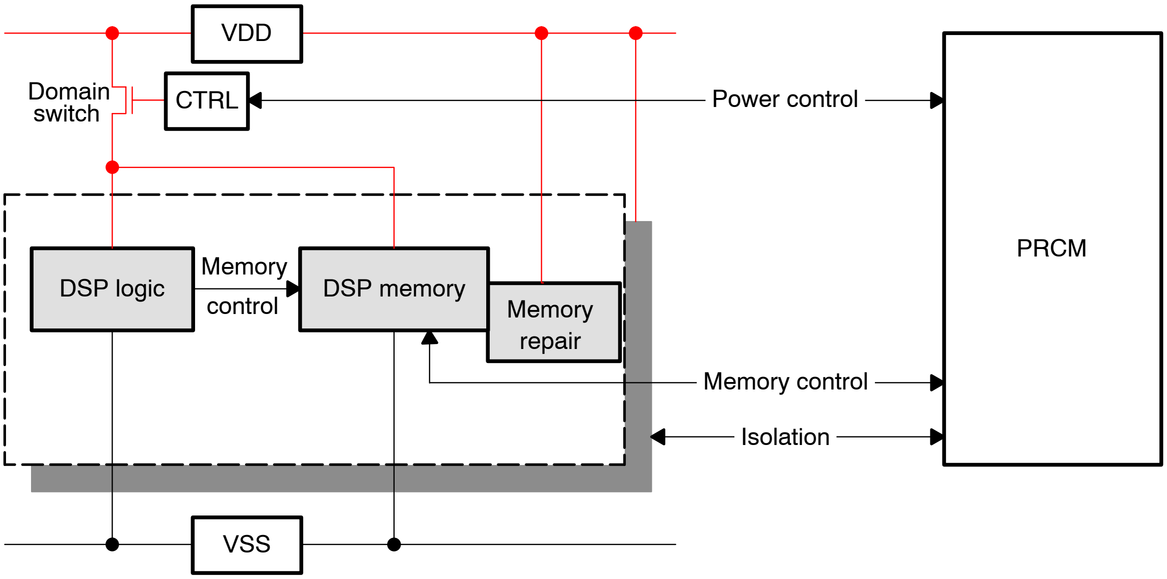
Figure 5−28. MPU Domain Power Supply



DSP Power Supply Control

The domain switch supplies DSP memory. It is controlled internally by the DSP core and not by the PRCM (see Figure 5−29).

Figure 5−29. DSP Domain Power Supply



CORE Domain Power Supply Control

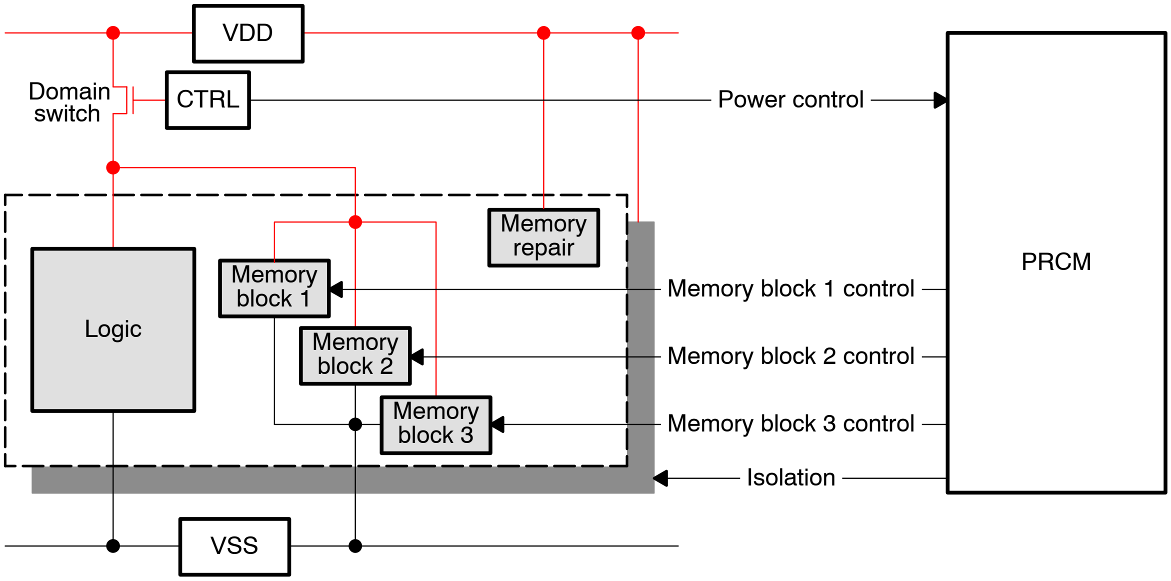
The OMAP2420 CORE power domain memory is composed of 3 blocks: block 1 is dedicated for secure RAM memory; blocks 2 and 3 are dedicated to the frame buffer. Table 5−26 lists the blocks with their definitions.

Table 5−26. OMAP2420 Internal Memory

	Total Size (K bytes)	Usage
Memory block 1	64	Secure RAM
Memory block 2	320	Frame buffer
Memory block 3	256	Frame buffer

The memory blocks are all supplied by the domain switch and are controlled independently by the PRCM module. See Figure 5−30.

Figure 5−30. Core Domain Power Supply



5.7 PRCM Interrupts

5.7.1 Definition

The PRCM generates two interrupts:

☐ PRCM_MPU_IRQ
☐ PRCM_DSP_IRQ

These interrupts are mapped to the MPU interrupt controller and the DSP interrupt controller, respectively.

Table 5−27 summarizes the events on which interrupts are generated.

Table 5−27. PRCM Interrupts Summary

Interrupt Name	Destination	Events
PRCM_MPU_IRQ	MPU Interrupt controller	Wake-up event type 1 occurred.
		Wake-up event type 2 occurred.
		Voltage transition is complete.
		End of on time event occurred.
		End of off time event occurred.
		Sleep (DSP domain) or wake-up (DSP domain) transition has completed.
PRCM_DSP_IRQ	DSP interrupt controller	Wake-up event type 1 occurred.
		Wake-up event type 2 occurred, or a force wake-up DSP domain transition is complete.
		Voltage transition is complete.

Notes:
1) Wake-up event type 1 is generated by a module that generates an interrupt or DMA request corresponding to the wake-up event.
2) Wake-up event type 2 is generated by a module that generates a wake-up event without triggering an interrupt (or DMA request) behind.
Wake-up event type 2 is also triggered by a DSP domain force wake-up event.

Wake-up event types are listed in Section 5.8, *Sleep and Wake−Up Processes*.

5.7.2 Interrupt Control

The PRCM interrupts can be masked using the PRCM interrupt enable registers, as follows:

1) PRCM_IRQENABLE_MPU that allows masking interrupt events to the MPU subsystem interrupt controller. The masking is based on an event-by-event scheme.

2) PRCM_IRQENABLE_DSP that allows masking interrupt events to the DSP subsystem interrupt controller. The masking is based on an event-by-event scheme.

Their status is readable in the PRCM interrupt status registers:

- PRCM_IRQSTATUS_MPU
- PRCM_IRQSTATUS_DSP

5.8 Sleep and Wake-Up Processes

5.8.1 Overview

To save dynamic consumption, the OMAP2420 idle scheme is based on:

- ❏ Efficient local clock autogating for each module
- ❏ The implementation of hardware handshake protocols between the PRCM and each module or subsystem

This enhanced idle control enables safe activation/deactivation of clocks without requiring complex software management.

This section describes power-management support and the required software control for the modules and subsystems.

5.8.2 Definitions

5.8.2.1 Initiator and Target Modules

The OMAP2420 includes two categories of modules or subsystems:

- ❏ Initiator: A module that can generate traffic on the chip interconnect (typically processors, MMU, DMA, etc.)
- ❏ Target: A module that cannot generate traffic on the chip interconnect, but can generate interrupts or a DMA request to the system (typically peripherals).

  Target modules can be divided into two subcategories:

  - ■ Wake-up targets with asynchronous wake-up capabilities
  - ■ Non-wake-up targets without wake-up capabilities

5.8.2.2 Active, Idle, and Standby Modes

An initiator module is in active mode when working. When not working (that is, not processing or generating traffic), an initiator module is in standby mode.

A target module is also in active mode when working and in idle mode when the system asks it to stop activity. In idle mode, only targets with wake-up capability can wake up the system by generating a wake-up event. The others are awakened only by a system request.

A module in idle mode is idle from a system standpoint (that is, no access can be performed to this module, it cannot generate an interrupt or DMA request). It can be functional. Clocks may be cut or not, depending on their wake-up feature requirements. The software controls whether clocks are effectively cut during this mode.

5.8.2.3 Hardware Handshake Protocols

The PRCM handles hardware handshake protocols for target and initiator modules.

- ☐ The IDLE mode handshake allows the PRCM to enter an idle mode while awaiting a target. The target acknowledges when it is ready.
- ☐ The MSTANDBY handshake is initiated by an initiator module to inform the PRCM that it enters standby mode and does not generate traffic on the interconnect.

5.8.3 Power-Management Software Control for Modules and Subsystems

Most OMAP2420 modules and subsystems support power-management features. Control at the module or subsystem level occurs using the <*IP-Name*>_SYSCONFIG register.

The <*IPName*>_SYSCONFIG register includes the following parameters:

- ☐ Master-interface power management (initiators)
  - Force-standby: The initiator module reports that it is in standby only when it is disabled.
  - No-standby: The initiator module never reports that it is in standby.
  - Smart-standby: The initiator module reports that it is in standby mode, depending on its internal activity.
- ☐ Slave interface power management
  - Force-idle: The target module always acknowledges an idle request issued from the PRCM.
  - No-idle: The target module never acknowledges an idle request issued from the PRCM.
  - Smart-idle: The target module acknowledges an idle request issued from the PRCM, depending on its internal activity.
- ☐ Clock(s) activity during idle mode

The CLOCKACTIVITY two-bit field indicates whether the module clocks (interface and functional) will be switched off after the module enters idle mode. (There is one bit per clock.) This field can be extended if a module has more than one functional clock.

Table 5−28 shows CLOCKACTIVITY field use for a module with one interface clock and one functional clock.

Table 5−28. CLOCKACTIVITY Configuration

<*IPName*>_SYSCONFIG[9:8] (ClockActivity Bit Field)	**Interface Clock**	**Functional Clock**
00	Off	Off
10	On	Off
01	Off	On
11	On	On

If the control bit is on, the system power manager does not switch off the clock during idle mode.

CAUTION

The power-management software driver must ensure overall system consistency.

- Internal-interconnect clock-gating strategy

  In addition to the global power-management scheme, a local power-optimization scheme can be accomplished in the module by gating interconnect clock subdomains. This auto idle feature is enabled by setting the AUTOIDLE control bit in the <*IPName*>_SYSCONFIG register.

  The decision to gate the interconnect clock is based on interface activity, regardless of the hardware handshake protocols. This control bit is also used to enable auto-gating of the module functional clock domains. Auto-gating is independent of the hardware handshake protocol.

  Table 5−29 summarizes the power-management features supported by each OMAP2420 module or subsystem.

Table 5−29. Power-Management Features of Modules and Subsystems

Module/ Subsystem	**Type**	**IDLE Handshake**					**STANDBY Handshake**				**Auto-Idle**
		Force-Idle	**No-Idle**	**Smart-Idle**	**Note**	**Wake-Up**	**Force-Stdby**	**No-Stdby**	**Smart-Stdby**	**Note**	
MPU											
MPU	Initiator								Always	1	
Interrupt controller	Target			Always		Yes					Yes
DSP											
DSPSS main part (UMA, DMA)	Initiator						Yes	Yes	Yes	2	
IPI	Target	Yes	Yes	Yes							Yes
Interrupt controller	Target			Always	1	Yes					Yes
Camera											
Camera core	Target	Yes	Yes	No	3						Yes
CamDMA	Initiator						Yes	Yes	Yes	4	Yes
Display	Initiator						Yes	Yes	Yes		
L3 interconnect	Target			Always	1						
L4 interconnect	Target			Always	1						

Table 5−29. Power-Management Features of Modules and Subsystems (Continued)

SDMA	Initiator						Yes	Yes	Yes		Yes
GPT[1...12]	Target	Yes	Yes	Yes		Yes					Yes
FAC	Target	Yes	Yes	Yes							Yes
SPI[1...2]	Target	Yes	Yes	Yes		Yes					Yes
GPIO[1...4]	Target	Yes	Yes	Yes		Yes					Yes
OCM RAM	Target			Always	1						Always
OCM ROM	Target			Always	1						Always
SMS	Target	Yes	Yes	Yes							Yes
SDRC	Target			Always	1						
GPMC	Target	Yes	Yes	Yes							Yes
Mailbox	Target	Yes	Yes	Yes							Yes
UART[1...3]	Target	Yes	Yes	Yes		Yes					Yes

1) Not programmable by software
2) Configured at DMA level (see Chapter 10)
3) Smart-idle not implemented
4) Never use smart-standby (see Chapter 14)

CAUTION

The following modules do not support handshake protocols: MMC/SDIO, I^{2}C, McBSP, 32k SYNC Timer, WDT[1..4], EAC, and HDQ/1-Wire. The software must ensure correct clock management relative to the idle/standby states.

5.8.4 Sleep Transitions

5.8.4.1 Clock Sleep Transition

A sleep transition consists of stopping activity and clocks in a clock domain. When this is complete, the clock domain is identified as idle.

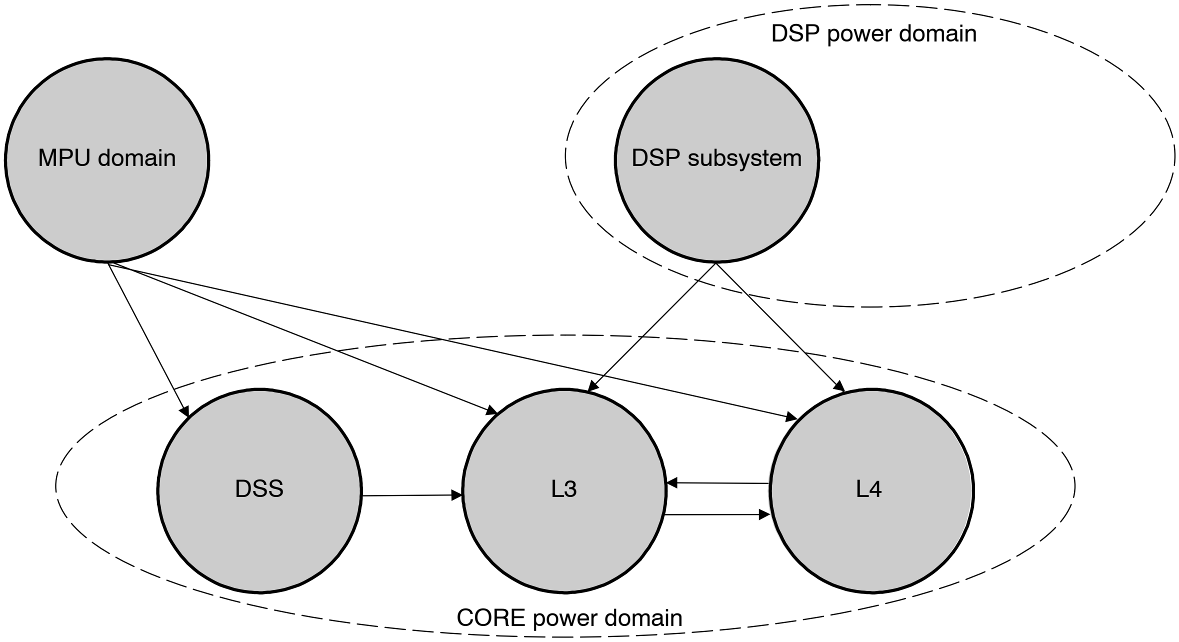
To achieve a sleep transition, ensure that:

- All initiator modules in the clock domain are in standby mode.
- All target modules in the clock domain are in idle mode.
- All functional and interface clocks are shut down.

Each clock domain can be dependent on another, in which case its sleep transition relies on the other clock domain state. Dependencies are based on functional constraints.

Figure 5−31 shows the dependencies that exist among the clock domains.

Figure 5−31. Clock Domain Dependencies



When all domains are idle, the full device is identified as idle.

> **The L4 clock domain is an exception: it can be idled even if GPT1 and GPIOs (in the WKUP domain) keep their functional clock active (but the interface clock is off). This allows the full device to be idled while keeping wake-up capabilities.**
>
> CAUTION

5.8.4.2 Power Sleep Transition

A power domain that contains one or more clock domains, all identified as idle, is in inactive state (all clocks in the power domain are shut off).

In the inactive state, the domain is still powered and ready for deeper sleep transition. The domain can, therefore, enter retention or off mode, depending on the related power state control register setting.

> **Direct sleep transition from the retention mode to the off mode is not feasible. Once in retention mode, only a wake-up transition is permitted.**
>
> CAUTION

5.8.5 Wake-Up Transitions

5.8.5.1 Wake-Up Actions

A wake-up event can be generated only from an active domain.

A module in a power domain in retention or off state cannot generate a wake-up event; thus, when the full device is in retention or off state, only a wake-up issued from the wake-up domain (always active) can wake up the device.

A wake-up event can provoke either:

- A clock wake-up transition from inactive to active state
- A power and clock wake-up transition from off or retention state to active state

Therefore, it has several consequences, depending on the domain power state:

- When the power domain is off, a wake-up sets it in on mode, resets it, and reactivates its clocks (depending on the PRCM clock register settings). For the MPU power domain, MPU boots on reset release. DSP reset release is controlled by software.
- When the power domain is in retention, a wake-up sets it in on mode and reactivates its clocks (depending on the PRCM clock-register settings). When the MPU is in dormant mode, the MPU power domain is reset and the MPU boots on reset release. Processors require an interrupt request to exit from their standby mode.
- When the power domain is on and inactive, a wake-up reactivates its clocks, depending on the PRCM clock register settings. Processors require an interrupt request to exit standby mode.

5.8.5.2 Wake-Up Dependencies

Dependencies between power domains allow a wake-up event issued in a particular power domain to cause another domain to wake up.

This feature reduces wake-up latency by managing simultaneous power domain wakeups automatically instead of sequentially by software.

Table 5−30 summarizes the available wake-up dependencies.

5.8.5.3 Wake-Up Interrupts

The PRCM can generate two different interrupts to the two processors (MPU and DSP). These interrupts can be caused either by PRCM internal events or by wake-up events.

The PRCM provides a wake-up enable and wake-up status registers for each power domain. Each wake-up occurrence is logged in the PRCM logic and must be cleared by software by clearing the related wake-up status bit.

When available, a corresponding interrupt can be enabled and generated to the MPU or DSP processor and must be cleared by clearing its status register.

CAUTION

If a wake-up event is logged and not cleared, it prevents the power domain from any new sleep transition.

5.8.5.4 Wake-Up and Interrupts Summary

Table 5−30 through Table 5−34 summarize all sources of wake-up for each power domain, including the wake-up source, software control, and related interrupt, when it exists.

Table 5−30. MPU Power Domain Wake-Up Events

Internal Wake-Up Event	Originator Module	PRCM Software Control	MASK Method	Interrupt-Type	Interrupt Mapped to
DSP power domain dependency	PRCM	Yes	PM_WKDEP_MPU register (tied)	No	N/A
CORE power domain dependency	PRCM	Yes	PM_WKDEP_MPU register	No	N/A
WKUP power domain dependency	PRCM	Yes	PM_WKDEP_MPU register	No	N/A
Event generator off time	PRCM	Yes	PM_EVGENCTRL_MPU register	Event generator off	MPU
Voltage setup time	PRCM	No		Voltage transition completed	MPU
MPU wake-up interrupt	MPU interrupt controller	No	See Chapter 11	No	N/A

Table 5−31. CORE Power Domain Wake-Up Events

Internal Wake-Up Event	Originator Module	PRCM Software Control	MASK Method	Interrupt Type	Interrupt Mapped to
MPU power domain dependency	PRCM	Fixed	PM_WKDEP_CORE register (tied)	No	N/A
DSP power domain dependency	PRCM	Fixed	PM_WKDEP_CORE register (tied)	No	N/A
WKUP power domain dependency	PRCM	Fixed	PM_WKDEP_CORE register (tied)	No	N/A
UART2 WKUP	UART2	Yes	PM_WKEN1_CORE register	1	MPU, DSP

Table 5−31. CORE Power Domain Wake-Up Events (Continued)

Internal Wake-Up Event	Originator Module	PRCM Software Control	MASK Method	Interrupt Type	Interrupt Mapped to
UART1 WKUP	UART1	Yes	PM_WKEN1_CORE register	1	MPU, DSP
McSPI2 WKUP	McSPI2	Yes	PM_WKEN1_CORE register	1	MPU, DSP
McSPI1 WKUP	McSPI1	Yes	PM_WKEN1_CORE register	1	MPU, DSP
GP Timer 12 WKUP	GP Timer 12	Yes	PM_WKEN1_CORE register	1	MPU, DSP
GP Timer 11 WKUP	GP Timer 11	Yes	PM_WKEN1_CORE register	1	MPU, DSP
GP Timer 10 WKUP	GP Timer 10	Yes	PM_WKEN1_CORE register	1	MPU, DSP
GP Timer 9 WKUP	GP Timer 9	Yes	PM_WKEN1_CORE register	1	MPU, DSP
GP Timer 8 WKUP	GP Timer 8	Yes	PM_WKEN1_CORE register	1	MPU, DSP
GP Timer 7 WKUP	GP Timer 7	Yes	PM_WKEN1_CORE register	1	MPU, DSP
GP Timer 6 WKUP	GP Timer 6	Yes	PM_WKEN1_CORE register	1	MPU, DSP
GP Timer 5 WKUP	GP Timer 5	Yes	PM_WKEN1_CORE register	1	MPU, DSP
GP Timer 4 WKUP	GP Timer 4	Yes	PM_WKEN1_CORE register	1	MPU, DSP
GP Timer 3 WKUP	GP Timer 3	Yes	PM_WKEN1_CORE register	1	MPU, DSP
GP Timer 2 WKUP	GP Timer 2	Yes	PM_WKEN1_CORE register	1	MPU, DSP
UART3 WKUP	UART3	Yes	PM_WKEN2_CORE register	1	MPU, DSP
MMC WKUP	MMC	Yes	PM_WKEN1_CORE register	2	MPU, DSP
USB WKUP	USB	Yes	PM_WKEN2_CORE register	2	MPU, DSP

Table 5−32. DSP Power Domain Wake-Up Events

Internal Wake-Up Event	**Originator Module**	**PRCM Software Control**	**MASK Method**	**Interrupt Type**	**Interrupt Mapped to**
MPU domain dependency	PRCM	Yes	PM_WKDEP_DSP register	No	N/A
Core domain dependency	PRCM	Yes	PM_WKDEP_DSPP register	No	N/A
Wake-up domain dependency	PRCM	Yes	PM_WKDEP_DSP register	No	N/A
Force transition state wake-up	PRCM	Yes	PM_PWSTCTRL_ DSP	2	DSP
Force transition state sleep	PRCM	Yes	PM_PWSTCTRL_ DSP register	Transition completed	MPU
IPI access	DSP sub-system IPI	Yes	NO	No	N/A

Table 5−33. WKUP Power Domain—Internal Events

Internal Wake-Up Event	**Originator Module**	**PRCM Software Control**	**MASK Method**	**Interrupt Type**	**Interrupt Mapped to**
GP Timer 1 WKUP	PRCM	Yes	PM_WKEN_WKUP register	1	MPU, DSP
GPIOs (OR of all GPIOs WKUP)	PRCM	Yes	PM_WKEN_WKUP register	1	MPU, DSP

Table 5–34. WKUP Power Domain—External Events

External Wake-Up Event (Pins)	Originator Module	PRCM Software Control	MASK Method	Interrupt Type	Interrupt Mapped to
SPI1 (CS0 pin)	Connected GPIO	No	See Chapter 23	1	MPU, DSP
SPI2 (CS0 pin)	Connected GPIO	No	See Chapter 23	1	MPU, DSP
UART1 (CTS pin)	Connected GPIO	No	See Chapter 23	1	MPU, DSP
UART2 (CTS pin)	Connected GPIO	No	See Chapter 23	1	MPU, DSP
UART3 (CTS pin)	Connected GPIO	No	See Chapter 23	1	MPU, DSP
McBSP1	Connected GPIO	No	See Chapter 23	1	MPU, DSP
McBSP2	Connected GPIO	No	See Chapter 23	1	MPU, DSP
USB	Connected GPIO	No	See Chapter 23	1	MPU, DSP
MMC	Connected GPIO	No	See Chapter 23	1	MPU, DSP

5.9 Programming Model

5.9.1 Overview

The PRCM contains a rich set of registers that allows programming of clocks, reset, and power for each power domain.

These registers are fully programmable or readable by the MPU and DSP. However, DSP power domain registers are mapped in a specific memory location for security (DSP can be restricted to access only to its registers).

All registers are detailed in Section 5.10, *Power, Reset, and Clock-Management Registers*.

There are three types of registers:

- ❏ Configuration registers: These registers allow configuration of the PRCM interface port and interrupt lines.
- ❏ Power domain registers: The same set of registers is defined for each power domain, allowing consistent programming.
- ❏ Global registers: These registers are defined for global control of the device: clock sources, voltage control, and general-purpose register.

This section defines the register types and describes, case-by-case, the programming of each subsystem or module.

5.9.2 Definitions

The PRCM propagates two kinds of clocks in the OMAP2420:

- ❏ Interface clock: Ensures correct communication between any module and the interconnect (L3 or L4). In most cases, the interface clock supplies the module interface and registers.
- ❏ Functional clock: Supplies the functional part of a module or a subsystem. In some cases, a module or a subsystem can require several functional clocks. Without a functional clock, a module is not operational.

Clock control registers provide granular control over these two clocks. The control may differ, depending on the module or subsystem implementation.

5.9.3 Configuration Registers

The PRCM contains seven system configuration registers:

- ❏ PRCM_REVISION: Indicates the PRCM revision code; read-only
- ❏ PRCM_SYSCONFIG: Holds an autoidle bit used to control the PRCM clock autogating feature
- ❏ PRCM_IRQSTATUS_MPU (MPU INTERRUPT STATUS REGISTER): Provides the status of all PRCM internal events that can generate an interrupt. This register applies on interrupt line 0 mapped to the MPU interrupt controller.

Six events can activate this interrupt line:

- Transition in progress complete
- End of on period in the event generator
- End of off period in the event generator
- Voltage transition complete
- Type 1 wake-up event occurred.
- Type 2 wake-up event occurred.

The software must first read the register to identify the interrupt cause, and then clear the pending interrupt by setting the corresponding bit to 1.

CAUTION

When the interrupt is caused by a wake-up event, you must also clear the wake-up source by clearing the related wake-up status bit.

- PRCM_IRQENABLE_MPU (MPU INTERRUPT ENABLE REGISTER): Allows masking/unmasking independently each of the four internal sources of interrupt:

  - Transition in progress complete
  - End of on or off period in the event generator
  - Voltage transition complete
  - Wake-up event type 1 or 2 occurred

  This register applies on interrupt line 0 mapped to the MPU interrupt controller.

- PRCM_IRQSTATUS_DSP (DSP INTERRUPT STATUS REGISTER): Provides the status of all PRCM internal events that can generate an interrupt; applies on interrupt line 1 mapped to the DSP interrupt controller.

  Three events can activate this interrupt line:

  - Voltage transition complete
  - Type 1 wake-up event occurred
  - Type 2 wake-up event occurred

  The software must first read the register to identify the interrupt cause, then clear the pending interrupt by setting the corresponding bit to 1.

CAUTION

When the interrupt is caused by a wake-up event, you must also clear the wake-up source by clearing the related wake-up status

- ❏ PRCM_IRQENABLE_DSP (DSP INTERRUPT ENABLE REGISTER): Allows masking/unmasking independently each of the two internal sources of interrupt:
  - ■ Voltage transition complete
  - ■ Wake-up event occurred

  This register applies on interrupt line 1 mapped to the DSP interrupt controller.

5.9.4 Power Domain Registers

An identical set of registers controls each of the five power domains. Implementation varies according to the power domain. This section describes the action of each register.

5.9.4.1 Clock Control

Clock Registers Description

The following registers control the clock settings:

- ❏ CM_FCLKEN_<domain_name> (FUNCTIONAL CLOCK ENABLE REGISTER): Allows control of the functional clock activity of each module or subsystem

  All OMAP module functional clocks can be controlled individually by software except the MPU, interconnects, and memory subsystems, for which the clock is automatically controlled by the PRCM.

  The OMAP2420 includes the following functional clock control registers:
  - ■ CM_FCLKEN1_CORE: Peripherals, display subsystem
  - ■ CM_FCLKEN2_CORE: UART3 and USB peripherals
  - ■ CM_FCLKEN_DSP: DSP
  - ■ CM_FCLKEN_WKUP: Peripherals

  The functional clock enable register has an immediate effect: the PRCM effectively initiates a procedure to shut down the clock. The clock source is shut down when the module is ready for quiescence and if other modules do not require the clock.

  Functional clock control is managed by software only, except at wake-up, when it can be automatically activated.

  CM_FCLKEN2_CORE controls peripherals that must be active at power-up for flashing. All bits are, therefore, set at reset.

Note:

The functional clock supplies the functional part of a module or subsystem. In some cases, a module or subsystem can require several functional clocks. A module is not operational without a functional clock.

- ❏ CM_ICLKEN_<domain_name> (INTERFACE CLOCK ENABLE REGISTER): Allows control of the interface clock activity of each module or subsystem.

This register provides control over each OMAP module.

The OMAP2420 includes the following interface clock control registers:

- CM_ICLKEN1_CORE: Peripherals, display subsystem
- CM_ICLKEN2_CORE: Boot peripherals
- CM_ICLKEN_DSP: DSP intrusive port-interface clock control
- CM_ICLKEN_WKUP: Peripherals

The interface clock enable register has an immediate effect and causes the source of the interface clock to be effectively cut (if the bit is cleared and the same source is not required by another module) or activated (if the bit is set).

CM_ICLKEN2_CORE controls peripherals that must be active at power-up for flashing; thus, all bits are set at reset.

Note:

The interface clock ensures correct communication between any module and the interconnect (L3 or L4), in most cases supplying the module interface and registers.

You can manage a fine clock control using those two registers. In any case, when the functional clock or interface clock is disabled, the module cannot communicate with the device and is required to enter idle mode.

For example, a peripheral can have its interface clock disabled while its functional clock remains active. It is idle from the device standpoint, but it can detect an external event. This configuration typically allows a main part of the device to go to idle state while keeping a peripheral active and ready to wake up from an external event.

A module can have a particular hardware dependency between its functional and interface clocks. This relation can be programmable and is defined by the ClockActivity bit field in the related SYSCONFIG configuration register.

The CM_FCLKEN_<domain_name> and CM_ICLKEN_<domain_name> registers must be programmed consistently.

All clocks are disabled at reset, except:

- MPU clocks
- L3, L4 clocks
- Memory controllers (GPMC, SDRC, SMS) and SDMA clocks

UART and USB clocks (used for flashing at boot time) are activated at power-up. Their control is not sensitive to warm reset.

- CM_IDLEST_<domain_name> (idle status register): Allows checking that a target module (or subsystem) is in idle mode (that is, idled from a system standpoint) or an initiator module (or subsystem) is in standby mode.

CAUTION

The software must avoid access to a target module in idle mode that cannot answer and may generate an error.

The idle mode for any module depends on the configuration of the CM_FCLKEN_<domain_name> and CM_ICLKEN_<domain_name> registers. Idle mode can be controlled automatically by hardware, depending on the configuration of the CM_AUTOIDLE_<domain_name> register.

For the DSP subsystem, standby mode is reached after the processor performs the IDLE instruction.

The OMAP2420 includes the following idle status registers:

- CM_IDLEST1_CORE: Peripherals, display subsystem
- CM_IDLEST2_CORE: Boot peripherals
- CM_IDLEST_DSP: DSP intrusive port interface, DSP standby status
- CM_IDLEST_WKUP: Peripherals

The IDLE status bit is cleared when the PRCM sends an idle request to the module and is automatically set when the module exits its idle mode.

For initiator modules, the STANDBY status bit is cleared when the module enters standby mode and is automatically set when the module resumes its activity.

❑ CM_AUTOIDLE_<domain_name> (auto-idle register): This register allows enabling/disabling automatic control of the module clock interface. This register holds an AUTO bit per module that belongs to the related power domain.

When activated, the interface clock is automatically managed with the power state transition and is automatically reenabled on wake-up, provided the related bit in CM_ICLKEN_<domain_name> register is set (see Table 5−35).

Table 5−35. Interface Clock Auto-Idle Settings

CM_AUTOIDLE. En_x[1]	CM_ICLKEN.AUTO_x[1]	Interface clock
0	0	Disabled
0	1	Enabled
1	0	Disabled
1	1	Automatic enabling/disabling

1) x represents any module.

The OMAP2420 includes the following auto-idle control registers:

  - CM_AUTOIDLE1_CORE: Peripherals, display subsystem
  - CM_AUTOIDLE2_CORE: Boot peripherals
  - CM_AUTOIDLE3_CORE: Memory controllers, SDMA
  - CM_AUTOIDLE_DSP: DSP Intrusive port interface clock control
  - CM_AUTOIDLE_WKUP: Peripherals

- ☐ CM_CLKSEL_<domain_name> (clock-select register): Controls the module or subsystem input clock frequency selection. It therefore only deals with modules or subsystems for which 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 DPLL output clock (CORE_CLK).

  This register also provides control over DSP synchronizers.

  The OMAP2420 includes the following clock select registers:

  - CM_CLKSEL_MPU: MPU clock selection
  - CM_CLKSEL1_CORE: Display subsystem, L3 and L4 interconnects, peripherals
  - CM_CLKSEL2_CORE: General-purpose timers
  - CM_CLKSEL_DSP: DSP, synchronizers
  - CM_CLKSEL_WKUP: General-purpose timer 1

  To operate a safe and secure clock configuration change, clocks change requests (except GP timers, DDS1, and DSS2 clock source selection) do not take effect until the VALID_CONFIG bit in the PRCM_CLKCFG_CTRL register is set (PRCM_CLKCFG_CTRL[0]). When the configuration change occurs, this bit is automatically cleared.

  Thus, you must change one or more CM_CLKSEL_<domain_name> registers first to set the new clock configuration, and then validate it by setting the VALID_CONFIG bit.

Note:

The CM_CLKSEL2_CORE register and CM_CLKSEL_WKUP register can be changed without setting the VALID_CONFIG bit.

  A new written value in a CM_CLKSEL_<domain_name> register is effectively updated only after the configuration is correctly validated. Any read access before the effective validation returns the previous register values.

- ☐ Clock configuration validation is required for all clocks in the device issued from the DPLL core clock, but does not apply to DSS1, DSS2, and the GP timer functional clock source selection.

- ☐ A change between the DSS1 system clock source and DSS1 divided core clock does not need validation (this applies also to DSS2 clock source changes). Hence, a change to the following registers takes effect immediately:

  - CM_CLKSEL_CORE2.CLKSEL_GPT [12 :2]
  - CM_CLKSEL_WKUP.CLKSEL_GPT1
  - CM_CLKSEL_CORE1.CLKSEL_DSS2

  However, clock configuration validation is required to change the core clock division values of the DSS1 functional clock.

> **CAUTION**
>
> **The software ensures a consistent clock configuration as defined in Section 5.4.3, *Clock Configurations*. Two inconsistent programming cases can occur:**
>
> - **The software requires a clock divider not implemented.**
> - **The software uses correct dividers that generate clock mismatch paths in the device.**
>
> **In the first case, the hardware flags the erroneous programming by setting the CONFIG_STATUS bit in the PRCM_CLKCFG_STATUS register (PRCM_CLKFG_STATUS[0]. Setting the VALID_CONFIG bit in the PRCM_CLKCFG_CTRL register has no effect. The CONFIG_STATUS bit is cleared on the next valid configuration.**
>
> **The second case is a valid configuration from a hardware standpoint (the error flag is not set) but can provoke a dysfunction that can set the device out of order. This is repairable by debug or reset.**

> **Note:**
>
> The DPLL goes to bypass mode on global warm reset. Clock divisions (driven by the CM_CLKSEL_<domain_name> shadow registers) remain unchanged to ensure that modules are not over-clocked during the DPLL transition.
>
> All CM_CLKSEL_<domain_name> registers are reset, but their default values are effective only after the software sets the VALID_CONFIG bit (PRCM_CLKCFG_CTRL[0]) to 1.

- CM_CLKSTCTRL_<domain_name> (clock state control register): Controls the hardware-supervised state transition between the active and inactive states.

  The OMAP242 includes the following clock state control registers:

  - CM_CLKSTCTRL_MPU: MPU subsystem clock
  - CM_CLKSTCTRL_CORE: Display subsystem, L3 and L4 clock domains
  - CM_CLKSTCTRL_DSP: DSP clock domain

  The clock state control register holds one bit per clock region (see Section 5.4.2.2, *Clock Distribution*, for clock domain definition) inside a power domain. If the bit is set, the related clock domain is automatically cut when all module idle and standby conditions are met.

Thus, when the hardware-supervised transition is enabled:

1) The MPU domain clock is automatically cut when the MPU completes the execution of the WAIT FOR INTERRUPT instruction.
2) The DSP domain clock is automatically cut when the DSP completes its IDLE instruction and the DSP subsystem is ready for standby.
3) The L3 domain clock is automatically cut when all initiators are in standby mode (including the MPU and DSP) and slave ports are idled (including L4).
4) The L4 domain clock is automatically cut when all peripherals and slave ports are idled.
5) The display subsystem interface clock is automatically cut when it stops fetching data through the interconnect. The display functional clock is controlled only by the CM_FCLKEN1_CORE register.

Clearing a bit in the CM_CLKSTCTRL_<domain_name> register prevents the related clock domain from being stopped automatically, and a power transition cannot occur.

Setting the FORCESTATE bit field in the PM_PWSTCTRL_<domain_name> register (PM_PWSTCTRL_<domain_name>[1:0]) provides direct software control over a clock domain sleep transition. Clearing the bit wakes up the clock domain.

Programming Examples

The following examples show how to:

- ☐ Control two different module clocks
- ☐ Read the module idle status
- ☐ Manage the DSP intrusive port clock

UART1 Clock Control

- ☐ UART1 uses one fixed functional clock (48 MHz):

1) Select a 48-MHz source (48M_SOURCE bit: CM_CLKSEL1_PLL[3]).
2) If the source is internal, enable the APLL 96-MHz by setting the EN_96M_PLL bit field (CM_CLKEN_PLL[3:2]) or AUTO_96M bit field (CM_CLKEN_PLL[3:2]).
3) Enable the functional clock by setting the EN_UART1 bit in the CM_FCLKEN1_CORE register (CM_FCLKEN1_CORE[21]).
4) Enable the interface clock by setting the EN_UART1 bit in the CM_ICLKEN1_CORE register (CM_ICLKEN1_CORE[21]) and the AUTO_UART1 bit in the CM_AUTOIDLE1_CORE register (CM_AUTOIDLE1_CORE[21]) when automatic control is desired.

Clearing the EN_UART1 bit in the CM_ICLKEN1_CORE register (CM_ICLKEN1_CORE[21]) puts the UART1 in idle mode where it can generate only a wake-up event, not an interrupt or DMA request.

GPT2 Clock Control

General-purpose timer 2 has an asynchronous functional clock that can work with three different sources: system clock (SYS_CLK), external alternate clock (EXTALT_CLK), and 32-kHz clock (FUNC_32K_CLK).

1) Select the clock source with the CLKSEL_GPT2 field in the CM_CLKSEL2_CORE register (CM_CLKSEL2_CORE[3:2]).
2) Enable the functional clock by setting the EN_GPT2 bit in the CM_FCLKEN1_CORE register (CM_FCLKEN1_CORE[4]).
3) Enable the interface clock by setting the EN_GPT2 bit in the CM_ICLKEN1_CORE register (CM_ICLKEN1_CORE[4]) and the AUTO_GPT2 bit in the CM_AUTOIDLE1_CORE register (CM_AUTOIDLE1_CORE[4]) when automatic control is desired.

5.9.4.2 Reset Control

Reset Registers Description

The following registers control reset settings:

- ❑ RM_RSTCTRL_<domain_name> (reset control register): Controls local domain software reset

Most OMAP modules include an individual software reset that can be controlled using the related module SYS_CONFIG configuration register.

The RM_RSTCTRL_<domain_name> register is used for subsystems or modules (DSP) that do not support this local reset or that require specific system control.

In addition, the RM_RSTCTRL_WKUP register handles the global device software reset control.

The OMAP2420 includes the following reset control registers:

- ❑ RM_RSTCTRL_DSP: DSP software reset (set by default)
- ❑ RM_RSTCTRL_WKUP: DPLL and global software reset control auto cleared

The RM_RSTIME_WKUP (reset time) register provides control over reset duration. Two durations are configurable:

- ❑ RSTIME1: Minimum duration (by default, two cycles of the 32-kHz clock) that controls the assertion of the external reset signal used for external devices such as flash memories.

  At power-up, the DPLL and the OMAP2420 control module are reset. This duration also ensures that all power domains are switched on and stabilized on a global software reset. This time can include voltage ramp time.

- RSTIME2: Minimum duration (by default, 16 cycles of the L4 clock) to control power domain reset when domain clocks are on. The default value (16 cycles of the L4 clock) is an optimized value, so the user does not need to modify this value.

Cold and global warm reset are applied during RSTIME1+ RSTIME2. In other cases, when a power domain individually transitions from off to active, reset is applied only during RSTIME2 (plus power domain wake-up time).

RSTIME1 and RSTIME2 can be reprogrammed for different behaviors after initial power-up.

The RM_RSTST_<domain_name> (reset status) register logs any source that has generated a reset in the related power domain. Depending on the domain, several causes of reset can be logged:

- Global device cold reset
- Global device warm reset
- Power domain transition (off to active)
- Core domain transition (because of security firewall configurations, when the core transitions from off to active, the MPU and DSP power domains are reset.

The RM_RSTST_WKUP register also logs all causes of global warm reset (external reset and MPU watchdog reset), as well as the DPLL reset.

The OMAP2420 includes the following reset status registers:

- RM_RSTST_MPU
- RM_RSTST_DSP
- RM_RSTST_WKUP

The bit for each of these registers is set on the effective release of the respective reset signal.

CAUTION

Reset status bits must be cleared by software.

Programming Examples

On power-up, the PRCM automatically directs all power domains to active and applies a global reset on the device during RSTIME1 + RSTIME2. Release of reset depends on the default values set in the RM_RSTCTRL_<domain_name> registers:

- The MPU, CORE, and WKUP power domains automatically exit from reset.

- ☐ The DSP power domain remains under reset. The MPU controls DSP reset deassertion to manage start-up.

For the DSP reset sequence, the MPU must:

1) Enable the DSP clock (CM_FCLKEN_DSP[0] and CM_ICLKEN_DSP[1])
2) Release the RST2_DSP bit of the RM_RSTCTRL_DSP register (RM_RSTCTRL_DSP[1]) to make the IPI and MMU functional
3) Download the boot code in DSP memories and configure the MMU
4) Release the RST1_DSP bit of the RM_RSTCTRL_DSP register (RM_RSTCTRL_DSP[0]) to make UMA and DMA functional

After this sequence completes, the DSP can boot and configure the DMA.

5.9.4.3 Wake-Up Control

Wake-Up Registers Description

The following registers control the wake-up settings:

- ☐ PM_WKEN_<domain_name> (wake-up enable register): Allows enabling or disabling the wake-up of the related power domain on a module or subsystem wake-up event.

  The wake-up enable register applies only to modules that can generate wake-up events.

  The OMAP2420 includes the following wake-up enable registers:

  - ■ PM_WKEN1_CORE: Peripheral wake-up control
  - ■ PM_WKEN2_CORE: UART1 and USB modules wake-up control
  - ■ PM_WKEN_WKUP: GPT1 and GPIO wake-up control

Note:

The four GPIO wake-up signals are ORed and therefore controlled as a unique signal by the the PRCM module.

Wakeups issued from the two processor (MPU and DSP) interrupt controllers are not maskable in the PRCM module. However, they cannot be generated if corresponding interrupts are masked in the interrupt controllers.

Wake-ups issued from the DSP intrusive interface are not maskable in the PRCM module.

- ☐ PM_WKST_<domain_name> (wake-up status register): Logs all module wake-up events; holds one wake-up status bit per module with wake-up capability in the domain.

  The OMAP2420 includes the following wake-up status registers:

  - ■ PM_WKST1_CORE: Peripherals wake-up status
  - ■ PM_WKST2_CORE: UART3 and USB modules wake-up status
  - ■ PM_WKST_WKUP: GPT1 and GPIOs wake-up status.

> **This register must be cleared by software. No clear prevents a sleep transition of the related clock domain.**
>
> CAUTION

❑ PM_WKDEP_<domain_name> (wake-up dependency register): Allows enabling or disabling the wake-up of the domain on another domain wake-up; used to set dependency between domains.

The OMAP2420 includes the following wake-up dependency registers:

- PM_WKDEP_MPU: Dependency with CORE, DSP, and WKUP power domains
- PM_WKDEP_CORE: Dependency with MPU, DSP, and WKUP power domains
- PM_WKDEP_DSP: Dependency with MPU, CORE, and WKUP domains

The WKUP power domain is always alive and has no wake-up dependency control.

Programming Examples

The software must program the PM_WKEN_<domain_name> register to ensure that the device can always wake up from idle mode.

The wake-up sequence behaves as follows:

1) A peripheral (in either the core or the wake-up domain), the DSP intrusive port, the MPU event generator, or an interrupt controller (MPU or DSP) generates a wake-up event.
2) If enabled in the corresponding PM_WKEN_<domain_name> register, this event wakes up the related domain. First, it ensures that the domain is switched on and then activates all of domain clock interfaces, the peripheral functional clock, or the processor clock (if the wake-up issues from the MPU event generator or an interrupt controller). This presupposes that clock sources (PLLs) are also waked up.
3) According to the programming of the PM_WKDEP_<domain_name> register, one or more domains can be awakened simultaneously.

The software must then enable all required functional clocks in the CM_FCLKEN_<domain_name> registers and clear the PM_WKST_<domain_name> registers.

When awakened, the system operates as follows:

1) If the wake-up is issued from the DSP intrusive port, the MPU accesses DSP memory.

2) If the wake-up is issued from an interrupt controller, the concerned processor (MPU or DSP) processes the pending interrupt.

3) If the wake-up is issued from the MPU event generator, the MPU processes the dedicated interrupt generated by the PRCM when it is enabled in the PRCM_IRQENABLE_MPU register.

4) If the wake-up is issued from a peripheral, an interrupt generation or a DMA request follows the wake-up event and is sent to the concerned processor (MPU or DSP) or to a DMA (dDMA or sDMA).

   Three kinds of interrupts can occur:

   - Module wake-up interrupt (interrupt related to the wake-up event)
   - Module functional interrupt (not directly related to the wake-up event, generated, for example, when a FIFO is full)
   - PRCM interrupt (which informs the processor that a wake-up event occurred)

   The software must activate only one of the three interrupts and mask the others.

5.9.4.4 Event Generator Control

Event Generator Registers

The following registers apply only to the ARM1136 processor:

- ❑ PM_EVGENCTRL_MPU (event generator control register): Provides control of event generator settings
- ❑ PM_ EVGENONTIM_MPU (event generator on-time register): Allows the on duration of the event generator to be set
- ❑ PM_ EVGENOFFTIM_MPU (event generator off-time register): Allows the off duration of the event generator to be set

Note:

The event generator on-time interrupt is generated immediately after the ENABLE bit PM_EVGENCTRL_MPU[0] is set.

Programming Examples

The event generator feature allows control of a mode in which the MPU power domain is switched alternately on and off (or dormant mode). This implements an efficient activity modulation of the MPU power state with minimum support from software. The vent generator must not be used to put the core domain in off mode, because the event generator needs the system clock to be active (GPT1 must be used in that case).

When the event generator is activated, the PRCM ensures that the core domain always follows MPU domain activity. Consequently, the EN_MPU bit in the PM_WKDEP_CORE register (PM_WKDEP_CORE[1]) is always set to 1 and is read-only.

The event generator is a counter that can be loaded successively with two distinct values; on and off durations (number of system clock cycles) are set in the PM_EVGENOFFTIM_MPU and PM_ EVGENOFFTIM_MPU registers.

The PM_EVGENCTRL_MPU register allows enabling/disabling the event generator feature (enable bit: PM_EVGENCTRL_MPU[0]), and controls how to load on and off values in the counter (ONLOADMODE and OFFLOADMODE fields: PM_EVGENCTRL_MPU[2:1] and PM_EVGENCTRL_MPU[4:3]).

There are three ways to load the counter with the next counting value:

- Load the update of the corresponding PM_EVGENONTIM_MPU or PM_EVGENOFFTIM_MPU register.
- Load the on-time value when the MPU exits standby mode and load the off-time value when the MPU completes its WAIT FOR INTERRUPT instruction).
- Auto-load the on-time value when the off time expires, and auto-load the off-time value when the on time expires.

The system uses internal ENDONTIME and ENDOFFTIME signals to generate interrupt and/or wake-up signals as follows:

- The assertion of the ENDONTIME signal must result in the generation of an interrupt to the processor. For example, when a counter load is driven by the MPU standby status, the software can use this interrupt to initiate the WAIT FOR INTERRUPT instruction.
- The assertion of the ENDOFFTIME signal must result in a wake-up event to the domain wake-up controller. When the wake-up transition completes and the processor clock restarts, an interrupt must also be generated to the processor.
- The PRCM_IRQENABLE_MPU register manages interrupt masking.

5.9.4.5 Power-Management Control

Power-Management Registers

The following registers provide control of power-management settings:

- PM_PWSTCTRL_<domain_name> (power state control register): Allows control of the power state transition of the domain and holds the following bit fields:
    - POWERSTATE: Allows selection of the required power state (on, retention, and off) for the next transition
    - LOGICRETSTATE: Allows specifying whether the logic is retained or switched off when the domain goes in retention

Note:

This feature is domain-dependent and not programmable (read-only).

- MEMRETSTATE[1 to X]: Allows specifying whether the memory block is retained or switched off when the domain goes to retention

Note:

This feature is domain-dependent and not programmable (read-only).

- MEMONSTATE[1 to X]: Allows specifying whether the memory block in domain is on, retained, or switched off when the domain is on

Note:

This feature is domain-dependent and not programmable (read-only).

- MEMORYCHANGE: Allows updating the memory state according to the MEMONSTATE[1 to X] setting
- FORCESTATE: Allows starting a domain transition as soon as all conditions are met in the domain, regardless of the dependency of other domains

  When the bit is set, the software must first ensure that all conditions are met to start the transition.

  When a domain is idled, clearing this bit provokes a wake-up transition, and an associated interrupt can be generated to the MPU.

The OMAP2420 includes the following power state control registers:

- PM_PWSTCTRL_MPU
- PM_PWSTCTRL_CORE
- PM_PWSTCTRL_DSP

☐ PM_PWSTST_<domain_name> (power state status register): Provides the status of the power state transition of the domain and holds the following bit fields:

- POWERSTATEST: Indicates the current power domain state

  For the MPU power domain, this field is not applicable (because the MPU cannot process any read access in the off and retention modes).
- LASTSTATEENTERED: When returning to active mode, the software can use this bit field to identify from which state it returns (retention or off) and thus which wake-up procedure it must use to execute (memory restore, configuration restore, complete or partial boot, etc.).

  This bit field is defined for the MPU and DSP power domains and is updated when the wake-up transition occurs.
- MEMSTATE[1 to X]St: Indicates the current memory block power state

- CLKACTIVITY: The software can check this bit for clock activity in the domain. If there is clock activity, a power transition cannot be operated.
- INTRANSITION: The software can poll this bit for the end of the power-domain state transition.

The OMAP2420 includes the following power-state status registers:

- PM_PWSTST_MPU
- PM_PWSTST_CORE
- PM_PWSTST_DSP

Programming Examples

- ☐ **Core Memories Power-State Control**

The following sequence describes how to control the memory-blocks power state in the core domain when in active mode. In the OMAP2420, core memory is split into three blocks: secure RAM (block 1), frame buffer (block 2), and frame buffer (block 3).

This example uses only half of the frame buffer (typically for display low-power refresh mode). This example assumes that the POWERSTATE field is on in the PM_PWSTCTRL_CORE register (PM_PWSTCTRL_CORE[1:0]).

1) Program the MEM3ONSTATE bit to off or retained (if required) in the PM_PWSTCTRL_CORE register (PM_PWSTCTRL_CORE[15:14]).
2) Set the MEMORYCHANGE bit in the PM_PWSTCTRL_CORE register (PM_PWSTCTRL_CORE[20]) to validate the memory state transition.
3) The change occurs only when the L3 clock domain is idled.

- ☐ **DSP Power-Domain Transition**

The following sequence describes how to operate a DSP power-domain state transition from active to off:

1) Program the POWERSTATE field to off in the PM_PWSTCTRL_DSP register (PM_PWSTCTRL_DSP[1:0]).
2) The transition occurs when the DSP subsystem is in standby mode, which is achieved after the DSP executes its IDLE instruction.

At this stage, if there is no pending interrupt or MPU access bits in the CM_CLKSTCTRL_DSP register set (CM_CLKSTCTRL_DSP[0] and CM_CLKSTCTRL_DSP[8]), the PRCM stops the DSP clocks.

3) The MPU can poll the INTRANSITION bit in the PM_PWSTST_DSP register (PM_PWSTST_DSP[20]) to know when the transition state is-complete. The MPU can also poll the STANDBY_DSP bits in the CM_IDLEST_DSP register (CM_IDLEST_DSP[0] and CM_IDLEST_DSP[8]) to track standby-mode assertion.

Note:

There is no MPU software control of the DSP memory power states, which are automatically supervised by hardware (according to the power state setting) and independently managed by the DSP software for DSP memory.

If at least one of the AUTOSTATE_DSP bits is cleared, the transition cannot occur until the software sets the FORCESTATE bit in the PM_PWSTCTRL_DSP register (PM_PWSTCTRL_DSP[18]).

❑ **MPU Power Domain Transition**

The following sequence describes how to operate two different MPU power-domain state transitions from active to retention modes.

Active to Full Retention

1) Program the POWERSTATE field to retention in the PM_PWSTCTRL_MPU register (PM_PWSTCTRL_MPU1:0]).
2) Program the LOGICRETSTATE bit in the PM_PWSTCTRL_MPU register (PM_PWSTCTRL_MPU [2]) so the logic remains retained.
3) The transition occurs when the MPU power domain goes to standby mode. after it executes its WAIT FOR INTERRUPT instruction. At this stage, if there is no pending interrupt and if the AUTOSTATE_MPU bit in the CM_CLKSTCTRL_MPU register (CM_CLKSTCTRL_MPU[0]) is set, the PRCM stops the MPU clocks.
4) The MPU power domain enters full retention mode (retaining logic and cache memory).

Active to Dormant Mode

1) Program the POWERSTATE field to retention in the PM_PWSTCTRL_MPU register (PM_PWSTCTRL_MPU[1:0]).
2) Program the LOGICRETSTATE bit in the PM_PWSTCTRL_MPU register (PM_PWSTCTRL_MPU[2]) so that the logic goes off in retention mode.
3) The transition occurs when the MPU domain goes to standby mode after it executes its WAIT FOR INTERRUPT instruction.

   At this stage, if there is no pending interrupt and if the AUTOSTATE_MPU bit in the CM_CLKSTCTRL_MPU register (CM_CLKSTCTRL_MPU[0]) is set, the PRCM stops the MPU clocks.
4) The MPU domain enters dormant mode (logic is off while cache memory is retained).

5.9.5 Global Registers

5.9.5.1 External Voltage Control

❑ PRCM_VOLTSETUP (voltage-source setup time register): Allows control of the setup time of the main V_{DD} power supply. This constant is power IC-dependent.

At power-up reset, this register is not used, because the external power supply must be stable. After boot, the software must set the correct value to ensure correct sequences when performing voltage scaling or sleep transitions.

Note:

The validation of the voltage (PRCM_src_volt_ok observability signal) comes after the value in the register is read, plus a delay equal to five cycles of the 32-kHz clock.

- ❑ PRCM_VOLTCTRL (voltage source control register): Allows control of the main V_{DD} supply; can act directly on the external power IC.

The OMAP2420 main V_{DD} supplies the five power domains. The I/O domain is powered independently. The PLL domain can be connected to the main V_{DD} or be supplied independently.

Main V_{DD} can scale between low-voltage point and high-voltage point.

The PRCM allows predefining of two different voltage levels (L0 and L1) and drives externally the VMODE signal that informs the power IC which level is required. Programmed by the MPU through a serial command (for example, SPI port), the power IC can allocate any voltage value to L1 and L0 (for example, L1 = 0 V, L0 = high voltage, or L1 = low voltage and L0 = high voltage, etc.). L0 is assumed to be the maximum voltage.

Thus, device voltage can be L0 or L1 and is set with the VOLT_LEVEL bit. The VMODE signal is updated either automatically when all internal power domain transitions are complete or immediately on software request, depending on the FORCE_EXTVOLT bit setting (when using this bit, the software ensures that all conditions are met to operate a safe and correct voltage change).

In addition, the PRCM associates a voltage level with two predefined device system modes:

- ❑ Retention voltage: Desired voltage when either all power domains are in the retention state or at least one domain is in retention and the others are off. It can be L0 or L1 and is set with the SETRET_LEVEL bit.
- ❑ Off voltage: Desired voltage when all power domains are in the off state. It can be L0 or L1 and set with the SETOFF_LEVEL bit.

These two voltage modes can be automatically controlled when setting the AUTO_EXTVOLT bit. If cleared, this bit forces the main V_{DD} to the L0 voltage.

When the AUTO_EXTVOLT bit is set, voltage is automatically controlled and scaled according to SETRET_LEVEL and SETOFF_LEVEL settings when all power domains are in retention or off state.

VMODE polarity can be controlled by the EXTVOL_POL bit (PRCM_POLCTRL register) when the VMODE signal is not used at power-up.

- ❑ PRCM_VOLTST (VOLTAGE STATUS REGISTER): Reading this register informs the software of the current voltage level: L0 or L1.

Note:

If the FORCE_EXTVOLT bit PRCM_VOLTCTRL[14] = 1, regardless of the setting of the AUTO_EXTVOLT bit PRCM_VOLTCTRL[15] (0 or 1), the VMODE signal always behaves according to the VOLT_LEVEL field PRCM_VOLTCTRL[1:0] setting (for a fixed polarity of the VMODE signal, that is, the EXTVOL_POL bit PRCM_POLCTRL[0] is not changed).

5.9.5.2 Clock Source Control

- ❑ PRCM_CLKSRC_CTRL (clock source control register): Dedicated to clock source control of the OMAP2420, through three mechanisms:
  - ■ Source of the system clock (external square clock input or oscillator)
  - ■ Automatic oscillator or CLKREQ. This signal informs the external system (for example, a power IC) of the OMAP2420 system clock requirements and can be kept always asserted or deasserted automatically only when all power domains are off (or off or in retention).
  - ■ The input divider (1, 2) of the system clock

Deasserting CLKREQ stops the system clock input (if the external clock source is connected) or causes the oscillator (if quartz connected) to go to bypass mode. This saves power, but requires the longest latency (for example, oscillator stabilization) to wake up the system.

After power-up, programming the PRCM_CLKSRC_CTRL register disables the oscillator (unless the command specifically enables it).

The SYSCLKSEL bit field default value is 0x3. To ensure predictable behavior, use the value only for design boot. Do not use the value after the device is booted or for an additional application.

The software ensures correct device behavior and must set one of two effective configurations for the SYSCLKSEL bitfield: 0x0 or 0x1.

- ❑ PRCM_CLKSSETUP (clock source setup register): Allows setting the setup time of the oscillator system clock based on the number of 32-kHz clock cycles. This duration corresponds to the time needed by the oscillator to be stable before propagating the system clock in the OMAP2420.

  This duration is not taken at device power-up because the stabilization time duration is controlled by the external system. It is reset to 0 and must be programmed before the first internal power state transition.

Note:

The validation of the clocking (PRCM_clk_ok observability signal) comes after the value in the register is read, plus a delay equal to nine cycles of the 32-kHz clock.

- PRCM_CLKCFG_CTRL (clock configuration control register): Allows correct initiation and synchronization of a device clock configuration change. An update of one or more CM_CLKSEL_<domain_name> registers is effective only when the software sets the VALID_CONFIG bit (PRCM_CLKCFG_CTRL[0]) to 1.

  On assertion of this bit, the PRCM can safely synchronize all new clock ratios and align clock phases. This allows clock scaling to be operated on-the-fly without requiring the system to be idled.

  Clock configuration validation is not required when selecting DSS1, DSS2, or GP timer clock sources.

  Reading the CM_CLKSEL_<domain_name> registers before validation returns previous values. The same action returns the new values after the VALID_CONFIG bit is set.

- PRCM_CLKCFG_STATUS (clock configuration status register): Handles an error flag, CONFIG_STATUS bit, which is set when the CM_CLKSEL_<domain_name> registers so that non-implemented clock dividers are required. This can happen when the software does not configure a validated clock setting.

  Setting the CONFIG_STATUS bit (PRCM_CLKCFG_STATUS[0] = 1) prevents the PRCM_CLKCFG_CTRL CLOCK_VALID bit (PRCM_CLKCFK_CTRL[0]) from being set. This means that while CONFIG_STATUS is set, the clock setting that caused the error flag is blocked by the hardware. The CONFIG_STATUS bit is automatically cleared on the next valid clock configuration.

> **CAUTION**
>
> **Software checks the CONFIG_STATUS bit after any clock configuration change. It is strongly recommended that the PRCM_CLKCFG_CTRL register VALID_STATUS bit be set only when CONFIG_STATUS has been checked.**

- PRCM_CLKOUTCTRL (clock out control register): Provides control over the two OMAP2420 external output clocks

  The OMAP2420 generates two output clocks that can be used externally for functionality or testing. The following settings allow control of the clocks:

  - Selection of the clock source (DPLL output clock, system clock, 54-MHz clock, or 96-MHz clock)
  - Division of the clock source by 1, 2, 4, 8, or 16
  - Enabling/disabling the output clocks

When the clocks are disabled, their static polarity depends on the CLOCKOUT_POL and CLOCKOUT2_POL bits in the POLCTRL[10] and PRCM_POLCTRL[9] registers.

5.9.5.3 Polarity (Voltage Status) Register

- PRCM_POLCTRL (voltage status register): Allows setting the polarity of the four external signals controlled by the PRCM:
  - SYS.nVMODE
  - SYS.CLKREQ

    At device power-up, reset values are SYS.nVMODE = 1 and SYS.CLKREQ = 1 (both asserted high). According to system implementation, these signals can be used or not used during device power-up.

    In addition, the voltage status register controls the polarity of the output clocks when they are inactive:
  - SYS.CLKOUT (reset at power-up only)
  - SYS.CLKOUT2 (reset at power-up only)

The voltage status register is reset at power-up only.

5.9.5.4 General-Purpose Registers

- GENERAL_PURPOSE [1 to 20]: These 20 registers can be used as a retention memory. For example, before the CORE power domain goes to off mode, the SDRC configuration can be saved on the general-purpose registers. Then, when the CORE power domain wakes up, the MPU can restore quickly the SDRAM configuration and operate a fast boot.

5.9.6 Clock Generator Registers

- CM_CLKEN_PLL (PLL enable register): Provides software control of the three PLLs:
  - The APLLs can be either stopped or enabled (lock mode).
  - The DPLL can be set in low-power bypass (long relock latency), fast-relock bypass (short relock latency), or enabled (lock mode). In bypass modes, the PLL outputs the reference clock (REF_CLK) divided by (N+1), where N is the DPLL divider.
- CM_IDLEST_CKGEN (source-clock idle status register): Documents activity of all master clocks issued from the PLLs:
  - 54-MHz clock activity: Indicates when the APLL is locked
  - 96-MHz clock activity: Indicates when the APLL is locked
  - 12-, 48-, and 96-MHz clock activity
  - DPLL output clock is REF_CLK, DPLL-locked clock, or 32 kHz (low clock).
- CM_AUTOIDLE_PLL (PLL auto-idle register): Allows enabling/disabling of automatic control of PLL activity. If required, the PLLs can be programmed to stop if all reporting clocks are stopped. The DPLL can automatically switch to bypass mode (fast relock or low power) when the

CORE_CLK is unused and switch back to locked mode when a reporting clock is required. APLLs are also automatically stopped/enabled according to clock requirements.

- CM_CLKSEL1_PLL (source clock selection register): Provides selection control of the APLL clock sources, selects the CORE_CLK source, and allows setting the DPLL multipliers and dividers to generate the required CORE_CLK.
  - 54-MHz clock can issue from the internal 54-MHz APLL or the EXTALTCLK clock input.
  - 12-MHz and 48-MHz clocks can issue from the internal 96-MHz APLL or from the EXTALTCLK clock input.
  - 2-bit field allows setting the DPLL divider value (0 to 15) and the multiplier value (0 to 1023). When set to 0 or 1, the DPLL goes to bypass mode.
  - This register is reset on power-up only.
- CM_CLKSEL2_PLL (source clock selection register): Handles one field that allows selection of the DPLL-locked output clock, DPLL-locked output clock × 2, or 32 kHz to provide the CORE_CLK.

  The switch between the DPLL clock output and DPLL clock output X2 allows quick frequency scaling and is intended to be used with voltage scaling; the software must select the DPLL clock output when it requires entering a low-power (lp) clock configuration. At high-voltage point, the DPLL clock output X2 must be selected, to reach maximum performance.

  This register is reset on any global reset.

5.9.7 MPU Clock Domain Management

5.9.7.1 Clocks Summary

The MPU clock domain includes the following clocks, which are controlled together:

- MPU_FCLK
- MPU_ICLK
- INT_M_FCLK
- INT_M_ICLK

5.9.7.2 Sleep Sequence

Processing an MPU clock-domain sleep sequence requires the following conditions:

- MPU has executed its STANDBYWFI instruction
- CM_CLKSTCTRL_MPU register AUTOSTATE bit is enabled.

The MPU clock is never shut down when the AUTOSTATE bit is cleared.

Note:

When the MPU clock is effectively turned off, the MPU domain is ready for power domain transition (to either retention or off mode).

5.9.7.3 Wake-Up Sequence

Use the following wake-up events to activate the MPU clock:

- An interrupt sent to the MPU (if the CORE power domain is on)
- The event generator counter expired (ENDOFTIME event)
- Voltage ramp complete (voltage setup time has expired)
- CORE power domain dependency wakeup (CORE power domain has awakened)
- WKUP power domain dependency wakeup (a wake-up is issued from the WKUP power domain).
- DSP power domain dependency wakeup (DSP power domain wakeup).
- Global warm reset

Notes:

1) The MPU interrupt controller clock is gated with the MPUSS clock.
2) When the MPUSS clock is active, any unmasked interrupt causes the MPU to wake up.

CAUTION

After a wake-up transition, a new sleep transition is possible only if the MPU is active for at least one SLEEP_CLK cycle (12, 13, or 19.2 MHz) duration.

5.9.7.4 Clock Selection

The MPU clock frequency is selectable by writing in the CM_CLKSEL_MPU bit field register (CM_CLKSEL_MPU[4:0]). The MPU frequency can be equal to CORE_CLK/[1, 2, 4, 6, or 8].

Note:

The CORE_CLK value depends on the DPLL programming (M and N values in the CM_CLKSEL1_PLL register) and on DPLL output selection (CM_CLKSEL2_PLL). See Section 5.4.3, *Clock Configurations*, for clock configuration settings.

Clock frequency change occurs only after validation of the VALID_CONFIG bit in the PRCM_CLKCFG_CTRL register (PRCM_CLKCFG_CTRL[0]).

5.9.8 DSP Clock Management

5.9.8.1 Clock Summary

The DSP clock domain includes the following clocks, which are controlled together:

- DSP_FCLK
- DSP_ICLK
- INT_D_FCLK
- INT_D_ICLK

5.9.8.2 Sleep Sequence

Either of two modes can shut down the DSPSS clock:

- Auto-control mode

  The clock shuts down when the following events occur:

  - The CM_CLKSTCTRL_DSP register AUTOSTATE_DSP bit is set to 1.
  - The CM_AUTOIDLE_DSP register AUTO_DSP_IPI bit is set to 1.
  - The CM_FCLKEN_DSP register EN_DSP bit and the CM_ICLKEN_DSP register EN_DSP_IPI bit are set to 1.
  - The DSP executes its IDLE instruction.

  From a system standpoint, auto-control mode is used when the DSP clock is required to be automatically cut as soon it is in standby mode, and then automatically activated on a wake-up event.

- Force mode

  The clock shuts down when Case A or Case B occurs:

  Case A

  - The CM_AUTOIDLE_DSP register AUTO_DSP_IPI bit is set to 1.
  - The DSP executes its IDLE instruction.
  - The CM_FCLKEN_DSP register EN_DSP bit and the CM_ICLKEN_DSP register EN_DSP_IPI bit remain set.
  - The PM_PWSTCTRL_DSP register FORCESTATE bit is set to 1.

  Case B

  - The CM_AUTOIDLE_DSP register AUTO_DSP_IPI bit is set to 0.
  - The DSP executes its IDLE instruction.
  - After the DSP is in standby mode, the CM_FCLKEN_DSP register EN_DSP bit and CM_ICLKEN_DSP register EN_DSP_IPI bit are set to 0.
  - The PM_PWSTCTRL_DSP register FORCESTATE bit is set to 1.

  From a system standpoint, this configuration is set when the device is in a functional mode where the DSP is not required. The MPU disables the DSP.

  When DSP activity is required, the MPU wakes up the DSP by clearing the PM_PWSTCTRL_DSP register FORCESTATE bit. In case B, the wakeup does not reactivate the DSP clock.

Notes:

1) The DSP interrupt controller clock is gated with the DSPSS clock.
2) When the DSP clock is effectively off, the DSP domain is ready for power domain transition (to retention or off mode).
3) The DSPSS standby status is logged in the CM_IDLEST_DSP register (ST_DSP bit). This bit is cleared when the DSP is active.
4) DSP domain clock activity (including the DSP) status is logged in the PM_PWSTST_DSP register (clock activity bit).

The software must not clear the EN.DSP bit while the DSP clock is running, because unpredictable results can occur.

WARNING

5.9.8.3 DSP Intrusive Port Interface Control (IPI)

Whenever the DSP IPI is unused while the DSP core is running, the IPI can be idled by clearing the CM_ICLKEN_DSP.EN_DSP_IPI bit (CM_ICLKEN_DSP[1]). In this case, the part of the DSPSS is forced in idle but the DSP SS clocks remain ON. This action allows internal power savings in DSPSS. The combination of the AUTO_DSP_IPI (CM_AUTOIDLE_DSP[1]) and EN_DSP_IPI (CM_ICLKEN_DSP[1]) bits provide the control listed in Table 5−36.

Table 5−36. Auto-Idle and EN.PI Bits Combinations

Auto-Idle	EN.IPI	Actin
1	1	This mode allows automatic control of the IPI idle mode.
1	0	Forbidden by software rules
0	1	The DSPSS clock is kept active. Domain transition is not possible.
0	0	IPI is forced in idle mode.

The DSPSS IPI idle status is logged in the CM_IDLEST_DSP register ST_IPI bit (CM_IDLEST_DSP[1]).

Note:

When the clock is on and the DSP is running, the MPU is not intended to access the IPI it idled (EN_DSP_IPI bit cleared).

5.9.8.4 Wake-Up Sequence

If the EN_DSP and/or EN_DSP_IPI bits are enabled, the DSPSS clock can be activated in the following ways:

- ❑ Generating FORCE WAKEUP by clearing FORCESTATE

- Sending an interrupt to the DSP (The CORE power domain must be on.)
- The MPU accessing the DSP subsystem
- A CORE power domain dependency wakeup
- An MPU power domain dependency wakeup
- WKUP power domain dependency wakeup (wakeup issued from the WKUP power domain)

Note:

1) On wakeup, IPI idle is maintained, if EN.IPI = 0.
2) When the DSPSS clock is active, an unmasked interrupt or DSP reset causes a DSP processor wakeup (that is, it exits standby mode and starts running).

CAUTION

Wake-up dependencies that wake up the DSP domain impact the DSP. A hardware restriction prevents the full DSP power domain from returning to idle unless the DSP subsystem is reactivated after the wakeup.

CAUTION

After a wake-up transition, a new sleep transition is possible only if the DSP is active at least during a SLEEP_CLK cycle (12, 13, or 19.2 MHz) duration.

5.9.8.5 Clock Selection and Synchronization

The DSP subsystem contains two clock regions:

- UMA top (and DSP interrupt controller)
- DSP I/F (remaining DSP subsystem: DMA, MMU, IPI)

The UMA clock frequency is selected using the CLKSEL_DSP field in the CM_CLKSEL_DSP register (CM_CLKSEL_DSP[4:0]).

The UMA frequency can be equal to CORE_CLK/[1, 2, 3, 4, 6, 8, or 12].

The DSP I/F clock frequency is a divided (by 1 or 2) value of the UMA clock frequency division. The division value is selected using the CLKSEL_DSP_IF field in the CM_CLKSEL_DSP register (CM_CLKSEL_DSP[6:5]).

Depending on the ratio (only 1 and 2/3 are supported) between the DSP I/F and L3 clocks, the software can be required to disable or enable the DSP synchronizer to ensure correct communication between the DSP and interconnects. The clock selection rule is shown in Table 5−37.

Table 5−37. Clock Selection Rule

[DSP I/F Clock]/[Core L3 Clock] Ratio	Synchronizer
2/3	Enabled
1	Disabled (in bypass)

Synchronizer control is managed through the CM_CLKSEL_DSP register SYNC_DSP bit (CM_CLKSEL_DSP[7]).

Notes:

1) The CORE_CLK value depends on DPLL programming (M and N values in CM_CLKSEL1_PLL) and DPLL output selection (CM_CLKSEL2_PLL register). See Section 5.9.13, *DPLL Programming*, for details.
2) The CORE_L3_CLK is defined by the CM_CLKSEL1_CORE register CLKSEL_L3 bit field (CM_CLKSEL1_CORE[4:0].
3) See Section 5.4.3, *Clock Configurations*, for clock configuration settings.
4) A clock frequency change occurs only after validation of the VALID_CONFIG bit in the PRCM_CLKCFG_CTRL register (PRCM_CLKCFG_CTRL[0]).

5.9.9 Display Subsystem Clock Management

5.9.9.1 Clocks Summary

The PRCM provides three functional clocks and two interface clocks to the DSS (display subsystem):

- ❑ DSS_CLK1
- ❑ DSS_CLK2
- ❑ DSS_54M_CLK
- ❑ DSS_L3_ICLK
- ❑ DSS_L4_ICLK

The same gating controls DSS_L3_ICLK and DSS_L4_ICLK. All other clocks are independently gated.

5.9.9.2 Sleep Sequence

Use the auto-control mode to shut down the DSS clocks. The DSS interface clock shuts down when the following events occur:

- ❑ The CM_CLKSTCTRL_CORE register AUTOSTAT_DSS bit is set to 1.
- ❑ The CM_ICLKEN1_CORE register EN_DSS bit is set to 1.
- ❑ The CM_AUTOIDLE1_CORE register AUTO_DSS bit is set to 1.
- ❑ The DSS DMA is in standby mode.
- ❑ The MPU is in standby mode.

From a system standpoint, this configuration is set when the DSS clocks are required to be automatically cut when the DSS is in standby mode, and then automatically activated on a wake-up event.

Notes:

1) Clocks can shut down manually when EN_DSS1 = 0, EN_DSS2 = 0, EN_DSS = 0, and EN_TV=0 (the display feature is disabled). However, to complete the DSS clock domain sleep transition, the CM_CLKSTCTRL_CORE register AUTOSTAT_DSS bit must be set to 1, and the DSS DMA and MPU must be in standby mode.
2) The EN_TV bit has an immediate effect: a clearing/setting stops/activates the DSS_54M_CLK.
3) There is no automatic control over functional clocks DSS_CLK1 and DSS_CLK2.
4) The DSS standby status is logged in the CM_IDLEST1_CORE register ST_DSS bit.

The combination of AUTOSTATE and AUTOIDLE bits provides the control listed in Table 5−38.

Table 5−38. AUTOSTATE and AUTOIDLE Bits

AUTOSTATE	AUTOIDLE	Action
1	1	DSS clock domain automatic sleep transition is allowed.
1	0	DSS domain clock sleep transition occurs only if EN_DSS = 0.
0	X	DSS domain clock sleep transition cannot occur.

5.9.9.3 Wake-Up Sequence

If the EN_DSS1 or/and EN_DSS2 or/and EN_DSS are enabled, the DSS clocks can be activated when the MPU domain exits standby mode.

Note:

When EN_DSS = 1, only the DSS interface clock is enabled. Software enables the DSS_CLK1 and DSS_CLK2 functional clocks.

CAUTION

After a wake-up transition, a new sleep transition is possible only if DSS DMA is active for the duration of at least one SLEEP_CLK cycle (12, 13, or 19.2 MHz).

5.9.9.4 Clock Selection

The DSS clock configuration includes the following:

- Two configurable functional clocks (DSS_CLK1 and DSS_CLK2)
- One unconfigurable interface clock (DSS_L3_ICLK)
- One fixed functional clock (DSS_54M_CLK)

To select the DSS1 clock frequency, write in the CM_CLKSEL1_CORE register CLKSEL_DSS1 field. The DSS1 clock frequency can be equal to the system clock or to CORE_CLK/[1, 2, 3, 4, 6, 8, or 12].

To select the DSS2 clock frequency, write in the CM_CLKSEL1_CORE register CLKSEL_DSS2 field. The DSS2 clock frequency can be equal to the system clock or the 48-MHz clock.

The I/F clock (DSS_L3_ICLK) frequency is equal to the L3 clock (CORE_L3_ICLK).

The TV out clock (DSS_54M_CLK) is fixed to 54 MHz.

Notes:

1) The CORE_CLK value depends on DPLL programming (M and N values in CM_CLKSEL1_PLL) and on the DPLL output selection (CM_CLKSEL2_PLL).
2) CORE_L3_ICLK is defined in the CM_CLKSEL1_CORE register CLKSEL_L3 field.
3) A DSS1 clock-frequency division change occurs only after validation in the PRCM_CLKCFG_CTRL register VALID_CONFIG field.

5.9.9.5 Display Low-Power Refresh Mode

Introduction

The display low-power refresh mode minimizes power consumption while keeping the display state active. Only the wake-up and core domains are active in this mode. The MPU is either off or dormant. The DSP domain is off. Only the graphics memory is refreshed. The frame buffer is in internal SRAM.

Two low-power display modes can be defined:

- Still-picture low-power refresh (display content does not change):
  - The MPU is either off or dormant.
  - The frame buffer (FB) is internal.
  - Unused SRAM is kept in retention.
  - Only the DSS is active.
  - The L3 interconnect clock is activated only when the DSS accesses the FB.
  - The DPLL stays in bypass mode.
- Screen saver (still background plus animated small picture):
  - The MPU regularly wakes up to modify the display content. When unused, the MPU is either in inactive or dormant mode.
  - The DSS is always active, and the MPU is active for a small percentage of time.
  - The interconnect clock is activated during DSS accesses and during MPU wake-up periods.
  - The DPLL locks when the MPU wakes up, then switches back to bypass mode.
  - A timer is active.

Clocking Scheme

The two functional clock inputs (DSS_CLK1 and DSS_CLK2) support various display formats and three main operating modes:

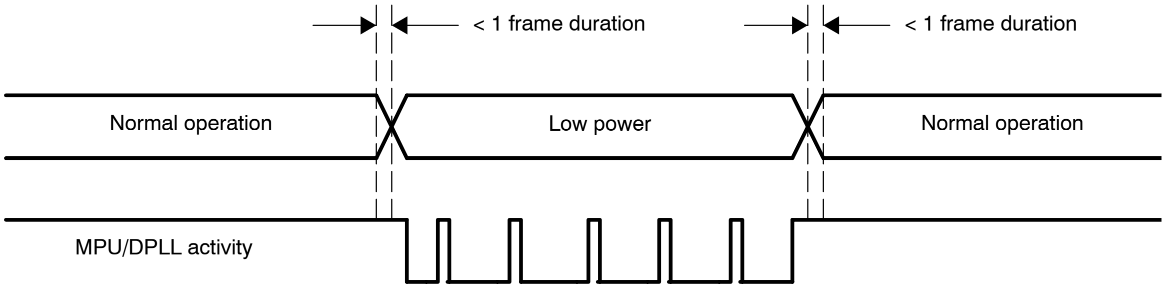
- Normal operation
- Voltage scaling
- Low-power refresh

The DSS can automatically switch between DSS_CLK1 and DSS_CLK2 synchronously to VFP signal without MPU support. This synchronized switch avoids visible effects on the screen.

Hardware Mechanism

When switching from normal mode to low-power refresh mode, the input frequency of the display subsystem must be reduced and its pixel frequency dividers must be reprogrammed. Using two clock inputs, the change operates synchronously with the VFP event and independently of the MPU. The synchronization can therefore require a frame duration of up to one between the request and the effective clock change (see Figure 5−32).

Figure 5−32. Switch Between Normal and Low Power (Screen Saver) Modes



Normal Operation

In normal operation (graphic and video scaling operation), the DSS requires a high functional clock (for example, a QVGA TFT panel requires at least 24 MHz). CORE_CLK or 48-MHz sources must be used.

Low-Power Mode

Often in low-power mode (simple still-picture refresh or screen saver), only the system clock (12, 13, or 19.2 MHz) is necessary, allowing the PLLs to be deactivated and saving significant power consumption.

During low-power mode, the display subsystem clock input remains unchanged, but device core frequency can vary.

The device toggles between two steps in low-power mode:

1) The display controller continuously outputs data on the LCD interface, flushing the graphics FIFO. While the FIFO provides data, the core domain is idle and the L3, OCM RAM, and SDRC clocks stop. SDRAM is in self-refresh mode. DPLL is in bypass mode, depending on the case.

2) When the FIFO must be refilled (DMA exits from standby mode on a threshold event, and the SDRAM exits from self-refresh), core clocks (L3, OCM RAM, SDRC/SMS) are enabled. Depending on the case, the DPLL either relocks to active mode or remains in bypass.

   When the FIFO is full, the DMA enters standby mode and the sequence returns to step 1. The display controller continues flushing the graphics FIFO during step 2.

This behavior requires specific core domain clock programming as described below.

Programming Model

This section describes typical programming for entering and exiting low-power mode.

The sequences include three main phases:

- ❑ Request
- ❑ Initiation
- ❑ Completion

The following DISPC interrupts implement the sequences:

- Frame done triggers the initiation phase.
- Programmed line triggers the completion phase.

The request phase can be triggered asynchronously with respect to the DISPC activity. If the request phase executes during LCD frame N, the initiation phase executes between frames N and N+1 and the completion phase executes in frame N+1.

Note:

Because the LCD output is disabled between frames N and N+1 to execute the initiation sequence, the time spacing between frames N and N+1 is longer than between any other two frames. (To avoid undesirable screen effects, ensure that the additional delay is small compared to the one-frame duration.)

Normal Operation Transition (Input Clock is DSS_CLK1 = CORE_CLK/N) to Low-Power Mode (DSS_CLK2 = SYSTEM_CLK)

- Initial configuration
  - DSS_CLK1 active (CM_FCLKEN1_CORE.EN_DSS1 = 1).
  - DSS_CLK2 selected (in CM_CLKSEL1_CORE).
  - The hardware-supervised clock control in the CORe power domain is configured as follows (allows L3 and L4 clocks to be shut down when the display does not access the memory):
    - CM_CLKSTCTRL_CORE.AUTOSTAT_L3 = 1
    - CM_CLKSTCTRL_CORE.AUTOSTAT_L4 = 1
    - CM_CLKSTCTRL_CORE.AUTOSTAT_DSS = 0 (DSS remains active)
- Request phase (triggered by any event)
  - Enable frame-done DISPC interrupt (DISPC_IRQENABLE, set FRAMEMASK bit).
  - Disable the LCD output of the DISPC (DISPC_CONTROL, clear LCDENABLE bit).
- Initiation phase (triggered by frame-done DISPC interrupt)
  - Enable the programmed-line DISPC interrupt (DISPC_IRQENABLE, set the PROGRAMMEDLINENUMBER bit).
  - Select DSS2 (APLL_CLOCK in the DSS) as the DISPC functional clock (DSS_CONTROL, set the DPLL_APLL_CLK bit).
  - Modify the pixel divisor in the DISPC.
  - Enable DSS2 in the PRCM (CM_FCLKEN1_CORE, set the EN_DSS2 bit).

  - Clear all DISPC event status flags (DISPC_IRQSTATUS, read it, and write back the same value).
  - Program PRCM dividers to match clock configuration VII.
  - Select the DPLL autoidle mode (CM_AUTOIDLE_PLL, set AUTO_DPLL to LOWPW, if the DPLL should relock in low-power mode; otherwise, set to DIS).
  - Change the DPLL multiplier/divisor (with CM_CLKSEL1_PLL, fields DPLL_MULT and DPLL_DIV—the maximum frequency allowed is 100 MHz).

Note:

Changing the DPLL multiplier and/or divisor involves a momentary loss of lock.

  - Switch the core clock to x1 DPLL output (CM_CLKSEL2_PLL, set CORE_CLK_SRC to SEL1).
  - Validate the new clock configuration and wait for the change to take effect (PRCM_CLKCFG_CTRL, set the VALID_CONFIG bit, check that CONFIG_STATUS in PRCM_CLKCFG_STATUS is NO_ERROR, and then wait for resetting VALID_CONFIG in PRCM_CLKCFG_CTRL).
  - Select the DPLL mode (CM_CLKEN_PLL, set EN_DPLL to Lock, if the DPLL should relock in low-power mode; otherwise, set to LP_BYPASS).
  - Enable the LCD output and activate the new DISPC LCD output configuration (DISPC_CONTROL, set the LCDENABLE and GOLCD bits).

- ☐ Completion phase (triggered by the programmed-line DISPC interrupt)
  - Disable the DSS1 clock in the PRCM (PRCM_FCLKEN1_CORE, clear the EN_DSS1 bit).
  - At this stage, the MPU can cease activity until the next wake-up event. For example, the following sequence indicates how the MPU can enter dormant mode:
    - Switches off DSS1_CLK (CM_FCLKEN1_CORE.EN_DSS1 = 0).
    - Enters the device in idle/wake-up mode (some peripherals using the system clock potentially can remain active).
    - Saves the context and programs the next state to be dormant: PM_PWSTCTRL_MPU. LogicRETState = 0; PM_PWSTCTRL_MPU.PowerState = 1.
    - Executes wait-for-interrupt instruction
  - The DPLL automatically switches to bypass mode after the MPU sleep transition completes (requires CM_AUTOIDLE_PLL.AUTO_DPLL = 3).

- Low-power-mode transition (DSS_CLK2 = SYSTEM_CLK) to normal operation (input clock is DSS_CLK1 = CORE_CLK/N)—initial configuration:
  - DSS_CLK2 active (CM_FCLKEN1_CORE.EN_DSS2 = 1)
  - The DSS_CLK2 frequency source is the system clock (CM_CLKSEL1_CORE.CLKSEL_DSS2 = 0).
- Request phase (triggered by any event)
  - Enable frame-done DISPC interrupt (DISPC_IRQENABLE, set the FRAMEMASK bit).
  - Disable the LCD output of the DISPC (DISPC_CONTROL, clear the LCDENABLE bit).
- Initiation phase (triggered by the frame-done DISPC interrupt).
  - The MPU is active or wakes up on any wake-up event (for example, GPIO, timer, peripheral).
  - Lock the DPLL (CM_CLKEN_PLL, set EN_DPLL to Lock).
  - Program the PRCM dividers to match the previous clock configuration I (used before entering low-power mode).
  - Validate the new clock configuration and wait for the change to take effect (PRCM_CLKCFG_CTRL, set the VALID_CONFIG bit, check that CONFIG_STATUS in PRCM_CLKCFG_STATUS is NO_ERROR, then wait for resetting VALID_CONFIG in PRCM_CLKCFG_CTRL).
  - Switch the core clock to x2 DPLL output (CM_CLKSEL2_PLL, set CORE_CLK_SRC to SEL2.
  - Change the DPLL multiplier/divisor to previous values (CM_CLKSEL1_PLL, fields DPLL_MULT and DPLL_DIV).

Note:

Changing the DPLL multiplier and/or divisor involves a momentary loss of lock.

  - Switch off DPLL autoidle (CM_AUTOIDLE_PLL, set AUTO_DPLL to DIS).
  - Enable the programmed-line DISPC interrupt (DISPC_IRQENABLE, set the PROGRAMMEDLINENUMBER bit).
  - Select DSS1 (DPLL_CLOCK in the DSS) as the DISPC functional clock (DSS_CONTROL, clear the DPLL_APLL_CLK bit).
  - Modify the pixel divisor in the DISPC (DISPC_DIVISOR, PCD field set to the previous value).
  - Enable DSS1 in the PRCM (CM_FCLKEN1_CORE, set the EN_DSS1 bit).
  - Clear all DISPC event status flags (DISPC_IRQSTATUS, read it and write back the same value).

  - ■ Enable the LCD output and activate the new DISPC LCD output configuration (DISPC_CONTROL, set the LCDENABLE and GOLCD bits).

- ❑ Completion phase (triggered by the programmed-line DISPC interrupt)

  Disable the DSS_CLK2 clock in the PRCM (PRCM_FCLKEN1_CORE, clear the EN_DSS2 bit).

Notes:

1) Other display attributes must be set and tuned to perform the low-power refresh mode. In particular, high- and low- FIFO thresholds should be set as far from each other as possible. See the display specification for details.
2) When the DPLL switches in bypass mode, the DSS requires the L3 and L4 clock divisors to be set to 1, which explains the use of configuration VII when entering low-power mode.
3) During the DSS clock input change, both clocks (DSS_CLK1 and DSS_CLK2) must be supplied to the DSS.

5.9.9.6 Display Subsystem Setting for Voltage Scaling

When the device is in a mode where voltage varies dynamically to adapt power to performance requirements, it is recommended not to use the DPLL clock (CORE_CLK/N) to supply the DSS, because this clock can change frequently, based on voltage.

It is suggested to supply the DSS with the 48-MHz fixed clock source for operation at high performance and resolution independently of the remaining device clocks scaling. In this case, it is not necessary to program the PRCM and DSS each time the voltage scales.

The following sequence shows how to run the DSS at 48 MHz:

1) The DSS_CLK1 must be active (CM_FCLKEN1_CORE.EN_DSS1 = 1).
2) The DSS_CLK1 frequency is selected (in CM_CLKSEL1_CORE).

To activate the 48-MHz source:

1) The MPU enables the DSS_CLK2 (CM_FCLKEN1_CORE.EN_DSS2 = 1) system clock for the DSS.
2) The MPU selects the 48-MHz clock for the DSS (CM_CLKSEL1_CORE.CLKSEL_DSS2 = 1). The APLL 96 MHz must be manually or automatically activated.
3) The MPU programs the DSS (pixel clock divider, selection of clock input 2 (DSS_CLK2)), and sets the GO bit (the GOLCD and GODIGITAL bits).
4) The MPU can execute a wait-for-interrupt instruction (and then the PRCM cuts MPU clocks).

5) On the next VFP event, the DSS operates all changes (divider and input clock selection) safely, and then sends an interrupt to the MPU.

6) On the interrupt, the MPU can wake up, switch off DSS_CLK1 (CM_FCLKEN1_CORE.EN_DSS1 = 0), and resume its activity.

5.9.10 L3 Domain Clock Management

5.9.10.1 Clocks, Initiators and Targets, Signaling Summary

Clocks and Initiators/Targets Summary

The PRCM provides the following clock signals to supply the L3 clock domain:

- ❑ CORE_L3_ICLK
- ❑ CORE_L4_USB_CLK

Initiators and Targets

The L3 clock domain includes the initiators and target modules/subsystems shown in Table 5−39.

Table 5−39. Initiators and Targets

Initiators	Targets
USB	USB
SDMA	SDRC
CAMERA IF	SMS
	GPMC
	OCM ROM
	OCM RAM
	L3 interconnect

5.9.10.2 Sleep Sequence

L3 clock control is automatically supervised by hardware. Clock control depends on L3 initiators and targets and on external clock domain activity: MPU, DSP, display subsystem, and L4.

The L3 clocks shut down when any of the following events occurs:

- ❑ CM_CLKSTCTRL_CORE.AUTOSTAT_L3 = 1
- ❑ [CM_AUTOIDLE(X)_CORE.AUTOIDLE = 1
  or CM_ICLKEN_CORE(X).EN = 0 when possible] for all L3 targets
- ❑ All L3 initiators are in standby mode.
- ❑ The MPU domain is in standby mode (and the MPU INTC is idled).

- The DSP domain is in standby mode (and all related targets are idled).
- The DSS is in standby mode.
- All targets in the L4 domain are idled.

Notes:

1) If all these conditions are met (except CM_CLKSTCTRL_CORE.AUTOSTAT_L3 = 0), the L3 domain does not operate a sleep transition. Thus, the software can keep L3 domain clocks always active by clearing the CM_CLKSTCTRL_CORE register AUTOSTAT_L3 bit.
2) When the L3 clock is disabled and the USB and CAMERA functional clocks are disabled [related CM_FCLKEN_CORE(X).EN = 0], all clocks in the L3 domain shut down (L3 clock domain is idled).
3) When the L4, L3, and DSS clock domains are idled, the CORE power domain is also idled and ready for a power transition.

5.9.10.3 Wake-Up Sequence

L3 clock activation is hardware-supervised only and managed in the following ways:

- DSS wake-up (DSS DMA exits standby mode)
- L4 domain wake-up event
- USB wake-up
- Voltage setup wake-up (voltage ramp complete)
- MPU power domain dependency wakeup (not controllable by software)
- DSP power domain dependency wakeup (not controllable by software)
- WKUP domain dependency wakeup (not controllable by software)

After a wake-up transition, a new sleep transition is possible only if the L3 initiators are active during at least one pulse of a SLEEP_CLK cycle.

CAUTION

5.9.10.4 Clock Selection

To select the L3 clock frequency, write in the CM_CLKSEL1_CORE register CLKSEL_L3 field.

The L3 frequency can be equal to CORE_CLK/[1, 2, 4, 6, 8, 12, or 16].

To select the DSS interface clock, write in the CM_CLKSEL1_CORE register CLKSEL_L4 field.

To define L4, select a ratio between the L4 and L3 clock frequencies of either 1 or 2.

To select the USB interface clock, write in the CM_CLKSEL1_CORE register CLKSEL_USB register field.

To define L4, select a ratio between the USB interface and L3 clock frequencies of 1 (used at boot only), 2, or 4.

Notes:

1) The CORE_CLK value depends on DPLL programming (M and N values in CM_CLKSEL1_PLL) and on DPLL output selection (CM_CLKSEL2_PLL).
2) See Section 5.4, *Clock Configurations*, for clock configuration settings.
3) A clock frequency change occurs only after validation in the PRCM_CLKCFG_CTRL register VALID_CONFIG field.

5.9.11 L4 Domain Clock Management

5.9.11.1 Clocks, Initiators, and Targets Summary

Clocks

The PRCM provides two clocks to supply the L4 clock domain:

- CORE_L4_CLK (core power domain)
- WU_L4_ICLK (wake-up power domain)

Targets

The L4 clock domain includes the following target modules:

- GPT1 to GPT12
- GPIO1 to GPIO4
- WDT1 to WDT4
- McBSP1, McBSP2
- MMC
- EAC
- UART1 to UART3
- SPI1, SPI2
- I2C1, I2C2
- HDQ
- FAC
- 32k_SYNC_TIMER
- Mailboxes
- OMAP control
- L4 interconnect

5.9.11.2 Sleep Sequence

Hardware automatically supervises L4 clock control, which depends on the L4 targets and on external clock domain activity: L3, MPU, and DSP.

L4 clocks are shut down when the following events occur:

- CM_CLKSTCTRL_CORE.AUTOSTAT_L4 = 1
- [CM_AUTOIDLE(X)_CORE.AUTOIDLE = 1 or CM_ICLKEN_CORE(X).en = 0] for all L4 targets
- SDMA is in standby mode.
- L3 is inactive (it has drained all its pending transactions).
- The MPU domain is in standby mode (and MPU INTC is idled).
- The DSP domain is in standby mode (and all related targets are idled).

Note:

1) If all these conditions are met (except CM_CLKSTCTRL_CORE.AUTOSTAT_L4 = 0), the L4 domain does not operate sleep transition. Therefore, the software can keep the L4 domain clocks always active by clearing the CM_CLKSTCTRL_CORE register AUTOSTAT_L4 bit.

2) When the L4, L3, and DSS clock domains are idled, the CORE power domain is also idled and ready for a power transition.

CAUTION

When the L4 clocks and all L4 targets functional clocks are disabled (except GPIO and GP timer 1 functional clocks that can be either disabled or maintained enabled), the L4 clock domain is idled. In idle mode, the GP timer 1 and GPIO modules can be kept active to allow the device to wake up.

5.9.11.3 Wake-Up Sequence

L4 clock activation, which is hardware-supervised only, is managed as follows:

- The MPU wakes up.
- The DSP wakes up.
- L3 wakes up.
- Any GPIO, GPT, UART, MMC, or SPI wakes up.

5.9.11.4 Clock Selection

To select the L4 clock frequency, write in the CM_CLKSEL1_CORE register CLKSEL_L4 field.

L4 is defined by selecting a ratio of either 1 or 2 between the L4 and L3 clock frequencies.

Notes:

1) The CORE_CLK value depends on DPLL programming (the M and N values in CM_CLKSEL1_PLL) and on DPLL output selection (CM_CLKSEL2_PLL).
2) The CORE_L3_ICLK is defined in the CM_CLKSEL1_CORE register CLKSEL_L3 field.
3) See Section 5.4, *Clock Configurations*, for clock configuration settings.
4) A clock frequency change occurs only after validation in PRCM_CLKCFG_CTRL register VALID_CONFIG field.

5.9.12 Full Device Idle and Wake-Up Sequence

This section describes how to set the full device in idle/retention/off mode and how to awaken it.

5.9.12.1 Sleep Sequence

To put the device in idle mode, the software must proceed as follows:

1) Ensure that all module interface clocks are either disabled (CM_ICLKEN_<power domain>.MODULE bit cleared) or under automatic control (that is, the CM_ICLKEN_<power domain>.MODULE bit and CM_AUTOIDLE_<power domain>.MODULE bit are both set to 1).
2) Ensure that all clock domain (MPU, DSP, L3, L4, and DSS) idle transitions are possible by setting all CM_CLKSTCTRL_<power domain>.AUTOSTATE bits to 1. (In some cases, the software can force a domain to idle by setting the PM_PWSTCTRL_<power domain>.FORCESTATE bit.)
3) Ensure that all functional clocks are stopped (CM_FCLKEN_<power domain>.MODULE bits cleared). In particular, the DSS clock must be disabled. Only the GPT1 and GPIOs clocks can remain active when these peripherals must be used to wake up the device.
4) Ensure that at least one wake-up source is valid.
5) Ensure that all initiators stop their activity (that is, they are either disabled or quiet).
6) Place the SDRAM in self-refresh mode, and then run the last instructions in internal memory.
7) Execute the MPU standby WFI instruction.

When the MPU and all L3 initiators (including the DSP, and DSS) enter standby mode, the L4 clock domain, followed by the L3 clock domain, enters idle mode, and the device is idled.

5.9.12.2 Device State

When entering idle mode, each of the power domains (MPU, DSP, and CORE) enters a power state (inactive, retention, or off) according to the PM_PWSTCTRL_<power domain>.POWERSTATE bit settings.

When at least one power domain remains in inactive state, the device is in idle mode.

When all four power domains are either in retention or off mode, the device is in retention mode.

When all four power domains are in off mode, the device is in off mode.

5.9.12.3 Voltage and Clock Requests

When the device enters either retention or off mode, SYS.nVMODE and SYS.CLKREQ can be updated according to the PRCM_VOLTCTRL and PRCM_CLKSRC_CTRL.AUTOEXTCLKMODE bit settings.

Figure 5−36 represents a sequence in which the device enters off mode, which causes SYS.nVMODE and SYS.CLKREQ to be asserted low. Voltage is thus dropped down and the system clock is shut down (either externally or internally).

The following settings correspond to this sequence:

- ❏ PRCM_ CLKSRC_CTRL.AUTOEXTCLKMODE = 0x1
- ❏ PRCM_VOLTCTRL.AUTO_EXTVOLT = 0x1
- ❏ PRCM_VOLTCTRL.SETOFF_LEVEL = 0x1

Note:

The CLKREQ does not turn off the system clock when going into idle unless auto-voltage scaling is enabled: PRCM_VOLTCTRL.AUTO_EXTVOLT = 0x1. Auto-voltage scaling does not have to change the voltage level, but it does have to be enabled.

5.9.12.4 Wake-Up Sequence

The device can wake up from inactive, retention, or off mode from any wake-up issued by a GPIO or the GP timer 1 in the WKUP power domain, if they have been activated.

An external warm reset also causes the device to wake up (see Figure 5−24).

Note:

To put the device in idle mode a second time, immediately after the wake-up sequence, the software must proceed as follows. This scenario can be generalized to all modules capable of type-1 wake-up events (for this example, GPT1 waking up the device).

When the interrupt is generated, the ISR (interrupt service routine) handles the interrupt.

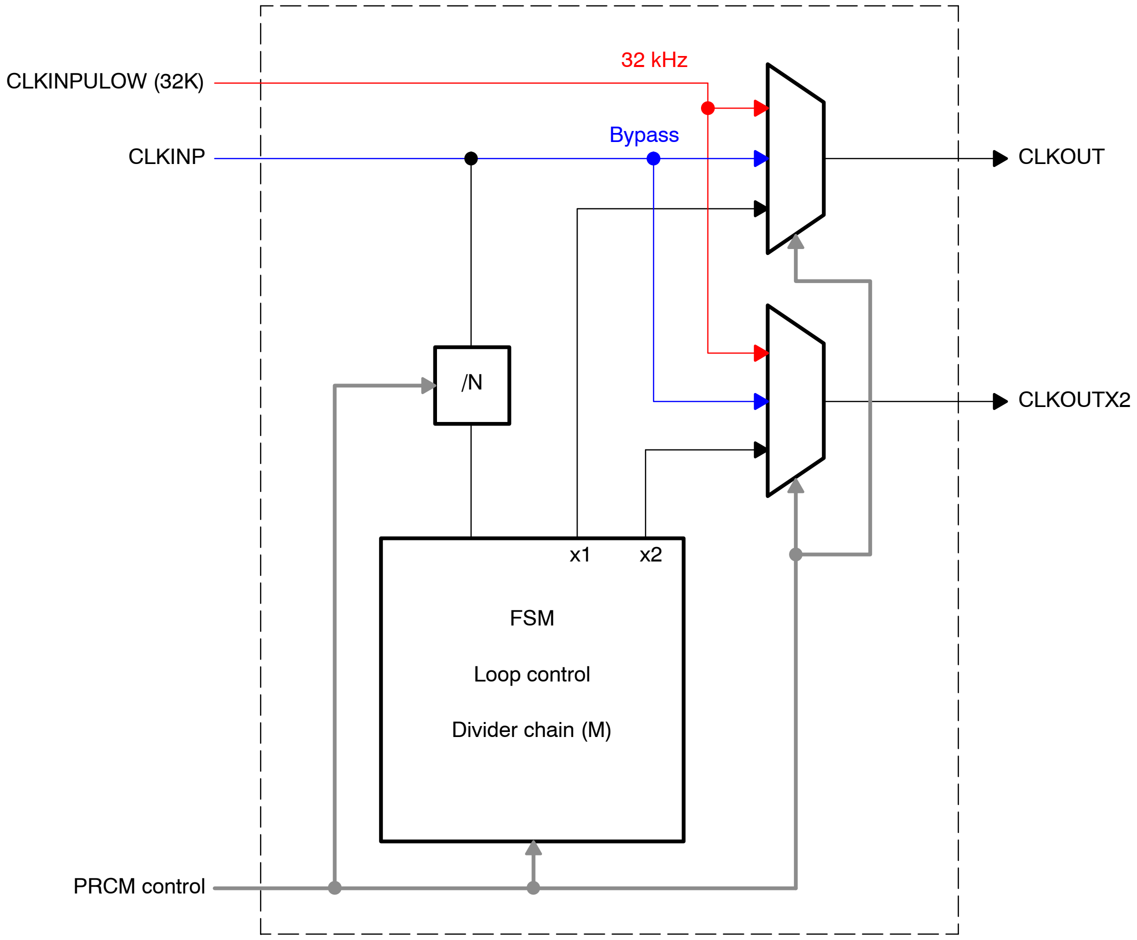
1) Check in the PRCM_IRQSTATUS_MPU register to see which kind of interrupt is generated.
   In this example, it is a type-1 wake-up event (WKUP1_ST bit PRCM_IRQSTATUS_MPU[0]).
2) A GPT1 wake-up event is expected. This can be checked by looking at the ST_GPT1 bit PM_WKST_WKUP[0].
3) Stop the GPT1 timer.
4) Clear the wake-up status for GPT1 by writing 1 in the ST_GPT1 bit PM_WKST_WKUP[0].
5) Clear the GPT1 pending overflow interrupt by writing 1 in the GPT1 timer OVF_IT_FLAG bit TISR[1].
6) Clear the interrupt status register by writing 1 in the WKUP1_ST bit PRCM_IRQSTATUS_MPU[0].
7) Disable the PRCM type-1 wake-up event by writing 0 in the WKUP1_EN bit PRCM_IRQENABLE_MPU[0].

5.9.13 DPLL Programming

5.9.13.1 Overview

Figure 5−33 is a functional representation of DPLL clock generation and control.

Figure 5−33. DPLL Functional Diagram



The OMAP2420 DPLL has two clock inputs:

- CLKINP: Reference input clock (system clock 12 MHz, 13 MHz, or 19.2 MHz)
- CLKINPULOW: 32-kHz clock

The OMAP2420 DPLL has one clock output:

- CLKOUTX2: Output clock

The DPLL analog/digital cell is fully controlled by the PRCM.

5.9.13.2 DPLL Clock Generation

The DPLL output clocks are generated according to the following equations:

- ☐ CLKOUT = CLKINP * M/(N+1)
- ☐ CLKOUTX2 = CLKINP * M/(N+1) * 2

5.9.13.3 Power Consumption and Latencies

Lock latencies depend on internal reference frequency, defined as:

FINT = CLKINP/(N+1)

Therefore, it is recommended to minimize the N value to reduce lock latency (see Table 5−40).

Table 5−40. Power Consumption

Mode	CLKOUTX2	Lock Time	Power
Off	OFF	1.5 µs + 350 FINT cycles	< 1 µA
STOPMODE	OFF	1.5 µs + 20 FINT cycles	< 1 µA
Active	CLKINP * M/(N+1) *2	N/A	< 3 mA
Idle bypass Low power (default)	CLKINP	1.5 µs + 20 FINT cycles	Bypass active current
Idle Bypass fast relock	CLKINP	0.05 µs + 20 FINT cycles	< 200 µA + bypass active current

Note:

For bypass fast relock mode, relock time assumes a maximum 10°C temperature drift, no voltage change, and multiplier M values larger than 16; otherwise, behavior is not guaranteed. A typical use of this fast relock mode is an LCD low-power refresh scenario. In this case, the bypass duration is short enough to ensure an acceptable temperature drift. If it is not possible to ensure all the conditions for fast relock usage, it is mandatory to use the default low-power bypass.

5.9.13.4 Programming Model

The DPLL is enabled/disabled (bypass) by writing in the CM_CLKEN_PLL register. A write has an immediate effect.

Three modes are programmable:

- Bypass mode (low power)
- Bypass mode (short relock)
- Active mode (DPLL locked)

  The DPLL clock out selection is controlled by the CM_CLKSEL2_PLL register.

Three output values are available:

- 32-kHz output
- DPLL primary clock
- DPLL primary clock x 2

  DPLL clock status is logged in the CM_IDLEST_CKGEN register.

The ST_CORE_CLK bit field indicates whether:

- DPLL is in bypass (output is system clock)
- DPLL is locked
- DPLL is in low frequency mode (output is 32 kHz)

  DPLL automatic control is defined in the CM_AUTOIDLE_PLL register. The DPLL can be automatically stopped (stop mode) or set in bypass mode (low-power or fast relock modes) when the CORE_CLK is not required. It is automatically restarted (to lock mode) when the CORE_CLK is required.

  DPLL output frequency is programmed in the CM_CLKSEL1_PLL register.

  - M value is the multiplier, from 0 to 1023.
  - N value is the divider, from 0 to 15.

Notes:

- When M = 0 or 1, the DPLL is bypassed and its internal oscillator is shut down. This mode is different from low-power mode and from fast-relock bypass mode, and it must not be used.

- Not all DPLL settings require validation in the PRCM_CLKCFG_CTRL.VALID_CONFIG register.

Notes:

To put the device in idle mode a second time, immediately after the wake-up sequence, the software must proceed as follows. This scenario can be generalized to all modules capable of type-1 wake-up events (for this example, GPT1 is waking up the device).
When the interrupt is generated, the ISR handles the interrupt.

1) Check the PRCM_IRQSTATUS_MPU register to see which kind of interrupt is generated.

In this example, it is a type-1 wake-up event (WKUP1_ST bit PRCM_IRQSTATUS_MPU[0]).

2) A GPT1 wake-up event is expected. This can be checked by looking at the ST_GPT1 bit PM_WKST_WKUP[0].
3) Stop the GPT1 timer.
4) Clear the wake-up status for GPT1 by writing 1 in the ST_GPT1 bit PM_WKST_WKUP[0].
5) Clear the GPT1 pending overflow interrupt by writing 1 in the GPT1 timer OVF_IT_FLAG bit TISR[1].
6) Clear the interrupt status register by writing 1 in the WKUP1_ST bit PRCM_IRQSTATUS_MPU[0].
7) Disable the PRCM type-1 wake-up event by writing 0 in the WKUP1_EN bit PRCM_IRQENABLE_MPU[0].

5.9.14 Frequency Scaling

5.9.14.1 Introduction

The software can scale one or more module frequencies and adapt them to effective workload, providing the following benefits:

- Power savings when autogating is not possible
- Lower device temperature and, consequently, less device leakage

Frequency scaling can be achieved through several ways:

- DPLL M, N reprogramming
  - Latency: Because of DPLL relock, the maximum is:
    1.5 µs + 350 x FINT (FINT = SYSTEM_CLK/(N+1))cycles
  - Impact: Entire DPLL clock tree (CORE_CLK)
- DPLL output selection (x1, x2, 32 kHz)
  - Latency: Few DPLL clock cycles
  - Impact: Entire DPLL clock tree (CORE_CLK)
- Subsystem or module clock division or selection
  - Latency: Because of resynchronization inside the PRCM, up to 1440 DPLL clock (CORE_CLK) cycles
  - Impact: Subsystem or module only

Note:

FINT = SYSTEM_CLK/(N+1).

CAUTION

The OMAP2420 supports frequency scaling on its internal CORE_CLK, whether using the dual DPLL output (DPLL_CLKOUT or DPLL_CLKOUT x2) or changing DPLL N and M parameters, without affecting global chip behavior. This can be used for lower-frequency operating modes or before entering low-power modes (voltage scaling). The software must ensure that all system applicative prerequisites have been taken into account before using such scaling:
– SDRC and GPMC ac timing parameters may need to be updated according to the new frequency selected.
– SDRC autorefresh counter value may need to be updated according to the new frequency selected.
– When using a DDR memory with DLLs/DCDLs locked, the MPU must not access it during DLLs/DCDLs relock time after frequency change.
– The software must ensure that any ongoing transaction from/to external peripherals is not corrupted and does not cause a loss of data as a result of the frequency change.

5.9.14.2 Programming Model

- ❏ DPLL M, N reprogramming
    - On-the-fly operation (the device is not required to be idled)
    - Write in CM_CLKSEL1_PLL register: M and N value settings.
- ❏ DPLL output selection (x1, x2, 32 KHz)
    - On-the-fly operation (the device is not required to be idled)
    - Write in CM_CLKSEL2_PLL register: CORE_CLK selection.
- ❏ Subsystem or module clock division or selection
    - On-the-fly operation (the device is not required to be idled)
    - Write in one or more CM_CLKSEL_<domain> registers.
    - Write in PRCM_CLKCFG_CTRL register: Set valid bit.
    - If required, poll PRCM_CLKCFG_STATUS register: Check valid status.

5.9.14.3 Example

The following example describes two scenarios in which the DSP processor effective workload does not require its maximum functional clock frequency.

Reducing this frequency decreases related clock tree consumption when DSP is not idled as soon as it is inactive.

In the two following sequences, the assumption is that the device is initially programmed with the clock configuration 1 (DSP frequency is programmed to be 220 MHz).

Scenario 1: The DSP core is scaled from 220 MHz to 110 MHz. The DSP synchronizer is active:

1) Change the DSP core frequency to 110 MHz: (CM_CLKSEL_DSP.CLKSEL_DSP = 6)

2) Align the DSP clock interface with the DSP core clock: (CM_CLKSEL_DSP.CLKSEL_DSP_IF = 1)

3) Valid new clock configuration (PRCM_CLKCFG_CTRL. Valid_config = 1)

Scenario 2: The DSP core is scaled from 220 MHz to 165 MHz. The DSP synchronizer is disabled:

1) Change the DSP core frequency to 165 MHz: (CM_CLKSEL_DSP.CLKSEL_DSP = 4)

2) Align the DSP clock interface with the DSP core clock: (CM_CLKSEL_DSP.CLKSEL_DSP_IF = 1)

3) Set the DSP synchronizer to be in bypass mode: (CM_CLKSEL_DSP.SYNC_DSP = 0)

4) Valid new clock configuration (PRCM_CLKCFG_CTRL. VALID_CONFIG = 1)

The two sequences can be performed on-the-fly while the DSP is operating.

5.9.15 Voltage Scaling

5.9.15.1 Introduction

The software can scale the voltage (identical for each power domain) to adapt power consumption to performance requirements.

The OMAP2420 supports two operating voltage levels:

- Low-voltage point
- High-voltage point

Low-voltage operation is useful to reduce leakage and active power consumption:

- For medium workload activity
- When MPU, DSP, and core domains are in off or retention mode.

At low voltage, the DPLL clock tree must be divided by 2. Peripheral clocks remain unchanged, regardless of the voltage, except for the camera interface. For more information, see Section 5.4.3.2, *Peripheral Clocks*.

Depending on performance requirements, an L3 clock frequency change can require reprogramming of GPMC ac timings and SDRC ac timings.

Voltage scaling causes the DLLs and DCDLs to relock in the SDRAM controller. The associated latency is about 1000 x L3 clock cycles.

If PLL voltage is scaled with core device voltage, the APLLs and DPLL must relock. During relock, the DPLL is in bypass. APLLs do not output clock during locking.

Voltage scaling can be handled in four different scenarios:

- The device operates during scaling.
- The device is idled, scaling occurs, and then the device wakes up and resumes activity.
- The device enters off or retention mode and the PRCM automatically manages the voltage transition.
- The voltage scaling method must be force, not automatic, and the MPU and CORE power states must be set to on to use the voltage transition interrupt.

While the MPU is operating during voltage scaling, it must avoid accessing the SDRAM during DLL and DCDL relock. Ensure that no other initiator in the device accesses the external SDRAM during this period.

5.9.15.2 Preliminary Settings

To operate voltage scaling, the software must perform the following preliminary steps:

1) Program voltage levels and maximum ramp time.
   a) Program voltage values associated with VMODE signal states in the external power IC (for example, Menelaus).

      Example: L1 level → VMODE = 0 → VDD = Low-voltage point
      L0 level → VMODE = 1 → VDD = High-voltage point

Note:

VMODE polarity can be defined in the PRCM_POLCTRL register.

   b) Configure VDD setup time in the PRCM_VOLTSETUP register.
2) Ensure that the display controller functional clock remains unchanged during scaling:
   a) Enable the display clock DSS2_CLK in the CM_FCLKEN1_CORE register.
   b) Set the DSS functional frequency to DSS2_CLK (48 MHz or system clock source) in the CM_CLKSEL1_CORE register.
   c) Validate the new clock configuration in the PRCM_CLKCFG_CTRL register.
   d) Configure the new pixel clock dividers and clock input selection in the DSS.
   e) On display VSYNC interrupt, disable the display clock DSS1_CLK in the CM_FCLKEN1_CORE register.
   f) Validate the new clock configuration in the PRCM_CLKCFG_CTRL register.

5.9.15.3 Scenario 1

Description: The DPLL remains stable and the device operates during voltage scaling.

Transition high-voltage point to low-voltage point

Execute the following instructions from internal memory only:

1) Change the autorefresh frequency in the SDRC to correlate with the upcoming slower frequency.
2) Divide the DPLL clock tree by 2 in the PRCM CM_CLKSEL2_PLL register.
3) Set the voltage level to L1 and force the voltage transition in the PRCM_VOLTCTRL register.
4) Start the counter for the voltage and DLL/DCDL stabilization time (using either a timer or processor delay loop), where [voltage + DLL/DCDL]

stabilization time is equal to [PRCM_VOLTSETUP.SETUP_TIME value + (960 x L3 clock cycles)].

5) If necessary, set new ac timings for the SDRC controller.
6) If necessary, set new ac timings for the GPMC controller.
7) On counter overflow or delay loop expiration, the MPU can access the external SDRAM memory.

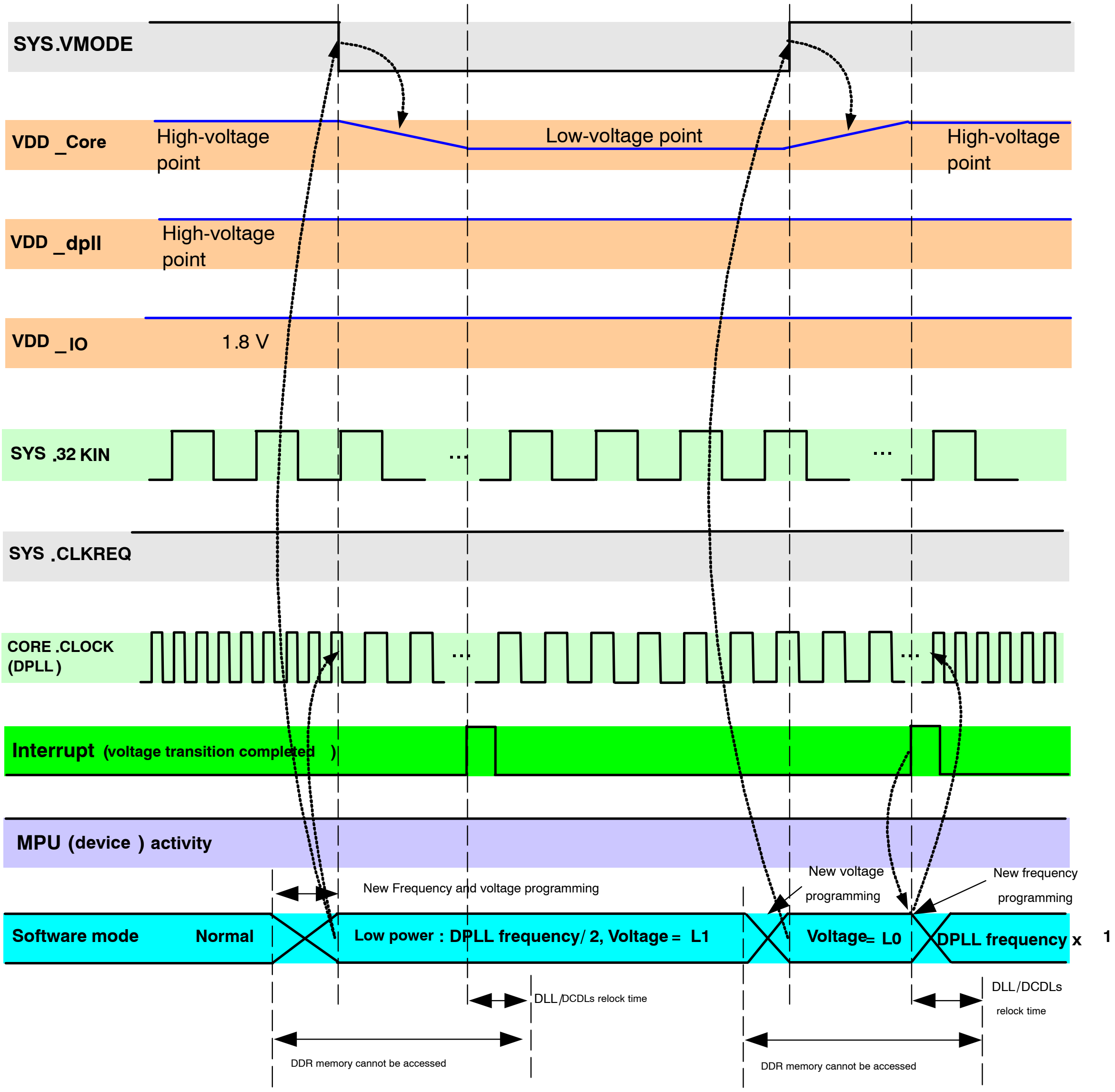
Transition low-voltage point to high-voltage point

Execute the following instructions from internal memory only:

1) Change the autorefresh frequency in the SDRC to correlate with the upcoming faster frequency.
2) If necessary, set new ac timings for the SDRC controller.
3) If necessary, set new ac timings for the GPMC controller.
4) Set voltage level to L0 and force voltage transition in the PRCM_VOLTCTRL register.
5) Start the counter for the voltage stabilization time (using either a timer or a processor delay loop), where [voltage] stabilization time is equal to [PRCM_VOLTSETUP.SETUP_TIME value].
6) On counter overflow or delay loop expiration, multiply the DPLL clock by 2 in the PRCM CM_CLKSEL2_PLL register.
7) Start the counter for the DLL/DCDL stabilization time (using either a timer or a processor delay loop), where [DLL/DCDL] stabilization time is equal to [(960 x L3 clock cycles)].
8) On counter overflow or delay loop expiration, the MPU can access external SDRAM memory.

Figure 5−34 shows the voltage scaling scenario 1.

Figure 5–34. Voltage Scaling Sequence—Scenario 1



5.9.15.4 Scenario 2

Description: The device is idled and not operating during voltage scaling.

Conditions of transition:

- ☐ The device must be idled; that is, all processors and initiators must be quiet, all peripherals must be in idle mode, and all clocks must be turned off.
- ☐ Any wake-up event that occurs while the device is idle prevents a voltage transition interrupt.
- ☐ The PRCM_VOLTSETUP.SETUP_TIME value must be set with a valid voltage setup value.
- ☐ PRCM_VOLTCTRL.AUTO_EXTVOLT does not have to be set.
- ☐ Execute the following instructions from internal memory only.

Transition high-voltage point to low-voltage point

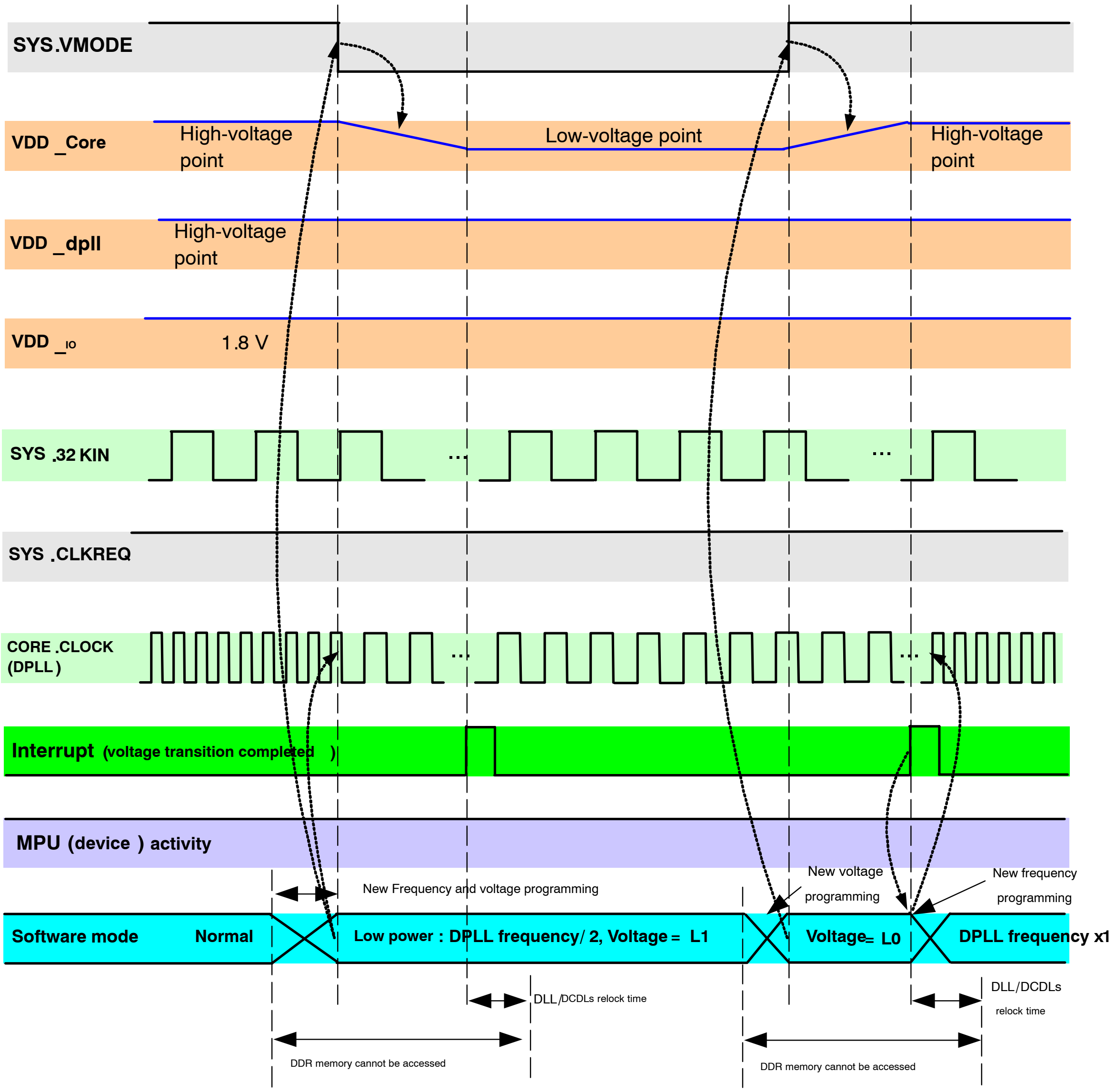
1) The MPU and CORE POWERSTATE bits must be set to ON (PM_PWSTCTRL_MPU.POWERSTATE = 0x0 and PM_PWSTCTRL_CORE.POWERSTATE = 0x0).
2) Enable the voltage transition interrupt in the PRCM_IRQENABLE register.
3) Divide the DPLL clock by 2 in the CM_CLKSEL2_PLL register.
4) Change the autorefresh frequency in the SDRC to correlate with the upcoming slower frequency.
5) Set VOLT_LEVEL to L1 in the PRCM_VOLTCTRL register.
6) The software stops device activity and then executes an MPU WFI instruction.
7) Voltage ramp starts when the entire device is inactive.
8) On the voltage transition interrupt, the MPU wakes up and resumes device activity in internal memory.
9) Start the counter (using either a timer or a processor delay loop) for the DLL/DCDL stabilization time (960 x L3 clock cycles).
10) If required, set new ac timings for the SDRC controller.
11) If required, set new ac timings for the GPMC controller.
12) On counter overflow or delay loop expiration, the MPU can access the external SDRAM memory.

Transition low-voltage point to high-voltage point

1) The MPU and CORE POWERSTATE bits must be set to ON (PM_PWSTCTRL_MPU.POWERSTATE = 0x0 and PM_PWSTCTRL_CORE.POWERSTATE = 0x0).
2) Enable the voltage transition interrupt in the PRCM_IRQENABLE register.
3) Set VOLT_LEVEL to L0 in the PRCM_VOLTCTRL register.
4) The software stops device activity and then executes the MPU WFI instruction.
5) Voltage ramp starts when the device is idled.
6) On voltage transition completed event, the MPU wakes up executing from internal memory.
7) Multiply the DPLL clock tree by 2: Configuration of the CM_CLKSEL2_PLL register.
8) Start counter (using either a timer or a processor delay loop) for the DLL/DCDL stabilization time (960 x L3 clock cycles).
9) If required, set new ac timings for the SDRC controller.
10) If required, set new ac timings for the GPMC controller.
11) Change the autorefresh frequency in the SDRC for faster frequency.
12) On counter overflow or delay loop expiration, the MPU can access external SDRAM memory.

Figure 5−35 shows the voltage scaling scenario 2.

Figure 5–35. Voltage Scaling Sequence—Scenario 2



5.9.15.5 Scenario 3

Description: Scenario 3 describes a hardware automatic control. Voltage is scaled down automatically when the device enters off or retention mode. This automatic control can be disabled.

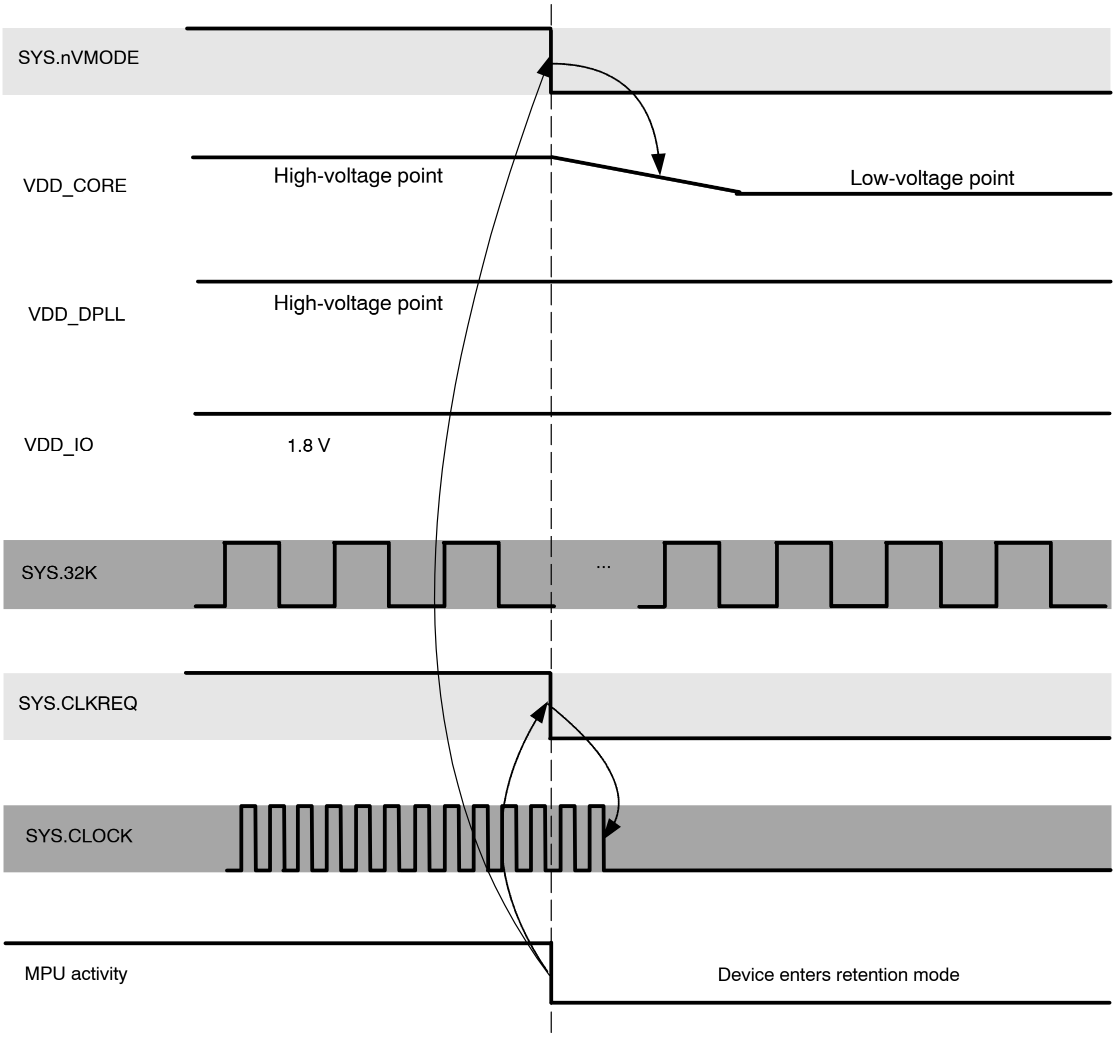
The device enters off or retention mode. If automatic control is enabled, the VMODE signal state changes when the device enters one of these low-power states, asking the power IC to scale down the voltage.

Note:

VMODE can remain unchanged if the device is already operating in low-power mode (L1).

Figure 5–36 shows the device sleep-to-retention sequence.

Figure 5−36. Sleep-to-Retention Sequence—Scenario 3

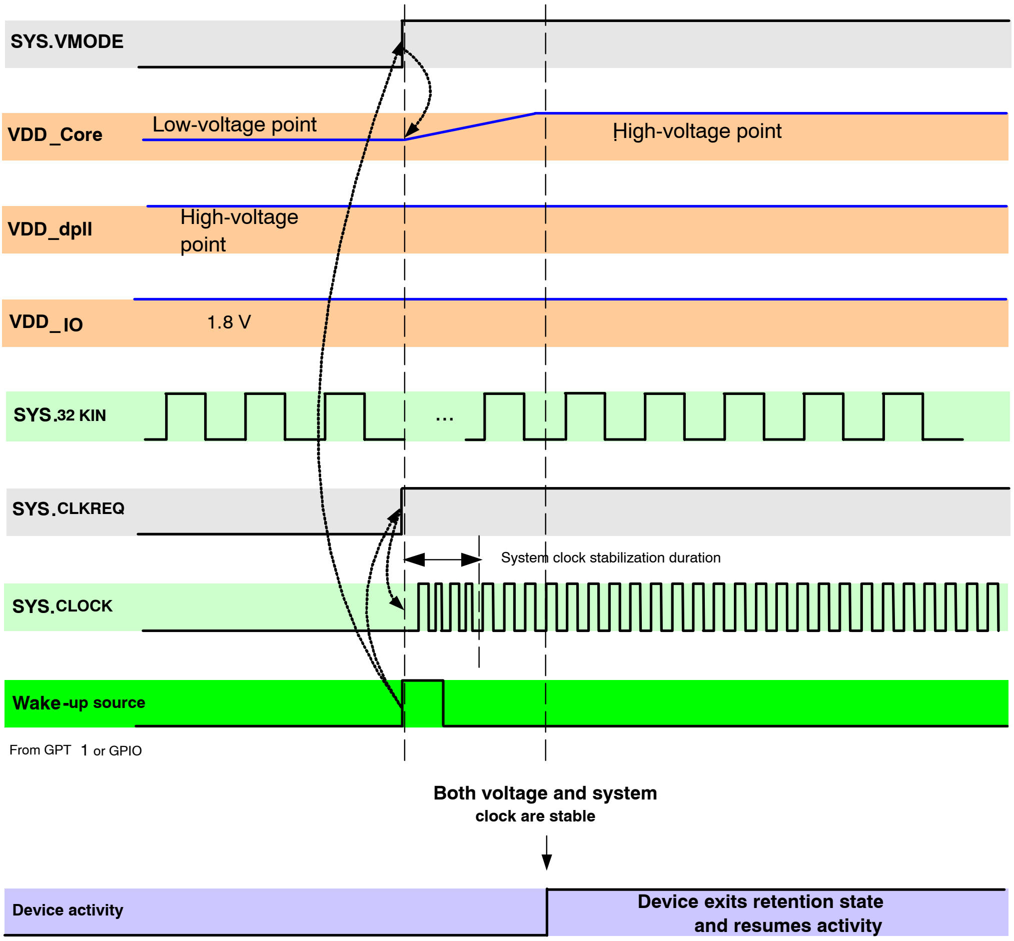


The device is awakened from off or retention mode:

On a wake-up event, VMODE automatically switches to the state it had when the device was active.

Figure 5−37 shows the device wake-up from off or retention mode.

Figure 5–37. Wake-Up From Retention Sequence—Scenario 3



5.10 Power, Reset, and Clock Management Registers

Table 5−41 is the PRCM instance summary.

Table 5−41. Instance Summary

Module Name	Base Address	Size
PRCM	0x4800 8000	4K bytes

5.10.1 PRCM Register Summary

CAUTION

PRCM registers are limited to 32-bit data access. 16-bit and 8-bit data accesses are not allowed because they can corrupt register contents.

Table 5−42 lists the PRCM registers. Table 5−43 through Table 5−131 describe the register bits.

Note:

The Reset Source column indicates whether the register is reset only on power-up (cold reset: C) or on any global reset (warm reset: W).

Table 5−42. PRCM Registers

Register Name	Type	Register Width (Bits)	Physical Address	Reset Source
PRCM_REVISION	R	32	0x4800 8000	C
PRCM_SYSCONFIG	RW	32	0x4800 8010	W
PRCM_IRQSTATUS_MPU	RW	32	0x4800 8018	W
PRCM_IRQENABLE_MPU	RW	32	0x4800 801C	W
PRCM_VOLTCTRL	RW	32	0x4800 8050	W
PRCM_VOLTST	R	32	0x4800 8054	W
PRCM_CLKSRC_CTRL	RW	32	0x4800 8060	C
PRCM_CLKOUT_CTRL	RW	32	0x4800 8070	C
PRCM_CLKCFG_CTRL	RW	32	0x4800 8080	W
PRCM_CLKCFG_STATUS	R	32	0x4800 8084	W
PRCM_VOLTSETUP	RW	32	0x4800 8090	C
PRCM_CLKSSETUP	RW	32	0x4800 8094	C
PRCM_POLCTRL	RW	32	0x4800 8098	C
GENERAL_PURPOSE1	RW	32	0x4800 80B0	C

Table 5−42. PRCM Registers (Continued)

Register Name	Type	Register Width (Bits)	Physical Address	Reset Source
GENERAL_PURPOSE2	RW	32	0x4800 80B4	C
GENERAL_PURPOSE3	RW	32	0x4800 80B8	C
GENERAL_PURPOSE4	RW	32	0x4800 80BC	C
GENERAL_PURPOSE5	RW	32	0x4800 80C0	C
GENERAL_PURPOSE6	RW	32	0x4800 80C4	C
GENERAL_PURPOSE7	RW	32	0x4800 80C8	C
GENERAL_PURPOSE8	RW	32	0x4800 80CC	C
GENERAL_PURPOSE9	RW	32	0x4800 80D0	C
GENERAL_PURPOSE10	RW	32	0x4800 80D4	C
GENERAL_PURPOSE11	RW	32	0x4800 80D8	C
GENERAL_PURPOSE12	RW	32	0x4800 80DC	C
GENERAL_PURPOSE13	RW	32	0x4800 80E0	C
GENERAL_PURPOSE14	RW	32	0x4800 80E4	C
GENERAL_PURPOSE15	RW	32	0x4800 80E8	C
GENERAL_PURPOSE16	RW	32	0x4800 80EC	C
GENERAL_PURPOSE17	RW	32	0x4800 80F0	C
GENERAL_PURPOSE18	RW	32	0x4800 80F4	C
GENERAL_PURPOSE19	RW	32	0x4800 80F8	C
GENERAL_PURPOSE20	RW	32	0x4800 80FC	C
CM_CLKSEL_MPU	RW	32	0x4800 8140	W
CM_CLKSTCTRL_MPU	RW	32	0x4800 8148	W
RM_RSTST_MPU	RW	32	0x4800 8158	C
PM_WKDEP_MPU	RW	32	0x4800 81C8	W
PM_EVGENCTRL_MPU	RW	32	0x4800 81D4	W
PM_EVEGENONTIM_MPU	RW	32	0x4800 81D8	W
PM_EVEGENOFFTIM_MPU	RW	32	0x4800 81DC	W
PM_PWSTCTRL_MPU	RW	32	0x4800 81E0	W
PM_PWSTST_MPU	R	32	0x4800 81E4	W
CM_FCLKEN1_CORE	RW	32	0x4800 8200	W
CM_FCLKEN2_CORE	RW	32	0x4800 8204	C
CM_ICLKEN1_CORE	RW	32	0x4800 8210	W

Table 5−42. PRCM Registers (Continued)

Register Name	Type	Register Width (Bits)	Physical Address	Reset Source
CM_ICLKEN2_CORE	RW	32	0x4800 8214	C
CM_IDLEST1_CORE	R	32	0x4800 8220	W
CM_IDLEST2_CORE	R	32	0x4800 8224	C
Reserved	R	32	0x4800 822C	W
CM_AUTOIDLE1_CORE	RW	32	0x4800 8230	W
CM_AUTOIDLE2_CORE	RW	32	0x4800 8234	C
CM_AUTOIDLE3_CORE	RW	32	0x4800 8238	W
Reserved	RW	32	0x4800 823C	W
CM_CLKSEL1_CORE	RW	32	0x4800 8240	W
CM_CLKSEL2_CORE	RW	32	0x4800 8244	W
CM_CLKSTCTRL_CORE	RW	32	0x4800 8248	W
PM_WKEN1_CORE	RW	32	0x4800 82A0	W
PM_WKEN2_CORE	RW	32	0x4800 82A4	W
PM_WKST1_CORE	RW	32	0x4800 82B0	W
PM_WKST2_CORE	RW	32	0x4800 82B4	W
PM_WKDEP_CORE	R	32	0x4800 82C8	W
PM_PWSTCTRL_CORE	RW	32	0x4800 82E0	W
PM_PWSTST_CORE	R	32	0x4800 82E4	W
CM_FCLKEN_WKUP	RW	32	0x4800 8400	W
CM_ICLKEN_WKUP	RW	32	0x4800 8410	W
CM_IDLEST_WKUP	RW	32	0x4800 8420	W
CM_AUTOIDLE_WKUP	RW	32	0x4800 8430	W
CM_CLKSEL_WKUP	RW	32	0x4800 8440	W
RM_RSTCTRL_WKUP	RW	32	0x4800 8450	W
RM_RSTTIME_WKUP	RW	32	0x4800 8454	C
RM_RSTST_WKUP	RW	32	0x4800 8458	C
PM_WKEN_WKUP	RW	32	0x4800 84A0	W
PM_WKST_WKUP	RW	32	0x4800 84B0	W
CM_CLKEN_PLL	RW	32	0x4800 8500	W
CM_IDLEST_CKGEN	R	32	0x4800 8520	W
CM_AUTOIDLE_PLL	RW	32	0x4800 8530	W

Table 5−42. PRCM Registers (Continued)

Register Name	Type	Register Width (Bits)	Physical Address	Reset Source
CM_CLKSEL1_PLL	RW	32	0x4800 8540	C
CM_CLKSEL2_PLL	RW	32	0x4800 8544	C
CM_FCLKEN_DSP	RW	32	0x4800 8800	W
CM_ICLKEN_DSP	RW	32	0x4800 8810	W
CM_IDLEST_DSP	R	32	0x4800 8820	W
CM_AUTOIDLE_DSP	RW	32	0x4800 8830	W
CM_CLKSEL_DSP	RW	32	0x4800 8840	W
CM_CLKSTCTRL_DSP	RW	32	0x4800 8848	W
RM_RSTCTRL_DSP	RW	32	0x4800 8850	W
RM_RSTST_DSP	RW	32	0x4800 8858	C
PM_WKDEP_DSP	RW	32	0x4800 88C8	W
PM_PWSTCTRL_DSP	RW	32	0x4800 88E0	W
PM_PWSTST_DSP	R	32	0x4800 88E4	W
PRCM_IRQSTATUS_DSP	RW	32	0x4800 88F0	W
PRCM_IRQENABLE_DSP	RW	32	0x4800 88F4	W

5.11 Register Descriptions

Table 5−43. PRCM_REVISION

Address Offset	0x000		
Physical Address	0x4800 8000	**Instance**	PRCM
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] indicates major revision. [3:0] indicates minor revision Examples: 0x10 for 1.0, 0x21 for 2.1	R	

Table 5−44. PRCM_SYSCONFIG

Address Offset	0x010		
Physical Address	0x4800 8010	**Instance**	PRCM
Description	This register controls the OCP interface 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																															AUTOIDLE

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000000
0	AUTOIDLE	Internal clock-gating strategy 0x0: OCP clock is free-running. 0x1: Automatic clock-gating strategy is enabled, based on OCP interface activity.	RW	0

Table 5−45. PRCM_IRQSTATUS_MPU

Address Offset	0x018		
Physical Address	0x4800 8018	**Instance**	PRCM
Description	This interrupt status register regroups the statuses of module internal events that can generate interrupts. Writing 1 to a bit resets this bit. This register applies on 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																										TRANSITION_ST	EVGENOFF_ST	EVGENON_ST	VOLTTRANS_ST	WKUP2_ST	WKUP1_ST

Bits	Field Name	Description		Type	Reset
31:6	Reserved	Write 0s for future compatibility. Read returns 0.		RW	0x0000000
5	TRANSITION_ST	Domain transition completed event status		RW	0
		Read 0x0:	Transition complete event is false.		
		Write 0x0:	Status bit is unchanged.		
		Read 0x1:	Transition complete event is true (pending).		
		Write 0x1:	Status bit is reset.		
4	EVGENOFF_ST	Event generator end-of-OFFtime status		RW	0
		Read 0x0:	End-of-OFF time event is false.		
		Write 0x0:	Status bit is unchanged.		
		Read 0x1:	End-of-OFF time event is true (pending)		
		Write 0x1:	Status bit is reset.		
3	EVGENON_ST	Event generator end-of-ON time status		RW	0
		Read 0x0:	End-of-ON time event is false.		
		Write 0x0:	Status bit is unchanged.		
		Read 0x1:	End-of-ON time event is true (pending).		
		Write 0x1:	Status bit is reset.		

Table 5−45. PRCM_IRQSTATUS_MPU (Continued)

Bits	Field Name	Description		Type	Reset
2	VOLTTRANS_ST	Voltage transition complete event status		RW	0
		Read 0x0:	Voltage transition complete event is false.		
		Write 0x0:	Status bit is unchanged.		
		Read 0x1:	Voltage transition complete event is true (pending).		
		Write 0x1:	Status bit is reset.		
1	WKUP2_ST	Wake-up event 2 status		RW	0
		Read 0x0:	Wake-up event is false.		
		Write 0x0:	Status bit is unchanged.		
		Read 0x1:	Wake-up event is true (pending).		
		Write 0x1:	Status bit is reset.		
0	WKUP1_ST	Wake-up event 1 status		RW	0
		Read 0x0:	Wake-up event is false.		
		Write 0x0:	Status bit is unchanged.		
		Read 0x1:	Wake-up event is true (pending).		
		Write 0x1:	Status bit is reset.		

Table 5−46. PRCM_IRQENABLE_MPU

Address Offset	0x01C		
Physical Address	0x4800 801C	**Instance**	PRCM
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																										TRANSITION_EN	EVGENOFF_EN	EVGENON_EN	VOLTTRANS_EN	WKUP2_EN	WKUP1_EN

Bits	Field Name	Description	Type	Reset
31:6	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000000
5	TRANSITION_EN	Transition complete event mask 0x0: Transition complete event is masked. 0x1: Transition complete event generates an interrupt.	RW	0
4	EVGENOFF_EN	Event generator end-OFF-time mask 0x0: End of OFF time event is masked. 0x1: End of OFF time event generates an interrupt.	RW	0
3	EVGENON_EN	Event generator end-ON-time mask 0x0: End of ON time event is masked. 0x1: End of ON time event generates an interrupt.	RW	0
2	VOLTTRANS_EN	Voltage transition complete event mask 0x0: Voltage transition complete event is masked. 0x1: Voltage transition complete event generates an interrupt.	RW	0
1	WKUP2_EN	Wake-up event 2 mask 0x0: Wake-up event is masked. 0x1: Wake-up event generates an interrupt.	RW	0
0	WKUP1_EN	Wake-up event 1 mask 0x0: Wake-up event is masked. 0x1: Wake-up event generates an interrupt.	RW	0

Table 5−47. PRCM_VOLTCTRL

Address Offset	0x050		
Physical Address	0x4800 8050	**Instance**	PRCM
Description	This register provides control over the main V_{DD} supply. It allows a direct control of the external power IC.		
Type	RW		

31	30	29	28	27	26	25	24	23	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_EXTVOLT	FORCE_EXTVOLT	SETOFF_LEVEL		Reserved			MEMRETCTRL	SETRET_LEVEL		Reserved				VOLT_LEVEL	

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000
15	AUTO_EXTVOLT	Auto voltage transition 0x0: VMODE signal is kept unchanged when all power domains are in retention state or OFF state. 0x1: VMODE signal is set to RETENTION or OFF voltage levels when all power domains are in retention state or OFF state, respectively.	RW	0
14	FORCE_EXTVOLT	Forces the voltage transition. 0x0: Update is automatically handled. 0x1: Immediate update of external VMODE signals with the VoltLevel value.	RW	0

Table 5−47. PRCM_VOLTCTRL (Continued)

Bits	Field Name	Description	Type	Reset
13:12	SETOFF_LEVEL	Sets the voltage level value corresponding to the device OFF mode 0x0: OFF is defined by voltage level 0. 0x1: OFF is defined by voltage level 1. 0x2: Reserved 0x3: Reserved	RW	0x1
11:9	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
8	MEMRETCTRL	Global memory retention control 0x0: Overall memory retention feature is disabled. 0x1: Overall memory retention feature is enabled.	RW	0
7:6	SETRET_LEVEL	Sets the voltage level value corresponding to the device RETENTION voltage 0x0: Retention is defined by voltage level 0. 0x1: Retention is defined by voltage level 1. 0x2: Reserved 0x3: Reserved	RW	0x1
5:2	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
1:0	VOLT_LEVEL	Sets the device core voltage. Controls the setting of the VMODE external signal. 0x0: Level 0 0x1: Level 1 0x2: Reserved 0x3: Reserved	RW	0x0

Table 5−48. PRCM_VOLTST

Address Offset	0x054		
Physical Address	0x4800 8054	**Instance**	PRCM
Description	This register provides the status of the main V_{DD} supply level.		
Type	R		

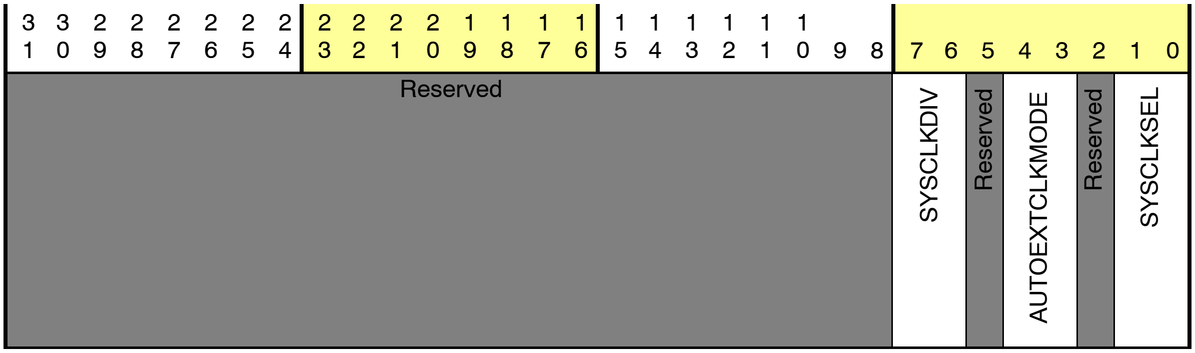
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																														ST_VOLT_LEVEL	

Table 5−48. PRCM_VOLTST (Continued)

Bits	Field Name	Description	Type	Reset
31:2	Reserved	Read returns 0.	RW	0x00000000
1:0	ST_VOLTLEVEL	Voltage level status 0x0: Current voltage level is L0. 0x1: Current voltage level is L1. 0x2: Reserved 0x3: Reserved	R	0x0

Table 5−49. PRCM_CLKSRC_CTRL

Address Offset	0x060		
Physical Address	0x4800 8060	**Instance**	PRCM
Description	This register provides control of the device source clock.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:8	Reserved	Write 0s for future compatibility. Read returns 0.	RW	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	Write 0s for future compatibility. Read returns 0.	RW	0

Table 5−49. PRCM_CLKSRC_CTRL (Continued)

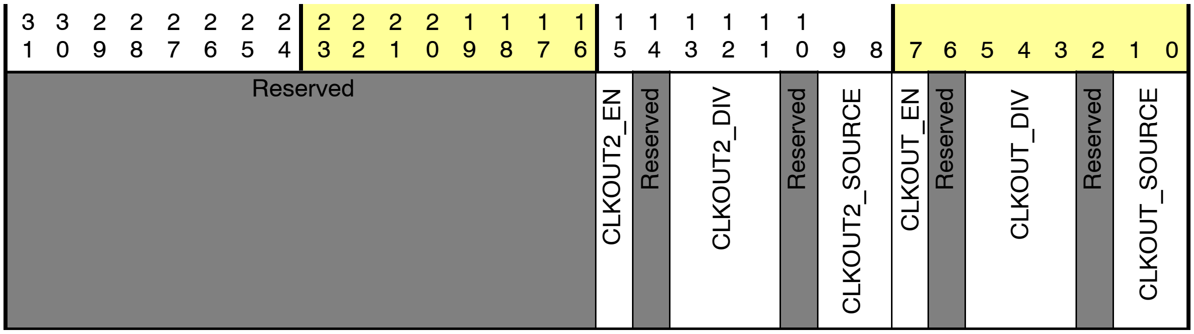
Bits	Field Name	Description	Type	Reset
4:3	AUTOEXTCLK-MODE	This field allows configuration of the external clock input control (CLKREQ) or the oscillator control. 0x0: CLKREQ is kept asserted or the oscillator is always active. 0x1: CLKREQ is deasserted or the oscillator is put in power-down mode only when all domains are in OFF state. 0x2: Reserved 0x3: CLKREQ is deasserted or the oscillator is put in power-down mode when all domains are in RETENTION or OFF state.	RW	0x0
2	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
1:0	SYSCLKSEL	This field selects the source of the system clock 0x0: External square clock input (oscillator is bypassed) 0x1: Oscillator input 0x2: Reserved 0x3: Default (oscillator is active on reset)	RW	0x3[1]

---

1) The SYSCLKSEL bit field reset value is 0x3. In this configuration, either an external square clock input or the internal oscillator can be used, but with low performance. For full performance for a given clock source, use 0x1 for the oscillator or 0x0 for the external square clock.

Table 5−50. PRCM_CLKOUT_CTRL

Address Offset	0x0070		
Physical Address	0x4800 8070	**Instance**	PRCM1
Description	This register provides control of the two device output clocks.		
Type	RW		



31	30	29	28	27	26	25	24	23	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	CLKOUT2_SOURCE		CLKOUT_EN	Reserved	CLKOUT_DIV			Reserved	CLKOUT_SOURCE	

Table 5−50.PRCM_CLKOUT_CTRL (Continued)

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x000000
15	CLKOUT2_EN	This bit controls the external output clock 2 activity. 0x0: SYS_CLKOUT2 is disabled. 0x1: SYS_CLKOUT2 is enabled.	RW	0
14	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
13:11	CLKOUT2_DIV	This field controls the external output clock 2 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
10	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
9:8	CLKOUT2_ SOURCE	This field selects the external output clock 2 source. 0x0: Source is CORE_CLK. 0x1: Source is system clock. 0x2: Source is 96-MHz clock. 0x3: Source is 54-MHz clock.	RW	0x3
7	CLKOUT_EN	This bit controls the external output clock activity. 0x0: SYS_CLKOUT is disabled. 0x1: SYS_CLKOUT is enabled.	RW	0
6	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
5:3	CLKOUT_DIV	This field controls the external output clock division. Other enums: Reserved 0x0: SYS_CLKOUT/1 0x1: SYS_CLKOUT/2 0x2: SYS_CLKOUT/4 0x3: SYS_CLKOUT/8 0x4: SYS_CLKOUT/16	RW	0x0
2	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
1:0	CLKOUT_ SOURCE	This field selects the external output clock source. 0x0: Source is CORE_CLK. 0x1: Source is system clock. 0x2: Source is 96-MHz clock. 0x3: Source is 54-MHz clock.	RW	0x3

Table 5−51. PRCM_CLKCFG_CTRL

Address Offset	0x080		
Physical Address	0x4800 8080	**Instance**	PRCM
Description	This register allows correct initiation and synchronization of a device clock configuration change.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															VALID_CONFIG

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000000
0	VALID_CONFIG	Valid clock configuration 0x0: Device clock configuration is validated. 0x1: When set to 1, this bit validates the clock configuration defined in the clock selection (All CM_CLKSEL_<domain> registers, except DSS1, DSS2, and GP timer clock sources selection) and ensures that clock changes are correctly handled and synchronized. Before setting this bit, the software must ensure that the newly selected clock configuration is correct by checking the PRCM_CLKCFG_ STATUS.CONFIGSTATUS bit (which, when set, prevents the clock configuration from validating). This bit is automatically cleared when the clocks are effectively changed.	RW	0

Table 5–52. PRCM_CLKCFG_STATUS

Address Offset	0x084		
Physical Address	0x4800 8084	**Instance**	PRCM
Description	This register logs the clock validation 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																															CONFIG_STATUS

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Read returns 0.	R	0x00000000
0	CONFIG_ STATUS	Flags invalid clock configuration 0x0: Clock configuration is validated. 0x1: Indicates that an error occurred because of missing dividers. When set, this bit prevents the PRCM_CLKCFG_CTRL.VALID_CONFIG bit from being set.	R	0

Table 5−53. PRCM_VOLTSETUP

Address Offset	0x090		
Physical Address	0x4800 8090	**Instance**	PRCM
Description	This register allows setting the setup time of the V_{DD} supply.		
Type	RW		

31	30	29	28	27	26	25	24	23	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	Write 0s for future compatibility. Read returns 0.	RW	0x0000
15:0	SETUP_TIME	Number of 32kHz clock cycles for the SETUP duration	RW	0x0000

Table 5−54. PRCM_CLKSSETUP

Address Offset	0x094		
Physical Address	0x4800 8094	**Instance**	PRCM
Description	This register allows setting the setup time of the oscillator system clock (sys_clk), based on number of 32 kHz 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
Reserved																SETUP_TIME															

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000
15:0	SETUP_TIME	Number of 32-kHz clock cycles for the SETUP duration	RW	0x0000

Table 5−55. PRCM_POLCTRL

Address Offset	0x0098		
Physical Address	0x4800 8098	**Instance**	PRCM1
Description	This register allows the polarity of device output control signals to be set.		
Type	RW		

31	30	29	28	27	26	25	24	23	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	CLKOUT_POL	CLKREQ_POL	Reserved							EXTVOL_POL

Bits	Field Name	Description	Type	Reset
31:11	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x000000
10	CLKOUT2_POL	Controls the external output clock 2 polarity when disabled 0x0: SYS_CLKOUT2 is active low. 0x1: SYS_CLKOUT2 is active high.	RW	0
9	CLKOUT_POL	Controls the external output clock polarity when disabled 0x0: SYS_CLKOUT is active low. 0x1: SYS_CLKOUT is active high.	RW	0
8	CLKREQ_POL	Controls the polarity of the CLKREQ signal 0x0: CLKREQ is active low. 0x1: CLKREQ is active high.	RW	1
7:1	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
0	EXTVOL_POL	Controls the polarity of the VMODE signal 0x0: VMODE signal is active low. 0x1: VMODE signal is active high.	RW	0

Table 5−56. GENERAL_PURPOSE1

Address Offset	0x0B0		
Physical Address	0x4800 80B0	**Instance**	PRCM
Description	This register allows storing information when domains are in OFF 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
DATA																															

Bits	**Field Name**	**Description**	**Type**	**Reset**
31:0	DATA	Data value	RW	0x00000000

Table 5−57. GENERAL_PURPOSE2

Address Offset	0x0B4		
Physical Address	0x4800 80B4	**Instance**	PRCM
Description	This register allows storing information when domains are in OFF 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
DATA																															

Bits	**Field Name**	**Description**	**Type**	**Reset**
31:0	DATA	Data value	RW	0x00000000

Table 5−58. GENERAL_PURPOSE3

Address Offset	0x0B8		
Physical Address	0x4800 80B8	**Instance**	PRCM
Description	This register allows storing information when domains are in OFF 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
DATA																															

Bits	**Field Name**	**Description**	**Type**	**Reset**
31:0	DATA	Data value	RW	0x00000000

Table 5−59. GENERAL_PURPOSE4

Address Offset	0x0BC		
Physical Address	0x4800 80BC	**Instance**	PRCM
Description	This register allows storing information when domains are in OFF 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
DATA																															

Bits	**Field Name**	**Description**	**Type**	**Reset**
31:0	DATA	Data value	RW	0x00000000

Table 5−60. GENERAL_PURPOSE5

Address Offset	0x0C0		
Physical Address	0x4800 80C0	**Instance**	PRCM
Description	This register allows storing information when domains are in OFF 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
DATA																															

Bits	**Field Name**	**Description**	**Type**	**Reset**
31:0	DATA	Data value	RW	0x00000000

Table 5−61. GENERAL_PURPOSE6

Address Offset	0x0C4		
Physical Address	0x4800 80C4	**Instance**	PRCM
Description	This register allows storing information when domains are in OFF 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
DATA																															

Bits	**Field Name**	**Description**	**Type**	**Reset**
31:0	DATA	Data value	RW	0x00000000

Table 5−62. GENERAL_PURPOSE7

Address Offset	0x0C8		
Physical Address	0x4800 80C8	**Instance**	PRCM
Description	This register allows storing information when domains are in OFF 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
DATA																															

Bits	Field Name	Description	Type	Reset
31:0	DATA	Data value	RW	0x00000000

Table 5−63. GENERAL_PURPOSE8

Address Offset	0x0CC		
Physical Address	0x4800 80CC	**Instance**	PRCM
Description	This register allows storing information when domains are in OFF 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
DATA																															

Bits	Field Name	Description	Type	Reset
31:0	DATA	Data value	RW	0x00000000

Table 5−64. GENERAL_PURPOSE9

Address Offset	0x0D0		
Physical Address	0x4800 80D0	**Instance**	PRCM
Description	This register allows storing information when domains are in OFF 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
DATA																															

Bits	Field Name	Description	Type	Reset
31:0	DATA	Data value	RW	0x00000000

Table 5−65. GENERAL_PURPOSE10

Address Offset	0x0D4		
Physical Address	0x4800 80D4	**Instance**	PRCM
Description	This register allows storing information when domains are in OFF 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
DATA																															

Bits	Field Name	Description	Type	Reset
31:0	DATA	Data value	RW	0x00000000

Table 5−66. GENERAL_PURPOSE11

Address Offset	0x0D8		
Physical Address	0x4800 80D8	**Instance**	PRCM
Description	This register allows storing information when domains are in OFF 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
DATA																															

Bits	Field Name	Description	Type	Reset
31:0	DATA	Data value	RW	0x00000000

Table 5−67. GENERAL_PURPOSE12

Address Offset	0x0DC		
Physical Address	0x4800 80DC	**Instance**	PRCM
Description	This register allows storing information when domains are in OFF 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
DATA																															

Bits	Field Name	Description	Type	Reset
31:0	DATA	Data value	RW	0x00000000

Table 5−68. GENERAL_PURPOSE13

Address Offset	0x0E0		
Physical Address	0x4800 80E0	**Instance**	PRCM
Description	This register allows storing information when domains are in OFF 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
DATA																															

Bits	Field Name	Description	Type	Reset
31:0	DATA	Data value	RW	0x00000000

Table 5−69. GENERAL_PURPOSE14

Address Offset	0x0E4		
Physical Address	0x4800 80E4	**Instance**	PRCM
Description	This register allows storing information when domains are in OFF 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
DATA																															

Bits	Field Name	Description	Type	Reset
31:0	DATA	Data value	RW	0x00000000

Table 5−70. GENERAL_PURPOSE15

Address Offset	0x0E8		
Physical Address	0x4800 80E8	**Instance**	PRCM
Description	This register allows storing information when domains are in OFF 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
DATA																															

Bits	Field Name	Description	Type	Reset
31:0	DATA	Data value	RW	0x00000000

Table 5−71. GENERAL_PURPOSE16

Address Offset	0x0EC		
Physical Address	0x4800 80EC	**Instance**	PRCM
Description	This register allows storing information when domains are in OFF 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
DATA																															

Bits	**Field Name**	**Description**	**Type**	**Reset**
31:0	DATA	Data value	RW	0x00000000

Table 5−72. GENERAL_PURPOSE17

Address Offset	0x0F0		
Physical Address	0x4800 80F0	**Instance**	PRCM
Description	This register allows storing information when domains are in OFF 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
DATA																															

Bits	**Field Name**	**Description**	**Type**	**Reset**
31:0	DATA	Data value	RW	0x00000000

Table 5−73. GENERAL_PURPOSE18

Address Offset	0x0F4		
Physical Address	0x4800 80F4	**Instance**	PRCM
Description	This register allows storing information when domains are in OFF 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
DATA																															

Bits	**Field Name**	**Description**	**Type**	**Reset**
31:0	DATA	Data value	RW	0x00000000

Table 5−74. GENERAL_PURPOSE19

Address Offset	0x0F8		
Physical Address	0x4800 80F8	**Instance**	PRCM
Description	This register allows storing information when domains are in OFF mode.		
Type	RW		

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="32">DATA</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
31:0	DATA	Data value	RW	0x00000000

Table 5−75. GENERAL_PURPOSE20

Address Offset	0x0FC		
Physical Address	0x4800 80FC	**Instance**	PRCM
Description	This register allows storing information when domains are in OFF mode.		
Type	RW		

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="32">DATA</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
31:0	DATA	Data value	RW	0x00000000

Table 5−76. CM_CLKSEL_MPU

Address Offset	0x140		
Physical Address	0x4800 8140	**Instance**	PRCM
Description	MPU core clock selection. This clock is divided from Core_clk (DPLL output).		
Type	RW		

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="27">Reserved</td><td colspan="5">CLKSEL_MPU</td></tr>
</table>

Table 5−76. CM_CLKSEL_MPU (Continued)

Bits	Field Name	Description	Type	Reset
31:5	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000000
4:0	CLKSEL_MPU	MPU clock selection; other enums: Reserved 0x0: Reserved 0x1: Core_clk/1 used by clock configuration VII and VIII 0x2: Core_clk/2 used by clock configuration I, II, III, and IV 0x4: Core_clk/4[1] 0x6: Core_clk/6[1] 0x8: Core_clk/8[1]	RW	0x01

1) Value not recommended by the regular OMAP2420 clock configurations. PRCM configuration is not validated at the OMAP2420 level.

Table 5−77. CM_CLKSTCTRL_MPU

Address Offset	0x148		
Physical Address	0x4800 8148	**Instance**	PRCM
Description	This register controls the hardware-supervised 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																															AUTOSTATE_MPU

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000000
0	AUTOSTATE_MPU	Auto State control 0x0: Hardware-supervised control of MPU domain is disabled. 0x1: Hardware-supervised control of MPU domain is enabled.	RW	0

Table 5−78. RM_RSTST_MPU

Address Offset	0x158		
Physical Address	0x4800 8158	**Instance**	PRCM
Description	This register logs the different reset sources of the MPU domain. Each bit is set on release of the respective reset source 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																												COREWKUP_RST	DOMAINWKUP_RST	GLOBALWMPU_RST	GLOBALCOLD_RST

Bits	Field Name	Description	Type	Reset
31:4	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000000
3	COREWKUP_RST	Core domain wake-up reset 0x0: No core domain reset Write 0x0: Status bit is unchanged. 0x1: MPU domain is reset with a CORE power domain reset. Write 0x1: Status bit is reset.	RW	0
2	DOMAINWKUP_RST	Power domain wake-up reset 0x0: No power domain wake-up reset Write 0x0: Status bit is unchanged. 0x1: MPU domain is reset following an MPU power domain wake-up. Write 0x1: Status bit is reset.	RW	0
1	GLOBAL WMPU_RST	Global warm reset 0x0: No global warm reset Write 0x0: Status bit is unchanged. 0x1: MPU domain is reset on a global warm reset. Write 0x1: Status bit is reset.	RW	0
0	GLOBAL-COLD_RST	Global cold reset 0x0: No global cold reset Write 0x0: Status bit is unchanged. 0x1: MPU domain is reset on a global cold reset. Write 0x1: Status bit is reset.	RW	1

Table 5−79. PM_WKDEP_MPU

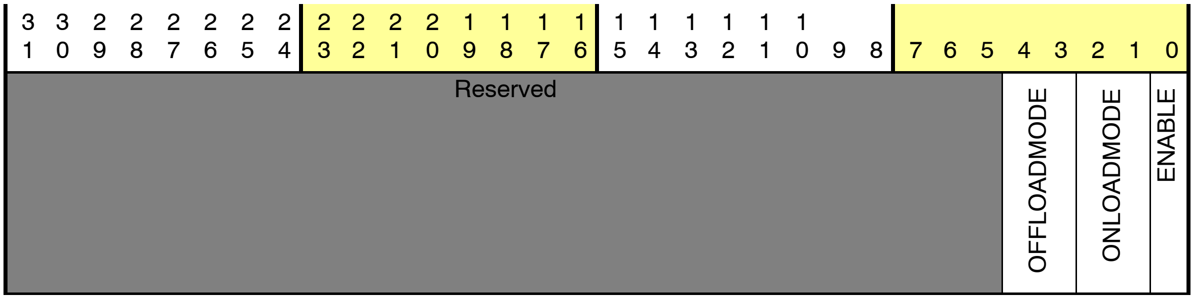
Address Offset	0x1C8		
Physical Address	0x4800 81C8	**Instance**	PRCM
Description	This register allows enabling or disabling the wake-up of the MPU domain on a wake-up event of another domain.		
Type	RW		

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_WKUP	Reserved	Reserved	Reserved	EN_CORE

Bits	Field Name	Description	Type	Reset
31:5	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000000
4	EN_WKUP	Wake-up domain dependency 0x0: MPU domain is independent of WKUP domain module wake-up event. 0x1: MPU domain is waked up on WKUP domain module wake-up event.	RW	1
3	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
2	Reserved	Reserved	RW	1
1	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
0	EN_CORE	Core domain dependency 0x0: MPU domain is independent of core domain wake-up event. 0x1: MPU domain is waked up on core domain wake-up event.	RW	1

Table 5−80. PM_EVGENCTRL_MPU

Address Offset	0x1D4		
Physical Address	0x4800 81D4	**Instance**	PRCM
Description	This register allows control 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	Write 0s for future compatibility . Read returns 0.	RW	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: Reserved 0x2: Load on MPU standby signal deassertion 0x3: Auto load	RW	0x2
0	ENABLE	Event generator control 0x0: Disable event generator. 0x1: Enable event generator.	RW	0

Table 5−81. PM_EVEGENONTIM_MPU

Address Offset	0x1D8		
Physical Address	0x4800 81D8	**Instance**	PRCM
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

Table 5−82. PM_EVEGENOFFTIM_MPU

Address Offset	0x1DC		
Physical Address	0x4800 81DC	**Instance**	PRCM
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

Table 5−83. PM_PWSTCTRL_MPU

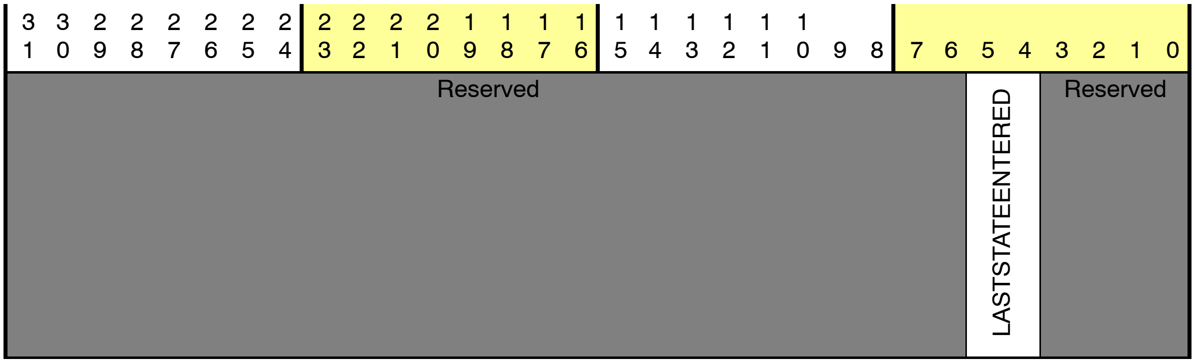
Address Offset	0x1E0		
Physical Address	0x4800 81E0	**Instance**	PRCM
Description	This register controls the MPU domain power state transition.		
Type	RW		

31	30	29	28	27	26	25	24	23	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	LOGICRETSTATE	POWERSTATE	

Bits	Field Name	Description		Type	Reset
31:12	Reserved	Write 0s for future compatibility. Read returns 0.		RW	0x00000
11:10	MEMONSTATE	Memory state when domain is ON		R	0x0
		0x0:	Memory is always ON when domain is ON.		
9:4	Reserved	Write 0s for future compatibility. Read returns 0.		RW	0x00
3	MEMRETSTATE	Memory state when domain is RETENTION		R	1
		0x1:	Cache memory is always retained when domain is in RETENTION state.		
2	LOGICRETSTATE	Logic state when domain is RETENTION		RW	1
		0x0:	Logic is OFF when domain is in RETENTION state.		
		0x1:	Logic is retained when domain is in RETENTION state.		
1:0	POWERSTATE	Power state control		RW	0x0
		0x0:	ON state		
		0x1:	RETENTION state		
		0x2:	Reserved		
		0x3:	OFF state		

Table 5−84. PM_PWSTST_MPU

Address Offset	0x1E4		
Physical Address	0x4800 81E4	**Instance**	PRCM
Description	This register provides a status on the MPU domain power state.		
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																										LASTSTATEENTERED		Reserved			

Bits	Field Name	Description	Type	Reset
31:6	Reserved	Read returns 0.	R	0x0000000
5:4	LASTSTATE ENTERED	Last power state entered 0x0: MPU domain was ON. 0x1: MPU domain was in RETENTION. 0x2: Reserved 0x3: MPU domain was OFF.	R	0x3
3:0	Reserved	Read returns 0.	R	0x0

Table 5−85. CM_FCLKEN1_CORE

Address Offset	0x200		
Physical Address	0x4800 8200	**Instance**	PRCM
Description	Controls module functional clock activity		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN_CAM	Reserved	EN_WDT4	EN_WDT3	Reserved	EN_MMC	EN_FAC	EN_EAC	EN_HDQ	EN_UART2	EN_UART1	EN_I2C2	EN_I2C1	EN_MCSPI2	EN_MCSPI1	EN_MCBSP2

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN_MCBSP1	EN_GPT12	EN_GPT11	EN_GPT10	EN_GPT9	EN_GPT8	EN_GPT7	EN_GPT6	EN_GPT5	EN_GPT4	EN_GPT3	EN_GPT2	Reserved	EN_TV	EN_DSS2	EN_DSS1

Bits	Field Name	Description	Type	Reset
31	EN_CAM	Camera 96-MHz clock control 0x0: Camera 96-MHz clock is disabled. 0x1: Camera 96-MHz clock is enabled.	RW	0
30	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
29	EN_WDT4	Watchdog timer 4 clock control 0x0: Watchdog timer 4 functional clock is disabled. 0x1: Watchdog timer 4 functional clock is enabled.	RW	0
28	EN_WDT3	Watchdog timer 3 clock control 0x0: Watchdog timer 3 functional clock is disabled. 0x1: Watchdog timer 3 functional clock is enabled.	RW	0
27	Reserved	Reserved	RW	0
26	EN_MMC	MMC functional clock control 0x0: MMC functional clock is disabled. 0x1: MMC functional clock is enabled.	RW	0
25	EN_FAC	FAC functional clock control 0x0: FAC functional clock is disabled. 0x1: FAC functional clock is enabled.	RW	0
24	EN_EAC	EAC functional clock control 0x0: EAC functional clock is disabled. 0x1: EAC functional clock is enabled.	RW	0

Table 5−85. CM_FCLKEN1_CORE (Continued)

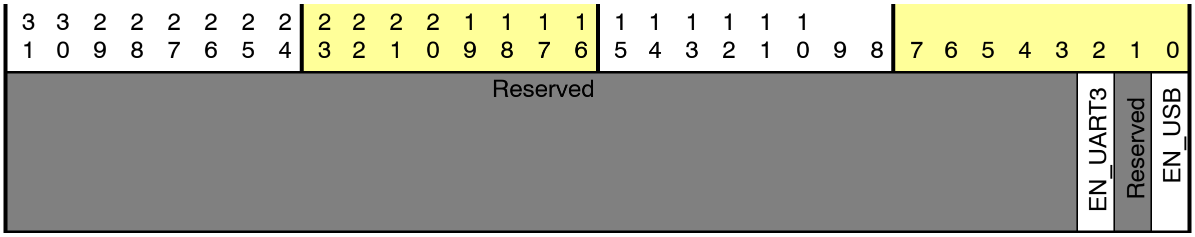
Bits	Field Name	Description	Type	Reset
23	EN_HDQ	HDQ functional clock control 0x0: HDQ functional clock is disabled. 0x1: HDQ functional clock is enabled.	RW	0
22	EN_UART2	UART 2 functional clock control 0x0: UART 2 functional clock is disabled. 0x1: UART 2 functional clock is enabled.	RW	0
21	EN_UART1	UART 1 functional clock control 0x0: UART 1/ IRda functional clock is disabled. 0x1: UART 1/IRda functional clock is enabled.	RW	0
20	EN_I2C2	I2C 2 functional clock control 0x0: I2C 2 functional clock is disabled. 0x1: I2C 2 functional clock is enabled.	RW	0
19	EN_I2C1	I2C 1 functional clock control 0x0: I2C 1 functional clock is disabled. 0x1: I2C 1 functional clock is enabled.	RW	0
18	EN_MCSPI2	MsSPI 2 functional clock control 0x0: McSPI 2 functional clock is disabled. 0x1: McSPI 2 functional clock is enabled.	RW	0
17	EN_MCSPI1	MsSPI 1 functional clock control 0x0: McSPI 1 functional clock is disabled. 0x1: McSPI 1 functional clock is enabled.	RW	0
16	EN_MCBSP2	McBSP 2 functional clock control 0x0: McBSP 2 functional clock is disabled. 0x1: McBSP 2 functional clock is enabled.	RW	0
15	EN_MCBSP1	McBSP 1 functional clock control 0x0: McBSP 1 functional clock is disabled. 0x1: McBSP 1 functional clock is enabled.	RW	0
14	EN_GPT12	GPT 12 functional clock control 0x0: GP Timer 12 functional clock is disabled. 0x1: GP Timer 12 functional clock is enabled.	RW	0
13	EN_GPT11	GPT 11 functional clock control 0x0: GP Timer 11 functional clock is disabled. 0x1: GP Timer 11 functional clock is enabled.	RW	0
12	EN_GPT10	GPT 10 functional clock control 0x0: GP Timer 10 functional clock is disabled. 0x1: GP Timer 10 functional clock is enabled.	RW	0

Table 5−85. CM_FCLKEN1_CORE (Continued)

Bits	Field Name	Description		Type	Reset
11	EN_GPT9	GPT 9 functional clock control		RW	0
		0x0:	GP Timer 9 functional clock is disabled.		
		0x1:	GP Timer 9 functional clock is enabled.		
10	EN_GPT8	GPT 8 functional clock control		RW	0
		0x0:	GP Timer 8 functional clock is disabled.		
		0x1:	GP Timer 8 functional clock is enabled.		
9	EN_GPT7	GPT 7 functional clock control		RW	0
		0x0:	GP Timer 7 functional clock is disabled.		
		0x1:	GP Timer 7 functional clock is enabled.		
8	EN_GPT6	GPT 6 functional clock control		RW	0
		0x0:	GP Timer 6 functional clock is disabled.		
		0x1:	GP Timer 6 functional clock is enabled.		
7	EN_GPT5	GPT 5 functional clock control		RW	0
		0x0:	GP Timer 5 functional clock is disabled.		
		0x1:	GP Timer 5 functional clock is enabled.		
6	EN_GPT4	GPT 4 functional clock control		RW	0
		0x0:	GP Timer 4 functional clock is disabled.		
		0x1:	GP Timer 4 functional clock is enabled.		
5	EN_GPT3	GPT 3 functional clock control		RW	0
		0x0:	GP Timer 3 functional clock is disabled.		
		0x1:	GP Timer 3 functional clock is enabled.		
4	EN_GPT2	GPT 2 functional clock control		RW	0
		0x0:	GP Timer 2 functional clock is disabled.		
		0x1:	GP Timer 2 functional clock is enabled.		
3	Reserved	Reserved		RW	0
2	EN_TV	TV OUT functional clock control		RW	0
		0x0:	TV remote functional clock is disabled.		
		0x1:	TV remote functional clock is enabled.		
1	EN_DSS2	Display subsystem functional clock 2 control		RW	0
		0x0:	Display subsystem functional clock 2 is disabled.		
		0x1:	Display subsystem functional clock 2 is enabled.		
0	EN_DSS1	Display subsystem functional clock 1 control		RW	0
		0x0:	Display subsystem functional clock 1 is disabled.		
		0x1:	Display subsystem functional clock 1 is enabled.		

Table 5−86. CM_FCLKEN2_CORE

Address Offset	0x204		
Physical Address	0x4800 8204	**Instance**	PRCM
Description	Controls module functional clock activity		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:3	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000000
2	EN_UART3	UART 3 functional clock control 0x0: UART 3 functional clock is disabled. 0x1: UART 3 functional clock is enabled.	RW	1
1	Reserved	Reserved	RW	1
0	EN_USB	USB functional clock control 0x0: USB functional clock is disabled. 0x1: USB functional clock is enabled.	RW	1

Table 5−87. CM_ICLKEN1_CORE

Address Offset	0x210		
Physical Address	0x4800 8210	**Instance**	PRCM
Description	Controls module interface clock activity		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN_CAM	EN_MAILBOXES	EN_WDT4	EN_WDT3	Reserved	EN_MMC	EN_FAC	EN_EAC	EN_HDQ	EN_UART2	EN_UART1	EN_I2C2	EN_I2C1	EN_MCSPI2	EN_MCSPI1	EN_MCBSP2

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN_MCBSP1	EN_GPT12	EN_GPT11	EN_GPT10	EN_GPT9	EN_GPT8	EN_GPT7	EN_GPT6	EN_GPT5	EN_GPT4	EN_GPT3	EN_GPT2	Reserved	Reserved		EN_DSS

Bits	Field Name	Description		Type	Reset
31	EN_CAM	Camera interface clock control		RW	0
		0x0:	Camera interface clock is disabled.		
		0x1:	Camera interface clock is enabled.		
30	EN_MAILBOXES	Mailboxes interface clock control		RW	0
		0x0:	Mailboxes interface clock is disabled.		
		0x1:	Mailboxes interface clock is enabled.		
29	EN_WDT4	Watchdog timer 4 interface clock control		RW	0
		0x0:	Watchdog timer 4 interface clock is disabled.		
		0x1:	Watchdog timer 4 interface clock is enabled.		
28	EN_WDT3	Watchdog timer 3 interface clock control		RW	0
		0x0:	Watchdog timer 3 interface clock is disabled.		
		0x1:	Watchdog timer 3 interface clock is enabled.		
27	Reserved	Reserved		RW	0
26	EN_MMC	MMC interface clock control		RW	0
		0x0:	MMC interface clock is disabled.		
		0x1:	MMC interface clock is enabled.		
25	EN_FAC	FAC interface clock control		RW	0
		0x0:	FAC interface clock is disabled.		
		0x1:	FAC interface clock is enabled.		

Table 5−87. CM_ICLKEN1_CORE (Continued)

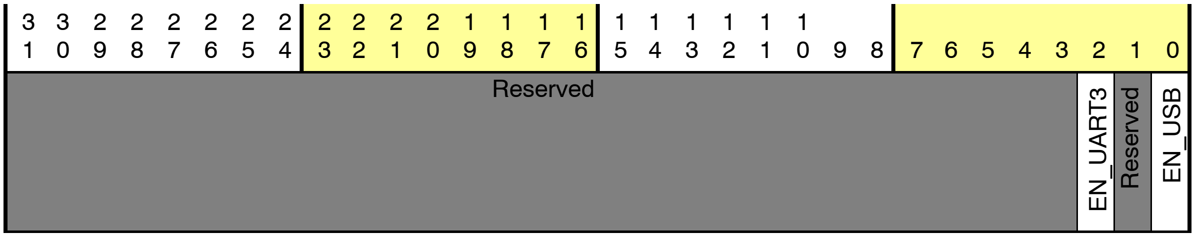
Bits	Field Name	Description	Type	Reset
24	EN_EAC	EAC interface clock control 0x0: EAC interface clock is disabled. 0x1: EAC interface clock is enabled.	RW	0
23	EN_HDQ	HDQ interface clock control 0x0: HDQ interface clock is disabled. 0x1: HDQ interface clock is enabled.	RW	0
22	EN_UART2	UART 2 interface clock control 0x0: UART 2 interface clock is disabled. 0x1: UART 2 interface clock is enabled.	RW	0
21	EN_UART1	UART 1 interface clock control 0x0: UART 1/ IRda interface clock is disabled. 0x1: UART 1/IRda interface clock is enabled.	RW	0
20	EN_I2C2	I2C 2 interface clock control 0x0: I2C 2 interface clock is disabled. 0x1: I2C 2 interface clock is enabled.	RW	0
19	EN_I2C1	I2C 1 interface clock control 0x0: I2C 1 interface clock is disabled. 0x1: I2C 1 interface clock is enabled.	RW	0
18	EN_MCSPI2	MsSPI 2 interface clock control 0x0: McSPI 2 interface clock is disabled. 0x1: McSPI 2 interface clock is enabled.	RW	0
17	EN_MCSPI1	MsSPI 1 interface clock control 0x0: McSPI 1 interface clock is disabled. 0x1: McSPI 1 interface clock is enabled.	RW	0
16	EN_MCBSP2	McBSP 2 interface clock control 0x0: McBSP 2 interface clock is disabled. 0x1: McBSP 2 interface clock is enabled.	RW	0
15	EN_MCBSP1	McBSP 1 interface clock control 0x0: McBSP 1 interface clock is disabled. 0x1: McBSP 1 interface clock is enabled.	RW	0
14	EN_GPT12	GPT 12 interface clock control 0x0: GP Timer 12 interface clock is disabled. 0x1: GP Timer 12 interface clock is enabled.	RW	0
13	EN_GPT11	GPT 11 interface clock control 0x0: GP Timer 11 interface clock is disabled. 0x1: GP Timer 11 interface clock is enabled.	RW	0

Table 5−87. CM_ICLKEN1_CORE (Continued)

Bits	Field Name	Description	Type	Reset
12	EN_GPT10	GPT 10 interface clock control 0x0: GP Timer 10 interface clock is disabled. 0x1: GP Timer 10 interface clock is enabled.	RW	0
11	EN_GPT9	GPT 9 interface clock control 0x0: GP Timer 9 interface clock is disabled. 0x1: GP Timer 9 interface clock is enabled.	RW	0
10	EN_GPT8	GPT 8 interface clock control 0x0: GP Timer 8 interface clock is disabled. 0x1: GP Timer 8 interface clock is enabled.	RW	0
9	EN_GPT7	GPT 7 interface clock control 0x0: GP Timer 7 interface clock is disabled. 0x1: GP Timer 7 interface clock is enabled.	RW	0
8	EN_GPT6	GPT 6 interface clock control 0x0: GP Timer 6 interface clock is disabled. 0x1: GP Timer 6 interface clock is enabled.	RW	0
7	EN_GPT5	GPT 5 interface clock control 0x0: GP Timer 5 interface clock is disabled. 0x1: GP Timer 5 interface clock is enabled.	RW	0
6	EN_GPT4	GPT 4 interface clock control 0x0: GP Timer 4 interface clock is disabled. 0x1: GP Timer 4 interface clock is enabled.	RW	0
5	EN_GPT3	GPT 3 interface clock control 0x0: GP Timer 3 interface clock is disabled. 0x1: GP Timer 3 interface clock is enabled.	RW	0
4	EN_GPT2	GPT 2 interface clock control 0x0: GP Timer 2 interface clock is disabled. 0x1: GP Timer 2 interface clock is enabled.	RW	0
3	Reserved	Reserved	RW	0
2:1	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
0	EN_DSS	Display subsystem interface clock control 0x0: Display subsystem interface clock is disabled. 0x1: Display subsystem interface clock is enabled.	RW	0

Table 5−88. CM_ICLKEN2_CORE

Address Offset	0x214		
Physical Address	0x4800 8214	**Instance**	PRCM
Description	Controls module interface clock activity		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:3	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000000
2	EN_UART3	UART 3 interface clock control 0x0: UART 3 interface clock is disabled. 0x1: UART 3 interface clock is enabled.	RW	1
1	Reserved	Reserved	RW	1
0	EN_USB	USB functional clock control 0x0: USB interface clock is disabled. 0x1: USB interface clock is enabled.	RW	1

Table 5−89. CM_IDLEST1_CORE

Address Offset	0x220		
Physical Address	0x4800 8220	**Instance**	PRCM
Description	Core module access availability monitoring. This register is read-only and is 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_MAILBOXES	ST_WDT4	ST_WDT3	Reserved	ST_MMC	ST_FAC	ST_EAC	ST_HDQ	ST_UART2	ST_UART1	ST_I2C2	ST_I2C1	ST_MCSPI2	ST_MCSPI1	ST_MCBSP2	ST_MCBSP1	ST_GPT12	ST_GPT11	ST_GPT10	ST_GPT9	ST_GPT8	ST_GPT7	ST_GPT6	ST_GPT5	ST_GPT4	ST_GPT3	ST_GPT2	Reserved	Reserved		ST_DSS

Bits	Field Name	Description	Type	Reset
31	Reserved	Read returns 0.	R	0
30	ST_MAILBOXES	Mailboxes idle status 0x0: Mailboxes cannot be accessed. An access attempt can return an error. 0x1: Mailboxes can be accessed.	R	0
29	ST_WDT4	Watchdog timer 4 idle status 0x0: Watchdog timer 4 cannot be accessed. An access attemtp can return an error. 0x1: Watchdog timer 4 can be accessed.	R	0
28	ST_WDT3	Watchdog timer 3 idle status 0x0: Watchdog timer 3 cannot be accessed. An access attempt can return an error. 0x1: Watchdog timer 3 can be accessed.	R	0
27	Reserved	Reserved	R	0
26	ST_MMC	MMC idle status 0x0: MMC cannot be accessed. An access attempt can return an error. 0x1: MMC can be accessed.	R	0

Table 5−89. CM_IDLEST1_CORE (Continued)

Bits	Field Name	Description	Type	Reset
25	ST_FAC	FAC idle status 0x0: FAC cannot be accessed. An access attempt can return an error. 0x1: FAC can be accessed.	R	0
24	ST_EAC	EAC idle status 0x0: EAC cannot be accessed. An access attempt can return an error. 0x1: EAC can be accessed.	R	0
23	ST_HDQ	HDQ idle status 0x0: HDQ cannot be accessed. An access attempt can return an error. 0x1: HDQ can be accessed.	R	0
22	ST_UART2	UART 2 idle status 0x0: UART 2 cannot be accessed. An access attempt can return an error. 0x1: UART 2 can be accessed.	R	0
21	ST_UART1	UART 1 idle status 0x0: UART 1/ IRda cannot be accessed. An access attempt can return an error. 0x1: UART 1/IRda can be accessed.	R	0
20	ST_I2C2	I2C 2 idle status 0x0: I2C 2 cannot be accessed. An access attempt can return an error. 0x1: I2C 2 can be accessed.	R	0
19	ST_I2C1	I2C 1 idle status 0x0: I2C 1 cannot be accessed. An access attempt can return an error. 0x1: I2C 1 can be accessed.	R	0
18	ST_MCSPI2	MsSPI 2 idle status 0x0: McSPI 2 cannot be accessed. An access attempt can return an error. 0x1: McSPI 2 can be accessed.	R	0
17	ST_MCSPI1	MsSPI 1 idle status 0x0: McSPI 1 cannot be accessed. An access attempt can return an error. 0x1: McSPI 1 can be accessed.	R	0
16	ST_MCBSP2	McBSP 2 idle status 0x0: McBSP 2 cannot be accessed. An access attempt can return an error. 0x1: McBSP 2 can be accessed.	R	0

Table 5−89. CM_IDLEST1_CORE (Continued)

Bits	Field Name	Description	Type	Reset
15	ST_MCBSP1	McBSP 1 idle status 0x0: McBSP 1 cannot be accessed. An access attempt can return an error. 0x1: McBSP 1 can be accessed.	R	0
14	ST_GPT12	GPT 12 idle status 0x0: GP Timer 12 cannot be accessed. An access attempt can return an error. 0x1: GP Timer 12 can be accessed.	R	0
13	ST_GPT11	GPT 11 idle status 0x0: GP Timer 11 cannot be accessed. An access attempt can return an error. 0x1: GP Timer 11 can be accessed.	R	0
12	ST_GPT10	GPT 10 idle status 0x0: GP Timer 10 cannot be accessed. An access attempt can return an error. 0x1: GP Timer 10 can be accessed.	R	0
11	ST_GPT9	GPT 9 idle status 0x0: GP Timer 9 cannot be accessed. An access attempt can return an error. 0x1: GP Timer 9 can be accessed.	R	0
10	ST_GPT8	GPT 8 idle status 0x0: GP Timer 8 cannot be accessed. An access attempt can return an error. 0x1: GP Timer 8 can be accessed.	R	0
9	ST_GPT7	GPT 7 idle status 0x0: GP Timer 7 cannot be accessed. An access attempt can return an error. 0x1: GP Timer 7 can be accessed.	R	0
8	ST_GPT6	GPT 6 idle status 0x0: GP Timer 6 cannot be accessed. An access attempt can return an error. 0x1: GP Timer 6 can be accessed.	R	0
7	ST_GPT5	GPT 5 idle status 0x0: GP Timer 5 cannot be accessed. An access attempt can return an error. 0x1: GP Timer 5 can be accessed.	R	0
6	ST_GPT4	GPT 4 idle status 0x0: GP Timer 4 cannot be accessed. An access attempt can return an error. 0x1: GP Timer 4 can be accessed.	R	0

Table 5−89. CM_IDLEST1_CORE (Continued)

Bits	Field Name	Description	Type	Reset
5	ST_GPT3	GPT 3 idle status 0x0: GP Timer 3 cannot be accessed. An access attempt can return an error. 0x1: GP Timer 3 can be accessed.	R	0
4	ST_GPT2	GPT 2 idle status 0x0: GP Timer 2 cannot be accessed. An access attempt can return an error. 0x1: GP Timer 2 can be accessed.	R	0
3	Reserved	Reserved	R	0
2:1	Reserved	Read returns 0.	R	0x0
0	ST_DSS	Display subsystem standby status 0x0: Display is in standby mode. 0x1: Display subsystem is in normal mode.	R	0

Table 5−90. CM_IDLEST2_CORE

Address Offset	0x224		
Physical Address	0x4800 8224	**Instance**	PRCM
Description	Core module access availability monitoring. This register is read-only and is 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_UART3	Reserved	ST_USB

Bits	Field Name	Description	Type	Reset
31:3	Reserved	Read returns 0.	R	0x00000000
2	ST_UART3	UART 3 idle status 0x0: UART 3 cannot be accessed. An access attempt can return an error. 0x1: UART 3 can be accessed.	R	0
1	Reserved	Reserved	R	0
0	ST_USB	USB idle status 0x0: USB cannot be accessed. An access attempt can return an error. 0x1: USB can be accessed.	R	0

Table 5−91. CM_AUTOIDLE1_CORE

Address Offset	0x230		
Physical Address	0x4800 8230	**Instance**	PRCM
Description	This register allows automatic control of core 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
AUTO_CAM	AUTO_MAILBOXES	AUTO_WDT4	AUTO_WDT3	Reserved	AUTO_MMC	AUTO_FAC	AUTO_EAC	AUTO_HDQ	AUTO_UART2	AUTO_UART1	AUTO_I2C2	AUTO_I2C1	AUTO_MCSPI2	AUTO_MCSPI1	AUTO_MCBSP2	AUTO_MCBSP1	AUTO_GPT12	AUTO_GPT11	AUTO_GPT10	AUTO_GPT9	AUTO_GPT8	AUTO_GPT7	AUTO_GPT6	AUTO_GPT5	AUTO_GPT4	AUTO_GPT3	AUTO_GPT2	Reserved	Reserved		AUTO_DSS

Bits	Field Name	Description	Type	Reset
31	AUTO_CAM	Camera auto clock control 0x0: Camera clock is unrelated to the domain state transition. 0x1: Camera clock is automatically enabled or disabled with the domain state transition.	RW	0
30	AUTO_MAIL-BOXES	Mailboxes auto clock control 0x0: Mailboxes interface clock is unrelated to the domain state transition. 0x1: Mailboxes interface clock is automatically enabled or disabled with the domain state transition.	RW	0
29	AUTO_WDT4	Watchdog timer 4 auto clock control 0x0: Watchdog timer 4 interface clock is unrelated to the domain state transition. 0x1: Watchdog timer 4 interface clock is automatically enabled or disabled with the domain state transition.	RW	0
28	AUTO_WDT3	Watchdog timer 3 auto clock control 0x0: Watchdog timer 3 interface clock is unrelated to the domain state transition. 0x1: Watchdog timer 3 interface clock is automatically enabled or disabled with the domain state transition.	RW	0

Table 5−91. CM_AUTOIDLE1_CORE (Continued)

Bits	Field Name	Description	Type	Reset
27	Reserved	Reserved	RW	0
26	AUTO_MMC	MMC auto clock control 0x0: MMC interface clock is unrelated to the domain state transition. 0x1: MMC interface clock is automatically enabled or disabled with the domain state transition.	RW	0
25	AUTO_FAC	FAC auto clock control 0x0: FAC interface clock is unrelated to the domain state transition. 0x1: FAC interface clock is automatically enabled or disabled with the domain state transition.	RW	0
24	AUTO_EAC	EAC auto clock control 0x0: EAC interface clock is unrelated to the domain state transition. 0x1: EAC interface clock is automatically enabled or disabled with the domain state transition.	RW	0
23	AUTO_HDQ	HDQ auto clock control 0x0: HDQ interface clock is unrelated to the domain state transition. 0x1: HDQ interface clock is automatically enabled or disabled with the domain state transition.	RW	0
22	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 with the domain state transition.	RW	0
21	AUTO_UART1	UART 1 auto clock control 0x0: UART 1/ IRda interface clock is unrelated to the domain state transition. 0x1: UART 1/IRda interface clock is automatically enabled or disabled with the domain state transition.	RW	0
20	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 with the domain state transition.	RW	0

Table 5−91. CM_AUTOIDLE1_CORE (Continued)

Bits	Field Name	Description	Type	Reset
19	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 with the domain state transition.	RW	0
18	AUTO_MCSPI2	MsSPI 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 with the domain state transition.	RW	0
17	AUTO_MCSPI1	MsSPI 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 with the domain state transition.	RW	0
16	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 with the domain state transition.	RW	0
15	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 with the domain state transition.	RW	0
14	AUTO_GPT12	GPT 12 auto clock control 0x0: GP Timer 12 interface clock is unrelated to the domain state transition. 0x1: GP Timer 12 interface clock is automatically enabled or disabled with the domain state transition.	RW	0
13	AUTO_GPT11	GPT 11 auto clock control 0x0: GP Timer 11 interface clock is unrelated to the domain state transition. 0x1: GP Timer 11 interface clock is automatically enabled or disabled with the domain state transition.	RW	0
12	AUTO_GPT10	GPT 10 auto clock control 0x0: GP Timer 10 interface clock is unrelated to the domain state transition. 0x1: GP Timer 10 interface clock is automatically enabled or disabled with the domain state transition.	RW	0

Table 5−91. CM_AUTOIDLE1_CORE (Continued)

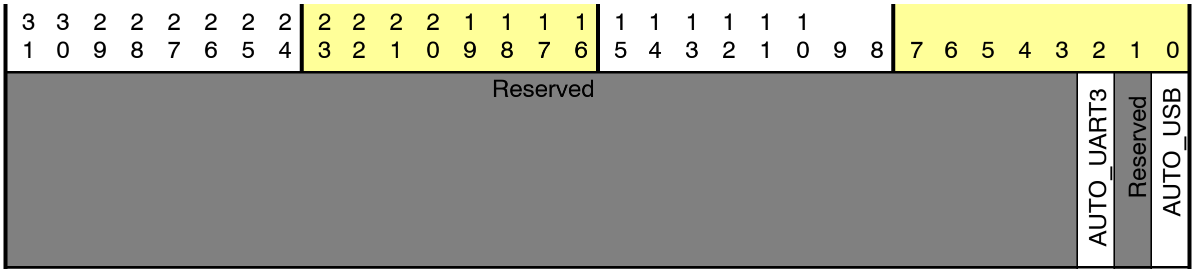
Bits	Field Name	Description	Type	Reset
11	AUTO_GPT9	GPT 9 auto clock control 0x0: GP Timer 9 interface clock is unrelated to the domain state transition. 0x1: GP Timer 9 interface clock is automatically enabled or disabled with the domain state transition.	RW	0
10	AUTO_GPT8	GPT 8 auto clock control 0x0: GP Timer 8 interface clock is unrelated to the domain state transition. 0x1: GP Timer 8 interface clock is automatically enabled or disabled with the domain state transition.	RW	0
9	AUTO_GPT7	GPT 7 auto clock control 0x0: GP Timer 7 interface clock is unrelated to the domain state transition. 0x1: GP Timer 7 interface clock is automatically enabled or disabled with the domain state transition.	RW	0
8	AUTO_GPT6	GPT 6 auto clock control 0x0: GP Timer 6 interface clock is unrelated to the domain state transition. 0x1: GP Timer 6 interface clock is automatically enabled or disabled with the domain state transition.	RW	0
7	AUTO_GPT5	GPT 5 auto clock control 0x0: GP Timer 5 interface clock is unrelated to the domain state transition. 0x1: GP Timer 5 interface clock is automatically enabled or disabled with the domain state transition.	RW	0
6	AUTO_GPT4	GPT 4 auto clock control 0x0: GP Timer 4 interface clock is unrelated to the domain state transition. 0x1: GP Timer 4 interface clock is automatically enabled or disabled with the domain state transition.	RW	0

Table 5−91. CM_AUTOIDLE1_CORE (Continued)

Bits	Field Name	Description	Type	Reset
5	AUTO_GPT3	GPT 3 auto clock control 0x0: GP Timer 3 interface clock is unrelated to the domain state transition. 0x1: GP Timer 3 interface clock is automatically enabled or disabled with the domain state transition.	RW	0
4	AUTO_GPT2	GPT 2 auto clock control 0x0: GP Timer 2 interface clock is unrelated to the domain state transition. 0x1: GP Timer 2 interface clock is automatically enabled or disabled with the domain state transition.	RW	0
2:1	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
0	AUTO_DSS	Display subsystem auto clock control 0x0: Display subsystem interface clock is unrelated to the domain state transition. 0x1: Display subsystem interface clock is automatically enabled or disabled with the domain state transition.	RW	0

Table 5−92. CM_AUTOIDLE2_CORE

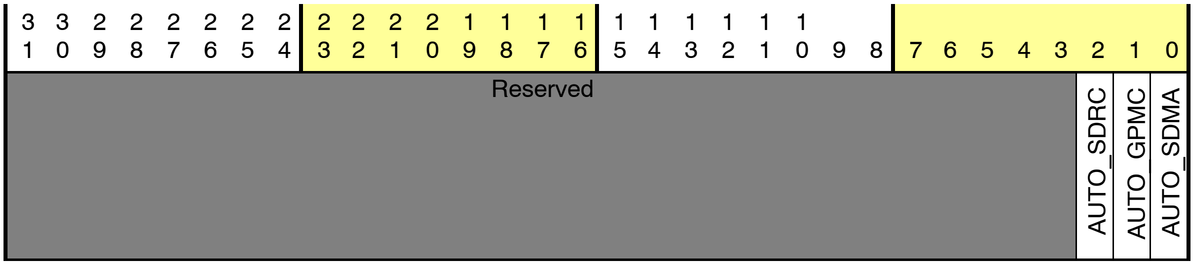
Address Offset	0x234		
Physical Address	0x4800 8234	**Instance**	PRCM
Description	This register allows automatic control of core module interface clock activity.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:3	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000000
2	AUTO_UART3	UART 3 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 with the domain state transition.	RW	0
1	Reserved	Reserved	RW	0
0	AUTO_USB	USB auto clock control 0x0: USB interface clock is unrelated to the domain state transition. 0x1: USB interface clock is automatically enabled or disabled with the domain state transition.	RW	0

Table 5−93. CM_AUTOIDLE3_CORE

Address Offset	0x238		
Physical Address	0x4800 8238	**Instance**	PRCM
Description	This register allows automatic control of core module interface clock activity.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:3	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000000
2	AUTO_SDRC	SDRC-SMS auto clock control 0x0: SDRC interface clock is unrelated to the domain state transition. 0x1: SDRC interface clock is automatically enabled or disabled with the domain state transition.	RW	1
1	AUTO_GPMC	GPMC auto clock control 0x0: GPMC interface clock is unrelated to the domain state transition. 0x1: GPMC interface clock is automatically enabled or disabled with the domain state transition.	RW	1
0	AUTO_SDMA	SDMA auto clock control 0x0: SDMA interface clock is unrelated to the domain state transition. 0x1: SDMA interface clock is automatically enabled or disabled with the domain state transition.	RW	1

Table 5−94. CM_CLKSEL1_CORE

Address Offset	0x240		
Physical Address	0x4800 8240	**Instance**	PRCM
Description	Core module clock selection		
Type	RW		

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="4">Reserved</td><td colspan="3">CLKSEL_USB</td><td colspan="5">Reserved</td><td colspan="5">Reserved</td><td>Reserved</td><td>Reserved</td><td colspan="5">Reserved</td><td>Reserved</td><td colspan="2">CLKSEL_L4</td><td colspan="5">CLKSEL_L3</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
31:28	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
27:25	CLKSEL_USB	Selects USB interface clock; other enums: Reserved 0x0: Reserved 0x1: USB interface clock is L3_clk/1 (boot mode only). 0x2: USB interface clock is L3_clk/2. 0x4: USB interface clock is L3_clk/4.	RW	0x1
24:20	Reserved	Reserved	RW	0x01
19:15	Reserved	Reserved	RW	0x00
14	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
13	Reserved	Reserved	RW	0
12:8	Reserved	Reserved	RW	0x01
7	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
6:5	CLKSEL_L4	Selects peripheral interconnect clock (L4_clk) 0x0: Reserved 0x1: L4 clock is L3_clk/1. 0x2: L4 clock is L3_clk/2. 0x3: Reserved	RW	0x1

1) Value not recommended by regular OMAP2420 clock configurations. PRCM configuration is not validated at the OMAP2420 level.

2) Value not recommended by regular OMAP2420 clock configurations. PRCM configuration is not validated at the OMAP2420 level.

Table 5−94. CM_CLKSEL1_CORE (Continued)

Bits	Field Name	Description	Type	Reset
4:0	CLKSEL_L3	Selects L3 interconnect clock (L3_clk); other enums: Reserved 0x0: Reserved 0x1: Core_clk/1 used by clock configuration VII and VIII 0x2: Core_clk/2[4] 0x4: Core_clk/4 used by clock configuration I and III 0x6: Core_clk/6 used by clock configuration II and IV 0x8: Core_clk/8[4] 0xC: Core_clk/12[4] 0x10: Core_clk/16[4]	RW	0x01

---

3) Value not recommended by regular OMAP2420 clock configurations. PRCM configuration is not validated at the OMAP2420 level.

4) Value not recommended by regular OMAP2420 clock configurations. PRCM configuration is not validated at the OMAP2420 level.

Table 5−95. CM_CLKSEL2_CORE

Address Offset	0x244		
Physical Address	0x4800 8244	**Instance**	PRCM
Description	Core module 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_GPT12	CLKSEL_GPT11	CLKSEL_GPT10	CLKSEL_GPT9	CLKSEL_GPT8	CLKSEL_GPT7	CLKSEL_GPT6	CLKSEL_GPT5	CLKSEL_GPT4	CLKSEL_GPT3	CLKSEL_GPT2	Reserved

Bits	Field Name	Description	Type	Reset
31:24	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
23:22	CLKSEL_GPT12	Selects GPT12 source clock 0x0: Source is Func_32k_clk. 0x1: Source is Sys_clk. 0x2: Source is ExtAltClk. 0x3: Reserved	RW	0x0
21:20	CLKSEL_GPT11	Selects GPT11 source clock 0x0: Source is Func_32k_clk. 0x1: Source is Sys_clk. 0x2: Source is ExtAltClk. 0x3: Reserved	RW	0x0
19:18	CLKSEL_GPT10	Selects GPT10 source clock 0x0: Source is Func_32k_clk. 0x1: Source is Sys_clk. 0x2: Source is ExtAltClk. 0x3: Reserved	RW	0x0
17:16	CLKSEL_GPT9	Selects GPT9 source clock 0x0: Source is Func_32k_clk. 0x1: Source is Sys_clk. 0x2: Source is ExtAltClk. 0x3: Reserved	RW	0x0
15:14	CLKSEL_GPT8	Selects GPT8 source clock 0x0: Source is Func_32k_clk. 0x1: Source is Sys_clk. 0x2: Source is ExtAltClk. 0x3: Reserved	RW	0x0
13:12	CLKSEL_GPT7	Selects GPT7 source clock 0x0: Source is Func_32k_clk. 0x1: Source is Sys_clk. 0x2: Source is ExtAltClk. 0x3: Reserved	RW	0x0

Table 5−95. CM_CLKSEL2_CORE (Continued)

Bits	Field Name	Description	Type	Reset
11:10	CLKSEL_GPT6	Selects GPT6 source clock 0x0: Source is Func_32k_clk. 0x1: Source is Sys_clk. 0x2: Source is ExtAltClk. 0x3: Reserved	RW	0x0
9:8	CLKSEL_GPT5	Selects GPT5 source clock 0x0: Source is Func_32k_clk. 0x1: Source is Sys_clk. 0x2: Source is ExtAltClk. 0x3: Reserved	RW	0x0
7:6	CLKSEL_GPT4	Selects GPT4 source clock 0x0: Source is Func_32k_clk. 0x1: Source is Sys_clk. 0x2: Source is ExtAltClk. 0x3: Reserved	RW	0x0
5:4	CLKSEL_GPT3	Selects GPT3 source clock 0x0: Source is Func_32k_clk. 0x1: Source is Sys_clk. 0x2: Source is ExtAltClk. 0x3: Reserved	RW	0x0
3:2	CLKSEL_GPT2	Selects GPT2 source clock 0x0: Source is Func_32k_clk. 0x1: Source is Sys_clk. 0x2: Source is ExtAltClk. 0x3: Reserved	RW	0x0
1:0	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0

Table 5−96. CM_CLKSTCTRL_CORE

Address Offset	0x248		
Physical Address	0x4800 8248	**Instance**	PRCM
Description	This register controls the hardware-supervised state transition of three core clock subdomains 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																													AUTOSTAT_DSS	AUTOSTAT_L4	AUTOSTAT_L3

Table 5−96.CM_CLKSTCTRL_CORE (Continued)

Bits	Field Name	Description	Type	Reset
31:3	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000000
2	AUTOSTAT_DSS	Display subsystem interface clock auto state control 0x0: Hardware supervised control of display subsystem clock is disabled. 0x1: Hardware supervised control of display subsystem clock is enabled.	RW	0
1	AUTOSTAT_L4	L4 interconnect clock auto state control 0x0: Hardware supervised control of L4 interconnect clock is disabled. 0x1: Hardware supervised control of L4 interconnect clock is enabled.	RW	0
0	AUTOSTAT_L3	L3 interconnect clock auto state control 0x0: Hardware supervised control of L3 interconnect clock is disabled. 0x1: Hardware supervised control of L3 interconnect clock is enabled.	RW	0

Table 5−97. PM_WKEN1_CORE

Address Offset	0x2A0		
Physical Address	0x4800 82A0	**Instance**	PRCM
Description	This register allows enabling/disabling of module wake-up events.		
Type	RW		

31:27	26	25:23	22	21	20:19	18	17	16:15	14	13	12	11	10	9	8	7	6	5	4	3	2:0
Reserved	EN_MMC	Reserved	EN_UART2	EN_UART1	Reserved	EN_MCSPI2	EN_MCSPI1	Reserved	EN_GPT12	EN_GPT11	EN_GPT10	EN_GPT9	EN_GPT8	EN_GPT7	EN_GPT6	EN_GPT5	EN_GPT4	EN_GPT3	EN_GPT2	Reserved	Reserved

Bits	Field Name	Description	Type	Reset
31:27	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
26	EN_MMC	MMC wake-up enable 0x0: MMC wake-up is disabled. 0x1: MMC wake-up event is enabled.	RW	1
25:23	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
22	EN_UART2	UART 2 wake-up enable 0x0: UART 2 wake-up is disabled. 0x1: UART 2 wake-up event is enabled.	RW	1
21	EN_UART1	UART 1 wake-up enable 0x0: UART 1 wake-up is disabled. 0x1: UART 1 wake-up event is enabled.	RW	1
20:19	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0

Table 5−97. PM_WKEN1_CORE (Continued)

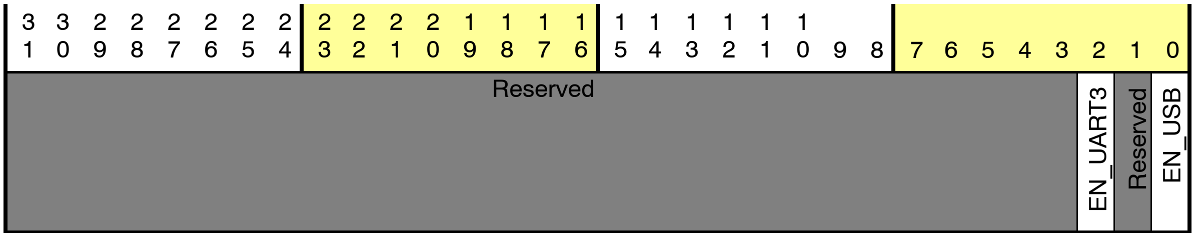
Bits	Field Name	Description	Type	Reset
18	EN_MCSPI2	McSPI 2 wake-up enable 0x0: SPI 2 wake-up is disabled. 0x1: SPI 2 wake-up event is enabled.	RW	1
17	EN_MCSPI1	McSPI 1 wake-up enable 0x0: SPI 1 wake-up is disabled. 0x1: SPI 1 wake-up event is enabled.	RW	1
16:15	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
14	EN_GPT12	GPT 12 wake-up enable 0x0: GPT 12 wake-up is disabled. 0x1: GPT 12 wake-up event is enabled.	RW	1
13	EN_GPT11	GPT 11 wake-up enable 0x0: GPT 11 wake-up is disabled. 0x1: GPT 11 wake-up event is enabled.	RW	1
12	EN_GPT10	GPT 10 wake-up enable 0x0: GPT 10 wake-up is disabled. 0x1: GPT 10 wake-up event is enabled.	RW	1
11	EN_GPT9	GPT 9 wake-up enable 0x0: GPT 9 wake-up is disabled. 0x1: GPT 9 wake-up event is enabled.	RW	1
10	EN_GPT8	GPT 8 wake-up enable 0x0: GPT 8 wake-up is disabled. 0x1: GPT 8 wake-up event is enabled.	RW	1
9	EN_GPT7	GPT 7 wake-up enable 0x0: GPT 7 wake-up is disabled. 0x1: GPT 7 wake-up event is enabled.	RW	1
8	EN_GPT6	GPT 6 wake-up enable 0x0: GPT 6 wake-up is disabled. 0x1: GPT 6 wake-up event is enabled.	RW	1
7	EN_GPT5	GPT 5 wake-up enable 0x0: GPT 5 wake-up is disabled. 0x1: GPT 5 wake-up event is enabled.	RW	1
6	EN_GPT4	GPT 4 wake-up enable 0x0: GPT 4 wake-up is disabled. 0x1: GPT 4 wake-up event is enabled.	RW	1
5	EN_GPT3	GPT 3 wake-up enable 0x0: GPT 3 wake-up is disabled. 0x1: GPT 3 wake-up event is enabled.	RW	1
4	EN_GPT2	GPT 2 wake-up enable 0x0: GPT 2 wake-up is disabled. 0x1: GPT 2 wake-up event is enabled.	RW	1

Table 5−97. PM_WKEN1_CORE (Continued)

Bits	Field Name	Description	Type	Reset
3	Reserved	Reserved	RW	1
2:0	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0

Table 5−98. PM_WKEN2_CORE

Address Offset	0x2A4		
Physical Address	0x4800 82A4	**Instance**	PRCM
Description	This register allows enabling/disabling of module wake-up events.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:3	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000000
2	EN_UART3	UART 3 wake-up enable 0x0: UART 3 wake-up is disabled. 0x1: UART 3 wake-up event is enabled.	RW	1
1	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
0	EN_USB	USB wake-up enable 0x0: USB wake-up is disabled. 0x1: USB wake-up event is enabled.	RW	1

Table 5−99. PM_WKST1_CORE

Address Offset	0x2B0		
Physical Address	0x4800 82B0	**Instance**	PRCM
Description	This register logs module wake-up events. 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					ST_MMC	Reserved			ST_UART2	ST_UART1	Reserved		ST_MCSPI2	ST_MCSPI1	Reserved		ST_GPT12	ST_GPT11	ST_GPT10	ST_GPT9	ST_GPT8	ST_GPT7	ST_GPT6	ST_GPT5	ST_GPT4	ST_GPT3	ST_GPT2	Reserved	Reserved		

Bits	Field Name	Description		Type	Reset
31:27	Reserved	Write 0s for future compatibility. Read returns 0.		RW	0x00
26	ST_MMC	MMC wake-up status		RW	0
		Read 0x0:	MMC wake-up did not occur or was masked.		
		Write 0x0:	Status bit is unchanged.		
		Read 0x1:	MMC wake-up occurred.		
		Write 0x1:	Status bit is reset.		
25:23	Reserved	Write 0s for future compatibility. Read returns 0.		RW	0x0
22	ST_UART2	UART 2 wake-up status		RW	0
		Read 0x0:	UART 2 wake-up did not occur or was masked.		
		Write 0x0:	Status bit is unchanged.		
		Read 0x1:	UART 2 wake-up occurred.		
		Write 0x1:	Status bit is reset.		
21	ST_UART1	UART 1 wake-up status		RW	0
		Read 0x0:	UART 1 wake-up did not occur or was masked.		
		Write 0x0:	Status bit is unchanged.		
		Read 0x1:	UART 1 wake-up occurred.		
		Write 0x1:	Status bit is reset.		
20:19	Reserved	Write 0s for future compatibility. Read returns 0.		RW	0x0
18	ST_MCSPI2	McSPI 2 wake-up status		RW	0
		Read 0x0:	SPI 2 wake-up did not occur or was masked.		
		Write 0x0:	Status bit is unchanged.		
		Read 0x1:	SPI 2 wake-up occurred.		
		Write 0x1:	Status bit is reset.		

Table 5−99. PM_WKST1_CORE (Continued)

Bits	Field Name	Description		Type	Reset
17	ST_MCSPI1	McSPI 1 wake-up status		RW	0
		Read 0x0:	SPI 1 wake-up did not occur or was masked.		
		Write 0x0:	Status bit is unchanged.		
		Read 0x1:	SPI 1 wake-up occurred.		
		Write 0x1:	Status bit is reset.		
16:15	Reserved	Write 0s for future compatibility. Read returns 0.		RW	0x0
14	ST_GPT12	GPT 12 wake-up status		RW	0
		Read 0x0:	GPT 12 wake-up did not occur or was masked.		
		Write 0x0:	Status bit is unchanged.		
		Read 0x1:	GPT 12 wake-up occurred.		
		Write 0x1:	Status bit is reset.		
13	ST_GPT11	GPT 11 wake-up status		RW	0
		Read 0x0:	GPT 11 wake-up did not occur or was masked.		
		Write 0x0:	Status bit is unchanged.		
		Read 0x1:	GPT 11 wake-up occurred.		
		Write 0x1:	Status bit is reset.		
12	ST_GPT10	GPT 10 wake-up status		RW	0
		Read 0x0:	GPT10 wake-up did not occur or was masked.		
		Write 0x0:	Status bit is unchanged.		
		Read 0x1:	GPT 10 wake-up occurred.		
		Write 0x1:	Status bit is reset.		
11	ST_GPT9	GPT 9 wake-up status		RW	0
		Read 0x0:	GPT 9 wake-up did not occur or was masked.		
		Write 0x0:	Status bit is unchanged.		
		Read 0x1:	GPT 9 wake-up occurred.		
		Write 0x1:	Status bit is reset.		
10	ST_GPT8	GPT 8 wake-up status		RW	0
		Read 0x0:	GPT 8 wake-up did not occur or was masked.		
		Write 0x0:	Status bit is unchanged.		
		Read 0x1:	GPT 8 wake-up occurred.		
		Write 0x1:	Status bit is reset.		

Table 5−99. PM_WKST1_CORE (Continued)

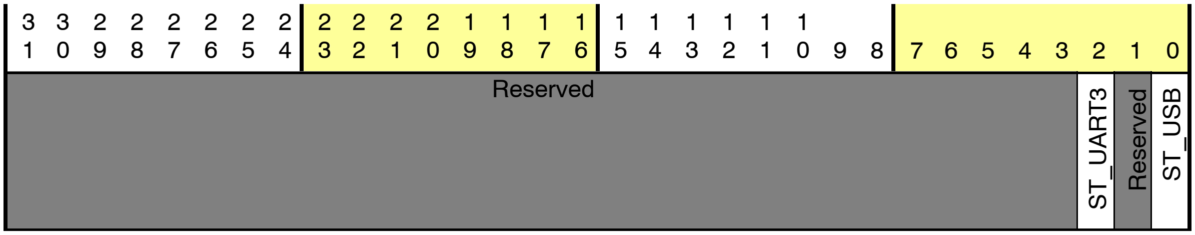
Bits	Field Name	Description		Type	Reset
9	ST_GPT7	GPT 7 wake-up status		RW	0
		Read 0x0:	GPT 7 wake-up did not occur or was masked.		
		Write 0x0:	Status bit is unchanged.		
		Read 0x1:	GPT 7 wake-up occurred.		
		Write 0x1:	Status bit is reset.		
8	ST_GPT6	GPT 6 wake-up status		RW	0
		Read 0x0:	GPT 6 wake-up did not occur or was masked.		
		Write 0x0:	Status bit is unchanged.		
		Read 0x1:	GPT 6 wake-up occurred.		
		Write 0x1:	Status bit is reset.		
7	ST_GPT5	GPT 5 wake-up status		RW	0
		Read 0x0:	GPT 5 wake-up did not occur or was masked.		
		Write 0x0:	Status bit is unchanged.		
		Read 0x1:	GPT 5 wake-up occurred.		
		Write 0x1:	Status bit is reset.		
6	ST_GPT4	GPT 4 wake-up status		RW	0
		Read 0x0:	GPT 4 wake-up did not occur or was masked.		
		Write 0x0:	Status bit is unchanged.		
		Read 0x1:	GPT 4 wake-up occurred.		
		Write 0x1:	Status bit is reset.		
5	ST_GPT3	GPT 3 wake-up status		RW	0
		Read 0x0:	GPT 3 wake-up did not occur or was masked.		
		Write 0x0:	Status bit is unchanged.		
		Read 0x1:	GPT 3 wake-up occurred.		
		Write 0x1:	Status bit is reset.		
4	ST_GPT2	GPT 2 wake-up status		RW	0
		Read 0x0:	GPT 2 wake-up did not occur or was masked.		
		Write 0x0:	Status bit is unchanged.		
		Read 0x1:	GPT 2 wake-up occurred.		
		Write 0x1:	Status bit is reset.		

Table 5−99. PM_WKST1_CORE (Continued)

Bits	Field Name	Description	Type	Reset
3	Reserved	Reserved	RW	0
2:0	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0

Table 5−100. PM_WKST2_CORE

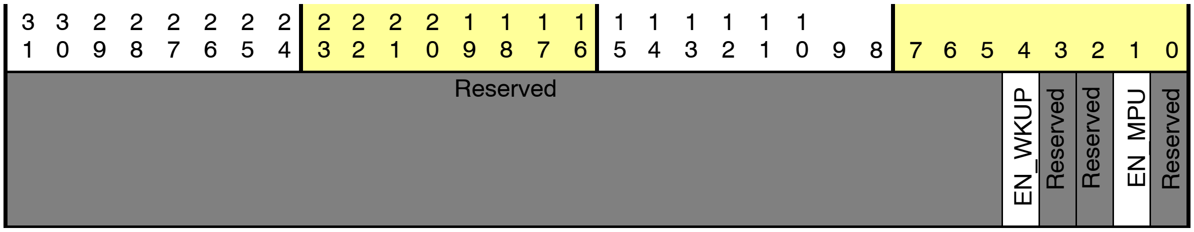
Address Offset	0x2B4		
Physical Address	0x4800 82B4	**Instance**	PRCM
Description	This register logs module wake-up events. Must be cleared by software.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:3	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000000
2	ST_UART3	UART 3 wake-up status Read 0x0: UART 3 wake-up did not occur or was masked. Write 0x0: Status bit is unchanged. Read 0x1: UART 3 wake-up occurred Write 0x1: Status bit is reset.	RW	0
1	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
0	ST_USB	USB wake-up status Read 0x0: USB wake-up did not occur or was masked. Write 0x0: Status bit is unchanged. Read 0x1: USB wake-up occurred Write 0x1: Status bit is reset.	RW	0

Table 5−101. PM_WKDEP_CORE

Address Offset	0x2C8		
Physical Address	0x4800 82C8	**Instance**	PRCM
Description	This register establishes core domain as the result of another domain wake-up. Read only.		
Type	R		



Bits	Field Name	Description	Type	Reset
31:5	Reserved	Read returns 0.	R	0x0000000
4	EN_WKUP	Wake-up domain dependency 0x1: Core domain is waked up on wake-up domain module wake-up event.	R	1
3	Reserved		R	1
2	Reserved		R	1
1	EN_MPU	MPU domain dependency 0x1: Core domain is waked up on MPU domain wake-up.	R	1
0	Reserved	Read returns 0.	RW	0

Table 5−102. PM_PWSTCTRL_CORE

Address Offset	0x2E0		
Physical Address	0x4800 82E0	**Instance**	PRCM
Description	This register controls the core domain power state transition.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											MEMORYCHANGE	Reserved				MEM3ONSTATE		MEM2ONSTATE		MEM1ONSTATE		Reserved				MEM3RETSTATE	MEM2RETSTATE	MEM1RETSTATE	LOGICRETSTATE	POWERSTATE	

Bits	Field Name	Description	Type	Reset
31:21	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x000
20	MEMORY-CHANGE	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.	RW	0
19:16	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
15:14	MEM3ONSTATE	Memory block 3 state when ON 0x0: Memory block 3 is ON when domain is ON. 0x1: Memory block 3 is retained when domain is ON. 0x2: Reserved 0x3: Memory block 3 is OFF when domain is ON.	RW	0x0

Table 5−102. PM_PWSTCTRL_CORE (Continued)

Bits	Field Name	Description		Type	Reset
13:12	MEM2ONSTATE	Memory block 2 state when ON		RW	0x0
		0x0:	Memory block 2 is ON when domain is ON.		
		0x1:	Memory block 2 is retained when domain is ON.		
		0x2:	Reserved		
		0x3:	Memory block 2 is OFF when domain is ON.		
11:10	MEM1ONSTATE	Memory block 1 state when ON		RW	0x0
		0x0:	Memory block 1 is ON when domain is ON.		
		0x1:	Memory block 1 is retained when domain is ON.		
		0x2:	Reserved		
		0x3:	Memory block 1 is OFF when domain is ON.		
9:6	Reserved	Write 0s for future compatibility. Read returns 0.		RW	0x0
5	MEM3RETSTATE	Memory block 3 when RETENTION		RW	0
		0x0:	Memory block 3 is OFF when domain is in RETENTION state.		
		0x1:	Memory block 3 is retained when domain is in RETENTION state.		
4	MEM2RETSTATE	Memory block 2 state when RETENTION		RW	0
		0x0:	Memory block 2 is OFF when domain is in RETENTION state.		
		0x1:	Memory block 2 is retained when domain is in RETENTION state.		
3	MEM1RETSTATE	Memory block 1 state when RETENTION		RW	0
		0x0:	Memory block 1 is OFF when domain is in RETENTION state.		
		0x1:	Memory block 1 is retained when domain is in RETENTION state.		
2	LOGICRETSTATE	Logic state when RETENTION		R	1
		0x1:	Logic is always retained when domain is in RETENTION state.		
1:0	POWERSTATE	Power state control		RW	0x0
		0x0:	ON State		
		0x1:	RETENTION state		
		0x2:	Reserved		
		0x3:	OFF state		

Table 5−103. PM_PWSTST_CORE

Address Offset	0x2E4		
Physical Address	0x4800 82E4	**Instance**	PRCM
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	CLKACTIVITY	Reserved			MEM3STATEST		MEM2STATEST		MEM1STATEST		Reserved				LASTSTATEENTERED		Reserved			

Bits	Field Name	Description	Type	Reset
31:21	Reserved	Read returns 0.	R	0x000
20	INTRANSITION	Domain transition status 0x0: No transition 0x1: Core power domain transition is in progress.	R	0
19	CLKACTIVITY	Clock activity status 0x0: No domain clock activity 0x1: Domain clock is active	R	0
18:16	Reserved	Write 0s for future compatibility. Read returns 0.	R	0x0
15:14	MEM3STATEST	Memory block 3 state status 0x0: Memory is ON. 0x1: Memory is in RETENTION state. 0x2: Reserved 0x3: Memory is OFF.	R	0x0
13:12	MEM2STATEST	Memory block 2 state status 0x0: Memory is ON. 0x1: Memory is in RETENTION state. 0x2: Reserved 0x3: Memory is OFF.	R	0x0

Table 5−103. PM_PWSTST_CORE (Continued)

Bits	Field Name	Description	Type	Reset
11:10	MEM1STATEST	Memory block 1 state status 0x0: Memory is ON. 0x1: Memory is in RETENTION state. 0x2: Reserved 0x3: Memory is OFF.	R	0x0
9:6	Reserved	Read returns 0.	R	0x0
5:4	LASTSTATE ENTERED	Last power state entered 0x0: Core domain was previously ON. 0x1: Core domain was previously in RETENTION state. 0x2: Reserved 0x3: Core domain was previously OFF.	R	0x3
3:0	Reserved	Read returns 0.	R	0x0

Table 5−104. CM_FCLKEN_WKUP

Address Offset	0x400		
Physical Address	0x4800 8400	**Instance**	PRCM
Description	Controls 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_MPU_WDT	EN_GPIOS	Reserved	EN_GPT1

Bits	Field Name	Description	Type	Reset
31:4	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000000
3	EN_MPU_WDT	MPU watchdog timer clock control 0x0: MPU watchdog timer functional clock is disabled. 0x1: MPU watchdog timer functional clock is enabled.	RW	1
2	EN_GPIOS	GPIO (1 to 4) clock control 0x0: GPIO (1 to 4) functional clock is disabled. 0x1: GPIO (1 to 4) functional clock is enabled.	RW	1
1	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
0	EN_GPT1	General-purpose timer 1 clock control 0x0: GPTimer1 functional clock is disabled. 0x1: GPTimer1 functional clock is enabled.	RW	0

Table 5−105. CM_ICLKEN_WKUP

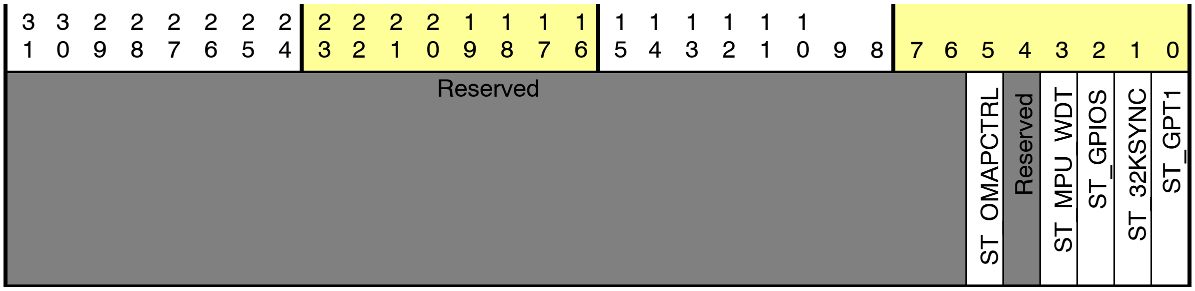
Address Offset	0x410		
Physical Address	0x4800 8410	**Instance**	PRCM
Description	Controls module interface clock activity		
Type	RW		

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_OMAPCTRL	Reserved	EN_MPU_WDT	EN_GPIOS	EN_32KSYNC	EN_GPT1

Bits	Field Name	Description	Type	Reset
31:6	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000000
5	EN_OMAPCTRL	OMAP control module interface clock control 0x0: OMAP control interface clock is disabled. 0x1: OMAP control interface clock is enabled.	RW	1
4	Reserved	Reserved		
3	EN_MPU_WDT	MPU WDT interface clock control 0x0: MPU watchdog timer interface clock is disabled. 0x1: MPU watchdog timer interface clock is enabled.	RW	1
2	EN_GPIOS	GPIO interface clock control 0x0: GPIO (1 to 4) interface clock is disabled. 0x1: GPIO (1 to 4) interface clock is enabled.	RW	1
1	EN_32KSYNC	32 kHz synchronization timer interface clock control 0x0: 32k synchronization timer interface clock is disabled. 0x1: 32k synchronization timer interface clock is enabled.	RW	1
0	EN_GPT1	GPT 1 interface clock control 0x0: GPTimer1 interface clock is disabled. 0x1: GPTimer1 interface clock is enabled.	RW	0

Table 5−106. CM_IDLEST_WKUP

Address Offset	0x420		
Physical Address	0x4800 8420	**Instance**	PRCM
Description	Wake-up domain modules access monitoring. This register is read only and automatically updated.		
Type	R		



Bits	Field Name	Description	Type	Reset
31:6	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000000
5	ST_OMAPCTRL	OMAP control module idle status 0x0: OMAP control module cannot be accessed. An access attempt can return an error. 0x1: OMAP control module can be accessed.	R	0
4	Reserved	Reserved		
3	ST_MPU_WDT	MPU WDT idle status 0x0: MPU WDT cannot be accessed. An access attempt can return an error. 0x1: MPU WDT can be accessed.	R	0
2	ST_GPIOS	GPIOs idle status 0x0: GPIO (1 to 4) cannot be accessed. An access attempt can return an error. 0x1: GPIO (1 to 4) can be accessed.	R	0
1	ST_32KSYNC	32-kHz synchronization timer idle status 0x0: 32-kHz synchronization timer cannot be accessed. An access attempt can return an error. 0x1: 32-kHz synchronization timer can be accessed.	R	0
0	ST_GPT1	GPT 1 idle status 0x0: GPTimer1 cannot be accessed. An access attempt can return an error. 0x1: GPTimer1 can be accessed.	R	0

Table 5−107. CM_AUTOIDLE_WKUP

Address Offset	0x430		
Physical Address	0x4800 8430	**Instance**	PRCM
Description	This register controls the automatic control of the wake-up module 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_OMAPCTRL	Reserved	AUTO_MPU_WDT	AUTO_GPIOS	AUTO_32KSYNC	AUTO_GPT1

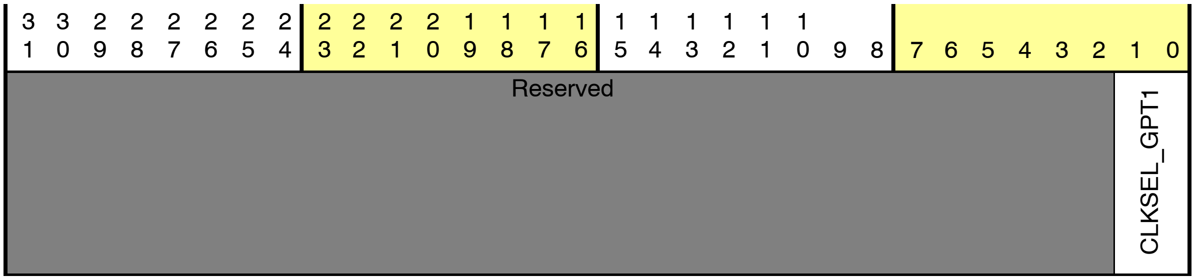
Bits	Field Name	Description	Type	Reset
31:6	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000000
5	AUTO_OMAPCTRL	OMAP control module autoidle control 0x0: OMAP control module interface clock is unrelated to core domain activity. 0x1: OMAP control module interface clock is automatically enabled/disabled according to core domain activity.	RW	0
4	Reserved	Reserved		
3	AUTO_MPU_WDT	MPU Watchdog timer autoidle control 0x0: MPU watchdog timer interface clock is unrelated to core domain activity. 0x1: MPU watchdog timer interface clock is automatically enabled/disabled according to core domain activity.	RW	0
2	AUTO_GPIOS	GPIO autoidle control 0x0: GPIO (1 to 4) interface clock is unrelated to CORE domain activity. 0x1: GPIOs (1 to 4) interface clock is automatically enabled/disabled according to core domain activity.	RW	0

Table 5−107. CM_AUTOIDLE_WKUP (Continued)

Bits	Field Name	Description	Type	Reset
1	AUTO_32KSYNC	32-kHz synchronization timer autoidle control 0x0: 32-kHz synchronization timer interface clock is unrelated to core domain activity. 0x1: 32-kHz synchronization timer interface clock is automatically enabled or disabled according to core domain activity.	RW	0
0	AUTO_GPT1	GPTimer 1 autoidle control 0x0: GP timer 1 interface clock is unrelated to core domain activity. 0x1: GP timer 1 interface clock is automatically enabled or disabled according to core domain activity.	RW	0

Table 5−108. CM_CLKSEL_WKUP

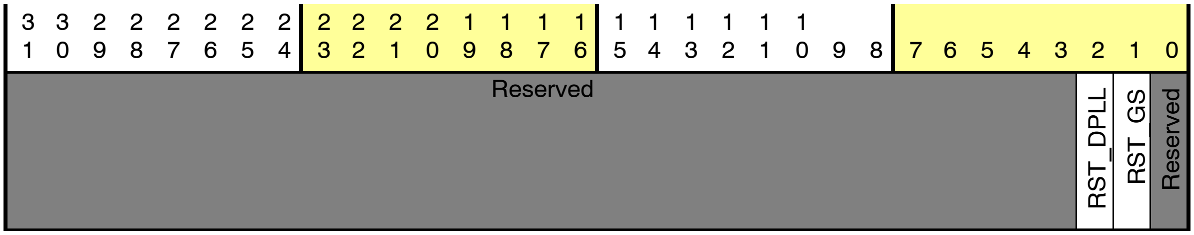
Address Offset	0x440		
Physical Address	0x4800 8440	**Instance**	PRCM
Description	Wake-up domain module source clock selection		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:2	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000000
1:0	CLKSEL_GPT1	Selects GPT1 source clock 0x0: Source is Func_32k_clk. 0x1: Source is Sys_clk. 0x2: Source is ExtAltClk. 0x3: Reserved	RW	0x0

Table 5−109. RM_RSTCTRL_WKUP

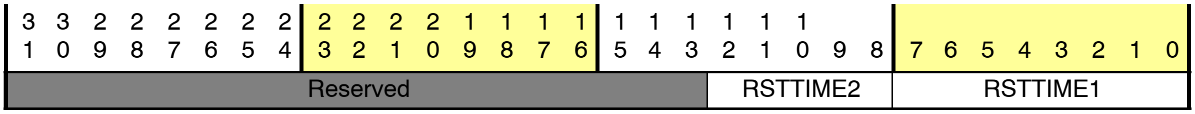
Address Offset	0x450		
Physical Address	0x4800 8450	**Instance**	PRCM
Description	Global software and DPLL reset control. This register is auto-cleared. Only writing 1 is possible. A read returns 0.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:3	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000000
2	RST_DPLL	DPLL reset control 0x0: DPLL reset is cleared. 0x1: Resets the DPLL and asserts a global software reset. The software must ensure that SDRAM is correctly put in self-refresh mode before applying this reset.	RW	0
1	RST_GS	Global software reset control 0x0: Global software reset is cleared. 0x1: Asserts a global software reset.	RW	0
0	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0

Table 5−110. RM_RSTTIME_WKUP

Address Offset	0x0454		
Physical Address	0x4800 8454	**Instance**	PRCM1
Description	Reset duration control		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:13	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000
12:8	RSTTIME2	(Power domain) reset duration 2. Reset is asserted during (1 + RSTTIME2) L4 clock cycles.	RW	0x10
7:0	RSTTIME1	(Global) reset duration 1. Reset is asserted during (1 + RSTTIME1) 32-kHz clock cycles.	RW	0x02

Table 5−111. RM_RSTST_WKUP

Address Offset	0x458		
Physical Address	0x4800 8458	**Instance**	PRCM
Description	This register logs the global reset sources. Each bit is set on release of the respective reset source 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	EXTWMPU_RST	Reserved	MPU_WD_RST	Reserved	Reserved	GLOBALWMPU_RST	GLOBALCOLD_RST

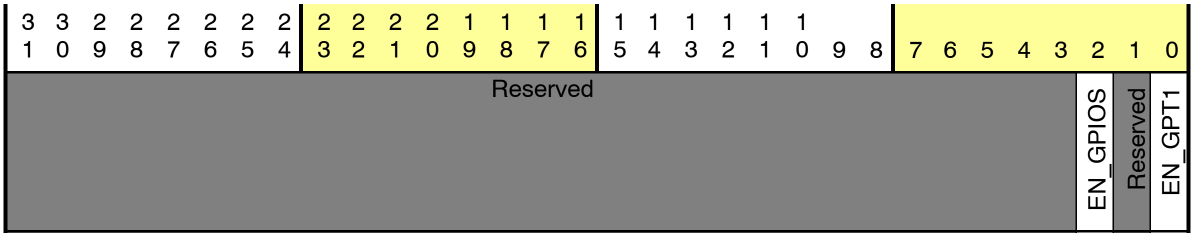
Bits	Field Name	Description		Type	Reset
31:7	Reserved	Write 0s for future compatibility. Read returns 0.		R	0x0000000
6	EXTWMPU_RST	External warm reset event		RW	0
		Read 0x0:	No global warm reset.		
		Write 0x0:	Status bit is unchanged.		
		Read 0x1:	Global external warm reset occurred.		
		Write 0x1:	Status bit is reset.		
5	Reserved	Reserved			
4	MPU_WD_RST	MPU watchdog reset event		RW	0
		Read 0x0:	No MPU watchdog reset.		
		Write 0x0:	Status bit is unchanged.		
		Read 0x1:	MPU watchdog reset occurred.		
		Write 0x1:	Status bit is reset.		
3	Reserved	Reserved			
2	Reserved	Write 0s for future compatibility. Read returns 0.		RW	0

Table 5−111. RM_RSTST_WKUP (Continued)

Bits	Field Name	Description	Type	Reset
1	GLOBAL WMPU_RST	Global software reset event Read 0x0: No global software reset. Write 0x0: Status bit is unchanged. Read 0x1: Global software reset occurred. Write 0x1: Status bit is reset.	RW	0
0	GLOBAL-COLD_RST	Power-up (cold) reset event Read 0x0: No power-on reset. Write 0x0: Status bit is unchanged. Read 0x1: Power-on reset occurred. Write 0x1: Status bit is reset.	RW	1

Table 5−112. PM_WKEN_WKUP

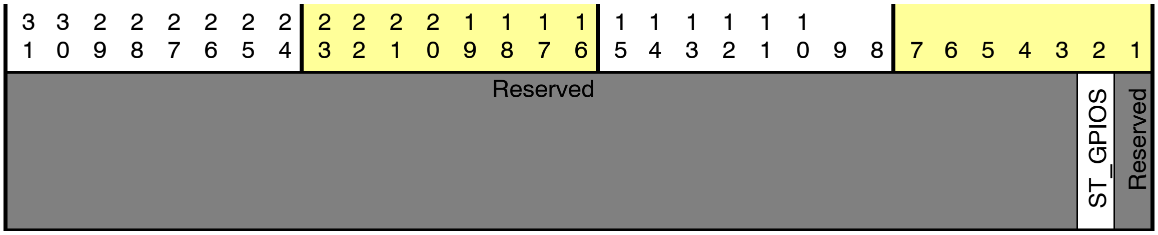
Address Offset	0x4A0		
Physical Address	0x4800 84A0	**Instance**	PRCM
Description	This register allows enabling/disabling of module wake-up events.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:3	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000000
2	EN_GPIOS	0: GPIO wake-up is disabled. 1: GPIO wake-up path is enabled.	RW	1
1	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
0	EN_GPT1	0: GP timer 1 wake-up is disabled. 1: GP timer 1 wake-up path is enabled.	RW	1

Table 5−113. PM_WKST_WKUP

Address Offset	0x4B0		
Physical Address	0x4800 84B0	**Instance**	PRCM
Description	This register logs module wake-up events. Must be cleared by software.		
Type	RW		



Bits	Field Name	Description		Type	Reset
31:3	Reserved	Write 0s for future compatibility. Read returns 0.		RW	0x00000000
2	ST_GPIOS	GPIO wake-up status		RW	0
		Read 0x0:	GPIO wake-up did not occur or was masked.		
		Write 0x0:	Status bit is unchanged.		
		Read 0x1:	GPIO wake-up occurred.		
		Write 0x1:	Status bit is reset.		
1	Reserved	Write 0s for future compatibility. Read returns 0.		RW	0
0	ST_GPT1	GPT1 wake-up status		RW	0
		Read 0x0:	GP timer 1 wake-up did not occur or was masked.		
		Write 0x0:	Status bit is unchanged.		
		Read 0x1:	GP timer 1 wake-up occurred.		
		Write 0x1:	Status bit is reset.		

Table 5−114. CM_CLKEN_PLL

Address Offset	0x500		
Physical Address	0x4800 8500	**Instance**	PRCM
Description	This register allows control of the PLL 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_54M_PLL		Reserved		EN_96M_PLL		EN_DPLL	

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x000000
7:6	EN_54M_PLL	APLL 54 MHz control 0x0: APLL is stopped. 0x1: Reserved 0x2: Reserved 0x3: APLL is enabled in lock mode.	RW	0x0
5:4	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
3:2	EN_96M_PLL	APLL 96-MHz control 0x0: APLL is stopped. 0x1: Reserved 0x2: Reserved 0x3: APLL is enabled in lock mode.	RW	0x3
1:0	EN_DPLL	DPLL control 0x0: Reserved 0x1: DPLL is in low-power bypass. 0x2: DPLL is in fast-relock bypass. 0x3: DPLL is enabled in lock mode.	RW	0x1

Table 5−115. CM_IDLEST_CKGEN

Address Offset	0x520		
Physical Address	0x4800 8520	**Instance**	PRCM
Description	This register allows monitoring the master clock activity. This register is read-only and is 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_54M_APLL	ST_96M_APLL	Reserved	ST_54M_CLK	ST_12M_CLK	ST_48M_CLK	Reserved	ST_96M_CLK	ST_CORE_CLK	

Bits	Field Name	Description		Type	Reset
31:10	Reserved	Read returns 0.		R	0x000000
9	ST_54M_APLL	APLL 54 MHz activity		R	0
		0x0:	APLL is in power-down mode.		
		0x1:	APLL is locked.		
8	ST_96M_APLL	APLL 96-MHz activity		R	0
		0x0:	APLL is in power-down mode.		
		0x1:	APLL is locked.		
7	Reserved	Read returns 0.		R	0
6	ST_54M_CLK	Functional clock 54 MHz activity		R	0
		0x0:	Func_54M_clk is not active.		
		0x1:	Func_54M_clk is active.		
5	ST_12M_CLK	Functional clock 12 MHz activity		R	0
		0x0:	Func_12M_clk is not active.		
		0x1:	Func_12M_clk is active.		
4	ST_48M_CLK	Functional clock 48 MHz activity		R	0
		0x0:	FUNC_48M_CLK is not active.		
		0x1:	FUNC_48M_CLK is active.		
3	Reserved	Read returns 0.		R	0
2	ST_96M_CLK	Functional clock 96-MHz activity		R	0
		0x0:	FUNC_96M_CLK is not active.		
		0x1:	FUNC_96M_CLK is active.		
1:0	ST_CORE_CLK	Core clock activity		R	0x1
		0x0:	Reserved		
		0x1:	Core_clk is ref_clk.		
		0x2:	Core_clk is DPLL output frequency (lock).		
		0x3:	Core_clk is 32 kHz.		

Table 5−116. CM_AUTOIDLE_PLL

Address Offset	0x530		
Physical Address	0x4800 8530	**Instance**	PRCM
Description	This register provides automatic control of the PLL 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_54M		Reserved		AUTO_96M		AUTO_DPLL	

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x000000
7:6	AUTO_54M	54 MHz APLL automatic control 0x0: Auto control is disabled. 0x1: Reserved 0x2: Reserved 0x3: APLL is automatically stopped when the Func_54M_clk is not required. It is also restarted automatically.	RW	0x0
5:4	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
3:2	AUTO_96M	96-MHz APLL automatic control 0x0: Auto control is disabled. 0x1: Reserved 0x2: Reserved 0x3: APLL is automatically stopped when related clocks are not required. It is also restarted automatically.	RW	0x3
1:0	AUTO_DPLL	DPLL automatic control 0x0: Auto control disabled 0x1: DPLL is automatically in bypass (low power) when the Core_clk is not required. It is also restarted automatically. 0x2: DPLL is automatically in bypass (fast relock) when the Core_clk is not required. It is also restarted automatically. 0x3: DPLL is automatically stopped (enabled) when all required conditions are met.	RW	0x0

Table 5−117. CM_CLKSEL1_PLL

Address Offset	0x540		
Physical Address	0x4800 8540	**Instance**	PRCM
Description	This register controls the selection of the master clock frequencies. This register is reset on power-up only.		
Type	RW		

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="6">Reserved</td><td colspan="3">APLLS_CLKIN</td><td>Reserved</td><td colspan="10">DPLL_MULT</td><td colspan="4">DPLL_DIV</td><td colspan="2">Reserved</td><td>54M_SOURCE</td><td>Reserved</td><td>48M_SOURCE</td><td colspan="3">Reserved</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
31:26	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
25:23	APLLS_CLKIN	APLLs clock input selection; other enums: Reserved 0x0: Clock input is 19.2 MHz. 0x2: Clock input is 13 MHz. 0x3: Clock input is 12 MHz.	RW	0x0
22	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
21:12	DPLL_MULT	DPLL multiplier factor M (0 to 1023)	RW	0x000
11:8	DPLL_DIV	DPLL divider factor N (0 to 15)	RW	0x0
7:6	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
5	54M_SOURCE	Selection of Func_54M_clk source 0x0: Source is the 54 MHz APLL. 0x1: Source is ExtAlt_clk.	RW	0
4	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
3	48M_SOURCE	Selection of Func_12M_clk and FUNC_48M_CLK source 0x0: Source is the 96-MHz APLL. 0x1: Source is ExtAlt_clk.	RW	0
2:0	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0

Table 5−118. CM_CLKSEL2_PLL

Address Offset	0x544		
Physical Address	0x4800 8544	**Instance**	PRCM1
Description	This register controls the selection of the master clock frequencies. This bit is reset on power-up only.		
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																														CORE_CLK_SRC	

Bits	Field Name	Description	Type	Reset
31:2	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000000
1:0	CORE_CLK_SRC[1]	Selection of Core_clk source 0x0: Source is 32-kHz clock. 0x1: Source is DPLL clockout. 0x2: Source is DPLL clockout X 2. 0x3: Reserved	RW	0x2

1) If warm reset is asserted, CM_CLKSEL2_PLL is not reset and the CORE_CLK_SRC field is kept at its previous value. The software must set this field to the default value: 0x2 to be in the same configuration as after release of a cold reset.

Table 5−119. CM_FCLKEN_DSP

Address Offset	0x800		
Physical Address	0x4800 8800	**Instance**	PRCM
Description	This registers controls the DSP domain 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																					Reserved	Reserved	Reserved	Reserved							EN_DSP

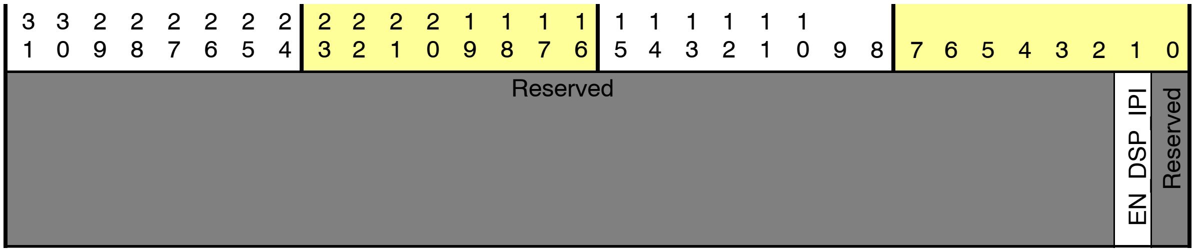
Bits	Field Name	Description	Type	Reset
31:11	Reserved	Write 0s for future compatibility. Reads return 0	RW	0x000000
10	Reserved		RW	0
9	Reserved	Write 0s for future compatibility. Reads return 0	RW	0
8	Reserved		RW	0

Table 5−119. CM_FCLKEN_DSP (Continued)

Bits	Field Name	Description	Type	Reset
7:1	Reserved	Write 0s for future compatibility Reads return 0	RW	0x00
0	EN_DSP	DSP clock control 0x0: DSP-SS functional clock is disabled. 0x1: DSP-SS functional clock is enabled.	RW	0

Table 5−120. CM_ICLKEN_DSP

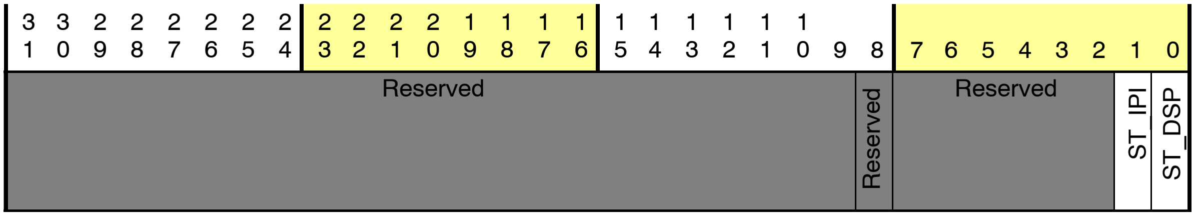
Address Offset	0x810		
Physical Address	0x4800 8810	**Instance**	PRCM
Description	This registers controls the DSP domain interface clock activity.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:2	Reserved	Write 0s for future compatibility. Reads return 0.	RW	0x00000000
1	EN_DSP_IPI	DSP intrusive port clock activity 0x0: DSP IPI clock is disabled. 0x1: DSP IPI clock is enabled.	RW	0
0	Reserved	Write 0s for future compatibility. Reads return 0	RW	0

Table 5−121. CM_IDLEST_DSP

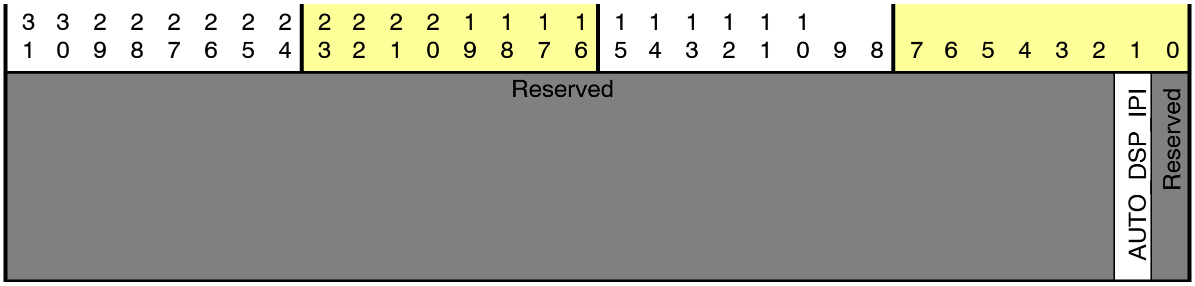
Address Offset	0x820		
Physical Address	0x4800 8820	**Instance**	PRCM
Description	DSP standby status and access availability monitoring. This register is read only and automatically updated.		
Type	R		



Bits	Field Name	Description	Type	Reset
31:9	Reserved	Reads return 0	R	0x000000
8	Reserved		R	0
7:2	Reserved	Read returns 0.	R	0x00
1	ST_IPI	DSP intrusive port idle status 0x0: DSP intrusive port cannot be accessed. Any access may return an error. 0x1: DSP Intrusive port can be accessed	R	0
0	ST_DSP	DSP subsystem standby status 0x0: DSP subsystem is in standby mode. 0x1: DSP subsystem is active.	R	0

Table 5−122. CM_AUTOIDLE_DSP

Address Offset	0x830		
Physical Address	0x4800 8830	**Instance**	PRCM
Description	This register controls the automatic control of the DSP domain interface clock activity.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:2	Reserved	Write 0s for future compatibility. Reads return 0	RW	0x00000000
1	AUTO_DSP_IPI	DSP IPI auto clock control 0x0: DSP Intrusive Port Interface clock is unrelated to the domain state transition. 0x1: DSP Intrusive Port Interface clock is automatically enabled or disabled along the domain state transition.	RW	0
0	Reserved	Write 0s for future compatibility. Reads return 0	RW	0

Table 5−123. CM_CLKSEL_DSP

Address Offset	0x840		
Physical Address	0x4800 8840	**Instance**	PRCM
Description	DSP clocks 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																		Reserved	Reserved					SYNC_DSP	CLKSEL_DSP_IF		CLKSEL_DSP				

Bits	Field Name	Description	Type	Reset
31:14	Reserved	Write 0s for future compatibility. Reads return 0	RW	0x00000
13	Reserved		RW	0
12:8	Reserved	Reserved	RW	0x01
7	SYNC_DSP	DSP Synchronizer control 0x0: Disable the DSP clock synchronizer. 0x1: Enable the DSP clock synchronizer.	RW	0
6:5	CLKSEL_DSP_IF	Selects DSP clock interface 0x0: Reserved 0x1: DSP interface clock is DSP_clk/1 0x2: DSP interface clock is DSP_clk/2 0x3: Reserved	RW	0x1
4:0	CLKSEL_DSP	Reserved 0x0: Reserved 0x1: Core_clk/1 0x2: Core_clk/2 0x3: Core_clk/3 0x4: Core_clk/4 0x6: Core_clk/6 0x8: Core_clk/8 0xC: Core_clk/12	RW	0x01

Table 5−124. CM_CLKSTCTRL_DSP

Address Offset	0x848		
Physical Address	0x4800 8848	**Instance**	PRCM
Description	This register controls the hardware supervised state transition of the two DSP clock sub-domains between ACTIVE and INACTIVE state.		
Type	RW		

31	30	29	28	27	26	25	24	23	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							AUTOSTATE_DSP

Bits	Field Name	Description	Type	Reset
31:9	Reserved	Write 0s for future compatibility. Reads return 0.	RW	0x000000
8	Reserved		RW	0
7:1	Reserved	Write 0s for future compatibility. Reads return 0.	RW	0x00
0	AUTO-STATE_DSP	DSP clock auto state control 0x0: Hardware supervised control of DSP sub-system clock is disabled. 0x1: Hardware supervised control of DSP sub-system clock is enabled.	RW	0

Table 5−125. RM_RSTCTRL_DSP

Address Offset	0x850		
Physical Address	0x4800 8850	**Instance**	PRCM
Description	This register controls DSP subsystems resets.		
Type	RW		

31:9	8	7:2	1	0
Reserved	Reserved	Reserved	RST2_DSP	RST1_DSP

Bits	Field Name	Description	Type	Reset
31:9	Reserved	Write 0s for future compatibility. Reads return 0.	RW	0x000000
8	Reserved		RW	1
7:2	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
1	RST2_DSP	DSP reset 2 control (MMU, LUT, IPI) 0x0: DSP IPI and MMU reset is cleared. 0x1: Resets DSP IPI and MMU	RW	1
0	RST1_DSP	DSP reset 1 control (UMA and DMA) 0x0: DSP UMA and DMA reset is cleared. 0x1: Resets DSP UMA and DMA	RW	1

Table 5−126. RM_RSTST_DSP

Address Offset	0x858		
Physical Address	0x4800 8858	**Instance**	PRCM
Description	This register logs the reset sources of DSP domain. Each bit is set on release of the respective reset source signal. Must be cleared by software.		
Type	RW		

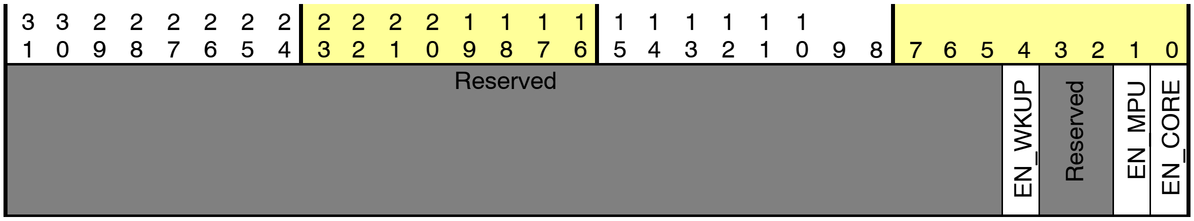
31:9	8	7:6	5	4	3	2	1	0
Reserved	Reserved	Reserved	DSP_SW_RST2	DSP_SW_RST1	COREWKUP_RST	DOMAINWKUP_RST	GLOBALWMPU_RST	GLOBALCOLD_RST

Table 5−126. RM_RSTST_DSP (Continued)

Bits	Field Name	Description	Type	Reset
31:9	Reserved	Write 0s for future compatibility. Reads return 0.	RW	0x000000
8	Reserved		RW	0
7:6	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
5	DSP_SW_RST2	DSP software reset 2 (MMU, LUT, IPI) 0x0: No DSP software reset 2. Write 0x0: Status bit is unchanged. 0x1: DSP-SS is reset on DSP software reset 2. Write 0x1: Status bit is reset.	RW	0
4	DSP_SW_RST1	DSP software reset 1 (UMA and DMA) Read 0x0: No DSP software reset 1. Write 0x0: Status bit is unchanged. Read 0x1: DSP-SS is reset on DSP software reset 1. Write 0x1: Status bit is reset.	RW	0
3	CORE WKUP_RST	Core domain wake-up reset Read 0x0: No core power domain reset. Write 0x0: Status bit is unchanged. Read 0x1: DSP domain is reset along with a core power domain reset. Write 0x1: Status bit is reset.	RW	0
2	DOMAIN WKUP_RST	Power domain wake-up reset Read 0x0: No DSP domain wake-up. Write 0x0: Status bit is unchanged. Read 0x1: DSP domain is reset following a DSP domain wake-up. Write 0x1: Status bit is reset.	RW	0
1	GLOBAL WMPU_RST	Global warm reset Read 0x0: No Global warm reset. Write 0x0: Status bit is unchanged. Read 0x1: DSP domain is reset on global warm reset. Write 0x1: Status bit is reset.	RW	0
0	GLOBAL-COLD_RST	Global cold reset Read 0x0: No global cold reset. Write 0x0: Status bit is unchanged. Read 0x1: DSP domain is reset on a global cold reset Write 0x1: Status bit is reset.	RW	1

Table 5−127. PM_WKDEP_DSP

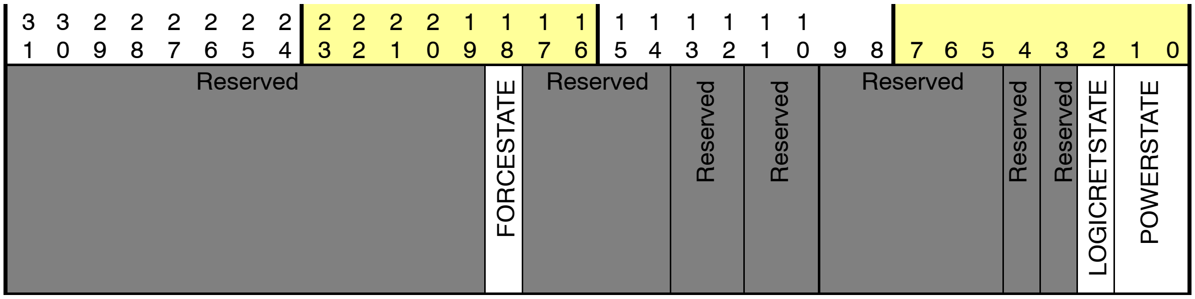
Address Offset	0x8C8		
Physical Address	0x4800 88C8	**Instance**	PRCM
Description	This register allows enabling or disabling the wake-up of the DSP domain on another domain wake-up.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:5	Reserved	Write 0s for future compatibility Reads return 0	RW	0x0000000
4	EN_WKUP	Wake-up domain dependency 0x0: DSP domain is independent of WKUP domain modules wake-up. 0x1: DSP domain is woken up on WKUP domain modules wake-up.	RW	1
3:2	Reserved	Write 0s for future compatibility. Reads return 0.	RW	0x0
1	EN_MPU	MPU domain dependency 0x0: DSP domain is independent of MPU domain wake-up. 0x1: DSP domain is woken up on MPU domain wake-up.	RW	1
0	EN_CORE	Core domain dependency 0x0: DSP domain is independent of core domain wake-up. 0x1: DSP domain is woken up on core domain wake-up.	RW	1

Table 5−128. PM_PWSTCTRL_DSP

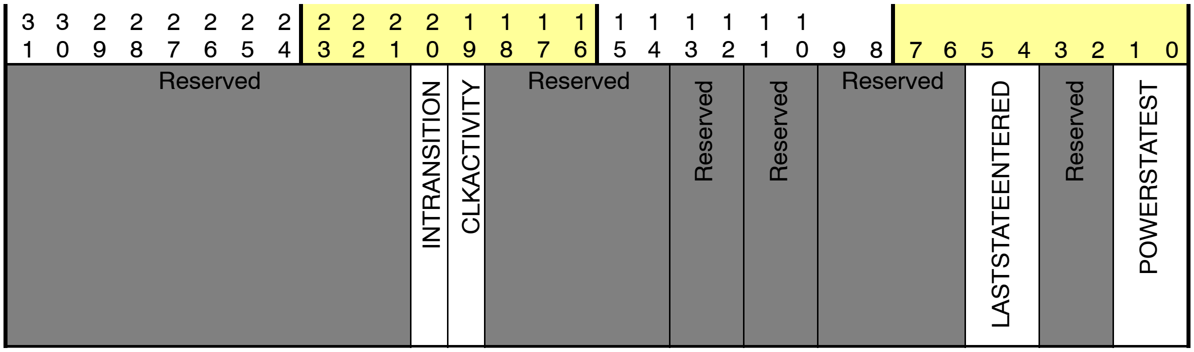
Address Offset	0x8E0		
Physical Address	0x4800 88E0	**Instance**	PRCM
Description	This register controls the DSP domain power state transition.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:19	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000
18	FORCESTATE	Software transition control 0x0: No transition or start wake-up transition. 0x1: Start software supervised state transition of DSP domain.	RW	0
17:14	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
13:12	Reserved		R	0x0
11:10	Reserved	DSP memories are internally controlled by DSP software. Write 0s for future compatibility. Read returns 0.	RW	0x0
9:5	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
4	Reserved		R	1
3	Reserved	DSP memories are internally controlled by DSP soft. Write 0s for future compatibility. Read returns 0.	RW	0
2	LOGICRETSTATE	Logic state when RETENTION 0x1: Logic is always retained when domain is in RETENTION state.	R	1
1:0	POWERSTATE	Power state control 0x0: ON state 0x1: RETENTION state 0x2: Reserved 0x3: OFF state	RW	0x0

Table 5−129. PM_PWSTST_DSP

Address Offset	0x8E4		
Physical Address	0x4800 88E4	**Instance**	PRCM
Description	This register provides a status on the power state transition of the DSP power domain.		
Type	R		



Bits	Field Name	Description	Type	Reset
31:21	Reserved	Write 0s for future compatibility. Read returns 0.	R	0x000
20	INTRANSITION	Power domain transition status 0x0: No transition 0x1: DSP power domain transition is in progress.	R	0
19	CLKACTIVITY	Clock activity status 0x0: No power domain clock activity 0x1: Power domain clock is active.	R	0
18:14	Reserved	Write 0s for future compatibility. Read returns 0.	R	0x00
13:12	Reserved	Reserved	R	0x0
11:10	Reserved	DSP memory status is readable in UMA registers only. Write 0s for future compatibility. Read returns 0.	R	0x0
9:6	Reserved	Write 0s for future compatibility. Read returns 0.	R	0x0
5:4	LASTSTATE ENTERED	Last power state entered 0x0: DSP power domain was ON. 0x1: DSP power domain was in RETENTION state. 0x2: Reserved 0x3: DSP power domain was OFF.	R	0x3
3:2	Reserved	Write 0s for future compatibility. Read returns 0.	R	0x0
1:0	POWERSTATEST	Current power state status 0x0: Power domain is ON. 0x1: Power domain is in RETENTION state. 0x2: Reserved 0x3: Power domain is OFF.	R	0x0

Table 5−130. PRCM_IRQSTATUS_DSP

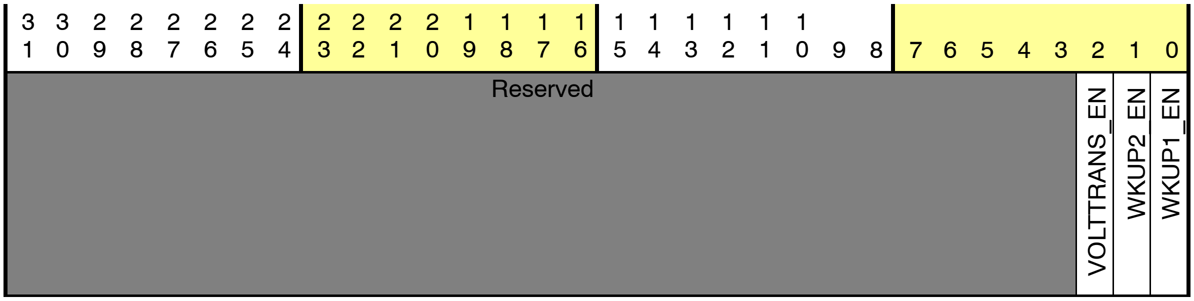
Address Offset	0x8F0		
Physical Address	0x4800 88F0	**Instance**	PRCM
Description	This interrupt status register regroups the statuses of module internal events that can generate interrupts. Writing 1 to a bit resets the bit. This register applies on interrupt line 1 mapped to the DSP interrupt controller.		
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																													VOLTTRANS_ST	WKUP2_ST	WKUP1_ST

Bits	Field Name	Description		Type	Reset
31:3	Reserved	Write 0s for future compatibility. Read returns 0.		RW	0x00000000
2	VOLTTRANS_ST	Voltage transition complete event status		RW	0
		Read 0x0:	Voltage transition complete event is false.		
		Write 0x0:	Status bit is unchanged.		
		Read 0x1:	Voltage transition complete event is true (pending).		
		Write 0x1:	Status bit is reset.		
1	WKUP2_ST	Wake-up event 2 status		RW	0
		Read 0x0:	Wake-up event is false.		
		Write 0x0:	Status bit is unchanged.		
		Read 0x1:	Wake-up event is true (pending).		
		Write 0x1:	Status bit is reset.		
0	WKUP1_ST	Wake-up event 1 status		RW	0
		Read 0x0:	Wake-up event is false.		
		Write 0x0:	Status bit is unchanged.		
		Read 0x1:	Wake-up event is true (pending).		
		Write 0x1:	Status bit is reset.		

Table 5−131. PRCM_IRQENABLE_DSP

Address Offset	0x8F4		
Physical Address	0x4800 88F4	**Instance**	PRCM
Description	The interrupt enable register allows masking/unmasking of module internal sources of interrupt, on a event-by-event basis. This register applies on interrupt line 0 mapped to the MPU interrupt controller.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:3	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000000
2	VOLTTRANS_EN	Voltage transition completed event mask 0x0: Voltage transition complete event is masked. 0x1: Voltage transition complete event generates an interrupt.	RW	0
1	WKUP2_EN	Wake-up event 2 mask 0x0: Wake-up event is masked. 0x1: Wake-up event generates an interrupt.	RW	0
0	WKUP1_EN	Wake-up event 1 mask 0x0: Wake-up event is masked. 0x1: Wake-up event generates an interrupt.	RW	0

Chapter 6

Internal Interconnect

This chapter describes the OMAP2420 device internal interconnect.

Topic **Page**

6.1 OMAP2420 Interconnect Overview

6.1.1 Terminology

- Initiator: A module is an initiator when it can initiate read and write requests to the chip interconnect (processors, DMA, etc.). Some modules are both initiator and target.
- Target: Unlike an initiator, a target module cannot generate read/write requests to chip interconnects, but instead responds to these requests. However, it may generate interrupts or DMA requests to the system (typically peripherals and memory controllers). Some modules are both initiator and target.
- OCP: OCPIP1.0 and OCPIP2.0 protocol (open-core protocol): point-to-point standard protocol between one master port and one slave port. This only document describes only aspects related to using the OMAP2420; for more information about OCP, see www.ocpip.org.
- OCP master port: A port that can generate OCP commands from an initiator module
- OCP slave port: A port that responds to OCP commands connected to a target module
- Interconnect: All decoding, routing, and arbitration logic enabling connection between multiple initiator modules and multiple target modules connected on the interconnect
- L1: Level 1 cache/memory close to the MPU core (not discussed in this chapter)
- L2: Level 2 cache/memory between L1 memories and L3 memories (not discussed here)
- L3: Level 3 interconnect, based on OCP. Handles transfers from initiators to targets with a maximum data width of 64 bits in the OMAP2420. System memories such as external SDRAM and external flash are L3 memories.
- L4: Level 4 interconnect, based on OCP. Peripheral interconnect, handles transfer from the L3 interconnect to any target module of the L4, with a maximum data width of 32 bits in the OMAP2420.
- LLRC: Low-latency response crossbar: subpart of the L3 interconnect optimized for low latency
- SB: Silicon backplane™: a subsection of the L3 interconnect optimized for bandwidth scheduling.
- Agent: Each module is connected to an interconnect by an agent, which is a bridge (sometimes configurable) between the module and the interconnect. A target module is connected by a target agent, while an initiator module is connected by an initiator agent.
- TM: Target module

- IM: Initiator module
- TA: Target agent
- IA: Initiator agent
- SBIM: Silicon backplane initiator module agent registers. All initiator modules interfaced with SB use the same set of registers for their initiator agent. All instances of SBIM are identical (the same registers and functionalities) except that they are mapped at different base addresses.
- SBTM: Silicon backplane target module agent registers. All target modules interfaced with the SB use the same set of registers for their target agent. All instances of SBTM are identical (the same registers and functionalities) except that they are mapped at different base addresses.
- SBIM+SBTM: Silicon backplane initiator and target module agent registers. When a module is both initiator and target, its initiator and target agents are sometimes collapsed (SBIM+SBTM) or they are split at a different base addresses (separate SBIM and SBTM). If agents are collapsed, initiator and target agent registers share the same base address and some registers are also shared. Apart from the fact that some of the registers are shared, SBIM+SBTM behavior is identical to an assembly composed of a stand-alone SBIM and a stand-alone SBTM. All instances of SBIM+SBTM are identical (the same registers and functionalities) except they are mapped at different base addresses.
- CONNID: Connection identifier: an initiator module identifier. A CONNID is transmitted in-band with the request and is used for security and error logging. There are four different CONNID types in the system, and one module can be identified by different CONNID types depending on where the information is found in the system:
  - System CONNID: Main CONNID, used for:
    - Firewall setting in L3
    - Error logging in LLRC, GPMC, and SDRC
    - Firewall error logging except in SBFW for DSP
  - : Silicon backplane CONNID, used for:
    - L3 SBTM target agent error−logging registers
    - SBFW firewall error−logging registers for DSP
  - L4CONNID: Used in L4 interconnect for nonsecure transactions
- Firewall: A firewall is a programmable security feature integrated with each target agent for which a module requires a secure data path. A firewall can change access permissions from the different initiator modules to the protected target module address space.
- Address space (L3 interconnect only): One target module or submodule can have up to four address spaces that are not necessarily contiguous. Permissions are region-based in an address space.

- In-band signal: Any OCP signal that is part of a clearly identified OCP transfer. Typically, command, address, byte enables, etc. Their behavior is defined by OCP semantics.
- Out-of-band: Any OCP signal, the behavior of which is not associated with a precise OCP transaction. The OCP standard does not define any specific semantics for these signals. Typically, interrupt requests, DMA requests, asynchronous error flags are out-of-band signals.
- Firewall comparison mechanism: Comparison made in a firewall between access in-band qualifiers and access permissions programmed in firewall registers. If the comparison is successful, access is allowed; otherwise, access is denied.
- MREQ in-band qualifiers: 2 MREQ qualifiers can be used in a firewall comparison mechanism:
  - MREQ_DEBUG: 1 = Debug access, 0 = Functional access
  - MREQ_TYPE: 1 = Instruction, 0 = Data
- MREQ combination: Used for firewall configuration to determine REQ_INFO_PERMISSION value. Pattern composed of the combination of selected MREQ in−band parameters like MREQ_debug, MREQ_secure, and MREQ_Type. MREQ combination size depends on the number of MREQ in−band qualifiers used by the firewall comparison mechanism.
- REQ_INFO_PERMISSION: Pattern used for firewall configuration. Permission is defined based on the MREQ in−band qualifier value.
- Thread: Logical channel. Independent transfer streams are separate and belong to different threads.
- Multithreaded port: A physical port that can 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 other threads. But the order between threads can change between requests and responses. Thread management is used for bandwidth optimization and is handled automatically by the system.

6.1.2 L3 and L4 Roles

Four communication layers, L1 through L4, exist in the OMAP2420. L1 is internal to CPUs; it concerns data exchange with the internal L1 cache memory subsystem, and it is the closest memory to the MPU CPU core. L2 is reserved for L2 cache, but is not used in OMAP2420. The chip-level interconnect is composed of of L3 and L4 and enables communication among all modules and subsystems. L3 handles many types of data transfer and in particular, exchanges with system on-chip/external memories. L4 handles only transfers to basic peripherals.

Interconnect layers enable communication between all instances of the OMAP2420. Interconnects assume routing of all data exchanges at chip level. This interconnect is based on OCP.

Each module/subsystem is connected to an interconnect port through a dedicated interface agent, which can be configured to tune the access, depending the characteristics of the module.

Some data paths are secured by a configurable firewall, which can filter accesses according to different criteria.

L3 and L4 interconnect default setting is fully functional and enables all possible functional data paths. However, it is possible to change the interconnect parameters to better fit the user's expectation.

6.1.2.1 Topology of the Interconnect

- L3: Handles transfers from initiators to targets with a maximum data width of 64 bits. The L3 interconnect is a big-endian platform. It can be divided into two submodules:
  - LLRC: Low-latency response crossbar, which is a custom routing between modules optimized for latency. It is a true crossbar structure, connecting all processors to the L3 memories: SDRAM, flash, on-chip L3 RAM, and ROM.
  - SB: Silicon backplane TM that is optimized for bandwidth allocation, including a 64-bit shared data path.
- L4: Support for 32-bit data-width transfer to peripherals. This backplane is optimized to support the interconnection of a high number of peripherals. The L4 assumes little-endian transactions for all narrower targets (8-bit and 16-bit), when performing data packing and unpacking.

Modules can be connected to an interconnect through an agent: initiator agent (IA) for initiator module (IM) and target agent (TA) for target modules (TM), respectively.

Some agents include a firewall (FW) for security. A firewall can limit target access permission to a restricted number of initiator modules. These firewalls can be configured by software. See Section 6.2.5.7, *L3 Firewall*, and Section 6.4.4.5, *Firewall*.

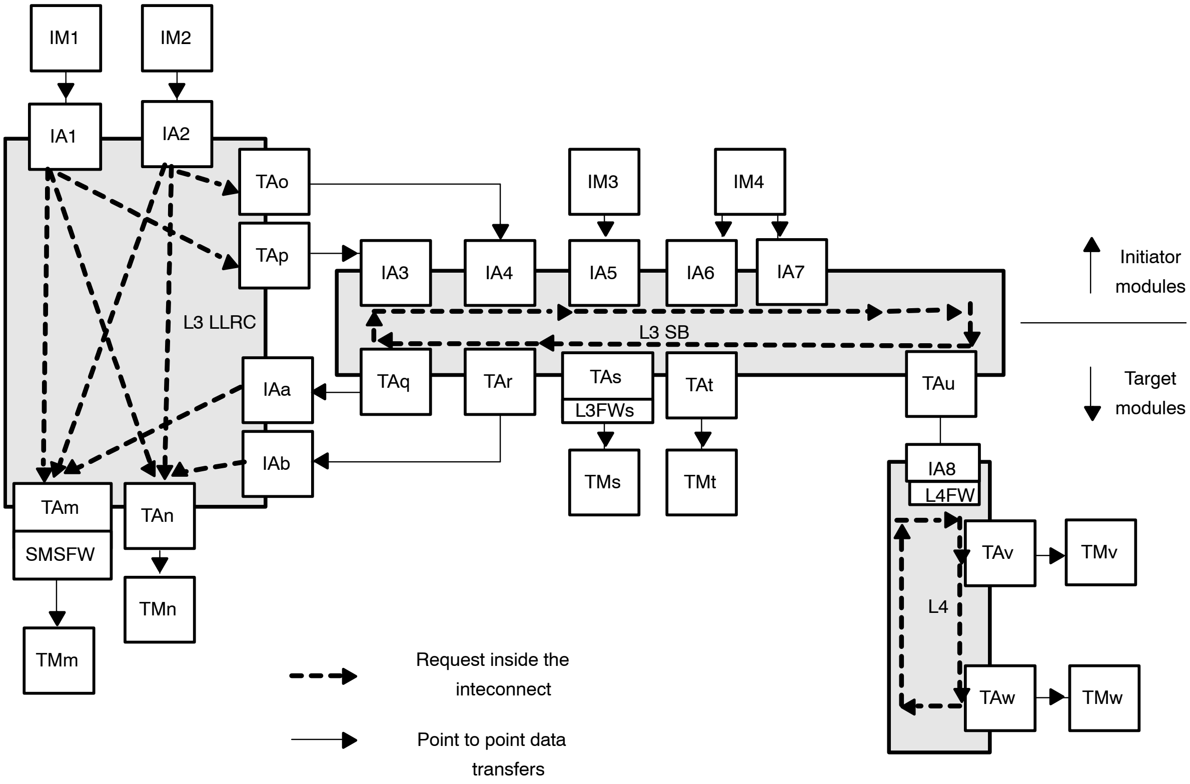
Any transaction in the system interconnect is tagged by an in-band qualifier, which uniquely identifies the initiator at a given interconnect point. This signal field is named CONNID, and is used in the error−logging logic, as well as in the firewall−control logic. This ID is not constant over the entire interconnect network; care must be taken to use the correct ID at the correct location. CONNID tables can be found in Table 6−31.

6.1.2.2 Block Diagram

Figure 6−1 shows the structure of the interconnect.

- A module is always associated with at least one agent, to ensure connection with the interconnect.
- Interconnection between interconnect subsystems is handled by a handshake mechanism between a target agent of an interconnect with an initiator agent of the other interconnect.
- All data exchanges outside the interconnect are point-to-point accesses.
- There are different types of firewall: L3FW (for all SB firewalls and some LLRC firewalls), L4FW (for the L4 interconnect firewalls), SMSFW firewall (specific firewall for the SMS module).
- A target module of the L4 can be accessed only through the L3 SB.

Figure 6−1. Interconnect Block Diagram



The port numbers in Figure 6−1 are not exhaustive. Figure 6−1 shows the various numbers of modules that can be interconnected with this architecture and how blocks are organized.

6.1.2.3 Module Distribution

LLRC Modules

Table 6−1. LLRC Initiator Agents

Initiator Agents	
Name	**Description**
MPU SS	MPU subsystem (based on ARM1136 processor core)
DSP SS	DSP subsystem (based on TMS320C55x DSP core)

Table 6−2. LLRC Target Agents

Target Agents	
Name	**Description**
SMS	SDRAM memory scheduler
GPMC	General-purpose memory controller (for flash memory, SRAM, SROM, etc.)
OCM−ROM	On-chip memory ROM
OCM−RAM	On-chip memory RAM

SB Agents

Table 6−3. SB Initiator Agents

Initiator Agents	
Name	**Description**
MPU2SB	Bridge from LLRC to SB, to allow MPU to access SB targets
s-DMA R	System DMA read port
s-DMA W	System DMA write port
DSS	Display subsystem
CAMERA SS	Camera subsystem
USB	Universal serial bus on the go
DSP2SB	Bridge from LLRC to SB, to allow DSP to access SB targets

Note: vvvvvvvvvvvvvv

Table 6−4. SB Target Agents

Target Agents	
Name	**Description**
SB2SMS	Bridge from SB to LLRC, to allow SB initiators to access SMS and SDRC targets.
SB2GPMC	Bridge from SB to LLRC, to allow SB initiators to access GPMC target.
L4	SB target agent for L4 interconnect
SB2OCM	Bridge from SB to LLRC, to allow SB initiators to access on−chip memory RAM and ROM
DSP Mem	DSP core memory

L4 Agents

Table 6–5. L4 Initiator Agent

Initiator Agents	
Name	**Description**
L4 IA	Initiator agent
L4 LA	Link agent
L4 AP	Access and protection: L4 mapping and access permission

Table 6–6. L4 Target Agents

Target Agents	
Module Name	**Description**
System control and pinout	System control module (SCM) and pinout configuration
32K timer	Timer 32K
PRCM	Power reset and clock management
Quad GPIO	4x general-purpose I/O blocks (32 GPIO each)
WD timer1/2	2x watchdog timer
WD timer 3 (DSP)	DSP watchdog timer
DSS	Display subsystem configuration port
Camera	Camera subsystem configuration port
sDMA	System DMA (sDMA) configuration port
USB	Universal serial bus on the go (OTG) slave port
UART1	Universal asynchronous receiver transmitter (modem or general-purpose)
UART2	Universal asynchronous receiver transmitter (modem or general-purpose)
UART3	Universal asynchronous receiver transmitter (modem/IrDA/CIR)
I2C1	Multimaster inter-integrated circuit 1
I2C2	Multimaster inter-integrated circuit 2
McBSP1	Multichannel buffered serial port 1
McBSP2	Multichannel buffered serial port 2
GP timer 1	General-purpose timer 1: Gauging/MPU
GP timer 2	General-purpose timer 2: MPU
GP timer 3	General-purpose timer 3: MPU
GP timer 4	General-purpose timer 4: MPU
GP timer 5	General-purpose timer 5: MPU/DSP L1&L2
GP timer 6	General-purpose timer 6: MPU/DSP L1&L2
GP timer 7	General-purpose timer 7: MPU/DSP
GP timer 8	General-purpose timer 8: MPU/DSP
GP timer 9	General-purpose timer 9: MPU + Debug

Table 6−6. L4 Target Agents (Continued)

Target Agents	
Module Name	**Description**
GP timer 10	General-purpose timer 10: MPU + Debug
GP timer 11	General-purpose timer 11: MPU + Debug
GP timer 12	General-purpose timer 12: MPU + Debug
EAC	Enhanced audio controller
FAC	Frame adjustment counter
IPC	Interprocessor communication
SPI1	Serial port interface 1
SPI2	Serial port interface 2
MMC SDIO	Multimedia memory controller SDIO
HDQ/1-Wire	Single-wire serial-link low rate (application example: battery monitoring)

6.1.2.4 Connectivity Matrix

Table 6−7 lists all of the functional paths between the L3 interconnect initiator module and target module.

Table 6−7. Connectivity Matrix

Initiator	**Initiator Ports**	**Target**				
		L4	**LLRC**			**Silicon Backplane**
		L4 Peripheral Interconnect[1]	**SMS**	**GPMC**	**OCMRAM & ROM**	**DSP Memory**
MPU	MPU RD	+	+	+	+	+
	MPU WR	+	+	+	+	+
	MPU instruction	+	+	+	+	+
LCD	LCD		+	+	+	
Camera	Camera		+	+	+	
	MMU		+	+	+	
DSP DMA	dDMA RD	+	+	+	+	
	dDMA WR	+	+	+	+	
DSP	DSP Data	+	+	+	+	+
	DSP Instruction		+	+	+	+
	DSP MMU[2]		+	+	+	+
USB	USB		+	+	+	
System DMA	sDMA RD	+	+	+	+	+
	sDMA WR	+	+	+	+	+
	MMU					

1) A functional data path always exists from an L4 initiator agent to any L4 target module. As a consequence, all L3 initiator modules for which a data path exists to L4 can access all L4 peripherals.

2) When an MMU miss occurs and the DSP MMU is the original initiator of the transfer, DSP MMU connectivity indicates the only possible paths. In other cases, transfer requests are initiated by other DSP subsystem initiator modules and pass through the DSP MMU for address remapping. In such cases, the DSP MMU can remap accesses according to the permission of the original initiator (dDMA RD, dDMA WR, DSP data, DSP instruction). The functional path to be considered is the path from the original initiator module to the target module without considering whether it is remapped by the DSP MMU.
Cell contains a + when a functional path exists.
Cell is blank when no functional path exists.

6.2 L3 Interconnect

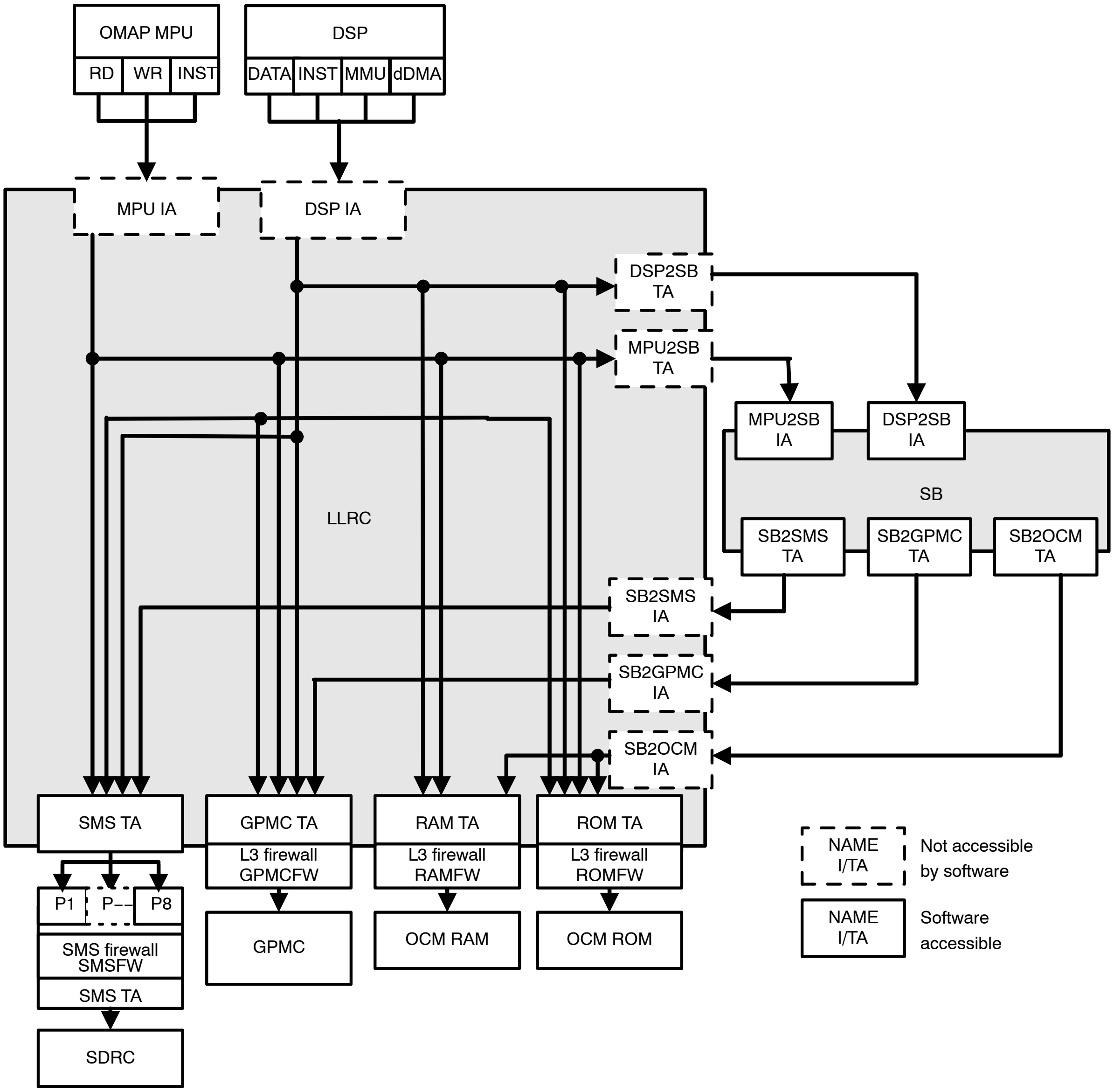
6.2.1 Low-Latency Response Crossbar Overview

The low-latency response crossbar (LLRC) interconnect is a crossbar that is optimized for minimum-latency; transfers from the initiator to the target are controlled by addresss-decoding control muxes. The LLRC has the following distinguishing features:

- ❑ There is no address miss in the LLRC. Any request that does not target one of the LLRC memory controllers (SDRC/GPMC/OCM-RAM/ROM) is forwarded to the SB where it can either reach an SB target module or generate an SB address miss.
- ❑ There is no time-out in the LLRC. System-level time-out is distributed in the SB section of the L3, the L4 interconnect, and the GPMC target core. Protocol-wise, other memory targets like external SDRAM (SMS/SDRC cores) or on-chip RAM/ROM memories cannot time-out.
- ❑ Aliasing effects exist in LLRC; see Section 6.2.5.6, *Aliasing Effects (LLRC Only)*.

Figure 6−2 shows all the possible data paths between modules inside LLRC, interconnection with SB through agents, all LLRC agents, and related SB agents.

Figure 6−2. LLRC Block Diagram



LLRC agents cannot be configured and cannot be controlled by software:

- Initiator agents contain no registers.
- Only registers accessible by software in the target agents are error−logging registers.

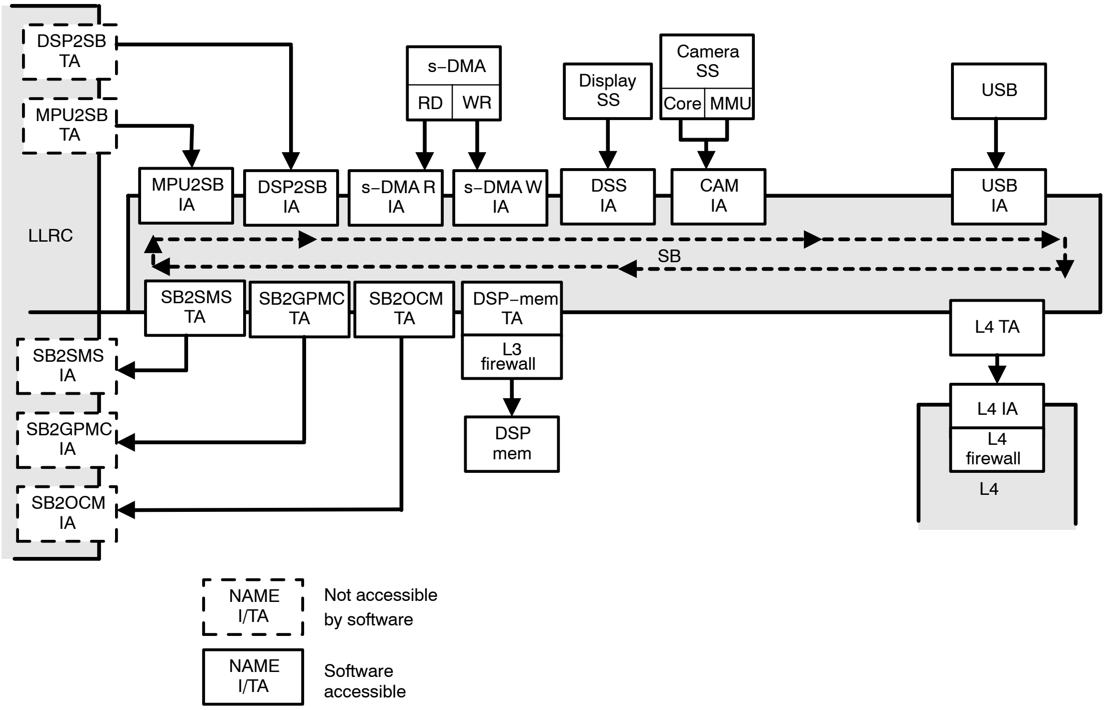
6.2.2 Silicon Backplane Overview

L3 silicon backplane is an architecture optimized for high−bandwidth transfer. When a target responds to a request, a transfer can occur. SB has the following characteristics:

- Address miss can occur in the L3 SB.
- Time-out can occur in the L3 SB.

- ❑ There is no address aliasing in L3 SB.
- ❑ Latency is not deterministic and depends on traffic collisions.

Figure 6−3. Silicon Backplane Block Diagram



L3 silicon backplane agents provide identification, time-out configuration, or error-management registers that are interface-accessible by software. There is a common register organization for all target agents (SBTM) and initiator agents (SBIM) in the L3 SB.

6.2.3 Port List and Description

Some modules have several related agents, for several possible reasons:

1) A module has different non-merged ports (sDMA).

2) A module is an LLRC initiator that can access SB: Transfer is routed outside the LLRC through an LLRC target agent, then addresses the SB through an SB initiator agent (MPU).

3) A module is an LLRC target and that be accessed by SB: Transfer is routed outside the SB through an SB target agent, then addresses the LLRC through a LLRC initiator agent (access to SMS, OCM, RAM/ROM, GPMC from SB initiators).

6.2.3.1 Initiator Modules

This section includes a table for each initiator module, providing a snapshot of the interconnect port(s): number of physical ports, port width, burst capability, presence of potential programming registers, and CONNID information that must be used to identify the transactions generated by that port at various locations in the system.

An additional table provides the names and types of all related agents. For a thorough description of how to program the agents, see Section 6.3, *L3 Interconnect Registers*, and Section 6.5, *L4 Interconnect Registers*.

6.2.3.2 LLRC Initiators

MPU Subsystem

Requests from the data unit and the instruction unit are submitted to the L3 interconnect using three separate ports. To optimize bandwidth, requests between these ports can be treated out of order by the system. This reordering is handled automatically by the system.

Table 6−8. MPU Subsystem Ports

Shared Port	Bus Width (bits)	Burst Support	Configuration Registers IM	Firewall Setting CONNID Type	Error−Logging CONNID
RD Data	64	Yes	No	SystemConnID-CONNID Value: 1	SB error log: SBCONNID Value: E Firewall and other L3 error log: SystemCONNID Value: 1
WR Data		Yes		SystemConnID-CONNID Value: 1	SB error log: SBCONNID Value: E Firewall and other L3 error log: SystemCONNID Value: 1
INST		Yes		SystemConnID-CONNID Value: 1	SB error log: SBCONNID Value: E Firewall and other L3 error log: SystemCONNID Value: 1

Table 6−9. MPU Subsystem−Related Agents

Related Agents				
Agent Type	**Agent Name**	**Module Name**	**Firewall Protection**	**Configuration Registers**
L3 LLRC IA	Not software-visible	MPU SS	No	No
L3 LLRC TA	Not software-visible	MPU2SB	No	No
L3 SB IA	IMMPU2SB	MPU2SB	No	SBIM

DSP Subsystem

Requests from DSP subsystem internal initiators are merged by the DSP subsystem in one 64-bit bus. To optimize bandwidth, these requests can be treated out of order by the system. This reordering is handled automatically by the system and implies no behavioral change.

Even though requests are merged, each DSP DMA request is identified by a different CONNID identifier.

Table 6−10. DSP Subsystem Ports

Shared Port	Bus Width (bits)	Burst Support	Configuration Registers IM	Firewall Setting CONNID Type	Error-Logging CONNID[1]
DSP Data	64	No	No	SystemCONNID Value: 7	SB error log: SBCONNID Value: B Firewall and other L3 error log: SystemCONNID Value: 7
DSP INST	64	Yes	No	SystemCONNID Value: 7	SB error log: SBCONNID Value: B Firewall and other L3 error log: SystemCONNID Value: 7
DSP MMU	64	No	No	SystemCONNID Value: 7	SB error log: SBCONNID Value: B Firewall and other L3 error log: SystemCONNID Value: 7
DSP DMA	64	Yes	No	SystemCONNID Value: 2	SB error log: SBCONNID Value: 8/9/A† Firewall and other L3 error log: SystemCONNID Value: 2

1) At SB level, different CONNIDs identify dDMA. These different CONNIDs correspond to different threads: 2 threads for RD (8/9) and one thread for WR (A). RD threads and WR thread are merged by the DSP subsystem on the DSP port. Request ordering is respected within a thread, while requests between different threads can be treated out of order by the system to optimize bandwidth. This reordering is handled automatically by the system and implies no behavioral change.

Table 6−11. DSP Subsystem−Related Agents

Related Agents				
Agent Type	**Agent Name**	**Module Name**	**Firewall Protection**	**Configuration Registers**
L3 LLRC IA	Not visible	DSP SS	No	No
L3 LLRC TA	Not visible	DSP2SB	No	No
L3 SB IA	IM7	DSP2SB	No	SBIM

6.2.3.3 SB Initiators

Display Subsystem

Table 6–12. Display Subsystem Ports

Port	Bus Width (bits)	Burst Support	Configuration Registers IM	Firewall Setting CONNID Type	Error-Logging CONNID
DSS	32	Yes	Yes	SystemCONNID Value: 8	SB error log: SBCONNID Value: D Firewall and other L3 error log: SystemCONNID Value: 8

Camera

Table 6–13. Camera Subsystem Ports

Port	Bus Width (bits)	Burst Support	Configuration Registers IM	Firewall Setting CONNID Type	Error-Logging CONNID
Camera core	64	Yes	Yes	SystemCONNID Value: B	SB error log: SBCONNID Value: C Firewall and other L3 error log: SystemCONNID Value: B
cMMU		Yes		SystemCONNID Value: B	SB error log: SBCONNID Value: C Firewall and other L3 error log: SystemCONNID Value: B

Table 6–14. Camera Subsystem-Related Agent

Related Agent				
Agent Type	**Agent Name**	**Module Name**	**Firewall Protection**	**Configuration Registers**
L3 SB IA	IM4	CAM	No	SBIM

System DMA

There are two ports and two different initiator agents, one for sDMA RD and one for sDMA WR, but from a protection point of view, the permissions are identical for both ports and consequently the same identifier is assigned to the two ports.

Table 6−15. sDMA Ports

Shared Port	Bus Width (bits)	Burst Support	Configuration Registers IM	Firewall Setting CONNID Type	Error-Logging CONNID[1]
sDMA RD	32	Yes	Yes	SystemCONNID Value: 3	SB error log: SBCONNID Value: 10/11/12/1B/1E/1F Firewall and other L3 error log: SystemCONNID Value: 3
sDMA WR	32	Yes	Yes	SystemCONNID Value: 3	SB error log: SBCONNID Value: 10/11/12/1B/1E/1F firewall and other L3 error log: SystemCONNID Value: 3

1) At the SB level, there are different CONNID to identify sDMA. These different CONNIDs correspond to different threads: 4 threads for RD (10/11/12/1B) and two threads for WR (1E/1F). RD threads are merged by the sDMA on the RD port; WR threads are merged by sDMA on the WR port. Request ordering is respected within a thread, while requests between different threads can be treated out of order by the system to optimize bandwidth. This reordering is handled automatically by the system and implies no behavioral change.

Table 6−16. sDMA−Related Agents

Related Agents				
Agent Type	**Agent Name**	**Module Name**	**Firewall Protection**	**Configuration Registers**
L3 SB IA	IM1	sDMA R	No	SBIM
L3 SB IA	IM2	sDMA W	No	SBIM

USB

Table 6−17. USB Port

Port	Bus Width (bits)	Burst Support	Configuration Registers IM	Firewall Setting CONNID Type	Error-Logging CONNID
USB	32	No	Yes	SystemCONNID Value: 9	SB error log: SBCONNID Value: 16 Firewall and other L3 error log: SystemCONNID Value: 9

Table 6−18. USB Related Agent

Related Agent				
Agent Type	**Agent Name**	**Module Name**	**Firewall Protection**	**Configuration Registers**
L3 SB IA	IM6	USB	No	SBIM

6.2.3.4 Target Modules (LLRC Targets)

SMS/SDRC

The SMS port is a multithreaded 64-bit port shared by eight different threads. Incoming requests to the SMS are collapsed into the eight threads by the target agent based on the initiator of the request. A thread can be dedicated to an initiator, or can be used for a group of initiators, as indicated in Table 6−19. The thread in that section is identical to the concept of queues described in the SDRC section of Chapter 12. Arbitration between the threads is controlled by the SMS. Priorities are software-programmable in the SMS module (see Chapter 12, *Memory Subsystem*). The granted request is forwarded to the SDRC memory controller.

Request order in a frame is preserved, while requests between different threads can be treated out of order by the SMS.

Table 6−19. SMS Ports

LLRC Incoming Port Bus width (bits)	SMS Threads	Burst Support	Configuration Registers TM	Aliasing	Firewall	Outgoing SDRC Port Bus Width (bits)
64	SMS Thread 0 MPU-SS	Yes	Error log only	No	Yes; specific SMS firewall	64
64	SMS Thread 1 DSP−SS CPU/ MMU	Yes	Error log only	No	Yes; specific SMS firewall	64
64	SMS Thread 2 sDMA−RD dDMA	Yes	Error log only	No	Yes; specific SMS firewall	64
64	SMS Thread 4	Yes	Error log only	No	Yes; specific SMS firewall	64
64	SMS Thread 5 sDMA-WR USB	Yes	Error log only	No	Yes; specific SMS firewall	64
64	SMS Thread 6 Camera-SS	Yes	Error log only	No	Yes; specific SMS firewall	64
64	SMS Thread 7 DSS	Yes	Error log only	No	Yes; specific SMS firewall	64

Table 6−20. SMS−Related Agents

Related Agents				
Agent Type	**Agent Name**	**Module Name**	**Firewall Protection**	**Configuration Registers**
L3 LLRC TA	Not visible	SMS	SMSFW	LLRC error log
L3 LLRC IA	Not visible	SB2SMS	No	No
L3 SB TA	TM1	SB2SMS	No	SBTM

GPMC

Table 6−21. GPMC Port

Port	Bus Width (bits)	Burst Support	Configuration Registers TM	Aliasing	Firewall
GPMC	64	Yes	Error log only	No	Yes; generic L3 firewall

Table 6−22. GPMC−Related Agents

Related Agents				
Agent Type	**Agent Name**	**Module Name**	**Firewall Protection**	**Configuration Registers**
L3 LLRC TA	Not visible	GPMC	GPMCFW	LLRC error log
L3 LLRC IA	Not visible	SB2GPMC	No	No
L3 SB TA	TM2	SB22GPMC	No	SBTM

OCM ROM

Table 6−23. OCM ROM Port

Port	Bus Width (bits)	Burst Support	Configuration Registers TM	Aliasing	Firewall
OCM ROM	32	Yes	Error log only	Yes	Yes; generic L3 firewall

Note: Requests from SB to OCM−RAM and OCM−ROM are merged by the TM3. TM3 is common to OCM−ROM and OCM−RAM.

Table 6−24. OCM−Related Agents

Related Agents				
Agent Type	**Agent Name**	**Module Name**	**Firewall Protection**	**Configuration Registers**
L3 LLRC TA	Not visible	OCM−ROM	ROMFW	LLRC error log
L3 LLRC IA	Not visible	SB2OCM	No	No
L3 SB TA	TM3	SB2OCM	No	SBTM

OCM RAM

Table 6−25. OCM RAM Port

Port	Bus Width (bits)	Burst Support	Configuration Registers TM	Aliasing	Firewall
OCM RAM	32	Yes	Error log only	Yes	Yes; generic L3 firewall

Note: Requests from SB to OCM RAM and OCM ROM are merged by the TM3. TM3 is common to OCM ROM and OCM RAM.

Table 6−26. OCM RAM−Related Agent

Related Agents				
Agent Type	**Agent Name**	**Module Name**	**Firewall Protection**	**Configuration Registers**
L3 LLRC TA	Not visible	OCM−RAM	RAMFW	LLRC error log
L3 LLRC IA	Not visible	SB2OCM	No	No
L3 SB TA	TM3	SB2OCM	No	SBTM

6.2.3.5 Target Modules (SB Targets)

DSP Memory

Table 6−27. DSP Memory Port

Port	Bus Width (bits)	Burst Support	Configuration Registers TM	Aliasing	Firewall
DSP mem	32	Yes	Yes; L3 SB TM	No	Yes; generic L3 firewall

Table 6−28. DSP Memory−Related Agent

Related Agent				
Agent Type	**Agent Name**	**Module Name**	**Firewall Protection**	**Configuration Registers**
L3 SB TM	TM4	DSP Mem	DSPFW	SBTM

L4 Interconnect for Access to Peripherals

The L4 interconnect port is a multithreaded 32−bit ports shared by four 32−bit threads. The firewall is implemented in the L4 interconnect entity (see Section 6.4, *L4 Interconnect*).

Table 6−29. L3 Target Port to L4 Interconnect Initiator

Port Bus Width (bits)	Ports	Burst Support	Configuration Registers TM	Aliasing	Firewall
32	L4T Thread0: sDMA	Yes	Yes; L3 SB TM	No	Yes; specific L4 firewall
32	L4T Thread1: DSP SS	Yes	Yes; L3 SB TM	No	Yes; specific L4 firewall
32	L4T Thread2: sDMA	Yes	Yes; L3 SB TM	No	Yes; specific L4 firewall
32	L4T Thread3: MPU	Yes	Yes; L3 SB TM	No	Yes; specific L4 firewall

Table 6−30. L3 Related Agent for L4 Interconnect Initiator

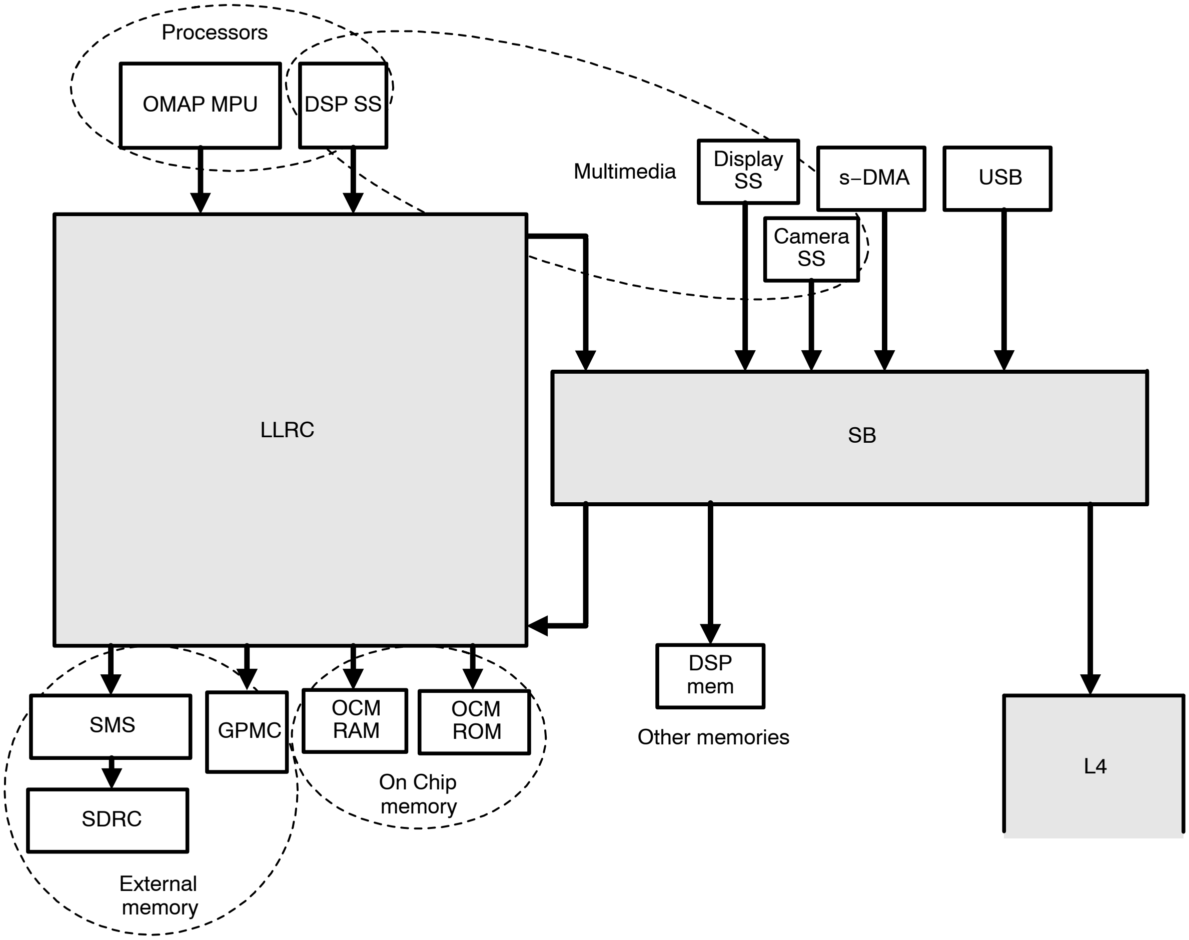
Related Agents				
Agent Type	**Agent Name**	**Module Name**	**Firewall Protection**	**Configuration Registers**
L3 SB TM	L4TA	L4	L4FW[1]	No

[1] Firewall protection L4F(AP: Address and Protection) is described in Section 6.4, *L4 Interconnect.*

6.2.4 L3 Functional Overview

Figure 6−4 is a block diagram of the L3 interconnect.

Figure 6−4. L3 Block Diagram



6.2.4.1 Feature List

The L3 interconnect has the following features:

❑ 64-bit L3 multipath interconnect to eliminate on-chip bottlenecks

❑ SDRAM memory scheduler (SMS) that ensures quality-of-service (QoS) for real-time hardware operators while maintaining optimal memory latency for CPU accesses

- True big-endian platform
- Transaction error tracking and logging
- One interrupt signaling transaction error

6.2.5 Functional Description

6.2.5.1 Initiator Identification in the OMAP2420

A CONNID is an initiator module identifier. The same initiator module is referenced by different CONNIDs, depending on where the access takes place.

An initiator module can have three CONNIDs:

- SystemCONNID:
  - Used to set read/write permissions in all L3 firewalls. See the *ConnID, Read Permission,* and *Write Permission* subsection of Section 6.2.5.7.
  - Used in all L3 firewall error-logging registers
  - Used in L3 LLRC error-logging registers
- SBCONNID: Used in L3 SB error-logging registers to identify the initiator module that caused the error, in both initiator and target agent.
- L4CONNID: This CONNID is used in the L4 micronetwork only for transactions that do not have the secure attribute. This L4CONNID is a subset of systemCONNID.

Figure 6−5 shows the CONNID types.

Note:

The system CONNID is also used in the SMS/SDRC/GPMC modules to identify the owner of a transaction that causes an error in these modules.

Figure 6−5. CONNIDs Used in the Interconnect

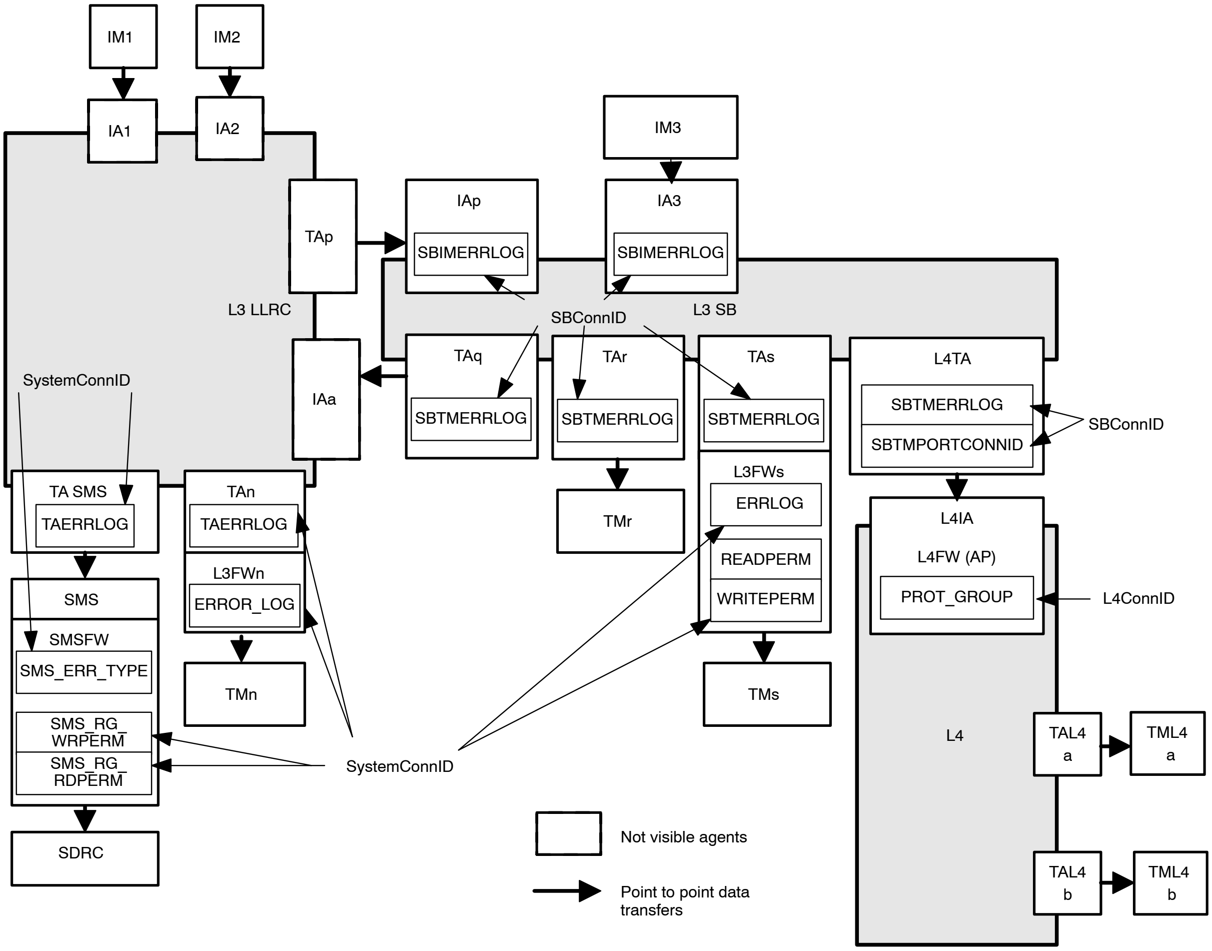


Table 6−31 corresponds to Figure 6−5 and indicates the CONNID type for each initiator. When no CONNID is available for an initiator, this is indicated by NR (not relevant).

Table 6–31. CONNID List and Affectation

		L3 Interconnect	L4 Interconnect	
Subsystem	**Port**	**sbCONNID**	**L4 CONNID**	**L4SecCONNID**
MPU	MPU data RD	E	1	9
	MPU data WR			
	MPU instruction			
LCD	LCD	D	NR	NR
Camera	Camera	C	NR	NR
	MMU		NR	NR
DSP DMA[1]	dDMA RD	8	2	NR
		9		
	dDMA WR	A		
DSP	DSP data	B	7	NR
	DSP instruction			
	DSP MMU			
USB	USB	16	NR	NR
System DMA[†]	sDMA RD	10	3	B
		11		
		12		
		1B		
	sDMA WR	1E		
		1F		

1) sDMA and dDMA have multiple SBCONNIDs for one SystemCONNID/L4CONNID/L4SecCONNID because each SBCONNID refers to a different thread. From a user point of view, all SBCONNIDs identify the same initiator.

The system CONNID, L4CONNID, and L4SecCONNID are all encoded on 4 bits. The CONNIDs are related as follows:

- If an L4CONNID exists for the initiator module, the three lower bits of the system CONNID are the L4CONNID.
- The fourth bit of the L4CONNID corresponds to the secure bit. If it is read/set to 0, the transaction is performed by an initiator without the secure attribute; this is L4CONNID. If this fourth bit is set/read to 1, it corresponds to an initiator with the secure attribute; this is L4SecCONNID.

6.2.5.2 Burst Support

The L3 interconnect includes full hardware support for burst accesses; on each path from one initiator module to one target module, the interconnect handles burst protocol and can adapt to target module capabilities to ensure maximum efficiency. Burst adaptation is totally transparent to the user and nothing can be configured apart from the burst programming on the initiator side.

6.2.5.3 Endianness Management

The L3 interconnect is an endian-aware platform; this means that all transfers are tagged with an in-band endianness qualifier. This endian qualifier takes into account only data-width conversion (required when initiator and target module use different data widths). When data-width conversion occurs, data are packed and unpacked with respect to their endianness to preserve data integrity.

Packing consists of merging several back-to-back transfers at consecutive addresses on an N-bit data bus in one transfer on a larger data bus. The ratio between the narrow side and the wide side is generally 2 or 4.

Unpacking is the opposite operation: splitting one transfer on the wider path into a sequence of transfers at consecutive addresses on the narrower path.

Packing or unpacking operations imply no endianness conversion; there is no byte reordering.

WARNING

The software manages byte order to respect the endianness of data during transfers between initiators and targets of the interconnect. Requests must have the same endianness as the data transferred.

Figure 6−6 shows a 32-bit-to-64-bit packing operation in little-endian mode. The first 32-bit transfer, which corresponds to the lowest 32-bit address, is put into the lowest part of the 64-bit word; the second transfer, which corresponds to the highest 32-bit address, is put into the highest part of the 64-bit word. The unpacking operation is symmetric.

Figure 6−6. Little−Endian Packing/Unpacking Operation

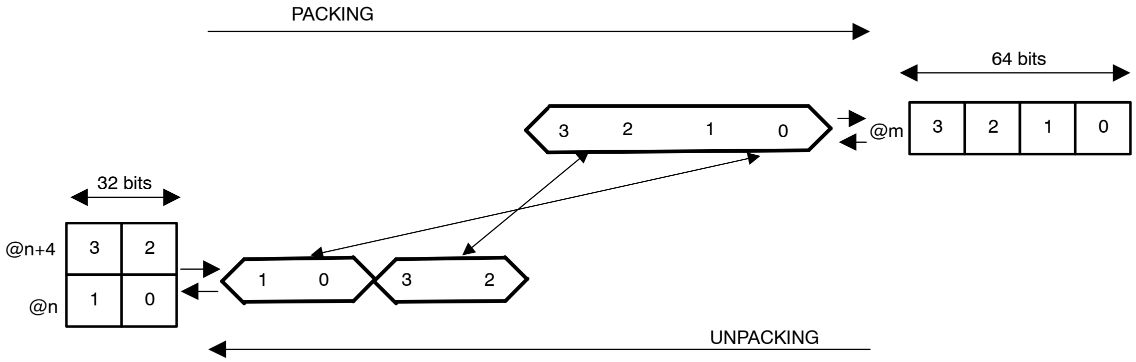
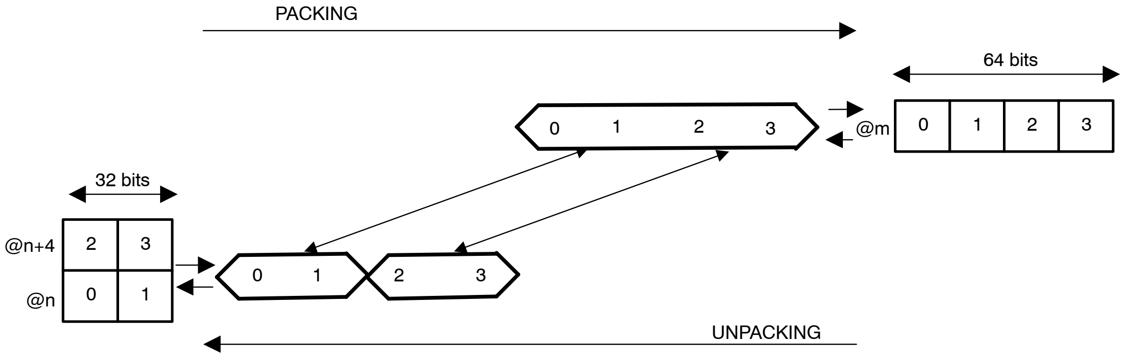


Figure 6−7 shows a 32-bit to 64-bit packing operation in big endian mode. The first 32-bit transfer, which corresponds to the lowest 32-bit address, is put into the highest part of the 64-bit word; the second transfer, which corresponds to the highest 32-bit address, is put into the lowest part of the 64-bit word. The unpacking operation is symmetric.

Figure 6−7. Big Endian Packing/Unpacking Operation



If endianness conversion is necessary, it does not take place in the interconnect. Only three kinds of module can perform endianness conversion in the OMAP2420: DMA, MMU, and IPI.

With endianness conversion, bytes are reordered in big− or little−endian style and the in-band endianness qualifier is also converted. After conversion is performed, the new data with its new in-band qualifier is delivered to the interconnect, which can manage a data-width conversion with the same rules using the new properties of the data.

6.2.5.4 Posting Effects

The interconnect handles posted writes and nonposted writes. Because data throughput performance is generally better with posted writes, this model is the most commonly used.

On a posted write, after the data, address, and all in band attributes are transferred to the L3 interconnect, a response is given as early as possible to the initiator module by the L3 interconnect. At this point, the initiator module can consider that the transfer is finished. Subsequent transfers are assumed to have no dependency on the write previously posted, and can be issued with no constraint. The L3 interconnect is responsible for completing the write operation at the target independently of the initiator module.

On a nonposted write, a response is given to the initiator module only when the target module is hit and has internally completed the transaction. Consequently, transaction latency is significantly longer than for posted writes. Nonposted writes decrease overall system performance in terms of data throughput, because the initiator is likely to be locked until the nonposted transaction completes. Nonposted writes are used when posted writes are likely to affect system stability.

Typical Example for a Non posted Write

When a module asserts an interrupt line, the interrupt line remains asserted until the interrupt processing is finished. At the end of the interrupt routine, the interrupt must be deasserted by writing to a dedicated register in the module. After this is done, the interrupt controller must be informed that the interrupt process is finished by acknowledgement of the service in another dedicated register in the interrupt controller. A new interrupt can then be treated.

If the posted write semantic is used for accesses to both the module and the interrupt controller, the module is not necessarily hit before the interrupt controller. Then the interrupt line remains asserted when the interrupt controller begins a new treatment. As a consequence, a dummy interrupt is detected by the interrupt controller and results in a system error.

To avoid this kind of system issue, use of a nonposted write semantic automatically ensures that the module is hit (interrupt line deasserted) before the acknowledging write is issued to the interrupt controller.

One important consequence of the difference between a posted and a nonposted write is error signaling.

For a posted write, if an error occurs on a transfer after a response is given to the initiator module, the error cannot be reported in-band (it is impossible for the initiator module to identify which transfer caused the error).

An error can be reported in-band on a nonposted write; the error is attached to the current transfer, making error identification much easier (see Section 6.2.6, *Error Management in the L3 Interconnect*, Section 6.4.4.6, *Error Management in the L4 Interconnect*, and Section 6.6, *Guidelines for OMAP2420 Device Interconnect Error Handling*).

6.2.5.5 Atomic Read/Write Pair Accesses

The interconnect supports atomic read/write accesses. This mechanism is generally used for semaphore handling. Only processors can generate such requests.

When an initiator issues such read/write pairs, the system guarantees that no other request from any other initiator will be inserted between the read and the write. After the read is forwarded to the target, the target is locked until the associated write is received.

Atomic read/write pairs are issued by the MPUs when executing the SWAP instruction (MPU-SS ARM1136).

Atomic read/write pairs are issued by the DSP core when executing read-modify-write instructions (such as the test-and-set instruction used for semaphore management).

These atomic read/write transaction pairs are supported by a limited number of target modules: SMS, GPMC, OCM-RAM, and DSP memory. When an atomic read/write pair access is addressed to a target module for which the instruction is not supported, an error is generated.

WARNING

For the locking mechanism to work correctly on a target core that is protected behind a firewall, access permission at the semaphore address must be given for both read and write access.

- ☐ **If read is permitted, write is not permitted. In that case, a lock is set on the target on receiving the read. The lock is not released, because of firewall rejection on the write access. This error can be detected and handled by the transaction error manager.**
- ☐ **If read is not permitted, write is permitted. In that case, an error is returned on the read access. Similarly, this error can be detected and processed by the transaction error manager.**

6.2.5.6 Aliasing Effects (LLRC Only)

An alias is used when it is possible to access the same memory area while accessing different addresses. Aliasing never occurs in SB. Aliases exist in LLRC for performance optimization; only the 5 upper bits of the address are decoded by LLRC TA.

Example:

Mapping of OCM ROM and RAM

Description:

L3 physical address: Address coded on 32 bits = 4G-byte space

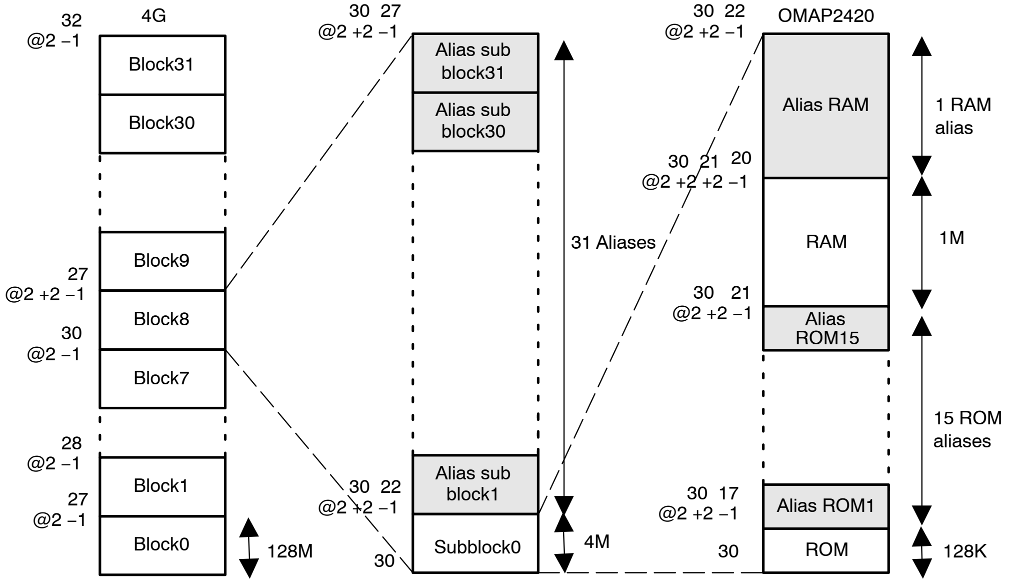
OCM ROM: Address coded on 17 bits = 128K-byte block

OCM RAM: Address coded on 20 bits = 1M-byte block

LLRC decoding: Address decoded on 5 bits = 32 blocks of 128M bytes in 4G-byte space.

Figure 6−8 shows the aliasing scheme in the LLRC.

Figure 6−8. Aliasing Scheme in LLRC



Each block can address one TA of the LLRC.

Each alias subblock is identical to subblock 0; addressing any alias in spite of block 0 has exactly the same effect as addressing subblock 0. For example, a write in alias subblock 1 at address $2^{30}+2^{22}$+offset is reported identically and instantaneously to any aliases and to subblock 0 at address 2^{30} + offset (with offset < 2^{22}).

In each subblock, there is one alias for RAM and there are 15 for ROM. Each access to any alias of ROM or RAM is strictly identical to access of ROM or RAM.

There are 31*15 = 465 aliases for ROM and 31 aliases for RAM.

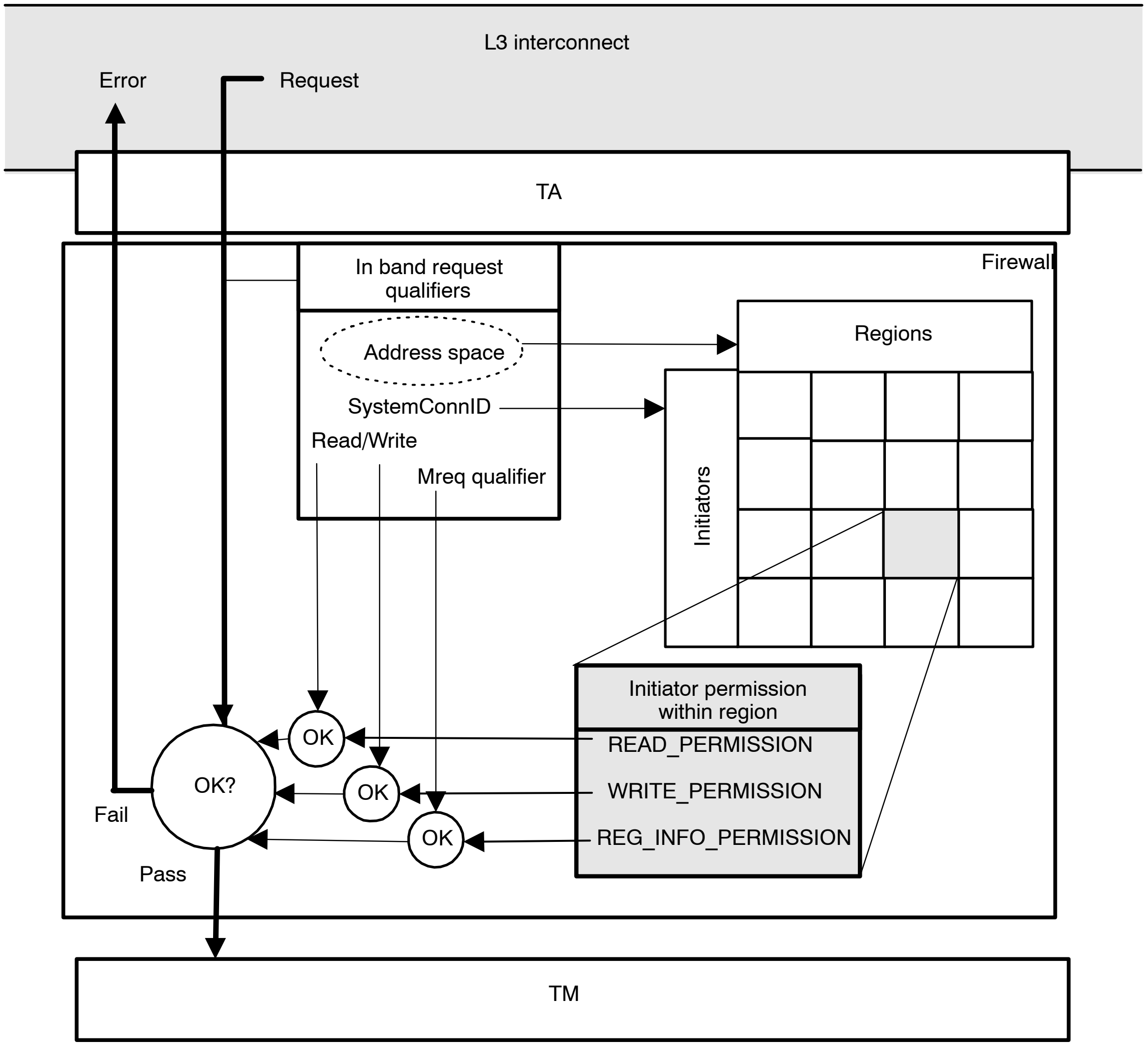
From the security firewall perspective, any alias has the same security settings as the main memory area. Any protected region created in the main memory area is automatically active in all alias areas.

6.2.5.7 L3 Firewall

The purpose of the protection mechanism is to designate protection regions in the address space of certain targets. Access to these regions is granted only for certain initiators based on transaction attributes, read or write, or in-band attributes (sent through MREQINFO).

Figure 6−9 shows the comparison mechanism performed by the L3 firewall to control access permission to a target module for each L3 request.

Figure 6−9. Firewall Comparison Mechanism



As Figure 6−9 shows, different request qualifiers are considered to determine whether this access is authorized:

❑ The address field and the address space field are used to find out which region is hit. The region ID selects one column of the firewall matrix.

❑ The region ID identifies a unique set of permission registers: Read_ Permission, Write_Permission, ReqInfo_Permission.

❑ The transaction CONNID is decoded and used to get the permission of that initiator with respect to that region.

❑ The access type (read or write) and the transaction attributes (MReq in-band qualifiers) are used to grant or not grant the access. The access is granted if the request type (read or write) is permitted (as indicated in the Read_Permission or Write_Permission registers) and if the transaction in-band qualifiers are compatible with the ReqInfo_Permission register. Otherwise, the access is denied, the request is not forwarded to the target, and an error is returned.

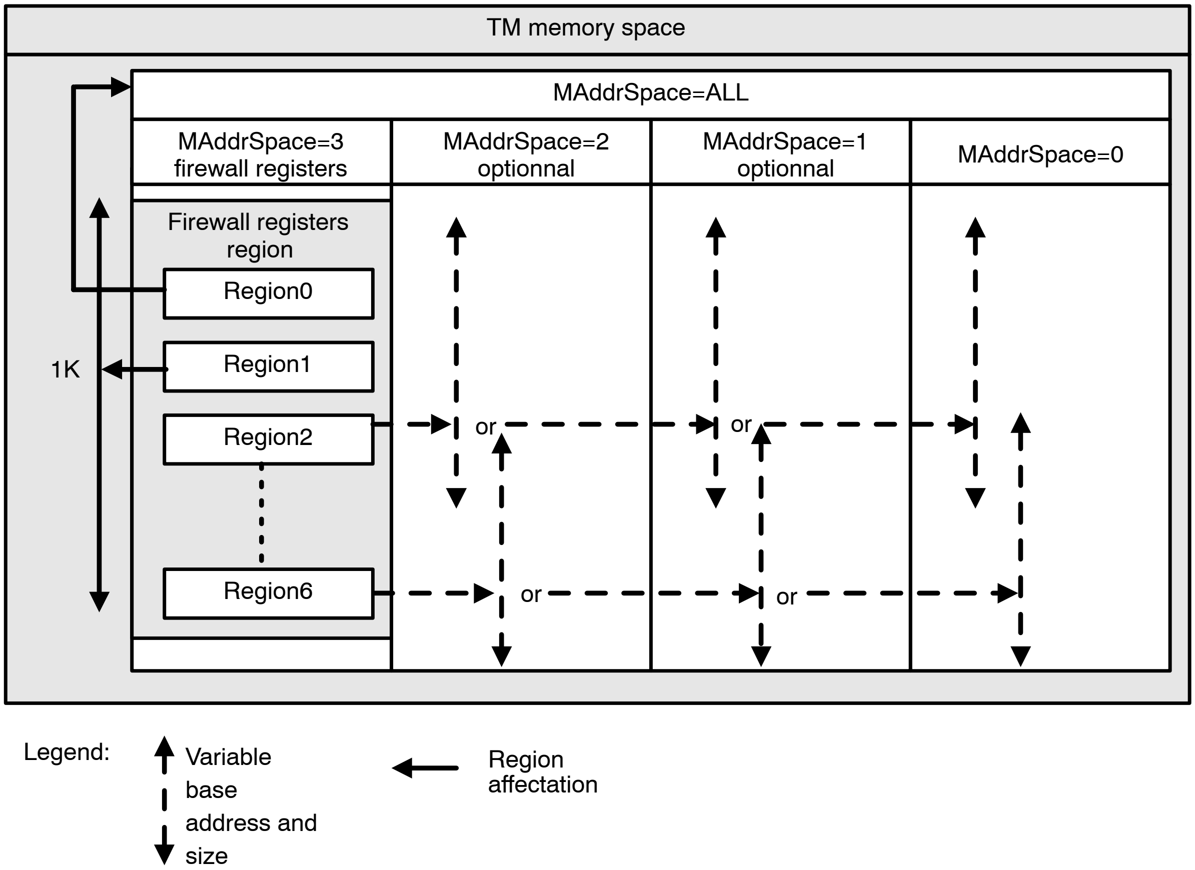
Software programs the L3 firewall registers, which fixes each initiator module permission region-by-region for a target module.

Regions Description

Protection regions are used to subdivide a given target address space. Each target can have multiple protection regions.

There are two special regions: Region0 and Region1. Region0 is the default region; Region1 is the firewall register region. All other regions are normal regions (see the *Normal Region* subsection of Section 6.2.5.7).

Figure 6−10. L3 Firewall Region Overview



For more information about the MAddrSpace, see the *Address Space Configuration* subsection of Section 6.2.5.7).

Special Regions

Region 0 is the default firewall for the entire target; this region spans the target address space. ADDR_MATCH0 register is not accessible. Its configuration corresponds to all possible addresses for the target, including all the ADDR_SPACE (no other region can be programmed like this because only one ADDR_SPACE can be programmed for one region in other regions). This region is the only region that can be overlapped. It is systematically overlapped when other regions are set.

Region 1 is the firewall register region. Its address space, size, base address, read permission, and write permission are fixed and cannot be modified. This is the only region in address space 3.

Normal Regions

All normal regions are identical and have identical capabilities.

The possible protection regions can be configured only for a memory space that is power-of-two in size and size aligned (described in Table 6−32) in the SIZE register field: ADDMATCH_N[8:4]. A given request either maps to a specific protection region or is considered to have hit the default protection region.

Table 6−32 explains the size encoding.

Table 6−32. L3 Firewall Size Parameter Definition

Possible Configuration		
Size	**Page Size**	**Base_Addr**
0x0	Region disabled	0x0000000
0x1	1K byte secure	0x0000000
		0x0000400
		0x0000800
		……
0x2	2K byte secure	0x0000000
		0x0000800
		0x0001000
		……
0xn	2^(n−1)K byte secure	
…	…	

When the size parameter is set to 0, the region is disabled.

See Table 6−33 for the summarized number of regions by L3 firewall for each protected target module. 2+ n regions means 2 special regions, (region 0 and region 1) plus regions that can be configured for user purposes.

Table 6−33. Regions by Protected Target

	GPMC	DSP Memory	OCM−ROM	OCM−RAM
Number of regions	2 + 5	2 + 2	2 + 2	2 + 5

Overlay Regions

Software ensures that user-defined regions never overlap, unless one region is given the overlay attribute in the ADDR_MATCH register. At a given time, only one region can be declared as an overlay region. Region 0 cannot be declared an overlay region.

If an address hits both an overlay and a nonoverlay region, the overlay region protections are applied. An overlay region can overlap part or all of nonoverlay regions. In this manner, exactly one set of protections applies to any given access.

Hardware behavior in the case of overlapping nonoverlay protection regions is undefined.

An overlay region must be used to mask a region that is being reprogrammed, to avoid security holes during this reconfiguration.

To change the protection settings of the region:

1) A free region must be available to be used as an overlay region.

2) Program this region as an overlay region so that its parameters match the region to be changed. The final programming must be the ADDR_MATCH register, which includes the overlay attribute and the size parameter to enable the region.

3) Disable the region to be changed by setting the size parameter to 0.

4) Set up all region control registers with the new configuration. The final programming must be the ADDR_MATCH register, which includes the size parameter to enable the region.

5) Disable the overlay region by setting its size to 0.

Use this procedure each time there is an overlap between the region as defined originally and the newly defined region.

CONNID, Read Permission, and Write Permission

CONNID corresponds to a number that identifies the initiator. For read permission and write permission, the system CONNID is used. Table 6–34 shows summarized system CONNID values for each initiator.

Table 6–34. SystemCONNID Summary Table

SystemCONNID	ID_Bit
Reserved	0
MPU [RD, WR, INSTR]	1
dDMA [RD, WR]	2
sDMA [RD, WR]	3

Table 6−34. SystemCONNID Summary Table (Continued)

SystemCONNID	**ID_Bit**
Reserved	5
Reserved	6
DSP [DATA, INSTR, MMU]	7
LCD	8
USB	9
Reserved	A
CAMERA [CAMERA, MMU]	B
Reserved	C
Reserved	D
Reserved	E

Read permission and write permission are configured using two bit vectors; one for read access permission (READPERM) and one for write access permission (WRITEPERM). A target is accessible by an initiator if the bit with the position corresponding to the CONNID number [ID_BIT] is set at 1 in the bit vector register.

Example:

In this example, the target module region is set to enable read accesses from initiator modules with the CONNID 1, 9 or 14(0xE hexadecimal). This means that the target region can be read by MPU and USB. Other read transactions to this region are rejected.

The target module region is set to enable write accesses from initiator modules with the CONNID of 1 or 3. This mean that the target region can be written by MPU or SDMA. Other write transactions to this region are rejected.

ADDR_SPACE Configuration

Some target modules have multiple address spaces (up to three: 0,1, 2). L3 registers related to TA and FW are always mapped to address space 3. Like any other address space defined in the target module, these registers can be protected by the firewall mechanism.

Mapping of functions to ADDR_SPACE is described in Table 6−35 and must be used for firewall programming.

Table 6−35. Address Spaces for Protected Targets

Can protect regions in target space:	GPMC	DSP Memory	OCM−ROM	OCM−RAM
Using firewall address space 0	GPMC (address space 0)	DSP mem (address space 0 = dmemory)	OCM−ROM	OCM−RAM
Using firewall address space 1	GPMC registers (address space 1)	DSP mem (address space 1 = dMMU)		
Using firewall address space 2		DSP mem (address space 2 = dIPI)		
Using firewall address space 3: TA firewall registers	GPMC firewall registers GPMC TA error log registers SMS TA error log registers	DSP mem firewall registers	OCM−RO firewall registers OCM−ROM TA error−log registers	OCM−RAM firewall registers OCM−RAM TA error−log registers

REQ_INFO_PERMISSION Configuration

The firewall comparison mechanism enables access of a protected target when a correct combination of three MREQ in-band parameters (MREQDebug and MREQType) is transmitted. The MREQ combination is a fixed 3-bit pattern corresponding to a combination of the parameters.

Valid MREQ combinations are defined for each L3 firewall based on the REQ_INFO_PERMISSION value, which is programmed by software.

REQ_INFO_PERMISSION encoding covers all possible MREQ combinations. Each valid MREQ combination corresponds to a ReqBit that must be set to 1 in the REQ_INFO_PERMISSION field to enable the MREQ combination for the firewall.

REQ_INFO_PERMISSION is correctly set when its value corresponds to the bit-to-bit addition of all the valid allowed bit vectors.

Table 6−36 summarizes REQ_INFO_PERMISSION configuration.

Table 6−36. REQ_INFO_PERMISSION Setting

MREQDebug	MREQType[0]	ReqBit	REQ_INFO_PERMISSION[ReqBIT]	
0 normal/ 1 debug	0 data/1 instruction		Bit vector when allowed	Bit vector when forbidden
0	0	0	0b00000001	0b00000000
0	1	1	0b00000010	0b00000000

Table 6−36. REQ_INFO_PERMISSION Setting (Continued)

MREQDebug	**MREQType[0]**	**ReqBit**	**REQ_INFO_PERMISSION[ReqBIT]**	
0 normal/ 1 debug	**0 data/1 instruction**		**Bit vector when allowed**	**Bit vector when forbidden**
1	0	2	0b00000100	0b00000000
1	1	3	0b00001000	0b00000000
0	**0**	**4**	0b00010000	0b00000000
0	**1**	**5**	0b00100000	0b00000000
1	0	6	0b01000000	0b00000000
1	1	7	0b10000000	0b00000000
	+			
Example of Result REQ_INFO_PERMISSION value			0x10 = 0b00010000 **0x30 = 0b00110000** 0x5F = 0b01011111	

Note: Bold values (ReqBit 4 and ReqBit 5) indicate two MREQ combinations valid for the firewall (MREQ = 100 and MREQ = 101). Consequently, the REQ_INFO_PERMISSION value is a bit-to-bit addition between bit vector 4 and bit vector 5: 0b00010000 + 0b00100000 = 0b00110000.

Depending on the target, the L3 firewall does not have the same capabilities. The DSP firewall does not integrate all the MREQ in-band parameters, but only a subset: MREQDebug is considered, while MREQType is ignored. If this is the case, no comparison mechanism is performed inside the firewall between the MREQType transaction attribute and the REQ_INFO_PERMISSION value. Even bits of REQ_INFO_PERMISSION have no effect on firewall behavior.

Table 6−37. MREQ In Band Qualifiers Considered by Firewall for Each Target

	GPMC	**DSP Memory**	**OCM−ROM**	**OCM−RAM**
MREQ in band parameters comparison performed on:	MREQDebug MREQSecure MREQType	MREQ Debug	MREQ Debug	MREQ Debug MREQ

L3 Firewall Registers Overview

Table 6−38. L3 Firewall Registers Overview

Register Name	Register Offset	Register Field Name	Field Modifiability	Field Reset Value	Parameter Comments
Region 0 **The default setting can be changed only with correct access according to region 1.**					
ADD MATCH0	Embedded In L3; none accessible	ADDR_ SPACE	Hard coded	All	Corresponds to all target memory space
		SIZE	Hard coded	Maximum	Corresponds to all target memory space
		OVERLAY	Hard coded	0	Allows the region to overlap another non-overlay region
		BASE_ ADDR	Hard coded	Target memory space start address	Target−dependent
REQINFO PERM0	0x48	REQ_ INFO [31 :0]	Yes	0xFF	All ReqInfo encodings allowed (debuggable, nonsecure, and all access types)
READ PERM0	0x50	READ_ PERMISSION [15:0]	Yes	0xFFFF	All initiators have read permission
WRITE PERM0	0x58	WRITE_ PERMISSION [15:0]	Yes	0xFFFF	All initiators have write permission
Region 1 **This is the target register memory space region, by default configured as MPU−only access.**					
ADD MATCH1	0x60	ADDR_ SPACE [1:0]	Hard coded	3	All target embedded registers always on MAddrSpace = 3
		SIZE [8:4]	Hard coded	1	All target registers are meant to fit in a 1k page.
		OVER LAY [9]	Hard coded	0	Can be bypassed by others firewall
		BASE_ ADDR [31:10]	Hard coded	target register memory space address	Beware that BASE_ADRR must account for the ADDR_SPACE parameter
REQINFO PERM1	0x68	REQ_ INFO [31 :0]	Yes	FF	All ReqInfo encodings are allowed (debuggable, nonsecure, and all access types)
READ PERM1	0x70	READ_ PERMISSION [15:0]	Hard coded	0x2	Read access reserved to initiatorID [1] = ARM11
WRITE PERM1	0x78	WRITE_ PERMISSION [15:0]	Hard coded	0x2	Write access reserved to initiatorID [1] = ARM11

Table 6−38. L3 Firewall Registers Overview (Continued)

Register Name	Register Offset	Register Field Name	Field Modifiability	Field Reset Value	Parameter Comments
Region 2 to be configured if needed					
ADD MATCH2	0x80	ADDR_SPACE [1:0]	Y	0	
		SIZE [8:4]	Y	0	The regions are power−of−two in size and size aligned with Base_Addr reference When Size = 0x0, the firewall is deactivated.
		OVERLAY [9]	Y	0	
		BASE_ ADDR [31:10]	Y	0	
REQINFO PERM2	0x88	REQ_ INFO [31 :0]	Y	FF	
READPERM2	0x90	READ_ PERMISSION [15:0]	Y	FFFF	
WRITE PERM2	0x98	WRITE_ PERMISSION [15:0]	Y	FFFF	
Up to 3 more regions, depending on the target					

L3 Firewall Error Logging Registers

Table 6−39. L3 Firewall Error Logging Registers

Register Name	Register Offset	Register Field Name	Field Modifiability	Field Reset Value	Parameters Comments
Error Log					
ERRLOG	0x20	CMD [2:0]	Read only	0x0	This field logs the OCP command code (MCmd) of the request that caused the protection violation.
		REGION [7:4]	Read only	0x0	This field logs the region number of the region that matched and denied access to the request that caused the protection violation.

Table 6−39. L3 Firewall Error Logging Registers (Continued)

Register Name	Register Offset	Register Field Name	Field Modifiability	Field Reset Value	Parameters Comments
ERRLOG	0x20	INITIATOR_ID [13:8]	Read only	0x0	This field logs the SystemCOnnID of the request that caused the protection violation and could have been used by the firewall to deny access. **Initiator ID** **ID_Bit** Reserved 0 ARM11 [RD, WR, INSTR] 1 dDMA [RD, WR] 2 sDMA [RD, WR] 3 Reserved 4 Reserved 5 . . . . Reserved 6 DSP [DATA, INSTR, MMU] 7 LCD 8 USB 9 . . . . Reserved A CAMERA [CAMERA, MMU] B Reserved C Reserved D Reserved E F
		REQ_INFO [19:16]	Read only	0x0	This field logs the MREQInfo bits of the request that caused the protection violation and could have been used by the firewall to deny access.
		CODE [27 :24]	Read/Write	0x0	0b0000 No violation 0b0011 A violation occurred.
		MULT[31]	Read/Write	0x0	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 writing a nonzero value to the CODE field and 1 to the MULT bit.

6.2.5.8 Exported Reset Values for OCM RAM Firewall

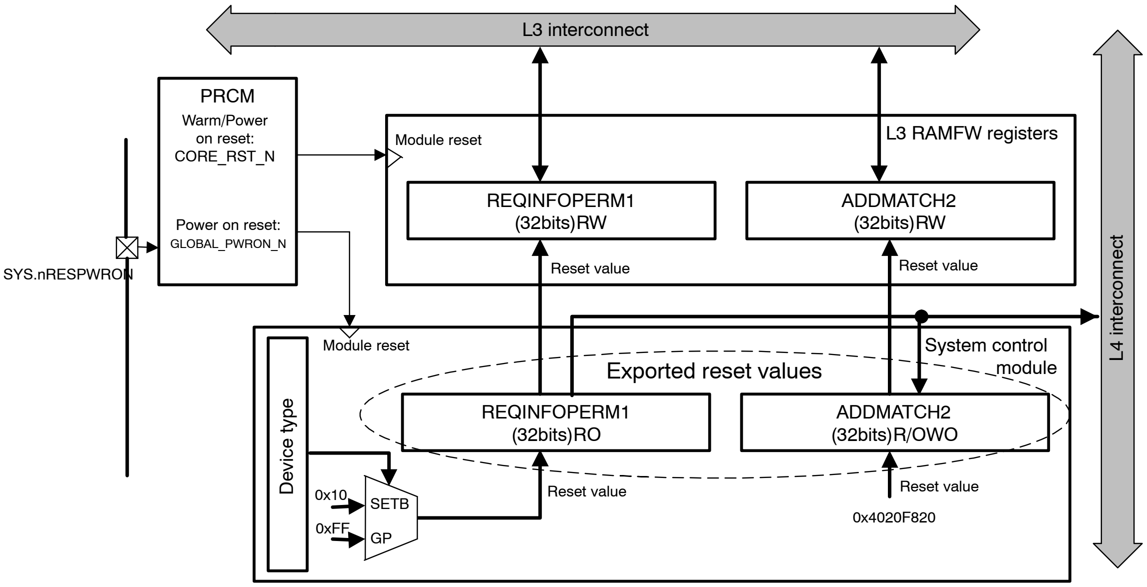
Two of the OCM RAM firewall register reset values are exported in the system control module (SCM). Exported means that the L3 interconnect firewall register takes the value contained in the SCM register exported reset value at the release of the L3 interconnect reset (CORE_RST).

Once the L3 interconnect reset is released, there is no difference from other registers. The register values of RAMFW registers can be overwritten by software by accessing the L3 interconnect register address. A modification on an exported reset value in the SCN does not affect the value of RAMFW registers, because there is no warm reset.

The REQINFOPERM1 exported reset value is a read-only value that depends on the chip device type: 0xFF if for a general-purpose device, otherwise, 0x10 (test, or bad device).

The ADDMATCH2 exported reset value is read/one-write-only (R/OWO); only the first write access can modify its value. Write access is taken into account even if the register value is unchanged; subsequent write attempts are ignored, and no more write actions are effective until the next power-on reset.

Figure 6−11. RAM Firewall Exported Reset Values



Consequently, the ADDMATCH2 reset value can be changed by secure software if it does not correspond to system requirements. This new reset value must be applied to the SCM register before a CORE_RST deassertion. Once changed, this new reset value is applied to the RAMFW ADDMATCH2 register when a WARM_RESET is performed.

6.2.5.9 SMS Firewall

The SMS firewall is similar to the L3 firewall; however, it has a limitation on attributes and number of regions:

- Read/write access permissions are allocated to initiators on a per−region basis.
- The memory regions are programmable, using a start address and an end address defined with a 64K-byte granularity and aligned on 64K-byte addresses.
- There is an implicit default region that is not configurable. Any access that does not hit one of the four user-defined regions is allowed. This is a public region.
- Up to four distinct regions can be defined; the software must ensure that they do not overlap.

Note:

Only differences relative to the L3 firewall are detailed in this section.

Security Related Registers

Security-related registers are programmable only by a secure transaction.

A region can be given the specific access permissions, depending on whether the access is a read or a write, and depending on the in-band request qualifiers:

- The read permission is initiator-based and controlled using the SMS_RG_RDPERMi register.
- The write permission is initiator-based and controlled using the SMS_RG_WRPERMi register.
- The MREQ in-band parameter considered is MReqDebug. It is possible to specify for each REQ_INFO_PERMISSION pattern whether the access is accepted or not (there is one valid bit for each REQ_INFO_PERMISSION pattern).
- REQ_INFO_PERMISSION is controlled using the region attributes register SMS_RG_ATTi.

SMS_RG_ATT Configuration: REQ_INFO_PERMISSION

This mechanism is identical to REQ_INFO_PERMISSION in the L3 firewall except that only one MREQ parameter is used: MREQ_DEBUG (MREQ_TYPE is discarded). Table 6−40 shows the REQ_INFO_PERMISSION setting for the SMS firewall. Table 6−41 summarizes the SMS firewall permission−setting registers.

Table 6−40. REQ_INFO_PERMISSION Setting for SMS Firewall

MREQ Debug	**ReqBit**	**REQ_INFO_PERMISSION[ReqBIT]**	
0 normal/1 debug		Bit vector when allowed	Bit vector when forbidden
0	0	0b0001	0b0000
1	1	0b0010	0b0000
+	+	0x0 = 0b0000 **0x30 = 0b1100** 0xF = 0b1111	
Example of Result REQ_INFO_PERMISSION value			

Note: Bold values (ReqBit 2 and ReqBit 3) indicate two MREQ combinations valid for the firewall (MREQ = 102 and MREQ = 103).

SMS Firewall Registers Overview

Table 6−41. SMS Firewall Permission Setting Registers Summary

Register Name	**Register Offset**	**Register Field Name**	**Field Reset Value**	**Parameters Comments**
				Region 0
SMS_RG_ START0 [30:0]	0x40	StartAddress	0x0	Region 0 start address aligned on 64K-bytes boundary (15 bits wide); [15:0] must be 0s
SMS_RG_ END0 [30:0]	0x44	EndAddress	0x0	Region 0 end address aligned on 64K-bytes boundary (15 bits wide); [15:0] must be 0s
SMS_RG_ ATT0[7:4]	0x48	ReqPerm	0xF	ReqBit [0] = MReqDebugs ReqBit [1] = MReqSecure RegPerm[ReqBit] = 1 Pass RegPerm [ReqBit] = 0 Fail If ReqPerm = 0xFF : All ReqBits are allowed. If ReqPerm = 0x00 : All ReqBits are forbidden.
SMS_RG_ WRPERM0 [15:0]	0x80	WRITE_ PERMISSION	0x0	**Comparison** **Initiator ID** **ID_Bit** Reserved 0 ARM11 [RD, WR, INSTR] 1 dDMA [RD, WR] 2 sDMA [RD, WR] 3 Reserved 4 Reserved 5 Reserved 6 DSP [DATA, INSTR, MMU] 7 LCD 8 USB 9 Reserved A CAMERA [CAMERA, MMU] B Reserved C Reserved D Reserved E Reserved F WRITE_PERMISSION [ID_Bit] = 1 PassWRITE_PERMISSION [ID_Bit] = 0 Fail

Table 6–41. SMS Firewall Permission Setting Registers Summary (Continued)

Register Name	Register Offset	Register Field Name	Field Reset Value	Parameters Comments
SMS_RG_ RDPERM0 [15:0]	0x84	READ_ PERMISSION	0x0	**Comparison** **Initiator ID** **ID_Bit** Reserved 0 ARM11 [RD, WR, INSTR] 1 dDMA [RD, WR] 2 sDMA [RD, WR] 3 Reserved 4 Reserved 5 . . . . Reserved 6 DSP [DATA, INSTR, MMU] 7 LCD 8 USB 9 . . . . Reserved A CAMERA [CAMERA, MMU] B Reserved C Reserved D . . . . Reserved E Reserved F READ_PERMISSION [ID_Bit] = 1 Pass READ_PERMISSION [ID_Bit] = 0 Fail
Region 1 –3				
Region 1 to 3 configuration is the same as region 0. For the offset descriptions, see Section 6.3.1.1, *LLRC Register Descriptions*.				

When a security violation occurs, an in-band error is sent back in the OCP transaction.

6.2.6 Error Management in the L3 Interconnect

The L3 interconnect provides mechanisms for handling either internally detected errors or errors reported by modules attached to the L3 target ports (LLRC or SB). Specific hardware distributed in the agents facilitates error detection and error logging. Logging registers can then be used by the error–management software to identify error sources and (optionally) to recover from an error state.

When possible, errors are reported in-band to the initiator that owns the faulty transaction. When this is not possible (for example, on posted write operations), errors are reported out-of-band.

Additionally, in-band errors detected in the SB part can be redundantly reported using out-of-band signaling in a programmable fashion.

Out-of-band error information from the LLRC is merged with out-of-band error information from the SB before being provided as a unified interrupt request to the MPU subsystem.

There is no time-out detection in the LLRC, whereas the SB part of the L3 interconnect features fully programmable time-out capabilities. Time-outs in the L4 are managed by the L4 internal mechanisms, and are described in the *Time-out* subsection of Section 6.4.4.6.

6.2.6.1 SB Error Management: Controls and Logging Information

The SB includes error-management-related resources distributed across the IM and TM agents:

- Controls available in the IM agents
- Status available in the IM agents
- Controls available in the TM agents
- Status available in the TM agents

Figure 6−12 shows the error−reporting architecture available in all IM and TM agents.

In this section, control and status bits are referenced using a REGISTER_NAME(bit(s)) :BITFIELD_NAME notation.

IM Agents—Error Management Control

An IM agent includes several control bit fields related to error management, all regrouped in the SBIMCONFIG_L(_H) registers. These bit fields are described in this section. A local IM software reset bit is also provided.

- SBIMCONFIG_L(6:4): SBREQTIMEOUT

  This field controls the request time-out for transfers initiated by the attached initiator core. The time-out occurs after an interval of between one and two times the time-out period, as shown in Table 6−42.

Table 6−42. L3 SB Time-Out: Initiator Agent Settings

SBREQTIMEOUT [2:0]	Time-Out Period
0 0 0	Time-out disabled
0 0 1	2^6 = 64 L3 clock cycles
0 1 0	2^8 = 256 L3 clock cycles
0 1 1	2^{10} = 1024 L3 clock cycles
1 0 0	2^{12} = 4096 L3 clock cycles
1 0 1	2^{14} = 16384 L3 clock cycles
1 1 0	2^{16} = 65536 L3 clock cycles
1 1 1	2^{18} = 262144 L3 clock cycles

If an IM fails to complete a request with a TM within the time-out interval defined, the associated TM returns an ERR response. This action clears the request in the IM, allowing later requests (for example, for error handling) to proceed. The IM also logs and reports the time-out error either in-band or out-of-band (see the next control description for the SBIMCONFIG_H(4): SBTOERRMODE bit).

The request is lost from the system and cannot be recovered. The time-out event is simultaneously logged into the associated TM agent; see the SBTMSTATE_H(5): SBTMTIMEOUT description later in this section.

When a request time-out occurs and the IM clears its internal state, an additional state relating to the request can remain in a TM. The system software clears this state through a reset of the affected TM and attached target core by writing 1 to the SBCLRESET field in the TM. If the final system target was an L4 target, additional cleanup is also required in the L4 interconnect.

At reset, the SBREQTIMEOUT field is set to 7 for all initiator modules; that is, time-out is enabled with the longest possible value.

- SBIMCONFIG_H(4): SBTOERRMODE

  When a time-out condition is detected in an initiator agent, an ERR response is auto-generated by the agent to the initiator module. Optionally, if this bit is set to 1, the event is simultaneously reported sideband on the SB aggregated error signal and can, therefore, generate an interrupt on the MPU.

  This option is enabled at reset for all IM agents and is typically used if the initiator core has no interrupt support when detecting a transaction error, or if this capability is not used (that is, a single interrupt line is preferred).

- SBIMCONFIG_H(2): SBIBERRMODE

  When an in-band error response is received by an initiator agent, this in-band response is forwarded to the initiator module. Optionally, if this bit is set to 1, the event is simultaneously reported sideband on the SB aggregated error signal and can therefore generate an interrupt on the MPU.

  This option is enabled at reset except for the MPU2SB bridge. This option is typically used if the initiator core has no interrupt support when detecting a transaction error, or if this capability is not used (that is, a single interrupt line is preferred). On the MPU2SB bridge, the in-band error automatically generates a bus−error exception on the MPU. If the option is activated on the MPU2SB port, the L3 interconnect exception interrupt is also issued, resulting in dual reporting that must be handled correctly by the error-management software.

- SBTMSTATE_L(0): SBCLRESET (despite its name, which includes a reference to a TM, this register is effectively part of any IM agent): Global reset control for the IM agent. Must be set to 1 for at least 16 L3 clock cycles, and then deasserted under software control. Not available for the MPU2SB agent to avoid a deadlock condition of the L3 interconnect, because the SB overall control is expected to come from the MPU.

IM Agents—Error−Management Status

Additionally, an IM agent includes several status bitfields related to error management. These status bits can be read and cleared under software control by writing 0. Distributed across several IM registers (SBIMSTATE, SBIMERRLOG, SBIMERRLOGA), they are described in the following sections.

- SBIMSTATE(18): SBTOERR

When a time-out occurs, this bit logs the time-out condition, and the corresponding request is discarded.

- ☐ SBIMSTATE(17): SBIBERR

  This bit is set on multiple error conditions, to indicate that an in-band error response was received from a target, that an illegal command was sent, or that a request hit a hole in the SB address map.

- ☐ SBIMERRLOG(31): SBIMERRLOGMULT

  While SBIMERRCODE is not 0 (see the next bitfield description); that is, a previous error is already detected and recorded, this bit is turned on when an error is detected.

  Once set, this bit can be cleared only by writing 0 to it in conjunction with writing 0 to SBERRCODE or by resetting the agent.

  When multiple errors occur, all logged information corresponds to the first error that was detected.

- ☐ SBIMERRLOG(27:24): SBIMERRCODE

  This field provides information about the type of error that was encountered. The error code values appear in the table below:

SBIMERRCODE [3:0][1]	**Error Type**
0 0 0 0	No error
0 0 0 1	No target (address hole) or request not supported by target
0 0 1 0	Time-out condition detected
0 0 1 1	In-band error received from target

Notes: 1) All other encodings are reserved.

Once set by hardware, this field can be cleared only by writing 0 to it in conjunction with writing 0 to SBIMERRLOGMULT (see the previous bitfield description) or by resetting the agent. On clearing these fields, the value of the SBIMERRLOG and SBIMERRLOGA fields is undefined.

- ☐ SBIMERRLOG(15:8): SBIMERRCONNID

  This 8-bit field specifies the SBConnID of the request producing the error. Table 6–31 shows how to translate the information back into SYSTEMCONNID (only the 5 LSBs must be considered) and in-band information (3 MSBs).

- ☐ SBIMERRLOG(2:0): SBIMERRCMD

  This 3-bit field specifies the type of the request producing the error.

sbCmd[2:0]	**Transfer Type**	**Comment**
0 0 0	Reserved	Reserved
0 0 1	WritePost	Posted write
0 1 0	Read	Regular read

sbCmd[2:0]	Transfer Type	Comment
0 1 1	ReadEx	Exclusive read
1 0 0	ReadRtry	SB internal opcode, functionally a regular read
1 0 1	WriteNonPost	Nonposted write
1 1 0	WriteRtry	SB internal opcode, functionally either a posted or nonposted write
1 1 1	Reserved	Reserved

- SBIMERRLOGA(31:0): SBIMERRADDR

  This register provides the 32-bit address of the request producing the error.

  - On single accesses, this register provides the 32-bit address of the request producing the error.
  - On burst accesses, this register provides the 32-bit address of the request producing the error in the burst. Therefore, the address logged can be any address of the burst producing the error.

TM Agents—Error-Management Controls

There is no user-programmable control field for error management in SB TM agents. Two bitfields in the SBTMCONFIG registers are hardwired as described below. A local TM software reset bit is also provided.

- SBTMCONFIG(12): SBOBERRMODE

  This bit is hardwired to 1: Any ERR response received on the target port that cannot be forwarded in-band to the originator TM because the request was a posted write is logged and aggregated on the SB internal error signal. Therefore, such errors are visible on the MPU2SB TM error signal and can generate an interrupt on the MPU.

- SBTMCONFIG(8): SBSERRMODE

  This bit is hardwired to 1: An out-of-band error signal asserted by a target module is automatically aggregated on the internal SB error signal. The error is then visible on the MPU2SB TM error signal and can generate an interrupt on the MPU. In practice, this concerns only the L4 port, which is the only SB target with out-of-band error-reporting capability.

- SBTMSTATE_L(0): SBCLRESET

  Global reset control for the TM agent. Must be set to 1 for at least 16 L3 clock cycles, then deasserted under software control.

TM Agents—Error-Management Status

TM agents also include several status bitfields related to error management. These status bits can be read and cleared under software control by writing 0. Distributed across several registers (SBTMSTATE_H, SBTMSTATE_L, SBTMERRLOG, SBTMERRLOGA), they are described in the following sections.

- SBTMSTATE_H(5).SBTMTIMEOUT

  All TMs have time-out support capability, working with time-out detection logic located in the IMs. This bit is set when a time-out condition occurs for a particular request with that TM.

  The TM then returns a time-out ERR response to the IM, and the TM enters a time-out error state (indicated by the SBTMTIMEOUT status bit). While SBTMTIMEOUT is set, the TM responds with ERR to all nonregister requests targeting it (that is, addressed to the target module), but continues to respond normally to TA register requests.

  SBTMTIMEOUT can be cleared only through a reset of the TM (using SBTMSTATE_L.SBCLRESET). Transfers for which a time-out occurs are lost from the system and cannot be recovered.

- SBTMSTATE_H(4): SBERRRESP

  This bit is set whenever an ERR response is received at the TM but cannot be reported in-band because the corresponding request is already posted (posted write only). Once set by hardware, this bit can be cleared by writing 0 to it.

- SBTMSTATE_H(0): SBSERR

  This bit is set when a positive edge is detected on the out-of-band error flagged by the target module. The only target that can produce such a signal is the L4 interconnect; therefore, this bit is always 0 for all other TMs. Once set by the hardware, this bit can be cleared only by writing 0 to it.

- SBTMERRLOG(31): SBTMERRLOGMULT

  While SBTMERRCODE is not zero, this bit is turned on when an error is detected and signals multiple errors. Once set by hardware, this bit can be cleared only by writing 0 to it in conjunction with writing 0 to SBTMERRCODE or by resetting the TM.

- SBTMERRLOG(27:24): SBTMERRCODE

  This bit field identifies the error type. The error code values appear in the table below:

SBTMERRCODE [3:0][2]	**Error Type**
0 0 0 0	No error
0 1 0 0	Posted write error (in-band error reporting was not possible)

2) All other encodings are reserved

  Once set by hardware, this field can be cleared only by writing 0 to it in conjunction with writing 0 to SBTMERRLOGMULT or by resetting the TM.

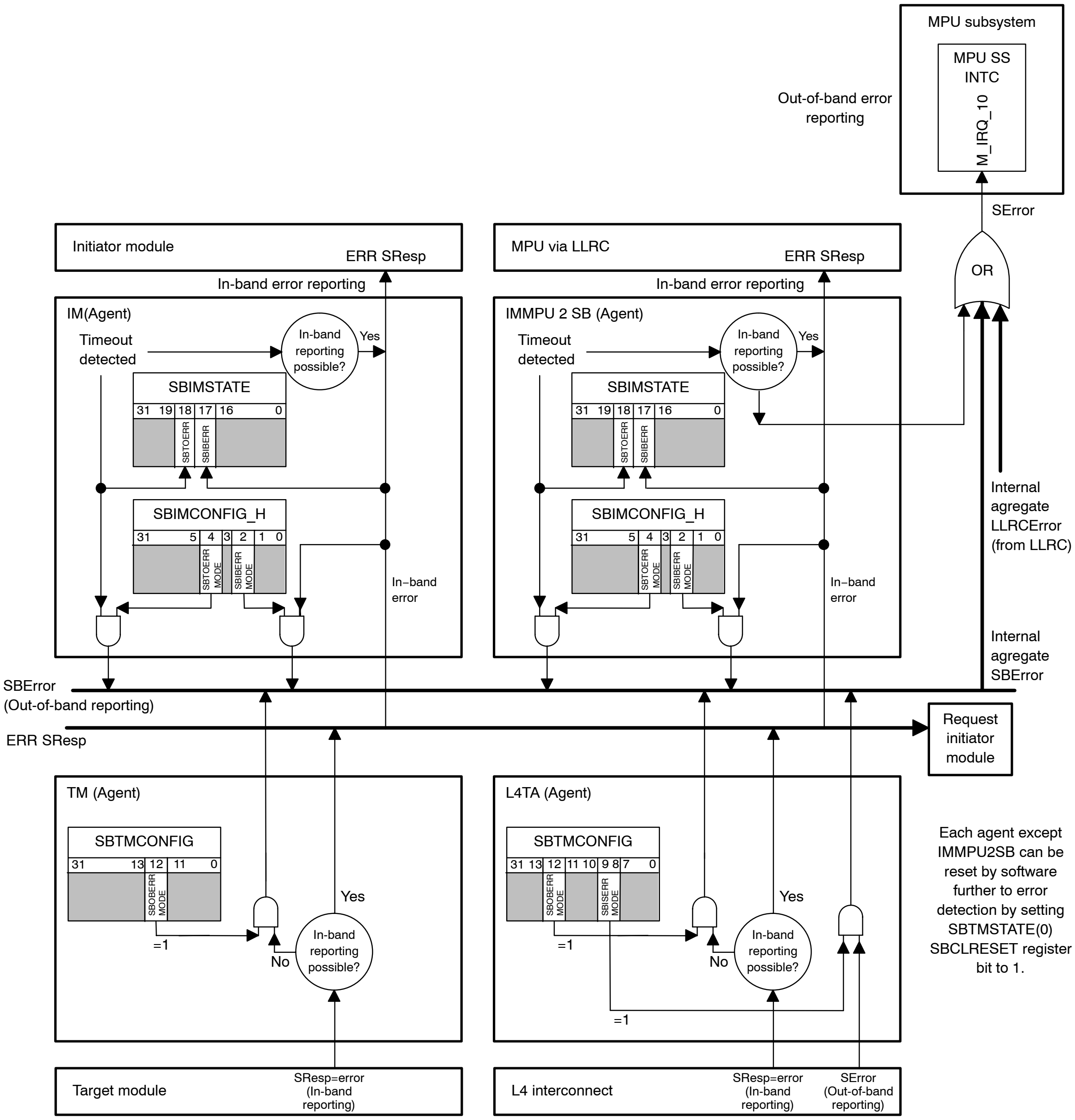
- SBTMERRLOG(15:8): SBTMERRCONNID

  This 8-bit field specifies the SBConnID of the request producing the error. Table 6−31 shows how to translate the information back into SystemConnID (only the 5 LSBs must be considered) and in-band information (3 MSBs).

- SBTMERRLOGA(31:0): SBTMERRADDR

  This register provides the 32-bit address of the request producing the error.

Figure 6−12. SB Error Management



6.2.6.2 LLRC Error Management: Logging Information

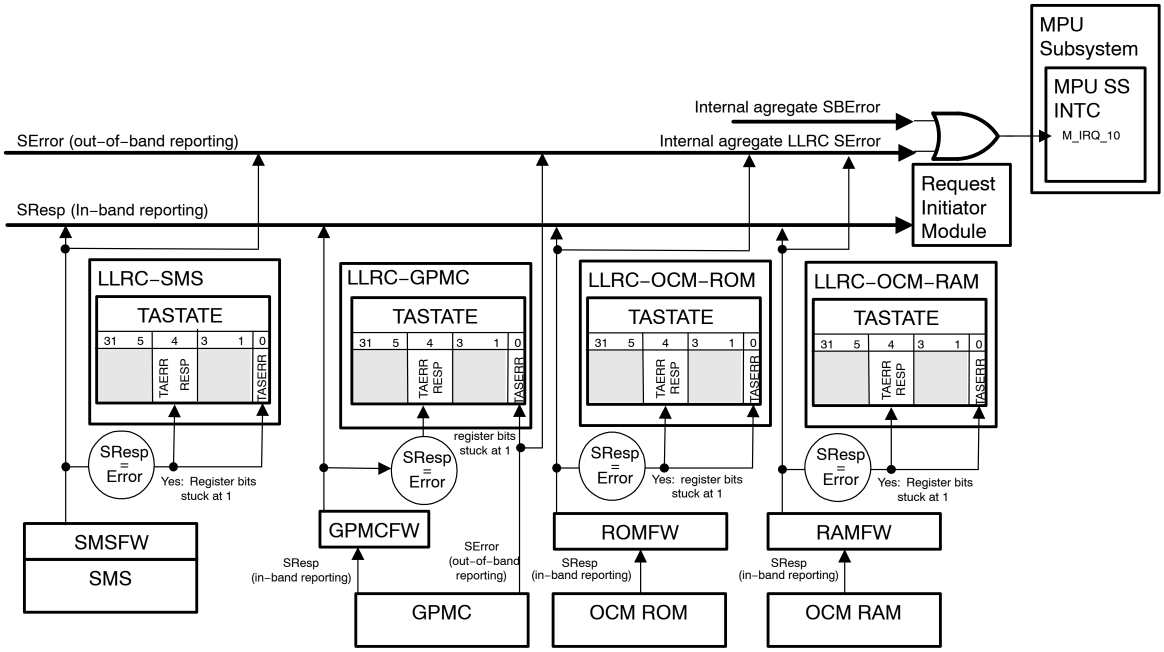
Except in the firewall logic, there is no error detection inside the LLRC. Security-specific error registers are not described in this section. See Section 6.2.5.7, *L3 Firewall*.

The out-of-band error signal from the SB is ORed with the LLRC error signal. The merged signal is forwarded to the MPU subsystem interrupt controller.

Errors reported by the attached target cores or their L3 firewall are logged unconditionally in the LLRC logging registers, as shown in Figure 6−13. There is no software programmability in the LLRC to handle error management.

The logging registers are in the OCM RAM TA, the OCM ROM TA, and the GPMC TA.

Figure 6−13. LLRC Error Logging Registers



Any in-band ERR response received from the target module or from the firewall logic is logged simultaneously in the TAERRRESP and the TASERR bits. TASERR bits are aggregated on the LLRC error sideband signal. The error response is also forwarded in-band to the initiator, if possible (always, except for some posted writes).

The GPMC is the only module that can assert a sideband error signal (SError). When a rising edge is detected on that signal, the event is logged in the TASERR bit. Because the GPMC TASERR bit is aggregated on the LLRC error signal, an interrupt can be generated to the MPU.

A set of three registers is used for logging errors: TASTATE, TAERRLOG, and TAERRLOGA. TAERRLOG and TAERRLOGA are only for the OCM RAM and OCM ROM, as the SMS/SDRC and GPMC cores already include their own extended error logging registers.

The TAERRLOG and TAERRLOGA registers record information about the first error occurrence. If a second error occurs before this register is cleared by software, the TAERRLOGMULT bit is set, but other fields are not overwritten.

Bitfields are referenced here using a REGISTER_NAME(bit(s)): BITFIELD_NAME notation.

- TASTATE(4): TAERRRESP

  This field is set when an in-band ERR response is received. If possible, for nonposted transactions, the in-band ERR response is forwarded to the initiator. Once set by hardware, this bit can be cleared by writing 0 to it.

- ☐ TASTATE(0): TASERR

  This field is set either when an in-band ERR response is received, or when a positive edge is detected on the target module sideband error signal (GPMC only). Once set by hardware, this bit can be cleared only by writing 0 to it.

- ☐ TAERRLOG(31): TAERRLOGMULT

  While TAERRCODE is not zero, this bit is turned on when an error is detected. Once set by hardware, this bit can be cleared only by writing 0 to it in conjunction with writing 0 to TAERRCODE.

- ☐ TAERRLOG(27:24): TAERRCODE

  The error code values appear in the table below:

TAERRCODE [3:0][1]	Error Type
0 0 0 0	No error
0 1 0 0	ERR response received

1) All other encodings are reserved.

Once set by hardware, this field can be cleared only by writing 0 to it in conjunction with writing 0 to TAERRLOGMULT.

- ☐ TAERRLOG(11:8): TAERRCONNID

  Specifies the SystemConnID associated with the request producing the error

- ☐ TAERRLOG(2:0): TAERRCMD

  Specifies the access type (OCP command) for the request producing the error, as indicated in the table below:

TAERRCMD[2:0]	Transfer Type	Comment
0 0 0	Reserved	Reserved
0 0 1	WritePost	Posted write
0 1 0	Read	Regular read
0 1 1	ReadEx	Exclusive read
1 0 0	Reserved	Reserved
1 0 1	WriteNonPost	Nonposted write
1 1 0	Reserved	Reserved
1 1 1	Reserved	Reserved

- ☐ TAERRLOGA(31:0): TAERRADDR

  Provides the address of the request producing the error

6.2.7 Clocking, Reset, and Power-Management Scheme

6.2.7.1 Clocking

Only one clock is used in the L3 interconnect: Core_L3_ICLK

Clock	Frequency	Name	Comments
Interface	Up to 165 MHz	Core_L3_ICLK	Source, control, and gating handled by PRCM

6.2.7.2 Reset

There are no software resets for the L3 interconnect, only hardware reset capabilities.

Hardware Reset

The L3 interconnect receives a reset signal, CORE_RST, from the PRCM. This is the reset signal to the CORE power domain. See Chapter 5, *Power, Reset, and Clock Management.*

Interconnect	Reset Domain
L3 interconnect	CORE_RST

6.2.7.3 Power Management

Power Domain

The L3 interconnect connects to the CORE power domain, which is capable of dynamically switching between low voltage and high voltage.

Interconnect	Power Domain
L3 interconnect	CORE

Power Saving

The L3 interconnect can automatically gate the clock going to the SB when a programmable number of inactivity cycles are observed. This number of inactivity cycles can take any value between 0 and 65535. The register field SBCLCONTROL is in the IMMPU2SB register: SBTMSTATE_L[31:16] at address: 0x6800 0D98 and only 8 bits are used: SBTMSTATE_L[23:16].

6.2.8 Programming Guide

6.2.8.1 Initialization

At the release of power-on reset, L3 firewall default configuration enables all accesses to target modules, except for ROM and RAM for which the default region 0 is configured to enable access for MPU with secure attribute only. Software configures the firewall to avoid bad use of hardware resources.

6.2.8.2 General Recommendation

L3 interconnect registers must be read or written with little-endian attributes; otherwise, the result is undefined. Additionally, registers must be accessed only by using byte-enable patterns of all 0s or all 1s on a 32-bit aligned word (that is, all 0s or all 1s on a 32-bit interface, all 0s or all 1s on 32-bit MSBs, and 32-bit LSBs on a 64-bit interface); otherwise, the result is unpredictable (but no error is generated).

Overlapping between protection regions that do not have the overlay attribute must be avoided and result in unpredictable behavior.

6.2.8.3 Typical Example of Firewall Programming

This example reprograms the RAM firewall to enable accesses from sDMA with nonsecure attribute on the higher 512K bytes of OCM RAM memory.

1) A protection region must be chosen to apply the new access permission on the upper 512K bytes of the OCM-RAM memory area.
    a) Region 0 and region 1 cannot be chosen because they are special regions (default and firewall regions).
    b) Region 2 is already used. Region 2 is applied by default at reset on 2K bytes of OCM-RAM.
    c) Out of the seven protection regions that apply to the OCM-RAM, four protection regions are available for this reconfiguration: Regions 3, 4, 5, and 6.

2) This example uses protection region 3. The new access permissions can be programmed for protection region 3 by correct configuration of OCM_RAM firewall registers: ADDMATCH3, REQINFOPERM3, READPERM3, and WRITEPERM3.
    a) To avoid any border case during firewall programming, region 3 must be disabled. To disable the region, program ADDMATCH3_L with:
        i) ADDMATCH3_L[8:4] SIZE register field with:

        Value: 0x0
        Region disabled

    b) Set REQINFOPERM3_L with:

        REQINFOPERM3[7:0] REQINFO register field with:

        Value 0xFF
        All MREQ combinations allowed

    c) Set READPERM3_L with:

        READPERM3_L[15:0] READPERM register field with:

        Value: 0x8
        Feel bit vector with sDMA SystemCONNID value (0x3)

    d) Set WRITEPERM3_L with:

        WRITEPERM3_L[15:0] WRITEPERM register field with:

Value: 0x8
Feel bit vector with sDMA SystemCONNID value (0x3)

e) Set ADDMATCH3_L with:

i) ADDMATCH3_L[1:0] ADDRSPACE register field with:

Value: 00
OCM_RAM memory space is in address space 0

ii) ADDMATCH3_L[8:4] SIZE register field with:

Value: 0xA: Half OCM_RAM capacity: 512K bytes

Protection Region 3 selected size = 1K byte*(2^{size-1}) = 2^{10-1}K bytes = 512K bytes

iii) ADDMATCH3_L(9) OVERLAY register bit with:

Value: 0
No overlay region required because no security hole risk.

iv) ADDMATCH3_L[31:10] BASEADD register with:

Value: 0x40280000 on 22 bits
Base address of the half upper part of the OCM_RAM

3) Region 3 is set and can be accessed only by sDMA with any kind of secure attribute.

6.3 L3 Interconnect Registers

6.3.1 L3 LLRC Error Logging Registers

Table 6−43 shows the base address and address space for the LLRC target agents.

Table 6−43. L3 LLRC Error Logging Registers Instance Summary

Module Name	Base Address	Size
LLRC−OCM−RAM	0x6800 5400	512 bytes
LLRC−OCM−ROM	0x6800 5C00	512 bytes
LLRC−GPMC	0x6800 6400	512 bytes
LLRC−SMS	0x6800 6C00	512 bytes

This section provides information about the LLRC error−logging registers.

6.3.1.1 LLRC Register Descriptions

Table 6−44 lists the LLRC registers. Table 6−45 through Table 6−47 describe the register bits.

Table 6−44. LLRC Registers

Register Name	Type	Register Width (Bits)	Physical Address
TAERRLOGA	R	32	Base + 0x148
TAERRLOG	RW	32	Base + 0x150
TASTATE	RW	32	Base + 0x19C

Table 6−45. TAERRLOGA Register

Address Offset	0x148		
Physical Address	0x6800 5548	**Instance**	LLRC−OCM−RAM
	0x0000 5D48		LLRC−OCM−ROM
Description	Logs information about TA error conditions − LSBs		
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
TAERRADDR																															

Bits	Field Name	Description	Type	Reset
31:0	TAERRADDR	Address of request with error	R	0x−−−−−−−−

Table 6−46. TAERRLOG Register

Address Offset	0x150		
Physical Address	0x6800 5550	**Instance**	LLRC−OCM−RAM
	0x0000 5D50		LLRC−OCM−ROM
Description	Logs information about TA error conditions − LSBs		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAERRLOGMULT	RESERVED			TAERRCODE				Reserved												TAERRCONNID				Reserved					TAERRCMD		

Bits	Field Name	Description	Type	Reset
31	TAERRLOGMULT	Multiple errors detected	RW	0
30:28	Reserved	Read returns 0.	R	0x0
27:24	TAERRCODE	Error code 0x0: No error 0x4: Error response cannot be delivered in-band.	RW	0x0
23:12	Reserved	Read returns 0.	R	0x000
11:8	TAERRCONNID	System ConnID of request with error	R	0x−
7:3	Reserved	Read returns 0.	R	0x00
2:0	TAERRCMD	OCP MCmd of request with error	R	0x−

Table 6−47. TASTATE Register

Address Offset	0x19C		
Physical Address	0x6800 559C	**Instance**	LLRC−OCM−RAM
	0x0000 5D9C		LLRC−OCM−ROM
	0x0000 659C		LLRC−GPMC
	0x0000 6D9C		LLRC−SMS
Description	Error-logging register for OCM and GPMC. GPMC also logs reporting of SMS TA side-band errors − LSBs − reserved		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											TAERRRESP	RESERVED			TASERR

Bits	Field Name	Description	Type	Reset
31:5	Reserved	Read returns 0.	R	0x0000000
4	TAERRRESP	Detected ERR response	RW	0
3:1	Reserved	Read returns 0.	R	0x0
0	TASERR	Positive edge detected in SError	RW	0

6.3.2 L3 SB Registers

Table 6−48 shows the base address and address area for each module (agents and firewall) of the L3 SB interconnect.

Table 6−48 lists the L3 SB registers.

Table 6−48. L3 SB Registers Instance Summary

Module Name	Base Address	Size	Port
IM1	0x6800 0400	512 bytes	sDMA RD
IM2	0x6800 0600	512 bytes	sDMA WR
IM3	0x6800 0800	512 bytes	DSS
IM7	0x6800 0A00	512 bytes	DSP2SB
IMMPU2SB	0x6800 0C00	512 bytes	MPU2SB
IM6	0x6800 1200	512 bytes	USB
IM4	0x6800 1400	512 bytes	Camera
L4TA	0x6800 2400	512 bytes	L4 target agent
DSPFW	0x6800 2800	512 bytes	DSP memory firewall
TM4	0x6800 2E00	512 bytes	DSP memory
TM1	0x6800 4100	512 bytes	SB2SMS
TM3	0x6800 4300	512 bytes	SB2OCM
TM2	0x6800 4500	512 bytes	SB2GPMC
RAMFW	0x6800 5000	512 bytes	OCM RAM firewall
ROMFW	0x6800 5800	512 bytes	OCM ROM firewall
GPMCFW	0x6800 6000	512 bytes	GPMC firewall

6.3.2.1 L3 IM Register Descriptions

Table 6−49 summarizes the IM registers. Replace XX in the physical address with the correct value using a value from Table 6−48; Y corresponds to X+1.

Table 6−49 lists the IM1-8 registers. Table 6−50 through Table 6−56 describe the register bits.

Table 6−49. IM1−8 Registers

Register Name	Type	Register Width (Bits)	Physical Address
SBIMERRLOGA	R	32	0x6800 XXA8
SBIMERRLOG	RW	32	0x6800 XXB0
SBIMSTATE	RW	32	0x6800 XY90
SBTMSTATE	RW	32	0x6800 XY98
SBIMCONFIG_L	RW	32	0x6800 XYA8
SBIMCONFIG_H	RW	32	0x6800 XYAC
SBID	R	32	0x6800 XYF8

Table 6−50. SBIMERRLOGA Register

Address Offset	0x0A8		
Physical Address	0x6800 04A8	**Instance**	IM1
	0x6800 06A8		IM2
	0x6800 08A8		IM3
	0x6800 0AA8		IM7
	0x6800 10A8		IM5
	0x6800 12A8		IM6
	0x6800 14A8		IM4
	0x6800 20A8		IM8
Description	Logs information about IM error conditions		
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
SBIMERRADDR																															

Bits	Field Name	Description	Type	Reset
31:0	SBIMERRADDR	SB address of request with error	R	0x−−−−−−−−

Table 6–51. SBIMERRLOG Register

Address Offset	0x0B0		
Physical Address	0x6800 04B0	**Instance**	IM1
	0x6800 06B0		IM2
	0x6800 08B0		IM3
	0x6800 0AB0		IM7
	0x6800 10B0		IM5
	0x6800 12B0		IM6
	0x6800 14B0		IM4
	0x6800 20B0		IM8
Description	Logs information about IM error conditions		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SBIMERRLOGMULT	RESERVED			SBIMERRCODE				Reserved								SBIMERRCONNID								Reserved					SBIMERRCMD		

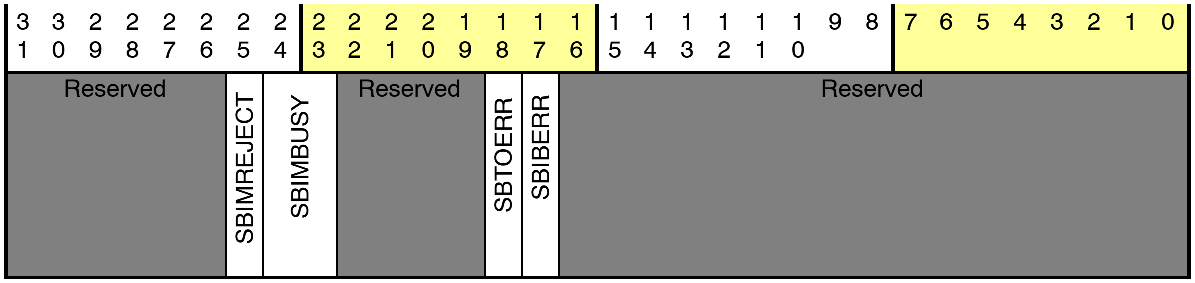
Bits	Field Name	Description	Type	Reset
31	SBIMERRLOG-MULT	1 if multiple errors detected	RW	0
30:28	Reserved	Read returns 0.	R	0x0
27:24	SBIMERRCODE	Error code 0x0: No error 0x1: No target or request not supported by target 0x2: Time-out 0x3: In-band error from target core 0x7: No target for broadcast request	RW	0x0
23:16	Reserved	Read returns 0.	R	0x00
15:8	SBIMERRCON-NID	SBConnID of request with error	R	0x––
7:3	Reserved	Reserved	R	0x––

Table 6−51. SBIMERRLOG (Continued)

Bits	Field Name	Description	Type	Reset
2:0	SBIMERRCMD	SBCmd of request with error 0x0: Idle 0x1: WritePost 0x2: Read 0x3: ReadEx 0x4: ReadRtry 0x5: WriteNonPost 0x6: WriteRetry 0x7: Broadcast	R	0x−

Table 6−52. SBIMSTATE Register

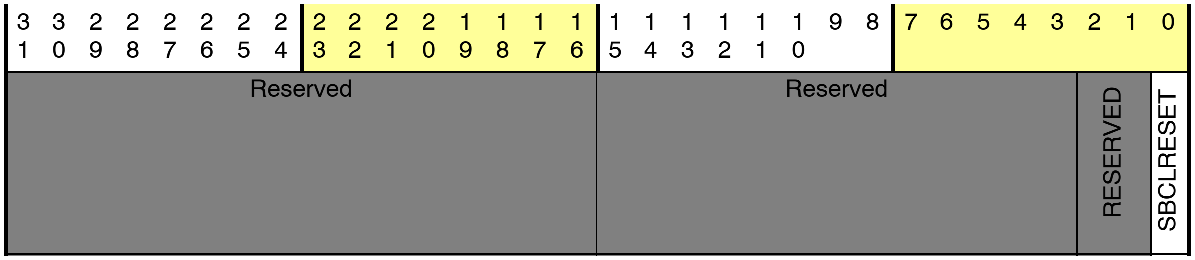
Address Offset	0x190		
Physical Address	0x6800 0590	**Instance**	IM1
	0x6800 0790		IM2
	0x6800 0990		IM3
	0x6800 0B90		IM7
	0x6800 1190		IM5
	0x6800 1390		IM6
	0x6800 1590		IM4
	0x6800 2190		IM8
Description	This register provides access to initiator agent state values.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:26	Reserved	Read returns 0.	R	0x00
25	SBIMREJECT	Rejects requests that start a new burst or ReadEx/Write pair	RW	0
24:23	SBIMBUSY	Agent has pending commands. 0x0: No requests and no response pending 0x1: No requests and no response pending, but open burst on OCP (any thread) 0x2: Requests pending with all SB protocol activities complete for all requests (that is, some OCP responses pending) 0x3: Requests pending SB protocol activities not yet complete for all requests	R	0x0
22:19	Reserved	Read returns 0.	R	0x0
18	SBTOERR	Time-out log	RW	0
17	SBIBERR	In-band error log	RW	0
16:0	Reserved	Read returns 0. **Caution** Reserved field SBIMSTATE[16:0] for IM3 (physical address = 0x6800 0990) is 0x00010. For the other IMs, this field value is 0x00000.	R	0x00000

Table 6–53. SBTMSTATE Register

Address Offset	0x198		
Physical Address	0x6800 0598	**Instance**	IM1
	0x6800 0798		IM2
	0x6800 0998		IM3
	0x6800 0B98		IM7
	0x6800 1198		IM5
	0x6800 1398		IM6
	0x6800 1598		IM4
	0x6800 2198		IM8
Description	SBTMSTATE is an unbuffered register that provides access to status information and target agent control values.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:16	Reserved	Read returns 0.	RW	0x0000
15:3	Reserved	Read returns 0.	RW	0x0000
2:1	Reserved	Read returns 0.	RW	0x0
0	SBCLRESET	Clear transactions and reset IM agent.	RW	0

Table 6−54. SBIMCONFIG_L Register

Address Offset	0x1A8		
Physical Address	0x6800 05A8	**Instance**	IM1
	0x6800 07A8		IM2
	0x6800 09A8		IM3
	0x6800 0BA8		IM7
	0x6800 11A8		IM5
	0x6800 13A8		IM6
	0x6800 15A8		IM4
	0x6800 21A8		IM8
Description	This register configures the operation of the initiator agent LSBs.		
Type	RW		

31	30	29	28	27	26	25	24	23	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									SBREQTIMEOUT			RESERVED	RESERVED		

Bits	Field Name	Description	Type	Reset
31:24	Reserved	Read returns 0.	R	0x00
23:16	Reserved	Reserved	R	0x−−
15:7	Reserved	Read returns 0.	R	0x000
6:4	SBREQTIMEOUT	Request time-out control 0x0: Time-out disabled 0x1: 2^6 silicon backplane cycles 0x2: 2^8 silicon backplane cycles 0x3: 2^10 silicon backplane cycles 0x4: 2^12 silicon backplane cycles 0x5: 2^14 silicon backplane cycles 0x6: 2^16 silicon backplane cycles 0x7: 2^18 silicon backplane cycles	RW	0x7
3	Reserved	Read returns 0.	R	0
2:0	Reserved	Reserved	R	0x−

Table 6−55. SBIMCONFIG_H Register

Address Offset	0x1AC		
Physical Address	0x6800 05AC	**Instance**	IM1
	0x6800 07AC		IM2
	0x6800 09AC		IM3
	0x6800 0BAC		IM7
	0x6800 11AC		IM5
	0x6800 13AC		IM6
	0x6800 15AC		IM4
	0x6800 21AC		IM8
Description	This register configures the operation of the initiator agent MSBs.		
Type	RW		

31	30	29	28	27	26	25	24	23	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	SBTOERRMODE	Reserved	SBIBERRMODE	Reserved	

Bits	Field Name	Description	Type	Reset
31:6	Reserved	Read returns 0.	R	0x0000000
5	Reserved	Read returns 0.	RW	0
4	SBTOERRMODE	IM time-out mapping 0x0: No effect 0x1: Drive sbError while Sbtoerr is set.	RW	1
3	Reserved	Read returns 0	RW	0
2	SBIBERRMODE	In-band error mapping 0x0: No effect 0x1: Drive sbError while Sbiberr is set.	RW	1
1:0	Reserved	Reserved	R	0x−

Table 6−56. SBID Register

Address Offset	0x1F8		
Physical Address	0x6800 05F8	**Instance**	IM1
	0x6800 07F8		IM2
	0x6800 09F8		IM3
	0x6800 0BF8		IM7
	0x6800 11F8		IM5
	0x6800 13F8		IM6
	0x6800 15F8		IM4
	0x6800 21F8		IM8
Description	Identification		
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
SBSWREV				Reserved																											

Bits	Field Name	Description	Type	Reset
31:28	SBSWREV	SB revision code per agent 0x0: Release 2.3 and earlier 0x1: Release 2.3	R	0x1
27:0	Reserved	Reserved	R	0x-------

6.3.2.2 IMMPU2SB Initiator Agent Register Descriptions

Table 6−57 summarizes the IMMPU2SB initiator agent, which is globally equivalent to other IM, with one additional feature: The SBTMSTATE[23:16] register field, SBCLCONTROL, used to save power. See the *Power Saving* subsection in Section 6.2.7.3.

Table 6−57 lists the IMMPU2SB registers. Table 6−58 through Table 6−66 describe the register bits.

Table 6−57. IMMPU2SB Registers

Register Name	Type	Register Width (Bits)	Physical Address
SBIMERRLOGA	R	32	0x6800 0CA8
SBIMERRLOG	RW	32	0x6800 0CB0
SBIMSTATE	RW	32	0x6800 0D90
SBTMSTATE_L	RW	32	0x6800 0D98
SBTMSTATE_H	RW	32	0x6800 0D9C
SBIMCONFIG_L	RW	32	0x6800 0DA8
SBIMCONFIG_H	RW	32	0x6800 0DAC
SBID	R	32	0x6800 0DF8

Table 6–58. SBIMERRLOGA Register

Address Offset	0x0A8		
Physical Address	0x6800 0CA8	**Instance**	IMMPU2SB
Description	Logs information about IM error conditions		
Type	R		

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="32">SBIMERRADDR</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
31:0	SBIMERRADDR	SB address of request with error	R	0x––––––––

Table 6–59. SBIMERRLOG Register

Address Offset	0x0B0		
Physical Address	0x6800 0CB0	**Instance**	IMMPU2SB
Description	Logs information about IM error conditions		
Type	RW		

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>SBIMERRLOGMULT</td><td colspan="3">RESERVED</td><td colspan="4">SBIBMERRCODE</td><td colspan="8">Reserved</td><td colspan="8">SBIMERRCONNID</td><td colspan="5">Reserved</td><td colspan="3">SBIMERRCMD</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
31	SBIMERRLOG-MULT	Multiple errors detected	RW	0
30:28	Reserved	Read returns 0.	R	0x0
27:24	SBIMERRCODE	Error code 0x0: No error 0x1: No target or request not supported by target 0x2: Time-out 0x3: In-band error from target core 0x7: No target for broadcast request	RW	0x0
23:16	Reserved	Read returns 0.	R	0x00
15:8	SBIMERR CONNID	SBConnID of request with error	R	0x––
7:3	Reserved	Read returns 0.	R	0x00

Table 6−59. SBIMERRLOG (Continued)

Bits	Field Name	Description	Type	Reset
2:0	SBIMERRCMD	SBCmd of request with error 0x0: Idle 0x1: WritePost 0x2: Read 0x3: ReadEx 0x4: ReadRtry 0x5: WriteNonPost 0x6: WriteRetry 0x7: Broadcast	R	0x−

Table 6−60. SBIMSTATE Register

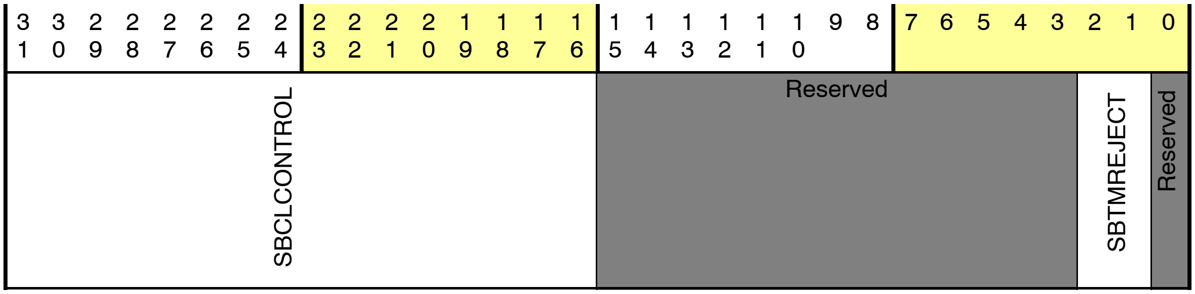
Address Offset	0x190		
Physical Address	0x6800 0D90	**Instance**	IMMPU2SB
Description	This register provides access to initiator agent state values.		
Type	RW		

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="6">Reserved</td><td>SBIMREJECT</td><td colspan="2">SBIMBUSY</td><td colspan="4">Reserved</td><td>SBTOERR</td><td>SBIBERR</td><td colspan="17">Reserved</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
31:26	Reserved	Read returns 0.	R	0x00
25	SBIMREJECT	Rejects requests that start a new burst or ReadEx/write pair	R	0
24:23	SBIMBUSY	Agent has pending commands. 0x0: No requests and no response pending 0x1: No requests and no response pending, but open burst on OCP (any thread) 0x2: Requests pending with all SB protocol activities complete for all requests (that is, some OCP responses pending) 0x3: Requests pending SB protocol activities not yet complete for all requests	R	0x0
22:19	Reserved	Read returns 0.	R	0x0
18	SBTOERR	Time-out log	RW	0
17	SBIBERR	In-band error log	RW	0
16:0	Reserved	Read returns 0.	R	0x00000

Table 6–61. SBTMSTATE_L Register

Address Offset	0x198		
Physical Address	0x6800 0D98	**Instance**	IMMPU2SB
Description	This register provides access to status information and target agent control values – LSBs.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:16	SBCLCONTROL	Number of inactivity cycles observed before SB clocks are gated Automatic L3 interconnect clock autogating feature refers to this parameter, which can be adjusted to optimize power consumption. 0x0000: Disable SB auto clock gating. 0x0001: One inactivity clock cycle observed before SB clocks are gated 0x0002: Two inactivity clock cycles observed before SB clocks are gated ... 0xXXX: 0xXXXX clock cycles observed before SB clocks are gated	RW	0x0000
15:3	Reserved	Read returns 0.	R	0x0000
2:1	SBTMREJECT	Controls rejection of new SB requests 0x0: Normal behavior 0x1: Included for backwards compatibility 0x2: Returns BUSY response for all nonregister requests that start a new burst. Requests within a burst are treated normally.	R	0x0
0	Reserved	Read returns 0	R	0

Table 6−62. SBTMSTATE_H Register

Address Offset	0x19C		
Physical Address	0x6800 0D9C	**Instance**	IMMPU2SB
Description	This register provides access to status information and target agent control values − MSBs.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																										SBTMTIMEOUT	SBERRRESP	SBTMRDE	SBTMBUSY	RESERVED	

Bits	Field Name	Description	Type	Reset
31:6	Reserved	Read returns 0.	R	0x0000000
5	SBTMTIMEOUT	1 if request has timed out; otherwise, 0	R	0
4	SBERRRESP	1 if detected error response to posted write; otherwise, 0	RW	0
3	SBTMRDE	1 if agent has ReadEx in progress; otherwise, 0	R	0
2	SBTMBUSY	1 if agent has pending commands; otherwise, 0	R	0
1:0	Reserved	Read returns 0.	R	0x0

Table 6−63. SBIMCONFIG_L Register

Address Offset	0x1A8
Physical Address	0x6800 0DA8 **Instance** IMMPU2SB
Description	This register configures the operation of the initiator agent LSBs.
Type	RW

31 30 29 28 27 26 25 24	23 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	SBREQTIMEOUT	RESERVED	RESERVED

Bits	Field Name	Description	Type	Reset
31:24	Reserved	Read returns 0.	R	0x00
23:16	Reserved	Reserved	R	0x−−
15:7	Reserved	Read returns 0.	R	0x000
6:4	SBREQTIMEOUT	Request time-out control 0x0: Time-out disabled 0x1: 2^6 silicon backplane cycles 0x2: 2^8 silicon backplane cycles 0x3: 2^10 silicon backplane cycles 0x4: 2^12 silicon backplane cycles 0x5: 2^14 silicon backplane cycles 0x6: 2^16 silicon backplane cycles 0x7: 2^18 silicon backplane cycles	RW	0x7
3	Reserved	Read returns 0.	R	0
2:0	Reserved	Reserved	R	0x−

Table 6−64. SBIMCONFIG_H Register

Address Offset	0x1AC
Physical Address	0x6800 0DAC **Instance** IMMPU2SB
Description	This register configures the operation of the initiator agent MSBs.
Type	RW

31 30 29 28 27 26 25 24 23 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	SBTOERRMODE	RESERVED	SBIBERRMODE	RESERVED

Table 6−64.SBIMCONFIG_H (Continued)

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Read returns 0.	R	0x000000
7:6	Reserved	Reads return 1.	RW	0x1
5	Reserved	Read returns 0.	RW	0
4	SBTOERRMODE	IM time-out mapping 0x0: No effect 0x1: Drive sbError while Sbtoerris set.	RW	1
3	Reserved	Read returns 0.	RW	0
2	SBIBERRMODE	In-band error mapping 0x0: No effect 0x1: Drive sbError while Sbiberr is set.	RW	0
1:0	Reserved	Reserved	R	0x−

Table 6−65. SBID_L Register

Address Offset	0x1F8		
Physical Address	0x6800 0DF8	**Instance**	IMMPU2SB
Description	Identification − Lower 32 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
SBSWREV				Reserved																											

Bits	Field Name	Description	Type	Reset
31:28	SBSWREV	SB revision code per agent 0x0: Release 2.3 and earlier 0x1: Release 2.3	R	0x1
27:0	Reserved	Reserved	R	0x−−−−−−−

Table 6−66. SBID_H Register

Address Offset	0x1FC		
Physical Address	0x6800 0DFC	**Instance**	IMMPU2SB
Description	Identification − Upper 32 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
SBVC																SBCC												SBRC			

Bits	Field Name	Description	Type	Reset
31:16	SBVC	Vendor code	R	0x50C5
15:4	SBCC	Core code	R	0x043
3:0	SBRC	Revision code	R	

6.3.2.3 L3 Firewall Registers

Table 6−67 summarizes all L3 interconnect firewalls, except SMSFW, which is different and described separately. Differences between firewalls are detailed in individual description tables in this section.

Table 6−67 lists the L3 firewall registers. Table 6−68 through Table 6−95 describe the register bits.

Table 6−67. L3 Firewall Registers

Register Name	Type	DSPFW	RAMFW	ROMFW	GPMCFW
ERRLOG	RW	0x6800 2820	0x6800 5020	0x6800 5820	0x6800 6020
REQINFOPERM0	RW	0x6800 2848	0x6800 5048	0x6800 5848	0x6800 6048
READPERM0	RW	0x6800 2850	0x6800 5050	0x6800 5850	0x6800 6050
WRITEPERM0	RW	0x6800 2858	0x6800 5058	0x6800 5858	0x6800 6058
ADDMATCH1	R	0x6800 2860	0x6800 5060	0x6800 5860	0x6800 6060
REQINFOPERM1	RW	0x6800 2868	0x6800 5068	0x6800 5868	0x6800 6068
READPERM1	R	0x6800 2870	0x6800 5070	0x6800 5870	0x6800 6070
WRITEPERM1	R	0x6800 2878	0x6800 5078	0x6800 5878	0x6800 6078
ADDMATCH2	RW	0x6800 2880	0x6800 5080	0x6800 5880	0x6800 6080
REQINFOPERM2	RW	0x6800 2888	0x6800 5088	0x6800 5888	0x6800 6088

Table 6−67. L3 Firewall Registers (Continued)

Register Name	Type	DSPFW	RAMFW	ROMFW	GPMCFW
READPERM2	RW	0x6800 2890	0x6800 5090	0x6800 5890	0x6800 6090
WRITEPERM2	RW	0x6800 2898	0x6800 5098	0x6800 5898	0x6800 6098
ADDMATCH3	R	0x6800 28A0	0x6800 50A0	0x6800 58A0	0x6800 60A0
REQINFOPERM3	R	0x6800 28A8	0x6800 50A8	0x6800 58A8	0x6800 60A8
READPERM3	R	0x6800 28B0	0x6800 50B0	0x6800 58B0	0x6800 60B0
WRITEPERM3	R	0x6800 28B8	0x6800 50B8	0x6800 58B8	0x6800 60B8
ADDMATCH4	R	NA	0x6800 50C0	NA	0x6800 60C0
REQINFOPERM4	R	NA	0x6800 50C8	NA	0x6800 60C8
READPERM4	R	NA	0x6800 50D0	NA	0x6800 60D0
WRITEPERM4	R	NA	0x6800 50D8	NA	0x6800 60D8
ADDMATCH5	R	NA	0x6800 50E0	NA	0x6800 60E0
REQINFOPERM5	R	NA	0x6800 50E8	NA	0x6800 60E8
READPERM5	R	NA	0x6800 50F0	NA	0x6800 60F0
WRITEPERM5	R	NA	0x6800 50F8	NA	0x6800 60F8
ADDMATCH6	R	NA	0x6800 5100	NA	0x6800 6100
REQINFOPERM6	R	NA	0x6800 5108	NA	0x6800 6108
READPERM6	R	NA	0x6800 5110	NA	0x6800 6110
WRITEPERM6	R	NA	0x6800 5118	NA	0x6800 6118

Table 6−68. ERRLOG Register

Address Offset	0x020	
Physical Address	0x6800 1820	**Instance**
	0x6800 2820	DSPFW
	0x6800 5020	RAMFW
	0x6800 5820	ROMFW
	0x6800 6020	GPMCFW
Description	Logs information about requests that incur a protection violation	
Type	RW	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MULT	RESERVED			CODE				Reserved					REQINFO			Reserved				INIATORID				REGION				RESERVED	CMD		

Bits	Field Name	Description	Type	Reset
31	MULT	1 if multiple errors detected	RW	0
30:28	Reserved	Read returns 0.	R	0x0
27:24	CODE	Error code 0x0: No error 0x3: Protection violation	RW	0x0
23:19	Reserved	Read returns 0.	R	0x00
18:16	REQINFO	MReqInfo bits of request with error	R	0x−
15:12	Reserved	Read returns 0.	R	0x0
11:8	INITIATORID	SBConnID of request with error	R	0x−
7:4	REGION	Protection region number to which request with error is mapped	R	0x−
3	Reserved	Read returns 0	R	0
2:0	CMD	OCP MCmd of request with error	R	0x−

Table 6−69. REQINFOPERM0 Register

Address Offset	0x048	
Physical Address	0x6800 1848	**Instance**
	0x6800 2848	DSPFW
	0x6800 5048	RAMFW
	0x6800 5848	ROMFW
	0x6800 6048	GPMCFW
Description	Request information permission configuration for region 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																								REQINFO							

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Read returns 0.	R	0x000000
7:0	REQINFO	Request information permission bits for region 0	RW	**RAMFW:** 0x10[1] **Others:** 0xFF

[1] Access reserved for MREQSECURE = 1 (secure access only), MREQTYPE = 0 (data access not instruction), MREQDEBUG = 0 (not in debug mode). Access enabled for secure data access not in debug mode.

Table 6−70. READPERM0 Register

Address Offset	0x050	
Physical Address	0x6800 1850	**Instance**
	0x6800 2850	DSPFW
	0x6800 5050	RAMFW
	0x6800 5850	ROMFW
	0x6800 6050	GPMCFW
Description	Read permission configuration for region 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																READPERM															

Table 6−70. READPERM0 Register (Continued)

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Read returns 0.	R	0x0000
15:0	READPERM	Read permission for region 0	RW	**RAMFW & ROMFW:** 0x2[1] **Others:** 0xFFFF

[1] Read access reserved for SystemConnID = 1: MPU: ARM11[RD,WR,INSTR]

Table 6−71. WRITEPERM0 Register

Address Offset	0x058		
Physical Address	0x6800 1858	**Instance**	
	0x6800 2858		DSPFW
	0x6800 5058		RAMFW
	0x6800 5858		ROMFW
	0x6800 6058		GPMCFW
Description	Write permission configuration for region 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																WRITEPERM															

Bits	Field Name	Description	Type	Reset
30:16	Reserved	Read returns 0.	R	0x0000
15:0	WRITEPERM	Write permission for region 0	RW	**RAMFW & ROMFW:** 0x2[1] **Others:** 0xFFFF

1) Write access reserved for SystemConnID = 1: MPU: ARM11[RD,WR,INSTR]

Table 6−72. ADDMATCH1 Register

Address Offset	0x060		
Physical Address	0x6800 1860	**Instance**	
	0x6800 2860		DSPFW
	0x6800 5060		RAMFW
	0x6800 5860		ROMFW
	0x6800 6060		GPMCFW
Description	Address match configuration for region 1		
Type	R		

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="22">BASEADD</td><td>OVERLAY</td><td colspan="5">SIZE</td><td colspan="2">RESERVED</td><td colspan="2">ADDRSPACE</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
31:10	BASEADD	Protection region 1 base address	R	0x006000
9	OVERLAY	Overlay enable bit for region 1 0: Nonoverlay region 1: Overlay region	R	0
8:4	SIZE	Protection region 1 size: 2^(SIZE−1) K bytes region area	R	0x01
3:2	Reserved	Read returns 0.	R	0x0
1:0	ADDRSPACE	Protection region address space	R	0x3

Table 6−73. REQINFOPERM1 Register

Address Offset	0x068		
Physical Address	0x6800 1868	**Instance**	
	0x6800 2868		DSPFW
	0x6800 5068		RAMFW
	0x6800 5868		ROMFW
	0x6800 6068		GPMCFW
Description	Request information permission configuration for region 1		
Type	RW		

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="24">Reserved</td><td colspan="8">REQINFO</td></tr>
</table>

Table 6−73. REQINFOPERM1 Register (Continued)

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Read returns 0.	R	0x000000
7:0	REQINFO	Request information permission bits for region1	RW	**ROMFW:** 0x10 **RAMFW:** EXP[1] **Others:** 0xFF

[1] Exported reset value. The register containing the reset value can be found in the SCM: CONTROL_OCM_RAM_PERM1[7:0] register field: OCMRAMPERM1. OCMRAMPERM1 is a read-only register that contains the reset value of REQINFO for RAMFW. The OCMRAMPERM1 value depends on the device type. For a general−purpose device, reset value = 0xFF.

Table 6−74. READPERM1 Register

Address Offset	0x070	
Physical Address	0x6800 1870	**Instance**
	0x6800 2870	DSPFW
	0x6800 5070	RAMFW
	0x6800 5870	ROMFW
	0x6800 6070	GPMCFW
Description	Read permission configuration for region 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																READPERM															

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Read returns 0.	R	0x0000
15:0	READPERM	Read permission for region 1	R	0x0002

Table 6−75. WRITEPERM1 Register

Address Offset	0x078		
Physical Address	0x6800 1878	**Instance**	
	0x6800 2878		DSPFW
	0x6800 5078		RAMFW
	0x6800 5878		ROMFW
	0x6800 6078		GPMCFW
Description	Write permission configuration for region 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																WRITEPERM															

Bits	Field Name	Description	Type	Reset
30:16	Reserved	Read returns 0.	R	0x0000
15:0	WRITEPERM	Write permission for region 1	R	0x0002

Table 6−76. ADDMATCH2 Register

Address Offset	0x080		
Physical Address	0x6800 1880	**Instance**	
	0x6800 2880		DSPFW
	0x6800 5080		RAMFW
	0x6800 5880		ROMFW
	0x6800 6080		GPMCFW
Description	Address match configuration for region 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
BASEADD																						OVERLAY	SIZE					RESERVED		ADDRSPACE	

Table 6−76.ADDMATCH2 (Continued)

Bits	Field Name	Description	Type	Reset
31:10	BASEADD	Protection region base address	**ROMFW:** R **Others:** RW	**ROMFW:** 0x100020 **RAMFW:** EXP[1] **Others:** 0x000000
9	OVERLAY	Protection region 2 overlay control 0: Nonoverlay region 1: Overlay region	**ROMFW:** R **Others:** RW	0
8:4	SIZE	Protection region 2 size: 2^(SIZE−1) K bytes region area	**ROMFW:** R **Others:** RW	**ROMFW:** 0x06 **RAMFW:** EXP[2] **Others** 0x00
3:2	Reserved	Read returns 0.	R	0x0
1:0	ADDRSPACE	Protection region 2 address space	**ROMFW:** R **Others:** RW	**RAMF:** EXP[3] **Others:** 0x0

1) Exported reset value: The reset value can be found in the SCM: CONTROL_OCM_RAM_PUB_ADD2[31:10]. The value contained in this register field becomes the reset value for BASEADD. The reset value for CONTROL_OCM_RAM_PUB_ADD2[31:10] is 0x10083E. CONTROL_OCM_RAM_PUB_ADD2[31:10] can be overwritten one time; it is a read/one write only register. The export value is taken into account and effectively exported in BASEADD at the release of the warm-up reset. This reset value can be overwritten at any time by accessing the BASEADD field.

2) Exported reset value: The reset value can be found in the SCM: CONTROL_OCM_RAM_PUB_ADD2[8:4]. The value contained in this register field becomes the reset value for SIZE. The reset value for CONTROL_OCM_RAM_PUB_ADD2[8:4] is 0x02. CONTROL_OCM_RAM_PUB_ADD2[8:4] can be overwritten one time; it is a read/ one write only register. The export value is taken into account and effectively exported in SIZE at the release of the warm-up reset. This reset value can be overwritten at any time by accessing the SIZE field.

3) Exported reset value: The reset value can be found in the SCM: CONTROL_OCM_RAM_PUB_ADD2[1:0]. The value contained in this register field becomes the reset value for ADDRSPACE. The reset value for CONTROL_OCM_RAM_PUB_ADD2[31:10] is 0x0. CONTROL_OCM_RAM_PUB_ADD2[1:0] can be overwritten one time; it is a read/one write only register. The export value is taken into account and effectively exported in ADDRSPACE at the release of the warm-up reset. This reset value can be overwritten at any time by accessing the ADDRSPACE field.

Table 6−77. REQINFOPERM2 Register

Address Offset	0x088	
Physical Address	0x6800 1888	**Instance**
	0x6800 2888	DSPFW
	0x6800 5088	RAMFW
	0x6800 5888	ROMFW
	0x6800 6088	GPMCFW
Description	Request information permission configuration for region 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																								REQINFO							

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Read returns 0.	R	0x000000
7:0	REQINFO	Request information permission bits for region 2	RW	**ROMFW:** 0x301† **Others:** 0xFF

1) Access reserved to MreqSecure = 1 (secure access only), MreqType = 0/1 (data or instruction), MreqDebug = 0 (not in debug mode). Access enabled for secure data/instruction access not in debug mode.

Table 6−78. READPERM2 Register

Address Offset	0x090	
Physical address	0x6800 1890	**Instance**
	0x6800 2890	DSPFW
	0x6800 5090	RAMFW
	0x6800 5890	ROMFW
	0x6800 6090	GPMCFW
Description	Read permission configuration for region 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																READPERM															

Table 6−78. READPERM2 (Continued)

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Read returns 0.	R	0x0000
15:0	READPERM	Read permission for region 2	**ROMFW:** R **Others:** RW	**ROMFW:** 0x0002[1] **Others:** 0xFFFF

[1] Read access reserved for SystemConnID = 1: MPU: ARM11[RD,WR,INSTR]

Table 6−79. WRITEPERM2 Register

Address Offset	0x098	
Physical Address	0x6800 1898	**Instance**
	0x6800 2898	DSPFW
	0x6800 5098	RAMFW
	0x6800 5898	ROMFW
	0x6800 6098	GPMCFW
Description	Write permission configuration for region 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																WRITEPERM															

Bits	Field Name	Description	Type	Reset
30:16	Reserved	Read returns 0.	R	0x0000
15:0	WRITEPERM	Write permission for region 2	**ROMFW:** R **Others:** RW	**ROMFW:** 0x0000[1] **Others:** 0xFFFF

[1] Write access forbidden for any initiator.

Table 6−80. ADDMATCH3 Register

Address Offset	0x0A0		
Physical Address	0x6800 28A0	**Instance**	DSPFW
	0x6800 50A0		RAMFW
	0x6800 58A0		ROMFW
	0x6800 60A0		GPMCFW
Description	Address match configuration for region 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
BASEADD																						OVERLAY	SIZE					RESERVED		ADDRSPACE	

Bits	Field Name	Description	Type	Reset
31:10	BASEADD	Protection region 3 base address	**ROMFW:** R **Others:** RW	**ROMFW:** 0x100040 **Others:** 0x000000
9	OVERLAY	Protection region 3 overlay control 0: Nonoverlay region 1: Overlay region	**ROMFW:** R **Others:** RW	0
8:4	SIZE	Protection region 3 size: 2^(SIZE−1) K-bytes region area	**ROMFW:** R **Others:** RW	**ROMFW:** 0x06 **Others:** 0x00
3:2	Reserved	Read returns 0.	R	0x0
1:0	ADDRSPACE	Protection region 3 address space	**ROMFW:** R **Others:** RW	0x0

Table 6–81. REQINFOPERM3 Register

Address Offset	0x0A8		
Physical Address	0x6800 28A8	**Instance**	DSPFW
	0x6800 50A8		RAMFW
	0x6800 58A8		ROMFW
	0x6800 60A8		GPMCFW
Description	Request information permission configuration for region 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																								REQINFO							

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Read returns 0.	R	0x000000
7:0	REQINFO	Request information permission bits for region 3	RW	**ROMFW:** 0x30[1] **Others:** 0xFF

[1] Access reserved to MreqSecure = 1 (secure access only), MreqType = 0/1 (data or instruction), MreqDebug = 0 (not in Debug mode). Access enabled for secure data/instruction access not in debug mode.

Table 6–82. READPERM3 Register

Address Offset	0x0B0		
Physical Address	0x6800 28B0	**Instance**	DSPFW
	0x6800 50B0		RAMFW
	0x6800 58B0		ROMFW
	0x6800 60B0		GPMCFW
Description	Read permission configuration for region 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																READPERM															

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Read returns 0.	R	0x0000
15:0	READPERM	Read permission for region 3	RW	**ROMFW:** 0x0002[1] **Others**: 0xFFFF

[1] Read access reserved for SystemConnID = 1: MPU: ARM11[RD,WR,INSTR]

Table 6−83. WRITEPERM3 Register

Address Offset	0x0B8		
Physical Address	0x6800 28B8	**Instance**	DSPFW
	0x6800 50B8		RAMFW
	0x6800 58B8		ROMFW
	0x6800 60B8		GPMCFW
Description	Write permission configuration for region 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																WRITEPERM															

Bits	Field Name	Description	Type	Reset
30:16	Reserved	Read returns 0.	R	0x0000
15:0	WRITEPERM	Write permission for region 3	RW	**ROMFW:** 0x0000[†] **Others**: 0xFFFF

[†] Write access forbidden for any initiator.

Table 6−84. ADDMATCH4 Register

Address Offset	0x0C0		
Physical Address	0x6800 50C0	**Instance**	RAMFW
	0x6800 60C0		GPMCFW
Description	Address match configuration for region 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
BASEADD																						OVERLAY	SIZE					RESERVED		ADDRSPACE	

Table 6−84.ADDMATCH4 (Continued)

Bits	Field Name	Description	Type	Reset
31:10	BASEADD	Protection region 4 base address	RW	0x000000
9	OVERLAY	Protection region 4 overlay control 0: Nonoverlay region 1: Overlay region	RW	0
8:4	SIZE	Protection region 4 size: 2^(SIZE−1)K bytes region area	RW	0x00
3:2	Reserved	Read returns 0.	R	0x0
1:0	ADDRSPACE	Protection region 4 address space	RW	0x0

Table 6−85. REQINFOPERM4 Register

Address Offset	0x0C8		
Physical Address	0x6800 50C8	**Instance**	RAMFW
	0x6800 60C8		GPMCFW
Description	Request Info permission configuration for region 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																								REQINFO							

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Read returns 0.	R	0x000000
7:0	REQINFO	Request information permission bits for region 4	RW	0xFF

Table 6−86. READPERM4 Register

Address Offset	0x0D0		
Physical Address	0x6800 50D0	**Instance**	RAMFW
	0x6800 60D0		GPMCFW
Description	Read permission configuration for region 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																READPERM															

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Read returns 0.	R	0x0000
15:0	READPERM	Read permission for region 4	RW	0xFFFF

Table 6−87. WRITEPERM4 Register

Address Offset	0x0D8		
Physical Address	0x6800 50D8	**Instance**	RAMFW
	0x6800 60D8		GPMCFW
Description	Write permission configuration for region 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																WRITEPERM															

Bits	Field Name	Description	Type	Reset
30:16	Reserved	Read returns 0.	R	0x0000
15:0	WRITEPERM	Write permission for region 4	RW	0xFFFF

Table 6−88. ADDMATCH5 Register

Address Offset	0x0E0		
Physical Address	0x6800 50E0	**Instance**	RAMFW
	0x6800 60E0		GPMCFW
Description	Address match configuration for region 5		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BASEADD																						OVERLAY	SIZE					RESERVED		ADDRSPACE	

Bits	Field Name	Description	Type	Reset
31:10	BASEADD	Protection region 5 base address	RW	0x000000
9	OVERLAY	Protection region 5 overlay control 0 Nonoverlay region 1 Overlay region	RW	0
8:4	SIZE	Protection region 5 size: 2^(SIZE−1) K bytes region area	RW	0x00
3:2	Reserved	Read returns 0.	R	0x0
1:0	ADDRSPACE	Protection region 5 address space	RW	0x0

Table 6–89. REQINFOPERM5 Register

Address Offset	0x0E8		
Physical Address	0x6800 50E8	**Instance**	RAMFW
	0x6800 60E8		GPMCFW
Description	Request information permission configuration for region 5		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								REQINFO							

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Read returns 0.	R	0x000000
7:0	REQINFO	Request information permission bits for region 5	RW	0xFF

Table 6–90. READPERM5 Register

Address Offset	0x0F0		
Physical Address	0x6800 50F0	**Instance**	RAMFW
	0x6800 60F0		GPMCFW
Description	Read permission configuration for region 5		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																READPERM															

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Read returns 0.	R	0x0000
15:0	READPERM	Read permission for region 5	RW	0xFFFF

Table 6–91. WRITEPERM5 Register

Address Offset	0x0F8		
Physical Address	0x6800 50F8	**Instance**	RAMFW
	0x6800 60F8		GPMCFW
Description	Write permission configuration for region 5		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																WRITEPERM															

Bits	Field Name	Description	Type	Reset
30:16	Reserved	Read returns 0.	R	0x0000
15:0	WRITEPERM	Write permission for region 5	RW	0xFFFF

Table 6−92. ADDMATCH6 Register

Address Offset	0x100		
Physical Address	0x6800 5100	**Instance**	RAMFW
	0x6800 6100		GPMCFW
Description	Address match configuration for region 6		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BASEADD																						OVERLAY	SIZE					RESERVED		ADDRSPACE	

Bits	Field Name	Description	Type	Reset
31:10	BASEADD	Protection region 6 base address	RW	0x000000
9	OVERLAY	Protection region 6 overlay control 0: Nonoverlay region 1: Overlay region	RW	0
8:4	SIZE	Protection region size: 2^(SIZE−1)K bytes region area	RW	0x00
3:2	Reserved	Read returns 0.	R	0x0
1:0	ADDRSPACE	Protection region 6 address space	RW	0x0

Table 6−93. REQINFOPERM6 Register

Address Offset	0x108		
Physical Address	0x6800 5108	**Instance**	RAMFW
	0x6800 6108		GPMCFW
Description	Request information permission configuration for region 6		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								REQINFO							

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Read returns 0.	R	0x000000
7:0	REQINFO	Request information permission bits for region 6	RW	0xFF

Table 6–94. READPERM6 Register

Address Offset	0x110		
Physical Address	0x6800 5110	**Instance**	RAMFW
	0x6800 6110		GPMCFW
Description	Read permission configuration for region 6		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																READPERM															

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Read returns 0.	R	0x0000
15:0	READPERM	Read permission for region 6	RW	0xFFFF

Table 6–95. WRITEPERM6 Register

Address Offset	0x118		
Physical Address	0x6800 5118	**Instance**	RAMFW
	0x6800 6118		GPMCFW
Description	Write permission configuration for region 6		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																WRITEPERM															

Bits	Field Name	Description	Type	Reset
30:16	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x0000
15:0	WRITEPERM	Write permission for region 6	RW	0xFFFF

6.3.2.4 IMTM1 Register Descriptions

Table 6–96 lists the IMTT1 registers. Table 6–97 through Table 6–107 describe the register bits.

Table 6–96. IMTM1 Registers

Register Name	Type	Register Width (Bits)	Physical Address
SBIMERRLOGA	R	32	0x6800 1EA8
SBIMERRLOG	RW	32	0x6800 1EB0
SBTMERRLOGA	R	32	0x6800 1F48
SBTMERRLOG	RW	32	0x6800 1F50
SBIMSTATE	RW	32	0x6800 1F90
SBTMSTATE_L	RW	32	0x6800 1F98
SBTMSTATE_H	RW	32	0x6800 1F9C
SBIMCONFIG_L	RW	32	0x6800 1FA8
SBIMCONFIG_H	RW	32	0x6800 1FAC
SBTMCONFIG	R	32	0x6800 1FBC
SBID	R	32	0x6800 1FF8

Table 6–97. SBIMERRLOGA Register

Address Offset	0x0A8		
Physical Address	0x6800 1EA8	**Instance**	IMTM1
Description	Logs information about IM error conditions		
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
SBIMERRADDR																															

Bits	Field Name	Description	Type	Reset
31:0	SBIMERRADDR	SB address of request with error	R	0x––––––––

Table 6−98. SBIMERRLOG Register

Address Offset	0x0B0		
Physical Address	0x6800 1EB0	**Instance**	IMTM1
Description	Logs information about IM error conditions		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SBIMERRLOGMULT	RESERVED			SBIMERRCODE				Reserved								SBIMERRCONNID								Reserved					SBIMERRCMD		

Bits	Field Name	Description		Type	Reset
31	SBIMERRLOG-MULT	Multiple errors detected		RW	0
30:28	Reserved	Read returns 0.		R	0x0
27:24	SBIMERRCODE	Error code		RW	0x0
		0x0:	No error		
		0x1:	No target or request not supported by target		
		0x2:	Time-out		
		0x3:	In-band error from target core		
		0x7:	No target for broadcast request		
23:16	Reserved	Read returns 0.		R	0x00
15:8	SBIMERR CONNID	SBConnID of request with error		R	0x−−
7:3	Reserved	Read returns 0.		R	0x00
2:0	SBIMERRCMD	SBCmd of request with error		R	0x−
		0x0:	Idle		
		0x1:	WritePost		
		0x2:	Read		
		0x3:	ReadEx		
		0x4:	ReadRtry		
		0x5:	WriteNonPost		
		0x6:	WriteRetry		
		0x7:	Broadcast		

Table 6−99. SBTMERRLOGA Register

Address Offset	0x148		
Physical Address	0x6800 1F48	**Instance**	IMTM1
Description	Logs information about TM error conditions		
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
SBTMERRADDR																															

Bits	Field Name	Description	Type	Reset
31:0	SBTMERRADDR	SB address of request with error	R	0x--------

Table 6−100. SBTMERRLOG Register

Address Offset	0x150		
Physical Address	0x6800 1F50	**Instance**	IMTM1
Description	Logs information about TM error conditions		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SBTMERRLOGMULT	RESERVED			SBTMERRCODE				Reserved								SBTMERRCONNID								Reserved					Reserved		

Bits	Field Name	Description	Type	Reset
31	SBTMERRLOG- MULT	1 if multiple errors detected	RW	0
30:28	Reserved	Read returns 0.	R	0x0
27:24	SBTMERRCODE	Error code 0x0: No error 0x4: Error response cannot be delivered in−band.	RW	0x0
23:16	Reserved	Read returns 0.	R	0x00
15:8	SBTMERR CONNID	SBConnID of request with error	R	0x−−
7:3	Reserved	Read returns 0.	R	0x00
2:0	Reserved	Reserved	R	0x−

Table 6−101. SBIMSTATE Register

Address Offset	0x190		
Physical Address	0x6800 1F90	**Instance**	IMTM1
Description	This register provides access to initiator agent state values.		
Type	RW		

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="6">Reserved</td><td>SBIMREJECT</td><td colspan="2">SBIMBUSY</td><td colspan="4">Reserved</td><td>SBTOERR</td><td>SBIBERR</td><td colspan="17">Reserved</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
31:26	Reserved	Read returns 0.	R	0x00
25	SBIMREJECT	Rejects requests that start a new burst or ReadEx/Write pair	RW	0
24:23	SBIMBUSY	Agent has pending commands. 0x0: No requests and no response pending 0x1: No requests and no response pending, but open burst on OCP (any thread) 0x2: Requests pending with all SB protocol activities complete for all requests (that is, some OCP responses pending) 0x3: Requests pending SB protocol activities not yet complete for all requests	R	0x0
22:19	Reserved	Read returns 0.	R	0x0
18	SBTOERR	Time-out log	RW	0
17	SBIBERR	In-band error log	RW	0
16:0	Reserved	Read returns 0.	R	0x00000

Table 6−102. SBTMSTATE_L Register

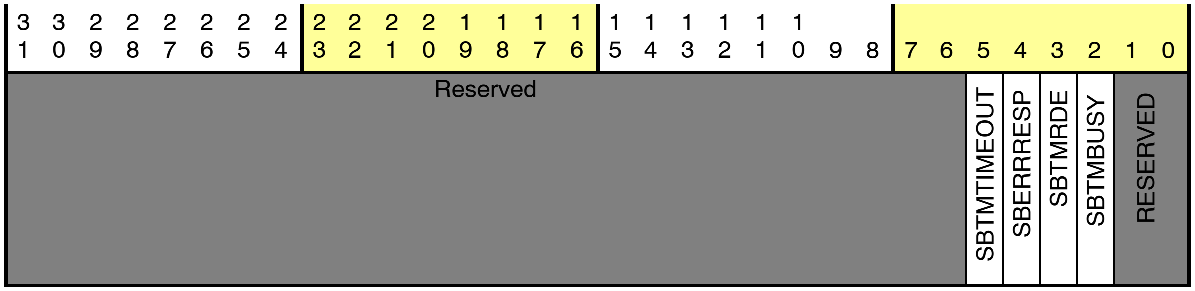
Address Offset	0x198		
Physical Address	0x6800 1F98	**Instance**	IMTM1
Description	This register provides access to status information and target agent control values − LSBs.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													FSBTMREJECT		SBCLRESET

Bits	Field Name	Description	Type	Reset
31:3	Reserved	Read returns 0.	RW	0x00000000
2:1	SBTMREJECT	Controls rejection of new SB requests 0x0: Normal behavior 0x1: Included for backwards compatibility 0x2: Return BUSY response for all nonregister requests that start a new burst. Requests within a burst are treated normally.	RW	0x0
0	SBCLRESET	Clear transactions and reset IW or TW	RW	0

Table 6−103. SBTMSTATE_H Register

Address Offset	0x19C		
Physical address	0x6800 1F9C	**Instance**	IMTM1
Description	This register provides access to status information and target agent control values − MSBs.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:6	Reserved	Read returns 0.	RW	0x0000000
5	SBTMTIMEOUT	1 if request has timed out; otherwise, 0	R	0
4	SBERRRESP	1 if detected error response to posted write; otherwise, 0	RW	0
3	SBTMRDE	1 if agent has ReadEx in progress; otherwise, 0	R	0
2	SBTMBUSY	1 if agent has pending commands; otherwise, 0	R	0
1:0	Reserved	Read returns 0.	RW	0x0

Table 6−104. SBIMCONFIG_L Register

Address Offset	0x1A8		
Physical Address	0x6800 1FA8	**Instance**	IMTM1
Description	This register configures the operation of the initiator agent LSBs.		
Type	RW		

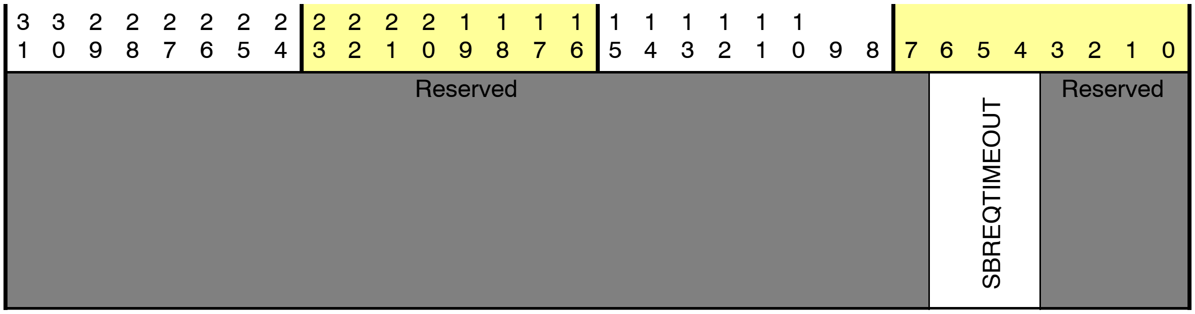


Table 6−104. SBIMCONFIG_L (Continued)

Bits	Field Name	Description	Type	Reset
31:7	Reserved	Read returns 0.	R	0x0000000
6:4	SBREQTIMEOUT	Request time-out control 0x0: Time-out disabled 0x1: 2^6 SB cycles 0x2: 2^8 SB cycles 0x3: 2^10 SB cycles 0x4: 2^12 SB cycles 0x5: 2^14 SB cycles 0x6: 2^16 SB cycles 0x7: 2^18 SB cycles	RW	0x7
3:0	Reserved	Read returns 0	R	0x0

Table 6−105. SBIMCONFIG_H Register

Address Offset	0x1AC		
Physical Address	0x6800 1FAC	**Instance**	IMTM1
Description	This register configures the operation of the initiator agent MSBs.		
Type	RW		

31	30	29	28	27	26	25	24	23	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	SBTOERRMODE	RESERVED	SBIBERRMODE	RESERVED	

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Read returns 0.	R	0x000000
7:6	Reserved	Read returns 0.	RW	0x0
5	Reserved	Read returns 0.	RW	0
4	SBTOERRMODE	IM time-out mapping 0x0: No effect 0x1: Drive sbError while Sbtoerr is set	RW	1
3	Reserved	Read returns 0.	RW	0
2	SBIBERRMODE	In-band error mapping 0x0: No effect 0x1: Drive sbError while Sbiberr is set	RW	1
1:0	Reserved	Read returns 0.	R	0x0

Table 6−106. SBTMCONFIG Register

Address Offset	0x1BC		
Physical Address	0x6800 1FBC	**Instance**	IMTM1
Description	This register configures the operation of the clock generator and the target agent response flag use.		
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																			SBOBERRMODE	Reserved											

Bits	Field Name	Description	Type	Reset
31:13	Reserved	Read returns 0.	R	0x00000
12	SBOBERRMODE	Controls out of band reporting of in-band OCP errors	R	1
11:0	Reserved	Read returns 0.	R	0x000

Table 6−107. SBID Register

Address Offset	0x1F8		
Physical Address	0x6800 1FF8	**Instance**	IMTM1
Description	Identification		
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
SBSWREV				Reserved																											

Bits	Field Name	Description	Type	Reset
31:28	SBSWREV	SB revision code per agent 0x0: Release 2.3 and earlier 0x1: Release 2.3	R	0x1
27:0	Reserved	Reserved	R	0x−−−−−−−

6.3.2.5 L4TA Register Descriptions

Table 6−108 lists the L4TA registers. Table 6−109 through Table 6−116 describe the register bits.

Table 6−108. L4TA Registers

Register Name	Type	Register Width (Bits)	Physical Address
SBTMPORTCONNID0	R	32	0x6800 24D8
SBTMPORTLOCK0	R	32	0x6800 24F8
SBTMERRLOGA	R	32	0x6800 2548
SBTMERRLOG	RW	32	0x6800 2550
SBTMSTATE_L	RW	32	0x6800 2598
SBTMSTATE_H	RW	32	0x6800 259C
SBTMCONFIG	R	32	0x6800 25BC
SBID	R	32	0x6800 25F8

Table 6−109. SBTMPORTCONNID0 Register

Address Offset	0x0D8		
Physical Address	0x6800 24D8	**Instance**	L4TA
Description	Reserved		
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								Reserved							

Bits	Field Name	Description	Type	Reset
31:24	Reserved	Reserved	R	0x−−
23:16	Reserved	Reserved	R	0x−−
15:8	Reserved	Reserved	R	0x−−
7:0	Reserved	Reserved	R	0x−−

Table 6–110. SBTMPORTLOCK0 Register

Address Offset	0x0F8		
Physical Address	0x6800 24F8	**Instance**	L4TA
Description	This register allows software to determine whether TM port locks exist.		
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							RESERVED	Reserved							RESERVED	Reserved							RESERVED

Bits	Field Name	Description	Type	Reset
31:25	Reserved	Read returns 0.	R	0x00
24	Reserved	Reserved	R	0
23:17	Reserved	Read returns 0.	R	0x00
16	Reserved	Reserved	R	0
15:9	Reserved	Read returns 0.	R	0x00
8	Reserved	Reserved	R	0
7:1	Reserved	Read returns 0.	R	0x00
0	Reserved	Reserved	R	0

Table 6–111. SBTMERRLOGA Register

Address Offset	0x148		
Physical Address	0x6800 2548	**Instance**	L4TA
Description	Logs information about TM error conditions		
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
SBTMERRADDR																															

Bits	Field Name	Description	Type	Reset
31:0	SBTMERRADDR	SB address of request with error	R	0x––––––––

Table 6−112. SBTMERRLOG Register

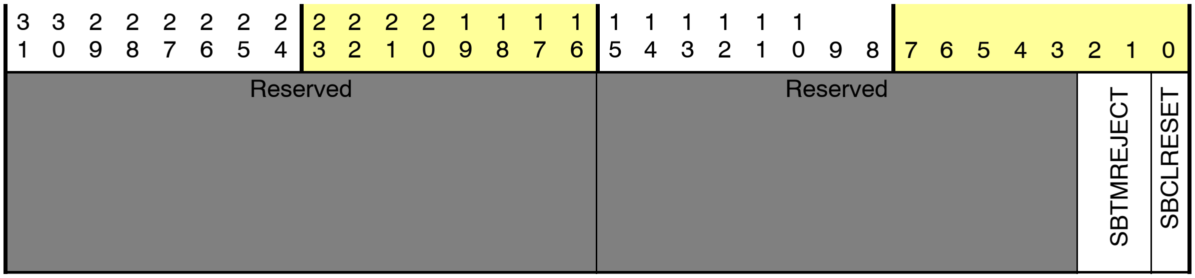
Address Offset	0x150		
Physical Address	0x6800 2550	**Instance**	L4TA
Description	Logs information about TM error conditions		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SBTMERRLOGMULT	RESERVED			SBTMERRCODE				Reserved								SBTMERRCONNID								Reserved					RESERVED		

Bits	Field Name	Description	Type	Reset
31	SBTMERRLOG-MULT	Multiple errors detected	RW	0
30:28	Reserved	Read returns 0.	R	0x0
27:24	SBTMERRCODE	Error code 0x0: No error 0x4: Error response cannot be delivered in−band.	RW	0x0
23:16	Reserved	Read returns 0.	R	0x00
15:8	SBTMERR CONNID	SBConnID of request with error	R	0x−−
7:3	Reserved	Read returns 0.	R	0x00
2:0	Reserved	Reserved	R	0x−

Table 6−113. SBTMSTATE_L Register

Address Offset	0x198		
Physical Address	0x6800 2598	**Instance**	L4TA
Description	This register provides access to status information and target agent control values − LSBs.		
Type	RW		

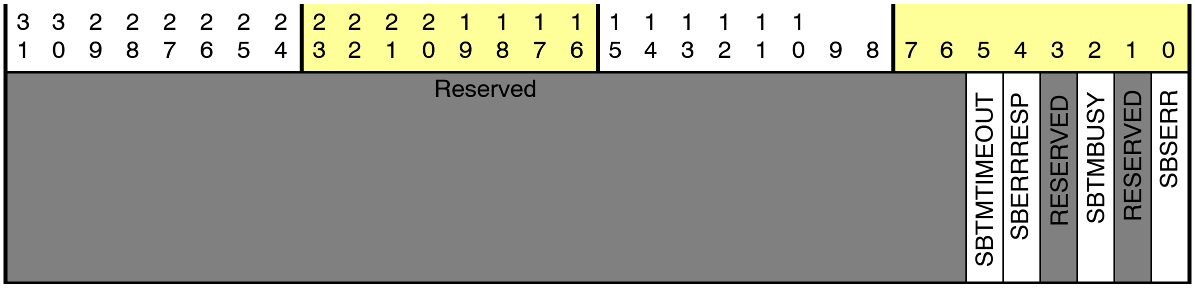


31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																Reserved													SBTMREJECT		SBCLRESET

Bits	Field Name	Description	Type	Reset
31:3	Reserved	Read returns 0.	R	0x0000000
2:1	SBTMREJECT	Controls rejection of new SB requests 0x0: Normal behavior 0x1: Included for backwards compatibility 0x2: Return BUSY response for all nonregister requests that start a new burst. Requests within a burst are treated normally.	RW	0x0
0	SBCLRESET	Clear transactions and reset IW or TW	RW	0

Table 6−114. SBTMSTATE_H Register

Address Offset	0x19C		
Physical Address	0x6800 259C	**Instance**	L4TA
Description	This register provides access to status information and target agent control values − MSBs.		
Type	RW		



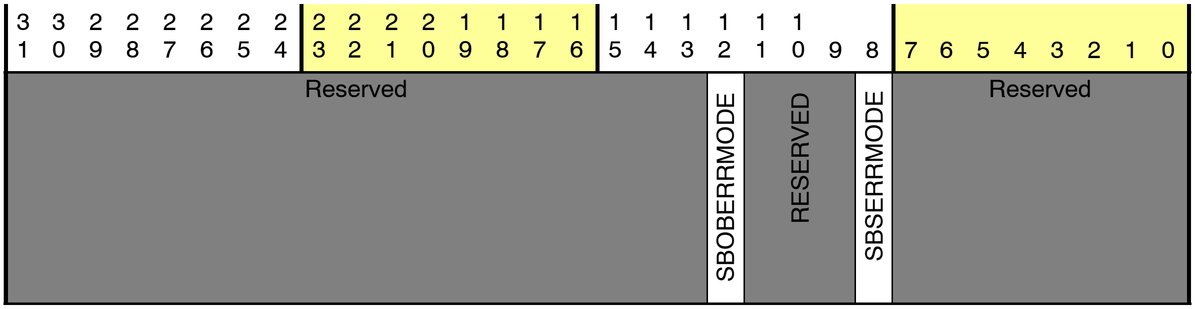
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																										SBTMTIMEOUT	SBERRRESP	RESERVED	SBTMBUSY	RESERVED	SBSERR

Table 6−114. SBTMSTATE_H (Continued)

Bits	Field Name	Description	Type	Reset
31:6	Reserved	Read returns 0.	R	0x0000000
5	SBTMTIMEOUT	1 if request has timed out; otherwise, 0	R	0
4	SBERRRESP	1 if detected error response to posted write; otherwise, 0	RW	0
3	Reserved	Read returns 0.	R	0
2	SBTMBUSY	1 if agent has pending commands; otherwise, 0	R	0
1	Reserved	Read returns 0.	RW	0
0	SBSERR	1 if positive edge detected on Serror; otherwise, 0	RW	0

Table 6−115. SBTMCONFIG Register

Address Offset	0x1BC		
Physical Address	0x6800 25BC	**Instance**	L4TA
Description	This register configures the operation of the clock generator and the target agent response flag use.		
Type	R		



Bits	Field Name	Description	Type	Reset
31:13	Reserved	Read returns 0.	R	0x00000
12	SBOBERRMODE	Controls out-of-band reporting of in-band OCP errors	R	1
11:9	Reserved	Read returns 0.	R	0x0
8	SBSERRMODE	Serror mapping	R	1
7:0	Reserved	Read returns 0.	R	0x00

Table 6−116. SBID Register

Address Offset	0x1F8		
Physical Address	0x6800 25F8	**Instance**	L4TA
Description	Identification		
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
SBSWREV				Reserved																											

Bits	Field Name	Description	Type	Reset
31:28	SBSWREV	SB revision code per agent 0x0: Version 2.2 and earlier 0x1: Version 3.0	R	0x1
27:0	Reserved	Read returns 0.	R	0x0000000

6.3.2.6 TM Register Descriptions

Table 6−117 lists the TM registers.

Table 6−117. TM Instance Summary

Module Name	Base Address	Size	Port
TM4	0x6800 2E00	512 bytes	DSP memory
TM1	0x6800 4100	512 bytes	SB2SMS
TM3	0x6800 4300	512 bytes	SB2OCM
TM2	0x6800 4500	512 bytes	SB2GPMC

Table 6−118 summarizes the TM registers. Replace XX in the physical address with the correct value from Table 6−117.

Table 6−118 lists the TM1-7 registers. Table 6−119 through Table 6−124 describe the register bits.

Table 6−118. TM1−7 Registers

Register Name	Type	Register Width (Bits)	Physical Address
SBTMERRLOGA	R	32	0x6800 XX48
SBTMERRLOG	RW	32	0x6800 XX50
SBTMSTATE_L	RW	32	0x6800 XX98
SBTMSTATE_H	RW	32	0x6800 XX9C
SBTMCONFIG	R	32	0x6800 XXBC
SBID	R	32	0x6800 XXF8

Table 6–119. SBTMERRLOGA Register

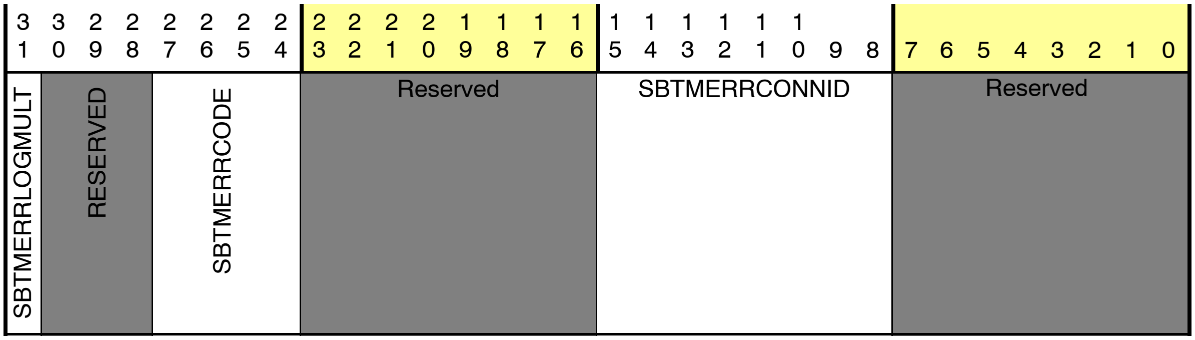
Address Offset	0x148		
Physical Address	0x6800 2F48	**Instance**	TM4
	0x6800 3748		TM7
	0x6800 3B48		TM5
	0x6800 3D48		TM6
	0x6800 4148		TM1
	0x6800 4348		TM3
	0x6800 4548		TM2
Description	Logs information about TM error conditions		
Type	R		

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
SBTMERRADDR																															

Bits	Field Name	Description	Type	Reset
31:0	SBTMERRADDR	SB address of request with error	R	0x00000000

Table 6−120. SBTMERRLOG Register

Address Offset	0x150		
Physical Address	0x6800 2F50	**Instance**	TM4
	0x6800 3750		TM7
	0x6800 3B50		TM5
	0x6800 3D50		TM6
	0x6800 4150		TM1
	0x6800 4350		TM3
	0x6800 4550		TM2
Description	Logs information about TM error conditions		
Type	RW		

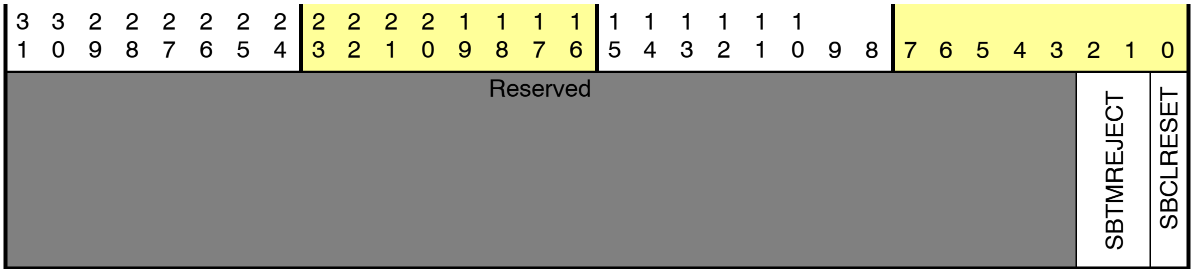


31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SBTMERRLOGMULT	RESERVED			SBTMERRCODE				Reserved								SBTMERRCONNID								Reserved							

Bits	Field Name	Description	Type	Reset
31	SBTMERRLOG-MULT	Multiple errors detected	RW	0
30:28	Reserved	Read returns 0.	R	0x0
27:24	SBTMERRCODE	Error code 0x0: No error 0x4: Error response cannot be delivered in-band	RW	0x0
23:16	Reserved	Read returns 0.	R	0x00
15:8	SBTMERR CONNID	SBConnID of request with error	R	0x−−
7:0	Reserved	Reserved	R	0x−−

Table 6−121. SBTMSTATE_L Register

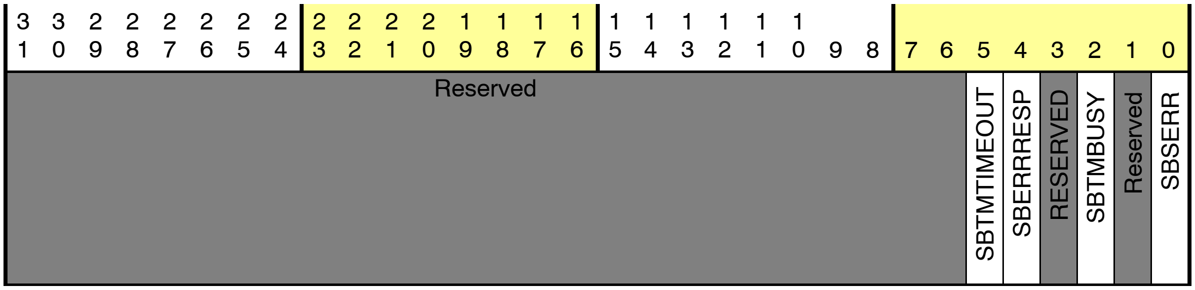
Address Offset	0x198		
Physical Address	0x6800 2F98	**Instance**	TM4
	0x6800 3798		TM7
	0x6800 3B98		TM5
	0x6800 3D98		TM6
	0x6800 4198		TM1
	0x6800 4398		TM3
	0x6800 4598		TM2
Description	This register provides access to status information and target agent control values − LSBs.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:3	Reserved	Read returns 0.	R	0x00000000
2:1	SBTMREJECT	Controls rejection of new SB requests 0x0: Normal behavior 0x1: Included for backwards compatibility 0x2: Return BUSY response for all non-register requests that start a new burst. Requests within a burst are treated normally.	RW	0x0
0	SBCLRESET	Clear transactions and reset IW or TW.	RW	0

Table 6−122. SBTMSTATE_H Register

Address Offset	0x19C		
Physical Address	0x6800 2F9C	**Instance**	TM4
	0x6800 379C		TM7
	0x6800 3B9C		TM5
	0x6800 3D9C		TM6
	0x6800 419C		TM1
	0x6800 439C		TM3
	0x6800 459C		TM2
Description	This register provides access to status information and target agent control values − MSBs.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:6	Reserved	Read returns 0.	R	0x0000000
5	SBTMTIMEOUT	1 if request has timed out; otherwise, 0	R	0
4	SBERRRESP	1 if detected error response to posted write; otherwise, 0	RW	0
3	Reserved	Read returns 0.	R	0
2	SBTMBUSY	1 if agent has pending commands; otherwise, 0	R	0
1	Reserved	Read returns 0.	RW	0
0	SBSERR	1 if positive edge detected on Serror; otherwise, 0	RW	0

Table 6−123. SBTMCONFIG Register

Address Offset	0x1BC		
Physical Address	0x6800 2FBC	**Instance**	TM4
	0x6800 37BC		TM7
	0x6800 3BBC		TM5
	0x6800 3DBC		TM6
	0x6800 41BC		TM1
	0x6800 43BC		TM3
	0x6800 45BC		TM2
Description	This register configures the operation of the clock generator and the target agent response flag use.		
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																			SBOBERRMODE	RESERVED		RESERVED		Reserved							

Bits	Field Name	Description	Type	Reset
31:13	Reserved	Read returns 0.	R	0x00000
12	SBOBERRMODE	Controls out of band reporting of in-band OCP errors	R	1
11:10	Reserved	Read returns 0.	R	0x0
9:8	Reserved	Reserved	R	0x−
7:0	Reserved	Reserved	R	0x−−

Table 6–124. SBID Register

Address Offset	0x1F8		
Physical Address	0x6800 2FF8	**Instance**	TM4
	0x6800 37F8		TM7
	0x6800 3BF8		TM5
	0x6800 3DF8		TM6
	0x6800 41F8		TM1
	0x6800 43F8		TM3
	0x6800 45F8		TM2
Description	Identification – Lower 32 –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
SBSWREV				Reserved																											

Bits	Field Name	Description	Type	Reset
31:28	SBSWREV	SB revision code per agent 0x0: Version 2.2 and earlier 0x1: Version 3.0	R	0x1
27:0	Reserved	Reserved	R	0x-------

6.3.2.7 SMS Register Descriptions

Table 6−125 lists the SMSFW registers. Table 6−126 through Table 6−130 describe the register bits.

Table 6−125. SMSFW Registers

Register Name	Type	Register Width (Bits)	Physical Address
SMS_RG_START0 – SMS_RG_START3	RW	32	0x6800 8040– 0x6800 8070
SMS_RG_END0 – SMS_RG_END3	RW	32	0x6800 8044– 0x6800 8074
SMS_RG_ATT0 – SMS_RG_ATT3	RW	32	0x6800 8048– 0x6800 8078
SMS_RG_WRPERM0 – SMS_RG_WRPERM3	RW	32	0x6800 8080– 0x6800 80B0
SMS_RG_RDPERM0 – SMS_RG_RDPERM3	RW	32	0x6800 8084– 0x6800 80B4
SMS_REGS_SECURITY	RW	32	0x6800 80C0

Table 6−126. SMS_RG_START0–SMS_RG_START3 Registers

Address Offset	0x0040–0x0070 in 0x10 byte increments		
Physical Address	0x6800 8040–0x6800 8070	**Instance**	SMSFW
Description	This register provides the region i start address (lowest address in the region), with a 64KB 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 0 for future compatibility. Read returns 0.	RW	0
30:16	STARTADDRESS	Region i start address (included in the region). Aligned on 64KB boundary. [15:0] must be written with 0s.	RW	0x0000
15:0	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x0000

Table 6−127. SMS_RG_END0−SMS_RG_END3 Registers

Address Offset	0x0044−0x0074 in 0x10 byte increments		
Physical Address	0x6800 8044−0x6800 8074	**Instance**	SMSFW
Description	This register provides the region i end address (lowest address outside the region), with a 64KB 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 0 for future compatibility. Read returns 0.	RW	0
30:16	ENDADDRESS	Region i end address (not included in the region) Aligned on 64KB boundary. [15:0] must be written with 0s.	RW	0x0000
15:0	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x0000

Table 6−128. SMS_RG_ATT0−SMS_RG_ATT3 Registers

Address Offset	0x0048−0x0078 in 0x10 byte increments		
Physical Address	0x6800 8048−0x6800 8078	**Instance**	SMSFW
Description	This register provides the protection attributes for region i. Can be programmed only by a secure transaction.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								REQPERM							

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x000000
7:4	REQPERM	Region access permission with respect to request qualifiers (secure transaction and debug transaction)	RW	0xF
3:0				

Table 6−129. SMS_RG_WRPERM0−SMS_RG_WRPERM3 Registers

Address Offset	0x0080−0x00B0 in 0x10 byte increments		
Physical Address	0x6800 8080−0x6800 80B0	**Instance**	SMSFW
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 0 for future compatibility. Read returns 0.	RW	0x0000
15:0	CONNIDVECTOR	Vector providing read permission to up to 16 initiators. Any nonauthorized access generates a security violation. For bit[k], k = 0 to 15: 0: Initiator (ConnID=k) has read permission. 1: Initiator (ConnID=k) has write permission.	RW	0x0000

Table 6−130. SMS_RG_RDPERM0−SMS_RG_RDPERM3 Registers

Address Offset	0x0084−0x00B4 in 0x10 byte increments		
Physical Address	0x6800 8084−0x6800 80B4	**Instance**	SMSFW
Description	This register provides the list of all initiators that have permission to read 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 0 for future compatibility. Read returns 0.	RW	0x0000
15:0	CONNIDVECTOR	Vector providing write permission to up to 16 initiators. Any unauthorized access generates a security violation. For bit[k], k = 0 to 15: 0: Initiator (ConnID = k) has write permission. 1: Initiator (ConnID = k) has read permission.	RW	0x0000

6.4 L4 Interconnect

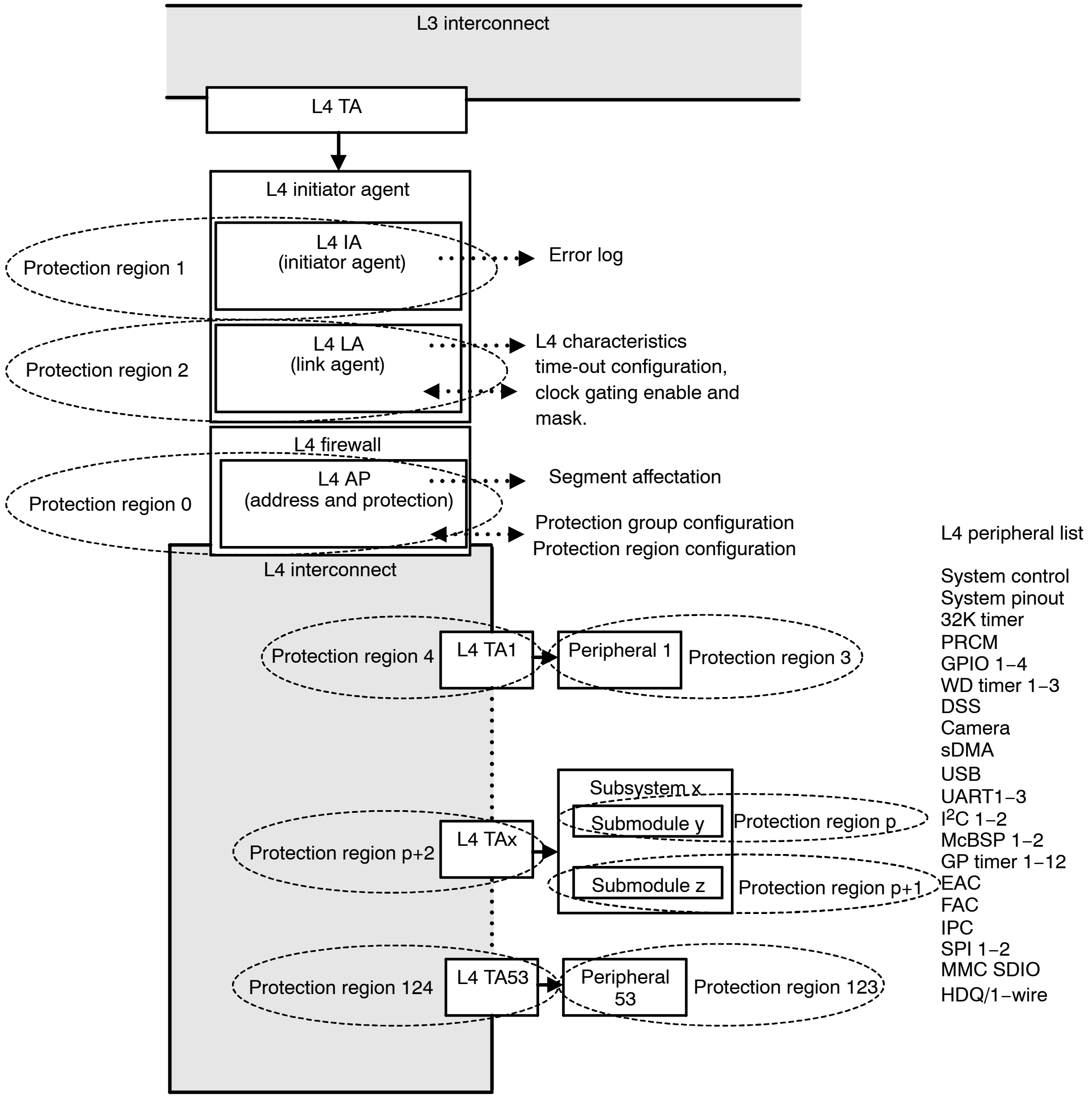
6.4.1 L4 Interconnect Overview

L4 interconnect handles only transfers to target modules; it is a 32-bit peripheral interconnect. There is only one initiator agent connected to the L3 SB interconnect; this initiator agent is required in all transfers because it is the only L4 interconnect initiator.

This backplane is optimized to support the interconnection of a high number of peripherals (44 peripherals are connected in the OMAP2420). The L4 interconnect assumes a little-endian transaction for narrower targets (8 bits and 16 bits) when performing data packing and unpacking.

This architecture, with only one initiator module distributing transactions to all target modules, allows L4 interconnect firewall functionalities to be centralized at the L4 initiator level. The L4 firewall filters the accesses with respect to one of the height−configurable protection groups defined in the L4 AP for each target module and target agent. Each module, submodule, or agent that belongs to a different protection region can be protected by one of the height−protection groups.

Figure 6−14. L4 interconnect Block Diagram



6.4.2 Port List and Description

6.4.2.1 Initiator Agent

Table 6−131. L4 Initiator Agent Regions

Initiator Agent				
Region Name	**Region ID**	**Composite Flag Mask Index**	**Segment**	**Bus Width**
Initiator agent	1	0	2	32
Link agent	2			32
Address and protection	0			32

6.4.2.2 Target Module/Agent

L4 Target Agent Connection to Peripheral Overview

There are 53 different L4 interconnect target agents divided in two types: L4TA (39 instances) and L4TAO (14 instances).

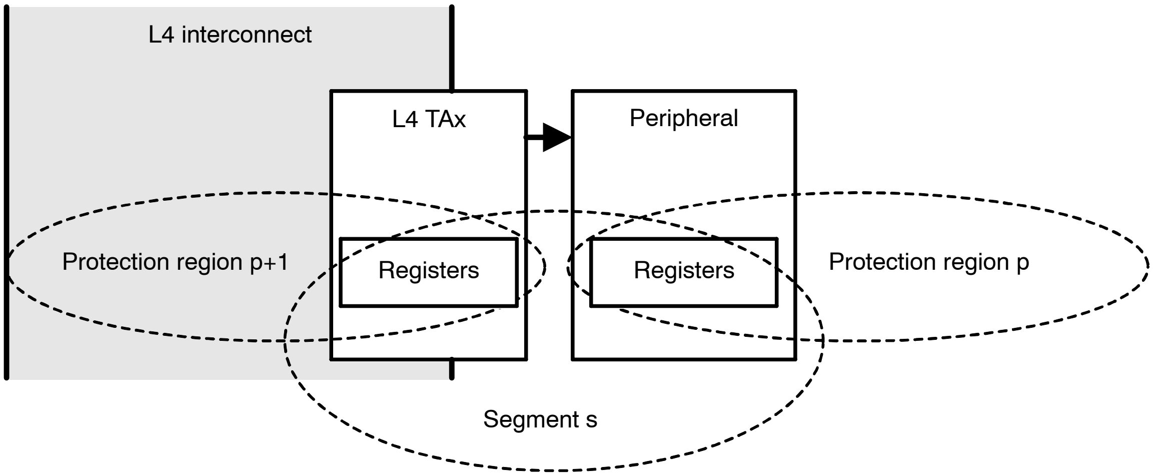
L4TA and L4TAO are functionally equivalent, except for error reporting (see Section 6.4.4.6, *Error Management in the L4 Interconnect*). The only difference is the value of the read-only register bit, AGENT_CONTROL(24): SERROR_REP register field (1 for each L4TAO and 0 for each L4TA).

44 of the 53 target agents are connected to functional peripherals; the other 9 are reserved for emulation and testing.

Each L4 target agent corresponds to one L4 interconnect port. A port can be connected to one module or one subsystem. One subsystem contains several modules. Each module in the subsystem belongs to a different protection region. One module address space can be divided into different protection regions. There are three possible configurations:

1) Basic connection between target agent and peripheral with one protection region (see Figure 6−15)

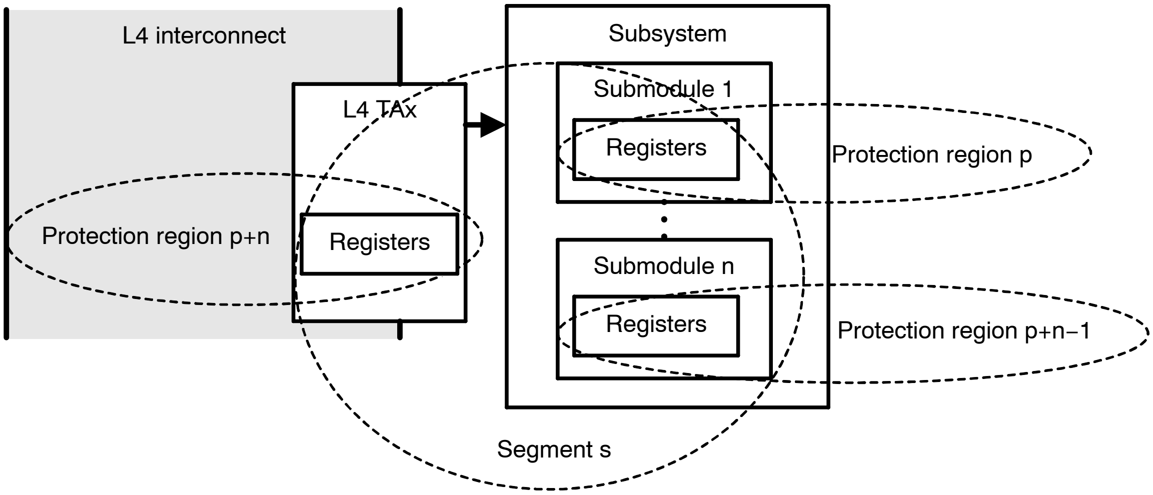
Figure 6−15. Connection Between TA and Peripheral With One Protection Region



Most L4 peripherals are interconnected this way: basic connection with one protection region used for the peripheral and the following protection region used for its TA.

2) Connection of target agent with a subsystem (see Figure 6−16)

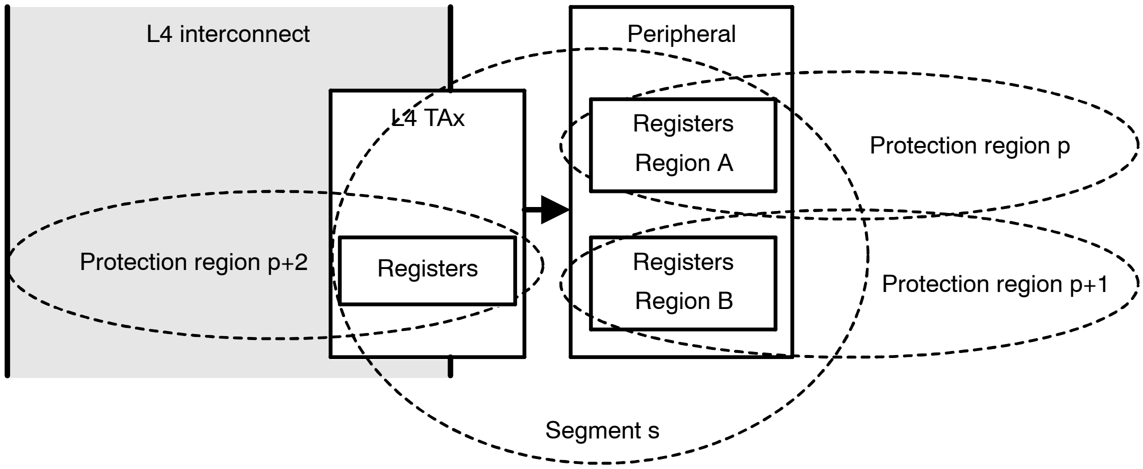
Figure 6−16. Connection Between TA and Subsystem



Subsystem example: QuadGPIO, dual WD timer, DSS, camera

3) Connection between target agent and one module with multiple protection regions (see Figure 6−17)

Figure 6−17. Connection Between TA and Peripheral With Multiple Protection Regions



L4 Interconnect Ports and Region Distribution

The following tables show all correspondence between modules, target agents, and protection regions port-by-port. (Region ID is the protection region identifier.)

Note:

A target module and its target agent are always mapped on the same segment.

The Allowed Access column shows the possible data size access allowed to peripherals. (Other data sizes result in error generation or bad behavior. See each module for more information.)

Table 6−132. System Control Module and Pinout Port: Basic Connection to Peripheral with One Protection Region

Region Name	Region ID	Composite Flag Mask Index	Segment	Allowed Access
System control and pinout	3	1	0	32/16/8
L4TAO1	4			32/16/8

Table 6−133. 32K Timer Port: Basic Connection to Peripheral with One Protection Region

Region Name	Region ID	Composite Flag Mask Index	Segment	Allowed Access
32K Timer	5	2	0	32/16
L4TAO2	6			32/16/8

Table 6−134. Power Reset and Clock Management Port: Peripheral With Multiple Protection Regions

Region Name	Region ID	Composite Flag Mask Index	Segment	Allowed Access
PRCM Region A	7	3	0	32
PRCM Region B	8			32
L4TAO	9			32/16/8

Reserved for testing.

Table 6−135. GPIO 1/2/3/4 Port: Subsystem

Region Name	Region ID	Composite Flag Mask Index	Segment	Allowed Access
GPIO1	14	6	0	32/16/8
GPIO2	16			32/16/8
GPIO3	18			32/16/8
GPIO4	19			32/16/8
Quad GPIO Top	15			32/16/8
L4TA3	17			32/16/8

Table 6−136. WD Timer 1/2 Port: Subsystem

Region Name	Region ID	Composite Flag Mask Index	Segment	Allowed Access
WD Timer 2 (OMAP)	22	7	1	32/16/8
Dual WD timer TOP	21			32/16/8
L4TA4	23			32/16/8

Table 6−137. GP Timer 1 Port: Basic Connection to Peripheral with One Protection Region

Region Name	Region ID	Composite Flag Mask Index	Segment	Allowed Access
GP Timer 1	24	8	1	32/16/8
L4 TA7	25			32/16/8

Table 6−138. Display SubSystem Port: Subsystem

Region Name	Region ID	Composite Flag Mask Index	Segment	Allowed Access
Display top	28	10	2	32/16/8
Display control	29			32/16/8
Display RFBI	30			32/16/8
Display encoder	31			32/16/8
L4TA10	32			32/16/8

Table 6−139. Camera Subsystem Port: Subsystem

Region Name	Region ID	Composite Flag Mask Index	Segment	Allowed Access
Camera top	33	11	2	32/16/8
Camera core	34			32/16/8
Camera DMA	35			32/16/8
Camera MMU	36			32/16/8
L4TA11	37			32/16/8

Table 6−140. System DMA Port: Subsystem

Region Name	Region ID	Composite Flag Mask Index	Segment	Allowed Access
sDMA	38	12	2	32/16/8
L4TA12	39			32/16/8

Table 6−141. USB OTG Port: Basic Connection to Peripheral with One Protection Region

Region Name	Region ID	Composite Flag Mask Index	Segment	Allowed Access
USB OTG	45	14	2	32/16/8
L4TAO4	46			32/16/8

Reserved for testing.

Table 6−142. UART 1 Port: Basic Connection to Peripheral with One Protection Region

Region Name	Region ID	Composite Flag Mask Index	Segment	Allowed Access
UART1	57	20	3	16/8
L4TA19	58			32/16/8

Table 6−143. UART 2 Port: Basic Connection to Peripheral with One Protection Region

Region Name	Region ID	Composite Flag Mask Index	Segment	Allowed Access
UART2	59	21	3	16/8
L4TA20	60			32/16/8

Table 6−144. UART 3 Port: Basic Connection to Peripheral with One Protection Region

Region Name	Region ID	Composite Flag Mask Index	Segment	Allowed Access
UART3	61	22	3	16/8
L4TA21	62			32/16/8

Table 6−145. I2C1 Port: Basic Connection to Peripheral with One Protection Region

Region Name	Region ID	Composite Flag Mask Index	Segment	Allowed Access
I2C1	63	23	3	16
L4TAO5	64			32/16/8

Table 6−146. I2C2 Port: Basic Connection to Peripheral with One Protection Region

Region Name	Region ID	Composite Flag Mask Index	Segment	Allowed Access
I2C2	65	24	3	16
L4TAO6	66			32/16/8

Table 6−147. McBSP1 Port: Basic Connection to Peripheral with One Protection Region

Region Name	Region ID	Composite Flag Mask Index	Segment	Allowed Access
McBSP1	67	25	3	16
L4TAO7	68			32/16/8

Table 6−148. McBSP2 Port: Basic Connection to Peripheral with One Protection Region

Region Name	Region ID	Composite Flag Mask Index	Segment	Allowed Access
McBSP2	69	26	3	16
L4TAO8	70			32/16/8

Table 6−149. Watchdog Timer 3 (DSP) Port: Basic Connection to Peripheral with One Protection Region

Region Name	Region ID	Composite Flag Mask Index	Segment	Allowed Access
WD Timer 3	71	27	1	32/16/8
L4TA5	72			32/16/8
L4TA6	74	28	1	32/16/8

Table 6−150. General-Purpose Timer 2 Port: Basic Connection to Peripheral with One Protection Region

Region Name	Region ID	Composite Flag Mask Index	Segment	Allowed Access
GPTimer2	75	29	1	32/16/8
L4TA8	76			32/16/8

Table 6−151. General-Purpose Timer 3, 4, 5, 6 Four Ports: Basic Connection to Peripheral with One Protection Region

Region Name	Region ID	Composite Flag Mask Index	Segment	Allowed Access
GPTimer 3, 4, 5, 6,	77,79,81,83,	30, 31, 32, 33	3	32/16/8
L4TA22,23,24, 25,	78,80,82,84,			32/16/8

Table 6−152. General-Purpose Timer 7, 8, 9, 10, 11, 12 Six Ports: Basic Connection to Peripheral with One Protection Region

Region Name	Region ID	Composite Flag Mask Index	Segment	Allowed Access
GPTimer 7,8,9,10, 11,12	85,87,89,91,93,95	34, 35, 36, 37, 38, 39	4	32/16/8
L4TA26,27,28,29 30,31	86,88,90,92,94,96			32/16/8

Table 6−153. Audio Application: Enhanced Audio Controller Port: Basic Connection to Peripheral with One Protection Region

Region Name	Region ID	Composite Flag Mask Index	Segment	Allowed Access
EAC	97	40	4	16
L4TA32	98			32/16/8

Table 6−154. Frame Adjustment Counter: FAC Port: Basic Connection to Peripheral with One Protection Region

Region Name	Region ID	Composite Flag Mask Index	Segment	Allowed Access
FAC	99	41	4	16
L4TA33	100			32/16/8

Table 6−155. Interprocessor Communication: IPC Port: Basic Connection to Peripheral with One Protection Region

Region Name	Region ID	Composite Flag Mask Index	Segment	Allowed Access
IPC	101	42	4	32/16/8
L4TA34	102			32/16/8

Table 6−156. SPI1 Port: Basic Connection to Peripheral with One Protection Region

Region Name	Region ID	Composite Flag Mask Index	Segment	Allowed Access
SPI1	103	43	4	32/16/8
L4TA35	104			32/16/8

Table 6−157. SPI2 Port: Basic Connection to Peripheral with One Protection Region

Region Name	Region ID	Composite Flag Mask Index	Segment	Allowed Access
SPI2	105	44	4	32/16/8
L4TA36	106			32/16/8

Table 6−158. MMC SDIO Port: Basic Connection to Peripheral with One Protection Region

Region Name	Region ID	Composite Flag Mask Index	Segment	Allowed Access
MMC SDIO	107	45	4	16/8
L4TAO9	108			32/16/8

Table 6−159. Reserved

Region Name	Region ID	Composite Flag Mask Index	Segment	Allowed Access
Reserved	109	NA	NA	NA
Reserved	110			NA

Table 6−160. Reserved

Region Name	Region ID	Composite Flag Mask Index	Segment	Allowed Access
Reserved	121	NA	NA	32
Reserved	122			32

Table 6−161. HDQ/1-Wire Port: Basic Connection to Peripheral with One Protection Region

Region Name	Region ID	Composite Flag Mask Index	Segment	Allowed Access
HDQ/1-Wire	123	52	5	32
L4TA39	124			32/16/8

6.4.3 L4 Interconnect Memory Mapping

6.4.3.1 L4 Interconnect Segmentation

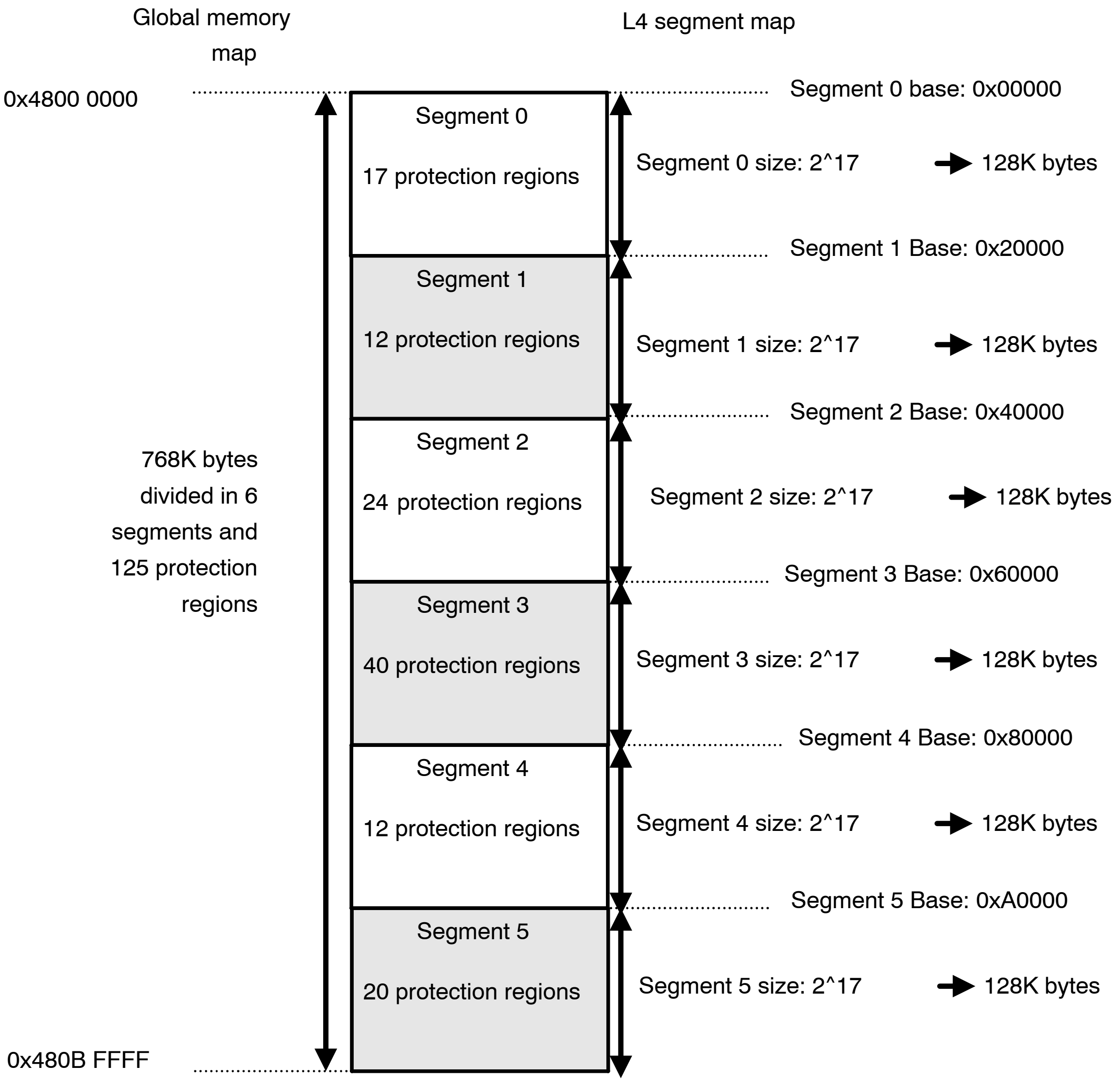
The L4 interconnect address space is 768K bytes wide and starts at physical address 0x4800 0000. It is divided into 6 different contiguous segments of 128K bytes. Each segment is defined by its base address and size in the L4AP register:

- Base: SEGMENT_L[23:0] gives the base address in L4 interconnect mapping. To obtain the real physical address at chip level, add 0x4800 0000 to BASE.

- Size: SEGMENT_H[4:0]. To obtain the segment size in bytes, register field SIZE must be considered as a power of two: 2 ^ SIZE = Segment size in bytes. All segments have identical size: Register field SIZE = 0x11, consecutively, segment size is 2^17 = 128K bytes.

The subdivision of the L4 interconnect address space is a consequence of the implementation, but no functionality is linked to the segments. All segments are equivalent and have no particular properties. Each protection region base address L4AP REGION_X_L[23:0] BASE (with X between 0 and 124) register field is the base address of the protection region within the segment.

Figure 6−18. L4 Interconnect Segment Definition



6.4.3.2 L4 Interconnect Detailed Memory Mapping

Table 6−162. L4 Interconnect Detailed Memory Mapping

Device Name	Start Address (hex)	Size	Description	Segment
OMAP2420 system control	0x4800 0000	4K bytes	Module	0
	0x4800 1000	4K bytes	L4 interconnect	0
Reserved	0x4800 2000	8K bytes		0
32KTIMER	0x4800 4000	4K bytes	Module	0
	0x4800 5000	4K bytes	L4 interconnect	0

1) Reserved area for MPU interrupt module. Accesses to the MPU interrupt module registers at this memory space are conditional to suitable ARM1136 CP15 initialization. See Chapter 3, *MPU Subsystem*. Module mINTC is not connected to the L4 interconnect, but is on a private MPU subsystem interconnect.

Table 6−162. L4 Interconnect Detailed Memory Mapping (Continued)

Device Name	Start Address (hex)	Size	Description	Segment
Reserved	0x4800 6000	8K bytes		0
PRCM: – DPLL – Clock manager – Power manager	0x4800 8000	2K bytes	Module region A	0
	0x4800 8800	2K bytes	Module region B	0
	0x4800 9000	4K bytes	L4 interconnect	0
Reserved	0x4800 A000	56K bytes		0
Quad GPIO: – GPIO1 – GPIO2 – GPIO3 – GPIO4	0x4801 8000	4K bytes	GPIO1 module	0
	0x4801 9000	4K bytes	Quad GPIO top (OCP splitter)	0
	0x4801 A000	4K bytes	GPIO2 module	0
	0x4801 B000	4K bytes	Quad GPIO L4 interconnect	0
	0x4801 C000	4K bytes	GPIO3 module	0
	0x4801 D000	4K bytes	Reserved	0
	0x4801 E000	4K bytes	GPIO4 module	0
	0x4801 F000	8K bytes	Reserved	0
Dual – WDTIMER2(OMAP)	0x4802 1000	4K bytes	Dual WDTIMER top (OCP splitter)	1
	0x4802 2000	4K bytes	WDTIMER 2 module	1
	0x4802 3000	4K bytes	Dual WDTIMER L4 interconnect	1
WDTIMER3(DSP)	0x4802 4000	4K bytes	Module	1
	0x4802 5000	4K bytes	L4 interconnect	1
Reserved	0x4802 6000	8K bytes	Reserved	1

1) Reserved area for MPU interrupt module. Accesses to the MPU interrupt module registers at this memory space are conditional to suitable ARM1136 CP15 initialization. See Chapter 3, *MPU Subsystem*. Module mINTC is not connected to the L4 interconnect, but is on a private MPU subsystem interconnect.

Table 6−162. L4 Interconnect Detailed Memory Mapping (Continued)

Device Name	Start Address (hex)	Size	Description	Segment
GPTIMER1	0x4802 8000	4K bytes	Module	1
	0x4802 9000	4K bytes	L4 interconnect	1
GPTIMER2	0x4802 A000	4K bytes	Module	1
	0x4802 B000	4K bytes	L4 interconnect	1
Reserved	0x4802 C000	80K bytes		1
L4-Configuration	0x4804 0000	2K bytes	Address/Protection (AP)	2
	0x4804 0800	2K bytes	Initiator Port (AP)	2
	0x4804 1000	4K bytes	Link Agent (LA)	2
Reserved	0x4804 2000	56K bytes		2
DISPLAY subsystem	0x4805 0000	1K bytes	Display subsystem top	2
	0x4805 0400	1K bytes	Display controller (DISP)	2
	0x4805 0800	1K bytes	Remote Frame Buffer Interface (RFBI)	2
	0x4805 0C00	1K bytes	Video encoder (VENC)	2
	0x4805 1000	4K bytes	L4 interconnect	2
CAMERA	0x4805 2000	1K bytes	Camera top	2
	0x4805 2400	1K bytes	Camera core	2
	0x4805 2800	1K bytes	Camera DMA	2
	0x4805 2C00	1K bytes	Camera MMU	2
	0x4805 3000	4K bytes	L4 interconnect	2
Reserved	0x4805 4000	24K bytes		2
SDMA	0x4805 6000	4K bytes	Module (L3)	2
	0x4805 7000	4K bytes	L4 interconnect	2
Reserved	0x4805 8000	24K bytes		2
USB	0x4805 E000	4K bytes	Module (L3)	2
	0x4805 F000	4K bytes	L4 interconnect	2
Reserved	0x4806 0000	40K bytes		2
UART1	0x4806 A000	4K bytes	Module	3
	0x4806 B000	4K bytes	L4 interconnect	3
UART2	0x4806 C000	4K bytes	Module	3
	0x4806 D000	4K bytes	L4 interconnect	3

1) Reserved area for MPU interrupt module. Accesses to the MPU interrupt module registers at this memory space are conditional to suitable ARM1136 CP15 initialization. See Chapter 3, *MPU Subsystem*. Module mINTC is not connected to the L4 interconnect, but is on a private MPU subsystem interconnect.

Table 6−162. L4 Interconnect Detailed Memory Mapping (Continued)

Device Name	Start Address (hex)	Size	Description	Segment
UART3 (with infrared)	0x4806 E000	4K bytes	Module	3
	0x4806 F000	4K bytes	L4 interconnect	3
I2C1	0x4807 0000	4K bytes	Module	3
	0x4807 1000	4K bytes	L4 interconnect	3
I2C2	0x4807 2000	4K bytes	Module	3
	0x4807 3000	4K bytes	L4 interconnect	3
McBSP1	0x4807 4000	4K bytes	Module	3
	0x4807 5000	4K bytes	L4 interconnect	3
McBSP2	0x4807 6000	4K bytes	Module	3
	0x4807 7000	4K bytes	L4 interconnect	3
GPTIMER3	0x4807 8000	4K bytes	Module	3
	0x4807 9000	4K bytes	L4 interconnect	3
GPTIMER4	0x4807 A000	4K bytes	Module	3
	0x4807 B000	4K bytes	L4 interconnect	3
GPTIMER5	0x4807 C000	4K bytes	Module	3
	0x4807 D000	4K bytes	L4 interconnect	3
GPTIMER6	0x4807 E000	4K bytes	Module	3
	0x4807 F000	4K bytes	L4 interconnect	3
GPTIMER7	0x4808 0000	4K bytes	Module	4
	0x4808 1000	4K bytes	L4 interconnect	4
GPTIMER8	0x4808 2000	4K bytes	Module	4
	0x4808 3000	4K bytes	L4 interconnect	4
GPTIMER9	0x4808 4000	4K bytes	Module	4
	0x4808 5000	4K bytes	L4 interconnect	4
GPTIMER10	0x4808 6000	4K bytes	Module	4
	0x4808 7000	4K bytes	L4 interconnect	4
GPTIMER11	0x4808 8000	4K bytes	Module	4
	0x4808 9000	4K bytes	L4 interconnect	4
GPTIMER12	0x4808 A000	4K bytes	Module	4
	0x4808 B000	4K bytes	L4 interconnect	4
Reserved	0x4808 C000	16K bytes		4

1) Reserved area for MPU interrupt module. Accesses to the MPU interrupt module registers at this memory space are conditional to suitable ARM1136 CP15 initialization. See Chapter 3, *MPU Subsystem*. Module mINTC is not connected to the L4 interconnect, but is on a private MPU subsystem interconnect.

Table 6−162. L4 Interconnect Detailed Memory Mapping (Continued)

Device Name	Start Address (hex)	Size	Description	Segment
EAC	0x4809 0000	4K bytes	Module	4
	0x4809 1000	4K bytes	L4 interconnect	4
FAC	0x4809 2000	4K bytes	Module	4
	0x4809 3000	4K bytes	L4 interconnect	4
MAILBOX	0x4809 4000	4K bytes	Module	4
	0x4809 5000	4K bytes	L4 interconnect	4
Reserved	0x4809 6000	8K bytes		4
SPI1	0x4809 8000	4K bytes	Module	4
	0x4809 9000	4K bytes	L4 interconnect	4
SPI2	0x4809 A000	4K bytes	Module	4
	0x4809 B000	4K bytes	L4 interconnect	4
MMC/SDIO	0x4809 C000	4K bytes	Module	4
	0x4809 D000	4K bytes	L4 interconnect	4
Reserved	0x4809 E000	80K bytes		5
HDQ (1 wire)	0x480B 2000	4K bytes	Module	5
	0x480B 3000	4K bytes	L4 interconnect	5
Reserved	0x480B 4000	296K bytes		5
MPU interrupt (mINT)[1]	0x480F E000	4K bytes		5

[1] Reserved area for MPU interrupt module. Accesses to the MPU interrupt module registers at this memory space are conditional to suitable ARM1136 CP15 initialization. See Chapter 3, *MPU Subsystem*. Module mINTC is not connected to the L4 interconnect, but is on a private MPU subsystem interconnect.

6.4.4 L4 Interconnect Functional Description

6.4.4.1 Initiator Identification in L4 Interconnect

A CONNID is an initiator module identifier. The L4 interconnect uses specific CONNIDs derived from the SystemCONNID used in the L3 interconnect. The same initiator module is referenced by two different CONNIDs, depending on whether access is performed with the secure attribute.

- ☐ L4CONNID: This 4-bit CONNID is used in L4 interconnect only for all transactions that do not have the secure attribute. This L4CONNID is a subset of systemCONNID (3 lower bits), and the fourth MSB bit is always 0.

Table 6−163. L4 Firewall InitiatorID Description

Initiator	Initiator Sub	L4 Access Possible	L4 InitiatorID
MPU	MPU data RD	Yes	1
	MPU data WR		
	MPU instruction		
LCD	LCD	No	
Camera	Camera	No	
	MMU	No	
DSP DMA	DSP DMA RD	Yes	2
USB	USB	No	
System DMA	System DMA RD	Yes	3
	System DMA WR		
DSP	DSP Data	Yes	7
	DSP Instruction		
	DSP MMU		

6.4.4.2 CONNID and CONNID_Bit_Vector of Protection_Group

CONNID corresponds to a number that identifies the initiator.

CONNID_Bit_Vector is a 1-bit vector that identifies read and write permissions for one initiator module to access one target that belongs to one protection_ group. It is possible to set a maximum of eight different protection groups (0 to 7) in the L4 interconnect. Read and write permissions are identical for read and write accesses in L4 interconnect targets. A target is accessible by an initiator if the bit position corresponding to the CONNID number is set at 1 in the CONNID bit vector. The protection group can be programmed by accessing the L4AP PROTGROUP_X[15:0] CONNIDBITVECTOR register field.

Example:

The protection group 1 setting enables access permission for the initiator module with CONNID 3, 9, or 11(0xB). Table 6−164 provides the correspondence between CONNID and the initiator modules, respectively: sDMA, MPU with the secure attribute, and sDMA with the secure attribute.

Protection group 1 can be applied to multiple protection regions. Each protection region X that is configured with protection group 1 enables permission access for these three initiators only: REGION_X_H[22:20] PROT_GROUP_ID register field value set to 001.

One protection group can be applied to several protection regions (no limitation). A protection group can be changed by software at any time.

One protection region applying to TM or TA is protected by one and only one protection group at a given time (the software can change the protection group of a protection region by modifying the PROT_GROUP_ID register field at any time).

6.4.4.3 Burst Support

The L4 interconnect does not limit burst support.

6.4.4.4 Endianness Management

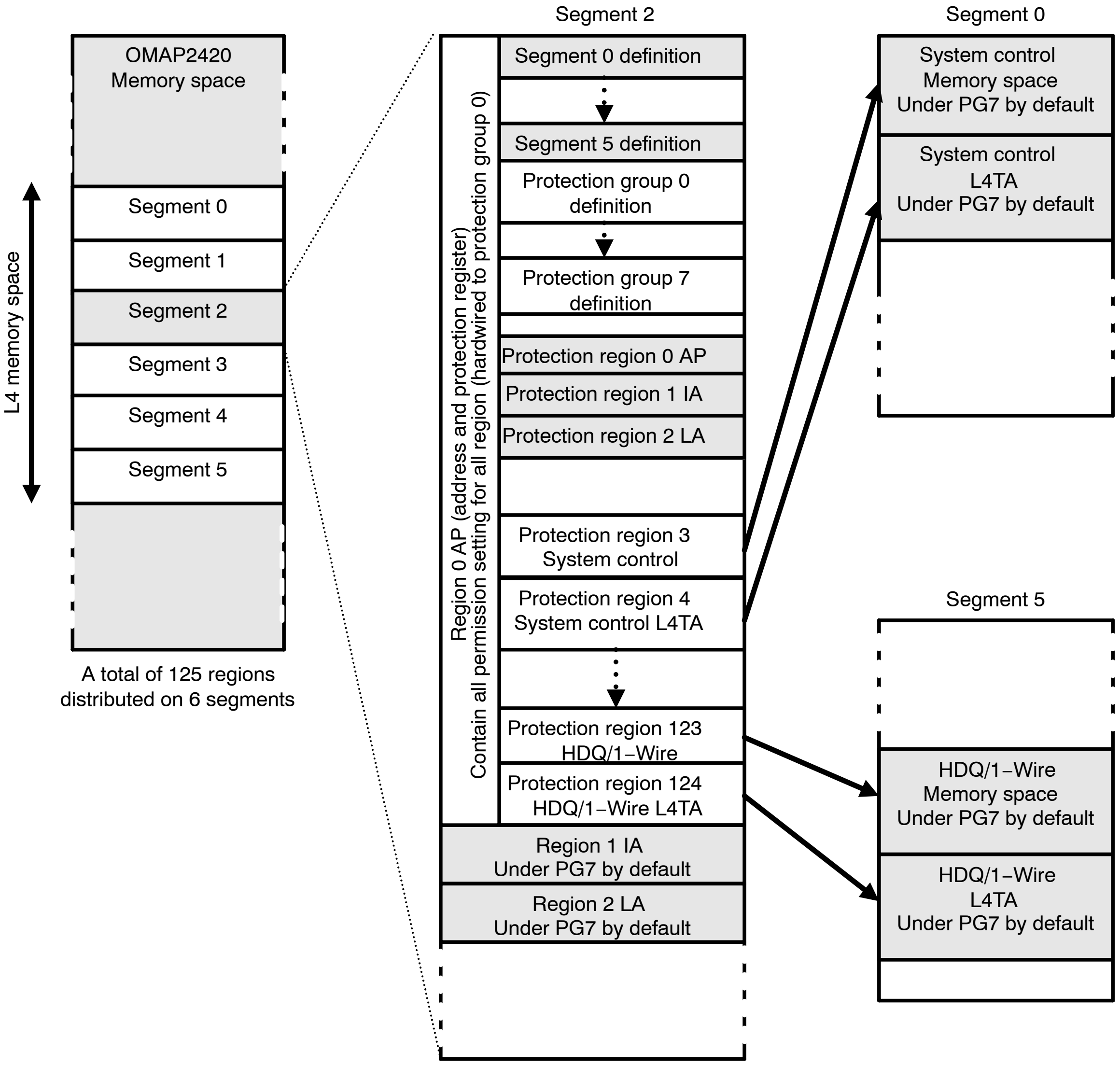
L4 interconnect is little-endian only. Any initiator accessing the L4 interconnect module must consider byte ordering and perform a conversion, if necessary.

6.4.4.5 Firewall

Because of the high number of peripherals connected to the L4 interconnect, the L4 firewall uses groups of initiators. Eight protection groups can be defined:

- ☐ A protection group is a list of initiators, described in the CONNID_bit_vector, having the same access permission to all regions referring to that protection group.
- ☐ An initiator is identified during an access by the L4initiatorID.
- ☐ The protection group is programmable.
- ☐ At a given time, any initiator can belong to several protection groups.
- ☐ A target region i refers to a protection group using the protection region i register, defined in the L4AP module.
- ☐ At a given time, a target region can refer to a unique protection group that is programmable for all regions.

Figure 6−19. L4 Firewall Functional Overview



L4 Firewall Default Configuration

All protection regions except protection region 0 are by default set with protection group 7 (PG7), which is configured to enable all access. Protection region 0 corresponding to L4AP (address and protection) is by default under protection group 0 (PG0), which is MPU access only.

L4 Firewall Address and Protection (AP) Registers Setting

Table 6−164. L4 Firewall Implementation Overview

Register Type	Address Offset	Register	Bits	Field	Reset Value	R/W	Description
Segment	0x100…0x128	SEGMENT_ (0:6)_L	23:0	BASE	–	RO	Segment base address
		SEGMENT_ (0:6)_H	4:0	SIZE	0x11	RO	Segment size
Protection groups	0x200	PROT GROUP_(0)	15:0	CONNID BITVECTOR	0x0202	R/W	**Initiator** — **L4 Initiator ID** ARM1136 — 1 dDMA — 2 sDMA — 3 Reserved — 4 Reserved — 6 DSP — 7 Reserved — 9 Reserved — B CONNIDBITVECTOR (L4InitiatorID) = 1: Access permission CONNIDBITVECTOR (L4InitiatorID) = 0: Access forbidden
Protection groups	0x208…0x238	PROT GROUP_(1:7)	15:0	CONNID BITVECTOR	0xFFFF	R/W	Configuration is the same as for protection group 0.
Region	0x300…0x6E8	REGION_ (0:124)_H	31:28	MADDR SPACE	[1]	RO	Target Region MaddrSpace value (unused)
			27:24	SEGMENT_ ID	[1]	RO	Segment ID of the region
			22:20	PROT_ GROUP_ID	[1]	R/W	Protection group ID
			19:17	BYTE_DATA_ WIDTH	0x2	RO	Target region data byte width (always 32 bits)
			14:8	PHY_ target_ID	––	RO	Physical target ID (unused)
			5:1	SIZE	[1]	RO	Region size
			0	ENABLE	0x1	R/W	Enable access to the region
		REGION_ (0:124)_L	23:0	BASE	[1]	RO	Region base address within segment

1) See Section 6.5, L4 *Interconnect Registers*, for the reset value of each field, depending on the region.

Example of Firewall Programming

Firewall configuration requires accesses to the AP region (region 0); only the MPU can access this region.

6.4.4.6 Error Management in the L4 Interconnect

General Mechanisms

The L4 interconnect provides mechanisms for handling either internally detected errors or errors reported by modules attached to L4 target ports. Hardware support facilitates error logging and clean-up of state to allow error−recovery software to treat the errors.

As an L3 target, when possible, the L4 interconnect reports errors to the L3 interconnect in-band. In-band error reporting is the default and recommended configuration. Therefore, all this section assumes that the INBAND_ERROR_REP bit in the L4IA AGENT_CONTROL register is set to 1.

Errors Detected Internally by L4 Interconnect Logic

- No target core found

  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 in the L4IA AGENT_STATUS(27): INBAND_ERROR register bit.

  An address hole error code is also logged in the ERROR_LOG(25:24): CODE register field.

  The ERROR_LOG register also includes a bitfield named ERROR_LOG(31): MULTI, which is asserted when multiple errors are detected. In this case, the error code corresponds to the first error that occurred.

- Request protection violation

  This error indicates that the L4ConnID of a request is not one of the allowed ConnIDs associated with the target region. This error is reported using an in-band error and is written to the INBAND_ERROR field of the L4IA AGENT_STATUS register. A protection violation error code is also saved in the ERROR_LOG register CODE field.

- The target core does not respond before a time-out expires

  A time-out mechanism can be enabled on a per-target basis (L4TAx/L4TAOx AGENT_CONTROL register). If the mechanism is enabled for a target core, and commands are not accepted or responses are not returned within the expected delay, an error event is generated by the L4 interconnect.

  The error is logged in the target agent AGENT_STATUS register. The affected target agent enters an error state that causes it to send error responses to any new request targeted to it. To recover from this state, the target agent must be reset by system software. The time-out is counted starting from the time a command is presented to the OCP, independently of the state of the target SCmdAccept.

The L4 interconnect time-out mechanism is described more in detail in the next section.

Errors Detected by Target Modules and Directly Forwarded to the L3

- ☐ In-band ERR response that is received at a target agent from the attached module

  The error is logged in the L4IA AGENT_STATUS(27): INBAND_ ERROR register field.

- ☐ Out-of-band SError assertion from a target module

  This feature can be enabled on a per-agent basis in the L4TA AGENT_CONTROL register (bit field AGENT_CONTROL(24): SERROR_REP set to 1). If enabled (enabled for all L4TAO and disabled for all L4TA), SError assertion is the combination of all target cores having that report capability in the L4 interconnect, and the aggregated signal is forwarded to the L3. The error is also logged in the L4TA AGENT_STATUS(24): SERROR register bit.

  Upon SError assertion on the L4 port, L3 logic can generate an interrupt to the MPU. See the *Composition Error Flag* subsection of this section.

  As a general policy in the L4 interconnect, any error-related logging bit can be cleared under software control by writing1 to that bit.

Time-Out

The L4 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. Selection is done by programming the L4LA NETWORK_ CONTROL_L(10:8): TIMEOUT_BASE register field. This is illustrated in Table 6−167. All values in the table are expressed in number of L4 clock cycles.

Table 6−165. L4 Time-Out: Time Base Programming

TIMEOUT_BASE	1X−Time Base	4X−Time Base	16X−Time Base	64X−Time Base
0	All L4 time-out features are disabled.			
1	64	256	1024	4096
2	256	1024	4096	16384
3	1024	4096	16384	65536
4	4096	16384	65536	262144

The reset value is 4, resulting in the longest possible time-out.

The selected time base signals are available at any target agent. Each target agent can then be programmed to refer to one of these four time base signals, using the L4TAxx/L4TAOxx AGENT_CONTROL(10:8): REQ_TIMEOUT register field, as shown in Table 6−166.

Table 6−166. L4 Time-Out: Target Agent Programming

REQ_TIMEOUT	Target Agent Time-Out Reference
0	Time-out locally disabled
1	1X−time base
2	4X−time base
3	16X−time base
4	64X−time base

A time-out condition is detected when the command acceptance or the response is not received after a delay of between 1 and 3 time base periods.

Example:

☐ L4 frequency = 100 MHz
☐ TIMEOUT_BASE = 4 in L4LA NETWORK_CONTROL_L register
☐ REQ_TIMEOUT = 2 in the AGENT_CONTROL for agent A
☐ REQ_TIMEOUT = 4 in the AGENT_CONTROL for agent B

At agent A, the time base unit is 16384 cycles; that is,163 μs.

A time-out is issued when a request to the attached module is not accepted or gets no response after a delay of between 163μs and 489μs.

At agent B, the time base unit is 262144 cycles; that is, 2621 μs.

A time-out is issued when a request to the attached module is not accepted or gets no response after a delay of between 2621μs and 7863 μs.

On detection of a time-out condition, the agent auto-generates an error response to the L4IA, which is forwarded to the L3. The agent also logs the time-out to the L4TA AGENT_STATUS(8): REQ_TIMEOUT register status bit.

After the time-out is detected and logged, the behavior of the attached module is ignored. A new request targeting 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 the time-out error, the software is assumed to reset the target agent using the L4TA AGENT_CONTROL(0): OCP_RESET bitfield, and then to reset the faulty module (an L4 module has an individual soft−reset bit, which must be used once its agent is reset to restore the access on the OCP interface).

Applying a Target Agent Software Reset

Writing 1 to the L4TA AGENT_CONTROL(0) register OCP_RESET field 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 events occur:

- Requests sent to the target module receive ERR responses. Therefore, if the faulty request was part of a DMA transfer, it is necessary to stop the DMA to avoid unwanted errors.
- Requests sent to the target agent register block are processed as usual.
- The L4TA AGENT_STATUS(8): REQ_TIMEOUT status bit is cleared.

Writing 0 to the L4TA AGENT_CONTROL(0): OCP_RESET field terminates the software reset period.

The attached module must then be reset to complete the recovery.

Composite Error Flag

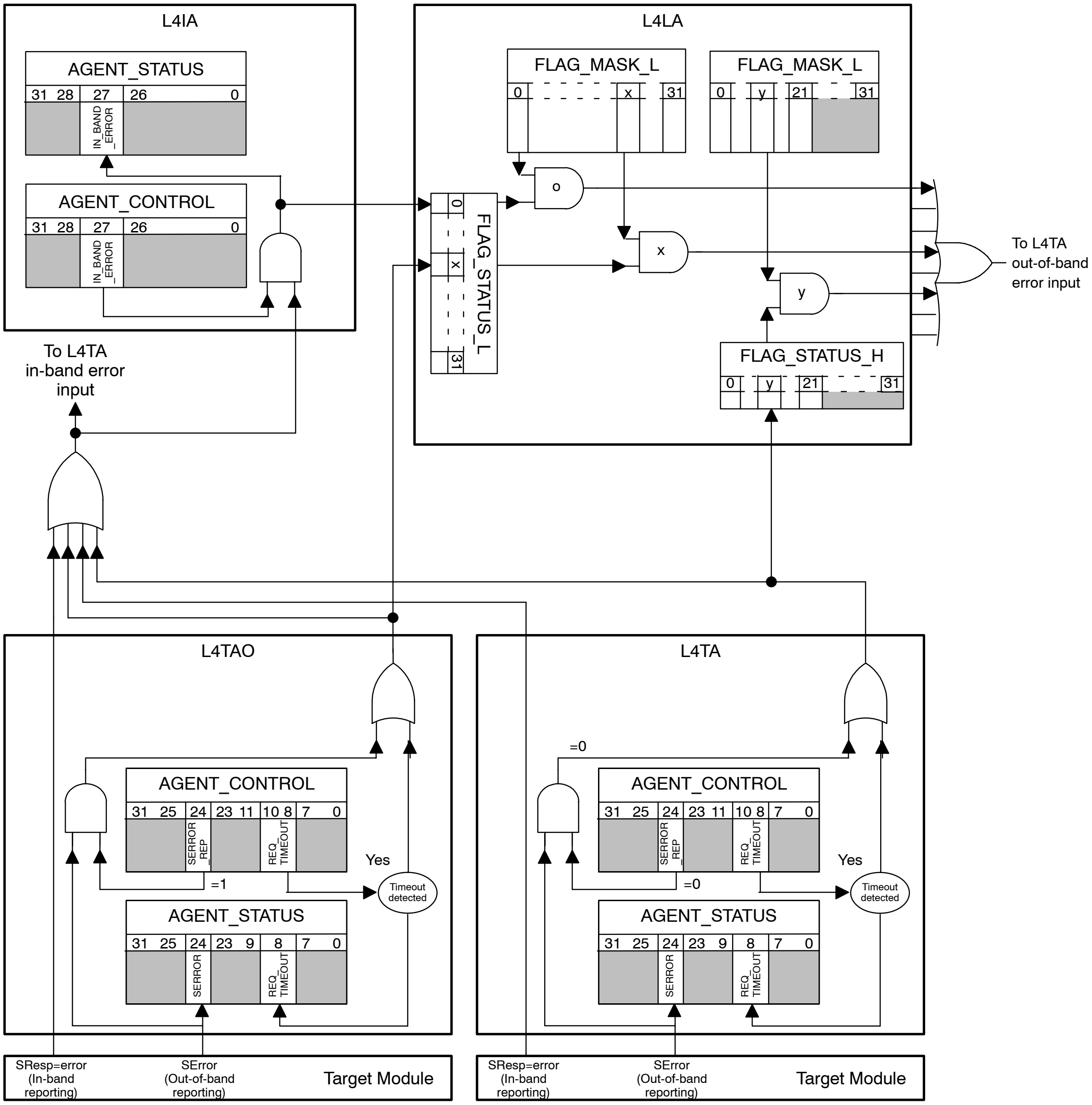
L4 interconnect errors can be reported out-of-band to the L3 aggregate SBError if the composite error flag in the L4LA is configured correctly. L4TAO agents can report out-of-band errors for both time-out and Sresp = error (in-band error detected). L4TA can report out-of-band error for time-out only. L4IA can report out-of-band error for Sresp = error (in-band error detected) only.

- Composite error mask flag: L4LA FLAG_MASK_L(31:0) and FLAG_MASK_H(21:0) allow masking of incoming errors from each L4 interconnect agent. The composite error flag avoids module error propagation to the SBError aggregate signal when the module corresponding flag mask is set. Each module is identified by a composite error flag index.
- Composite flag status: L4LA FLAG_STATUS_L(31:0) and FLAG_STATUS_H(21:0) log the error status for each module. The error status is positioned even if the module reporting is masked by the composite error mask flag.

Table 6–167. Composite Error Flag Mask

Flag Mask Index	Module	Mask Register Bit	Flag Mask Index	Module	Mask Register Bit
0	L4IA initiator agent	FLAG_MASK_L(0)	28	Reserved	FLAG_MASK_L(28)
1	Control and pinout module	FLAG_MASK_L(1)	29	GP timer 2	FLAG_MASK_L(29)
2	32K timer	FLAG_MASK_L(2)	30	GP timer 3	FLAG_MASK_L(30)
3	PRCM	FLAG_MASK_L(3)	31	GP timer 4	FLAG_MASK_L(31)
4	Reserved		32	GP timer 5	FLAG_MASK_H(0)
5	Reserved		33	GP timer 6	FLAG_MASK_H(1)
6	Quad GPIO	FLAG_MASK_L(6)	34	GP timer 7	FLAG_MASK_H(2)
7	WD timer1/2	FLAG_MASK_L(7)	35	GP timer 8	FLAG_MASK_H(3)
8	GP timer 1	FLAG_MASK_L(8)	36	GP timer 9	FLAG_MASK_H(4)
9	Reserved		37	GP timer 10	FLAG_MASK_H(5)
10	DSS (display subsystem)	FLAG_MASK_L(10)	38	GP timer 11	FLAG_MASK_H(6)
11	Camera subsystem	FLAG_MASK_L(11)	39	GP timer 12	FLAG_MASK_H(7)
12	sDMA	FLAG_MASK_L(12)	40	EAC	FLAG_MASK_H(8)
13	Reserved	FLAG_MASK_L(13)	41	FAC	FLAG_MASK_H(9)
14	USB	FLAG_MASK_L(14)	42	IPC	FLAG_MASK_H(10)
15	Reserved		43	SPI1	FLAG_MASK_H(11)
16	Reserved		44	SPI2	FLAG_MASK_H(12)
17	Reserved		45	MMC SDIO	FLAG_MASK_H(13)
18	Reserved		46	Reserved	
19	Reserved		47	Reserved	FLAG_MASK_H(15)
20	UART1	FLAG_MASK_L(20)	48	Reserved	FLAG_MASK_H(16)
21	UART2	FLAG_MASK_L(21)	49	Reserved	FLAG_MASK_H(17)
22	UART3	FLAG_MASK_L(22)	50	Reserved	FLAG_MASK_H(18)
23	I2C1	FLAG_MASK_L(23)	51	Reserved	FLAG_MASK_H(19)
24	I2C2	FLAG_MASK_L(24)	52	Reserved	
25	McBSP1	FLAG_MASK_L(25)	53	HDQ/1-Wire	FLAG_MASK_H(21)
26	McBSP2	FLAG_MASK_L(26)			
27	WD Timer 3 (DSP)	FLAG_MASK_L(27)			

Figure 6−20. L4 Interconnect Error Reporting



L4TAO Agent peripheral list:

System control
System pinout
32K timer
PRCM
USB
UART1−3
I2C 1−2
McBSP 1−2
MMC SDIO

L4TA Agent peripheral list:

GPIO 1−4
WD timer 1−3
DSS
Camera
sDMA
UART1−3
GP timer 1−12
EAC
FAC
IPC
SPI 1−2
HDQ/1−wire

L4 Firewall Error Logging (L4IA)

Table 6–168. L4 Firewall Error Logging Initiator Agent Registers Description

Address Base/ Offset	Register	Bits	Field	Reset Value	R/W	Description
0x20	AGENT_CONTROL	27	INBAND_ERROR_ REP	0x1	RW	Initiator interface error–reporting enable
0x28	AGENT_STATUS	27	INBAND_ERROR	0x0	RO	Error status
0x58	ERROR_LOG	25:24	CODE	0x0	RO	Initiator request error code: 0: No error 1: Reserved 2: Address hole error
		31	MULTI	0x0	RO	Multiple errors detected

6.4.5 Clocking, Reset, and Power-Management Scheme

6.4.5.1 Clocking

Only one clock is used in the L4 interconnect: Core_L4_ICLK.

Clock	Frequency	Name	Comments
Interface	Core_L3_ICLK or Core_L3_ICLK/2	Core_L4_ICLK	Source, control, and gating handled by PRCM

6.4.5.2 Reset

There is no software reset, only hardware reset, which is controlled by the PRCM and applied to the L4 interconnect and all the connected L4 target agents and L4 target modules.

6.4.5.3 Hardware Reset

The L4 interconnect receives a reset signal (CORE_RST) from the PRCM . CORE_RST is the reset signal to the CORE power domain. (See

Chapter 5, *Power, Reset, and Clock Management,* for more information. When asserted, CORE_RST is propagated to all modules connected to the L4 interconnect.

Interconnect	Reset Domain
L4 interconnect	CORE_RST

6.4.5.4 Power Management

Power Domain

The L4 interconnect connects to the CORE power domain, which can dynamically switch between low voltage and high voltage.

Interconnect	Power Domain
L4 interconnect	CORE

Module Power Saving

The L4 interconnect automatically handles internal clock autogating features to reduce power consumption. This functionality is active by default. It is possible to deactivate it by writing 1 to the CLOCKGATE_DISABLE register bit: NETWORK_CONTROL_H[24] of the L4LA.

System Power Management and Wakeup

As part of the system-wide power-management scheme, the L4 interconnect can go to IDLE state at the request of the PRCM (for more information, see Chapter 5, *Power, Reset, and Clock Management*). The L4 interconnect is always in smart-idle mode; that is, it goes to IDLE state after receiving the request from the PRCM, after all transfer requests in process are serviced. This functionality is handled by hardware; the L4 interconnect sends back an acknowledgement to the PRCM if the idle request is accepted.

6.5 L4 Interconnect Registers

Table 6−169 lists the L4 registers.

Table 6−169. L4 Interconnect Instance Registers

Module Name	**Base Address**	**Size**	**Port**
L4TAO1	0x4800 1000	512 bytes	System control and pinout
L4TAO2	0x4800 5000	512 bytes	32K timer
L4TAO3	0x4800 9000	512 bytes	PRCM
L4TA1	0x4801 3000	512 bytes	Reserved
L4TA2	0x4801 5000	512 bytes	Reserved
L4TA3	0x4801 B000	512 bytes	Quad GPIO
L4TA4	0x4802 3000	512 bytes	Dual WD timer (1/2)
L4TA5	0x4802 5000	512 bytes	WD timer 3 (DSP)
L4TA6	0x4802 7000	512 bytes	Reserved
L4TA7	0x4802 9000	512 bytes	GP timer1
L4TA8	0x4802 B000	512 bytes	GP timer2
L4AP	0x4804 0000	2K bytes	Address and protection
L4IA	0x4804 0800	512 bytes	Initiator agent
L4LA	0x4804 1000	512 bytes	Link agent
L4TA9	0x4804 A000	512 bytes	Reserved
L4TA10	0x4805 1000	512 bytes	DSS
L4TA11	0x4805 3000	512 bytes	Camera
L4TA12	0x4805 7000	512 bytes	sDMA
L4TA13	0x4805 C000	512 bytes	Reserved
L4TAO4	0x4805 F000	512 bytes	USB OTG
L4TA14	0x4806 1000	512 bytes	Reserved
L4TA15	0x4806 3000	512 bytes	Reserved
L4TA16	0x4806 5000	512 bytes	Reserved
L4TA17	0x4806 7000	512 bytes	Reserved
L4TA18	0x4806 9000	512 bytes	Reserved
L4TA19	0x4806 B000	512 bytes	UART1
L4TA20	0x4806 D000	512 bytes	UART2
L4TA21	0x4806 F000	512 bytes	UART3
L4TAO5	0x4807 1000	512 bytes	I^2C1
L4TAO6	0x4807 3000	512 bytes	I^2C2
L4TAO7	0x4807 5000	512 bytes	McBSP1
L4TAO8	0x4807 7000	512 bytes	McBSP2

Table 6−169. L4 Interconnect Instance Registers(Continued)

Module Name	Base Address	Size	Port
L4TA22	0x4807 9000	512 bytes	GP timer3
L4TA23	0x4807 B000	512 bytes	GP timer4
L4TA24	0x4807 D000	512 bytes	GP timer5
L4TA25	0x4807 F000	512 bytes	GP timer6
L4TA26	0x4808 1000	512 bytes	GP timer7
L4TA27	0x4808 3000	512 bytes	GP timer8
L4TA28	0x4808 5000	512 bytes	GP timer9
L4TA29	0x4808 7000	512 bytes	GP timer10
L4TA30	0x4808 9000	512 bytes	GP timer11
L4TA31	0x4808 B000	512 bytes	GP timer12
L4TA32	0x4809 1000	512 bytes	EAC
L4TA33	0x4809 3000	512 bytes	FAC
L4TA34	0x4809 5000	512 bytes	IPC
L4TA35	0x4809 9000	512 bytes	SPI1
L4TA36	0x4809 B000	512 bytes	SPI2
L4TAO9	0x4809 D000	512 bytes	MMC SDIO
L4TAO10	0x4809 F000	512 bytes	Reserved
L4TAO11	0x480A1000	512 bytes	Reserved
L4TAO12	0x480A3000	512 bytes	Reserved
L4TAO13	0x480A5000	512 bytes	Reserved
L4TA37	0x480A7000	512 bytes	Reserved
L4TA38	0x480AA000	512 bytes	Reserved
L4TAO14	0x480B1000	512 bytes	Reserved
L4TA39	0x480B3000	512 bytes	HDQ/1-Wire

6.5.1 L4TAO

Table 6−170 summarizes the instances of L4TAO. X X in the physical address can be replaced by a real value using the instance summary table.

Table 6−170 lists the L4TAO1-14 registers. Table 6−171 through Table 6−173 describe the register bits.

Table 6−170. L4TAO1−14 Registers

Register Name	Type	Register Width (Bits)	Physical Address
COMPONENT	R	32	0x480X X000
AGENT_CONTROL	RW	32	0x480X X020
AGENT_STATUS	R	32	0x480X X028

Table 6−171. COMPONENT Register

Address Offset	0x000		
Physical Address	0x4800 1000	**Instance**	L4TAO1: System control and pinout
	0x4800 5000		L4TAO2: 32K timer
	0x4800 9000		L4TAO3: PRCM
	0x4805 F000		L4TAO4: USB OTG
	0x4807 1000		L4TAO5: I2C1
	0x4807 3000		L4TAO6: I2C2
	0x4807 5000		L4TAO7: McBSP1
	0x4807 7000		L4TAO8: McBSP2
	0x4809 D000		L4TAO9: MMC/SDIO
	0x4809 F000		L4TAO10: Reserved
	0x480A1000		L4TAO11: Reserved
	0x480A3000		L4TAO12: Reserved
	0x480A5000		L4TAO13: Reserved
	0x480B1000		L4TAO14: Reserved
Description	Logs information about L4 revision code and interconnect 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
CODE																REV															

Bits	Field Name	Description	Type	Reset
31:16	CODE	Interconnect code	R	
15:0	REV	L4 revision code	R	

Table 6−172. AGENT_CONTROL Register

Address Offset	0x020		
Physical Address	0x4800 1020	**Instance**	L4TAO1: System control and pinout
	0x4800 5020		L4TAO2: 32K timer
	0x4800 9020		L4TAO3: PRCM
	0x4805 F020		L4TAO4: USB OTG
	0x4807 1020		L4TAO5: I2C1
	0x4807 3020		L4TAO6: I2C2
	0x4807 5020		L4TAO7: McBSP1
	0x4807 7020		L4TAO8: McBSP2
	0x4809 D020		L4TAO9: MMC/SDIO
	0x4809 F020		L4TAO10: Reserved
	0x480A1020		L4TAO11: Reserved
	0x480A3020		L4TAO12: Reserved
	0x480A5020		L4TAO13: Reserved
	0x480B1020		L4TAO14: Reserved
Description	Agent control register bit allocation		
Type	RW		

31	30	29	28	27	26	25	24	23	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	Write 0 for future compatibility. Read returns 0.	RW	0x00
24	SERROR_REP	Serror reporting enable Fixed at 1 by hardware. Out-of-band reporting is always enabled for L4TAO agents.	R	1
23:11	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x0000
10:8	REQ_TIMEOUT	Request time-out control 0x0: No time-out 0x1: 1 x base cycle 0x2: 4 x base cycle 0x3: 16 x base cycle 0x4: 64 x base cycle	RW	0x2

Table 6−172. AGENT_CONTROL (Continued)

Bits	Field Name	Description	Type	Reset
7:1	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x00
0	OCP_RESET	Reset OCP Interface (clears time-out flag)	RW	0

Table 6−173. AGENT_STATUS Register

Address Offset	0x028		
Physical Address	0x4800 1028	**Instance**	L4TAO1: System control and pinout
	0x4800 5028		L4TAO2: 32K timer
	0x4800 9028		L4TAO3: PRCM
	0x4805 F028		L4TAO4: USB OTG
	0x4807 1028		L4TAO5: I2C1
	0x4807 3028		L4TAO6: I2C2
	0x4807 5028		L4TAO7: McBSP1
	0x4807 7028		L4TAO8: McBSP2
	0x4809 D028		L4TAO9: MMC/SDIO
	0x4809 F028		L4TAO10: Reserved
	0x480A1028		L4TAO11: Reserved
	0x480A3028		L4TAO12: Reserved
	0x480A5028		L4TAO13: Reserved
	0x480B1028		L4TAO14: Reserved
Description	Records Req_time-out and Serror fields − LSBs		
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							SERROR	Reserved															REQ_TIMEOUT	Reserved							

Bits	Field Name	Description	Type	Reset
31:25	Reserved	Read returns 0.	R	0x00
24	SERROR	Value of OCP Serror signal	RW	0
23:9	Reserved	Read returns 0.	R	0x0000
8	REQ_TIMEOUT	Timed out request status	RW	0
7:0	Reserved	Read returns 0.	R	0x00

6.5.2 L4TA

Table 6−174 summarizes the 39 instances of L4TA. X X in the physical address can be replaced by a real value using the instance summary table.

Table 6−174 lists the L4TA1 registers. Table 6−175 through Table 6−207 describe the register bits.

Table 6−174. L4TA1 Registers

Register Name	Type	Register Width (Bits)	Physical Address
COMPONENT	R	32	0x480X X000
AGENT_CONTROL	RW	32	0x480X X020
AGENT_STATUS	R	32	0x480X X028

Table 6−175. COMPONENT Register

Address Offset	0x000		
Physical address	0x4801 3000	**Instance**	L4TA1: Reserved
	0x4801 5000		L4TA2: Reserved
	0x4801 B000		L4TA3: Quad GPIO
	0x4802 3000		L4TA4: Dual WD timer (1/2)
	0x4802 5000		L4TA5: WD timer3 (DSP)
	0x4802 7000		L4TA6: Reserved
	0x4802 9000		L4TA7: GP timer1
	0x4802 B000		L4TA8: GP timer2
	0x4804 A000		L4TA9: Reserved
	0x4805 1000		L4TA10: DSS
	0x4805 3000		L4TA11: Camera
	0x4805 7000		L4TA12: sDMA
	0x4805 C000		L4TA13: Reserved
	0x4806 1000		L4TA14: Reserved
	0x4806 3000		L4TA15: Reserved
	0x4806 5000		L4TA16: Reserved
	0x4806 7000		L4TA17: Reserved
	0x4806 9000		L4TA18: Reserved
	0x4806 B000		L4TA19: UART1
	0x4806 D000		L4TA20: UART2
	0x4806 F000		L4TA21: UART3
	0x4807 9000		L4TA22: GP timer3
	0x4807 B000		L4TA23: GP timer4
	0x4807 D000		L4TA24: GP timer5
	0x4807 F000		L4TA25: GP timer6
	0x4808 1000		L4TA26: GP timer7

Table 6–175. COMPONENT (Continued)

Physical address	0x4808 3000	**Instance**	L4TA27: GP timer8
	0x4808 5000		L4TA28: GP timer9
	0x4808 7000		L4TA29: GP timer10
	0x4808 9000		L4TA30: GP timer11
	0x4808 B000		L4TA31: GP timer12
	0x4809 1000		L4TA32: EAC
	0x4809 3000		L4TA33: FAC
	0x4809 5000		L4TA34: IPC
	0x4809 9000		L4TA35: SPI1
	0x4809 B000		L4TA36: SPI2
	0x480A7000		L4TA37: Reserved
	0x480AA000		L4TA38: Reserved
	0x480B3000		L4TA39: HDQ/1-Wire
Description	Logs information about L4 revision code and interconnect 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
CODE																REV															

Bits	**Field Name**	**Description**	**Type**	**Reset**
31:16	CODE	Interconnect code	R	
15:0	REV	L4 revision code	R	

Table 6–176. AGENT_CONTROL Register

Address Offset	0x020		
Physical Address	0x4801 3020	**Instance**	L4TA1: Reserved
	0x4801 5020		L4TA2: Reserved
	0x4801 B020		L4TA3: Quad GPIO
	0x4802 3020		L4TA4: Dual WD timer (1/2)
	0x4802 5020		L4TA5: WD timer3 (DSP)
	0x4802 7020		L4TA6: Reserved
	0x4802 9020		L4TA7: GP timer1
	0x4802 B020		L4TA8: GP timer2
	0x4804 A020		L4TA9: Reserved
	0x4805 1020		L4TA10: DSS
	0x4805 3020		L4TA11: Camera
	0x4805 7020		L4TA12: sDMA
	0x4805 C020		L4TA13: Reserved
	0x4806 1020		L4TA14: Reserved
	0x4806 3020		L4TA15: Reserved

Table 6−176. AGENT_CONTROL (Continued)

Physical Address	0x4806 5020	**Instance**	L4TA16: Reserved
	0x4806 7020		L4TA17: Reserved
	0x4806 9020		L4TA18: Reserved
	0x4806 B020		L4TA19: UART1
	0x4806 D020		L4TA20: UART2
	0x4806 F020		L4TA21: UART3
	0x4807 9020		L4TA22: GP timer3
	0x4807 B020		L4TA23: GP timer4
	0x4807 D020		L4TA24: GP timer5
	0x4807 F020		L4TA25: GP timer6
	0x4808 1020		L4TA26: GP timer7
	0x4808 3020		L4TA27: GP timer8
	0x4808 5020		L4TA28: GP timer9
	0x4808 7020		L4TA29: GP timer10
	0x4808 9020		L4TA30: GP timer11
	0x4808 B020		L4TA31: GP timer12
	0x4809 1020		L4TA32: EAC
	0x4809 3020		L4TA33: FAC
	0x4809 5020		L4TA34: IPC
	0x4809 9020		L4TA35: SPI1
	0x4809 B020		L4TA36: SPI2
	0x480A7020		L4TA37: Reserved
	0x480AA020		L4TA38: Reserved
	0x480B3020		L4TA39: HDQ/1-Wire
Description	Request Info permission configuration for region 0 − LSBs − Reserved for future use		
Type	RW		

31	30	29	28	27	26	25	24	23	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	Write 0 for future compatibility. Read returns 0.	RW	0x00
24	SERROR_REP	Serror reporting enable Fixed at 0 by hardware. Out-of-band reporting is not possible for L4TA agents.	R	0

Table 6–176. AGENT_CONTROL (Continued)

Bits	Field Name	Description	Type	Reset
23:11	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x0000
10:8	REQ_TIMEOUT	Request time-out control 0x0: No time-out 0x1: 1x Base cycles 0x2: 4x Base cycles 0x3: 16x Base cycles 0x4: 64x Base cycles	RW	0x2
7:1	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x00
0	OCP_RESET	Reset OCP Interface (clears time-out flag)	RW	0

Table 6−177. AGENT_STATUS Register

Address Offset	0x028		
Physical Address	0x4801 3028	**Instance**	L4TA1: Reserved
	0x4801 5028		L4TA2: Reserved
	0x4801 B028		L4TA3: Quad GPIO
	0x4802 3028		L4TA4: Dual WD timer (1/2)
	0x4802 5028		L4TA5: WD timer3 (DSP)
	0x4802 7028		L4TA6: Reserved
	0x4802 9028		L4TA7: GP timer1
	0x4802 B028		L4TA8: GP timer2
	0x4804 A028		L4TA9: ARM 11 ETB Test/emulation*
	0x4805 1028		L4TA10: DSS
	0x4805 3028		L4TA11: Camera
	0x4805 7028		L4TA12: sDMA
	0x4805 C028		L4TA13: Reserved
	0x4806 1028		L4TA14:Reserved
	0x4806 3028		L4TA15: Reserved
	0x4806 5028		L4TA16: Reserved
	0x4806 7028		L4TA17: Reserved
	0x4806 9028		L4TA18: Reserved
	0x4806 B028		L4TA19: UART1
	0x4806 D028		L4TA20: UART2
	0x4806 F028		L4TA21: UART3
	0x4807 9028		L4TA22: GP timer3
	0x4807 B028		L4TA23: GP timer4
	0x4807 D028		L4TA24: GP timer5
	0x4807 F028		L4TA25: GP timer6
	0x4808 1028		L4TA26: GP timer7
	0x4808 3028		L4TA27: GP timer8
	0x4808 5028		L4TA28: GP timer9
	0x4808 7028		L4TA29: GP timer10
	0x4808 9028		L4TA30: GP timer11
	0x4808 B028		L4TA31: GP timer12
	0x4809 1028		L4TA32: EAC
	0x4809 3028		L4TA33: FAC
	0x4809 5028		L4TA34: IPC
	0x4809 9028		L4TA35: SPI1
	0x4809 B028		L4TA36: SPI2
	0x480A7028		L4TA37: Reserved
	0x480AA028		L4TA38: Reserved
	0x480B3028		L4TA39: HDQ/1-Wire
Description	Records Req_time-out and Serror fields − LSBs		
Type	R		

Table 6–177. AGENT_STATUS (Continued)

31	30	29	28	27	26	25	24	23	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															REQ_TIMEOUT	Reserved							

Bits	Field Name	Description	Type	Reset
31:25	Reserved	Read returns 0.	R	0x00
24	SERROR	Value of OCP Serror signal It is always 0 as the out-of-band reporting is not enabled for L4TA agents.	RW	0
23:9	Reserved	Read returns 0.	R	0x0000
8	REQ_TIMEOUT	Timed out request status	RW	0
7:0	Reserved	Read returns 0.	R	0x00

6.5.3 L4AP: Address and Protection

Table 6−178 lists the L4AP registers. Table 6−179 through Table 6−207 describe the register bits.

Table 6−178. L4AP Registers

Register Name	Type	Register Width (Bits)	Physical Address
COMPONENT	R	32	0x4804 0000
SEGMENT_L_0	R	32	0x4804 0100
SEGMENT_H_0	R	32	0x4804 0104
SEGMENT_L_1	R	32	0x4804 0108
SEGMENT_H_1	R	32	0x4804 010C
SEGMENT_L_2	R	32	0x4804 0110
SEGMENT_H_2	R	32	0x4804 0114
SEGMENT_L_3	R	32	0x4804 0118
SEGMENT_H_3	R	32	0x4804 011C
SEGMENT_L_4	R	32	0x4804 0120
SEGMENT_H_4	R	32	0x4804 0124
SEGMENT_L_5	R	32	0x4804 0128
SEGMENT_H_5	R	32	0x4804 012C
PROTGROUP_0	RW	32	0x4804 0200
PROTGROUP_0_H	RW	32	0x4804 0204
PROTGROUP_1	RW	32	0x4804 0208
PROTGROUP_2	RW	32	0x4804 0210
PROTGROUP_3	RW	32	0x4804 0218
PROTGROUP_4	RW	32	0x4804 0220
PROTGROUP_5	RW	32	0x4804 0228
PROTGROUP_6	RW	32	0x4804 0230
PROTGROUP_7	RW	32	0x4804 0238
REGION_0_L	R	32	0x4804 0300
REGION_0_H	R	32	0x4804 0304
REGION_1_L	R	32	0x4804 0308
REGION_1_H	RW	32	0x4804 030C
REGION_2_L	R	32	0x4804 0310
REGION_2_H	RW	32	0x4804 0314
REGION_3_L	R	32	0x4804 0318
REGION_3_H	RW	32	0x4804 031C
REGION_4_L	R	32	0x4804 0320

Table 6−178. L4AP Registers

Register Name	Type	Register Width (Bits)	Physical Address
REGION_4_H	RW	32	0x4804 0324
REGION_5_L	R	32	0x4804 0328
REGION_5_H	RW	32	0x4804 032C
REGION_6_L	R	32	0x4804 0330
REGION_6_H	RW	32	0x4804 0334
REGION_7_L	R	32	0x4804 0338
REGION_7_H	RW	32	0x4804 033C
REGION_8_L	R	32	0x4804 0340
REGION_8_H	RW	32	0x4804 0344
REGION_9_L	R	32	0x4804 0348
REGION_9_H	RW	32	0x4804 034C
REGION_10_L	R	32	0x4804 0350
REGION_10_H	RW	32	0x4804 0354
REGION_11_L	R	32	0x4804 0358
REGION_11_H	RW	32	0x4804 035C
REGION_12_L	R	32	0x4804 0360
REGION_12_H	RW	32	0x4804 0364
REGION_13_L	R	32	0x4804 0368
REGION_13_H	RW	32	0x4804 036C
REGION_14_L	R	32	0x4804 0370
REGION_14_H	RW	32	0x4804 0374
REGION_15_L	R	32	0x4804 0378
REGION_15_H	RW	32	0x4804 037C
REGION_16_L	R	32	0x4804 0380
REGION_16_H	RW	32	0x4804 0384
REGION_17_L	R	32	0x4804 0388
REGION_17_H	RW	32	0x4804 038C
REGION_18_L	R	32	0x4804 0390
REGION_18_H	RW	32	0x4804 0394
REGION_19_L	R	32	0x4804 0398
REGION_19_H	RW	32	0x4804 039C
REGION_20_L	R	32	0x4804 03A0
REGION_20_H	RW	32	0x4804 03A4
REGION_21_L	R	32	0x4804 03A8

Table 6−178. L4AP Registers

Register Name	Type	Register Width (Bits)	Physical Address
REGION_21_H	RW	32	0x4804 03AC
REGION_22_L	R	32	0x4804 03B0
REGION_22_H	RW	32	0x4804 03B4
REGION_23_L	R	32	0x4804 03B8
REGION_23_H	RW	32	0x4804 03BC
REGION_24_L	R	32	0x4804 03C0
REGION_24_H	RW	32	0x4804 03C4
REGION_25_L	R	32	0x4804 03C8
REGION_25_H	RW	32	0x4804 03CC
REGION_26_L	R	32	0x4804 03D0
REGION_26_H	RW	32	0x4804 03D4
REGION_27_L	R	32	0x4804 03D8
REGION_27_H	RW	32	0x4804 03DC
REGION_28_L	R	32	0x4804 03E0
REGION_28_H	RW	32	0x4804 03E4
REGION_29_L	R	32	0x4804 03E8
REGION_29_H	RW	32	0x4804 03EC
REGION_30_L	R	32	0x4804 03F0
REGION_30_H	RW	32	0x4804 03F4
REGION_31_L	R	32	0x4804 03F8
REGION_31_H	RW	32	0x4804 03FC
REGION_32_L	R	32	0x4804 0400
REGION_32_H	RW	32	0x4804 0404
REGION_33_L	R	32	0x4804 0408
REGION_33_H	RW	32	0x4804 040C
REGION_34_L	R	32	0x4804 0410
REGION_34_H	RW	32	0x4804 0414
REGION_35_L	R	32	0x4804 0418
REGION_35_H	RW	32	0x4804 041C
REGION_36_L	R	32	0x4804 0420
REGION_36_H	RW	32	0x4804 0424
REGION_37_L	R	32	0x4804 0428
REGION_37_H	RW	32	0x4804 042C
REGION_38_L	R	32	0x4804 0430

Table 6−178. L4AP Registers

Register Name	Type	Register Width (Bits)	Physical Address
REGION_38_H	RW	32	0x4804 0434
REGION_39_L	R	32	0x4804 0438
REGION_39_H	RW	32	0x4804 043C
REGION_40_L	R	32	0x4804 0440
REGION_40_H	RW	32	0x4804 0444
REGION_41_L	R	32	0x4804 0448
REGION_41_H	RW	32	0x4804 044C
REGION_42_L	R	32	0x4804 0450
REGION_42_H	RW	32	0x4804 0454
REGION_43_L	R	32	0x4804 0458
REGION_43_H	RW	32	0x4804 045C
REGION_44_L	R	32	0x4804 0460
REGION_44_H	RW	32	0x4804 0464
REGION_45_L	R	32	0x4804 0468
REGION_45_H	RW	32	0x4804 046C
REGION_46_L	R	32	0x4804 0470
REGION_46_H	RW	32	0x4804 0474
REGION_47_L	R	32	0x4804 0478
REGION_47_H	RW	32	0x4804 047C
REGION_48_L	R	32	0x4804 0480
REGION_48_H	RW	32	0x4804 0484
REGION_49_L	R	32	0x4804 0488
REGION_49_H	RW	32	0x4804 048C
REGION_50_L	R	32	0x4804 0490
REGION_50_H	RW	32	0x4804 0494
REGION_51_L	R	32	0x4804 0498
REGION_51_H	RW	32	0x4804 049C
REGION_52_L	R	32	0x4804 04A0
REGION_52_H	RW	32	0x4804 04A4
REGION_53_L	R	32	0x4804 04A8
REGION_53_H	RW	32	0x4804 04AC
REGION_54_L	R	32	0x4804 04B0
REGION_54_H	RW	32	0x4804 04B4
REGION_55_L	R	32	0x4804 04B8

Table 6−178. L4AP Registers

Register Name	Type	Register Width (Bits)	Physical Address
REGION_55_H	RW	32	0x4804 04BC
REGION_56_L	R	32	0x4804 04C0
REGION_56_H	RW	32	0x4804 04C4
REGION_57_L	R	32	0x4804 04C8
REGION_57_H	RW	32	0x4804 04CC
REGION_58_L	R	32	0x4804 04D0
REGION_58_H	RW	32	0x4804 04D4
REGION_59_L	R	32	0x4804 04D8
REGION_59_H	RW	32	0x4804 04DC
REGION_60_L	R	32	0x4804 04E0
REGION_60_H	RW	32	0x4804 04E4
REGION_61_H	R	32	0x4804 04EC
REGION_62_L	R	32	0x4804 04F0
REGION_62_H	RW	32	0x4804 04F4
REGION_63_L	R	32	0x4804 04F8
REGION_63_H	RW	32	0x4804 04FC
REGION_64_L	R	32	0x4804 0500
REGION_64_H	RW	32	0x4804 0504
REGION_65_L	R	32	0x4804 0508
REGION_65_H	RW	32	0x4804 050C
REGION_66_L	R	32	0x4804 0510
REGION_66_H	RW	32	0x4804 0514
REGION_67_L	R	32	0x4804 0518
REGION_67_H	RW	32	0x4804 051C
REGION_68_L	R	32	0x4804 0520
REGION_68_H	RW	32	0x4804 0524
REGION_69_L	R	32	0x4804 0528
REGION_69_H	RW	32	0x4804 052C
REGION_70_L	R	32	0x4804 0530
REGION_70_H	RW	32	0x4804 0534
REGION_71_L	R	32	0x4804 0538
REGION_71_H	RW	32	0x4804 053C
REGION_72_L	R	32	0x4804 0540
REGION_72_H	RW	32	0x4804 0544

Table 6−178. L4AP Registers

Register Name	Type	Register Width (Bits)	Physical Address
REGION_73_L	R	32	0x4804 0548
REGION_73_H	RW	32	0x4804 054C
REGION_74_L	R	32	0x4804 0550
REGION_74_H	RW	32	0x4804 0554
REGION_75_L	R	32	0x4804 0558
REGION_75_H	RW	32	0x4804 055C
REGION_76_L	R	32	0x4804 0560
REGION_76_H	RW	32	0x4804 0564
REGION_77_L	R	32	0x4804 0568
REGION_77_H	RW	32	0x4804 056C
REGION_78_L	R	32	0x4804 0570
REGION_78_H	RW	32	0x4804 0574
REGION_79_L	R	32	0x4804 0578
REGION_79_H	RW	32	0x4804 057C
REGION_80_L	R	32	0x4804 0580
REGION_80_H	RW	32	0x4804 0584
REGION_81_L	R	32	0x4804 0588
REGION_81_H	RW	32	0x4804 058C
REGION_82_L	R	32	0x4804 0590
REGION_82_H	RW	32	0x4804 0594
REGION_83_L	R	32	0x4804 0598
REGION_83_H	RW	32	0x4804 059C
REGION_84_L	R	32	0x4804 05A0
REGION_84_H	RW	32	0x4804 05A4
REGION_85_L	R	32	0x4804 05A8
REGION_85_H	RW	32	0x4804 05AC
REGION_86_L	R	32	0x4804 05B0
REGION_86_H	RW	32	0x4804 05B4
REGION_87_L	R	32	0x4804 05B8
REGION_87_H	RW	32	0x4804 05BC
REGION_88_L	R	32	0x4804 05C0
REGION_88_H	RW	32	0x4804 05C4
REGION_89_L	R	32	0x4804 05C8
REGION_89_H	RW	32	0x4804 05CC

Table 6−178. L4AP Registers

Register Name	Type	Register Width (Bits)	Physical Address
REGION_90_L	R	32	0x4804 05D0
REGION_90_H	RW	32	0x4804 05D4
REGION_91_L	R	32	0x4804 05D8
REGION_91_H	RW	32	0x4804 05DC
REGION_92_L	R	32	0x4804 05E0
REGION_92_H	RW	32	0x4804 05E4
REGION_93_L	R	32	0x4804 05E8
REGION_93_H	RW	32	0x4804 05EC
REGION_94_L	R	32	0x4804 05F0
REGION_94_H	RW	32	0x4804 05F4
REGION_95_L	R	32	0x4804 05F8
REGION_95_H	RW	32	0x4804 05FC
REGION_96_L	R	32	0x4804 0600
REGION_96_H	RW	32	0x4804 0604
REGION_97_L	R	32	0x4804 0608
REGION_97_H	RW	32	0x4804 060C
REGION_98_L	R	32	0x4804 0610
REGION_98_H	RW	32	0x4804 0614
REGION_99_L	R	32	0x4804 0618
REGION_99_H	RW	32	0x4804 061C
REGION_100_L	R	32	0x4804 0620
REGION_100_H	RW	32	0x4804 0624
REGION_101_L	R	32	0x4804 0628
REGION_101_H	RW	32	0x4804 062C
REGION_102_L	R	32	0x4804 0630
REGION_102_H	RW	32	0x4804 0634
REGION_103_L	R	32	0x4804 0638
REGION_103_H	RW	32	0x4804 063C
REGION_104_L	R	32	0x4804 0640
REGION_104_H	RW	32	0x4804 0644
REGION_105_L	R	32	0x4804 0648
REGION_105_H	RW	32	0x4804 064C
REGION_106_L	R	32	0x4804 0650
REGION_106_H	RW	32	0x4804 0654

Table 6−178. L4AP Registers

Register Name	Type	Register Width (Bits)	Physical Address
REGION_107_L	R	32	0x4804 0658
REGION_107_H	RW	32	0x4804 065C
REGION_108_L	R	32	0x4804 0660
REGION_108_H	RW	32	0x4804 0664
REGION_109_L	R	32	0x4804 0668
REGION_109_H	RW	32	0x4804 066C
REGION_110_L	R	32	0x4804 0670
REGION_110_H	RW	32	0x4804 0674
REGION_111_L	R	32	0x4804 0678
REGION_111_H	RW	32	0x4804 067C
REGION_112_L	R	32	0x4804 0680
REGION_112_H	RW	32	0x4804 0684
REGION_113_L	R	32	0x4804 0688
REGION_113_H	RW	32	0x4804 068C
REGION_114_L	R	32	0x4804 0690
REGION_114_H	RW	32	0x4804 0694
REGION_115_L	R	32	0x4804 0698
REGION_115_H	RW	32	0x4804 0698
REGION_116_L	R	32	0x4804 06A0
REGION_116_H	RW	32	0x4804 06A4
REGION_117_L	R	32	0x4804 06A8
REGION_117_H	RW	32	0x4804 06AC
REGION_118_L	R	32	0x4804 06B0
REGION_118_H	RW	32	0x4804 06B4
REGION_119_L	R	32	0x4804 06B8
REGION_119_H	RW	32	0x4804 06BC
REGION_120_L	R	32	0x4804 06C0
REGION_120_H	RW	32	0x4804 06C4
REGION_121_L	R	32	0x4804 06C8
REGION_121_H	RW	32	0x4804 06CC
REGION_122_L	R	32	0x4804 06D0
REGION_122_H	RW	32	0x4804 06D4
REGION_123_L	R	32	0x4804 06D8
REGION_123_H	RW	32	0x4804 06DC

Table 6−178. L4AP Registers

Register Name	Type	Register Width (Bits)	Physical Address
REGION_124_L	R	32	0x4804 06E0
REGION_124_H	RW	32	0x4804 06E4

Table 6−179. COMPONENT Register

Address Offset	0x000		
Physical Address	0x4804 0000	**Instance**	L4AP
Description	Logs information about L4 revision code and interconnect 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
CODE																REV															

Bits	Field Name	Description	Type	Reset
31:16	CODE	Interconnect code	R	
15:0	REV	L4 revision code	R	

Table 6−180. SEGMENT_L_0 Register

Address Offset	0x100		
Physical Address	0x4804 0100	**Instance**	L4AP
Description	Logs information about the segment base address		
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								BASE																							

Bits	Field Name	Description	Type	Reset
31:24	Reserved	Read returns 0.	R	0x00
23:0	BASE	Segment 0 base address	R	0x000000

Table 6−181. SEGMENT_H_0 Register

Address Offset	0x104		
Physical Address	0x4804 0104	**Instance**	L4AP
Description	Logs information about the segment 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																											SIZE				

Bits	Field Name	Description	Type	Reset
31:5	Reserved	Read returns 0.	R	0x0000000
4:0	SIZE	Segment 0 size	R	0x11

Table 6−182. SEGMENT_L_1 Register

Address Offset	0x108		
Physical Address	0x4804 0108	**Instance**	L4AP
Description	Logs information about the segment base address		
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								BASE																							

Bits	Field Name	Description	Type	Reset
31:24	Reserved	Read returns 0.	R	0x00
23:0	BASE	Segment 1 base address	R	0x020000

Table 6−183. SEGMENT_H_1 Register

Address Offset	0x10C		
Physical Address	0x4804 010C	**Instance**	L4AP
Description	Logs information about the segment 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																											SIZE				

Bits	Field Name	Description	Type	Reset
31:5	Reserved	Read returns 0.	R	0x0000000
4:0	SIZE	Segment 1 size	R	0x11

Table 6−184. SEGMENT_L_2 Register

Address Offset	0x110		
Physical Address	0x4804 0110	**Instance**	L4AP
Description	Logs information about the segment base address		
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								BASE																							

Bits	Field Name	Description	Type	Reset
31:24	Reserved	Read returns 0.	R	0x00
23:0	BASE	Segment 2 base address	R	0x040000

Table 6−185. SEGMENT_H_2 Register

Address Offset	0x114		
Physical Address	0x4804 0114	**Instance**	L4AP
Description	Logs information about the segment 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																											SIZE				

Bits	Field Name	Description	Type	Reset
31:5	Reserved	Read returns 0.	R	0x0000000
4:0	SIZE	Segment 2 size	R	0x11

Table 6−186. SEGMENT_L_3 Register

Address Offset	0x118		
Physical Address	0x4804 0118	**Instance**	L4AP
Description	Logs information about the segment base address		
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								BASE																							

Bits	Field Name	Description	Type	Reset
31:24	Reserved	Read returns 0.	R	0x00
23:0	BASE	Segment 3 base address	R	0x060000

Table 6−187. SEGMENT_H_3 Register

Address Offset	0x11C		
Physical Address	0x4804 011C	**Instance**	L4AP
Description	Logs information about the segment 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																											SIZE				

Bits	Field Name	Description	Type	Reset
31:5	Reserved	Read returns 0.	R	0x0000000
4:0	SIZE	Segment 3 size	R	0x11

Table 6–188. SEGMENT_L_4 Register

Address Offset	0x120		
Physical Address	0x4804 0120	**Instance**	L4AP
Description	Logs information about the segment base address		
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								BASE																							

Bits	Field Name	Description	Type	Reset
31:24	Reserved	Read returns 0.	R	0x00
23:0	BASE	Segment 4 base address	R	0x080000

Table 6–189. SEGMENT_H_4 Register

Address Offset	0x124		
Physical Address	0x4804 0124	**Instance**	L4AP
Description	Logs information about the segment 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																											SIZE				

Bits	Field Name	Description	Type	Reset
31:5	Reserved	Read returns 0.	R	0x0000000
4:0	SIZE	Segment 4 size	R	0x11

Table 6–190. SEGMENT_L_5 Register

Address Offset	0x128		
Physical Address	0x4804 0128	**Instance**	L4AP
Description	Logs information about the segment base address		
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								BASE																							

Bits	Field Name	Description	Type	Reset
31:24	Reserved	Read returns 0.	R	0x00
23:0	BASE	Segment 5 base address	R	0x0A0000

Table 6−191. SEGMENT_H_5 Register

Address Offset	0x12C		
Physical Address	0x4804 012C	**Instance**	L4AP
Description	Logs information about the segment 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																											SIZE				

Bits	Field Name	Description	Type	Reset
31:5	Reserved	Read returns 0.	R	0x0000000
4:0	SIZE	Segment 5 size	R	0x11

Table 6−192. PROTGROUP_0 Register

Address Offset	0x200		
Physical Address	0x4804 0200	**Instance**	L4AP
Description	Protection group 0 CONNID − LSBs		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CONNIDBITVECTOR															

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x0000
15:0	CONNID BITVECTOR	Protection group 0.CONNID vector on reset ARM only	RW	0x0202

Find below the description for protection group 1 to protection group 7. Protection groups 1 to 7 have identical values at reset release.

Table 6−193. PROTGROUP_1−7 Register

Address Offset	0x208,0x210,0x218,0x220,0x228,0x230,0x238				
Physical Address	0x4804 0208	**Instance**	L4AP	**PROTGROUP**	1
	0x4804 0210			**PROTGROUP**	2
	0x4804 0218			**PROTGROUP**	3
	0x4804 0220			**PROTGROUP**	4
	0x4804 0228			**PROTGROUP**	5
	0x4804 0230			**PROTGROUP**	6
	0x4804 0238			**PROTGROUP**	7
Description	Protection group 1 CONNID − LSBs				
Type	RW				

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CONNIDBITVECTOR															

Bits	**Field Name**	**Description**	**Type**	**Reset**
31:16	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x0000
15:0	CONNID BITVECTOR	Protection group N (With N = 1 to 7). CONNID vector − Set by ARM at boot	RW	0xFFFF

Table 6−194. REGION_0_L Register

Address Offset	0x300		
Physical Address	0x4804 0300	**Instance**	L4AP
Description	Region 0 AP block − LSBs		
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								BASE																							

Bits	**Field Name**	**Description**	**Type**	**Reset**
31:24	Reserved	Write 0 for future compatibility. Returns 0s.	RW	0x00
23:0	BASE	Region − base address inside segment	R	0x000000

Table 6−195. REGION_0_H Register

Address Offset	0x304		
Physical Address	0x4804 0304	**Instance**	L4AP
Description	Region 0 AP block MSBs		
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
MADDRSPACE				SEGMENT_ID				RESERVED	PROT_GROUP_ID			BYTE_GROUP_ID			RESERVED		PHI_target_ID							RESERVED		SIZE					ENABLE

Bits	Field Name	Description	Type	Reset
31:28	MADDRSPACE	Unused	R	0x0
27:24	SEGMENT_ID	Segment ID of the region	R	0x2
23	Reserved	Read returns 0.	R	0
22:20	PROT_GROUP_ID	Unused	R	0x0
19:17	BYTE_DATA_WIDTH	Target OCP data byte width	R	0x2
16:15	Reserved	Read returns 0.	R	0x0
14:8	PHI_target_ID	Physical target ID	R	0x00
7:6	Reserved	Read returns 0.	R	0x0
5:1	SIZE	Region size (address bits): 2^SIZE = Region byte size	R	0x11
0	ENABLE	Enable access to region 0. 0x0: Disable access to region 0: Read to the region area returns 0. Write to the region area has no effect. 0x1: Enable access to region 0.	R	1

Table 6−196. REGION_1_L Register

Address Offset	0x308		
Physical Address	0x4804 0308	**Instance**	L4AP
Description	Region 1 AP block − LSBs		
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								BASE																							

Bits	Field Name	Description	Type	Reset
31:24	Reserved	Write 0 for future compatibility. Returns 0s.	RW	0x00
23:0	BASE	Region − base address inside segment	R	0x000800

Table 6−197. REGION_1_H Register

Address Offset	0x30C		
Physical Address	0x4804 030C	**Instance**	L4AP
Description	Region 0 AP block MSBs		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MADDRSPACE				SEGMENT_ID				RESERVED	PROT_GROUP_ID			BYTE_GROUP_ID			RESERVED		PHI_target_ID							RESERVED		SIZE					ENABLE

Bits	Field Name	Description	Type	Reset
31:28	MADDRSPACE	Unused	R	0x1
27:24	SEGMENT_ID	Segment ID of the region	R	0x2
23	Reserved	Read returns 0.	R	0
22:20	PROT_ GROUP_ID	Protection group ID	RW	0x7
19:17	BYTE_DATA_ WIDTH	Target OCP data byte width	R	0x2
16:15	Reserved	Read returns 0.	R	0x0
14:8	PHI_target_ID	Unused	R	0x00
7:6	Reserved	Read returns 0.	R	0x0
5:1	SIZE	Region size (address bits): 2^SIZE = Region byte size	R	0x07

Table 6−197. REGION_1_H (Continued)

Bits	Field Name	Description	Type	Reset
0	ENABLE	Enable access to the region 0x0: Disable access to region 1: Read to the region area returns 0. Write to the region area has no effect. 0x1: Enable access to region 1.	R	1

Table 6−198. REGION_2_L Register

Address Offset	0x310		
Physical Address	0x4804 0310	**Instance**	L4AP
Description	Region 2 LA block − LSBs		
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								BASE																							

Bits	Field Name	Description	Type	Reset
31:24	Reserved	Write 0 for future compatibility. Returns 0s.	RW	0x00
23:0	BASE	Region − base address inside segment	R	0x001000

Table 6−199. REGION_2_H Register

Address Offset	0x314		
Physical Address	0x4804 0314	**Instance**	L4AP
Description	Region 0 LA block MSBs		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MADDRSPACE				SEGMENT_ID				RESERVED	PROT_GROUP_ID			BYTE_GROUP_ID			RESERVED		PHI_target_ID							RESERVED		SIZE					ENABLE

Bits	Field Name	Description	Type	Reset
31:28	MADDRSPACE	Unused	R	0x5
27:24	SEGMENT_ID	Segment ID of the region	R	0x2

Table 6−199. REGION_2_H (Continued)

Bits	Field Name	Description	Type	Reset
23	Reserved	Read returns 0.	R	0
22:20	PROT_ GROUP_ID	Protection group ID	RW	0x7
19:17	BYTE_DATA_ WIDTH	Target OCP data byte width	R	0x1
16:15	Reserved	Read returns 0.	R	0x0
14:8	PHI_target_ID	Unused	R	0x01
7:6	Reserved	Read returns 0.	R	0x0
5:1	SIZE	Region size (address bits): 2^SIZE = Region byte size	R	0x09
0	ENABLE	Enable access to the region. 0x0: Disable access: Read to the region area returns 0. Write to the region area has no effect. 0x1: Enable access.	R	1

Table 6−200. Protection Region 3−124 Summary Table

Register Name	Region Name	Offset	Physical Address	Segment	Protected Region Base Address	Protected region size (field value: size in bytes)
REGION_0_L	Address and protection	0x300	0x4804 0300	2	0x4804 0000	0xB:2K
REGION_0_H		0x304	0x4804 0304			
REGION_1_L	initiator agent	0x308	0x4804 0308	2	0x4804 0800	0xB:2K
REGION_1_H		0x30C	0x4804 030C			
REGION_2_L	Link agent	0x310	0x4804 0310	2	0x4804 1000	0xC:4K
REGION_2_H		0x314	0x4804 0314			
REGION_3_L	System control and pinout	0x318	0x4804 0318	0	0x4800 0000	0xC:4K
REGION_3_H		0x31C	0x4804 031C			
REGION_4_L	L4TAO1	0x320	0x4804 0320	0	0x4800 1000	0xC:4K
REGION_4_H		0x324	0x4804 0324			
REGION_5_L	32K timer	0x328	0x4804 0328	0	0x4800 4000	0xC:4K
REGION_5_H		0x32C	0x4804 032C			
REGION_6_L	L4TAO2	0x330	0x4804 0330	0	0x4800 5000	0xC:4K
REGION_6_H		0x334	0x4804 0334			
REGION_7_L	PRCM region A	0x338	0x4804 0338	0	0x4800 8000	0xB:2K
REGION_7_H		0x33C	0x4804 033C			

Notes: 1) Reserved for testing and emulation.

Table 6−200. Protection Region 3−124 Summary Table (Continued)

Register Name	Region Name	Offset	Physical Address	Segment	Protected Region Base Address	Protected region size (field value: size in bytes)
REGION_8_L	PRCM Region B	0x340	0x4804 0340	0	0x4800 8800	0xB:2K
REGION_8_H		0x344	0x4804 0344			
REGION_9_L	L4TAO3	0x348	0x4804 0348	0	0x4800 9000	0xC:4K
REGION_9_H		0x34C	0x4804 034C			
REGION_10_L	Test[1]	0x350	0x4804 0350	NA	NA	NA
REGION_10_H		0x354	0x4804 0354			
REGION_11_L	Test[1]	0x358	0x4804 0358	NA	NA	NA
REGION_11_H		0x35C	0x4804 035C			
REGION_12_L	Test[1]	0x360	0x4804 0360	NA	NA	NA
REGION_12_H		0x364	0x4804 0364			
REGION_13_L	Test[1]	0x368	0x4804 0368	NA	NA	NA
REGION_13_H		0x36C	0x4804 036C			
REGION_14_L	GPIO1	0x370	0x4804 0370	0	0x4801 9000	0xC:4K
REGION_14_H		0x374	0x4804 0374			
REGION_15_L	Quad GPIO Top	0x378	0x4804 0378	0	0x4801 8000	0xC:4K
REGION_15_H		0x37C	0x4804 037C			
REGION_16_L	GPIO2	0x380	0x4804 0380	0	0x4801 A000	0xC:4K
REGION_16_H		0x384	0x4804 0384			
REGION_17_L	L4TA3	0x388	0x4804 0388	0	0x4801 E000	0xC:4K
REGION_17_H		0x38C	0x4804 038C			
REGION_18_L	GPIO3	0x390	0x4804 0390	0	0x4801 B000	0xC:4K
REGION_18_H		0x394	0x4804 0394			
REGION_19_L	GPIO4	0x398	0x4804 0398	0	0x4800 C000	0xC:4K
REGION_19_H		0x39C	0x4804 039C			
REGION_20_L	Reserved	0x3A0	0x4804 03A0	1	0x4802 1000	0xC:4K
REGION_20_H		0x3A4	0x4804 03A4			
REGION_21_L	Dual WD Timer TOP	0x3A8	0x4804 03A8	1	0x4802 0000	0xC:4K
REGION_21_H		0x3AC	0x4804 03AC			
REGION_22_L	WD Timer 2 (OMAP)	0x3B0	0x4804 03B0	1	0x4802 2000	0xC:4K
REGION_22_H		0x3B4	0x4804 03B4			
REGION_23_L	L4TA4	0x3B8	0x4804 03B8	1	0x4802 3000	0xC:4K
REGION_23_H		0x3BC	0x4804 03BC			

Notes: 1) Reserved for testing and emulation.

Table 6−200. Protection Region 3−124 Summary Table (Continued)

Register Name	Region Name	Offset	Physical Address	Segment	Protected Region Base Address	Protected region size (field value: size in bytes)
REGION_24_L	GP Timer1	0x3C0	0x4804 03C0	1	0x4802 8000	0xC:4K
REGION_24_H		0x3C4	0x4804 03C4			
REGION_25_L	L4TA7	0x3C8	0x4804 03C8	1	0x4802 9000	0xC:4K
REGION_25_H		0x3CC	0x4804 03CC			
REGION_26_L	Emulation[1]	0x3D0	0x4804 03D0	NA	NA	NA
REGION_26_H		0x3D4	0x4804 03D4			
REGION_27_L	Emulation[1]	0x3D8	0x4804 03D8	NA	NA	NA
REGION_27_H		0x3DC	0x4804 03DC			
REGION_28_L	Display TOP	0x3E0	0x4804 03E0	2	0x4805 0000	0xA:1K
REGION_28_H		0x3E4	0x4804 03E4			
REGION_29_L	Display Control	0x3E8	0x4804 03E8	2	0x4805 0400	0xA:1K
REGION_29_H		0x3EC	0x4804 03EC			
REGION_30_L	Display RFBI	0x3F0	0x4804 03F0	2	0x4805 0800	0xA:1K
REGION_30_H		0x3F4	0x4804 03F4			
REGION_31_L	Display encoder	0x3F8	0x4804 03F8	2	0x4805 0C00	0xA:1K
REGION_31_H		0x3FC	0x4804 03FC			
REGION_32_L	L4TA10	0x400	0x4804 0400	2	0x4805 1000	0xC:4K
REGION_32_H		0x404	0x4804 0404			
REGION_33_L	Camera TOP	0x408	0x4804 0408	2	0x4805 2000	0xA:1K
REGION_33_H		0x40C	0x4804 040C			
REGION_34_L	Camera Core	0x410	0x4804 0410	2	0x4805 2400	0xA:1K
REGION_34_H		0x414	0x4804 0414			
REGION_35_L	Camera DMA	0x418	0x4804 0418	2	0x4805 2800	0xA:1K
REGION_35_H		0x41C	0x4804 041C			
REGION_36_L	Camera MMU	0x420	0x4804 0420	2	0x4805 2C00	0xA:1K
REGION_36_H		0x424	0x4804 0424			
REGION_37_L	L4TA11	0x428	0x4804 0428	2	0x4805 3000	0xC:4K
REGION_37_H		0x42C	0x4804 042C			
REGION_38_L	sDMA	0x430	0x4804 0430	2	0x4805 6000	0xC:4K
REGION_38_H		0x434	0x4804 0434			
REGION_39_L	L4TA12	0x438	0x4804 0438	2	0x4805 7000	0xC:4K
REGION_39_H		0x43C	0x4804 043C			

Notes: 1) Reserved for testing and emulation.

Table 6−200. Protection Region 3−124 Summary Table (Continued)

Register Name	Region Name	Offset	Physical Address	Segment	Protected Region Base Address	Protected region size (field value: size in bytes)
REGION_40_L	Reserved	0x440	0x4804 0440	2	0x4805 8000	0xC:4K
REGION_40_H		0x444	0x4804 0444			
REGION_41_L	Reserved	0x448	0x4804 0448	2	0x4805 9000	0xC:4K
REGION_41_H		0x44C	0x4804 044C			
REGION_42_L	Reserved	0x450	0x4804 0450	2	0x4805 A000	0xC:4K
REGION_42_H		0x454	0x4804 0454			
REGION_43_L	Reserved	0x458	0x4804 0458	2	0x4805 B000	0xC:4K
REGION_43_H		0x45C	0x4804 045C			
REGION_44_L	L4TA13	0x460	0x4804 0460	2	0x4805 C000	0xC:4K
REGION_44_H		0x464	0x4804 0464			
REGION_45_L	USB OTG	0x468	0x4804 0468	2	0x4805 E000	0xC:4K
REGION_45_H		0x46C	0x4804 046C			
REGION_46_L	L4TAO4	0x470	0x4804 0470	2	0x4805 F000	0xC:4K
REGION_46_H		0x474	0x4804 0474			
REGION_47_L	Emulation[1]	0x478	0x4804 0478	NA	NA	NA
REGION_47_H		0x47C	0x4804 047C			
REGION_48_L	Emulation[1]	0x480	0x4804 0480	NA	NA	NA
REGION_48_H		0x484	0x4804 0484			
REGION_49_L	Emulation[1]	0x488	0x4804 0488	NA	NA	NA
REGION_49_H		0x48C	0x4804 048C			
REGION_50_L	Emulation[1]	0x490	0x4804 0490	NA	NA	NA
REGION_50_H		0x494	0x4804 0494			
REGION_51_L	Emulation[1]	0x498	0x4804 0498	NA	NA	NA
REGION_51_H		0x49C	0x4804 049C			
REGION_52_L	Emulation[1]	0x4A0	0x4804 04A0	NA	NA	NA
REGION_52_H		0x4A4	0x4804 04A4			
REGION_53_L	Emulation[1]	0x4A8	0x4804 04A8	NA	NA	NA
REGION_53_H		0x4AC	0x4804 04AC			
REGION_54_L	Emulation[1]	0x4B0	0x4804 04B0	NA	NA	NA
REGION_54_H		0x4B4	0x4804 04B4			
REGION_55_L	Emulation[1]	0x4B8	0x4804 04B8	NA	NA	NA
REGION_55_H		0x4BC	0x4804 04BC			

Notes: 1) Reserved for testing and emulation.

Table 6−200. Protection Region 3−124 Summary Table (Continued)

Register Name	Region Name	Offset	Physical Address	Segment	Protected Region Base Address	Protected region size (field value: size in bytes)
REGION_56_L	Emulation[1]	0x4C0	0x4804 04C0	NA	NA	NA
REGION_56_H		0x4C4	0x4804 04C4			
REGION_57_L	UART1	0x4C8	0x4804 04C8	3	0x4806 A000	0xC:4K
REGION_57_H		0x4CC	0x4804 04CC			
REGION_58_L	L4TA19	0x4D0	0x4804 04D0	3	0x4806 B000	0xC:4K
REGION_58_H		0x4D4	0x4804 04D4			
REGION_59_L	UART2	0x4D8	0x4804 04D8	3	0x4806 C000	0xC:4K
REGION_59_H		0x4DC	0x4804 04DC			
REGION_60_L	L4TA20	0x4E0	0x4804 04E0	3	0x4806 D000	0xC:4K
REGION_60_H		0x4E4	0x4804 04E4			
REGION_61_L	UART3	0x4E8	0x4804 04E8	3	0x4806 E000	0xC:4K
REGION_61_H		0x4EC	0x4804 04EC			
REGION_62_L	L4TA21	0x4F0	0x4804 04F0	3	0x4806 F000	0xC:4K
REGION_62_H		0x4F4	0x4804 04F4			
REGION_63_L	I2C1	0x4F8	0x4804 04F8	3	0x4807 0000	0xC:4K
REGION_63_H		0x4FC	0x4804 04FC			
REGION_64_L	L4TAO5	0x500	0x4804 0500	3	0x4807 1000	0xC:4K
REGION_64_H		0x504	0x4804 0504			
REGION_65_L	I2C2	0x508	0x4804 0508	3	0x4807 2000	0xC:4K
REGION_65_H		0x50C	0x4804 050C			
REGION_66_L	L4TAO6	0x510	0x4804 0510	3	0x4807 3000	0xC:4K
REGION_66_H		0x514	0x4804 0514			
REGION_67_L	McBSP1	0x518	0x4804 0518	3	0x4807 4000	0xC:4K
REGION_67_H		0x51C	0x4804 051C			
REGION_68_L	L4TAO7	0x520	0x4804 0520	3	0x4807 5000	0xC:4K
REGION_68_H		0x524	0x4804 0524			
REGION_69_L	McBSP2	0x528	0x4804 0528	3	0x4807 6000	0xC:4K
REGION_69_H		0x52C	0x4804 052C			
REGION_70_L	L4TAO8	0x530	0x4804 0530	3	0x4807 7000	0xC:4K
REGION_70_H		0x534	0x4804 0534			
REGION_71_L	WD Timer3	0x538	0x4804 0538	1	0x4802 4000	0xC:4K
REGION_71_H		0x53C	0x4804 053C			

Notes: 1) Reserved for testing and emulation.

Table 6−200. Protection Region 3−124 Summary Table (Continued)

Register Name	Region Name	Offset	Physical Address	Segment	Protected Region Base Address	Protected region size (field value: size in bytes)
REGION_72_L	L4TA5	0x540	0x4804 0540	1	0x4802 5000	0xC:4K
REGION_72_H		0x544	0x4804 0544			
REGION_73_L	Reserved	0x548	0x4804 0548	NA	NA	NA
REGION_73_H		0x54C	0x4804 054C			
REGION_74_L	L4 TA6	0x550	0x4804 0550	1	0x4802 7000	0xC:4K
REGION_74_H		0x554	0x4804 0554			
REGION_75_L	GP Timer 2	0x558	0x4804 0558	1	0x4802 A000	0xC:4K
REGION_75_H		0x55C	0x4804 055C			
REGION_76_L	L4TA8	0x560	0x4804 0560	1	0x4802 B000	0xC:4K
REGION_76_H		0x564	0x4804 0564			
REGION_77_L	GP Timer 3	0x568	0x4804 0568	3	0x4807 8000	0xC:4K
REGION_77_H		0x56C	0x4804 056C			
REGION_78_L	L4TA22	0x570	0x4804 0570	3	0x4807 9000	0xC:4K
REGION_78_H		0x574	0x4804 0574			
REGION_79_L	GP Timer 4	0x578	0x4804 0578	3	0x4807 A000	0xC:4K
REGION_79_H		0x57C	0x4804 057C			
REGION_80_L	L4TA23	0x580	0x4804 0580	3	0x4807 B000	0xC:4K
REGION_80_H		0x584	0x4804 0584			
REGION_81_L	GP Timer 5	0x588	0x4804 0588	3	0x4807 C000	0xC:4K
REGION_81_H		0x58C	0x4804 058C			
REGION_82_L	L4TA24	0x590	0x4804 0590	3	0x4807 D000	0xC:4K
REGION_82_H		0x594	0x4804 0594			
REGION_83_L	GP Timer 6	0x598	0x4804 0598	3	0x4807 E000	0xC:4K
REGION_83_H		0x59C	0x4804 059C			
REGION_84_L	L4TA25	0x5A0	0x4804 05A0	3	0x4807 F000	0xC:4K
REGION_84_H		0x5A4	0x4804 05A4			
REGION_85_L	GP Timer7	0x5A8	0x4804 05A8	3	0x4808 0000	0xC:4K
REGION_85_H		0x5AC	0x4804 05AC			
REGION_86_L	L4TA26	0x5B0	0x4804 05B0	3	0x4808 1000	0xC:4K
REGION_86_H		0x5B4	0x4804 05B4			
REGION_87_L	GPTimer8	0x5B8	0x4804 05B8	3	0x4808 2000	0xC:4K
REGION_87_H		0x5BC	0x4804 05BC			

Notes: 1) Reserved for testing and emulation.

Table 6−200. Protection Region 3−124 Summary Table (Continued)

Register Name	Region Name	Offset	Physical Address	Segment	Protected Region Base Address	Protected region size (field value: size in bytes)
REGION_88_L	L4TA27	0x5C0	0x4804 05C0	3	0x4808 3000	0xC:4K
REGION_88_H		0x5C4	0x4804 05C4			
REGION_89_L	GP Timer 9	0x5C8	0x4804 05C8	3	0x4808 4000	0xC:4K
REGION_89_H		0x5CC	0x4804 05CC			
REGION_90_L	L4TA28	0x5D0	0x4804 05D0	3	0x4808 5000	0xC:4K
REGION_90_H		0x5D4	0x4804 05D4			
REGION_91_L	GP Timer 10	0x5D8	0x4804 05D8	3	0x4808 6000	0xC:4K
REGION_91_H		0x5DC	0x4804 05DC			
REGION_92_L	L4TA29	0x5E0	0x4804 05E0	3	0x4808 7000	0xC:4K
REGION_92_H		0x5E4	0x4804 05E4			
REGION_93_L	GP Timer 11	0x5E8	0x4804 05E8	4	0x4808 8000	0xC:4K
REGION_93_H		0x5EC	0x4804 05EC			
REGION_94_L	L4TA30	0x5F0	0x4804 05F0	4	0x4808 9000	0xC:4K
REGION_94_H		0x5F4	0x4804 05F4			
REGION_95_L	GP Timer 12	0x5F8	0x4804 05F8	4	0x4808 A000	0xC:4K
REGION_95_H		0x5FC	0x4804 05FC			
REGION_96_L	L4TA31	0x600	0x4804 0600	4	0x4808 B000	0xC:4K
REGION_96_H		0x604	0x4804 0604			
REGION_97_L	EAC	0x608	0x4804 0608	4	0x4809 0000	0xC:4K
REGION_97_H		0x60C	0x4804 060C			
REGION_98_L	L4TA32	0x610	0x4804 0610	4	0x4809 1000	0xC:4K
REGION_98_H		0x614	0x4804 0614			
REGION_99_L	FAC	0x618	0x4804 0618	4	0x4809 2000	0xC:4K
REGION_99_H		0x61C	0x4804 061C			
REGION_100_L	L4TA33	0x620	0x4804 0620	4	0x4809 3000	0xC:4K
REGION_100_H		0x624	0x4804 0624			
REGION_101_L	IPC	0x628	0x4804 0628	4	0x4809 4000	0xC:4K
REGION_101_H		0x62C	0x4804 062C			
REGION_102_L	L4TA34	0x630	0x4804 0630	4	0x4809 5000	0xC:4K

Notes: 1) Reserved for testing and emulation.

Table 6−200. Protection Region 3−124 Summary Table (Continued)

Register Name	Region Name	Offset	Physical Address	Segment	Protected Region Base Address	Protected region size (field value: size in bytes)
REGION_ 102_H		0x634	0x4804 0634			
REGION_ 103_L	SPI1	0x638	0x4804 0638	4	0x4809 8000	0xC:4K
REGION_ 103_H		0x63C	0x4804 063C			
REGION_ 104_L	L4TA35	0x640	0x4804 0640	4	0x4809 9000	0xC:4K
REGION_ 104_H		0x644	0x4804 0644			
REGION_ 105_L	SPI2	0x648	0x4804 0648	5	0x4809 A000	0xC:4K
REGION_ 105_H		0x64C	0x4804 064C			
REGION_ 106_L	L4TA36	0x650	0x4804 0650	5	0x4809 B000	0xC:4K
REGION_ 106_H		0x654	0x4804 0654			
REGION_ 107_L	MMCSDIO	0x658	0x4804 0658	5	0x4809 C000	0xC:4K
REGION_ 107_H		0x65C	0x4804 065C			
REGION_ 108_L	L4TA09	0x660	0x4804 0660	5	0x4809 D000	0xC:4K
REGION_ 108_H		0x664	0x4804 0664			
REGION_ 109_L	Reserved	0x668	0x4804 0668	NA	NA	NA
REGION_ 109_H		0x66C	0x4804 066C			
REGION_ 110_L	Reserved	0x670	0x4804 0670	NA	NA	NA
REGION_ 110_H		0x674	0x4804 0674			
REGION_ 111_L	Reserved	0x678	0x4804 0678	5	0x480A 0000	0xC:4K
REGION_ 111_H		0x67C	0x4804 067C			

Notes: 1) Reserved for testing and emulation.

Table 6−200. Protection Region 3−124 Summary Table (Continued)

Register Name	Region Name	Offset	Physical Address	Segment	Protected Region Base Address	Protected region size (field value: size in bytes)
REGION_112_L	L4TAO11	0x680	0x4804 0680	5	0x480A 1000	0xC:4K
REGION_112_H		0x684	0x4804 0684			
REGION_113_L	Reserved	0x688	0x4804 0688	5	0x480A 2000	0xC:4K
REGION_113_H		0x68C	0x4804 068C			
REGION_114_L	L4TA012	0x690	0x4804 0690	5	0x480A 3000	0xC:4K
REGION_114_H		0x694	0x4804 0694			
REGION_115_L	Reserved	0x698	0x4804 0698	5	0x480A 4000	0xC:4K
REGION_115_H		0x698	0x4804 0698			
REGION_116_L	L4TAO13	0x6A0	0x4804 06A0	5	0x480A 5000	0xC:4K
REGION_116_H		0x6A4	0x4804 06A4			
REGION_117_L	Reserved	0x6A8	0x4804 06A8	5	0x480A 6000	0xC:4K
REGION_117_H		0x6AC	0x4804 06AC			
REGION_118_L	L4TA37	0x6B0	0x4804 06B0	5	0x480A 7000	0xC:4K
REGION_118_H		0x6B4	0x4804 06B4			
REGION_119_L	Reserved	0x6B8	0x4804 06B8	5	0x480A 8000	0xD:8K
REGION_119_H		0x6BC	0x4804 06BC			
REGION_120_L	L4TA38	0x6C0	0x4804 06C0	5	0x480A 9000	0xC:4K
REGION_120_H		0x6C4	0x4804 06C4			
REGION_121_L	Reserved	0x6C8	0x4804 06C8	NA	NA	NA

Notes: 1) Reserved for testing and emulation.

Table 6−200. Protection Region 3−124 Summary Table (Continued)

Register Name	Region Name	Offset	Physical Address	Segment	Protected Region Base Address	Protected region size (field value: size in bytes)
REGION_121_H		0x6CC	0x4804 06CC			
REGION_122_L	Reserved	0x6D0	0x4804 06D0	NA	NA	NA
REGION_122_H		0x6D4	0x4804 06D4			
REGION_123_L	HDQ/1−Wire	0x6D8	0x4804 06D8	5	0x480B 2000	0xC:4K
REGION_123_H		0x6DC	0x4804 06DC			
REGION_124_L	L4TA39	0x6E0	0x4804 06E0	5	0x480B 3000	0xC:4K
REGION_124_H		0x6E4	0x4804 06E4			

Notes: 1) Reserved for testing and emulation.

Table 6−201. REGION_3−124_L Register

Address Offset	See Table 6−200.		
Physical Address	See Table 6−200	**Instance**	L4AP
Description	Region 3−124 block − LSBs		
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								BASE																							

Bits	Field Name	Description	Type	Reset
31:24	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x00
23:0	BASE	Region − Base address inside segment	R	See Table 6−200

Table 6−202. REGION_3−124_H Register

Address Offset	0x31C		
Physical Address	0x4804 031C	**Instance**	L4AP
Description	Region 3 control block MSBs		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MADDRSPACE				SEGMENT_ID				RESERVED	PROT_GROUP_ID			BYTE_GROUP_ID			RESERVED		PHI_target_ID							RESERVED		SIZE					ENABLE

Bits	Field Name	Description	Type	Reset
31:28	MADDRSPACE	Unused	R	NA
27:24	SEGMENT_ID	Segment ID of the region	R	0x0
23	Reserved	Read returns 0.	R	0
22:20	PROT_GROUP_ID	Protection group ID	RW	0x7
19:17	BYTE_DATA_WIDTH	Target OCP data byte width	R	0x2
16:15	Reserved	Read returns 0.	R	0x0
14:8	PHI_target_ID	Physical target ID, fixed value for each region, no functional use for this value	R	NA
7:6	Reserved	Read returns 0.	R	0x0
5:1	SIZE	Region size (address bits): 2^SIZE = Region byte size	R	See Table 6–200

Table 6–202. REGION_3–124_H (Continued)

Bits	Field Name	Description	Type	Reset
0	ENABLE	Enable access to region. 0x0: Disable access to the region: Read access to region area returns 0. Write access to region area has no effect. 0x1: Enable access to the region.	R	1

6.5.4 L4IA

Table 6–203 lists the L4IA registers. Table 6–204 through Table 6–207 describe the register bits.

Table 6–203. L4IA Registers

Register Name	Type	Register Width (Bits)	Physical Address
COMPONENT	R	32	0x4804 0800
AGENT_CONTROL	RW	32	0x4804 0820
AGENT_STATUS	RW	32	0x4804 0828
ERROR_LOG	RW	32	0x4804 0858

Table 6−204. COMPONENT Register

Address Offset	0x000		
Physical Address	0x4804 0800	**Instance**	L4IA
Description	Logs information about L4 revision code and interconnect 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
CODE																REV															

Bits	Field Name	Description	Type	Reset
31:16	CODE	Interconnect code	R	
15:0	REV	L4 revision code	R	

Table 6−205. AGENT_CONTROL Register

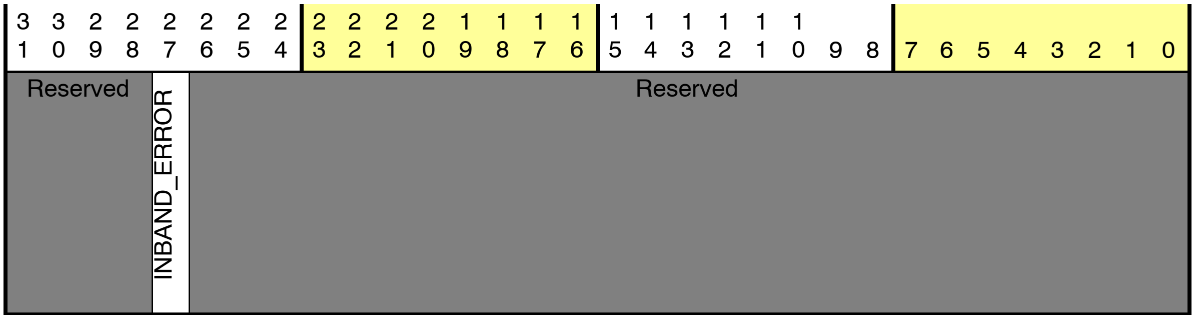
Address Offset	0x020		
Physical Address	0x4804 0820	**Instance**	L4IA
Description	L4 IA agent control register		
Type	RW		

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				INBAND_ERROR_REP	Reserved																										

Bits	Field Name	Description	Type	Reset
31:28	Reserved	Read returns 0.	R	0x0
27	INBAND_ ERROR_REP	Initiator interface error−reporting enable	RW	1
26:0	Reserved	Write 0 for future compatibility. Read returns 0.	R	0x0000000

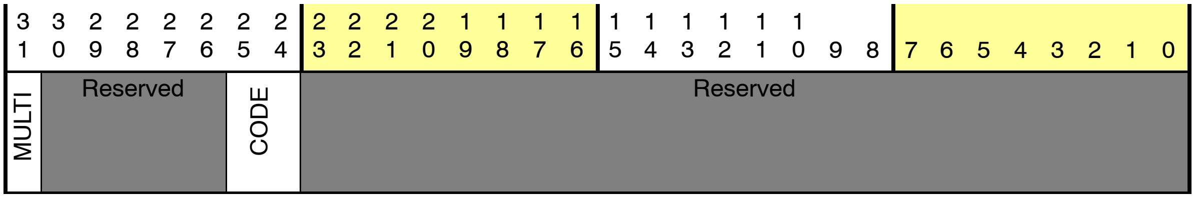
Table 6−206. AGENT_STATUS Register

Address Offset	0x028		
Physical Address	0x4804 0828	**Instance**	L4IA
Description	Agent status		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:28	Reserved	Read returns 0.	R	0x0
27	INBAND_ ERROR	Error status	RW	0
26:0	Reserved	Read returns 0.	R	0x0000000

Table 6−207. ERROR_LOG Register



Bits	Field Name	Description	Type	Reset
31	MULTI	Multiple errors detected	RW	0
30:26	Reserved	Read returns 0.	R	0x00
25:24	CODE	Initiator request error code	RW	0x0
23:0	Reserved	Read returns 0.	R	0x000000

6.5.5 L4LA

Table 6−208 lists the L4LA1 registers. Table 6−209 through Table 6−218 describe the register bits.

Table 6−208. L4LA1 Register Summary

Register Name	Type	Register Width (Bits)	Physical Address
COMPONENT	R	32	0x4804 1000
NETWORK	R	32	0x4804 1014
INITIATOR_INFO_L	RW	32	0x4804 1018
INITIATOR_INFO_H	RW	32	0x4804 101C
NETWORK_CONTROL_L	RW	32	0x4804 1020
NETWORK_CONTROL_H	RW	32	0x4804 1024
FLAG_MASK_L	RW	32	0x4804 1100
FLAG_MASK_H	RW	32	0x4804 1104

Table 6−208. L4LA1 Register Summary (Continued)

Register Name	Type	Register Width (Bits)	Physical Address
FLAG_STATUS_L	R	32	0x4804 1110
FLAG_STATUS_H	RW	32	0x4804 1114

Table 6−209. COMPONENT Register

Address Offset	0x000
Physical Address	0x4804 1000
Instance	L4LA
Description	Logs information about L4 revision code and interconnect 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
CODE																REV															

Bits	Field Name	Description	Type	Reset
31:16	CODE	Interconnect code	R	
15:0	REV	L4 revision code	R	

Table 6−210. NETWORK Register

Address Offset	0x014
Physical Address	0x4804 1014
Instance	L4LA
Description	Logs information about the vendor 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
ID																															

Bits	Field Name	Description	Type	Reset
31:0	ID	ID of the interconnect	R	

Table 6−211. INITIATOR_INFO_L Register

Address Offset	0x018
Physical Address	0x4804 1018
Instance	L4LA
Description	Request information permission configuration for region 0 − LSBs − Reserved for future use
Type	RW

31	30	29	28	27	26	25	24	23	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			

Table 6−211. INITIATOR_INFO_L Register (Continued)

Bits	Field Name	Description	Type	Reset
31:28	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x0
27:24	PROT_GROUPS	Number of protection groups	R	0x8

Bits	Field Name	Description	Type	Reset
23:16	NUMBER_ REGIONS	Number of regions	R	0x7D
15:4	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x000
3:0	SEGMENTS	Number of segments	R	0x6

Table 6–212. INITIATOR_INFO_H Register

Address Offset	0x01C		
Physical Address	0x4804 101C	**Instance**	L4LA
Description	Read permission configuration for region 0 – MSBs		
Type	RW		

31	30	29	28	27	26	25	24	23	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			RESERVED			ADDR_WIDTH				

Bits	Field Name	Description	Type	Reset
31:19	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x000000
18:16	THREADS	Number of initiator threads	R	0x4
15	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0
14:12	CONNID_ WIDTH	L4_CONNID encoding size (bit)	R	0x4
11	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0
10:8	BYTE_DATA_ WIDTH	Initiator subsystem data width 0x1: 16 bits 0x2: 32 bits	R	0x2
7:5	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x0
4:0	ADDR_WIDTH	Initiator subsystem address width: Potential L4 address space 1M byte (only 768K allocated)	R	0x14

Table 6–213. NETWORK_CONTROL_L Register

Address Offset	0x020		
Physical address	0x4804 1020	**Instance**	L4LA
Description	Network control information – LSBs		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					TIMEOUT_BASE			Reserved							

Bits	Field Name	Description	Type	Reset
31:11	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x000000
10:8	TIMEOUT_ BASE	Time-out period base cycles 0x0: Time-out disabled 0x1: 26 clock cycles 0x2: 28 clock cycles 0x3: 210 clock cycles 0x4: 212 clock cycles	R	0x4
7:0	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x00

Table 6–214. NETWORK_CONTROL_H Register

Address Offset	0x024		
Physical Address	0x4804 1024	**Instance**	L4LA
Description	Network control information – MSBs		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							CLOCKGATE_DISABLE	RESERVED			THREAD0_PRI	Reserved											Reserved								

Table 6–214. NETWORK_CONTROL_H Register (Continued)

Bits	Field Name	Description	Type	Reset
31:25	Reserved	Read returns 0.	R	0x00
24	CLOCKGATE_ DISABLE	Clock–gating disable (1 – No gating)	RW	0
23:21	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x0
20	THREAD0_PRI	Thread priority (0 – equal)	R	1
19:9	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x000
8:0	Reserved	Read returns 0.	R	0x000

Table 6–215. FLAG_MASK_L Register

Address Offset	0x100		
Physical Address	0x4804 1100	**Instance**	L4LA
Description	Mask LSBs		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MASK																															

Bits	Field Name	Description	Type	Reset
31:0	MASK	Composite flag mask LSBs	RW	0xFFFFFFFF

Table 6–216. FLAG_MASK_H Register

Address Offset	0x104		
Physical Address	0x4804 1104	**Instance**	L4LA
Description	Mask MSBs		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						MASK																									

Bits	Field Name	Description	Type	Reset
31:26	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x00
25:0	MASK	Composite flag mask MSBs	RW	0x3FFFFFF

Table 6–217. FLAG_STATUS_L Register

Address Offset	0x110		
Physical Address	0x4804 1110	**Instance**	L4LA
Description	Status LSBs		
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
STATUS																															

Bits	Field Name	Description	Type	Reset
31:0	STATUS	Composite flag status (before masking) LSBs	R	0x00000000

Table 6–218. FLAG_STATUS_H Register

Address Offset	0x114		
Physical Address	0x4804 1114	**Instance**	L4LA
Description	Status MSBs		
Type	RW		

31	30	29	28	27	26	25	24	23	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
31:26	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x00
25:0	STATUS	Composite flag status (before masking) MSBs	R	0x0000000

6.6 Guidelines for OMAP2420 Device Interconnect Error Handling

It is recommended that the MPU run centralized error-management software. Three situations can activate the software:

- An access from the MPU (ARM1136) to a system resource generates an error in the OMAP2420 interconnect. The access is generally nonposted (that is, either a read or a nonposted write, even if in some very specific cases, a posted write can also return an in-band error). In that case, an in-band error response is propagated back to the MPU port, where it generates a bus error exception, which must be treated by the error-management handler.
- An access from any initiator, including the MPU, generates an error reported using the L3 interrupt error line to the MPU subsystem INTC (M_IRQ_10). The same exception handler must be called.
- An access from any initiator, excluding the MPU, issues a transaction that returns an in-band error. The initiator module (for instance, the camera core or the system DMA) can assert a private interrupt line to report the error, if that capability is enabled. That interrupt is generally handled by the device driver for that module. The device driver must call the global error management handler to restore the interconnect setting; device drivers must not access interconnect register resources.

This means that the same error can be reported multiple times, and the software is responsible for handling such situations correctly. The third possible situation described can generally be avoided by not using the interrupt capability of the initiator modules and by configuring the interconnect so that any ERR response is also reported on the out-of-band L3 interrupt line.

All time-out capabilities in the interconnect must be enabled to avoid situations resulting in a deadlock condition.

Any error detected in the interconnect is potentially identified to the MPU error−management handler. The following sequence is a typical error-management software routine:

1) Scan all error−logging registers in the L3 interconnect:
    - TASTATE in the LLRC TAs
    - SBIMSTATE in the SB IM agents
    - SBTMSTATE in the SB TM agents
2) Identify registers that have logged error conditions (can be multiple errors):
    - Errors detected in LLRC GPMC or SMS TAs:
        - Check the module logging registers to determine what error condition occurred.
        - Clean up all logging registers (first at module level, then in the interconnect).

- Errors detected in LLRC OCM−RAM or OCM−ROM TAs:
    - Use the extended error−logging registers to get information about the error.
    - Clean up the logging registers.
- Errors detected in an SB IM or a SB TM (excluding the L4 TM):
    - Use the extended error−logging registers to get information about the error.
    - Clean up the logging registers.
- Errors detected in SB L4 TM:
    - Read the L4IA AGENT_STATUS.
    - Scan the L4TA AGENT_STATUS registers in all L4 target agents.
    - Identify every register that logged an error.
    - Clean up the logging registers, first in the L4 interconnect, then in the L3 interconnect.

3) When all registers containing logged errors have been cleaned up, the interrupt line is deasserted by the hardware, and the interrupt service can be acknowledged in the interrupt controller.

Specific actions that can be taken in particular error conditions relative to particular cores are out of the scope of this section.

Similarly, complex sequences to recover from error conditions are not discussed here. Error analysis is expected to be done mainly during the development of the platform software.

Chapter 7

Interprocessor Communication

This chapter discusses interprocessor communication (IPC) between the on-chip processors of the OMAP2420 device.

Topic **Page**

7.1 IPC Overview

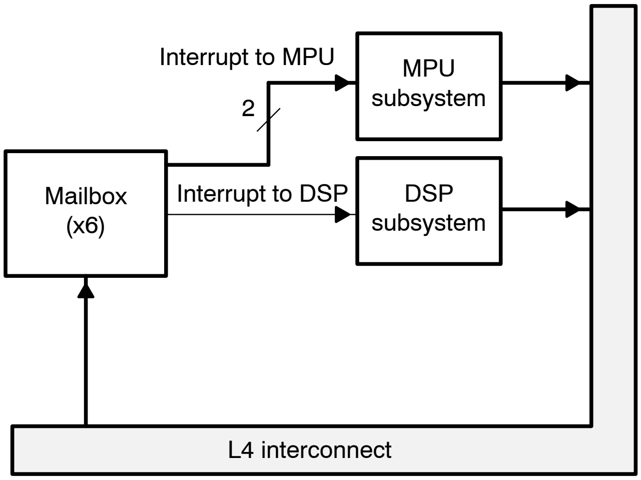
Interprocessor communication (IPC) between the on-chip processors of the OMAP2420 device is accomplished using a queued mailbox-interrupt mechanism that allows software to establish a communication channel between a sender (the sending processor) and a receiver (the receiving processor) through a set of registers and associated interrupt signals controlled by the mailbox peripheral.

The mailbox has the following features:

- ☐ Six mailbox message queues, each allowing 1-way communication between two users
- ☐ Flexible assignment of receiver and sender for each mailbox through interrupt configuration
- ☐ 32-bit message width
- ☐ Four-message FIFO depth for each message queue
- ☐ Message reception and queue-not-full notification by the interrupt
- ☐ Power management and wake-up support
- ☐ Automatic idle mode for power savings

Figure 7−1 is a block diagram of the OMAP2420 IPC.

Figure 7−1. Simplified Block Diagram of OMAP2420 IPC



7.2 Module Integration

The mailbox peripheral receives its reset, clock, and power from the OMAP power, reset, and clock management (PRCM) module.

7.2.1 Power Source

The mailbox peripheral connects to the CORE power domain, which can dynamically switch between low voltage and high voltage.

7.2.2 Clocks

Only one clock domain exists in the mailbox. The mailbox module receives one input clock from the PRCM, CORE_L4_CLK; the precise clock frequency depends on PRCM programming. This clock input is gated internally to the mailbox logic, allowing the clock to be turned off to lower the operating power when the mailbox is not active. For more information, see the description of automatic idle in Section 7.3.4, *Power Management.*

7.2.3 Reset

The mailbox peripheral receives a reset signal from the PRCM module (the CORE_RST signal, which is the reset signal to the CORE power domain). For more information, see Chapter 5, *Power, Reset, and Clock Management.* The mailbox also supports software reset of the module by accessing the mailbox register bit MAILBOX_SYSCONFIG[1] (SOFTRESET bit). For more information about soft reset, see Table 7−10.

7.2.4 Interrupts

Table 7−1 lists interrupt mapping from the mailbox modules to the three internal processors.

Table 7−1. IPC Interrupt Mapping

Interrupt Name	Mapping	Destination
MAIL_U0_MPU_IRQ	M_IRQ_26	MPU interrupt controller
MAIL_U1_DSP_IRQ	D_IRQ_14	DSP subsystem interrupt controller
	SINT5	DSP core interrupt controller
MAIL_U3_MPU_IRQ	M_IRQ_34	MPU interrupt controller

7.2.5 L4 Interconnect Interface

The IPC module has an interface with the L4 interconnect through a dedicated target agent (TA). This TA, which is part of the L4 interconnect, provides status and configuration registers as listed in Table 7−2. For a complete description, see Chapter 6, *Internal Interconnect*.

Table 7−2. L4 Interconnect Registers

Register Name	Type	Register Width (Bits)	Physical Address
COMPONENT	R	32	0x4809 5000
AGENT_CONTROL	R/W	32	0x4809 5020
AGENT_STATUS	R	32	0x4809 5028

7.2.6 Mailbox Register Summary

Table 7−3 summarizes the MLB1 registers.

Table 7−3. MLB1 Register Summary

Register Name	Type	Register Width (Bits)	Physical Address
MAILBOX_REVISION	R	32	0x4809 4000
MAILBOX_SYSCONFIG	RW	32	0x4809 4010
MAILBOX_SYSSTATUS	R	32	0x4809 4014
MAILBOX_MESSAGE_0 – MAILBOX_MESSAGE_5	RW	32	0x4809 4040– 0x4809 4054
MAILBOX_FIFOSTATUS_0 – MAILBOX_FIFOSTATUS_5	R	32	0x4809 4080– 0x4809 4094
MAILBOX_MSGSTATUS_0 – MAILBOX_MSGSTATUS_5	R	32	0x4809 40C0– 0x4809 40D4
MAILBOX_IRQSTATUS_0 – MAILBOX_IRQSTATUS_3	RW	32	0x4809 4100– 0x4809 4118
MAILBOX_IRQENABLE_0 – MAILBOX_IRQENABLE_3	RW	32	0x4809 4104– 0x4809 411C

7.3 Functional Description

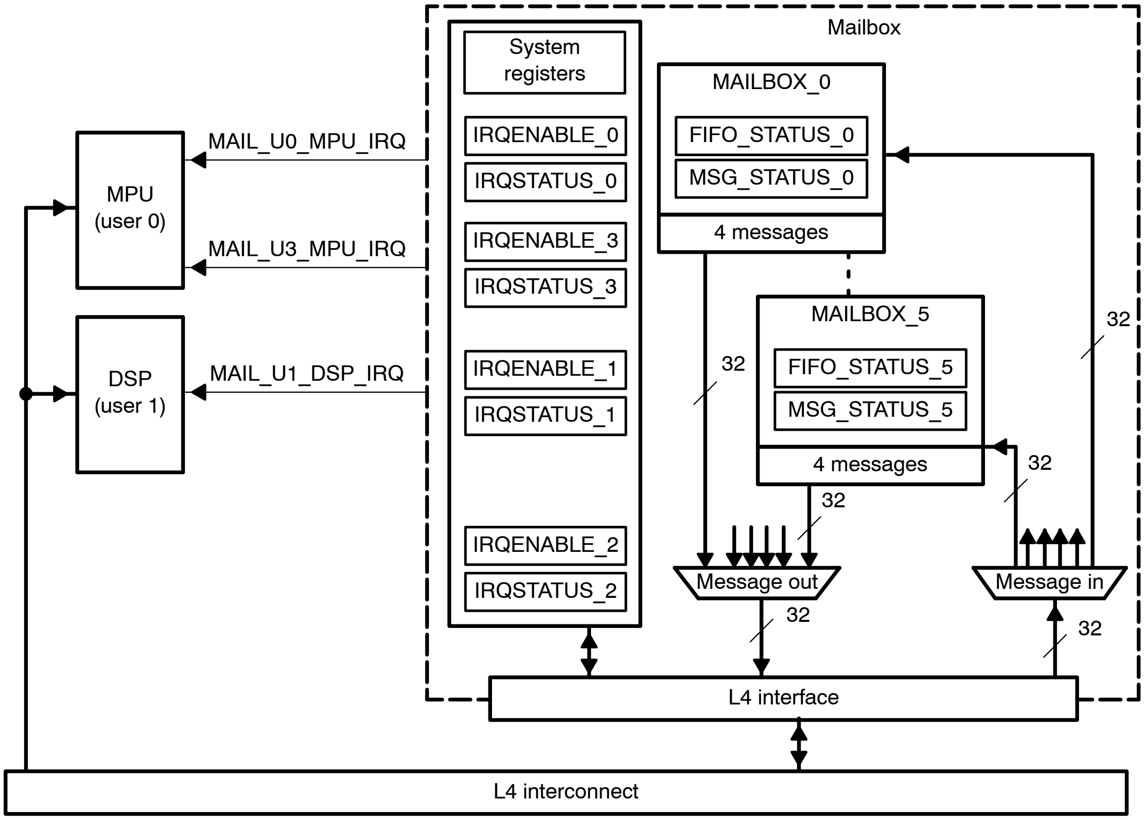
The mailbox module provides a means of communication through message queues between the microprocessor unit (MPU) and the digital signal processor (DSP). There are six individual mailbox submodules, or FIFOs, each capable of being associated with any of three processors through the MAILBOX_IRQENABLE[0:3] registers. The specific user numbers for the three processors are provided in Table 7−4.

Table 7−4. User Number Assignment

User Number	**User**
0, 3	MPU
1	DSP

Each user has a dedicated interrupt signal from the mailbox peripheral and a dedicated pair of interrupt-enable and status registers. A user can query its interrupt status register (MAILBOX_IRQSTATUS[0:3]) in the mailbox through the L4 interconnect to determine the origin of the interrupt. Each user can mask interrupts using its dedicated IRQ enable register (mailbox register bits MAILBOX_IRQENABLE[3:0]). Two interrupts are routed to the MPU from the mailbox so that the MPU serves as user 0 and user 3. This lets the system software separate the imaging and DSP interrupt service routines by dedicating the MPU interrupt to a DSP communication channel.

Figure 7−2. Mailbox Detailed Block Diagram



7.3.1 Mailbox Assignment

A receiver can be assigned to a mailbox by setting the new message interrupt enable bit corresponding to the desired mailbox in the MAILBOX_IRQENABLE_*u* register, where *u* is the number of the receiving user (see Table 7−4). To retrieve a message from the mailbox, the user reads from the MAILBOX_MESSAGE_*m* register, where *m* is the mailbox number (0 to 5).

An alternate method to using the interrupts is for the receiving processor(s) to poll the MAILBOX_FIFOSTATUS_*m* and/or MAILBOX_MSGSTATUS_*m* registers to determine when to send or retrieve a message from the mailbox. This method makes it unnecessary to assign a receiver to a mailbox. (Mailbox assignment is not explicit, so the software must avoid multiple receivers using a single mailbox, which can result in incoherence.)

Senders can be assigned to a mailbox by setting the QUEUE_NOT_FULL interrupt-enable bit corresponding to the desired mailbox in the MAILBOX_IRQENABLE_*u* register, where *u* is the number of the receiving user. However, this is not recommended, because it can cause the sending processor to be continually interrupted. Instead, it is recommended that senders use register polling to first check the status of either the MAILBOX_FIFOSTATUS_*m* or MAILBOX_MSGSTATUS_*m* registers, and to determine whether space is available to write the message to the corresponding MAILBOX_MESSAGE_*m* register. The QUEUE_NOT_FULL interrupt should be used by the sender when the initial mailbox status check indicates that it is full; in this case, the sender can enable the QUEUE_NOT_FULL interrupt for that mailbox in its MAILBOX_IRQENABLE_*u* register. This allows the sender to be notified by interrupt only when a FIFO queue has at least one available entry.

The status of the new message and QUEUE_NOT_FULL interrupts for a user can be determined by reading the MAILBOX_IRQSTATUS_*u* register, and an interrupt can be acknowledged (and subsequently cleared) by writing 1 to the corresponding bit in the same register location. See the MPU example in Section 7.4.4, *Example: MPU Communication to/from the Imaging and Video Accelerator and DSP*.

Note:

Assigning multiple senders or multiple receivers to the same mailbox is not recommended.

7.3.2 Sending and Receiving Messages

When a 32-bit message is written to the MAILBOX_MESSAGE_*m* register, the message is added to the FIFO queue, unless the queue is already full (four messages), in which case, the message is discarded.

Queue overflow can be avoided by first reading the MAILBOX_FIFOSTATUS_*m* register to check that the mailbox message queue is not full before writing a new message to it.

Reading the MAILBOX_MESSAGE_*m* register returns the message at the beginning of the FIFO queue and removes it from the queue. If the FIFO queue is empty when the MAILBOX_MESSAGE_*m* register is read, 0 is returned.

The new message interrupt is asserted when at least one message is in the mailbox message FIFO queue. The number of messages in the mailbox message FIFO queue can be determined by reading the MAILBOX_MSGSTATUS_*m* register.

7.3.3 16-Bit Register Access

To maintain backward compatibility with the previous version of the mailbox module, the OMAP2420 device allows 16-bit register read and write access. The 16-bit half-words are organized in little-endian fashion; that is, the least-significant 16 bits are at the low address and the most-significant 16 bits are at the high address (low address + 0x02).

All mailbox module registers except for the MAILBOX_MESSAGE_*m* register-can be read or written to directly using individual 16-bit accesses with no restrictions on interleaving. Handling of 16-bit accesses to the MAILBOX_MESSAGE_*m* registers is slightly more restricted (see Section 7.3.3.1, *16-Bit Access to MAILBOX_MESSAGE_m Registers*).

7.3.3.1 16-Bit Access to MAILBOX_MESSAGE_m Registers

While all mailbox module registers can be read or written to directly using individual 16-bit accesses, the MAILBOX_MESSAGE_*m* registers have some restrictions that must be followed for correct operation. The MAILBOX_MESSAGE_*m* registers must always be accessed by either single 32-bit accesses or by two consecutive 16-bit accesses. When using 16-bit accesses to the MAILBOX_MESSAGE_*m* registers, the access order must be least-significant half-word first (low address) and most-significant half-word last (high address). This requirement is the result of the update operation by the MESSAGE FIFO of the MAILBOX_MSGSTATUS_*m* registers. Updating of the FIFO queue contents and the associated status registers and possible interrupt generation occurs only when the most-significant 16 bits of a MAILBOX_MESSAGE_*m* are accessed.

Note:

When using 16-bit accesses, it is critical to ensure that the mailbox being used has only one assigned receiver and only one assigned sender.

7.3.4 Power Management

To conserve system power, the mailbox module can shut off its clock internally or, at the request of the clock manager in the PRCM, enter a low-power idle mode. Power−management behavior for the mailbox module is controlled by the MAILBOX_SYSCONFIG register. Section 7.3.4.1, *Internal Clock-Gating Control*, and Section 7.3.4.2, *External Clock-Gating Control*, describe how these modes work and how to configure them.

7.3.4.1 Internal Clock-Gating Control

To conserve power, the mailbox supports an automatic idle mode when no activity is detected on the mailbox L4 interconnect interface. This auto-idle mode is enabled or disabled by the MAILBOX_SYSCONFIG[0] AUTOIDLE bit. When auto-idle mode is enabled (AUTOIDLE bit asserted), when there is no activity on the L4 interconnect interface, the mailbox clock is disabled internally to the module, thereby reducing power consumption. When new activity is detected on the L4 interconnect interface, the interconnect clock restarts without a latency penalty. After reset, auto-idle mode is by default disabled. To save power, it is recommended to enable auto-idle mode.

7.3.4.2 External Clock-Gating Control

As part of the OMAP2420 system-wide power-management scheme, the mailbox module supports a communication protocol with the PRCM that allows the PRCM to request that the mailbox enter a low-power state. On mailbox acknowledgment of a low-power mode request from the PRCM, the clock to the mailbox module is gated off at the PRCM clock generator. This low-power mode offers lower power consumption than internal clock-gating, because the clock is disabled at the source. The PRCM register bit CM_ICLKEN1_CORE[30] controls the mailbox clock with a setting of 1 to enable the clock and a setting of 0 to disable the clock. (For details, see Chapter 5, *Power, Reset, and Clock Management.*) The mailbox module can be configured through the MAILBOX_ SYSCONFIG register to be in one of the following acknowledgment modes:

- No-idle mode: The mailbox module never goes to idle state (no acknowledgment is sent to the PRCM on a low-power-mode change request). In this mode, the PRCM times out and removes the idle request to the mailbox.

- Force-idle mode: The mailbox module goes to idle state immediately on receiving a low-power-mode request from the PRCM. (In this mode, software must ensure that there are no asserted output interrupts before requesting that this module go to idle state.)

- Smart-idle mode: After receiving a low-power-mode request from the PRCM, the mailbox module goes to idle state only after all asserted output interrupts are acknowledged.

Table 7−5 shows the configuration of the clock-gating registers.

Table 7−5. Power Mode (Clock-Gating) Registers

Power Management Mode	MAILBOX_SYSCONFIG (Offset 0x010)	
	Bit [4:3]	Bit 0

PRCM idle request	00: Force-idle mode 01: No-idle mode 10: Smart-idle mode 11: Reserved	–
Automatic idle	–	0 = L4 input clock enabled 1 = L4 input clock gated off internal to the mailbox module

Whenever the CORE power domain transitions to a lower power state or when the mailbox clock-enable bit (PRCM register bit CM_ICLKEN_CORE[30]) is deasserted, the PRCM can issue a request to the mailbox to enter idle mode.

The PRCM register bit CM_AUTOIDLE_CORE[30] controls whether the mailbox interface clock is enabled or disabled in sync with the CORE power domain state transition.

Mailbox idle status can be read from PRCM register bit CM_IDLEST1_CORE[30] (1: mailboxes can be accessed, 0: mailboxes cannot be accessed).

7.4 Programming Model

7.4.1 Assigning Mailboxes

Before communication, mailboxes can be explicitly assigned to a user through the appropriate MAILBOX_IRQENABLE_*u* register, as described in Section 7.3, *Functional Description*. Ensure that only one sender and one receiver are assigned per mailbox.

7.4.2 Mailbox Communication Preparation

Before trying to communicate with another user, the sender must first determine that the mailbox message FIFO queue is not full. This can be accomplished by one of two methods:

- Polling the MAILBOX_FIFOSTATUS_*m* or the MAILBOX_ MSGSTATUS_*m* register to determine whether an open slot is available to write a message
- When the QUEUE_NOT_FULL interrupt was previously enabled, an interrupt to the sender from the MAILBOX_IRQSTATUS_*u* register indicates to the sender that the mailbox has an available slot. (The QUEUE_NOT_FULL interrupt is only to be enabled and used by a sender in the situation where a previous status check found that the mailbox was full.

  In this case, the sender can enable the corresponding MAILBOX_ IRQENABLE_*u* interrupt to be notified by an interrupt when the mailbox is available. After detection, the QUEUE_NOT_FULL interrupt should be disabled by the sender until another full status is detected at the FIFOSTATUS or MSGSTATUS registers.

The receiver can also detect new messages from another user in two ways:

- Polling the MAILBOX_FIFOSTATUS_*m* or the MAILBOX_ MSGSTATUS_*m* registers
- Through a new message interrupt service routine (ISR) (The receiver must enable the appropriate interrupt at the MAILBOX_IRQENABLE_*u* register.)

7.4.3 Mailbox Communication Sequence

Once the sender establishes that a message slot is available in the mailbox FIFO, the following steps can be performed to transmit a message to another user.

1) The sender writes the message to the MAILBOX_MESSAGE_*m* register. This results in the following actions:
   a) The message is put in the FIFO queue, and the MAILBOX_ MSGSTATUS_*m* and MAILBOX_FIFOSTATUS_*m* registers are updated.

   b) If the FIFO queue was empty, a maskable new message interrupt is generated to the receiver to which the mailbox is allocated; otherwise, the interrupt is already asserted and remains so.

2) The receiver can use an ISR or it can poll the MAILBOX_FIFOSTATUS_*m* or the MAILBOX_MSGSTATUS_*m* registers to detect new messages.

   a) If using interrupts, the receiver enters the ISR when it detects the new message interrupt; it checks the MAILBOX_IRQSTATUS_*u* to determine the source of the interrupt and checks the MAILBOX_MSGSTATUS_*m* registers to determine the number of messages in the FIFO queue.

   b) If polling, the receiver can check the appropriate MAILBOX_ MSGSTATUS_*m* register(s) to determine whether there are any pending messages to read; the receiver can also determine how many messages are available by reading the mailbox register bits MAILBOX_MSGSTATUS[6:0] (NBOFMSGMBM field).

3) When the receiver determines that a message is pending in a mailbox, the receiver repeatedly reads the MAILBOX_MESSAGE_*m* register to remove all messages from the FIFO queue until a read of the MAILBOX_MSGSTATUS_*m* register indicates that no more messages are available.

4) After reading all messages, the receiver can acknowledge the new message interrupt by writing 1 to the appropriate bit in the MAILBOX_ IRQSTATUS_*u* register to clear the interrupt flag before exiting the ISR. (This acknowledgement must be done if using an ISR.)

7.4.4 MPU Communication to/from the Imaging and Video Accelerator and DSP Example

In this example, the OMAP2420 MPU subsystem is configured to communicate with the imaging and video accelerator and DSP subsystems using four mailbox FIFOs. The mailbox is also configured to turn on the auto-idle feature and to acknowledge PRCM low-power-mode requests only after clearing pending interrupts. Figure 7−3 shows this configuration example. Table 7−6 provides the register settings for this example. In this sample configuration, the MPU uses mailbox 0 to send messages to the DSP, and mailbox 2 to send messages to the imaging and video accelerator. The DSP and imaging and video accelerator are thus connected to mailbox 0 and mailbox 2 as receivers, and so these two mailboxes are dedicated communication channels from MPU to DSP and from MPU to imaging and video accelerator, respectively. The MPU communicates to the DSP by alternately checking the status of MAILBOX_0 and, if mailbox 0 is not full, writing to the MAILBOX_MESSAGE_0 register using the L4 interface. Before each write access, the MPU must poll the MAILBOX_MSGSTATUS_0 register to determine when that mailbox is full, and, if the mailbox is full, the MPU can enable the NotFullEnable0MB0 interrupt in the MAILBOX_ IRQENABLE_0 register to be notified when the mailbox is once again available.

Figure 7−3. Example 1 Pictorial

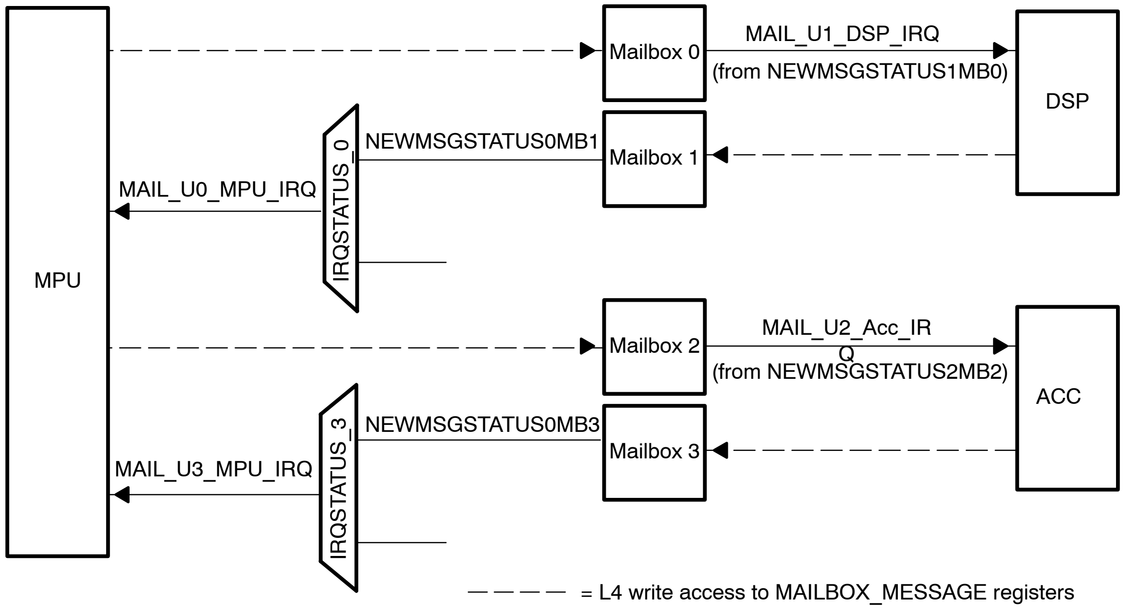


Table 7−6. Register Settings

Register Name	Offset Address	Register Setting	Description
MAILBOX_SYSCONFIG	0x01	0000 0011h	Bits[4:3] = 10, Smart-idle mode Bit 0 = 1, Auto-idle on
MAILBOX_IRQENABLE_0	0x104	0000 0004h	MPU Mailbox 1 new message status IRQ enabled
MAILBOX_IRQENABLE_1	0x10C	0000 0001h	DSP Mailbox 0 new message status IRQ enabled
MAILBOX_IRQENABLE_2	0x114	0000 0010h	Imaging and video accelerator Mailbox 2 new message status IRQ enabled
MAILBOX_IRQENABLE_3	0x11C	0000 0040h	MPU Mailbox 3 new message status IRQ enabled

7.5 Mailbox Registers

7.5.1 Instance Summary

Table 7−7 shows the base address and address space for the MLB1 module instances.

Table 7−7. Instance Summary

Module Name	Base Address	Size
MLB1	0x4809 4000	4K bytes

7.5.2 Register Summary

This section provides information about the MLB module instance in this product. Table 7−8 summarizes the MLB1 registers. Table 7−9 through Table 7−16 describe the registers separately.

Table 7−8. MLB1 Register Summary

Register Name	Type	Register Width (Bits)	Offset Address
MAILBOX_REVISION	R	32	0x00
MAILBOX_SYSCONFIG	RW	32	0x10
MAILBOX_SYSSTATUS	R	32	0x14
MAILBOX_MESSAGE_0 – MAILBOX_MESSAGE_5	RW	32	0x40− 0x54
MAILBOX_FIFOSTATUS_0 – MAILBOX_FIFOSTATUS_5	R	32	0x80− 0x94
MAILBOX_MSGSTATUS_0 – MAILBOX_MSGSTATUS_5	R	32	0xC0− 0xD4
MAILBOX_IRQSTATUS_0 – MAILBOX_IRQSTATUS_3	RW	32	0x100− 0x118
MAILBOX_IRQENABLE_0 – MAILBOX_IRQENABLE_3	RW	32	0x104− 0x11C

7.5.3 Mailbox Register Descriptions

Table 7−9. Mailbox Revision Register

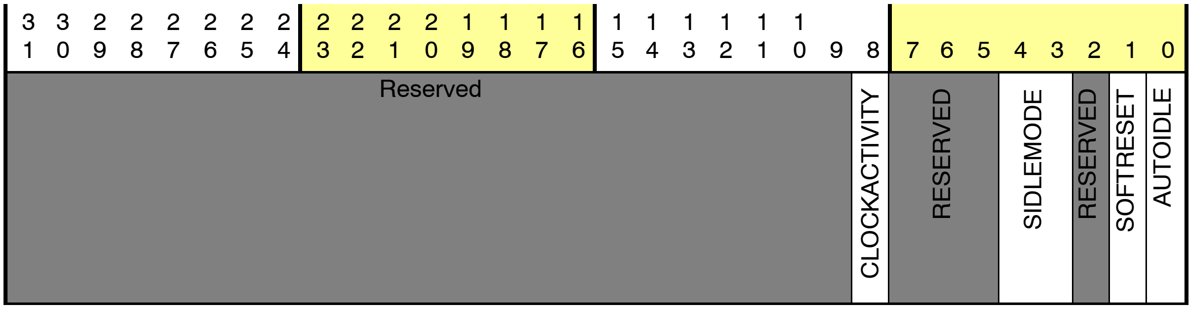
Address Offset	0x0000
Physical Address	0x4809 4000
Description	This register contains the IP revision code.
Type	R

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 Examples: 0x10 for 1.0, 0x21 for 2.1	R	

Table 7−10. Mailbox System Configuration Register

Address Offset	0x0010
Physical Address	0x4809 4010
Description	This register controls the L4 interface parameters.
Type	RW



Bits	Field Name	Description	Type	Reset
31:9	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x000000
8	CLOCKACTIVITY	Clock activity during wake-up mode period. Clock can always be switched off. Read returns 0.	R	0
7:5	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x0
4:3	SIDLEMODE	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 based on the internal activity of the module. 0x3: Reserved. Do not use.	RW	0x0
2	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0
1	SOFTRESET	Software reset. This bit is automatically reset by the hardware. Read 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 L4 interface activity.	RW	0

Table 7−11. Mailbox System Status Register

Address Offset	0x0014
Physical Address	0x4809 4014
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	Read returns 0.	R	0x000000
7:1	Reserved	Read returns 0.	R	0x00
0	RESETDONE	Internal reset monitoring 0x0: Internal module reset in ongoing. 0x1: Reset completed†	R	0[1]

Notes: 1) During reset the value is 0, but the value read just after the reset is 1, because ICLK/FCLK is already operating.

Table 7−12. Mailbox Message Registers

0x40 = MAILBOX_0 message register
0x44 = MAILBOX_1 message register
0x48 = MAILBOX_2 message register
0x4C = MAILBOX_3 message register
0x50 = MAILBOX_4 message register
0x54 = MAILBOX_5 message register
0x58 – 0x7C = Reserved

Address Offset	0x0040−0x0054 in 0x4 byte increments
Physical Address	0x4809 4040−0x4809 4054
Description	The message register stores the next to−be−read message of the mailbox m, where m is MAILBOX_0 through MAILBOX_5.
Type	RW

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MESSAGEVALUEMBM																															

Bits	Field Name	Description	Type	Reset
31:0	MESSAGE VALUEMBM	Message in mailbox m	RW	0x00000000

Table 7−13. Mailbox FIFO Status Register

0x80 = Mailbox 0 FIFO status register
0x84 = Mailbox 1 FIFO status register
0x88 = Mailbox 2 FIFO status register
0x90 = Mailbox 4 FIFO status register
0x94 = Mailbox 5 FIFO status register
0x98 – 0xBC = Reserved

Address Offset	0x0080−0x0094 in 0x4-byte increments
Physical Address	0x4809 4080−0x4809 4094
Description	The FIFO status register contains the status of the mailbox internal FIFO.
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																															FIFOFULLMBM

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Read returns 0.	R	0x00000000
0	FIFOFULLMBM	Full flag for mailbox m	R	0

Table 7−14. Mailbox Message Status Registers

0xC0 = Mailbox 0 message status
0xC4 = Mailbox 1 message status
0xC8 = Mailbox 2 message status
0xD0 = Mailbox 4 message status
0xD4 = Mailbox 5 message status
0xD8 – 0xFC = Reserved

Address Offset	0x00C0−0x00D4 in 0x4-byte increments
Physical Address	0x4809 40C0−0x4809 40D4
Description	The message status register contains the status of the messages in the mailbox.
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																									NBOFMSGMBM						

Bits	Field Name	Description	Type	Reset
31:7	Reserved	Read returns 0.	R	0x0000000
6:0	NBOFMSGMBM	Number of messages in mailbox m	R	0x00

Table 7−15. Mailbox IRQ Status Registers

0x100 = IRQ status register for MPU module (user 0)
0x108 = IRQ status register for DSP module (user 1)
0x110 = IRQ status register for imaging and video accelerator module (user 2)
0x118 = IRQ status register for MPU module (user 3)

Address Offset	0x0100−0x0118 in 0x8-byte increments	**Instance**	MLB1
Physical Address	0x4809 4100−0x4809 4118		
Description	The interrupt status register has the status for each event that can be responsible for the generation of an interrupt to the corresponding user. Writing 1 to a given bit resets this 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																				NOTFULLSTATUSUUMB5	NEWMSGSTATUSUUMB5	NOTFULLSTATUSUUMB4	NEWMSGSTATUSUUMB4	NOTFULLSTATUSUUMB3	NEWMSGSTATUSUUMB3	NOTFULLSTATUSUUMB2	NEWMSGSTATUSUUMB2	NOTFULLSTATUSUUMB1	NEWMSGSTATUSUUMB1	NOTFULLSTATUSUUMB0	NEWMSGSTATUSUUMB0

Bits	Field Name	Description	Type	Reset
31:12	RESERVED	Write 0 for future compatibility. Read returns 0.	RW	0
11	NOTFULLSTATUSUUMB5	NewMessage status bit for user u, mailbox 5	RW	0
10	NEWMSGSTATUSUUMB5	NewMessage status bit for user u, mailbox 5	RW	0
9	NOTFULLSTATUSUUMB4	NotFull status bit for user u, mailbox 4	RW	0
8	NEWMSGSTATUSUUMB4	NewMessage status bit for user u, mailbox 4	RW	0
7	NOTFULLSTATUSUUMB3	NotFull status bit for user u, mailbox 3	RW	0
6	NEWMSGSTATUSUUMB3	NewMessage status bit for user u, mailbox 3	RW	0
5	NOTFULLSTATUSUUMB2	NotFull status bit for user u, mailbox 2	RW	0
4	NEWMSGSTATUSUUMB2	NewMessage status bit for user u, mailbox 2	RW	0
3	NOTFULLSTATUSUUMB1	NotFull status bit for user u, mailbox 1	RW	0
2	NEWMSGSTATUSUUMB1	NewMessage status bit for user u, mailbox 1	RW	0
1	NOTFULLSTATUSUUMB0	NotFull status bit for user u, mailbox 0	RW	0
0	NEWMSGSTATUSUUMB0	NewMessage status bit for user u, mailbox 0	RW	0

Table 7−16.Mailbox IRQ Enable Registers

0x104 = IRQ enable register for MPU module (user 0)
0x10C = IRQ enable register for DSP module (user 1)
0x114 = IRQ enable register for imaging and video accelerator module (user 2)
0x11C = IRQ enable register for MPU module (user 3)

Address Offset	0x0104−0x011C in 0x8-byte increments
Physical Address	0x4809 4104−0x4809 411C
Description	The interrupt−enable register enables masking/unmasking of the module internal interrupt source to the corresponding user.
Type	RW

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOTFULLENABLEUUMB15	NEWMSGENABLEUUMB15	NOTFULLENABLEUUMB14	NEWMSGENABLEUUMB14	NOTFULLENABLEUUMB13	NEWMSGENABLEUUMB13	NOTFULLENABLEUUMB12	NEWMSGENABLEUUMB12	NOTFULLENABLEUUMB11	NEWMSGENABLEUUMB11	NOTFULLENABLEUUMB10	NEWMSGENABLEUUMB10	NOTFULLENABLEUUMB9	NEWMSGENABLEUUMB9	NOTFULLENABLEUUMB8	NEWMSGENABLEUUMB8	NOTFULLENABLEUUMB7	NEWMSGENABLEUUMB7	NOTFULLENABLEUUMB6	NEWMSGENABLEUUMB6	NOTFULLENABLEUUMB5	NEWMSGENABLEUUMB5	NOTFULLENABLEUUMB4	NEWMSGENABLEUUMB4	NOTFULLENABLEUUMB3	NEWMSGENABLEUUMB3	NOTFULLENABLEUUMB2	NEWMSGENABLEUUMB2	NOTFULLENABLEUUMB1	NEWMSGENABLEUUMB1	NOTFULLENABLEUUMB0	NEWMSGENABLEUUMB0

Bits	Field Name	Description	Type	Reset
31	NOTFULL ENABLEUUMB15	NotFull Enable bit for user u, mailbox 15	RW	0
30	NEWMSG ENABLEUUMB15	NewMessage Enable bit for user u, mailbox 15	RW	0
29	NOTFULL ENABLEUUMB14	NotFull Enable bit for user u, mailbox 14	RW	0
28	NEWMSG ENABLEUUMB14	NewMessage Enable bit for user u, mailbox 14	RW	0
27	NOTFULL ENABLEUUMB13	NotFull Enable bit for user u, mailbox 13	RW	0
26	NEWMSG ENABLEUUMB13	NewMessage Enable bit for user u, mailbox 13	RW	0
25	NOTFULL ENABLEUUMB12	NotFull Enable bit for user u, mailbox 12	RW	0
24	NEWMSG ENABLEUUMB12	NewMessage Enable bit for user u, mailbox 12	RW	0
23	NOTFULL ENABLEUUMB11	NotFull Enable bit for user u, mailbox 11	RW	0
22	NEWMSG ENABLEUUMB11	NewMessage Enable bit for user u, mailbox 11	RW	0
21	NOTFULL ENABLEUUMB10	NotFull Enable bit for user u, mailbox 10	RW	0
20	NEWMSG ENABLEUUMB10	NewMessage Enable bit for user u, mailbox 10	RW	0
19	NOTFULLENABLEUUMB9	NotFull Enable bit for user u, mailbox 9	RW	0

Table 7−16. Mailbox IRQ Enable Registers (Continued)

Bits	Field Name	Description	Type	Reset
18	NEWMSGENABLEUUMB9	NewMessage Enable bit for user u, mailbox 9	RW	0
17	NOTFULLENABLEUUMB8	NotFull Enable bit for user u, mailbox 8	RW	0
16	NEWMSGENABLEUUMB8	NewMessage Enable bit for user u, mailbox 8	RW	0
15	NOTFULLENABLEUUMB7	NotFull Enable bit for user u, mailbox 7	RW	0
14	NEWMSGENABLEUUMB7	NewMessage Enable bit for user u, mailbox 7	RW	0
13	NOTFULLENABLEUUMB6	NotFull Enable bit for user u, mailbox 6	RW	0
12	NEWMSGENABLEUUMB6	NewMessage Enable bit for user u, mailbox 6	RW	0
11	NOTFULLENABLEUUMB5	NotFull Enable bit for user u, mailbox 5	RW	0
10	NEWMSGENABLEUUMB5	NewMessage Enable bit for user u, mailbox 5	RW	0
9	NOTFULLENABLEUUMB4	NotFull Enable bit for user u, mailbox 4	RW	0
8	NEWMSGENABLEUUMB4	NewMessage Enable bit for user u, mailbox 4	RW	0
7	NOTFULLENABLEUUMB3	NotFull Enable bit for user u, mailbox 3	RW	0
6	NEWMSGENABLEUUMB3	NewMessage Enable bit for user u, mailbox 3	RW	0
5	NOTFULLENABLEUUMB2	NotFull Enable bit for user u, mailbox 2	RW	0
4	NEWMSGENABLEUUMB2	NewMessage Enable bit for user u, mailbox 2	RW	0
3	NOTFULLENABLEUUMB1	NotFull Enable bit for user u, mailbox 1	RW	0
2	NEWMSGENABLEUUMB1	NewMessage Enable bit for user u, mailbox 1	RW	0
1	NOTFULLENABLEUUMB0	NotFull Enable bit for user u, mailbox 0	RW	0
0	NEWMSGENABLEUUMB0	NewMessage Enable bit for user u, mailbox 0	RW	0

Chapter 8

System Control Module

This chapter describes the system control module (SCM) in the OMAP2420 multimedia device.

Topic **Page**

8.1 System Control Module Overview

The system control module (SCM) allows software to control the static modes supported by the OMAP2420. The SCM is an L4-interconnect-compliant module, with the following exceptions:

- The SCM is sensitive only to the OMAP2420 internal power-on reset. L4 interconnect reset has no impact on the SCM.
- Idle mode, as defined in the L4 interconnect, is hard-coded to force-idle mode.

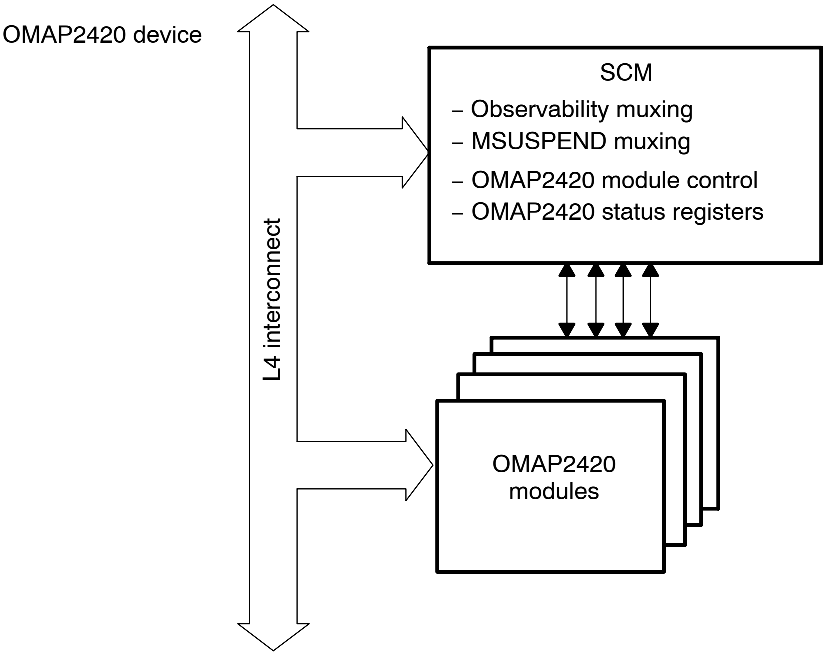
The SCM is the primary point of control for the following areas of the OMAP2420 modules:

- Device status
- Peripheral sensitivity to microprocessor unit (MPU) and/or digital signal processor (DSP) MSUSPEND signals
- Static device configuration
- Debug and observability I/O multiplexing
- Functional I/O multiplexing*
- Pad configuration (pullup or pulldown enable)*

* These functions are detailed in Chapter 24, *Pinout Overview.*

Figure 8−1 is an overview of the SCM.

Figure 8−1. Module Overview



The OMAP2420 system control module primarily implements a bank of registers accessible (read/write) by the software, and some read-only registers that contain status information.

Read/write registers can be static device configuration registers (32-bit read/write registers for module-specific configuration).

8-bit access is also allowed.

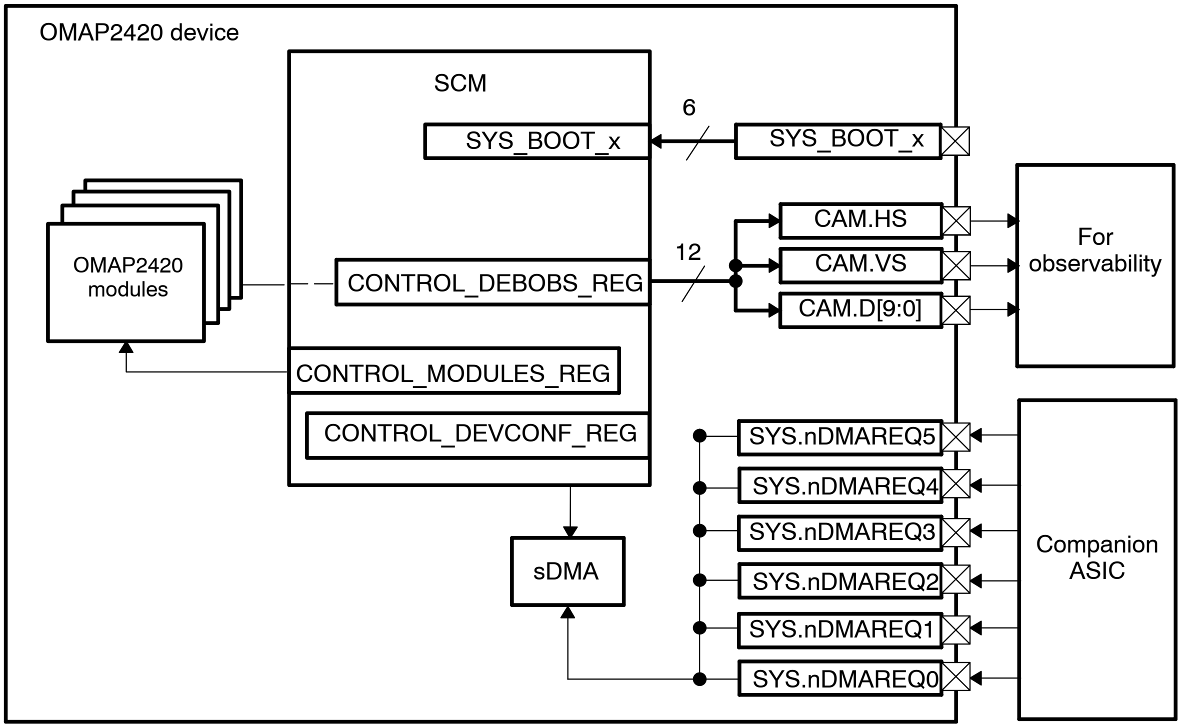
8.2 SCM Environment

8.2.1 Module Interface

The SCM configures other OMAP2420 modules; for example, USB, and McBSP. Except for pad configuration, (see Chapter 24, *Pinout Overview*), the SCM controls the multiplexing of OMAP2420 internal-module signals that are routed to external pins for hardware debugging. The SCM also allows external system DMA (sDMA) request pin sensitivity to be set.

Figure 8−2 shows the OMAP2420 configured to observe OMAP2420 debug signals.

Figure 8−2. Module Application System Overview



The six SYS.BOOT[5:0] pins are sampled and latched onto a status register after a power-on reset. The lower four SYS.BOOT[3:0] pins let you choose the memory space where the MPU reads the first instruction when booting.

The ROM code uses the remaining upper two SYS.BOOT[5:4] pins for peripheral booting and flashing. After booting, these pins can be used for other functions.

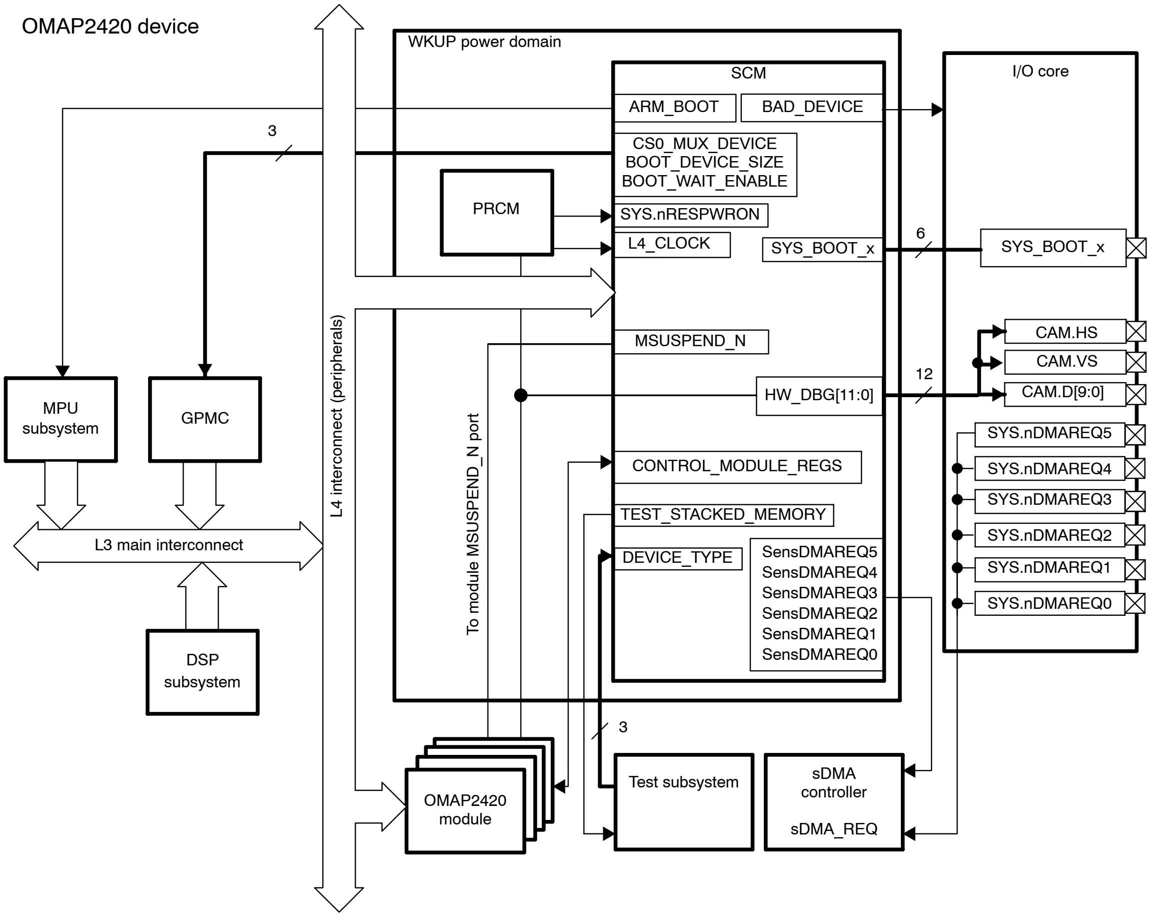
Figure 8−2 describes the CAM.HS, CAM.VS, and CAM.D[9:0] pins when mapped at the OMAP2420 device boundary to observe hardware debug signals from the OMAP2420 modules.

8.3 SCM Integration

The SCM is connected on the L4 interconnect. The power, reset, and clock management (PRCM) module clock, L4_CLOCK, provides the module system clock. The PRCM also provides the reset signal, GLOBAL_PWRON_N, which corresponds to an internal power-on reset.

The SCM is not sensitive to a warm reset or L4 reset.

Figure 8−3. Module Integration Overview



The software sets configuration registers to desired values based on the requested device configuration. The software can set static device configuration registers at any time; the registers are effective immediately.

8.3.1 Clocking, Reset, and Power-Management Scheme

8.3.1.1 Hardware Requests

The SCM does not generate interrupt or wake-up requests.

8.3.1.2 Reset

The SCM is sensitive only to the internal power-on reset and device type.

The internal power-on reset is not a direct image of the power-on reset input pin. The PRCM generates the internal power-on reset signal (GLOBAL_PWRON_N) and activates the internal power-on reset.

Note:

This chapter refers to the internal power-on reset from the perspective of the SCM.

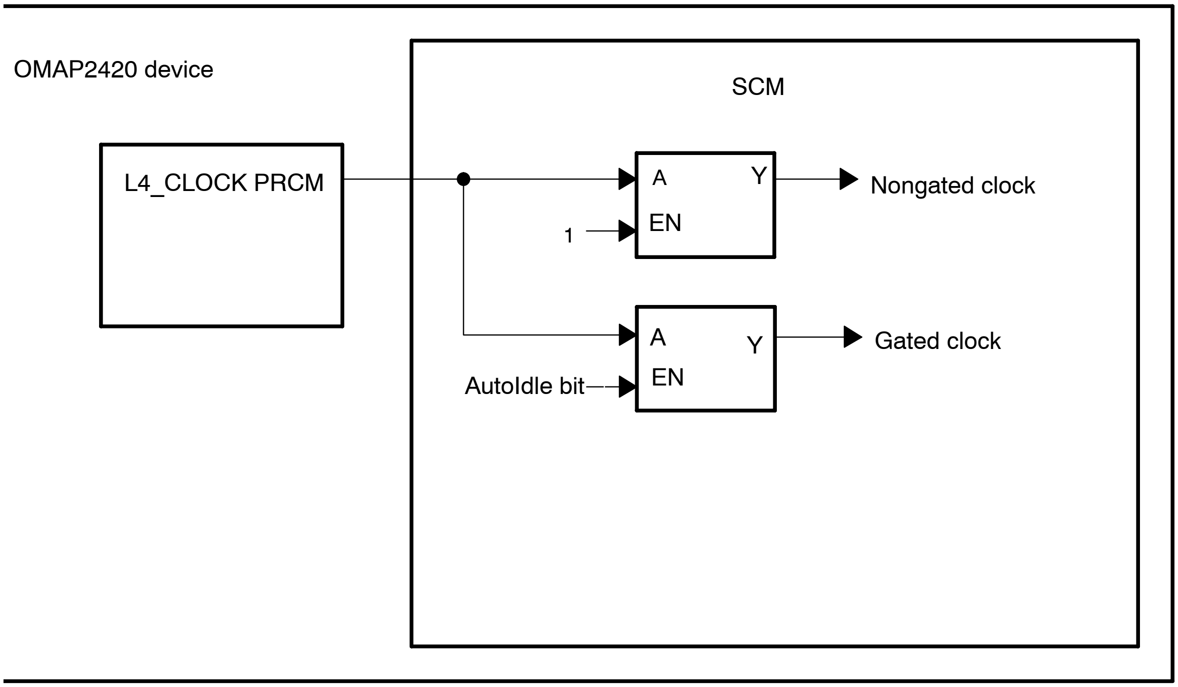
8.3.1.3 Power Management

The SCM is powered by the wake-up (WKUP) power domain.

L4_CLOCK can be gated in the module when there is no access to the module according to SCM register bit CONTROL_SYSCONFIG[0].

In the module, a second clock from L4_CLOCK (nongated) is used to sample the states of input signals to the modules, which are then read through the security status register and the security error status register. Because of this, L4_CLOCK cannot be gated; it must be free-running at all times. Also, this clock is used for signal resynchronization.

Figure 8−4. Internal Clock Implementation



8.3.2 L4 Interconnect Interface

The SCM has an interface with the L4 interconnect through a dedicated target agent (TA). This TA, which is part of the L4 interconnect, provides status and configuration registers as listed in Table 8−1. By default, TA register values are functional, but it is possible to overwrite them. For a complete description, see Chapter 6, *Internal Interconnect*.

Table 8−1. L4 Interconnect Registers

Register Name	Type	Register Width (Bits)	Physical Address
COMPONENT	R	32	0x4800 1000
AGENT_CONTROL	RW	32	0x4800 1020
AGENT_STATUS	R	32	0x4800 1028

8.3.3 SCM Register Summary

Table 8−2 shows all SCM registers and their physical addresses. For a detailed description, see Section 8.6, *SCM Registers*. See Chapter 24, *Pinout Overview*, for details about SCM register bits CONTROL_PADCONF_x.

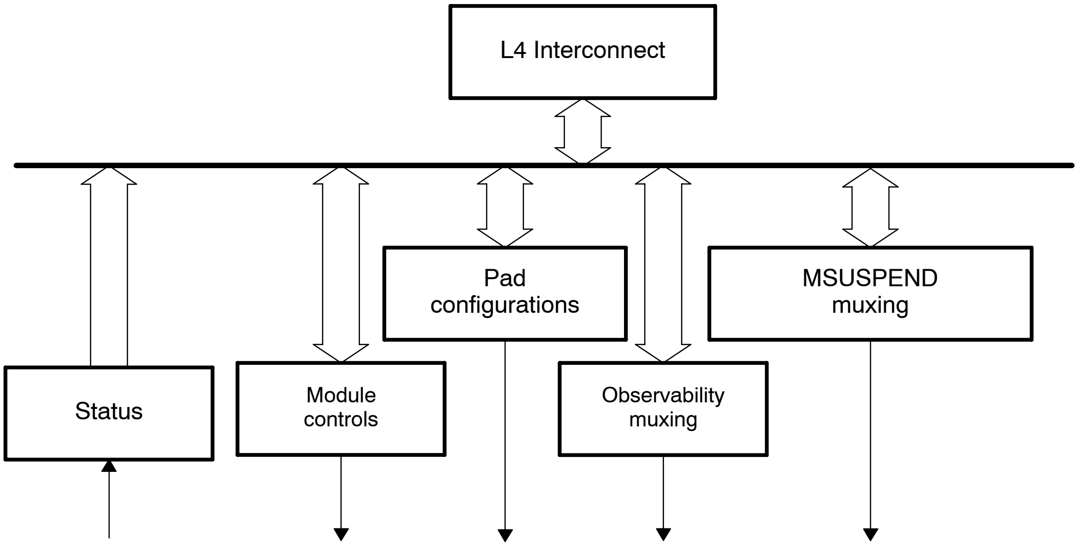
Table 8−2. SCM Register Summary

Register Name	Type	Register Width (Bits)	Physical Address
CONTROL_REVISION	R	32	0x4800 0000
CONTROL_SYSCONFIG	RW	32	0x4800 0010
CONTROL_DEVCONF	RW	32	0x4800 0274
CONTROL_STATUS	R	32	0x4800 02F8

8.4 SCM Functional Description

The SCM controls various OMAP2420 modules, settings through register selection, and internal signals routed to dedicated pins for hardware debugging. Figure 8−5 is functional block diagram of the SCM.

Figure 8−5. SCM Functional Block Diagram



This section details the functionality of suspend, security/emulation, status, and control registers. For a description of pad configuration and multiplexing details, see Chapter 24, *Pinout Overview*.

8.5 SCM Programming Model

8.5.1 Module Initialization

In initialization mode, only modules used at boot time are associated with the pads. Other module inputs are internally tied and output pads are turned off when the feature is available.

For pad configuration, pullup/down fields are set according to the OMAP2420 pin list; for more information about pad configuration, see Chapter 24, *Pinout Overview*.

Device type (emulator, secure, general-purpose, etc.) affects reset values and access rights for some emulation- and security-related configuration bits (see Section 8.6, *SCM Registers*).

On general-purpose devices (no secure mode available), some features are not available. These inaccessible registers define default/fixed device configuration/behavior.

8.5.2 Feature Settings

8.5.2.1 Set the MMC/SDIO Module Input Clock

MMC/SDIO module input clock selection

SCM register bit CONTROL_DEVCONF[24]

0 = Input clock is from the external pin.

1 = Internal loopback: module input clock is copied from the module output clock.

8.5.2.2 Set USB Port 0, 1, 2 Top-Level Transceiver Interface Mode

The core internal registers also control the final USB mode. For correct operation, top-level and internal core settings must be coherent.

- ☐ Port 0: SCM register bit CONTROL_DEVCONF[23:22]

  00 = Unidir

  01 = Unidir TLL

  10 = Bidir

  11 = Bidir TLL

- ☐ Port 1: SCM register bit CONTROL_DEVCONF[21:20]

  00 = Unidir

  01 = Unidir TLL

  10 = Bidir

  11 = Bidir TLL

- Port 2: SCM register bit CONTROL_DEVCONF[19:18]

  00 = Unidir

  01 = Unidir TLL

  10 = Bidir

  11 = Bidir TLL

8.5.2.3 Set 5-Pins TLL Mode Enable for USB Port 2

Used with the SCM register bit CONTROL_DEVCONF[19:18] bit field setting in Section 8.5.2.2, *Set USB Port 0, 1, 2 Top-Level Transceiver Interface Mode*

- SCM register bit CONTROL_DEVCONF[17]

  0 = 5-pin TLL mode disabled

  1 = 5-pin TLL mode enabled

8.5.2.4 Set the McBSP2 Internal Clock

Gates the internal interconnect clock and selects the CLKR input for the module. There are no external pins McBSP2_CLKR and McBSP2_FSR for the module McBSP2. For this module, clock input is from the McBSP2_CLKX pin, and FSR input is from the McBSP2_FSX pin.

- SCM register bit CONTROL_DEVCONF[7]

  0 = Enable internal clock

  1 = Inactive internal clock

- SCM register bit CONTROL_DEVCONF[6]

  0 = Clock is from the internal functional clock

  1 = Clock is from the external pin McBSP2_CLKS

8.5.2.5 Set the McBSP1 Internal Clock

Gates the internal interconnect clock and selects FSR, CLKR, and CLKS as clock input for the module.

- SCM register bit CONTROL_DEVCONF[5]

  0 = Enable internal clock

  1 = Inactive internal clock

- SCM register bit CONTROL_DEVCONF[4]

  0 = FSR is from the pin McBSP1_FSR.

  1 = FSR is from the pin McBSP1_FSX.

- SCM register bit CONTROL_DEVCONF[3]

  0 = CLKR is from the pin McBSP1_CLKR.

  1 = CLKR is from the pin McBSP1_CLKX.

- SCM register bit CONTROL_DEVCONF[2]

  0 = CLKS is from the internal functional clock.

  1 = CLKS is from the pin McBSP_CLKS.

8.6 SCM Registers

Table 8−3 summarizes the SCM instance.

Table 8−3. Instance Summary

Module Name	Base Address	Size
SYSC1	0x4800 0000	1K byte

8.6.1 SCM Register Mapping Summary

Supported accesses: 32-bit, 16-bit, and 8-bit (little−endian) are supported; multibyte accesses are valid only on word boundaries (32-bit alignment for 32-bit access, 16-bit alignment for 16-bit access).

Table 8−4 lists the SYSC1 registers. Table 8−5 through Table 8−8 describe the register bits.

Table 8−4. SYSC1 Registers

Register Name	Type	Register Width (Bits)	Offset
CONTROL_REVISION	R	32	0x4800 0000
CONTROL_SYSCONFIG	RW	32	0x4800 0010
CONTROL_DEVCONF	RW	32	0x4800 0274
CONTROL_STATUS	R	32	0x4800 02F8

8.6.2 Register Descriptions

Table 8–5. CONTROL_REVISION

Address Offset	0x000		
Physical Address	0x4800 0000	**Instance**	SYSC1
Description	Control module revision 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																								REVISION							

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Read returns 0.	R	0x000000
7:0	REVISION	RTL revision number [7:4] Major revision [3:0] Minor revision	R	

Table 8–6. CONTROL_SYSCONFIG

Address Offset	0x010		
Physical Address	0x4800 0010	**Instance**	SYSC1
Description	Set parameters relative to the idle mode of the SCM.		
Type	RW		

31	30	29	28	27	26	25	24	23	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	Reserved	R	0x0000000
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. Idle request is acknowledged based on the internal activity of the module. 0x3: Reserved—Do not use.	R	0x0

Table 8−6. CONTROL_SYSCONFIG (Continued)

Bits	Field Name	Description	Type	Reset
2	ENAWAKEUP	Wake-up enable. Not used in the module.	R	0
1	SOFTRESET	Software reset. Not used in the module. 0x0: Normal mode 0x1: The module is reset.	R	0
0	AUTOIDLE	Internal L4 interconnect clock-gating strategy 0x0: L4 interconnect clock is free-running 0x1: Automatic L4 interconnect clock-gating strategy is applied, based on L4 interconnect interface activity.	RW	0

Table 8−7. CONTROL_DEVCONF

Address Offset	0x274		
Physical Address	0x4800 0274	**Instance**	SYSC1
Description	Static device configuration register; module-dedicated functions.		
Type	RW		

31	30	29	28	26	27	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SENSDMAREQ5	SENSDMAREQ4	SENSDMAREQ3	SENSDMAREQ2	Reserved	Reserved	Reserved	MMCSDIOADPCLKISEL	USBT0WRMODEI		USBT1WRMODEI		USBT2WRMODEI		USBT2TLL5PI	USB0PUENACTLOI	USBSTANDBYCTRL	Reserved	Reserved			FACINPUTSEL			MCBSP2_AUXON	MCBSP2_CLKS	MCBSP1_AUXON	MCBSP1_FSR	MCBSP1_CLKR	MCBSP1_CLKS	SENSDMAREQ1	SENSDMAREQ0

Table 8−7. CONTROL_DEVCONF (Continued)

Bits	Field Name	Description	Type	Reset
31	SENSDMAREQ5	Set sensitivity on SYS.nDMAREQ5 input pin. 0x0: Transition sensitivity 0x1: Edge sensitivity	RW	0
30	SENSDMAREQ4	Set sensitivity on SYS.nDMAREQ4 input pin. 0x0: Transition sensitivity 0x1: Edge sensitivity	RW	0
29	SENSDMAREQ3	Set sensitivity on SYS.nDMAREQ3 input pin. 0x0: Transition sensitivity 0x1: Edge sensitivity	RW	0
28	SENSDMAREQ2	Set sensitivity on SYS.nDMAREQ2 input pin. 0x0: Transition sensitivity 0x1: Edge sensitivity	RW	0
27	Reserved	Reserved for future use	RW	1
26	Reserved	Reserved for future use		
25	Reserved	Reserved for future use		
24	MMCSDIOADP CLKISEL	MMC/SDIO module input clock selection 0x0: Input clock is from the external pin. 0x1: Internal loopback; module input clock copied from the module output clock	RW	0
23:22	USBT0WRMODEI	USB port 0 top-level transceiver interface mode control. The core internal registers also control the final USB mode. For correct operation, top-level and internal core settings must be coherent . 0x0: Unidi 0x1: Unidi TLL 0x2: Bidi 0x3: Bidi TLL	RW	0x0
21:20	USBT1WRMODEI	USB port 1 top-level transceiver interface mode control. The core internal registers also control the final USB mode. For correct operation, top-level and internal core settings must be coherent.	RW	0x0

Table 8−7. CONTROL_DEVCONF (Continued)

Bits	Field Name	Description	Type	Reset
		0x0: Unidi 0x1: Unidi TLL 0x2: Bidi 0x3: Bidi TLL		
19:18	USBT2WRMODEI	USB port 2 top-level transceiver interface mode control. The core internal registers also control the final USB mode. For correct operation, top-level and internal core settings must be coherent. 0x0: Unidi 0x1: Unidi TLL 0x2: Bidi 0x3: Bidi TLL	RW	0x0
17	USBT2TLL5PI	5_pin TLL mode enable for USB port 2 (used with USBT2WRMODEI bit field setting) 0x0: 5_pin TLL mode disabled 0x1: 5_pin TLL mode enabled	RW	0
16	USB0PUENACT-LOI	USB0 pullup enable polarity control 0x0: Pullup enable active high 0x1: Pullup enable active low	RW	0
15	USBSTANDBY CTRL	Software control of the USB standby signal 0x0: USB module standby signal not asserted 0x1: USB module standby signal asserted	RW	0
14	Reserved			
13:11	Reserved	Reserved bits	R	0x0
10:8	FACINPUTSEL	Frame adjustment counter FAC_SYNC input selection 0x0: McBSP1_FSX 0x1: McBSP2_FSX 0x2: EAC_AC_FC 0x3: EAC_MD_FS 0x4: EAC_BT_FS	RW	0x0

Table 8−7. CONTROL_DEVCONF (Continued)

Bits	Field Name	Description	Type	Reset
7	MCBSP2_ AUXON	Gate the internal OCP clock for the McBSP2 module. 0x0: Enable internal clock. 0x1: Inactive internal clock.	RW	0
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 internal functional clock. 0x1: CLKS is from the external pin McBSP_CLKS.	RW	0
5	MCBSP1_ AUXON	Gate the internal OCP clock for the McBSP1 module. 0x0: Enable internal clock 0x1: Inactive internal clock	RW	0
4	MCBSP1_FSR	Select the FSR input for the module McBSP1. 0x0: FSR is from the McBSP1_FSR pin. 0x1: FSR is from the McBSP1_FSX pin.	RW	0
3	MCBSP1_CLKR	Select the CLKR input for the module McBSP1. 0x0: CLKR is from the McBSP1_CLKR pin. 0x1: CLKR is from the McBSP1_CLKX pin.	RW	0
2	MCBSP1_CLKS	Select the CLKS input for the module McBSP1. 0x0: CLKS is from the internal functional clock. 0x1: CLKS is from the McBSP_CLKS external pin.	RW	0
1	SENSDMAREQ1	Set sensitivity on SYS.nDMAREQ1 input pin. 0x0: Transition sensitivity 0x1: Edge sensitivity	RW	0
0	SENSDMAREQ0	Set sensitivity on SYS.nDMAREQ0 input pin. 0x0: Transition sensitivity 0x1: Edge sensitivity	RW	0

Table 8–8. CONTROL_STATUS

Address Offset	0x2F8		
Physical Address	0x4800 02F8	**Instance**	SYSC1
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																					Reserved			Reserved		SYSBOOT_5	SYSBOOT_4	SYSBOOT_3	SYSBOOT_2	SYSBOOT_1	SYSBOOT_0

Bits	Field Name	Description	Type	Reset
31:11	Reserved	Reserved	R	0x000000
10:8	Reserved	Device type captured at reset time 0b000 ⇒ Reserved 0b001 ⇒ Reserved 0b011 ⇒ Reserved	R	0x0
7:6	Reserved	Reserved	R	0x00
5	SYSBOOT_5	Sys.Boot pin values sampled at power-on reset	R	0
4	SYSBOOT_4	Sys.Boot pin values sampled at power-on reset	R	0
3	SYSBOOT_3	Sys.Boot pin values sampled at power-on reset	R	0
2	SYSBOOT_2	Sys.Boot pin values sampled at power-on reset	R	0
1	SYSBOOT_1	Sys.Boot pin values sampled at power-on reset	R	0
0	SYSBOOT_0	Sys.Boot pin values sampled at power-on reset	R	0

Chapter 9

Memory Management Units

This chapter describes the memory−management units (MMUs) in the OMAP2420 multimedia device.

Topic **Page**

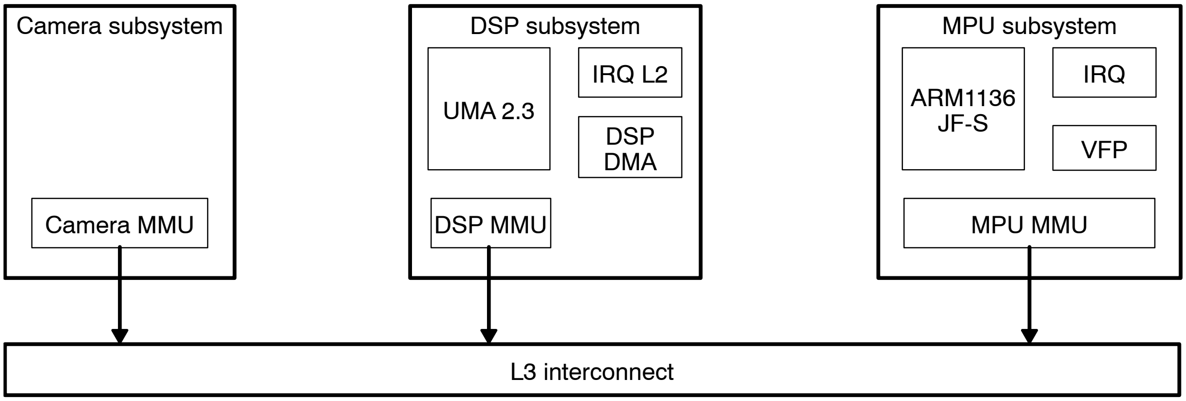
9.1 MMU Overview

The OMAP2420 contains three memory management units (MMUs):

- MPU MMU
- DSP MMU
- Camera MMU

Figure 9−1 shows the MMU instances in the OMAP2420.

Figure 9−1. OMAP2420 MMU Instances



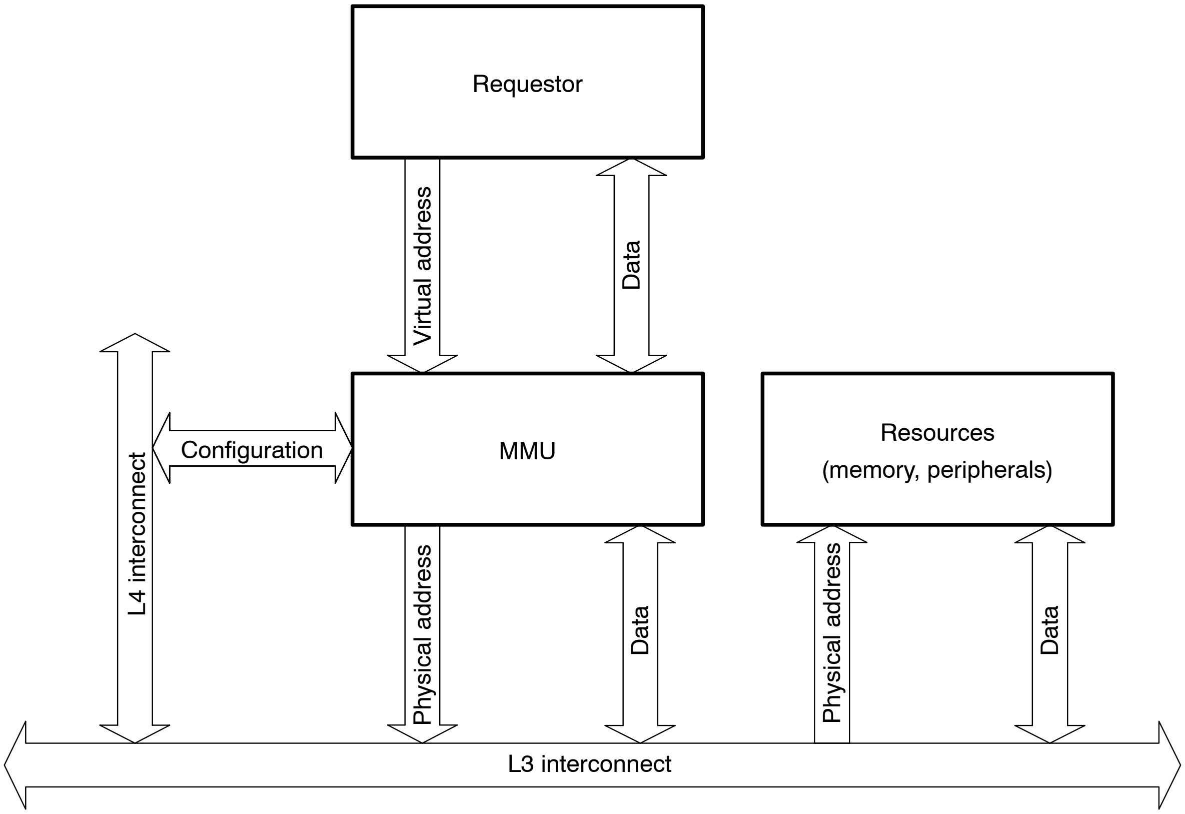
The DSP MMU and the camera MMU share the same architecture and are covered in this chapter. The MPU MMU implements a different architecture and is covered in Chapter 3, *MPU Subsystem*.

9.2 MMU Integration

The MMU communicates accesses from the requestor (through its data access open−core protocol [OCP] target) to the L3 main interconnect (through its OCP initiator), performing virtual−to−physical address translation. The MMU is programmed through its configuration OCP slave connected to the L4 OCP interconnect. MMU error conditions are signaled as interrupts to the system microprocessor unit (MPU).

Figure 9−2 shows the typical system integration of an MMU instance.

Figure 9−2. MMU System Integration



9.2.1 Clock Domains

Each MMU instance has two clock domains: the functional clock domain for the MMU, which is synchronous with the clock for the OCP slave and master access ports, and the clock domain for the OCP slave configuration port used to configure the MMU instance.

Both MMU clocks are derived from a common reference clock and synchronous enable signals provided by the power, reset, and clock management module (PRCM). Each MMU instance can be put in idle mode by the PRCM by signaling an idle request. For details about the PRCM, see Chapter 5, *Power, Reset, and Clock Management.*

9.2.2 Power Management

The OMAP2420 functional units are grouped in power domains. A power domain is a section of the device with independent and dedicated power rails. There are five power domains: See Chapter 5 for information about OMAP2420 power management.

- MPU
- CORE
- DSP
- Graphics
- WKUP

For information about OMAP2420 power management, see Chapter 5, *Power, Reset, and Clock Management.*

The MMU instances belong to different power domains. Table 9−1 shows the correspondence between the DSP and camera MMU instances and the power domains.

Table 9−1. Power Domains of the MMU Instances

MMU Instance	**Power Domain**
DSP MMU	DSP
Camera MMU	CORE

9.2.3 Reset

The MMU instances are reset with their respective reset domains. Table 9−2 shows the correspondence between the MMU instances and the reset domains.

Table 9−2. Reset Domains of the MMU Instances

MMU Instance	**Reset Domain**
DSP MMU	DSP_RST2
Camera MMU	CORE_RST

When an MMU instance is released from reset, its translation lookaside buffer (TLB) is empty and the MMU is disabled.

9.2.4 Interrupts

Each MMU instance can generate an interrupt to the MPU on the occurrence of a predefined set of events:

- Multi-hit fault

  There is more than one TLB entry for the given virtual address.

- Table walk fault

  A table walk data read generated an error.

- ❑ Emulation miss

  A TLB miss was caused by an emulation access.

- ❑ Translation fault

  No translation is found for the given virtual address. The hardware table walker is enabled, but no valid page table entry exists for the requested address.

- ❑ TLB miss with table walk disabled

  No translation is found in the TLB for the given virtual address, and the table−walking logic is disabled.

Each of these events can be individually enabled and disabled using the MMU_IRQENABLE register. If such an event occurs and it is enabled, an interrupt is generated to the MPU. The MPU can use the MMU_INTERRUPTSTATUS register to find the cause of the interrupt.

The MMU_FAULT_AD register holds the virtual address of the translation that caused the interrupt. A separate register (MMU_EMU_FAULT_AD) indicates the address of the last emulation event that caused an MMU interrupt.

Table 9−3 shows the generated interrupt versus the MMU instance.

Table 9−3. MPU Interrupt Mapping of the MMU Instances

MMU Instance	**MPU Interrupt**
DSP MMU	DSP MMU IRQ (IRQ 28)
Camera MMU	Camera IRQ (IRQ 24)

Note:

There is no dedicated camera MMU interrupt in the MPU subsystem.

Details about interrupt generation in the camera subsystem are outlined in Chapter 14, *Camera Subsystem*. For details about the MPU interrupt scheme, see Chapter 3, *MPU Subsystem*.

9.2.5 MMU Register Summary

Each MMU instance is configured by the MPU through a set of 18 configuration registers, which are identified in Table 9−4. For a complete reference of these registers and their content, see Section 9.5, *MMU Registers*.

Table 9−4. MMU Registers Summary

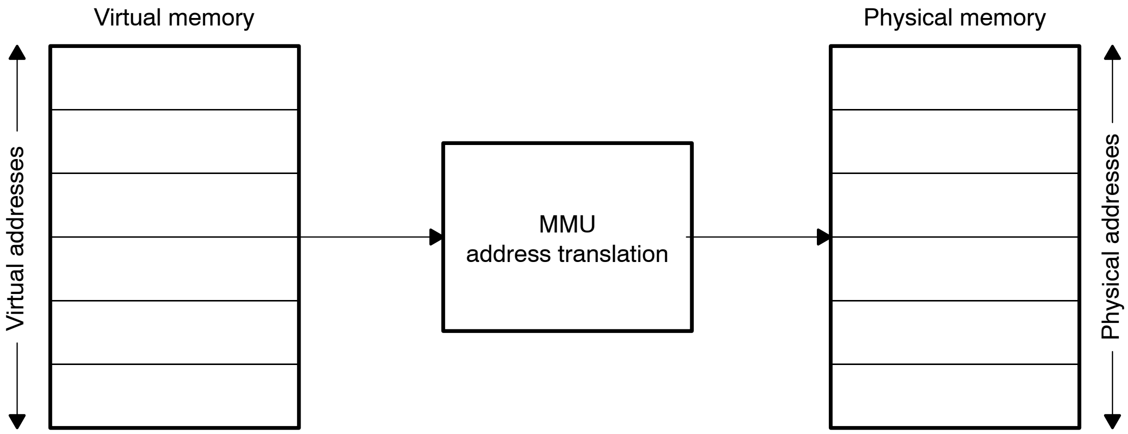
Register Name	**Access Type**
MMU_REVISION	R
MMU_SYSCONFIG	RW
MMU_SYSSTATUS	R
MMU_IRQSTATUS	RW
MMU_IRQENABLE	RW
MMU_WALKING_ST	R
MMU_CNTL	RW
MMU_FAULT_AD	R
MMU_TTB	RW
MMU_LOCK	RW
MMU_LD_TLB	RW
MMU_CAM	RW
MMU_RAM	RW
MMU_GFLUSH	RW
MMU_FLUSH_ENTRY	RW
MMU_READ_CAM	R
MMU_READ_RAM	R
MMU_EMU_FAULT_AD	R

9.3 MMU Functional Description

MMUs handle the translation between physical addresses and virtual addresses. Virtual addresses are issued by the requestor (DSP or camera) to the respective MMU (DSP MMU or camera MMU). The MMU translates these virtual addresses into physical addresses that are used to access the actual resource (memory).

Figure 9−3 shows the relationship between physical address, virtual address, and the MMU.

Figure 9−3. MMU Address Translation



9.3.1 MMU Benefits

The use of an MMU offers two major benefits:

- ❑ Memory defragmentation—Fragmented physical memory can be translated into contiguous virtual memory without moving any data.
- ❑ Memory protection—Illegal, that is, nonallowed accesses to memory locations can be detected and prevented.

Figure 9−4 shows two typical MMU use cases.

Figure 9−4. MMU Usage Examples

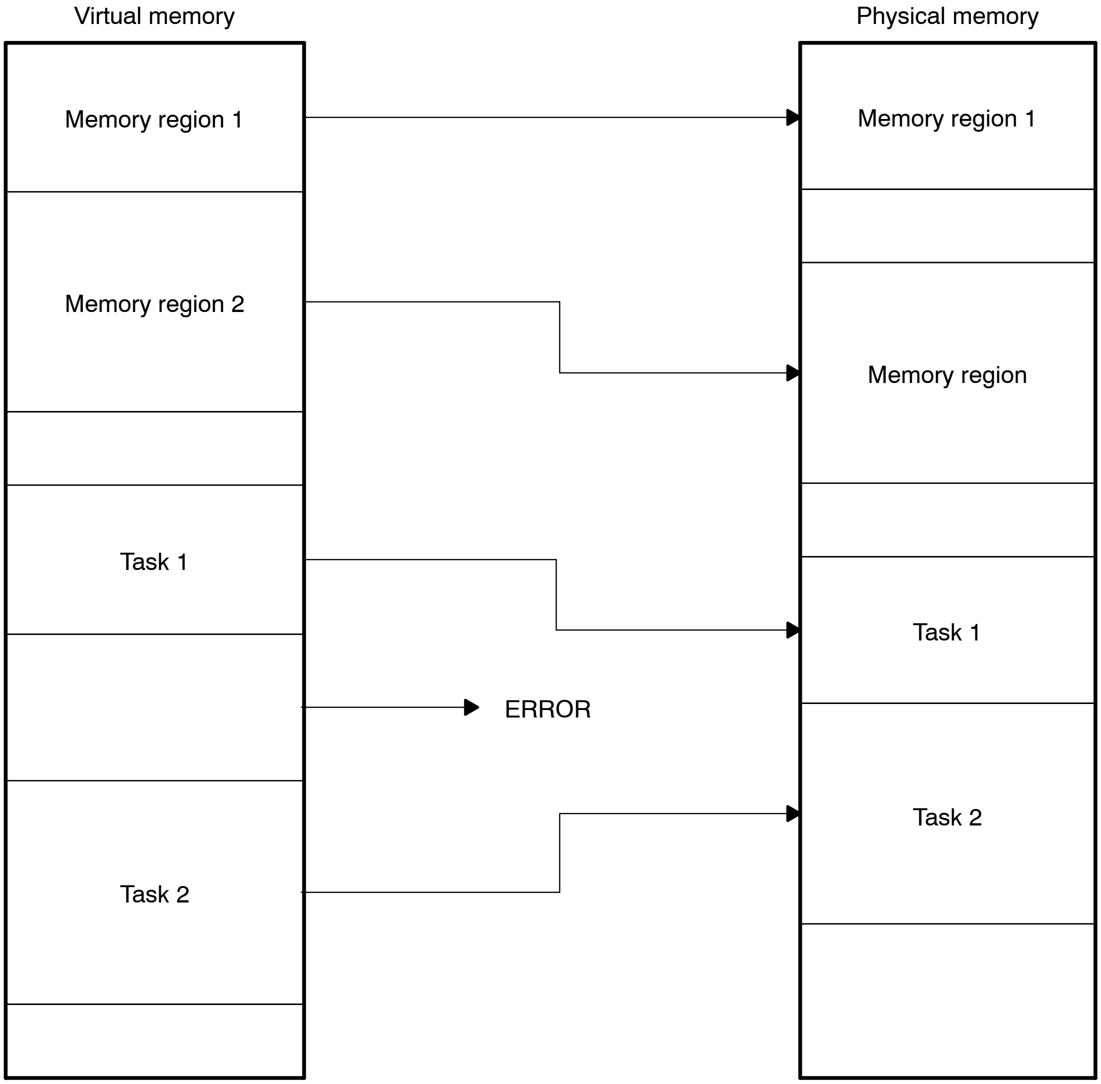


Figure 9−4 shows two benefits of using an MMU. There, memory region 1 and memory region 2 are fragmented in physical memory. Using the MMU, they appear as one contiguous memory region in the virtual memory space.

On the other hand, task 1 and task 2 are adjacent to one another in physical memory. In systems without an MMU, there is a danger that task 1 can accidentally write to the memory area allocated to task 2, and vice versa. Because there is a region of unmapped memory between the two tasks, allocating them into two separate virtual memory regions prevents this problem. Any erroneous access to this region results in an error that is easily detected.

9.3.2 MMU Architecture

The MMU translation process is based on translation entries. These entries are stored in translation tables. One first-level translation table and—optionally—several second-level translation tables can exist.

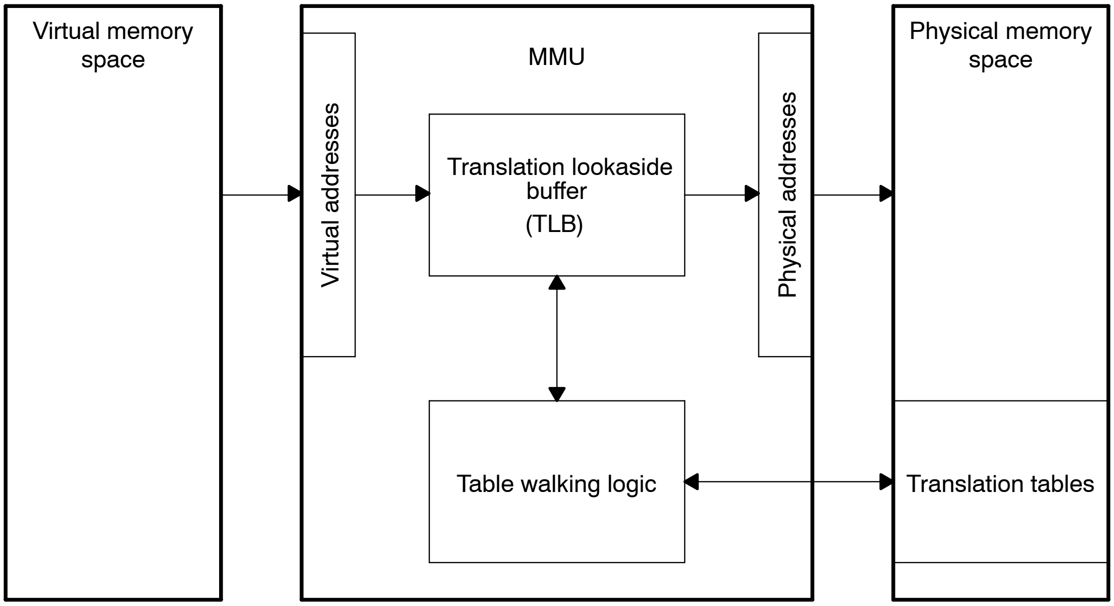
Each table entry describes the translation of one contiguous memory region. For a description of the structure of these tables, see Section 9.3.3, *Translation Tables*.

Two major functional units exist in the MMU to automatically provide address translation based on the table entries:

- The table walker automatically retrieves the correct translation table entry for a requested translation. If 2-level translation is used (for the translation of small memory pages), the table walker also automatically reads the required second-level translation table entry.
- The TLB stores recently used translation entries. The TLB acts like a cache of the translation table.

This basic MMU architecture is outlined in Figure 9−5.

Figure 9−5. MMU Architecture

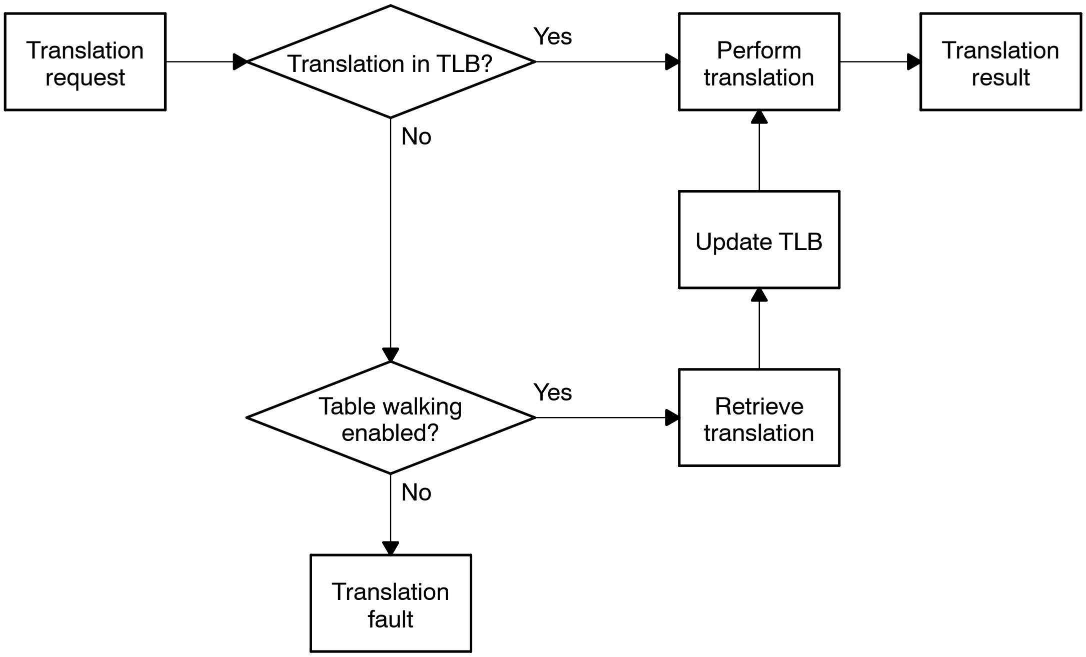


9.3.2.1 MMU Address Translation Process

When an address translation is requested (that is, for every memory access with the MMU enabled), the MMU first checks whether the translation is already contained in the TLB. The TLB acts like a cache, storing recent translations. It can also be programmed manually to ensure that time-critical data is translated without delay.

If the requested translation is not in the TLB, the table−walking logic retrieves this translation from the translation table(s). It then updates the TLB and the address translation is performed. This process is summarized in Figure 9−6.

Figure 9−6. Translation Process



9.3.3 Translation Tables

The translation of virtual to physical addresses is based on entries in translation tables. These entries define the following properties:

- ❏ Address translation (the correspondence between virtual and physical addresses)
- ❏ Size of the memory region this entry translates
- ❏ Endianness, data access size, and the mixed property of this memory region. The mixed property defines whether the data size information needed for the endianness conversion is derived from the detected CPU access size or from the specified data access size parameter.

The translation tables are indexed by the virtual address. Each virtual address corresponds to one entry in the translation table.

9.3.3.1 Translation Table Hierarchy

When developing a table-based address translation scheme, one of the most important design parameters is the memory page size each translation table entry describes. Using a larger page sizes means a smaller translation table.

Using a smaller page size greatly increases the efficiency of dynamic memory allocation and defragmentation. That is why many operating systems (OSs) can operate on memory blocks as small as 4KB. Smaller page size, however, implies a more complex table structure.

A quick calculation shows that using 4−KB memory pages with one translation table requires one million entries to span the entire 4−GB address range. Such a table itself would be 32MB; obviously, this is not feasible.

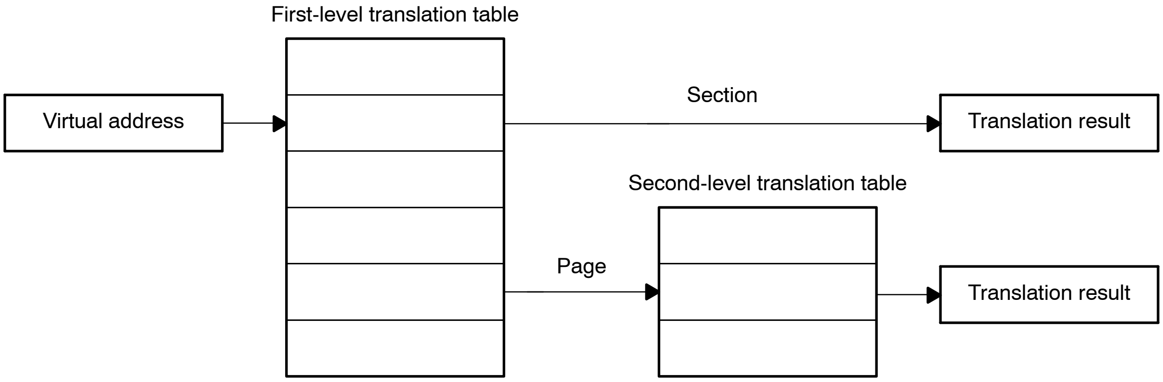
However, using larger pages greatly reduces the functionality of the OS memory management. To reconcile these two requirements, a 2-level hierarchy is implemented. In this hierarchy, one first-level translation table describes the translation properties based on 1−MB memory regions.

Each entry in this first-level translation table can specify the following:

- ❑ The translation properties for a large memory section. Such memory sections can be either 1MB (section) or 16MB (supersection). In this case, all translation parameters are specified in the first-level translation table entry.
- ❑ A pointer to a second-level translation table that specifies individual translation properties based on smaller pages within the 1−MB page of memory. These pages can be either 64KB (large page) or 4KB (small page). In this case, the actual translation parameters are specified in the second-level translation table entry. The first-level translation table entry specifies only the base address of the second-level translation table.

This hierarchical approach means that additional translation information for smaller pages need be provided only when such pages are actually used. Figure 9−7 shows this hierarchy.

Figure 9−7. Translation Hierarchy



The structure of the first and second-level translation tables and their entries are described in more detail in Section 9.3.3.2 and Section 9.3.3.3, respectively.

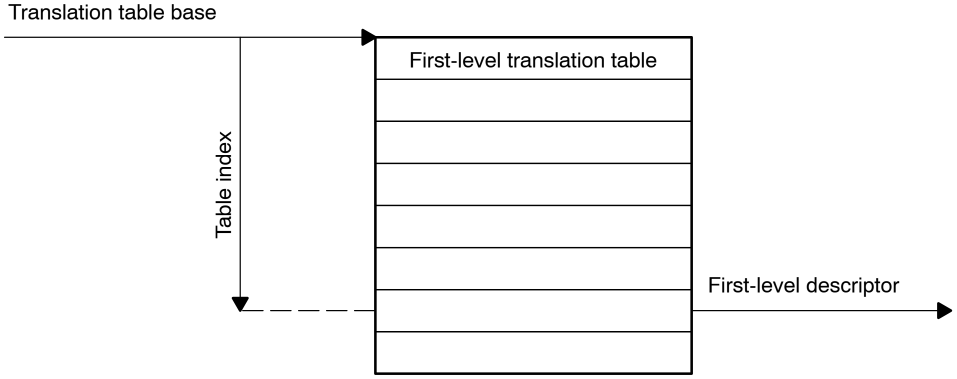
9.3.3.2 First-Level Translation Table

The first-level translation table describes the translation properties for 1−MB sections. 4096 32-bit entries—so-called first-level descriptors—are required to describe a 4−GB address range. 16 first-level descriptors are required for the DSP with its 16−MB virtual address space.

The first-level translation table start address must be aligned on a multiple of the table size with a 128-byte minimum. Consequently, an alignment of at least 16K bytes is required for a complete 4096-entry table (at least the last 14 address bits must be 0).

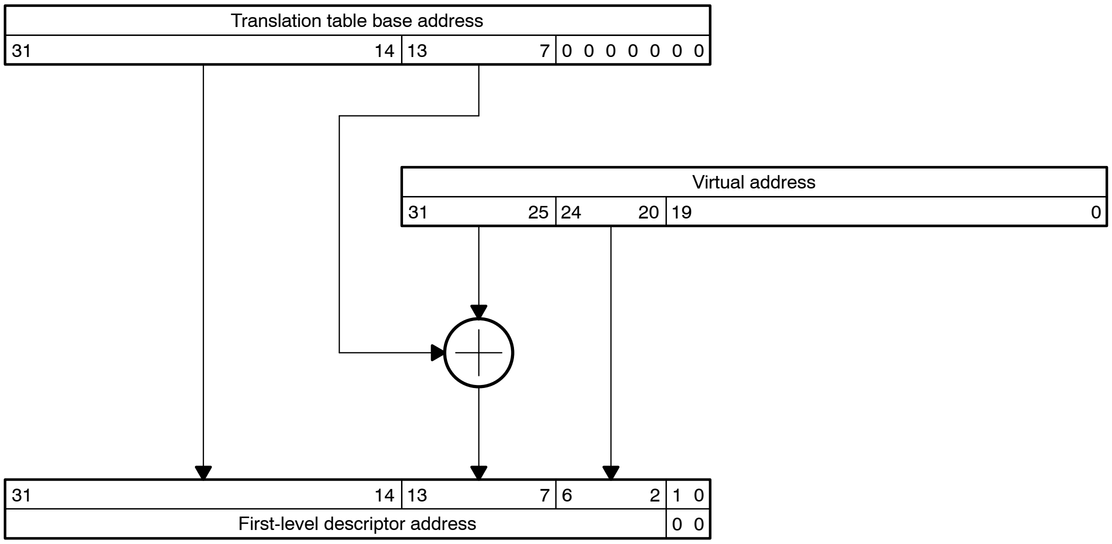
The start address of the first-level translation table is specified by the so-called translation table base. The table is indexed by the upper 12 bits of the virtual address. This mechanism is shown in Figure 9−8.

Figure 9−8. First−Level Descriptor Address Calculation



In summary, translation table base and translation table index together define the first-level descriptor address. Figure 9−9 outlines the mechanism used to calculate this address.

Figure 9−9. Detailed First−Level Descriptor Address Calculation



As an example to illustrate this mechanism, consider a translation table base address of 0x8000:0000 and a virtual address of 0x1234:5678. In this case, the first-level descriptor address is 0x80000:0000 + (0x123 <<2) = 0x8000:048C.

First−Level Descriptor Format

Each first-level descriptor provides either the complete address translation for a 1−MB section or provides a pointer to a second-level translation table. The first-level descriptor format is shown in Table 9−5.

Table 9−5. First-Level Descriptor Format

First-Level Descriptor Format												
31:24	23:20	19	18	17	16	15	14:12	11:10	9:2	1	0	
										0	0	Fault
Second-Level Translation Table Base Address									X	0	1	Page
Section Base Address		X	0	M	X	E	X	ES	X	1	0	Section
Supersection Base Address	X		1	M	X	E	X	ES	X	1	0	Supersection
										1	1	Fault

Notes: 1) M = Mixed region 0 = Page-based endianness, 1 = Access-based endianness
2) E = Endianess 0 = Little−endian, 1 = Big−endian
3) ES = Element size 00 = 8-bit, 01= 16-bit, 10 = 32-bit, 11 = No endianness conversion
4) X = Don't care

First-Level Page Descriptor Format

If a translation granularity smaller than 1MB is required, a 2-level translation process is used. In this case, the first-level block descriptor specifies only the start address of a second-level translation table. The second-level translation table entries specify the actual translation properties.

First−Level Section Descriptor Format

Each section descriptor in the first-level translation table specifies the complete translation properties for a 1−MB section or a 16−MB supersection. Because each descriptor in the first-level translation table describes 1MB of memory, supersection descriptors must be repeated 16 times.

In addition to the address translation itself, three parameters are specified in such section descriptors:

- Endianness
- Element size
- Mixed region

The endianness parameter specifies whether the memory section uses big− or little−endian data format.

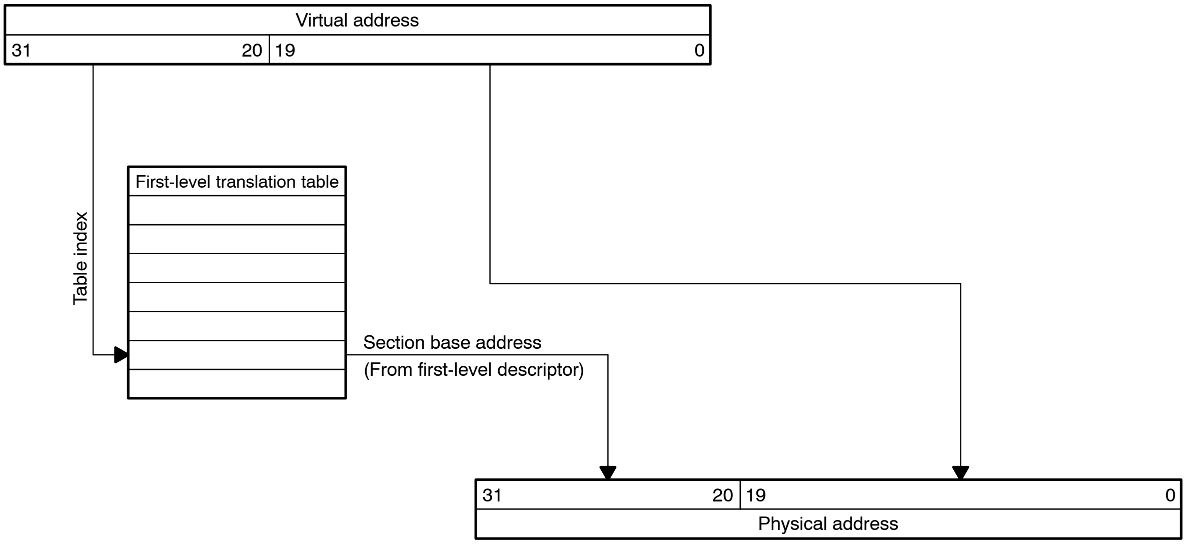
The element size parameter can specify the data access size for all data items in the defined section. This size can be 8, 16, or 32 bits. Correct knowledge of the data access size is required to perform endianness conversion. You can also specify that no endianness conversion be performed.

The mixed region parameter specifies whether the information about the data access size is detected from the access itself (access-based detection) or whether the specified element size parameter is used (page-based detection). The latter mechanism can, for instance, be employed when several smaller accesses are packed into a larger access (for example, two 16-bit accesses in one 32-bit access). Without specified data access size in such a case, the access size is detected as 32-bit and endianness conversion is performed accordingly. This leads to an incorrect result. To avoid this problem, explicitly specify the data access size for this memory section.

Section Translation Summary

Sections and supersections can be translated based solely on the information in the first-level translation table. Figure 9−10 summarizes the address translation process for a section.

Figure 9−10. Section Translation Summary

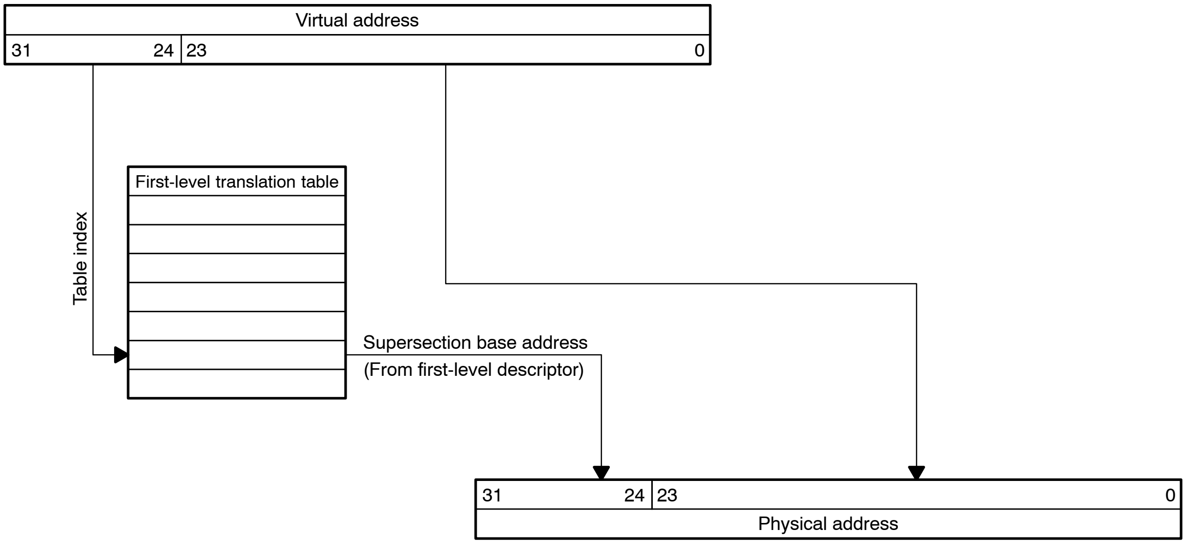


Supersection Translation Summary

The translation of a supersection is similar to that of a section. The difference is that for a supersection, only bits 31 through 24 index into the first-level translation table. The last 4 bits of the table index are implicitly assumed to be 0, as there are 16 identical consecutive entries for a supersection.

Figure 9−11 summarizes the translation process for a supersection.

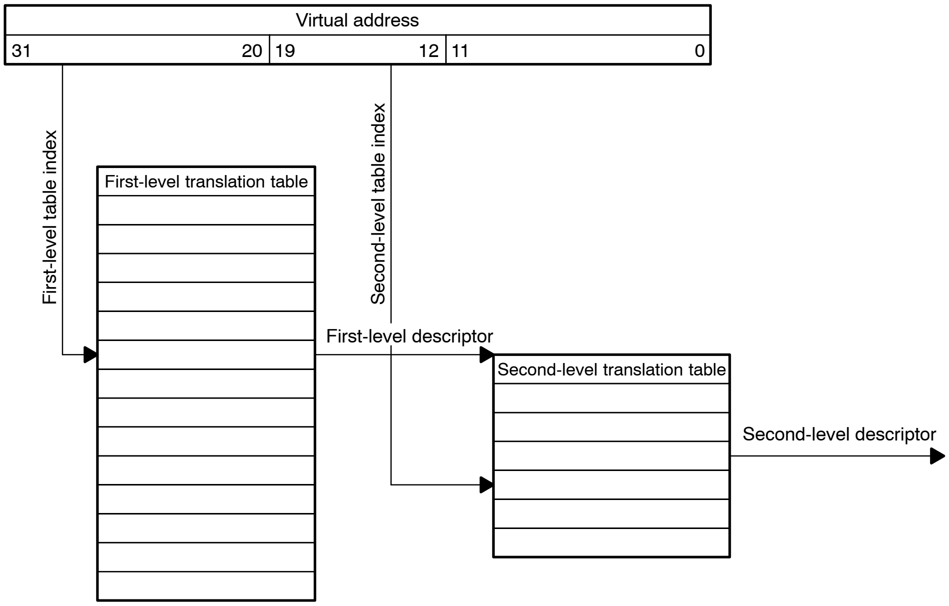
Figure 9−11. Supersection Translation Summary


Virtual address
31
24
23
0
Table index
First-level translation table
Supersection base address
(From first-level descriptor)
31
24
23
0
Physical address

9.3.3.3 Two-Level Translation

Two-level translation is used when a translation granularity better than 1MB is required. In that case, the first-level descriptor provides a pointer to the base address of a second-level translation table. This second-level table is indexed by bits 19 through 12 of the virtual address. Figure 9−12 shows this indexing mechanism.

Figure 9−12. Two-Level Translation



Each second-level translation table describes the translation of 1MB of address space in pages of 64KB (large page) or 4KB (small page). It consists of 256 second-level descriptors describing 4KB each. In the case of a large page, the same descriptor must be repeated 16 times.

Second-Level Descriptor Format

Second-level descriptors provide all necessary information for the translation of a large or small page. In this, second−level descriptors are similar to first-level section descriptors. Table 9−6 shows the format of such second-level descriptors. The translation parameters—endianness, element size, and mixed region—have the same meaning as for sections. See Section 9.3.3.2, *First−Level Translation Table*.

Table 9−6. Second-Level Descriptor Format

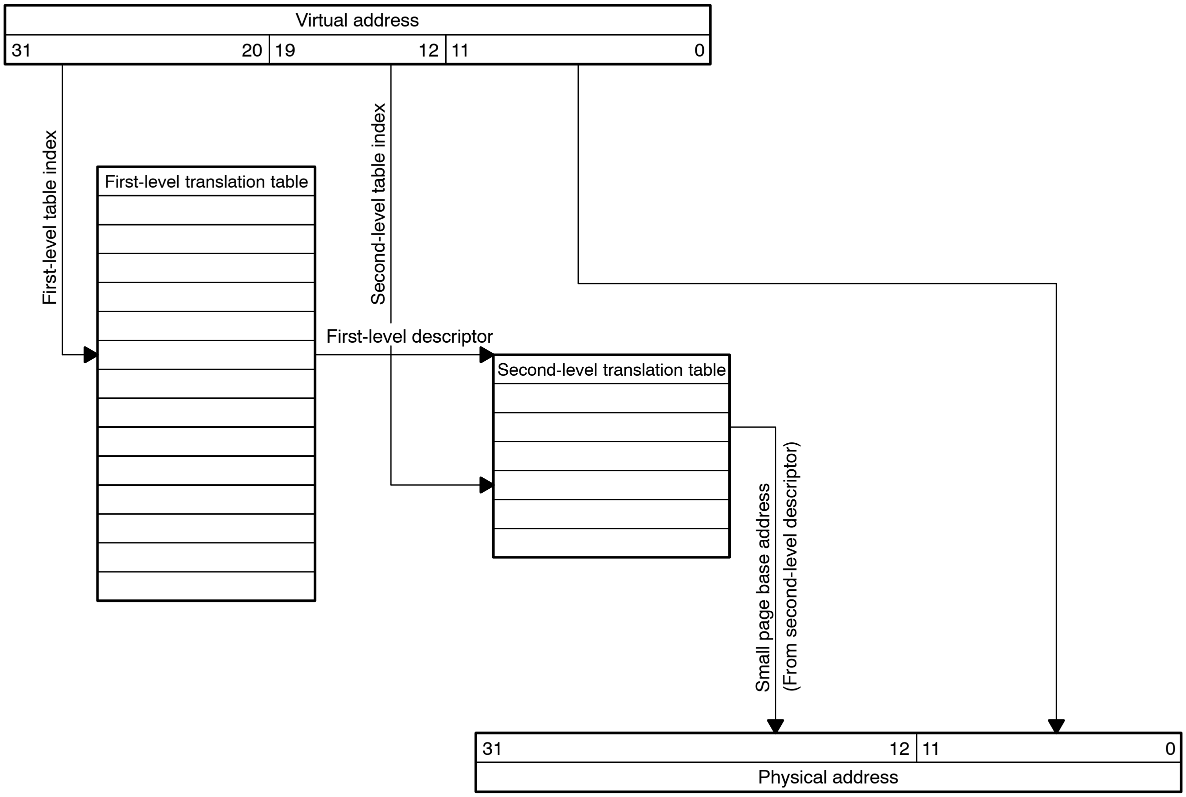
Second-Level Descriptor Format										
31:16	15:12	11	10	9	8:6	5:4	3:2	1	0	
								0	0	Fault
Large Page Base Address	X	M	X	E	X	ES	X	0	1	Large Page
Small Page Base Address		M	X	E	X	ES	X	1	X	Small Page

Notes: 1) M = Mixed region 0 = Page-based endianness, 1 = Access-based endianness
2) E = Endianess 0 = Little−endian, 1 = Big−endian
3) ES = Element size 00 = 8-bit, 01= 16-bit, 10 = 32-bit, 11 = No endianness conversion
4) X = Don't care

Small−Page Translation Summary

Figure 9−13 summarizes the translation process for small pages.

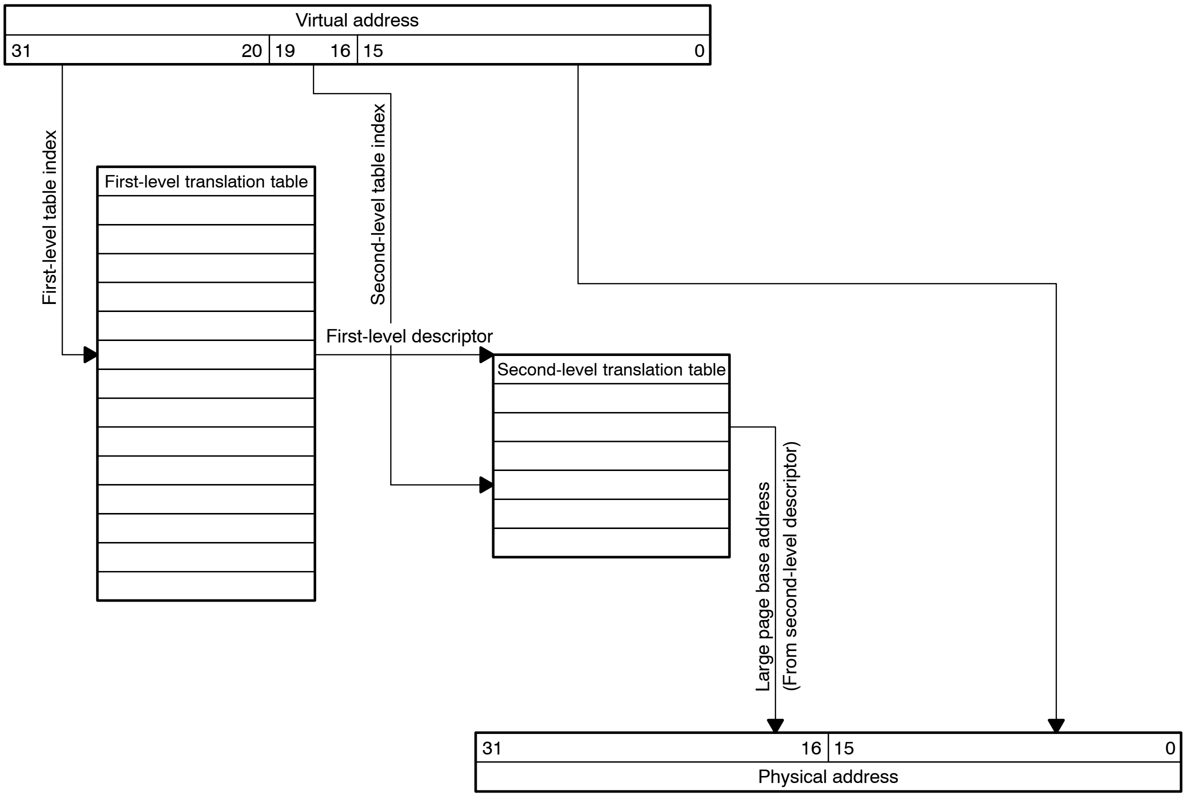
Figure 9−13. Small−Page Translation Summary



Large−Page Translation Summary

The translation of a large page is similar to that of a small page. The difference is that for a large page, only bits 19 through 16 index into the second-level translation table. The last 4 bits of the table index are implicitly assumed to be 0, as there are 16 identical consecutive entries for a large page. Figure 9−14 summarizes the translation process for large pages.

Figure 9−14. Large−Page Translation Summary



9.3.4 TLB

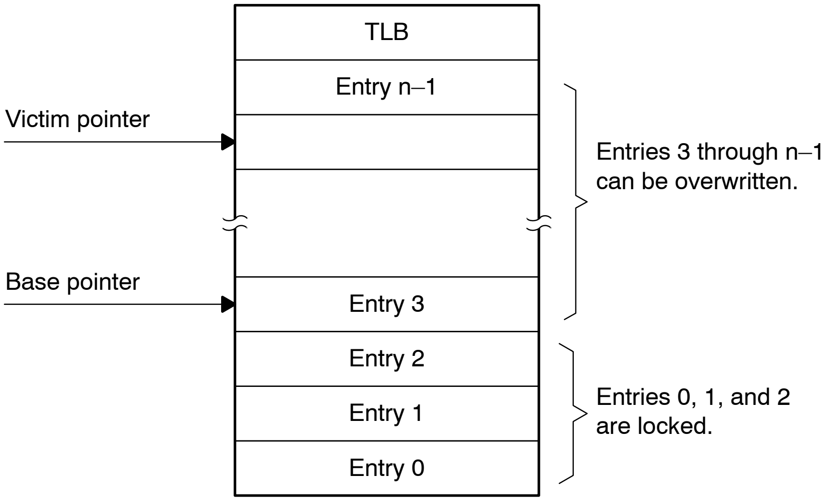
Translating virtual to physical addresses is required for each memory access in systems using an MMU. To accelerate this translation process, a cache, or translation lookaside buffer (TLB), holds the results of recent translations.

For every translation, the MMU internal logic first checks whether the requested translation is already cached in the TLB. If the translation is already cached, this translation is used; otherwise it is retrieved from the translation tables and the TLB is updated. If the TLB is already full, one of its entries must be replaced. This entry is selected randomly.

The first *n* TLB entries (with *n* < total number of TLB entries) can be protected—or locked—against being overwritten by setting the TLB base pointer to *n*. See Figure 9−15. If this mechanism is used, only nonprotected entries can be overwritten. The victim pointer indicates the next TLB entry to be written.

> **Note:**
>
> The last TLB entry (Entry n−1) is always unprotected.

Figure 9−15. TLB Entry Lock Mechanism


TLB
Entry n–1
Victim pointer
Entries 3 through n–1 can be overwritten.
Base pointer
Entry 3
Entry 2
Entry 1
Entry 0
Entries 0, 1, and 2 are locked.

NOTE: The last TLB entry (Entry n–1) is always unprotected.

TLB entries are automatically written by the table−walking logic. They can also be manually written. This is typically done to ensure that the translation of time-critical data accesses is already present in the TLB so that they execute as fast as possible. These entries must be locked to prevent them from being overwritten.

9.3.4.1 TLB Entry Format

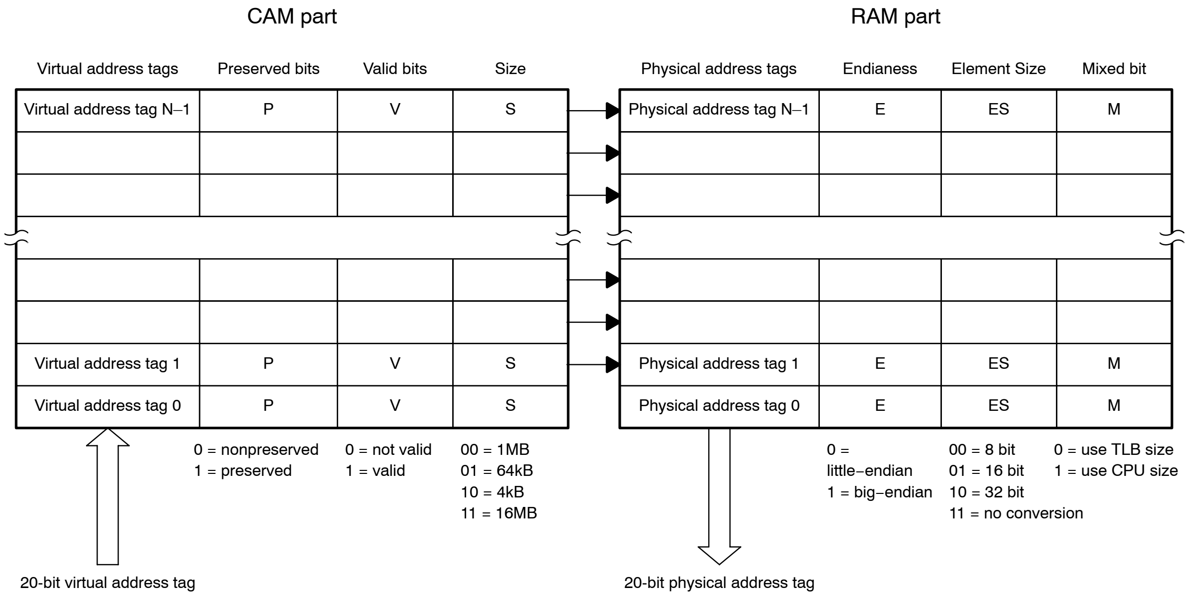
TLB entries consist of two parts:

- ❑ The CAM part contains the virtual address tag used to check whether a virtual address translation is in the TLB. The TLB acts like a fully associative cache addressed by the virtual address tag. It also contains the section/page size, as well as the preserved and the valid parameters.
- ❑ The RAM part contains the address translation that belongs to the virtual address tag. It also contains the endianness, element size, and mixed parameters described in Section 9.3.3.2, *First−Level Translation Table.*

The *valid* parameter specifies whether an entry is valid. The *preserved* parameter determines the behavior of an entry in the event of a TLB flush. If an entry is set as preserved, it is not deleted on a TLB flush. Preserved entries must be deleted manually. The procedure for deleting TLB entries is described in Section 9.4.2.3, *Deleting TLB Entries*.

Figure 9−16 shows the TLB entry structure.

Figure 9−16. TLB Entry Structure



9.3.5 MMU Error Handling

The following types of faults can occur:

- TLB miss with table walker disabled

  No translation is found for the required virtual addres. If the hardware table walker is disabled, a fault is generated.

- Translation fault

  No translation is found for the required virtual address (TLB miss). The table walker is enabled, but no page table entry exists for the given virtual address.

- Table walk fault

  A table walk results in a memory read error.

- Multihit fault

  More than one valid entry exists in the TLB for the given virtual address.

When a fault occurs and its corresponding interrupt is enabled, an interrupt is signaled to the MPU. The interrupt service routine (ISR) is responsible for fault recovery. The requestor is stalled by the MMU while the fault is handled. For example, for a TLB miss, the ISR can load the missing entry into the TLB.

The ISR can determine the cause of the fault interrupt by reading the MMU_IRQSTATUS register. The virtual address that caused the fault can be determined by reading the MMU_FAULT_AD register.

In the case of a TLB miss, the MMU continues servicing the request when a valid TLB entry is written. In the case of a translation fault, a table−walk fault, or a multihit fault, the ISR first addresses the cause of this fault and then releases the MMU by writing to the interrupt status register. The MMU then continues servicing the request.

9.3.6 MMU Endianness Handling

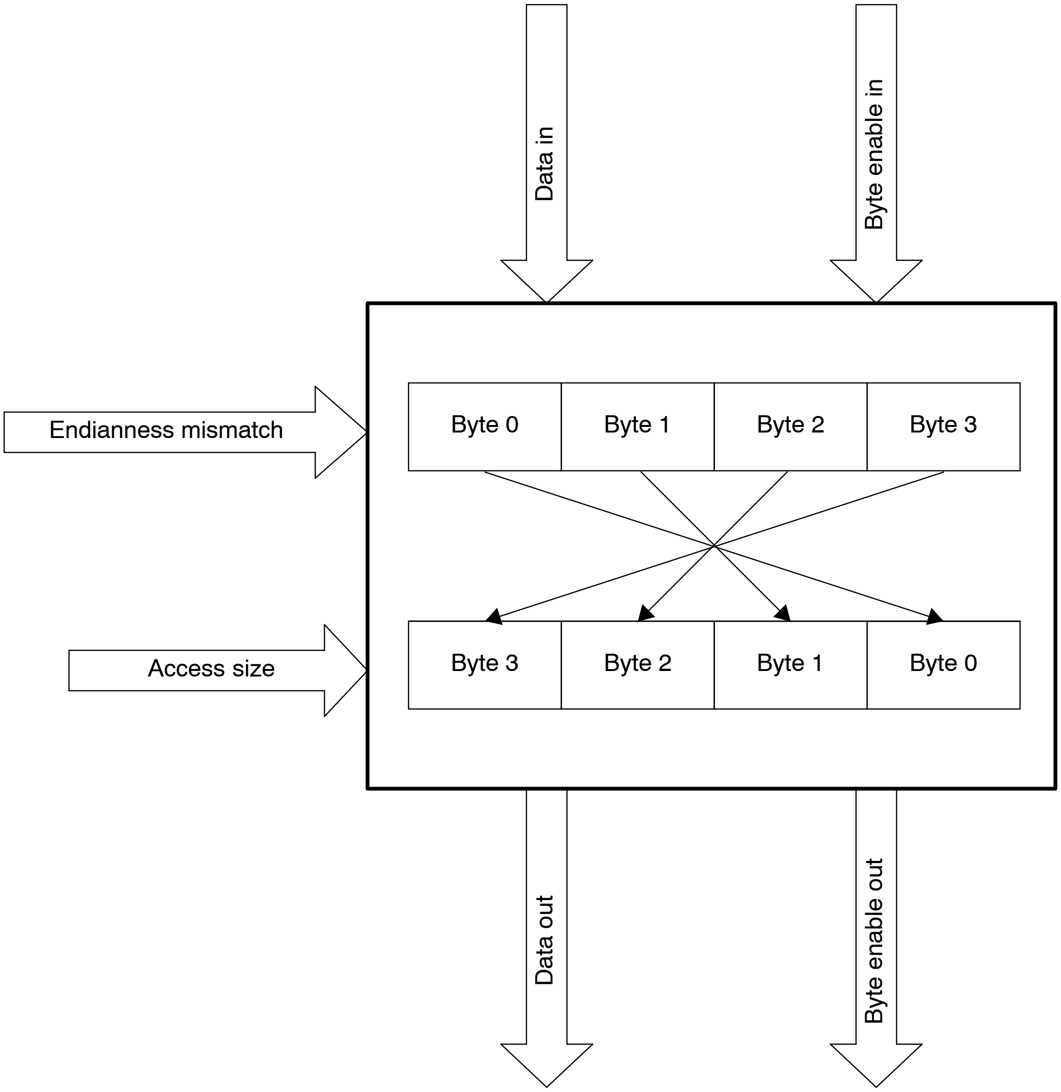
The OMAP2420 system has both big− and little−endian components. Endianness conversion is required to transfer data efficiently between components of different endianness. Endianness conversion can be performed by the initiator of a data transfer or by the MMU.

The MMU checks for endianness mismatches during the request processing between the page-based endianness parameter from the TLB and the initiator endianness qualifier. In case of a mismatch, endianness conversion can be performed in the MMU.

The MMU must know the element size to perform endianness conversion. This element size information is either taken from the TLB and page tables if the corresponding M (mixed) bit = 0, or detected from the initiator access if the M bit = 1.

When the MMU is disabled, no endianness conversion is performed on requests. Figure 9−17 shows the function of the endianness conversion block in the MMU for the case of 32-bit data.

Figure 9−17. Endianness Conversion in the MMU for 32-Bit Data



9.3.7 MMU Instance Design Parameters

The different MMU instances have different design parameters, most notably for the size of the virtual address space and the number of TLB entries. Table 9−7 shows the correspondence between the MMU instances and these design parameters.

Table 9−7. Design Parameters of the MMU Instances

MMU Instance	Virtual Address Space	TLB Entries
DSP MMU	16MB	32 entries
Camera MMU	4GB	8 entries

9.4 MMU Programming Model

The MMU components must be set up before the MMU can be used for address translation. The following steps are required:

- ❑ Initialize translation tables and the table−walker logic if translation tables are used.
- ❑ Write TLB entries if required.
- ❑ Enable the MMU.

9.4.1 Translation Table Setup

If translation tables are used for the MMU address translation, they must be correctly set up, and the table−walking logic must be enabled.

The first step is to build the translation tables (first-level translation table and second-level translation tables, if necessary) and place them in memory. The start address of the translation tables must be aligned according to their size.

This means that a 4096-entry first-level translation table must be aligned on a 16KB boundary (4096 entries ∗ 4 bytes = 16KB); that is. the last 14 bits of the translation table start address must be 0. The minimum alignment is 128-bytes; that is, the last 7 bits of the start address must always be 0, independent of the translation table size.

After the translation tables are written to memory, the translation table base address (the most−significant bits [MSBs] of the first-level translation table start address) must be set using the TTB_ADDRESS_REGISTER register. For a complete description of the MMU control registers, see Section 9.5, *MMU Registers*.

After the translation tables are written to memory and the translation table base is set, the table−walking logic can be enabled by setting the TWLENABLE bit in the MMU_CNTL register.

9.4.2 TLB Setup

In many cases, the TLB is initialized to hold some commonly used or very time-critical address translations. These address translations must be protected from being overwritten by the table−walking logic.

9.4.2.1 Writing TLB Entries

Initially after reset, the TLB is empty. TLB entries can then be initialized as needed, starting from entry 0. To initialize TLB entries, follow this procedure:

1) Load the virtual address, the preserved bit, the valid bit, and the section/ page size into the MMU_CAM register.
2) Load the physical address and the endianness, element size, and mixed bits into the MMU_RAM register.

3) Specify the TLB entry to write by setting the CURRENTVICTIM pointer in the MMU_LOCK register. Start with TLB entry 0 and increment for each subsequent entry to be written.
4) Set the LDTLBITEM bit of the MMU_LD_TLB register to load the specified entry.
5) Repeat Steps 1 through 4 for all entries to be written. Remember to increment the CURRENTVICTIM pointer with each entry.

9.4.2.2 Protecting TLB Entries

The first *n* TLB entries (with *n* < total number of TLB entries) can be protected from being overwritten with new translations. This is useful to ensure that commonly used or time-critical translations are always in the TLB and do not require retrieval using the table−walking process.

The entry protection mechanism is shown in Figure 9−15. To protect the first *n* TLB entries, set the BASEVALUE field in the MMU_LOCK register to *n*.

9.4.2.3 Deleting TLB Entries

Two mechanisms exist to delete TLB entries. All unpreserved TLB entries (TLB entries written with the preserved bit set to 0), can be deleted by invoking a TLB flush. Such a TLB flush is invoked by setting the GLOBALFLUSH bit in the MMU_GFLUSH register.

Individual TLB entries can be flushed, regardless of the preserved bit setting, by specifying its virtual address in the MMU_CAM register and setting the FLUSHENTRY bit in the MMU_FLUSH_ENTRY register.

Because the preserved bit does not prevent replacement by the table−walking logic, it must be used only on protected TLB entries.

9.4.2.4 Reading TLB Entries

TLB entries can be read to determine the TLB content at run time. The TLB entry number is specified by setting the CURRENTVICTIM pointer. The CAM and RAM parts of the TLB entry can then be read in the MMU_READ_CAM and MMU_READ_RAM registers, respectively.

9.4.3 MMU Enable

The MMU is enabled by setting the MMUENABLE bit in the MMU_CNTL register. Clearing this bit disables the MMU. When the MMU is disabled, no translation is done—virtual addresses are treated as physical addresses—and no endianness conversion is performed.

9.5 MMU Registers

9.5.1 MMU Instance Register Sets

The DSP MMU and camera MMU instances are programmed using three identical sets of configuration registers. Table 9−8 lists the base address of each register set.

Table 9−8. MMU Configuration Register Set Base Addresses

MMU Instance	Base Address	Size
DSP MMU	0x5A00:0000	4KB
Camera MMU	0x4805:2C00	1KB

Note:

The MPU MMU is configured using CP15. For details, see Chapter 3, *MPU Subsystem*.

9.5.2 MMU Register Set

Each MMU instance except the MPU MMU is programmed using a set of 18 configuration registers. Table 9−9 lists these registers and their access types, access sizes, and offsets against their base address.

Note:

The registers and their bit fields in this set are identical for all three instances, except for the MMU_LOCK register, which has bit fields for the camera MMU that are different from those for the DSP MMU.

Table 9−9. MMU Register Set

Register Name	Access Type	Access Size	Address Offset
MMU_REVISION	R	32 bits	0x00
MMU_SYSCONFIG	RW	32 bits	0x10
MMU_SYSSTATUS	R	32 bits	0x14
MMU_IRQSTATUS	RW	32 bits	0x18
MMU_IRQENABLE	RW	32 bits	0x1C
MMU_WALKING_ST	R	32 bits	0x40
MMU_CNTL	RW	32 bits	0x44
MMU_FAULT_AD	R	32 bits	0x48
MMU_TTB	RW	32 bits	0x4C
MMU_LOCK	RW	32 bits	0x50
MMU_LD_TLB	RW	32 bits	0x54
MMU_CAM	RW	32 bits	0x58

Table 9–9. MMU Register Set (Continued)

Register Name	Access Type	Access Size	Address Offset
MMU_RAM	RW	32 bits	0x5C
MMU_GFLUSH	RW	32 bits	0x60
MMU_FLUSH_ENTRY	RW	32 bits	0x64
MMU_READ_CAM	R	32 bits	0x68
MMU_READ_RAM	R	32 bits	0x6C
MMU_EMU_FAULT_AD	R	32 bits	0x70

9.5.3 MMU Register Descriptions

Table 9–10. MMU_REVISION

Address Offset	0x000		
Physical Address	0x5A00:0000 0x4805:2C00	**Instance**	DSP MMU Camera MMU
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 Examples: 0x10 for 1.0, 0x21 for 2.1	R	

Table 9−11. MMU_SYSCONFIG

Address Offset	0x010		
Physical Address	0x5A00:0010 0x4805:2C10	**Instance**	DSP MMU Camera MMU
Description	This register controls the OCP interface 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																						CLOCKACTIVITY		RESERVED			IDLEMODE		RESERVED	SOFTRESET	AUTOIDLE

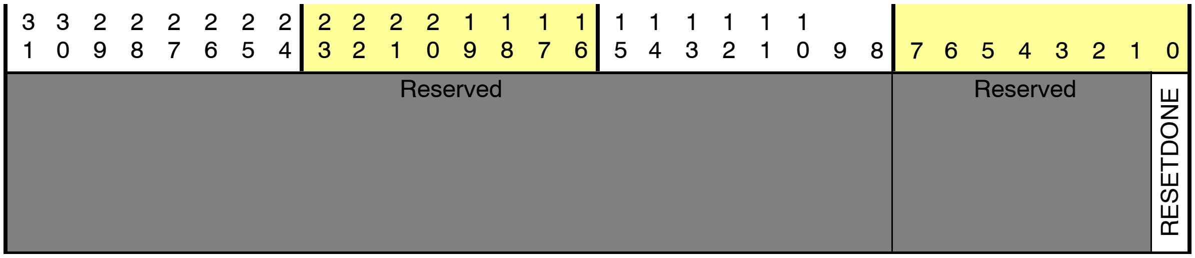
Bits	Field Name	Description	Type	Reset
31:10	Reserved	Read returns 0. Write 0 for future compatibility.	R	0x000000
9:8	CLOCKACTIVITY	Clock activity during wake-up mode 0x0: Functional and OCP clocks can be switched off. Read 0x1, 0x2, 0x3 never happens. Write 0 for future compatibility.	R	0x0
7:5	Reserved	Read returns 0. Write 0 for future compatibility.	R	0x0
4:3	IDLEMODE	Idle mode 0x0: Force−idle. Idle request is acknowledged unconditionally. 0x1: No−idle. Idle request is never acknowledged. 0x2: Smart−idle. An idle request is acknowledged based on the internal activity of the module. 0x3: Reserved—Do not use.	RW	0x0
2	Reserved	Read returns 0. Write 0 for future compatibility.	R	0
1	SOFTRESET	Software reset. This bit is automatically reset by the hardware. Read returns 0. Read 0x0: Always returns 0 Write 0x0: No functional effect Read 0x1: Never happens Write 0x1: The module is reset.	RW	0

Table 9−11. MMU_SYSCONFIG (Continued)

Bits	Field Name	Description	Type	Reset
0	AUTOIDLE	Internal OCP clock−gating strategy 0x0: OCP clock is free-running. 0x1: Automatic OCP clock−gating strategy is applied, based on OCP interface activity.	RW	0

Table 9−12. MMU_SYSSTATUS

Address Offset	0x014		
Physical Address	0x5A00:0014 0x4805:2C14	**Instance**	DSP MMU Camera MMU
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	Read returns 0.	R	0x000000
7:1	Reserved	Read returns 0. Reserved for OCP-socket status information	R	0x00
0	RESETDONE	Internal reset monitoring 0x0: Internal module reset is ongoing. 0x1: Reset complete1	R	–

1) During reset, the value is 0, but the value read just after the reset is 1, because ICLK/FCLK is already operating.

Table 9−13. MMU_IRQSTATUS

Address Offset	0x018		
Physical Address	0x5A00:0018 0x4805:2C18	**Instance**	DSP MMU Camera MMU
Description	This interrupt status register regroups the statuses of module internal events that can generate 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																											MULTIHITFAULT	TABLEWALKFAULT	EMUMISS	TRANSLATIONFAULT	TLBMISS

Bits	Field Name	Description	Type	Reset
31:5	Reserved	Read returns 0. Write 0 for future compatibility.	R	0x0000000
4	MULTIHITFAULT	Error caused by multiple matches in the TLB Read 0x0: MultiHitFault false Write 0x0: MultiHitFault status bit unchanged Read 0x1: MultiHitFault is true (pending). Write 0x1: MultiHitFault status bit is reset.	RW	0
3	TABLEWALKFAULT	Error response received during a table walk Read 0x0: TableWalkFault false Write 0x0: TableWalkFault status bit unchanged Read 0x1: TableWalkFault is true (pending). Write 0x1: TableWalkFault status bit is reset.	RW	0
2	EMUMISS	Unrecoverable TLB miss during debug (hardware TWL disabled) Read 0x0: EMUMiss false Write 0x0: EMUMiss status bit unchanged Read 0x1: EMUMiss is true (pending). Write 0x1: EMUMiss status bit is reset.	RW	0

Table 9−13. MMU_IRQSTATUS (Continued)

Bits	Field Name	Description		Type	Reset
1	TRANSLATION-FAULT	Invalid descriptor in translation tables (translation fault)		RW	0
		Read 0x0:	TranslationFault false		
		Write 0x0:	TranslationFault status bit unchanged		
		Read 0x1:	TranslationFault is true (pending).		
		Write 0x1:	TranslationFault status bit is reset.		
0	TLBMISS	Unrecoverable TLB miss (hardware TWL disabled)		RW	0
		Read 0x0:	TLBMiss false		
		Write 0x0:	TLBMiss status bit unchanged		
		Read 0x1:	TLBMiss is true (pending).		
		Write 0x1:	TLBMiss status bit is reset.		

Table 9−14. MMU_IRQENABLE

Address Offset	0x01C		
Physical Address	0x5A00:001C 0x4805:2C1C	**Instance**	DSP MMU Camera MMU
Description	The interrupt enable register allows masking/unmasking the module internal sources of interrupt, on a event-by-event basis.		
Type	RW		

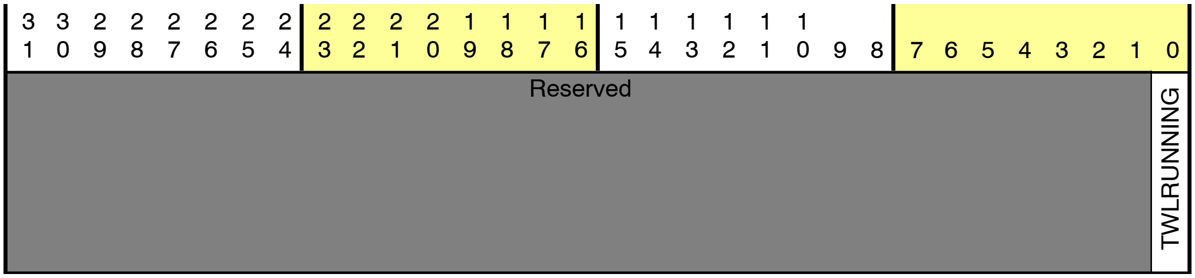
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											MULTIHITFAULT	TABLEWALKFAULT	EMUMISS	TRANSLATIONFAULT	TLBMISS

Table 9−14. MMU_IRQENABLE (Continued)

Bits	Field Name	Description	Type	Reset
31:5	Reserved	Read returns 0. Write 0 for future compatibility.	R	0x0000000
4	MULTIHITFAULT	Error caused by multiple matches in the TLB 0x0: MultiHitFault is masked. 0x1: MultiHitFault event generates an interrupt if fault occurs.	RW	0
3	TABLEWALKFAULT	Error response received during a table walk 0x0: TableWalkFault is masked. 0x1: TableWalkFault event generates an interrupt if error occurs.	RW	0
2	EMUMISS	Unrecoverable TLB miss during debug (hardware TWL disabled) 0x0: EMUMiss interrupt is masked. 0x1: EMUMiss event generates an interrupt if miss occurs.	RW	0
1	TRANSLATION-FAULT	Invalid descriptor in translation tables (translation fault) 0x0: TranslationFault is masked. 0x1: TranslationFault event generates an interrupt if fault occurs.	RW	0
0	TLBMISS	Unrecoverable TLB miss (hardware TWL disabled) 0x0: TLBMiss interrupt is masked. 0x1: TLBMiss event generates an interrupt if miss occurs.	RW	0

Table 9−15. MMU_WALKING_ST

Address Offset	0x040		
Physical Address	0x5A00:0040 0x4805:2C40	**Instance**	DSP MMU Camera MMU
Description	This register provides status information about the table−walking logic.		
Type	R		



Bits	Field Name	Description	Type	Reset
31:1	Reserved	Read returns 0.	R	0x00000000
0	TWLRUNNING	Table−walking logic is running. 0x0: TWL complete 0x1: TWL running	R	0

Table 9−16. MMU_CNTL

Address Offset	0x044		
Physical Address	0x5A00:0044 0x4805:2C44	**Instance**	DSP MMU Camera MMU
Description	This register programs the MMU features.		
Type	RW		

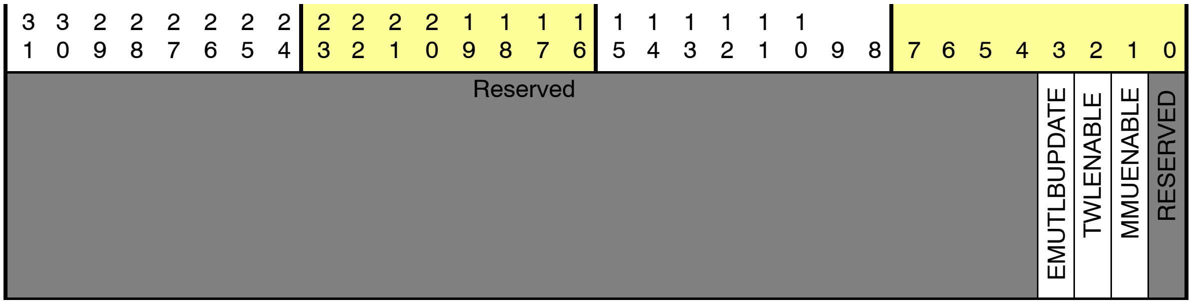


Table 9−16. MMU_CNTL (Continued)

Bits	Field Name	Description	Type	Reset
31:4	Reserved	Read returns 0. Write 0 for future compatibility.	R	0x0000000
3	EMUTLBUPDATE	Enable TLB update on emulator table walk. 0x0: Emulator TLB update disabled 0x1 Emulator TLB update enabled	RW	0
2	TWLENABLE	Table−walking logic enable 0x0: TWL disabled 0x1: TWL enabled	RW	0
1	MMUENABLE	MMU enable 0x0 MMU disabled 0x1: MMU enabled	RW	0
0	Reserved	Read returns 0. Write 0 for future compatibility.	R	0

Table 9−17. MMU_FAULT_AD

Address Offset	0x048		
Physical Address	0x5A00:0048 0x4805:2C48	**Instance**	DSP MMU Camera MMU
Description	This register contains the virtual address that generated the interrupt.		
Type	R		

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="32">FAULTADDRESS</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
31:0	FAULTADDRESS	Virtual address of the access that generated a fault	R	0x00000000

Table 9−18. MMU_TTB

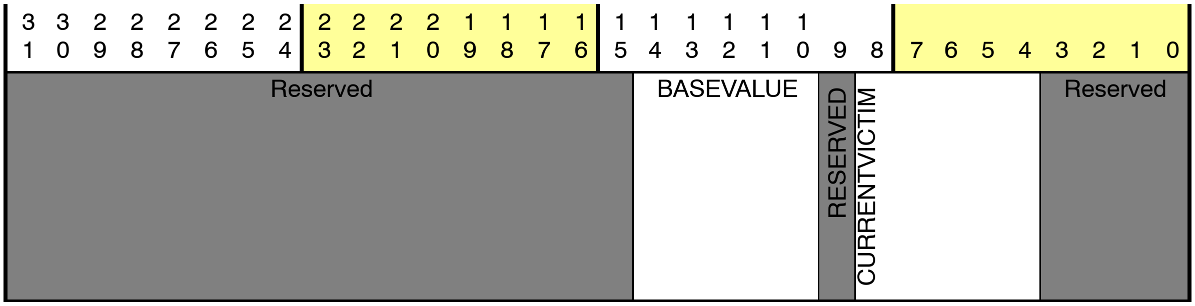
Address Offset	0x04C		
Physical Address	0x5A00:004C 0x4805:2C4C	**Instance**	DSP MMU Camera MMU
Description	This register contains the translation table base address.		
Type	RW		

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="25">TTBADDRESS</td><td colspan="7">Reserved</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
31:7	TTBADDRESS	Translation table base address	RW	0x0000000
6:0	Reserved	Read returns 0. Write 0 for future compatibility.	R	0x00

Table 9−19.MMU_LOCK (DSP MMU Instance)

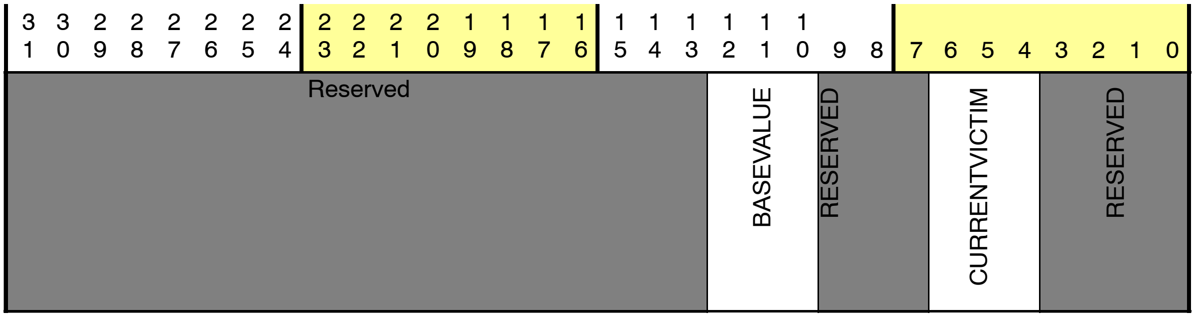
Address Offset	0x050		
Physical Address	0x5A00:0050	**Instance**	DSP MMU
Description	This register locks some TLB entries or specifies the TLB entry to be read.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:15	Reserved	Read returns 0. Write 0 for future compatibility.	R	0x00000
14:10	BASEVALUE	Locked entries base value	RW	0x00
9	Reserved	Read returns 0. Write 0 for future compatibility.	R	0
8:4	CURRENT VICTIM	Current entry to be updated by the TWL, by the software, or by the TLB entry to be read Write value: TLB entry to be updated by software or by the TLB entry to be read Read value: TLB entry to be updated by table−walk logic	RW	0x00
3:0	Reserved	Read returns 0. Write 0 for future compatibility.	R	0x0

Table 9–20. MMU_LOCK – Camera MMU Instance

Address Offset	0x050		
Physical Address	0x4805:2C50	**Instance**	Camera MMU
Description	This register locks some TLB entries. or specifies the TLB entry to be read		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:13	Reserved	Read returns 0. Write 0 for future compatibility.	R	0x00000
12:10	BASEVALUE	Locked entries base value	RW	0x00
9:7	Reserved	Read returns 0. Write 0 for future compatibility.	R	0
6:4	CURRENT VICTIM	Current entry to be updated by the TWL, by the software, or by the TLB entry to be read Write value: TLB entry to be updated by software or by the TLB entry to be read Read value: TLB entry to be updated by table–walk logic.	RW	0x00
3:0	Reserved	Read returns 0. Write 0 for future compatibility.	R	0x0

Table 9–21. MMU_LD_TLB

Address Offset	0x054		
Physical Address	0x5A00:0054	**Instance**	DSP MMU
	0x4805:2C54		Camera MMU
Description	This register loads a TLB entry (CAM + RAM).		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															LDTLBITEM

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Write 0 for future compatibility Read returns 0.	RW	0x00000000
0	LDTLBITEM	Write (load) data in the TLB Read 0x0: Always returns 0 Write 0x0: No functional effect Read 0x1: Never happens Write 0x1: Load TLB data	RW	0

Table 9–22. MMU_CAM

Address Offset	0x058		
Physical Address	0x5A00:0058 0x4805:2C58	**Instance**	DSP MMU Camera MMU
Description	This register holds a CAM entry.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VATAG																				Reserved								P	V	PAGESIZE	

Bits	Field Name	Description		Type	Reset
31:12	VATAG	Virtual address tag		RW	0x00000
11:4	Reserved	Read returns 0. Write 0 for future compatibility.		R	0x00
3	P	Preserved bit		RW	0
		0x0:	TLB entry can be flushed.		
		0x1:	TLB entry is protected against flush.		
2	V	Valid bit		RW	0
		0x0:	TLB entry is invalid.		
		0x1:	TLB entry is valid.		
1:0	PAGESIZE	Page size		RW	0x0
		0x0:	Section (1MB)		
		0x1:	Large page (64KB)		
		0x2	Small page (4KB)		
		0x3:	Supersection (16MB)		

Table 9−23. MMU_RAM

Address Offset	0x05C		
Physical Address	0x5A00:005C 0x4805:2C5C	**Instance**	DSP MMU Camera MMU
Description	This register holds a RAM entry.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PHYSICALADDRESS																				RESERVED		ENDIANNESS	ELEMENTSIZE		MIXED	Reserved					

Bits	Field Name	Description	Type	Reset
31:12	PHYSICAL ADDRESS	Physical address of the page	RW	0x00000
11:10	Reserved	Read returns 0. Write 0 for future compatibility.	R	0x0
9	ENDIANNESS	Endianness of the page 0x0: Little−endian 0x1: Big−endian	RW	0
8:7	ELEMENTSIZE	Element size of the page (8, 16, 32, no translation) 0x0: 8 bits 0x1: 16 bits 0x2: 32 bits 0x3: No translation	RW	0x0
6	MIXED	Mixed page attribute (use CPU element size) 0x0: Use TLB element size. 0x1: Use CPU element size.	RW	0
5:0	Reserved	Read returns 0. Write 0 for future compatibility.	R	0x00

Table 9−24. MMU_GFLUSH

Address Offset	0x060		
Physical Address	0x5A00:0060 0x4805:2C60	**Instance**	DSP MMU Camera MMU
Description	This register flushes all nonprotected TLB entries.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															GLOBALFLUSH

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x00000000
0	GLOBALFLUSH	Flush all nonprotected TLB entries when set. Read 0x0: Always returns 0 Write 0x0: No functional effect Read 0x1: Never happens Write 0x1: Flush all nonprotected TLB entries.	RW	0

Table 9–25. MMU_FLUSH_ENTRY

Address Offset	0x064		
Physical Address	0x5A00:0064 0x4805:2C64	**Instance**	DSP MMU Camera MMU
Description	This register flushes the entry pointed to by the CAM virtual 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																															FLUSHENTRY

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0x00000000
0	FLUSHENTRY	Flush the TLB entry pointed to by the virtual address (VATag) in the MMU_CAM register, even if this entry is set protected. Read 0x0: Always returns 0 Write 0x0: No functional effect Read 0x1 Never happens Write 0x1: Flush all the TLB entries specified by the CAM register.	RW	0

Table 9–26. MMU_READ_CAM

Address Offset	0x068		
Physical Address	0x5A00:0068 0x4805:2C68	**Instance**	DSP MMU Camera MMU
Description	This register reads CAM data from a CAM entry.		
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
VATAG																				Reserved								P	V	PAGESIZE	

Table 9–26. MMU_READ_CAM (Continued)

Bits	Field Name	Description	Type	Reset
31:12	VATAG	Virtual address tag	R	0x00000
11:4	Reserved	Read returns 0.	R	0x00
3	P	Preserved bit 0x0: TLB entry can be flushed. 0x1: TLB entry is protected against flush.	R	0
2	V	Valid bit 0x0: TLB entry is invalid. 0x1: TLB entry is valid.	R	0
1:0	PAGESIZE	Page size 0x0: Section (1MB) 0x1: Large page (64KB) 0x2: Small page (4KB) 0x3: Supersection (16MB)	R	0x0

Table 9–27. MMU_READ_RAM

Address Offset	0x06C		
Physical Address	0x5A00:006C	**Instance**	DSP MMU
	0x4805:2C6C		Camera MMU
Description	This register reads RAM data from a RAM entry.		
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
PHYSICALADDRESS																				RESERVED		ENDIANNESS	ELEMENTSIZE		MIXED	Reserved					

Bits	Field Name	Description	Type	Reset
31:12	PHYSICAL ADDRESS	Physical address of the page	R	0x00000
11:10	Reserved	Read returns 0.	R	0x0

Table 9−26. MMU_READ_RAM (Continued)

Bits	Field Name	Description	Type	Reset
9	ENDIANNESS	Endianness of the page 0x0: Little−endian 0x1: Big−endian	R	0
8:7	ELEMENTSIZE	Element size of the page (8, 16, 32 bits, or no translation) 0x0: 8 bits 0x1: 16 bits 0x2: 32 bits 0x3: No translation	R	0x0
6	MIXED	Mixed page attribute (use CPU element size) 0x0: Use TLB element size. 0x1: Use CPU element size.	R	0
5:0	Reserved	Read returns 0.	R	0x00

Table 9−28. MMU_EMU_FAULT_AD

Address Offset	0x070		
Physical Address	0x5A00:0070 0x4805:2C70	**Instance**	DSP MMU Camera MMU
Description	This register contains the last virtual address of a fault caused by the debugger.		
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
EMUFAULTADDRESS																															

Bits	Field Name	Description	Type	Reset
31:0	EMUFAULT ADDRESS	Virtual address of the last emulator access that generated a fault	R	0x00000000

Chapter 10

DMA

This chapter describes the generic direct memory access (DMA) module used in the digital signal processor (DSP) subsystem direct memory access (DDMA) module, camera subsystem direct memory access (CamDMA) module, and the system direct memory access (sDMA) module.

Topic **Page**

10.1 DMA Module Overview

The direct memory access (DMA) module/controller provides high-performance data transfers between memories and peripheral devices with low processor use. A DMA transfer is programmed using a logical DMA channel, allowing the transfer to be optimally tailored to application requirements.

The OMAP2420 architecture includes six DMA controllers. Three of the DMA controllers use the generic DMA module. The other three use DMA modules specific to their subsystems.

This chapter describes the generic DMA module used in the digital signal processor (DSP) subsystem (DDMA), camera subsystem (CamDMA), and the system DMA (sDMA).

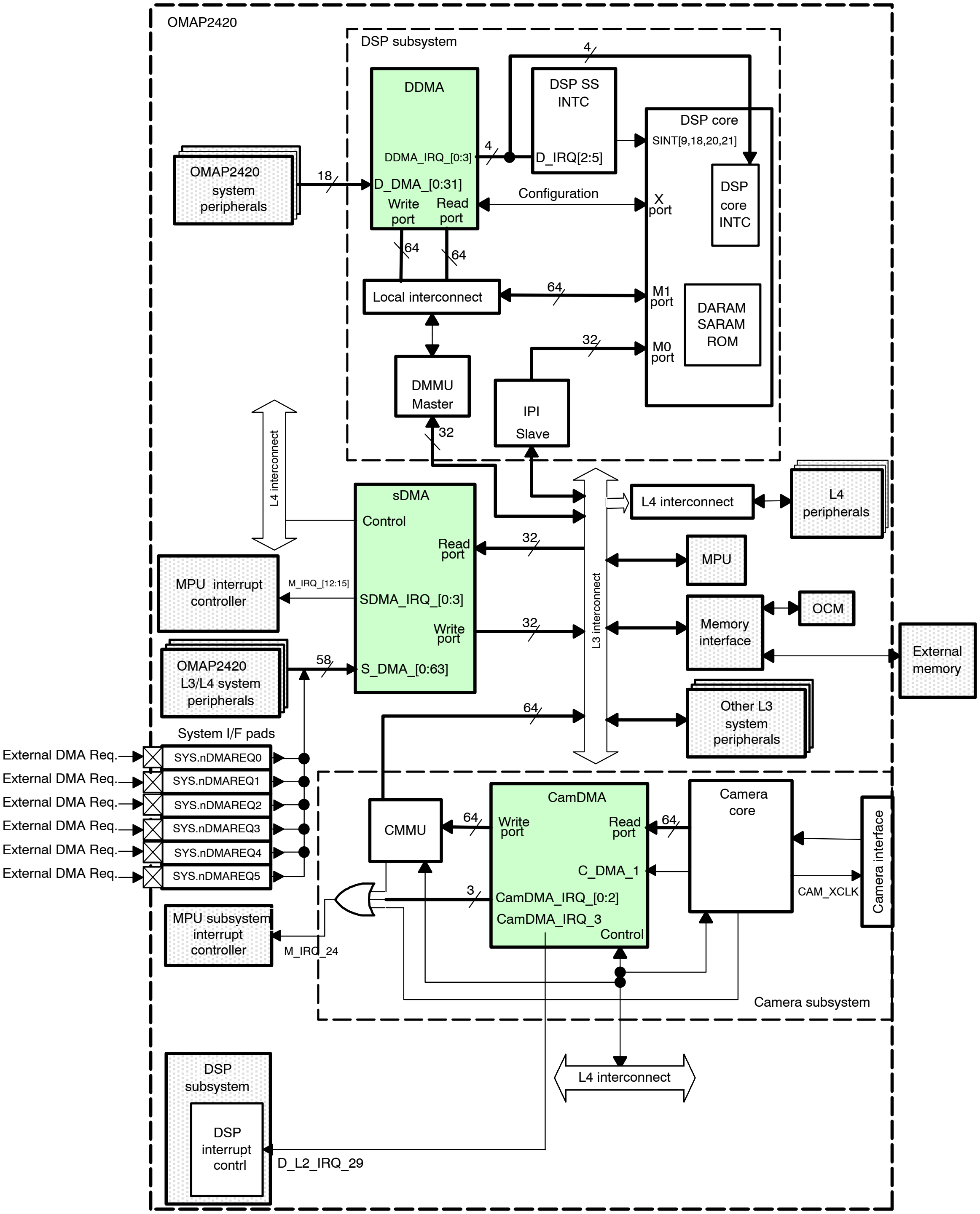
For information about a particular DMA module instance, see Section 10.3, *DMA Module Integration*, Chapter 4, *DSP Subsystem*, Chapter 12, *Memory Subsystem*, and Chapter 14, *Camera Subsystem*.

The generic DMA controller includes the following main features:

- Data transfer in either direction between:
  - Memory and memory
  - Memory and a peripheral device
- Multiple 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
  - Per-channel secure transaction attribute (depending on the DMA module instance)
- First-come-first-served DMA scheduling with high- and low-priority queues
- Relative arbitration rate between high- and low-priority channels
- When multiple DMA requests enable channels simultaneously, the channel with the lower number has higher priority.
- Four programmable interrupt request output lines
- Power-management support
- Auto-idle power-saving support
- Graphics acceleration (depending on the DMA module instance)

Figure 10−1 shows the DMA instances in the OMAP2420.

Figure 10−1. Overview of the Three DMA Module Instances: sDMA, CamDMA, and DDMA



The DMA 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 three generic DMA modules instantiated use a common programming model. The DDMA is configured only by the DSP; the sDMA and the CamDMA can be configured by the MPU or the DSP through the L4 interconnect.

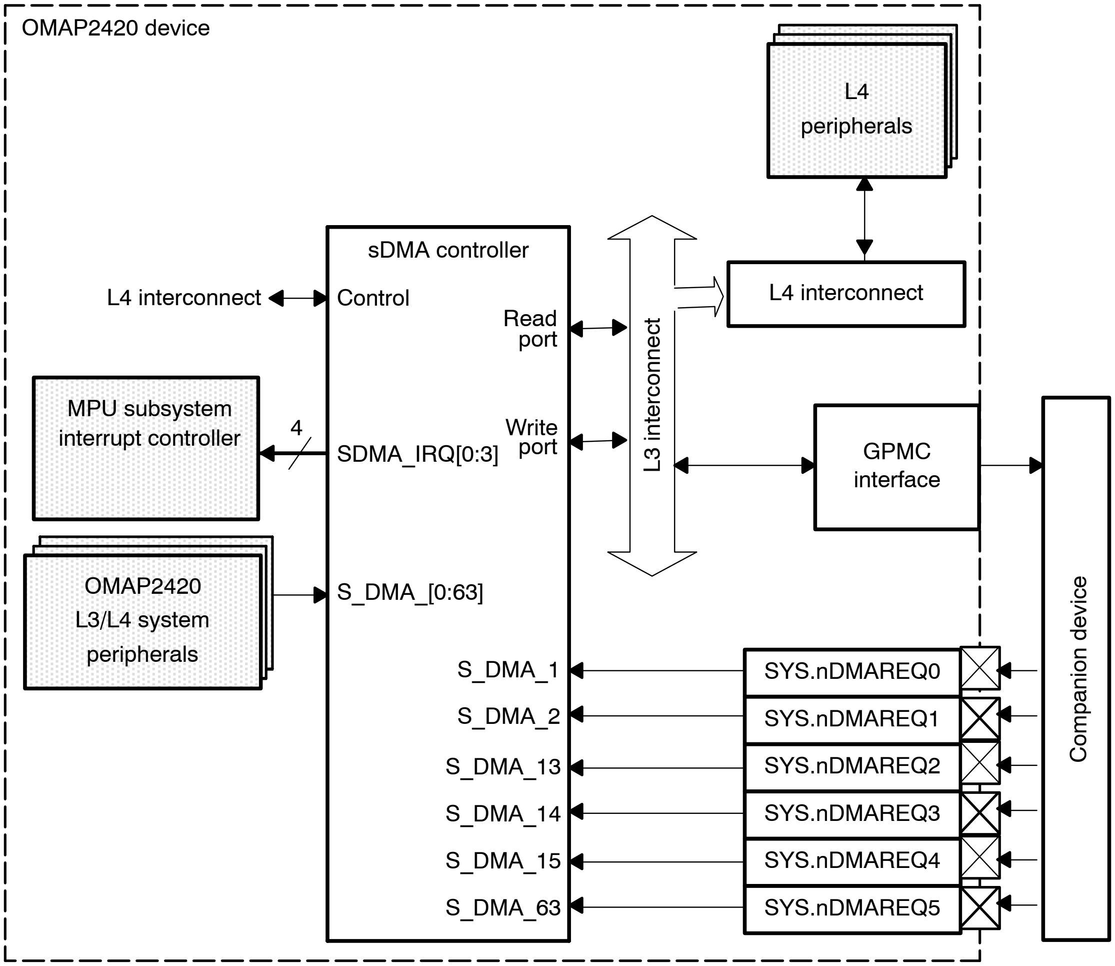
The sDMA and the DDMA can be used generically; the CamDMA has a dedicated source. The DDMA can access the DSP subsystem memory through the local interconnect. OMAP2420 system memory and peripherals are accessed through the DSP MMU and the L3 interconnect. The sDMA can access the system memory and peripherals through the L3 interconnect and can access the DSP subsystem memory through the DSP subsystem intrusive port interface (IPI) using the L3 interconnect. For more information about the IPI, see Chapter 4, *DSP Subsystem.*

The CamDMA can access OMAP2420 system memory and DSP subsystem memory through the camera MMU and the L3 interconnect.

10.2 sDMA Controller Environment

Only the sDMA can have direct connection to external pins. Figure 10−2 shows an example of how the pins can be used.

Figure 10−2. Example of External DMA Requests Use to sDMA Controller



The six SYS.nDMAREQ[5:0] 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. 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, where the source memory is located in the external device.

10.2.1 External sDMA Request Interface Description

External DMA request signals are not available on external pins by default after a cold reset. See Chapter 8, *System Control Module*, for instructions for multiplexing two signal lines to pins.

Table 10−1 describes the external sDMA request input/output (I/O) signals.

Table 10−1. I/O Description of External DMA Request Pins

Signal Name	I/O[1]	Description	Reset
SYS.nDMAREQ0	I	External DMA request to sDMA controller	Unknown
SYS.nDMAREQ1	I	External DMA request to sDMA controller	Unknown
SYS.nDMAREQ2	I	External DMA request to sDMA controller	Unknown
SYS.nDMAREQ3	I	External DMA request to sDMA controller	Unknown
SYS.nDMAREQ4	I	External DMA request to sDMA controller	Unknown
SYS.nDMAREQ5	I	External DMA request to sDMA controller	Unknown

1) I = Input, O = Output

10.2.1.1 sDMA Request Scheme

The DMA module supports two hardware DMA request line schemes:

- Edge-sensitive
- Transition-sensitive

The selection of the DMA request line as either edge- or transition-sensitive can be configured in the system control module (SCM) register bits CONTROL_DEVCONF[1:0]. The default scheme is transition-sensitive. All peripherals internal to the OMAP2420 use the transition-sensitive scheme.

Figure 10−3 shows the DMA request captured on a falling edge using the edge-sensitive scheme.

Figure 10−3. Edge-Sensitive DMA Request Scheme

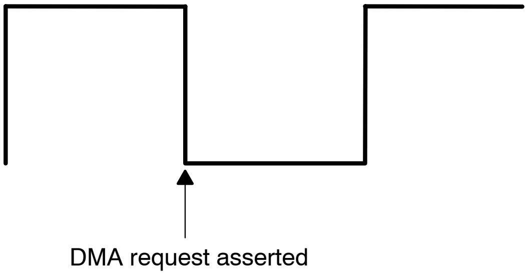
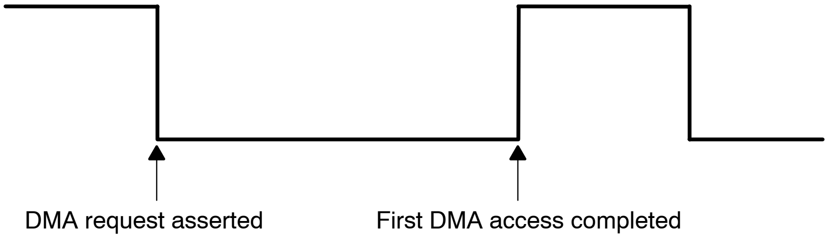


Figure 10−4 shows a transition-sensitive DMA request in which the line must be maintained low (asserted) until the first DMA access completes, after which the line must be maintained high (deasserted) for a minimum of one clock cycle (CORE_L3_CLK).

- When deassertion time is less than one clock cycle, the sDMA may 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 10−4. Transition-Sensitive DMA Request Scheme



All peripheral DMA requests in the OMAP2420 are tied to be transition-sensitive. However, an external DMA request can be configured to be either transition-sensitive or edge-sensitive. When a DMA request is configured in transition-sensitive mode and a request is pending before the channel is enabled, the sDMA considers this DMA request after enabling the corresponding channel. This ensures that a DMA request asserted before the channel is enabled is accepted in transition-sensitive mode.

10.3 DMA Module Integration

Table 10−2 lists the features available for DMA instantiation. The sDMA supports all available features; the DDMA and the CamDMA have reduced feature sets.

Table 10−2. Features Available for Each DMA Instantiation

	Transfer Feature	sDMA	DDMA	CamDMA
Per Logical Channels Support	Data transfer in either direction: Memory-memory Memory-peripheral device	√	√	√[1]
	First-come-first-served logical channel scheduling with high- and low-priority queues	√	√	√
	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	√	√	√
	Graphics acceleration (transparent copy and constant fill)	√		
	Posted write	√	√	√
	Prefetch	√	√	
	Auto-idle and software-controlled power down	√	√	√
	Four programmable interrupt request output lines	√	√	√
	LCh interleaving	√	√	√

1) Only from camera to memory; not bidirectional

Each DMA controller is based on the same architecture but with a different configuration. Table 10−3 summarizes the physical configuration of the sDMA, DDMA, and CamDMA.

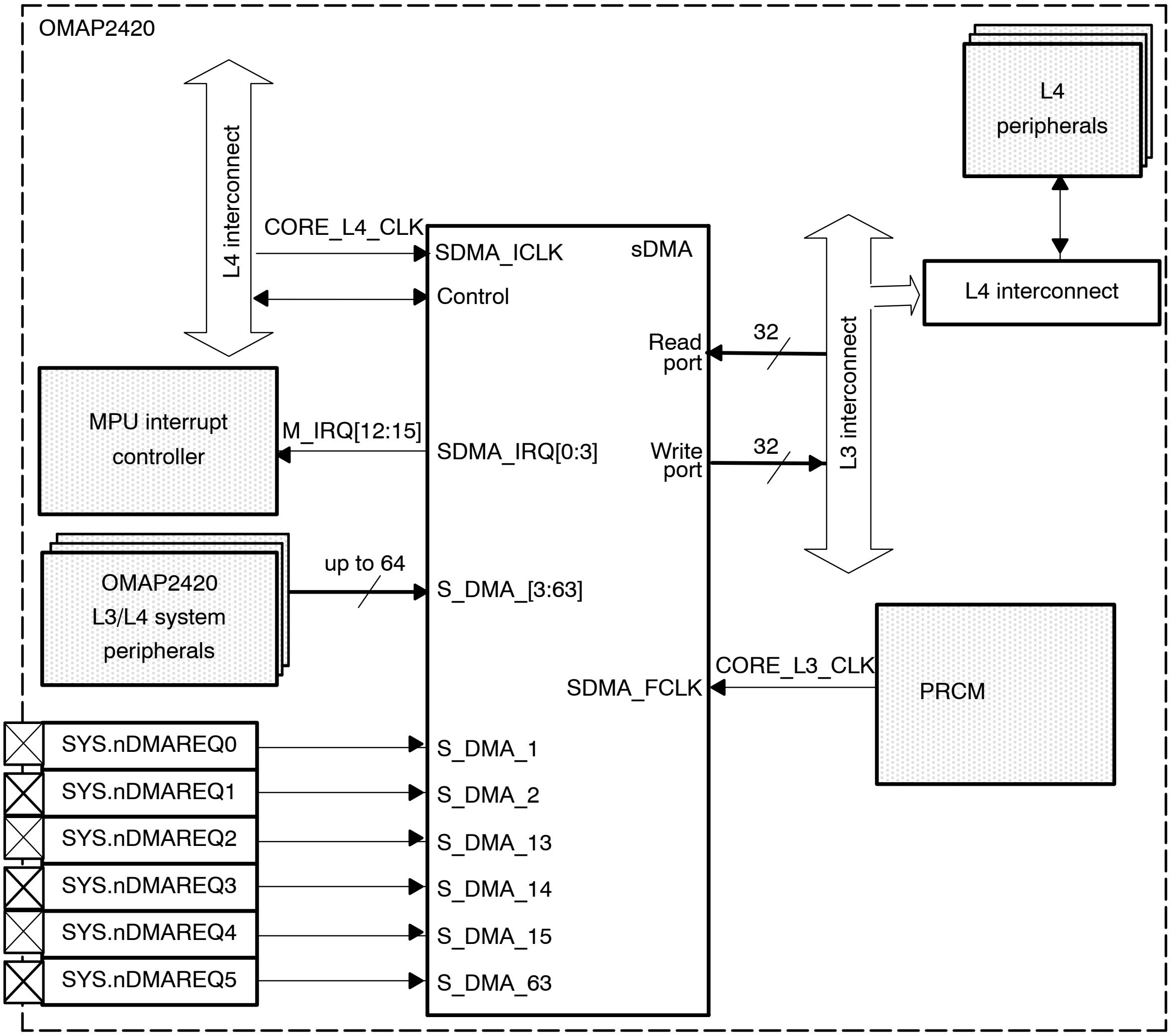
Table 10−3. Physical Differences Among the Three DMA Instantiations

Physical Feature	sDMA	DDMA	CamDMA
Data FIFO queue depth	256 x 32 bits	128 x 64 bits	16 x 64 bits
Read and write data port width	32 bits	64 bits	64 bits
Number of logical DMA channels	32	24	4
Maximum number of DMA requests	64	32	1
Graphics acceleration (transparent copy and constant fill) option	Yes	No	No

10.3.1 sDMA Controller Description

Figure 10−5 highlights sDMA controller integration in the OMAP2420.

Figure 10−5. sDMA Controller Integration



10.3.1.1 Clocking, Reset, and Power-Management Scheme

Clocking

The sDMA controller uses two clock domains:

- ❑ CORE_L4_CLK is used for the configuration port.
- ❑ CORE_L3_CLK is used both as a functional clock for all internal logic and for the two master read and write ports.

The sDMA controller supports the software-controlled standby mode with an input clock shutoff. There are no power, reset, and clock management (PRCM) clock-enable registers specific to sDMA. With the PRCM register bit (CM_AUTOIDLE3_CORE[0]), it is possible to select the sDMA behavior for the whole clock domain.

In other words, setting this bit to 1 allows SDMA_ICLK (or SDMA_FCLK) to be cut off automatically when the L4 (or L3) clock domain is to be cut. For more information about power management and clock idle, see Section 10.4.13, *Power Management*.

Table 10−4 and Table 10−5 show the recommended operation frequencies for the CORE_L3_CLK for high voltage and low voltage. For information about the CORE_L4_CLK configuration, see Chapter 5, *Power, Reset, and Clock Management.*

CAUTION

Clock configurations depend on the core voltage and maximum clock frequencies. Values described in this document may not apply to production devices.

Table 10−4. Valid sDMA, L3 Interconnect Clock Configurations for High-Voltage Operation

Clock Configuration	DPLL Clock Output (CORE_CLK) (MHz)	CORE_L3_CLK (MHz)	CORE_L3_CLK Selection: CM_CLKSEL1_CORE[4:0] Setting	PLL Clock Selection: CM_CLKSEL2_PLL[1:0] Setting
1	660	165	00100b	10
2	600	100	00110b	10
3	532	133	00100b	10
4	660	110	00110b	10

Table 10−5 describes low-power modes in which clock ratios are unchanged but DPLL output clock/2 is selected. These low-power modes allow scaling the chip voltage down to low voltage, while high voltage is required in normal mode.

Table 10−5. Valid sDMA, L3 Interconnect Clock Configurations for Low-Voltage Operation

Clock Configuration	DPLL Clock Output (CORE_CLK) (MHz)	CORE_L3_CLK (MHz)	CORE_L3_CLK Selection: CM_CLKSEL1_CORE[4:0] Setting	PLL Clock Selection: CM_CLKSEL2_PLL[1:0] Setting
1	330	82.5	00100b	01
2	300	50	00110b	01
3	268	66.5	00100b	01
4	330	55	00110b	01

Hardware Reset

The sDMA controller is part of the CORE_RST reset domain.

Software Reset

The sDMA controller can be reset independently by software using the DMA4_OCP_SYSCONFIG register.

The DMA4_OCP_SYSCONFIG[1] bit controls the software reset; setting this bit to 1 enables an active software reset functionally equivalent to hardware reset.

Power Domain

The sDMA controller is part of the CORE power domain.

10.3.1.2 Hardware Requests

Interrupts to MPU Subsystem

Table 10−6 shows how the sDMA module interrupt lines are connected to the interrupt lines of the MPU interrupt controller.

Table 10−6. sDMA Interrupt Mapping to MPU Subsystem

IRQ	Source	Description
M_IRQ_12	SDMA_IRQ0	sDMA interrupt request 0
M_IRQ_13	SDMA_IRQ1	sDMA interrupt request 1
M_IRQ_14	SDMA_IRQ2	sDMA interrupt request 2
M_IRQ_15	SDMA_IRQ3	sDMA interrupt request 3

DMA Requests to sDMA Controller

All peripherals internal to the OMAP2420 use the transition-sensitive scheme for DMA requests. For details about the transition-sensitive scheme, see Section 10.2.1.1, *sDMA Request Scheme*.

Table 10−7 describes the sDMA request mappings.

Table 10−7. sDMA Requests Mapping

sDMA Request Number	sDMA Request Line	Source	Description
1	S_DMA_0	Reserved	
2	S_DMA_1	SYS.nDMAREQ0	External DMA request 0 (system expansion)
3	S_DMA_2	SYS.nDMAREQ1	External DMA request 1 (system expansion)
4	S_DMA_3	GPMC_DMA	GPMC request from prefetch engine
5	S_DMA_4	Reserved	
6	S_DMA_5	DSS_DMA	Display subsystem—frame update request

1) Shared DMA request − also mapped on DDMA

Table 10−7. sDMA Requests Mapping (Continued)

sDMA Request Number	sDMA Request Line	Source	Description
7	S_DMA_6	Reserved	
8	S_DMA_7	Reserved	
9	S_DMA_8	Reserved	
10	S_DMA_9	Reserved	
11	S_DMA_10	Reserved	
12	S_DMA_11	Reserved	
13	S_DMA_12	Reserved	
14	S_DMA_13	SYS.nDMAREQ2	External DMA request 2 (system expansion)
15	S_DMA_14	SYS.nDMAREQ3	External DMA request 3 (system expansion)
16	S_DMA_15	SYS.nDMAREQ4	External DMA request 4 (system expansion)
17	S_DMA_16	EAC_AC_DMA_RD	EAC audio codec port − read request[1]
18	S_DMA_17	EAC_AC_DMA_WR	EAC audio codec port—write request[1]
19	S_DMA_18	EAC_MD_UL_DMA_RD	EAC modem/voice port—uplink read request[1]
20	S_DMA_19	EAC_MD_UL_DMA_WR	EAC modem/voice port—uplink write request[1]
21	S_DMA_20	EAC_MD_DL_DMA_RD	EAC modem/voice port—downlink read request [0]
22	S_DMA_21	EAC_MD_DL_DMA_WR	EAC modem/voice port—downlink write request [0]
23	S_DMA_22	EAC_BT_UL_DMA_RD	EAC Bluetooth® port—uplink read request[1]
24	S_DMA_23	EAC_BT_UL_DMA_WR	EAC Bluetooth—uplink write request[1]
25	S_DMA_24	EAC_BT_DL_DMA_RD	EAC Bluetooth—downlink read request[1]
26	S_DMA_25	EAC_BT_DL_DMA_WR	EAC Bluetooth port—downlink write request[1]
27	S_DMA_26	I2C1_DMA_TX	I^{2}C module 1—transmit request
28	S_DMA_27	I2C1_DMA_RX	I^{2}C module 1—receive request
29	S_DMA_28	I2C2_DMA_TX	I^{2}C module 2—transmit request
30	S_DMA_29	I2C2_DMA_RX	I^{2}C module 2—receive request
31	S_DMA_30	MCBSP1_DMA_TX	McBSP module 1—transmit request[1]
32	S_DMA_31	MCBSP1_DMA_RX	McBSP module 1—receive request[1]
33	S_DMA_32	MCBSP2_DMA_TX	McBSP module 2—transmit request[1]
34	S_DMA_33	MCBSP2_DMA_RX	McBSP module 2—receive request[1]
35	S_DMA_34	SPI1_DMA_TX0	McSPI module 1—transmit request channel 0
36	S_DMA_35	SPI1_DMA_RX0	McSPI module 1—receive request channel 0
37	S_DMA_36	SPI1_DMA_TX1	McSPI module 1—transmit request channel 1
38	S_DMA_37	SPI1_DMA_RX1	McSPI module 1—receive request channel 1

1) Shared DMA request − also mapped on DDMA

Table 10−7. sDMA Requests Mapping (Continued)

sDMA Request Number	sDMA Request Line	Source	Description
39	S_DMA_38	SPI1_DMA_TX2	McSPI module 1—transmit request channel 2
40	S_DMA_39	SPI1_DMA_RX2	McSPI module 1—receive request channel 2
41	S_DMA_40	SPI1_DMA_TX3	McSPI module 1—transmit request channel 3
42	S_DMA_41	SPI1_DMA_RX3	McSPI module 1—receive request channel 3
43	S_DMA_42	SPI2_DMA_TX0	McSPI module 2—transmit request channel 0
44	S_DMA_43	SPI2_DMA_RX0	McSPI module 2—receive request channel 0
45	S_DMA_44	SPI2_DMA_TX1	McSPI module 2—transmit request channel 1
46	S_DMA_45	SPI2_DMA_RX1	McSPI module 2—receive request channel 1
47	S_DMA_46	Reserved	
48	S_DMA_47	Reserved	
49	S_DMA_48	UART1_DMA_TX	UART module 1—transmit request
50	S_DMA_49	UART1_DMA_RX	UART module 1—receive request
51	S_DMA_50	UART2_DMA_TX	UART module 2—transmit request
52	S_DMA_51	UART2_DMA_RX	UART module 2—receive request
53	S_DMA_52	UART3_DMA_TX	UART module 3—transmit request[1]
54	S_DMA_53	UART3_DMA_RX	UART module 3—receive request[1]
55	S_DMA_54	USB0_DMA_TX0	USB client (port 0)—transmit request channel 0
56	S_DMA_55	USB0_DMA_RX0	USB client (port 0)—receive request channel 0
57	S_DMA_56	USB0_DMA_TX1	USB client (port 0)—transmit request channel 1
58	S_DMA_57	USB0_DMA_RX1	USB client (port 0)—receive request channel 1
59	S_DMA_58	USB0_DMA_TX2	USB client (port 0)—transmit request channel 2
60	S_DMA_59	USB0_DMA_RX2	USB client (port 0)—receive request channel 2
61	S_DMA_60	MMC_DMA_TX	MMC/SD transmit request
62	S_DMA_61	MMC_DMA_RX	MMC/SD receive request
63	S_DMA_62	Reserved	
64	S_DMA_63	SYS.nDMAREQ5	External DMA request 5 (system expansion)

1) Shared DMA request − also mapped on DDMA

10.3.2 sDMA Register Summary

Table 10−8 shows the sDMA registers and their physical addresses. For detailed descriptions of the registers, see Section 10.6.2, *Common Register Descriptions for the CamDMA, DDMA, and sDMA Controllers*.

Table 10−8. sDMA Register Summary

Register Name	Type	Register Width (Bits)	Offset Address	sDMA Physical Address
DMA4_REVISION	R	32	0x0000	0x4805 6000
DMA4_IRQSTATUS_L0	RW	32	0x0008	0x4805 6008
DMA4_IRQSTATUS_L1	RW	32	0x000C	0x4805 600C
DMA4_IRQSTATUS_L2	RW	32	0x0010	0x4805 6010
DMA4_IRQSTATUS_L3	RW	32	0x0014	0x4805 6014
DMA4_IRQENABLE_L0	RW	32	0x0018	0x4805 6018
DMA4_IRQENABLE_L1	RW	32	0x001C	0x4805 601C
DMA4_IRQENABLE_L2	RW	32	0x0020	0x4805 6020
DMA4_IRQENABLE_L3	RW	32	0x0024	0x4805 6024
DMA4_SYSSTATUS	R	32	0x0028	0x4805 6028
DMA4_OCP_SYSCONFIG	RW	32	0x002C	0x4805 602C
DMA4_CAPS_0	R	32	0x0064	0x4805 6064
DMA4_CAPS_2	R	32	0x006C	0x4805 606C
DMA4_CAPS_3	R	32	0x0070	0x4805 6070
DMA4_CAPS_4	R	32	0x0074	0x4805 6074
DMA4_GCR	RW	32	0x0078	0x4805 6078
DMA4_CCR0 − DMA4_CCR*n*	RW	32	0x0080+ (*c**0x60)	0x4805 6080− 0x4805 6C20
DMA4_CLNK_CTRL0 − DMA4_CLNK_CTRL*n*	RW	32	0x0084+ (*c**0x60)	0x4805 6084− 0x4805 6C24
DMA4_CICR0 − DMA4_CICR*n*	RW	32	0x0088+ (*c**0x60)	0x4805 6088− 0x4805 6C28
DMA4_CSR0 − DMA4_CSR*n*	RW	32	0x008C+ (*c**0x60)	0x4805 608C− 0x4805 6C2C
DMA4_CSDP0 − DMA4_CSDP*n*	RW	32	0x0090+ (*c**0x60)	0x4805 6090− 0x4805 6C30
DMA4_CEN0 − DMA4_CEN*n*	RW	32	0x0094+ (*c**0x60)	0x4805 6094− 0x4805 6C34

Table 10−8.sDMA Register Summary (Continued)

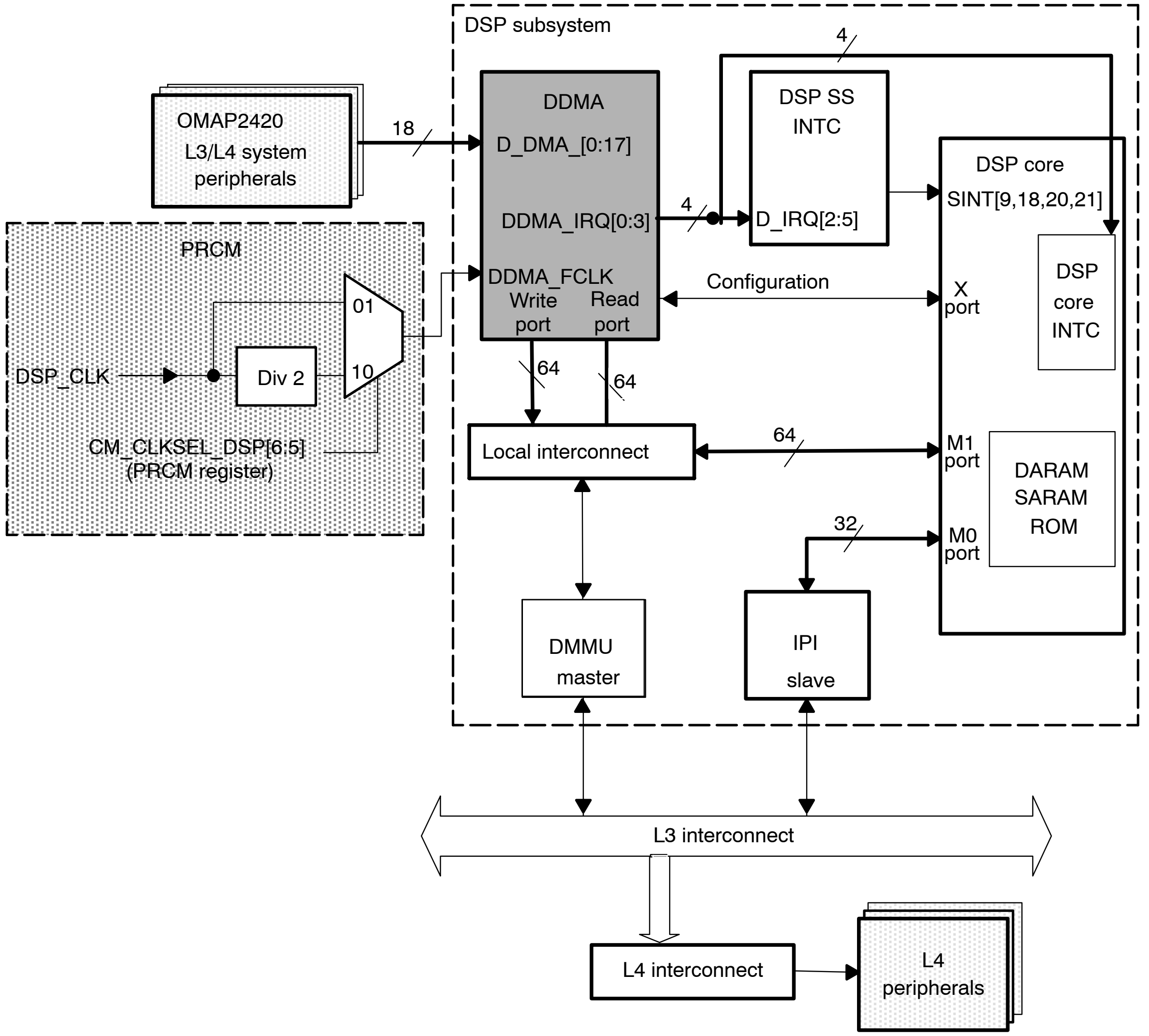
Register Name	Type	Register Width (Bits)	Offset Address	sDMA Physical Address
DMA4_CFN0 − DMA4_CFN*n*	RW	32	0x0098+ (*c**0x60)	0x4805 6098− 0x4805 6C38
DMA4_CSSA0 − DMA4_CSSA*n*	RW	32	0x009C+ (*c**0x60)	0x4805 609C− 0x4805 6C3C
DMA4_CDSA0 − DMA4_CDSA*n*	RW	32	0x00A0+ (*c**0x60)	0x4805 60A0− 0x4805 6C40
DMA4_CSEI0 − DMA4_CSEI*n*	RW	32	0x00A4+ (*c**0x60)	0x4805 60A4− 0x4805 6C44
DMA4_CSFI0 − DMA4_CSFI*n*	RW	32	0x00A8+ (*c**0x60)	0x4805 60A8− 0x4805 6C48
DMA4_CDEI0 − DMA4_CDEI*n*	RW	32	0x00AC+ (*c**0x60)	0x4805 60AC− 0x4805 6C4C
DMA4_CDFI0 − DMA4_CDFI*n*	RW	32	0x00B0+ (*c**0x60)	0x4805 60B0− 0x4805 6C50
DMA4_CSAC0 − DMA4_CSAC*n*	R	32	0x00B4+ (*c**0x60)	0x4805 60B4− 0x4805 6C54
DMA4_CDAC0 − DMA4_CDAC*n*	R	32	0x00B8+ (*c**0x60)	0x4805 60B8− 0x4805 6C58
DMA4_CCEN0 − DMA4_CCEN*n*	R	32	0x00BC+ (*c**0x60)	0x4805 60BC− 0x4805 6C5C
DMA4_CCFN0 − DMA4_CCFN*n*	R	32	0x00C0+ (*c**0x60)	0x4805 60C0− 0x4805 6C60
DMA4_COLOR0 − DMA4_COLOR*n*	RW	32	0x00C4+ (*c**0x60)	0x4805 60C4− 0x4805 6C64

Note: *c* is the logical channel numbered 0x0 through 0x1F.

10.3.3 DDMA Controller Description

Figure 10−6 highlights the DDMA controller integration in the OMAP2420.

Figure 10−6. DDMA Controller Integration



10.3.3.1 Clocking, Reset, and Power-Management Scheme

The clocking, power, and reset scheme used for the DDMA is also extensively described in Chapter 4, *DSP Subsystem.* For details, see Chapter 5, *Power, Reset, and Clock Management.*

Clocking

As shown in Figure 10−6, the DDMA runs at the DSP_CLK frequency along with the DSP core and other DSP subsystems. Another option is to run the DDMA, DSP MMU, and IPI on half that frequency. This is selected by the CM_CLKSEL_DSP register in the PRCM module.

The DDMA controller uses two clock domains.

- The DSP_CLK clock (or half-its frequency) is used for the functional clock for all internal logic and the two master read and write ports.
- The X-port clock is used for the configuration port, which also runs on the DSP_CLK clock.

Hardware Reset

The DDMA controller is part of the DSP1_RST reset domain.

Software Reset

The DDMA controller can be reset independently by software using the DMA4_OCP_SYSCONFIG register.

The DMA4_OCP_SYSCONFIG[1] bit controls the software reset. Setting this bit to 1 enables an active software reset that is functionally equivalent to a hardware reset. To determine when the reset is complete, the software can poll the DDMA register bit DMA4_SYSSTATUS[0]. This bit is set to 1 by hardware when the reset is complete.

Power Domain

The DDMA controller is part of the DSP power domain.

10.3.3.2 Hardware Requests

Interrupts to DSP Subsystem

Table 10−9 and Table 10−10 show the DDMA controller interrupts into the DSP core interrupt controller (DSP core INTC) and the DSP subsystem interrupt controller (DSP SS INTC). Table 10−11 describes DDMA request mapping.

Table 10−9. DDMA Interrupt Mapping to DSP Core Interrupt Controller

IRQ	Source	Description	Hardware Priority
SINT9	DDMA_IRQ1	DDMA interrupt request 1	13
SINT18	DDMA_IRQ0	DDMA interrupt request 0	12
SINT20	DDMA_IRQ2	DDMA interrupt request 2	21
SINT21	DDMA_IRQ3	DDMA interrupt request 3	22

Table 10−10. DDMA Interrupt Mapping to the DSP Subsystem Interrupt Controller

IRQ	Source	Description
D_IRQ_2	DDMA_IRQ0	DDMA interrupt request 0
D _IRQ_3	DDMA_IRQ1	DDMA interrupt request 1
D _IRQ_4	DDMA_IRQ2	DDMA interrupt request 2
D _IRQ_5	DDMA_IRQ3	DDMA interrupt request 3

DMA Requests to DDMA Controller

All peripherals internal to the OMAP2420 use the transition-sensitive scheme for DMA requests. Table 10−11 lists the DDMA request mapping.

Table 10−11. DDMA Request Mapping

DDMA Request Number	DDMA Request Line	Source	Description
1	D_DMA_0	Reserved	
2	D_DMA_1	Reserved	
3	D_DMA_2	EAC.AC_DMA_RD	EAC audio codec port—read request[1]
4	D_DMA_3	EAC.AC_DMA_WR	EAC audio codec port—write request[1]
5	D_DMA_4	EAC.MD_UL_DMA_RD	EAC modem/voice port—uplink read request[1]
6	D_DMA_5	EAC.MD_UL_DMA_WR	EAC modem/voice port—uplink write request[1]
7	D_DMA_6	EAC.MD_DL_DMA_RD	EAC modem/voice port—downlink read request[1]
8	D_DMA_7	EAC.MD_DL_DMA_WR	EAC modem/voice port—downlink write request[1]
9	D_DMA_8	EAC.BT_UL_DMA_RD	EAC Bluetooth port—uplink read request[1]
10	D_DMA_9	EAC.BT_UL_DMA_WR	EAC Bluetooth—uplink write request[1]
11	D_DMA_10	EAC.BT_DL_DMA_RD	EAC Bluetooth—downlink read request[1]
12	D_DMA_11	EAC.BT_DL_DMA_WR	EAC Bluetooth port—downlink write request[1]
13	D_DMA_12	MCBSP1_DMA_TX	MCBSP module 1—transmit request[1]
14	D_DMA_13	MCBSP1_DMA_RX	MCBSP module 1—receive request[1]
15	D_DMA_14	MCBSP2_DMA_TX	MCBSP module 2—transmit request[1]
16	D_DMA_15	MCBSP2_DMA_RX	MCBSP module 2—receive request[1]
17	D_DMA_16	UART3_DMA_TX	UART module 3—transmit request[1]
18	D_DMA_17	UART3_DMA_RX	UART module 3—receive request[1]
19	D_DMA_18	Reserved	
20	D_DMA_19	Reserved	
21	D_DMA_20	Reserved	
22	D_DMA_21	Reserved	
23	D_DMA_22	Reserved	

Table 10−11. DDMA Request Mapping (Continued)

DDMA Request Number	DDMA Request Line	Source	Description
24	D_DMA_23	Reserved	
25	D_DMA_24	Reserved	
26	D_DMA_25	Reserved	
27	D_DMA_26	Reserved	
28	D_DMA_27	Reserved	
29	D_DMA_28	Reserved	
30	D_DMA_29	Reserved	
31	D_DMA_30	Reserved	
32	D_DMA_31	Reserved	

1) Shared DMA request—also mapped on sDMA

10.3.4 DDMA Register Summary

Table 10−12 shows the DDMA registers and their physical addresses. For detailed descriptions of the registers, see Section 10.6, *DMA Registers*. The physical addresses listed in Table 10−12 are byte addresses; divide by 2 to get DSP half-word addresses.

It is possible to access the registers by addressing memory space or I/O space. In Table 10−12, IOMA refers to the page index to access the DSP I/O space addresses from DSP memory space. IOMA is programmed in the DSP subsystem IPI register; IPI_IOMAP. IOMA is fixed in DSP BIOS.

Table 10−12. DDMA Register Summary

Register Name	Type	Offset Address	Register Width (Bits)	DDMA Physical Address Memory Space	DDMA Physical Address I/O Space
DMA4_REVISION	R	0x0000	32	IOMA+0xC000	0x6000
DMA4_IRQSTATUS_L0	RW	0x0008	32	IOMA+0xC008	0x6008
DMA4_IRQSTATUS_L1	RW	0x000C	32	IOMA+0xC00C	0x600C
DMA4_IRQSTATUS_L2	RW	0x0010	32	IOMA+0xC010	0x6010
DMA4_IRQSTATUS_L3	RW	0x0014	32	IOMA+0xC014	0x6014
DMA4_IRQENABLE_L0	RW	0x0018	32	IOMA+0xC018	0x6018
DMA4_IRQENABLE_L1	RW	0x001C	32	IOMA+0xC01C	0x601C
DMA4_IRQENABLE_L2	RW	0x0020	32	IOMA+0xC020	0x6020
DMA4_IRQENABLE_L3	RW	0x0024	32	IOMA+0xC024	0x6024
DMA4_SYSSTATUS	R	0x0028	32	IOMA+0xC028	0x6028
DMA4_OCP_SYSCONFIG	RW	0x002C	32	IOMA+0xC02C	0x602C

Note: *c* is the logical channel numbered 0x0 through 0x17 for DDMA.

Table 10−12. DDMA Register Summary (Continued)

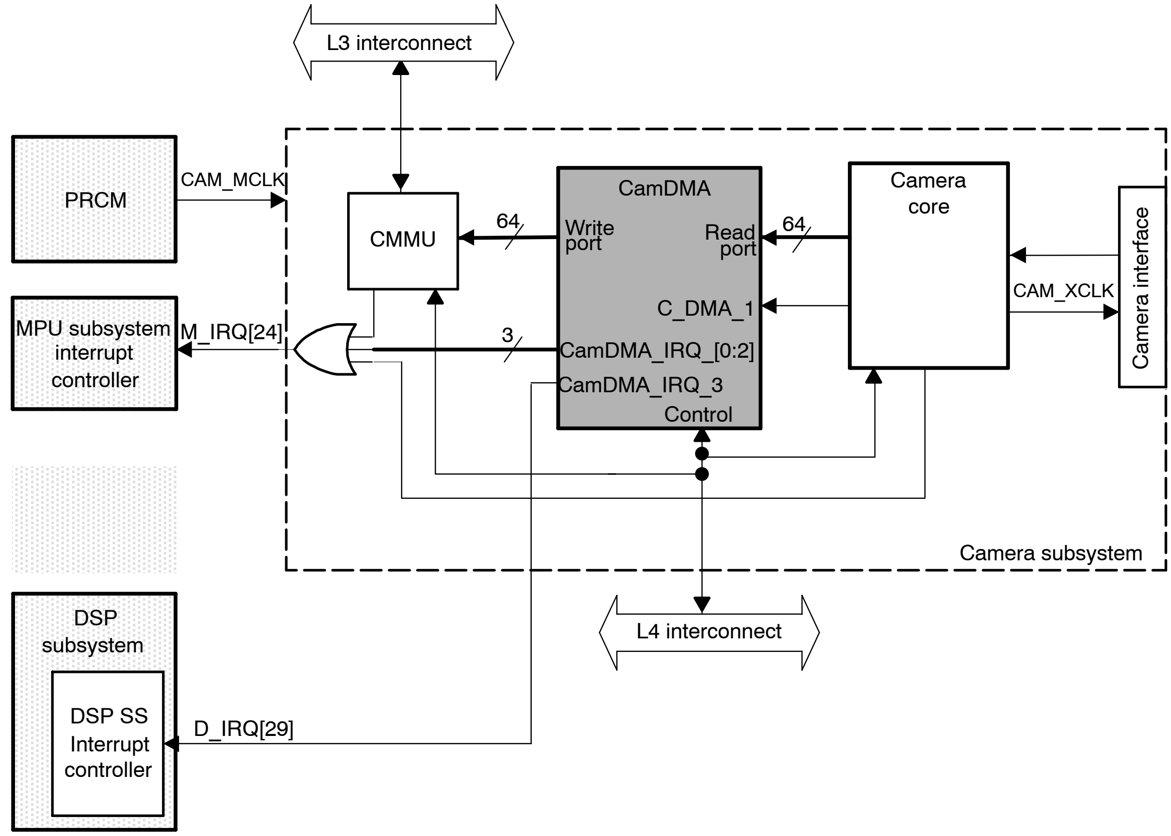
Register Name	Type	Offset Address	Register Width (Bits)	DDMA Physical Address Memory Space	DDMA Physical Address I/O Space
DMA4_CAPS_0	R	0x0064	32	IOMA+0xC064	0x6064
DMA4_CAPS_2	R	0x006C	32	IOMA+0xC06C	0x606C
DMA4_CAPS_3	R	0x0070	32	IOMA+0xC070	0x6070
DMA4_CAPS_4	R	0x0074	32	IOMA+0xC074	0x6074
DMA4_GCR	RW	0x0078	32	IOMA+0xC078	0x6078
DMA4_CCR0 – DMA4_CCR*n*	RW	0x0080+ (*c**0x60)	32	IOMA+0xC080– IOMA+0xC920	0x6080−0x6920
DMA4_CLNK_CTRL0 – DMA4_CLNK_CTRL*n*	RW	0x0084+ (*c**0x60)	32	IOMA+0xC084– IOMA+0xC924	0x6084−0x6924
DMA4_CICR0 – DMA4_CICR*n*	RW	0x0088+ (*c**0x60)	32	IOMA+0xC088– IOMA+0xC928	0x6088−0x6928
DMA4_CSR0 – DMA4_CSR*n*	RW	0x008C+ (*c**0x60)	32	IOMA+0xC08C– IOMA+0xC92C	0x608C−0x692C
DMA4_CSDP0 – DMA4_CSDP*n*	RW	0x0090+ (*c**0x60)	32	IOMA+0xC090– IOMA+0xC930	0x6090−0x6930
DMA4_CEN0 – DMA4_CEN*n*	RW	0x0094+ (*c**0x60)	32	IOMA+0xC094– IOMA+0xC934	0x6094−0x6934
DMA4_CFN0 – DMA4_CFN*n*	RW	0x0098+ (*c**0x60)	32	IOMA+0xC098– IOMA+0xC938	0x6098−0x6938
DMA4_CSSA0 – DMA4_CSSA*n*	RW	0x009C+ (*c**0x60)	32	IOMA+0xC09C– IOMA+0xC93C	0x609C−0x693C
DMA4_CDSA0 – DMA4_CDSA*n*	RW	0x00A0+ (*c**0x60)	32	IOMA+0xC0A0– IOMA+0xC940	0x60A0−0x6940
DMA4_CSEI0 – DMA4_CSEI*n*	RW	0x00A4+ (*c**0x60)	32	IOMA+0xC0A4– IOMA+0xC944	0x60A4−0x6944
DMA4_CSFI0 – DMA4_CSFI*n*	RW	0x00A8+ (*c**0x60)	32	IOMA+0xC0A8– IOMA+0xC948	0x60A8−0x6948
DMA4_CDEI0 – DMA4_CDEI*n*	RW	0x00AC+ (*c**0x60)	32	IOMA+0xC0AC– IOMA+0xC94C	0x60AC−0x694C
DMA4_CDFI0 – DMA4_CDFI*n*	RW	0x00B0+ (*c**0x60)	32	IOMA+0xC0B0– IOMA+0xC950	0x60B0−0x6950
DMA4_CSAC0 – DMA4_CSAC*n*	R	0x00B4+ (*c**0x60)	32	IOMA+0xC0B4– IOMA+0xC954	0x60B4−0x6954
DMA4_CDAC0 – DMA4_CDAC*n*	R	0x00B8+ (*c**0x60)	32	IOMA+0xC0B8– IOMA+0xC958	0x60B8−0x6958
DMA4_CCEN0 – DMA4_CCEN*n*	R	0x00BC+ (*c**0x60)	32	IOMA+0xC0BC– IOMA+0xC95C	0x60BC−0x695C
DMA4_CCFN0 – DMA4_CCFN*n*	R	0x00C0+ (*c**0x60)	32	IOMA+0xC0C0– IOMA+0xC960	0x60C0−0x6960

Note: *c* is the logical channel numbered 0x0 through 0x17 for DDMA.

10.3.5 CamDMA Controller Description

Figure 10−7 highlights CamDMA controller integration in the OMAP2420.

Figure 10−7. CamDMA Controller Integration



10.3.5.1 Clocking, Reset, and Power-Management Scheme

The clocking, reset, and power-management scheme used for the CamDMA is also described in Chapter 5, *Power, Reset, and Clock Management.*

Clocking

The camera subsystem operates from a fixed functional clock of 96 MHz provided by the PRCM module. The CamDMA uses this clock as the functional clock for all internal logic and for the two master read and write ports. The DMA is configured with the L4 interconnect.

Table 10−13 describes the CamDMA controller clocks.

Table 10−13. CamDMA Controller Clocks

	Frequency	Name	Comments
Functional clock	96 MHz	CAM_MCLK	Source is PRCM module.
Interconnect clock	Depends on PRCM register settings	CAM_ICLK	Source is PRCM module.

Hardware Reset

The CamDMA controller is part of the CORE_RST reset domain.

Software Reset

The CamDMA controller can be reset independently by software using the CamDMA_OCP_SYSCONFIG register.

The CamDMA_OCP_SYSCONFIG[1] bit controls the software reset. Setting the bit to 1 enables an active software reset functionally equivalent to a hardware reset. To determine when the reset is complete, the software can poll the DDMA register bit DMA4_SYSSTATUS[0]. Hardware resets the bit to 1 when reset is complete.

Power Domain

The CamDMA controller is part of the CORE power domain.

10.3.5.2 Hardware Requests

Interrupts to MPU Subsystem

When an interrupt is generated, the camera subsystem register, CAM_IRQSTATUS must be read to determine the source of the interrupt (the MMU interrupt, the DMA interrupt 0, the DMA interrupt 1, the DMA interrupt 2, or the camera core interrupt). If the CAM_IRQSTATUS[0], CAM_IRQSTATUS[1], or CAM_IRQSTATUS[2] bit is set to 1, the interrupt source comes from the DMA module.

The next step is to determine which logical channel generates the interrupt by reading the corresponding CamDMA_IRQSTATUS_*i* register, where the corresponding CamDMA interrupt lines 0−2 are represented. See Chapter 11, *Interrupt Controller*, for information about how to handle interrupts in the DMA module.

Table 10−14 describes the camera subsystem and CamDMA interrupt mapping to the MPU subsystem.

Table 10−14. Camera Subsystem and CamDMA Interrupt Mapping to MPU Subsystem

IRQ	Name	Description
M_IRQ_24	CAM_MPU_IRQ	Camera interface interrupt request from the camera MMU or CamDMA interrupt lines 0−2

DMA Requests to CamDMA Controller

The camera core can generate two different DMA requests, CC_DMAREQN[0] and CC_DMAREQN[1], to the CamDMA. However, both requests share the same DMA request line, C_DMA_1. By design, the two DMA requests cannot be active at the same time. The DMA request line/number is 1 for CamDMA; therefore, always set the SYNCHRO_CONTROL bit field to 1 in the CamDMA_CCR register.

The CC_DMAREQN[0] DMA request is generated when the write camera core FIFO counter reaches the trigger level and is active until the DMA controller reads the number of 32-bit words equal to the FIFO threshold. The next CC_DMAREQN[0] DMA request occurs when the number of remaining 32-bit words is above the threshold and the DMA completes the previous transfer. Several CC_DMAREQ[0] DMA requests are generated during a typical frame transfer.

The other kind of DMA request, CC_DMAREQN[1], is used when the image frame to transfer is not a multiple of the packet size to transfer each DMA request. This request is used to drain the camera FIFO before the next frame comes. CC_DMAREQN[1] is generated when the frame is ended and the remaining data are still in the camera core FIFO.

The deassertion of the DMA request occurs when the camera FIFO is emptied by the CamDMA. A maximum of one CamDMA_REQ2 request can be generated per frame transfer.

Note:

The FIFO threshold can be different from the packet size.

The maximum transfer size, regardless of the packet size, is always:

> Block_size = Number_of_Frame_in_Block * Number_of_Element_in_Frame * Element_Size

For example, if the image frame is 176*144 pixels of 1 byte (meaning 25,344 bytes or 6,336 32-bits words) and the threshold is 10 (32-bit words), the camera core module generates 633 CC_DMAREQN[0] and one CC_DMAREQN[1] for the remaining six 32-bit words. The logical channel is deactivated when the entire block is transferred. This can be configured using one logical channel.

Configure it to be synchronized on DMA request line 1 by setting the DMA4_CCR register field SYNCHRO_CONTROL equal to 1 for every logical channel used.

The CamDMA always uses synchronized transfers at the source and nonsynchronized transfers at the destination.

10.3.6 CamDMA Register Summary

Table 10−15 shows the CamDMA registers and their physical addresses. For detailed descriptions of the registers, see Section 10.6, *DMA Registers*. All registers are 32 bits wide.

Table 10−15. CamDMA Register Summary

Register Name	Type	Register Width (Bits)	Offset Address	CamDMA Physical Address
DMA_REVISION	R	32	0x0000	0x4805 2800
DMA_IRQSTATUS_L0	RW	32	0x0008	0x4805 2808

Note: *c* is the logical channel numbered 0x0 through 0x3.

Table 10−15. CamDMA Register Summary (Continued)

Register Name	Type	Register Width (Bits)	Offset Address	CamDMA Physical Address
DMA_IRQSTATUS_L1	RW	32	0x000C	0x4805 280C
DMA_IRQSTATUS_L2	RW	32	0x0010	0x4805 2810
DMA_IRQSTATUS_L3	RW	32	0x0014	0x4805 2814
DMA_IRQENABLE_L0	RW	32	0x0018	0x4805 2818
DMA_IRQENABLE_L1	RW	32	0x001C	0x4805 281C
DMA_IRQENABLE_L2	RW	32	0x0020	0x4805 2820
DMA_IRQENABLE_L3	RW	32	0x0024	0x4805 2824
DMA_SYSSTATUS	R	32	0x0028	0x4805 2828
DMA_OCP_SYSCONFIG	RW	32	0x002C	0x4805 282C
DMA4_CAPS_0	R	32	0x0064	0x4805 2864
DMA4_CAPS_2	R	32	0x006C	0x4805 286C
DMA4_CAPS_3	R	32	0x0070	0x4805 2870
DMA4_CAPS_4	R	32	0x0074	0x4805 2874
DMA_GCR	RW	32	0x0078	0x4805 2878
DMA4_CCR0 – DMA4_CCR*n*	RW	32	0x0080+ (*c**0x60)	0x4805 2880– 0x4805 29A0
DMA4_CLNK_CTRL0 – DMA4_CLNK_CTRL*n*	RW	32	0x0084+ (*c**0x60)	0x4805 2884– 0x4805 29A4
DMA4_CICR0 – DMA4_CICR*n*	RW	32	0x0088+ (*c**0x60)	0x4805 2888– 0x4805 29A8
DMA4_CSR0 – DMA4_CSR*n*	RW	32	0x008C+ (*c**0x60)	0x4805 288C– 0x4805 29AC
DMA4_CSDP0 – DMA4_CSDP*n*	RW	32	0x0090+ (*c**0x60)	0x4805 2890– 0x4805 29B0
DMA4_CEN0 – DMA4_CEN*n*	RW	32	0x0094+ (*c**0x60)	0x4805 2894– 0x4805 29B4
DMA4_CFN0 – DMA4_CFN*n*	RW	32	0x0098+ (*c**0x60)	0x4805 2898– 0x4805 29B8
DMA4_CSSA0 – DMA4_CSSA*n*	RW	32	0x009C+ (*c**0x60)	0x4805 289C– 0x4805 29BC
DMA4_CDSA0 – DMA4_CDSA*n*	RW	32	0x00A0+ (*c**0x60)	0x4805 28A0– 0x4805 29C0
DMA4_CSEI0 – DMA4_CSEI*n*	RW	32	0x00A4+ (*c**0x60)	0x4805 28A4– 0x4805 29C4
DMA4_CSFI0 – DMA4_CSFI*n*	RW	32	0x00A8+ (*c**0x60)	0x4805 28A8– 0x4805 29C8
DMA4_CDEI0 – DMA4_CDEI*n*	RW	32	0x00AC+ (*c**0x60)	0x4805 28AC– 0x4805 29CC

Note: *c* is the logical channel numbered 0x0 through 0x3.

Table 10−15. CamDMA Register Summary (Continued)

Register Name	Type	Register Width (Bits)	Offset Address	CamDMA Physical Address
DMA4_CDFI0 − DMA4_CDFI*n*	RW	32	0x00B0+ (*c**0x60)	0x4805 28B0− 0x4805 29D0
DMA4_CSAC0 − DMA4_CSAC*n*	R	32	0x00B4+ (*c**0x60)	0x4805 28B4− 0x4805 29D4
DMA4_CDAC0 − DMA4_CDAC*n*	R	32	0x00B8+ (*c**0x60)	0x4805 28B8− 0x4805 29D8
DMA4_CCEN0 − DMA4_CCEN*n*	R	32	0x00BC+ (*c**0x60)	0x4805 28BC− 0x4805 29DC
DMA4_CCFN0 − DMA4_CCFN*n*	R	32	0x00C0+ (*c**0x60)	0x4805 28C0− 0x4805 29E0

Note: *c* is the logical channel numbered 0x0 through 0x3.

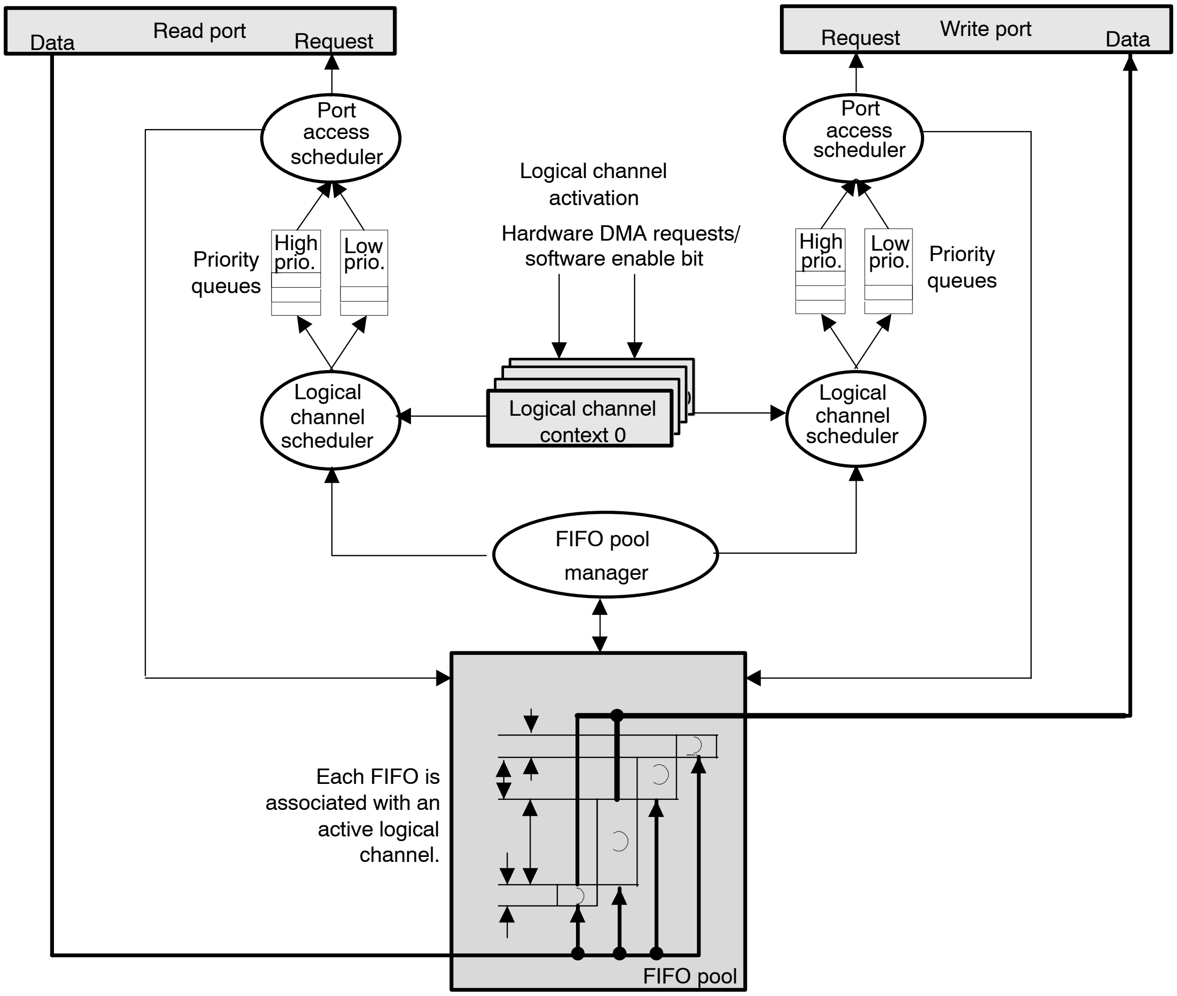
10.4 DMA Functional Description

The DMA module provides high-performance data transfers between memories and peripheral devices with low processor use. A DMA transfer is programmed using a logical DMA channel, which allows the transfer to be optimally tailored to the requirements of the application.

10.4.1 Logical Channel Transfer Overview

As shown in Figure 10−8, the DMA module has one read port and one write port operating independently of each other.

Figure 10−8. DMA Controller Top-Level Block Diagram


Read port
Data
Request
Write port
Request
Data
Port access scheduler
Port access scheduler
Logical channel activation
Hardware DMA requests/ software enable bit
Priority queues
High prio.
Low prio.
High prio.
Low prio.
Priority queues
Logical channel scheduler
Logical channel context 0
Logical channel scheduler
FIFO pool manager
Each FIFO is associated with an active logical channel.
FIFO pool

Buffering is provided between the read and write ports through a FIFO queue memory pool, which is shared dynamically between the active logical channels.

- ❑ Logical channel synchronization

  A logical channel is described as hardware-synchronized if the DMA transfers are triggered by DMA requests from a hardware device. Alternately, a logical channel is described as nonsynchronized if the DMA transfer is triggered by software.

- Logical channel activation

  A logical channel becomes active in the following situations:

  - For hardware-synchronized transfers, when the logical channel is enabled and the hardware DMA request line is asserted
  - For software-triggered (nonsynchronized) transfers, when the software enables the logical channel

- Logical channel transfer composition

  A DMA transfer is automatically divided into a number of transactions. Depending on the logical channel context configured, the 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 use of the read and write ports. The scheduling of logical channel transfers is similar for the 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 channel with the lower number is queued first. This mechanism provides a first-come-first-served scheduling scheme between the concurrently active logical channels.

  In addition, the read and write ports have both a high-priority and a low-priority queue. The priority bit in the logical channel DMA4_CCR register determines whether a logical channel is queued on the high-priority or the low-priority queue. The relative weighting of the scheduling of the high-priority queue to the low-priority queue is programmable from 1:1 to 16:1 using the DMA global channel register (DMA4_GCR).

- Read/write port access scheduling policy

  When 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-priority or the low-priority queue. When the DMA access is complete but the full DMA transfer is not, the logical channel is returned to the tail of the queue. Because port access scheduling is per-access, a logical channel can be queued this way several times during a single transfer.

The DMA 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, two conditions must be satisfied:

- At least one channel is requesting.
- At least one free thread ID is available.

For 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 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 entire channel transfer.

If only one channel is active, one thread ID is allocated during the channel transfer; accesses can occur with up to four consecutive bursts (4x32) without rescheduling the channel at the end of each burst transfer. If nonburst alignment occurs at the beginning of the transfer, the channel is rescheduled for each smaller access until the burst is aligned. If the end of the transfer is not burst-aligned, the channel is rescheduled for each one of the remaining smaller accesses.

❑ 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 10.4.10, *Interrupt Generation*).

10.4.2 FIFO Queue Memory Pool

A FIFO queue memory pool provides buffering between the read and write ports. This space is allocated dynamically by the hardware to a number of FIFO queues, each queue 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 down 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 using the 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 causes the channel to exceed this limit. This means that the effective number of entries used by a particular logical channel is limited to:

configured maximum entries per channel + channel maximum configured burst size (in words) − 1

See Table 10−3 for details about the size of the FIFO queue memory pool for the DMA module instance.

10.4.3 Addressing Modes

A DMA transfer block consists of a number of frames. Each frame consists of a number of elements and each element can have a size of 8, 16, or 32 bits. Therefore:

transfer block size = number of frames x number of elements per frame x element size

The number of frames, number of elements per frame, and element size 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 four addressing modes:

- ❑ Constant: the address remains the same for consecutive element accesses.
- ❑ Post-increment: the address increases by the element size (even across consecutive frames).
- ❑ Single-index: the address increases by the element size plus the element index value minus one (even across consecutive frames).
- ❑ Double-index: the address increases by the element size plus the element index value minus one within a frame. When a full frame is transferred, the address increases by the element size plus the frame index value minus one.

The element size and element and frame index values are expressed in bytes. The element and frame index values can be positive or negative.

When calculating the element and frame index values, remember that when an element is accessed, the logical channel address pointer equals the address of the last byte (highest address) of the accessed element. When the correct value for the element or frame index is added to the logical channel address pointer, the result should be the address of the first byte (lowest address) of the next element to be accessed. The element and frame index values must be configured so that the address of each element in the transfer is aligned on an element size boundary.

Consequently, single-index addressing mode with element index = 1 or double-index addressing mode with element index = 1 and frame index = 1 is equivalent to post-increment addressing.

Note:

The source and destination start addresses must also be aligned on an element size boundary.

If the address of an element to be accessed is not aligned on an element size boundary, the transfer is stopped and an address error interrupt occurs, if enabled (see Section 10.4.10, *Interrupt Generation*).

The number of frames in a block is configured using the DMA4_CFN register; the number of elements per frame is configured using the DMA4_CEN register; the element size is configured using the DMA4_CSDP register; the source and destination start addresses are configured using the DMA4_CSSA register and the DMA4_CDSA register; the source and destination addressing modes are configured using the DMA4_CCR register; and the source element index, source frame index, destination element index, and destination frame index are configured using the DMA4_CSEI, DMA4_CSFI, DMA4_CDEI, and DMA4_CDFI registers, respectively.

The addressing profiles are expressed in Equation 10−1 through Equation 10−6.

Equation 10−1. Constant Addressing

$$A(n + 1) = A(n)$$

Equation 10−2. Post-Increment Addressing

$$A(n + 1) = A(n) + ES$$

Equation 10−3. Single-Indexed Addressing

$$A(n + 1) = A(n) + ES + (EI - 1)$$

Equation 10−4. Double-Indexed Addressing

When not at the end of a frame or a transfer (that is, as long as element counter $\neq 0$):

$$A(n + 1) = A(n) + ES + (EI - 1)$$

When at the end of a frame but not at the end of the transfer (that is, as long as element counter $= 0$ and frame counter $\neq 0$):

$$A(n + 1) = A(n) + ES + (FI - 1)$$

Calculate the element and frame index as follows:

Equation 10−5. Element Index

$$EI = ((\text{Stride } EI - 1) * ES) + 1$$

Equation 10−6. Frame Index

$$FI = ((\text{Stride } FI - 1) * ES) + 1$$

where:

A(n): Byte address of the element *n* within the transfer

ES: Element size in bytes, $ES \in \{1, 2, 4\}$

EI: Element index in bytes, specified in a configuration register, $-32768 \leq EI \leq 32767$

Stride EI: Number of elements difference between the start of 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. The number is decreased by 1 for each element transferred. The initial value is configured in the register DMA channel element number (DMA4_CEN).

FI: Frame index in bytes, specified in a configuration register, $-2147483648 \leq FI \leq 2147483647$

Stride FI: The difference in the number of elements between the start of the last element of the current frame and the beginning of first element of the next frame

Frame counter: A counter that is (re)initiated with the number of frames per transfer. Decreased by 1 for each frame transferred. The initial value is configured in the register DMA channel frame number (DMA4_CFN).

Figure 10−9 shows how a stride EI and FI are defined. When handling with 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 90-degree clockwise image rotation example shown in Figure 10−13. The double-index addressing example shown in Figure 10−9 has ES = 4, EN = 2, EI = 5, FI = 5, and FN = 2.

Figure 10−9 through Figure 10−12 show examples of addressing mode configurations.

Figure 10−9. Example Showing Double-Index Addressing, Elements, Frames, and Strides

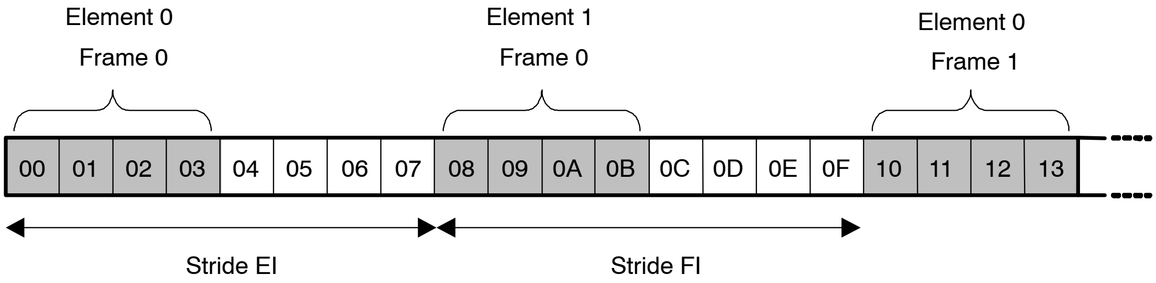


Table 10−16 lists the parameter values for the addressing mode examples in Figure 10−10 through Figure 10−12.

Table 10−16. 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
Element size	4 (32-bit)	4 (32-bit)	2 (16-bit)
Number of elements per frame	3	2	2
Element index	1	5	1
Number of frames	1	2	2
Frame index	n/a	5	−9

Figure 10−10. Addressing Mode Example (a)

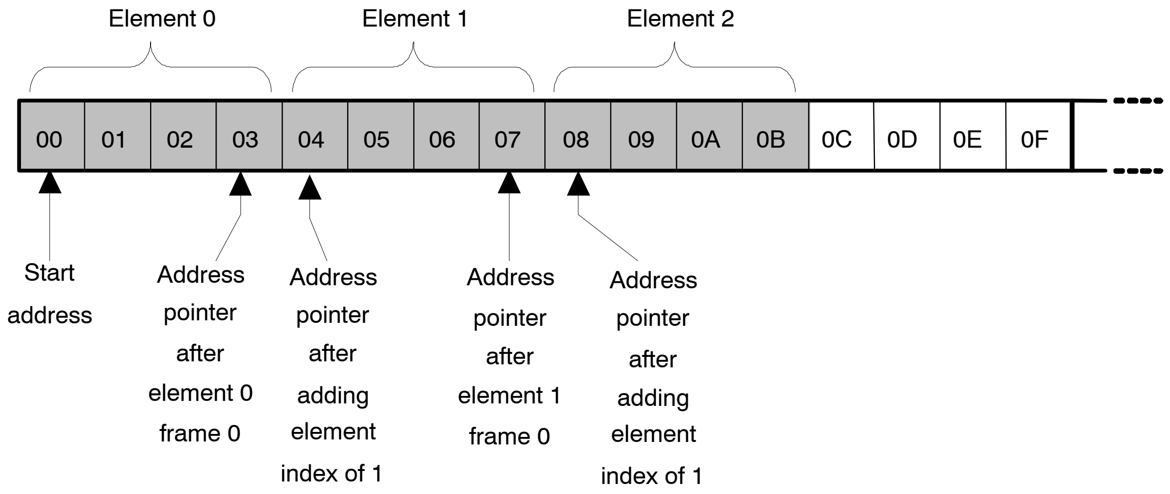

Element 0
Element 1
Element 2
00
01
02
03
04
05
06
07
08
09
0A
0B
0C
0D
0E
0F
Start address
Address pointer after element 0 frame 0
Address pointer after adding element index of 1
Address pointer after element 1 frame 0
Address pointer after adding element index of 1

Figure 10−11.Addressing Mode Example (b)

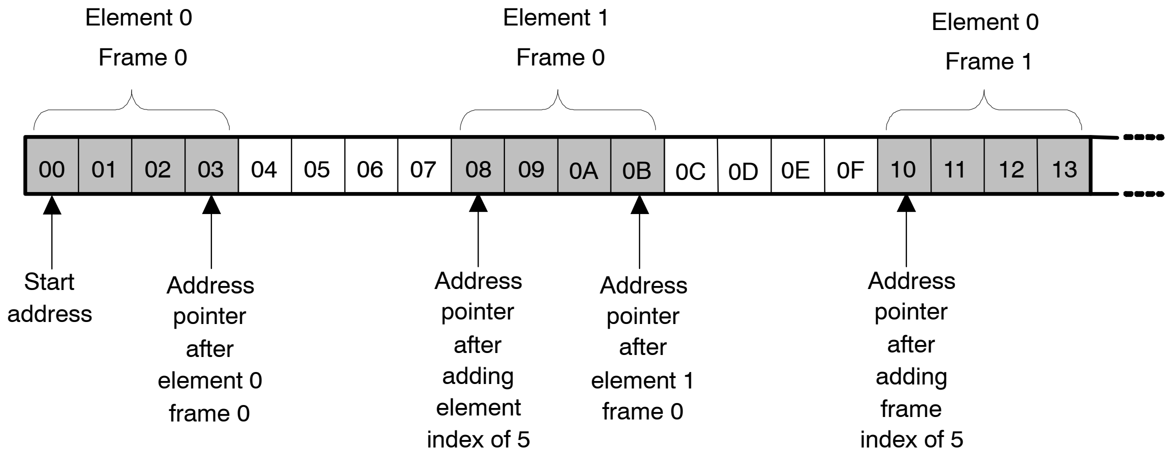

Element 0 Frame 0
Element 1 Frame 0
Element 0 Frame 1
00
01
02
03
04
05
06
07
08
09
0A
0B
0C
0D
0E
0F
10
11
12
13
Start address
Address pointer after element 0 frame 0
Address pointer after adding element index of 5
Address pointer after element 1 frame 0
Address pointer after adding frame index of 5

Figure 10−12. Addressing Mode Example (c)

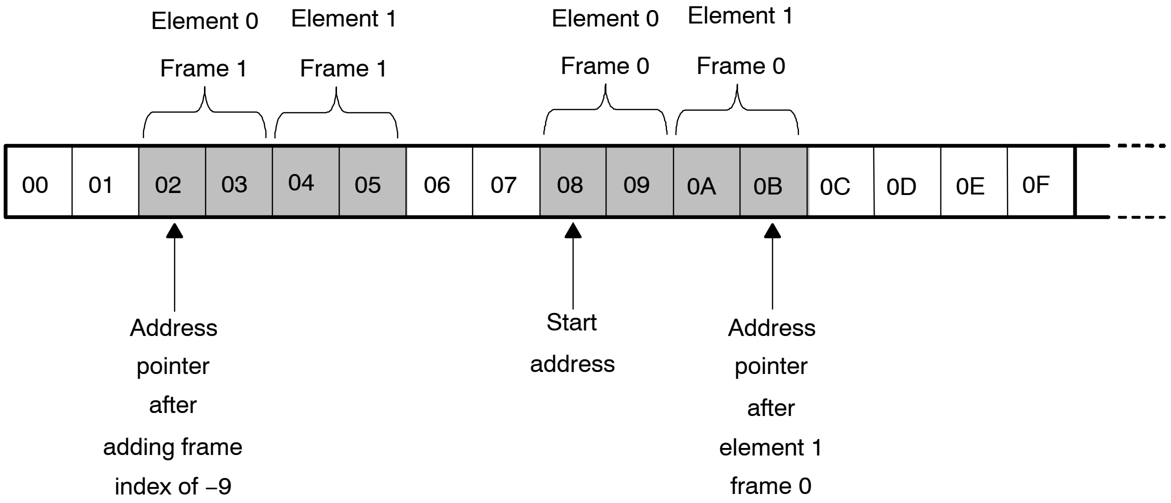


Table 10−17 and Figure 10−13 show the configuration to perform a 90-degree clockwise image rotation of a 320 x 240 pixel, 16-bit image. The element index, frame size, and frame index values are configured so that the image is rotated line by line starting at the left end of the top line.

Note:

The frame index 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 10−5 and Equation 10−6 are used to calculate destinations FI and EI. The example assumes that the image lines are stored at consecutive addresses in memory, meaning that both EI and FI on source side is 1.

See Section 10.5.5, *90-Degree Clockwise Image Rotation*, for a description of how to program this example.

Observe that:

- One pixel = one element
- One line = one DMA frame
- Pixel size = element size (ES)

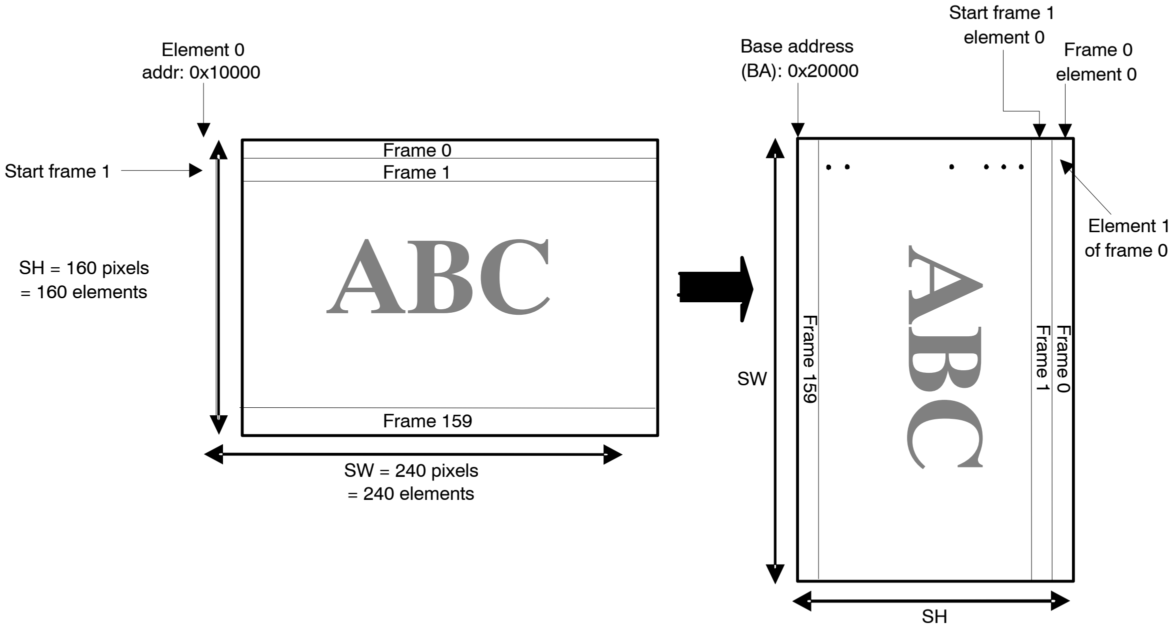
Table 10−17. Example Parameter Values for a 90-Degree Clockwise Image Rotation

Parameter	Source Value	Destination Value
Bits per pixel	16	16
Element size (ES)	2	2
Image width	240 elements (SW)[1]	SH elements
Image height	160 elements (SH)[2]	SW elements
Stride elements (stride EI)	1 element	SH elements
Stride frames (stride FI)	1 element	−((SW−1)*SH+1) = −38241 elements
Start address	0x100000	0x200000 + (SH – 1) x ES = 0x20013E
Number of elements per frame (EN)	SW	SW
Element index (EI)	((Stride EI – 1) * ES) + 1 = 1	((Stride EI – 1) * ES) + 1 = 319
Number of frames (FN)	SH	SH
Frame index (FI)	((Stride FI – 1) * ES) + 1 = 1	((Stride FI – 1) * ES) + 1 = −76481

Notes: 1) SW = size width

Notes: 2) SH = size height

Figure 10–13. Example of a 90-Degree Clockwise Image Rotation



Notes: 1) SH = size height

2) SW = size width

10.4.4 Packed Accesses

When the logical channel element size is less than the DMA module read/write port size, and the addressing profile supports it (that is, the post-increment mode or single- or double-index mode with element index = 1), the number of elements to transfer in each read/write port access can be maximized by specifying that the source or destination is packed using the channel DMA4_CSDP register. Thus:

- For a read/write port size of 32 bits, the source or destination can be configured as packed for transfer element sizes 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 element sizes 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 the length of a transfer, the first or last packed access might be only partially filled. This is indicated to the source or destination using byte-enable signals.

Note:

When constant addressing mode is specified, only nonpacked accesses are used and specifying the accesses as packed has no effect.

10.4.5 Burst Transactions

When the source or destination and addressing profile support transfer performance, it can be improved 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 using the DMA4_CSDP register, with the read burst size programmable independent 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 must be configured to align with the burst size. However, 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 constant addressing mode, the source or destination must be specified as packed, for burst transactions to occur.

Note:

When the FIFO is full (that is, the maximum FIFO depth allocated by channel is reached), the DMA does not send a read request to the memory or peripheral.

Before issuing a new read request, the DMA waits until the FIFO has enough space (equal to the channel source burst size).

10.4.6 Endianism Conversion

The source and destination are each specified as either little-endian or big-endian using the DMA4_CSDP register for the particular logical channel. If the endianism of the source and destination differ and the logical channel element size is less than the DMA 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 element size that equals 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, not to perform a conversion) using the logical DMA channel DMA4_CSDP register.

10.4.7 Transfer Synchronization

A logical channel can be programmed for either software-triggered or hardware-synchronized transfers.

A transfer is software-triggered if the logical channel is set up and started by software and continues without hardware synchronization. A software-triggered transfer is specified by setting all of the channel DMA register bits DMA4_CCR[4:0] to 0. The transfer starts as soon as the DMA register bit DMA4_CCR[7] is set (that is, enters the scheduling process).

A transfer is hardware-synchronized if the logical channel activation is driven by requests from either the source or destination. A hardware-synchronized transfer is specified by configuring the DMA request line number in the channel DMA4_CCR 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 tables for each instance. See Table 10−7 for sDMA mapping and Table 10−11 for DDMA mapping.

Specify the DMA request number in the channel DMA register bits DMA4_CCR[4:0] and DMA4_CCR[20:19]. After the DMA register bit DMA4_CCR[7] is set, the logical channel becomes enabled but not activated (that is, does not enter the scheduling process) until the first DMA request is received.

Note:

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 hardware synchronization, the amount of data transferred for each assertion of the DMA request line is configured using the frame synchronization (FS) and block synchronization (BS) bits in the logical channel DMA4_CCR register, DMA register bits DMA4_CCR[5] and DMA4_CCR[18], respectively. The amount of data can be:

- A single transfer element
- A frame consisting of a number of elements
- A block consisting of a number of frames
- A packet consisting of a number of elements

Packets allow the size of each part of the full DMA transfer to be configured independently of the organization of the data to be transferred (typically described as a number of frames). This can be useful when the source or the 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.

The CamDMA is an exception to this rule. On the CamDMA source side, no address bus is implemented because it is always read from a constant address (the camera core FIFO). However, because the camera core FIFO and the sDMA use a 64-bit data bus between them, the most effective transfer mode is a 64-bit burst mode (that is, 8 x 64 bits per time). To support 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 the constant address mode; thus, it must be configured for the post-increment address mode.

The constant address mode can be used on the CamDMA source port but must be specified as nonpacked.

The packet size is configured based on the source/destination synchronization select bit in the DMA4_CCR register, using either the channel DMA4_CSFI register (source synchronized) or the DMA4_CDFI 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.

Regardless of the packet size, the maximum transfer size is always:

Block_size = Number_of_Frame_in_Block * Number_of_Element_in_Frame * Element_Size

☐ Synchronized at the source

The DMA module optimizes the transfer 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 10.4.10, *Interrupt Generation*).

For a source-synchronized transfer, buffering can be enabled or disabled by setting the BUFFERING_DISABLE bit field of the DMA4_CCR register.

For a packet source synchronization with buffering disable on and the packed/burst across the packets boundary, the last packed/burst transaction is split in optimized smaller accesses to complete the packet transfer size.

☐ Synchronized at the destination

The performance of a hardware-synchronized transfer can be improved by using the prefetch mode, enabled using the channel DMA register bit DMA4_CCR[23]. Data is prefetched on the read port side before the DMA request is 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 10.4.2, *FIFO Queue Memory Pool*).

Buffering disable is not allowed for a destination synchronized transfer.

Note:

- Behavior is undefined if prefetch is enabled when a transfer is synchronized with 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 DMA4_CSDP register = 2).
  However, in packet synchronization mode, the last transaction of each packet in the transfer is WNP, only if the buffering disable is on (even if the write mode is specified as WLNP).
- Whether buffering is enabled or disabled, the packet interrupt is not generated in the packet source synchronized mode.

CAUTION

The BUFFERING_DISABLE bit field of the DMA4_CCR register must be filled with the allowed value, as specified in Table 10–18.

Table 10–18. Allowed Values for the BUFFERING_DISABLE Bit Field

	BUFFERING_DISABLE **(0: buffering enable, 1: buffering disable)**	
Destination synchronized	0	Allowed
	1	Not allowed
Source synchronized	0	Allowed
	1	Allowed

- Synchronization error (event drop)

  If one DMA request is serviced, a second DMA request is pending (not yet scheduled), and a third DMA request comes in, the DMA4 generates an event drop interrupt and ignores this last request and any new incoming DMA request until the current transfer completes and the pending DMA request is scheduled.

10.4.8 Chained Logical Channel Transfers

Chaining multiple logical channels permits the execution (without repeated software intervention) of transfers consisting of multiple parts, thus resulting in better performance compared with 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, permitting 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 using each channel DMA4_CLNK_CTRL register. When the transfer for the first channel completes, the next channel in the chain is activated. The number of channels in the chain that are configured for hardware-synchronized transfers is flexible (although typically is all, none, or just the first one). To specify that any or all of the channels in a chain are software-triggered or nonsynchronized, the DMA request line number must be set to 0.

The last channel in a chain can be chained to the first channel, creating a continuously looping chain. The continuously looping transfer can be stopped on the fly at a specific channel by disabling the ENABLE_LNK bit in the channel DMA4_CLNK_CTRL register. The looping transfer stops after the specified channel transfer completes.

For more information about the programming model, see Section 10.5, *DMA Programming Model*.

10.4.9 Reprogramming an Active Channel

A currently active logical DMA channel can be disabled using a bit in the channel DMA4_CCR register. When an ongoing transaction completes, the read and write active bits in the DMA4_CCR register are reset and the channel can be reprogrammed for a new transfer, if required.

10.4.10 Interrupt Generation

The DMA module has four interrupt request output lines, IRQ0 to IRQ3. For information about IRQ routing for the sDMA, DDMA, and CamDMA, see Section 10.3.1, *sDMA Controller Description*, Section 10.3.3, *DDMA Controller Description*, and Section 10.3.5, *CamDMA Controller Description*.

One or more logical channels can be programmed to generate an interrupt request on any of these lines when any of the maskable DMA events in Table 10−19 occurs.

Table 10−19. Logical DMA Channel Events

Event	Description
End of packet	A packet transfer completes.
End of block	A block transfer completes.

Table 10−19. Logical DMA Channel Events (Continued)

Event	Description
End of frame	A frame transfer completes.
Half of frame	Half of the current frame completes.
Start of last frame	The first element of the last frame completes.
Transaction error	A system error occurred during an attempted DMA transaction.
Address error	An attempt was made to perform DMA access to an address not aligned on an element size boundary.
Synchronization error	A new DMA request arrived before the completion of the transfer because of the previous DMA request (event drop).

The logical DMA channels that generate an interrupt on a particular IRQ output are specified using the DMA4_IRQENABLE_*i* register (where *i* is the IRQ number: 0, 1, 2, or 3). Events that generate an interrupt for a particular channel can be configured using the channel DMA4_CICR register.

When an interrupt is detected, the logical DMA channel generating the event can first be identified by reading the DMA4_IRQSTATUS_*i* register. The event causing the interrupt can then be identified by reading the interrupt status using the relevant DMA channel DMA4_CSR register.

10.4.11 Posted and Nonposted Writes

A logical channel can be configured in its DMA register bits DMA4_CSDP[17:16] to use one of three write-access handshake modes for the destination:

❑ Nonposted write: Each write must complete before a transfer can continue or complete.

❑ Posted write: The transfer continues without waiting for each write to complete (can improve performance with slow devices).

❑ Posted with final write nonposted: The transfer continues without waiting for each write to complete, but the final write of each frame completes before the frame transfer can complete.

10.4.12 Reset

After a software or hardware reset, all fields in the logical channel registers have undefined values except for 5 bits in the DMA4_CCR and DMA4_CICR registers. Therefore, when programming a channel for the first time, it is necessary to configure the remaining fields in all channel registers before enabling the channel.

After a reset, the global registers take their specified reset values.

10.4.13 Power Management

The DMA module provides two methods to reduce power consumption:

- Interconnect clock auto-idle
- Automatic standby mode

10.4.13.1 Auto-Idle

The interconnect clock auto-idle power-saving mode is enabled or disabled in the DMA register bit DMA_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, and this reduces power consumption. When there is new activity on the interconnect interface, the interconnect clock is restarted without a latency penalty. After reset, this mode is disabled by default. It is recommended that this mode be enabled, to reduce power consumption.

10.4.13.2 System Power Management

As part of the system-wide power-management scheme, the module can enter standby state at the request of the PRCM module (for more information, see Chapter 5, *Power, Reset, and Clock Management*).

The module can be configured in the DMA register bits DMA_OCP_SYSCONFIG[13:12] to be in one of the following standby modes:

- No-standby mode: The module never goes to standby state.
- Force-standby mode: The module goes to standby state only when all DMA channels are disabled.
- Smart-standby mode: The module enters standby state when either of the following conditions occurs:
  - 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.

10.5 DMA Programming Model

10.5.1 Setup Configuration

Because most fields are undefined after a software or hardware reset, all fields in the logical channel registers must be programmed to default values for any channels used.

Before programming DMA transfers, the priority arbitration rate and the maximum FIFO depth must be configured using the DMA4_GCR register, and any required interrupts must be enabled using the DMA4_IRQENABLE_L0 – DMA4_IRQENABLE_L3 registers and the logical channel DMA4_CICR registers.

Software must clear the DMA4_CSR register and the IRQSTATUS bit for the different interrupt lines before enabling the channel.

10.5.2 Software-Triggered (Nonsynchronized) Transfer

A software-triggered DMA transfer can be programmed as follows:

1) Configure the transfer parameters in the logical DMA channel registers:
   - DMA4_CSDP
     - Transfer element size (8-bit, 16-bit, or 32-bit): DMA register bits DMA4_CSDP[1:0]
     - Read and write port access types (single/burst): DMA register bits DMA4_CSDP[8:7] and DMA4_CSDP[15:14]
     - Source and destination endianism: DMA register bits DMA4_CSDP[21] and DMA4_CSDP[19]
     - Write mode (posted or nonposted): DMA register bits DMA4_CSDP[17:16]
     - Source or destination packed or nonpacked (if the element size is less than the read/write port size): DMA register bits DMA4_CSDP[6] and DMA4_CSDP[13]
   - DMA4_CEN: Number of elements per frame
   - DMA4_CFN: Number of frames per transfer block
   - DMA4_CSSA and DMA4_CDSA: Source and destination start address (aligned with transfer element size)
   - DMA4_CCR
     - Read and write port addressing modes: DMA register bits DMA4_CCR[13:12] and DMA4_CCR[15:14]
     - Low or high priority: DMA register bit DMA4_CCR[6]
     - DMA request number (set to 0 for software-triggered transfer): DMA register bits DMA4_CCR[4:0] = 0 and DMA4_CCR[20:19] = 0

- DMA4_CSEI, DMA4_CSFI, DMA4_CDEI, and DMA4_CDFI: Source and destination element and frame indexes (depending on addressing mode)

2) Start the transfer using the enable bit in the channel DMA4_CCR register: DMA register bit DMA4_CCR[7].

10.5.3 Hardware-Synchronized Transfer

A hardware-synchronized DMA transfer can be programmed as follows:

1) Configure the transfer parameters in the logical DMA channel registers:
   - DMA4_CSDP
     - Transfer element size (8-bit, 16-bit, or 32-bit): DMA register bits DMA4_CSDP[1:0]
     - Read and write port access types (single/burst): DMA register bits DMA4_CSDP[8:7] and DMA4_CSDP[15:14]
     - Source and destination endianism: DMA register bits DMA4_CSDP[21] and DMA4_CSDP[19]
     - Write mode (posted or nonposted): DMA register bits DMA4_CSDP[17:16]
     - Source or destination packed or nonpacked (if the element size is less than the read/write port size): DMA register bits DMA4_CSDP[6] and DMA4_CSDP[13]
   - DMA4_CEN: Number of elements per frame
   - DMA4_CFN: Number of frames per transfer block
   - DMA4_CSSA and DMA4_CDSA: Source and destination start address (aligned with transfer element size)
   - DMA4_CCR
     - Read and write port addressing modes (one of these must be constant addressing mode for synchronization to a packet, depending on whether the source or destination generates the DMA requests): DMA register bits DMA4_CCR[13:12] and DMA4_CCR[15:14]
     - Prefetch enabled or disabled: DMA register bit DMA4_CCR[23]
     - Low or high priority: DMA register bit DMA4_CCR[6]
     - DMA request number (corresponding to the incoming DMA request line with which the transfer is synchronized): DMA register bits DMA4_CCR[4:0] and DMA4_CCR[20:19]
     - Amount of data to transfer per DMA request (using frame and block synchronization bits): DMA register bits DMA4_CCR[5] and DMA4_CCR[18]
     - Whether read from source or write to destination is triggered by the DMA request: DMA register bit DMA4_CCR[24]
     - Buffering enabled or disabled: DMA register bit DMA4_CCR[25]

- DMA4_CSEI, DMA4_CSFI, DMA4_CDEI, and DMA4_CDFI: Source and destination element and frame indexes (depending on the addressing mode)
- DMA4_CSFI or DMA4_CDFI: Number of elements per packet (only for synchronization to a packet, and only one of these registers, depending on whether the source or destination sends the DMA requests)
- Enable the transfer using the enable bit in the channel DMA4_CCR register: DMA register bit DMA4_CCR[7].

2) The first transfer starts when the DMA request line is asserted.

10.5.4 Chained Transfer

Program a chained DMA transfer as follows:

1) Configure the transfer parameters for each logical DMA channel in the chain as in Step 1 for either software-triggered (nonsynchronized) or hardware-synchronized transfers (see Section 10.5.2, *Software-Triggered (Nonsynchronized) Transfer*, and Section 10.5.3, *Hardware-Synchronized Transfer*).

2) For each channel in the chain, configure the DMA4_CLNK_CNTRL register as follows:
   - The next logical DMA channel number (for a looping chained transfer link last-channel to first-channel number) in the DMA register bits DMA4_CLNK_CNTRL[4:0]
   - Include the logical channel to the chain and enable the link by setting the DMA register bit DMA4_CLNK_CNTRL[15].
   - For a nonlooping chain, the last logical channel in the chain must have the DMA register bit DMA4_CLNK_CNTRL[15] set to 0 to indicate the end of the chain.

3) Enable the transfer using the enable bit in the first logical channel DMA register bit DMA4_CCR[7]. All other channels in the chain must be disabled. Each channel is enabled automatically in turn when the previous logical channel transfer completes. A software-triggered (nonsynchronized) 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 DMA4_CLNK_CTRL[15] (ENABLE_LNK bit set to 0x0) of the final channel for which to perform a transfer.

10.5.5 90-Degree Clockwise Image Rotation

The 90-degree clockwise image rotation example described in Section 10.4.3, *Addressing Modes*, can be programmed as follows:

1) Configure the transfer parameters in the logical DMA channel registers:
   - DMA4_CSDP
     - Transfer element size = 16-bit (16 BPP): DMA register bits DMA4_CSDP[1:0]
     - Read and write port access types = maximum burst size supported by memory device: DMA register bits DMA4_CSDP[8:7] and DMA4_CSDP[15:14]
     - Source and destination endianism: DMA register bits DMA4_CSDP[21] and DMA4_CSDP[19]
     - Write mode = posted with last element nonposted: DMA register bits DMA4_CSDP[17:16]
     - Source and destination packed = Yes (although the destination writes do not benefit because the element index > 1): DMA register bits DMA4_CSDP[6] and DMA4_CSDP[13]
   - DMA4_CEN: Number of elements per frame = 240
   - DMA4_CFN: Number of frames per transfer block = 160
   - DMA4_CSSA: Source start address = 0x100000
   - DMA4_CDSA: Destination start address = 0x200131E
   - DMA4_CCR
     - Read and write port addressing modes = double-index addressing mode for both or post-increment addressing on the source and double-index addressing on the destination: DMA register bits DMA4_CCR[13:12] and DMA4_CCR[15:14]
     - Low or high priority: DMA register bit DMA4_CCR[6]
     - DMA request number = 0 [for software-triggered (nonsynchronized transfers]: DMA register bits DMA4_CCR[4:0] and DMA4_CCR[20:19]
   - DMA4_CSEI: Source element index = 1
   - DMA4_CSFI: Source frame index = 1
   - DMA4_CDEI: Destination element index = 319
   - DMA4_CDFI: Destination frame index = −76481
2) Start the transfer using the enable bit in the channel DMA4_CCR register.

10.5.6 CamDMA Configuration Example

Consider how to transfer a complete picture from the camera to OMAP system memory. An image is identified by the following elements:

- ☐ PICTURE_WIDTH (number of pixels per line)
- ☐ PICTURE_HEIGHT (number of lines in a frame)

- ❏ PIXEL_TYPE (number of bytes per pixel)
- ❏ THRESHOLD_VALUE (amount of 32-bit data to transfer from the camera core to the CamDMA at each DMA request)

The complete transfer size can then be expressed as shown in Equation 10−7.

Equation 10−7. Image Transfer Size

[nr of bytes] = PICTURE_HEIGHT x PICTURE_WIDTH x PIXEL_TYPE

In DMA terms, this translates to the following:

PICTURE_HEIGHT = number of lines = number of DMA frames

PICTURE_WIDTH = number of pixels per frame = number of elements per frame

PIXEL_TYPE = pixel size in byte = element size = DATA_TYPE (can be 1, 2, or 4)

CAUTION

Avoid confusing the camera frame with a DMA frame. The camera frame = one image; a DMA frame = one image line.

The following shows Equation 10−7 fully translated to DMA terminology:

Image transfer size [nr of bytes] = nr of frames per image x nr of elements per frame x element size

Equation 10−8 shows the image transfer size expressed in registers and register field names:

Equation 10−8. Image Transfer Size

[nr of bytes] = DMA4_CEN x DMA4_CFN x DMA4_CCR[DATA_TYPE]

However, the camera interface probably cannot hold one complete image line in the camera core FIFO to buffer the data. Instead, each line is transferred in several subsequent transfers. Each of these subsequent transfers is as large as FIFO THRESHOLD_VALUE is defined. When FIFO THRESHOLD_VALUE is reached, a DMA request is asserted.

This situation benefits from source-packed synchronized transfers. This feature enables the DMA to make several transfers per image line, so-called packets, without counting down the frame number, hence still keeping valid.

To achieve optimal CamDMA performance, configure the CamDMA registers as described in Section 10.5.6.1, *Synchronization*.

10.5.6.1 Synchronization

In the DMA4_CCR register, configure the following bit fields:

FS=BS = 1 (to become packet synchronized)

SEL_SRC_DST_SYNC= src sync (to be synchronized on the source port)

SYNCHRO_CONTROL = 1 (There is only one DMA request line for the camera.)

10.5.6.2 Packet to Transfer per DMA Request

DMA4_CSFI = nr of elements to transfer per DMA request = THRESHOLD_VALUE x (4/PIXEL_TYPE)

DMA4_CSDP[DATA_TYPE] = PIXEL_TYPE

10.5.6.3 Image Size to Transfer

DMA4_CFN = PICTURE_HEIGHT

DMA4_CEN = PICTURE_WIDTH

DMA4_CCR[DATA_TYPE] is already set.

10.5.6.4 Addressing

In register DMA4_CSDP, configure the following bit fields:

SRC_PACKED = 1

DST_PACKED = 1

SRC_BURST = 8 x 64 bit

DST_BURST = 8 x 64 bit

DMA4_CSSA = 0x0 (The source start address is a constant address for reading the camera core FIFO.)

DMA4_CDSA = 0x0 (The destination start address for the camera is hardcoded, MMU translate.)

DMA4_CCR[src_amode] = post-incremented (when the source is specified to be packed)

Source constant addressing cannot be used when the source is packed. However, you can specify that SRC_AMODE be post-incrementing, because there is no address bus between the DMA read port and the camera core. This is the best way to achieve higher performance.

When the source uses the constant addressing mode, the source must be specified as nonpacked. Consequently, only one data (pixel-type: 8-bit, 16-bit) is read by the DMA (over the 64 bits) for each access, which is not good for performance.

DMA4_CCR[dst_amode] = post-incremented, single-incremented, or double-incremented addressing

DMA4_CSEI can be ignored.

DMA4_CSFI is already set (see Section 10.5.6.2, *Packet to Transfer per DMA Request*).

DMA4_CDEI = the appropriate configuration for the application

DMA4_CDFI = the appropriate configuration for the application

10.6 DMA Registers

Table 10−20. Instance Summary

Module Name	Base Address	Size
DDMA	IOMA + 0xC000	4K bytes
CamDMA	0x4805 2800	1K byte
sDMA	0x4805 6000	4K bytes

10.6.1 Register Summary for CamDMA, DDMA, and sDMA Controllers

Table 10−21 shows the DMA controller registers and their physical addresses. For a detailed description, see Section 10.6.2, *Common Register Descriptions for CamDMA, DDMA, and sDMA Controllers*.

To access these registers, address the memory space or the I/O space. The physical addresses provided are byte addresses; divide by 2 to derive the DSP half-word addresses (see Chapter 4, *DSP Subsystem*). All registers are 32 bits wide.

Table 10−21. sDMA, DDMA, and CamDMA Controller Register Summary

Register Name	Type	Offset Address	DDMA Physical Address Memory Space	DDMA Physical Address I/O Space	CamDMA Physical Address	sDMA Physical Address
DMA4_REVISION	R	0x0000	IOMA+0xC000	0x6000	0x4805 2800	0x4805 6000
DMA4_IRQSTATUS_L0	RW	0x0008	IOMA+0xC008	0x6008	0x4805 2808	0x4805 6008
DMA4_IRQSTATUS_L1	RW	0x000C	IOMA+0xC00C	0x600C	0x4805 280C	0x4805 600C
DMA4_IRQSTATUS_L2	RW	0x0010	IOMA+0xC010	0x6010	0x4805 2810	0x4805 6010
DMA4_IRQSTATUS_L3	RW	0x0014	IOMA+0xC014	0x6014	0x4805 2814	0x4805 6014
DMA4_IRQENABLE_L0	RW	0x0018	IOMA+0xC018	0x6018	0x4805 2818	0x4805 6018
DMA4_IRQENABLE_L1	RW	0x001C	IOMA+0xC01C	0x601C	0x4805 281C	0x4805 601C
DMA4_IRQENABLE_L2	RW	0x0020	IOMA+0xC020	0x6020	0x4805 2820	0x4805 6020
DMA4_IRQENABLE_L3	RW	0x0024	IOMA+0xC024	0x6024	0x4805 2824	0x4805 6024
DMA4_SYSSTATUS	R	0x0028	IOMA+0xC028	0x6028	0x4805 2828	0x4805 6028
DMA4_OCP_ SYSCONFIG	RW	0x002C	IOMA+0xC02C	0x602C	0x4805 282C	0x4805 602C
DMA4_CAPS_0	R	0x0064	IOMA+0xC064	0x6064	0x4805 2864	0x4805 6064
DMA4_CAPS_2	R	0x006C	IOMA+0xC06C	0x606C	0x4805 286C	0x4805 606C

Notes:

1) The variable *c* is the logical channel numbered 0x0 through 0x17 for DDMA, 0x3 for CamDMA, and 0x1F for sDMA.
2) The variable *n* represents the maximum number of logical channels minus 1, or 23 DDMA logical channels, 3 CamDMA logical channels, and 31 sDMA logical channels. (The DDMA has a maximum of 24 logical channels, the CamDMA has a maximum of 4 logical channels, and the sDMA has a maximum of 32 logical channels.)
3) IOMA refers to the page index used to access the DSP I/O space addresses from the DSP memory space. IOMA is programmed in the DSP subsystem IPI register (IPI_IOMAP. IOMA), the value of which is fixed by the DSP BIOS.

Table 10−21. sDMA, DDMA, and CamDMA Controller Register Summary (Continued)

Register Name	Type	Offset Address	DDMA Physical Address Memory Space	DDMA Physical Address I/O Space	CamDMA Physical Address	sDMA Physical Address
DMA4_CAPS_3	R	0x0070	IOMA+0xC070	0x6070	0x4805 2870	0x4805 6070
DMA4_CAPS_4	R	0x0074	IOMA+0xC074	0x6074	0x4805 2874	0x4805 6074
DMA4_GCR	RW	0x0078	IOMA+0xC078	0x6078	0x4805 2878	0x4805 6078
DMA4_CCR0 − DMA4_CCR*n*	RW	0x0080+ (*c**0x60)	IOMA+0xC080− IOMA+0xC920	0x6080−0x 6920	0x4805 2880− 0x4805 29A0	0x4805 6080− 0x4805 6C20
DMA4_CLNK_CTRL0 − DMA4_CLNK_CTRL*n*	RW	0x0084+ (*c**0x60)	IOMA+0xC084− IOMA+0xC924	0x6084−0x 6924	0x4805 2884− 0x4805 29A4	0x4805 6084− 0x4805 6C24
DMA4_CICR0 − DMA4_CICR*n*	RW	0x0088+ (*c**0x60)	IOMA+0xC088− IOMA+0xC928	0x6088−0x 6928	0x4805 2888− 0x4805 29A8	0x4805 6088− 0x4805 6C28
DMA4_CSR0 − DMA4_CSR*n*	RW	0x008C+ (*c**0x60)	IOMA+0xC08C− IOMA+0xC92C	0x608C−0x 692C	0x4805 288C− 0x4805 29AC	0x4805 608C− 0x4805 6C2C
DMA4_CSDP0 − DMA4_CSDP*n*	RW	0x0090+ (*c**0x60)	IOMA+0xC090− IOMA+0xC930	0x6090−0x 6930	0x4805 2890− 0x4805 29B0	0x4805 6090− 0x4805 6C30
DMA4_CEN0 − DMA4_CEN*n*	RW	0x0094+ (*c**0x60)	IOMA+0xC094− IOMA+0xC934	0x6094−0x 6934	0x4805 2894− 0x4805 29B4	0x4805 6094− 0x4805 6C34
DMA4_CFN0 − DMA4_CFN*n*	RW	0x0098+ (*c**0x60)	IOMA+0xC098− IOMA+0xC938	0x6098−0x 6938	0x4805 2898− 0x4805 29B8	0x4805 6098− 0x4805 6C38
DMA4_CSSA0 − DMA4_CSSA*n*	RW	0x009C+ (*c**0x60)	IOMA+0xC09C− IOMA+0xC93C	0x609C−0x 693C	0x4805 289C− 0x4805 29BC	0x4805 609C− 0x4805 6C3C
DMA4_CDSA0 − DMA4_CDSA*n*	RW	0x00A0+ (*c**0x60)	IOMA+0xC0A0− IOMA+0xC940	0x60A0−0x 6940	0x4805 28A0− 0x4805 29C0	0x4805 60A0− 0x4805 6C40
DMA4_CSEI0 − DMA4_CSEI*n*	RW	0x00A4+ (*c**0x60)	IOMA+0xC0A4− IOMA+0xC944	0x60A4−0x 6944	0x4805 28A4− 0x4805 29C4	0x4805 60A4− 0x4805 6C44
DMA4_CSFI0 − DMA4_CSFI*n*	RW	0x00A8+ (*c**0x60)	IOMA+0xC0A8− IOMA+0xC948	0x60A8−0x 6948	0x4805 28A8− 0x4805 29C8	0x4805 60A8− 0x4805 6C48

Notes:

1) The variable *c* is the logical channel numbered 0x0 through 0x17 for DDMA, 0x3 for CamDMA, and 0x1F for sDMA.
2) The variable *n* represents the maximum number of logical channels minus 1, or 23 DDMA logical channels, 3 CamDMA logical channels, and 31 sDMA logical channels. (The DDMA has a maximum of 24 logical channels, the CamDMA has a maximum of 4 logical channels, and the sDMA has a maximum of 32 logical channels.)
3) IOMA refers to the page index used to access the DSP I/O space addresses from the DSP memory space. IOMA is programmed in the DSP subsystem IPI register (IPI_IOMAP. IOMA), the value of which is fixed by the DSP BIOS.

Table 10−21. sDMA, DDMA, and CamDMA Controller Register Summary (Continued)

Register Name	Type	Offset Address	DDMA Physical Address Memory Space	DDMA Physical Address I/O Space	CamDMA Physical Address	sDMA Physical Address
DMA4_CDEI0 − DMA4_CDEI*n*	RW	0x00AC + (*c**0x60)	IOMA+0xC0AC− IOMA+0xC94C	0x60AC−0 x694C	0x4805 28AC− 0x4805 29CC	0x4805 60AC− 0x4805 6C4C
DMA4_CDFI0 − DMA4_CDFI*n*	RW	0x00B0+ (*c**0x60)	IOMA+0xC0B0− IOMA+0xC950	0x60B0−0x 6950	0x4805 28B0− 0x4805 29D0	0x4805 60B0− 0x4805 6C50
DMA4_CSAC0 − DMA4_CSAC*n*	R	0x00B4+ (*c**0x60)	IOMA+0xC0B4− IOMA+0xC954	0x60B4−0x 6954	0x4805 28B4− 0x4805 29D4	0x4805 60B4− 0x4805 6C54
DMA4_CDAC0 − DMA4_CDAC*n*	R	0x00B8+ (*c**0x60)	IOMA+0xC0B8− IOMA+0xC958	0x60B8−0x 6958	0x4805 28B8− 0x4805 29D8	0x4805 60B8− 0x4805 6C58
DMA4_CCEN0 − DMA4_CCEN*n*	R	0x00BC + (*c**0x60)	IOMA+0xC0BC− IOMA+0xC95C	0x60BC−0 x695C	0x4805 28BC− 0x4805 29DC	0x4805 60BC− 0x4805 6C5C
DMA4_CCFN0 − DMA4_CCFN*n*	R	0x00C0+ (*c**0x60)	IOMA+0xC0C0− IOMA+0xC960	0x60C0−0x 6960	0x4805 28C0− 0x4805 29E0	0x4805 60C0− 0x4805 6C60
DMA4_COLOR0 − DMA4_COLOR*n*	RW	0x00C4+ (*c**0x60)	NA	NA	NA	0x4805 60C4− 0x4805 6C64

Notes:

1) The variable *c* is the logical channel numbered 0x0 through 0x17 for DDMA, 0x3 for CamDMA, and 0x1F for sDMA.
2) The variable *n* represents the maximum number of logical channels minus 1, or 23 DDMA logical channels, 3 CamDMA logical channels, and 31 sDMA logical channels. (The DDMA has a maximum of 24 logical channels, the CamDMA has a maximum of 4 logical channels, and the sDMA has a maximum of 32 logical channels.)
3) IOMA refers to the page index used to access the DSP I/O space addresses from the DSP memory space. IOMA is programmed in the DSP subsystem IPI register (IPI_IOMAP. IOMA), the value of which is fixed by the DSP BIOS.

10.6.2 Common Register Description for CamDMA, DDMA, and sDMA Controllers

This section describes the registers used in the generic DMA controller. The same set of register descriptions is used for the DDMA, CamDMA, and sDMA. Register functionality is the same for all instantiations. The difference is in the number of logical channels and that some capabilities are not implemented in all instances. Footnotes are implemented where there are differences. Capabilities and physical differences between the instantiations are also summarized in Section 10.3, *DMA Module Integration*.

In the register descriptions, the variable *n* represents the maximum number of logical channels minus 1; that is, 23 DDMA logical channels, 3 CamDMA logical channels, and 31 sDMA logical channels. (The DDMA has a maximum of 24 logical channels, the CamDMA has a maximum of 4 logical channels, and the sDMA has a maximum of 32 logical channels.)

The physical addresses are byte addresses; divide by 2 to get the DSP half-word addresses. The term IOMA refers to the page index used to access the DSP I/O space addresses from the DSP memory space. IOMA is programmed in the DSP subsystem IPI register (IPI_IOMAP. IOMA), the value of which is fixed by the DSP BIOS.

Note:

Some registers have no reset value (marked as −) because of the hardware implementation in memory. Thus, the software must ensure that the register programming is correct.

Table 10−22. DMA4_REVISION

Address Offset	0x000		
Physical Address Memory	IOMA + 0xC000	**Instance**	DDMA
Physical Address I/O	0x6000	**Instance**	DDMA
Physical Address	0x4805 2800	**Instance**	CamDMA
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]: DMA major revision code [3:0]: DMA minor revision code	R	

Table 10–23. DMA4_IRQSTATUS_L0

Address Offset	0x008		
Physical Address Memory	IOMA + 0xC008	**Instance**	DDMA
Physical Address I/O	0x6008	**Instance**	DDMA
Physical Address	0x4805 2808	**Instance**	CamDMA
Physical Address	0x4805 6008	**Instance**	sDMA
Description	The interrupt status register regroups the statuses of DMA channels that can generate interrupts over line L0.		
Type	RW		

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="28">Reserved</td><td colspan="4">For CamDMA:
CH_*n*_0_L0</td></tr>
<tr><td colspan="8">Reserved</td><td colspan="24">For DDMA: CH_*n*_0_L0</td></tr>
<tr><td colspan="32">For sDMA: CH_*n*_0_L0</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
31:*n*+1	Reserved	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x0
n:0	CH_*n*_0_L0	Channel on L0: When an interrupt is detected on line L0, the status of interrupting channel i is read in bit field i. Read 0: Channel interrupt L0 false Read 1: Channel interrupt L0 true (pending) Write 0: Channel interrupt L0 status bit unchanged Write 1: Channel interrupt L0 status bit is reset.	RW	0x0

Table 10–24. DMA4_IRQSTATUS_L1

Address Offset	0x00C		
Physical Address Memory	IOMA + 0xC00C	**Instance**	DDMA
Physical Address I/O	0x600C	**Instance**	DDMA
Physical Address	0x4805 280C	**Instance**	CamDMA
Physical Address	0x4805 600C	**Instance**	sDMA
Description	The interrupt status register regroups the statuses of DMA channels that can generate interrupts over line L1.		
Type	RW		

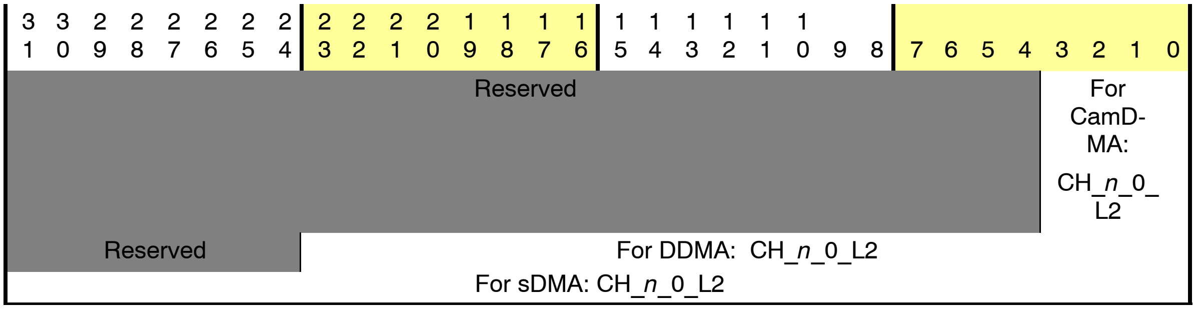
<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="28">Reserved</td><td colspan="4">For CamDMA:
CH_*n*_0_L1</td></tr>
<tr><td colspan="8">Reserved</td><td colspan="24">For DDMA: CH_*n*_0_L1</td></tr>
<tr><td colspan="32">For sDMA: CH_*n*_0_L1</td></tr>
</table>

Table 10–24. DMA4_IRQSTATUS_L1 (Continued)

Bit	Field Name	Description	Type	Reset
31: *n+1*	Reserved	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x0
n:0	CH_ *n* _0_L1	Channel on L1: When an interrupt is detected on line L1, the status of interrupting channel i is read in bit field i. Read 0: Channel interrupt L1 false Read 1: Channel interrupt L1 true (pending) Write 0: Channel interrupt L1 status bit unchanged Write 1: Channel interrupt L1 status bit is reset.	RW	0x0

Table 10–25. DMA4_IRQSTATUS_L2

Address Offset	0x010		
Physical Address Memory	IOMA + 0xC010	**Instance**	DDMA
Physical Address I/O	0x6010	**Instance**	DDMA
Physical Address	0x4805 2810	**Instance**	CamDMA
Physical Address	0x4805 6010	**Instance**	sDMA
Description	The interrupt status register regroups the statuses of DMA channels that can generate interrupts over line L2.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31: *n+1*	Reserved	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x0
n:0	CH_ *n* _0_L2	Channel on L2: When an interrupt is detected on line L2, the status of interrupting channel i is read in bit field i. Read 0: Channel interrupt L2 false Read 1: Channel interrupt L2 true (pending) Write 0: Channel interrupt L2 status bit unchanged Write 1: Channel interrupt L2 status bit is reset.	RW	0x0

Table 10−26. DMA4_IRQSTATUS_L3

Address Offset	0x014		
Physical Address Memory	IOMA + 0xC014	**Instance**	DDMA
Physical Address I/O	0x6014	**Instance**	DDMA
Physical Address	0x4805 2814	**Instance**	CamDMA
Physical Address	0x4805 6014	**Instance**	sDMA
Description	The interrupt status register regroups all the status of the DMA channels that can generate an interrupt over line L3.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								For CamDMA: CH_*n*_0_L3							
Reserved								For DDMA: CH_*n*_0_L3																							
For sDMA: CH_*n*_0_L3																															

Bits	Field Name	Description	Type	Reset
31: *n+1*	Reserved	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x0
n:0	CH_ *n* _0_L3	Channel i on L3: When an interrupt is detected on line L3, the status of interrupting channel i is read in bit field i. Read 0: Channel 31 interrupt L3 false Read 1: Channel 31 interrupt L3 true (pending) Write 0: Channel 31 interrupt L3 status bit unchanged Write 1: Channel 31 interrupt L3 status bit is reset.	RW	0x0

Table 10−27. DMA4_IRQENABLE_L0

Address Offset	0x018		
Physical Address Memory	IOMA + 0xC018	**Instance**	DDMA
Physical Address I/O	0x6018	**Instance**	DDMA
Physical Address	0x4805 2818	**Instance**	CamDMA
Physical Address	0x4805 6018	**Instance**	sDMA
Description	The interrupt enable register allows masking/unmasking of the module internal sources of interrupt, on line L0		
Type	RW		

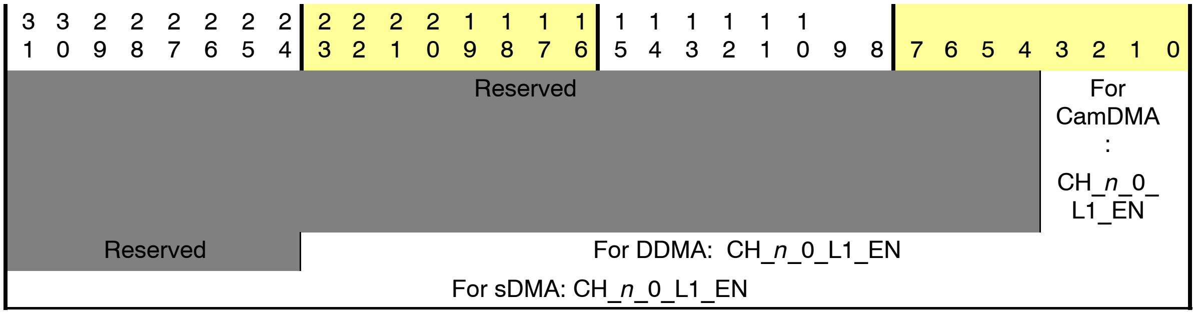
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								For CamDMA: CH_*n*_0_L0_EN							
Reserved								For DDMA: CH_*n*_0_L0_EN																							
For sDMA: CH_*n*_0_L0_EN																															

Table 10–27. DMA4_IRQENABLE_L0 (Continued)

Bits	Field Name	Description		Type	Reset
31: *n+1*	Reserved	Reserved. Write 0s for future compatibility. Read returns 0.		RW	0x0
n:0	CH_n_0_L0_EN			RW	0x0
		0x0:	Channel interrupt L0 is masked.		
		0x1:	Channel interrupt L0 generates an interrupt.		

Table 10–28. DMA4_IRQENABLE_L1

Address Offset	0x01C		
Physical Address Memory	IOMA + 0xC01C	**Instance**	DDMA
Physical Address I/O	0x601C	**Instance**	DDMA
Physical Address	0x4805 281C	**Instance**	CamDMA
Physical Address	0x4805 601C	**Instance**	sDMA
Description	The interrupt enable register allows masking/unmasking of the module internal sources of interrupt, on line L1.		
Type	RW		



Bits	Field Name	Description		Type	Reset
31: *n+1*	Reserved	Reserved. Write 0s for future compatibility. Read returns 0.		RW	0x0
n:0	CH_n_0_L1_EN			RW	0x0
		0x0:	Channel interrupt L1 is masked.		
		0x1:	Channel interrupt L1 generates an interrupt when it occurs.		

Table 10−29. DMA4_IRQENABLE_L2

Address Offset	0x020		
Physical Address Memory	IOMA + 0xC020	**Instance**	DDMA
Physical Address I/O	0x6020	**Instance**	DDMA
Physical Address	0x4805 2820	**Instance**	CamDMA
Physical Address	0x4805 6020	**Instance**	sDMA
Description	The interrupt enable register allows masking/unmasking of the module internal sources of interrupt, on line L2		
Type	RW		

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="28">Reserved</td><td colspan="4">For CamDMA: CH_*n*_0_L2_EN</td></tr>
<tr><td colspan="8">Reserved</td><td colspan="24">For DDMA: CH_*n*_0_L2_EN</td></tr>
<tr><td colspan="32">For sDMA: CH_*n*_0_L2_EN</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
31: *n+1*	Reserved	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x0
n:0	CH_n_0_L2_EN	0x0: Channel interrupt L2 is masked. 0x1: Channel interrupt L2 generates an interrupt when it occurs.	RW	0x0

Table 10−30. DMA4_IRQENABLE_L3

Address Offset	0x024		
Physical Address Memory	IOMA + 0xC024	**Instance**	DDMA
Physical Address I/O	0x6024	**Instance**	DDMA
Physical Address	0x4805 2824	**Instance**	CamDMA
Physical Address	0x4805 6024	**Instance**	sDMA
Description	The interrupt enable register allows masking/unmasking of the module internal sources of interrupt, on line L3		
Type	RW		

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="28">Reserved</td><td colspan="4">For CamDMA: CH_*n*_0_L3_EN</td></tr>
<tr><td colspan="8">Reserved</td><td colspan="24">For DDMA: CH_*n*_0_L3_EN</td></tr>
<tr><td colspan="32">For sDMA: CH_*n*_0_L3_EN</td></tr>
</table>

Table 10–30. DMA4_IRQENABLE_L3 (Continued)

Bits	Field Name	Description	Type	Reset
31: *n+1*	Reserved	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x0
n:0	CH_n_0_L3_EN		RW	0x0
		0x0: Channel interrupt L3 is masked.		
		0x1: Channel interrupt L3 generates an interrupt when it occurs.		

Table 10–31. DMA4_SYSSTATUS

Address Offset	0x028		
Physical Address Memory	IOMA + 0xC028	**Instance**	DDMA
Physical Address I/O	0x6028	**Instance**	DDMA
Physical Address	0x4805 2828	**Instance**	CamDMA
Physical Address	0x4805 6028	**Instance**	sDMA
Description	The register provides status information about the module, excluding interrupt status information (see the interrupt status 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																															RESETDONE

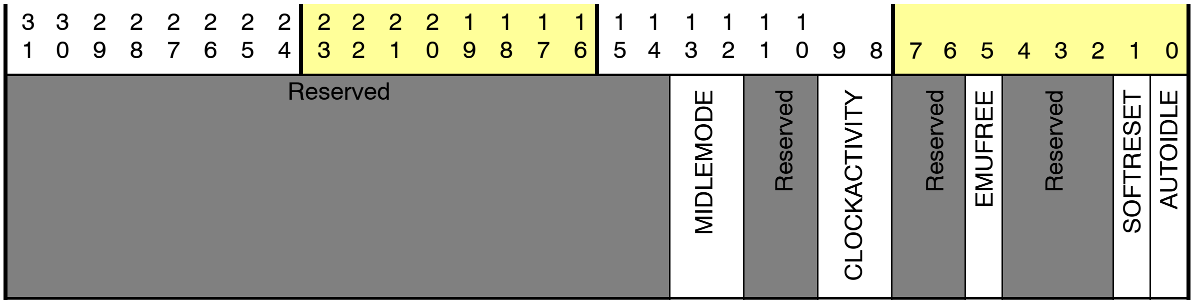
Bits	Field Name	Description	Type	Reset
31:1	Reserved	Reserved for module-specific status information	RW	0x00000000
0	RESETDONE	Internal reset monitoring	R	–
		0x0: Internal module reset is ongoing.		
		0x1: Reset complete[1]		

---

[1] During reset, the value is 0, but the value read just after the reset is 1, because ICLK/FCLK is already operating.

Table 10–32. DMA4_OCP_SYSCONFIG

Address Offset	0x02C		
Physical Address Memory	IOMA + 0xC02C	**Instance**	DDMA
Physical Address I/O	0x602C	**Instance**	DDMA
Physical Address	0x4805 282C	**Instance**	CamDMA
Physical Address	0x4805 602C	**Instance**	sDMA
Description	This register controls the OCP interface parameters.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:14	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000
13:12	MIDLEMODE	Read write power management, standby/wait control 0x0: Force standby: MSTANDBY is asserted only when all DMA channels are disabled. 0x1: No standby: MSTANDBY is never asserted. 0x2: MSTANDBY is asserted if at least one of the following two conditions is satisfied: All channels are disabled. There is no nonsynchronized channel enabled and or no DMA request input is asserted and there is no request in the read and write port scheduler state-machine. **Note:** MIDLEMODE cannot be set to 0x2 for CamDMA. 0x3: Reserved for second smart-standby mode if needed	RW	0x0
11:10	Reserved	Reserved for clock activity extension	RW	0x0
9:8	CLOCKACTIVITY	Clock activity during wakeup 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

Table 10–32. DMA4_OCP_SYSCONFIG (Continued)

Bits	Field Name	Description	Type	Reset
5	EMUFREE	Enable sensitivity to MSUSPEND 0x0: DMA4 freezes its internal logic on MSUSPEND assertion. 0x1: DMA4 ignores the MSUSPEND input.	RW	0
4:2	Reserved	Reserved Write 0s for future compatibility.	RW	0x0
1	SOFTRESET	Software reset. Set this bit to 1 to trigger a module reset. The bit is automatically reset by the hardware. Read returns 0. 0x0: No effect 0x1: Reset	RW	0
0	AUTOIDLE	Internal OCP clock-gating strategy 0x0: OCP clock is free-running. 0x1: Automatic OCP clock gating strategy is applied, based on OCP interface activity.	RW	0

Table 10–33. DMA4_CAPS_0

Address Offset	0x064		
Physical Address Memory	IOMA + 0xC064	**Instance**	DDMA
Physical Address I/O	0x6064	**Instance**	DDMA
Physical Address	0x4805 2864	**Instance**	CamDMA
Physical Address	0x4805 6064	**Instance**	sDMA
Description	DMA capabilities register 0 LSW		
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												CONST_FILL_CPBLTY	TRANSPARENT_BLT_CPBLTY	Reserved																	

Table 10–33. DMA4_CAPS_0 (Continued)

Bits	Field Name	Description	Type	Reset
31:20	Reserved	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x000
19	CONST_FILL_CPBLTY[1]	Constant_Fill_Capability 0: No LCH supports constant fill copy. 1: Any LCH supports constant fill copy.	R	1
18	TRANSPARENT_BLT_CPBLTY†	Transparent_BLT_Capability 0: No LCH supports transparent BLT copy. 1: Any LCH supports transparent BLT copy.	R	1
17:0	Reserved	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x00000

1) This feature is not supported for DDMA or CamDMA; hence, the reset value is 0.

Table 10–34. DMA4_CAPS_2

Address Offset	0x06C		
Physical Address Memory	IOMA + 0xC06C	**Instance**	DDMA
Physical Address I/O	0x606C	**Instance**	DDMA
Physical Address	0x4805 286C	**Instance**	CamDMA
Physical Address	0x4805 606C	**Instance**	sDMA
Description	DMA capabilities register 2		
Type	R		

31–9	8	7	6	5	4	3	2	1	0
Reserved	SEPARATE_SRC_AND_DST_INDEX_CPBLTY	DST_DOUBLE_INDEX_ADRS_CPBLTY	DST_SINGLE_INDEX_ADRS_CPBLTY	DST_POST_INCRMNT_ADRS_CPBLTY	DST_CONST_ADRS_CPBLTY	SRC_DOUBLE_INDEX_ADRS_CPBLTY	SRC_SINGLE_INDEX_ADRS_CPBLTY	SRC_POST_INCREMENT_ADRS_CPBLTY	SRC_CONST_ADRS_CPBLTY

Table 10–34. DMA4_CAPS_2 (Continued)

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	1
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	1
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	1
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	1
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	1
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	1
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	1
1	SRC_POST_ INCREMENT_ SRC_POST_ INCREMENT_ ADRS_CPBLTY	Source_post_increment_address_capability 0x0: Does not support post-increment address mode in the source port 0x1: Supports post-increment address mode in the source port	R	1

Table 10–34. DMA4_CAPS_2 (Continued)

Bits	Field Name	Description		Type	Reset
0	SRC_CONST_ SRC_CONST_ ADRS_CPBLTY	Source_constant_address_capability		R	1
		0x0:	Does not support constant address mode in the source port		
		0x1:	Supports constant address mode in the source port		

Table 10–35. DMA4_CAPS_3

Address Offset	0x070		
Physical Address Memory	IOMA + 0xC070	**Instance**	DDMA
Physical Address I/O	0x6070	**Instance**	DDMA
Physical Address	0x4805 2870	**Instance**	CamDMA
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	LCH_INTERLEAVE_CPBLTY	Reserved	FRAME_SUNCHR_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		R	1
		0x0:	Does not support synchronization transfer on block boundary		
		0x1:	Supports synchronization transfer on block boundary		
6	PKT_SYNCHR_ CPBLTY	Packet_synchronization_capability		R	1
		0x0:	Does not support synchronization transfer on packet boundary		
		0x1:	Supports synchronization transfer on packet boundary		

Table 10–35. DMA4_CAPS_3 (Continued)

Bits	Field Name	Description	Type	Reset
5	CHANNEL_ CHAINING_ CPBLTY	Channel_chaining_capability 0x0: Does not support logical channel chaining 0x1: Supports logical channel chaining	R	1
4	LCH_ INTERLEAVE_ CPBLTY	LCh_interleave_capability 0x0: Does not support logical channel interleaving 0x1: Supports logical channel interleaving	R	1
3:2	Reserved	Reserved. Write 0s for future compatibility. Read returns 0.	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	1
0	ELMNT_ SYNCHR_ CPBLTY	Element_synchronization_capability 0x0: Does not support synchronization transfer on element boundary 0x1: Supports synchronization transfer on element boundary	R	1

Table 10–36. DMA4_CAPS_4

Address Offset	0x074		
Physical Address Memory	IOMA + 0xC074	**Instance**	DDMA
Physical Address I/O	0x6074	**Instance**	DDMA
Physical Address	0x4805 2874	**Instance**	CamDMA
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																								PKT_INTERRUPT_CPBLTY	SYNC_STATUS_CPBLTY	BLOCK_INTERRUPT_CPBLTY	LAST_FRAME_INTERRUPT_CPBLTY	FRAME_INTERRUPT_CPBLTY	HALF_FRAME_INTERRUPT_CPBLTY	EVENT_DROP_INTERRUPT_CPBLTY	Reserved

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x000000
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	1
6	SYNC_STATUS_ CPBLTY	Sync_status_capability 0x0: Does not support synchronized transfer status bit generation 0x1: Supports synchronized transfer status bit generation	R	1
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	1

Table 10–36. DMA4_CAPS_4 (Continued)

Bits	Field Name	Description	Type	Reset
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	1
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	1
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	1
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	1
0	Reserved	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0

Table 10–37. DMA4_GCR

Address Offset	0x078		
Physical Address Memory	IOMA + 0xC078	**Instance**	DDMA
Physical Address I/O	0x6078	**Instance**	DDMA
Physical Address	0x4805 2878	**Instance**	CamDMA
Physical Address	0x4805 6078	**Instance**	sDMA
Description	Channel DMA global 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	ARBITRATION_RATE	Reserved	MAX_CHANNEL_FIFO_DEPTH

Table 10–37. DMA4_GCR (Continued)

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
15:8	Reserved	Reserved. Write 0s for future compatibility. Read returns 0	RW	0x00
7:0	MAX_CHANNEL_ FIFO_DEPTH	Maximum FIFO depth allocated to one logical channel. Maximum FIFO depth cannot be 0x0. It should be at least 0x1 or greater. If channel limit is less than destination burst size, not enough data will accumulate 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 10–38. DMA4_CCR0 – DMA4_CCRn

Address Offset	0x080–0x1A0 in 0x60 byte increments		
Physical Address	0x0000 0080–0x0000 01A0	**Instance**	CDMA
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						BUFFERING_DISABLE	SEL_SRC_DST_SYNC†	PREFETCH	Resrved	Resrved	SYNCHRO_CONTROL_UPPER		BS	TRANSPARENT_COPY_ENABLE	CONST_FILL_ENABLE	DST_AMODE		SRC_AMODE		Reserved	WR_ACTIVE	RD_ACTIVE	SUSPEND_SENSITIVE	ENABLE	PRIO	FS	SYNCHRO_CONTROL				

Table 10−38. DMA4_CCR0 − DMA4_CCRn (Continued)

Bits	Field Name	Description	Type	Reset
31:26	Reserved	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x00
25	BUFFERING_ DISABLE	This bit allows default buffering to be disabled when the transfer is source synchronized. 0x0: Buffering is enabled across element/packet/frame when source is synchronized to element/packet/frame. 0x1: Buffering is disabled across element/packet/frame when source is synchronized to element/packet/frame. Buffering disable is not allowed in destination synchronized mode.	RW	−
24	SEL_SRC_ DST_SYNC[1]	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 That is, a read from SRC or a write to DST is triggered by a DMA request. 0x0: The destination triggers on the DMA request. A write to DST is synchronized on the DMA request and a read from SRC is controlled by the PREFETCH bit. If synchronized on packet, the packet element number is specified in the CDFI register. 0x1: The source triggers on the DMA request. A read from SRC is synchronized on the DMA request and a write to DST depends on the number of data inside the FIFO and the destination transfer. If synchronized on packet, the packet element number is specified in the CSFI register.	RW	−
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. Prefetch = 1 and Sel_Src_Dst_Sync = 1 must not occur at the same time. This mode is not supported.	RW	−
22	Reserved	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0
21	Reserved	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0
20:19	SYNCHRO_ CONTROL_ UPPER	Channel synchronization control upper (used with the 5 bits of synchronized channel DMA4_CCR[4:0]) Used together, as two most-significant bits (MSBs), with the five bits of the synchronized channel bit field.	RW	0x−

1) Only synchronized source is supported for CamDMA. The software must not write 0 to this field before enabling the channel.

2) Must be set to 00001 for CamDMA source synchronization

Table 10–38. DMA4_CCR0 – DMA4_CCRn (Continued)

Bits	Field Name	Description	Type	Reset
18	BS	Block synchronization This bit is used in with the FS bit to determine how the DMA request is serviced in a synchronized transfer.	RW	–
17	TRANSPARENT_COPY_ENABLE	Transparent copy enable 0x0: Transparent copy mode is disabled. 0x1: Transparent copy mode is enabled.	RW	–
16	CONST_FILL_ENABLE	Constant fill enable 0x0: Constant fill mode is disabled. 0x1: Constant fill mode is enabled.	RW	–
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	0
10	WR_ACTIVE	Indicates whether the channel write context is active 0x0: Channel is not active on the write port. 0x1: Channel is active on the write port.	R	–
9	RD_ACTIVE	Indicates whether the channel read context is active 0x0: Channel is not active on the read port. 0x1: Channel is active on the read port.	R	–
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, all current OCP services (single transaction or burst transaction as specified in the corresponding CSDP register) must be complete before any more transactions are stopped.	RW	–

1) Only synchronized source is supported for CamDMA. The software must not write 0 to this field before enabling the channel.

2) Must be set to 00001 for CamDMA source synchronization

Table 10−38. DMA4_CCR0 − DMA4_CCRn (Continued)

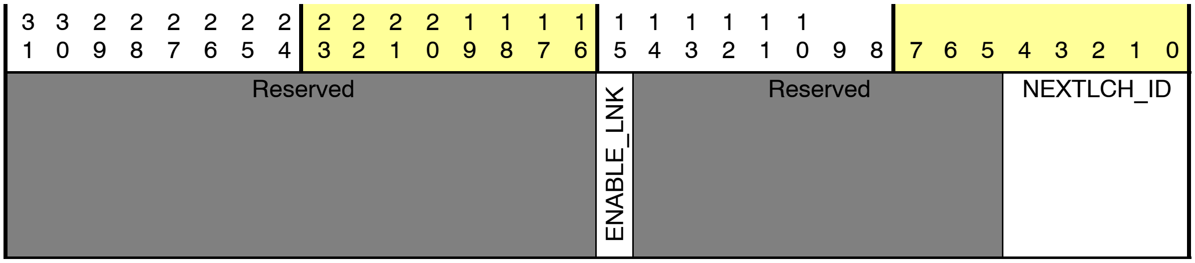
Bits	Field Name	Description	Type	Reset
7	ENABLE	Logical channel enable. The software must clear the CSR register and the IRQSTATUS bit for the different interrupt lines before enabling the channel. 0x0: The logical channel is disabled. 0x1: The logical channel is enabled.	RW	0
6	PRIO	Channel priority 0x0: Channel has low priority during the arbitration process. 0x1: Channel has high priority during the arbitration process.	RW	–
5	FS	Frame synchronization This bit is used with the BS bit to determine how the DMA request is serviced in a synchronized transfer. FS = 0 and BS = 0: An element is transferred when a DMA request is made. FS = 0 and BS = 1: An entire block is transferred when a DMA request is made. FS = 1 and BS = 0: An entire frame is transferred when a DMA request is made. FS = 1 and BS = 1: A packet is transferred when a DMA request is made. All these transfers can be interleaved on the port with other DMA requests.	RW	–
4:0	SYNCHRO_ CONTROL	Channel synchronization control This bit field used in with the second_level_ synchro_control_upper (as two MSB) 0000000: Reserved for nonsynchronized LCH transfer xxxxxxx: Which DMA request line number to enter is found in the DMA request mapping tables (Table 10−7 and Table 10−11) for sDMA and DDMA, respectively. CamDMA has only one DMA request[2].	RW	0x−−

1) Only synchronized source is supported for CamDMA. The software must not write 0 to this field before enabling the channel.

2) Must be set to 00001 for CamDMA source synchronization

Table 10–39. DMA4_CLNK_CTRL0 – DMA4_CLNK_CTRLn

Address Offset	0x084–0x924 in 0x60 byte increments		
Physical Address Memory	IOMA + 0xC084–IOMA + 0xC924	**Instance**	DDMA
Physical Address I/O	0x6084–0x6924	**Instance**	DDMA
Physical Address	0x4805 2884–0x4805 29A4	**Instance**	CamDMA
Physical Address	0x4805 6084–0x4805 6C24	**Instance**	sDMA
Description	Channel link control register		
Type	RW		



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 disables channel linking 0: Channel linking mode is disabled. When set on the fly to 0, the current channel completes the transfer and stops the chain linking. 1: Channel linking mode is enabled. The logical channel defined in NextLCH_ID is enabled at the end of the current transfer.	RW	–
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 the logical channel chaining queue.	RW	0x––

Table 10−40. DMA4_CICR0 − DMA4_CICRn

Address Offset	0x088−0x928 in 0x60 byte increments		
Physical Address Memory	IOMA + 0xC088−IOMA + 0xC928	**Instance**	DDMA
Physical Address I/O	0x6088−0x6928	**Instance**	DDMA
Physical Address	0x4805 2888−0x4805 29A8	**Instance**	CamDMA
Physical Address	0x4805 6088−0x4805 6C28	**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																				MISALIGNED_ERR_IE	Reserved	Reserved	TRANS_ERR_IE	PKT_IE	Reserved	BLOCK_IE	LAST_IE	FRAME_IE	HALF_IE	DROP_IE	Reserved

Bits	Field Name	Description	Type	Reset
31:12	Reserved	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0
11	MISALIGNED_ ERR_IE	Enables the address misaligned error event interrupt 0: Disables the misaligned address error event interrupt 1: Enables the misaligned address error event interrupt	RW	–
10	Reserved	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0
9	Reserved	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0
8	TRANS_ERR_IE	Enables the transaction error event interrupt 0: Disables the transaction error event interrupt 1: Enables the transaction error event interrupt	RW	–
7	PKT_IE	Enables the end-of-packet interrupt 0: Disables the end-of-packet transfer interrupt 1: Enables the end-of-packet transfer interrupt	RW	–
6	Reserved	Reserved. Write 0s for future compatibility. Read returns 0	RW	–
5	BLOCK_IE	Enables the end-of-block interrupt 0: Disables the end-of-block interrupt 1: Enables the end-of-block interrupt	RW	–

Table 10−40. DMA4_CICR0 – DMA4_CICRn (Continued)

Bits	Field Name	Description	Type	Reset
4	LAST_IE	Last-frame interrupt enable (start of last frame) 0: Disables the last-frame interrupt 1: Enables the last-frame interrupt	RW	–
3	FRAME_IE	Frame interrupt enable (end of frame) 0: Disables the end-of-frame interrupt 1: Enables the end-of-frame interrupt	RW	–
2	HALF_IE	Enables or disables the half-frame interrupt. 0: Disables the half-frame interrupt 1: Enables the half-frame interrupt	RW	–
1	DROP_IE	Synchronization event drop interrupt enable (request collision) 0: Disables the event drop interrupt 1: Enables the event drop interrupt	RW	–
0	Reserved	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0

Table 10−41. DMA4_CSR0 – DMA4_CSRn

Address Offset	0x08C−0x92C in 0x60 byte increments		
Physical Address Memory	IOMA + 0xC08C−IOMA + 0xC92C	**Instance**	DDMA
Physical Address I/O	0x608C–0x692C	**Instance**	DDMA
Physical Address	0x4805 288C−0x4805 29AC	**Instance**	CamDMA
Physical Address	0x4805 608C−0x4805 6C2C	**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																				MISALIGNED_ADRS_ERR	Reserved	Reserved	TRANS_ERR	PKT	SYNC	BLOCK	LAST	FRAME	HALF	DROP	Reserved

Bits	Field Name	Description	Type	Reset
31:12	Reserved	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0

Table 10–41. DMA4_CSR0 – DMA4_CSRn (Continued)

Bits	Field Name	Description	Type	Reset
11	MISALIGNED_ ADRS_ERR	Misaligned address error event Read 1: An address error occurred. Read 0: No address error Write 0: Status bit unchanged Write 1: Status bit is reset.	RW	–
10	Reserved	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0
9	Reserved	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0
8	TRANS_ERR	Transaction error event Read 1: A transaction error occurred. Read 0: No transaction error Write 0: Status bit unchanged Write 1: Status bit is reset.	RW	–
7	PKT	End of Packet transfer Read 1: The current packet was transferred. Read 0: The current packet transfer was not complete. Write 0: Status bit unchanged Write 1: Status bit is reset.	RW	–
6	SYNC	Synchronization status of a channel Read 1: Logical channel is servicing a synchronized DMA request. Read 0: Logical channel is not scheduled or servicing a nonsynchronized DMA request. Write 0: Status bit unchanged Write 1: Status bit is reset.	RW	–
5	BLOCK	End-of-block event Read 1: The current block was transferred. Read 0: The current block transfer was not complete. Write 0: Status bit unchanged Write 1: Status bit is reset.	RW	–
4	LAST	Last frame (start of last frame) Read 1: The start of the last frame to transfer was reached. Read 0: The start of the last frame to transfer was not reached. Write 0: Status bit unchanged Write 1: Status bit is reset.	RW	–
3	FRAME	End-of-frame event Read 1: The end of the current transferred frame was reached. Read 0: The end of the current transferred frame was not reached. Write 0: Status bit unchanged Write 1: Status bit is reset.	RW	–

Table 10−41. DMA4_CSR0 – DMA4_CSRn (Continued)

Bits	Field Name	Description	Type	Reset
2	HALF	Half-of-frame event. Read 1: Half of the current transferred frame was transferred. Read 0: Half of the current transferred frame was not transferred. Write 0: Status bit unchanged Write 1: Status bit is reset.	RW	–
1	DROP	Synchronization event drop occurred during the transfer. Read 1: A synchronization collision occurred. Read 0: No synchronization collision Write 0: Status bit unchanged Write 1: Status bit is reset.	RW	–
0	Reserved	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0

Table 10−42. DMA4_CSDP0 – DMA4_CSDPn

Address Offset	0x090−0x930 in 0x60 byte increments		
Physical Address Memory	IOMA + 0xC090−IOMA + 0xC930	**Instance**	DDMA
Physical Address I/O	0x6090–0x6930	**Instance**	DDMA
Physical Address	0x4805 2890−0x4805 29B0	**Instance**	CamDMA
Physical Address	0x4805 6090−0x4805 6C30	**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_ENCIAN_LOCK	WRITE_MODE		DST_BURST_EN		DST_PACKED	WR_ADD_TRSLT				SRC_BURST_EN		SRC_PACKED	RD_ADD_TRSLT				DATA_TYPE	

[†] Only supported in DDMA. Not supported in sDMA or CamDMA.

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 endianism control 0: Source has little-endian type 1: Source has big-endian type	RW	–
20	SRC_ENDIAN_ LOCK	Endianism lock 0: Endianism adapt 1: Endianism lock	RW	–

[†] Only supported in DDMA. Not supported in sDMA or CamDMA.

Table 10−42. DMA4_CSDP0 − DMA4_CSDPn (Continued)

Bits	Field Name	Description	Type	Reset
19	DST_ENDIAN	Channel destination endianism control 0: Destination has little-endian type 1: Destination has big-endian type	RW	−
18	DST_ENDIAN_LOCK	Endianism lock 0: Endianism adapt 1: Endianism lock	RW	−
17:16	WRITE_MODE	Used to enable writing mode with or without posting 00: Write nonposted (WRNP) 01: Write (posted) 10: Transfer continues without waiting for each write to complete, but the final write of each frame is complete before the frame transfer can complete (posted with final write not posted) 11: 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. 00: Single access 01: 16 bytes or 4x32-bit/2x64-bit burst access 10: 32 bytes or 8x32-bit/4x64-bit burst access 11: 64 bytes or 16x32-bit/8x64-bit burst access	RW	0x−
13	DST_PACKED	Destination receives packed data. 1: The destination target is packed. 0: The destination target is nonpacked.	RW	−
12:9	WR_ADD_TRSLT†	Enables the MReqAddressTranslate sideband signal in the write port side. Must be set to 0x0 for proper operation.	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. 00: Single access 01: 16 bytes or 4x32-bit/2x64-bit burst access 10: 32 bytes or 8x32-bit/4x64-bit burst access 11: 64 bytes or 16x32-bit/8x64-bit burst access	RW	0x−
6	SRC_PACKED	Source provides packed data. 1: The source target is packed. 0: The source target is nonpacked.	RW	−
5:2	RD_ADD_TRSLT†	Enables the MReqAddressTranslate sideband signal in the read port side. Must be set to 0x0 for correct operation.	RW	0x−
1:0	DATA_TYPE	Defines the type of the data moved in the channel. 00: 8-bit scalar 01: 16-bit scalar 10: 32-bit scalar 11: Reserved	RW	0x−

† Only supported in DDMA. Not supported in sDMA or CamDMA.

Table 10−43. DMA4_CEN0 − DMA4_CENn

Address Offset	0x094−0x934 in 0x60 byte increments		
Physical Address Memory	IOMA + 0xC094−IOMA + 0xC934	**Instance**	DDMA
Physical Address I/O	0x6094–0x6934	**Instance**	DDMA
Physical Address	0x4805 2894−0x4805 29B4	**Instance**	CamDMA
Physical Address	0x4805 6094−0x4805 6C34	**Instance**	sDMA
Description	Channel element number		
Type	RW		

31	30	29	28	27	26	25	24	23	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 in a frame (unsigned) to transfer	RW	0x––––––

Table 10−44. DMA4_CFN0 − DMA4_CFNn

Address Offset	0x098−0x938 in 0x60 byte increments		
Physical Address Memory	IOMA + 0xC098−IOMA + 0xC938	**Instance**	DDMA
Physical Address I/O	0x6098–0x6938	**Instance**	DDMA
Physical Address	0x4805 2898−0x4805 29B8	**Instance**	CamDMA
Physical Address	0x4805 6098−0x4805 6C38	**Instance**	sDMA
Description	Channel frame number		
Type	RW		

31	30	29	28	27	26	25	24	23	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	0x––––
15:0	CHANNEL_ FRAME_NBR	Number of frames in the block to be transferred (unsigned)	RW	0x0000

Table 10−45. DMA4_CSSA0 – DMA4_CSSAn

Address Offset	0x09C−0x93C in 0x60 byte increments		
Physical Address Memory	IOMA+ 0xC09C−IOMA + 0xC93C	**Instance**	DDMA
Physical Address I/O	0x609C–0x693C	**Instance**	DDMA
Physical Address	0x4805 289C−0x4805 29BC	**Instance**	CamDMA
Physical address	0x4805 609C−0x4805 6C3C	**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 10−46. DMA4_CDSA0 – DMA4_CDSAn

Address Offset	0x0A0−0x940 in 0x60 byte increments		
Physical Address Memory	IOMA + 0xC0A0−IOMA + 0xC940	**Instance**	DDMA
Physical Address I/O	0x60A0–0x6940	**Instance**	DDMA
Physical Address	0x4805 28A0−0x4805 29C0	**Instance**	CamDMA
Physical Address	0x4805 60A0−0x4805 6C40	**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 10−47. DMA4_CSEI0 – DMA4_CSEIn

Address Offset	0x0A4−0x944 in 0x60 byte increments		
Physical Address Memory	IOMA + 0xC0A4−IOMA + 0xC944	**Instance**	DDMA
Physical Address I/O	0x60A4–0x6944	**Instance**	DDMA
Physical Address	0x4805 28A4−0x4805 29C4	**Instance**	CamDMA
Physical Address	0x4805 60A4−0x4805 6C44	**Instance**	sDMA
Description	Channel source element 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																CHANNEL_SRC_ELMNT_INDEX															

Table 10−47. DMA4_CSEI0 – DMA4_CSEIn (Continued)

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 10−48. DMA4_CSFI0 – DMA4_CSFIn

Address Offset	0x0A8−0x948 in 0x60 byte increments		
Physical Address Memory	IOMA + 0xC0A8−IOMA + 0xC948	**Instance**	DDMA
Physical Address I/O	0x60A8−0x6948	**Instance**	DDMA
Physical Address	0x4805 28A8−0x4805 29C8	**Instance**	CamDMA
Physical Address	0x4805 60A8−0x4805 6C48	**Instance**	sDMA
Description	Channel source frame index 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_16BIT_PKT_ELNT_NBR																															

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 DMA4_CCR[SEL_SRC_DST_SYNC]=1; the bit field [15:0] gives the number of elements in the packet. The field [31:16] is not used for the packet size.	RW	0x−−−−−−−−

Table 10−49. DMA4_CDEI0 – DMA4_CDEIn

Address Offset	0x0AC−0x94C in 0x60 byte increments		
Physical Address Memory	IOMA + 0xC0AC−IOMA + 0xC94C	**Instance**	DDMA
Physical Address I/O	0x60AC–0x694C	**Instance**	DDMA
Physical Address	0x4805 28AC−0x4805 29CC	**Instance**	CamDMA
Physical Address	0x4805 60AC−0x4805 6C4C	**Instance**	sDMA
Description	Channel destination element 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																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 10−50. DMA4_CDFI0 – DMA4_CDRIn

Address Offset	0x0B0−0x950 in 0x60 byte increments		
Physical Address Memory	IOMA + 0xC0B0−IOMA + 0xC950	**Instance**	DDMA
Physical Address I/O	0x60B0–0x6950	**Instance**	DDMA
Physical Address	0x4805 28B0−0x4805 29D0	**Instance**	CamDMA
Physical Address	0x4805 60B0−0x4805 6C50	**Instance**	sDMA
Description	Channel destination frame index or 16-bit packet size		
Type	RW		

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1										
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
CH_DST_FRM_IDX_OR_16BIT_PKT_ELNT_NBR																															

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 DMA4_CCR[SEL_SRC_DST_SYNC]=0; the bit field [15:0] gives the number of elements in the packet. The field [31:16] is not used for the packet size.	RW	0x--------

Table 10−51. DMA4_CSAC0 – DMA4_CSACn

Address Offset	0x0B4−0x954 in 0x60 byte increments		
Physical Address Memory	IOMA + 0xC0B4−IOMA + 0xC954	**Instance**	DDMA
Physical Address I/O	0x60B4–0x6954	**Instance**	DDMA
Physical Address	0x4805 28B4−0x4805 29D4	**Instance**	CamDMA
Physical Address	0x4805 60B4−0x4805 6C54	**Instance**	sDMA
Description	Channel source address counter		
Type	R		

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1										
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
SRC_ELMNT_ADRS																															

Bits	Field Name	Description	Type	Reset
31:0	SRC_ELMNT_ADRS[1]	Current source address counter value	R	0x--------

[1] Hard-coded to 0x0000 for the CamDMA.

Table 10–52. DMA4_CDAC0 – DMA4_CDACn

Address Offset	0x0B8–0x958 in 0x60 byte increments		
Physical Address Memory	IOMA + 0xC0B8–IOMA + 0xC958	**Instance**	DDMA
Physical Address I/O	0x60B8–0x6958	**Instance**	DDMA
Physical Address	0x4805 28B8–0x4805 29D8	**Instance**	CamDMA
Physical Address	0x4805 60B8–0x4805 6C58	**Instance**	sDMA
Description	Channel destination address counter		
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
DST_ELMNT_ADRS																															

Bits	Field Name	Description	Type	Reset
31:0	DST_ELMNT_ADRS	Current destination address counter value	R	0x--------

Table 10–53. DMA4_CCEN0 – DMA4_CCENn

Address Offset	0x0BC–0x95C in 0x60 byte increments		
Physical Address Memory	IOMA + 0xC0BC–IOMA + 0xC95C	**Instance**	DDMA
Physical Address I/O	0x60BC–0x695C	**Instance**	DDMA
Physical Address	0x4805 28BC–0x4805 29DC	**Instance**	CamDMA
Physical Address	0x4805 60BC–0x4805 6C5C	**Instance**	sDMA
Description	Channel current transferred element number in the current frame		
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								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 10–54. DMA4_CCFN0 – DMA4_CCFNn

Address Offset	0x0C0–0x960 in 0x60 byte increments		
Physical Address Memory	IOMA + 0xC0C0–IOMA + 0xC960	**Instance**	DDMA
Physical Address I/O	0x60C0–0x6960	**Instance**	DDMA
Physical Address	0x4805 28C0–0x4805 29E0	**Instance**	CamDMA
Physical Address	0x4805 60C0–0x4805 6C60	**Instance**	sDMA
Description	Channel current transferred frame number in the current transfer		
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																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 10–55. DMA4_COLOR0 – DMA4_COLORn

Address Offset	0x0C4–0x964 in 0x60 byte increments		
Physical Address Memory	IOMA + 0xC0C4–IOMA + 0xC964	**Instance**	DDMA
Physical Address I/O	0x60C4–0x6964	**Instance**	DDMA
Physical Address	Not supported	**Instance**	CamDMA
Physical Address	0x4805 60C4–0x4805 6C64	**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_OR_SOLID_COLOR_PTRN																							

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[1]	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 and enhance processing when data is packed at the graphic module input. The same reasoning applies for the 16-bit data type.	RW	0x––––––

1) Not supported for CamDMA or DDMA.

Chapter 11

Interrupt Controller

This chapter describes the interrupt controller (INTC) module used in the microprocessor unit (MPU) and digital signal processor (DSP) subsystems of the OMAP2420 device.

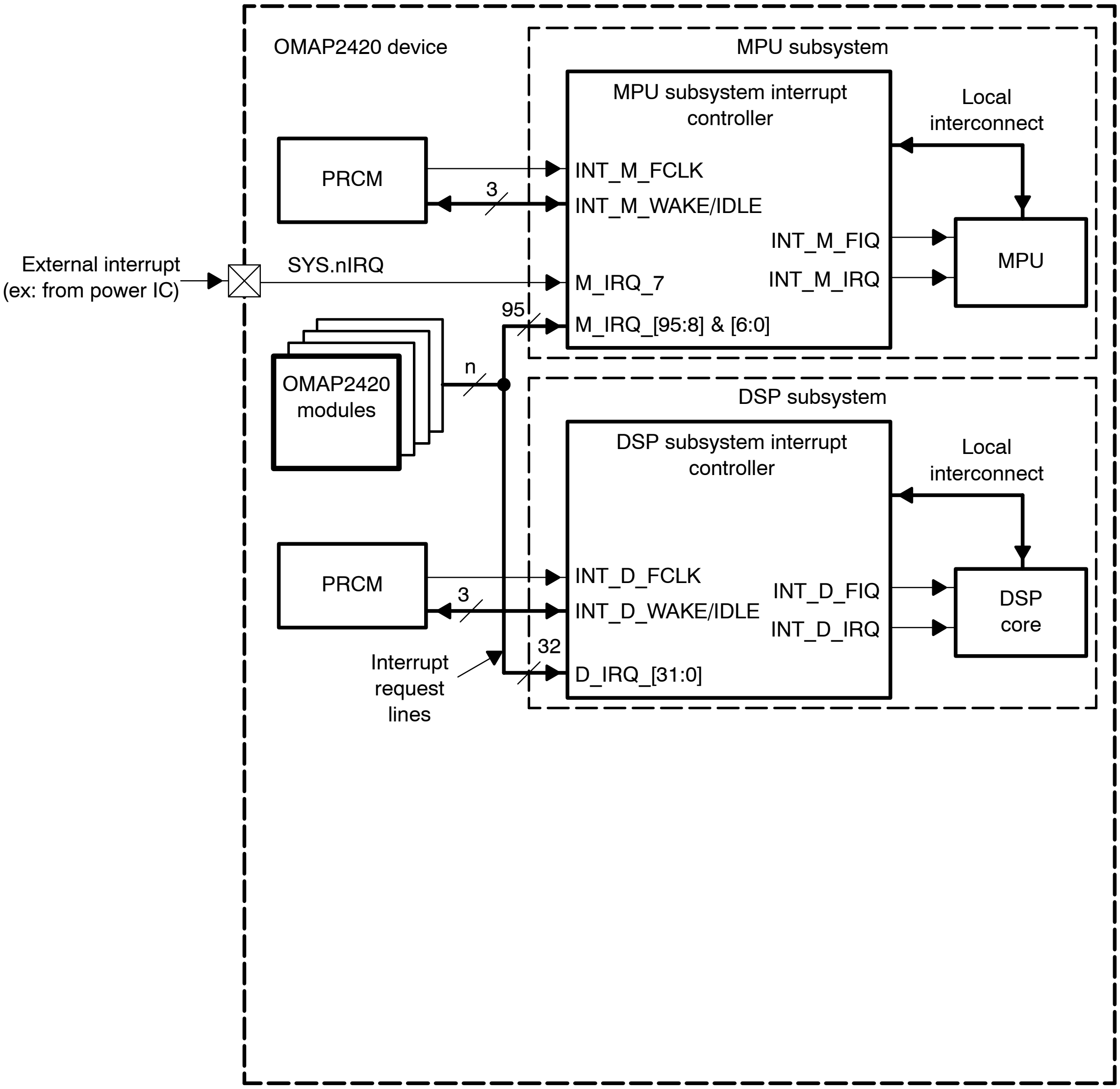
Topic **Page**

11.1 Interrupt Controller Overview

The interrupt controller (INTC) module is a generic module used in the microprocessor unit (MPU) subsystem, the digital signal processor (DSP) subsystem. For details specific to a particular instance of the module, see the MPU subsystem INTC description (Section 11.3.1), DSP subsystem INTC description (Section 11.3.2).

All INTCs use a common programming model. Each INTC communicates to its respective processor through a private local interconnect and runs either at the processor clock frequency or half the processor clock frequency.

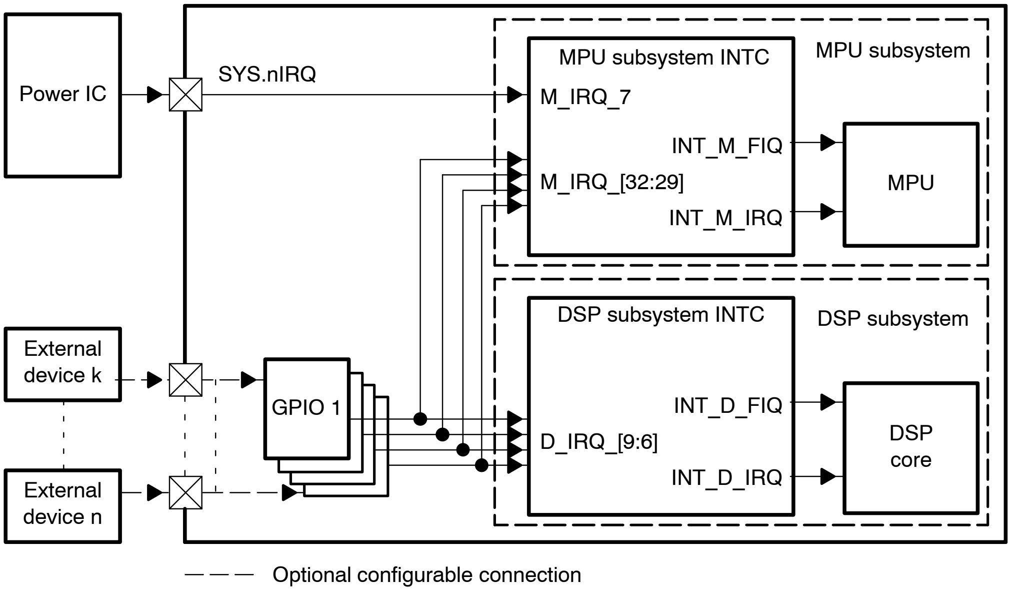
Figure 11−1. Interrupt Controllers in the OMAP2420 System



11.2 Interrupt Controller Environment

The INTC can handle two types of interrupts that originate from external devices (see Figure 11−2).

Figure 11−2. Interrupts From External Devices



One interrupt is provided by an external pin for the MPU subsystem INTC. This external interrupt line is dedicated to a power IC. An interrupt can be generated for any event—a system wake-up event, for a particularly important example.

If this interrupt is masked and the system is idle, the system cannot wake up by this interrupt mechanism. The external interrupt is no different from other interrupt lines; the interrupt is active at low level and is acknowledged by software following the common programming model.

The other method of generating additional external interrupts to the MPU or DSP core is by using general-purpose input/output (GPIO) functions (see Chapter 23, *General−Purpose Interface*).

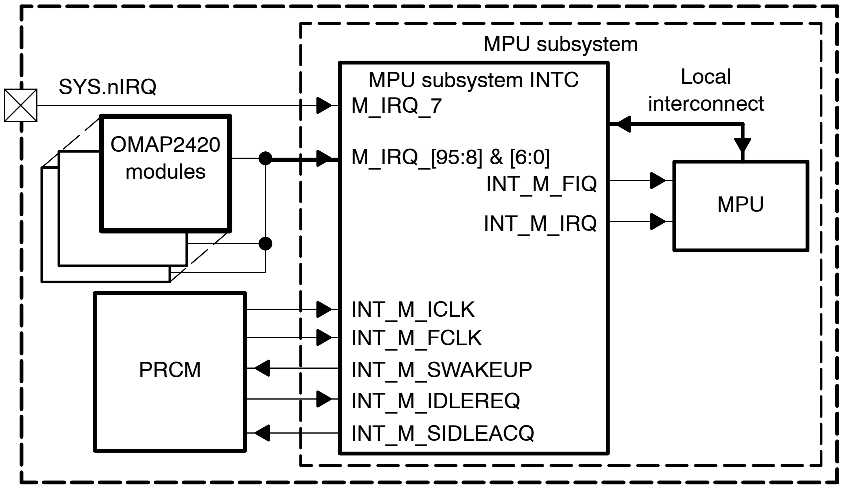
11.3 MPU Subsystem Interrupt Controller Integration

11.3.1 MPU Subsystem Interrupt Controller Description

This module handles up to 96 interrupt requests directed to the MPU subsystem. The requests can be steered/prioritized as MPU FIQ or IRQ interrupt requests.

Figure 11−3 shows the integration of the INTC in the MPU subsystem.

Figure 11−3. MPU Subsystem INTC Integration



The features specific to this instance of the MPU subsystem INTC module are:

- ❏ Up to 96 level-sensitive interrupt inputs
- ❏ Individual priority (up to 64) for each interrupt input
- ❏ 95 incoming interrupt lines connected to internal module interrupts
- ❏ One incoming interrupt line from an external device: Power IC

11.3.1.1 Clocking, Reset, and Power-Management Scheme

This section describes the methodology to address the INTC on the peripheral.

Clocking

The MPU subsystem INTC runs at the MPU clock rate (see Chapter 5, *Power, Reset, and Clock Management*).

The interface clock used for register access runs at L3_CLK frequency (always identical to the MPU interface clock; see Chapter 5, *Power, Reset, and Clock Management*).

Table 11−1 lists the INTC clock rates in the MPU subsystem.

Table 11−1. MPU Subsystem INTC Clock Rates

Clock	Frequency	Name	Comments
Functional	Same as MPU_FCLK	INT_M_FCLK	Source is PRCM module, gating controlled by MPU.
Interface	1,1/2,1/3,1/4 INT_M_FCLK	INT_M_ICLK	Source is PRCM module, gating controlled by MPU.

Hardware Reset

The MPU subsystem INTC receives a reset signal from the power, reset, and clock management (PRCM) module—the CORE_RST signal, which is the reset signal to the CORE power domain (see Chapter 5, *Power, Reset, and Clock Management*).

MPU Subsystem Peripheral	Reset Domain
MPU subsystem INTC	CORE_RST

Software Reset

The INTC_SYSCONFIG[1] bit controls the software reset; setting this bit to 1 triggers a software reset functionally equivalent to a hardware reset. See Table 11−15.

Power Domain

The MPU subsystem INTC connects into the CORE power domain, which can switch dynamically between low voltage and high voltage (see Chapter 5).

MPU Subsystem Peripheral	Power Domain
MPU subsystem INTC	CORE

11.3.1.2 Hardware Requests

Interrupt Request Lines

Table 11−2 lists the incoming and outgoing interrupt lines. Table 11−3 provides the interrupt mapping to the MPU subsystem.

Table 11−2. Interrupt Lines Incoming and Outgoing

Type	Number	Name	Mapping	Comments
Interrupt request inputs	Up to 96	M_IRQ_[95:0]	See Table 11−3.	Inputs to the module; sources come from various modules.
Interrupt request outputs	2	INT_M_FIQ	ARM1136_FIQ	Fast interrupt
		INT_M_IRQ	ARM1136_IRQ	Normal interrupt

CAUTION

An interrupt source can be physically mapped more than once to a single interrupt within the MPU subsystem, DSP subsystem, DSP core. With these shared interrupts, it is important for the application to have only one interrupt unmasked at a time.

Table 11−3. Interrupt Mapping to MPU Subsystem

IRQ	Source	Description
M_IRQ_0	EMUINT	MPU emulation [1]
M_IRQ_1	COMMRX	MPU emulation [1]
M_IRQ_2	COMMTX	MPU emulation [1]
M_IRQ_3	BENCH [#]	MPU emulation [1]
M_IRQ_4	Reserved	
M_IRQ_5	Reserved	
M_IRQ_6	Reserved	
M_IRQ_7	SYS.nIRQ	External interrupt (active low)
M_IRQ_8	Reserved	
M_IRQ_9	Reserved	
M_IRQ_10	L3_IRQ	L3 interconnect (transaction error)
M_IRQ_11	PRCM_MPU_IRQ	PRCM module
M_IRQ_12	SDMA_IRQ0	System DMA interrupt request 0
M_IRQ_13	SDMA_IRQ1	System DMA interrupt request 1
M_IRQ_14	SDMA_IRQ2	System DMA interrupt request 2
M_IRQ_15	SDMA_IRQ3	System DMA interrupt request 3
M_IRQ_16	Reserved	
M_IRQ_17	Reserved	
M_IRQ_18	Reserved	
M_IRQ_19	Reserved	
M_IRQ_20	GPMC_IRQ	General-purpose memory controller module
M_IRQ_21	Reserved	
M_IRQ_22	Reserved	
M_IRQ_23	EAC_IRQ	Audio controller [3]
M_IRQ_24	CAM_IRQ0	Camera interface interrupt request 0
M_IRQ_25	DSS_IRQ	Display subsystem module [3]
M_IRQ_26	MAIL_MPU_IRQ	Mailbox MPU interrupts
M_IRQ_27	Reserved	
M_IRQ_28	DSP_MMU_IRQ	DSP subsystem MMU interrupt
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	Reserved	
M_IRQ_34	MAIL_U3_MPU_IRQ	Mailbox user 3 interrupt request

Notes:
1) These interrupts are internally generated within the MPU subsystem
2) From the ARM1136 performance monitor control register (PMUIRQ)
3) Shared interrupt − also mapped on DSP subsystem INTC
4) Shared interrupt − also mapped on DSP core INTC

Table 11−3. Interrupt Mapping to MPU Subsystem (Continued)

IRQ	Source	Description
M_IRQ_35	WDT3_IRQ	Watchdog timer module 3 overflow
M_IRQ_36	WDT4_IRQ	Watchdog timer module 4 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 [3] [4]
M_IRQ_42	GPT6_IRQ	General-purpose timer module 6 [3] [4]
M_IRQ_43	GPT7_IRQ	General-purpose timer module 7 [3] [4]
M_IRQ_44	GPT8_IRQ	General-purpose timer module 8 [3] [4]
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	GPT12_IRQ	General-purpose timer module 12
M_IRQ_49	Reserved	
M_IRQ_50	Reserved	
M_IRQ_51	Reserved	
M_IRQ_52	Reserved	
M_IRQ_53	Reserved	
M_IRQ_54	Reserved	
M_IRQ_55	Reserved	
M_IRQ_56	I2C1_IRQ	I^{2}C module 1
M_IRQ_57	I2C2_IRQ	I^{2}C module 2
M_IRQ_58	HDQ_IRQ	HDQ/1-Wire
M_IRQ_59	McBSP1_IRQ_TX	McBSP module 1 transmit [3]
M_IRQ_60	McBSP1_IRQ_RX	McBSP module 1 receive [3]
M_IRQ_61	Reserved	[3]
M_IRQ_62	McBSP2_IRQ_TX	McBSP module 2 transmit [3]
M_IRQ_63	McBSP2_IRQ_RX	McBSP module 2 receive [3]
M_IRQ_64	Reserved	[3]
M_IRQ_65	SPI1_IRQ	McSPI module 1
M_IRQ_66	SPI2_IRQ	McSPI module 2
M_IRQ_67	Reserved	
M_IRQ_68	Reserved	
M_IRQ_69	Reserved	

Notes:
1) These interrupts are internally generated within the MPU subsystem
2) From the ARM1136 performance monitor control register (PMUIRQ)
3) Shared interrupt − also mapped on DSP subsystem INTC
4) Shared interrupt − also mapped on DSP core INTC

Table 11–3. Interrupt Mapping to MPU Subsystem (Continued)

IRQ	Source	Description
M_IRQ_70	Reserved	
M_IRQ_71	Reserved	
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) [3]
M_IRQ_75	USB_IRQ_GEN	USB device general interrupt
M_IRQ_76	USB_IRQ_NISO	USB device non-ISO
M_IRQ_77	USB_IRQ_ISO	USB device ISO
M_IRQ_78	USB_IRQ_HGEN	USB host general interrupt
M_IRQ_79	USB_IRQ_HSOF	USB host start of frame
M_IRQ_80	USB_IRQ_OTG	USB OTG
M_IRQ_82	Reserved	
M_IRQ_83	MMC_IRQ	MMC/SD module
M_IRQ_84	MS_IRQ	MS-PRO module
M_IRQ_85	FAC_IRQ	FAC module
M_IRQ_86	Reserved	
M_IRQ_87	Reserved	
M_IRQ_88	Reserved	
M_IRQ_89	Reserved	
M_IRQ_90	Reserved	
M_IRQ_91	Reserved	
M_IRQ_92	Reserved	
M_IRQ_93	Reserved	
M_IRQ_94	Reserved	
M_IRQ_95	Reserved	

Notes:
1) These interrupts are internally generated within the MPU subsystem
2) From the ARM1136 performance monitor control register (PMUIRQ)
3) Shared interrupt – also mapped on DSP subsystem INTC
4) Shared interrupt – also mapped on DSP core INTC

Wake-Up/Idle Requests

See the behavioral description in Section 11.4.4, *System Power Management and Wake-Up*. Table 11−4 lists the power management requests.

Table 11−4. Power Management Requests

Request	Description
INT_M_SWAKEUP	Wake-up request generated by MPU subsystem INTC to PRCM module.
INT_M_IDLEREQ	Idle request from PRCM to MPU subsystem INTC; if accepted by MPU subsystem INTC, INT_M_SIDLEACK is sent back to PRCM module.
INT_M_SIDLEACK	Acknowledge for idle request. Send back to PRCM module before the MPU subsystem INTC goes into idle mode.

11.3.1.3 Pin List and Pad Multiplexing With Other Functions

One interrupt is mapped as a direct external signal on the OMAP device; however, additional external interrupts signals can be obtained using GPIO functions. Black cells in the mode columns in Table 11−5 indicate a mode is not used. Gray cells in the mode columns indicate this mode must be selected to obtain the MPU subsystem interface INTC signal (SYS.nIRQ) on the pin. White cells in the mode columns indicate alternate function.

Table 11−5. Pin Multiplexing for SYSn.IRQ

MPU Subsystem INTC	OMAP2420 Device		Description	Alternate Function				
Interface	Pin Name	Ball		Mode0	Mode1	Mode2	Mode3	Mode4
SYS.nIRQ	SYS.nIRQ	W19	External interrupt signal (active low)	▨	■	■	GPIO.60	■

11.3.1.4 MPU Subsystem INTC Register Summary

Table 11−6 lists the MPU subsystem INTC registers and their physical addresses.

> **Note:**
>
> Address mapping of MPU subsystem INTC belongs to L4 interconnect address mapping, even if a local interconnect with MPU is implemented. The only impacts are:
>
> - Only the MPU can address the MPU subsystem INTC registers.
> - No target agent (TA) is available for MPU subsystem INTC (contrary to other modules included in L4 interconnect address space).

Table 11−6. MPU Subsystem INTC Register Summary

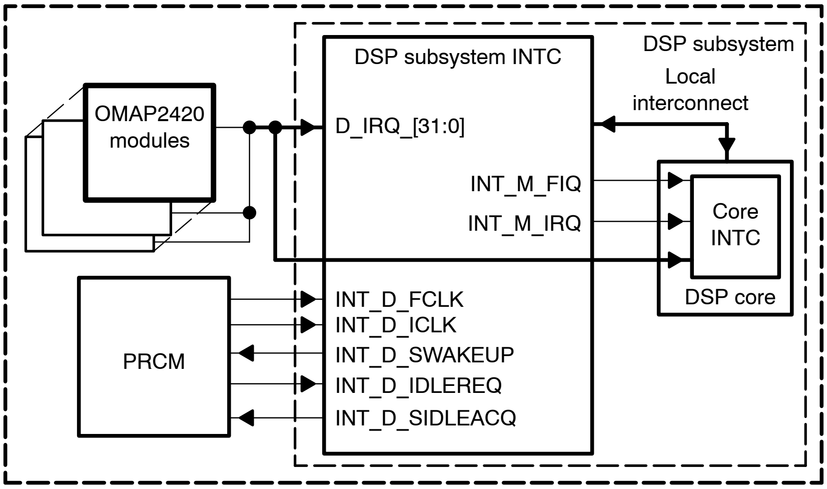
Register Name	Type	Register Width (Bits)	Physical Address
INTC_REVISION	R	32	0x480F E000
INTC_SYSCONFIG	RW	32	0x480F E010
INTC_SYSSTATUS	R	32	0x480F E014
INTC_SIR_IRQ	R	32	0x480F E040
INTC_SIR_FIQ	R	32	0x480F E044
INTC_CONTROL	RW	32	0x480F E048
INTC_PROTECTION	RW	32	0x480F E04C
INTC_IDLE	RW	32	0x480F E050
INTC_ITR0	R	32	0x480F E080
INTC_MIR0	RW	32	0x480F E084
INTC_MIR_CLEAR0	RW	32	0x480F E088
INTC_MIR_SET0	RW	32	0x480F E08C
INTC_ISR_SET0	RW	32	0x480F E090
INTC_ISR_CLEAR0	RW	32	0x480F E094
INTC_PENDING_IRQ0	R	32	0x480F E098
INTC_PENDING_FIQ0	R	32	0x480F E09C
INTC_ITR1	R	32	0x480F E0A0
INTC_MIR1	RW	32	0x480F E0A4
INTC_MIR_CLEAR1	RW	32	0x480F E0A8
INTC_MIR_SET1	RW	32	0x480F E0AC
INTC_ISR_SET1	RW	32	0x480F E0B0
INTC_ISR_CLEAR1	RW	32	0x480F E0B4
INTC_PENDING_IRQ1	R	32	0x480F E0B8
INTC_PENDING_FIQ1	R	32	0x480F E0BC
INTC_ITR2	R	32	0x480F E0C0
INTC_MIR2	RW	32	0x480F E0C4
INTC_MIR_CLEAR2	RW	32	0x480F E0C8
INTC_MIR_SET2	RW	32	0x480F E0CC
INTC_ISR_SET2	RW	32	0x480F E0D0
INTC_ISR_CLEAR2	RW	32	0x480F E0D4
INTC_PENDING_IRQ2	R	32	0x480F E0D8
INTC_PENDING_FIQ2	R	32	0x480F E0DC
INTC_ILR0−INTC_ILR31	RW	32	0x480F E100−0x480F E17C
INTC_ILR32−INTC_ILR63	RW	32	0x480F E180−0x480F E1FC
INTC_ILR64−INTC_ILR95	RW	32	0x480F E200−0x480F E27C

11.3.2 DSP Subsystem Interrupt Controller Description

This module is intended to handle the interrupts directed to the DSP subsystem. It can handle up to 32 requests, which can be steered/prioritized as DSP core INTC FIQ or IRQ interrupt requests.

Figure 11−4 shows the integration of the INTC in the DSP subsystem.

Figure 11−4. DSP Subsystem INTC Integration



The features specific to this instance of the INTC module are:

- Up to 32 level-sensitive interrupt inputs
- Individual priority (up to 32) for each interrupt input

The DSP core contains its own INTC, the core INTC, which uses its own programming model. For more information on the core INTC, see Chapter 4, *DSP Subsystem*.

The same interrupt source can be physically mapped on both the DSP core INTC and the DSP subsystem INTC. See Table 11−9.

11.3.2.1 Clocking, Reset, and Power-Management Scheme

Clocking

The DSP subsystem INTC runs at the DSP core clock (DSP_FCLK) rate (see Chapter 5, *Power, Reset, and Clock Management*).

The interface clock used for register access runs at DSPSS_I_CLK frequency (always identical to the DSP interface clock—see Chapter 5, *Power, Reset, and Clock Management*).

CAUTION

When the ratio between the L3 interconnect clock (CORE_L3_ICLK) and the DSP subsystem INTC interface clock (INT_D_ICLK) is not 1, a synchronizer must be enabled in the PRCM module by setting the SYNCHRO_DSP bit (CM_CLKSEL_DSP[7]) to 1.

See Chapter 5, *Power, Reset, and Clock Management*.

Table 11−7 lists the clock rates in the DSP subsystem INTC.

Table 11−7. DSP Subsystem INTC Clock Rates

Clock	Frequency	Name	Comments
Functional	Same as DSP_FCLK	INT_D_FCLK	Source is PRCM module, gating controlled by DSP.
Interface	1,1/2 INT_D_FCLK = DSP_ICLK	INT_D_ICLK	Source is PRCM module, gating controlled by DSP.

Hardware Reset

The DSP subsystem INTC receives a reset signal from the PRCM module—the DINT_RST signal, which is a dedicated reset signal to the DSP subsystem INTC module. See Chapter 5, *Power, Reset, and Clock Management,* for details.

DSP Subsystem Peripheral	Reset Domain
DSP subsystem INTC	CORE_RST

Software Reset

The INTC_SYSCONFIG[1] bit controls the software reset; setting this bit to 1 triggers a software reset functionally equivalent to a hardware reset. See Table 11−3**.**

Power Domain

The DSP subsystem INTC connects to the CORE power domain, which can dynamically switch between low voltage and high voltage. See Section 5.6.1.3, *Voltage Scaling*.

DSP Subsystem Peripheral	Power Domain
DSP subsystem INTC	CORE

11.3.2.2 Hardware Requests

Interrupt Request Lines

Table 11−8 lists the incoming and outgoing interrupt lines. Table 11−9 provides the interrupt mapping to the DSP subsystem.

Table 11−8. Interrupt Lines Incoming and Outgoing

Type	Number	Name	Mapping	Comments
Interrupt request inputs	Up to 32	D_IRQ_[31:0]	See Table 11−6.	Inputs to the module, sources come from various module.
Interrupt request outputs	2	INT_D_FIQ	DSP_CORE_FIQ	Fast interrupt
		INT_D_IRQ	DSP_CORE_IRQ	Normal interrupt

CAUTION

An interrupt source can be physically mapped more than once to a single interrupt within the MPU subsystem, DSP subsystem, or DSP core INTCs. With these shared interrupts, it is important for the application to have only one interrupt unmasked at a time.

CAUTION

The DSP core INTC is on the DSP power domain. An interrupt event to the DSP core INTC cannot wake up the DSP from inactive, retention, or off power state. If wake-up capability is required, use the DSP subsystem INTC.

Table 11−9. Interrupt Mapping to DSP Subsystem

IRQ	Source	Description
D_IRQ_0	Reserved	
D_IRQ_1	PRCM_DSP_IRQ	PRCM module
D_IRQ_2	DDMA_IRQ0	DSP DMA interrupt request 0 [1,2]
D_IRQ_3	DDMA_IRQ1	DSP DMA interrupt request 1 [1,2]
D_IRQ_4	DDMA_IRQ2	DSP DMA interrupt request 2 [1,2]
D_IRQ_5	DDMA_IRQ3	DSP DMA interrupt request 3 [1,2]
D_IRQ_6	GPIO1_DSP_IRQ	GPIO module 1
D_IRQ_7	GPIO2_DSP_IRQ	GPIO module 2
D_IRQ_8	GPIO3_DSP_IRQ	GPIO module 3
D_IRQ_9	GPIO4_DSP_IRQ	GPIO module 4
D_IRQ_10	GPT5_IRQ	General-purpose timer module 5 [1,3]
D_IRQ_11	GPT6_IRQ	General-purpose timer module 6 [1,3]
D_IRQ_12	GPT7_IRQ	General-purpose timer module 7 [3]
D_IRQ_13	GPT8_IRQ	General-purpose timer module 8 [3]
D_IRQ_14	MAIL_DSP_IRQ	Mailbox DSP interrupts [1]
D_IRQ_15	EAC_IRQ	Audio controller [3]
D_IRQ_16	McBSP1_IRQ_TX	McBSP module 1 transmit [3]
D_IRQ_17	McBSP1_IRQ_RX	McBSP module 1 receive [3]
D_IRQ_18	Reserved	[3]
D_IRQ_19	McBSP2_IRQ_TX	McBSP module 2 transmit [3]
D_IRQ_20	McBSP2_IRQ_RX	McBSP module 2 receive [3]
D_IRQ_21	Reserved	[3]
D_IRQ_22	UART3_IRQ	UART module 3 (also infrared) [3]
D_IRQ_23	Reserved	

Notes:
1) Shared interrupt − also mapped on DSP core INTC
2) These interrupts are internally generated in the DSP subsystem.
3) Shared interrupt − also mapped on MPU

Table 11−9. Interrupt Mapping to DSP Subsystem (Continued)

IRQ	Source	Description
D_IRQ_24	Reserved	
D_IRQ_25	Reserved	
D_IRQ_26	Reserved	
D_IRQ_27	Reserved	
D_IRQ_28	Reserved	
D_IRQ_29	CAM_IRQ1	Camera module interrupt request 1
D_IRQ_30	DSS_IRQ	Display subsystem module [3]
D_IRQ_31	Reserved	

Notes: 1) Shared interrupt − also mapped on DSP core INTC

2) These interrupts are internally generated in the DSP subsystem.

3) Shared interrupt − also mapped on MPU

Wake-Up Requests

See the behavioral description in Section 11.4.4, *System Power Management and Wake-Up*.

Table 11−10 lists the power management requests.

Table 11−10. Power Management Requests

Request	Description
INT_D_SWAKEUP	Wake-up request generated by DSP subsystem INTC to PRCM module
INT_D_IDLEREQ	Idle request from PRCM to DSP subsystem INTC; if accepted by DSP subsystem INTC, INT_D_SIDLEACK is sent back to PRCM module.
INT_D_SIDLEACK	Acknowledge for idle request. Send back to PRCM module before DSP subsystem INTC goes into idle mode.

11.3.2.3 DSP Subsystem INTC Register Summary

Table 11−11 shows the DSP subsystem INTC registers and their physical addresses. For a detailed description, see Section 11.6, *Registers*. The physical addresses given in Table 11−11 are byte addresses; divide by 2 to get DSP half-word addresses.

Note:

It is possible to access the registers by addressing memory space or I/O space (see Chapter 4, *DSP Subsystem*).

Table 11−11. DSP Subsystem INTC Register Summary

Register Name	Type	Register Width (Bits)	Physical Address Memory Space	Physical Address I/O Space
INTC_REVISION	R	32	IOMA + 0x9000	0x4800
INTC_SYSCONFIG	RW	32	IOMA + 0x9010	0x4810

Table 11−11. DSP Subsystem INTC Register Summary (Continued)

Register Name	Type	Register Width (Bits)	Physical Address Memory Space	Physical Address I/O Space
INTC_SYSSTATUS	R	32	IOMA + 0x9014	0x4814
INTC_SIR_IRQ	R	32	IOMA + 0x9040	0x4840
INTC_SIR_FIQ	R	32	IOMA + 0x9044	0x4844
INTC_CONTROL	RW	32	IOMA + 0x9048	0x4848
INTC_PROTECTION	RW	32	IOMA + 0x904C	0x484C
INTC_IDLE	RW	32	IOMA + 0x9050	0x4850
INTC_ITR0	R	32	IOMA + 0x9080	0x4880
INTC_MIR0	RW	32	IOMA + 0x9084	0x4884
INTC_MIR_CLEAR0	RW	32	IOMA + 0x9088	0x4888
INTC_MIR_SET0	RW	32	IOMA + 0x908C	0x488C
INTC_ISR_SET0	RW	32	IOMA + 0x9090	0x4890
INTC_ISR_CLEAR0	RW	32	IOMA + 0x9094	0x4894
INTC_PENDING_IRQ0	R	32	IOMA + 0x9098	0x4898
INTC_PENDING_FIQ0	R	32	IOMA + 0x909C	0x489C
INTC_ILR0−INTC_ILR31	RW	32	IOMA + 0x9100− IOMA + 0x917C	0x2500− 0x257C

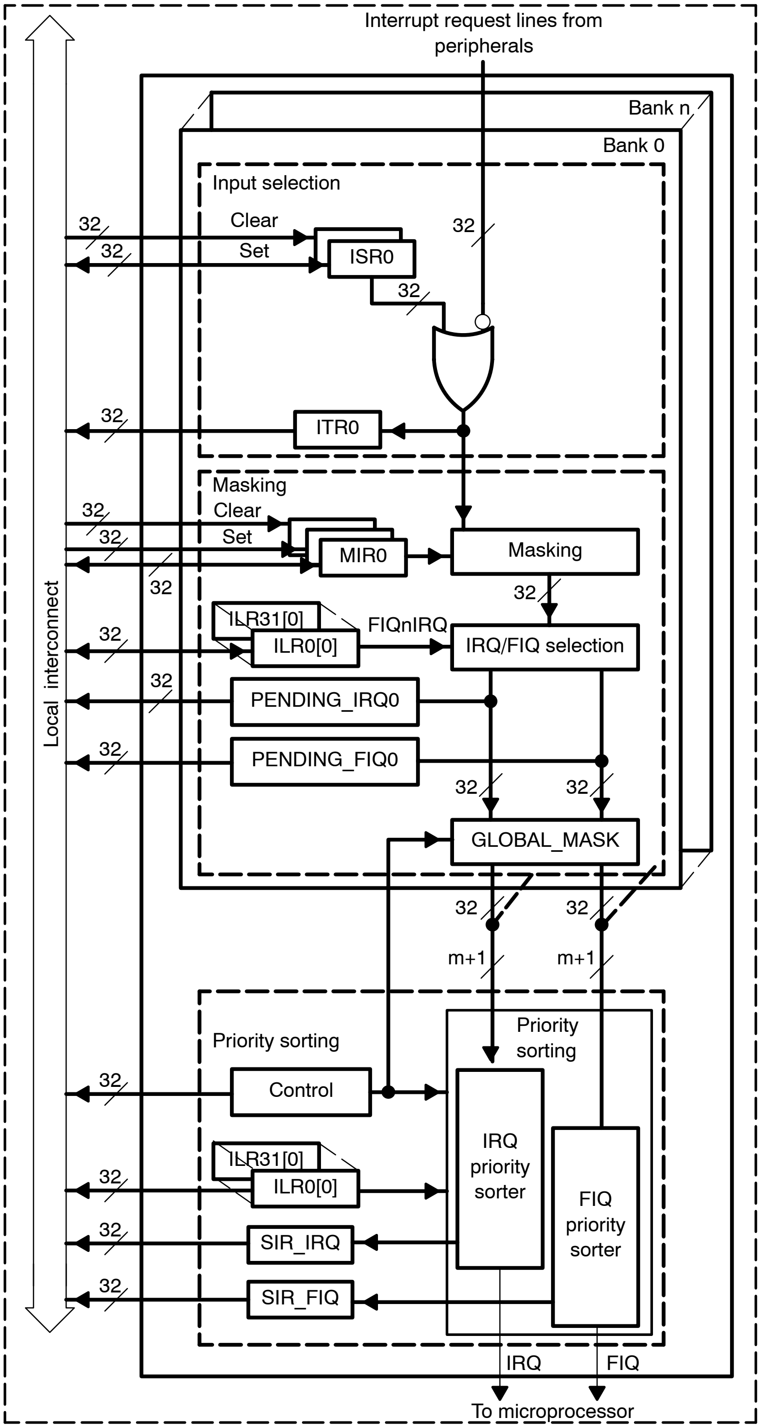
11.4 Interrupt Controller Functional Description

The main features of the INTC module are:

- Individual priority 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
- Atomic bit set and clear capability for interrupt mask and software interrupt registers
- Global interrupt mask
- Power-management and wake-up support
- Auto-idle power-saving support

The INTC processes incoming interrupts by masking and priority sorting, and it generates the interrupt requests to the processor to which it is attached.

Figure 11–5. Top-Level Block Diagram



n between 0 and 2

m = 32.(n+1)-1: number of incoming interrupts -1

11.4.1 Interrupt Processing

11.4.1.1 Input Selection

The INTC supports level-sensitive incoming interrupt detection only. A peripheral asserting an interrupt maintains it until software handles the interrupt and instructs the peripheral to deassert the interrupt.

A software interrupt is generated if the corresponding bit in the INTC_ISR*n* register via the INTC_ISR_SET*n* register is set. The software interrupt is cleared when the corresponding bit in INTC_ISR_CLEAR*n* register is written. Typical use of this feature is software debug.

11.4.1.2 Masking

Detection of interrupts on each incoming interrupt line can be enabled or disabled independently using the interrupt mask register, INTC_MIR*n* (*n* may be 0, 1, or 2, depending on the number of incoming interrupt lines supported).

In response to an unmasked incoming interrupt, the INTC 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 relevant field in the INTC_ILR*m* register (*m* is the number of the incoming interrupt line).

The current incoming interrupt status before masking is readable from the INTC_ITR*n* register. The interrupt status after masking and IRQ/FIQ selection is readable from the INTC_PENDING_IRQ*n* and INTC_PENDING_FIQ*n* registers.

In addition, it is possible to globally mask incoming interrupt detection through a bit in the INTC_CONTROL register. In this state, further incoming interrupts cause the INTC_ITR*n*, INTC_PENDING_IRQ*n,* and INTC_PENDING_FIQ*n* registers to be updated as normal; however, no interrupt events are passed to the priority sorting stage. Any currently asserted IRQ or FIQ interrupt request due to a previously occurring interrupt remains asserted until the interrupt is acknowledged by writing to the relevant bit in the INTC_CONTROL register. Additionally, any interrupt that has already entered the priority sorting stage when the global mask is asserted results in an IRQ or FIQ interrupt request being asserted. This interrupt request remains asserted until it is similarly acknowledged.

Incoming interrupt detection is unmasked again by writing to the appropriate bit in the INTC_CONTROL register.

11.4.1.3 Priority Sorting

Each incoming interrupt line has a priority level assigned to it (0 being the highest). Both the priority level and the interrupt request type are configured using the INTC_ILR*m* register. If more than one incoming interrupt with the same priority level and interrupt request type occurs simultaneously, the highest numbered interrupt is serviced first.

When one or more unmasked incoming interrupts are detected, the interrupt request type is determined from the contents of the corresponding INTC_ILR*m* register. If no other interrupt is currently being processed for this type of interrupt request, one of two priority sorters determines the highest priority incoming interrupt number and puts this in the INTC_SIR_IRQ or INTC_SIR_FIQ register. The priority sorters for FIQ and IRQ are independent and can execute in parallel if interrupts assigned to IRQ and FIQ occur close together.

Once the interrupting peripheral device is serviced and the incoming interrupt is deasserted, the appropriate bit in the INTC_CONTROL register must be written to indicate to the INTC that the interrupt has been handled. If there are any pending unmasked incoming interrupts for this interrupt request type, then the INTC restarts the appropriate priority sorter; otherwise, the IRQ or FIQ interrupt line is deasserted.

To minimize the interrupt latency when an unmasked interrupt occurs, the IRQ or FIQ interrupt request is generated in advance of the priority sorting completion. The IRQ/FIQ interrupt generation takes three INTC functional clock cycles, if the TURBO bit of the INTC_IDLE register is set to 0, or five INTC functional clock cycles, if the TURBO bit of the INTC_IDLE register is set to 1 (but the power consumption is reduced while waiting for an interrupt).

The priority sorting takes 10 functional clock cycles, which is likely to be less than the minimum number of cycles required for the MPU to switch to the interrupt context from reception of the IRQ or FIQ event.

Note:

Masking an interrupt on-the-fly while the sorting engine is in the wait-for-interrupt state and the MPU interrupt flag is clear (FIR or IRQ enabled) can result in a spurious interrupt, if the associated interrupt line is asserted in a 10-cycle window ahead of the masking access.

The spurious interrupt occurs because of the multicycle sorting process and because IRQ or FIQ interrupt requests are generated in advance. The masking can affect the priority sorting result so that the INTC_SIR_IRQ or INTC_SIR_FIQ register content may not be valid.

If the interrupt assertion cannot be controlled at the source, masking in the INTC should be applied either when the sorting is stopped or under the MPU interrupt masking period followed by a NEWIRQAGR or NEWFIQAGR bit setting in the INTC_CONTROL register to release any spurious event due to the associated interrupt line assertion.

Any read of the INTC_SIR_IRQ or INTC_SIR_FIQ registers during the priority sorting process is stalled until it is complete and the relevant register has been updated. However, in practice, the delay between the interrupt request being generated and the interrupt service routine being executed is such that priority sorting is complete before the INTC_SIR_IRQ or INTC_SIR_FIQ registers are read.

For information on the functional clock frequency of the specific instance of the INTC module, see Section 11.3.1.1, *Clocking, Reset, and Power-Management Scheme*.

11.4.2 Register Protection

Access to the INTC module registers is restricted to the host processor privileged (supervisor) mode if the corresponding bit in the INTC_PROTECTION register is set. Access to the INTC_PROTECTION register is always restricted to privileged mode. For more details about the host processor privileged mode (if supported), see Chapter 3, Chapter 4.

11.4.3 Module Power Saving

The INTC module provides an auto-idle function in its three clock domains:

- ☐ Resynchronization block clock
- ☐ Interconnect clock
- ☐ Functional clock

The resynchronization block clock permits resynchronizing external asynchronous interrupts before masking these interrupts. The input synchronizers have an auto-idle power-saving mode enabled or disabled through the TURBO bit of the INTC_IDLE register. The TURBO bit field allows the choice of latency versus standby power. At reset, the auto-idle is disabled and functional clock gating is applied. If the auto-idle is enabled, the standby power is reduced, but the IRQ or FIQ interrupt latency increases from three to five functional clock cycles. This enable can be changed dynamically according to the current needs of the device.

The interconnect clock auto-idle power-saving mode is enabled or disabled through the AUTOIDLE bit of the INTC_SYSCONFIG register. When this mode is enabled and there is no activity on the interconnect interface, the interconnect clock is disabled internally to the module, thus reducing 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. For reduced power consumption, it is recommended to enable this mode.

The functional clock auto-idle power-saving mode is enabled or disabled through the FUNCIDLE bit of the INTC_IDLE register. When this mode is enabled and there are no active (that is, being processed or generating an IRQ or FIQ interrupt) or pending incoming interrupts, the functional clock is disabled internally to the module, thus reducing power consumption.

When a new unmasked incoming interrupt is detected, the functional clock is restarted and the INTC processes the interrupt as usual without any latency penalty. After reset, this mode is enabled by default.

11.4.4 System Power Management and Wake-Up

As part of the system-wide power-management scheme, the INTC module can go into idle state at the request of the PRCM module (for details, see Chapter 5. The INTC module is always in smart-idle mode (that is, it goes into idle state after it receives the request from the PRCM module, once all asserted unmasked incoming interrupts are acknowledged, and neither IRQ nor FIQ interrupt requests are being generated). This functionality is handled by hardware. The INTC returns an acknowledgement to the PRCM module when the idle request is accepted.

When an unmasked incoming interrupt occurs while in idle state, the INTC forwards a wake-up signal to the PRCM module, which wakes up the system in an orderly fashion. This includes bringing the INTC out of idle state, at which point the incoming interrupt is processed as normal.

Note:

Other higher priority interrupts may occur before the INTC is out of idle state, in which case the interrupt number read by software from the INTC_SIR_IRQ or INTC_SIR_FIQ register may not be the one that caused the system to wake up.

11.5 Programming Model

11.5.1 Initialization Sequence

1) Set the INTC_ILR*m* registers for each interrupt line: Select FIQ or IRQ, assign priority level.

2) Set INTC_MIR*n* register: Enable interrupts (by default all interrupt lines are masked).

Note:

Texas Instruments recommends using writes to the INTC_MIR_SET*n* and INTC_MIR_CLEAR*n* registers to program INTC_MIR*n*, even if it is possible for backwards compatibility to write directly to the INTC_MIR*n* register.

11.5.2 Interrupt Processing Sequence

After the INTC_MIR*n* and INTC_ILR*m* registers are configured to enable and assign priorities to incoming interrupts, the interrupt processing sequence is as follows:

Step 1: One or more unmasked incoming interrupts are received.

Step 2: Depending on the setting of the FIQnIRQ bit in the INTC_ILR*n* register corresponding to the incoming interrupt, either an IRQ or FIQ is generated.

Step 3: The INTC performs the priority sorting and updates the INTC_SIR_IRQ or INTC_SIR_FIQ register with the current interrupt number.

Step 4: When the host processor recognizes the IRQ or FIQ, software execution jumps to the interrupt service routine (ISR) that determines the source of the interrupt by reading the INTC_SIR_IRQ or INTC_SIR_FIQ register. It executes code specific to the peripheral generating the interrupt, handling the event and deasserting the interrupt condition at the peripheral.

Step 5: Software sets either the NEWIRQAGR or the NEWFIQAGR bit in the INTC_CONTROL register.

Step 6: The INTC processes any other pending interrupts or deasserts IRQ/FIQ if there are no interrupts.

Note:

The priority sorting mechanism is frozen during an interrupt processing sequence. If an interrupt condition occurs during this time, the interrupt is not lost and is sorted once NEWIRQAGR/NEWFIQAGR is set (priority sorting is reactivated).

The priority level can be changed at any time; the following procedure is recommended:

1) Mask all interrupts by writing all bits at 1 in the INTC_MIR_SET*n* register.

2) Read INTC_SIR_IRQ and INTC_SIR_FIQ.

3) Program INTC_ILR_m with a new value.

4) Unmask the interrupts writing in the INTC_MIR_CLEAR*n* register.

11.6 Registers

11.6.1 Instance Summary

Table 11−12 shows the base address and address space for the INTC module instances.

Table 11−12. INTC Instance Summary

Module Name	Base Address	Size
MPU_SS_INTC	0x480F E000	4K bytes
DSP_SS_INTC	IOMA + 0x9000 or 0x2400†	2K bytes

† DSP memory space mapping or DSP I/O space mapping

11.6.2 Register Summary

This section provides information on the INTC module instance within the OMAP2420. Table 11−14 through Table 11−46 describe each register within the module instance. Some registers exist only in the MPU subsystem INTC instance. For more information, see Section 11.3, *MPU Subsystem Interrupt Controller Integration*.

It is possible to access these registers by addressing memory space or I/O space. For information on DSP subsystem INTC addressing, see Chapter 4, *DSP Subsystems*. The physical addresses provided are byte addresses; divide by 2 to get DSP half-word addresses. IOMA value is fixed by DSP BIOS. See Chapter 4, *DSP Subsystems,* for details.

In Table 11−13, the INTC_INTRO to INTC_PENDING_FIQ2 registers contain 32 bits, or 1 bit for each interrupt. The interrupts are assigned in ascending order, starting at bit 0 from INTC_ITR0 to INTC_PENDING_FIQ0 and continuing successively from INTC_ITR1 to INTC_PENDING_FIQ1 (and from INTC_ITR2 to INTC_PENDING_FIQ2, for the 96 interrupt cases).

Table 11−13. INTC Registers Summary

Register Name	Type	Offset Address	MPU_SS_ INTC Physical Address	DSP_SS_ INTC Physical Address Memory Space	DSP_SS_ INTC Physical Address I/O Space
INTC_REVISION	R	0x 0000	0x480F E000	IOMA + 0x9000	0x4800
INTC_SYSCONFIG	RW	0x 0010	0x480F E010	IOMA + 0x9010	0x4810
INTC_SYSSTATUS	R	0x 0014	0x480F E014	IOMA + 0x9014	0x4814
INTC_SIR_IRQ	R	0x 0040	0x480F E040	IOMA + 0x9040	0x4840
INTC_SIR_FIQ	R	0x 0044	0x480F E044	IOMA + 0x9044	0x4844
INTC_CONTROL	RW	0x 0048	0x480F E048	IOMA + 0x9048	0x4848
INTC_PROTECTION	RW	0x 004C	0x480F E04C	IOMA + 0x904C	0x484C
INTC_IDLE	RW	0x 0050	0x480F E050	IOMA + 0x9050	0x4850
INTC_ITR0	R	0x 0080	0x480F E080	IOMA + 0x9080	0x4880
INTC_MIR0	RW	0x 0084	0x480F E084	IOMA + 0x9084	0x4884

Table 11−13. INTC Registers Summary (Continued)

Register Name	Type	Offset Address	MPU_SS_ INTC Physical Address	DSP_SS_ INTC Physical Address Memory Space	DSP_SS_ INTC Physical Address I/O Space
INTC_MIR_CLEAR0	RW	0x 0088	0x480F E088	IOMA + 0x9088	0x4888
INTC_MIR_SET0	RW	0x 008C	0x480F E08C	IOMA + 0x908C	0x488C
INTC_ISR_SET0	RW	0x 0090	0x480F E090	IOMA + 0x9090	0x4890
INTC_ISR_CLEAR0	RW	0x 0094	0x480F E094	IOMA + 0x9094	0x4894
INTC_PENDING_IRQ0	R	0x 0098	0x480F E098	IOMA + 0x9098	0x4898
INTC_PENDING_FIQ0	R	0x 009C	0x480F E09C	IOMA + 0x909C	0x489C
INTC_ITR1	R	0x 00A0	0x480F E0A0	NA	NA
INTC_MIR1	RW	0x 00A4	0x480F E0A4	NA	NA
INTC_MIR_CLEAR1	RW	0x 00A8	0x480F E0A8	NA	NA
INTC_MIR_SET1	RW	0x 00AC	0x480F E0AC	NA	NA
INTC_ISR_SET1	RW	0x 00B0	0x480F E0B0	NA	NA
INTC_ISR_CLEAR1	RW	0x 00B4	0x480F E0B4	NA	NA
INTC_PENDING_IRQ1	R	0x 00B8	0x480F E0B8	NA	NA
INTC_PENDING_FIQ1	R	0x 00BC	0x480F E0BC	NA	NA
INTC_ITR2	R	0x 00C0	0x480F E0C0	NA	NA
INTC_MIR2	RW	0x 00C4	0x480F E0C4	NA	NA
INTC_MIR_CLEAR2	RW	0x 00C8	0x480F E0C8	NA	NA
INTC_MIR_SET2	RW	0x 00CC	0x480F E0CC	NA	NA
INTC_ISR_SET2	RW	0x 00D0	0x480F E0D0	NA	NA
INTC_ISR_CLEAR2	RW	0x 00D4	0x480F E0D4	NA	NA
INTC_PENDING_IRQ2	R	0x 00D8	0x480F E0D8	NA	NA
INTC_PENDING_FIQ2	R	0x 00DC	0x480F E0DC	NA	NA
INTC_ILR0−INTC_ILR31	RW	0x 0100− 0x 017C	0x480F E100− 0x480F E17C	IOMA + x9100− IOMA + 0x917C	0x5000− 0x507C
INTC_ILR32−INTC_ILR63	RW	0x 0180− 0x 01FC	0x480F E180− 0x480F E1FC	NA	NA
INTC_ILR64−INTC_ILR95	RW	0x 0200− 0x 027C	0x480F E200− 0x480F E27C	NA	NA

11.6.3 Register Descriptions

Table 11−14. INTC_REVISION

Address Offset	0x000		
Physical Address	0x480FE000	**Instance**	MPU_SS_INTC
Physical Address Memory	IOMA + 0x9000	**Instance**	DSP_SS_INTC
Physical Address I/O	0x4800	**Instance**	DSP_SS_INTC
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 Examples: 0x10 for 1.0, 0x21 for 2.1	R	

Table 11−15. INTC_SYSCONFIG

Address Offset	0x010		
Physical Address	0x480FE010	**Instance**	MPU_SS_INTC
Physical Address Memory	IOMA + 0x9010	**Instance**	DSP_SS_INTC
Physical Address I/O	0x4810	**Instance**	DSP_SS_INTC
Description	This register controls the various parameters of the OCP 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

Table 11−15. INTC_SYSCONFIG (Continued)

Bits	Field Name	Description	Type	Reset
31:2	Reserved	Write 0s for future compatibility. Read returns 0.	R	0x00000000
1	SOFTRESET	Software reset. Set this bit to trigger a module reset. The bit is automatically reset by the hardware. During reads, it always returns 0. Write 0x0: No functional effect Write 0x1: The module is reset.	RW	0
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	0

Table 11−16. INTC_SYSSTATUS

Address Offset	0x014		
Physical Address	0x480FE014	**Instance**	MPU_SS_INTC
Physical Address Memory	IOMA + 0x9014	**Instance**	DSP_SS_INTC
Physical Address I/O	0x4814	**Instance**	DSP_SS_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	Reserved for module-specific status information. Read returns 0.	R	0x00000000
0	RESETDONE	Internal reset monitoring 0x0: Internal module reset is ongoing. 0x1: Reset completed[†]	R	0

[†] During reset the value is 0, but the value read just after the reset is 1 because ICLK/FCLK is already operating.

Table 11−17. INTC_SIR_IRQ

Address Offset	0x040		
Physical Address	0x480FE040	**Instance**	MPU_SS_INTC
Physical Address Memory	IOMA + 0x9040	**Instance**	DSP_SS_INTC
Physical Address I/O	0x4840	**Instance**	DSP_SS_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
Reserved																									ACTIVEIRQ						

Bits	Field Name	Description	Type	Reset
31:7	Reserved	Reads returns 0	R	0x0000000
6:0	ACTIVEIRQ	Active IRQ number	R	0x00

Table 11−18. INTC_SIR_FIQ

Address Offset	0x044		
Physical Address	0x480FE044	**Instance**	MPU_SS_INTC
Physical Address Memory	IOMA + 0x9044	**Instance**	DSP_SS_INTC
Physical Address I/O	0x4844	**Instance**	DSP_SS_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
Reserved																									ACTIVEFIQ						

Bits	Field Name	Description	Type	Reset
31:7	Reserved	Reads returns 0	R	0x0000000
6:0	ACTIVEFIQ	Active FIQ number	R	0x00

Table 11−19. INTC_CONTROL

Address Offset	0x048		
Physical Address	0x480FE048	**Instance**	MPU_SS_INTC
Physical Address Memory	IOMA + 0x9048	**Instance**	DSP_SS_INTC
Physical Address I/O	0x4848	**Instance**	DSP_SS_INTC
Description	This register contains the global mask and new interrupt agreement 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																													GLOBALMASK	NEWFIQAGR	NEWIRQAGR

Bits	Field Name	Description	Type	Reset
31:3	Reserved	Write 0s for future compatibility. Read returns 0.	R	0x00000000
2	GLOBALMASK	Global mask operation 0x0: Normal operation 0x1: All IRQ and FIQ are masked; IRQ_SECURE_MASK_n is asserted if no interrupt is in progress or in INTC pipeline	RW	0
1	NEWFIQAGR	Reset FIQ output and enable new FIQ generation. Write 0x0: No functional effect Write 0x1: Reset FIQ output and enable new FIQ generation Reads always return 0.	RW	0
0	NEWIRQAGR	New IRQ generation Write 0x0: No functional effect Write 0x1: Reset IRQ output and enable new IRQ generation Reads always return 0.	RW	0

Table 11−20. INTC_PROTECTION

Address Offset	0x04C		
Physical Address	0x480FE04C	**Instance**	MPU_SS_INTC
Physical Address Memory	IOMA + 0x904C	**Instance**	DSP_SS_INTC
Physical Address I/O	0x484C	**Instance**	DSP_SS_INTC
Description	This register controls protection of the other registers. This register can only be accessed in privileged mode, regardless of the current value of the protection 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																															PROTECTION

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Write 0s for future compatibility. Read returns 0.	R	0x00000000
0	PROTECTION	Protection mode 0x0: Protection mode disabled (default) 0x1: Protection mode enabled. When enabled, only privileged mode can access registers.	RW	0

Table 11−21. INTC_IDLE

Address Offset	0x050		
Physical Address	0x480FE050	**Instance**	MPU_SS_INTC
Physical Address Memory	0x00FC9050	**Instance**	DSP_SS_INTC
Physical Address I/O	0x4850	**Instance**	DSP_SS_INTC
Description	This register can disable the functional clock auto-idle and input synchronizer clock 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

Table 11–21. INTC_IDLE (Continued)

Bits	Field Name	Description	Type	Reset
31:2	Reserved	Write 0s for future compatibility. Reads return 0.	R	0x00000000
1	TURBO	Input synchronizer clock auto-gating 0x0: Input synchronizer clock is free running (default). 0x1: Input synchronizer clock is auto-gated based on interrupt input activity.	RW	0
0	FUNCIDLE	Functional clock auto-idle mode 0x0: Functional clock-gating strategy is applied (default). 0x1: Functional clock is free-running.	RW	0

Table 11–22. INTC_ITR0

Address Offset	0x080		
Physical Address	0x480FE080	**Instance**	MPU_SS_INTC
Physical Address Memory	IOMA + 0x9080	**Instance**	DSP_SS_INTC
Physical Address I/O	0x4880	**Instance**	DSP_SS_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	0x––––––––

Table 11–23. INTC_MIR0

Address Offset	0x084		
Physical Address	0x480FE084	**Instance**	MPU_SS_INTC
Physical Address Memory	IOMA + 0x9084	**Instance**	DSP_SS_INTC
Physical Address I/O	0x4884	**Instance**	DSP_SS_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	RW	0xFFFFFFFF

Table 11–24. INTC_MIR_CLEAR0

Address Offset	0x088		
Physical Address	0x480FE088	**Instance**	MPU_SS_INTC
Physical Address Memory	IOMA + 0x9088	**Instance**	DSP_SS_INTC
Physical Address I/O	0x4888	**Instance**	DSP_SS_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	Write 1 clears the mask bit to 0, read returns 0.	W	0x--------

Table 11–25. INTC_MIR_SET0

Address Offset	0x08C		
Physical Address	0x480FE08C	**Instance**	MPU_SS_INTC
Physical Address Memory	IOMA + 0x908C	**Instance**	DSP_SS_INTC
Physical Address I/O	0x488C	**Instance**	DSP_SS_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	Write 1 sets the mask bit to 1, read returns 0.	W	0x--------

Table 11–26. INTC_ISR_SET0

Address Offset	0x090		
Physical Address	0x480FE090	**Instance**	MPU_SS_INTC
Physical Address Memory	IOMA + 0x9090	**Instance**	DSP_SS_INTC
Physical Address I/O	0x4890	**Instance**	DSP_SS_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																															

Table 11-30. INTC_ISR_SET0 (Continued)

Bits	Field Name	Description	Type	Reset
31:0	ISRSET	Read returns the currently active software interrupts; write 1 sets the software interrupt bits to 1.	RW	0x00000000

Table 11–27. INTC_ISR_CLEAR0

Address Offset	0x094		
Physical Address	0x480FE094	**Instance**	MPU_SS_INTC
Physical Address Memory	IOMA + 0x9094	**Instance**	DSP_SS_INTC
Physical Address I/O	0x4894	**Instance**	DSP_SS_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	Write 1 clears the software interrupt bits to 0; read returns 0	W	0x––––––––

Table 11–28. INTC_PENDING_IRQ0

Address Offset	0x098		
Physical Address	0x480FE098	**Instance**	MPU_SS_INTC
Physical Address Memory	IOMA + 0x9098	**Instance**	DSP_SS_INTC
Physical Address I/O	0x4898	**Instance**	DSP_SS_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

Table 11−29. INTC_PENDING_FIQ0

Address Offset	0x09C		
Physical Address	0x480FE09C	**Instance**	MPU_SS_INTC
Physical Address Memory	IOMA + 0x909C	**Instance**	DSP_SS_INTC
Physical Address I/O	0x489C	**Instance**	DSP_SS_INTC
Description	This register contains the FIQ status after masking.		
Type	R		

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1										
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	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

Table 11−30. INTC_ITR1

Address Offset	0x0A0		
Physical Address	0x480FE0A0	**Instance**	MPU_SS_INTC
Description	This register shows the raw interrupt input status before masking.		
Type	R		

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1										
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	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	0x−−−−−−−−

Table 11−31. INTC_MIR1

Address Offset	0x0A4		
Physical Address	0x480FE0A4	**Instance**	MPU_SS_INTC
Description	This register contains the interrupt mask.		
Type	RW		

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1										
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
MIR																															

Bits	Field Name	Description	Type	Reset
31:0	MIR	Interrupt mask	RW	0xFFFFFFFF

Table 11–32. INTC_MIR_CLEAR1

Address Offset	0x0A8		
Physical Address	0x480FE0A8	**Instance**	MPU_SS_INTC
Description	This register is used to clear the interrupt mask 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
MIRCLEAR																															

Bits	Field Name	Description	Type	Reset
31:0	MIRCLEAR	Write 1 clears the mask bit to 0; read returns 0.	RW	0x––––––––

Table 11–33. INTC_MIR_SET1

Address Offset	0x0AC		
Physical Address	0x480FE0AC	**Instance**	MPU_SS_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	Write 1 sets the mask bit to 1; read returns 0.	W	0x––––––––

Table 11–34. INTC_ISR_SET1

Address Offset	0x0B0		
Physical Address	0x480FE0B0	**Instance**	MPU_SS_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	Read returns the active software interrupts; write 1 sets the software interrupt bits to 1.	RW	0x00000000

Table 11−35. INTC_ISR_CLEAR1

Address Offset	0x0B4		
Physical Address	0x480FE0B4	**Instance**	MPU_SS_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	Writing 1 clears the software interrupt bits to 0; read returns 0	W	0x--------

Table 11−36. INTC_PENDING_IRQ1

Address Offset	0x0B8		
Physical Address	0x480FE0B8	**Instance**	MPU_SS_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

Table 11−37. INTC_PENDING_FIQ1

Address Offset	0x0BC		
Physical Address	0x480FE0BC	**Instance**	MPU_SS_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

Table 11−38. INTC_ITR2

Address Offset	0x0C0		
Physical Address	0x480FE0C0	**Instance**	MPU_SS_INTC
Description	This register shows the raw interrupt input status before masking.		
Type	R		

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="32">ITR</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
31:0	ITR	Interrupt status before masking	R	0x--------

Table 11−39. INTC_MIR2

Address Offset	0x0C4		
Physical Address	0x480FE0C4	**Instance**	MPU_SS_INTC
Description	This register contains the interrupt mask.		
Type	RW		

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="32">MIR</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
31:0	MIR	Interrupt mask	RW	0xFFFFFFFF

Table 11−40. INTC_MIR_CLEAR2

Address Offset	0x0C8		
Physical Address	0x480FE0C8	**Instance**	MPU_SS_INTC
Description	This register is used to clear the interrupt mask bits.		
Type	W		

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="32">MIRCLEAR</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
31:0	MIRCLEAR	Writing 1 clears the mask bit to 0; read returns 0.	W	0x--------

Table 11−41. INTC_MIR_SET2

Address Offset	0x0CC		
Physical Address	0x480FE0CC	**Instance**	MPU_SS_INTC
Description	This register is used to set the interrupt mask 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
MIRSET																															

Bits	Field Name	Description	Type	Reset
31:0	MIRSET	Writing 1 sets the mask bit to 1; read returns 0.	RW	0x––––––––

Table 11−42. INTC_ISR_SET2

Address Offset	0x0D0		
Physical Address	0x480FE0D0	**Instance**	MPU_SS_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	Read returns the currently active software interrupts; Writing 1 sets the software interrupt bits to 1.	RW	0x00000000

Table 11−43. INTC_ISR_CLEAR2

Address Offset	0x0D4		
Physical Address	0x480FE0D4	**Instance**	MPU_SS_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	Write 1 clears the software interrupt bits to 0; Read returns 0.	W	0x––––––––

Table 11−44. INTC_PENDING_IRQ2

Address Offset	0x0D8		
Physical Address	0x480FE0D8	**Instance**	MPU_SS_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

Table 11−45. INTC_PENDING_FIQ2

Address Offset	0x0DC		
Physical Address	0x480FE0DC	**Instance**	MPU_SS_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

Table 11−46. INTC_ILR0−INTC_ILR31/INTC_ILR32−INTC_ILR63/ INTC_ILR64−INTC_ILR95

Address Offset	0x100−0x17C in 0x4-byte increments		
	0x180−0x1FC in 0x4-byte increments		
	0x200−0x27C in 0x4-byte increments		
Physical Address	0x480FE100−0x480FE17C	**Instance**	MPU_SS_INTC
	0x480FE180−0x480FE1FC		MPU_SS_INTC
	0x480FE200−0x480FE27C		MPU_SS_INTC
Physical Address Memory	IOMA + 0x9100−IOMA + 0x917C	**Instance**	DSP_SS_INTC
Physical Address I/O	0x5000−0x507C	**Instance**	DSP_SS_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

Table 11−46. INTC_ILR0−INTC_ILR31/INTC_ILR32−INTC_ILR63/ INTC_ILR64−INTC_ILR95 (Continued)

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Write 0s for future compatibility. Read returns 0s.	R	0x000000
7:2	PRIORITY	Interrupt priority	RW	0x00
1	Reserved	Write 0 for future compatibility. Read returns 0.	R	0
0	FIQNIRQ	Interrupt IRQ FiQ mapping 0x0: Interrupt is routed to IRQ. 0x1: Interrupt is routed to FIQ.	RW	0

Chapter 12

Memory Subsystem

This chapter describes the memory subsytem on the OMAP2420 device, including the general-purpose memory controller (GPMC), synchromous dynamic random access memory (SDRAM) controller subsystem (SDRC), and on-chip memory (OCM) subsystem.

Topic **Page**

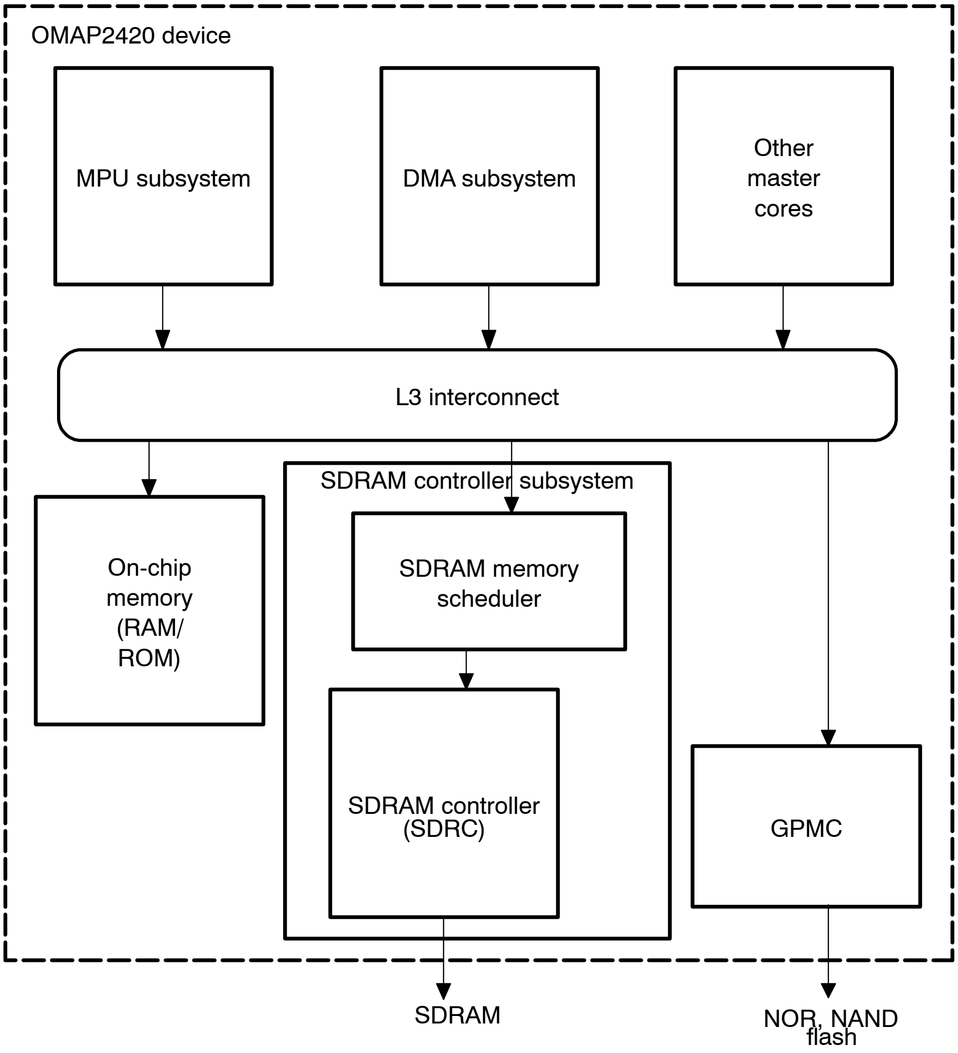
12.1 GPMC Overview

The general-purpose memory controller (GPMC) is the OMAP2420 unified memory controller dedicated to interfacing external memory devices:

- ☐ Asynchronous SRAM-like memories and application-specific integrated circuit (ASIC) devices
- ☐ Asynchronous, page mode, and synchronous burst NOR flash
- ☐ NAND flash

Figure 12−1 shows the GPMC environment.

Figure 12−1. GPMC Environment



12.1.1 Feature List

The GPMC is the OMAP2420 16-bit external memory controller. The GPMC data access engine provides a flexible programming model to interface all known standard memories. The GPMC supports various accesses:

- Asynchronous read/write access
- Asynchronous read page access (4-8-16 Word16)
- Synchronous read access
- Synchronous read burst access without wrap capability (4-8-16 Word16)
- Synchronous read burst access with wrap capability (4-8-16 Word16)
- Synchronous write burst access
- Address and data multiplexed access

The GPMC can be interfaced with a wide range of external devices:

- Asynchronous or synchronous 8-bit-wide memory or ASIC device (without page/burst support)
- Asynchronous or synchronous 16-bit-wide memory or ASIC device (with page/burst support)
- Asynchronous or synchronous 16-bit-wide address and data multiplexed memory or ASIC device (with page/burst support)
- 8-bit and 16-bit NAND flash devices

The GPMC supports up to 8 chip-select regions of programmable size and base addresses (from 16M bytes to 128M bytes) in a total address space of 1G byte.

12.2 GPMC Environment

Figure 12−2 through Figure 12−4 show different GPMC external connection options:

- ❑ GPMC to 16-bit nonmultiplexed memory
- ❑ GPMC to 16-bit address/data multiplexed memory
- ❑ GPMC to 16-bit NAND device

Figure 12−2 shows connections between the GPMC module and a 16-bit nonmultiplexed external device/memory.

Figure 12−2. GPMC to 16-Bit Nonmultiplexed Memory

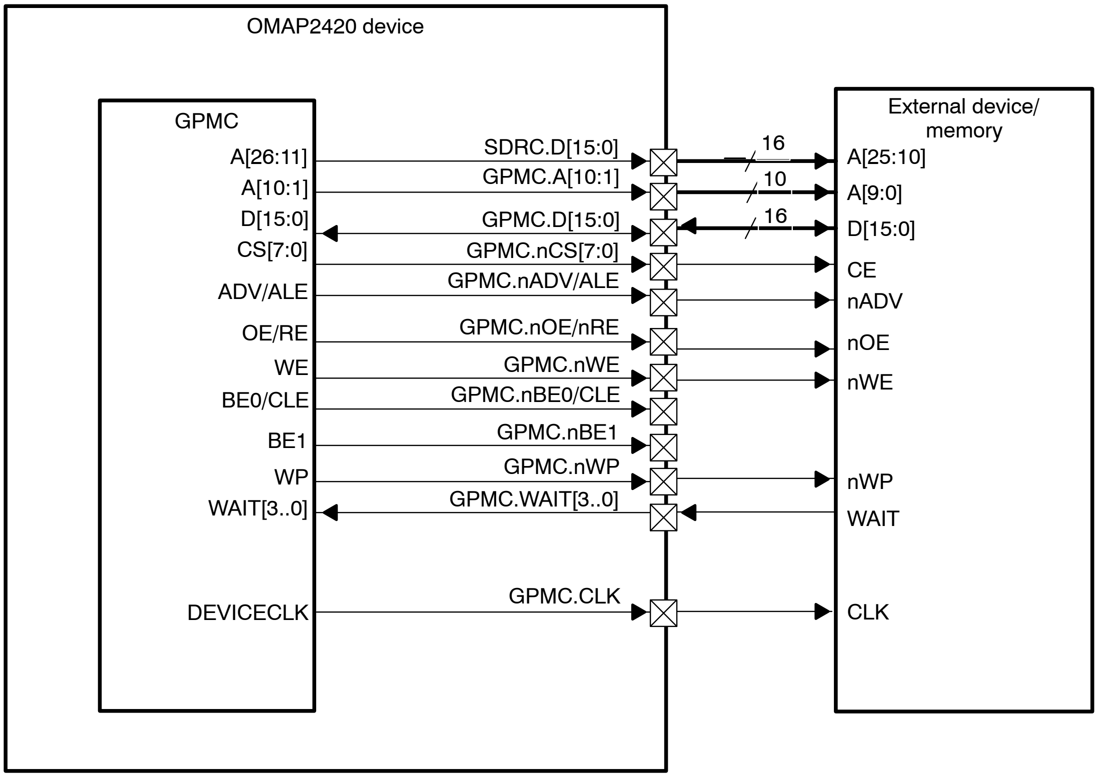
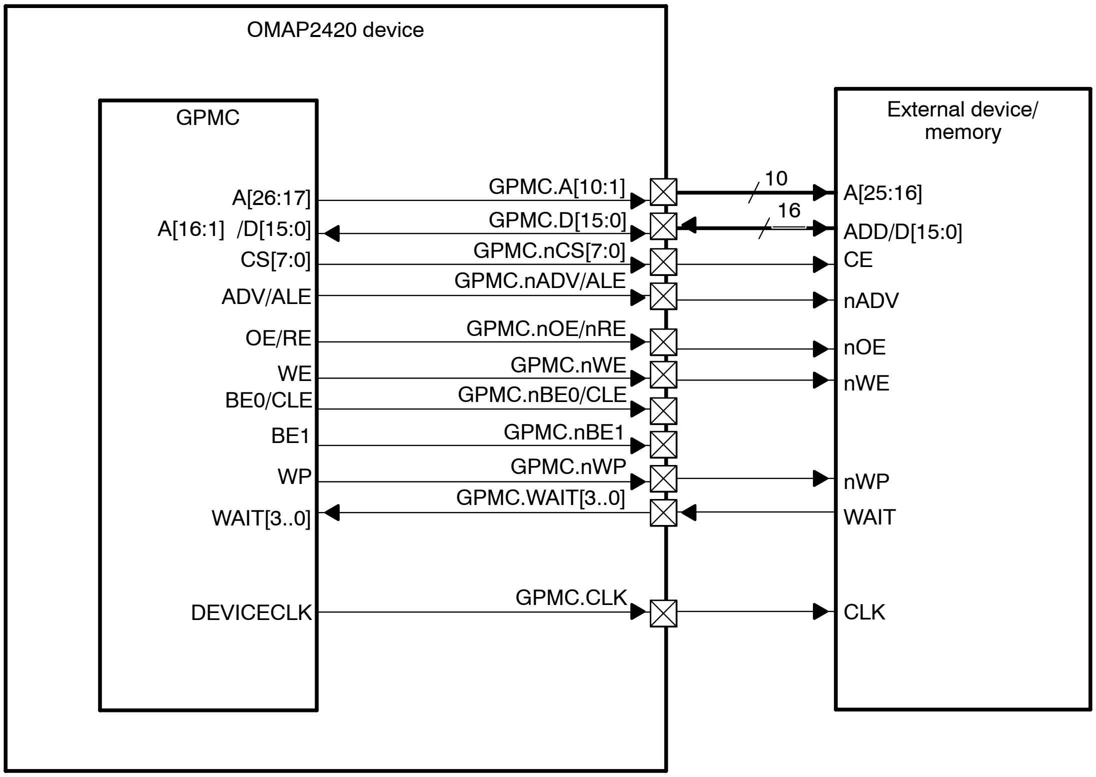


Figure 12−3 shows a connection between the GPMC module and a 16-bit address/data multiplexed external memory device.

Figure 12−3. GPMC to 16-Bit Address/Data Multiplexed Memory

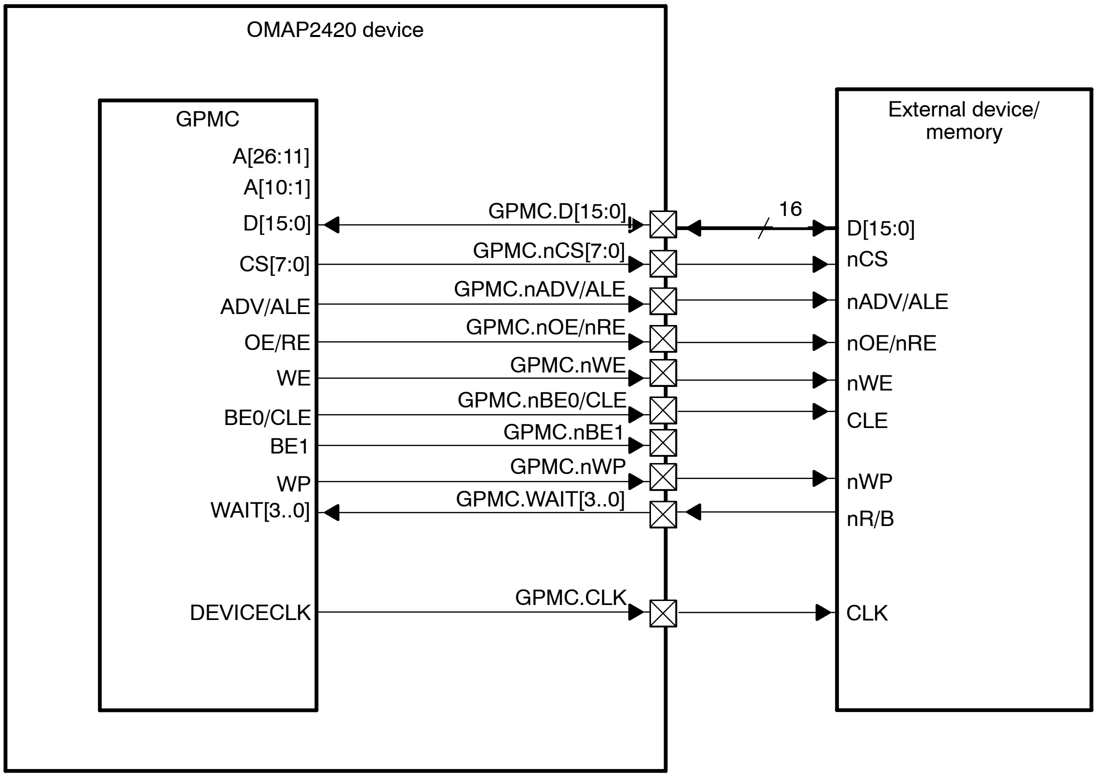


Note:

An 8-bit device must be connected to D[7:0] data bus of the GPMC. The OMAP2420 does not provide the A0 byte address line required for random-byte-addressable 8-bit-wide device interfacing (for both multiplexed and nonmultiplexed protocol). This limits the use of 8-bit-wide device interfacing to byte-alias access.

Figure 12−4 shows a connection between the GPMC and a 16-bit NAND device.

Figure 12−4. GPMC to 16-Bit NAND Device



Note:

An 8-bit NAND device must be connected to the D[7:0] data bus of the GPMC.

All GPMC controller subsystem input/output (I/O) pins are listed in Table 12−1.

Table 12−1. GPMC I/O Description

Signal Name	I/O[1]	Description
SDRC.D[15:0]	O	Address bus: A[26:11] in nonmultiplexed mode
GPMC.A[10:1]	O	Address bus: A[26:17] in address/data multiplexed mode A[10:1] in nonmultiplexed mode
GPMC.D[15:0]	I/O	Data bus: A[16:1]/D[15:0] in address data/multiplexed mode D[15:0] in nonmultiplexed mode
GPMC.nCS[7:0]	O	Chip-selects (active low)
GPMC.CLK	I/O	External clock
GPMC.nADV/ALE	O	Address valid (active low) if NOR protocol memory is selected Address latch enable (active high) if NAND protocol memory is selected
GPMC.nOE/nRE	O	Output enable (active low) if NOR protocol memory is selected Read enable (active low) if NAND protocol memory is selected

Table 12−1. GPMC I/O Description (Continued)

Signal Name	I/O[1]	Description
GPMC.nWE	O	Write enable (active low)
GPMC.nBE0/CLE	O	Lower byte enable (active low) Command latch enable (active high) if NAND protocol memory is selected
GPMC.nBE1	O	Upper byte 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	Data bus signal direction control

1) I = Input, O = Output

Table 12−2 shows the use of address and data synchromous dynamic random access memory (SDRAM) controller pins, according to the connected external device.

Table 12−2. GPMC Pin Multiplexing Options

GPMC Pin	Nonmultiplexed Address Data 16-Bit Device	Multiplexed Address Data 16-Bit Device	Nonmultiplexed Address Data 16-Bit Device with LIMITEDADDRESS Bit Enabled	16-Bit NAND Device	8-Bit NAND Device
SDRC.D[15]	A25	Not used	Not used	Not used	Not used
SDRC.D[14]	A24	Not used	Not used	Not used	Not used
SDRC.D[13]	A23	Not used	Not used	Not used	Not used
SDRC.D[12]	A22	Not used	Not used	Not used	Not used
SDRC.D[11]	A21	Not used	Not used	Not used	Not used
SDRC.D[10]	A20	Not used	Not used	Not used	Not used
SDRC.D[0]	A19	Not used	Not used	Not used	Not used
SDRC.D[8]	A18	Not used	Not used	Not used	Not used
SDRC.D[7]	A17	Not used	Not used	Not used	Not used
SDRC.D[6]	A16	Not used	Not used	Not used	Not used
SDRC.D[5]	A15	Not used	Not used	Not used	Not used
SDRC.D[4]	A14	Not used	Not used	Not used	Not used
SDRC.D[3]	A13	Not used	Not used	Not used	Not used
SDRC.D[2]	A12	Not used	Not used	Not used	Not used
SDRC.D[1]	A11	Not used	Not used	Not used	Not used
SDRC.D[0]	A10	Not used	Not used	Not used	Not used

Table 12−2. GPMC Pin Multiplexing Options (Continued)

GPMC Pin	Nonmultiplexed Address Data 16-Bit Device	Multiplexed Address Data 16-Bit Device	Nonmultiplexed Address Data 16-Bit Device with LIMITEDADDRESS Bit Enabled	16-Bit NAND Device	8-Bit NAND Device
GPMC.A[10]	A9	A25	A9	Not used	Not used
GPMC.A[9]	A8	A24	A8	Not used	Not used
GPMC.A[8]	A7	A23	A7	Not used	Not used
GPMC.A[7]	A6	A22	A6	Not used	Not used
GPMC.A[6]	A5	A21	A5	Not used	Not used
GPMC.A[5]	A4	A20	A4	Not used	Not used
GPMC.A[4]	A3	A19	A3	Not used	Not used
GPMC.A[3]	A2	A18	A2	Not used	Not used
GPMC.A[2]	A1	A17	A1	Not used	Not used
GPMC.A[1]	A0	A16	A0	Not used	Not used
GPMC.D[15]	D15	A15/D15	D15	D15	Not used
GPMC.D[14]	D14	A14/D14	D14	D14	Not used
GPMC.D[13]	D13	A13/D13	D13	D13	Not used
GPMC.D[12]	D12	A12/D12	D12	D12	Not used
GPMC.D[11]	D11	A11/D11	D11	D11	Not used
GPMC.D[10]	D10	A10/D10	D10	D10	Not used
GPMC.D[9]	D9	A9/D9	D9	D9	Not used
GPMC.D[8]	D8	A8/D8	D8	D8	Not used
GPMC.D[7]	D7	A7/D7	D7	D7	D7
GPMC.D[6]	D6	A6/D6	D6	D6	D6
GPMC.D[5]	D5	A5/D5	D5	D5	D5
GPMC.D[4}	D4	A4/D4	D4	D4	D4
GPMC.D[3]	D3	A3/D3	D3	D3	D3
GPMC.D[2]	D2	A2/D2	D2	D2	D2
GPMC.D[1]	D1	A1/D1	D1	D1	D1
GPMC.D[0]	D0	A0/D0	D0	D0	D0

Note:

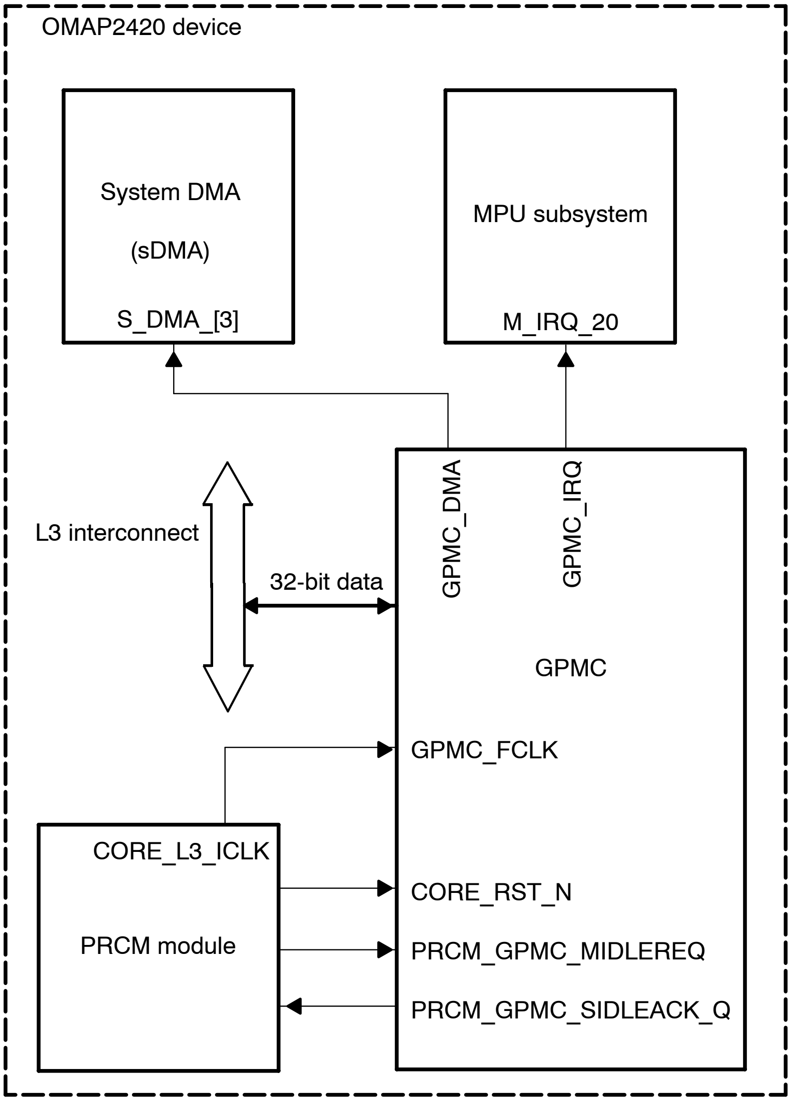
The OMAP2420 does not provide the A0 byte address line required for random byte addressable 8-bit-wide device interfacing (for both multiplexed and nonmultiplexed protocols). This limits the use of 8-bit-wide device interfacing to byte-alias access.

12.3 GPMC Integration

12.3.1 Description

Figure 12−5 shows how the GPMC module interacts with other modules in the OMAP2420.

Figure 12−5. GPMC Integration to the OMAP2420 Device



12.3.2 Clocking, Reset, and Power-Management Scheme

12.3.2.1 Clocking

GPMC_FCLK comes internally from the power, reset, and clock−management (PRCM) module and runs at the level 3 (L3) interconnect frequency.

This clock is used as the functional clock for the GPMC module.

The GPMC_FCLK source is PRCM CORE_L3_ICLK output.

CORE_L3_ICLK belongs to the L3 interconnect clock domain.

At the PRCM level, the GPMC module can be configured to prevent/allow the entire domain to be shut down, using the AUTO_GPMC bit of the CM_AUTOIDLE3_CORE register.

If the AUTO_GPMC bit is set to 0, the GPMC does not allow the clock domain (CORE_L3_ICLK) to shut down.

If the AUTO_GPMC bit is set to 1, the GPMC accepts an idle request from the PRCM module when the software shuts off the domain. The PRCM_GPMC_IDLEREQ request is then asserted and the GPMC sends back PRCM_ GPMC_SIDLEACK according to its idle mode setting.

Note:

The domain is shut down, provided all other modules belonging to the same clock domain (GPMC,OCM_ROM,OCM_RAM…) are able to accept this request.

For more information, see Chapter 5, *Power, Reset, and Clock Management.*

12.3.2.2 Hardware Reset

Global reset of the GPMC controller is done by activating the CORE_RST_N signal in the CORE_RST domain (see Chapter 5, *Power, Reset, and Clock Management*).

12.3.2.3 Software Reset

GPMC modules can be reset under software control using the GPMC_SYSCONFIG soft reset bit in the SYSCONFIG[1] register. When software reset is on, all registers and the finite state−machine (FSM) are reset immediately and unconditionally.

12.3.2.4 Power Domain, Power Saving, and Reset Management

SDRAM controller power is supplied by the CORE power domain (for information about the CORE power domain, see Chapter 5, *Power, Reset, and Clock Management)*.

- The GPMC complies with system power-management guidelines.
- The GPMC supports autoidle mode to dynamically reduce power consumption by internally disabling the functional clock when no requests are pending and no accesses are ongoing.
  - This dynamic autoidle mode can be enabled through the AUTOIDLE bit field of the GPMC_SYSCONFIG register.
- On PRCM module idle request, the GPMC sends an idle request acknowledge to allow the PRCM module to cut the GPMC source clock correctly.

- The idle request/acknowledge process can be configured through the IDLEMODE bit field of the GPMC_SYSCONFIG register to be in one of three idle modes:
  - Force-idle mode: The GPMC goes to idle state immediately on receiving the request from the PRCM module.
  - No-idle mode: The GPMC never goes to idle state.
  - Smart-idle mode: The GPMC goes into idle state when all ongoing transactions are complete. Smart-idle mode is strongly recommended.

12.3.2.5 Hardware Requests

One interrupt request (GPMC_IRQ) goes from the GPMC to the MPU subsystem (M_IRQ_20).

One DMA request (GPMC_DMA) goes from the GPMC module to the sDMA (S_DMA_3).

12.3.3 SDRC and GPMC I/O Sharing

The OMAP2420 external memory interface consists of two separate controllers:

- One 8-/16-bit general-purpose memory controller (GPMC)
- One 16-/32-bit SDRAM controller (SDRC)

According to attached memories and application pin count requirements, static multiplexing options exist between the GPMC and SRDC, depending on the interface configurations required. If a configuration does not require all GPMC/SDRC pins, GPIOs or alternate function signals can also be multiplexed (see Table 12−4 and Table 12−52).

When SDRC.D[15:0] I/Os are used by one module, they are off limits to the other until the next integrated circuit (IC) reset (power-on reset).

12.3.3.1 GPMC Address and Data Bus Use

Depending on the GPMC configuration on each chip-select, the address and data bus lines not required for a particular access protocol are not updated (changed from current value) and are not sampled when input (input data bus).

- When GPMC_CONFIG.LIMITEDADDRESS = 1, nonmultiplexed address/data NOR devices do not use the shared SDRC/GPMC I/O SDRC.D[15:0]. Only GPMC.A[10:0] address lines are used. This limits the memory support only 2K-byte addressable memories.
- Multiplexed address and data NOR devices do not use shared SDRC I/O: SDRC.D[15:0]. The address is multiplexed on the data bus.
- 8-bit-wide NOR devices do not use GPMC I/O: GPMC.D[15:8].
- 16-bit-wide NAND devices do not use the shared SDRC/GPMC I/O: SDRC.D[15:0] and GPMC I/O: GPMC.A [10:1].

- 8-bit-wide NAND devices do not use the shared SDRC I/O: SDRC.D[15:0], GPMC I/O: GPMC.A[10:1], or GPMC I/O: GPMC.D[15:8].

> **From the OMAP2420 power-on reset (IC reset), address and data lines are set to the default logical 1 value. This default value is used as a transparent value that allows OMAP device pin sharing (transparent multiplexing) between the GPMC and alternate I/O functions.**
>
> **The alternate functions are selected in place of the GPMC function if the GPMC function is not used. For example, alternate I/O functions can use GPMC.D[15:8] if GPMC use is restricted to 8-bit-wide device interfacing.**
>
> **Never configure a chip-select or initiate an access with an incompatible sharing configuration that could corrupt the transparent reset value.**

> **For a nonmultiplexed chip-select configuration, you can force transparent sharing of A[26:11] address lines by setting the LIMITEDADDRESS bit (GPMC_CONFIG register). In this case, only the GPMC.A[10:1] I/O address lines are usable so that nonmultiplexed devices can be supported through a limited 2K-byte address range.**
>
> **Set the LIMITEDADDRESS bit control before attempting to access a chip-select configured with a nonmultiplexed protocol. After the first nonmultiplexed access without this bit setup, SDRC[15:0] I/Os are lost for SDRC use until the next IC reset.**

12.3.3.2 SDRC and GPMC I/O Configuration Solutions

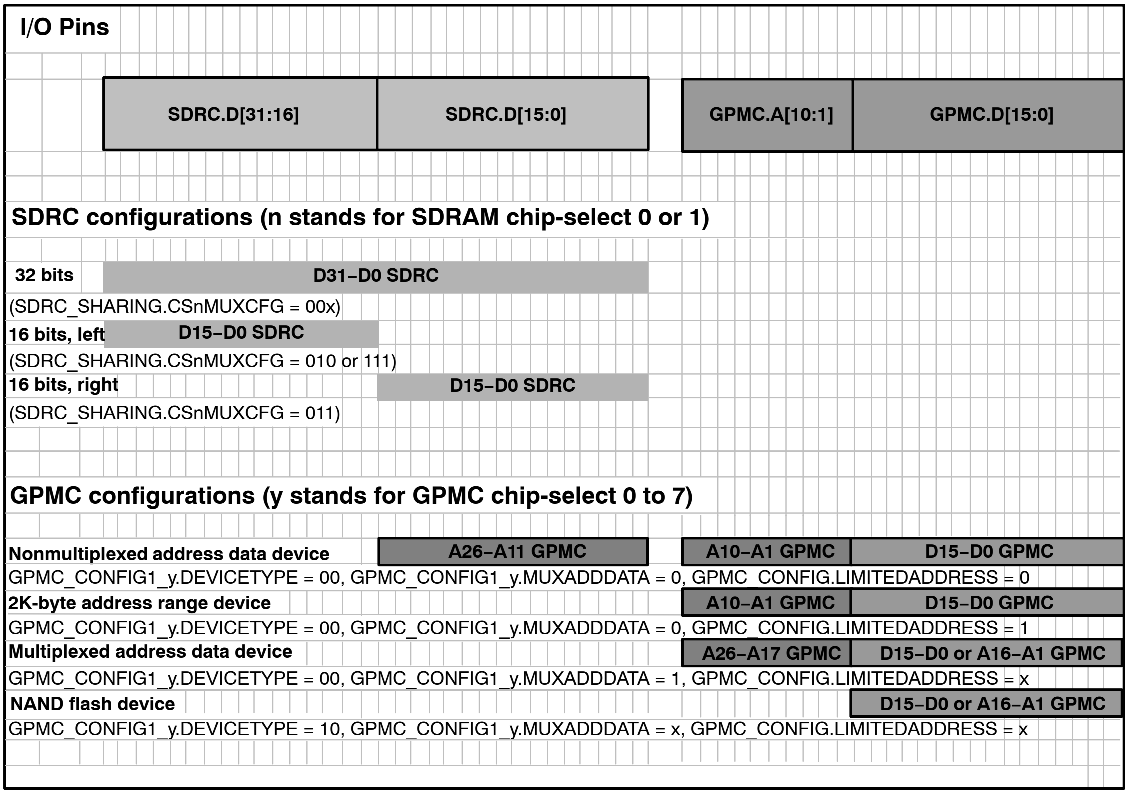
Figure 12−6 shows different solutions for connecting SDRAM devices and NOR flash devices, asynchronous/synchronous devices, and NAND flash devices.

All these memories are connected through several chip-selects (*n* and *y)* and share four I/O groups:

- SDRC.D[31:16]
  - Upper 16 bits of a 32-bit SDRAM device (data [31:16])
  - 16-bit SDRAM device (data [15:0])
- SDRC.D[15:0]
  - Lower 16 bits of a 32-bit SDRAM device (data [15:0])
  - 16-bit SDRAM device (data [15:0])
  - Nonmultiplexed address/data device (address [26:11]) when GPMC_CONFIG.LIMITEDADDRESS is not set
- GPMC.A[10:1]
  - Nonmultiplexed address/data device (address [10:1])
  - Multiplexed address/data device (address [26:17])

- ☐ GPMC.D[15:0]
  - ■ Nonmultiplexed address/data device (data [15:0])
  - ■ Multiplexed address/data device (address [16:1] and data [15:0])

Figure 12−6. SDRC/GPMC I/O Pin Configuration Options in Default Pinout Mode 0


I/O Pins
SDRC.D[31:16]
SDRC.D[15:0]
GPMC.A[10:1]
GPMC.D[15:0]
SDRC configurations (n stands for SDRAM chip-select 0 or 1)
32 bits
D31−D0 SDRC
(SDRC_SHARING.CSnMUXCFG = 00x)
16 bits, left
D15−D0 SDRC
(SDRC_SHARING.CSnMUXCFG = 010 or 111)
16 bits, right
D15−D0 SDRC
(SDRC_SHARING.CSnMUXCFG = 011)
GPMC configurations (y stands for GPMC chip-select 0 to 7)
Nonmultiplexed address data device
A26−A11 GPMC
A10−A1 GPMC
D15−D0 GPMC
GPMC_CONFIG1_y.DEVICETYPE = 00, GPMC_CONFIG1_y.MUXADDDATA = 0, GPMC_CONFIG.LIMITEDADDRESS = 0
2K-byte address range device
A10−A1 GPMC
D15−D0 GPMC
GPMC_CONFIG1_y.DEVICETYPE = 00, GPMC_CONFIG1_y.MUXADDDATA = 0, GPMC_CONFIG.LIMITEDADDRESS = 1
Multiplexed address data device
A26−A17 GPMC
D15−D0 or A16−A1 GPMC
GPMC_CONFIG1_y.DEVICETYPE = 00, GPMC_CONFIG1_y.MUXADDDATA = 1, GPMC_CONFIG.LIMITEDADDRESS = x
NAND flash device
D15−D0 or A16−A1 GPMC
GPMC_CONFIG1_y.DEVICETYPE = 10, GPMC_CONFIG1_y.MUXADDDATA = x, GPMC_CONFIG.LIMITEDADDRESS = x

The OMAP2420 device can support a 32-bit-wide external DDR/SDRAM interface only when the GPMC use is restricted to address and data multiplexed protocol (random synchronous or asynchronous memory and NAND device type).

These protocols leave the A[26:11] address lines (*SDRC.D[15:0] I/O)* free for transparent SDRC.D[15:0] pin sharing.

You must never configure a chip-select nor perform any access with an Incompatible sharing configuration can corrupt the transparent reset value.

12.3.3.3 SDRC and GPMC I/O Configuration Settings (in Default Pinout Mode 0)

SDRC I/O Configuration Setting (in Default Pinout Mode 0)

External 32- or 16-bit data SDRAM is selected by SDRC register fields SDRC_SHARING[14:12] (CS1MUXCFG) and SDRC register fields SDRC_SHARING[11:9] (CS0MUXCFG) with the following options:

- 32 bits = 00x
- 16 bits on SDRC.D[31:16] = 010 or 011
- 16 bits on SDRC.D[15:0] = 011

The reset value is 16 bits on SDRC.D[31:16] to authorize an external boot through the GPMC on a nonmultiplexed address/data device.

GPMC I/O Configuration Setting (in Default Pinout Mode 0)

In this section, *y* stands for chip-select value (CS*y).*

The nonmultiplexed address/data device, if not limited to a 2K-byte address range, is selected by programming the following register fields:

- SDRC register SDRC_SHARING[11:9] (CS0MUXCFG) = 010
- SDRC register SDRC_SHARING[14:12] (CS1MUXCFG) = 010
- GPMC register GPMC_CONFIG1_y[11:10] (CSy DEVICETYPE) = 00
- GPMC_CONFIG1_y[9] (CSy MUXADDDATA) = 0
- GPMC_CONFIG[1] (LIMITEDADDRESS) = 0

CAUTION

GPMC_CONFIG1_y.DEVICETYPE = 0, GPMC_CONFIG1_ y.MUXADDATA = 0 with 32-bit external DDR, and GPMC_CONFIG.LIMITEDADDRESS = 0 are not supported.

The nonmultiplexed address/data device, if limited to a 2K-byte address range, is selected by programming the following register fields:

- GPMC register GPMC_CONFIG1_y[11:10] (CSy DEVICETYPE) = 00
- GPMC_CONFIG1_y[9] (CSy MUXADDDATA) = 0
- GPMC_CONFIG[1] (LIMITEDADDRESS) = 1

Note:

The LIMITEDADDRESS field applies only to nonmultiplexed address/data devices and has no effect on other device types (multiplexed address/data, NAND).

The multiplexed address/data device is selected by programming the following register fields:

- GPMC register GPMC_CONFIG1_y[11:10] (CSy DEVICETYPE) = 00
- GPMC_CONFIG1_y[9] (CSy MUXADDDATA) = 1

The NAND device is selected by programming the following register field:

- GPMC register GPMC_CONFIG1_y[11:10] (CSy DEVICETYPE) = 10

Note:

The CSy MUXADDDATA field applies only to DEVICETYPE = 00 and has no effect on NAND-like devices.

12.3.3.4 GPMC CS0 Default Configuration at IC Reset

To ensure correct external boot with a GPMC access from IC reset time on CS0, three external pins (SYS.BOOT[2:0]) are sampled to configure three GPMC register field reset values of the GPMC_CONFIG1_0 register:

- GPMC_CONFIG1_0[22]: WAITREADMONITORING bit reset value
- GPMC_CONFIG1_0[13:12]: DEVICESIZE bits reset value
- GPMC_CONFIG1_0[9]: MUXADDDATA bit reset value

The SYS.BOOT[3] external pin determines internal or external boot.

Table 12−3. GPMC CS0 Default Configuration at IC Reset

SYS.BOOT[3:0]	Internal/ Boot Code	Wait Monitoring	Memory Interface Type	GPMC_ CONFIG1_0_ WAITREADMONITORING Reset Value	GPMC_ CONFIG1_0_ DEVICESIZE Reset Value	GPMC_ CONFIG1_0_ MUXADDDATA Reset Value
0000	Reserved					
0001	Reserved					
0010	Reserved					
0011	Reserved					
1000	Internal	Active low	Nonmultiplexed 16-bit	0x1	0x1	0x0
1001	Internal	Don't care	Nonmultiplexed 16-bit	0x0	0x1	0x0
1010	Internal	Active low	Address/data multiplexed 16-bit	0x1	0x1	0x1
1011	Internal	Don't care	Address/data multiplexed 16-bit	0x0	0x1	0x1
1100	Internal	–	NAND 8-bit[1]	0x0	0x0	0x0
1101	Internal	–	NAND *16-bit[1]	0x0	0x1	0x0

1) DEVICETYPE is always 0x0 at reset and is programmed by internal ROM code to NAND (0x2), if necessary.

> CAUTION
>
> **With internal boot code, the entire CS0 configuration can be modified before the first CS0 access. This modification into internal boot code is necessary for two device types:**
>
> - **NAND device attached to CS0**
> - **Nonmultiplexed 2K-byte address range device used to free SDRC[15:0] I/O and attached to CS0.**

12.3.4 Pin List and Pad Multiplexing With Other Functions

In the Mode 0 through Mode 5 columns in Table 12−4, gray shading indicates that the mode is used for the corresponding pin, black shading indicates that the mode is not used for the corresponding pin, and white cells indicate an alternate function.

Table 12−4. GPMC Pin List and Pad Multiplexing

GPMC	Description	DIR	Ball	ALTERNATE FUNCTIONS					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
GPMC.A1	General-purpose memory address bit 1	O	J3		dss.d18		gpio.12		
GPMC.A2	General-purpose memory address bit 2	O	H4		dss.d19		gpio.11		
GPMC.A3	General-purpose memory address bit 3	O	H3		dss.d20		gpio.10		
GPMC.A4	General-purpose memory address bit 4	O	G3		dss.d21		gpio.9		
GPMC.A5	General-purpose memory address bit 5	O	F4		dss.d22		gpio.8		
GPMC.A6	General-purpose memory address bit 6	O	F3		dss.d23		gpio.7		
GPMC.A7	General-purpose memory address bit 7	O	E4			sys.nd mareq2	gpio.6		
GPMC.A8	General-purpose memory address bit 8	O	G4			sys.nd mareq3	gpio.5		
GPMC.A9	General-purpose memory address bit 9	O	D3			sys.nd mareq4	gpio.4		
GPMC.A10	General-purpose memory address bit 10	O	E3			sys.nd mareq5	gpio.3		
GPMC.A11	General-purpose memory address bit 11	IO	C2	Muxed with SDRC.D0					
GPMC.A12	General-purpose memory address bit 12	IO	H7	Muxed with SDRC.D1					
GPMC.A13	General-purpose memory address bit 13	IO	D4	Muxed with SDRC.D2					
GPMC.A14	General-purpose memory address bit 14	IO	B2	Muxed with SDRC.D3					
GPMC.A15	General-purpose memory address bit 15	IO	C4	Muxed with SDRC.D4					
GPMC.A16	General-purpose memory address bit 16	IO	C3	Muxed with SDRC.D5					
GPMC.A17	General-purpose memory address bit 17	IO	C5	Muxed with SDRC.D6					
GPMC.A18	General-purpose memory address bit 18	IO	H8	Muxed with SDRC.D7					

Table 12−4. GPMC Pin List and Pad Multiplexing (Continued)

GPMC	Description	DIR	Ball	ALTERNATE FUNCTIONS					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
GPMC.A19	General-purpose memory address bit 19	IO	G11	Muxed with SDRC.D8					
GPMC.A20	General-purpose memory address bit 20	IO	H11	Muxed with SDRC.D9					
GPMC.A21	General-purpose memory address bit 21	IO	G10	Muxed with SDRC.D10					
GPMC.A22	General-purpose memory address bit 22	IO	H10	Muxed with SDRC.D11					
GPMC.A23	General-purpose memory address bit 23	IO	A11	Muxed with SDRC.D12					
GPMC.A24	General-purpose memory address bit 24	IO	D6	Muxed with SDRC.D13					
GPMC.A25	General-purpose memory address bit 25	IO	H9	Muxed with SDRC.D14					
GPMC.A26	General-purpose memory address bit 26	IO	C6	Muxed with SDRC.D15					
GPMC.D0	GPMC data bit 0	IO	M3						
GPMC.D1	GPMC data bit 1	IO	M4						
GPMC.D2	GPMC data bit 2	IO	N3						
GPMC.D3	GPMC data bit 3	IO	N4						
GPMC.D4	GPMC data bit 4	IO	P3						
GPMC.D5	GPMC data bit 5	IO	R3						
GPMC.D6	GPMC data bit 6	IO	P4						
GPMC.D7	GPMC data bit 7	IO	R4						
GPMC.D8	GPMC data bit 8	IO	M8				gpio.20		
GPMC.D9	GPMC data bit 9	IO	M7				gpio.19		
GPMC.D10	GPMC data bit 10	IO	L7				gpio.18		
GPMC.D11	GPMC data bit 11	IO	K8				gpio.17		
GPMC.D12	GPMC data bit 12	IO	L8				gpio.16		
GPMC.D13	GPMC data bit 13	IO	K7				gpio.15		
GPMC.D14	GPMC data bit 14	IO	J8				gpio.14		
GPMC.D15	GPMC data bit 15	IO	J7				gpio.13		

Table 12−4. GPMC Pin List and Pad Multiplexing (Continued)

GPMC	Description	DIR	Ball	ALTERNATE FUNCTIONS					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
GPMC.nCS0	GPMC chip-select bit 0	O	L4						
GPMC.nCS1	GPMC chip-select bit 1	O	N8				gpio.22		
GPMC.nCS2	GPMC chip-select bit 2	O	E2				gpio.23		
GPMC.nCS3	GPMC chip-select bit 3	O	N2		gpmc. io.dir		gpio.24		
GPMC.nCS4	GPMC chip-select bit 4	O	F1				gpio.25		
GPMC.nCS5	GPMC chip-select bit 5	O	H1				gpio.26		
GPMC.nCS6	GPMC chip-select bit 6	O	K1				gpio.27		
GPMC.nCS7	GPMC chip-select bit 7	O	L2		gpmc.io.dir		gpio.28		
GPMC.IO_ DIR	GPMC I/O direction control	O	N2	gpmc.ncs3			gpio.24		
			L2	gpmc.ncs7			gpio.28		
GPMC.CLK	GPMC clock	IO	J4				gpio.21		
GPMC. nADV/ALE	Address valid or address latch enable	O	K3						
GPMC.nOE	Output enable	O	H2						
GPMC.nWE	Write enable	O	K4						
GPMC.nBE0	Lower byte enable. Also used for command latch enable	O	P7				gpio.29		
GPMC.nBE1	Upper byte enable	O	R1				gpio.30		
GPMC.nWP	Flash write protect	O	G2				gpio.31		
GPMC.WAIT0	External indication of wait	I	L3						
GPMC.WAIT1	External indication of wait	I	N7				gpio.33		
GPMC.WAIT2	External Indication of wait	I	M1				gpio.34		
GPMC.WAIT3	External Indication of wait	I	P1				gpio.35		

12.3.5 GPMC Register Summary

Table 12−5 shows the type, size, and physical addresses of all GPMC registers.

Table 12−5. GPMC1 Register Summary

Register Name	Type	Register Width (Bits)	Physical Address
GPMC_REVISION	R	32	0x6800 A000
GPMC_SYSCONFIG	RW	32	0x6800 A010
GPMC_SYSSTATUS	R	32	0x6800 A014
GPMC_IRQSTATUS	RW	32	0x6800 A018
GPMC_IRQENABLE	RW	32	0x6800 A01C
GPMC_TIMEOUT_CONTROL	RW	32	0x6800 A040
GPMC_ERR_ADDRESS	RW	32	0x6800 A044
GPMC_ERR_TYPE	RW	32	0x6800 A048
GPMC_CONFIG	RW	32	0x6800 A050
GPMC_STATUS	RW	32	0x6800 A054
GPMC_CONFIG1_0	RW	32	0x6800 A060
GPMC_CONFIG2_0	RW	32	0x6800 A064
GPMC_CONFIG3_0	RW	32	0x6800 A068
GPMC_CONFIG4_0	RW	32	0x6800 A06C
GPMC_CONFIG5_0	RW	32	0x6800 A070
GPMC_CONFIG6_0	RW	32	0x6800 A074
GPMC_CONFIG7_0	RW	32	0x6800 A078
GPMC_NAND_COMMAND_0	W	32	0x6800 A07C
GPMC_NAND_ADDRESS_0	W	32	0x6800 A080
GPMC_NAND_DATA_0	RW	32	0x6800 A084
GPMC_CONFIG1_1	RW	32	0x6800 A090
GPMC_CONFIG2_1	RW	32	0x6800 A094
GPMC_CONFIG3_1	RW	32	0x6800 A098
GPMC_CONFIG4_1	RW	32	0x6800 A09C
GPMC_CONFIG5_1	RW	32	0x6800 A0A0
GPMC_CONFIG6_1	RW	32	0x6800 A0A4
GPMC_CONFIG7_1	RW	32	0x6800 A0A8
GPMC_NAND_COMMAND_1	W	32	0x6800 A0AC
GPMC_NAND_ADDRESS_1	W	32	0x6800 A0B0
GPMC_NAND_DATA_1	RW	32	0x6800 A0B4
GPMC_CONFIG1_2	RW	32	0x6800 A0C0
GPMC_CONFIG2_2	RW	32	0x6800 A0C4

Table 12−5. GPMC1 Register Summary (Continued)

Register Name	Type	Register Width (Bits)	Physical Address
GPMC_CONFIG3_2	RW	32	0x6800 A0C8
GPMC_CONFIG4_2	RW	32	0x6800 A0CC
GPMC_CONFIG5_2	RW	32	0x6800 A0D0
GPMC_CONFIG6_2	RW	32	0x6800 A0D4
GPMC_CONFIG7_2	RW	32	0x6800 A0D8
GPMC_NAND_COMMAND_2	W	32	0x6800 A0DC
GPMC_NAND_ADDRESS_2	W	32	0x6800 A0E0
GPMC_NAND_DATA_2	RW	32	0x6800 A0E4
GPMC_CONFIG1_3	RW	32	0x6800 A0F0
GPMC_CONFIG2_3	RW	32	0x6800 A0F4
GPMC_CONFIG3_3	RW	32	0x6800 A0F8
GPMC_CONFIG4_3	RW	32	0x6800 A0FC
GPMC_CONFIG5_3	RW	32	0x6800 A100
GPMC_CONFIG6_3	RW	32	0x6800 A104
GPMC_CONFIG7_3	RW	32	0x6800 A108
GPMC_NAND_COMMAND_3	W	32	0x6800 A10C
GPMC_NAND_ADDRESS_3	W	32	0x6800 A110
GPMC_NAND_DATA_3	RW	32	0x6800 A114
GPMC_CONFIG1_4	RW	32	0x6800 A120
GPMC_CONFIG2_4	RW	32	0x6800 A124
GPMC_CONFIG3_4	RW	32	0x6800 A128
GPMC_CONFIG4_4	RW	32	0x6800 A12C
GPMC_CONFIG5_4	RW	32	0x6800 A130
GPMC_CONFIG6_4	RW	32	0x6800 A134
GPMC_CONFIG7_4	RW	32	0x6800 A138
GPMC_NAND_COMMAND_4	W	32	0x6800 A13C
GPMC_NAND_ADDRESS_4	W	32	0x6800 A140
GPMC_NAND_DATA_4	RW	32	0x6800 A144
GPMC_CONFIG1_5	RW	32	0x6800 A150
GPMC_CONFIG2_5	RW	32	0x6800 A154
GPMC_CONFIG3_5	RW	32	0x6800 A158
GPMC_CONFIG4_5	RW	32	0x6800 A15C
GPMC_CONFIG5_5	RW	32	0x6800 A160
GPMC_CONFIG6_5	RW	32	0x6800 A164

Table 12−5. GPMC1 Register Summary (Continued)

Register Name	Type	Register Width (Bits)	Physical Address
GPMC_CONFIG7_5	RW	32	0x6800 A168
GPMC_NAND_COMMAND_5	W	32	0x6800 A16C
GPMC_NAND_ADDRESS_5	W	32	0x6800 A170
GPMC_NAND_DATA_5	RW	32	0x6800 A174
GPMC_CONFIG1_6	RW	32	0x6800 A180
GPMC_CONFIG2_6	RW	32	0x6800 A184
GPMC_CONFIG3_6	RW	32	0x6800 A188
GPMC_CONFIG4_6	RW	32	0x6800 A18C
GPMC_CONFIG5_6	RW	32	0x6800 A190
GPMC_CONFIG6_6	RW	32	0x6800 A194
GPMC_CONFIG7_6	RW	32	0x6800 A198
GPMC_NAND_COMMAND_6	W	32	0x6800 A19C
GPMC_NAND_ADDRESS_6	W	32	0x6800 A1A0
GPMC_NAND_DATA_6	RW	32	0x6800 A1A4
GPMC_CONFIG1_7	RW	32	0x6800 A1B0
GPMC_CONFIG2_7	RW	32	0x6800 A1B4
GPMC_CONFIG3_7	RW	32	0x6800 A1B8
GPMC_CONFIG4_7	RW	32	0x6800 A1BC
GPMC_CONFIG5_7	RW	32	0x6800 A1C0
GPMC_CONFIG6_7	RW	32	0x6800 A1C4
GPMC_CONFIG7_7	RW	32	0x6800 A1C8
GPMC_NAND_COMMAND_7	W	32	0x6800 A1CC
GPMC_NAND_ADDRESS_7	W	32	0x6800 A1D0
GPMC_NAND_DATA_7	RW	32	0x6800 A1D4
GPMC_PREFETCH_CONFIG1	RW	32	0x6800 A1E0
GPMC_PREFETCH_CONFIG2	RW	32	0x6800 A1E4
GPMC_PREFETCH_CONTROL	RW	32	0x6800 A1EC
GPMC_PREFETCH_STATUS	RW	32	0x6800 A1F0
GPMC_ECC_CONFIG	RW	32	0x6800 A1F4
GPMC_ECC_CONTROL	RW	32	0x6800 A1F8
GPMC_ECC_SIZE_CONFIG	RW	32	0x6800 A1FC
GPMC_ECC1_RESULT	RW	32	0x6800 A200
GPMC_ECC2_RESULT	RW	32	0x6800 A204
GPMC_ECC3_RESULT	RW	32	0x6800 A208
GPMC_ECC4_RESULT	RW	32	0x6800 A20C

Table 12−5.GPMC1 Register Summary (Continued)

Register Name	Type	Register Width (Bits)	Physical Address
GPMC_ECC5_RESULT	RW	32	0x6800 A210
GPMC_ECC6_RESULT	RW	32	0x6800 A214
GPMC_ECC7_RESULT	RW	32	0x6800 A218
GPMC_ECC8_RESULT	RW	32	0x6800 A21C
GPMC_ECC9_RESULT	RW	32	0x6800 A220
GPMC_TESTMODE_CTRL	RW	32	0x6800 A230
GPMC_PSA_LSB	R	32	0x6800 A234
GPMC_PSA_MSB	R	32	0x6800 A238

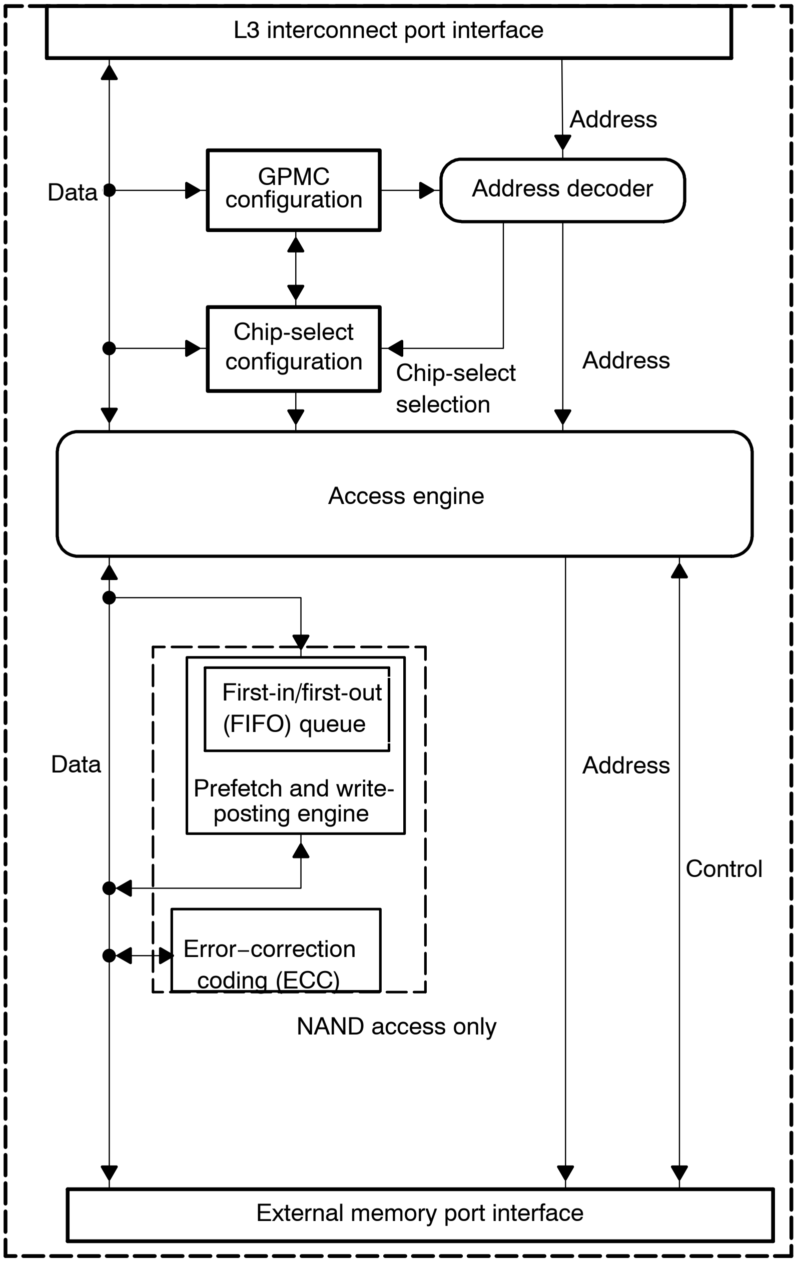
12.4 GPMC Functional Description

12.4.1 Description

The GPMC can be divided into six main blocks, as illustrated in Figure 12−7:

- L3 interconnect port interface
- Address decoding and GPMC configuration register file
- Access engine
- Prefetch and write-posting engine
- ECC
- External device/memory port interface

Figure 12−7. GPMC Functional Diagram


L3 interconnect port interface
Address
Data
GPMC configuration
Address decoder
Chip-select configuration
Chip-select selection
Address
Access engine
First-in/first-out (FIFO) queue
Prefetch and write-posting engine
Data
Address
Control
Error−correction coding (ECC)
NAND access only
External memory port interface

12.4.1.1 L3 Interconnect Interface

The GPMC is connected to the OMAP system L3 interconnect through a 32-bit data-width port.

The GPMC L3 interconnect interface is a pipelined interface including an 8 x Word32 write buffer.

Any OMAP system host can issue external access requests through the GPMC.

The OMAP system can issue the following requests through this interface:

- ❑ Byte, Word16, Word32
- ❑ Incrementing fixed-length bursts of 2, 4, and 8 words
- ❑ Wrapped (critical word access first) fixed-length bursts of 8 words

This interface also provides one interrupt and one DMA request line for specific event control.

12.4.1.2 Address Decoding, GPMC, and Chip-Select Configuration Register File

The address decoding logic is responsible for chip-select selection according to the chip-select base address register file content and according to the address request.

The register file 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, Word16, or Word32 accesses. The register file should be configured as a noncacheable, nonbufferable (nonposted write) region to prevent a desynchronization between the host execution (write request) and the completion of register configuration (write completed with register updated).

For GPMC memory locations, see Chapter 2, *Memory Mapping*. For GPMC register locations, see Section 12.3.5, *GMPC Register Summary*.

The access engine performs appropriate access to the external device according to the associated chip-select configuration and according to the system request.

The access engine drives the external interface control signals and applies the protocol interface according to the user-defined timing parameters and settings.

12.4.1.3 ECC

The GPMC includes error code correction (ECC) calculator circuitry that allows ECC calculation on the fly during data read or data program (write) operations.

ECC is based on a two-dimensional (row and column) bit parity accumulation called hamming code. Parity accumulation is done on the programmed number of bytes or Word16s read from the memory device or written to the memory device in stream mode.

There is no automatic error detection or correction and the software NAND driver must read the multiple ECC calculation result, compare it to the expected code value, and take appropriate action.

The ECC engine includes only one accumulation context, so it can be allocated to only one chip-select at a time.

12.4.1.4 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; 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) and in that sense it is primarily dedicated to NAND support. The engine does not include an address generator, and the request is limited to chip-select target identification. It includes a 64-byte first-in/first-out (FIFO) queue associated with a DMA request synchronization line for optimal DMA-based use.

For more information about prefetch and write-posting engine programming, see Section 12.5.10.4, *Prefetch and Write-Posting Engine*.

12.4.1.5 External Device/Memory Port Interface

The external port interface controls all addresses, data, and control signals required to interface GPMC-supported devices and memories.

12.5 GPMC Programming Model

The GPMC programming model offers maximum flexibility to support multiple access protocols for each of the eight configurable chip-selects. Use the appropriate optimized chip-select settings according to the associated external device characteristics:

- Protocol can be selected to support generic asynchronous or synchronous random-access devices (NOR flash, SRAM) or to support specific NAND devices.
- Address and data bus can be multiplexed on the same external bus.
- Read access can be independently defined to be performed as asynchronous or synchronous. Write access is defined as asynchronous.
- System requests (byte, word16, word32, burst) are performed through single or several accesses. External access profiles (single, multiple with optimized burst length, native or emulated wrap) are performed based on external device characteristics (bus width, data buffer size, native wrap support).
- System burst read request can be defined to be performed as synchronous burst or asynchronous page access (multiple read), or to be performed as successive single synchronous or asynchronous accesses (single read) when burst or page mode are not supported by the external memory or ASIC devices. 8-bit width devices are supported only in single synchronous or asynchronous read mode.
- System burst write requests must be defined to be performed as successive single accesses (single write) in asynchronous access mode.
- Similarly to a programmable internal wait state, an external wait pin can be monitored to dynamically control external access at the beginning (initial access time) 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 time periods:

- Read and write access duration
- Most GPMC external interface control signals assertion and deassertion times
- Data-capture time during read access
- External wait-pin monitoring time
- Duration of idle time to be inserted between accesses, when required

12.5.1 Chip-Select Base Address and Region Size Configuration

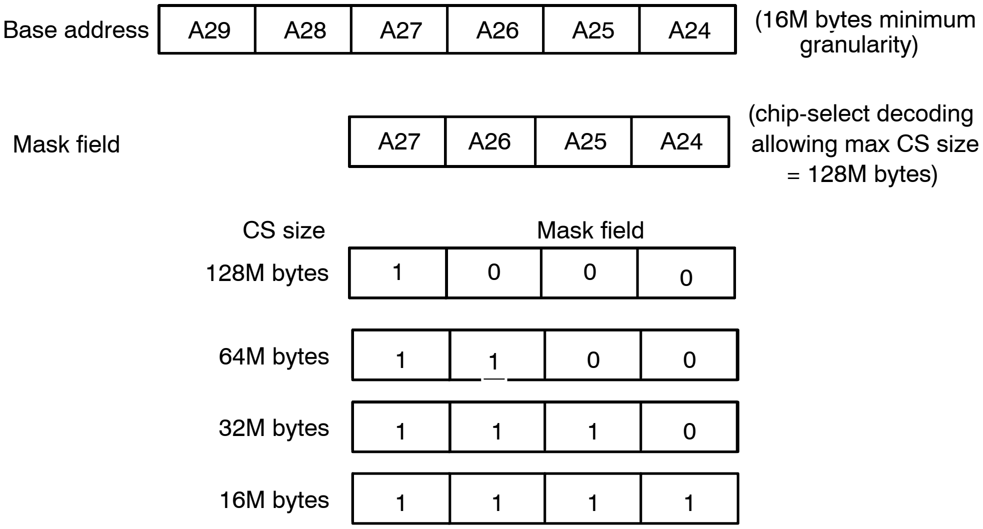
Any external memory or ASIC device attached to GPMC external interface can be accessed by any OMAP system host in the GPMC 1G-byte contiguous address space. For details, see Chapter 2, *Memory Mapping*.

The GPMC 1G-byte address space can be divided into a maximum of 8 chip-select regions with programmable base addresses and programmable sizes of 16, 32, 64, or 128M bytes.

Each chip-select has a base address register and a mask register. The two registers must be programmed with 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-two 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 12−8 (with A0 as the OMAP system byte-address line).
- 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 (the incoming address bit line is don't care). It is your responsibility to limit the register mask value to the following value according to the desired chip-select region size. Any other value has an undefined result. If multiple chip-select regions with overlapping addresses are concurrently enabled, any access to these chip-select regions is cancelled and a GPMC access error is posted.

Figure 12−8. CS Address Mapping and CS Decoding Mask



Note:

Register mask 0 value (all bits set to 0 for the 256M-byte region size) is a reserved value that is not supported in the current OMAP device.

Any chip-select configuration settings (base and mask address or any protocol and timing settings) must be done while the associated chip-select is disabled through the CSVALID bit field of the GPMC_CONFIG7[x] register. When no access to chip-select is ongoing, the chip-select configurationmust be disabled. This requires appropriate activity monitoring of the prefetch or write-posting engine if the engine is active on that chip-select. It also requires monitoring of the write buffer state to wait for any posted write completion to that chip-select. Similarly, software must ensure that the chip-select is enabled before trying to access it. To account for prefetch engine effects, after the instruction enabling a chip-select, a NOP instruction (equivalent to 64 bits) must be executed before accessing this chip-select.

Any access attempted to a nonvalid GPMC address region (CSVALID disabled or address decoding outside a valid chip-select) is cancelled and a GPMC access error is posted.

Chip-select 0 is the only chip-select enabled after a power-up or after a GPMC reset.

12.5.2 Access Protocol Configuration

12.5.2.1 Supported Devices

The access protocol of each chip-select can be independently specified through the DeviceType parameter (programmable in the GPMC_ CONFIG1[x] register) for:

- ☐ Random access synchronous or asynchronous memory like NOR flash, SRAM
- ☐ NAND flash asynchronous devices

Note:

NAND flash interfacing requires specific settings of the generic chip-select configuration parameters. Refer to NAND support for more details about the NAND flash GPMC programming model.

12.5.2.2 Access Size Adaptation and Device Width

Each chip-select can be independently configured through the DeviceSize parameter to interface with a 16-bit-wide device or an 8-bit-wide device. System requests with data widths greater than the external device data bus width are split into successive accesses according to the external device data bus width and according to little-endian data organization.

Note:

The OMAP2420 does not provide the A0 byte address line required for random byte addressable 8-bit-wide device interfacing (for both multiplexed and nonmultiplexed protocols). 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).

12.5.2.3 Address and Data Multiplexing Interface

For random synchronous or asynchronous memory interfacing (DeviceType = 00), an address and data-multiplexing protocol can be selected through the

MuxAddData parameter. The nADV signal must be used as the external device address latch control signal. For the associated chip-select configuration, the nADV assertion and deassertion time and the nOE assertion time must be set to the appropriate value so that external device address latch setup/hold time requirements are met. See Section 12.3, *GPMC Integration*.

Note:

This address and data-multiplexing interface is not applicable to NAND device interfacing. NAND devices require a specific address, command, and data multiplexing protocol. See Section 12.5.10, *NAND Device Programming Model*.

12.5.2.4 Address and Data Bus Use

See Section 12.3.3.1, *GPMC Address and Data Use Use*.

12.5.2.5 Asynchronous and Synchronous Access

For each chip-select configuration, read access can be specified to be processed asynchronously or synchronously through the ReadType parameter. For each chip-select configuration, write access can be specified to be processed asynchronously through the WriteType parameter.

Read or write access time and the related control signal are controlled through timing parameters that refer to GPMC_FCLK. This is applicable to both asynchronous and synchronous access.

The main difference of synchronous mode is the availability of a configurable clock interface (GPMC.CLK) to control the external device. The synchronous mode also affects the data capture scheme and the wait-pin monitoring scheme in read access.

For details about asynchronous and synchronous access, see Section 12.5.3.7, *GPMC.CLK and Control Signals Setup and Hold*, Section 12.5.3.8, *Access Time (AccessTime)*, and Section 12.5.4, *Wait-Pin Monitoring Control*.

For details about timing parameter settings, see the waveform examples in this chapter .

Note:

Address bus and nBE[1:0] are kept fixed for the completion of a synchronous burst read access and are updated for each beat of an asynchronous page read access.

12.5.2.6 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 the appropriate access size adaptation.

Depending on the external device page or burst capability, read and write accesses can be independently configured through ReadMultiple and WriteMultiple parameters associated with the ReadType and WriteType parameters.

Note:

- Asynchronous write page mode is not supported.
- 8-bit-wide device support is limited to nonburstable device (ReadMultiple, WriteMultiple are don't care).
- Not applicable to NAND device interfacing.

12.5.2.7 System Burst versus External Device Burst Support

The OMAP system can issue the following requests to the GPMC:

- Byte, Word16, Word32
- Incrementing fixed-length bursts of 2, 4, and 8 words
- Wrapped (critical word access first) fixed-length burst of eight words

To process the system request with the most optimized protocol, you must set the ReadMultiple (and ReadType) and WriteMultiple (and WriteType) parameters according to the burstable capability (synchronous or asynchronous) of the attached device.

The GPMC access engine issues only fixed-length burst, not continuous undefined length burst. The maximum length that can be issued is defined by the AttachedDevicePageLength parameter. The GPMC splits the system burst request in multiple burst beats when the AttachedDevicePageLength value is less than the system burst request length (including the appropriate access size adaptation according to the device width). Within the specified 4, 8, 16 words value, the AttachedDevicePageLength parameter value must correspond to the maximum burst length supported by the memory device configured in fixed burst length mode (as opposed to continuous burst mode).

To get optimized 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 WrapBurst parameter must be set to 1.

When the memory device does not offer (or is not configured to offer) native 16 Word16 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 native wrapping burst is not supported by the memory device, there is usually no behavior difference between a memory device configured in fixed burst length mode or in continuous burst mode (except for a potential power increase from memory-speculative data prefetch in continuous burst read). Since the GPMC access engine issues only fixed burst length and does not benefit from continuous burst mode, it is preferable to configure the memory device in fixed burst length mode, even though continuous burst mode is compatible with GPCM behavior.

The memory device maximum burst length (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 with smaller buffer size (4, 8) is supported as well, if the AttachedDevicePageLength parameter is set accordingly (in that case, the wrap mode must not be enabled in both memory and GPMC configuration).

The OMAP system issues only requests with an address (or starting address for nonwrapping burst request) (that is, request size boundary aligned). In case of eight words wrapping burst, the wrapping address always occurs on eight words boundary. Thus, all words requested must be available from the memory data buffer when buffer size equals or is greater than the AttachedDevicePageLength parameter value. It usually means that data can be read or written from/to the buffer at a constant rate (number of cycles between data); that is, 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.

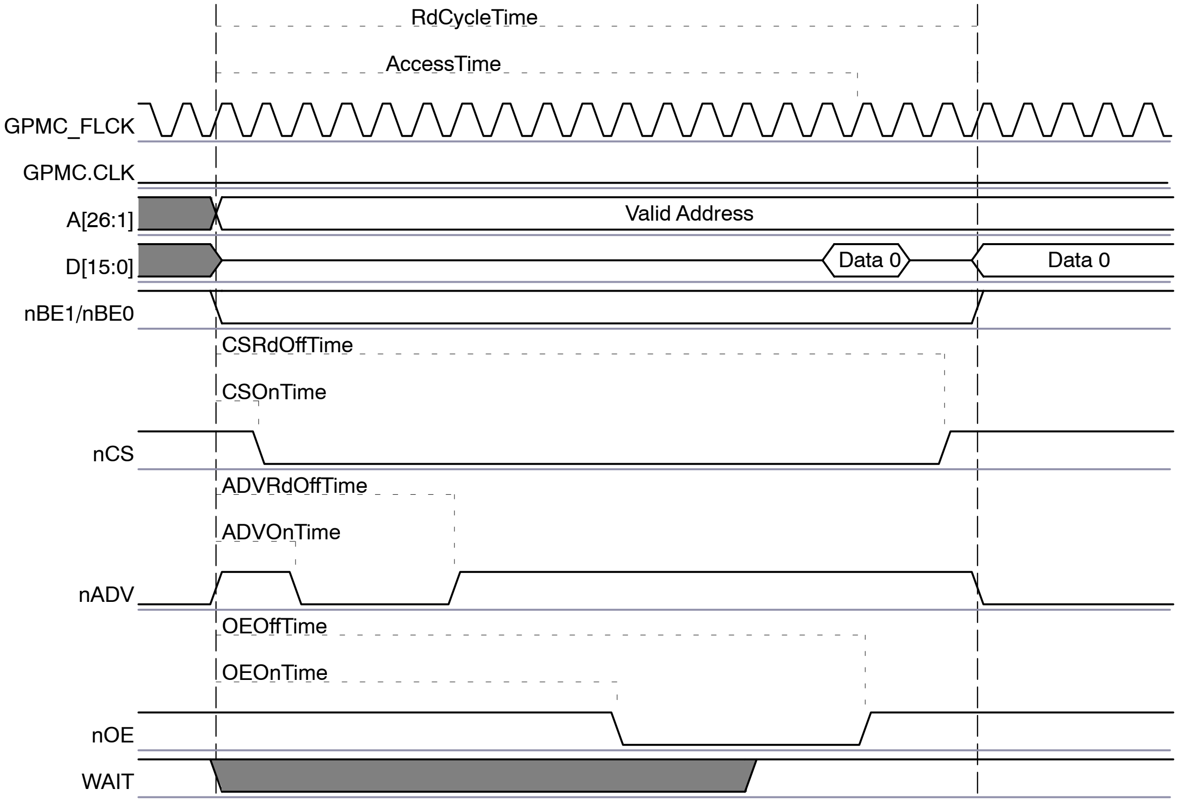
Note:

When the system burst request length is less than AttachedDevicePageLength, the GPMC proceeds with the necessary accesses.

12.5.3 Timing Setting

Most timing parameters of the protocol access used by the GPMC to interface various attached memories or devices are programmable on a chip-select basis and are described in this section. 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 12−9 illustrates an asynchronous single-read access performed on an asynchronous device.

Figure 12−9. Asynchronous Single Read on a Nonmultiplexed Address/Data Device



12.5.3.1 Read Cycle Time and Write Cycle Time (RdCycleTime/WrCycleTime)

The RdCycleTime and WrCycleTime parameters define the minimum address bus and nBE[1:0] valid time for read and write accesses. RdCycleTime and WrCycleTime are expressed in GPMC_FCLK cycles.

Read and write access time can be dynamically extended using external wait-pin monitoring. Control signal assertion and deassertion times are also delayed by wait-pin assertion. You must set the appropriate wait-pin monitoring window. When the wait signal is asserted in this window, the internal access counter is frozen and all control signals are kept at their current states.

In this document, the beginning of the access (read or write) is called the start-cycle time and the access completion is called the end-cycle time.

At end-cycle time, all control signals (nCS, nADV/ALE, nOE/nRE, nWE, nBE0/CLE) are deasserted to their respective reset values, regardless of their deassertion time parameters, if they were not already deasserted.

12.5.3.2 nCS: Chip-Select Signal Control Assertion/Deassertion Time (CSOnTime/CSRdOffTime/CSWrOffTime)

The CSOnTime parameter defines nCS signal assertion time based on start-cycle time. It is common for both read and write accesses.

For a read access, the CSRdOffTime parameter defines the nCS signal deassertion time based on start-cycle time.

For a write access, the CSWrOffTime parameter defines the nCS signal deassertion time based on start-cycle time.

12.5.3.3 nADV/ALE: Address Valid/Address Latch Enable Signal Control Assertion/Deassertion Time (ADVOnTime/ADVRdOffTime/ADVWrOffTime)

The ADVOnTime parameter defines the nADV/ALE signal assertion time based on start-cycle time. It is common to both read and write.

For a read access, the ADVRdOffTime parameter defines the nADV/ALE signal deassertion time based on start-cycle time.

For a write access, the ADVWrOffTime parameter defines the nADV/ALE signal deassertion time based on the start-cycle time.

12.5.3.4 nOE/nRE: Output Enable/Read Enable Signal Control Assertion/Deassertion Time (OEOnTime/OEOffTime)

The OEOnTime parameter defines the nOE/nRE signal assertion time based on start-cycle time. It is applicable only to read accesses.

The OEOffTime parameter defines the nOE/nRE signal deassertion time based on start-cycle time. It is applicable only to read accesses.

nOE/nRE is not asserted during the write cycle.

12.5.3.5 nWE: Write Enable Signal Control Assertion/Deassertion Time (WEOnTime/WEOffTime)

The WEOnTime parameter defines the nWE signal assertion time based on start-cycle time. It is applicable only to write accesses.

The WEOffTime parameter defines the nWE signal deassertion time based on start-cycle time. It is applicable only to write accesses.

nWE is not asserted during the read cycle.

12.5.3.6 GPMC.CLK

GPMC.CLK is the external clock provided to the attached synchronous memory or ASIC device.

GPMC.CLK is derived from GPMC_FCLK with a division factor of 1, 2, 3, or 4 set in the GPMCFCLKDIVIDER bit field.

GPMC.CLK is activated only when access is defined as synchronous read (keep asserted low during asynchronous access).

The ClkActivationTime parameter defines the number of GPMC_FCLK cycles from start-cycle time to GPMC.CLK first rising edge.

GPMC.CLK is stopped and asserted low when no access is ongoing.

Note: External Clock Cycle

To ensure the correct external clock cycle, apply the following rules:

(RdCycleTime − ClkActivationTime) must be a multiple of (GPMCFCLKDivider + 1).

The PageBurstAccessTime value must be a multiple of (GPMCFCLKDivider + 1).

12.5.3.7 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:

- ☐ The GPMC.CLK ClkActivationTime parameter 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, control signal transitions can be delayed by one half of a GPMC_FCLK period to provide minimum setup and hold times. This half-GPMC_FCLK period delay is enabled with the CSExtraDelay, ADVExtraDelay, OEExtraDelay, or WEExtraDelay parameter. This delay must be used carefully to prevent control-signal overlapping between successive accesses to different chip-selects. This implies that RdCycleTime and WrCycleTime are greater than the last control signal deassertion time including the extra half GPMC_FCLK cycle.

12.5.3.8 Access Time (AccessTime)

Access Time on Read Access

In asynchronous read mode (single and multiple access), AccessTime defines the number of GPMC_FCLK cycles from start-cycle time to the GPMC_FCLK rising edge used for the first data capture. The AccessTime parameter allows data capture time to be independent of nOE or nCS deassertion definition and allows data capture to not be done by a feedback of nOE (or nCS) rising edge.

In synchronous read mode (single and multiple access), data capture goes through a two-stage resynchronization pipeline.

The first stage corresponds to data capture on a user-defined GPMC.CLK rising edge. AccessTime defines the number of GPMC_FCLK cycles from start-cycle time to this first data capture. You must program AccessTime with the correct value corresponding to a GPMC.CLK rising edge with respect to the GPMC_FCLK divider value.

The second stage corresponds to the data internal resynchronization on GPMC_FCLK. To correctly complete the access, RdCycletime must be defined so that at least one or two GPMC_FCLK rising edges are available after access time completion.

If GPMC.CLK runs at the GPMC_FCLK frequency, RdCycleTime must be greater than or equal to AccessTime + 1.

If GPMC.CLK runs at a divided GPMC_FCLK frequency, RdCycleTime must be greater than or equal to AccessTime + 2.

In both asynchronous and synchronous modes, external wait-pin monitoring can be used to dynamically delay the capture time of the first (or single) word of a multiple (single) access. This corresponds to the assertion of the wait pin during the initial access time of the external device. AccessTime is used as a wait-invalid timing window and must be set to a correct value so that the wait pin is at a valid (asserted or not asserted) at the same cycle or a few cycles ahead of AccessTime completion. For more information about wait monitoring, see Section 12.5.4, *Wait-Pin Monitoring Control*.

Access Time on Write Access

In asynchronous write mode (single access), data are valid on the bus until completion of the access.

The AccessTime timing parameter value is don't care for the completion of the write access if wait external pin monitoring is not required to dynamically extend the write cycle time.

In asynchronous write mode with wait monitoring, WrCycleTime must be greater than or equal to AccessTime + 2.

In asynchronous write mode (single access), external wait-pin monitoring can be used to dynamically extend the write cycle time of the single word write access. This corresponds to the assertion of the wait pin during the initial access time of the external device. AccessTime is used as a wait-invalid timing window and AccessTime should be set to a correct value so that the wait pin is valid (asserted or not asserted) at the same cycle or a few cycles ahead of AccessTime completion. See Section 12.5.4, *Wait-Pin Monitoring Control*.

If wait monitoring is required for both read and write accesses, the invalid timing window specified by AccessTime is common to both read and write accesses. When wait monitoring is required only for write accesses, the AccessTime value must be a compromise between the read AccessTime requirement and the write wait-invalid timing window. For more information about wait monitoring, see Section 12.5.4 *Wait-Pin Monitoring Control*.

12.5.3.9 Page Burst Access Time (PageBurstAccessTime)

Page Burst Access Time on Read Access

PageBurstAccessTime defines the number of GPMC_FCLK cycles between successive word captures during asynchronous page (asynchronous ReadMultiple) and synchronous burst (synchronous ReadMultiple) read accesses.

As explained for synchronous mode, data capture goes through a two-stage resynchronization pipeline with the first capture stage on user-defined GPMC.CLK rising edge in ReadMultiple mode. PageBurstAccessTime must be programmed with the correct value corresponding to a GPMC.CLK rising edge with respect to the GPMC_FCLK divider value.

In both asynchronous page mode and synchronous burst mode, external wait-pin monitoring can be used to dynamically delay data capture inside the page (burst). This corresponds to the assertion of the wait pin at any time during the page (burst) access on the external device. For the successive word capture of the page (burst) PageBurstAccessTime is used as a wait-invalid timing window and PageBurstAccessTime should be set to the correct value so the wait pin is valid state (asserted or not asserted) at the same cycle or a few cycles ahead of PageBurstAccessTime completion. For more information about wait monitoring, see Section 12.5.4, *Wait-Pin Monitoring Control* .

Page Burst Access Time on Write Access

Asynchronous WriteMultiple page mode is not supported. PageBurstAccessTime has no meaning.

12.5.3.10 Idle Cycle Control Between Successive Accesses

Bus Turnaround (BusTurnAround)

To prevent data bus contention, it is necessary to delay an access following a read access to a slow memory/device (that is, nCS / nOE deasserted to data bus high-Z).

Bus turnaround starts after nCS or nOE deassertion time (whichever occurs first) and delays the next access start-cycle time.

After a read to a chip-select with BusTurnAround ≠ 0, the next access is delayed until BusTurnAround delay completion if the next access is:

- A write access to any chip-select (same or different from the previous chip-select read access)
- A read access to a chip-select different from the previous chip-select read access
- A read or write access to a chip-select associated with a multiplexed address and data device

Another way to prevent bus contention is to systematically shorten the nCS or nOE deassertion time on slow devices and extend the RdCycleTime value. Doing this prevents bus contention, but affects all accesses to this chip-select.

Idle Cycles Between Accesses to Same CS (Cycle2CycleSameCSEn, Cycle2CycleDelay)

Some devices require minimum chip-select inactive time between accesses. The Cycle2CycleSameCSEn parameter enables the insertion of a minimum number of GPMC_FCLK cycles defined by the Cycle2CycleDelay parameter between successive accesses of any type (read or write) to the same chip-select.

If Cycle2CycleSameCSEn is enabled, the next access to the same chip-select is delayed until Cycle2CycleDelay completes. Cycle2CycleDelay counter starts at CSRd/WrOffTime completion.

The same applies to successive accesses during Word32 or burst system request that are split into successive single accesses when single-access mode is used.

All control signals are kept to their default states during these idle GPMC_FCLK cycles.

Idle Cycles Between Accesses to Different CS (Cycle2CycleDiffCSEn, Cycle2CycleDelay)

Because of the pipelining behavior of the system, successive accesses to different chip-selects can occur back to back without idle cycles between accesses.

To prevent overlapping effects and to get the required control signal transition, a minimum of Cycle2CycleDelay inactive cycles are inserted between the start-cycle time to a chip-select and the previous ending access from a different chip-select. This applies to any type of access (read or write).

Note:

The Cycle2CycleDelay delay runs in parallel with the BusTurnAround delay. But BusTurnAround is a timing parameter of the finishing chip-select access, whereas Cycle2CycleDelay is a timing parameter of the following chip-select access. The effective minimum delay between successive accesses is based on the larger delay timing parameter and on access type combination (bus turnaround is not applicable to all access type combinations).

Table 12−6 shows idle cycle insertion configurations.

Table 12−6. Idle Cycle Insertion Configuration

1st Access	Bus Turn Around	2nd Access	Chip-Select	Add/Data Multi-plexed	Cycle2 Cycle SameCSEn	Cycle2 Cycle DiffCSEn	Idle Cycle Insertion Between Accesses
R/W	= 0	R/W	Any	Any	0	0	No idle cycles if the two accesses are well pipelined
R	> 0	R	Same	Non-muxed	0	0	No idle cycles if the two accesses are well pipelined
R	> 0	R	Different	Non-muxed	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 if the two accesses are well pipelined
R/W	= 0	R/W	Same	Any	1	0	Cycle2CycleDelay cycles are inserted
R/W	= 0	R/W	Different	Any	0	1	Cycle2CycleDelay cycles are inserted
R/W	> 0	R/W	Same	Any	1	0	Cycle2CycleDelay cycles are inserted. If BTA idle cycles already apply on these two back-to-back accesses. The effective delay is maximum (BusTurnAround, Cycle2CycleDelay).
R/W	> 0	R/W	Different	Any	0	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).

12.5.3.11 Slow Device Support (TimeParaGranularity)

All access timing parameters can be multiplied by 2 by setting the TIMEPARAGRANULARITY bit of the GPMC_CONFIG1[x] register. Increasing all access timing parameters permits support of very slow devices.

12.5.3.12 Write Protect (nWP)

The Write Protect signal, when connected to an attached memory device, can enable or disable the lockdown function of the attached memory.

12.5.3.13 Byte Enables (nBE1/nBE0)

Byte enable external signals (nBE0/nBE1) are not aligned with internal read request during asynchronous or synchronous single or multiple read accesses. All bytes are read externally, then byte selection occurs internally.

nBE0/nBE1 active values during read accesses are:

- For 16-bit devices:
  - nBE0 = 1b0
  - nBE1 = 1b0

- For 8-bit devices
  - nBE0 = 1b0
  - nBE1 = 1b1

12.5.3.14 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 read data 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 completion with the same data to prevent bus floating and power consumption

12.5.4 Wait-Pin Monitoring Control

GPMC access time (read and write) can be dynamically controlled by external wait-signal monitoring when external device access time is not deterministic and cannot be defined and controlled by the GPMC internal AccessTime and PageBurstAccessTime wait state generator. When enabled (WaitReadMonitoring, WaitWriteMonitoring), wait-pin monitoring enables:

- Dynamic control of the first (or single) word of a multiple (single) access corresponding to the assertion of the wait pin during the initial access time of the external device
- Dynamic access control at any time during the page (burst) access corresponding to the assertion of the wait pin at any time during the page (burst) access on the external device

The GPMC controller provides four input WAIT pins: WAITPIN0, WAITPIN1, WAITPIN2, and WAITPIN3, which allow direct plug-in and control of external devices with different wait-pin polarity:

- The WaitPinSelect parameter selects which input wait pin is used for the device attached to the corresponding chip-select. It also allows overlapping of wait-pin assertion from different devices without affecting access on a ready device for which a wait pin is not asserted.
- Wait-pin polarity is defined by the WaitxPinPolarity parameter. A WAIT pin configured to be active low means that low level on the WAIT signal (WAIT is said asserted) indicates that the data is not ready and the data bus is invalid. When the WAIT signal is deasserted, it means that data is valid.

The GPMC access engine can be configured to monitor the wait pin of the external memory device asynchronously or synchronously with GPMC.CLK, depending on whether the access type is synchronous or asynchronous (ReadType and WriteType).

12.5.4.1 Wait Monitoring During Asynchronous Read Access

When wait-pin monitoring is enabled for asynchronous read accesses (WaitReadMonitoring), the GPMC uses the programmed access time completion and the wait-deasserted state to latch the data bus.

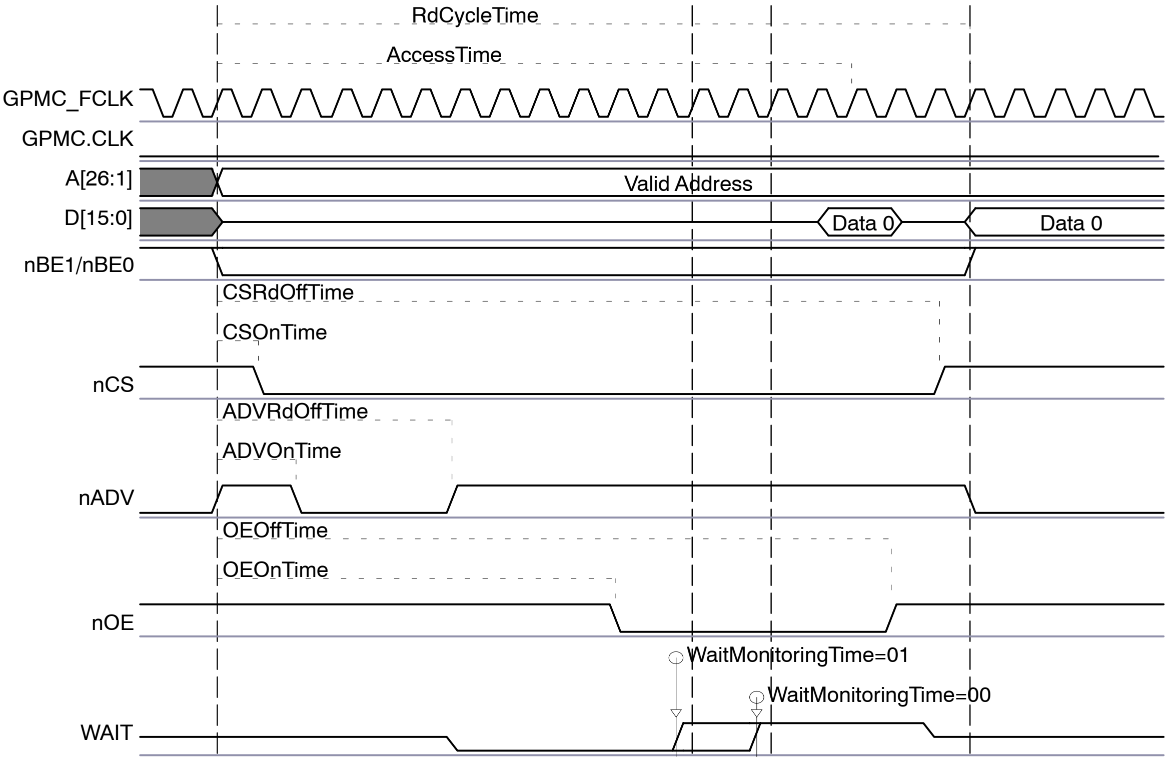
Because of internal resynchronization of the wait signal in the GPMC, the AccessTime parameter must be programmed so that the wait pin is valid a minimum of two GPMC_FCLK clock cycles before AccessTime completion (that is, the wait pin must be valid with required ac timing setup at AccessTime – 2). This wait-invalid timing window must be defined based on the attached memory characteristics.

- A wait monitored as asserted freezes the access (internal cycle counter is frozen). Control signals are kept at their current states. The data bus is considered invalid and is not captured.
- A wait monitored as deasserted enables access completion (the internal cycle counter is unlocked); the data bus is considered valid and is captured. The read access completes according to its configuration. All control signals are treated according to their respective timing values.

For multiple read accesses (asynchronous page mode), a wait is monitored (considered as valid) on every PageBurstAccessTime completion. As described previously, the signal must be valid (including ac timing setup) two GPMC_FCLK clock cycles ahead of PageBurstAccessTime completion. A wait monitored as deasserted completes the current access in the page (data is captured) and starts the next access in the page.

A slow device can present a significant delay between wait-pin deassertion and effective data valid time (including the required GPMC data ac timing setup). When this delay is greater than the two wait resynchronization GPMC_FCLK cycles, data cannot be sampled correctly. An extra delay can be added between wait-pin deassertion time detection and effective data capture time. This extra delay can be programmed in the WAITMONITORINGTIME bit field. The WaitMonitoringTime does not delay wait-pin assertion or deassertion detection time and does not suppress the 2 GPMC_FCLK pipelined detection. This extra delay is expressed in GPMC.CLK cycles, even though the access is defined as asynchronous and no GPMC.CLK is provided to the external device. Still, GPMCFCLKDivider is used as a divider from GPMC_FCLK, and it should be programmed to define the correct WaitMonitoringTime delay.

Figure 12−10. Wait Behavior During an Asynchronous Single Read Access (GPMCFCLKDivider = 1)



Note: WAIT signal is active low, WaitMonitoringTime = 00, 01.

12.5.4.2 Wait Monitoring During Asynchronous Write Access

When wait-pin monitoring is enabled for asynchronous write accesses (WaitWriteMonitoring), the wait-invalid timing window is defined by AccessTime. Because of internal resynchronization of the WAIT signal, the wait pin must be valid state two GPMC_FCLK cycles before AccessTime completion (the wait pin must be valid with the required ac timing setup at AccessTime – 2).

- ❑ A wait monitored as asserted freezes the access (the internal cycle counter is frozen). Control signals are kept at their current states. The access is on hold and the GPMC keeps the data bus driven.
- ❑ A wait monitored as deasserted enables access completion (the internal cycle counter is unlocked). The write access completes according to its configuration. All signals, including the data bus, are controlled according to their corresponding control timing values.

Asynchronous write page mode is not supported. Therefore, wait monitoring is not linked to PageBurstAccessTime completion for asynchronous write accesses.

A slow device can cause a significant delay between wait-pin deassertion time and the device data setup time requirement before write access completion (nWE or nCS deassertion). When this delay is greater than the two wait resynchronization GPMC_FCLK cycles, an extra delay can be added between wait-pin deassertion time detection and completion of the write access (unlock of the internal cycle counter). This extra delay can be programmed in the WAITMONITORINGTIME bit field. The WaitMonitoringTime does not delay wait-pin assertion or deassertion detection time and does not suppress the 2 GPMC_FCLK pipelined detection. This extra delay is expressed in GPMC.CLK cycles, even though the access is defined as asynchronous, and even though no GPMC.CLK is provided to the external device. Still, GPMCFCLKDivider is used as a divider from GPMC_FCLK, and it must be programmed to define the correct WaitMonitoringTime delay.

Another way to provide the appropriate extra data setup is to delay nWE or nCS deassertion from AccessTime completion so that an extra GPMC_FCLK cycle occurs between wait-deassertion detection and nWE and nCS deassertion.

12.5.4.3 Wait Monitoring During Synchronous Read Access

When wait-pin monitoring is enabled for synchronous read accesses (WaitReadMonitoring) the GPMC uses both the programmed AccessTime completion and the wait-deasserted state to latch the data bus. During synchronous accesses, the wait pin is captured synchronously with the rising GPMC.CLK edge.

The wait-monitoring cycle (that is, the GPMC.CLK edge on which the wait pin state is considered) can be the same as the data capture cycle it is supposed to handshake (or not capture if wait is asserted). It can also be one or two GPMC.CLK cycles ahead of the data cycle it is supposed to handshake. This pipelining support is effective for the whole burst access, and the wait-monitoring advance phase is applicable to every data phase of the burst access. This wait advance pipelining monitoring is programmed in the WAITMONITORINGTIME bit field and is expressed in GPMC.CLK cycles.

Depending on the WaitMonitoringTime programmed value, the WAIT pin should be valid (asserted or deasserted) on the rising edge of GPMC.CLK:

- Corresponding to AccessTime completion, which is the same as data valid when WaitMonitoringTime = 0
- Corresponding to WaitMonitoringTime x (GPMCFCLKDivider + 1) GPMC_FCLK cycles before AccessTime completion when WaitMonitoringTime ≠ 0

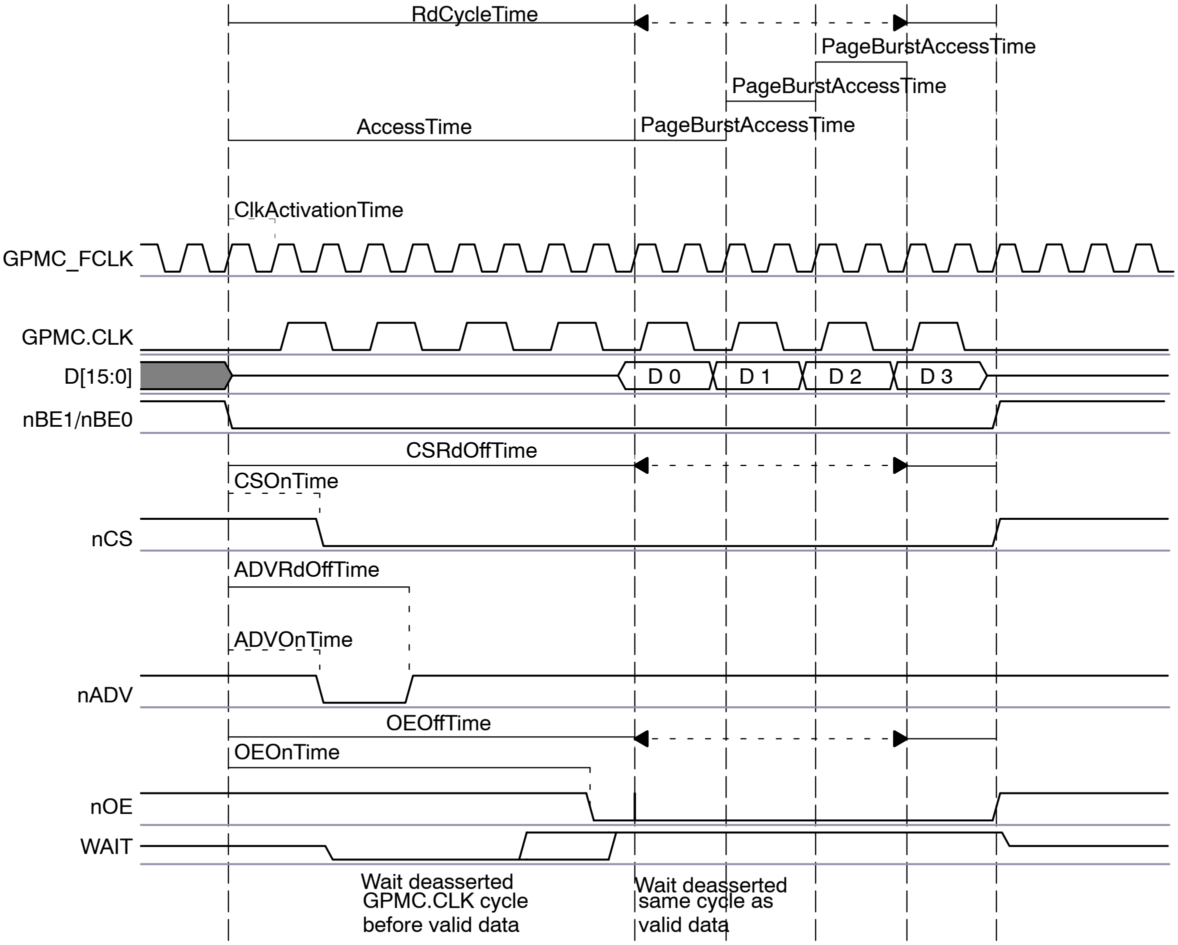
At the external wait-pin monitoring time:

- A wait monitored as asserted freezes the access (the internal cycle counter is frozen). Control signals are kept at their current state. The data bus is considered invalid and is not captured.
- A wait monitored as deasserted enables access completion (the internal cycle counter is unlocked); the data bus is considered valid and is captured. The read access completes according to its configuration. All control signals are treated according to their corresponding control timing values.

For multiple read accesses (synchronous burst mode), the wait pin is monitored (consider as valid including ac timing setup) on every PageBurstAccessTime completion or, as described previously, one or two GPMC_FCLK clock cycles ahead of PageBurstAccessTime completion according to the WaitMonitoringTime value.

A wait monitored as deasserted completes the current access in the burst (data is captured) and starts the next access in the burst.

Figure 12−11. Wait Behavior During a Synchronous Read Burst Access



Note: The WAIT signal is active low, WaitMonitoringTime = 00, 01.

12.5.4.4 Wait Monitoring During Synchronous Write Access

Wait monitoring is not supported for synchronous write access.

12.5.5 GPMC.IO_DIR Signal

The GPMC.IO_DIR signal is used for the GPMC data bus D[15:0] I/O direction. Depending on the top level pad multiplexing (see Table 12−4), this signal can be output and used externally to the OMAP2420.

The GPMC.IO_DIR signal is low during transmit (OUT) and high during receive (IN).

The reset value at boot time of the GPMC.IO_DIR is IN to allow transparent I/O sharing (see Section 12.5.3.13, *Byte Enables (nBE1/nBE0)*).

For write accesses, GPMC.IO_DIR stays OUT from start-cycle time to end-cycle time.

For read accesses, GPMC.IO_DIR goes from OUT to IN at OE# assertion time and stays IN until:

- ❑ If BusTurnAround is enabled:
  - ■ The GPMC.IO_DIR goes from IN to OUT at end-cycle time plus the programmable bus turnaround time
- ❑ If BusTurnAround is disabled:
  - ■ After an asynchronous read access, the GPMC.IO_DIR goes from IN to OUT at AccessTime + 1 GPMC_FCLK cycle or at RdCycleTime completion, whichever occurs last.
  - ■ After a synchronous read access, the GPMC.IO_DIR goes from IN to OUT at AccessTime + 2 GPMC_FCLK cycles or at RdCycleTime completion, 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 signal is OUT (see Section 12.5.3.14, *Bus Keeping Support*).

To prevent unnecessary toggling, the GPMC.IO_DIR stays IN between two successive read accesses to a nonmultiplexed device.

12.5.6 Generic Access Description

12.5.6.1 Asynchronous Access Description

In asynchronous operations:

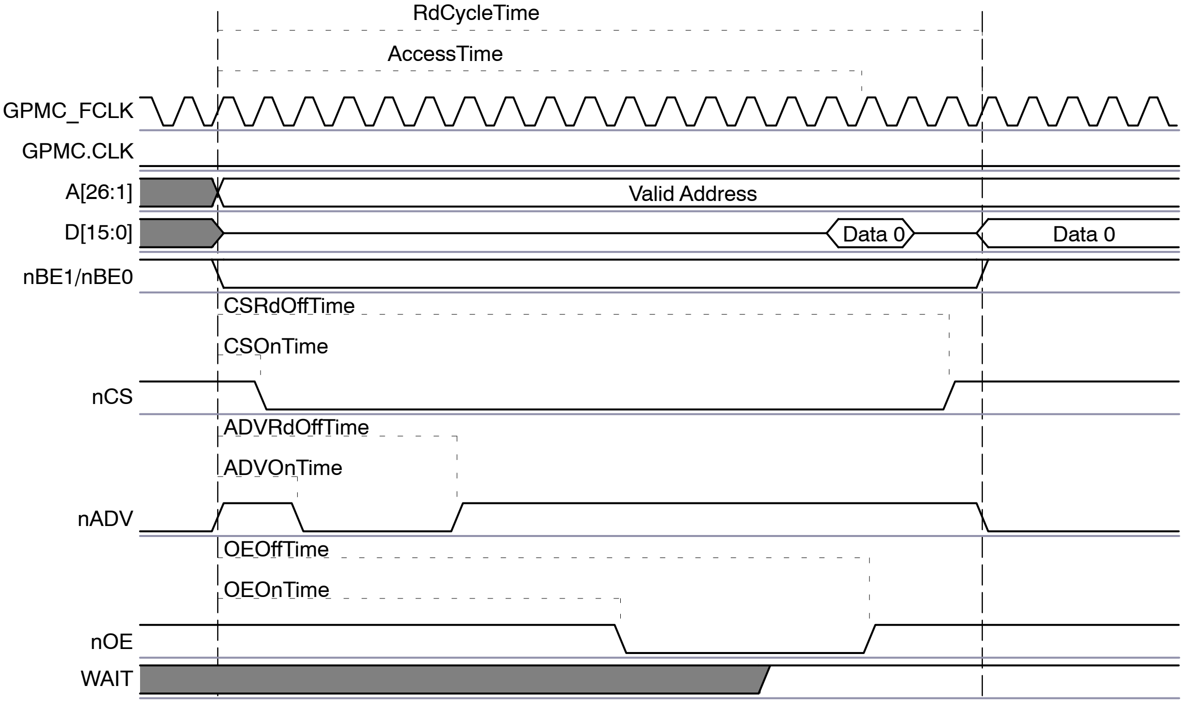
- ❑ GPMC.CLK is not provided outside the GPMC.
- ❑ The output GPMC.CLK signal is kept low.

Asynchronous Single Read

Asynchronous Single Read Operation on a Nonmultiplexed Device

Figure 12−12 shows an asynchronous single read operation on a nonmultiplexed device.

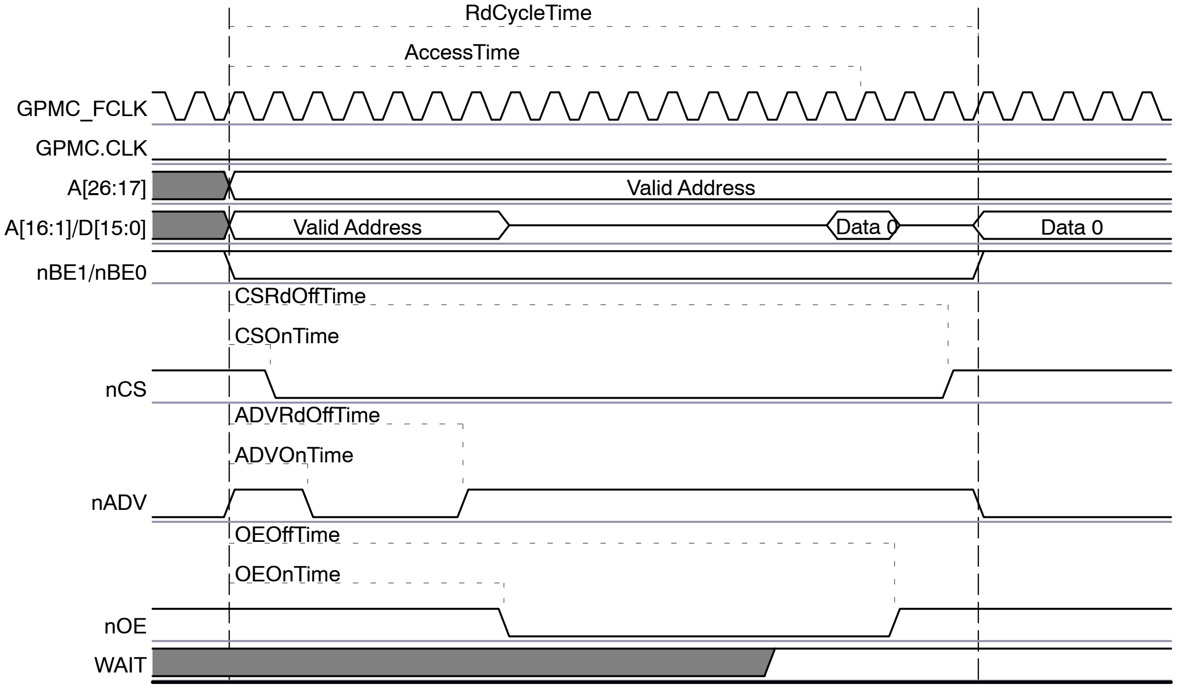
Figure 12−12. Asynchronous Single Read on an Address/Data Nonmultiplexed Device



Asynchronous Single Read Operation on an Address and Data Multiplexed Device

Figure 12−13 shows an asynchronous single write operation on an address and data multiplexed device

Figure 12−13. Asynchronous Single Read on an Address/Data Multiplexed Device



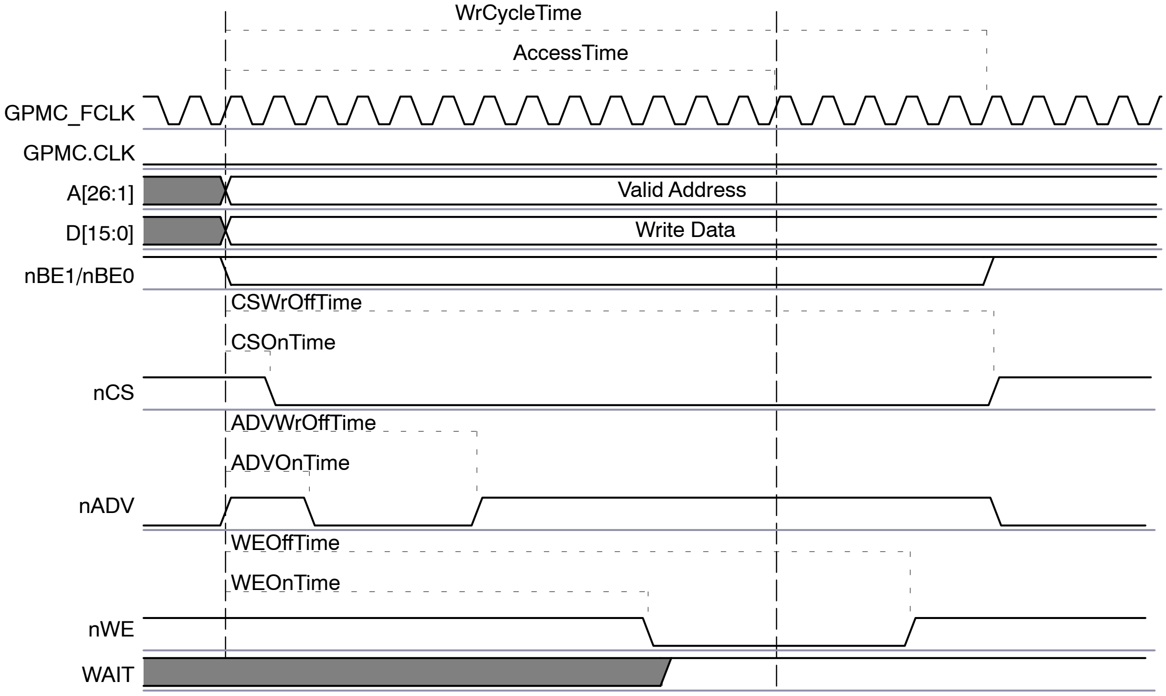
The GPMC drives the address bus until nOE assertion time when generating a read access to an address/data multiplexed device. For more information, see Section 12.5.2.3, *Address and Data Multiplexing Interface*.

Asynchronous Single Write

Asynchronous Single Write Operation on a Non-multiplexed Device

Figure 12−14 shows an asynchronous single write operation on a nonmultiplexed device.

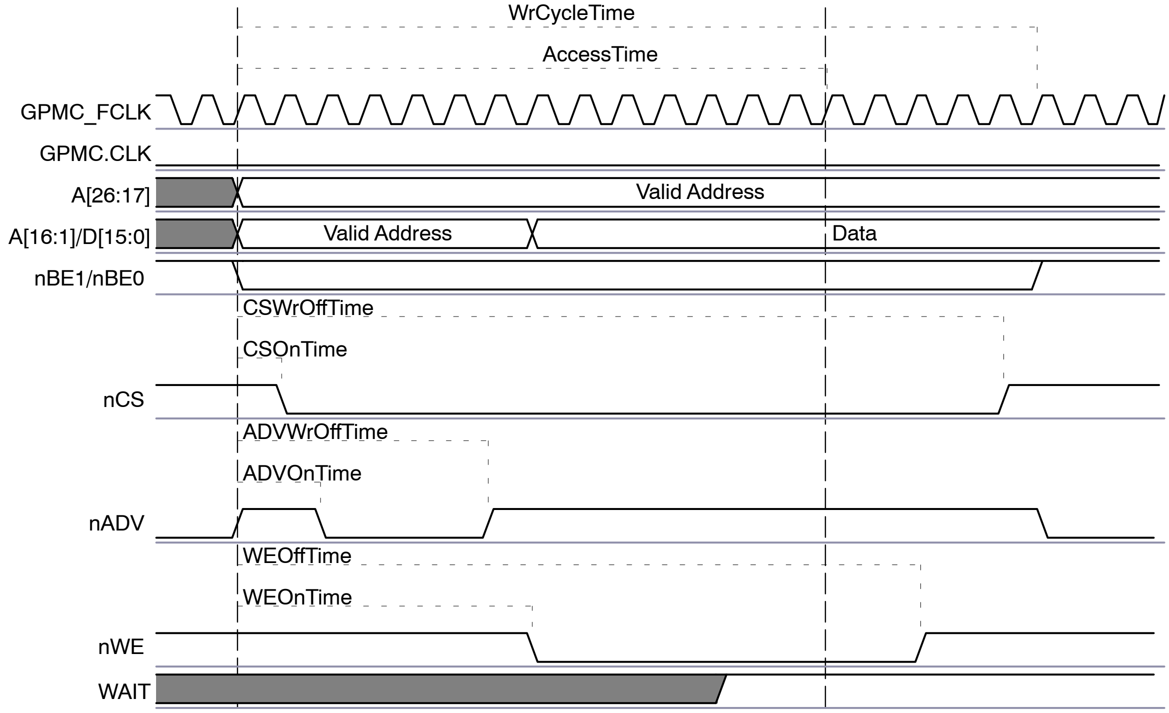
Figure 12−14. Asynchronous Single Write on an Address/Data Nonmultiplexed Device



Asynchronous Single Write Operation on an Address and Data Multiplexed Device

Figure 12−15 shows an asynchronous single write operation on an address and data multiplexed device.

Figure 12−15. Asynchronous Single Write on an Address/Data Multiplexed Device

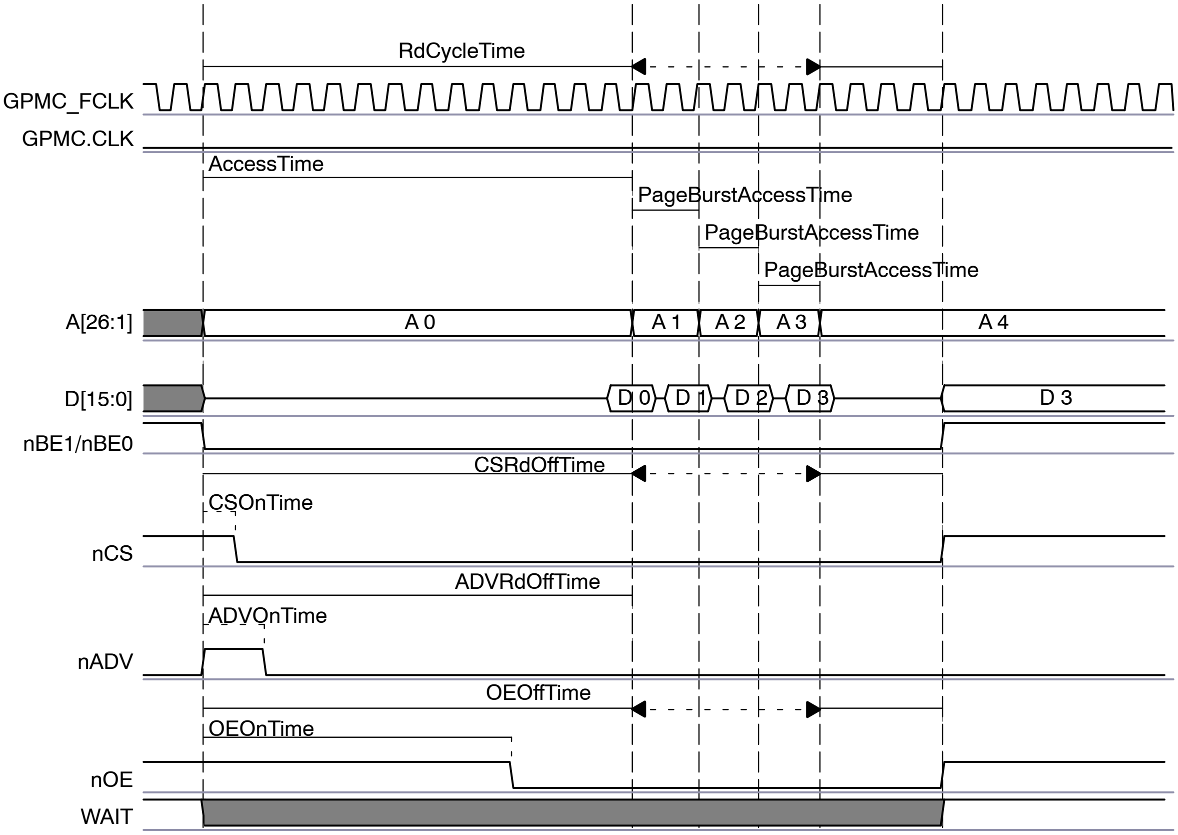


The GPMC drives the address bus until nWE assertion time when generating a write access to an address/data multiplexed device. For more information, see Section 12.5.2.3, *Address and Data Multiplexing Interface*.

Asynchronous Multiple (Page Mode) Read

Figure 12−16 shows an asynchronous multiple read operation.

Figure 12−16. Asynchronous Multiple (Page Mode) Read



At AccessTime completion, any control-signal timing is frozen during the multiple data transactions, corresponding to PageBurstAccessTime multiplied by the number of remaining data transactions.

During consecutive accesses, the GPMC increments the address after each data read completion.

The delay between successive data reads in the page is controlled by PageBurstAccessTime.

The total access time, RdCycleTime that must be programmed, corresponds to the AccessTime plus the remaining time required until access completion. The same comment applies to control signal OffTime parameters.

Note: Asynchronous Page Mode Read

Asynchronous page mode read is not supported for address and data muxed mode.

12.5.6.2 Synchronous Access Description

In synchronous operations, GPMC.CLK is provided outside the GPMC when accessing the memory device. Synchronous write burst is supported if GPMCFCLKDivider is not 0.

Synchronous Single Read

Figure 12−17 and Figure 12−18 show a synchronous single read operation with GPMCFCLKDivider equal to 0 and 1, respectively.

Figure 12−17. Synchronous Single Read (GPMCFCLKDivider = 0)

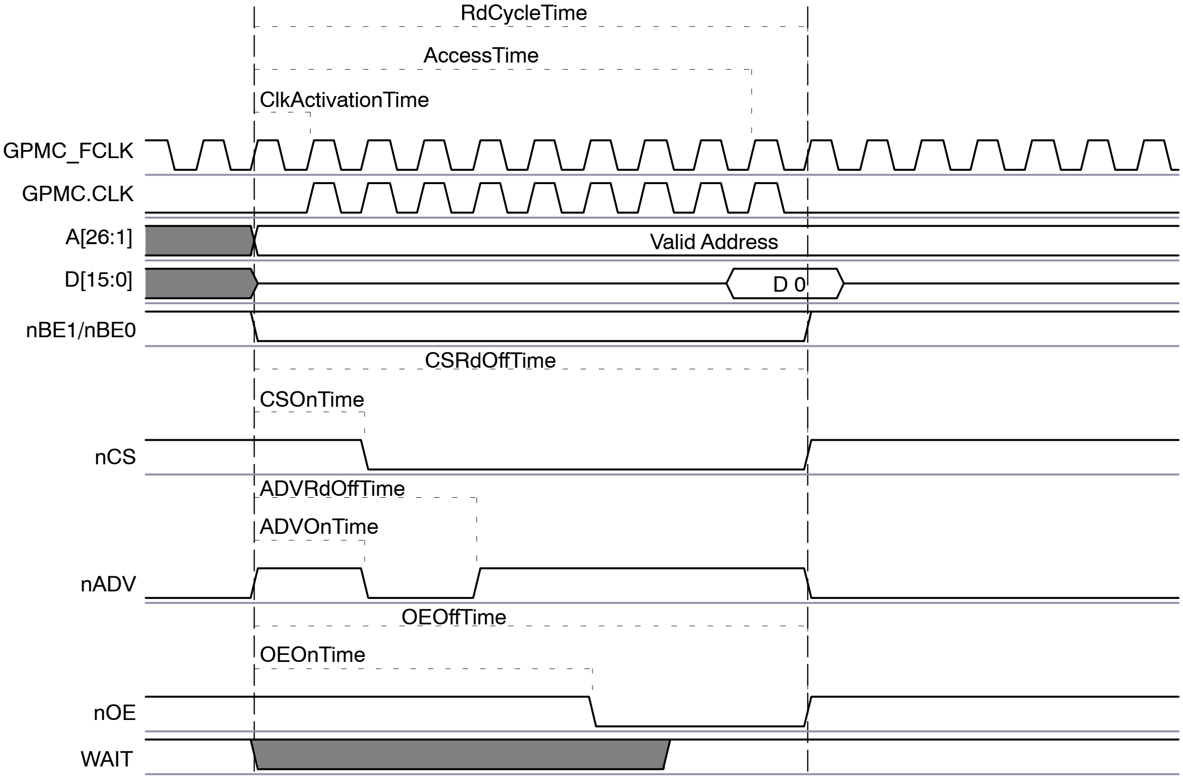
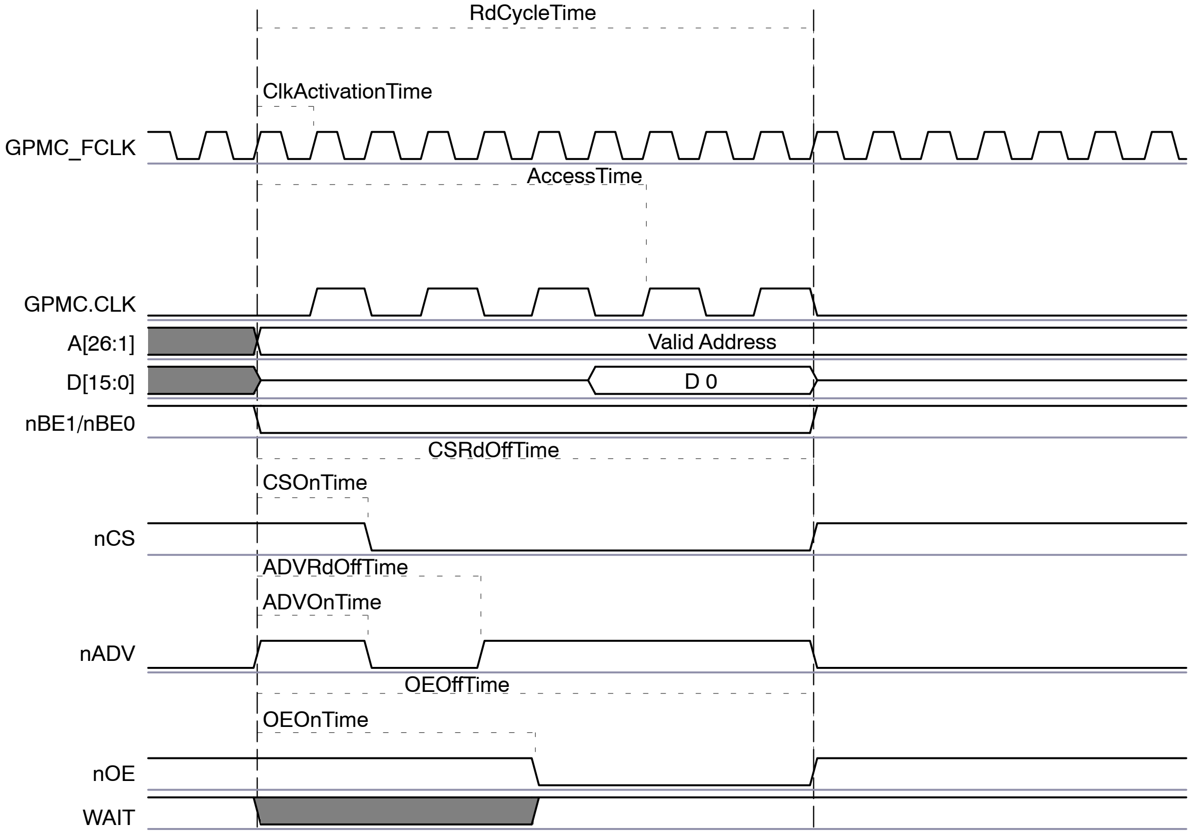


Figure 12−18. Synchronous Single Read (GPMCFCLKDivider = 1)



Synchronous Multiple (Burst) Read (4−8−16 Word Burst With Wraparound Capability)

Figure 12−19 and Figure 12−20 show a synchronous multiple read operation with GPMCFCLKDivider equal to 0 and 1, respectively.

Figure 12−19. Synchronous Multiple (Burst) Read (GPMCFCLKDivider = 0)

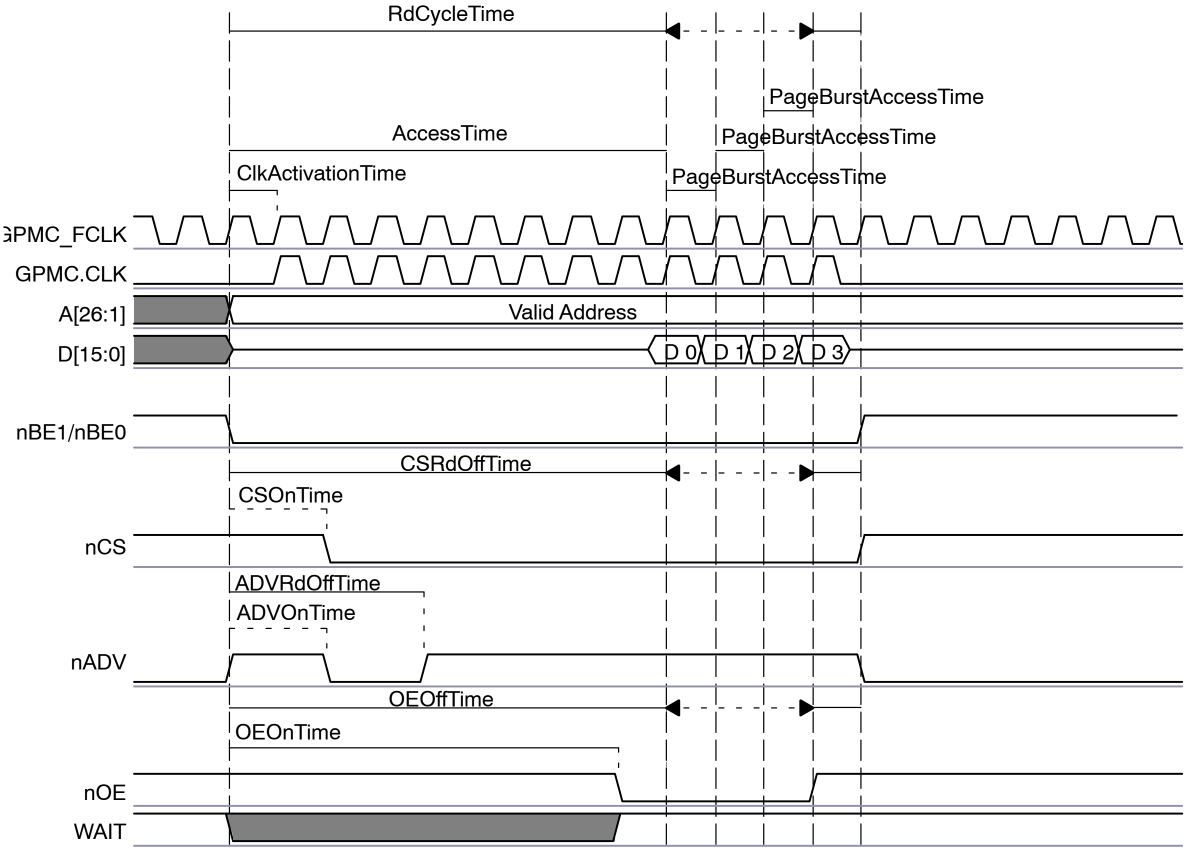
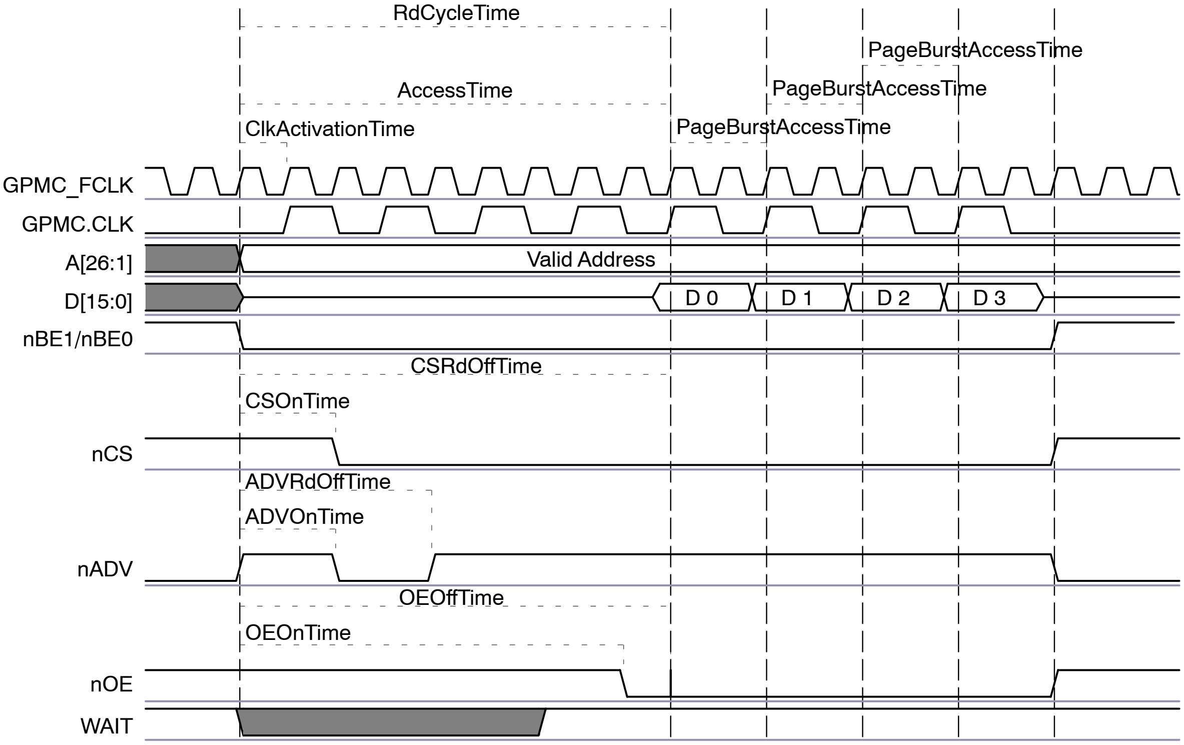


Figure 12−20. Synchronous Multiple (Burst) Read (GPMCFCLKDivider = 1)



At AccessTime completion, control-signal timing is frozen during the multiple data transactions, corresponding to PageBurstAccessTime multiplied by the number of remaining data transactions.

Initial latency for the first read data is controlled by AccessTime or by monitoring the WAIT signal.

Successive data reads are provided by the memory device at each GPMC.CLK cycle or every two GPMC.CLK cycles. The PageBurstAccessTime parameter must be set according to the GPMCFCLKDivider and memory device internal configuration.

Depending on the device page length, the GPMC can control device page crossing during a burst request by delaying the next data capture if wait monitoring is enabled.

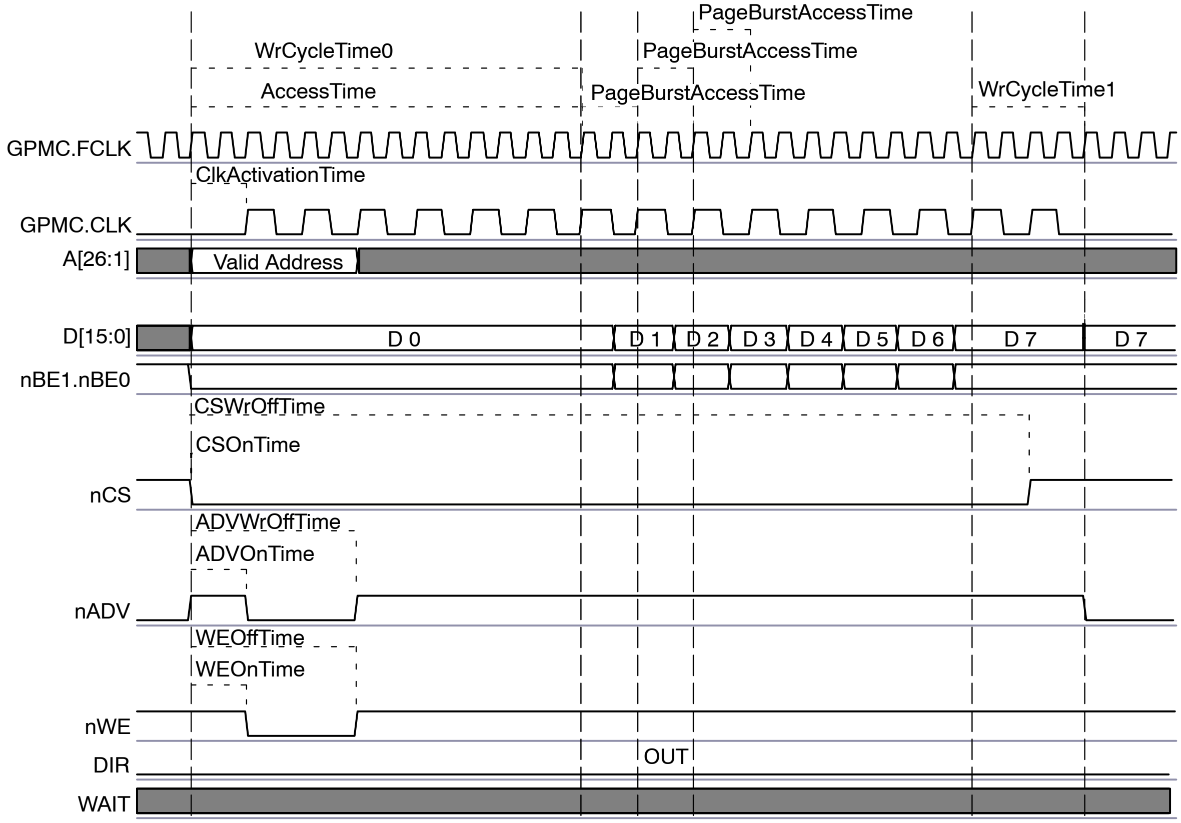
WARNING

In synchronous burst write mode, a GPMCFCLKDIVIDER value of 0 is not supported. The GPMC_CLK frequency must be lower or equal to GPMC_FCLK.

Synchronous Burst Write

Burst write mode provides synchronous single or consecutive accesses.

Figure 12−21. Synchronous Multiple (Burst) Write (GPMCFCLKDivider = 1)



Note: The WAIT signal is active low.

At AccessTime completion, control signal timing is frozen during the multiple data transactions, corresponding to PageBurstAccessTime multiplied by the number of remaining data transactions.

The first write data is driven by the GPMC at start-cycle time. The next write data of the burst is driven on the bus at AccessTime + 1 during PageBurstAccessTime GPMC.FCLK cycles. the last data of the synchronous burst write is driven until WrCycleTime completion.

Successive write data must be provided to the memory device each GPMC.CLK cycle.

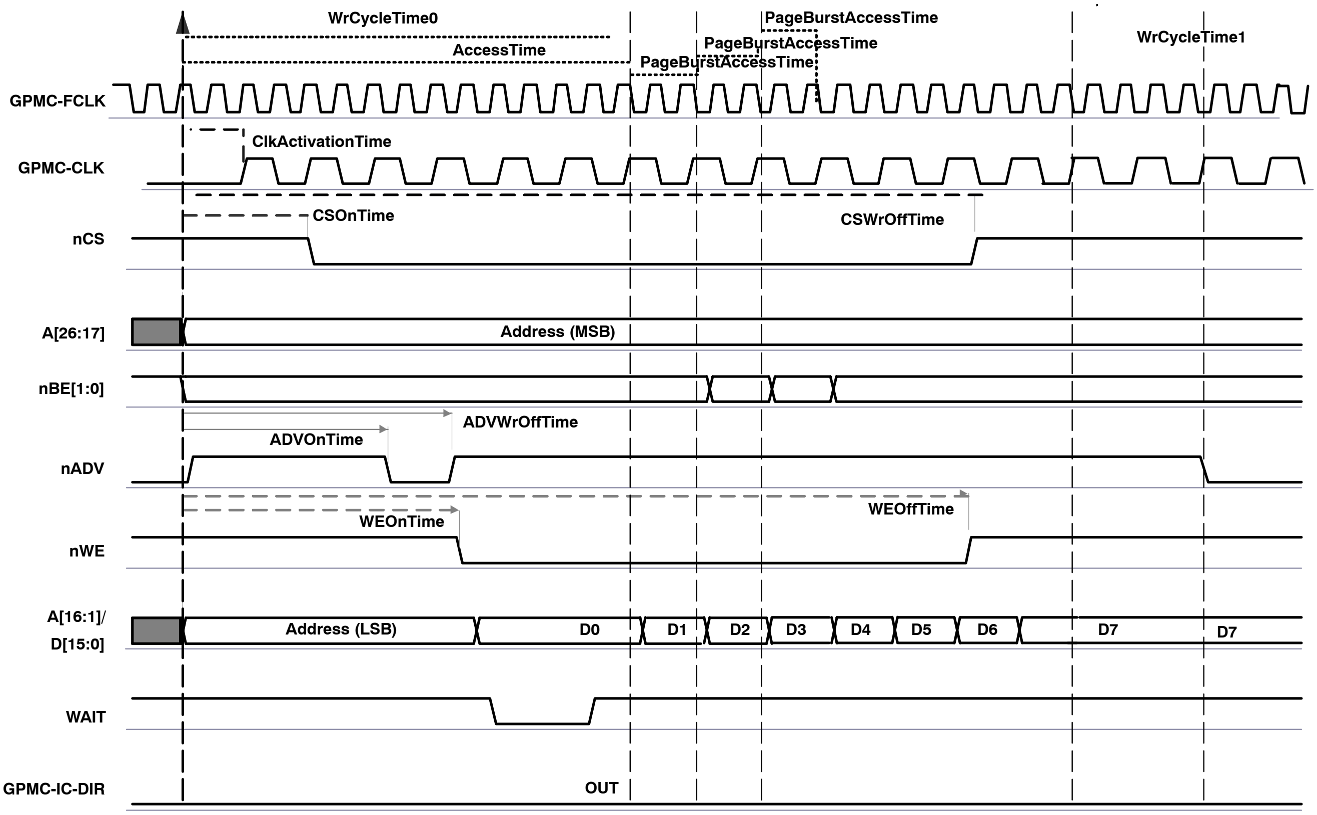
The total access time, WrCycleTime, that must be programmed corresponds to the AccessTime plus the remaining time needed until access completion (WrCycleTime0 + WrCycleTime1).

Note:

The following rules must be applied for correct operation:

- ☐ PageBurstAccessTime value must be a multiple of (GPMCFCLKDivider + 1).
- ☐ (AccessTime − ClkActivationTime) modulus (GPMCFCLKDivider + 1) must be different from GPMCFCLKDivider.

Figure 12−22. Synchronous Burst Write (GPMCFCLKDivider = 1) on a Multiplexed Address/Data Device



When generating a burst write access to an address/data multiplexed device, the GPMC drives the address on the multiplexed address/data bus until nWE is asserted. The GPMC drives the first data of the burst on the multiplexed address/data bus as soon as nWE is asserted.

Consequently, care must be taken to ensure that nADV is deasserted before nWE is asserted.

12.5.7 Write Buffer

The GPMC write buffer is used to store up to eight system write requests before processing single write accesses to external memory.

Up to eight consecutive posted-write accesses can be accepted and stored into the write buffer.

For nonposted write requests, the pipeline depth is 1.

The EMPTYWRITEBUFFERSTATUS status bit in the GPMC_STATUS register stores the empty status of the write buffer.

This status bit can be used to check that all posted write transactions stored in the write buffer are complete and the write buffer is empty. For example, this can be used to check that all write transactions to a particular chip-select configuration register file are complete before disabling this chip-select.

12.5.8 Error Handling

Different types of errors occur in the GPMC. When an error occurs, the error type information is stored in the GPMC_ERR_TYPE register and the address of the illegal access is stored in the GPMC_ERR_ADDRESS register. The GPMC keeps the first error abort information until the GPMC_ERR_TYPE register is reset. Subsequent accesses that cause errors are not logged until the error is cleared by hardware with the ERRORVALIDBIT field.

- ☐ 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 attempted outside the address valid range of 1K bytes.
- ☐ ErrorNotSuppMCmd occurs when an unsupported command request is decoded at the L3 interconnect interface.
- ☐ Error time-out: A time-out mechanism prevents the system from hanging. The start value of the 9-bit time-out counter is defined in the GPMC_TIMEOUT_CONTROL register and enabled with the TIMEOUTEN bit. When enabled, the counter starts at start-cycle time until it reaches 0 and data is not responded to from memory; a time-out error occurs. When data is sent from memory, this counter is reset to its start value. In case of multiple accesses (asynchronous page mode or synchronous burst mode), the counter is reset to its start value for each data access in the burst.

The GPMC does not generate interrupts on these types of errors. True abort to the MPU or interrupt generation is handled at the L3 level.

12.5.9 Boot Configuration

See Section 12.3.3.4, *GPMC CS0 Default Configuration at IC Reset*.

12.5.10 NAND Device Programming Model

NAND (8-bit and 16-bit) memory devices using a classical NAND asynchronous address and data-multiplexing scheme can be supported on any chip-select with the appropriate asynchronous configuration settings.

As with any other chip-select, the accesses to a chip-select allocated to a NAND device can be interleaved with accesses to chip-selects allocated to other external devices (the GPMC interface is not dedicated to a NAND device). This interleaved capability limits the chip enable don’t care NAND device support, because the chip-select allocated to the NAND device must be deasserted if accesses to other chip-selects are requested.

12.5.10.1 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. GPMC NAND support is data stream oriented (byte or Word16), as opposed to random memory map device support.

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 should access the NAND device ID to ensure that correct command and address formatting is 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 nBE0/CLE as Command Latch Enable (CLE active high, default state value at low) during command program access. The GPMC address lines are not used (previous value not changed) during a NAND access.

Chip-Select Configuration for NAND Interfacing in Byte or Word Stream Mode

The GPMC_CONFIG7_CS[x] register associated with a NAND device region interfaced in byte or word stream mode can be initialized with the minimum size of 16M bytes, since only one address location in the chip-select memory region is used to access NAND data array.

The GPMC_CONFIG1_CS[x] register associated with a NAND device region must be initialized in asynchronous read and write modes with the parameters given in Table 12−7 to allow correct command, address, and data access controls. Not complying with the following settings corrupts the NAND interface protocol.

Table 12−7. Chip-Select Configuration for NAND Interfacing

Bit Field	Register	Value	Comments
WrapBurst	GPMC_CONFIG1_CSx	0	No wrap
ReadMultiple	GPMC_CONFIG1_CSx	0	Single access
ReadType	GPMC_CONFIG1_CSx	0	Asynchronous mode
WriteMultiple	GPMC_CONFIG1_CSx	0	Single access
WriteType	GPMC_CONFIG1_CSx	0	Asynchronous mode
ClkActivationTime	GPMC_CONFIG1_CSx	00	
AttachedDevicePageLength	GPMC_CONFIG1_CSx	Don’t care	Single access mode
WaitReadMonitoring	GPMC_CONFIG1_CSx	0	Wait not monitored by GPMC access engine
WaitWriteMonitoring	GPMC_CONFIG1_CSx	0	Wait not monitored by GPMC access engine
WaitMontoringTime	GPMC_CONFIG1_CSx	Don’t care	Wait not monitored by GPMC access engine
WaitPinSelect	GPMC_CONFIG1_CSx		Select which wait is monitored by edge detectors

Table 12−7. Chip-Select Configuration for NAND Interfacing (Continued)

Bit Field	Register	Value	Comments
DeviceSize	GPMC_CONFIG1_CSx	00 or 01	
DeviceType	GPMC_CONFIG1_CSx	10	NAND device in stream mode
MuxAddData	GPMC_CONFIG1_CSx	0	Non-multiplexed mode
TimeParaGranularity	GPMC_CONFIG1_CSx	0	Timing achieved with best GPMC clock granularity
GPMCFCLKDivider	GPMC_CONFIG1_CSx	Don't care	Asynchronous mode

The GPMC_CONFIG1_CS[x] to GPMC_CONFIG4_CS[x] register associated with a NAND device region must be initialized with the correct control signal timing value according to the NAND device timing parameters.

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 with the correct value 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_NAND_COMMAND_CS[x] and GPMC_NAND_ADDRESS_CS[x] location with the correct command and address values. The GPMC_NAND_COMMAND_CS[x] and GPMC_NAND_ADDRESS_CS[x] 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.
- To improve performance and prevent host stalling, the successive address and command words should be posted in the GPMC write buffer. The NANDForcePostedWrite configuration bit in the GPMC_CONFIG register enables write accesses to GPMC_NAND_COMMAND_CS[x] and GPMC_NAND_ADDRESS_CS[x] to be systematically posted even if accesses to GPMC register file should be defined as a noncacheable nonbufferable (nonposted write) region.
- Up to 8 consecutive posted-write accesses can be accepted and stored in the write buffer. For nonposted write and read requests, the pipeline depth is 1.
- The EMPTYWRITEBUFFERSTATUS status bit in the GPMC_STATUS register stores the empty status of the write buffer.
- GPMC_NAND_COMMAND_CS[x] and GPMC_NAND_ADDRESS_CS[x] are Word32 locations, which means that any Word32 or Word16 ac-

cess is split into 4-byte and 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 carefully consider the splitting and little-endian ordering scheme. When only one byte command or address phase is required, only byte write access to GPMC_NAND_COMMAND_CS[x] and GPMC_NAND_ADDRESS_CS[x] should be used, and any of the four byte locations of the registers are valid.

- ❏ The same applies in case of GPMC_NAND_COMMAND_CS[x] and GPMC_NAND_ADDRESS_CS[x] Word32 write access to a 16-bit wide NAND device (split into 2 Word16 access). In case of Word16 write access, the MSByte of the word16 value must be set according to the NAND device requirement (usually 0). Any of two Word16 locations or any of the four byte locations of the registers are valid.

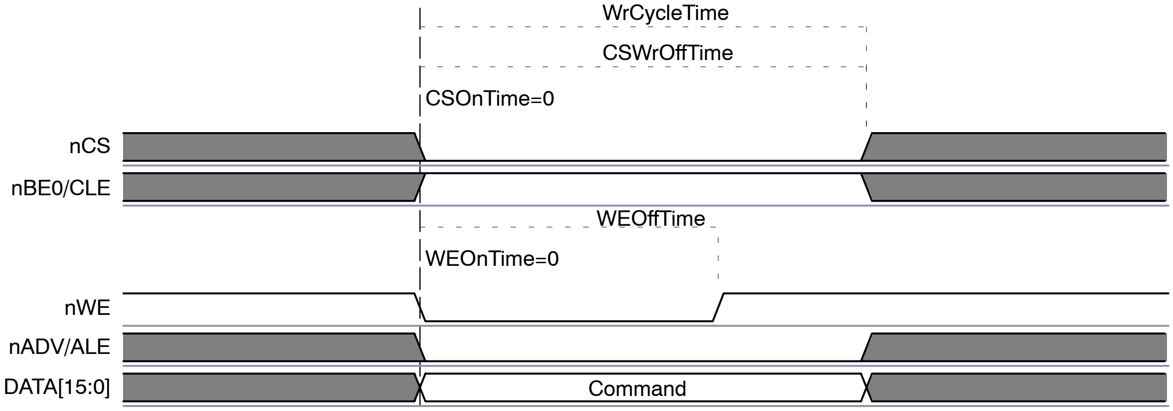
Command Latch Cycle

Writing data at the GPMC_NAND_COMMAND_CS[x] location places the data as the NAND command value on the bus according to an 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 12−23 shows the NAND command latch cycle.

Figure 12−23. NAND Command Latch Cycle



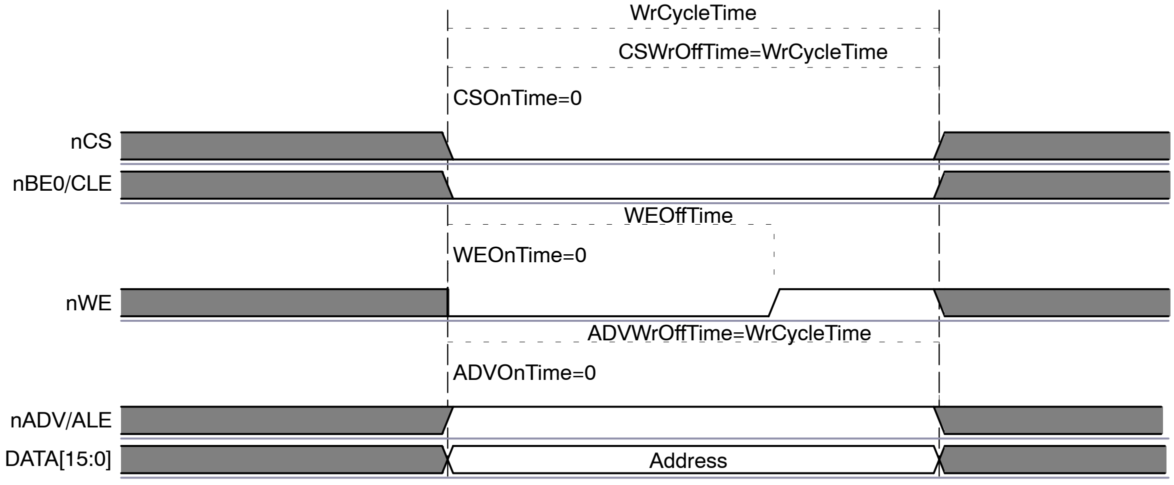
Address Latch Cycle

Writing data at the GPMC_NAND_ADDRESS_CS[x] location places the data as the NAND partial address value on the bus according to 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 12−24 shows the NAND address latch cycle.

Figure 12−24. NAND Address Latch Cycle



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_NAND_DATA_CS[x] location mapped in the chip-select associated control register region. The associated chip-select signal timing control must be programmed according to the NAND device timing specification

Note:

GPMC_NAND_DATA_CS[x] is not a physical register, but an address location that should not be read back after write access (read accesses are routed directly to the NAND device).

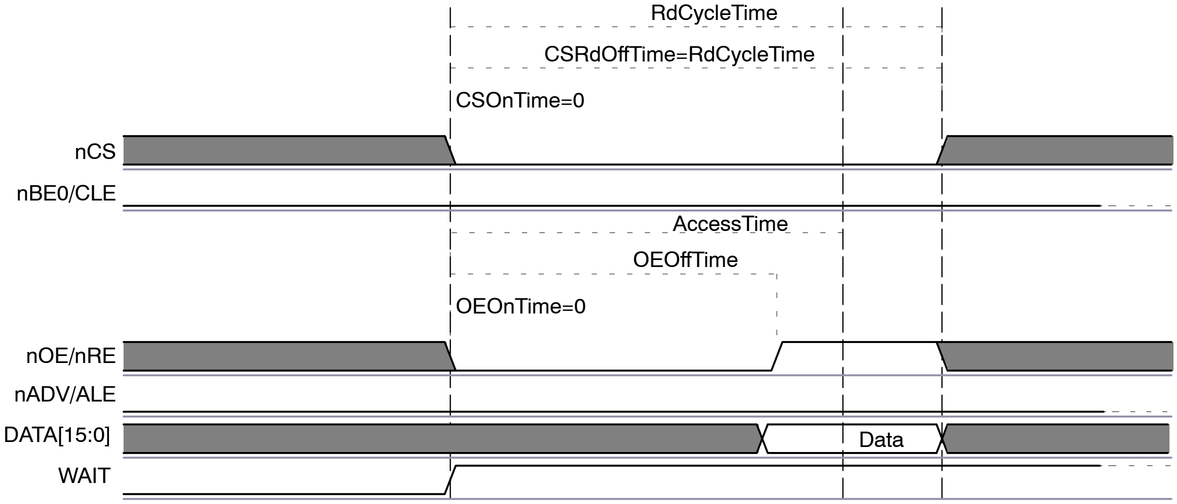
Reading data from the GPMC_NAND_DATA_CS[x] location or from any location in the associated chip-select memory region activates an asynchronous read access.

- ☐ nCS is controlled by CSOnTime and CSRdOffTime timing parameters.
- ☐ nRE is controlled by OEOnTime and OEOffTime timing parameters.
- ☐ To take advantage of nRE high to data invalid minimum timing value, the AccessTime can be set so that data are effectively captured after nRE

deassertion. This allows optimization of NAND read access cycle time completion.

ALE, CLE, and nWE are maintained inactive. Figure 12−25 shows the NAND data read cycle.

Figure 12−25. NAND Data Read Cycle

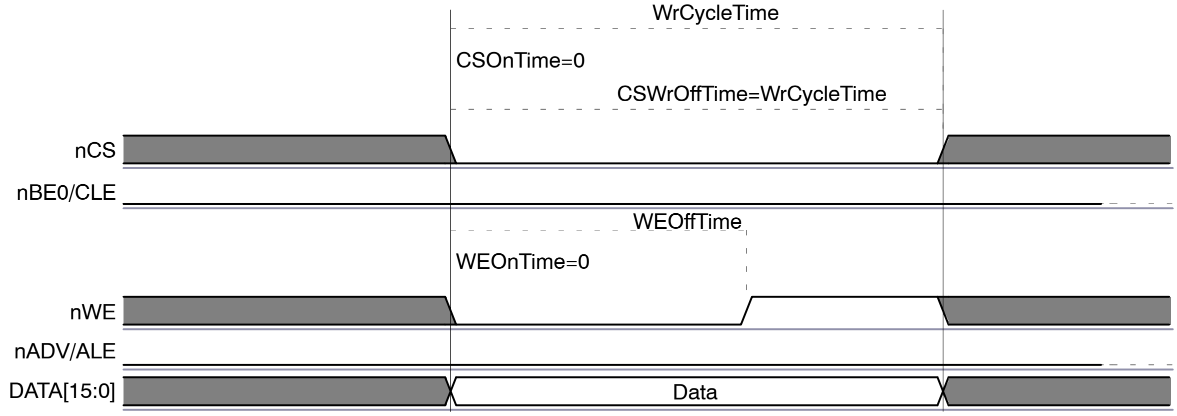


Writing data to the GPMC_NAND_DATA_CS[x] location or to any location in the associated chip-select memory region activates an asynchronous write access.

- ❑ nCE 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 12−26 shows the NAND data write cycle.

Figure 12−26. NAND Data Write Cycle



NAND Device General CS Timing Control Requirements

For most NAND devices, read data access time is constrained by the nCS-to-data-valid timing and not constrained by the nRE-to-data-valid timing. Successive accesses with nCS deassertions between accesses are affected by this timing constraint. Since accesses to a NAND device can be interleaved with other chip-select accesses, it cannot be ensured that nCS stays low between two accesses to the same chip-select. Moreover, an nCS deassertion time between NAND accesses to the same chip-select is likely to be required, as explained next. The nCS deassertion requires that CycleTime and AccessTime be programmed 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 the RdCycleTime, WrCycleTime, AccessTime, CSRdOffTime, CSWrOffTime, ADVRdOffTime, ADVWrOffTime, OEOffTime, and WEOffTime on back-to-back NAND accesses (noninterleaved accesses) and can suppress the minimum nCS high pulse width between accesses. For more information about optimized prefetch engine access, see Section 12.5.10.4, *Prefetch and Write-Posting Engine*.

Some NAND devices require minimum write-to-read idle time, especially for device status read accesses after 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 a timing violation.

NAND devices usually have an important nRE-high-to-data bus in 3-state mode. This requires a bus turnaround setting (BusTurnAround = 1) so that the next access to a different chip-select (interleaved accesses) is delayed until BusTurnAround delay completion. Back-to-back NAND read accesses (non-interleaved read-to-read accesses) are not affected by the programmed bus turnaround delay.

Read and Write Access Size Adaptation

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 D0−D7 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.

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 accesses are 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, since 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 returned 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.

In general, byte access to a 16-bit-wide NAND device should be prohibited, especially when ECC calculation is enabled. 8-bit or 16-bit ECC is corrupted by a byte read to a 16-bit-wide NAND device, because the nonrequested byte is considered invalid on 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.

Since any 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, any incrementing 32-bit burst requests are translated in multiple 32-bit sequential accesses following the 32-bit to 8- or 16-bit device access adaptation.

12.5.10.2 NAND Device Ready Pin Usage

The NAND memory device provides a ready pin to indicate data availability after block/page opening and to indicate data programming completion. The ready pin can be connected to one of the four wait GPMC input pins and data read accesses must not be attempted while the ready pin is not set (device not ready), even though the associated chip-select WAITREADMONITORING bit field is set. Duration of NAND device busy state after block/page opening is so long that accesses performed while the ready pin is not set can stall GPMC access until a system time-out occurs. It is preferred to not let the GPMC access engine monitor the wait pin associated with the NAND device (WaitReadMonitoring = 0 and WaitWriteMonitoring = 0 in GPMC_CONFIG1_CS[x]).

Ready Pin Monitored by Software Polling

The ready signal state can be monitored through the GPMC_STATUS WAIT*N*STATUS bit. The software should 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 invalid window is completed from the last read command written to the NAND device.

Ready Pin Monitored by Hardware Interrupt

Each wait GPMC input pin can generate an interrupt on wait-to-no-wait transition detection. Depending on the programmed wait polarity (WaitxPinPolarity), 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, if WaitxPinPolarity is active low or active high.

The wait transition detector must be cleared before any transition detection by writing 1 to the WAITxEDGEDETECTIONSTATUS bit field of the GPMC_IRQ-

STATUS register according to the wait pin used for the NAND device-ready signal monitoring. The transition detector requires a wait active time detection of a minimum of two GPMC_FCLK cycles to detect a wait-to-no-wait transition. Software must incorporate precautions to clear the wait transition pin detector before completion of the wait (busy) time period.

A wait-to-no-wait transition detection can issue a GPMC interrupt if the WAITxEDGEDETECTIONENABLE bit field in the GPMC_IRQENABLE register is set and if the WAITxEDGEDETECTIONSTATUS bit field in the GPMC_IRQSTATUS register is set.

The WAITMONITORINGTIME field does not affect the wait-to-no-wait transition time detection.

It is also possible to poll the WAITxEDGEDETECTIONSTATUS bit field in the GPMC_IRQSTATUS register according to the wait pin used for the NAND device ready signal monitoring.

12.5.10.3 ECC Use

The ECC engine includes only one accumulation context; therefore, it can be allocated to only one chip-select at a time, and parallel computation on different chip-selects is not possible. Because it is allocated to a particular chip-select, ECC computation is not affected by interleaved GPMC access to other chip-selects and devices. ECC accumulation is processed sequentially in the order of the data read from the memory or written to the memory on the allocated chip-select. The ECC engine does not differentiate between read or write access to the allocated chip-select.

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 this section must be limited to data read or write until the desired number of ECC calculations is complete.

ECC Result Register and ECC Computation Accumulation Size

The GPMC includes up to nine ECC result registers (GPMC_ECCx_RESULT) to store ECC computation results when the desired number of bytes or Word16s have been computed.

The ECC result registers are used sequentially to store one ECC result in one ECC result register, the next ECC result in the next ECC result register on the list, and so on until the last ECC computation. The GPMC_ECCx_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 or written in sequence from or to the NAND device.

The ECCPOINTER bit field in the GPMC_ECC_CONTROL register should be set to the correct value to select which ECC result register should be used first in the list for the incoming ECC computation process. The ECCPOINTER bit can be read to determine which ECC register is used in the next ECC result storage for the ongoing ECC computation. The GPMC_ECCx_RESULT register value is considered valid when the ECCPOINTER bit is equal to x+1. The ECCPOINTER value is frozen at 10 and ECC computing is stopped (ECCENABLE = 0) when the GPMC_ECC9_RESULT register is updated.

The ECC accumulator must be reset before any ECC computation accumulation process. The ECCCLEAR bit field in the GPMC_ECC_CONTROL register should be set to 1 (nonpersistent bit) to clear the accumulator and all ECC result registers.

For each ECC result (each GPMC_ECCx_RESULT), the number of bytes or Word16s used for ECC computing accumulation can be selected between two programmable values.

The ECCxRESULTSIZE bit field selects which programmable size (ECCSIZE0 or ECCSIZE1) value should be used for this ECC result (stored in the GPMC_ECCx_RESULT register).

The ECCSIZE0 and ECCSIZE1 bit fields allow selection of the number of bytes or Word16s to be used for ECC computation accumulation. Any 2N values from 2 to a maximum of 512 are allowed.

The flexibility of the number of ECCs computed and the flexibility of the number of bytes or Word16s used in the successive ECC computations enable 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 ECC on the NAND spare byte.

For example, with a 2K-byte data page 8-bit-wide NAND device, 8 ECCs accumulated on 256 bytes can be computed and added to one extra ECC computed on the 24-spare-byte area where the 8 ECC results used for comparison and correction with the computed data page ECC are stored. The GPMC provides 9 GPMC_ECC_RESULT registers to store the results. In that case, ECCSIZE0 is set to 256, ECCSIZE1 is set to 24, ECC[1:8]RESULTSIZE set to 0, and ECC9RESULTSIZE is set to 1.

ECC Enabling

The ECCCS bit field in the GPMC_ECC_CONTROL register selects the allocated chip-select. The ECCENABLE bit field in the GPMC_ECC_CONTROL register enables ECC computation on the next detected read or write access to the selected chip-select.

The ECCPOINTER, ECCCLEAR, ECCSIZE, ECCxRESULTSIZE, ECC16B, and ESSCS bit fields must not be changed or cleared while an ECC computation process is ongoing.

The ECC accumulator and the ECC result register must not be changed or cleared while an ECC computation process is ongoing.

Table 12−8 shows the ECC enable settings.

Table 12−8. 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
ECCPOINTER	GPMC_ECC_CONTROL	0−7	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 ECC size 1
ECCSIZE0	GPMC_ECC_SIZE_CONFIG	0x00−0xFF	Defines ECC size 0
ECC(x)RESULTSIZE ((x) 0 to 9)	GPMC_ECC_SIZE_CONFIG	0/1	Selects the size of ECCx result register
ECCENABLE	GPMC_ECC_CONFIG	1	Enables the ECC computation

ECC Calculation Based on 8-Bit Byte

8-bit based ECC computation is used for 8-bit-wide NAND device interfacing.

8-bit based ECC computation can be used for 16-bit width NAND device interfacing to get backward compatibility on the error-handling strategy used with 8-bit-wide NAND devices. In this case, 16-bit-wide data read or written to the NAND device is fragmented into two bytes. According to little-endian access, the LSB of the 16-bit width data is ordered first in the byte stream used for 8-bit based ECC computation

ECC Calculation Based on 16-Bit Word

16-bit word-based ECC computation is used for 16-bit width NAND device interfacing.

16-bit word-based computation is not supported when interfacing with an 8-bit-wide NAND device.

12.5.10.4 Prefetch and Write-Posting Engine

NAND device data access cycles are usually slower than the MCU system frequency; such NAND read or write accesses issued by the MCU can affect system performance, especially when a long read or write sequence is required for NAND page loading or programming. To minimize the impact on system performance, the GPMC includes a prefetch and write-posting engine that can be used to read or write to any chip-select location in a background manner.

The prefetch and write-posting engine uses an embedded 64 x Bytes (32 x 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). FIFO draining and filling (read and write) can be done by the MCU through interrupt synchronization (programmable byte threshold) or can be done by the sDMA through DMA request synchronization (programmable request byte size) in either prefetch or posting modes.

The prefetch and write-posting engine is a single-context engine that can be allocated to only one CS at a time for a read prefetch or a write-posting process.

The engine does not support automatic command and address phase programming and is limited to the chip-select memory region read or write accesses. It is also limited to NAND data stream accesses. The engine model use relies on the MCU NAND software driver to control block and page opening with the correct data address pointer initialization before the engine can start to read or write from/to the NAND memory device.

The engine reads/writes data from/to the selected chip-select based on FIFO location/byte availability and until the total programmed number of bytes has been read/written.

Any host concurrent accesses to a different chip-select are correctly interleaved with the ongoing engine accesses. The engine has the lowest priority access so that host accesses to a different chip-select have a minimum time delay.

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 on a different chip-select. If the PFPWENROUNDROBIN bit of the GPMC_PREFETCH_CONFIG1 register is enabled, the arbitration grants the prefetch and write-posting engine a programmable number of requests based on the PFPWWEIGHTEDPRIO bit field of the GPMC_PREFETCH_CONFIG1 register.

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 and write-posting engine (read/write) request is routed to the access engine with the chip-select destination ID. After the necessary 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. The destination chip-select configuration should 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.

General Programming Model

The engine can be configured only if the STARTENGINE bit field of the GPMC_PREFETCH_CONTROL register is set at 0.

The engine must be correctly configured in prefetch or write-posting mode and should be allocated to a NAND chip-select before it is started. The chip-select allocation is set in the ENGINECSSELECTOR bit field of the GPMC_PREFETCH_CONFIG1 register.

In both prefetch and write-posting modes, the engine uses byte or Word16 access requests, respectively, for an 8-bit wide or 16-bit-wide NAND device attached to the allocated chip-select. Accordingly, the FIFOTHRESHOLD and TRANSFERCOUNT bit fields must be expressed in N x Bytes or 2N x Bytes.

The FIFO entry on the l3 interconnect port side is accessible at any location of the associated chip-select memory region when the ENABLEENGINE bit field of the GPMC_PREFETCH_CONFIG1 register is set. When the ENABLEENGINE bit is set, any host accesses to this chip-select are re-routed to the FIFO entry. Accessing the NAND device allocated to this chip-select from the host is still feasible through the GPMC_NAND_COMMAND_CSx, GPMC_NAND_ADDRESS_CSx, and GPMC_NAND_DATA_CSx locations.

Even though the FIFO entry on the L3 interconnect port is 32 bits wide, it can be accessed with Byte, Word16, or Word32 access size, according to little-endian format.

Control of the FIFO is facilitated by the use of interrupts or DMA requests associated with the FIFO threshold. The FIFOPOINTER bit field in the GPMC_PREFETCH_STATUS register monitors the number of available bytes to be read in prefetch mode or the number of free empty byte places to be written in post-write mode. The COUNTVALUE bit field in the GPMC_PREFETCH_STATUS register monitors the number of remaining bytes to be read or to be written by the engine according to the TRANSFERCOUNT value. The FIFOPOINTER and COUNTVALUE bit fields are always expressed in bytes, even though a 16-bit-wide NAND device is attached to the allocated chip-select.

In prefetch mode, when the FIFO pointer is 0, meaning that the FIFO is empty, any host read access gets the last available byte. In case of Word32 or Word16 read access size, the last available byte is duplicated to fit the requested word size. In post-write mode, when the FIFO pointer is 0, meaning that the FIFO is full; any host write overwrites the last byte FIFO location. There is no underflow/overflow error reporting in the GPMC.

Prefetch Mode

The prefetch mode is selected when the ACCESSMODE bit in the GPMC_PREFETCH_CONFIG1 register 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 the NAND data array. The engine is started by setting the STARTENGINE bit field to 1 in the GPMC_PREFETCH_CONTROL register. The STARTENGINE bit field is automatically cleared on prefetch process completion.

If required, the ECC calculator engine must be started (configured, context reset, and enabled) before the prefetch engine is started so that the ECC is correctly computed on all data read by the prefetch engine on this allocated chip-select.

When the SYNCHROMODE bit field in the GPMC_PREFETCH_CONFIG1 register is cleared, the prefetch engine starts requesting data when the STARTENGINE bit field is set. In that configuration, the host must monitor its own NAND device ready pin so that it sets the STARTENGINE bit only when the NAND device is in ready state, meaning that data are valid for prefetching.

When the SYNCHROMODE bit field in the GPMC_PREFETCH_CONFIG1 register is set, the prefetch engine starts requesting data when a wait-to-no-wait transition event is detected. The transition detector must be cleared before any transition detection. The WAITPINSELECTOR bit field in the GPMC_PREFETCH_CONFIG1 register selects which wait-pin edge detector triggers the prefetch engine in this synchronized mode.

If the STARTENGINE bit field is set after the NAND Cmd/Add phase (page opening command), the start engine is effective only after read NAND Cmd/Add phase completion. To prevent GPMC stall during this NAND Cmd/Add phase, set the start engine before NAND Cmd/Add phase completion in synchro mode. Prefetch starts when a wait-to-no-wait transition is detected.

The prefetch engine issues a read request to ensure that the FIFO is always filled with the maximum amount of data until the programmed TRANSFERCOUNT value in the GPMC_PREFETCH_CONFIG2 register is completed.

Table 12−9.Prefetch Mode Configuration

Bit Field	Register	Value	Comments
STARTENGINE	GPMC_PREFETCH_CONTROL	0	Prefetch engine can be configured only if STARTENGINE is set to 0.
ENGINECSSELECTOR	GPMC_PREFETCH_CONFIG1	0 to 7	Selects the chip-select associated with a NAND device where the prefetch engine is active
ACCESS MODE	GPMC_PREFETCH_CONFIG1	0	Selects prefetch mode
FIFOTHRESHOLD	GPMC_PREFETCH_CONFIG1		Selects the maximum number of bytes read or written by the host on DMA or interrupt request
TRANSFERCOUNT	GPMC_PREFETCH_CONFIG2		Selects the number of bytes to be read or written by the engine to the selected chip-select
SYNCHROMODE	GPMC_PREFETCH_CONFIG1	0/1	Selects when the engine starts access to the chip-select
WAITPINSELECTOR	GPMC_PREFETCH_CONFIG1	0 to 3	(If SynchroMode = 1) Selects wait-pin edge detector
ENABLEOPTIMIZEDACCESS	GPMC_PREFETCH_CONFIG1	0/1	See *Optimizing NAND Access with Prefetch and Write-Posting Engine* in Section 12.5.10.4.
CYCLEOPTIMIZATION	GPMC_PREFETCH_CONFIG1		
ENABLEENGINE	GPMC_PREFETCH_CONFIG1	1	Engine enabled
STARTENGINE	GPMC_PREFETCH_CONTROL	1	

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 state can be monitored through the FIFO pointer or with the THRESHOLDSTATUS bit in the GPMC_PREFETCH_STATUS register. The FIFO pointer indicates the current number of available data to be read and the FIFOTHRESHOLDSTATUS bit set to 1 indicates that at least FIFO threshold bytes are available and can be read from the FIFO.

An interrupt can be issued by the GPMC if the FIFOEVENTENABLE bit field is set in the GPMC_IRQENABLE register. The FIFO interrupt event is logged in the FIFOEVENTSTATUS bit field of the GPMC_IRQSTATUS register. To resume the interrupt, the MPU must read all the available bytes (or enough bytes to get below the FIFO threshold), and the FIFOEVENTSTATUS bit field should be cleared to detect further interrupt events. The FIFOEVENTSTATUS bit field must always be cleared before the FIFOEVENTENABLE bit field is set, to resume the out-of-date logged interrupt event.

Prefetch completion can be monitored through the COUNTVALUE bit field in the GPMC_PREFETCH_STATUS register. The COUNTVALUE bit field indicates the number of remaining data to be requested, according to the TRANSFERCOUNT value. An interrupt can be issued by the GPMC on prefetch process completion (COUNTVALUE = 0) if the TERMINALCOUNTEVENTENABLE bit field is set in the GPMC_IRQENABLE register. The TERMINALCOUNT interrupt event is logged and the TERMINALCOUNTEVENTSTATUS bit field is set in the GPMC_IRQSTATUS register. To resume the interrupt, the MPU must clear the TERMINALCOUNTEVENTSTATUS bit field. The TERMINALCOUNTEVENTSTATUS bit field must always be cleared before TERMINALCOUNTEVENTENABLE bit field is set, to resume the out-of-date logged interrupt event.

The COUNTVALUE value is valid only when the prefetch engine is active (started) and an interrupt is issued only when the COUNTVALUE = 0, causing the prefetch engine to go from active to stopped automatically.

The number of bytes to be prefetched, which are programmed in the TRANSFERCOUNT bit field, should be a multiple of the FIFOTHRESHOLD value to get the correct number of interrupts to allow a deterministic and transparent FIFO control (always the same number of bytes read from the ISR and the FIFO empty after the last interrupt). If this is not the case, 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 DMAMODE bit field in the GPMC_PREFETCH_CONFIG1 register 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 read from the FIFO during the DMA request acknowledge access equals the number of bytes programmed in the FIFOTHRESHOLD bit field. The DMA request remains active until the respective number of bytes has 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 request, so that when the number of bytes to be prefetched is not a multiple of FIFO-THRESHOLD, the remaining bytes (below the threshold) in the FIFO can be read by the DMA channel on the last DMA request. This assumes that the number of remaining bytes to be read is known and controlled by the DMA channel programming model.

Any active DMA request is resumed 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 an out-of-date active DMA request does not trigger spurious DMA transfers.

Write-Posting Mode

The posting mode is selected when the ACCESSMODE bit in the GPMC_ PREFETCH_CONFIG1 register is set.

The MCU NAND SW driver must issue the correct data address pointer initialization (PAGE PROGRAM) command before the engine can start to write data to a NAND memory device. The engine is started by setting the STARTENGINE bit field to 1 in the GPMC_PREFETCH_CONTROL register. The STARTENGINE bit field is automatically cleared on posting process completion. When all data are 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 required, the ECC calculator engine must be started (configured, context reset, and enabled) before the posting engine is started, so that the ECC is correctly calculated on all data written by the prefetch engine on the allocated chip-select.

In posting mode, the SYNCHROMODE bit field in the GPMC_PREFETCH_ CONFIG1 register must be cleared so that posting starts as soon as the STARTENGINE bit field is set and the FIFO is not empty.

If the STARTENGINE bit field is set after the NAND Cmd/Add phase (page opening command), the start engine is effective only after read NAND Cmd/ Add phase completion. To prevent GPMC stall during this NAND Cmd/Add phase, set the start engine before NAND Cmd/Add phase completion and ensure that the associated DMA channel is enabled after the NAND Cmd/Add phase.

The posting engine issues a write request as soon as valid data are available in the FIFO and until the programmed TRANSFERCOUNT value in the GPMC_PREFETCH_CONFIG2 register is completed.

The StartEngine bit field is automatically cleared on posting process completion. When all data are 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).

Table 12−10. Write Posting Mode Configuration

Bit Field	Register	Value	Comments
STARTENGINE	GPMC_PREFETCH_CONTROL	0	Write posting engine can be configured only if STARTENGINE is set to 0.
ENGINECSSELECTOR	GPMC_PREFETCH_CONFIG1	0 to 7	Selects the chip-select associated with a NAND device where the prefetch engine is active
ACCESSMODE	GPMC_PREFETCH_CONFIG1	1	Selects write-posting mode
FIFOTHRESHOLD	GPMC_PREFETCH_CONFIG1		Selects the maximum number of bytes read or written by the host on DMA or interrupt request
TRANSFERCOUNT	GPMC_PREFETCH_CONFIG2		Selects the number of bytes to be read or written by the engine from/to the selected chip-select
SYNCHROMODE	GPMC_PREFETCH_CONFIG1	0	Engine starts the access to chip-select as soon as STARTENGINE is set.
ENABLEOPTIMIZEDACCESS	GPMC_PREFETCH_CONFIG1	0/1	See *Optimizing NAND Access with Prefetch and Write-Posting Engine* in Section 12.5.10.4.
CYCLE OPTIMIZATION	GPMC_PREFETCH_CONFIG1		
ENABLEENGINE	GPMC_PREFETCH_CONFIG1	1	Engine enabled
STARTENGINE	GPMC_PREFETCH_CONTROL	1	

FIFO Control in Posting Mode

The FIFO can be filled directly by the MPU or by an sDMA channel.

In MPU filling mode, the FIFO state can be monitored through the FIFO pointer or with the FIFOTHRESHOLDSTATUS bit in the GPMC_PREFETCH_STATUS register. The FIFO pointer indicates the current number of available free byte places, 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 FIFOEVENTENABLE bit field is set in the GPMC_IRQENABLE register. The FIFO interrupt event is logged and the FIFOEVENTSTATUS bit field is set in the GPMC_IRQSTATUS register. To resume the interrupt, the MPU must write enough bytes to fill the FIFO (or enough bytes to get below the threshold) and the FIFOEVENTSTATUS bit field must be cleared to get further interrupt events. The FIFOEVENTSTATUS bit field should always be cleared before the FIFOEVENTENABLE bit field is set, to resume the out-of-date logged interrupt event.

Posting completion can be monitored through the COUNTVALUE bit field in the GPMC_PREFETCH_STATUS register. COUNTVALUE indicates the number of remaining data to be written according to the TRANSFERCOUNT value. An interrupt can be issued by the GPMC on posting process completion (COUNTVALUE = 0) if the TERMINALCOUNTEVENTENABLE bit field is set in the GPMC_IRQENABLE register. The TERMINALCOUNT interrupt event is logged and the TERMINALCOUNTEVENTSTATUS bit field is set in the the GPMC_IRQSTATUS register. To resume the interrupt, the MPU must clear the TERMINALCOUNTEVENTSTATUS bit field. The TERMINALCOUNTEVENTSTATUS bit field must always be cleared before the TERMINALCOUNTEVENTENABLE bit field is set, to resume the out-of-date logged interrupt event.

Note:

The COUNTVALUE value is valid only when the posting engine is active (started) and an interrupt is issued only when COUNTVALUE = 0, that is, when the posting engine automatically goes from active to stopped.

In DMA filling mode, the DMAMODE bit field in the GPMC_PREFETCH_CONFIG1 register 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 an equal number of bytes programmed in FIFOTHRESHOLD is written to the FIFO during the DMA request acknowledge access. The DMA request is still active until the associated number of bytes has been effectively written to the FIFO and no other DMA request can be issued until the ongoing active request is complete.

Any potentially active DMA request is resumed when the write-posting engine goes from inactive to active prefetch (STARTENGINE = 1). The associated DMA channel must always be enabled by the MPU after setting the STARTENGINE bit.

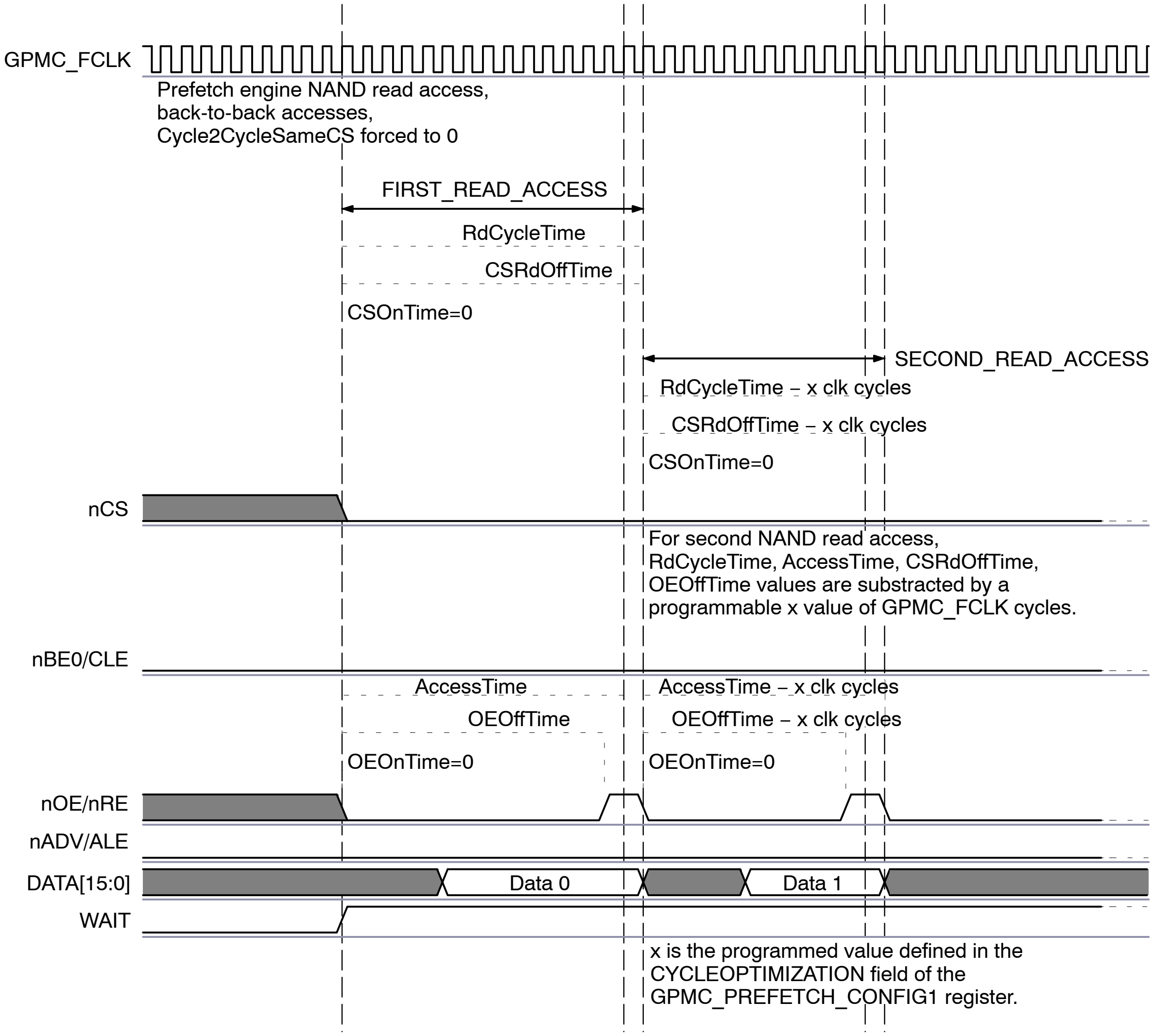
Optimizing NAND Access with the Prefetch and Write-Posting Engine

Access time to a NAND memory device can be optimized on back-to-back access if the associated nCS 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 (interleaved access) have occurred. Also, it eliminates the Cycle2CycleDelay even if Cycle2CycleSameCSEn is set. This capability is limited to the prefetch and write-posting engine accesses only; MPU access to a NAND memory device (through chip-select memory region or through GPMC_NAND_DATA_CS[x] location) is not optimized.

The ENABLEOPTIMIZEDACCESS bit field in the GPMC_PREFETCH_CONFIG1 register must be set to enable optimized access. The CYCLEOPTIMIZATION bit field in the GPMC_PREFETCH_CONFIG1 register defines the number of GPMC_FCLK cycles to be suppressed from the RdCycleTime, WrCycleTime, AccessTime, CSOffTime, ADVOffTime, OEOffTime, and WEOffTime timing parameters to optimize access time.

Figure 12−27 highlights the difference between nonoptimized and optimized accesses.

Figure 12−27. NAND Read Cycle Optimization Timing Description


GPMC_FCLK
Prefetch engine NAND read access,
back-to-back accesses,
Cycle2CycleSameCS forced to 0
FIRST_READ_ACCESS
RdCycleTime
CSRdOffTime
CSOnTime=0
SECOND_READ_ACCESS
RdCycleTime − x clk cycles
CSRdOffTime − x clk cycles
CSOnTime=0
nCS
For second NAND read access,
RdCycleTime, AccessTime, CSRdOffTime,
OEOffTime values are substracted by a
programmable x value of GPMC_FCLK cycles.
nBE0/CLE
AccessTime
OEOffTime
AccessTime − x clk cycles
OEOffTime − x clk cycles
OEOnTime=0
OEOnTime=0
nOE/nRE
nADV/ALE
DATA[15:0]
Data 0
Data 1
WAIT
x is the programmed value defined in the
CYCLEOPTIMIZATION field of the
GPMC_PREFETCH_CONFIG1 register.

12.6 GPMC Registers

This section provides information about the GPMC1 module instance in this product. Table 12−11 summarizes the GPMC1 registers. Each register in the module instance is described separately in the remaining parts of this section.

12.6.1 GPMC Register Summary

Module Name	Base Address	Size
GPMC1	0x6800 A000	4K bytes

Table 12−11. GPMC1 Register Summary

Register Name	Type	Register Width (Bits)	Offset
GPMC_REVISION	R	32	0x000
GPMC_SYSCONFIG	RW	32	0x010
GPMC_SYSSTATUS	R	32	0x014
GPMC_IRQSTATUS	RW	32	0x018
GPMC_IRQENABLE	RW	32	0x01C
GPMC_TIMEOUT_CONTROL	RW	32	0x040
GPMC_ERR_0DDRESS	RW	32	0x044
GPMC_ERR_TYPE	RW	32	0x048
GPMC_CONFIG	RW	32	0x050
GPMC_STATUS	RW	32	0x054
GPMC_CONFIG1_0	RW	32	0x060
GPMC_CONFIG2_0	RW	32	0x064
GPMC_CONFIG3_0	RW	32	0x068
GPMC_CONFIG4_0	RW	32	0x06C
GPMC_CONFIG5_0	RW	32	0x070
GPMC_CONFIG6_0	RW	32	0x074
GPMC_CONFIG7_0	RW	32	0x078
GPMC_NAND_COMMAND_0	W	32	0x07C
GPMC_NAND_ADDRESS_0	W	32	0x080
GPMC_NAND_DATA_0	RW	32	0x084
GPMC_CONFIG1_1	RW	32	0x090
GPMC_CONFIG2_1	RW	32	0x094
GPMC_CONFIG3_1	RW	32	0x098
GPMC_CONFIG4_1	RW	32	0x09C
GPMC_CONFIG5_1	RW	32	0x0A0
GPMC_CONFIG6_1	RW	32	0x0A4
GPMC_CONFIG7_1	RW	32	0x0A8
GPMC_NAND_COMMAND_1	W	32	0x0AC

Table 12−11. GPMC1 Register Summary (Continued)

Register Name	Type	Register Width (Bits)	Offset
GPMC_NAND_ADDRESS_1	W	32	0x0B0
GPMC_NAND_DATA_1	RW	32	0x0B4
GPMC_CONFIG1_2	RW	32	0x0C0
GPMC_CONFIG2_2	RW	32	0x0C4
GPMC_CONFIG3_2	RW	32	0x0C8
GPMC_CONFIG4_2	RW	32	0x0CC
GPMC_CONFIG5_2	RW	32	0x0D0
GPMC_CONFIG6_2	RW	32	0x0D4
GPMC_CONFIG7_2	RW	32	0x0D8
GPMC_NAND_COMMAND_2	W	32	0x0DC
GPMC_NAND_ADDRESS_2	W	32	0x0E0
GPMC_NAND_DATA_2	RW	32	0x0E4
GPMC_CONFIG1_3	RW	32	0x0F0
GPMC_CONFIG2_3	RW	32	0x0F4
GPMC_CONFIG3_3	RW	32	0x0F8
GPMC_CONFIG4_3	RW	32	0x0FC
GPMC_CONFIG5_3	RW	32	0x100
GPMC_CONFIG6_3	RW	32	0x104
GPMC_CONFIG7_3	RW	32	0x108
GPMC_NAND_COMMAND_3	W	32	0x10C
GPMC_NAND_ADDRESS_3	W	32	0x110
GPMC_NAND_DATA_3	RW	32	0x114
GPMC_CONFIG1_4	RW	32	0x120
GPMC_CONFIG2_4	RW	32	0x124
GPMC_CONFIG3_4	RW	32	0x128
GPMC_CONFIG4_4	RW	32	0x12C
GPMC_CONFIG5_4	RW	32	0x130
GPMC_CONFIG6_4	RW	32	0x134
GPMC_CONFIG7_4	RW	32	0x138
GPMC_NAND_COMMAND_4	W	32	0x13C
GPMC_NAND_ADDRESS_4	W	32	0x140
GPMC_NAND_DATA_4	RW	32	0x144
GPMC_CONFIG1_5	RW	32	0x150
GPMC_CONFIG2_5	RW	32	0x154
GPMC_CONFIG3_5	RW	32	0x158
GPMC_CONFIG4_5	RW	32	0x15C
GPMC_CONFIG5_5	RW	32	0x160

Table 12−11. GPMC1 Register Summary (Continued)

Register Name	Type	Register Width (Bits)	Offset
GPMC_CONFIG6_5	RW	32	0x164
GPMC_CONFIG7_5	RW	32	0x168
GPMC_NAND_COMMAND_5	W	32	0x16C
GPMC_NAND_ADDRESS_5	W	32	0x170
GPMC_NAND_DATA_5	RW	32	0x174
GPMC_CONFIG1_6	RW	32	0x180
GPMC_CONFIG2_6	RW	32	0x184
GPMC_CONFIG3_6	RW	32	0x188
GPMC_CONFIG4_6	RW	32	0x18C
GPMC_CONFIG5_6	RW	32	0x190
GPMC_CONFIG6_6	RW	32	0x194
GPMC_CONFIG7_6	RW	32	0x198
GPMC_NAND_COMMAND_6	W	32	0x19C
GPMC_NAND_ADDRESS_6	W	32	0x1A0
GPMC_NAND_DATA_6	RW	32	0x1A4
GPMC_CONFIG1_7	RW	32	0x1B0
GPMC_CONFIG2_7	RW	32	0x1B4
GPMC_CONFIG3_7	RW	32	0x1B8
GPMC_CONFIG4_7	RW	32	0x1BC
GPMC_CONFIG5_7	RW	32	0x1C0
GPMC_CONFIG6_7	RW	32	0x1C4
GPMC_CONFIG7_7	RW	32	0x1C8
GPMC_NAND_COMMAND_7	W	32	0x1CC
GPMC_NAND_ADDRESS_7	W	32	0x1D0
GPMC_NAND_DATA_7	RW	32	0x1D4
GPMC_PREFETCH_CONFIG1	RW	32	0x1E0
GPMC_PREFETCH_CONFIG2	RW	32	0x1E4
GPMC_PREFETCH_CONTROL	RW	32	0x1EC
GPMC_PREFETCH_STATUS	RW	32	0x1F0
GPMC_ECC_CONFIG	RW	32	0x1F4
GPMC_ECC_CONTROL	RW	32	0x1F8
GPMC_ECC_SIZE_CONFIG	RW	32	0x1FC
GPMC_ECC1_RESULT	RW	32	0x200
GPMC_ECC2_RESULT	RW	32	0x204
GPMC_ECC3_RESULT	RW	32	0x208
GPMC_ECC4_RESULT	RW	32	0x20C
GPMC_ECC5_RESULT	RW	32	0x210

Table 12−11. GPMC1 Register Summary (Continued)

Register Name	Type	Register Width (Bits)	Offset
GPMC_ECC6_RESULT	RW	32	0x214
GPMC_ECC7_RESULT	RW	32	0x218
GPMC_ECC8_RESULT	RW	32	0x21C
GPMC_ECC9_RESULT	RW	32	0x220
GPMC_TESTMODE_CTRL	RW	32	0x230
GPMC_PSA_LSB	R	32	0x234
GPMC_PSA_MSB	R	32	0x238

12.6.2 GPMC Register Descriptions

All GPMC registers are aligned on 32-bit address boundaries and can be accessed by single byte, Word16, Word32, and x2, x4, and x8 bursts of Word32 requests.

All register file accesses are in little-endian format.

Table 12−12. GPMC_REVISION

Address Offset	0x000		
Physical Address	0x6800 A000	**Instance**	GPMC1
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	
7:0	REV	IP revision [7:4] Major revision [3:0] Minor revision Examples: 0x10 for 1.0, 0x21 for 2.1	R	

Table 12−13. GPMC_SYSCONFIG

Address Offset	0x010		
Physical Address	0x6800 A010	**Instance**	GPMC1
Description	This register controls the the L3 interconnect interface 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																											IDLEMODE		Reserved	SOFTRESET	AUTOIDLE

Bits	Field Name	Description		Type	Reset
31:5	Reserved	Write 0s for future compatibility Read returns 0.		RW	0x0000000
4:3	IDLEMODE			RW	0x0
		0x0:	Force idle. An idle request is acknowledged unconditionally		
		0x1:	No idle. An idle request is never acknowledged		
		0x2:	Smart idle. An idle request is acknowledged based on internal module activity.		
		0x3:	Reserved. Do not use .		
2	Reserved	Write 0 for future compatibility. Read returns 0.		RW	0
1	SOFTRESET	Software reset (setting this bit to 1 triggers a module reset). This bit is automatically reset by hardware. Read returns 0.		RW	0
		0x0:	Normal mode		
		0x1:	The module is reset.		
0	AUTOIDLE	Internal L3 interconnect clock-gating strategy		RW	0
		0x0:	L3 interconnect clock is free-running.		
		0x1:	Automatic L3 interconnect clock-gating strategy is applied, based on L3 interconnect interface activity.		

Table 12−14. GPMC_SYSSTATUS

Address Offset	0x014		
Physical Address	0x6800 A014	**Instance**	GPMC1
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	Read returns 0.	R	0x000000
7:1	Reserved	Read returns 0 (reserved for L3 interconnect-socket status information).	R	0x00
0	RESETDONE	Internal reset monitoring 0x0: Internal module reset is ongoing. 0x1: Reset complete[1]	R	–

---

1) During reset, the value is 0, but the value read just after the reset is 1, because ICLK/FCLK is already operating.

Table 12−15. GPMC_IRQSTATUS

Address Offset	0x018		
Physical address	0x6800 A018	**Instance**	GPMC1
Description	This interrupt status register regroups the statuses of module internal events that can generate 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																				WAIT3EDGEDETECTIONSTATUS	WAIT2EDGEDETECTIONSTATUS	WAIT1EDGEDETECTIONSTATUS	WAIT0EDGEDETECTIONSTATUS	Reserved						TERMINALCOUNTSTATUS	FIFOEVENTSTATUS

Table 12−15. GPMC_IRQSTATUS (Continued)

Bits	Field Name	Description		Type	Reset
31:12	Reserved	Write 0s for future compatibility. Read returns 0.		RW	0x00000
11	WAIT3EDGE-DETECTION STATUS	Status of the Wait3 edge detection interrupt		RW	0
		Read 0x0:	Read 0: A transition on WAIT3 input pin was not detected.		
		Write 0x0:	Write 0: WAIT3EDGEDETECTION-STATUS bit unchanged		
		Read 0x1:	Read 1: A transition on WAIT3 input pin was detected.		
		Write 0x1:	Write 1: WAIT3EDGEDETECTION-STATUS bit is reset.		
10	WAIT2EDGE-DETECTION STATUS	Status of the Wait2 edge detection interrupt		RW	0
		Read 0x0:	Read 0: A transition on WAIT2 input pin was not detected.		
		Write 0x0:	Write 0: WAIT2EDGEDETECTION-STATUS bit unchanged		
		Read 0x1:	Read 1: A transition on WAIT2 input pin was detected.		
		Write 0x1:	Write 1: WAIT2EDGEDETECTION-STATUS bit is reset.		
9	WAIT1EDGE-DETECTION STATUS	Status of the Wait1 edge detection interrupt		RW	0
		Read 0x0:	Read 0: A transition on WAIT1 input pin was not detected.		
		Write 0x0:	Write 0: WAIT1EDGEDETECTION-STATUS bit unchanged		
		Read 0x1:	Read 1: A transition on WAIT1 input pin was detected.		
		Write 0x1:	Write 1: WAIT1EDGEDETECTION-STATUS bit is reset.		
8	WAIT0EDGE-DETECTION STATUS	Status of the Wait0 edge detection interrupt		RW	0
		Read 0x0:	Read 0: A transition on WAIT0 input pin was not detected.		
		Write 0x0:	Write 0: WAIT0EDGEDETECTION-STATUS bit unchanged		
		Read 0x1:	Read 1: A transition on WAIT0 input pin was detected.		
		Write 0x1:	Write 1: WAIT0EDGEDETECTION-STATUS bit is reset.		
7:2	Reserved	Write 0s for future compatibility. Read returns 0.		RW	0x00

Table 12−15. GPMC_IRQSTATUS (Continued)

Bits	Field Name	Description		Type	Reset
1	TERMINAL-COUNTSTATUS	Status of the TerminalCountEvent interrupt		RW	0
		Read 0x0:	Read 0: CountValue > 0.		

Bits	Field Name	Description		Type	Reset
		Write 0x0:	Write 0: TerminalCountEvent status bit unchanged		
		Read 0x1:	Read 1: CountValue = 0		
		Write 0x1:	Write 1: TerminalCountEvent status bit is reset.		
0	FIFOEVENT STATUS	Status of the FIFOEvent interrupt		RW	0
		Read 0x0:	Read 0: Less than FIFOThreshold bytes are available in prefetch mode and less than FIFOThreshold bytes free places are available in write-posting mode.		
		Write 0x0:	Write 0: FIFOEvent status bit unchanged		
		Read 0x1:	Read 1: At least FIFOThreshold bytes are available in prefetch mode and at least FIFOThreshold bytes free places are available in write posting mode.		
		Write 0x1:	Write 1: FIFOEvent status bit is reset.		

Table 12–16. GPMC_IRQENABLE

Address Offset	0x01C		
Physical Address	0x6800 A01C	**Instance**	GPMC1
Description	The interrupt enable register allows masking/unmasking of 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

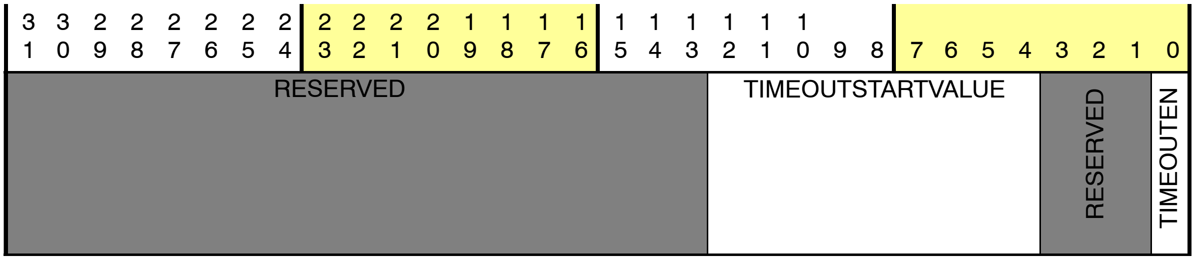
Bits	Field Name	Description	Type	Reset
31:12	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000
11	WAIT3EDGE DETECTION ENABLE	Enables the Wait3 edge detection interrupt 0x0: Wait3EdgeDetection interrupt is masked. 0x1: Wait3EdgeDetection event generates an interrupt.	RW	0
10	WAIT2EDGE DETECTION ENABLE	Enables the Wait2 edge detection interrupt 0x0: Wait2EdgeDetection interrupt is masked. 0x1: A Wait2EdgeDetection event generates an interrupt.	RW	0
9	WAIT1EDG EDETECTION ENABLE	Enables the Wait1 edge detection interrupt 0x0: Wait1EdgeDetection interrupt is masked. 0x1: A Wait1EdgeDetection event generates an interrupt.	RW	0
8	WAIT0EDG EDETECTION ENABLE	Enables the Wait0 edge detection interrupt 0x0: Wait0EdgeDetection interrupt is masked. 0x1: A Wait0EdgeDetection event generates an interrupt.	RW	0
7:2	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00

Table 12−16. GPMC_IRQENABLE (Continued)

Bits	Field Name	Description	Type	Reset
1	TERMINAL-COUNT EVENTENABLE	Enables TerminalCountEvent interrupt issuing in pre-fetch or write-posting mode 0x0: TerminalCountEvent interrupt is masked. 0x1: TerminalCountEvent interrupt is not masked.	RW	0
0	FIFOEVENT ENABLE	Enables the FIFOEvent interrupt 0x0: FIFOEvent interrupt is masked. 0x1: FIFOEvent interrupt is not masked.	RW	0

Table 12−17. GPMC_TIMEOUT_CONTROL

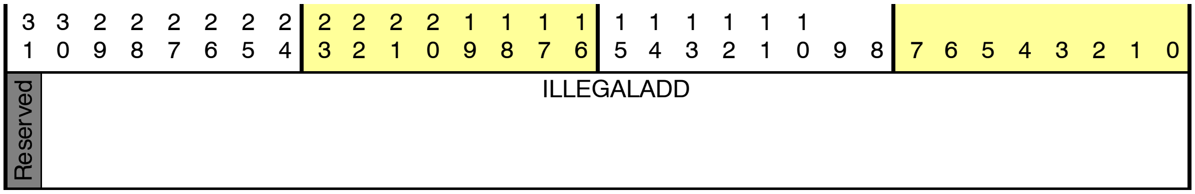
Address Offset	0x00000040		
Physical Address	0x6800 A040	**Instance**	GPMC1
Description	The GPMC_TIMEOUT_CONTROL register allows the user to set the start value of the time-out counter.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:13	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000
12:4	TIMEOUTSTART-VALUE	Start value of the time-out counter (0x000 corresponds to 0 GpmcClk cycle, 0x001 corresponds to 1 GmpcClk cycle, and 0x1FF corresponds to 511 GpmcClk cyles)	RW	0x1FF
3:1	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
0	TIMEOUTEN	Enables the time-out feature 0x0: Time-out is disabled. 0x1: Time-out is enabled.	RW	0

Table 12−18. GPMC_ERR_ADDRESS

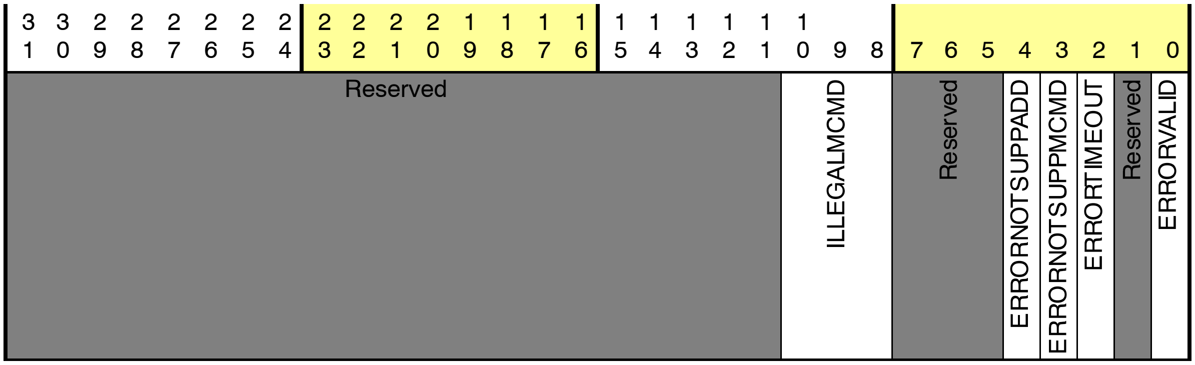
Address Offset	0x044		
Physical Address	0x6800 A044	**Instance**	GPMC1
Description	The GPMC_ERR_ADDRESS register stores the address of the illegal access when an error occurs.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
30:0	ILLEGALADD	Address of illegal access: A30 (0 for memory region, 1 for GPMC register region), and A29−A0 (1G byte maximum)	R	0x00000000

Table 12−19. GPMC_ERR_TYPE

Address Offset	0x048		
Physical Address	0x6800 A048	**Instance**	GPMC1
Description	The GPMC_ERR_TYPE register stores the type of error when an error occurs.		
Type	RW		



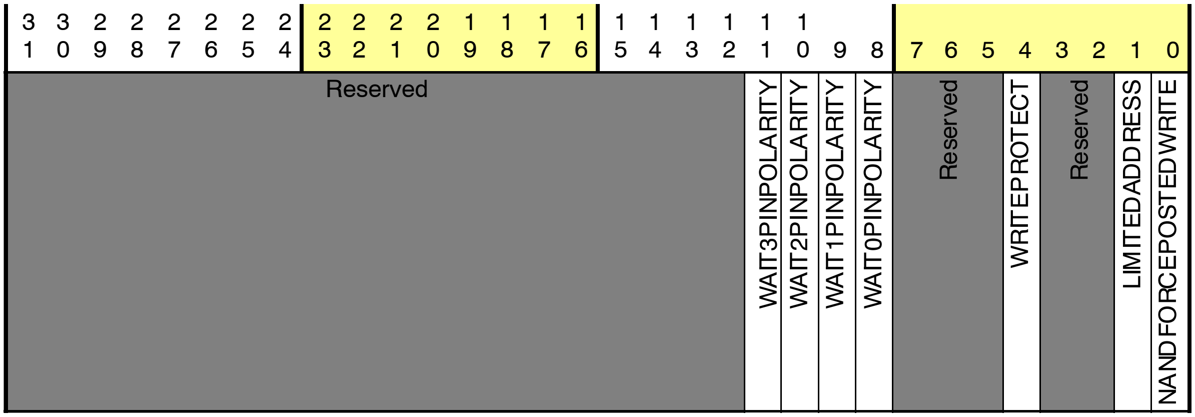
Bits	Field Name	Description	Type	Reset
31:11	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x000000
10:8	ILLEGALMCMD	System command of the transaction that caused the error	R	0x0
7:5	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
4	ERRORNOT SUPPADD	Unsupported address error 0x0: No error occurs. 0x1: The error is caused by an unsupported address.	R	0

Table 12−19. GPMC_ERR_TYPE (Continued)

Bits	Field Name	Description	Type	Reset
3	ERRORNOT-SUPPMCMD	Nonsupported command error 0x0: No error occurs. 0x1: The error is caused by an unsupported command.	R	0
2	ERRORTIMEOUT	Time-out error 0x0: No error occurs. 0x1: The error is caused by a time-out.	R	0
1	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
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	0

Table 12−20. GPMC_CONFIG

Address Offset	0x050		
Physical Address	0x6800 A050	**Instance**	GPMC1
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

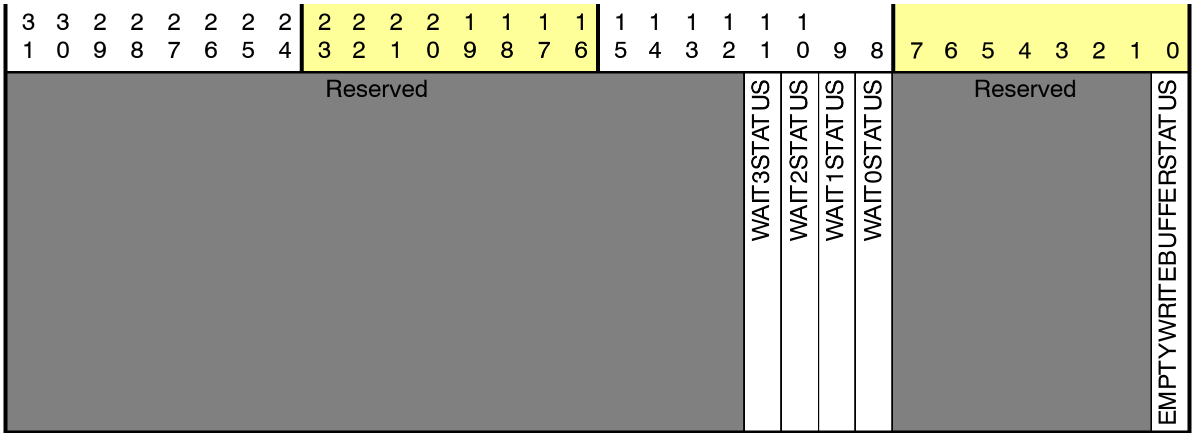
Bits	Field Name	Description	Type	Reset
31:12	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000
11	WAIT3PIN POLARITY	Selects the polarity of input pin WAIT3 0x0: WAIT3 active low 0x1: WAIT3 active high	RW	1
10	WAIT2PIN POLARITY	Selects the polarity of input pin WAIT2 0x0: WAIT2 active low 0x1: WAIT2 active high	RW	0

Table 12−20. GPMC_CONFIG (Continued)

Bits	Field Name	Description	Type	Reset
9	WAIT1PIN POLARITY	Selects the polarity of input pin WAIT1 0x0: WAIT1 active low 0x1: WAIT1 active high	RW	1
8	WAIT0PIN POLARITY	Selects the polarity of input pin WAIT0 0x0: WAIT0 active low 0x1: WAIT0 active high	RW	0
7:5	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
4	WRITEPROTECT	Controls the WP output pin level 0x0: WP output pin is low. 0x1: WP output pin is high.	RW	0
3:2	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
1	LIMITED ADDRESS	Limited address device support 0x0: No effect 0x1: A26−A11 are not modified during an external memory access.	RW	0
0	NANDFORCE-POSTEDWRITE	Enables the force posted-write feature to NAND Cmd/ Address/data location 0x0: Disables force posted write 0x1: Enables force posted write	RW	0

Table 12−21. GPMC_STATUS

Address Offset	0x054		
Physical address	0x6800 A054	**Instance**	GPMC1
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

Table 12−21. GPMC_STATUS (Continued)

Bits	Field Name	Description	Type	Reset
31:12	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000
11	WAIT3STATUS	Copy of input pin WAIT3 (reset value is WAIT3 input pin sampled at IC reset) 0x0: WAIT3 deasserted 0x1: WAIT3 asserted	R	–
10	WAIT2STATUS	Copy of input pin WAIT2 (reset value is WAIT2 input pin sampled at IC reset) 0x0: WAIT2 deasserted 0x1: WAIT2 asserted	R	–
9	WAIT1STATUS	Copy of input pin WAIT1 (reset value is WAIT1 input pin sampled at IC reset) 0x0: WAIT1 deasserted 0x1: WAIT1 asserted	R	–
8	WAIT0STATUS	Copy of input pin WAIT0 (reset value is WAIT0 input pin sampled at IC reset) 0x0: WAIT0 deasserted 0x1: WAIT0 asserted	R	–
7:1	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
0	EMPTYWRITE-BUFFERSTATUS	Stores the empty status of the write buffer 0x0: Write buffer is not empty. 0x1: Write buffer is empty.	R	1

Table 12−22. GPMC_CONFIG1_0

Address Offset	0x060		
Physical Address	0x6800 A060	**Instance**	GPMC1
Description	The configuration 1 register sets signal control parameters per chip-select.		
Type	RW		

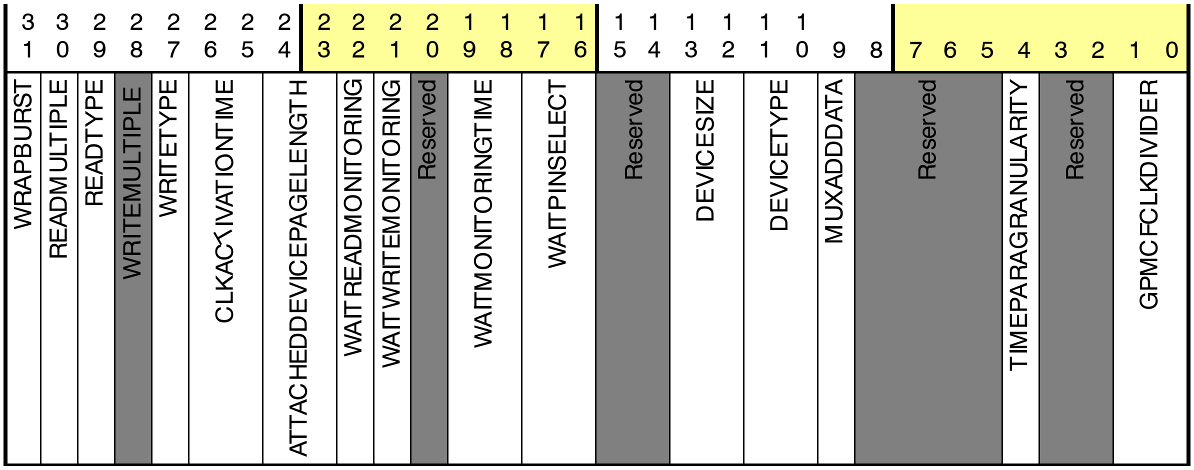


Table 12−22. GPMC_CONFIG1_0 (Continued)

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	0
30	READMULTIPLE	Selects the read single or multiple access 0x0: Single access 0x1: Multiple access (burst if synchronous, page if asynchronous)	RW	0
29	READTYPE	Selects the read mode operation 0x0: Read asynchronous 0x1: Read synchronous	RW	0
28	WRITEMULTIPLE	Selects the read single or multiple access 0x0: Single access 0x1: Multiple access (burst if synchronous, page if asynchronous)	RW	0
27	WRITETYPE	Selects the write mode operation 0x0: Write asynchronous 0x1: Write synchronous	RW	0
26:25	CLKACTIVATION-TIME	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: Not defined	RW	0x0
24:23	ATTACHED DEVICEPAGE-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
22	WAITREAD MONITORING	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	–

Table 12−22. GPMC_CONFIG1_0 (Continued)

Bits	Field Name	Description		Type	Reset
21	WAITWRITE MONITORING	Selects the WAIT monitoring configuration for write accesses		RW	0
		0x0:	Wait pin is not monitored for write accesses.		
		0x1:	Wait pin is monitored for write accesses.		
20	Reserved	Write 0s for future compatibility. Read returns 0.		RW	0
19:18	WAIT MONITORING-TIME	Selects input pin wait monitoring time		RW	0x0
		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 cycles before valid data.		
		0x3:	Not defined		
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)		RW	0x0
		0x0:	Wait input pin is WAIT0.		
		0x1:	Wait input pin is WAIT1.		
		0x2:	Wait input pin is WAIT2.		
		0x3:	Wait input pin is WAIT3.		
15:14	Reserved	Write 0s for future compatibility. Read 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 0 for CS1-7)		RW	0x−
		0x0:	8 bit		
		0x1:	16 bit		
		0x2:	Reserved		
		0x3:	Reserved		
11:10	DEVICETYPE	Selects the attached device type		RW	0x0
		0x0:	NOR flash like, asynchronous, and synchronous devices		
		0x1:	Reserved		
		0x2:	NAND flash-like devices, stream mode		
		0x3:	Reserved		
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)		RW	−
		0x0:	Nonmultiplexed attached device		
		0x1:	Address and data-multiplexed attached device		

Table 12−22. GPMC_CONFIG1_0 (Continued)

Bits	Field Name	Description	Type	Reset
8:5	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
4	TIMEPARA-GRANULARITY	Signals timing latencies scalar factor (Rd/WRCycleTime, AccessTime, PageBurstAccessTime, CSOnTime, CSRd/WrOffTime, ADVOnTime, ADVRd/WrOffTime, OEOnTime, OEOffTime, WEOnTime, WEOffTime, Cycle2CycleDelay, BusTurnAround, and TimeOutStartValue) 0x0: x1 latencies 0x1: x2 latencies	RW	0
3:2	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
1:0	GPMCFCLK DIVIDER	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

Table 12–23. GPMC_CONFIG2_0

Address Offset	0x064		
Physical Address	0x6800 A064	**Instance**	GPMC1
Description	Chip-select 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. Read returns 0.	RW	0x000
20:16	CSWROFFTIME	nCS deassertion time from start-cycle time for write accesses (0x00 corresponds to 0 GPMC_FCLK cycle, 0x01 corresponds to 1 GPMC_FCLK cycle, 0x1F corresponds to 31 GPMC_FCLK cycles)	RW	0x10
15:13	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
12:8	CSRDOFFTIME	nCS deassertion time from start-cycle time for read accesses (0x00 corresponds to 0 GPMC_FCLK cycle, 0x01 corresponds to 1 GPMC_FCLK cycle, 0x1F corresponds to 31 GPMC_FCLK cycles)	RW	0x10
7	CSEXTRADELAY	nCS adds half GPMC_FCLK cycle 0x0: nCS timing control signal is not delayed. 0x1: nCS timing control signal is delayed half GPMC_FCLK clock cycle.	RW	0
6:4	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
3:0	CSONTIME	nCS assertion time from start-cycle time (0x0 corresponds to 0 GPMC_FCLK cycle, 0x1 corresponds to 1 GPMC_FCLK cycle, 0xF corresponds to 15 GPMC_FCLK cycles)	RW	0x1

Table 12–24. GPMC_CONFIG3_0

Address Offset	0x068		
Physical Address	0x6800 A068	**Instance**	GPMC1
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	ADVEXTRADELAY			ADVONTIME			

Bits	Field Name	Description	Type	Reset
31:21	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x000
20:16	ADVWROFFTIME	nADV deassertion time from start-cycle time for write accesses (0x00 corresponds to 0 GPMC_FCLK cycle, 0x01 corresponds to 1 GPMC_FCLK cycle, 0x1F corresponds to 31 GPMC_FCLK cycles)	RW	0x02
15:13	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
12:8	ADVRDOFFTIME	nADV deassertion time from start-cycle time for read accesses(0x00 corresponds to 0 GPMC_FCLK cycle, 0x01 corresponds to 1 GPMC_FCLK cycle, 0x1F corresponds to 31 GPMC_FCLK cycles)	RW	0x02
7	ADVEXTRADELAY	nADV adds half GPMC_FCLK cycle 0x0: nADV timing control signal is not delayed. 0x1: nADV timing control signal is delayed half GPMC_FCLK clock cycle.	RW	0
6:4	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
3:0	ADVONTIME	nADV assertion time from start-cycle time (0x0 corresponds to 0 GPMC_FCLK cycle, 0x1 corresponds to 1 GPMC_FCLK cycle, 0xF corresponds to 15 GPMC_FCLK cycles)	RW	0x1

Table 12−25. GPMC_CONFIG4_0

Address Offset	0x06C		
Physical Address	0x6800 A06C	**Instance**	GPMC1
Description	nWE and nOE 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			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 0.	RW	0x0
28:24	WEOFFTIME	nWE deassertion time from start-cycle time (0x00 corresponds to 0 GPMC_FCLK cycle, 0x01 corresponds to 1 GPMC_FCLK cycle, 0x1F corresponds to 31 GPMC_FCLK cycles)	RW	0x10
23	WEEXTRADELAY	nWE adds half GPMC_FCLK cycle 0x0: nWE timing control signal is not delayed. 0x1: nWE timing control signal is delayed half GPMC_FCLK clock cycle.	RW	0
22:20	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
19:16	WEONTIME	nWE assertion time from start-cycle time (0x0 corresponds to 0 GPMC_FCLK cycle, 0x1 corresponds to 1 GPMC_FCLK cycle, 0xF corresponds to 15 GPMC_FCLK cycles)	RW	0x3
15:13	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
12:8	OEOFFTIME	nOE deassertion time from start-cycle time (0x00 corresponds to 0 GPMC_FCLK cycle, 0x01 corresponds to 1 GPMC_FCLK cycle, 0x1F corresponds to 31 GPMC_FCLK cycles)	RW	0x10
7	OEEXTRADELAY	nOE adds half GPMC_FCLK cycle. 0x0: nOE timing control signal is not delayed. 0x1: nOE timing control signal is delayed half GPMC_FCLK clock cycle .	RW	0
6:4	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
3:0	OEONTIME	nOE assertion time from start-cycle time (0x0 corresponds to 0 GPMC_FCLK cycle, 0x1 corresponds to 1 GPMC_FCLK cycle, 0xF corresponds to 15 GPMC_FCLK cycles)	RW	0x3

Table 12–26. GPMC_CONFIG5_0

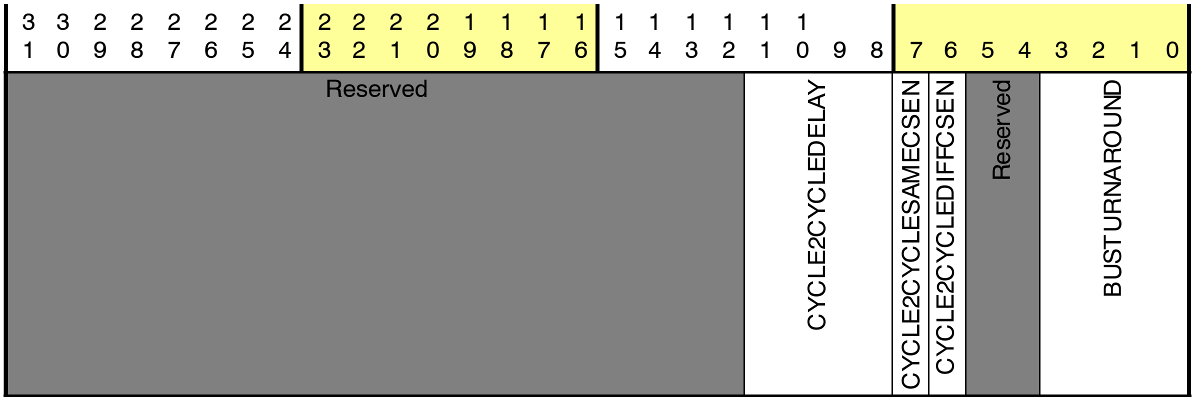
Address Offset	0x070		
Physical Address	0x6800 A070	**Instance**	GPMC1
Description	Access time and cycle time timing parameter configuration		
Type	RW		

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				PAGEBURSTACCESSTIME				Reserved			ACCESSTIME					Reserved			WRCYCLETIME					Reserved			RDCYCLETIME				

Bits	Field Name	Description	Type	Reset
31:28	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
27:24	PAGEBURST ACCESSTIME	Delay between successive words in a multiple access (0x0 corresponds to 0 GPMC_FCLK cycle, 0x1 corresponds to 1 GPMC_FCLK cycle, 0xF corresponds to 15 GPMC_FCLK cycles)	RW	0x1
23:21	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
20:16	ACCESSTIME	Delay between start-cycle time and first data valid (0x00 corresponds to 0 GPMC_FCLK cycle, 0x01 corresponds to 1 GPMC_FCLK cycle, 0x1F corresponds to 31 GPMC_FCLK cycles)	RW	0x0F
15:13	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
12:8	WRCYCLETIME	Total write cycle time (0x00 corresponds to 0 GPMC_FCLK cycle, 0x01 corresponds to 1 GPMC_FCLK cycle, 0x1F corresponds to 31 GPMC_FCLK cycles)	RW	0x11
7:5	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
4:0	RDCYCLETIME	Total read cycle time (0x00 corresponds to 0 GPMC_FCLK cycle, 0x01 corresponds to 1 GPMC_FCLK cycle, 0x1F corresponds to 31 GPMC_FCLK cycles)	RW	0x11

Table 12−27. GPMC_CONFIG6_0

Address Offset	0x074		
Physical Address	0x6800 A074	**Instance**	GPMC1
Description	Cycle2Cycle and BusTurnAround parameter configuration		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:12	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000
11:8	CYCLE2CYCLE-DELAY	Chip-select high pulse delay between successive accesses (0x0 corresponds to 0 GPMC_FCLK cycle, 0x1 corresponds to 1 GPMC_FCLK cycle, 0xF corresponds to 15 GPMC_FCLK cycles)	RW	0x0
7	CYCLE2CYCLE-SAMECSEN	Adds Cycle2CycleDelay between successive accesses to the same chip-select (any access type) 0x0: No delay between accesses 0x1: Add Cycle2CycleDelay	RW	0
6	CYCLE2CYCLE-DIFFCSEN	Adds Cycle2CycleDelay between successive accesses to a different chip-select (any access type) 0x0: No delay between the accesses 0x1: Add Cycle2CycleDelay	RW	0
5:4	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
3:0	BUSTURN AROUND	Bus turnaround latency between successive accesses to the same chip-select (read to write) or to a different chip-select (read to read and read to write) (0x0 corresponds to 0 GPMC_FCLK cycle, 0x1 corresponds to 1 GPMC_FCLK cycle, 0xF corresponds to 15 GPMC_FCLK cycles)	RW	0x0

Table 12−28. GPMC_CONFIG7_0

Address Offset	0x078		
Physical address	0x6800 A078	**Instance**	GPMC1
Description	Chip-select 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 0.	RW	0x00000
11:8	MASKADDRESS	Chip-select mask address	RW	0xF
7	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
6	CSVALID	Chip-select enable (reset value is 1 for CS0 and 0 for CS1−7) 0x0: Chip-select disabled 0x1: Chip-select enabled	RW	1
5:0	BASEADDRESS	Chip-select base address	RW	0x00

Table 12−29. GPMC_NAND_COMMAND_0

Address Offset	0x07C		
Physical Address	0x6800 A07C	**Instance**	GPMC1
Description	This is a reserved address that allows a local host to send commands to NAND type memory. For more information, see Section 12.5.10.1, *NAND Memory Device in byte or Word16 Stream Mode*.		
Type	W		

Table 12−30. GPMC_NAND_ADDRESS_0

Address Offset	0x080		
Physical Address	0x6800 A080	**Instance**	GPMC1
Description	This is a reserved address that allows a local host to send addresses to NAND type memory. For more information, see Section 12.5.10.1.		
Type	W		

Table 12−31. GPMC_NAND_DATA_0

Address Offset	0x084		
Physical Address	0x6800 A084	**Instance**	GPMC1
Description	This is a reserved address that allows a local host to read/write data from/to NAND type memory. For more information, see Section 12.5.10.1, *NAND Memory Device in byte or Word16 Stream Mode*.		
Type	RW		

Table 12–32. GPMC_CONFIG1_n

Address Offset	cs1, n = 1:0x090 cs2, n = 2: 0x0C0 cs3, n = 3: 0x0F0 cs4, n = 4: 0x120 cs5, n = 5: 0x150 cs6, n = 6: 0x180 cs7, n = 7: 0x1B0		
Physical Address	cs1, n = 1: 0x6800 A090 cs2, n = 2: 0x6800 A0C0 cs3, n = 3: 0x6800 A0F0 cs4, n = 4: 0x6800 A120 cs5, n = 5: 0x6800 A150 cs6, n = 6: 0x6800 A180 cs7, n = 7: 0x6800 A1B0	**Instance**	GPMC1
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
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 wrapping burst. Must be set if the attached device is configured in wrapping burst. 0x0: Synchronous wrapping burst not supported 0x1: Synchronous wrapping burst supported	RW	0
30	READMULTIPLE	Selects the read single or multiple access 0x0: Single access 0x1: Multiple access (burst if synchronous, page if asynchronous)	RW	0
29	READTYPE	Selects the read mode operation 0x0: Read asynchronous 0x1: Read synchronous	RW	0
28	WRITEMULTIPLE	Selects the write single or multiple access 0x0: Single access 0x1: Multiple access (burst if synchronous, considered single if asynchronous)	RW	0

Table 12–32. GPMC_CONFIG1_n (Continued)

Bits	Field Name	Description	Type	Reset
27	WRITETYPE	Selects the write mode operation 0x0: Write asynchronous 0x1: Write synchronous	RW	0
26:25	CLKACTIVATION-TIME	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: Not defined	RW	0x0
24:23	ATTACHED DEVICEPAGE-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
22	WAITREAD MONITORING	Selects the WAIT monitoring configuration for read accesses 0x0: Wait pin is not monitored for read accesses. 0x1: Wait pin is monitored for read accesses.	RW	0x0
21	WAITWRITE MONITORING	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	0
20	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
19:18	WAIT MONITORING-TIME	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 cycles before valid data. 0x3: Not defined	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.	RW	0x0

Table 12–32. GPMC_CONFIG1_n (Continued)

Bits	Field Name	Description	Type	Reset
17:16 cont'd	WAITPINSELECT cont'd	0x2: Wait input pin is WAIT2. 0x3: Wait input pin is WAIT3.	RW	0
15:14	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
13:12	DEVICESIZE	Selects the device size attached 0x0: 8 bit 0x1: 16 bit 0x2: Reserved 0x3: Reserved	RW	0x1
11:10	DEVICETYPE	Selects the attached device type 0x0: NOR flash, pSRAM, asynchronous devices 0x1: Reserved 0x2: NAND flash stream mode 0x3: Reserved	RW	0x0
9	MUXADDDATA	Enables the address and data-multiplexed protocol 0x0: Nonmultiplexed attached device 0x1: Address and data-multiplexed attached device	RW	0
8:5	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
4	TIMEPARAGRANULARITY	Signals timing latencies scalar factor (Rd/WRCycleTime, AccessTime, PageBurstAccessTime, CSOnTime, CSRd/WrOffTime, ADVOnTime, ADVRd/WrOffTime, OEOnTime, OEOffTime, WEOnTime, WEOffTime, Cycle2CycleDelay, BusTurnAround, and TimeOutStartValue) 0x0: x1 latencies 0x1: x2 latencies	RW	0
3:2	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
1:0	GPMCFCLK DIVIDER	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

Table 12–33. GPMC_CONFIG2_n

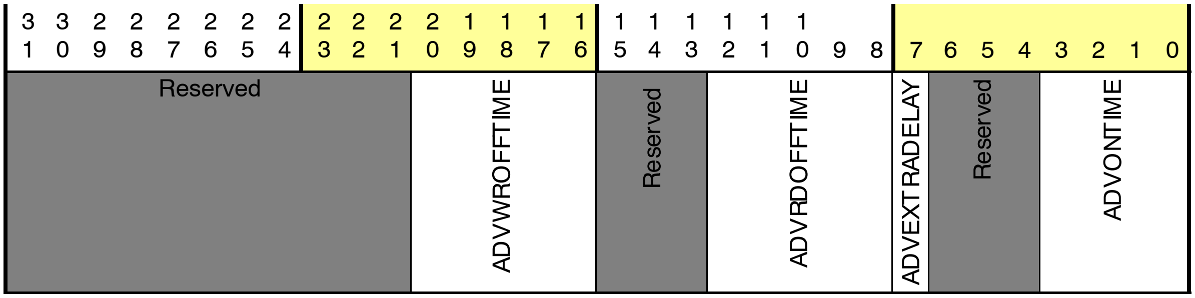
Address Offset	cs1, n = 1: 0x094 cs2, n = 2: 0x0C4 cs3, n = 3: 0x0F4 cs4, n = 4: 0x124 cs5, n = 5: 0x154 cs6, n = 6: 0x184 cs7, n = 7: 0x1B4		
Physical Address	cs1, n = 1:0x6800 A094 cs2, n = 2: 0x6800 A0C4 cs3, n = 3: 0x6800 A0F4 cs4, n = 4: 0x6800 A124 cs5, n = 5:0x6800 0x154 cs6, n = 6: 0x6800 A184 cs7, n = 7: 0x6800 A1B4	**Instance**	GPMC1
Description	Chip-select 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. Read returns 0.	RW	0x000
20:16	CSWROFFTIME	nCS deassertion time from start-cycle time for write accesses (0x00 corresponds to 0 GPMC_FCLK cycle, 0x01 corresponds to 1 GPMC_FCLK cycle, 0x1F corresponds to 31 GPMC_FCLK cycles)	RW	0x10
15:13	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
12:8	CSRDOFFTIME	nCS deassertion time from start-cycle time for read accesses (0x00 corresponds to 0 GPMC_FCLK cycle, 0x01 corresponds to 1 GPMC_FCLK cycle, 0x1F corresponds to 31 GPMC_FCLK cycles)	RW	0x10
7	CSEXTRADELAY	nCS adds half GPMC_FCLK cycle 0x0: nCS timing control signal is not delayed. 0x1: nCS timing control signal is delayed half GPMC_FCLK clock cycle.	RW	0
6:4	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
3:0	CSONTIME	nCS assertion time from start-cycle time (0x0 corresponds to 0 GPMC_FCLK cycle, 0x1 corresponds to 1 GPMC_FCLK cycle, 0xF corresponds to 15 GPMC_FCLK cycles)	RW	0x1

Table 12–34. GPMC_CONFIG3_n

Address Offset	cs1, n = 1: 0x098 cs2, n = 2: 0x0C8 cs3, n = 3: 0x0F8 cs4, n = 4: 0x128 cs5, n = 5: 0x158 cs6, n = 6: 0x188 cs7, n = 7: 0x1B8		
Physical Address	cs1, n = 1:0x6800 A098 cs2, n = 2: 0x6800 A0C8 cs3, n = 3: 0x6800 A0F8 cs4, n = 4: 0x6800 A128 cs5, n = 5:0x6800 0x158 cs6, n = 6: 0x6800 A188 cs7, n = 7: 0x6800 A1B8	**Instance**	GPMC1
Description	nADV signal timing parameter configuration		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:21	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x000
20:16	ADVWROFFTIME	nADV deassertion time from start-cycle time for write accesses (0x00 corresponds to 0 GPMC_FCLK cycle, 0x01 corresponds to 1 GPMC_FCLK cycle, 0x1F corresponds to 31 GPMC_FCLK cycles).	RW	0x02
15:13	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
12:8	ADVRDOFFTIME	nADV deassertion time from start-cycle time for read accesses(0x00 corresponds to 0 GPMC_FCLK cycle, 0x01 corresponds to 1 GPMC_FCLK cycle, 0x1F corresponds to 31 GPMC_FCLK cycles)	RW	0x02
7	ADVEXTRADELAY	nADV adds half GPMC_FCLK cycle 0x0: nADV timing control signal is not delayed. 0x1: nADV timing control signal is delayed half GPMC_FCLK clock cycle.	RW	0
6:4	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
3:0	ADVONTIME	nADV assertion time from start-cycle time (0x0 corresponds to 0 GPMC_FCLK cycle, 0x1 corresponds to 1 GPMC_FCLK cycle, 0xF corresponds to 15 GPMC_FCLK cycles)	RW	0x1

Table 12–35. GPMC_CONFIG4_n

Address Offset

cs1, n = 1: 0x09C
cs2, n = 2: 0x0CC
cs3, n = 3: 0x0FC
cs4, n = 4: 0x12C
cs5, n = 5: 0x15C
cs6, n = 6: 0x18C
cs7, n = 7: 0x1BC

Physical Address

cs1, n = 1:0x6800 A09C
cs2, n = 2: 0x6800 A0CC
cs3, n = 3: 0x6800 A0FC
cs4, n = 4: 0x6800 A12C
cs5, n = 5:0x6800 0x15C
cs6, n = 6: 0x6800 A18C
cs7, n = 7: 0x6800 A1BC

Instance GPMC1

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 0.	RW	0x0
28:24	WEOFFTIME	nWE deassertion time from start-cycle time (0x00 corresponds to 0 GPMC_FCLK cycle, 0x01 corresponds to 1 GPMC_FCLK cycle, 0x1F corresponds to 31 GPMC_FCLK cycles)	RW	0x10
23	WEEXTRADELAY	nWE adds half GPMC_FCLK cycle 0x0: nWE timing control signal is not delayed. 0x1: nWE timing control signal is delayed half GPMC_FCLK clock cycle.	RW	0
22:20	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
19:16	WEONTIME	nWE assertion time from start-cycle time (0x0 corresponds to 0 GPMC_FCLK cycle, 0x1 corresponds to 1 GPMC_FCLK cycle, 0xF corresponds to 15 GPMC_FCLK cycles)	RW	0x3
15:13	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0

Table 12–35. GPMC_CONFIG4_n (Continued)

Bits	Field Name	Description	Type	Reset
12:8	OEOFFTIME	nOE deassertion time from start-cycle time (0x00 corresponds to 0 GPMC_FCLK cycle, 0x01 corresponds to 1 GPMC_FCLK cycle, 0x1F corresponds to 31 GPMC_FCLK cycles)	RW	0x10
7	OEEXTRADELAY	nOE adds half GPMC_FCLK cycle 0x0: nOE timing control signal is not delayed. 0x1: nOE timing control signal is delayed half GPMC_FCLK clock cycle.	RW	0
6:4	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
3:0	OEONTIME	nOE assertion time from start-cycle time (0x0 corresponds to 0 GPMC_FCLK cycle, 0x1 corresponds to 1 GPMC_FCLK cycle, 0xF corresponds to 15 GPMC_FCLK cycles)	RW	0x3

Table 12–36. GPMC_CONFIG5_n

Address Offset	cs1, n = 1: 0x0A0 cs2, n = 2: 0x0D0 cs3, n = 3: 0x100 cs4, n = 4: 0x130 cs5, n = 5: 0x160 cs6, n = 6: 0x190 cs7, n = 7: 0x1C0		
Physical Address	cs1, n = 1:0x6800 A0A0 cs2, n = 2: 0x6800 A0D0 cs3, n = 3: 0x6800 A100 cs4, n = 4: 0x6800 A130 cs5, n = 5:0x6800 0x160 cs6, n = 6: 0x6800 A190 cs7, n = 7: 0x6800 A1C0	**Instance**	GPMC1
Description	Access time and cycle time 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				PAGEBURSTACCESSTIME				PAGEBURSTACCESSTIME			ACCESSTIME					Reserved			WRCYCLETIME					Reserved			RDCYCLETIME				

Bits	Field Name	Description	Type	Reset
31:28	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
27:24	PAGEBURST ACCESSTIME	Delay between successive words in a multiple access (0x0 corresponds to 0 GPMC_FCLK cycle, 0x1 corresponds to 1 GPMC_FCLK cycle, 0xF corresponds to 15 GPMC_FCLK cycles)	RW	0x1
23:21	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
20:16	ACCESSTIME	Delay between start-cycle time and first data valid (0x00 corresponds to 0 GPMC_FCLK cycle, 0x01 corresponds to 1 GPMC_FCLK cycle, 0x1F corresponds to 31 GPMC_FCLK cycles)	RW	0x0F
15:13	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
12:8	WRCYCLETIME	Total write cycle time (0x00 corresponds to 0 GPMC_FCLK cycle, 0x01 corresponds to 1 GPMC_FCLK cycle, 0x1F corresponds to 31 GPMC_FCLK cycles)	RW	0x11
7:5	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
4:0	RDCYCLETIME	Total read cycle time (0x00 corresponds to 0 GPMC_FCLK cycle, 0x01 corresponds to 1 GPMC_FCLK cycle, 0x1F corresponds to 31 GPMC_FCLK cycles)	RW	0x11

Table 12–37. GPMC_CONFIG6_n

Address Offset	cs1, n = 1: 0x0A4 cs2, n = 2: 0x0D4 cs3, n = 3: 0x104 cs4, n = 4: 0x134 cs5, n = 5: 0x164 cs6, n = 6: 0x194 cs7, n = 7: 0x1C4		
Physical Address	cs1, n = 1:0x6800 A0A4 cs2, n = 2: 0x6800 A0D4 cs3, n = 3: 0x6800 A104 cs4, n = 4: 0x6800 A134 cs5, n = 5:0x6800 0x164 cs6, n = 6: 0x6800 A194 cs7, n = 7: 0x6800 A1C4	**Instance**	GPMC1
Description	Cycle2Cycle and BusTurnAround 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																				CYCLE2CYCLEDELAY				CYCLE2CYCLESAMECSEN	CYCLE2CYCLEDIFFSCEN	Reserved		BUSTURNAROUND			

Bits	Field Name	Description	Type	Reset
31:12	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000
11:8	CYCLE2CYCLE-DELAY	Chip-select high pulse delay between successive accesses (0x0 corresponds to 0 GPMC_FCLK cycle, 0x1 corresponds to 1 GPMC_FCLK cycle, 0xF corresponds to 15 GPMC_FCLK cycles)	RW	0x0
7	CYCLE2CYCLE-SAMECSEN	Add Cycle2CycleDelay between successive accesses to the same chip-select (any access type) 0x0: No delay between accesses 0x1: Add Cycle2CycleDelay	RW	0
6	CYCLE2CYCLE-DIFFCSEN	Add Cycle2CycleDelay between successive accesses to a different chip-select (any access type) 0x0: No delay between accesses 0x1: Add Cycle2CycleDelay	RW	0
5:4	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0

Table 12–37. GPMC_CONFIG6_n (Continued)

Bits	Field Name	Description	Type	Reset
3:0	BUSTURN AROUND	Bus turnaround latency between successive accesses to the same chip-select (read to write) or to a different chip-select (read to read and read to write) (0x0 corresponds to 0 GPMC_FCLK cycle, 0x1 corresponds to 1 GPMC_FCLK cycle, 0xF corresponds to 15 GPMC_FCLK cycles)	RW	0x0

Table 12–38. GPMC_CONFIG7_n

Address Offset	cs1, n = 1: 0x0A8 cs2, n = 2: 0x0D8 cs3, n = 3: 0x108 cs4, n = 4: 0x138 cs5, n = 5: 0x168 cs6, n = 6: 0x198 cs7, n = 7: 0x1C8		
Physical Address	cs1, n = 1:0x6800 A0A8 cs2, n = 2: 0x6800 A0D8 cs3, n = 3: 0x6800 A108 cs4, n = 4: 0x6800 A138 cs5, n = 5:0x6800 0x168 cs6, n = 6: 0x6800 A198 cs7, n = 7: 0x6800 A1C8	**Instance**	GPMC1
Description	Chip-select 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																				MASK ADDRESS				Reserved	CSVALID	BASEADDRESS					

Bits	Field Name	Description	Type	Reset
31:12	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000
11:8	MASKADDRESS	Chip-select mask address	RW	0xF
7	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
6	CSVALID	Chip-select enable (reset value is 1 for CS0 and 0 for CS1–7) 0x0: Chip-select disabled 0x1: Chip-select enabled	RW	0
5:0	BASEADDRESS	Chip-select base address	RW	0x00

Table 12–39. GPMC_NAND_COMMAND_n

Address Offset	cs1, n = 1: 0x0AC cs2, n = 2: 0x0DC cs3, n = 3: 0x10C cs4, n = 4: 0x13C cs5, n = 5: 0x16C cs6, n = 6: 0x19C cs7, n = 7: 0x1CC		
Physical Address	cs1, n = 1:0x6800 A0AC cs2, n = 2: 0x6800 A0DC cs3, n = 3: 0x6800 A10C cs4, n = 4: 0x6800 A13C cs5, n = 5:0x6800 0x16C cs6, n = 6: 0x6800 A19C cs7, n = 7: 0x6800 A1CC	**Instance**	GPMC1
Description	This is a reserved address that allows a local host to send commands to NAND type memory. For more information, see Section 12.5.10.1, *NAND Memory Device in Byte or word16 Stream Mode*.		
Type	W		

Table 12–40. GPMC_NAND_ADDRESS_n

Address Offset	cs1, n = 1: 0x0B0 cs2, n = 2: 0x0E0 cs3, n = 3: 0x110 cs4, n = 4: 0x140 cs5, n = 5: 0x170 cs6, n = 6: 0x1A0 cs7, n = 7: 0x1D0		
Physical Address	cs1, n = 1:0x6800 A0B0 cs2, n = 2: 0x6800 A0E0 cs3, n = 3: 0x6800 A100 cs4, n = 4: 0x6800 A140 cs5, n = 5:0x6800 0x170 cs6, n = 6: 0x6800 A1A0 cs7, n = 7: 0x6800 A1D0	**Instance**	GPMC1
Description	This is a reserved address that allows a local host to send addresses to NAND type memory. For more information, see Section 12.5.10.1, *NAND Memory Device in Byte or word16 Stream Mode*.		
Type	W		

Table 12−41. GPMC_NAND_DATA_n

Address Offset	cs1, n = 1: 0x0B4 cs2, n = 2: 0x0E4 cs3, n = 3: 0x114 cs4, n = 4: 0x144 cs5, n = 5: 0x174 cs6, n = 6: 0x1A4 cs7, n = 7: 0x1D4		
Physical address	cs1, n = 1:0x6800 A0B4 cs2, n = 2: 0x6800 A0E4 cs3, n = 3: 0x6800 A114 cs4, n = 4: 0x6800 A144 cs5, n = 5:0x6800 0x174 cs6, n = 6: 0x6800 A1A4 cs7, n = 7: 0x6800 A1D4	**Instance**	GPMC1
Description	This is a reserved address that allows a local host to read/write data from/to NAND type memory. For more information, see Section 12.5.10.1, *NAND Memory Device in Byte or word16 Stream Mode*.		
Type	RW		

Table 12−42. GPMC_PREFETCH_CONFIG1

Address Offset	0x000001E0		
Physical Address	0x6800 A1E0	**Instance**	GPMC1
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			ENALBEOPTIMIZEDACCESS	ENGINECSSELECTOR			PFPWENROUNDROBIN	RESERVED			PFPWWEIGHTEDPRIO				RESERVED	FIFOTHRESHOLD							ENABLEENGINE	RESERVED	WAITPINSELECTOR		SYNCHROMODE	DMAMODE	ENDIANNISMTYPE	ACCESSMODE

Bits	Field Name	Description	Type	Reset
31	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0
30:28	CYCLE OPTIMIZATION	Define the number of GPMC_FCLK cycles to be subtracted from RdCycleTime, WrCycleTime, AccessTime, CSRdOffTime, CSWrOffTime, ADVRdOffTime, ADVWrOffTime, OEOffTime, WEOffTime (0x0 corresponds to 0 GPMC_FCLK cycle, 0x1 corresponds to 1 GPMC_FCLK cycle, 0x7 corresponds to 7 GPMC_FCLK cycles).	RW	0x0
27	ENABLE OPTIMIZED ACCESS	Enables access cycle optimization 0x0: Access cycle optimization is disabled.	RW	0

Table 12−42. GPMC_PREFETCH_CONFIG1 (Continued)

Bits	Field Name	Description	Type	Reset
		0x1: Access cycle optimization is enabled.		
26:24	ENGINECS SELECTOR	Selects the chip-select where the prefetch and write-posting engine is active (0x0 corresponds to CS0, 0x1 corresponds to CS1, 0x7 corresponds to CS7)	RW	0x0
23	PFPWENROUNDROBIN	Enables prefetch and write-posting engine round robin arbitration 0x0 Prefetch and write-posting engine round robin arbitration is disabled. 0x1 Prefetch and write-posting engine round robin arbitration is enabled.	RW	0
22:20	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
19:16	PFPWWEIGHTEDPRIO	When an arbitration occurs between a direct memory access and an RW 0x0 PFPW engine access, the direct memory access is always serviced. If PFPWENROUNDROBIN is enabled, 0x0: The next access is granted to the prefetch and write-posting engine. 0x1: The next two accesses are granted to the prefetch and write-posting engine. 0xF: The next 16 accesses are granted to the prefetch and write-posting engine.	RW	0x0
15	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
14:8	FIFO THRESHOLD	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 corresponds to 0 byte, 0x01 corresponds to 1 byte, 0x40 corresponds to 64 bytes)	RW	0x40
7	ENABLEENGINE	Enables the prefetch and write-posting engine 0x0: Prefetch and post-writing engine is disabled. 0x1: Prefetch and post-writing engine is enabled.	RW	0
6	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
5:4	WAITPIN SELECTOR	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 the chip−select	RW	0

Bits	Field Name	Description		Type	Reset
		0x0:	Engine starts the access to the chip-select as soon as StartEngine is set.		
		0x1:	Engine starts the access to the chip-select as soon as StartEngine is set and wait-to-no-wait edge detection on the selected wait pin.		

Table 12−42. GPMC_PREFETCH_CONFIG1 (Continued)

Bits	Field Name	Description		Type	Reset
2	DMAMODE	Selects interrupt synchronization or DMA request synchronization		RW	0
		0x0:	Interrupt synchronization is enabled. Only interrupt line is activated on FIFO threshold crossing.		
		0x1:	DMA request synchronization is enabled. A DMA request protocol is used.		
1	ENDIANNISM-TYPE	Write 0 for future compatibility. Read returns 0.		RW	0
		0x0:	Little-endian		
		0x1:	Big-endian		
0	ACCESSMODE	Selects prefetch read or write-posting accesses		RW	0
		0x0:	Prefetch read mode		
		0x1:	Write posting mode		

Table 12−43. GPMC_PREFETCH_CONFIG2

Address Offset	0x1E4		
Physical Address	0x6800 A1E4	**Instance**	GPMC1
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 0.	RW	0x00000
13:0	TRANSFER-COUNT	Selects the number of bytes to be read or written by the engine to the selected chip-select (0x0000 corresponds to 0 byte, 0x0001 corresponds to 1 byte, 0x2000 corresponds to 8K bytes)	RW	0x0000

Table 12−44. GPMC_PREFETCH_CONTROL

Address Offset	0x1EC		
Physical Address	0x6800 A1EC	**Instance**	GPMC1
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

Table 12−44. GPMC_PREFETCH_CONTROL (Continued)

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000000
0	STARTENGINE	Resets the FIFO pointer and starts the engine 0x0: Write 0 stops the engine. Read 0: Engine is stopped. 0x1: Write 1 resets the FIFO pointer to 0x0 in prefetch mode and 0x40 in write-posting mode and starts the engine. Read 1: Engine is running.	RW	0

Table 12−45. GPMC_PREFETCH_STATUS

Address Offset	0x1F0		
Physical Address	0x6800 A1F0	**Instance**	GPMC1
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	0
30:24	FIFOPOINTER	Number of available bytes to be read or number of free empty byte places to be written (0x00 corresponds to 0 byte available to be read or 0 free empty place to be written, 0x40 corresponds to 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 0.	RW	0x00
16	FIFOTHRESHOLDSTATUS	Set when FIFOPointer exceeds FIFOThreshold value 0x0: FIFOPointer smaller or equal to FIFO threshold. Writing to this bit has no effect. 0x1: FIFOPointer greater than FIFO threshold. Writing to this bit has no effect.	R	0

Table 12−45. GPMC_PREFETCH_STATUS (Continued)

Bits	Field Name	Description	Type	Reset
15:14	Reserved	Write 0s for future compatibility. Read returns 0.	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 corresponds to 0 byte remaining to be read or to be written, 0x0001 corresponds to 1 byte remaining to be read or to be written, 0x2000 corresponds to 8K bytes remaining to be read or to be written)	R	0x0000

Table 12−46. GPMC_ECC_CONFIG

Address Offset	0x1F4		
Physical Address	0x6800 A1F4	**Instance**	GPMC1
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																								ECC16B	Reserved			ECCCS			ECCENABLE

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x000000
7	ECC16B	Selects an ECC calculated on 16 columns 0x0: ECC calculated on 8 columns 0x1: ECC calculated on 16 columns	RW	0
6:4	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
3:1	ECCCS	Selects the chip-select 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	0

Table 12–47. GPMC_ECC_CONTROL

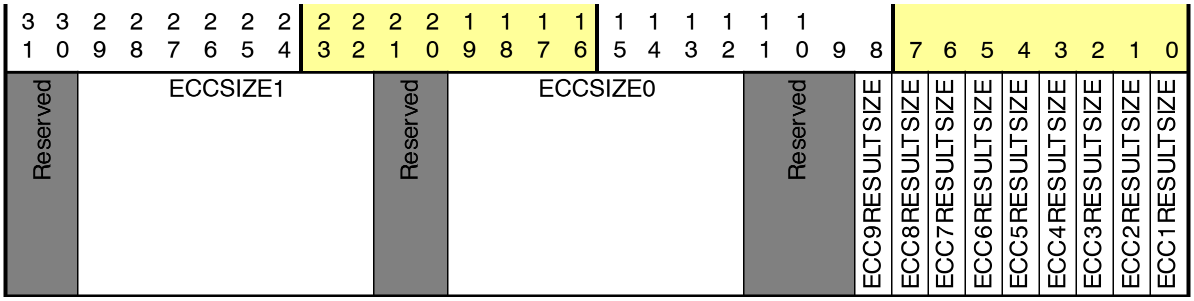
Address Offset	0x1F8		
Physical Address	0x6800 A1F8	**Instance**	GPMC1
Description	ECC control		
Type	RW		

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 0.	RW	0x000000
8	ECCCLEAR	Clear all ECC result registers (reads return 0 – writing 1 to this field clears all ECC result registers – writing 0 is ignored)	RW	0
7:4	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
3:0	ECCPOINTER	Selects ECC result register (reads of this field give the dynamic position of the ECC pointer – writes to this field select the ECC result register where the first ECC computation is stored); writing other values disables the ECC engine (ECCEnable bit of GPMC_ECC_CONFIG set to 0) 0x0: Writing 0000 disables the ECC engine (ECCEnable bit of GPMC_ECC_CONFIG set to 0). 0x1: ECC result register 1 is selected. 0x2: ECC result register 2 is selected. 0x3: ECC result register 3 is selected. 0x4: ECC result register 4 is selected. 0x5: ECC result register 5 is selected. 0x6: ECC result register 6 is selected. 0x7: ECC result register 7 is selected. 0x8: ECC result register 8 is selected. 0x9: ECC result register 9 is selected.	RW	0x0

Table 12–48. GPMC_ECC_SIZE_CONFIG

Address Offset	0x1FC		
Physical Address	0x6800 A1FC	**Instance**	GPMC1
Description	ECC size		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:30	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
29:22	ECCSIZE1	Defines ECC size 1 (0x00 corresponds to 2 bytes, 0x01 corresponds to 4 bytes, 0x02 corresponds to 6 bytes, 0x03 corresponds to 8 bytes, 0xFF corresponds to 512 bytes)	RW	0xFF
21:20	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
19:12	ECCSIZE0	Defines ECC size 0 (0x00 corresponds to 2 bytes, 0x01 corresponds to 4 bytes, 0x02 corresponds to 6 bytes, 0x03 corresponds to 8 bytes, 0xFF corresponds to 512 bytes)	RW	0xFF
11:9	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
8	ECC9RESULTSIZE	Selects ECC size for ECC 9 result register 0x0: ECCSize0 is selected. 0x1: ECCSize1 is selected.	RW	0
7	ECC8RESULTSIZE	Selects ECC size for ECC 8 result register 0x0: ECCSize0 is selected. 0x1: ECCSize1 is selected.	RW	0
6	ECC7RESULTSIZE	Selects ECC size for ECC 7 result register 0x0: ECCSize0 is selected. 0x1: ECCSize1 is selected.	RW	0
5	ECC6RESULTSIZE	Selects ECC size for ECC 6 result register 0x0: ECCSize0 is selected. 0x1: ECCSize1 is selected .	RW	0
4	ECC5RESULTSIZE	Selects ECC size for ECC 5 result register 0x0: ECCSize0 is selected. 0x1: ECCSize1 is selected.	RW	0

Table 12–48. GPMC_ECC_SIZE_CONFIG (Continued)

Bits	Field Name	Description	Type	Reset
3	ECC4RESULT-SIZE	Selects ECC size for ECC 4 result register 0x0: ECCSize0 is selected. 0x1: ECCSize1 is selected.	RW	0
2	ECC3RESULT-SIZE	Selects ECC size for ECC 3 result register 0x0: ECCSize0 is selected. 0x1: ECCSize1 is selected.	RW	0
1	ECC2RESULT-SIZE	Selects ECC size for ECC 2 result register 0x0: ECCSize0 is selected. 0x1: ECCSize1 is selected.	RW	0
0	ECC1RESULT-SIZE	Selects ECC size for ECC 1 result register 0x0: ECCSize0 is selected. 0x1: ECCSize1 is selected.	RW	0

Table 12−49. GPMC_ECCy_RESULT

Address Offset	y = 1: 0x200
	y = 2: 0x204
	y = 3: 0c208
	y = 4: 020C
	y = 5: 0x210
	y = 6: 0x214
	y = 7: 0x218
	y = 8: 0x21C
	y = 9: 0x220		

Physical Address	y = 1: 0x6800 A200
	y = 2: 0x6800 A204
	y = 3: 0x6800 A208
	y = 4: 0x6800 A20C
	y = 5: 0x6800 A210
	y = 6: 0x6800 A214
	y = 7: 0x6800 A218
	y = 8: 0x6800 A21C
	y = 9: 0x6800 A220
		Instance	GPMC1

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	P258O	F128C	P64O	P32O	P16O	P8O	P4O	P2O	P1O	Reserved				P2048E	P1024E	P512E	P258E	F128E	P64E	P32E	P16E	P8E	P4E	P2E	P1E

Bits	Field Name	Description	Type	Reset
31:28	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
27	P2048O	Odd Row Parity bit 2048, used only for ECC computed on 512 bytes	R	0
26	P1024O	Odd Row Parity bit 1024	R	0
25	P512O	Odd Row Parity bit 512	R	0
24	P256O	Odd Row Parity bit 256	R	0
23	P128O	Odd Row Parity bit 128	R	0
22	P64O	Odd Row Parity bit 64	R	0
21	P32O	Odd Row Parity bit 32	R	0
20	P16O	Odd Row Parity bit 16	R	0
19	P8O	Odd Row Parity bit 8	R	0
18	P4O	Odd Column Parity bit 4	R	0
17	P2O	Odd Column Parity bit 2	R	0
16	P1O	Odd Column Parity bit 1	R	0
15:12	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
11	P2048E	Even Row Parity bit 2048, used only for ECC computed on 512 bytes	R	0

Table 12–49. GPMC_ECCy_RESULT (Continued)

Bits	Field Name	Description	Type	Reset
10	P1024E	Even Row Parity bit 1024	R	0
9	P512E	Even Row Parity bit 512	R	0
8	P256E	Even Row Parity bit 256	R	0
7	P128E	Even Row Parity bit 128	R	0
6	P64E	Even Row Parity bit 64	R	0
5	P32E	Even Row Parity bit 32	R	0
4	P16E	Even Row Parity bit 16	R	0
3	P8E	Even Row Parity bit 8	R	0
2	P4E	Even Column Parity bit 4	R	0
1	P2E	Even Column Parity bit 2	R	0
0	P1E	Even Column Parity bit 1	R	0

12.7 SDRAM Controller Subsystem Overview

The SDRAM controller subsystem module provides connectivity between the OMAP2420 application processors and external (or stacked) DRAM memory components. The module includes support for single-data-rate SDRAM (low-power SDR) and double-data-rate SDRAM (mobile DDR).

The SDRAM controller subsystem provides a high-performance interface to a variety of fast memory devices. It comprises two submodules:

- The SDRAM memory scheduler (SMS), consisting of the scheduler, and virtual rotated frame-buffer modules
- The SDRAM controller (SDRC)

Note:

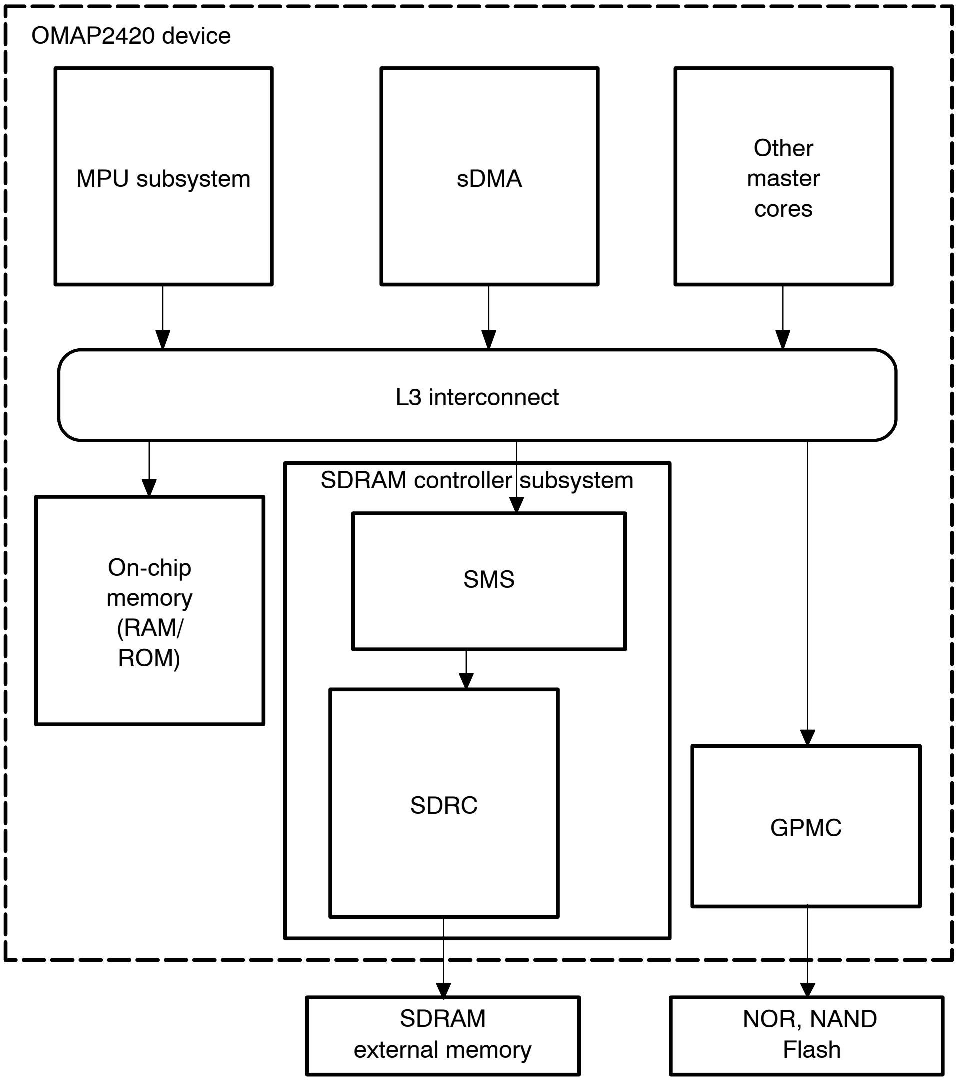
Stacked SDRAM is not used in the OMAP2420 device.

WARNING

DDR SDRAM and SDR SDRAM

It is not possible to connect the memory types (DDR SDRAM and SDR SDRAM).

Figure 12–28. SDRAM Controller Subsystem Environment



12.7.1 Features

The main features of the SDRAM controller subsystem module are:

- Virtual rotated frame-buffer module
  - Minimizes SDRAM page-miss penalty when accessing rotated (non-sequentially addressed) lines in a frame buffer
  - Four independent contexts
  - Supports rotations of 0°, 90°, 180°, or 270°
  - Transparent to software applications
- Memory-access scheduler
  - Optimizes latency and bandwidth use between initiators
  - Three quality-of-service (QoS) classes of initiator

  - Eight FIFO request queues divided between the QoS classes
  - Programmable arbitration between and within the QoS classes

- SDRAM controller (SDRC)
  - Supports two independently configurable memory areas with separate chip-selects
  - Supports the following memory types:
    - Low-power SDR SDRAM
    - Mobile DDR SDRAM
  - Supports 16M-bit, 32M-bit, 64M-bit, 128M-bit, 256M-bit, 512M-bit, 1G-bit, and 2G-bit devices
  - Memory device organization
    - 2-bank support for 16M bits and 32M bits
    - 4-bank support for 64M bits, 128M bits, 256M bits, 512M bits, 1G bits and 2G bits
    - Flexible row/column address multiplexing schemes
    - 16-bit or 32-bit external data width
  - Fully pipelined on SDRAM interface
  - Burst support
    - System burst support

      –Incrementing burst support

      –Wrapping burst support for critical word-first cache line refill
    - Memory burst support

      –System burst for SDR SDRAM: System burst translated into memory burst of 2

      –System burst for mobile DDR SDRAM: System burst translated into memory burst size of 4

      –Read interrupt by read, write interrupt by write
  - Supports standard and enhanced low-power memory features
- Power-management support

Note:

Regular and DDR1 SDRAM devices are not supported in the OMAP2420, because of electrical incompatibility.

12.8 SDRAM Controller Subsystem Environment

12.8.1 SDRAM Controller Subsystem Description

Figure 12−29 shows the SDRAM controller subsystem interfacing with two external memories: one 16-bit SDR SDRAM and one 32-bit SDR memory. Figure 12−30 shows the SDRAM controller subsystem interfacing with two external memories: one 16-bit DDR SDRAM and one 32-bit DDR memory.

Figure 12−29. SDRAM Controller Subsystem Connections to SDR SDRAM

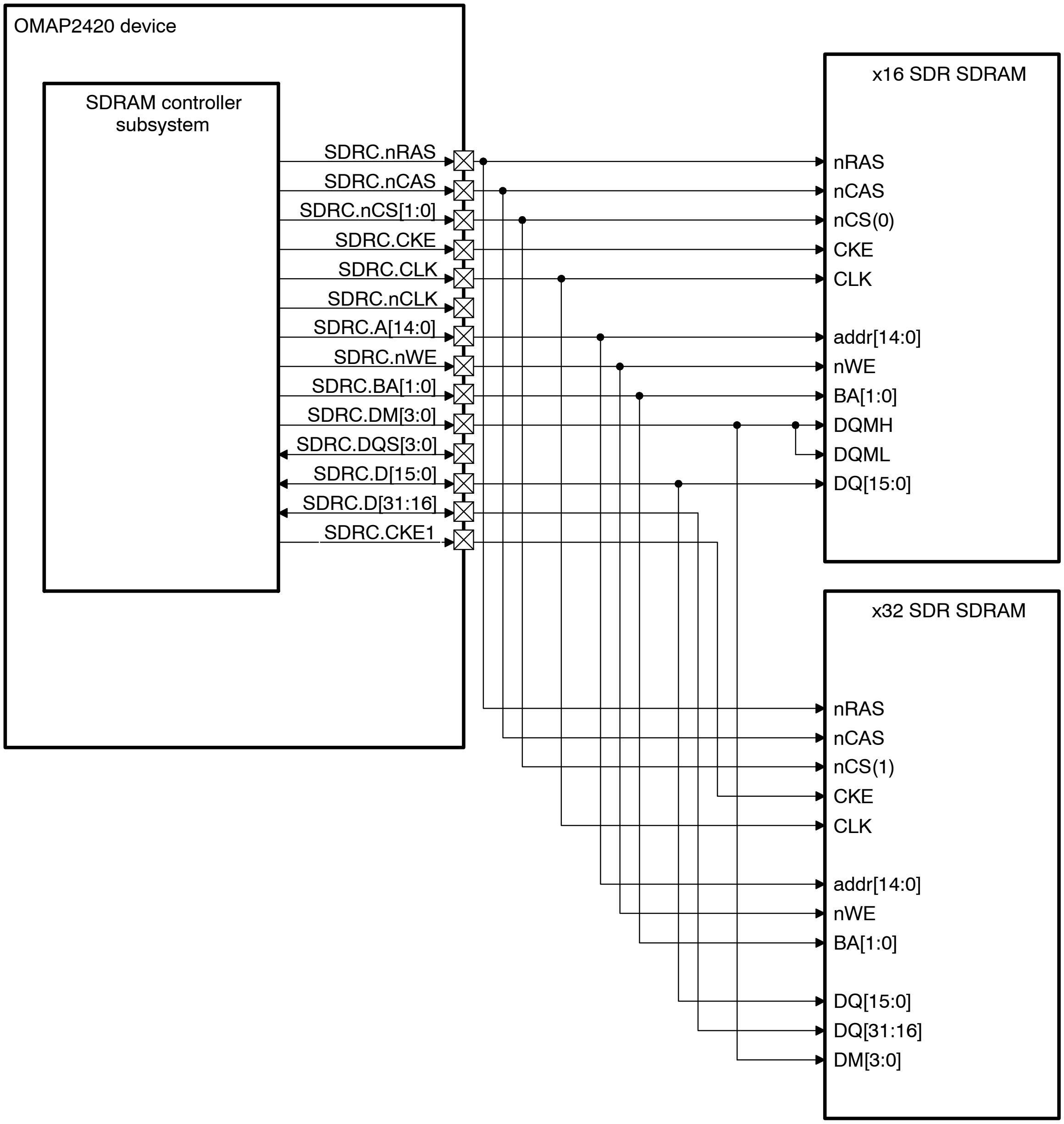
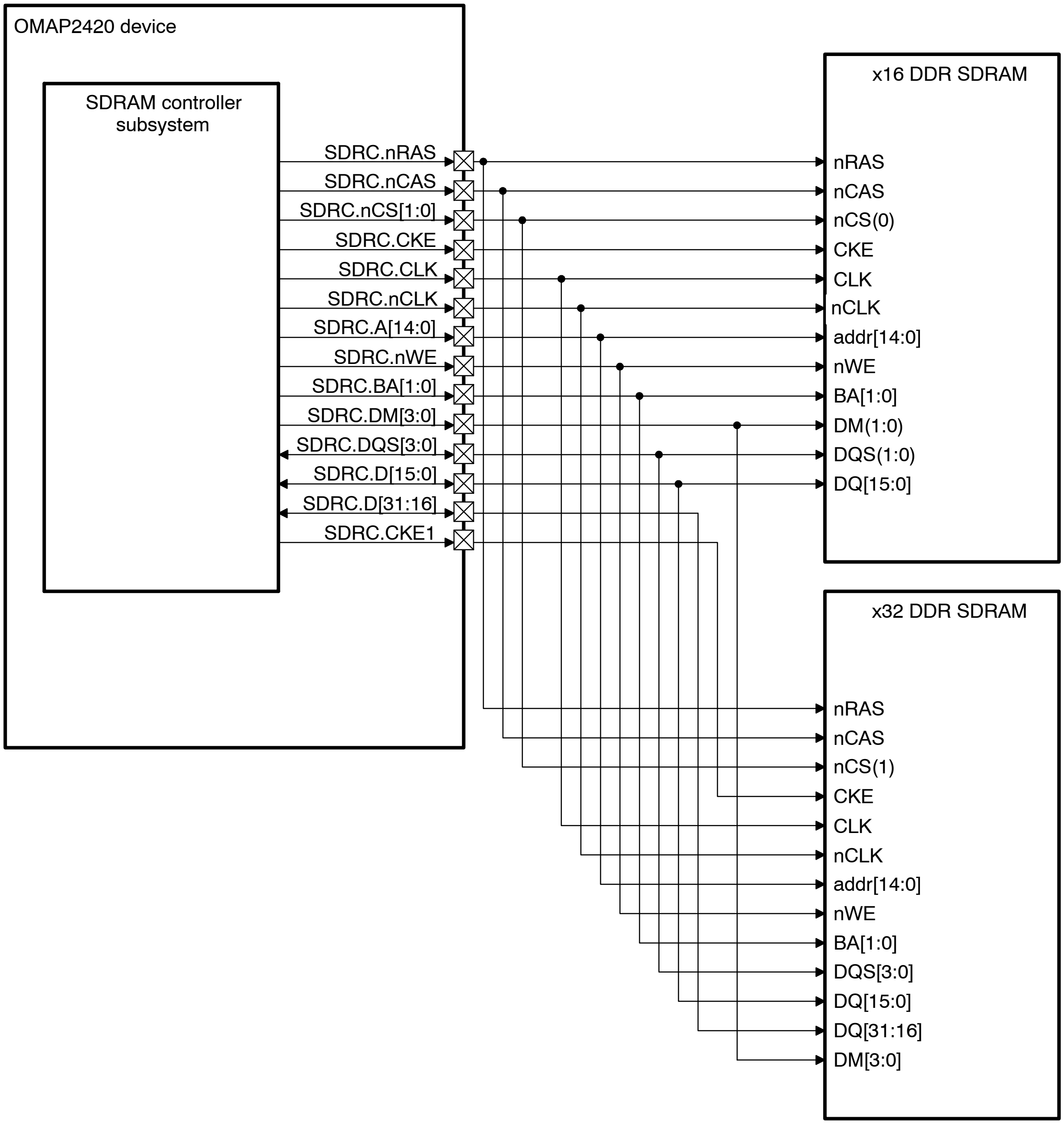


Figure 12−30. SDRAM Controller Subsystem Connections to DDR SDRAM



DDR SDRAM and SDR SDRAM

It is not possible to connect the DDR SDRAM and SDR SDRAM memory types on the OMAP2420.

WARNING

Table 12−50 lists SDRAM controller subsystem I/O pins.

Table 12−50. SDRAM Controller Subsystem I/O Description

Pin	Type	Description
SDRC.D[31:16]	I/O	Data bus, either 16 or 32 bits wide, stacked or non-stacked, depending on configuration options
SDRC.D[15:0]	I/O	
SDRC.BA[1:0]	O	Bank address 1−0
SDRC.A[14:0]	O	Address
SDRC.nCS[1:0]	O	Chip-select 1−0 (active low)
SDRC.CLK	I/O	Clock
SDRC.nCLK	O	Clock invert (DDR only)
SDRC.CKE0	O	Clock enable
SDRC.CKE1	O	Clock enable
SDRC.nRAS	O	Row address strobe (active low)
SRC.nCAS	O	Column address strobe (active low)
SDRC.nWE	O	Write enable (active low)
SDRC.DM[3:0]	O	Data mask/OE for SDR read operation
SDRC.DQS[3:0]	I/O	Data strobe (DDR only)

12.8.2 External Interface Configuration

12.8.2.1 CS0, CS1 Memory Spaces

The SDRC can be connected independently to two external SDRAM memories. Low-power SDR SDRAMs are supported in sizes of 16M bits, 32M bits, 64M bits, 128M bits, 256M bits, 1G bits, and 2G bits.

Note:

The OMAP2420 device does not support either regular or DDR1 SDRAM (see Section 12.11.3.1, *Chip-Select Configuration*).

12.8.2.2 Alternating Current Timing Control

The SDRC controller permits the configuration of many of the ac timing parameters controlling the SDRAM transaction timings.

For any memory type attached to the SDRC, there is a requirement to adapt all transaction timing parameters to the attached device.

A totally independent, complete set of parameters exists for each chip-select.

The ac parameters and their quantification are summarized in Section 12.11.3.1, *Chip-Select Configuration*.

12.8.2.3 Address Multiplexing

Table 12−51 and Figure 12−31 through Figure 12−33 describe the address multiplexing configurations.

Table 12−51. SDRC Address Multiplexing Scheme Selection Versus SDRAM Configurations

x16 Memory Interface

Banks		**Column Address**		**Row Address**		**MUX Scheme**	**Total Size (M bits)**	**Number of Devices**	**Device Organization**
BA0	1	A0−A7	8	A0−A10	11	MUX1	16	1	1Mx16
BA0	1	A0−A7	8	A0−A11	12	MUX2	32	1	2Mx16
BA1,BA0	2	A0−A7	8	A0−A11	12	MUX2	64	1	4Mx16
BA1,BA0	2	A0−A8	9	A0−A11	12	MUX4	128	2	8Mx8
								1	8Mx16
BA1,BA0	2	A0−A8	9	A0−A12	13	MUX7	256	1	16Mx16
BA1,BA0	2	A0−A8	9	A0−A13	14	MUX26	512	1	32Mx16
BA1,BA0	2	A0−A9	10	A0−A11	12	MUX6	256	2	16Mx8
BA1,BA0	2	A0−A9	10	A0−A12	13	MUX10	512	2	32Mx8
								1	32Mx16
BA1,BA0	2	A0−A9	10	A0−A13	14	MUX13	1024	1	64Mx16
BA1,BA0	2	A0−A9;A11	11	A0−A12	13	MUX12	1024	2	64Mx8

x32 Memory Interface

Banks		**Column Address**		**Row Address**		**MUX Scheme**	**Total Size (M bits)**	**Number of Devices**	**Device Organization**
BA0	1	A0−A7	8	A0−A11	12	MUX5	64	2	2Mx16
BA1,BA0	2	A0−A7	8	A0−A10	11	MUX3	64	1	2Mx32
BA1,BA0	2	A0−A7	8	A0−A11	12	MUX5	128	1	4Mx32
BA1,BA0	2	A0−A7	8	A0−A12	13	MUX9	256	1	8Mx32
BA1,BA0	2	A0−A7	8	A0−A13	14	MUX27	512	1	16Mx32
BA1,BA0	2		8	A0−A14	15	MUX28	1024	1	32Mx32
BA1,BA0	2	A0−A8	9	A0−A11	12	MUX8	256	4	8Mx8
								2	8Mx16
BA1,BA0	2	A0−A8	9	A0−A12	13	MUX24	512	1	16Mx32
BA1,BA0	2	A0−A9	10	A0−A11	12	MUX11	512	4	16Mx8
BA1,BA0	2	A0−A9	10	A0−A12	13	MUX14	1024	4	32Mx8
								2	32Mx16
BA1,BA0	2	A0−A9	10	A0−A13	14	MUX16	2048	2	64Mx16
BA1,BA0	2	A0−A9	10	A0−A14	15	MUX29	4096	1	128Mx32
BA1,BA0	2	A0−A9;A11	11	A0−A12	13	MUX15	2048	4	64Mx8

Table 12−51 is valid per chip-select. You can use different multiplexing schemes on CS0 and CS1. (As shown in the table, the total size in megabits corresponds to a single chip-select.)

Figure 12−31 through Figure 12−33 show all of the SDRAM MUX schemes, including the mapping of the 32-bit system address of the column and row addresses of the memory device.

Note:

The bank-select bit (BA1) does not exist for 2–bank devices.

Figure 12–31. SDRC SDR/DDR–SDRAM System Address Multiplexing Schemes (1 of 3)

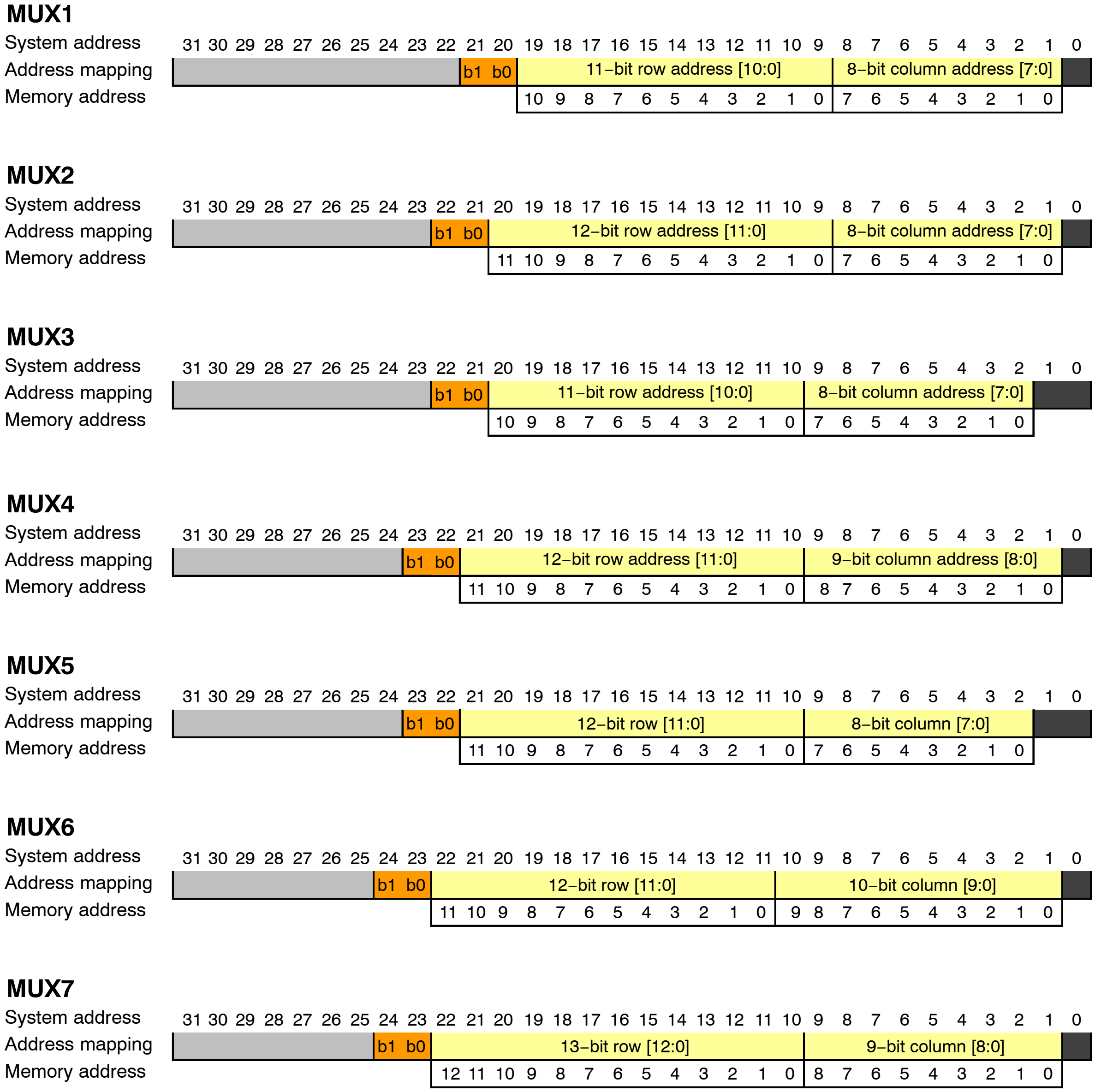


Figure 12−32. SDRC SDR/DDR−SDRAM System Address Multiplexing Schemes (2 of 3)

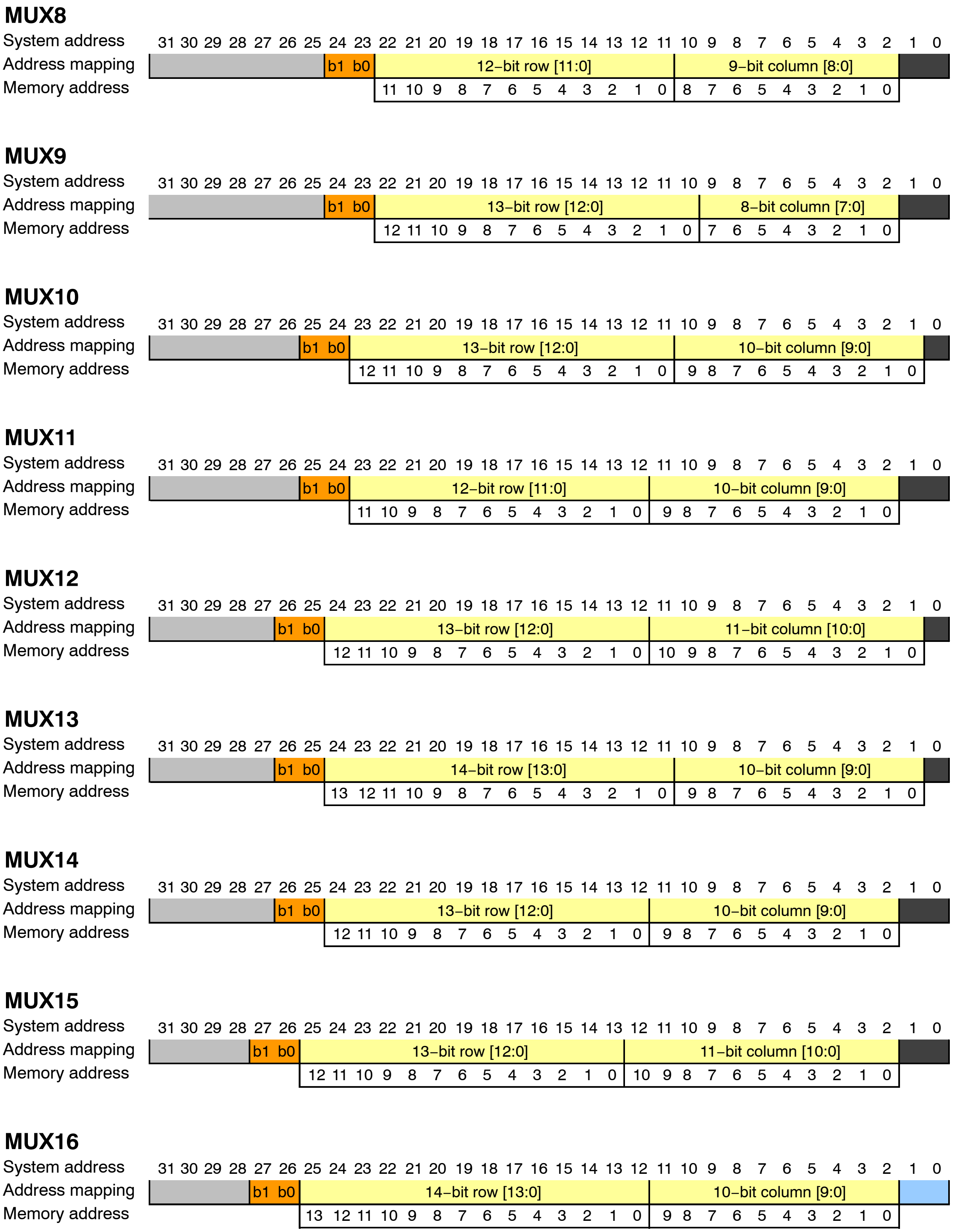
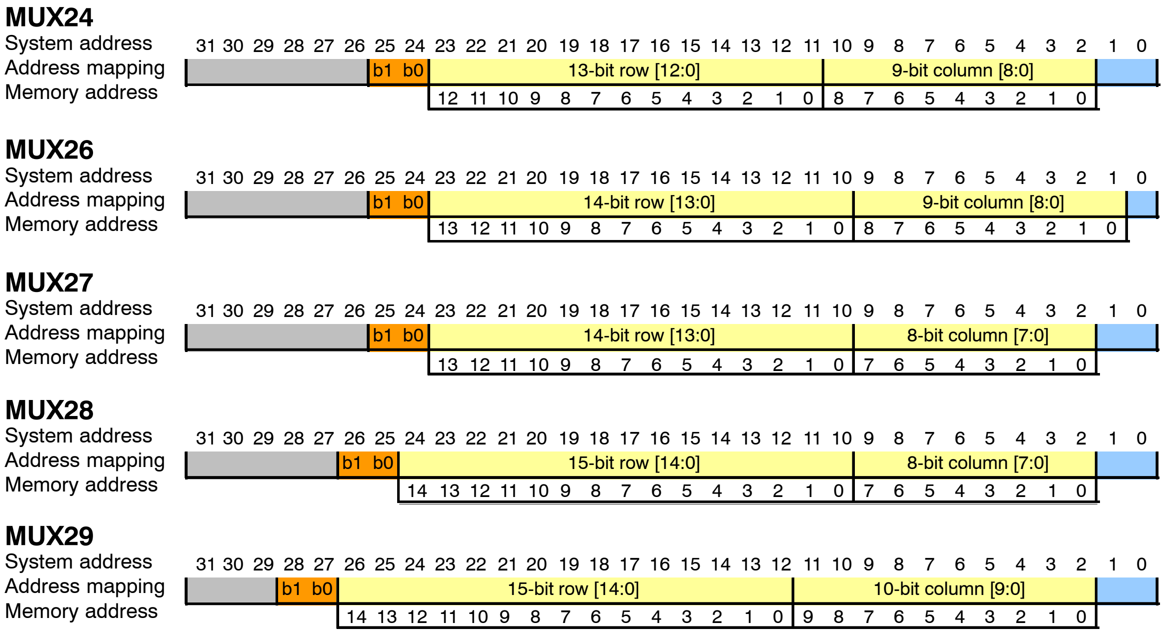


Figure 12–33. SDRC SDR/DDR–SDRAM System Address Multiplexing Schemes (3 of 3)

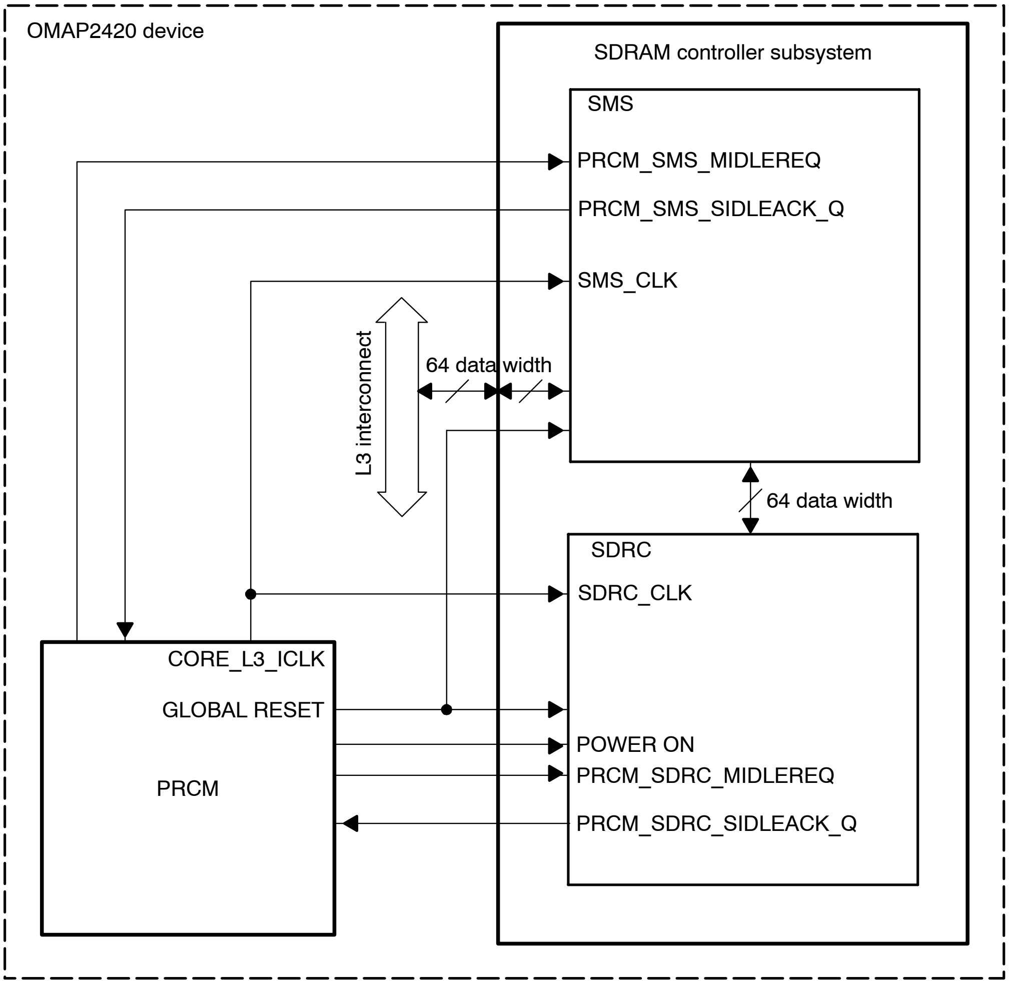


12.9 SDRAM Controller Subsystem Integration

12.9.1 Description

Figure 12−34 shows how the SMS and SDRC modules are integrated into OMAP2420 devices and how they interact with the PRCM module.

Figure 12−34. SDRC Integration to the OMAP2420 Device



12.9.2 Clocking, Reset, and Power Management Scheme

12.9.2.1 Clocking

SMS

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 SMS_CLK source is the PRCM CORE_L3_ICLK output; CORE_L3_ICLK belongs to the L3 interconnect clock domain.

SDRC

The SDRC_CLK comes from the PRCM module and runs at the L3 interconnect frequency. It is also used as a functional clock for the SDRC module.

The SDRC_CLK source is the PRCM CORE_L3_ICLK output; CORE_L3_ICLK belongs to the L3 interconnect clock domain.

At the PRCM level, the SDRC module can be configured to prevent/allow the entire domain to be shut down using the AUTO_SDRC bit in the CM_AUTOIDLE3_CORE register. If the bit is set to 0, SDRC does not permit the clock domain (CORE_L3_CLK) to shut down. If the bit is set to 1, SDRC can accept an idle request from the PRCM module when the software shuts off the domain clock. PRCM_SDRC_MIDLEREQ is then asserted and the SDRC sends back PRCM_SDRC_SIDLEACK_Q, according to its MIDLE_MODE setting.

Note:

The domain clock is shut down only if all other modules that receive the clock are able to accept this request.

For details, see Chapter 5, *Power, Reset, and Clock Management.*

12.9.2.2 Hardware Reset

Global reset of the SDRAM controller is done by activation of the CORE_RST_N signal in the CORE_RST domain (see Chapter 5, *Power, Reset, and Clock Management*).

One global reset signal, GLOBALRESET, is qualified by the signal 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 POWERON = 0 and GLOBALRESET is applied), all registers and state machines in the SDRC are asynchronously reset.
- ☐ On a warm reset (that is, any other system reset conditions under control of the chip top-level power manager; POWERON = 1 when GLOBALRESET is applied), the SDRC registers and FSM are not reset, but the external SDRAM memory can be placed in self-refresh mode.

12.9.2.3 Software Reset

The SMS and SDRC modules can be reset under software control using the software reset bits in the SMS_SYSCONFIG soft reset bit field (SMS_SYS_CONFIG[1]) and SDRC_SYSCONFIG soft reset bit field (SDRC_SYSCONFIG[1]) registers.

Software reset has the same effect as the hardware cold reset; that is, all registers and the FSM are reset immediately and unconditionally.

Power Domain

SDRAM controller power is supplied by the CORE power domain. For details, see Chapter 5, *Power, Reset, and Clock Management.*

Section 12.10.4, *SDRAM Controller*, describes power-saving features and different idle modes.

12.9.3 Pin List and Pad Multiplexing With Other Functions

In the Mode columns of Table 12−52, black cells indicate that the mode is not used. Gray cells indicate that the mode is used. White cells indicate an alternate function.

Table 12−52. SDRC Subsystem Pin List and Pad Multiplexing

SDRC Interface	Description	Dir	2420 Ball	Alternate Functions				
				Mode 0	Mode 1	Mode 2	Mode 3	Mode 4
SDRC.D0	SDRAM data bit 0	I/O	C2	I/O sharing with GPMC.A[11]				
SDRC.D1	SDRAM data bit 1	I/O	H7	I/O sharing with GPMC.A[12]				
SDRC.D2	SDRAM data bit 2	I/O	D4	I/O sharing with GPMC.A[13]				
SDRC.D3	SDRAM data bit 3	I/O	B2	I/O sharing with GPMC.A[14]				
SDRC.D4	SDRAM data bit 4	I/O	C4	I/O sharing with GPMC.A[15]				
SDRC.D5	SDRAM data bit 5	I/O	C3	I/O sharing with GPMC.A[16]				
SDRC.D6	SDRAM data bit 6	I/O	C5	I/O sharing with GPMC.A[17]				
SDRC.D7	SDRAM data bit 7	I/O	H8	I/O sharing with GPMC.A[18]				
SDRC.D8	SDRAM data bit 8	I/O	G11	I/O sharing with GPMC.A[19]				
SDRC.D9	SDRAM data bit 9	I/O	H11	I/O sharing with GPMC.A[20]				
SDRC.D10	SDRAM data bit 10	I/O	G10	I/O sharing with GPMC.A[21]				
SDRC.D11	SDRAM data bit 11	I/O	H10	I/O sharing with GPMC.A[22]				
SDRC.D12	SDRAM data bit 12	I/O	A11	I/O sharing with GPMC.A[23]				

Table 12−52. SDRC Subsystem Pin List and Pad Multiplexing (Continued)

SDRC Interface	**Description**	**Dir**	**2420 Ball**	**Alternate Functions**				
				Mode 0	**Mode 1**	**Mode 2**	**Mode 3**	**Mode 4**
SDRC.D13	SDRAM data bit 13	I/O	D6	I/O sharing with GPMC.A[24]				
SDRC.D14	SDRAM data bit 14	I/O	H9	I/O sharing with GPMC.A[25]				
SDRC.D15	SDRAM data bit 15	I/O	C6	I/O sharing with GPMC.A[26]				
SDRC.D16	SDRAM data bit 16	I/O	B15					
SDRC.D17	SDRAM data bit 17	I/O	H12					
SDRC.D18	SDRAM data bit 18	I/O	D15					
SDRC.D19	SDRAM data bit 19	I/O	C15					
SDRC.D20	SDRAM data bit 20	I/O	G13					
SDRC.D21	SDRAM data bit 21	I/O	B16					
SDRC.D22	SDRAM data bit 22	I/O	B17					
SDRC.D23	SDRAM data bit 23	I/O	C16					
SDRC.D24	SDRAM data bit 24	I/O	G14					
SDRC.D25	SDRAM data bit 25	I/O	D18					
SDRC.D26	SDRAM data bit 26	I/O	B19					
SDRC.D27	SDRAM data bit 27	I/O	B20					
SDRC.D28	SDRAM data bit 28	I/O	H13					
SDRC.D29	SDRAM data bit 29	I/O	C19					
SDRC.D30	SDRAM data bit 30	I/O	C18					
SDRC.D31	SDRAM data bit 31	I/O	C20					
SDRC.BA0	SDRAM bank select 0	O	C11					
SDRC.BA1	SDRAM bank select 1	O	D10					
SDRC.A0	SDRAM address bit 0	O	B9					
SDRC.A1	SDRAM address bit 1	O	D8					
SDRC.A2	SDRAM address bit 2	O	B7					
SDRC.A3	SDRAM address bit 3	O	D7					
SDRC.A4	SDRAM address bit 4	O	B6					
SDRC.A5	SDRAM address bit 5	O	C7					
SDRC.A6	SDRAM address bit 6	O	G8					
SDRC.A7	SDRAM address bit 7	O	C8					
SDRC.A8	SDRAM address bit 8	O	D9					
SDRC.A9	SDRAM address bit 9	O	B10					

Table 12−52. SDRC Subsystem Pin List and Pad Multiplexing (Continued)

SDRC Interface	Description	Dir	2420 Ball	Alternate Functions				
				Mode 0	**Mode 1**	**Mode 2**	**Mode 3**	**Mode 4**
SDRC.A10	SDRAM address bit 10	O	C9					
SDRC.A11	SDRAM address bit 11	O	C10					
SDRC.A12	SDRAM address bit 12	O	D11				gpio_2	
SDRC.A13	SDRAM address bit 13	O	B4				gpio_1	
SDRC.A14	SDRAM address bit 14	O	B3				gpio_0	
SDRC.nCS0	Chip-select 0	O	D12					
SDRC.nCS1	Chip-select 1	O	C12				gpio_37	
SDRC.CLK	Clock	I/O	C14					
SDRC.nCLK	Clock invert	O	B14					
SDRC.CKE0	Clock enable0	O	D13					
SDRC.CKE1	Clock enable1	O	B13				gpio_38	
SDRC.NRAS	SDRAM row access	O	B12					
SDRC.NCAS	SDRAM column address strobe	O	D14					
SDRC.NWE	SDRAM write enable	O	C13					
SDRC.DM0	Data mask 0	O	B5					
SDRC.DM1	Data mask 1	O	G12					
SDRC.DM2	Data mask 2	O	C17					
SDRC.DM3	Data mask 3	O	D17					
SDRC.DQS0	Data strobe 0	I/O	D5					
SDRC.DQS1	Data strobe 1	I/O	G9					
SDRC.DQS2	Data strobe 2	I/O	D16					
SDRC.DQS3	Data strobe 3	I/O	B18					

12.9.4 SDRC and GPMC I/O Sharing

The OMAP2420 external memory interface comprises two separate controllers:

- ❏ One 8-/16-bit general-purpose memory controller (GPMC)
- ❏ One 16-/32-bit SDRAM controller (SDRC)

According to attached memories and application pin count requirements, static multiplexing options exist between the GPMC and SRDC, depending on the interface configurations required. If a configuration does not require all GPMC/SDRC pins, GPIOs or alternate function signals can also be multiplexed (see Table 12−4 and Table 12−52).

When SDRC.D[15:0] I/Os are used by one module, they are off-limits to the other until the next IC reset (power-on reset).

12.9.4.1 GPMC Address and Data Bus Usage

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).

- When GPMC_CONFIG.LIMITEDADDRESS = 1, nonmultiplexed address/data NOR devices do not use the shared SDRC/GPMC I/O SDRC.D[15:0]. Only GPMC.A[10:0] address lines are used. This limits the memory support to only 2K-byte addressable memories.
- Multiplexed address and data NOR devices do not use shared SDRC I/O: SDRC.D[15:0]. The address is multiplexed on the data bus.
- 8-bit-wide NOR devices do not use GPMC I/O: GPMC.D[15:8].
- 16-bit-wide NAND devices do not use the shared SDRC/GPMC I/O: SDRC.D[15:0] and GPMC I/O: GPMC.A [10:1].
- 8-bit-wide NAND devices do not use the shared SDRC I/O: SDRC.D[15:0], GPMC I/O: GPMC.A[10:1], or GPMC I/O: GPMC.D[15:8].

> CAUTION
>
> **From the OMAP2420 power-on reset (IC reset), address and data lines are set to the default logical 1 value. This default value is used as a transparent value that allows OMAP device pin sharing (transparent multiplexing) between the GPMC and alternate I/O functions.**
>
> **Alternate functions are selected in place of the GPMC function if the GPMC function is not used. For example, alternate I/O functions can use [D15−D8] if GPMC use is restricted to 8-bit-wide device interfacing.**
>
> **Never configure a chip-select or initiate an access with an incompatible sharing configuration that can corrupt the transparent reset value.**

> CAUTION
>
> **For a nonmultiplexed chip-select configuration, you can force the transparent sharing of A[26:A11] address lines by setting the LIMITEDADDRESS bit control (GPMC_CONFIG register). In this case, only the GPMC.A[10:1] I/O address lines are usable so that nonmultiplexed devices can be supported through a limited 2K-byte address range.**
>
> **Set the LIMITEDADDRESS bit control before attempting an access to a chip-select configured with a nonmultiplexed protocol. After the first nonmultiplexed access is performed without this bit setup, SDRC[15:0] I/O are lost for SDRC use until the next IC reset.**

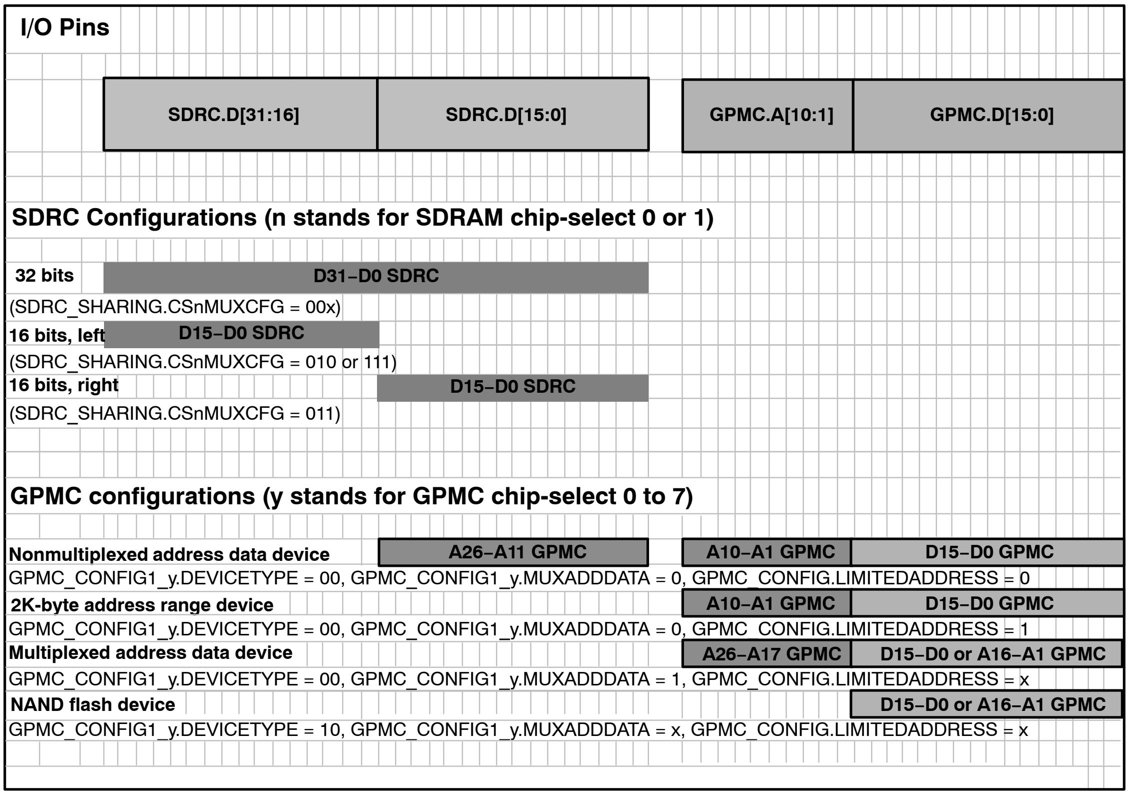
12.9.4.2 SDRC and GPMC I/O Configuration Solutions

Figure 12−35 shows different solutions for connecting SDRAM devices and NOR flash devices, asynchronous/synchronous devices and NAND flash devices.

All these memories are connected through several chip-selects (noted *n* and *y)* and share four I/O groups:

- SDRC.D[31:16]
  - Upper 16 bits of a 32-bit SDRAM device (data [31:16])
  - 16-bit SDRAM device (data [15:0])
- SDRC.D[15:0]
  - Lower 16 bits of a 32-bit SDRAM device (data [15:0])
  - 16-bit SDRAM device (data [15:0])
  - Nonmultiplexed address/data device (address [26:11]) when GPMC_CONFIG.LIMITEDADDRESS is not set
- GPMC.A[10:1]
  - Nonmultiplexed address/data device (address [10:1])
  - Multiplexed address/data device (address [26:17])
- GPMC.D[15:0]
  - Nonmultiplexed address/data device (data [15:0])
  - Multiplexed address/data device (address [16:1] and data [15:0])

Figure 12–35. SDRC/GPMC I/O Pin Configuration Options in Default Pinout Mode 0


I/O Pins
SDRC.D[31:16]
SDRC.D[15:0]
GPMC.A[10:1]
GPMC.D[15:0]
SDRC Configurations (n stands for SDRAM chip-select 0 or 1)
32 bits
D31–D0 SDRC
(SDRC_SHARING.CSnMUXCFG = 00x)
16 bits, left
D15–D0 SDRC
(SDRC_SHARING.CSnMUXCFG = 010 or 111)
16 bits, right
D15–D0 SDRC
(SDRC_SHARING.CSnMUXCFG = 011)
GPMC configurations (y stands for GPMC chip-select 0 to 7)
Nonmultiplexed address data device
A26–A11 GPMC
A10–A1 GPMC
D15–D0 GPMC
GPMC_CONFIG1_y.DEVICETYPE = 00, GPMC_CONFIG1_y.MUXADDDATA = 0, GPMC_CONFIG.LIMITEDADDRESS = 0
2K-byte address range device
A10–A1 GPMC
D15–D0 GPMC
GPMC_CONFIG1_y.DEVICETYPE = 00, GPMC_CONFIG1_y.MUXADDDATA = 0, GPMC_CONFIG.LIMITEDADDRESS = 1
Multiplexed address data device
A26–A17 GPMC
D15–D0 or A16–A1 GPMC
GPMC_CONFIG1_y.DEVICETYPE = 00, GPMC_CONFIG1_y.MUXADDDATA = 1, GPMC_CONFIG.LIMITEDADDRESS = x
NAND flash device
D15–D0 or A16–A1 GPMC
GPMC_CONFIG1_y.DEVICETYPE = 10, GPMC_CONFIG1_y.MUXADDDATA = x, GPMC_CONFIG.LIMITEDADDRESS = x

CAUTION

The OMAP2420 device can support a 32-bit-wide external DDR/SDRAM interface only when GPMC use is restricted to address and data-multiplexed protocol (random synchronous or asynchronous memory) and NAND device type.

These protocols leave the A[26:A11] address lines (SDRC.D[15:0] I/O) free for transparent SDRC [D15−D0] pin sharing.

Never configure a chip-select or initiate an access with an incompatible sharing configuration that could corrupt the transparent reset value.

12.9.4.3 SDRC and GPMC I/O Configuration Settings (in Default Pinout Mode 0)

SDRC I/O Configuration Setting (in Default Pinout Mode 0)

External 32- or 16-bit data SDRAM is selected by SDRC register fields SDRC_SHARING[14:12] (CS1MUXCFG) and SDRC register fields SDRC_SHARING[11:9] (CS0MUXCFG) with the following options:

- 32 bits = 00x
- 16 bits on SDRC.D[31:16] = 010 or 011
- 16 bits on SDRC.D[15:0] = 011

The reset value is 16 bits on SDRC.D[31:16] to authorize external boot through the GPMC on nonmultiplexed address/data device.

GPMC I/O Configuration Setting (in Default Pinout Mode 0)

In this section, *y* stands for chip-select value (CS*y).*

The nonmultiplexed address/data device, if not limited to 2K-byte address range, is selected with the following register programming:

- SDRC register SDRC_SHARING[11:9] (CS0MUXCFG) = 010
- SDRC register SDRC_SHARING[14:12] (CS1MUXCFG) = 010
- GPMC register GPMC_CONFIG1_y[11:10] (CSy DEVICETYPE) = 00
- GPMC_CONFIG1_y[9] (CSy MUXADDDATA) = 0
- GPMC_CONFIG[1] (LIMITEDADDRESS) = 0

CAUTION

GPMC_CONFIG1_y.DEVICETYPE = 0, GPMC_CONFIG1_y.MUXADDATA = 0 with 32-bit external DDR, and GPMC_CONFIG.LIMITEDADDRESS = 0 are not supported.

The nonmultiplexed address/data device, if limited to a 2K-byte address range, is selected with the following register programming:

- GPMC register GPMC_CONFIG1_y[11:10] (CSy DEVICETYPE) = 00
- GPMC_CONFIG1_y[9] (CSy MUXADDDATA) = 0
- GPMC_CONFIG[1] (LIMITEDADDRESS) = 1

Note:

The LIMITEDADDRESS field applies only to nonmultiplexed address/data devices and has no effect on other device types (multiplexed address/data, NAND).

The multiplexed address/data device is selected with the following register programming:

- GPMC register GPMC_CONFIG1_y[11:10] (CSy DEVICETYPE) = 00
- GPMC_CONFIG1_y[9] (CSy MUXADDDATA) = 1

The NAND device is selected with the following register programming:

- GPMC register GPMC_CONFIG1_y[11:10] (CSy DEVICETYPE) = 10

Note:

The CSy MUXADDDATA field applies only to DEVICETYPE = 00 and has no effect on NAND-like devices.

12.9.4.4 GPMC CS0 Default Configuration at IC Reset

To ensure correct external boot with a GPMC access from IC reset time on CS0, three external pins (SYS.BOOT[2:0]) are sampled to configure 3 GPMC register field reset values of the GPMC_CONFIG1_0 register:

- GPMC_CONFIG1_0[22]: WAITREADMONITORING bit reset value
- GPMC_CONFIG1_0[13:12]: DEVICESIZE bits reset value
- GPMC_CONFIG1_0[9]: MUXADDDATA bit reset value

The SYS.BOOT[3] external pin determines internal or external boot.

Table 12–53. GPMC CS0 Default Configuration at IC Reset

SYS.BOOT[3:0]	Internal/ BOOT CODE	Wait Monitoring	Memory Interface Type	GPMC_CONFIG1_0.WAITREADMONITORING	GPMC_CONFIG1_0. DEVICESIZE	GPMC_CONFIG1_0. MUXADDDATA
				Reset Value	Reset Value	Reset Value
0000	Reserved					
0001	Reserved					
0010	Reserved					
0011	Reserved					
1000	Internal	Active low	Non-multiplexed 16-bit	0x1	0x1	0x0
1001	Internal	Don't care	Non-multiplexed 16-bit	0x0	0x1	0x0
1010	Internal	Active low	Address/Data multiplexed 16-bit	0x1	0x1	0x1
1011	Internal	Don't care	Address/data multiplexed 16-bit	0x0	0x1	0x1
1100	Internal	–	NAND 8-bit[1]	0x0	0x0	0x0
1101	Internal	–	NAND 16-bit[1]	0x0	0x1	0x0

1) DEVICETYPE is always 0x0 at reset and is programmed by internal ROM code to NAND (0x2), if necessary.

CAUTION

With internal boot code, the entire CS0 configuration can be modified before the first CS0 access. This modification into internal boot code is necessary for the following device types:

- **NAND devices attached to CS0**
- **Nonmultiplexed 2K-byte address range device used to free SDRC[15:0] I/O and attached to CS0**

12.9.5 Register Summary

Table 12−54 and Table 12−55 summarize, respectively, the control registers for SMS and SDRC modules, listing their names, types, widths, and physical addresses.

Table 12−54. SMS Register Summary

Register Name	Type	Register Width (Bits)	Physical Address
SMS_REVISION	R	32	0x6800 8000
SMS_SYSCONFIG	RW	32	0x6800 8010
SMS_SYSSTATUS	R	32	0x6800 8014
Reserved		32	0x6800 8040−0x6800 8070
Reserved		32	0x6800 8044−0x6800 8074
Reserved		32	0x6800 8048−0x6800 8078
Reserved		32	0x6800 8080−0x6800 80B0
Reserved		32	0x6800 8084−0x6800 80B4
Reserved		32	0x6800 80C0
SMS_CLASS_ARBITER0	RW	32	0x6800 80D0
SMS_CLASS_ARBITER1	RW	32	0x6800 80D4
SMS_CLASS_ARBITER2	RW	32	0x6800 80D8
SMS_INTERCLASS_ARBITER	RW	32	0x6800 80E0
SMS_CLASS_ROTATION0 – SMS_CLASS_ROTATION2	RW	32	0x6800 80E4−0x6800 80EC
SMS_ERR_ADDR	R	32	0x6800 80F0
SMS_ERR_TYPE	RW	32	0x6800 80F4
SMS_POW_CTRL	RW	32	0x6800 80F8
SMS_ROT_CONTROL0 – SMS_ROT_CONTROL3	RW	32	0x6800 8100−0x6800 8130
SMS_ROT_SIZE__0_3	RW	32	0x6800 8104−0x6800 8134
SMS_ROT_PHYSICAL_BA__0_3	RW	32	0x6800 8108−0x6800 8138

Table 12−55. SDRC Register Summary

Register Name	Type	Register Width (Bits)	Physical Address
SDRC_REVISION	R	32	0x6800 9000
SDRC_SYSCONFIG	RW	32	0x6800 9010
SDRC_SYSSTATUS	R	32	0x6800 9014
SDRC_CS_CFG	RW	32	0x6800 9040
SDRC_SHARING	RW	32	0x6800 9044
SDRC_ERR_ADDR	R	32	0x6800 9048
SDRC_ERR_TYPE	RW	32	0x6800 904C
SDRC_DLLA_CTRL	RW	32	0x6800 9060
SDRC_DLLA_STATUS	R	32	0x6800 9064
SDRC_DLLB_CTRL	RW	32	0x6800 9068
SDRC_DLLB_STATUS	R	32	0x6800 906C
SDRC_POWER	RW	32	0x6800 9070
SDRC_MCFG_0	RW	32	0x6800 9080
SDRC_MR_0	RW	32	0x6800 9084
SDRC_EMR1_0	RW	32	0x6800 9088
SDRC_EMR2_0	RW	32	0x6800 908C
SDRC_EMR3_0	RW	32	0x6800 9090
SDRC_DCDL1_CTRL	RW	32	0x6800 9094
SDRC_DCDL2_CTRL	RW	32	0x6800 9098
SDRC_ACTIM_CTRLA_0	RW	32	0x6800 909C
SDRC_ACTIM_CTRLB_0	RW	32	0x6800 90A0
SDRC_RFR_CTRL_0	RW	32	0x6800 90A4
SDRC_MANUAL_0	RW	32	0x6800 90A8
SDRC_MCFG_1	RW	32	0x6800 90B0
SDRC_MR_1	RW	32	0x6800 90B4
SDRC_EMR1_1	RW	32	0x6800 90B8
SDRC_EMR2_1	RW	32	0x6800 90BC
SDRC_EMR3_1	RW	32	0x6800 90C0
SDRC_ACTIM_CTRLA_1	RW	32	0x6800 90C4
SDRC_ACTIM_CTRLB_1	RW	32	0x6800 90C8
SDRC_RFR_CTRL_1	RW	32	0x6800 90D4
SDRC_MANUAL_1	RW	32	0x6800 90D8

12.10 SDRAM Controller Subsystem Functional Description

The SDRAM memory scheduler (SMS) optimizes SDRAM memory use to provide the following advantages:

❑ Guaranteed QoS level required by each initiator in the system

❑ Security firewall that controls access protection to the memory

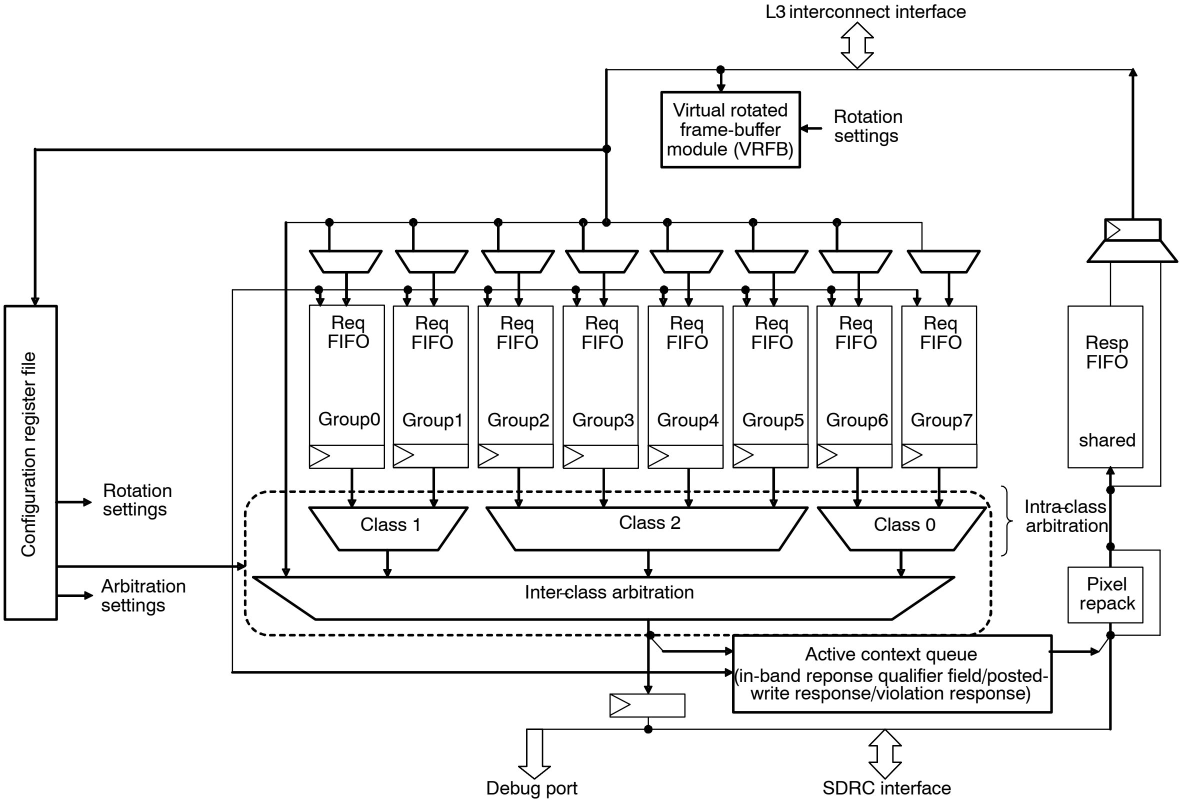
❑ Virtual rotated frame-buffer module that minimizes the SDRAM page-miss penalty when accessing rotated (nonsequentially addressed) lines in a graphic frame buffer

12.10.1 SDRAM Memory Scheduler

The SMS optimizes SDRAM memory use to provide the QoS level required by each initiator in the system. In addition, a virtual rotated frame-buffer module minimizes the SDRAM page-miss penalty when accessing rotated (addressed nonsequentially) lines in a graphics frame buffer.

Figure 12−36 is the top-level diagram of the SMS.

Figure 12−36. SMS Top-Level Diagram



12.10.1.1 Memory Access Scheduling

When a memory transaction request arrives in the memory-access scheduler after being processed by the virtual rotated frame-buffer module and security firewall, the request is sent to one of eight FIFO queues.

The allocation to a particular queue is fixed and depends on the source of the request. To prioritize concurrent transactions and optimize memory use, each FIFO queue (and hence the memory-access request source) is categorized into one of three arbitration classes:

- Class 0 (highest priority): For bandwidth-sensitive devices with severe real-time constraints
- Class 1: For latency-sensitive devices where system performance degrades severely when the average memory-access latency increases
- Class 2: For all other devices, possibly with high bandwidth requirements but without being highly latency-sensitive

Table 12−56 shows the mapping of FIFO queues and memory-access request sources to arbitration classes.

Table 12–56. Arbitration Class Allocation

Class	FIFO Queue	Source Device
0	6	Camera subsystem DMA
	7	Display subsystem
1	0	MPU subsystem CPU (instruction and data access)
	1	DSP subsystem CPU (instruction and data access)
2	2	sDMA (read) DSP subsystem DMA
	3	Reserved
	4	Reserved
	5	sDMA (write) USB

A two-level arbitration scheme determines from which FIFO queue the next memory transaction is granted.

Note:

In the register description, a group represents a FIFO queue of requests from the initiator.

12.10.1.2 Intra-Class Arbitration

The first level of arbitration is intraclass arbitration. One FIFO queue is selected from each class using a least-recently-used algorithm. However, one FIFO queue in a class can be specified as high priority using the SMS_CLASS_ARBITER*c* register (where *c* is the class number: 0, 1, or 2).

A burst transaction request can be submitted to the arbitration stage when either of the following occurs:

- ❑ The first request of the burst is received.
- ❑ A complete burst is buffered in the FIFO queue.

The request can be configured on a per-FIFO queue basis using the SMS_CLASS_ARBITER*c* register.

It is also possible to use the SMS_CLASS_ARBITER*c* register to specify how many consecutive transactions (1, 2, 3, or 4) a FIFO queue is granted at a time. This can result in better memory performance because of the high probability that two consecutive transactions in a FIFO queue will access consecutive memory addresses in the same SDRAM page.

This setting is ignored for consecutive transactions split by the virtual rotated frame-buffer module. Instead, the SMS_CLASS_ROTATION*c* register specifies for the class the number of consecutive transactions granted at a time to a FIFO queue.

Note:

The granting of multiple consecutive transactions is propagated to the second level of arbitration.

12.10.1.3 Interclass Arbitration

The second level of arbitration, interclass arbitration, decides from which of the three classes the next transaction should be granted. After class 0, which always has the highest priority, the relative priority of class 1 to class 2 is determined through programmable values in the SMS_INTERCLASS_ARBITER register, so that for a certain number of transactions, class 1 has priority and for a certain number of transactions, class 2 has priority.

Finally, the selected transaction request is passed to the SDRC module.

Note:

A burst access is not split, even if the burst continues over the priority window. The priority window is ignored until the end of burst transmission to the SDRC.

There is one exception when the burst is split: when the FIFO (8 words of 64 bits) cannot provide consecutive accesses.

If this occurs once, one idle is inserted by SMS.

If this occurs two consecutive times, another FIFO queue is selected and the burst is completed later.

12.10.1.4 Firewall

Access permissions for initiators in the system can be defined for up to four memory regions. The start and end of the memory regions are specified with 64K-byte granularity using the SMS_RG_START*r* and SMS_RG_END*r* registers, respectively (where *r* is the number of the region: 0 to 3). Memory outside these regions is treated as a fifth region without security protection that initiators can access unconditionally.

Each memory region’s protection attributes, write permission, and read permission can be configured using the SMS_RG_ATT*r*, SMS_RG_WRPERM*r* and SMS_RG_RDPERM*r* registers, respectively.

Attempted illegal access (for example, a write access to a write-protected region) is indicated as a security violation to the security module. The address where the security violation occurred and the type of violation can be read from the SMS_ERR_ADDR and SMS_ERR_TYPE registers, respectively.

12.10.1.5 Virtual Rotated Frame-Buffer Module

The virtual rotated frame-buffer (VRFB) module minimizes the SDRAM page-crossing penalty when the image in the frame buffer is accessed in different rotation view angles (0 degree view, 90 degree view, 180 degree view, and 270 degree view).

The VRFB allows this functionality by modifying incoming requests and virtual addresses generated by initiators connected to the L3 interconnect (LCD controller, DSP, MPU, etc.).

If the address of the request targets the virtual rotated frame-buffer address space, the request is sent to the rotation engine. The SMS translates the virtual address into physical SDRAM addresses and reinserts a request or multiple requests in the SMS request path to the SDRC.

The 256M-byte VRFB virtual address space is split into four independently configurable contexts, each with four views that can handle a frame-buffer rotated by 0 degrees , 90 degrees, 180 degrees, or 270 degrees, as shown in Table 12−57.

Table 12−57. Virtual Rotated Frame-Buffer Module Address Space

Context	Offset Address Range	Rotation
0	0x70000000 − 0x70FFFFFF	0°
	0x71000000 − 0x71FFFFFF	90°
	0x72000000 − 0x72FFFFFF	180°
	0x73000000 − 0x73FFFFFF	270°
1	0x74000000 − 0x74FFFFFF	0°
	0x75000000 − 0x75FFFFFF	90°
	0x76000000 − 0x76FFFFFF	180°
	0x77000000 − 0x77FFFFFF	270°
2	0x78000000 − 0x78FFFFFF	0°
	0x79000000 − 0x79FFFFFF	90°
	0x7A000000 − 0x7AFFFFFF	180°
	0x7B000000 − 0x7BFFFFFF	270°
3	0x7C00000 − 0x7CFFFFFF	0°
	0x7D000000–0x7DFFFFFF	90°
	0x7E000000–0x7EFFFFFF	180°
	0x7F000000–0x7FFFFFFF	270°

For each of the four contexts, three independent register sets configure the buffer physical base address, image height and width, and pixel size:

❑ SMS_ROT_CONTROLx
❑ SMS_ROT_SIZEx
❑ SMS_ROT_PHYSICAL_BAx

Note:

In the register set names, *x* represents the context number from 0 to 3.

The memory arrangement for the pixels of a frame buffer accessed through the rotation engine is page-based. A page is a rectangular array of pixels, the size of which must match the page size of the SDRAM component attached to the controller.

Page size is defined using the SMS_ROT_CONTROLx[PH] and SMS_ROT_CONTROLx[PW] register bit fields.

Image height and width (SMS_ROTx_SIZE [IMAGEHEIGHT] and SMS_ROTx_SIZE [IMAGEWIDTH]) in bytes must be multiples of page height and width, respectively.

Section 12.11.1.1, *Virtual Rotated Frame-Buffer Context Configuration*, shows a concrete example of configuration.

12.10.2 Module Power Saving

The SMS_SYSCONFIG register enables and disables the SMS module interconnect clock auto-idle power-saving mode. When this mode is enabled, all FIFO queues are empty, and there are no outstanding transactions in the SDRC module, the interconnect clock is disabled internally to the module (after a delay programmable using the SMS_POW_CTRL register), reducing power consumption.

When there is new activity on the interconnect interface, the interconnect clock is restarted without a latency penalty. After reset, this mode is disabled by default. Enabling this mode is recommended, to reduce power consumption.

12.10.3 System Power Management

The SMS can be configured using the SMS_SYSCONFIG register to be in one of three idle modes:

- ❑ No-idle mode (SIDLEMODE: 0x1)

  The module never goes into idle state.

- ❑ Force-idle mode (SIDLEMODE: 0x0)

  The module goes into idle state immediately after receiving the request from the PRCM module.

- ❑ Smart-idle mode (SIDLEMODE: 0x2)

  The module goes into idle state after receiving the request from the PRCM module and after all asserted output interrupts are acknowledged.

12.10.4 SDRC

The SDRC module provides two configurable memory areas. Each supports low-power SDR SDRAM and mobile DDR SDRAM from 16M bits to 2G bits.

Flexible row/column addressing schemes are possible with 2-bank support for 16M-bit and 32M-bit memories and 4-bank support for 64M-bit, 128M-bit, 256M-bit, 512M-bit, 1G-bit, and 2G-bit memories.

12.10.4.1 Bank Tracking

The SDRC contains hardware for tracking open pages on a per-bank basis. Four open pages are tracked per chip-select, or eight open pages, total.

To efficiently pipeline, the SDRC includes a request look-ahead FIFO that analyzes interconnect requests with respect to the status of the target bank. Three statuses are possible:

- Bank open on another row
- Bank idle
- Bank open on the same row

The SDRC state-machine generates the appropriate sequence of memory commands. All precharge and active commands are hidden as much as possible to optimize memory bandwidth use.

The look-ahead FIFO depth is 5x64-bit requests.

12.10.4.2 Refresh Management

Refresh management can be subdivided as follows:

- Self-refresh management
- Autorefresh management

Self-refresh is executed to place an attached SDRAM into an autonomous refresh mode, typically when the OMAP2420 enters an idle mode and switches off the SDRAM clock.

In self-refresh, the SDRAM supplies the row address generation required to refresh and retain data in the absence of external clocking.

Self-refresh can be entered using any of the following methods:

- Manual software command (entry to self-refresh mode)
- Automatically on a warm reset event (if the SDRC is programmed to enter self-refresh on warm reset)
- Automatically on an idle request sent by the system power manager (if the SDRC is programmed to enter self-refresh on idle request)
- Automatically after a (programmable) period of inactivity on the interconnect interface (if enabled through the CLKCTRL (0x2) bit in the SDRC_POWER register)

Self-refresh can be exited as follows:

- Manual software command (exit from self-refresh mode; see Section 12.11.4, *Manual Software Commands*).
- Automatically after receiving a read or write transaction to access memory

These self-refresh controls are shared by both chip-selects (but not by manual command registers that provide per-chip-select software control).

When autorefresh is enabled, a programmable hardware counter in the SDRC generates a periodic event that triggers either a single refresh or a burst of consecutive refresh commands. This is the standard refresh mode used when the system is active, while the running applications regularly access the SDRAM.

An autorefresh command can also be applied using SDRC_MANUAL_x (see Section 12.11.4, *Manual Software Commands*). This method can be used to generate a device-specific initialization sequence after power-up or after the memory device exits from a deep power-down mode state.

The autorefresh request period is a user-controlled parameter programmed to meet the specification of the memory device. Each time an autorefresh request is issued, the SDRC can service the autorefresh using any of the following commands:

- Single autorefresh command
- Burst of four autorefresh commands
- Burst of eight autorefresh commands

When a burst of four or eight is selected, the programmed period value is automatically scaled in hardware by 4 or 8. Consequently, the value to be programmed does not depend on the selected burst length.

12.10.4.3 System Power Management

Unlike the SMS, the SDRC can be configured only in smart-idle mode through the SDRC_SYSCONFIG register (SIDLEMODE: 0x2). The module goes to idle state after receiving the request from the PRCM module after all asserted output interrupts are acknowledged.

The SDRC acknowledge is conditioned by the internal activity of the SDRC.

12.10.4.4 Power-Saving Features

The SDRC has three operating modes:

- Page opening/closure policy
- Low-power dynamic operation modes
- Static low-power modes

Page Opening/Closure Policy

The OMAP2420 supports one page policy. The SDRC_POWER register must be set to 1.

The SDRC tracks open pages and determines whether the current access is to an open or closed page. If the accessed page is open, the SDRC executes the access immediately. If the accessed page is closed, the SDRC:

- Closes the current open page by issuing a PRECHARGE command to that bank. The CMDCODE bit field of SDRC_MANUAL_x is set to 0x1.
- Opens the accessed page by issuing an ACTIVE command to that bank
- Executes the access by issuing a READ or WRITE command

Up to four pages can be open simultaneously (one per bank). Pages remain open until one of the following events occurs:

- New read or write request to another page in the same bank
- Autorefresh request (a PRECHARGE_ALL command is issued first)
- Self-refresh entry request (a PRECHARGE_ALL command is issued first)

Dynamic Low-Power Operating Modes

The dynamic low-power operating modes of the SDRC are designed to:

- Control the external SDRAM clock(s)
- Control internal clock-gating of the SDRC when the interconnect interface is idle
- Control the self-refresh functionality

The PWDENA field is used to activate the power-down mode of the external memory by pulling the relevant CKE pin at a low level each time the memory interface is not used.

When the PWDENA field is enabled, the SDRC provides a free-running clock to external memory.

CKE is dynamically controlled based on the current memory command. There is a one-SDRC-clock cycle latency penalty when exiting from power-down. This mechanism is independent of the CLKCTRL field.

The SDRC has three modes of autoclock behavior when the interconnect interface is idle (that is, no outstanding active transactions). These modes are controlled using the CLKCTRL bit field and the AUTOCOUNT field of the SDRC_POWER register:

- Mode 0: The autoclock gating feature is disabled. No internal clock gating is performed in the SDRC in response to the detection of an idle state on the interconnect interface.
- Mode 1: A 16-bit counter starts decrementing when an interconnect idle condition is detected. The counter start value is loaded from the AUTOCOUNT 16-bit bit field. When this counter times out, internal clock gating within the SDRC is enabled.

  If an Interconnect active command is received before the counter times out, or if an internal autorefresh request is issued, the procedure is aborted, the counter stops decrementing, and the request is serviced immediately.
- Mode 2: This mode is similar to mode 1, but before internal clock gating, the SDRC places the SDRAM into self-refresh mode and turns off the external SDRAM clock. This is the lowest power mode.

To achieve maximum power saving, TI recommends use of both PWDENA and CLKCTRL-related features

All power-saving settings are common to the two chip-selects. But when two-independent chip-selects are used, only the accessed chip-select exits self-refresh.

If the SRFRONIDLEREQ field is enabled, the SDRC enters self-refresh mode on a hardware idle request from the PRCM module. The memory clock is automatically switched off, after which the SDRC sends an acknowledge back to the PRCM module.

In this situation, it is possible for the power manager to cut off the SDRC clock. Therefore, if connected to a DDR memory, the SDRC waits 1024 cycles (DLL relocking maximum time) before accessing the memory again when the system exits the idle state. This mechanism is independent of the CLKCTRL field.

CAUTION

The power-down feature (SDRC_POWER[2] PWDENA bit) must be disabled just before the deep power-down exit mode and reenabled after the next memory initialization.

Static Low Power Operating Modes

Software-driven controls for low-power operation modes include the capability of using SDRC_MANUAL_x to put the memory in self-refresh or deep power-down mode.

- In self-refresh mode, each chip-select can be controlled independently. If both chip-selects are in self-refresh, the external SDRAM clock can be switched off by setting the EXTCLKDIS control bit to 1 in the SDRC_ POWER register.

Another manual command must be used for the memory to exit self-refresh.

- ☐ In deep power-down mode, each chip-select is controlled independently. The external SDRAM clock can be switched off if both chip-selects are in self-refresh or deep power-down mode by setting the EXTCLKDIS control bit to 1 in the SDRC_POWER register.

Another manual command must be used for the memory to exit self-refresh or deep power-down mode.

After a memory exits from self-refresh or deep power-down mode (all data are lost), a full-initialization sequence must be sent to the device before it can be used.

12.10.4.5 SDRC Power Off Mode

In some applications, the SDRC power domain can be in off mode while the external memory is in self-refresh mode.

When the SDRC is in off mode, it is critical to prevent an unwanted exit from self-refresh when the context is restored. In off mode, SDRC outputs that control the SDRAM are set to maintain self-refresh (CKE low).

When a reset occurs, the default reset state of the SDRC is power-down enable (PDE), which means that the CKE pin is held low.

When exiting off mode:

- ☐ Power is restored to the SDRC.
- ☐ Software restores all registers except MRS and EMRS.
- ☐ Exit from self-refresh mode is achieved through the CMDCODE field of the SDRC_MANUAL_x register. Because exit to the self-refresh field is unconditional (the current state of the SDRC state-machine is not considered), ensure that autorefresh is disabled.
- ☐ MRS and EMRS can be restored to reflect the memory registers.
- ☐ The SDRAM is ready to use.

After context is successfully restored, the software can reinitialize the SDRAM or return the SDRAM to self-refresh. When the device successfully exits self-refresh, autorefresh is enabled again.

12.10.4.6 Digitally Controlled Delay Line

The SDRC includes digitally controlled delay technology for interfacing high-speed mobile DDR memory components.

A DLL is a calibration module used to track voltage and temperature variations dynamically, as well as to compensate for silicon process dispersion (process voltage temperature (PVT) tracking).

DLLs control several digitally controlled delay line (DCDL) companion modules by providing an 8-bit value, continuously updated so that it always reflects a nominal 72- or 90-degree delay with respect to the memory interface clock frequency, in all PVT conditions.

DLL/DCDL Architecture

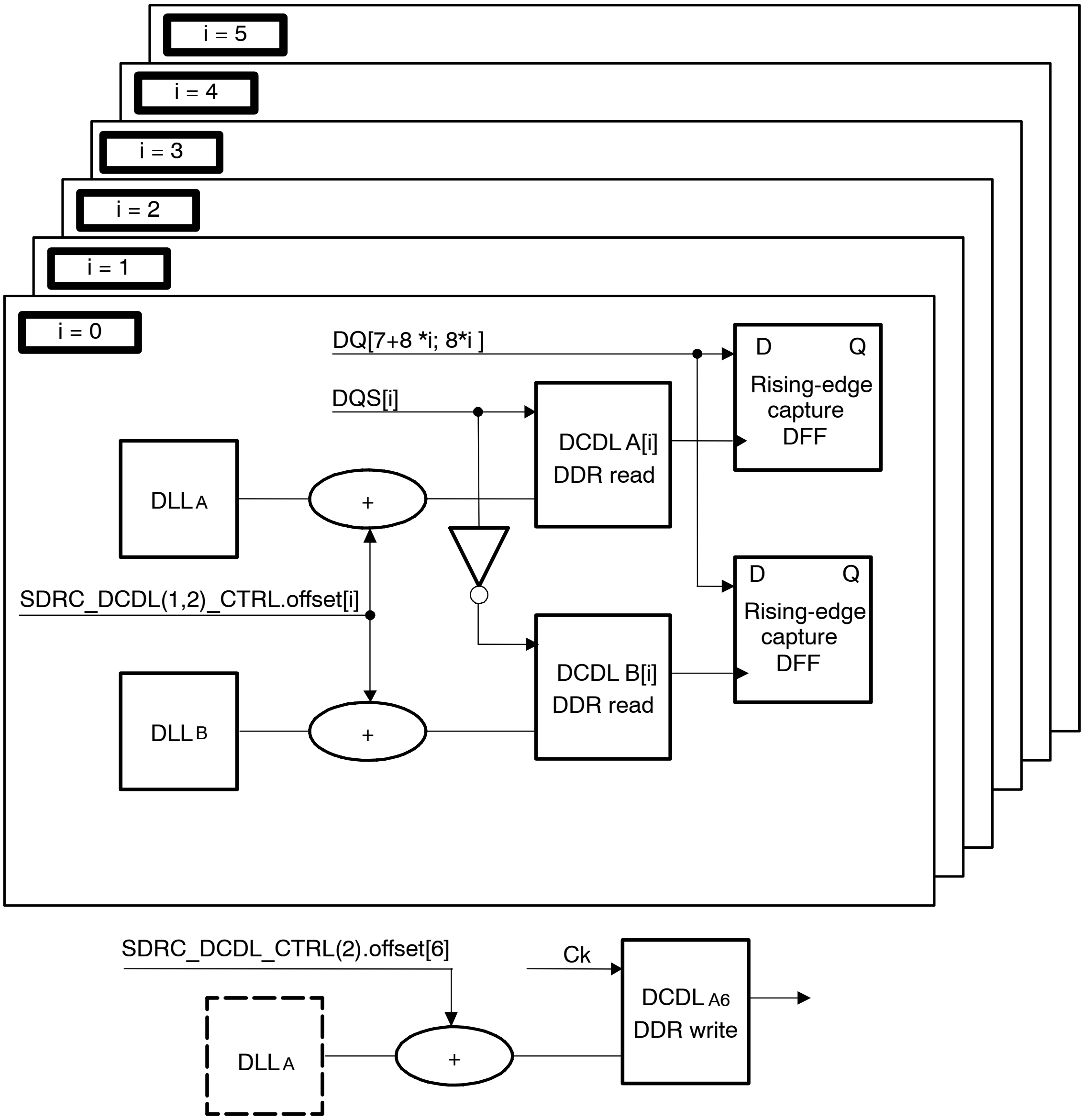
The DLL/DCDL architecture consists of:

- Two DLLs: Two separate DLL elements are used in the SDRC, DLLA, and DLLB. Both DLLs are used to control read timing (rising and falling clock edge); DLLA also controls write timing.
- Twelve DCDLs dedicated to generating the four DQS in DDR read mode, required by the worst-case data lane configuration, which is one chip-select associated with a 16-bit DDR and the others with a 32-bit DDR.
- One DCDL dedicated to generating the DLL write clock.

The SDRC operational range is from the minimum frequency accepted by the attached device up to the L3 maximum frequency. For an SDRAM device (including mobile DDR), the lowest frequency is in the range of a few tens of megahertz, depending on the refresh constraints.

Figure 12−37 shows the DLL/DCDL architecture.

Figure 12–37. DLL/DCDL Architecture



For the configuration of DLL/DCDL modules, see Section 12.11.3.4, *DLL/DCDL Configuration*.

CAUTION

The OMAP2420 device does not use i = 4 and i = 5, which are used only for devices with stacking memory.

12.10.5 Mode Registers

12.10.5.1 Mode Register

The mode register (MR) is common to all SDR and DDR SDRAM. It is a 12-bit register that controls the following parameters:

- ☐ Write burst mode (WBST)
- ☐ CAS latency (CASL)
- ☐ Serial/interleaved mode (SIL)
- ☐ Burst length (BL)

The mode register is accessible through SDRC_MR*a* (*a* is chip-select area 0 or 1). Writing to SDRC_MR*a* initiates an implicit load mode register command qualified by BA1, BA0 = 0, 0.

12.10.5.2 Extended Mode Register 1

The extended mode register 1 (EMR1) is specific to DDR SDRAM. It is a 12-bit register that controls the following parameters:

- ☐ Delay-locked loop (DLL) enable/disable
- ☐ Driver strength (DS)
- ☐ QFC enable

The QFC feature is not supported and must be set to disabled.

The DLL must be enabled during power-up initialization and for normal operation of the DDR SDRAM. It can be disabled for debugging. Returning to normal operation from debug mode requires that the DLL be enabled, after which 200 clock cycles must elapse before the first read is issued.

The DS field defines the output drive strength of the DDR SDRAM. Normal drive strength is specified as SSTL2 Class 2. Reduced drive strength is intended for lighter load and/or point-to-point environments. Some DDR SDRAM devices have a second DS qualifier at bit location 6, which combines with the first bit to give four possible values.

EMR1 is accessible through SDRC_EMR1*a*. Writing to SDRC_EMR1*a* initiates an implicit load mode register command qualified by BA1, BA0 = 0, 1.

12.10.5.3 Extended Mode Register 2

The extended mode register 2 (EMR2) is specific to low-power SDR and mobile DDR SDRAM devices. It is a 12-bit register that controls the following parameters:

- ☐ Temperature compensated self-refresh (TCSR)
- ☐ Partial array self-refresh (PASR)

EMR2 is accessible through SDRC_EMR2*a*. Writing to SDRC_EMR2*a* initiates an implicit load mode register command qualified by BA1, BA0 = 1, 0.

12.11 SDRAM Controller Subsystem Programming Model

This section contains programming guidelines to help set up SMS and SDRC registers according application requirements.

12.11.1 SMS Programming Model

12.11.1.1 Virtual Rotated Frame-Buffer Context Configuration

Using the rotation engine requires several initialization programming steps. After a rotation engine context is set up, an application can use it transparently, as if addressing a frame-buffer object with a standard raster-based memory arrangement.

Step 1: Define the page configuration. This operation normally depends on only the external memory device. The same page settings are then applied for any newly created rotated frame buffer.

Consider an SDRAM device with 1024-byte pages. The page can be defined as a 16x64 array. The page is not necessarily a square (32x32 is also a suitable value). It is recommended that the longest side correspond to the access direction requiring the maximum bandwidth.

For register settings, in any context that uses that page size:

Page width = 2^{pw} bytes, that is., pw = 6

Page height = 2^{ph} bytes, that is, ph = 4

Step 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 x 300 frame buffer, 16 bits per pixel

- Pixel size = 2^{ps} bytes, that is, ps = 1
- Number of pages per line: 400x2 64 = 12.5, rounded up to 13
- Number of pages per column: 300/16 = 18.75, rounded up to 19

For register settings, image size parameters correspond to the enlarged image and are programmed to:

- Image width = w pixels, that is, w = 13x64 /2 = 416
- Image height = h pixels, that is, h = 19x16 = 304

Note:

This example shows obtained values that are not integers but rounded up to the closest integer value, which results in a loss of physical memory that is generally negligible compared to the total size of the image.

In terms of memory allocation (in the physical memory), this corresponds to a 416x304x2 = 252 928-byte buffer.

The physical base address of this buffer must be aligned on a page boundary (in the 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.

Once determined, all these parameters must be loaded in the registers of the chosen context (there are four VRFB contexts with four independent sets of registers).

Example, context 1:

- Configure the physical base address of the frame buffer (for example: 2G bytes, start of CS0, and third quarter Q2):

  SMS_ROT_PHYSICAL_BA1 [PHYSICALBA] :0x20000000

- Configure the image height and width:

  Image height (416): SMS_ROT_SIZE1 [IMAGEHEIGHT]: 0x1A0

  Image width (304): SMS_ROT_SIZE1 [IMAGEWIDTH]: 0x130

- Configure the control parameters:

  Pixel size (for example: 2 bytes, 2^1 bytes): SMS_ROT_CONTROL1[PS]: 0x1

  Page size (for example: 1024 bytes = 64 bytes x 16 bytes)

  Page height (16 bytes, 2^4 bytes): SMS_ROT_CONTROL1[PH]: 0x4

  Page width (64 bytes, 2^6 bytes): SMS_ROT_CONTROL1[PW]: 0x6

Step 3: The frame buffer just created is now ready for use by the system initiators (MPU, sDMA, LCD controller, etc.).

From the perspective of these modules, the frame-buffer object can be accessed at the following virtual rotated frame-buffer address space memory locations:

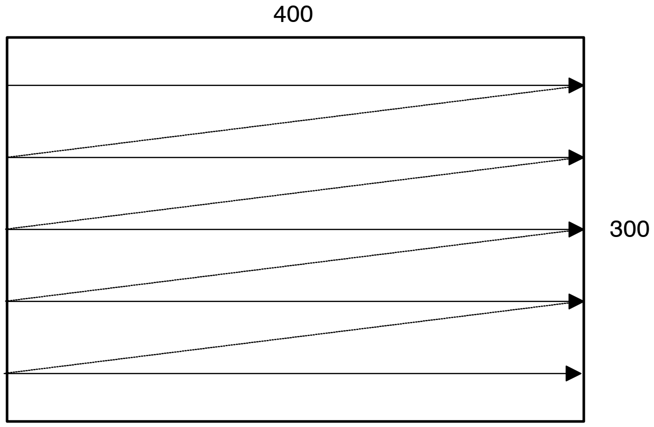
- Context 1 0-degree view: 0x7400 0000
- Context 1 90-degree view: 0x7500 0000
- Context 1 180-degree view: 0x7600 0000
- Context 1 270-degree view: 0x7700 0000

The start address of the view corresponds to the logical (x = 0,y = 0) origin of the image. The image pitch parameter, commonly defined as the distance between 2 vertically adjacent pixels, is fixed to 2048 pixels.

Example of use by the application:

In a system using an 400x300 LCD panel with native landscape orientation:

Figure 12−38. Natural Scan Order



The display buffer is the one created in the sample sequence.

When the application is running and is using 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 accelerators are using the 90-degree view.

When the application is running and is using the landscape orientation for the display (typically a video record/player or gaming application):

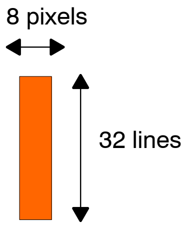
The LCD controller is still accessing the frame buffer using the 0-degree view.

The processor and other initiators such as 2D DMA accelerators are now also using the 0-degree view.

Programming Example

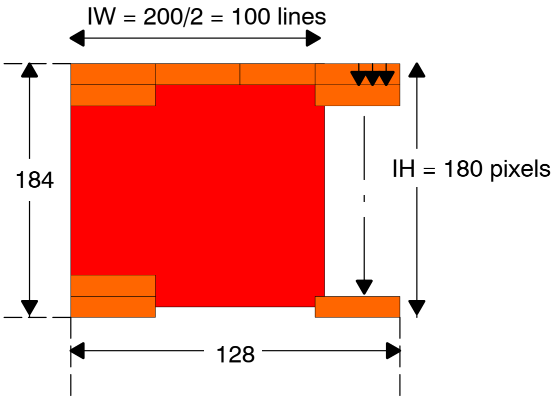
Image 200 x 180, YUV format, written to 90-degree view

1) VRFB page size



PS = 4 bytes/pixel
PW = 5 → 8 pixels
PH = 5 → 32 lines

2) Picture in SDRAM



Number of page column:
100/32 = 3,125 rounded up to 4 → 128
IMAGEHEIGHT = 128
Number of page per line:
180/8 = 22,5 rounded up to 23 → 184
IMAGEWIDTH = 184

Buffer size in physical memory is 128*184*4 = 94208 bytes.

More information about manipulating rotated frame buffers (180, 270 degrees, mirroring) is available in Chapter 15, *Display Subsystem*.

12.11.1.2 Memory-Access Scheduler Configuration

The memory-access scheduler is configured as follows:

- For each of the three classes, the arbitration parameters:
  - SMS_CLASS_ARBITER0–SMS_CLASS_ARBITER2
    - One high-priority FIFO queue in the class (HIGHPRIOVECTOR)
    - Number of consecutive transactions to perform (EXTENDEDGRRANT)
    - Burst transaction submitted to arbitration immediately or after the burst has been buffered (BURSTCOMPLETE)
  - SMS_CLASS_ROTATION0–SMS_CLASS_ROTATION2: Number of consecutive VRFB-split transactions to perform (NOFSERVICES)

12.11.2 SDRAM Controller Configuration

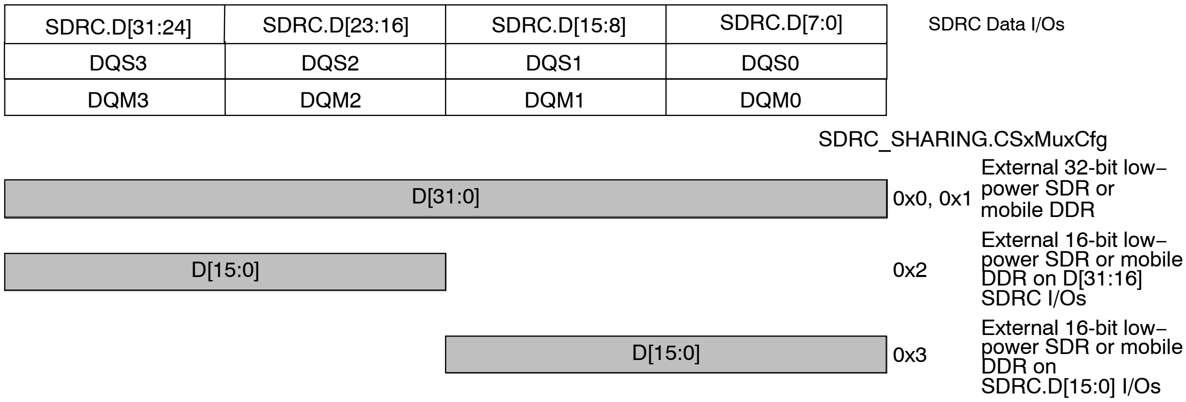
12.11.2.1 IP Revision

The IP revision code can be read in the REV bit field value of the SDRC_REVISION register.

12.11.2.2 I/O Data Lane Configuration

The external data bus interface of the SDRC is 32 bits wide. Depending on the attached memory data bus size and depending on the number of attached devices, you must select the appropriate data lane configuration per chip-select in the register SDRC_SHARING.CSxMuxCfg, as shown in Figure 12−39.

Figure 12−39. SDRC Data Lane Configurations


SDRC.D[31:24]
SDRC.D[23:16]
SDRC.D[15:8]
SDRC.D[7:0]
SDRC Data I/Os
DQS3
DQS2
DQS1
DQS0
DQM3
DQM2
DQM1
DQM0
SDRC_SHARING.CSxMuxCfg
D[31:0]
0x0, 0x1
External 32-bit low−power SDR or mobile DDR
D[15:0]
0x2
External 16-bit low−power SDR or mobile DDR on D[31:16] SDRC I/Os
D[15:0]
0x3
External 16-bit low−power SDR or mobile DDR on SDRC.D[15:0] I/Os

Note:

Ensure that static I/O sharing between the SDRC and the GPMC is not corrupted by a noncompatible data lane configuration. The OMAP2420 can support a 32-bit-wide external DDR/SDRAM interface only when GPMC use is restricted to address and data-multiplexed protocol (random synchronous or asynchronous memory) and NAND device type. Setting SDRC_SHARING.CSxMuxCfg to 0x0 or 0x1 is not compatible with an external address/data nonmultiplexed device.

12.11.3 SDRC Setup

A number of device parameters must be set before executing the initialization sequence.

12.11.3.1 Chip-Select Configuration

Chip-select CS0 is always located at 0x80000000 (with respect to the 32-bit interconnect address). There is no restriction on the presence of any SDRAM on either CS0 or CS1.

The total address space of the SDRC is 1G byte, divided into 128M-byte partitions. Each 128M-byte address space is also segmented into a 32M-byte address space. Each partition is a possible start address for CS1, except the partition occupied by CS0.

The start address for CS1 is defined by the CS1STARTHIGH and CS1STARTLOW bit fields in the SDRC_CS_CFG register.

Space 0, selected by the SDRC module using the chip-select 0 (CS0) output pin, is always at offset 0 from the SDRAM memory space base address. The size of this area is programmable through the RAMSIZE field of the SDRC_MCFG0 register.

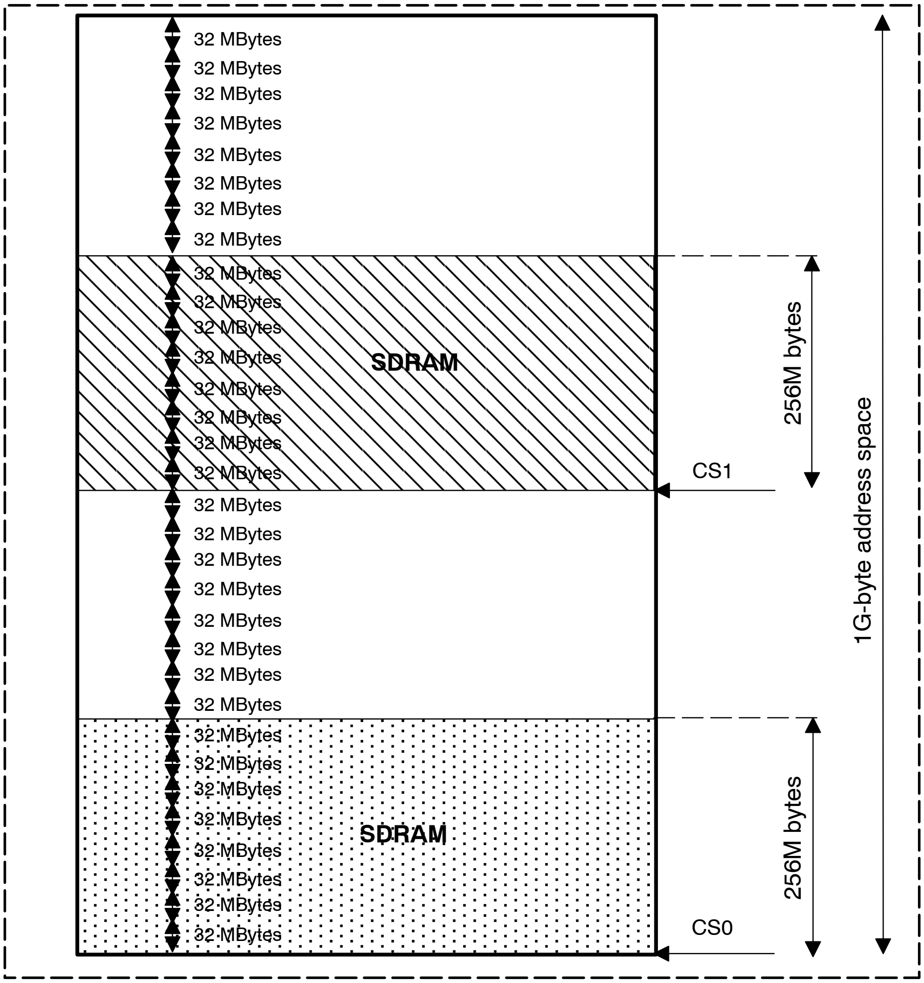
Space 1, selected by the SDRC module using the chip-select 1 (CS1) output pin, is located at an offset (configured by the CS1STARTHIGH and CS1STARTLOW bit fields in the SDRC_CS_CFG register) from the SDRAM memory space base address, programmable in 32M-byte increments. The size of this area is programmable through the RAMSIZE field of the SDRC_MCFG1 register.

Note:

Ensure that space 0 and 1 do not overlap each other or extend farther than the maximum 1G-byte SDRC memory space.

The type of device and the data bus width (16 bits or 32 bits) for each area are programmable through the SDRC_MCFG[x] register. Figure 12−40 shows a possible configuration.

Figure 12−40. CS0/CS1 Chip-Select Start Address Slots (With 32M Bytes Granularity for CS1)



12.11.3.2 Memory Configuration

The memory configuration is defined on a per-chip-select basis using SDRC_MCFG_1.

Table 12−58. Memory Configuration

Bit Field	Register	Comments
ADDRMUX	SDRC_MCFG_1	Address-multiplexing scheme
B32NOT16	SDRC_MCFG_1	The external device is x32-bit or x16 bit
DEEPPD	SDRC_MCFG_1	Set this bit if the .memory supports deep power-down
DDR type	SDRC_MCFG_1	Regular DDR or mobile
RAM type	SDRC_MCFG_1	Single-data-rate SDRAM (SDR/LP−SDR) Double-data-rate SDRAM (M−DDR)

Table 12−51 and Figure 12−31 through Figure 12−33 define the detailed address multiplexing programming scheme.

12.11.3.3 SDRAM ac Timing Parameters

The following ac parameters can be programmed (standard SDRAM terminology is used here) in clock cycles independently for each of the two memory areas using registers SDRC_ACTIM_CTRLA_a (a: 0 or 1 depending on the area) and SDRC_ACTIM_CTRLB_a.

Table 12−59. Programmable ac Parameters

SDRC ac Parameter	Description	Range (Clock Cycles)
tRFC	Autorefresh period	0−31
tRC	Row cycle time	0−31
tRAS	Row active time	0−15
tRP	Row precharge time	0−7
tRCD	Row-to-column delay time	0−7
tRRC	Active-to-active command period	0−7
tDPL/tWR/tCDLR	Data in to precharge command (Write recovery time)	0−7
tDAL	Data-in-to-active command	0−31
tXSR	Exit self-refresh to active time	0−255

Configuration example:

If you require a minimum tRC of 88 ns under 100 MHz, 88/10 = 9 (8.8 is rounded up).

Therefore, you must set 9 clock cycles.

Table 12−60 lists the nonprogrammable SDRC ac parameters, which are hard coded.

Table 12−60. Nonprogrammable ac Parameters

SDRC ac Parameter	Description	Range (Clock Cycles)
tCCD	Read/write command to read/write command	1
tCKED	CKE to clock disable or power-down entry mode	1
tPED	CKE to clock enable or power-down exit set up	1
tMRD	Last register command to active or refresh command	3
tDQM	DQM to data mask during writes	0
tDQZ	DQM to data high impedance during reads	2
tDQSS	Write command to first DQS latching transition	1
tRPRE	DQS read preamble	1

Note:

The SDRC uses tXSR only when exiting self-refresh mode, regardless of the first command issued. The SDRC systematically inserts an autorefresh command before it serves the first request.

12.11.3.4 DLL/DCDL Configuration

The DLLPHASE control bit is used to set up the nominal delay tracked by the DLL. This delay must be set up for 90 degrees (25 percent of the clock period) for DDR clock frequencies equal to or lower than 133 MHz, and 72 degrees (20 percent of the clock period) for higher DDR clock frequencies. The DLL locking range is from 66 MHz to L3 maximum frequency.

Below 66 MHz, the DLL must be used in unlock mode. Delays are not PVT-compensated, but this is compatible with the MDDR timing requirements in that frequency range.

The DLL can be loaded with an 8-bit delay programmable value with a valid range of 0−239 where each step represents 22 ps in nominal PVT conditions. This delay must be set with a granularity of 4; that is, Delay[1:0] must be set to 0b11.

The delay unit is loaded when a LOADDLL bit field low-to-high transition occurs. The load capability can be used either to shorten the locking time, by setting an initial value near the expected locked state value, or to assert a given value permanently (unlock mode).

When loading a value into the DLL, the DLL must be enabled (ENADLL set to 1) for the load to be effective, for at least two clock cycles.

To set up the controlled delay block for unlock mode:

- Write to the SDRC_DLLx_CTRL, with ENADLL set, LOADDLL set, DELAY field set to the expected value.

To set up the controlled delay block for lock mode (normal mode):

- Write to the SDRC_DLLx_CTRL register, with ENADLL set, LOADDLL set, DELAY field set to 0x77 to minimize the locking time.
- Write to the SDRC_DLLx_CTRL register, with LOADLL reset, to launch the tracking process.
- Wait for at least 480 TC clock cycles to be sure the DLL is locked before accessing the memory.

Once locked, the DLL can be periodically disabled under software control, if this period is short compared with the voltage and temperature variations. When disabled, the DLL counter is frozen at its current value. When enabled again, the tracking resumes from that DLL counter value.

The two registers SDRC_DCDL(1,2)_CTRL contain the following bit fields:

- Offset0: DQS0 adjustment, read mode, nominally 0
- Offset1: DQS1 adjustment, read mode, nominally 0
- Offset2: DQS2 adjustment, read mode, nominally 0
- Offset3: DQS3 adjustment, read mode, nominally 0
- Offset6: Data to DQS delay adjustment, write mode, nominally 0

DLL behavior can be monitored by the software using the SDRC DLL status registers (SDRC_DLLA_STATUS and SDRC_DLLB_STATUS). For each DLL, the current counter value is available (DLLCNT field), as well as two status bits: overflow (OVF field) and underflow (UDF field). Overflow is set when the DLL counter value is 255 during at least one SDRC clock cycle, and underflow is set when the DLL counter is 0 during at least one SDRC clock cycle.

These two status bits are both reset if the DLL is disabled (ENADLL is set to 0 in the DLL control register).

12.11.3.5 Mode Register Programming and Modes of Operation

Mode Register

The 12-bit mode register (MR) is common to all SDR and DDR SDRAMs. It controls the following parameters:

- Write burst mode
- CAS latency
- Serial/interleaved mode
- Burst length

When programming SDRC_MR_a bit fields, consider the following:

- CAS latencies of 1, 2, 3, 4, and 5 are supported (CASL).
- Only serial mode (not interleaved mode) is supported (SIL: 0x0).
- A burst length of 2 is supported for SDR SDRAM (BL: 0x2).
- A burst length of 4 is supported for DDR SDRAM (BL: 0x4).
- Burst lengths of 1 (BL: 0x0), 8 (BL: 0x3), or full page (BL: 0x7) are not supported.

The mode register is accessible through SDRC_MR*a*. Writing to SDRC_MR*a* initiates an implicit load mode register command qualified by BA1, BA0 = 0, 0.

Extended Mode Register 1

The 12-bit extended mode register 1 (EMR1) is specific to the DDR SDRAM- and controls the following parameters:

- Delay-locked loop (DLL) enable/disable
- Driver strength (DS)
- QFC enable

The QFC feature is not supported and must be disabled.

The DLL must be enabled during power-up initialization and normal operation of the DDR SDRAM, but it can be disabled for debugging. Returning to normal operation from debug mode requires the DLL to be enabled, after which 200 clock cycles must elapse before the first read is issued.

The DS field defines the output drive strength of the DDR SDRAM. The normal drive strength is specified to be SSTL2 class 2. The reduced drive strength is intended for lighter load and/or point-to-point environments. Some DDR SDRAM devices have a second DS qualifier at bit location 6 that combines with the first bit for four possible values.

EMR1 is accessible through SDRC_EMR1*a*. Writing to SDRC_EMR1*a* initiates an implicit load mode register command qualified by BA1, BA0 = 0, 1.

Extended Mode Register 2

The 12-bit extended mode register 2 (EMR2) is specific to low-power SDR and mobile DDR SDRAM devices. It is a 12-bit register that controls the following parameters:

- Temperature-compensated self-refresh
- Partial array self-refresh

EMR2 is accessible through SDRC_EMR2*a*. Writing to SDRC_EMR2*a* initiates an implicit load mode register command qualified by BA1, BA0 = 1, 0.

12.11.3.6 Autorefresh Management

The SDRAM refresh configuration register group controls refresh management in normal operation. This group contains two SDRC_RFR_CTRL registers that are defined on a per-chip-select basis and contain the following bit fields:

- ARE
- ARCV

The ARE field enables and disables autorefresh. Autorefresh bursts of 1, 4, and 8 are programmed using this field. The autorefresh burst starts when the 16-bit autorefresh counter decrements to 0.

The ARCV field loads the autorefresh counter with a 16-bit autorefresh value. The ARCV value is calculated using the formula:

> Autorefresh counter value = (memory refresh interval time/SDRC.CLK period) – margin (50 cycles).

The margin takes into account that there may be an ongoing access when the counter expires, delaying the effective refresh sequence.

The value to be programmed is independent of the burst-refresh configuration. If burst-refresh is selected, the value is automatically scaled to the burst-refresh size in hardware.

Autorefresh is enabled using the CMDCODE bit field of the relevant SDRC_MANUAL[x] register (CMDCODE: 0x2).

12.11.4 Manual Software Commands

The manual commands register SDRC_MANUAL_X (where X is the chip-select area 0 or 1) implements these software-driven commands:

- NOP command (CMDCODE: 0x0)

  When the CMDCODE bits SDRC_MANUAL_X[3:0] are programmed with 0x0, SDRC generates a NOP/INHIBIT command. The SDRC memory port signals have the following status:

  - NOP does not initiate any new operation, but is needed to complete operations requiring more than a single clock cycle-like auto-refresh.
  - Inhibit is also a NOP. nCS high disables the command decoder so that NRAS, NCAS, NWE and all the address inputs are ignored.

nCS	NRAS	NCAS	NWE	Command
H	X	X	X	INHIBIT
0	H	H	H	NOP

- PRECHARGE ALL command (CMDCODE: 0x1)

  When the CMDCODE bits SDRC_MANUAL_X[3:0] are programmed with 0x1, the SDRC generates a PRECHARGE ALL command. The SDRC memory port signals have the following status:

  - Before performing a PRECHARGE ALL command, all banks must satisfy the tRAS(min) requirement.
  - During the PRECHARGE ALL command, address A10 remains high. Bank information (BA0 and BA1) is don’t care.

- The PRECHARGE ALL command allows all banks to be precharged at the same time.
- At the end of tRP and after performing precharge to all the banks, all banks are in idle state.

nCS	NRAS	NCAS	NWE	Command
L	L	H	L	PRECHARGE ALL

❑ AUTOREFRESH command (CMDCODE: 0x2)

When the CMDCODE bits SDRC_MANUAL_X[3:0] are programmed with 0x2, the SDRC generates an AUTOREFRESH command. The SDRC memory port signals have the following status:

- In addition to the signal status described below, the CKE signal is at logic high.
- The AUTOREFRESH command can be asserted only with all banks in idle state and the device not in power-down mode (CKE is high in the previous cycle).
- The AUTOREFRESH command must be followed by NOPs until the autorefresh operation is complete.
- All banks are in idle state at the end of the autorefresh operation.

nCS	NRAS	NCAS	NWE	Command
L	L	L	H	AUTOREFRESH

❑ Enter deep power-down mode (CMDCODE0: 0x3).

When the CMDCODE bits SDRC_MANUAL_X[3:0] are programmed with 0x03, the SDRC executes ENTER DPDM for low-power devices (that is, LPSDR or MDDR devices that support deep power-down mode). The SDRC memory port signals have the following status.

The device enters deep power-down mode by holding nCS and NWE at logic low with NRAS and NCAS high at the rising edge of the clock; CKE is low.

nCS	NRAS	NCAS	NWE	CKE	Command
0	H	H	L	0	ENTER DEEP POWER DOWN MODE

❑ Exit deep power-down mode (CMDCODE: 0x4).

When the CMDCODE bits SDRC_MANUAL_X[3:0] are programmed with 0x4, SDRC executes the EXIT DPDM command. The device exits deep power-down mode when the INHIBIT command is sampled with CKE held at logic high.

CAUTION

Issuing a DPD exit command without having sent a DPD enter command can lead to unpredictable behavior.

- Enter self-refresh mode (CMDCODE: 0x5).

  When the CMDCODE bits SDRC_MANUAL_X[3:0] are programmed with 0x5, the SDRC executes the SELF-REFRESH ENTRY command. The signal statuses are the same as described for the AUROREFRESH command. In addition, CKE is held at logic low during self-refresh mode.

  Self-refresh mode is entered from the all banks idle state by asserting low on nCS, NRAS, NCAS, and CKE and high on NWE.

  After self-refresh mode is entered, the CKE state being low is the only input that matters; all other inputs, including the clock, are ignored to remain in self-refresh mode.

- Exit self-refresh mode (CMDCODE: 0x6).

  When the CMDCODE bits SDRC_MANUAL_X[3:0] are programmed with 0x6, the SDRC executes the EXIT SELF REFRESH command and, after meeting the tXSR timing parameter, executes one AUTOREFRESH command to adhere to the memory protocol. The EXIT SELF REFRESH command executes when CKE is detected high with a NOP command.

  Restarting the external clock and then asserting high on CKE exits self-refresh. These actions must be followed by NOPs for a minimum tXSR before the SDRAM reaches idle state to begin normal operation.

- Set CKE signal high (CMDCODE: 0x7).

  When the CMDCODE bits SDRC_MANUAL_X[3:0] are programmed with 0x7, CKE is set to high. This command is used, for example, during DDR memory initialization.

- Set CKE low (CMDCODE: 0x8).

  When the CMDCODE bits SDRC_MANUAL_X[3:0] are programmed with 0x8, CKE is set low. This command is used, for example, to put memory in power-down mode.

12.11.4.1 Low-Power SDR/Mobile DDR Initialization Sequence

The steps in the initialization sequence are as follows:

Step 1: Apply power and make sure CKE is set high.

Step 2: Maintain stable power, stable clock, and NOP input condition for a minimum of 200 μs (CMDCODE : 0x0).

Step 3: Issue precharge commands for all banks of the devices (CMDCODE: 0x1).

Step 4: Issue 2 or more autorefresh commands (CMDCODE: 0x2).

Step 5: Issue BA1 = 0, BA0 = 0 to load mode register.

Because the mode register powers up in an unknown state, it must be loaded before applying any operational command.

When MR is reprogrammed, a suitable amount of time needs to elapse before an active command can be issued. This is ensured by the hardware with tMRD fixed to 3 clock cycles.

12.11.4.2 Memory Power Management

Clock-Enable Management

The SDRC supports two autonomous clock enable pins (CKE0**,** CKE1), and each chip-select (CS0, CS1) has its own CKE signal. The power management of the CS0) memory is controlled by CKE0. The power management of the CS1 memory is controlled by CKE1. This allows the SDRAM memories associated with CS0 and CS1 to be independently placed in either deep power-down (DPD) or self-refresh (SR) mode.

External Clock Control Power Management

The SDRC power-management register (SDRC_POWER) contains the EXTCLKDIS field, which controls suspension of the external clock on a per-chip-select basis. Writing 1 to this field in the relevant register freezes the clock with a latency dependent on the SDRC_POWER register programming. Subsequently writing 0 to this field in the relevant register enables the clock with a latency dependent on the SDRC_POWER register programming.

Deep Power-Down Mode Power Management

Deep power-down mode is found in SDR, M−DDR. Power distribution to the entire memory array is cut. The programming model for deep power-down is as follows:

LP SDR/M−DDR/Deep Power-Down Mode Entry

- ☐ Precharge all banks (CMDCODE: 0x1).
- ☐ Enter deep power-down mode (CMDCODE: 0x3).

LP SDR/M−DDR/Deep Power-Down Mode Exit

- ☐ Exit deep power-down mode (CMDCODE: 0x4).

  The values of the mode register and the extended mode register are retained on exiting deep power-down.

Note:

Power-down entry/exit sequences depend on memory devices; for the complete sequence, see the applicable memory specification.

Self-Refresh Mode Power Management

The programming model for entering and exiting self-refresh mode is as follows:

Self-Refresh Entry

- Precharge all banks (CMDCODE: 0x1).
- NOP (CMDCODE: 0x0)
- Enter self-refresh mode (CMDCODE: 0x5).

There is no need for the software to disable the autorefresh in SDRC_RFR_CTRL before entering self-refresh. The autorefresh counter is reset in hardware, so that an autorefresh cycle is automatically generated immediately after self-refresh exit, before any other command.

Self-Refresh Exit

- Exit self-refresh mode (CMDCODE: 0x6).
- Reconfigure SDRC registers as required.
- Enable autorefresh by programming:
  - The relevant SDRC_RFR_CTRL register ARCV field.
  - The relevant SDRC_RFR_CTRL register ARE field to the desired refresh burst.

12.11.5 Error Management

All data transfers in the SDRC operate a system of full handshaking. A valid read or write request presented to the SDRC by the L3 interconnect sequencer results in the SDRC acknowledging the transfer by raising the SCMDACCEPT flag. Failure to do this within a defined temporal window constitutes an error. Errors can arise from the following sources:

- Transaction error while the memory is in deep power-down mode
- Interconnect transaction error
- Illegal initiator access
- Capture of the address of the last illegal access in the SDRC_ERR_ADDR register

If an error occurs, the software error handler performs the following actions:

- Interrogates the ERRORVALID field of the SDRC_ERR_TYPE register to verify the presence of an error.
- Interrogates the ERRORDPD field of the SDRC_ERR_TYPE register to ascertain the presence or absence of a transaction error resulting from the device being in deep power-down mode.

- Interrogates the ERRORMCMD field of the SDRC_ERR_TYPE register to ascertain the presence or absence of a transaction error resulting from an interconnect transaction error.
- Interrogates the ERRORCONNID field of the SDRC_ERR_TYPE register to ascertain the presence or absence of a transaction error resulting from an illegal access from an Interconnect initiator.
- Interrogates the ERRORADD filed of SDRC_ERR_TYPE register to ascertain the presence or absence of a transaction error resulting from an illegal address:
  - Interconnect access to an address outside the memory space (0x0)
  - Interconnect access to an address outside the register space (0x1)
  - Interconnect access to a memory location not refreshed (0x3)
- Writes 0 to the ERRORVALID field of the SDRC_ERR_TYPE register to clear the active error status.
- Executes error recovery from software.

12.12 SDRC Subsystem Registers

12.12.1 SDRC Subsystem Instance Summary

Table 12−61 shows the base address and address space for the SMS module instances.

Table 12−61. SMS Instance Summary

Module Name	Base Address	Size
SMS	0x6800 8000	64K bytes

Table 12−62 shows the base address and address space for the SDRC module instances.

Table 12−62. SDRC Instance Summary

Module Name	Base Address	Size
SDRC	0x6800 9000	64K bytes

12.12.2 SDRAM Controller Subsystem Register Summary

This section summarizes the SDRAM controller subsystem module registers.

Table 12−63. SMS Register Summary

Register Name	Type	Register Width (Bits)	Offset
SMS_REVISION	R	32	0x000
SMS_SYSCONFIG	RW	32	0x010
SMS_SYSSTATUS	R	32	0x014
Reserved		32	0x040−0x070
Reserved		32	0x044−0x074
Reserved		32	0x048−0x078
Reserved		32	0x080−0x0B0
Reserved		32	0x084−0x0B4
Reserved		32	0x0C0
SMS_CLASS_ARBITER0	RW	32	0x0D0
SMS_CLASS_ARBITER1	RW	32	0x0D4
SMS_CLASS_ARBITER2	RW	32	0x0D8
SMS_INTERCLASS_ARBITER	RW	32	0x0E0
SMS_CLASS_ROTATION0 − SMS_CLASS_ROTATION2	RW	32	0x0E4−0x0EC
SMS_ERR_ADDR	R	32	0x0F0
SMS_ERR_TYPE	RW	32	0x0F4
SMS_POW_CTRL	RW	32	0x0F8
SMS_ROT_CONTROL0 − SMS_ROT_CONTROL3	RW	32	0x100−0x130

Table 12−63. SMS Register Summary (Continued)

Register Name	Type	Register Width (Bits)	Offset
SMS_ROT_SIZE__0_3	RW	32	0x104−0x134
SMS_ROT_PHYSICAL_BA__0_3	RW	32	0x108−0x138

Table 12−64. SDRC Register Summary

Register Name	Type	Register Width (Bits)	Offset
SDRC_REVISION	R	32	0x000
SDRC_SYSCONFIG	RW	32	0x010
SDRC_SYSSTATUS	R	32	0x014
SDRC_CS_CFG	RW	32	0x040
SDRC_SHARING	RW	32	0x044
SDRC_ERR_ADDR	R	32	0x048
SDRC_ERR_TYPE	RW	32	0x04C
SDRC_DLLA_CTRL	RW	32	0x060
SDRC_DLLA_STATUS	R	32	0x064
SDRC_DLLB_CTRL	RW	32	0x068
SDRC_DLLB_STATUS	R	32	0x06C
SDRC_POWER	RW	32	0x070
SDRC_MCFG_0	RW	32	0x080
SDRC_MR_0	RW	32	0x084
SDRC_EMR1_0	RW	32	0x088
SDRC_EMR2_0	RW	32	0x08C
SDRC_EMR3_0	RW	32	0x090
SDRC_DCDL1_CTRL	RW	32	0x094
SDRC_DCDL2_CTRL	RW	32	0x098
SDRC_ACTIM_CTRLA_0	RW	32	0x09C
SDRC_ACTIM_CTRLB_0	RW	32	0x0A0
SDRC_RFR_CTRL_0	RW	32	0x0A4
SDRC_MANUAL_0	RW	32	0x0A8
SDRC_MCFG_1	RW	32	0x0B0
SDRC_MR_1	RW	32	0x0B4
SDRC_EMR1_1	RW	32	0x6800 90B8
SDRC_EMR2_1	RW	32	0x6800 90BC
SDRC_EMR3_1	RW	32	0x6800 90C0
SDRC_ACTIM_CTRLA_1	RW	32	0x6800 90C4

Table 12−64. SDRC Register Summary (Continued)

Register Name	Type	Register Width (Bits)	Offset
SDRC_ACTIM_CTRLB_1	RW	32	0x6800 90C8
SDRC_RFR_CTRL_1	RW	32	0x6800 90D4
SDRC_MANUAL_1	RW	32	0x6800 90D8

12.12.3 SMS Register Descriptions

Table 12−65. SMS_REVISION

Address Offset	0x000		
Physical Address	0x6800 8000	**Instance**	SMS1
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 code [7:4] Major revision [3:0] Minor revision Examples: 0x10 for 1.0, 0x21 for 2.1	R	

Table 12–66. SMS_SYSCONFIG

Address Offset	0x010		
Physical Address	0x6800 8010	**Instance**	SMS1
Description	This register controls the OCP interface 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																											SIDLEMODE		Reserved	SOFTRESET	AUTOIDLE

Bits	Field Name	Description	Type	Reset
31:5	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000000
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—An idle request is acknowledged based on the internal activity of the module. 0x3: Reserved—Do not use.	RW	0x0
2	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
1	SOFTRESET	Software reset 0x0: Normal mode (no reset applied) 0x1: Software reset is activated.	RW	0
0	AUTOIDLE	Internal OCP clock-gating strategy 0x0: OCP clock is free-running. 0x1: Automatic OCP clock gating strategy is applied based on OCP interface activity.	RW	0

Table 12−67. SMS_SYSSTATUS

Address Offset	0x014		
Physical Address	0x6800 8014	**Instance**	SMS1
Description	This register provides status 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	Reserved for module-specific status information Read returns 0.	R	0x000000
7:1	Reserved	Reserved for OCP socket status information. Read returns 0.	R	0x00
0	RESETDONE	Internal reset monitoring 0x0: Internal module reset is ongoing. 0x1: Reset completed. The module is ready to be used.[1]	R	–

[1] During reset, the value is 0, but the value read just after the reset is 1, because ICLK/FCLK is already operating.

Table 12−68. SMS_CLASS_ARBITER0

Address Offset	0x0D0		
Physical Address	0x6800 80D0	**Instance**	SMS1
Description	This register controls the arbitration parameters between the class i 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
BURSTCOMPLETE		Reserved						EXTENDEDGRANT				Reserved												HIGHPRIOVECTOR		Reserved					

Bits	Field Name	Description	Type	Reset
31:30	BURSTCOMPLETE	Delayed service until burst request complete BURSTCOMPLETE[k], k = 6 to 7 (group number). Group 6 is controlled by bit 30; group 7 is controlled by bit 31. 0x0: Group k request to arbiter issued as soon as the first burst request is available	RW	0x0

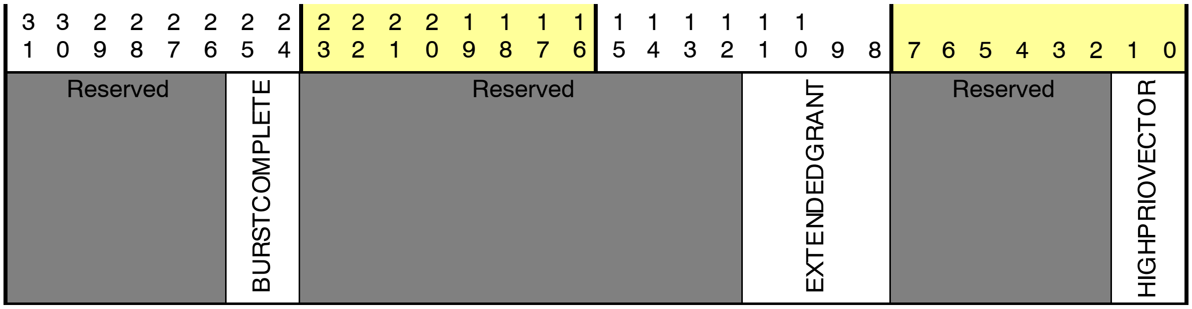
Bits	Field Name	Description	Type	Reset
		0x1: Group k request to arbiter delayed until a complete burst transaction is buffered		
29:24	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
23:20	EXTENDEDGRANT	Vector specifying the number of consecutive services a group can be granted consecutively: 2 bits per group EXTENDEDGRANT[2*k+1,2*k], k = 6 to 7 (group number). Group 6 is controlled by bits [21:20]; group 7 is controlled by bits [23:22]. 0x0: Not defined 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	0x1

Table 12−68. SMS_CLASS_ARBITER0 (Continued)

Bits	Field Name	Description	Type	Reset
19:8	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x000
7:6	HIGHPRIOVECTOR	Vector allocating a higher priority to one of the class members. One group at a time can be given this attribute. HIGHPRIOVECTOR[k], k = 6 to 7 (group number). Group 6 is controlled by bit 6; group 7 is controlled by bit 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 0.	RW	0x00

Table 12−69. SMS_CLASS_ARBITER1

Address Offset	0x0D4		
Physical Address	0x6800 80D4	**Instance**	SMS1
Description	This register controls the arbitration parameters between the class i request groups.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:26	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
25:24	BURSTCOMPLETE	Security violation error BURSTCOMPLETE[k], k = 0 to 1 (group number). Group 0 is controlled by bit 24; group 1 is controlled by bit 25. 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 0.	RW	0x000

Table 12−69. SMS_CLASS_ARBITER1 (Continued)

Bits	Field Name	Description	Type	Reset
11:8	EXTENDEDGRANT	Vector specifying the number of consecutive services a group can be granted consecutively. Two bits per group EXTENDEDGRANT[2*k+1,2*k], k = 0 to 1 (group number). Group 0 is controlled by bits [9:8]; group 1 is controlled by bits [11:10]. 0x0: Not defined 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	0x1
7:2	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
1:0	HIGHPRIOVECTOR	Vector allocating a higher priority to one of the class members. One group at a time can be given this attribute . HIGHPRIOVECTOR[k], k = 0 to 1 (group number). Group 0 is controlled by bit 0; group 1 is controlled by bit 1. 0x0: Group k has standard priority (LRU based). 0x1: Group k has the highest priority over all other class members.	RW	0x0

Table 12−70. SMS_CLASS_ARBITER2

Address Offset	0x0D8		
Physical Address	0x6800 80D8	**Instance**	SMS1
Description	This register controls the arbitration parameters between the class i 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		BURSTCOMPLETE				Reserved						EXTENDEDGRANT								Reserved						HIGHPRIOVECTOR				Reserved	

Bits	Field Name	Description	Type	Reset
31:30	Reserved	Write 0s for future compatibility. Read returns 0s	RW	0x0

Table 12−70. SMS_CLASS_ARBITER2 (Continued)

Bits	Field Name	Description	Type	Reset
29:26	BURSTCOMPLETE	BURSTCOMPLETE[k], k = 2 to 5 (group number). Group 2 is controlled by bit 26; group 3 is controlled by bit 27; group 4 is controlled by bit 28; group 5 is controlled by bit 29. 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 0.	RW	0x00
19:12	EXTENDEDGRANT	Vector specifying the number of consecutive services a group can be granted consecutively. 2 bits per group EXTENDEDGRANT[2*k+1,2*k], k = 2 to 5 (group number). Group 2 is controlled by bits [13:12]; group 3 is controlled by bits [15:14]; group 4 is controlled by bits [17:16]; group 5 is controlled by bits [19:18]. 0x0: Not defined 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	0x01
11:6	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
5:2	HIGHPRIOVECTOR	Vector allocating a higher priority to one of the class members. One group at a time can be given this attribute. HIGHPRIOVECTOR[k], k = 2 to 5 (group number). Group 2 is controlled by bit 2; group 3 is controlled by bit 3; group 4 is controlled by bit 4; group 5 is controlled by bit 5. 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 0.	RW	0x0

Table 12−71. SMS_INTERCLASS_ARBITER

Address Offset 0x0E0

Physical Address 0x6800 80E0 **Instance** SMS1

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 0.	RW	0x00
23:16	CLASS2PRIO	Class 2 high priority window width (clock cycle count)	RW	0x40
15:8	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
7:0	CLASS1PRIO	Class 1 high priority window width (clock cycle count)	RW	0x40

Table 12−72. SMS_CLASS_ROTATION0−SMS_CLASS_ROTATION2

Address Offset 0x0E4−0x0EC in 0x4 byte increments

Physical Address 0x6800 80E4−0x6800 80EC **Instance** SMS1

Description This register controls the number of consecutive services that can be allocated to a thread, the transactions of which 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																											NOF SERVICES				

Bits	Field Name	Description	Type	Reset
31:5	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000000
4:0	NOFSERVICES	Number of RE split transactions serviced consecutively when the thread gets granted by the arbitration logic: 0x00: Not defined 0x01: Transaction granted ... 0x1F: 31 transactions granted	RW	0x01

Table 12–73. SMS_POW_CTRL

Address Offset	0x0F8		
Physical Address	0x6800 80F8	**Instance**	SMS1
Description	This register controls SMS power management in conjunction with the regular OCP 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 0.	RW	0x000000
7:0	IDLEDELAY	Delay before auto-idle when no more traffic is in the SMS 8-bit value, in L3 clock cycle units	RW	0x80

Table 12−74. SMS_ROT_CONTROL0—SMS_ROT_CONTROL3

Address Offset	0x100−0x130 in 0x10 byte increments		
Physical Address	0x6800 8100−0x6800 8130	**Instance**	SMS1
Description	The control register configures the virtual rotated frame-buffer module for context 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																					PH			Reserved	PW			Reserved		PS	

Bits	Field Name	Description	Type	Reset
31:11	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x000000
10:8	PH	Exponent based 2 value, 2^ph indicates the page height in rows for context i.	RW	0x0
7	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
6:4	PW	Exponent based 2 value, 2^pw indicates the page width in bytes for context i.	RW	0x0
3:2	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
1:0	PS	Exponent-based 2 value, 2^ps indicates the pixel size in bytes for context i. The value 3 is invalid.	RW	0x0

Table 12−75. SMS_ROT_SIZE0—SMS_ROT_SIZE3

Address Offset	0x104−0x134 in 0x10 byte increments		
Physical Address	0x6800 8104−0x6800 8134	**Instance**	SMS1
Description	The control register configures bank organization for context 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					IMAGEHEIGHT											Reserved					IMAGEWIDTH										

Bits	Field Name	Description	Type	Reset
31:27	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
26:16	IMAGEHEIGHT	Image height in pixels for context i	RW	0x000
15:11	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
10:0	IMAGEWIDTH	Image width in pixels for context i	RW	0x000

Table 12−76. SMS_ROT_PHYSICAL_BA0—SMS_ROT_PHYSICAL_BA3

Address Offset	0x108−0x138 in 0x10 byte increments		
Physical Address	0x6800 8108−0x6800 8138	**Instance**	SMS1
Description	The control register allows configuration of the physical base address for context 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	PHYSICALBA																														

Bits	Field Name	Description	Type	Reset
31	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
30:0	PHYSICALBA	Physical base address of the frame buffer for context i in SDRAM	RW	0x00000000

12.12.4 SDRC Register Descriptions

Table 12−77. SDRC_REVISION

Address Offset	0x000		
Physical Address	0x6800 9000	**Instance**	SDRC1
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 code [7:4] Major revision [3:0] Minor revision Examples: 0x10 for 1.0, 0x21 for 2.1	R	

Table 12−78. SDRC_SYSCONFIG

Address Offset	0x010		
Physical Address	0x6800 9010	**Instance**	SDRC1
Description	This register controls the OCP interface 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																											IDLEMODE		Reserved	SOFTRESET	Reserved

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/ackcontrol 0x0: Reserved—Do not use. 0x1: Reserved—Do not use. 0x2: Smart idle—An idle request is acknowledged based on the internal activity of the module. Issued when the SDRC enters self-refresh. 0x3: Reserved—Do not use.	RW	0x2
2	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
1	SOFTRESET	Software reset 0x0: Normal mode (no reset applied) 0x1: Software reset activated	RW	0
0	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0

Table 12−79. SDRC_SYSSTATUS

Address Offset	0x014		
Physical Address	0x6800 9014	**Instance**	SDRC1
Description	This register provides status 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	Reserved for module-specific status information Read returns 0.	R	0x000000
7:1	Reserved	Reserved for OCP socket status information Read returns 0.	R	0x00
0	RESETDONE	Internal reset monitoring 0x0: Internal module reset is ongoing. 0x1: Reset completed—the module is ready to be used.[1]	R	–

[1] During reset, the value is 0, but the value read just after the reset is 1, because ICLK/FCLK is already operating.

Table 12−80. SDRC_CS_CFG

Address Offset	0x0040		
Physical Address	0x6800 9040	**Instance**	SDRC1
Description	This register configures the start address of CS1 address space. Must be aligned on a boundary that is a multiple of the size of the attached memory or of the next power of two, if the memory size is not a power of two.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																						CS1STARTLOW		Reserved				CS1STARTHIGH			

Bits	Field Name	Description	Type	Reset
31:10	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000000
9:8	CS1STARTLOW	CS1 address space start address (lower add bits a1:a0)/32MB unit	RW	0x0

Bits	Field Name	Description	Type	Reset
7:4	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
3:0	CS1STARTHIGH	CS1 address space start address (upper add bits a5:a4:a3:a2)/128MB unit	RW	0x4

Table 12–81. SDRC_SHARING

Address Offset	0x0044		
Physical Address	0x6800 9044	**Instance**	SDRC1
Description	This register controls the GPMC access with respect to the SDRC, in a pin-sharing 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								Reserved									CS1MUXCFG			CS0MUXCFG			SDRCTRISTATE	Reserved							

Bits	Field Name	Description	Type	Reset
31:15	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
14:12	CS1MUXCFG	Identifies the SDRC pins used by CS1 0x0: 32-bit SDRAM on data lane[31:0] 0x1: 32-bit SDRAM on data lane[31:0] 0x2: 16-bit SDRAM on data lane[31:16] 0x3: 16-bit SDRAM on data lane[16:0] 0x4: 32-bit SDRAM on data lane[47:16] 32-bit stacked DDR SDRAM on data lane[31:0] in the stacked version of the OMAP2420 0x5: 32-bit SDRAM on data lane[47:16] 32-bit stacked DDR SDRAM on data lane[31:0] in the stacked version of the OMAP2420 0x6: 16-bit SDRAM on data lane[47:32] 0x7: 16-bit SDRAM on data lane[31:16]	RW	0x2
11:9	CS0MUXCFG	Identifies the SDRC pins used by CS0 0x0: 32-bit SDRAM on data lane[31:0] 0x1: 32-bit SDRAM on data lane[31:0] 0x2: 16-bit SDRAM on data lane[31:16] 0x3: 16-bit SDRAM on data lane[16:0] 0x4: 32-bit SDRAM on data lane[47:16] 32-bit stacked DDR SDRAM on data lane[31:0] in the stacked version of the OMAP2420 0x5: 32-bit SDRAM on data lane[47:16] 32-bit stacked DDR SDRAM on data lane[31:0] in the stacked version of the OMAP2420 0x6: 16-bit SDRAM on data lane[47:32] 0x7: 16-bit SDRAM on data lane[31:16]	RW	0x3

Table 12−81. SDRC_SHARING (Continued)

Bits	Field Name	Description	Type	Reset
8	SDRCTRISTATE	Static 3-state command for the SDRC I/O pads 0x0: All SDRC interface pins are tristated. 0x1: Normal mode: SDRC drives the I/O pads based on memory traffic.	RW	1
7:0	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00

Table 12−82. SDRC_ERR_ADDR

Address Offset	0x048		
Physical Address	0x6800 9048	**Instance**	SDRC1
Description	This register captures the address of the last illegal access received on the OCP interface.		
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	ERROR ADDRESS	Address of illegal access	R	0x00000000

Table 12−83. SDRC_ERR_TYPE

Address Offset	0x04C		
Physical Address	0x6800 904C	**Instance**	SDRC1
Description	This register provides additional information about the last illegal access.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				ERRORCONNID				Reserved	ERRORCOMD			ERRORADD		ERRORDPD	ERRORVALID

Bits	Field Name	Description	Type	Reset
31:12	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000
11:8	ERRORCONNID	Identifies the illegal access initiator. OCP ConnID of the illegal access initiator: See the top-level documentation of the device using the SDRC module.	R	0x0
7	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
6:4	ERRORMCMD	OCP command of the transaction that caused the error (3-bit field)	R	0x0

Table 12–83. SDRC_ERR_TYPE (Continued)

Bits	Field Name	Description	Type	Reset
3:2	ERRORADD	Flag indicating that access is to an illegal address. 0x0: The OCP access was to an address outside the memory space (MADDRSPACE = 1). 0x1: The OCP access was to an address outside the register space (MADDRSPACE = 0). 0x2: No Err Add. Not an address error. 0x3: The PASR field was set to a value other than All bank. The access was to a memory location that is not refreshed (MADDRSPACE = 0).	R	0x2
1	ERRORDPD	Transaction error while the memory is in deep power-down mode 0x0: The memory was not in deep power-down mode when the error occurred. 0x1: The error was caused by an unexpected access while the memory was in deep power-down mode.	R	0
0	ERRORVALID	Error validity status—must be explicitly cleared with a write transaction 0x0: All error fields no longer valid 0x1: Error detected and logged in the other error fields	RW	0

Table 12–84. SDRC_DLLA_CTRL

Address Offset	0x0060		
Physical Address	0x6800 9060	**Instance**	SDRC1
Description	This register controls the SDRC DLL A resource used for fine-timing tuning on a double-data-rate 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																DELAY								Reserved				ENADLL	LOADDLL	DLLPHASE	TESTDLLENA

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000
15:8	DELAY	8-bit delay; valid range 0..224 one step = 22 ps +/– 8.5 ps DELAY[1:0] must be set to 0x3.	RW	0x00
7:4	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
3	ENADLL	Enables DLL 0x0 DLL disabled 0x1 DLL enabled	RW	0
2	LOADDLL	Selects the DLL functionality between lock or unlock mode. Setting 1 to this bit loads the DELAY field value into the DLL module and puts the DLL in unlock mode (assert DELAY value permanently). Writing 0 to this bit puts the DLL in lock mode (tracking counter started from last DELAY value asserted). 0x0: LOADDLL at 0 puts the DLL in lock mode (tracking counter started). 0x1: LOADDLL at 1 puts the DLL in unlock mode (asserts DELAY value permanently).	RW	0
1	DLLPHASE	Nominal digitally controlled delay when DLL is enabled 0x0: 72 degrees (20% of the clock period) 0x1: 90 degrees (25% of the clock cycle)—90 degrees must be used	RW	0
0	TESTDLLENA	Enables DLL test (used only for test) 0x0: DLL test disabled 0x1: DLL test enabled	RW	0

Table 12–85. SDRC_DLLA_STATUS

Address Offset	0x064		
Physical Address	0x6800 9064	**Instance**	SDRC1
Description	This register reflects the current status of the DLL A.		
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																DLLCNT								Reserved					LOCK	UDF	OVF

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Read returns 0.	R	0x0000
15:8	DLLCNT	Current DLL counter value for monitoring / debug	R	0x00
7:3	Reserved	Read returns 0.	R	0x00
2	LOCK	DLL lock status—reserved for future use 0x0: The DLL is not locked. 0x1: The DLL is locked.	R	0
1	UDF	DLL counter underflow status 0x0: No underflow 0x1: The DLL counter underflowed (< 0).	R	0
0	OVF	DLL counter overflow status 0x0: No overflow 0x1: The DLL counter overflowed (> 224).	R	0

Table 12–86. SDRC_DLLB_CTRL

Address Offset	0x0068		
Physical Address	0x6800 9068	**Instance**	SDRC1
Description	This register controls the SDRC DLL B resource used for fine-timing tuning on a double-data-rate 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																DELAY								Reserved				ENADLL	LOADDLL	DLLPHASE	TESTDLLENA

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000
15:8	DELAY	8-bit delay; valid range 0..224 one step = 22 ps +/– 8.5 ps. Delay[1:0] must be set to 0x3.	RW	0x00
7:4	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
3	ENADLL	Enables DLL 0x0: DLL disabled 0x1: DLL enabled	RW	0
2	LOADDLL	Selects the DLL functionality between lock or unlock mode. Setting this bit to 1 loads the DELAY field value into the DLL module and puts the DLL in unlock mode (assert DELAY value permanently). Writing 0 to this bit puts the DLL in lock mode (tracking counter started from last DELAY value asserted). 0x0: LoadDLL at 0 puts the DLL in lock mode (tracking counter started). 0x1: LoadDLL at 1 puts the DLL in unlock mode (assert DELAY value permanently).	RW	0
1	DLLPHASE	Nominal digitally controlled delay when DLL is enabled 0x0: 72 degrees (20% of the clock period) 0x1: 90 degrees (25% of the clock cycle)—90degrees must be used	RW	0
0	TESTDLLENA	Enables DLL test (used only for test) 0x0: DLL test disabled 0x1: DLL test enabled	RW	0

Table 12–87. SDRC_DLLB_STATUS

Address Offset	0x06C		
Physical address	0x6800 906C	**Instance**	SDRC1
Description	This register reflects the current status of the DLL B.		
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																DLLCNT								Reserved					LOCK	UDF	OVF

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Read returns 0.	R	0x0000
15:8	DLLCNT	Current DLL counter value for monitoring/debug	R	0x00
7:3	Reserved	Read returns 0.	R	0x00
2	LOCK	DLL lock status—reserved for future use 0x0: The DLL is not locked. 0x1: The DLL is locked.	R	0
1	UDF	DLL counter underflow status 0x0: No underflow 0x1: The DLL counter underflowed (< 0).	R	0
0	OVF	DLL counter overflow status 0x0: No overflow 0x1: The DLL counter overflowed (> 224).	R	0

Table 12–88. SDRC_POWER

Address Offset	0x070		
Physical Address	0x6800 9070	**Instance**	SDRC1
Description	The SDRC power management register defines the global power management policy (shared by CS0/CS1).		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								AUTOCOUNT																SRFONRESET	SRFRONIDLEREQ	CLKCTRL		EXTCLKDIS	PWDENA	Reserved	PAGEPOLICY

Bits	Field Name	Description	Type	Reset
31:24	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
23:8	AUTOCOUNT	16-bit programmable count value used for delayed automatic clock gating and self-refresh entry, assuming CLKCTRL field is not 0.	RW	0x0000
7	SRFRONRESET	Enter self-refresh when in reset mode. 0x0: Feature disabled 0x1: Feature enabled	RW	1
6	SRFRONIDLEREQ	Enter self-refresh when on hardware idle request. 0x0: Feature disabled 0x1: Feature enabled	RW	0
5:4	CLKCTRL	Clock control feature defines clock gating and self-refresh. 0x0: No auto clk feature turned on 0x1: Enable internal clk gating on time-out of Auto_cnt. 0x2: Enable self-refresh on time-out of Auto_cnt. 0x3: Reserved	RW	0x0
3	EXTCLKDIS	Disable the clock provided to the external or stacked memories. 0x0: Enable clock 0x1: Disable clock—logical 0 is applied.	RW	0

Table 12–88. SDRC_POWER (Continued)

Bits	Field Name	Description	Type	Reset
2	PWDENA	Activate power-down mode of the target memory through the CKE pin. 0x0: Power-down mode feature disabled 0x1: Power-down mode feature enabled	RW	1
1	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
0	PAGEPOLICY	Page/segment closure policy with respect to power versus bandwidth trade-off 0x1: High-power/high-bandwidth mode (HPHB)	RW	1

CAUTION

The power-down feature (SDRC_POWER[2] PWDENA bit) must be disabled just before the deep power-down exit mode and reenabled after the subsequent memory initialization.

Table 12–89. SDRC_MCFG_0

Address Offset	0x0080		
Physical address	0x6800 9080	**Instance**	SDRC1
Description	This register provides the memory 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							ADDRMUX					Reserved		RAMSIZE										Reserved			B32NOT16	DEEPPD	DDRTYPE	RAMTYPE	

Bits	Field Name	Description	Type	Reset
31:25	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
24:20	ADDRMUX	Address multiplexing scheme (see Section 12.8.2.3, *Address Multiplexing*) 0x0: Address mux scheme 1 0x1: Address mux scheme 2 0x2: Address mux scheme 3 0x3: Address mux scheme 4 0x4: Address mux scheme 5 0x5: Address mux scheme 6 0x6: Address mux scheme 7 0x7: Address mux scheme 8 0x8: Address mux scheme 9 0x9: Address mux scheme 10 0xA: Address mux scheme 11 0xB: Address mux scheme 12	RW	0x03

Table 12−89. SDRC_MCFG_0 (Continued)

Bits	Field Name	Description	Type	Reset
24:20	ADDRMUX	0xC: Address mux scheme 13 0xD: Address mux scheme 14 0xE: Address mux scheme 15 0xF: Address mux scheme 16 0x10: Reserved 0x11: Reserved 0x12: Reserved 0x13: Reserved 0x14: Reserved 0x15: Reserved 0x16: Reserved 0x17: Address mux scheme 24 0x18: Reserved 0x19 Address mux scheme 26 0x1A Address mux scheme 27 0x1B Address mux scheme 28 0x1C Address mux scheme 29 0x1D Undefined 0x1E Undefined 0x1F Undefined		
19:18	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
17:8	RAMSIZE	RAM address space size number of 2MB chunks	RW	0x000
7:5	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
4	B32NOT16	External SDRAM bus width 0x0: External SDRAM device is x16 bits. 0x1: External SDRAM device is x32 bits.	RW	0
3	DEEPPD	Indicates if the memory supports deep power-down mode 0x0: No deep power-down mode support 0x1: Memory supports deep power-down mode.	RW	0
2	DDRTYPE	DDR memory type (assuming RAM type = 01) 0x0: Mobile DDR 0x1: Reserved for future use	RW	0
1:0	RAMTYPE	Memory type 0x0: SDR−SDRAM (single-data rate) 0x1: DDR−SDRAM (double-data rate) 0x2: Reserved 0x3: Reserved	RW	0x0

Table 12–90. SDRC_MR_0

Address Offset	0x084		
Physical Address	0x6800 9084	**Instance**	SDRC1
Description	This 12-bit register corresponds to the traditional SDRAM MR register with standard bit fields. All 12 bits are loaded into memory; thus, future extensions can be supported. The SDRC keeps an internal copy register that can be used internally and returned when a read access is performed at that address. Load into memory on OCP write access using MRS command with BA!,BA0 = 0,0.		
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																				ZERO_1		MBST	ZERO_0		CASL			SIL	BL		

Bits	Field Name	Description	Type	Reset
31:12	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000
11:10	ZERO_1	Write 0s, as required by memory specifications. Read returns 0.	RW	0x0
9	WBST	Write burst support Not used in the stacked version of the OMAP2420 0x0: Write burst equals read burst. 0x1: Write burst disable (single-write access only)	RW	0
8:7	ZERO_0	Write 0s, as required by memory specifications. Read returns 0.	RW	0x0
6:4	CASL	CAS latency as defined in clock periods 0x1: CAS latency = 1 0x2: CAS latency = 2 0x3: CAS latency = 3 0x4: CAS latency = 4 0x5: CAS latency = 5	RW	0x2
3	SIL	Serial or interleaved mode 0x0: Serial mode (always used) 0x1: Interleaved mode (never used)	RW	0
2:0	BL	Memory burst length 0x0: Burst length = 1—Not supported 0x1: Burst length = 2—SDR memory only 0x2: Burst length = 4— DDR memory only 0x3: Burst length = 8—Not supported 0x4: Reserved 0x5: Reserved 0x6: Reserved 0x7: Full page—Not supported	RW	0x4

Table 12−91. SDRC_EMR1_0

Address Offset	0x088		
Physical Address	0x6800 9088	**Instance**	SDRC1
Description	This 12-bit register corresponds to the DDR1 EMR register, with standard bit fields. All 12 bits are loaded into memory; thus, future extensions can be supported. The SDRC keeps an internal copy register that can be used internally and returned when a read access is performed at that address. Load into memory on OCP write access using MRS command w/ BA1,BA0 = 0,1.		
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																				ZERO									QFC	DS	DLL

Bits	Field Name	Description	Type	Reset
31:12	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000
11:3	ZERO	Write 0s, as required by memory specifications. Read returns 0.	RW	0x000
2	QFC	QFC enable bit 0x0: Disabled 0x1: Enabled	RW	0
1	DS	DS driver strength bit 0x0: Normal 0x1: Reduced	RW	0
0	DLL	DLL enable bit 0x0: DLL enabled 0x1: DLL disabled	RW	0

Table 12−92. SDRC_EMR2_0

Address Offset	0x008C		
Physical Address	0x6800 908C	**Instance**	SDRC1
Description	This register corresponds to the low-power EMR register, as defined in the mobile DDR JEDEC standard. It is a 12-bit register, and all 12 bits are loaded into memory; thus, future extensions can be supported. The SDRC keeps an internal copy register, used internally, that is returned when a read access is performed at that address. Load into memory on OCP write access, using MRS command w/ BA1,BA0 = 1,0		
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																				ZERO_2		ATCS-R	ZERO_1	DS		TCSR		PASR			

Table 12–92. SDRC_EMR2_0 (Continued)

Bits	Field Name	Description	Type	Reset
31:12	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000
11:10	ZERO_2	Write 0s, as required by memory specifications. Read returns 0.	RW	0x0
9	ATCSR	Advanced temperature compensated self-refresh enable bit Used only in the stacked version of the OMAP2420 0x0: Enabled 0x1: Reserved (ATCSR disabled, not supported)	RW	0
8:7	ZERO_1	Write 0s, as required by memory specifications. Read returns 0.	RW	0x0
6:5	DS	Driver strength 0x0: Full-strength driver Normal-strength driver in the stacked version of the OMAP2420 0x1: Weak-strength driver 1/2-strength driver in the stacked version of the OMAP2420 0x2: 1/4-strength driver Used only in the stacked version of the OMAP2420 0x3: 1/8-strength driver Used only in the stacked version of the OMAP2420	RW	0x0
4:3	TCSR	Temperature -compensated self-refresh Not used in the stacked version of the OMAP2420 0x0: 70 degrees C maximum temperature 0x1: 45 degrees C maximum temperature 0x2: 15 degrees C max temperature 0x3: 85 degrees C maximum temperature	RW	0x0
2:0	PASR	Partial array self-refresh 0x0: All banks 0x1: 1/2-array 0x2: 1/4-array 0x3: Reserved 0x4: Reserved 0x5: 1/8-array Not used in the stacked version of the OMAP2420 0x6: 1/16-array Not used in the stacked version of the OMAP2420 0x7: Reserved	RW	0x0

Table 12−93. SDRC_EMR3_0

Address Offset	0x090		
Physical Address	0x6800 9090	**Instance**	SDRC1
Description	This 12-bit register is not yet implemented in an existing device. All 12 bits are loaded into memory for future extensions. The SDRC keeps an internal copy register that can be used internally and returned when a read access is performed at that address. Load into memory on OCP write access using MRS command with BA1,BA0 = 1,1.		
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																				FUTUREUSE											

Bits	Field Name	Description	Type	Reset
31:12	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000
11:0	FUTUREUSE	Write 0s. Not used by current memories. Read returns 0.	RW	0x000

Table 12−94. SDRC_DCDL1_CTRL

Address Offset	0x094		
Physical Address	0x6800 9094	**Instance**	SDRC1
Description	This register controls all DCDL delay adjustments.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		OFFSET2						Reserved		OFFSET1						Reserved		OFFSET0						Reserved							

Bits	Field Name	Description	Type	Reset
31:30	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
29:24	OFFSET2	One MSB bit defines whether OFFSET2 must be added or subtracted from the DQS2 nominal value for DQS2 adjustment in read mode. 0x0: DQS2/DCDL is subtracted. 0x1: DQS2/DCDL is added. DQS2/DCDL offset adjustment (step 22ps) 5 lower bits for unsigned value 0x1F: +31 0x1E: +30 … 0x01: +1 0x00: 0 0x20: 0 0x21: −1 … 0x3E: −30 0x3F: −31	RW	0x00

Table 12–94. SDRC_DCDL1_CTRL (Continued)

Bits	Field Name	Description	Type	Reset
23:22	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
21:16	OFFSET1	One MSB bit defines whether OFFSET1 must be added or subtracted from the DQS1 nominal value for DQS1 adjustment in read mode. Not used in the stacked version of the OMAP2420 0x0: DQS1/DCDL is subtracted. 0x1: DQS1/DCDL is added. DQS1/DCDL offset adjustment (step 22ps) 5 lower bits for unsigned value 0x1F: +31 0x1E: +30 … 0x01: +1 0x00: 0 0x20: 0 0x21: −1 … 0x3E: −30 0x3F: −31	RW	0x00
15:14	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
13:8	OFFSET0	One MSB bit defines whether OFFSET0 must be added or subtracted from the DQS0 nominal value for DQS0 adjustment in read mode. Not used in the stacked version of the OMAP2420 0x0: DQS0/DCDL is subtracted. 0x1: DQS0/DCDL is added. DQS0/DCDL offset adjustment (step 22ps) 5 lower bits for unsigned value 0x1F: +31 0x1E: +30 … 0x01: +1 0x00: 0 0x20: 0 0x21: −1 … 0x3E: −30 0x3F: −31	RW	0x00
7:0	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00

Table 12−95. SDRC_DCDL2_CTRL

Address Offset	0x098		
Physical Address	0x6800 9098	**Instance**	SDRC1
Description	This register controls all DCDL delay adjustments.		
Type	RW		

31 30	29 28 27 26 25 24	23 22	21 20 19 18 17 16	15 14	13 12 11 10 9 8	7 6	5 4 3 2 1 0
Reserved	OFFSET6	Reserved	OFFSET5	Reserved	OFFSET4	Reserved	OFFSET3

Bits	Field Name	Description	Type	Reset
31:30	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
29:24	OFFSET6	One MSB bit defines whether OFFSET6 must be added or subtracted from the DQS nominal value for DQS adjustment in write mode. 0x0: DQS/DCDL is subtracted. 0x1: DQS/DCDL is added. Write DCDL offset adjustment (step 22ps) 5 lower bits for unsigned value 0x1F: +31 0x1E: +30 … 0x01: +1 0x00: 0 0x20: 0 0x21: −1 … 0x3E: −30 0x3F: −31	RW	0x00
23:22	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0

Table 12−95. SDRC_DCDL2_CTRL (Continued)

Bits	Field Name	Description	Type	Reset
21:16	OFFSET5	One MSB bit defines whether OFFSET5 must be added or subtracted from the DQS5 nominal value for DQS5 adjustment in read mode. Used only in the stacked version of the OMAP2420 0x0: DQS5/DCDL is subtracted. 0x1: DQS5/DCDL is added. DQS5/DCDL offset adjustment (step 22ps) 5 lower bits for unsigned value 0x1F: +31 0x1E: +30 … 0x01: +1 0x00: 0 0x20: 0 0x21: −1 … 0x3E: −30 0x3F: −31	RW	0x00
15:14	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
13:8	OFFSET4	One MSB bit defines whether OFFSET4 must be added or subtracted from the DQS4 nominal value for DQS4 adjustment in write mode. Used only in the stacked version of the OMAP2420 0x0: DQS4/DCDL is subtracted. 0x1: DQS4/DCDL is added. DQS4/DCDL offset adjustment (step 22ps) 5 lower bits for unsigned value 0x1F: +31 0x1E: +30 … 0x01: +1 0x00: 0 0x20: 0 0x21: −1 … 0x3E: −30 0x3F: −31	RW	0x00

Table 12−95. SDRC_DCDL2_CTRL (Continued)

Bits	Field Name	Description	Type	Reset
7:6	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
5:0	OFFSET3	One MSB bit defines whether OFFSET3 must be added or subtracted from the DQS3 nominal value for DQS3 adjustment in read mode. 0x0: DQS3/DCDL is subtracted. 0x1: DQS3/DCDL is added. DQS3/DCDL offset adjustment (step 22ps) 5 lower bits for unsigned value 0x1F: +31 0x1E: +30 … 0x01: +1 0x00: 0 0x20: 0 0x21: −1 … 0x3E: −30 0x3F: −31	RW	0x00

Table 12−96. SDRC_ACTIM_CTRLA_0

Address Offset	0x09C		
Physical Address	0x6800 909C	**Instance**	SDRC1
Description	The ac parameter timing control register sets the ac parameter values in the clock cycle units to best match memory characteristics.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRFC					TRC					TRAS				TRP			TRCD			TRRD			TDPL			Reserved	TDAL				

Bits	Field Name	Description	Type	Reset
31:27	TRFC	Autorefresh-to-active	RW	0x00
26:22	TRC	Row cycle time	RW	0x00
21:18	TRAS	Row active time	RW	0x0
17:15	TRP	Row precharge time	RW	0x0
14:12	TRCD	Row-to-column delay time	RW	0x0
11:9	TRRD	Active-to-active command period	RW	0x0
8:6	TDPL	Data-in-to-precharge command (write recovery time tWR)	RW	0x0

Table 12–96. SDRC_ACTIM_CTRLA_0 (Continued)

Bits	Field Name	Description	Type	Reset
5	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
4:0	TDAL	Data-in-to-active command	RW	0x00

Table 12–97. SDRC_ACTIM_CTRLB_0

Address Offset	0x0A0		
Physical Address	0x6800 90A0	**Instance**	SDRC1
Description	The XSR parameter-timing control register sets the ac parameter values in clock-cycle units to best match memory characteristics.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								XSR							

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x000000
7:0	XSR	Self-refresh exit-to-active period	RW	0x00

Table 12–98. SDRC_RFR_CTRL_0

Address Offset	0x00A4		
Physical Address	0x6800 90A4	**Instance**	SDRC1
Description	SDRAM memory autorefresh 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								ARCV																Reserved						ARE	

Bits	Field Name	Description	Type	Reset
31:24	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
23:8	ARCV	Autorefresh counter value, to set the refresh period >. (Refresh interval [†]/clock period) − 50	RW	0x0000
7:2	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
1:0	ARE	Autorefresh enable 0x0: Autorefresh is disabled. 0x1: Counter is loaded with ARCV: 1 autorefresh command when autorefresh counter reaches 0. 0x2: Counter is loaded with 4xARCV: Burst of four autorefresh commands when autorefresh counter reaches 0. 0x3: Counter is loaded with 8xARCV: Burst of 8 autorefresh commands when autorefresh counter reaches 0.	RW	0x0

[†] Refresh interval in time unit.

Table 12−99. SDRC_MANUAL_0

Address Offset	0x0A8		
Physical Address	0x6800 90A8	**Instance**	SDRC1
Description	This register allows sending specific commands to external memory devices under software control. A write to this register generates the appropriate sequence based on the command code.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDPARAM																Reserved												CMDCODE			

Bits	Field Name	Description	Type	Reset
31:16	CMDPARAM	Manual command parameter, if any	RW	0x0000
15:4	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x000
3:0	CMDCODE	Memory command opcode. Other enums reserved for future implementations. 0x0: NOP command—no parameter 0x1: Precharge all command—no parameter 0x2: Autorefresh command—no parameter 0x3: Enter deep power-down—no parameter 0x4: Exit deep power-down—no parameter 0x5: Enter self-refresh—no parameter 0x6: Exit self-refresh—no parameter 0x7: Set CKE signal high—no parameter 0x8: Set CKE low—no parameter 0x9: Reserved 0xA: Reserved 0xB: Reserved 0xC: Reserved	RW	0x0

Table 12−100. SDRC_MCFG_1

Address Offset	0x00B0		
Physical Address	0x6800 90B0	**Instance**	SDRC1
Description	This register provides the memory 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							ADDRMUX					Reserved		RAMSIZE										Reserved			B32NOT16	DEEPPD	DDRTYPE	RAMTYPE	

Table 12−100. SDRC_MCFG_1 (Continued)

Bits	Field Name	Description	Type	Reset
31:25	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
24:20	ADDRMUX	Address multiplexing scheme (see Section 12.8.2.3, *Address Multiplexing*) 0x0: Address mux scheme 1 0x1: Address mux scheme 2 0x2: Address mux scheme 3 0x3: Address mux scheme 4 0x4: Address mux scheme 5 0x5: Address mux scheme 6 0x6: Address mux scheme 7 0x7: Address mux scheme 8 0x8: Address mux scheme 9 0x9: Address mux scheme 10 0xA: Address mux scheme 11 0xB: Address mux scheme 12 0xC: Address mux scheme 13 0xD: Address mux scheme 14 0xE: Address mux scheme 15 0xF: Address mux scheme 16 0x10: Reserved 0x11: Reserved 0x12: Reserved 0x13: Reserved 0x14: Reserved 0x15: Reserved 0x16: Reserved 0x17: Address mux scheme 24 0x18: Reserved 0x19 Address mux scheme 26 0x1A Address mux scheme 27 0x1B Address mux scheme 28 0x1C Address mux scheme 29 0x1D Undefined 0x1E Undefined 0x1F Undefined	RW	0x03
19:18	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
17:8	RAMSIZE	RAM address space size Number of 2MB chunks	RW	0x000
7:5	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0

Table 12–111. SDRC_MCFG_1 (Continued)

Bits	Field Name	Description	Type	Reset
4	B32NOT16	External SDRAM bus width 0x0: External SDRAM device is x16 bits. 0x1: External SDRAM device is x32 bits.	RW	0
3	DEEPPD	Indicates memory support of deep power-down mode 0x0: No deep power-down mode support 0x1: Deep power-down mode support	RW	0
2	DDRTYPE	DDR memory type (assuming RAM type = 01) 0x0: Mobile DDR 0x1: Reserved for future use	RW	0
1:0	RAMTYPE	Memory type 0x0: SDR–SDRAM (single-data rate) 0x1: DDR–SDRAM (double-data rate) 0x2: Reserved 0x3: Reserved	RW	0x0

Table 12–101. SDRC_MR_1

Address Offset	0x0B4		
Physical Address	0x6800 90B4	**Instance**	SDRC1
Description	This 12-bit register corresponds to the traditional SDRAM MR register with standard bit fields. All 12 bits are loaded into memory; thus, future extensions can be supported. The SDRC keeps an internal copy register that can be used internally and returned when a read access is performed at that address. Load into memory on OCP write access using MRS command with BA!,BA0 = 0,0.		
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																				ZERO_0		MBST	ZERO_0		CASL			SIL	BL		

Bits	Field Name	Description	Type	Reset
31:12	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000
11:10	ZERO_1	Write 0s, as required by memory specifications. Read returns 0.	RW	0x0
9	WBST	Write burst support Not used in the stacked version of the OMAP2420 0x0: Write burst equals read burst. 0x1: Write burst disable (single-write access only)	RW	0
8:7	ZERO_0	Write 0s, as required by memory specifications. Read returns 0.	RW	0x0

Table 12−101. SDRC_MR_1 (Continued)

Bits	Field Name	Description	Type	Reset
6:4	CASL	CAS latency defined in clock periods 0x1: CAS latency = 1 0x2: CAS latency = 2 0x3: CAS latency = 3 0x4: CAS latency = 4 0x5: CAS latency = 5	RW	0x2
3	SIL	Serial or interleaved mode 0x0: Serial mode (always used) 0x1: Interleaved mode (never used)	RW	0
2:0	BL	Memory burst length 0x0: Burst length = 1—Not supported 0x1: Burst length = 2—SDR memory only 0x2: Burst length = 4—DDR memory only 0x3: Burst length = 8—Not supported 0x4: Reserved 0x5: Reserved 0x6: Reserved 0x7: Full page—Not supported	RW	0x4

Table 12−102. SDRC_EMR1_1

Address Offset	0x0B8		
Physical Address	0x6800 90B8	**Instance**	SDRC1
Description	This 12-bit register corresponds to the DDR1 EMR register with standard bit fields. All 12 bits are loaded in memory, thus supporting future extensions. The SDRC keeps an internal copy register that can be used internally and returned when a read access is performed at that address. Load into memory on OCP write access using MRS command w/ BA1,BA0 = 0,1.		
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																				ZERO									QFC	DS	DLL

Bits	Field Name	Description	Type	Reset
31:12	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000
11:3	ZERO	Write 0s, as required by memory specifications. Read returns 0.	RW	0x000

Table 12–102. SDRC_EMR1_1 (Continued)

Bits	Field Name	Description	Type	Reset
2	QFC	QFC Enable bit 0x0: Disabled 0x1: Enabled	RW	0
1	DS	DS driver strength bit 0x0: Normal 0x1: Reduced	RW	0
0	DLL	DLL enable bit 0x0: DLL enabled 0x1: DLL disabled	RW	0

Table 12–103. SDRC_EMR2_1

Address Offset	0x00BC		
Physical Address	0x6800 90BC	**Instance**	SDRC1
Description	This register corresponds to the low-power EMR register, as defined in the mobile DDR JEDEC standard. It is a 12-bit register. All 12 bits are loaded in memory, thus supporting future extensions The SDRC keeps an internal copy register, used internally, that is returned when a read access is performed at that address. Load into memory on OCP write access, using MRS command w/ BA1,BA0 = 1,0		
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																				ZERO_2		ATCS-R	ZERO_1		DS		TCSR		PASR		

Bits	Field Name	Description	Type	Reset
31:12	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000
11:10	ZERO_2	Write 0s, as required by memory specifications. Read returns 0.	RW	0x0
9	ATCSR	Advanced temperature-compensated self-refresh enable bit Used only in the stacked version of the OMAP2420 0x0: Enabled 0x1: Reserved (ATCSR disabled, not supported)	RW	0
8:7	ZERO_1	Write 0s, as required by memory specifications. Read returns 0.	RW	0x0

Table 12−103. SDRC_EMR2_1 (Continued)

Bits	Field Name	Description	Type	Reset
6:5	DS	Driver strength 0x0: Full-strength driver Normal-strength driver in the stacked version of the OMAP2420 0x1: Weak-strength driver 1/2-strength driver in the stacked version of the OMAP2420 0x2: 1/4-strength driver Used only in the stacked version of the OMAP2420 0x3: 1/8-strength driver Used only in the stacked version of the OMAP2420	RW	0x0
4:3	TCSR	Temperature -compensated self-refresh Not used in the stacked version of the OMAP2420 0x0: 70 degrees maximum temperature 0x1: 45 degrees maximum temperature 0x2: 15 degrees maximum temperature 0x3: 85 degrees maximum temperature	RW	0x0
2:0	PASR	Partial array self-refresh 0x0: All banks 0x1: 1/2 array 0x2: 1/4 array 0x3: Reserved 0x4: Reserved 0x5: 1/8 array Not used in the stacked version of the OMAP2420 0x6: 1/16 array Not used in the stacked version of the OMAP2420 0x7: Reserved	RW	0x0

Table 12–104. SDRC_EMR3_1

Address Offset 0x0C0

Physical Address 0x6800 90C0 **Instance** SDRC1

Description This 12-bit register is not yet implemented in an existing device. All 12 bits are loaded into memory for future extensions. The SDRC keeps an internal copy register that can be used internally and returned when a read access is performed at that address. Load into memory on OCP write access using MRS command with BA1,BA0 = 1,1.

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																				FUTUREUSE											

Bits	Field Name	Description	Type	Reset
31:12	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000
11:0	FUTUREUSE	Write 0s. Not used by current memories. Read returns 0.	RW	0x000

Table 12–105. SDRC_ACTIM_CTRLA_1

Address Offset 0x0C4

Physical Address 0x6800 90C4 **Instance** SDRC1

Description The ac parameter timing control register sets the ac parameter values in clock-cycle units to best match memory characteristics.

Type RW

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRFC					TRC					TRAS				TRP			TRCD			TRRD			TDPL			Reserved	TDAL				

Bits	Field Name	Description	Type	Reset
31:27	TRFC	Autorefresh-to-active	RW	0x00
26:22	TRC	Row cycle time	RW	0x00
21:18	TRAS	Row active time	RW	0x0
17:15	TRP	Row precharge time	RW	0x0
14:12	TRCD	Row-to-column delay time	RW	0x0
11:9	TRRD	Active-to-active command period	RW	0x0
8:6	TDPL	Data-in-to-precharge command (write recovery time TWR)	RW	0x0
5	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
4:0	TDAL	Data-in-to-active command	RW	0x00

Table 12–106. SDRC_ACTIM_CTRLB_1

Address Offset 0x0C8

Physical Address 0x6800 90C8 **Instance** SDRC1

Description	The XSR parameter timing-control register sets the ac parameter values in clock-cycle units to best match memory characteristics.
Type	RW

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="24">Reserved</td><td colspan="8">XSR</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x000000
7:0	XSR	Self-refresh exit-to-active period	RW	0x00

Table 12–107. SDRC_RFR_CTRL_1

Address Offset	0x00D4		
Physical Address	0x6800 90D4	**Instance**	SDRC1
Description	SDRAM memory autorefresh control		
Type	RW		

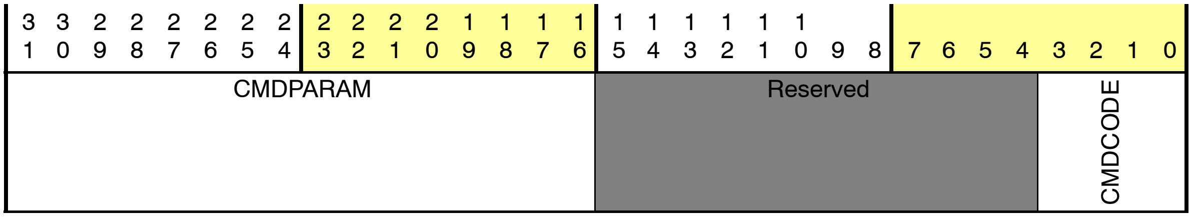
<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">Reserved</td><td colspan="16">ARCV</td><td colspan="6">Reserved</td><td colspan="2">ARE</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
31:24	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
23:8	ARCV	Autorefresh counter value to set the refresh period > (Refresh interval [†] / clock period) – 50	RW	0x0000
7:2	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
1:0	ARE	Autorefresh enable 0x0: Autorefresh is disabled. 0x1: Counter is loaded with ARCV: 1 autorefresh command when autorefresh counter reaches 0. 0x2: Counter is loaded with 4xARCV: Burst of four autorefresh commands when autorefresh counter reaches 0. 0x3: Counter is loaded with 8xARCV: Burst of eight autorefresh commands when autorefresh counter reaches 0.	RW	0x0

[†] Refresh interval in time unit

Table 12–108. SDRC_MANUAL_1

Address Offset	0x0D8		
Physical Address	0x6800 90D8	**Instance**	SDRC1
Description	This register allows sending specific commands to the external memory devices under software control. A write to this register generates the appropriate sequence, based on the command code.		
Type	RW		



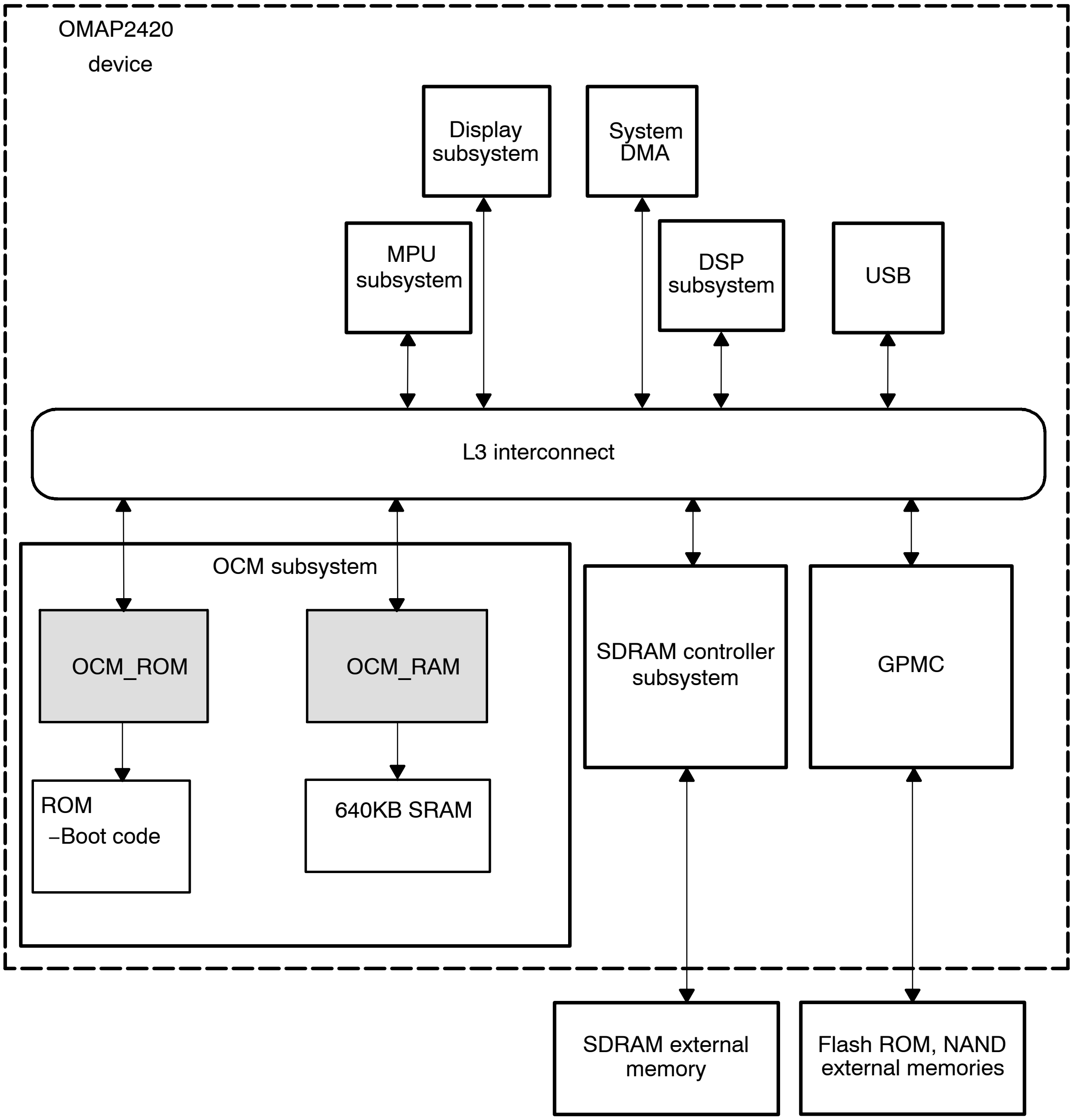
31:16	15:4	3:0
CMDPARAM	Reserved	CMDCODE

Bits	Field Name	Description	Type	Reset
31:16	CMDPARAM	Manual command parameter, if any	RW	0x0000
15:4	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x000
3:0	CMDCODE	Memory command opcode. Other enums reserved for future implementations. 0x0: NOP command—no parameter 0x1: Precharge command—no parameter 0x2: Autorefresh command—no parameter 0x3: Enter deep power-down—no parameter 0x4: Exit deep power-down—no parameter 0x5: Enter self-refresh—no parameter 0x6: Exit self-refresh—no parameter 0x7: Set CKE signal high—no parameter 0x8: Set CKE low—no parameter 0x9: Reserved 0xA: Reserved 0xB: Reserved 0xC: Reserved	RW	0x0

12.13 OCM Subsystem Overview

The on-chip memory subsystem consists of two separate on-chip memory controllers connected independently to an on-chip ROM and an on-chip RAM (the OCM_ROM and the OCM_RAM). Each memory controller has its own dedicated interface to the L3 interconnect.

Figure 12−41. OCM Subsystem Overview



Multiple L3 initiators (such as remote devices) have access to the RAM through, MPU subsystem, system DMA, display subsystem, DSP subsystem, and USB.

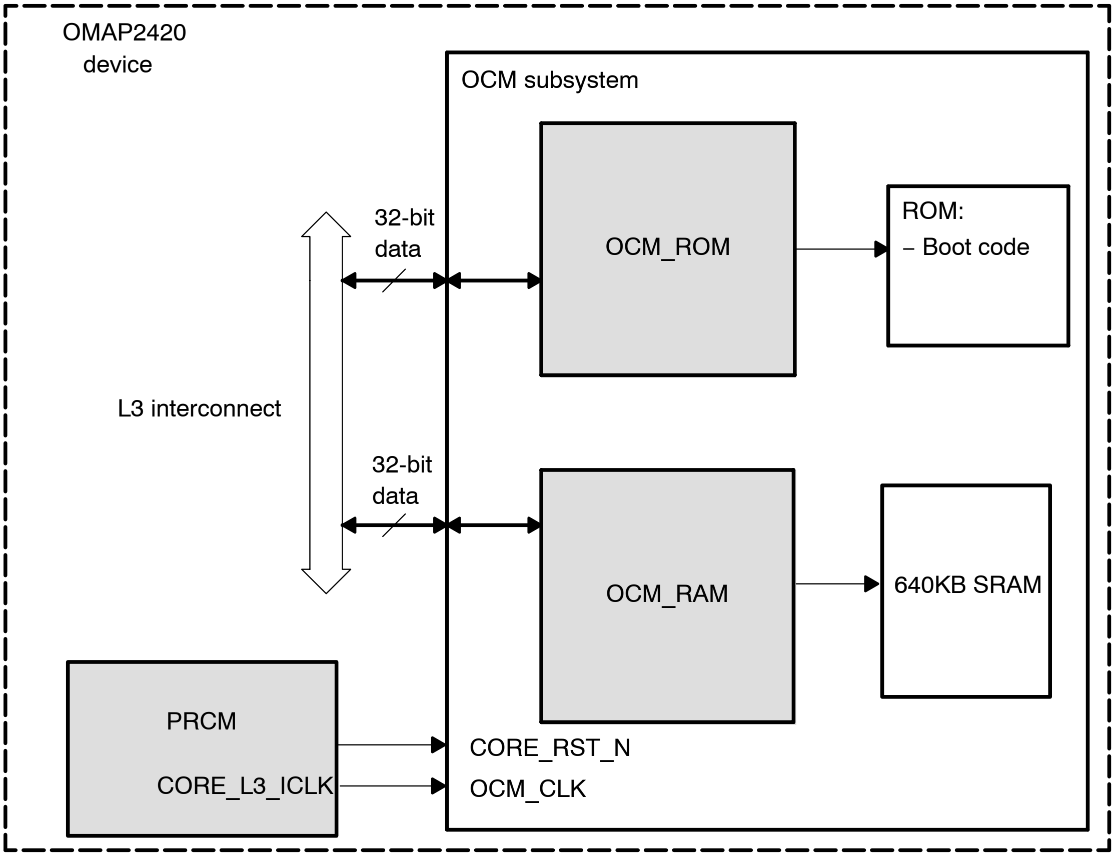
ROM is used for boot code.

12.14 OCM Subsystem Integration

12.14.1 Description

The OCM_ROM and OCM_RAM allow transactions between the system initiators and the multiple memories through the L3 interconnect.

Figure 12−42. OCM Subsystem Integration to the OMAP2420 Device



12.14.2 Clocking, Reset, and Power-Management Scheme

12.14.2.1 Clocking

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 functional clock for the OCM module.

For OCM, there is no register enabling the gating of OMC_CLK.

However, OCM does support the handshaking protocol with the PRCM module. OCM_CLK is gated if all modules (including the OCM) belonging to the L3 CLK domain send an SIdleAck back to the PRCM module after receiving the MIdleReq request.

For details, see Chapter 5, *Power, Reset, and Clock Management.*

The module performs automatic clock gating when the memory is not accessed by the system. There is no extra latency when the clock must be switched on after an idle state. The clock to the memory is dynamically gated.

12.14.2.2 Hardware Res

The module is globally reset by activating the CORE_RST_N signal in the CORE_RST domain (see Chapter 5, *Power, Reset, and Clock Management).*

12.14.2.3 Power Domain

OCM power is supplied by the CORE power domain (see Chapter 5, *Power Reset and Clock Management).*

12.15 OCM Subsystem Functional Description

12.15.1 OCM_ROM

The embedded ROM is used primarily for booting, flashing, and context restoring.

The OMAP2420 device embedded ROM (total 96K bytes) has the following characteristics:

- The public ROM is always accessible and contains the boot area.
- The OCM_ROM supports single and burst access transactions.
- The OCM_ROM operates at full interconnect clock frequency.

The memory space of the embedded ROM starts at 0x40000000 and ends at 0x4001FFFF.

12.15.2 OCM_RAM

The OCM_RAM contains 640K bytes of memory for the OMAP2420. OMAP2420 RAM is typically used as an on-chip frame buffer and as general-purpose SRAM. For example, 600K bytes of RAM memory can be used for a full 16 bpp VGA screen to be embedded.

The OMAP2420 RAM memory space starts at 0x40200000 and ends at 0x4029FFFF.

Chapter 13

Enhanced Audio Controller

This chapter describes the enhanced audio controller (EAC) in the OMAP2420 multimedia device.

Topic **Page**

13.1 EAC Overview

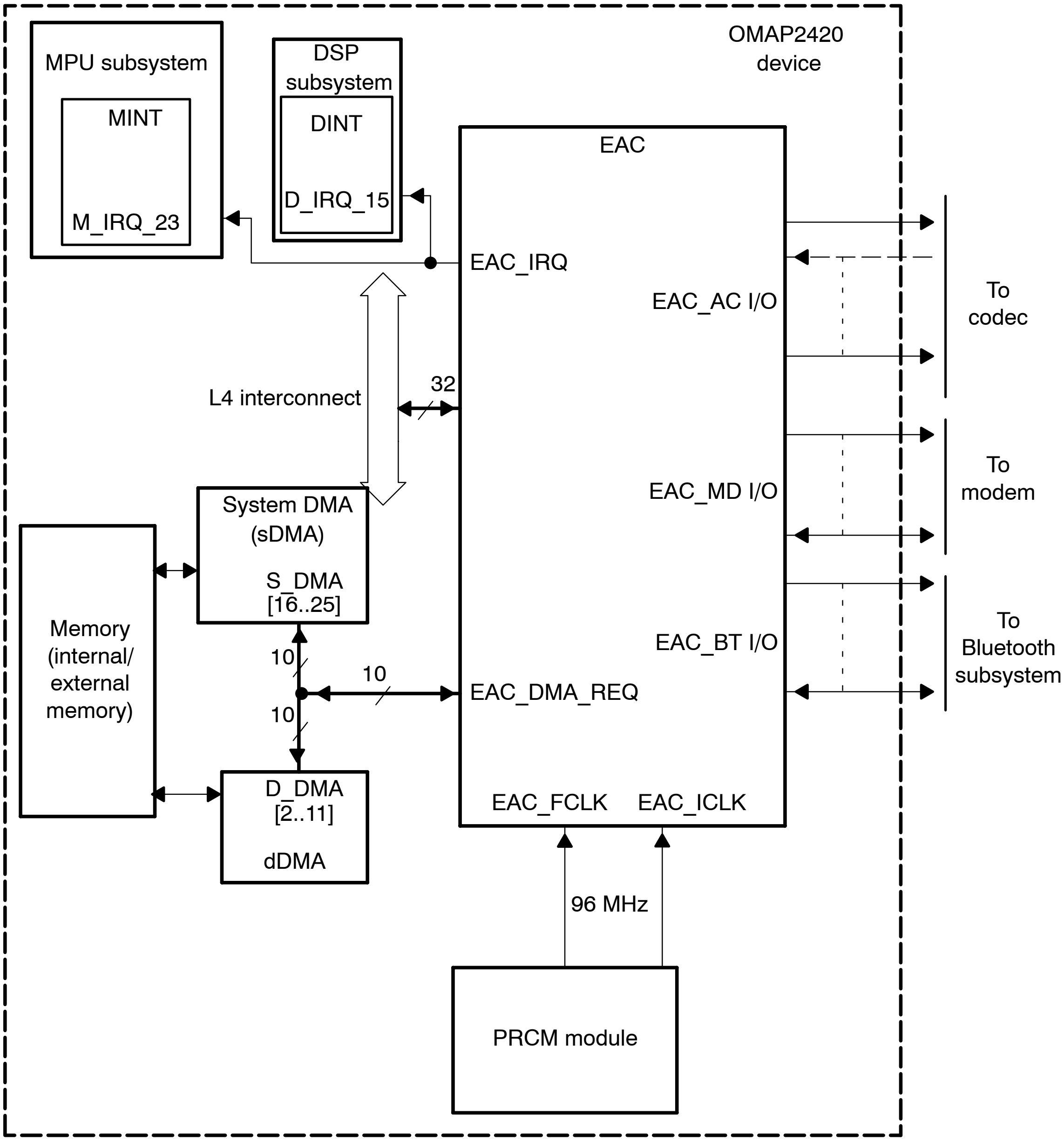
The enhanced audio controller (EAC) provides stand-alone audio and voice-stream processing without CPU (digital signal processor [DSP] or microprocessor unit [MPU]) support, including the following capabilities:

- ❏ Play and/or record pulse-code modulation (PCM) (.wav) files with various sampling rate frequencies and bit length formats
- ❏ Mix audio stream with voice stream at different sampling rates without CPU processing

The EAC provides connection of a modem and/or Bluetooth subsystem to any audio audio code/decode (codec) 1997 (AC97), inter-IC sound (I2S), or PCM codec while the MPU subsystem is in power-down mode. It eliminates the need for two codecs (one for the modem and one for high-quality audio).

Figure 13−1 is a general overview of the EAC.

Figure 13−1. EAC General Overview



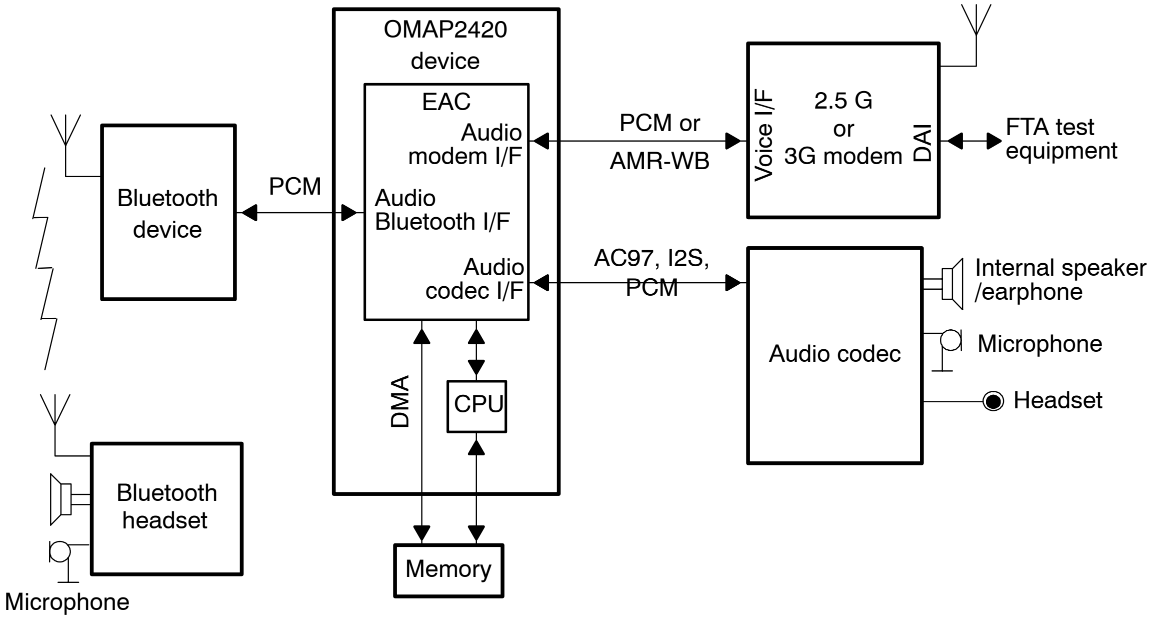
13.2 EAC Environment

The EAC is an audio-codec peripheral interface designed specifically for applications that require audio and voice data streaming. The EAC allows interconnection between several endpoints:

- ❏ Codec (audio, stereo, 44.1-/48-kHz sampling rates)
- ❏ Modem subsystem (voice, mono, 8-/16-kHz sampling rates)
- ❏ Bluetooth subsystem (voice, mono, 8-/16-kHz sampling rates)
- ❏ System memory using direct memory access (DMA) (audio files play/record, stereo, and various sampling rates)

Figure 13−2 shows the EAC environment.

Figure 13−2. EAC Environment



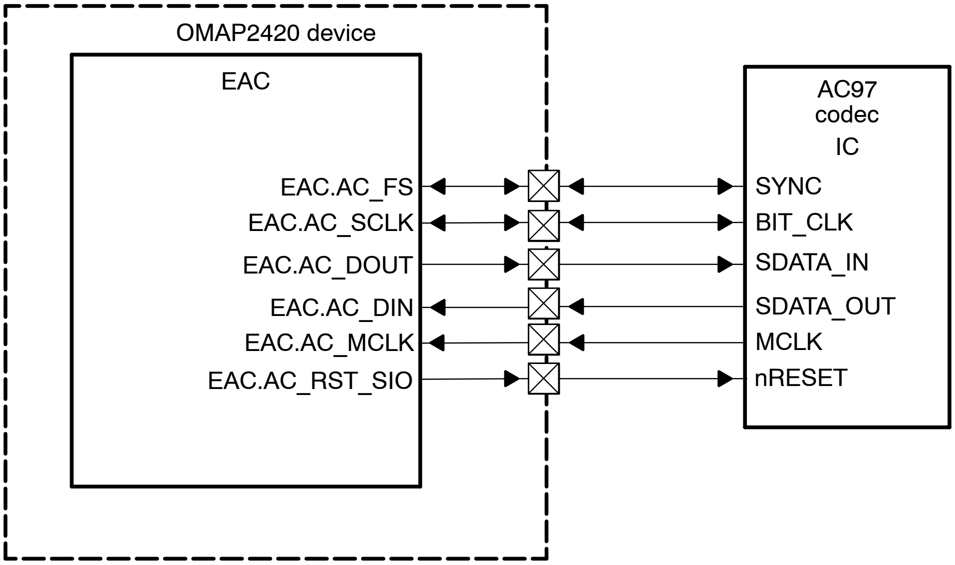
13.2.1 AC97 Codec Functional Interface

13.2.1.1 AC97 Codec Interface Description

In AC97 mode, the codec port interface can be configured as an ac link serial interface to the AC97 codec device. The ac link serial interface is a time-division multiplexed (TDM), slot-based serial interface used to transfer both audio data and command/status data to the codec device.

Figure 13−3 shows the EAC connected to an AC97 codec.

Figure 13−3. Connection of the EAC to an AC97 Codec



13.2.2 AC97 Codec Interface Description

Table 13−1 lists inputs and outputs (I/Os) for the AC97 codec port.

Table 13−1. AC97 Codec Port I/O Description

Signal Name	I/O[1]	Description	Codec Source/ Destination
EAC.AC_FS	I/O	Codec port interface frame synchronization	SYNC
EAC.AC_SCLK	I/O	Codec port interface serial clock	BIT_CLK
EAC.AC_DOUT	O	Codec port interface serial-data output	SDATA_IN
EAC.AC_DIN	I	Codec port interface serial-data input	SDATA_OUT
EAC.AC_MCLK	I	Master codec clock input	MCLK
EAC.AC_RST_SIO	O	Codec port interface reset output	nRESET

1) I = Input, O = Output

13.2.3 Protocol and Data Format

In this mode, the codec port interface is configured as a bidirectional full-duplex serial interface with a fixed rate of 48 kHz. Each 48-kHz frame is divided into 13 time slots, with the use of each time slot predefined by the *Audio Codec 97 Specification*, as shown in Table 13−2.

Table 13−2. AC97 Audio Frame

Time Slot	Out	In
0	Tag	Tag
1	Command address port	Status address port

Table 13−2. AC97 Audio Frame (Continued)

Time Slot	Out	In
2	Command data port	Status data port
3	PCM playback left channel	PCM record left channel
4	PCM playback right channel	PCM record right channel
5	Optional modem line 1 digital-to-audio converter (DAC)	Optional modem line 1 audio-to-digital converter (ADC)
6	Optional PCM center	Optional dedicated microphone record data
7	Optional PCM left surround	Reserved
8	Optional PCM right surround	Reserved
9	Optional low-frequency effect (LFE) DACs	Reserved
10	Optional modem line 2 DAC	Optional modem line 2 ADC
11	Optional modem headset DAC	Optional modem headset ADC
12	Optional modem general-purpose input-output (GPIO) control	Optional modem GPIO status

Each time slot is 20 serial clock cycles long except for time slot 0, which is only 16 serial clock cycles. The serial clock, which is referred to as the BIT_CLK for AC97 modes, is set to 12.288 MHz.

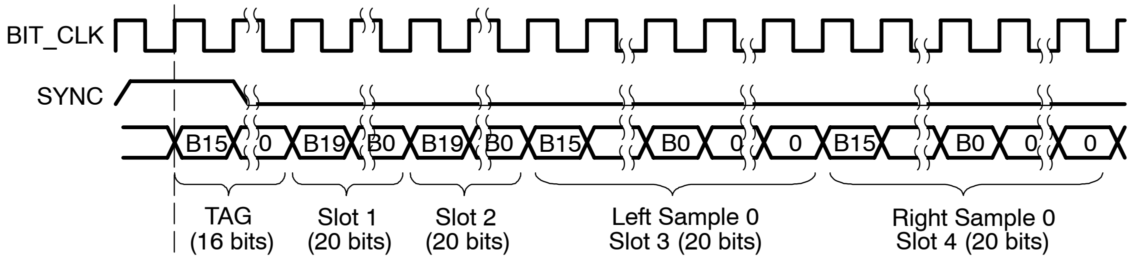
The BIT_CLK is derived from the codec master system clock (MCLK) input:

- If MCLK = 12.288 MHz, BIT_CLK = MCLK = 12.288 MHz.
- If MCLK = 24.576 MHz, BIT_CLK = MCLK/2 = 12.288 MHz.

SYNC is generated by dividing down the serial bit clock (BIT_CLK): 12.288 MHz/256 = 48 kHz.

Figure 13−4 shows the AC97 serial interface format.

Figure 13−4. AC97 Serial Interface Format



The beginning of all audio sample packets, or audio frames, transferred over the ac link is synchronized to the rising edge of the SYNC signal. Data are transitioned on the ac link on every rising edge of the BIT_CLK and are subsequently sampled on the receiving side of the ac link on each immediately following falling edge of the BIT_CLK.

☐ Tag phase

Portion of an audio frame where the SYNC signal is high, shifted one BIT_CLK forward. The tag phase lasts for the duration of time slot 0. The time slot 0 has only 16 bits. Each bit of time slot 0 conveys a valid tag for its corresponding time slot within the current audio frame (1 → valid data).

☐ Data phase

Portion of an audio frame where the SYNC signal is low, shifted one BIT_CLK forward. Each audio frame has 12 outgoing and 12 incoming data streams with 20-bit sample resolution.

Outgoing and incoming data streams are most-significant bit (MSB)-justified (MSB first). The unused least-significant bits (LSBs) are stuffed with 0s.

13.2.4 Audio Sample Exchange

The EAC codec port supports only stereo data; only slot 3 (left sample) and slot 4 (right sample) contain valid audio samples.

The codec port is directly interfaced to the audio-processing module with 16-bit resolution. Only the 16 MSBs of each audio sample are considered. Input sample LSBs are discarded; output sample LSBs are filled with zeros.

Note:

In AC97 mode, the frame frequency must be 48 kHz for data exchanges with the audio-processing module.

13.3 I2S Codec Functional Interface

I2S mode permits the codec port interface to be configured as an I2S-link serial interface to the I2S codec device. The I2S-link serial interface is a TDM slot-based serial interface that transfers audio data and command/status data to the codec device.

In the polarity-changed I2S mode, the codec port interface is configured as an I2S serial-link interface to an I2S codec device. The difference with I2S is the polarity of the frame-synchronization signal (EAC.AC_FS). In this mode, the first data (slot) is signaled high level.

Figure 13−5 shows the EAC connected to an I2S codec IC.

Figure 13−5. Connection of the EAC to an I2S Codec IC

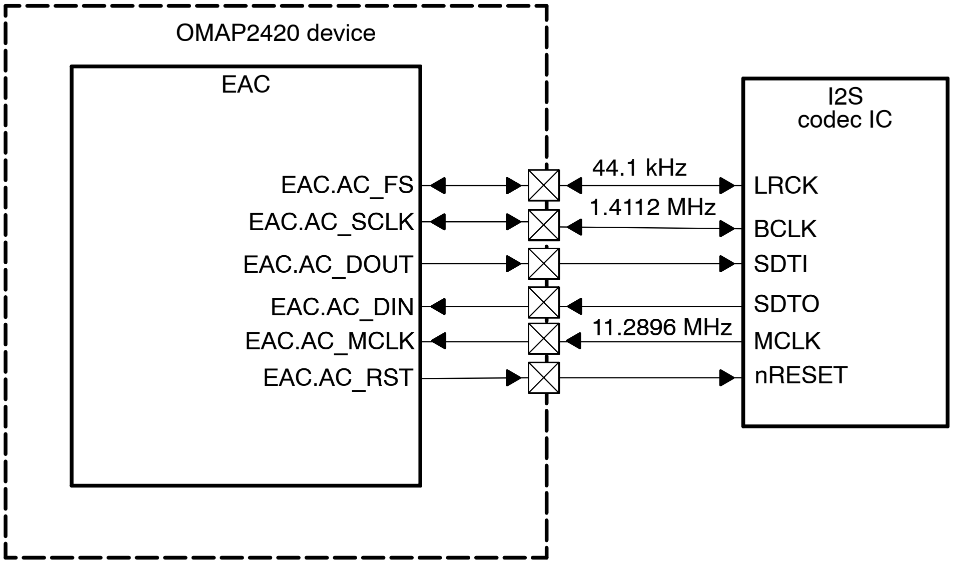


Table 13−3 lists I/O for the I2S codec port.

Table 13−3. I2S Codec Port I/O Description

Signal Name	I/O[1]	Description	Codec Source/ Destination
EAC.AC_FS	I/O	Codec port interface frame synchronization	LRCK
EAC.AC_SCLK	I/O	Codec port interface serial clock	BCLK
EAC.AC_DOUT	O	Codec port interface serial-data output	SDTI
EAC.AC_DIN	I	Codec port interface serial-data input	SDTO
EAC.AC_MCLK	I	Master codec clock input	MCLK
EAC.AC_RST	O	Codec port interface reset output	nRESET

1) I = Input, O = Output

In I2S mode, the codec port interface is configured as a bidirectional full-duplex serial interface with two time slots per frame. Time slot 0 is used for the left channel audio data, and time slot 1 is used for the right channel audio data. Each time slot is 16 serial clock cycles long, so the frame is 2 x 16 = 32 serial clock cycles long.

13.3.1 Protocol and Data Format

Table 13−4 lists the frame-synchronization (FS), master-clock (MCLK), and serial-clock (SCLK) frequencies for I2S mode in master mode.

Table 13−4. FS, MCLK, and SCLK Frequencies for I2S Mode in Master Mode

FS	MCLK (256 x FS)	SCLK (32 x FS)
44.1 kHz	11.2896 MHz	1.4112 MHz
48 kHz	12.288 MHz	1.536 MHz

Slave mode requires an external codec to provide frame synchronization (44.1 kHz and 48 kHz), and requires the serial clock and the codec port to be set in slave mode.

The LRCK signal has a 50-percent duty cycle and is low for the left channel time slot and high for the right channel time slot. In addition, LRCK is synchronous with the falling edge of SCLK. Serial data is shifted out on the falling edge of SCLK and shifted in on the rising edge of SCLK. For both the left and right channels, a 1-SCLK cycle delay occurs from the edge of LRCK before the MSB of the data is shifted out.

The I2S mode of the codec port interface includes a 16-bit transmit and a 16-bit receive shift register for each SDTO and SDTI signal, respectively. The interface automatically pads the unused bits with zeros. Serial data is transmitted in twos complement with the MSB first.

Figure 13−6 shows the I2S serial-interface format.

Figure 13−6. I2S Serial Interface Format

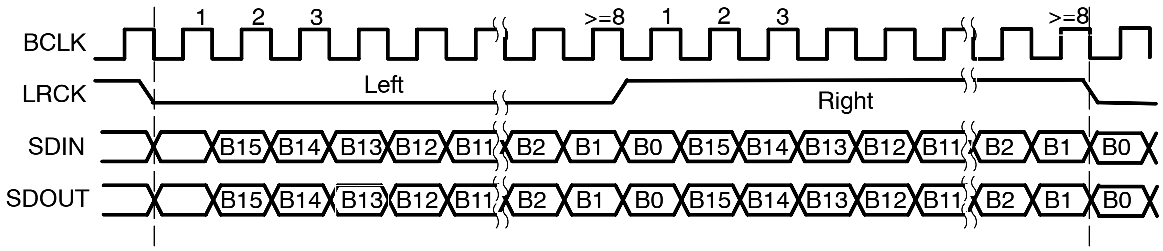
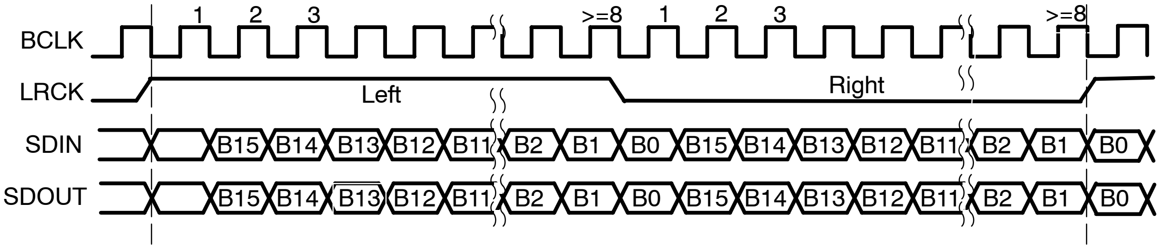


Figure 13−7 shows the polarity-changed I2S serial-interface format.

Figure 13−7. Polarity-Changed I2S Serial Interface Format



The transmitter always sends the MSB of the next word one clock period after the LRCK changes.

The codec port is directly interfaced to the audio-processing module with 16-bit resolution. Valid data bits and time slot length setting must be not less than 16 bits. In case of higher values, only the 16 MSBs of each left and right sample are considered. Input sample LSBs are discarded, and output sample LSBs are filled with zeros.

Note:

In I2S mode, the frame frequency must be 44.1 kHz or 48 kHz for data exchanges with the audio-processing module.

13.4 PCM Codec Functional Interface

In PCM mode, the codec port interface can be configured as a PCM-link serial interface to the PCM codec device. The PCM-link serial interface is a TDM slot-based serial interface that transfers both audio data and command/status data to the codec device.

Figure 13−8 shows the EAC connected to a PCM codec IC.

Figure 13−8. Connection of the EAC to a PCM Codec IC

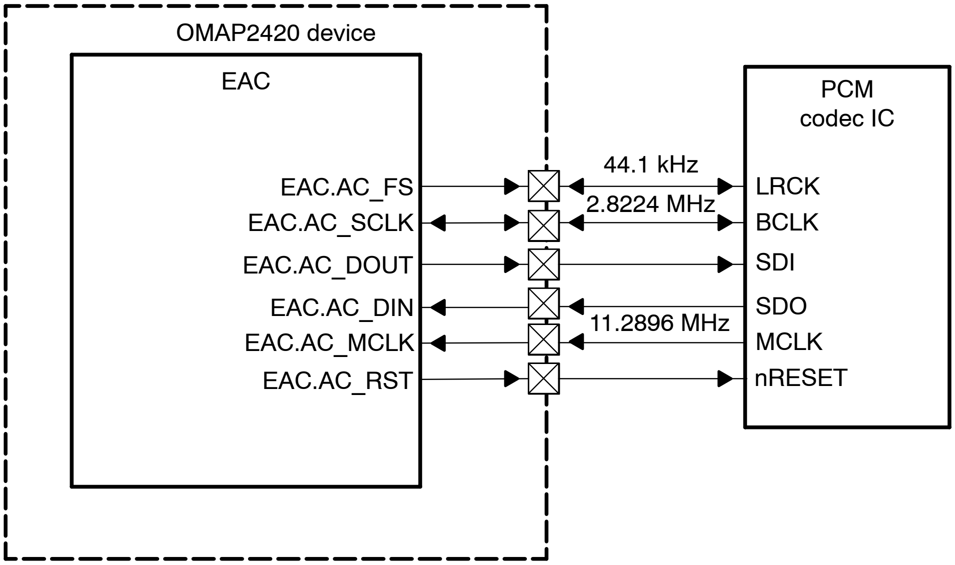


Table 13−5 lists I/O for the PCM codec port.

Table 13−5. PCM Codec Port I/O Description

Signal Name	**I/O**[1]	**Description**	**Codec Source/ Destination**
EAC.AC_FS	I/O	Modem port interface frame synchronization	LRCLK[2]
EAC.AC_SCLK	I/O	Modem port interface serial clock	BCLK[2]
EAC.AC_DOUT	O	Modem port interface serial-data output	SDI
EAC.AC_DIN	I	Modem port interface serial-data input	SDO
EAC.AC_MCLK	I	Master codec clock input	MCLK
EAC.AC_RST	O	Codec port interface reset output	nRESET

1) I = Input, O = Output

2) Clock name for an external PCM codec IC

13.4.1 Protocol and Data Format

In PCM mode, the codec port interface is configured as a bidirectional full-duplex serial interface with two time slots per frame. Time slot 0 is used for the left channel audio data, and time slot 1 is used for the right channel audio data. Each time slot is 32 serial-clock cycles long, so the frame is 2 x 32 = 64 serial-clock cycles in long.

Table 13−6 lists the FS, MCLK, and SCLK frequences for PCM mode.

Table 13−6. FS, MCLK, and SCLK Frequencies for PCM Mode

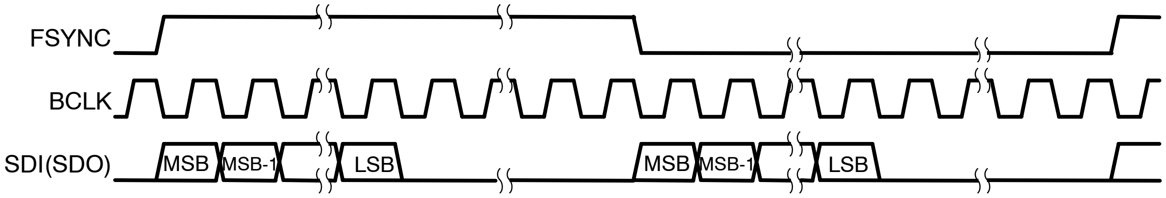
FS	**MCLK (256 x FS)**	**SCLK (64 x FS)**
44.1 kHz	11.2896 MHz	2.8224 MHz
48 kHz	12.288 MHz	3.072 MHz

The LRCK signal has a 50-percent duty cycle and is high for the left channel time slot and low for the right channel time slot. In addition, LRCK is synchronous with the falling edge of BCLK. Serial data is shifted out on the falling edge of BCLK and shifted in on the rising edge of BCLK. There is no BCLK cycle delay from the edge of LRCK before the MSB of the data is shifted out for either the left or right channel.

The PCM mode of the codec port interface includes a 16-bit transmit and a 16-bit receive shift register for each SDO and SDI signal, respectively.

Figure 13−9 shows the PCM serial-interface format.

Figure 13−9. PCM Serial Interface Format



As in I2S mode, the codec port is directly interfaced to the audio-processing module with 16-bit resolution. Valid data bits and time-slot length setting must not be less than 16 bits. In case of PCM mode with higher valid serial data bit values, only the 16 MSBs of each left and right sample are considered. Input sample LSBs are discarded, and output sample LSBs are filled with zeros.

Note:

In PCM mode, the frame frequency must be 44.1 kHz or 48 kHz for data exchanges with the audio-processing module.

13.5 Modem Port Functional Interface

The modem interface is a bidirectional 3-line interface dedicated to the transfer of data to and from external devices offering a 3-line serial interface. This serial port is based on a looped shift register, thus allowing both transmit (PISO) and receive (SIPO) modes. The modem serial port interface is a TDM time-slot-based scheme.

Figure 13−10 shows the EAC connected to a modem device.

Figure 13−10. Connections of the EAC to the Modem Device

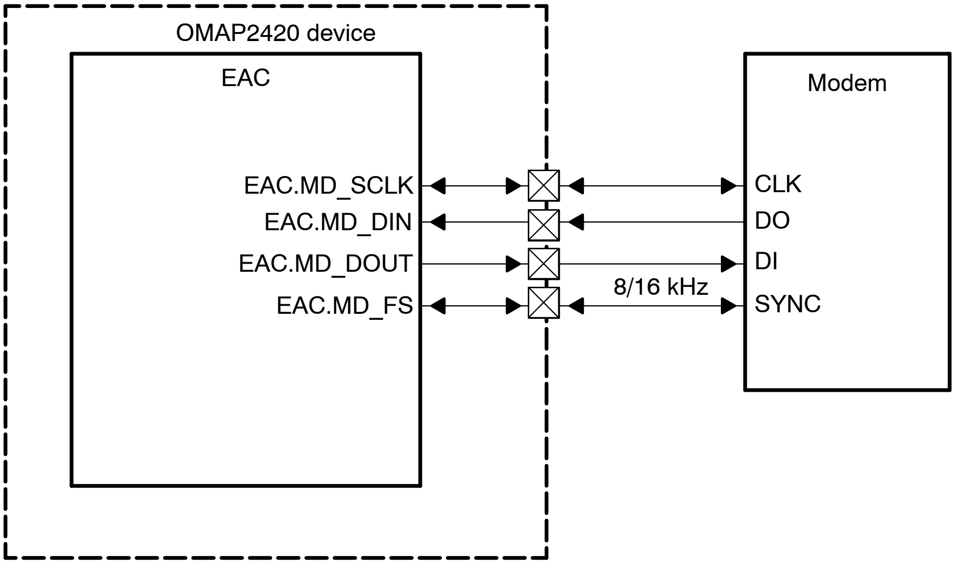


Table 13−7 lists I/O for the modem port.

Table 13−7. Modem Port I/O Description

Signal Name	I/O[1]	Description	Modem Source/ Destination
EAC.MD_SCLK	I/O	Modem port interface serial clock	CLK
EAC.MD_DIN	I	Modem port interface serial-data input	DO
EAC.MD_DOUT	O	Modem port interface serial-data output	DI
EAC.MD_FS	I/O	Modem port interface frame synchronization	SYNC

[1] I = Input, O = Output

13.5.1 Protocol and Data Format

Communications on the main channel are handled with frame synchronization (EAC.MD_FS). This signal is a one-EAC.MD_SCLK-wide pulse every 125 µs (8 kHz) or 62.5 µs (16 kHz) in nontransparent DMA mode. In transparent DMA mode, the frame synchronization frequency can be between 4 kHz–100 KHz. Communication between the EAC modem port interface and the modem can be in master or slave mode:

- ❑ Master mode: EAC.MD_FS and EAC.MD_SCLK are generated by the modem port interface.
- ❑ Slave mode: EAC.MD_FS and EAC.MD_SCLK are provided by the modem.

CAUTION

The transparent DMA mode applies only to slave mode. Master mode and nontransparent slave mode support only 8-kHz and 16-kHz frame synchronization.

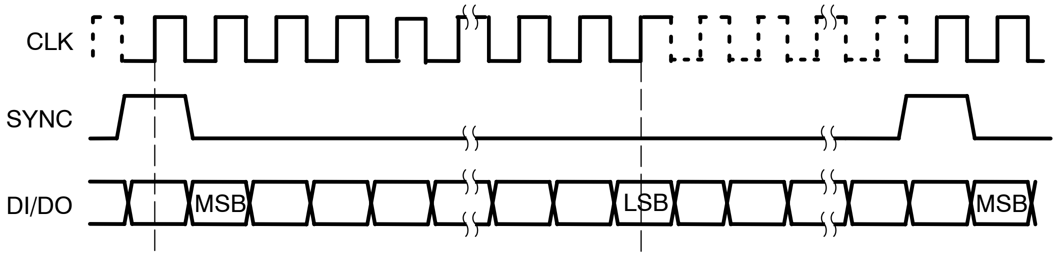
13.5.1.1 Serial Signals

The serial protocol includes the setting of frame synchronization MD_FSYNC and serial data DI/DO signals to the serial bit clock EAC.MD_SCLK.

- ❑ Example 1:
  - Serial clock polarity falling: MPMCCFR.CPB_MC = 0
  - Frame-synchronization polarity falling: MPMCCFR.FSP_MC = 0
  - Frame-synchronization level high: MPMCCFR.FSBL_MC = 1

Figure 13−11 shows the modem port serial signals for example 1.

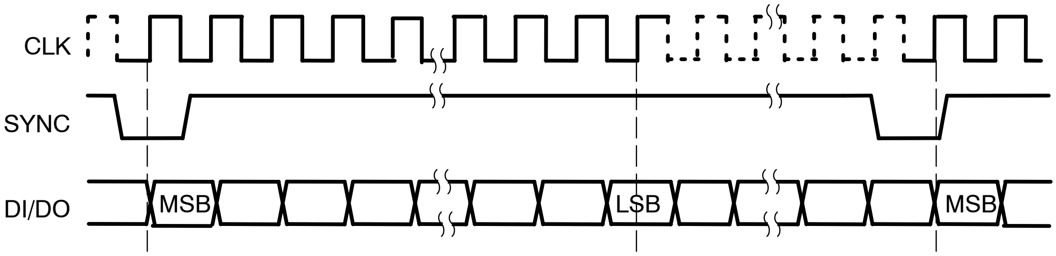
Figure 13−11. Modem Port Serial Signals—Example 1



- ❑ Example 2:
  - Serial-clock polarity rising: MPMCCFR.CPB_MC = 1
  - Frame-synchronization polarity falling: MPMCCFR.FSP_MC = 0
  - Frame-synchronization level low: MPMCCFR.FSBL_MC = 0

Figure 13−12 shows the modem port serial signals for example 2.

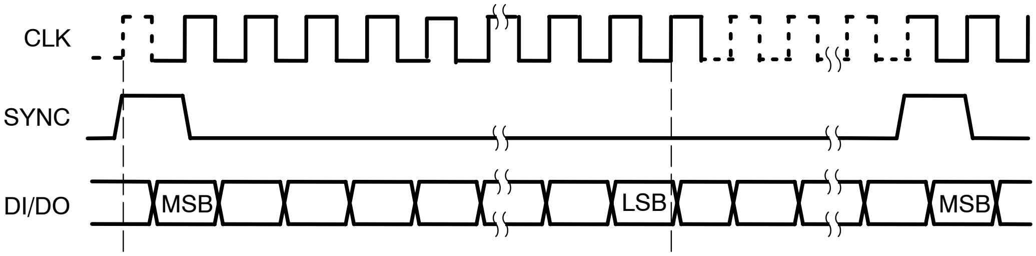
Figure 13−12. Modem Port Serial Signals—Example 2



- ❑ Example 3:
  - ■ Serial-clock polarity falling: MPMCCFR.CPB_MC = 0
  - ■ Frame-synchronization polarity rising: MPMCCFR.FSP_MC = 1
  - ■ Frame-synchronization level high: MPMCCFR.FSBL_MC = 1

Figure 13−13 shows the modem port serial signals for example 3.

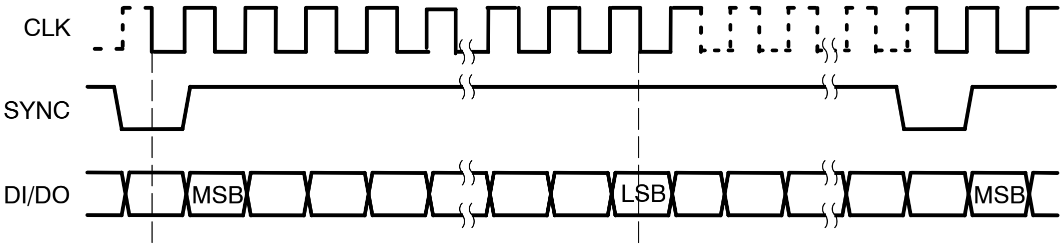
Figure 13−13. Modem Port Serial Signals—Example 3



- ❑ Example 4:
  - ■ Serial-clock polarity rising: MPMCCFR.CPB_MC = 1
  - ■ Frame-synchronization polarity rising: MPMCCFR.FSP_MC = 1
  - ■ Frame-synchronization level low: MPMCCFR.FSBL_MC = 0

Figure 13−14 shows the modem port serial signals for example 4.

Figure 13−14. Modem Port Serial Signals Example 4



13.5.1.2 Serial Data Length to Internal Data Relationship

The main channel interfaces to the audio processing module with fixed 16-bit resolution (uncompressed) or 8-bit resolution if compand/expand mode is selected. The serial part of the modem port adapts the serial stream bit number (WS_MC of MPMCCFR) to the internal sample width.

Note: Companding: Compressing and expanding. A quantization scheme for audio signals in which the input signal is compressed and then, after processing, is reconstructed at the output by expansion. The A-law companding scheme is used in Europe, and the μ-law companding scheme is used in the United States.

Serial Data Length = Internal Parallel Data Length

- ❑ Serial data length = 16 bits if compression is disabled.

- ❑ Serial data length = 8 bits if compand/expand compression is enabled.
- ❑ Example:
  - ■ 16 serial data bits per transfer: MPMCCFR.WS_MC = 0x0F
  - ■ Expand for data receive disabled: MPMCCFR.EXPAND = 0x0
  - ■ Compand for data transmit disabled: MPMCCFR.COMPAND = 0x0

Figure 13−15 shows that the modem port serial data length equals the internal parallel data-length example.

Figure 13−15. Modem Port Serial Data Length = Internal Parallel Data Length—Example

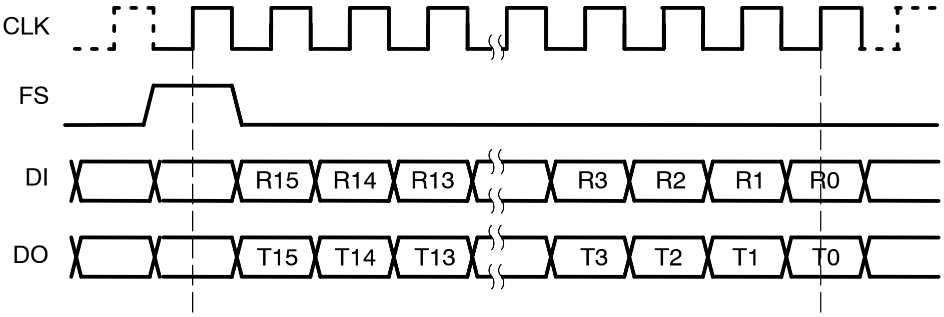


Figure 13−16 is the register view of Figure 13−15.

Figure 13−16. Modem Port Serial Data Length = Internal Parallel Data Length—Example: Register Description

	15															0
Audio sample OUT 16 bits	T 15	T 14	T 13	T 12	T 11	T 10	T 9	T 8	T 7	T 6	T 5	T 4	T 3	T 2	T 1	T 0
	15															0
Audio sample IN 16 bits	R 15	R 14	R 13	R 12	R 11	R 10	R 9	R 8	R 7	R 6	R 5	R 4	R 3	R 2	R 1	R 0

Serial Data Length is Greater Than Internal Parallel Data Length

- ❑ Serial data length is more than 16 bits if compression is disabled.
- ❑ Serial data length is more than 8 bits if compand/expand compression is enabled.
- ❑ For the receive path, extra LSBs (or MSBs) are discarded if data are aligned on MSB (or LSB).
- ❑ For the transmit path, missing LSBs (or MSBs) are padded with the DFILL_MC bit value for MSB (or LSB) alignment.

CAUTION

This interface is 8 bits or 16 bits only, but can support up to a 32-bit format on the serial link.

Extra bits (more than 8 bits or 16 bits) on the serial link are ignored in reception and are filled with a predefined value in transmission.

- Example 1: MSB alignment
  - 18 serial data bits per transfer: MPMCCFR.WS_MC = 0x11
  - Expand for data receive disabled: MPMCCFR.EXPAND = 0x0
  - Compand for data transmit disabled: MPMCCFR.COMPAND = 0x0
  - MSB alignment: MPMCCFR.DJUST_MC = 0
  - Serial data filling: MPMCCFR.DFILL_MC = 0/1

Figure 13−17 shows how modem port serial data length is greater than internal parallel data length in example 1.

Figure 13−17. Modem Port Serial Data Length is Greater Than Internal Parallel Data Length—Example 1

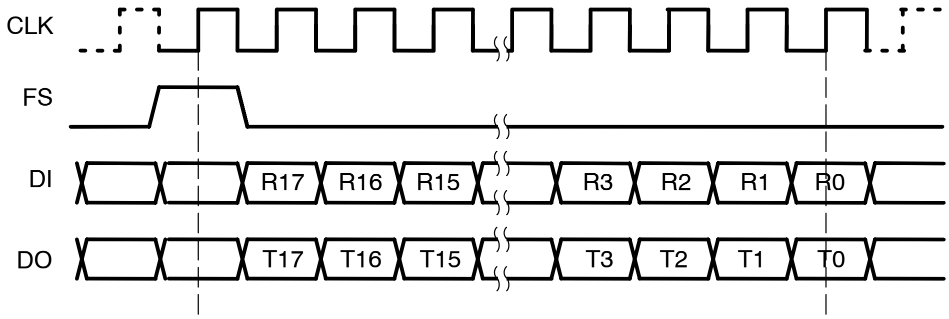


Figure 13−18 is the register view of how modem port serial data length is greater than internal parallel data length in example 1.

Figure 13−18. Modem Port Serial Data Length is Greater Than Internal Parallel Data Length—Example 1: Register Description With MSB Alignment

	15															0
Audio sample OUT 16 bits	T 17	T 16	T 15	T 14	T 13	T 12	T 11	T 10	T 9	T 8	T 7	T 6	T 5	T 4	T 3	T 2

	15															0
Audio sample IN 16 bits	R 17	R 16	R 15	R 14	R 13	R 12	R 11	R 10	R 9	R 8	R 7	R 6	R 5	R 4	R 3	R 2

- Example 2: LSB alignment
  - 18 serial data bits per transfer: MPMCCFR.WS_MC = 0x11
  - Expand for data receive disabled: MPMCCFR.EXPAND = 0x0
  - Compand for data transmit disabled: MPMCCFR.COMPAND = 0x0
  - LSB alignment: MPMCCFR.DJUST_MC = 1
  - Serial data filling: MPMCCFR.DFILL_MC = 0/1

Figure 13−19 shows how modem port serial data length is greater than internal parallel data length in example 2.

Figure 13−19. Modem Port Serial Data Length is Greater Than Internal Parallel Data Length—Example 2

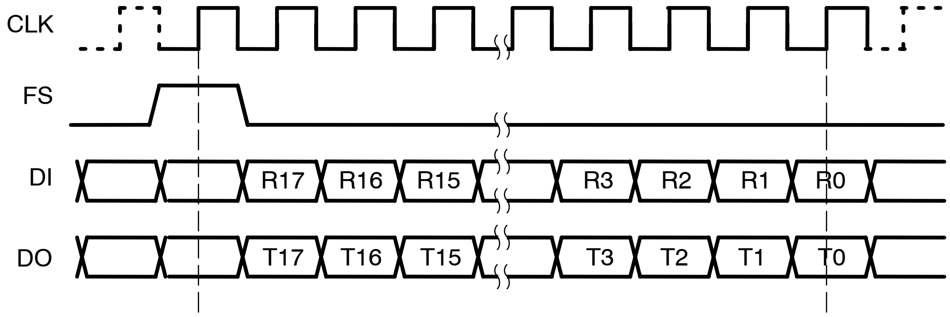


Figure 13−20 is the register view of how modem port serial data length is greater than internal parallel data length in example 2.

Figure 13−20. Modem Port Serial Data Length is Greater Than Internal Parallel Data Length—Example 2: Register Description With LSB Alignment

	15															0
Audio sample OUT 16 bits	T 15	T 14	T 13	T 12	T 11	T 10	T 9	T 8	T 7	T 6	T 5	T 4	T 3	T 2	T 1	T 0

	15															0
Audio sample IN 16 bits	R 15	R 14	R 13	R 12	R 11	R 10	R 9	R 8	R 7	R 6	R 5	R 4	R 3	R 2	R 1	R 0

Serial Data Length is Less Than Internal Parallel Data Length

- Serial data length is more than 16 bits if compression is disabled.
- Serial data length is more than 8 bits if compand/expand compression is enabled.
- For the receive path, extra LSBs (or MSBs) are discarded if data are aligned on the MSB (or the LSB).
- For the transmit path, missing LSBs (or MSBs) are padded with the DFILL_MC bit value for MSB (or LSB) alignment.
- Example 1: MSB alignment
  - 12 serial data bits per transfer: MPMCCFR.WS_MC = 0xB
  - Expand for data receive disabled: MPMCCFR.EXPAND = 0x0
  - Compand for data transmit disabled: MPMCCFR.COMPAND = 0x0
    MSB alignment: MPMCCFR.DJUST_MC = 0
  - Serial data filling: MPMCCFR.DFILL_MC = 0/1

Figure 13−21 shows how modem port serial data length is less than internal parallel data length in example 1.

Figure 13−21. Modem Port Serial Data Length is Less Than Internal Parallel Data Length—Example 1

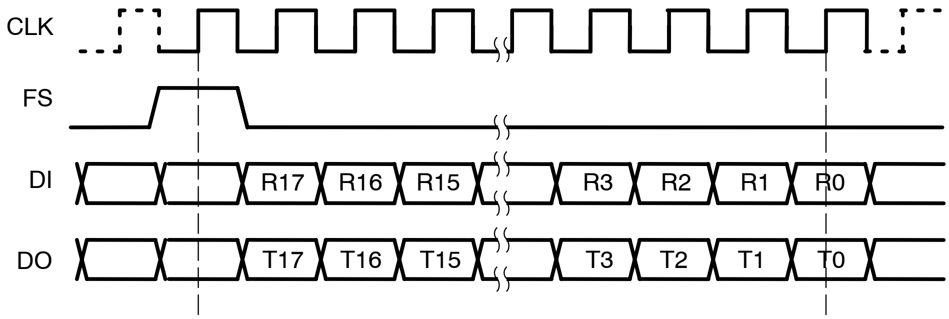


Figure 13−22 is the register view of how modem port serial data length is less than internal parallel data length in example 1.

Figure 13−22. Modem Port Serial Data Length is Less Than Internal Parallel Data Length—Example 1: Register Description With MSB Alignment

	15															0
Audio sample OUT	T 11	T 10	T 9	T 8	T 7	T 6	T 5	T 4	T 3	T 2	T 1	T 0				

	15															0
Audio sample IN	R 11	R 10	R 9	R 8	R 7	R 6	R 5	R 4	R 3	R 2	R 1	R 0	0/ 1	0/ 1	0/ 1	0/ 1

Note: Sample IN bits [3:0] are filled according to the DFILL_MC value.

- ❑ Example 2: LSB alignment
  - 12 serial data bits per transfer: MPMCCFR.WS_MC = 0xB
  - Expand for data receive disabled: MPMCCFR.EXPAND = 0x0
  - Compand for data transmit disabled: MPMCCFR.COMPAND = 0x0
  - LSB alignment: MPMCCFR.DJUST_MC = 1
  - Serial data filling: MPMCCFR.DFILL_MC = 0/1

Figure 13−23 shows how modem port serial data length is less than internal parallel data length in example 2.

Figure 13−23. Modem Port Serial Data Length is Less Than Internal Parallel Data Length—Example 2

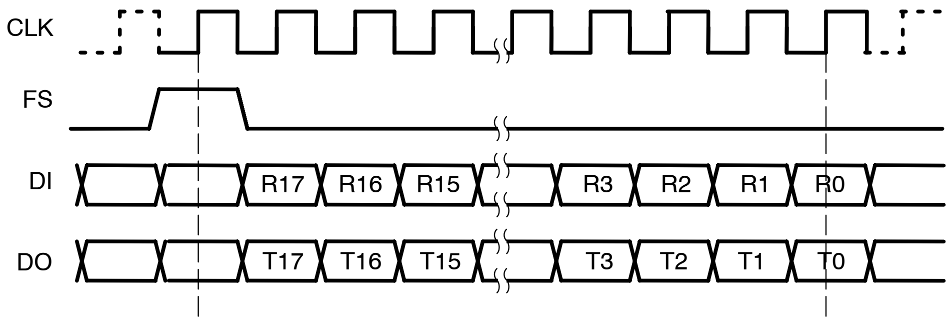


Figure 13−24 is the register view of how modem port serial data length is less than internal parallel data length in example 2.

Figure 13−24. Modem Port Serial Data Length is Less Than Internal Parallel Data Length—Example 2: Register Description With LSB Alignment

	15															0
Audio sample OUT					T11	T10	T9	T8	T7	T6	T5	T4	T3	T2	T1	T0
	15															0
Audio sample IN	0/1	0/1	0/1	0/1	R11	R10	R9	R8	R7	R6	R5	R4	R3	R2	R1	R0

Note: Sample IN bits [15:12] are filled according to the DFILL_MC value.

13.6 Bluetooth Port Functional Interface

The Bluetooth interface is a bidirectional 3-line interface dedicated to the transfer of data to and from external devices offering a 3-line serial interface. This serial port is based on a looped shift register, thus allowing both transmit (PISO) and receive (SIPO) modes. The Bluetooth serial port interface is a TDM time-slot-based scheme.

Figure 13−25 shows the EAC connected to a Bluetooth device.

Figure 13−25. Connections of the EAC to Bluetooth Device

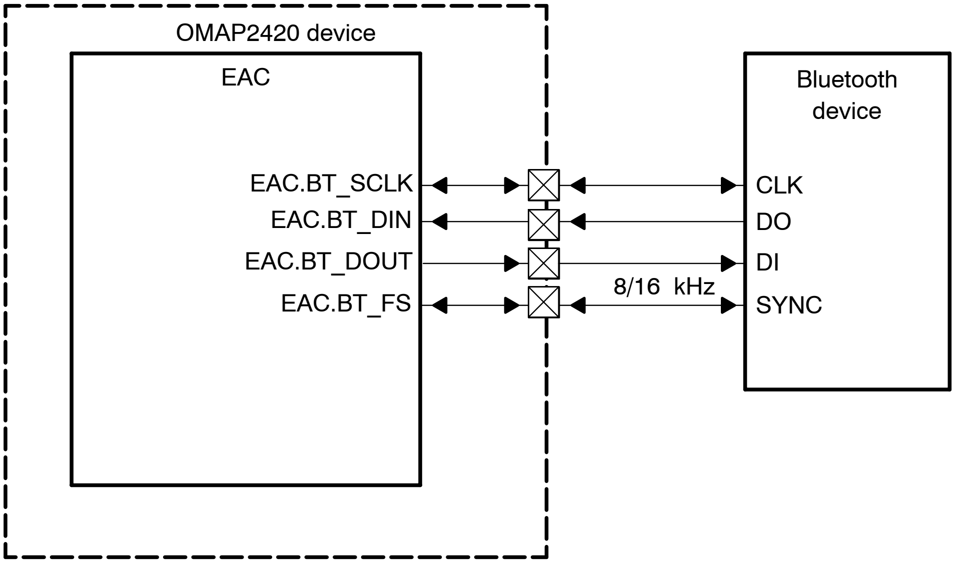


Table 13−8 lists the I/O for the Bluetooth device.

Table 13−8. Bluetooth Port I/O Description

Signal Name	I/O[1]	Description	Modem Source/ Destination
EAC.BT_SCLK	I/O	Bluetooth port interface serial clock	CLK
EAC.BT_DIN	I	Bluetooth port interface serial-data input	DO
EAC.BT_DOUT	O	Bluetooth port interface serial-data output	DI
EAC.BT_FS	I/O	Bluetooth port interface frame synchronization	SYNC

1) I = Input, O = Output

13.6.1 Protocol Chronograms

Communications on the main channel are handled with frame synchronization (EAC_BT_FS). This signal is a one-EAC_BT_SCLK-wide pulse every 125 µs (8 kHz) or every 62.5 µs (16 kHz). Communication between the EAC Bluetooth port interface and the external Bluetooth device can be in master or slave modes.

- ❑ Master mode: The Bluetooth port interface generates EAC_BT_FS and EAC_BT_SCLK.
- ❑ Slave mode: The external Bluetooth device generates EAC_BT_FS and EAC_BT_SCLK.

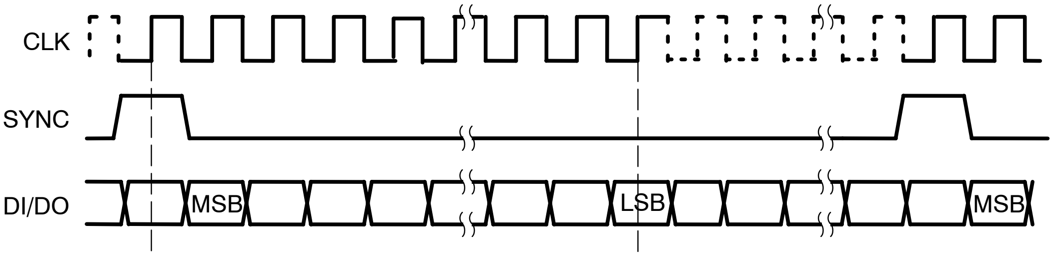
13.6.1.1 Serial Signals

Serial protocol includes setting the relationship between the frame synchronization EAC_BT_FS and serial data DI/DO to the serial bit clock EAC_BT_SCLK.

- ❑ Example 1:
  - ■ Serial-clock polarity falling: BPMCCFR.CPB_MC = 0
  - ■ Frame-synchronization polarity falling: BPMCCFR.FSP_MC = 0
  - ■ Frame-synchronization level high: BPMCCFR.FSBL_MC = 1

Figure 13−26 shows the Bluetooth port serial signals for example 1.

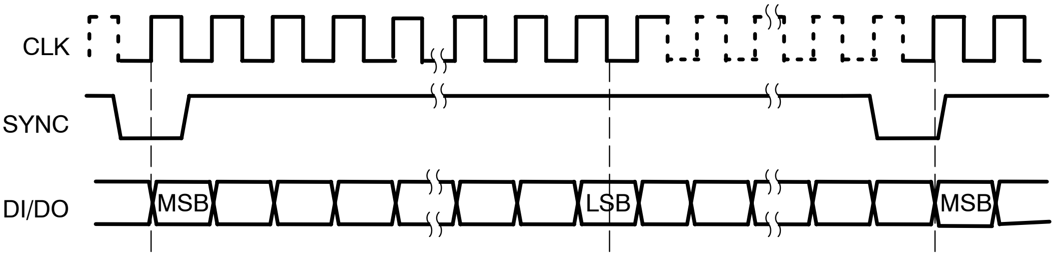
Figure 13−26. Bluetooth Port Serial Signals—Example 1



- ❑ Example 2:
  - ■ Serial-clock polarity rising: BPMCCFR.CPB_MC = 1
  - ■ Frame-synchronization polarity falling: BPMCCFR.FSP_MC = 0
  - ■ Frame-synchronization level low: BPMCCFR.FSBL_MC = 0

Figure 13−27 shows the Bluetooth port serial signals for example 2

Figure 13−27. Bluetooth Port Serial Signals—Example 2

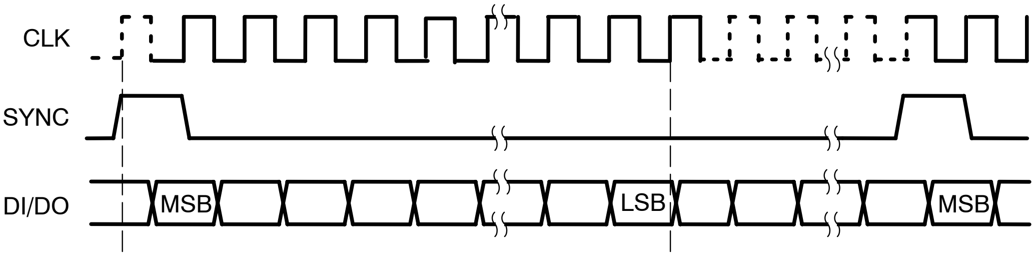


- ❑ Example 3:
  - ■ Serial-clock polarity falling: BPMCCFR.CPB_MC = 0

  - Frame-synchronization polarity rising: BPMCCFR.FSP_MC = 1
  - Frame-synchronization level high: BPMCCFR.FSBL_MC = 1

Figure 13−28 shows the Bluetooth port serial signals for example 3.

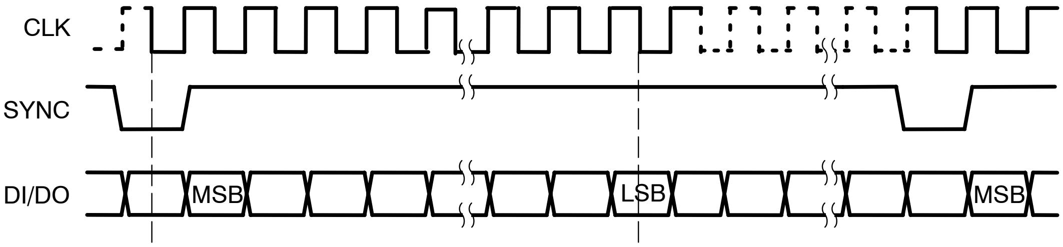
Figure 13−28. Bluetooth Port Serial Signals—Example3



- Example 4:
  - Serial clock polarity rising: BPMCCFR.CPB_MC = 1
  - Frame synchronization polarity rising: BPMCCFR.FSP_MC = 1
  - Frame synchronization level low: BPMCCFR.FSBL_MC = 0

Figure 13−29 shows the Bluetooth port serial signals for example 4.

Figure 13−29. Bluetooth Port Serial Signals—Example 4



13.6.1.2 Serial Data Length to Internal Data Relationship

The main channel interfaces with the audio-processing module with fixed 16-bit resolution (uncompressed) or 8-bit resolution if compand/expand mode is selected. The serial part of the Bluetooth port adapts the serial stream bit number (WS_MC of BPMCCFR) to the internal sample width.

Serial Data Length = Internal Parallel Data Length

- Serial data length = 16 bits if compression is disabled.
- Serial data length = 8 bits if compand/expand compression is enabled.
- Example:
  - 16 serial data bits per transfer: BPMCCFR.WS_MC = 0x0F
  - Expand for data receive disabled: BPMCCFR.EXPAND = 0x0

- Compand for data transmit disabled: BPMCCFR.COMPAND = 0x0

Figure 13−30 shows that the Bluetooth port serial data length equals the internal parallel data length for the example.

Figure 13−30. Bluetooth Port Serial Data Length = Internal Parallel Data Length—Example

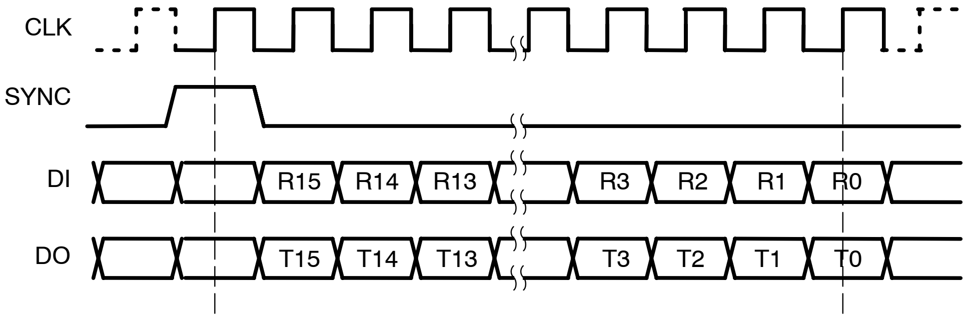


Figure 13−31 is the register view of how the Bluetooth port serial data length equals the internal parallel data length for the example.

Figure 13−31. Bluetooth Port Serial Data Length = Internal Parallel Data Length—Example: Register Description

	15															0
Audio sample OUT 16 bits	T 15	T 14	T 13	T 12	T 11	T 10	T 9	T 8	T 7	T 6	T 5	T 4	T 3	T 2	T 1	T 0
	15															0
Audio sample IN 16 bits	R 15	R 14	R 13	R 12	R 11	R 10	R 9	R 8	R 7	R 6	R 5	R 4	R 3	R 2	R 1	R 0

Serial Data Length is Greater Than Internal Parallel Data Length

- ❑ Serial data length is more than 16 bits if compression is disabled.
- ❑ Serial data length is more than 8 bits if compand/expand compression is enabled.
- ❑ For the receive path, extra LSBs (or MSBs) are discarded if data are aligned on MSB (or on LSB).
- ❑ For the transmit path, missing LSBs (or MSBs) are padded with DFILL_MC bit value for MSB (or LSB) alignment.

WARNING

This interface is 8 bits or 16 bits only, but can support up to a 32-bit format on the serial link.

Extra bits (that is, more than 8 bits or 16 bits) On the serial link are ignored in reception and are filled with a predefined value in transmission.

- Example 1: MSB alignment
  - 18 serial data bits per transfer: BPMCCFR.WS_MC = 0x11
  - Expand for data receive disabled: BPMCCFR.EXPAND = 0x0
  - Compand for data transmit disabled: BPMCCFR.COMPAND = 0x0
  - MSB alignment: BPMCCFR.DJUST_MC = 0
  - Serial data filling: BPMCCFR.DFILL_MC = 0/1

Figure 13−32 shows how Bluetooth port serial data length is greater than internal parallel data length in example 1.

Figure 13−32. Bluetooth Port Serial Data Length is Greater Than Internal Parallel Data Length—Example 1

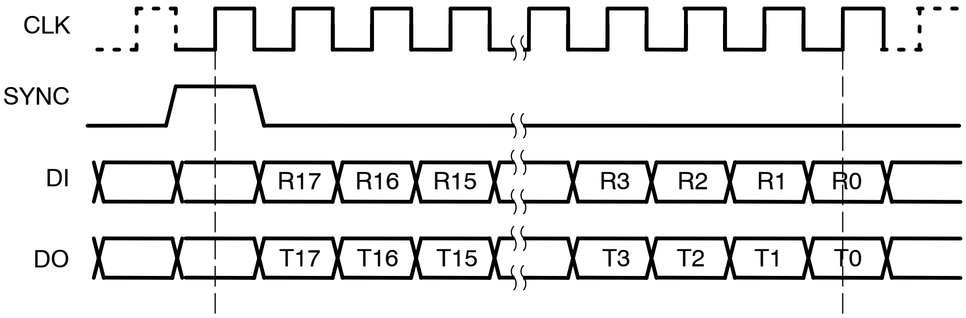


Figure 13−33 is the register view of how Bluetooth port serial data length is greater than internal parallel data length in example 1.

Figure 13−33. Bluetooth Port Serial Data Length is Greater Than Internal Parallel Data Length—Example 1: Register Description With MSB Alignment

	15															0
Audio sample OUT 16 bits	T 17	T 16	T 15	T 14	T 13	T 12	T 11	T 10	T 9	T 8	T 7	T 6	T 5	T 4	T 3	T 2
	15															0
Audio sample IN 16 bits	R 17	R 16	R 15	R 14	R 13	R 12	R 11	R 10	R 9	R 8	R 7	R 6	R 5	R 4	R 3	R 2

- Example 2: LSB alignment
  - 18 serial data bits per transfer: BPMCCFR.WS_MC = 0x11
  - Expand for data receive disabled: BPMCCFR.EXPAND = 0x0
  - Compand for data transmit disabled: BPMCCFR.COMPAND = 0x0
  - LSB alignment: BPMCCFR.DJUST_MC = 1
  - Serial data filling: BPMCCFR.DFILL_MC = 0/1

Figure 13−34 shows how Bluetooth port serial data length is greater than internal parallel data length in example 2.

Figure 13–34. Bluetooth Port Serial Data Length is Greater Than Internal Parallel Data Length—Example 2

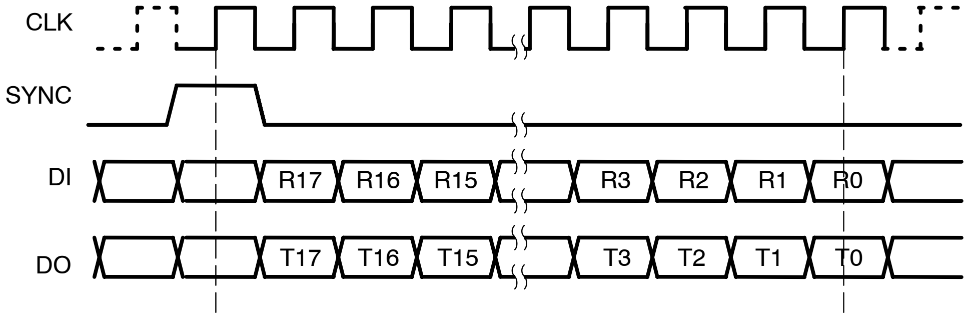


Figure 13–35 is the register view of how Bluetooth port serial data length is greater than internal parallel data length in example 2.

Figure 13–35. Bluetooth Port Serial Data Length is Greater Than Internal Parallel Data Length—Example 2: Register Description With LSB Alignment

	15															0
Audio sample OUT 16 bits	T 15	T 14	T 13	T 12	T 11	T 10	T 9	T 8	T 7	T 6	T 5	T 4	T 3	T 2	T 1	T 0
	15															0
Audio sample IN 16 bits	R 15	R 14	R 13	R 12	R 11	R 10	R 9	R 8	R 7	R 6	R 5	R 4	R 3	R 2	R 1	R 0

Serial Data Length is Less Than Internal Parallel Data Length

- Serial data length less than 16 bits if compression is disabled.
- Serial data length less than 8 bits if compand/expand compression is enabled.
- For the transmit path, extra LSBs (or MSBs) of audio sample that are not transmitted are discarded if data are aligned on the MSB (or the LSB).
- For the receive path, the received audio sample is rebuilt from the serial data bits received. Missing LSBs (or MSBs) are padded with the DFILL_MC bit value for MSB (or LSB) alignment.
- Example 1: MSB alignment
  - 12 serial data bits per transfer: BPMCCFR.WS_MC = 0xB
  - Expand for data receive disabled: BPMCCFR.EXPAND = 0x0
  - Compand for data transmit disabled: BPMCCFR.COMPAND = 0x0
  - MSB alignment. BPMCCFR.DJUST_MC = 0
  - Serial data filling: BPMCCFR.DFILL_MC = 0/1

Figure 13−36 shows how Bluetooth port serial data length is less than internal parallel data length in example 1.

Figure 13−36. Bluetooth Port Serial Data Length is Less Than Internal Parallel Data Length—Example 1

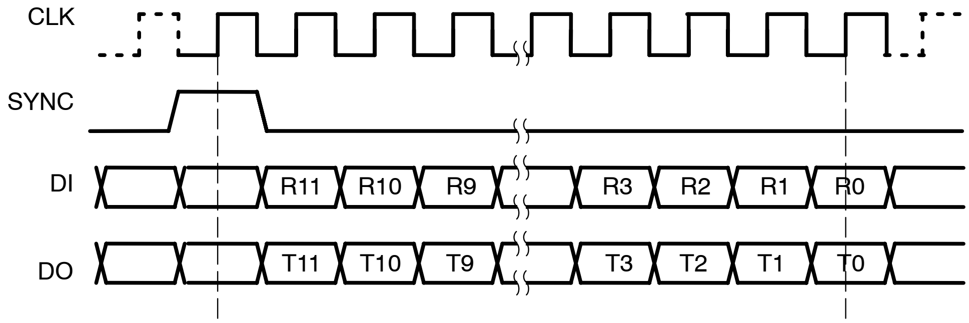


Figure 13−37 is the register view of how Bluetooth port serial data length is less than internal parallel data length in example 1.

Figure 13−37. Bluetooth Port Serial Data Length is Less Than Internal Parallel Data Length—Example 1: Register Description With MSB Alignment

	15															0
Audio sample OUT	T 11	T 10	T 9	T 8	T 7	T 6	T 5	T 4	T 3	T 2	T 1	T 0				

	15															0
Audio sample IN	R 11	R 10	R 9	R 8	R 7	R 6	R 5	R 4	R 3	R 2	R 1	R 0	0/ 1	0/ 1	0/ 1	0/ 1

Note: Sample IN bits [3:0] are filled according to the DFILL_MC value.

- ❑ Example 2: LSB alignment
  - 12 serial data bits per transfer: BPMCCFR.WS_MC = 0xB
  - Expand for data receive disabled: BPMCCFR.EXPAND = 0x0
  - Compand for data transmit disabled: BPMCCFR.COMPAND = 0x0
  - LSB alignment: BPMCCFR.DJUST_MC = 1
  - Serial data filling: BPMCCFR.DFILL_MC = 0/1

Figure 13−38 shows how Bluetooth port serial data length is less than internal parallel data length in example 2.

Figure 13−38. Bluetooth Port Serial Data Length is Less Than Internal Parallel Data Length—Example 2

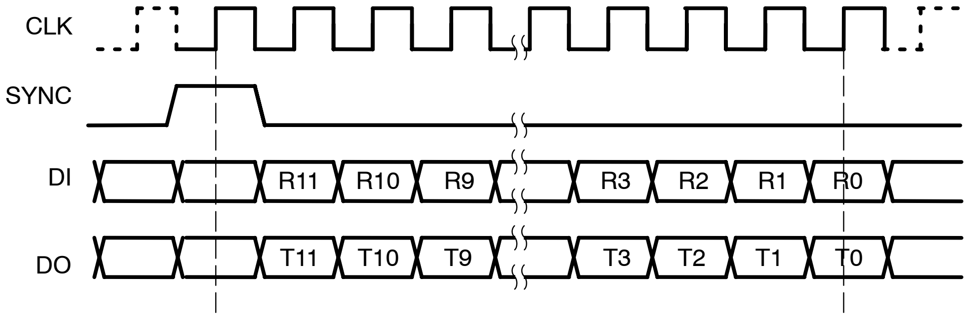


Figure 13−39 is the register view of how Bluetooth port serial data length is less than internal parallel data length in example 2.

Figure 13−39. Bluetooth Port Serial Data Length is Less Than Internal Parallel Data Length—Example 2: Register Description With LSB Alignment

	15															0
Audio sample OUT					T 11	T 10	T 9	T 8	T 7	T 6	T 5	T 4	T 3	T 2	T 1	T 0

	15															0
Audio sample IN	0/ 1	0/ 1	0/ 1	0/ 1	R 11	R 10	R 9	R 8	R 7	R 6	R 5	R 4	R 3	R 2	R 1	R 0

Note: Sample IN bits [15:12] are filled according to the DFILL_MC value.

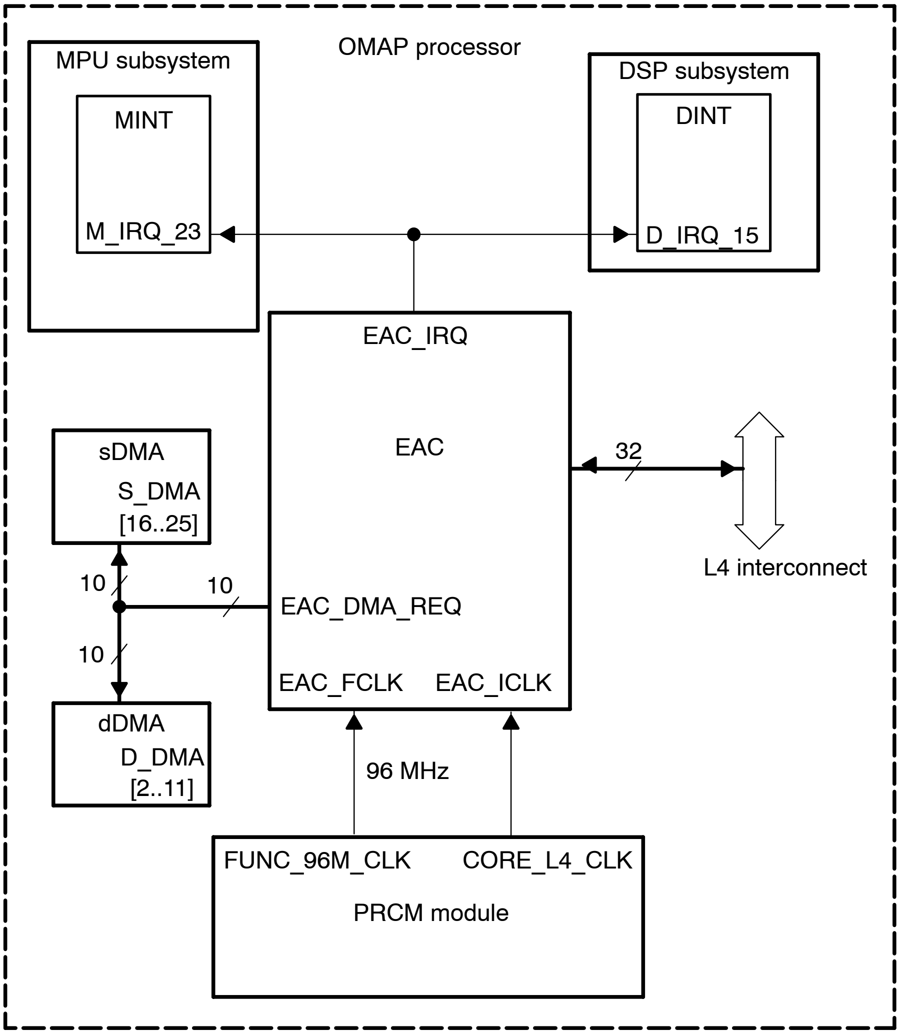
13.7 EAC Integration

This section describes EAC integration in the OMAP2420.

13.7.1 Description

Figure 13−40 shows the EAC integrated in the OMAP2420.

Figure 13−40. EAC Integration to the OMAP Processor



13.7.2 Clocking, Reset, and Power-Management Schemes

13.7.2.1 Clocking

The EAC module operates from a 96-MHz functional clock, EAC_FCLK. Its clock source from the power, reset, and clock management (PRCM) module is FUNC_96M_CLK.

At PRCM level, the EAC module can be configured to prevent/allow FUNC_96M_CLK to be shut down using the EN_EAC bit field in the CM_FCLKEN1_CORE register.

Note:

FUNC_96M_CLK shuts down only if all other modules that receive this clock are also configured for its shutdown.

EAC_ICLK comes internally from the PRCM module (its source is CORE_L4_CLK) and runs at the L4 clock interconnect frequency. CORE_L4_CLK can be shut down using the EN_EAC bit field in the CM_ICLKEN1_CORE register.

Note:

CORE_L4_CLK shuts down only if all other modules that receive this clock are also configured for its shutdown.

The PRCM also provides a way to synchronize EAC_ICLK and CORE power domain transitions using the AUTO_EAC bit in the CM_AUTOIDLE_CORE register. When this bit is set to 1, the EAC automatically follows the domain transitions.

Note:

The EAC does not support the IdleReq/SidleAck handshake protocol with the PRCM module. The software ensures correct clock management. EAC clocks can be cut regardless of any pending transactions or events at the EAC level.

For more information on clock management, see Chapter 5, *Power, Reset, and Clock Management*.

13.7.2.2 Hardware Reset

CORE_RST is the reset domain for the EAC.

13.7.2.3 Software Reset

There is one software reset. Setting the SOFTRESET bit in the EAC SYSCONFIG register to 1 triggers a module reset.

13.7.2.4 Power Domain

The EAC is in the CORE power domain.

13.7.2.5 Hardware Requests

The EAC provides one interrupt line connected to the MPU interrupt controller and DSP interrupt controller. The EAC provides 10 DMA requests, each connected to the system DMA (sDMA) and DSP DMA (dDMA) (see Table 13−9).

Table 13−9. Interrupt Line and DMA Requests

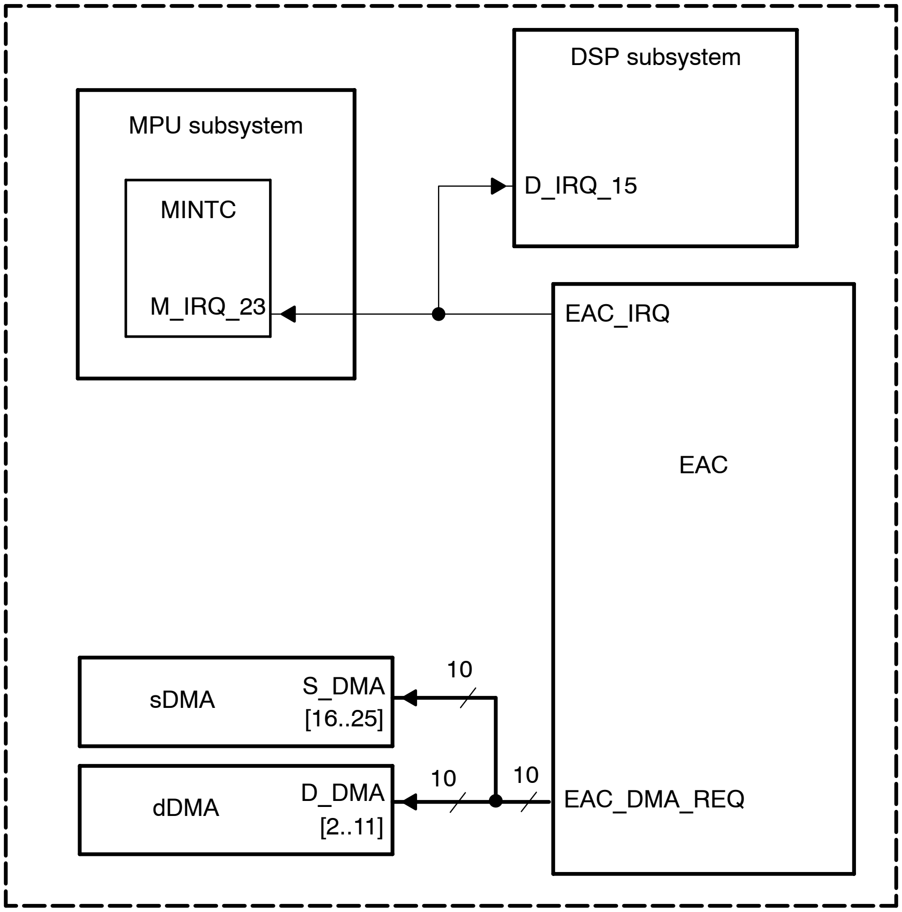
Name	Mapping		Comments
EAC_IRQ	M_IRQ_23	D_IRQ_15	Destination is MPU and DSP subsystem interrupts
EAC_AC_DMA_RD	S_DMA_16	D_DMA_2	EAC audio codec port—read request

Table 13−9. Interrupt Line and DMA Requests (Continued)

Name	Mapping		Comments
EAC_AC_DMA_WR	S_DMA_17	D_DMA_3	EAC audio codec port—write request
EAC_MD_UL_DMA_RD	S_DMA_18	D_DMA_4	EAC modem/voice port—uplink read request
EAC_MD_UL_DMA_WR	S_DMA_19	D_DMA_5	EAC modem/voice port—uplink write request
EAC_MD_DL_DMA_RD	S_DMA_20	D_DMA_6	EAC modem/voice port—downlink read request
EAC_MD_DL_DMA_WR	S_DMA_21	D_DMA_7	EAC modem/voice port—downlink write request
EAC_BT_UL_DMA_RD	S_DMA_22	D_DMA_8	EAC Bluetooth port—uplink read request
EAC_BT_UL_DMA_WR	S_DMA_23	D_DMA_9	EAC Bluetooth port—uplink write request
EAC_BT_DL_DMA_RD	S_DMA_24	D_DMA_10	EAC Bluetooth port—downlink read request
EAC_BT_DL_DMA_WR	S_DMA_25	D_DMA_11	EAC Bluetooth port—downlink write request

Figure 13−41 shows EAC requests.

Figure 13−41. EAC Requests Diagram



13.7.3 Pin List and Pad Multiplexing with Other Functions

Table 13−10 shows pin multiplexing for the EAC module. Gray shading indicates use of EAC I/O for that mode. Black shading indicates that the mode is not used.

Table 13−10. Pin Multiplexing for EAC Module

EAC	Description	DIR	Ball 2420	ALTERNATE FUNCTIONS					
Interface				Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
EAC.AC_MSCLK	Audio codec master clock input or output	I	V14	▨	■	■	GPIO.117	■	■
EAC.AC_FS	Audio codec interface frame synchronization	I/O	R14	▨	McBSP2.FSX	■	GPIO.114	■	■
EAC.AC_SCLK	Audio codec interface frame synchronization	I/O	Y15	▨	McBSP2.CLKX	■	GPIO.113	■	■
EAC.AC_DIN	Audio codec interface serial data input	I	W15	▨	McBSP2.DR	■	GPIO.115	■	■
EAC.AC_DOUT	Audio codec interface serial data output	O	V15	▨	McBSP2.DX	■	GPIO.116	■	■
EAC.AC_RST	Audio codec interface reset output (AC97 mode)	O	W16	▨	EAC.BT_DIN	LOOP_EAC.BT_DIN	GPIO.118	■	■
EAC.MD_SCLK	Modem voice interface serial clock	I/O	Y12		▨	■	GPIO.63	■	■
EAC.MD_FS	Modem voice interface frame synchronization	I/O	Y13		▨	■	GPIO.66	■	■
EAC.MD_DIN	Modem voice interface Serial data input	I	W13		▨	■	GPIO.64	■	■
EAC.MD_DOUT	Modem voice interface Serial data output	O	V13	FLAG RDY_RX	▨	■	GPIO.65	■	■
EAC.BT_SCLK	Bluetooth voice interface serial clock	I/O	R8	▨	■	■	GPIO.71	■	■
EAC.BT_FS	Bluetooth voice interface frame synchronization	I/O	P9	▨	■	■	GPIO.72	■	■
EAC.BT_DIN	Bluetooth voice interface serial data input	I	Y3	▨	■	■	GPIO.73	■	■
		I	W16	EAC.AC_RST	▨	LOOP_EAC.BT_DIN	GPIO.118	■	■
EAC.BT_DOUT	Bluetooth voice interface serial data output	O	W4	▨	■	STI.CLK	GPIO.74	■	■
LOOP_EAC.BT_DIN	Loop signal between EAC.BT_DIN pin and EAC.AC_RST pin	O	W16	EAC.AC_RST	EAC.BT_DIN	▨	GPIO.118	■	■

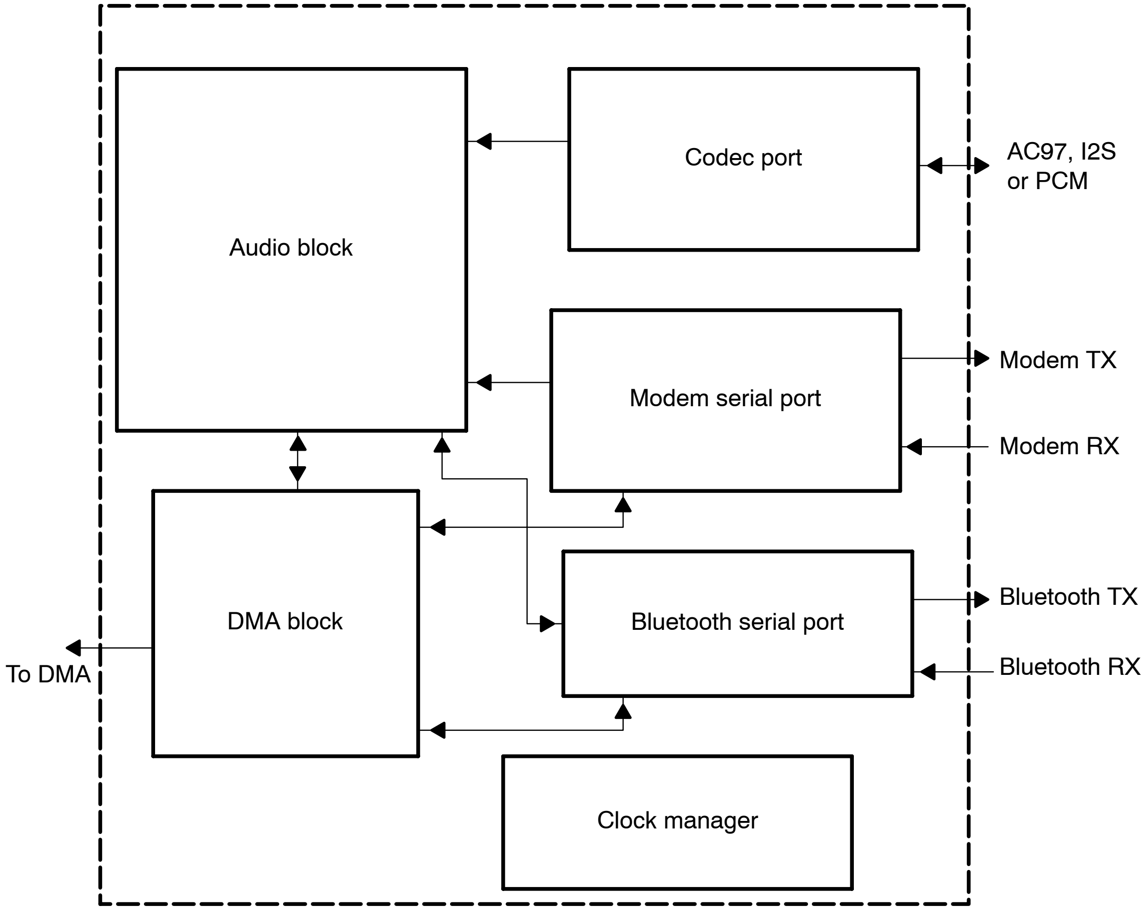
13.8 EAC Functional Description

The EAC is an audio codec peripheral interface designed specifically for applications that require audio and voice data streaming.

13.8.1 Description

Figure 13−42 is a block diagram of the EAC.

Figure 13−42. EAC Block Architecture



The EAC allows interconnection among several endpoints:

❏ Codec (audio, stereo, 44.1-/48-kHz sampling rates)

❏ Modem subsystem (voice, mono, 8-/16-kHz sampling rate)

❏ Bluetooth subsystem (voice, mono, 8-/16-kHz sampling rate)

❏ System memory using DMA (audio files play/record, stereo, various sampling rates)

❏ Audio block that features audio mixing, volume control, and sampling

Table 13−11 lists the possible combinations.

Table 13−11. EAC External Device Combination Table

		EAC Inputs			
		Modem Port 8,16 kHz	**Bluetooth Port 8,16 kHz**	**Codec Port 44.1, 48 kHz**	**DMA Read Output 8, 11.025, 16, 22.05, 24, 32, 44.1, 48 kHz**
EAC Outputs	**Modem Port 8,16 kHz**	–	√[1]	√[2]	√[2]
	Bluetooth Port 8,16 kHz	√[1]	–	√[2]	√[2]
	Codec Port 44.1,48 kHz	√[2]	√[2]	√[2, 3]	√[2]
	DMA Write Output 8, 11.025, 16, 22.05, 24, 32, 44.1, 48 kHz	√[2]	√[2]	√[2]	–

1) Without volume control
2) With volume control
3) Sidetone with volume control

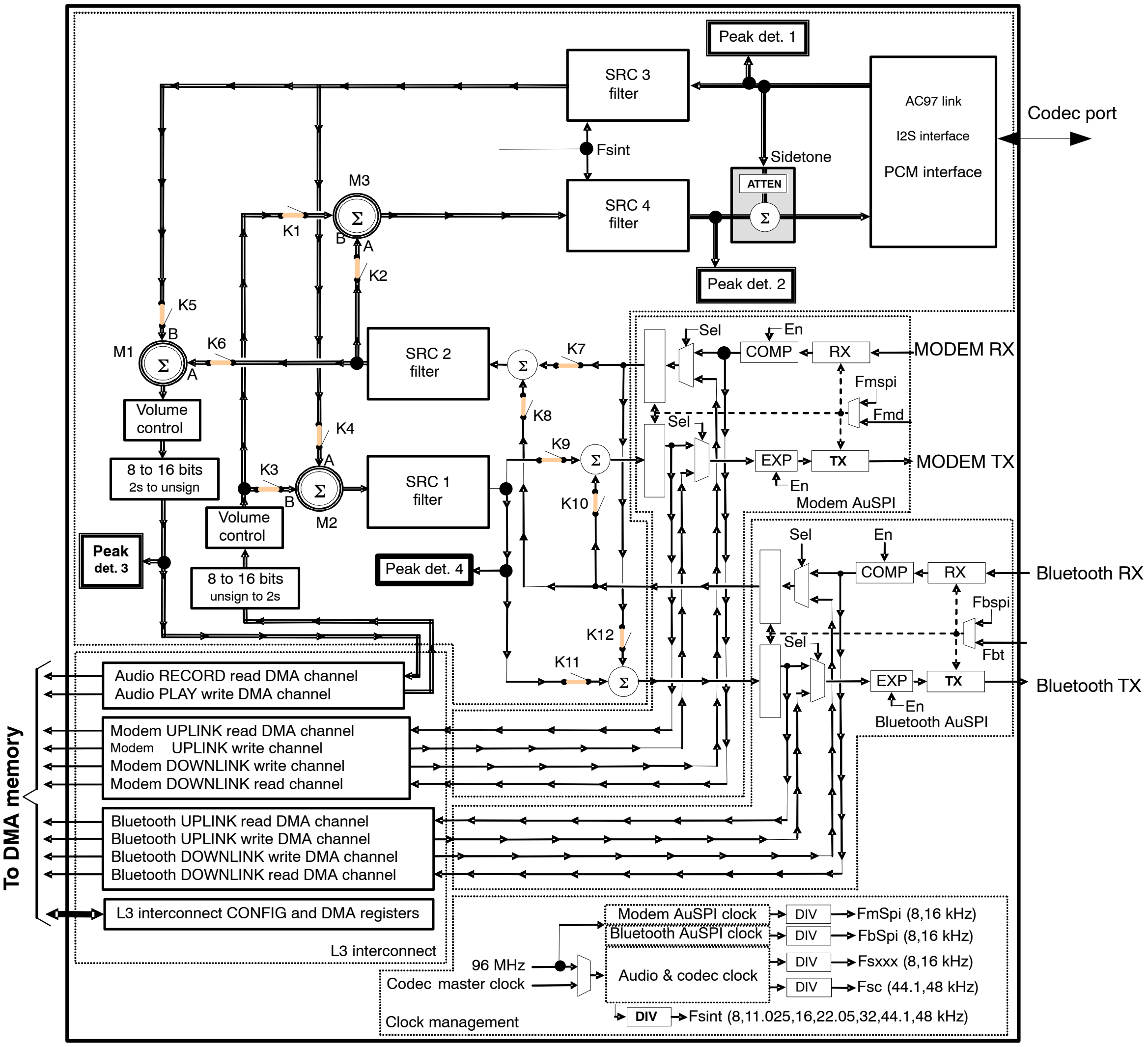
For each line (output), the output can be the sum of the possible inputs.

Connecting one or several of these external devices lets you perform several applications.

Between each application, the audio mixer switches (K) configuration must be changed.

Figure 13−44 shows the details of the EAC architecture shown in Figure 13−43.

Figure 13−44. EAC Block Architecture



13.8.1.1 Codec Serial Port

The codec serial port can be configured to support several industry-standard serial-interface protocols, including the AC97, I2S, and PCM modes. The codec port can be configured in master or slave mode.

13.8.1.2 Modem Serial-Port Block

The modem serial-port block includes the following interfaces and hardware:

- Serial port interfaces (AuSPIs): Handle voice data transfers between EAC and Bluetooth subsystems. AuSPIs can be configured to work as master or slave serial-port interfaces (SPIs).
- Companding hardware: Allows compression and expansion of data in either μ-law or A-law. This allows 14 or 13 bits of dynamic range, respectively. Any value outside this range is set to the most positive or negative value. The μ-law and A-law formats encode data into 8-bit code words. Companded data is always 8 bits wide.

13.8.1.3 Bluetooth Serial Port Block

The Bluetooth serial-port block includes the following interfaces and hardware:

- AuSPIs: Handle voice data transfers between EAC and modem subsystems. AuSPIs can be configured to work as master or slave SPIs.
- Companding hardware: Allows compression and expansion of data in either μ-law or A-law. This allows 14 or 13 bits of dynamic range, respectively. Any value outside this range is set to the most positive or negative value. The μ-law and A-law formats encode data into 8-bit code words. Companded data is always 8 bits wide.

13.8.1.4 Using DMA Requests

External CPU/DSP processing can be optionally inserted on data flow from/to both AuSPI interfaces. Four independent DMA requests per interface, individually selectable, allow read and/or write operations in system memory for uplink (AuSPI transmit) and downlink (AuSPI receive) paths.

- DMA_DL_WR (downlink memory write): Received data flow is stored in memory.
- DMA_DL_RD (downlink memory read): Processed data flow is re-input from memory to EAC.
- DMA_UL_WR (uplink memory write): Data flow from EAC audio/voice mix is stored in memory.
- DMA_UL_RD (uplink memory read): Processed data flow is re-input from memory to be transmitted.

The EAC includes two other independent DMA requests to play and/or record audio files from/to system memory:

- DMA_AC_RD (audio channel memory read): Audio file play from memory
- DMA_AC_WR (audio channel memory write): Audio file record into memory

The EAC also includes the following:

- Digital sampling rate conversion filters (SRC 1, 2, 3, 4, 5) to obtain the selected sample frequency and mix the signal sources
- Transparent DMA mode
  - In codec port transparent DMA mode, the codec interface is used in slave mode. The sent (or received) audio data bypass the sample rate converters and come directly to or go directly from the audio play (or record) DMA interface. The frequency of DMA requests equals the external frame synchronization. In the case of incorrect frame synchronization (clock glitches, inactivated frame synchronization) in I2S-like modes (I2S and polarity-changed I2S) in the transparent

mode, the module resynchronizes the transmit/receive for the external frame-synchronization signal.

  - In AuSPI (Bluetooth or modem) port transparent DMA mode, the interface is used in slave mode. The sent or received data come directly from or go directly to the dedicated DMA DL write channel (or dedicated DMA UL read channel). The frequency of the DMA requests equals the external frame synchronization.
  - The CP_TR_DMA bit of the AGCFR3 register enables the codec port transparent DMA.
  - The BT_TR_DMA bit of the AGCFR3 register enables the Bluetooth transparent DMA.
  - The MD_TR_DMA bit of the AGCFR3 register enables the modem transparent DMA.

- Sidetone: Close to the C-port, the sidetone adds a part of the input to the codec audio link to the uplink audio stream. It consists of a mixer and an attenuator. This block inserts a delay of less than four FS clock periods. In the worst case, the inserted delay is less than 0.1 ms.

  Three digital mixers make different audio scenarios possible:

  - Record the phone call in normal communication.
  - Play a system sound in normal communication.
  - Listen to music in normal communication.

  The digital mixers combine inputs according to the settings in the mixer X volume control registers.

  Gain level is given by:

  $$GAIN = 12\ dB - 0.5 * (127 - DATA)\ dB$$

- Audio mixer volume control

  Each mixer has a corresponding audio mixer volume control register that allows volume control of each mixer input.

- DMA volume control

  For audio transferred by or sent to DMA read or write channels, the left and right data volumes are controllable. All volume controls operate in 0.5-dB steps.

  Gain level is given by:

  $$GAIN = 12\ dB - 0.5 * (255 - DATA)\ dB$$

  Gain level on each DMA channel is configured by the audio master volume control register.

- Peak detector 1 (codec port input):

  This peak detector monitors the signal level on samples received from the codec (that is, the signal level of the external codec microinput).

Reading the APD1LCR register gives the signal level on the left channel, and the APD1RCR register is used for the right channel.

- Peak detector 2 (codec port output):

  This peak detector monitors the signal level on the SRC1 output after the sidetone mixer (that is, the level of the signal samples sent to the external codec).

  Reading the APD2LCR register gives the signal level on the left channel, and the APD2RCR register is used for the right channel.

- Peak detector 3 (DMA record):

  This peak detector monitors the signal level of the audio file written to memory through the DMA write channel (that is, the recording level).

  Reading the APD3LCR register gives the signal level on the left channel, and the APD3RCR register is used for the right channel.

- Peak detector 4 (AuSPI port output):

  This peak detector monitors the signal level on the SRC1 output (that is, the level of the signal samples sent through the modem and Bluetooth output ports).

13.8.1.5 Clock Manager

The EAC operates from a fixed functional clock of 96 MHz. It generates the audio clock needed to support I2S and AC97 (11.2896 and 12.288 MHz) from this 96-MHz clock. The external audio clock from the codec is also allowed.

The clock manager also generates the following clocks and sampling frequencies:

- Modem and Bluetooth clocks from the 96-MHz clock
- Sampling frequencies from audio play/record DMA at 8 kHz, 11.025 kHz, 16 kHz, 22.05 kHz, 24 kHz, 32 kHz, 44.1 kHz, and 48 kHz

Section 13.8.2, *Audio Applications*, includes a list of applications.

Note:

The EAC can operate in either slave or master mode using the MPMCCFR[8] (MCM) register, as follows:

0x0: Slave mode

0x1: Master mode

Figure 13−45 shows the functional clock tree with possible input clocks and possible output clocks for the EAC in master mode.

Figure 13–45. Clock Manager Schematic

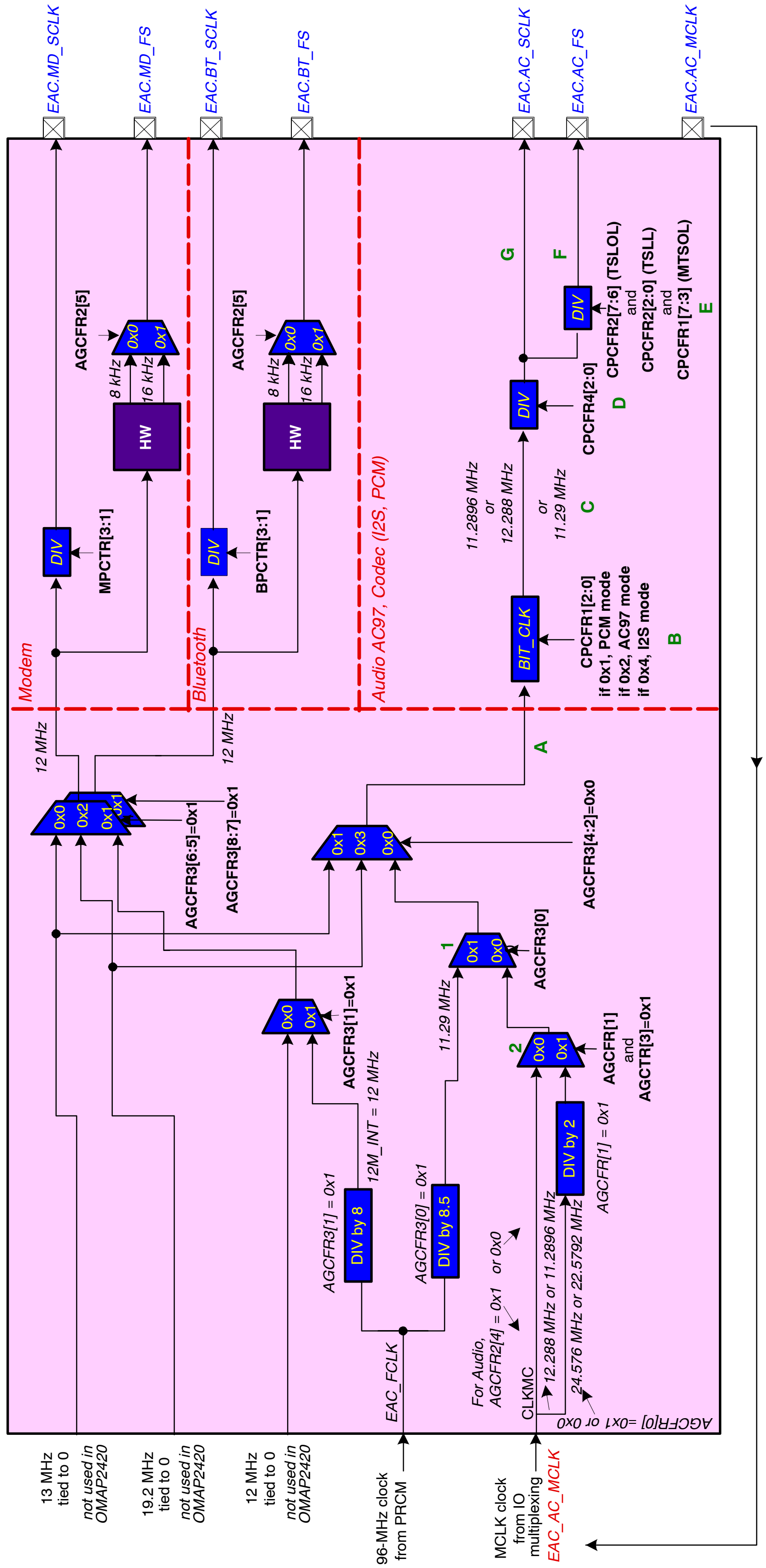


Table 13−12 details different possibilities for the EAC connected to an audio AC97 or a codec I2S/PCM.

Table 13−12. Possible Cases for Audio and Codec Parts

Signals	Description	Audio AC97	Codec I2S	Codec PCM
A	Possible values as input	12.288 MHz	11.2896 MHz	11.2896 MHz
			12.288 MHz	12.288 MHz
			11.29 MHz	11.29 MHz
B	Choose between audio AC97 or codec I2S/PCM	CPCFR1[2:0] (MODE): 0x2	CPCFR1[2:0] (MODE): 0x4	CPCFR1[2:0] (MODE): 0x1
C	Possible values after choice	12.288 MHz	11.2896 MHz	11.2896 MHz
			12.288 MHz	12.288 MHz
			11.29 MHz	11.29 MHz
D	Prescaler	CPCFR[2:0] (DIVB): 0x0 No division	CPCFR[2:0] (DIVB): 0x0: No division (DIVB=1) 0x1: Division by 2 0x2: Division by 3 0x3: Division by 4 0x4: Division by 5 0x5: Division by 6 0x6: Division by 7 0x7: Division by 8 * *[1]	CPCFR[2:0] (DIVB): 0x0: No division (DIVB=1) 0x1: Division by 2 0x2: Division by 3 0x3: Division by 4 0x4: Division by 5 0x5: Division by 6 0x6: Division by 7 0x7: Division by 8 * *[1]
E	MTSL (Number of time slots per audio frame)	CPCFR1[7:3] (MTSL) 0x7: 8 time slots per frame	CPCFR1[7:3] (MTSL) 0x0: One time slot per frame 0x1: Two time slots per frame ... etc. 0x1F: 32 time slots per frame	CPCFR1[7:3] (MTSL) 0x0: One time slot per frame 0x1: Two time slots per frame 0x1F: 32 time slots per frame
	TSLOL (time slot length for slot 0 in number of CLK_BIT serial clock cycles)	CPCFR2[7:6] (TSLOL) 0x0: same as TSLL	CPCFR2[7:6] (TSLOL) 0x0: Same as TSLL 0x1: 8 CLK_BIT cycles 0x2: 16 CLK_BIT cycles 0x3: 32 CLK_BIT cycles	CPCFR2[7:6] (TSLOL) 0x0: Same as TSLL 0x1: 8 CLK_BIT cycles 0x2: 16 CLK_BIT cycles 0x3: 32 CLK_BIT cycles
	TSLL (time slot length in number of CLK_BIT serial clock cycles)	CPCFR2[2:0] (TSLL) 0x5: 32 CLK_BIT cycles	CPCFR2[2:0] (TSLL) 0x0: 8 CLK_BIT cycles 0x1: 16 CLK_BIT cycles 0x2: 18 CLK_BIT cycles 0x3: 20 CLK_BIT cycles 0x4: 24 CLK_BIT cycles 0x5: 32 CLK_BIT cycles	CPCFR2[2:0] (TSLL) 0x0: 8 CLK_BIT cycles 0x1: 16 CLK_BIT cycles 0x2: 18 CLK_BIT cycles 0x3: 20 CLK_BIT cycles 0x4: 24 CLK_BIT cycles 0x5: 32 CLK_BIT cycles

Table 13−12. Possible Cases for Audio and Codec Parts (Continued)

Signals	Description	Audio AC97	Codec I2S	Codec PCM
F	Value of EAC.AC_FS	FS = C/DIVB 12.288 MHz	FS = C/DIVB	FS =C/DIVB
G	Value of EAC.AC_SCLK	SCLK= C/(MTSL*TSLL) 48 kHz	SCLK= C/(DIVB*MTSL*TSLL) 44.1 kHz 48 kHz	SCLK= C/(DIVB*MTSL*TSLL) 44.1 kHz 48 kHz

1) To obtain the frequencies highlighted in red and blue, select the value of DIVB enclosed in asterisks: * *.

The following example shows the serial clock (EAC.AC_SCLK) and frame-synchronization clock (EAC.AC_FS) values:

- For I2S codec
  - A = 11.2896 MHz
  - C = 11.2896 MHz
  - D = 0x7
  - E (MTSL) = 0x0
  - E (TSLOL) = 0x0
  - E (TSLL) = 0x5
  - EAC.AC_SCLK = 1.4112 MHz
  - EAC.AC_FS = G = 44.1 kHz
- For PCM codec
  - A = 12.2896 MHz
  - C = 12.2896 MHz
  - D = 0x7
  - E (MTSL) = 0x0
  - E (TSLOL) = 0x0
  - E (TSLL) = 0x5
  - EAC.AC_SCLK = 1.5362 MHz
  - EAC.AC_FS = G = 48 kHz

Note:

- D, E, and F are also valid for for I2S and PCM codecs. The highlighted lines are examples only.
- The value of F is fixed for audio AC97 (normalization).
- The value of G is fixed for audio AC97, codec I2S, and codec PCM (normalization).

CAUTION

Acceptable performance requires the following settings:

AGCFR3[1] = 0x1
AGCFR3[8:7] = 0x1
AGCFR3[6:5] = 0x1
AGCFR3[4:2] = 0x0
Enabling MCLK as the clock input requires the following setting:

AGCTR[3] = 0x1

13.8.2 Audio Applications

13.8.2.1 Normal Phone Call

- The modem AuSPI port and codec port are used.
- Voice coming from the modem AuSPI port is sent to the codec port. Voice coming from the codec device is sent to the modem AuSPI.
- Bluetooth AuSPI port input is not used.
- Audio mixer switch configuration: AMSCFR = 0x014A

13.8.2.2 Application: Play Audio File and Record Message

- The modem and Bluetooth devices are not used.
- The codec is set to I2S mode.
- Voice coming from the codec port is recorded in memory by the DMA.
- Audio file from memory through DMA can also be played on the codec.
- Audio mixer switch configuration: AMSCFR = 0x0010

13.8.2.3 Application: Normal Phone Call With Play

- Modem AuSPI port, codec port, and DMA are used.
- Play a system sound in normal communication. The voice recorded in memory is sent only to the codec port by DMA and not to the modem AuSPI port. The Bluetooth port is not used.
- Audio mixer switch configuration: AMSCFR = 0x014F

13.8.2.4 Application: Normal Phone Call With Record

- The modem AuSPI port, codec port, and DMA are used.
- Voice coming from the modem AuSPI port or codec port is recorded in memory by the DMA. The Bluetooth port is not used.
- Audio mixer switch configuration: AMSCFR = 0x017A

13.8.2.5 Application: Communication With Headset

- Communication with headset. The modem AuSPI port and the Bluetooth AuSPI port communicate bidirectionally.
- The DMA and codec ports are not used.
- Audio mixer switch configuration: AMSCFR = 0x0A00

13.8.2.6 Application: Communication With Headset and Record

- The Bluetooth AuSPI port, modem AuSPI port, and DMA are used.
- Communication with headset and record. The modem AuSPI port and the Bluetooth AuSPI port communicate bidirectionally. Voice coming from the modem AuSPI port is recorded in memory by the DMA. The codec port is not used.
- Audio mixer switch configuration: AMSCFR = 0x0A60

13.8.2.7 Application: Normal Phone Call With Listen to Music

- The modem AuSPI port, codec port, and DMA are used.
- Listen to music in normal communication so that caller and person being called can both hear. Voice recorded in memory is sent to the codec port and modem AuSPI port by the DMA. The Bluetooth AuSPI is not used.
- Audio mixer switch configuration: AMSCFR = 0x0E04

13.9 EAC Programming Model

As mentioned in Section 13.8, *EAC Functional Description*, with the help of external subsystems such as modem, Bluetooth, codec, and/or DMA memory, the EAC module can carry out multiple applications.

This section shows the programming model for some of the applications listed in Section 13.8.

13.9.1 Configuring the Codec

13.9.1.1 Configuring the Codec Port in AC97 Mode

CPCFR (codec port configuration register) registers must be configured as shown in this section.

CPCFR1: 0X62

- AC97 mode: MODE = 0X2
- 13 time slots per audio frame: MTSL = 0X0C

Table 13−13 shows the CPCFR1 configuration for AC97 mode.

Table 13−13. CPCFR1 Configuration for AC97 Mode

CFCFR1	RESERVED								MTSL					MODE		
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Val	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0

CPCFR2: 0X8B

- 16 serial clock cycles for time slot 0: TSOL = 0X2
- 16 data bits per audio time slot: BPTSL = 0X1 (only the 16 MSBs are used)
- 20 serial clock (BIT_CLK) cycles for all time slots except time slot 0: TSLL = 0X3

Table 13−14 shows the CPCFR2 configuration for AC97 mode.

Table 13−14. CPCFR2 Configuration for AC97 Mode

CFCFR2	RESERVED								TSOL		BPTSL			TSLL		
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Val	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	1

CPCFR3: 0XDA

- One BIT_CLK cycle delay of the serial data output and input signals in reference to the leading edge of the SYNC signal: DDLY = 0x1
- Data serial output enabled during the audio frame is not valid: TRSEN = 0X1.

- The CSYNC signal is generated at the positive edge of the CSCLK signal. Also, the CDATO signal is generated at the positive edge of the CSCLK signal, and the CDATI signal is sampled at the negative edge of the CSCLK signal: CLKBP = 0X0.
- The polarity of the codec port interface frame synchronization output signal (CSYNC) is active high: CSYNCP = 0x1.
- The length of the codec port interface frame synchronization output signal (CSYNC) is the same number of CSCLK cycles as for time slot 0: CSYNCL = 0X1.
- The direction of the codec port interface serial clock signal (CP_SCLK) is an input of the device: CSCLKD = 0x1.
- The direction of the codec port interface frame-synchronization signal (CP_SYNC) is an input from the device: CSYNCD = 0x1.

Table 13−15 shows the CPCFR3 configuration for AC97 mode.

Table 13−15. CPCFR3 Configuration for AC97 Mode

CPCFR3	Reserved								DDLY	TRSEN	CLKBP	CSYNCP	CSYNCL	RESERVED	CSCLKD	CSYNCD
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Val	0	0	0	0	0	0	0	0	1	1	0	1	1	0	1	1

CCPCFR4: 0X10

- Time slot 1 is used for the address and time slot 2 is used for the data: ASTL = 0X01.
- The source of the clock signal to be used for the divide-by-B circuit is the codec master system clock: CLKS = 0 (deleted).
- No divider on MCLK to generate CSCLK (MCLK = 12.288 MHz, 256 bits per frame: frame synchronization frequency = 48 kHz). DIVB = 0X0

Table 13−16 shows the CPCFR4 configuration for AC97 mode.

Table 13−16. CPCFR4 Configuration for AC97 Mode

CPCFR4	RESERVED								ATSL				CLKS	DIVB		
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Val	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

13.9.1.2 Configuring the Codec Port in I2S Mode

The CPCFR registers must be configured as shown in this section.

CPCFR1: 0X0C

- I2S mode: MODE = 0X4

- ❑ Two time slots per audio frame: MTSL = 0X1

Table 13−17 shows the CPCFR1 configuration for I2S mode.

Table 13−17. CPCFR1 Configuration for I2S Mode

CPCFR1	RESERVED								MTSL					MODE		
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Val	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0

CPCFR2: 0X0D

- ❑ Serial clock cycles for time slot 0 are the same as for other time slots: TSOL = 0X0.
- ❑ 16 data bits per audio time slot: BPTSL = 0X1 (only the 16 MSBs are used)
- ❑ 32 serial clock (BCLK) cycles for all time slots except time slot 0: TSLL = 0X5

Table 13−18 shows the CPCFR2 configuration for I2S mode.

Table 13−18. CPCFR2 Configuration for I2S Mode

CPCFR2	RESERVED								TSOL		BPTSL			TSLL		
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Val	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1

CPCFR3: 0XE8

- ❑ One BIT_CLK cycle delay of the serial data output and input signals in reference to the leading edge of the SYNC signal: DDLY = 0x1
- ❑ Data serial output enabled during the audio frame is not valid: TRSEN = 0X1.
- ❑ The CSYNC signal is generated at the negative edge of the CSCLK signal. Also, the CDATO signal is generated at the negative edge of the CSCLK signal, and the CDATI signal is sampled at the positive edge of the CSCLK signal: CLKBP = 0X1.
- ❑ The polarity of the codec port interface frame synchronization output signal (CSYNC) is active low during time slot 0: CSYNCP = 0x0 (time slot 0 = left sample).
- ❑ The length of the codec port interface frame synchronization output signal (CSYNC) is the same number of CSCLK cycles as for time slot 0: CSYNCL = 0X1.
- ❑ The codec port interface serial clock signal (CP_SCLK) is an output of the device: CSCLKD = 0x0.
- ❑ The codec port interface frame-synchronization signal (CP_SYNC) is an output from the device: CSYNCD = 0x0.

Table 13−19 shows the CPCFR3 configuration for I2S mode.

Table 13−19. CPCFR3 Configuration for I2S Mode

CPCFR3	RESERVED								DDLY	TRSEN	CLKBP	CSYNCP	CSYNCL	RESERVED	CSCLKD	CSYNCD
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Val	0	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0

CPCFR4: 0x03

- ❑ No address/data time slot: ATSL = 0X0
- ❑ The source of the clock signal to be used for the divide-by-B circuit is the codec master system clock: CLKS = 0 (deleted).
- ❑ Divide by 4 (MCLK = 11.289 MHz, 32 serial clock cycles per time slot: MCLK must be divided by 4 to achieve a 44.1-kHz frame frequency): DIVB = 0x3.

Table 13−20 shows the CPCFR4 configuration for I2S mode.

Table 13−20. CPCFR4 Configuration for I2S Mode

CPCFR4	RESERVED								ATSL				CLKS	DIVB		
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Val	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

13.9.1.3 Configuring the Codec Port in PCM Mode

This section describes how to configure the CPCFR registers.

CPCFR1: 0x09

- ❑ PCM mode: MODE = 0x1
- ❑ Two time slots per audio frame: MTSL = 0x1

Table 13−21 shows the CPCFR1 configuration for PCM mode.

Table 13−21. CPCFR1 Configuration for PCM Mode

CPCFR1	RESERVED								MTSL					MODE		
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Val	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1

CPCFR2: 0x0D

- ❑ Serial clock cycles for time slot 0 are the same as for other time slots: TSOL = 0x0
- ❑ 16 data bits per audio time slot: BPTSL = 0x1 (only the 16 MSBs are used)
- ❑ 32 serial clock (BCLK) cycles for all time slots except time slot 0: TSLL = 0x5

Table 13−22 shows the CPCFR2 configuration for PCM mode.

Table 13−22. CPCFR2 Configuration for PCM Mode

CPCFR2	RESERVED								TSOL		BPTSL			TSLL		
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Val	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1

CPCFR3: 0x78

- ❑ No BIT_CLK cycle delay of the serial data output and input signals in the reference to the leading edge of the SYNC signal: DDLY = 0x0
- ❑ Data serial output is enabled during the audio frame and is not valid: TRSEN = 0x1.
- ❑ The CSYNC signal is generated with the negative edge of the CSCLK signal. Also, the CDATO signal is generated with the negative edge of the CSCLK signal, and the CDATI signal is sampled with the positive edge of the CSCLK signal: CLKBP = 0x1.
- ❑ The polarity of the codec port interface frame synchronization output signal (CSYNC) is active high for time slot 0: CSYNCP = 0x1 (time slot 0: left sample).
- ❑ The length of the codec port interface frame synchronization output signal (CSYNC) is the same number of CSCLK cycles as for time slot 0: CSYNCL = 0x1.
- ❑ The direction of the codec port interface serial clock signal (CP_SCLK) is an output of the device: CSCLKD = 0x0.
- ❑ The direction of the codec port interface frame-synchronization signal (CP_SYNC) is an output from the device: CSYNCD = 0.

Table 13−23 shows the CPCFR3 configuration for PCM mode.

Table 13−23. CPCFR3 Configuration for PCM Mode

CPCFR3	Reserved								DDLY	TRSEN	CLKBP	CSYNCP	CSYNCL	Reserved	CSCLKD	CSYNCD
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Val	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0

CPCFR4: 0xF3

- ❑ No address/data time slot, but used as a workaround: ATSL = 0xF
- ❑ The source of the clock signal to be used for divide-by-B circuit is the codec master system clock: CLKS = 0 (deleted).
- ❑ Divide by 4 (MCLK = 11.289 MHz, 64 CSCLK cycle per frame: MCLK must be divided by 4 to achieve a 44.1-kHz frame frequency): DIVB = 0x3

Table 13−24 shows the CPCFR4 configuration for PCM mode.

Table 13−24. CPCFR4 Configuration for PCM Mode

CPCFR4	Reserved								ATSL		CLKS			DIVB		
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Val	0	0	0	0	0	0	0	0	1	1	1	1	0	0	1	1

13.9.1.4 Configuring the Codec Port in I2S Polarity-Changed

The CPCFR registers must be configured as shown in this section.

CPCFR1: 0x09

- Polarity-changed I2S: MODE = 0x1
- Two time slots per audio frame: MTSL = 0x1

Table 13−25 shows the CPCFR1 configuration for polarity-changed I2S mode.

Table 13−25. CPCFR1 Configuration for Polarity-Changed I2S Mode

CPCFR1	RESERVED								MTSL					MODE		
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Val	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1

CPCFR2: 0x0B

- Serial clock cycles for time slot 0 are the same as for other time slots: TSOL = 0x0
- 16 data bits per audio time slot: BPTSL = 0x1 (only the 16 MSBs are used)
- 20 serial clock (BCLK) cycles for all time slots except time slot 0: TSLL = 0x3

Table 13−26 shows the CPCFR2 configuration for polarity-changed I2S mode.

Table 13−26. CPCFR2 Configuration for Polarity-Changed I2S Mode

CPCFR2	RESERVED								TSOL		BPTSL			TSLL		
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Val	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1

CPCFR3: 0xFB

- One BIT_CLK cycle delay of the serial data output and input signals in reference to the leading edge of the SYNC signal: DDLY = 0x1
- Data serial output is enabled during the audio frame and is not valid: TRSEN = 0x1.
- The CSYNC signal is generated with the negative edge of the CSCLK signal. Also, the CDATO signal is generated with the negative edge of the CSCLK signal and the CDATI signal is sampled with the positive edge of the CSCLK signal: CLKBP = 0x1.
- The polarity of the codec port interface frame-synchronization output signal (CSYNC) is active high for time slot 0: CSYNCP = 0x1 (time slot 0: left sample).
- The length of the codec port interface frame-synchronization output signal (CSYNC) is the same number of CSCLK cycles as for time slot 0: CSYNCL = 0x1.

- ❑ The direction of the codec port interface serial clock signal (CP_SCLK) is the input of the device: CSCLKD = 0x1.
- ❑ The direction of the codec port interface frame-synchronization signal (CP_SYNC) is the input of the device: CSYNCD = 0x1.

Table 13−27 shows the CPCFR3 configuration for polarity-changed I2S mode.

Table 13−27. CPCFR3 Configuration for Polarity-Changed I2S Mode

CPCFR3	Reserved								DDLY	TRSEN	CLKBP	CSYNCP	CSYNCL	Reserved	CSCLKD	CSYNCD
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Val	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	1

CPCFR4: 0xF3

- ❑ No address/data time slot, but used as a workaround: ATSL = 0xF
- ❑ The source of the clock signal to be used for divide-by-B circuit is the codec master system clock: CLKS = 0 (deleted).
- ❑ Divide by 4 (MCLK = 11.289 MHz, 64 CSCLK cycle per frame: MCLK must be divided by 4 to achieve a 44.1-kHz frame frequency): DIVB = 0x3 (not relevant in slave mode).

Table 13−28 shows the CPCFR4 configuration for polarity-changed I2S mode.

Table 13−28. CPCFR4 Configuration for Polarity-Changed I2S Mode

CPCFR4	Reserved								ATSL		CLKS			DIVB		
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Val	0	0	0	0	0	0	0	0	1	1	1	1	0	0	1	1

13.9.2 Programming Model for Application: Play Audio File and Record Message

In this application, the modem and Bluetooth devices are not used.

The codec is set to I2S mode.

Voice coming from the codec port is recorded in memory by the DMA.

Audio files from memory through DMA can also be played on the codec.

This section provides a configuration example. You can change some of the following register values depending on your application needs.

Step 1: Set SYSCONFIG (software reset): 0x2.

Step 2: Set audio global control register (AGCTR): 0x000A.

- ■ The internal MCLK is enabled: MCLK_EN = 0x1.

- Audio processing is enabled: AUDEN = 0x1.

Step 3: Wait until the SYSSTATUS[0] bit is set to 1 (OCP_RESET_DONE).

Step 4: Set AGCFR3 (codec port configuration register 3): 0x04A3.

Set the frequency FSINT depending on application needs.

Example:

- FSINT is forced to 16 kHz: FSINT = 0x2.
- Bluetooth port clock is forced to 12 MHz by CLK12M_INT: BT_CKSRC = 0x1.
- Modem port clock is forced to 12 MHz by CLK12M_INT: MD_CKSRC = 0x1.
- MCLK_INT clock is used: AUD_CKSRC = 0x0.
- The internal 12-MHz clock comes from the 96-MHz clock divided by 8: CLK12M_INT = 0x1.
- The codec master clock comes from the 96-MHz clock divided by 8.5 (11.28 MHz): MCLKINT_SEL = 0x1.

Note:

Setting AUD_CKSRC in the AGCFR3 register to 0x0 selects MCLK_INT as the clock.

Setting BT_CKSRC or MD_CKSRC in the AGCFR3 register to 0x1 or AUD_CKSRC to 0x2 selects CLK12M_INT as the clock source.

You can select FSINT using the AGCFR3, AGCFR2, or AGCFR register. You must configure the modem, Bluetooth device, or audio codec port clocks using only the AGCFR3 or AGCFR register.

Step 5: Set AGCFR (codec port configuration register): 0x0648

Enables 16-bit stereo files and little-endianism.

Step 6: Wait until the SYSSTATUS[3] bit is set to 1 (AUD_RESET_DONE).

Step 7: Set AMSCFR (audio mixer switch configuration): 0x0011.

Only switches K1 and K5 are closed.

Step 8: Set CPCFR1, CPCFR2, CPCFR3, and CPCFR4 (codec port configuration registers):

See configuration in codec mode I2S in Section 13.9.1.2, *Configuring the Codec Port in I2S Mode*.

Step 9: Set CPTCTL (codec port interface control and status register): 0x28.

The C_PORT interface must be enabled: CPEN = 0x1.

Step 10: Set AGCTR (audio global control register): 0x180A.

- The audio file play operation DMAREN = 0x1.

- The message record operation DMAWEN = 0x1.
- Audio processing is enabled AUDEN = 0x1.

13.9.3 Programming Model for Application: Normal Phone Call With Record

This application uses the modem and codec devices. The DMA records voice coming from the modem AuSPI port and the codec port in memory.

The Bluetooth port is not used.

Step 1: Set SYSCONFIG (software reset): 0x2.

Step 2: Set AGCTR (audio global control register): 0x000A.

- The internal MCLK is enabled: MCLK_EN = 0x1.
- Audio processing is enabled: AUDEN = 0x1.

Step 3: Wait until the SYSSTATUS [0] bit is set to 1 (L4_INTERFACE_RESET_DONE).

Step 4: Set AGCFR3 (audio global configuration register 3): 0x04A3.

Set the frequency FSINT depending on application needs.

Example:

- FSINT is forced to 16 kHz: FSINT = 0x2.
- Bluetooth port clock is forced to 12 MHz by CLK12M_INT: BT_CKSRC = 0x1.
- Modem port clock is forced to 12 MHz by CLK12M_INT: MD_CKSRC = 0x1.
- MCLK_INT clock is used: AUD_CKSRC = 0x0.
- The internal 12-MHz clock comes from the 96-MHz clock divided by 8: CLK12M_INT = 0x1.
- The codec master clock comes from the 96-MHz clock divided 8.5 (11.28 MHz): MCLKINT_SEL = 0x1.

Note:

Setting AUD_CKSRC in the AGCFR3 register to 0x0 selects MCLK_INT as the clock.

Setting BT_CKSRC or MD_CKSRC in the AGCFR3 register to 0x1 or AUD_CKSRC to 0x2 selects CLK12M_INT as the clock source.

You can select FSINT using either the AGCFR3, AGCFR2, or AGCFR register. You must configure the modem, Bluetooth device, or audio codec port clocks using only the AGCFR3 or AGCFR register.

Step 5: AGCFR (audio global configuration register): 0x0248

Enables 16-bit mono audio files and little-endianism.

Step 6: Wait until the SYSSTATUS [5][4][3][2] bits (AUD_RESET_DONE, AUDRHEA_RESET_DONE, CP_RESET_DONE, and MD_RESET_DONE) are set to 1.

Step 7: Set AMSCFR (audio mixer switches configuration): 0x017C

Switches K2, K4, K5, K6, K7, and K9 are closed.

Step 8: Set CPCFR1, CPCFR2, CPCFR3, and CPCFR4:

See the configuration in codec mode AC97, Section 13.9.1.1, *Configuring the Codec Port in AC97 Mode*.

Step 9: Set CPTCTL (codec port interface control and status register): 0x28.

The C_PORT interface must be enabled: CPEN = 0x1

Step 10: Set MPMCCFR (modem port main configuration register): 0x010F.

- Fields MCM, FSP_MC, FSBL_MC, and CPB_MC of the MPMCCFR register are set according to the configuration of the modem.

  For this application, all registers are at their reset value. That means that the polarity is falling for the frame synchronization and the clock, and the expand and compand modes are disabled.
- The modem port is set in master mode: MCM = 0x1.
- 16-bit transmission on the channel: WS_MC = 0x0F

Step 11: Set MPCTR (modem port control register): 0x0089.

- The main modem channel is enabled: MC_EN = 0x1.
- The prescale clock divider is 16: PRE_MC = 0x100.
- The AuSPI clock is enabled: CKEN = 0x1.

Step 12: Set AGCTR (audio global control register): 0x0802.

- The message record operation DMAWEN = 0x1.
- Audio processing is enabled: AUDEN = 0x1.

13.9.4 Programming Model for Application: Communication With Headset

The modem AuSPI port and the Bluetooth AuSPI port communicate bidirectionally.

DMA and codec ports are not used.

Step 1: Set SYSCONFIG (software reset): 0x2.

Step 2: Set AGCTR (audio global control register): 0X04A3.

The internal MCLK is enabled: MCLK_EN = 0x1.

The audio processing is enabled: AUDEN = 0x1.

Step 3: Wait until the SYSSTATUS[0] bit is set to 1 (L4_INTERFACE_RESET_DONE).

Step 4: Set AGCFR3 (audio global configuration register 3): 0X04A3.

Set the frequency FSINT depending on application needs.

Example:

- FSINT is forced to 16 kHz: FSINT = 0x2.
- The Bluetooth port clock is forced to 12 MHz from CLK12M_INT: BT_CKSRC = 0x1.
- The modem port clock is forced to 12 MHz from CLK12M_INT: MD_CKSRC = 0x1.
- The MCLK_INT clock is used: AUD_CKSRC = 0x0.
- The internal 12-MHz clock comes from the 96-MHz clock divided by 8: CLK12M_INT = 0x1.
- The codec master clock comes from the 96-MHz clock divided by 8.5 (11.28 MHz): MCLKINT_SEL = 0x1.

Note:

Setting AUD_CKSRC in the AGCFR3 register to 0x0 selects MCLK_INT as the clock.

Setting BT_CKSRC or MD_CKSRC in the AGCFR3 register to 0x1 or AUD_CKSRC to 0x2 selects CLK12M_INT as the clock source.

Select FSINT using the AGCFR3, AGCFR2, or AGCFR register. You must configure the modem, Bluetooth device, and audio codec port clocks using the AGCFR3 or AGCFR register.

Step 5: Set AGCFR2 (audio global configuration register 2): 0X0248.

Enables 16-bit mono audio files and little-endianism.

Step 6: Wait until the SYSSTATUS [5][4][2][1] bits (AUD_RESET_DONE, AUDRHEA_RESET_DONE, MD_RESET_DONE, and BT_RESET_DONE) are 1.

Step 7: Set AMSCFR: 0x0A00.

Switches K10 and K12 are closed.

Step 8: Set MPMCCFR (modem port main configuration register): 0x010F.

- The fields MCM, FSP_MC, FSBL_MC, and CPB_MC are set according to the modem configuration.

  For this application, registers are at their reset values. That means that the polarity is falling for the frame synchronization and the clock, and that expand and compand mode are disabled.
- The modem port is set in master mode: MCM = 0x1.
- 16-bit transmission on the channel: WS_MC = 0x0F

Step 9: Set MPCTR (modem port control register): 0x0089.

- The main modem channel is enabled: MC_EN = 0x1.
- The prescale clock divider is 16: PRE_MC = 0x100.

- The AuSPI clock is enabled: CKEN = 0x1.

Step 10: Set BPMCCFR (modem port main configuration register): 0x010F.

- The fields MCM, FSP_MC, FSBL_MC, and CPB_MC are set according to the configuration of the codec.
- For this application, registers are at their reset values. That means that the polarity is falling for the frame synchronization and the clock, and that expand and compand modes are disabled.
- The Bluetooth port is set in master mode: MCM = 0x1.
- 16-bit transmission on the channel: WS_MC = 0x0F.

Step 11: Set BPCTR (modem port control register): 0x0089.

- The main Bluetooth channel is enabled: MC_EN = 0x1.
- The prescale clock divider is 16: PRE_MC = 0x100.
- The AuSPI clock is enabled: CKEN = 0x1.

13.9.5 Programming Model for Application: Normal Phone Call With Music Listening

Modem AuSPI port, codec port, and DMA are used.

Listen to music in normal communication so that caller and person being called can both hear. Voice recorded in memory is sent to the codec port and modem AuSPI port by DMA.

Bluetooth AuSPI is not used.

Step 1: Set SYSCONFIG (software reset): 0x2.

Step 2: Set AGCTR (audio global control register): 0X000A.

The internal MCLK is enabled: MCLK_EN = 0x1.

Audio processing is enabled: AUDEN = 0x1.

Step 3: Wait until the SYSSTATUS[0] bit is set to 1 (L4_INTERFACE_RESET_DONE).

Step 4: Set AGCFR3 (audio global configuration register 3): 0x0581.

Set the frequency FSINT depending on application needs.

Example:

- FSINT is forced to 16 kHz: FSINT = 0x2.
- The Bluetooth port clock is forced to 12 MHz by CLK12M_INT: BT_CKSRC = 0x1.
- The modem port clock is forced to 12 MHz by CLK12M_INT: MD_CKSRC = 0x1.
- The MCLK_INT clock is used: AUD_CKSRC = 0x0.
- The internal 12-MHz clock comes from the 96-MHz clock divided by 8: CLK12M_INT = 0x1.

- The codec master clock comes from the 96-MHz clock divided 8.5 (11.28 MHz): MCLKINT_SEL = 0x1.

Note:

Setting AUD_CKSRC in the AGCFR3 register to 0x0 selects MCLK_INT as the clock.

Setting BT_CKSRC or MD_CKSRC in the AGCFR3 register to 0x1 or AUD_CKSRC to 0x2 selects CLK12M_INT as the clock source.

Select FSINT using the AGCFR3, AGCFR2, or AGCFR register. You must configure the modem, Bluetooth device, and audio codec port clocks using the AGCFR3 or AGCFR register.

Step 5: Set AGCFR2 (audio global control registers 2): 0x0248.

Enables 16-bit audio mono files and little-endianism.

Step 6: Wait until the SYSSTATUS [5][4][3][2] bits (AUD_RESET_DONE, AUDRHEA_RESET_DONE, CP_RESET_DONE, and MD_RESET_ DONE) are 1.

Step 7: Set AMSCFR **(**audio mixer switches configuration): 0x014B.

Switches K1, K2, K3, K4, K7, and K9 are closed.

Step 8: Set CPCFR1, CPCFR2, CPCFR3, and CPCFR4.

See the configuration in codec mode I2S, Section 13.9.1.2, *Configuring the Codec Port in I2S Mode*.

Step 9: Set CPTCTL (codec port interface control and status register): 0x28.

The C_PORT interface must be enabled: CPEN = 0x1.

Step 10: Set MPMCCFR: 0x010F.

- The fields MCM, FSP_MC, FSBL_MC, and CPB_MC are set according to the modem configuration.
- For this application, registers are at their reset value. That means that the polarity is falling for the frame synchronization and the clock, and that expand and compand modes are disabled.
- 16-bit transmission on the channel: WS_MC = 0x0F

Step 11: Set MPCTR: 0x0089.

- The main modem channel is enabled: MC_EN = 0x1.
- The prescale clock divider is 16: PRE_MC = 0x100.
- The AuSPI clock is enabled: CKEN = 0x1.

Step 12: Set AGCTR (audio global control register): 0x1002.

- The message record operation is enabled: DMAREN = 0x1.
- Audio processing is enabled: AUDEN = 0x1.

13.10 EAC Registers

This section describes the EAC registers, including the module instance, a summary of registers, and individual descriptions of register bits.

13.10.1 Module Register Mapping Summary

Table 13−29 lists the module instance, base address, and size.

Table 13−29. Instance Summary

Module Name	Base Address	Size
EAC	0x4809 0000	64K bytes

13.10.2 EAC Register Summary

Table 13−30 lists the EAC registers with offset addresses.

CAUTION

EAC registers are limited to 16-bit data access; 8- and 32-bit data access are not allowed because they can corrupt register contents.

Table 13−30. EAC Register Summary

Register Name	Type	Register Width (Bits)	Physical Offset
CPCFR1	RW	16	0x0000
CPCFR2	RW	16	0x0004
CPCFR3	RW	16	0x0008
CPCFR4	RW	16	0x000C
CPTCTL	RW	16	0x0010
CPTTADR	RW	16	0x0014
CPTDATL	RW	16	0x0018
CPTDATH	RW	16	0x001C
CPTVSLL	RW	16	0x0020
CPTVSLH	RW	16	0x0024
MPCTR	RW	16	0x0040
MPMCCFR	RW	16	0x0044
BPCTR	RW	16	0x0060
BPMCCFR	RW	16	0x0064
AMSCFR	RW	16	0x0080

Table 13−30. EAC Register Summary (Continued)

Register Name	Type	Register Width (Bits)	Physical Offset
AMVCTR	RW	16	0x0084
AM1VCTR	RW	16	0x0088
AM2VCTR	RW	16	0x008C
AM3VCTR	RW	16	0x0090
ASTCTR	RW	16	0x0094
APD1LCR	R	16	0x0098
APD1RCR	R	16	0x009C
APD2LCR	R	16	0x00A0
APD2RCR	R	16	0x00A4
APD3LCR	R	16	0x00A8
APD3RCR	R	16	0x00AC
APD4R	R	16	0x00B0
ADWR	R	16	0x00B4
ADRDR	R	16	0x00B8
AGCFR	RW	16	0x00BC
AGCTR	RW	16	0x00C0
AGCFR2	RW	16	0x00C4
AGCFR3	RW	16	0x00C8
MBPDMACTR	RW	16	0x00CC
MPDDMARR	R	16	0x00D0
MPDDMAWR	W	16	0x00D4
MPUDMARR	R	16	0x00D8
MPUDMAWR	W	16	0x00E0
BPDDMARR	R	16	0x00E4
BPDDMAWR	W	16	0x00E8
BPUDMARR	R	16	0x00EC
BPUDMAWR	W	16	0x00F0
VERSION_ NUMBER	R	16	0x0100
SYSCONFIG	W	16	0x0104
SYSSTATUS	R	16	0x0108

Table 13−31 lists the target agent (TA) registers for the EAC module. For a detailed description, see Chapter 6, *Internal Interconnect*.

Table 13−31. L4TA32 Register Summary

Register Name	**Type**	**Register Width (Bits)**	**Physical Address**
COMPONENT	R	32	0x4809 1000
AGENT_CONTROL	RW	32	0x4809 1020
AGENT_STATUS	R	32	0x4809 1028

13.10.3 EAC Register Descriptions

Table 13–32 through t list the individual register bits.

Table 13–32. CPCFR1

Address Offset	0x000		
Physical Address	0x4809 0000	**Instance**	EAC1
Description	Codec port configuration register 1		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								MTSL					MODE		

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0.	R	0x00
7:3	MTSL	Number of time slots per audio frame 0x0: One time slot per frame 0x1: Two time slots per frame (default: I2S) ... 0x1F: 32 time slots per frame	RW	0x01
2:0	MODE	Codec port interface mode. Other values: reserved 0x1: PCM mode/polarity-changed I2S mode 0x2: AC97 mode 0x4: I2S mode (default)	RW	0x4

Table 13–33. CPCFR2

Address Offset	0x004		
Physical Address	0x4809 0004	**Instance**	EAC1
Description	Codec port configuration register 2		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TSLOL		BPTSL			TSLL		

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0.	R	0x00
7:6	TSLOL	Time slot 0 length in number of serial clock (CLK_BIT) cycles 0x0: Same as TSLL (default: I2S) 0x1: 8 CLK_BIT cycles 0x2: 16 CLK_BIT cycles 0x3: 32 CLK_BIT cycles	RW	0x0

Table 13–33. CPCFR2 (Continued)

Bits	Field Name	Description	Type	Reset
5:3	BPTSL	Number of data bits per audio time slot 0x0: 8 data bits per time slot 0x1: 16 data bits per time slot (default: I2S) 0x2: 18 data bits per time slot 0x3: 20 data bits per time slot 0x4: 24 data bits per time slot 0x5: 32 data bits per time slot 0x6: Reserved 0x7: Reserved	RW	0x1
2:0	TSLL	Time slot length (all except time slot 0) in number of serial clock (CLK_BIT) cycles 0x0: 8 CLK_BIT cycles 0x1: 16 CLK_BIT cycles (default: I2S) 0x2: 18 CLK_BIT cycles 0x3: 20 CLK_BIT cycles 0x4: 24 CLK_BIT cycles 0x5: 32 CLK_BIT cycles 0x6: Reserved 0x7: Reserved	RW	0x1

Table 13–34. CPCFR3

Address Offset	0x008		
Physical Address	0x4809 0008	**Instance**	EAC1
Description	Codec port configuration register 3		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DDLY	TRSEN	CLKBP	CSYNCP	CSYNCL	RESERVED	CSCLKD	CSYNCD

Table 13–34. CPCFR3 (Continued)

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0.	R	0x00
7	DDLY	Data delay: Data bits start according to SYNC signal leading edge. 0x0: No delay 0x1: One-cycle delay (default: I2S)	RW	1
6	TRSEN	3-state enable: Data serial output state during nonvalid audio frames 0x0: 3-state 0x1: Data serial output is enabled.	RW	0
5	CLKBP	CLOCK POLARITY 0x0: Rising edge (CP_SYNC and serial data are generated on serial clock rising edge, serial data input is sampled on falling edge). 0x1: Falling edge (default: I2S)	RW	1
4	CSYNCP	CP_SYNC (synchronization) polarity 0x0: Active low (default: I2S, time-slot 0 is left time slot) 0x1: Active high	RW	0
3	CSYNCL	CSYNC length. 0x0: Synchronization is a one-serial-clock-cycle pulse. 0x1: Synchronization length is equal to time slot 0 length (default: I2S).	RW	1
2	Reserved	Reserved (byte order)	RW	0
1	CSCLKD	CP_SCLK port (serial clock) direction 0x0: CP_SCLK is an output. 0x1: CP_SCLK is an input.	RW	1
0	CSYNCD	CP_SYNC (synchronization) direction 0x0: CP_SYNC is an output. 0x1: CP_SYNC is an input.	RW	1

Table 13–35. CPCFR4

Address Offset	0x00C		
Physical Address	0x4809 000C	**Instance**	EAC1
Description	Codec port configuration register 4		
Type	RW		

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">Reserved</td><td colspan="4">ATSL</td><td>CLKS</td><td colspan="3">DIVB</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0.	R	0x00
7:4	ATSL	Audio time slots for secondary communication address and data values. Other enums: reserved 0x0: For the I2S mode 0x1: For AC97 mode (time slot 1 for address, time slot 2 for data) 0xF: For PCM mode/polarity-changed I2S	RW	0x0
3	CLKS	Clock source (reserved)	RW	0
2:0	DIVB	CP_SCLK divider value This bit field allows predividing the clock to generate serial clocks. 0x0: No division 0x1: Divide by 2 0x2: Divide by 3 0x3: Divide by 4 (default: I2S) 0x4: Divide by 5 0x5: Divide by 6 0x6: Divide by 7 0x7: Divide by 8	RW	0x3

Table 13–36. CPTCTL

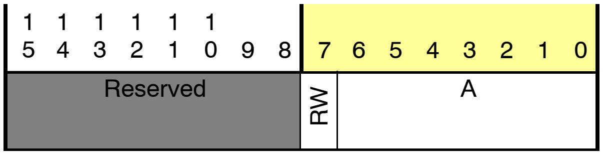
Address Offset	0x010		
Physical Address	0x4809 0010	**Instance**	EAC1
Description	Codec port interface control and status register		
Type	RW		

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">Reserved</td><td>RXF</td><td>RXIE</td><td>TXE</td><td>TXIE</td><td>CPEN</td><td colspan="2">CID</td><td>CRST</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0.	R	0x00
7	RXF	Receive data register full	RW	0
6	RXIE	Receive interrupt enable 0x0: Interrupt generation disabled 0x1: Interrupt generated on receive data register full	RW	0
5	TXE	Transmit data register empty	RW	1
4	TXIE	Transmit interrupt enable 0x0: Interrupt generation disabled 0x1: Interrupt generated on transmit data register empty	RW	0
3	CPEN	C port enable. The C port enable bit is used to control the C port interface. 0x0: Disabled 0x1: Enabled (access to configuration registers is disabled)	RW	0
2:1	CID	AC97 only: Codec ID selection between primary and secondary codec devices	RW	0x0
0	CRST	Codec reset 0x0: CP_NRST output is low. 0x1: CP_NRST output is high.	RW	0

Table 13–37. CPTTADR

Address Offset	0x014		
Physical Address	0x4809 0014	**Instance**	EAC1
Description	Codec port interface address register		
Type	RW		

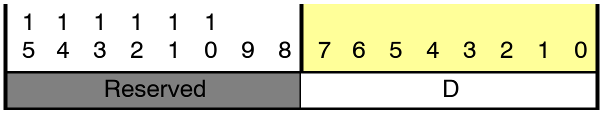


15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								RW	A						

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0.	R	0x00
7	RW	Command/status read/write operation 0x0: Command/status write operation 0x1: Command/status read operation	RW	0
6:0	A	Command/status address value for codec device control/status register to be accessed	RW	0x00

Table 13–38. CPTDATL

Address Offset	0x018		
Physical Address	0x4809 0018	**Instance**	EAC1
Description	Codec port interface data register		
Type	RW		

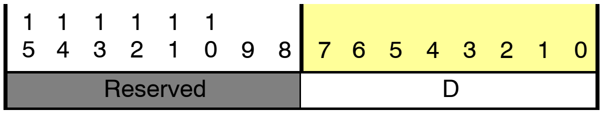


15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								D							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0.	R	0x00
7:0	D	Write command data value (LSB part) to be transmitted to codec device. Read status data value (LSB part) received from codec device.	RW	0x00

Table 13–39. CPTDATH

Address Offset	0x01C		
Physical Address	0x4809 001C	**Instance**	EAC1
Description	Codec port interface data register		
Type	RW		



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								D							

Table 13–39. CPTDATH (Continued)

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0.	R	0x00
7:0	D	Write command data value (MSB part) to be transmitted to codec device. Read status data value (MSB part) received from codec device.	RW	0x00

Table 13–40. CPTVSLL

Address Offset	0x020		
Physical Address	0x4809 0020	**Instance**	EAC1
Description	Codec port interface valid time slots register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								VTOL					RESERVED		

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0.	R	0x00
7:3	VTSL	Valid time slots. Bits 7 to 3 correspond to time slots 8 to 12 in slot 0. TAG set to 1 to indicate audio valid time slot.	RW	0x00
2:0	Reserved	Read returns 0.	R	0x0

Table 13–41. CPTVSLH

Address Offset	0x024		
Physical Address	0x4809 0024	**Instance**	EAC1
Description	Codec port interface valid time slots register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								VF	CRDY1	CRDY2	VTSL				

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0.	R	0x00
7	VF	Valid frame	RW	0
6	CRDY1	Primary codec ready	R	0
5	CRDY2	Secondary codec ready	R	0
4:0	VTSL	Valid time slots	RW	0x00

Table 13−42. MPCTR

Address Offset	0x040		
Physical Address	0x4809 0040	**Instance**	EAC1
Description	Modem port control register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								MC_EN	RESERVED			PRE_MC			CKEN

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0.	R	0x00
7	MC_EN	Main channel enable 0x0: Disabled 0x1: Enabled	RW	0
6:4	Reserved	Read returns 0.	RW	0x0
3:1	PRE_MC	Prescale clock divisor for main channel. Other values: reserved. 0x0: Clock divisor 1 0x1: Clock divisor 2 0x2: Clock divisor 4 0x3: Clock divisor 8 0x4: Clock divisor 16	RW	0x0
0	CKEN	Clock enable 0x0: Clocks are shut off. 0x1: Clocks are running.	RW	0

Table 13−43. MPMCCFR

Address Offset	0x044		
Physical Address	0x4809 0044	**Instance**	EAC1
Description	Modem port main channel configuration register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	DFILL_MC	DJUST_MC	COMPAND		EXPAND		MCM	FSP_MC	FSBL_MC	CPB_MC	WS_MC				

Table 13–43. MPMCCFR (Continued)

Bits	Field Name	Description	Type	Reset
15	Reserved	Read returns 0.	R	0
14	DFILL_MC	Justified data filling value 0x0: Filled with zeros 0x1: Filled with ones	RW	0
13	DJUST_MC	Data justification 0x0: Data justify left (MSB alignment) 0x1: Data justify right (LSB alignment)	RW	0
12:11	COMPAND	Compand mode for main channel transmit register 0x0: Disabled 0x1: μ-law 0x2: A-law 0x3: Reserved	RW	0x0
10:9	EXPAND	Expand mode for main channel receive register 0x0: Disabled 0x1: μ-law 0x2: A-law 0x3: Reserved	RW	0x0
8	MCM	Main channel mode 0x0: Slave mode 0x1: Master mode	RW	0
7	FSP_MC	Frame synchronization polarity for main channel 0x0: Falling 0x1: Rising	RW	0
6	FSBL_MC	Frame synchronization level for main channel 0x0: Low level 0x1: High level	RW	0
5	CPB_MC	Clock polarity for main channel 0x0: Falling 0x1: Rising	RW	0
4:0	WS_MC	The word size bits are set to define the bit number of the transmission for the main channel. 0x0: 1 bit 0x1: 2 bits 0x2: 3 bits 0x3: 4 bits 0x4: 5 bits 0x5: 6 bits 0x6: 7 bits	RW	0x00

Table 13−43. MPMCCFR (Continued)

Bits	Field Name	Description	Type	Reset
4:0 (cont'd)	WS_MC (cont'd)	0x7: 8 bits 0x8: 9 bits 0x9: 10 bits 0xA: 11 bits 0xB: 12 bits 0xC: 13 bits 0xD: 14 bits 0xE: 15 bits 0xF: 16 bits 0x10: 17 bits 0x11: 18 bits 0x12: 19 bits 0x13: 20 bits 0x14: 21 bits 0x15: 22 bits 0x16: 23 bits 0x17: 24 bits 0x18: 25 bits 0x19: 26 bits 0x1A: 27 bits 0x1B: 28 bits 0x1C: 29 bits 0x1D: 30 bits 0x1E: 31 bits 0x1F: 32 bits **Note:** This is an 8-bit or 16-bit interface only but can support up to a 32-bit format on the serial link. Extra bits on the serial link (more than 8 bits or 16 bits) are ignored in reception and filled with a predefined value in transmission.		

Table 13–44. BPCTR

Address Offset	0x060		
Physical Address	0x4809 0060	**Instance**	EAC1
Description	Bluetooth port control register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								MC_EN	RESERVED			PRE_MC			CKEN

Bits	Field Name	Description		Type	Reset
15:8	Reserved	Read returns 0.		R	0x00
7	MC_EN	Main channel enable		RW	0
		0x0:	Disabled		
		0x1:	Enabled		
6:4	Reserved	Read returns 0.		RW	0x0
3:1	PRE_MC	Prescale clock divisor for main channel Other values: reserved		RW	0x0
		0x0:	Clock divisor 1		
		0x1:	Clock divisor 2		
		0x2:	Clock divisor 4		
		0x3:	Clock divisor 8		
		0x4:	Clock divisor 16		
0	CKEN	AuSPI clock enable		RW	0
		0x0:	Clocks are shut off.		
		0x1:	Clocks are running.		

Table 13–45. BPMCCFR

Address Offset	0x064		
Physical Address	0x4809 0064	**Instance**	EAC1
Description	Bluetooth main channel port configuration register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	DFILL_MC	DJUST_MC	COMPAND		EXPAND		MCM	FSP_MC	FSBL_MC	CPB_MC	WS_MC				

Table 13−45. BPMCCFR (Continued)

Bits	Field Name	Description	Type	Reset
15	Reserved	Read returns 0.	R	0
14	DFILL_MC	Justified data filling value 0x0: Filled with zeros 0x1: Filled with ones	RW	0
13	DJUST_MC	Data justification 0x0: Data justify left (MSB alignment) 0x1: Data justify right (LSB alignment)	RW	0
12:11	COMPAND	Compand mode for main channel transmit register 0x0: Disabled 0x1: μ-law 0x2: A-law 0x3: Reserved	RW	0x0
10:9	EXPAND	Expand mode for main channel receive register 0x0: Disabled 0x1: μ-law 0x2: A-law 0x3: Reserved	RW	0x0
8	MCM	Main channel mode 0x0: Slave mode 0x1: Master mode	RW	0
7	FSP_MC	Frame synchronization polarity for main channel 0x0: Falling 0x1: Rising	RW	0
6	FSBL_MC	Frame synchronization level for main channel 0x0: Low level 0x1: High level	RW	0
5	CPB_MC	Clock polarity for main channel 0x0: Falling 0x1: Rising	RW	0
4:0	WS_MC	The word size bits are set to define the bit number of the transmission for the main channel. 0x0: 1 bit 0x1: 2 bits 0x2: 3 bits 0x3: 4 bits 0x4: 5 bits 0x5: 6 bits	RW	0x00

Table 13−45. BPMCCFR (Continued)

Bits	Field Name	Description	Type	Reset
4:0 (cont'd)	WS_MC (cont'd)	0x6: 7 bits 0x7: 8 bits 0x8: 9 bits 0x9: 10 bits 0xA: 11 bits 0xB: 12 bits 0xC: 13 bits 0xD: 14 bits 0xE: 15 bits 0xF: 16 bits 0x10: 17 bits 0x11: 18 bits 0x12: 19 bits 0x13: 20 bits 0x14: 21 bits 0x15: 22 bits 0x16: 23 bits 0x17: 24 bits 0x18: 25 bits 0x19: 26 bits 0x1A: 27 bits 0x1B: 28 bits 0x1C: 29 bits 0x1D: 30 bits 0x1E: 31 bits 0x1F: 32 bits **Note:** This is an 8-bit or 16-bit interface only but can support up to a 32-bit format on the serial link. Extra bits on the serial link (more than 8 bits or 16 bits) are ignored in reception and filled with a predefined value in transmission.		

Table 13−46. AMSCFR

Address Offset	0x080		
Physical Address	0x4809 0080	**Instance**	EAC1
Description	Audio mixer switch configuration register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				K12	K11	K10	K9	K8	K7	K6	K5	K4	K3	K2	K1

Bits	Field Name	Description	Type	Reset
15:12	Reserved	Read returns 0.	R	0x0
11	K12	K12 switch open/closed 0x0: K12 switch is open. 0x1: K12 switch is closed.	RW	0
10	K11	K11 switch open/closed 0x0: K11 switch is open. 0x1: K11 switch is closed.	RW	0
9	K10	K10 switch is open/closed 0x0: K10 switch is open. 0x1: K10 switch is closed.	RW	0
8	K9	K9 switch open/closed 0x0: K9 switch is open. 0x1: K9 switch is closed.	RW	0
7	K8	K8 switch open/closed 0x0: K8 switch is open. 0x1: K8 switch is closed.	RW	0
6	K7	K7 switch open/closed 0x0: K7 switch is open. 0x1: K7 switch is closed.	RW	0
5	K6	K6 switch open/closed 0x0: K6 switch is open. 0x1: K6 switch is closed.	RW	0
4	K5	K5 switch open/closed 0x0: K5 switch is open. 0x1: K5 switch is closed.	RW	0
3	K4	K4 switch open/closed 0x0: K4 switch is open. 0x1: K4 switch is closed.	RW	0

Table 13–46. AMSCFR (Continued)

Bits	Field Name	Description	Type	Reset
2	K3	K3 switch open/closed 0x0: K3 switch is open. 0x1: K3 switch is closed.	RW	0
1	K2	K2 switch open/closed 0x0: K2 switch is open. 0x1: K2 switch is closed.	RW	0
0	K1	K1 switch open/closed 0x0: K1 switch is open. 0x1: K1 switch is closed.	RW	0

Table 13–47. AMVCTR

Address Offset	0x084		
Physical Address	0x4809 0084	**Instance**	EAC1
Description	Audio master volume control register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GWO								GRO							

Bits	Field Name	Description	Type	Reset
15:8	GWO	Gain on write DMA operation	RW	0xE7
7:0	GRO	Gain on read DMA operation	RW	0xE7

Table 13–48. AM1VCTR

Address Offset	0x088		
Physical Address	0x4809 0088	**Instance**	EAC1
Description	Audio mixer 1 volume control register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUTE	GINB							Reserved	GINA						

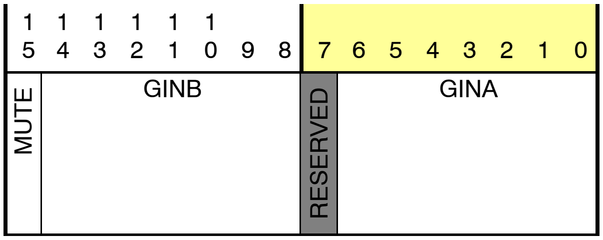
Bits	Field Name	Description	Type	Reset
15	MUTE	Mute/no mute on mixer output 0x0: No mute on mixer output 0x1: Mute on mixer output	RW	0
14:8	GINB	Gain on input B	RW	0x67

Table 13−48. AM1VCTR (Continued)

Bits	Field Name	Description	Type	Reset
7	Reserved	Read returns 0.	R	0
6:0	GINA	Gain on input A	RW	0x67

Table 13−49. AM2VCTR

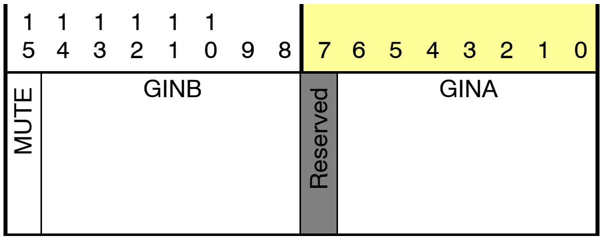
Address Offset	0x08C		
Physical Address	0x4809 008C	**Instance**	EAC1
Description	Audio mixer 2 volume control register		
Type	RW		



Bits	Field Name	Description	Type	Reset
15	MUTE	Mute/no mute on mixer output 0x0: No mute on mixer output 0x1: Mute on mixer output	RW	0
14:8	GINB	Gain on input B	RW	0x67
7	Reserved	Read returns 0.	R	0
6:0	GINA	Gain on input A	RW	0x67

Table 13−50. AM3VCTR

Address Offset	0x090		
Physical Address	0x4809 0090	**Instance**	EAC1
Description	Audio mixer 3 volume control register		
Type	RW		



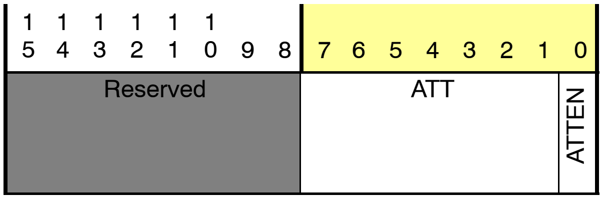
Bits	Field Name	Description	Type	Reset
15	MUTE	Mute/no mute on mixer output 0x0: No mute on mixer output 0x1: Mute on mixer output	RW	0
14:8	GINB	Gain on input B	RW	0x67

Table 13–50. AM3VCTR (Continued)

Bits	Field Name	Description	Type	Reset
7	Reserved	Read returns 0.	R	0
6:0	GINA	Gain on input A	RW	0x67

Table 13–51. ASTCTR

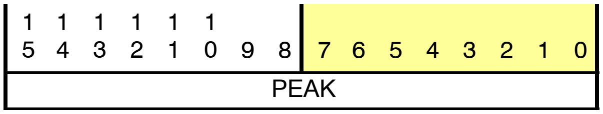
Address Offset	0x094		
Physical Address	0x4809 0094	**Instance**	EAC1
Description	Audio side tone control register		
Type	RW		



Bits	Field Name	Description		Type	Reset
15:8	Reserved	Modem interface clock source		R	0x00
7:1	ATT	Attenuation of the side tone		RW	0x67
0	ATTEN	Side tone enabled/disabled		RW	0
		0x0:	Side tone disabled		
		0x1:	Side tone enabled		

Table 13–52. APD1LCR

Address Offset	0x098		
Physical Address	0x4809 0098	**Instance**	EAC1
Description	Audio peak detector left channel register		
Type	R		



Bits	Field Name	Description	Type	Reset
15:0	PEAK	Peak value on the channel	R	0x0000

Table 13–53. APD1RCR

Address Offset	0x09C		
Physical Address	0x4809 009C	**Instance**	EAC1
Description	Audio peak detector 1 right channel register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PEAK															

Bits	Field Name	Description	Type	Reset
15:0	PEAK	Peak value on the channel	R	0x0000

Table 13–54. APD2LCR

Address Offset	0x0A0		
Physical Address	0x4809 00A0	**Instance**	EAC1
Description	Audio peak detector 2 left channel register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PEAK															

Bits	Field Name	Description	Type	Reset
15:0	PEAK	Peak value on the channel	R	0x0000

Table 13–55. APD2RCR

Address Offset	0x0A4		
Physical Address	0x4809 00A4	**Instance**	EAC1
Description	Audio peak detector 2 right channel register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PEAK															

Bits	Field Name	Description	Type	Reset
15:0	PEAK	Peak value on the channel	R	0x0000

Table 13–56. APD3LCR

Address Offset	0x0A8		
Physical Address	0x4809 00A8	**Instance**	EAC1
Description	Audio peak detector 3 left channel register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PEAK															

Bits	**Field Name**	**Description**	**Type**	**Reset**
15:0	PEAK	Peak value on the channel	R	0x0000

Table 13–57. APD3RCR

Address Offset	0x0AC		
Physical Address	0x4809 00AC	**Instance**	EAC1
Description	Audio peak detector 3 right channel register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PEAK															

Bits	**Field Name**	**Description**	**Type**	**Reset**
15:0	PEAK	Peak value on the channel	R	0x0000

Table 13–58. APD4R

Address Offset	0x0B0		
Physical Address	0x4809 00B0	**Instance**	EAC1
Description	Audio peak detector 4 register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PEAK															

Bits	**Field Name**	**Description**	**Type**	**Reset**
15:0	PEAK	Peak value on the channel	R	0x0000

Table 13−59. ADWR

Address Offset	0x0B4		
Physical Address	0x4809 00B4	**Instance**	EAC1
Description	Audio DMA write data register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WR_DATA															

Bits	Field Name	Description	Type	Reset
15:0	WR_DATA	DMA write operation data set by the hardware	R	0x0000

Table 13−60. ADRDR

Address Offset	0x0B8		
Physical Address	0x4809 00B8	**Instance**	EAC1
Description	Audio DMA read data register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA_RD															

Bits	Field Name	Description	Type	Reset
15:0	DATA_RD	DMA read operation data. Data in this register is read by the hardware.	R	0x0000

Table 13−61. AGCFR

Address Offset	0x0BC		
Physical Address	0x4809 00BC	**Instance**	EAC1
Description	Audio global configuration register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					MN_ST	B8_16	LI_BI	FSINT		FAUD_CKSRC		M_CKSRC	BT_CKSRC	MCLK_OUT	MCLK

Bits	Field Name	Description	Type	Reset
15:11	Reserved	Read returns 0.	R	0x00
10	MN_ST	Mono/stereo audio file 0x0: Mono 0x1: Stereo	RW	1

Table 13–61. AGCFR (Continued)

Bits	Field Name	Description	Type	Reset
9	B8_16	8-bit/16-bit audio file 0x0: 8 bits 0x1: 16 bits	RW	1
8	LI_BI	Audio file endianism 0x0: Little-endian 0x1: Big-endian	RW	0
7:6	FSINT	Intermediate sample frequency for DMA read and write operations 0x0: FSINT is 8 kHz. 0x1: FSINT is 11.025 kHz. 0x2: FSINT is 22.05 kHz. 0x3: FSINT is 44.1 kHz.	RW	0x1
5:4	AUD_CKSRC	Audio processing clock source 0x0: Codec master clock 0x1: Not used in OMAP2420 0x2: Not used in OMAP2420 0x3: Reserved	RW	0x0
3	M_CKSRC	Modem interface clock source 0x0: Not used in OMAP2420 0x1: Not used in OMAP2420	RW	1
2	BT_CKSRC	Bluetooth interface clock source 0x0: 12-MHz clock on CLK_12M input 0x1: 13-MHz clock on CLK_13M input	RW	0
1	MCLK_OUT	MCLK_OUT frequency 0x0: MCLK_OUT = MCLK 0x1: MCLK_OUT = MCLK/2	RW	0
0	MCLK	Clock on MCLK input 0x0: Clock on MCLK input is 11.2896/12.288 MHz 0x1: Clock on MCLK input is 22.5792/24.576 MHz (internally used audio clock is MCLK/2)	RW	1

Table 13−62. AGCTR

Address Offset	0x0C0		
Physical Address	0x4809 00C0	**Instance**	EAC1
Description	Audio global control register 2		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AUDRD	AUDRDI	AUDRDIEN	DMAREN	DMAWEN	Reserved							MCLK_EN	OSCMCLK_EN	AUDEN	EACPWD

Bits	Field Name	Description	Type	Reset
15	AUDRD	Audio ready 0x0: Audio not ready (either disabled by AUEN or disabled parameters setting changes) 0x1: Audio is ready for processing.	R	–
14	AUDRDI	Audio ready interrupt status 0x0: No interrupt generated 0x1: Interrupt generated	RW	0
13	AUDRDIEN	Audio ready interrupt enable 0x0: Interrupt generation disabled 0x1: Interrupt generated on audio ready condition	RW	0
12	DMAREN	Audio files play operation 0x0: DMA read operation disabled 0x1: DMA read operation enabled	RW	0
11	DMAWEN	Audio file record operation 0x0: DMA write operation disabled 0x1: DMA write operation enabled	RW	0
10:4	Reserved	Reads return 0s.	R	0x00
3	MCLK_EN	Internal MCLK enable 0x0: MCLK gated internally 0x1: MCLK enabled	RW	0
2	OSCMCLK_EN	OSCMCLK_EN output for MCLK oscillator control 0x0: OSCMCLK_EN low (oscillator disabled) 0x1: OSCMCLK_EN high (oscillator enabled)	RW	0
1	AUDEN	Audio processing enable/disable 0x0: Audio processing disabled 0x1: Audio processing enable	RW	0
0	EACPWD	EAC operation 0x0: Normal EAC operation 0x1: EAC in power-down mode The functional clocks are gated.	RW	0

Table 13−63. AGCFR2

Address Offset	0x0C4		
Physical Address	0x4809 00C4	**Instance**	EAC1
Description	Audio global configuration register, rev 2		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										BT_MD_WIDEBAND	MCLK_I2S_N11M_12M	I2S_N44K_48K	FSINT2		

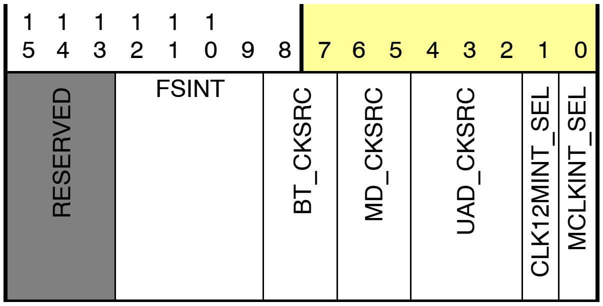
Bits	Field Name	Description		Type	Reset
15:6	Reserved	Read returns 0.		R	0x000
5	BT_MD_ WIDEBAND	Bluetooth device and modem AuSPIs wide-band mode		RW	0
		0x0:	Wide-band mode disabled (reset value for EAC rev 1 backward compatibility; only 8-kHz frame freq support)		
		0x1:	Wide-band mode enabled (both BT and MD AuSPIs operate on 16-kHz frame freq)		
4	MCLK_I2S_ N11M_12M	MCLK freq indicator for audio operations		RW	0
		0x0:	Clock on MCLK is 11.289 MHz (reset value for EAC EAC rev 1 backward compatibility; I2S requires 11.289 MHz).		
		0x1:	Clock on MCLK is 12.288MHz for I2S 48M master mode compatibility.		
3	I2S_N44K_48K	Frame sample frequency of I2S codec port. This bit field is used only to inform internal logic and does not generate this frequency value.		RW	0
		0x0:	Codec freq = 44.1 kHz (reset value for EAC rev 1 backward compatibility: 44.1-kHz support only).		
		0x1:	Codec freq = 48 kHz		
2:0	FSINT2	Intermediate sample frequency for DMA read and write operations		RW	0x7
		0x0:	FSINT is 8 kHz.		
		0x1:	FSINT is 11.025 kHz.		
		0x2:	FSINT is 22.05 kHz.		
		0x3:	FSINT is 44.1 kHz.		
		0x4:	FSINT is 48 kHz (EAC rev 2 new).		

Table 13−63. AGCFR2 (Continued)

Bits	Field Name	Description		Type	Reset
2:0	FSINT2	0x5:	Reserved		
(cont'd)	(cont'd)	0x6:	Reserved		
		0x7:	AGCFR/FSINT are used (reset value EAC rev1 backward compatibility).		

Table 13−64. AGCFR3

Address Offset	0x0C8		
Physical Address	0x4809 00C8	**Instance**	EAC1
Description	Audio global configuration register, rev 3		
Type	RW		



Bits	Field Name	Description		Type	Reset
15	CP_TR_DMA	Codec port transparent DMA (to audio DMAs)		RW	0x0
		0:	Transparent DMA is disabled.		
		1:	Transparent DMA is enabled.		
14	BT_TR_DMA	Bluetooth transparent DMA (to Bluetooth uplink write and downlink read DMAs)		RW	0x0
		0:	Transparent DMA is disabled.		
		1:	Transparent DMA is enabled.		
13	MD_TR_DMA	Modem transparent DMA (to modem uplink write and downlink read DMAs)		RW	0x0
		0:	Transparent DMA is disabled.		
		1:	Transparent DMA is enabled.		
12:9	FSINT	FSINT. Other values: reserved		RW	0xF
		0x0:	8 kHz		
		0x1:	11.025 kHz		
		0x2:	16 kHz		
		0x3:	22.05 kHz		
		0x4:	24 kHz		
		0x5:	32 kHz		
		0x6:	44.1 kHz		
		0x7:	48 kHz		
		0xF:	FSINT setting by AGCFR/AGCRF2		

Table 13−64. AGCFR3 (Continued)

Bits	Field Name	Description	Type	Reset
8:7	BT_CKSRC	Bluetooth port clock selection 0x0: Not used in OMAP2420 0x1: CLK12M_INT clock input 0x2: Not used in OMAP2420 0x3: Bluetooth port clock source selected by AGCFR	RW	0x3
6:5	MD_CKSRC	Modem port clock source 0x0: Not used in OMAP2420 0x1: CLK12M_INT clock 0x2: Not used in OMAP2420 0x3: Selected by AGCFR	RW	0x3
4:2	AUD_CKSRC	Audio and codec port clock source. Other values: reserved 0x0: MCLK_INT clock 0x1: Not used in OMAP2420 0x2: CLK12M_INT clock 0x3: Not used in OMAP2420 0x7: Selected by AGCFR	RW	0x7
1	CLK12MINT_SEL	CLK12M_INT (internal 12-MHz clock) source 0x0: Not used in OMAP2420 0x1: CLK96M clock input divided by 8	RW	0
0	MCLKINT_SEL	MCLK_INT (internal codec master clock) source 0x0: CLKMC input pin (OSC or square clock) 0x1: CLK96M (96 MHz) divided by 8.5 (11.28 MHz)	RW	0

Table 13−65. MBPDMACTR

Address Offset	0x0CC		
Physical Address	0x4809 00CC	**Instance**	EAC1
Description	Modem and Bluetooth ports DMA channel control register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				BPDDMA		BPUDMA		MPDDMA		MPUDMA		BPDATASCR		MPDATASCR	

Table 13−65. MBPDMACTR (Continued)

Bits	Field Name	Description	Type	Reset
15:12	Reserved	Read returns 0.	R	0x0
11:10	BPDDMA	Bluetooth port downlink/uplink DMA write channel enable 0x0: Bluetooth port downlink DMA write channel disabled and Bluetooth port downlink DMA read channel disabled 0x1: Bluetooth port downlink DMA write channel disabled and Bluetooth port downlink DMA read channel enabled 0x2: Bluetooth port downlink DMA write channel enabled and Bluetooth port downlink DMA read channel disabled 0x3: Bluetooth port downlink DMA write channel enabled and Bluetooth port downlink DMA read channel enabled	RW	0x0
9:8	BPUDMA	Bluetooth port uplink/downlink DMA read channel enable 0x0: Bluetooth port uplink DMA write channel disabled and Bluetooth port uplink DMA read channel disabled 0x1: Bluetooth port uplink DMA write channel disabled and Bluetooth port uplink DMA read channel enabled 0x2: Bluetooth port uplink DMA write channel enabled and Bluetooth port uplink DMA read channel disabled 0x3: Bluetooth port uplink DMA write channel enabled and Bluetooth port uplink DMA read channel enabled	RW	0x0
7:6	MPDDMA	Modem port downlink/uplink DMA read channel enable 0x0: Modem port downlink DMA write channel disabled and modem port downlink DMA read channel disabled 0x1: Modem port downlink DMA write channel disabled and modem port downlink DMA read channel enabled 0x2: Modem port downlink DMA write channel enabled and modem port downlink DMA read channel disabled 0x3: Modem port downlink DMA write channel enabled and modem port downlink DMA read channel enabled	RW	0x0

Table 13−65. MBPDMACTR (Continued)

Bits	Field Name	Description	Type	Reset
5:4	MPUDMA	Modem port uplink/downlink DMA write channel enable 0x0: Modem port uplink DMA write channel disabled and modem port uplink DMA read channel disabled 0x1: Modem port uplink DMA write channel disabled and modem port uplink DMA read channel enabled 0x2: Modem port uplink DMA write channel enabled and modem port uplink DMA read channel disabled 0x3: Modem port uplink DMA write channel enabled and modem port uplink DMA read channel enabled	RW	0x0
3:2	BPDATASCR	Bluetooth port downlink/uplink data source selection 0x0: Downlink data flow from Bluetooth AuSPI RX and uplink data flow from audio part 0x1: Downlink data flow from Bluetooth AuSPI RX and uplink data flow from Bluetooth port uplink DMA write channel 0x2: Downlink data flow from Bluetooth port downlink DMA write channel and uplink data flow from audio part 0x3: Downlink data flow from Bluetooth port downlink DMA write channel and uplink data flow from Bluetooth port uplink DMA write channel	RW	0x0
1:0	MPDATASCR	Modem port downlink/uplink data source selection 0x0: Downlink data flow from modem AuSPI RX and uplink data flow from audio 0x1: Downlink data flow from modem AuSPI RX and uplink data flow from modem port uplink DMA write channel 0x2: Downlink data flow from modem port downlink DMA write channel and uplink data flow from audio 0x3: Downlink data flow from modem port downlink DMA write channel and uplink data flow from modem port uplink DMA write channel	RW	0x0

Table 13−66. MPDDMARR

Address Offset	0x0D0		
Physical Address	0x4809 00D0	**Instance**	EAC1
Description	Modem port downlink DMA read register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA															

Bits	Field Name	Description	Type	Reset
15:0	DATA	Modem port downlink DMA read register	R	0x0000

Table 13−67. MPDDMAWR

Address Offset	0x0D4		
Physical Address	0x4809 00D4	**Instance**	EAC1
Description	Modem port downlink DMA write register		
Type	W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA															

Bits	Field Name	Description	Type	Reset
15:0	DATA	Modem port downlink DMA write register	W	0x0000

Table 13−68. MPUDMARR

Address Offset	0x0D8		
Physical Address	0x4809 00D8	**Instance**	EAC1
Description	Modem port uplink DMA read register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA															

Bits	Field Name	Description	Type	Reset
15:0	DATA	Modem port uplink DMA read data	R	0x0000

Table 13−69. MPUDMAWR

Address Offset	0x0E0		
Physical Address	0x4809 00E0	**Instance**	EAC1
Description	Modem port uplink DMA write register		
Type	W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA															

Bits	**Field Name**	**Description**	**Type**	**Reset**
15:0	DATA	Modem port uplink DMA write data	W	0x0000

Table 13−70. BPDDMARR

Address Offset	0x0E4		
Physical Address	0x4809 00E4	**Instance**	EAC1
Description	Bluetooth port downlink DMA read register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA															

Bits	**Field Name**	**Description**	**Type**	**Reset**
15:0	DATA	Bluetooth port downlink DMA read register	R	0x0000

Table 13−71. BPDDMAWR

Address Offset	0x0E8		
Physical Address	0x4809 00E8	**Instance**	EAC1
Description	Bluetooth port downlink DMA write register		
Type	W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA															

Bits	**Field Name**	**Description**	**Type**	**Reset**
15:0	DATA	Bluetooth port downlink DMA write data	W	0x0000

Table 13−72. BPUDMARR

Address Offset	0x0EC		
Physical Address	0x4809 00EC	**Instance**	EAC1
Description	Bluetooth uplink DMA read register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA															

Bits	Field Name	Description	Type	Reset
15:0	DATA	Bluetooth port uplink DMA read data	R	0x0000

Table 13−73. BPUDMAWR

Address Offset	0x0F0		
Physical Address	0x4809 00F0	**Instance**	EAC1
Description	Bluetooth uplink DMA write register		
Type	W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA															

Bits	Field Name	Description	Type	Reset
15:0	DATA	Bluetooth port uplink DMA write data	W	0x0000

Table 13−74. VERSION_NUMBER

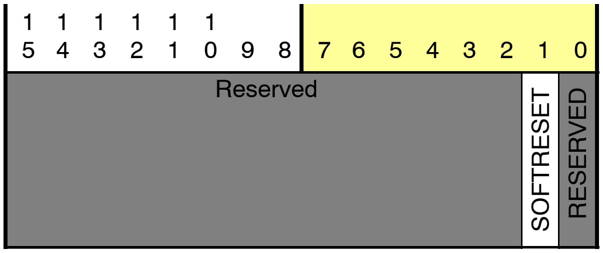
Address Offset	0x100		
Physical Address	0x4809 0100	**Instance**	EAC1
Description	IP revision code		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								REV							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0.	R	0x00
7:0	REV	Contains IP revision code	R	

Table 13−75. SYSCONFIG

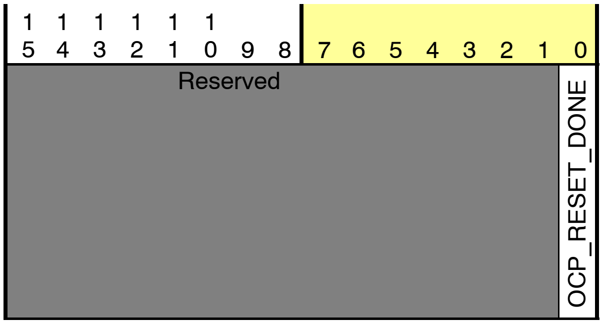
Address Offset	0x104		
Physical Address	0x4809 0104	**Instance**	EAC1
Description	Enables control of some OCP parameters		
Type	RW		



Bits	Field Name	Description	Type	Reset
15:2	Reserved	Read returns 0.	R	0x0000
1	SOFTRESET	Software reset. Set this bit to 1 to trigger a module reset. The bit is automatically reset by the hardware. Read 0x1: Cannot happen Write 0x1: The module is reset.	RW	0
0	Reserved	Read returns 0. Read 0x0: Read returns 0. Write 0x0: Normal mode	R	0

Table 13−76. SYSSTATUS

Address Offset	0x108		
Physical Address	0x4809 0108	**Instance**	EAC1
Description	Provides status information about the module		
Type	R		



Bits	Field Name	Description	Type	Reset
15:4	Reserved	Read returns 0.	R	0x000
3:1	Reserved	Reserved	R	0x0
0	OCP_RESET_DONE	1 if OCP-related parts also left reset state	R	0

Chapter 14

Camera Subsystem

This chapter describes the camera subsystem, which provides the system interface and functionality for connecting image-sensor modules to the OMAP2420 device.

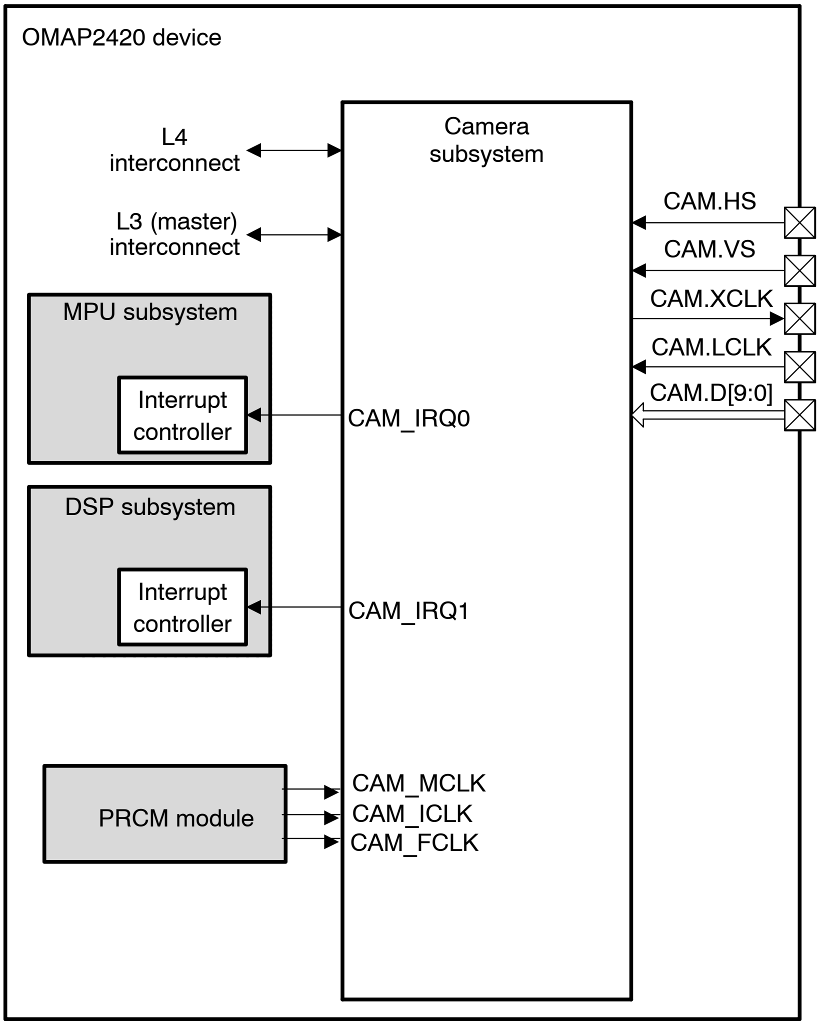
Topic **Page**

14.1 Camera Subsystem Overview

The camera subsystem provides the system interface and functionality for connecting image-sensor modules to the OMAP2420 device.

Figure 14−1 shows the camera subsystem.

Figure 14−1. Camera Subsystem Block Diagram



The main features of the camera subsystem are:

- ❑ Serial interface: Compact camera port (CCP) interface at 208 MHz
- ❑ Parallel interface: 8 or 10 bits (maximum clock of 96 MHz for 8-bit data and 48 MHz for 10-bit data)
- ❑ Support of multiple input formats, including Bayer, RGB, and ITU-R BT.656
- ❑ Embedded direct memory access (DMA) controller with its own memory management unit (MMU)

- One master interface connected to level 3 (L3) and one slave interface connected to level 4 (L4)

The camera subsystem consists of the camera core, DMA, and MMU.

The camera core ensures image-sensor data format detection and transmission to the system. The DMA splits pictures to the appropriate virtual address of the memory space. The MMU translates virtual address space into physical address space.

14.1.1 Camera Core Features

- Serial interface: CCP interface. Maximum clock of 208 MHz. Complies with MIPI CSI-1.
- Parallel interface: 8 or 10 bits. Maximum clock of 96 MHz for 8-bit data and 48 MHz for 10-bit data. Supports two operating modes:
  - BT656: ITU-R BT656-compatible parallel interface
  - No BT: General parallel interface with vertical and horizontal synchronization signals

14.1.2 DMA Features

- Two hardware DMA requests
- Four logical channels
- 64-bit interface width on read and write ports
- Streaming burst at the source
- Synchronized transfer on the source
- Buffering disable for synchronized source transfer
- Nonsynchronized transfer at the destination
- Constant addressing on the read port
- Packing/unpacking
- Channel chaining
- Streaming burst at destination

14.1.3 MMU Features

- Up to 4G bytes of virtual memory

14.2 Camera Subsystem Environment

The camera subsystem offers three configurations:

- ☐ Generic parallel interface
- ☐ ITU-R BT.656 parallel interface
- ☐ CCP serial interface

Figure 14−2 through Figure 14−4 show these configurations.

Figure 14−2. Camera Subsystem Parallel Interface in Generic Configuration

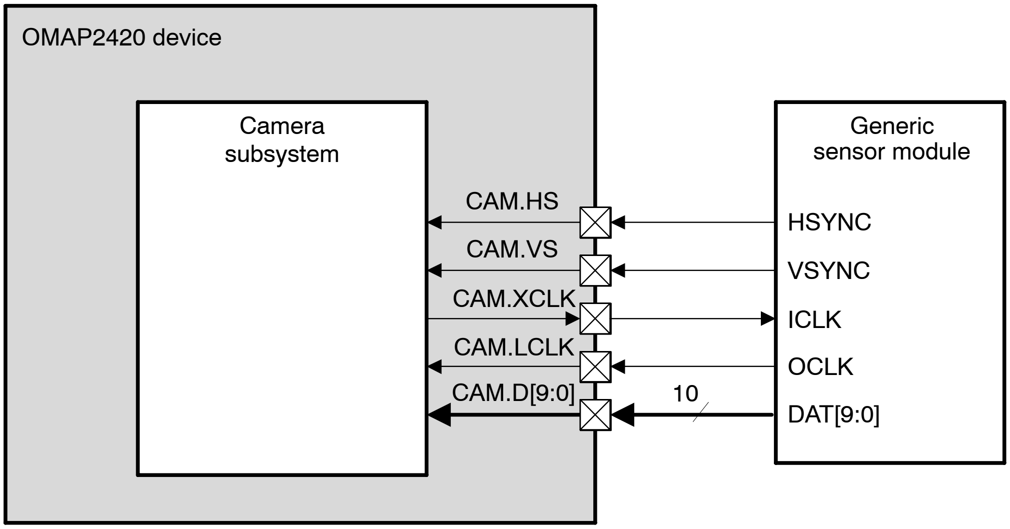
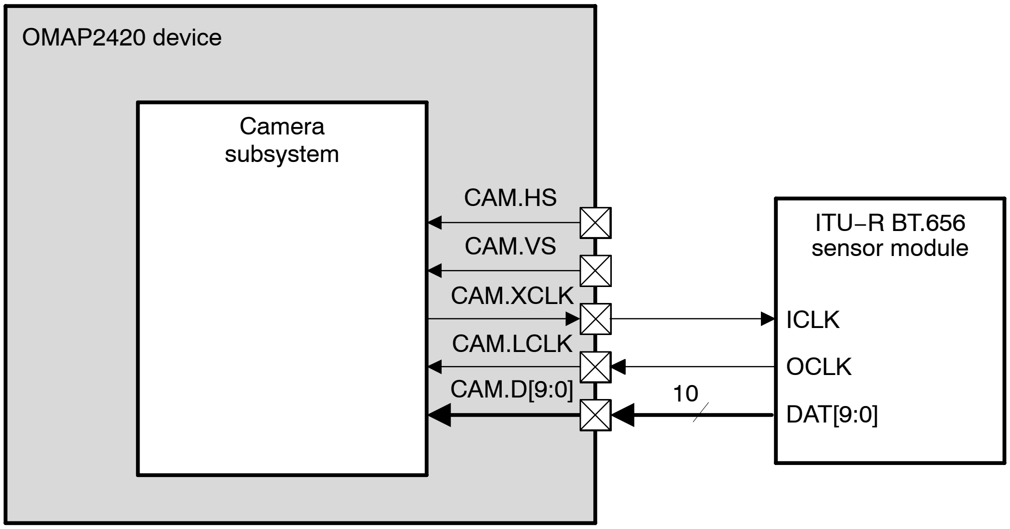


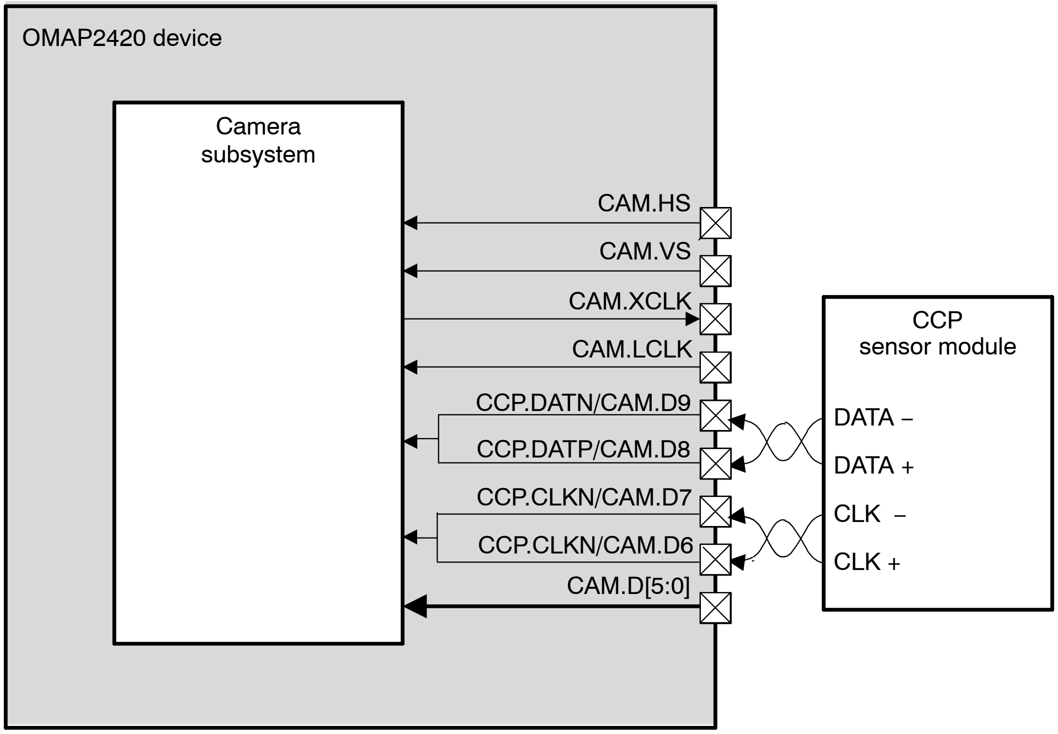
Figure 14−3. Camera Subsystem Parallel Interface in ITU-R BT.656 Configuration



CAUTION

For Figure 14−2 and Figure 14−3, if an 8-bit sensor is used, CAM.D[9:2], not CAM.D[9:0], must be used for the connection.

Figure 14−4. CCP Serial Interface Configuration

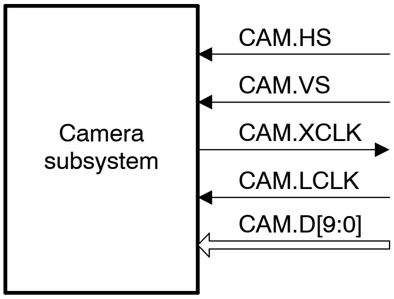


14.2.1 Parallel Generic Configuration Functional Interface

14.2.1.1 Basic Camera Pins for Parallel Generic Configuration

Figure 14−5 shows all possible interfaces of the camera subsystem in the parallel generic configuration.

Figure 14−5. Camera Generic Configuration Interface Signals



CAUTION

For Figure 14−5, if an 8-bit sensor is used, CAM.D[9:2], not CAM.D[9:0], must be used for the connection.

14.2.1.2 Parallel Generic Configuration Interface Description

Table 14−1 describes the generic configuration interface for the camera subsystem

Table 14−1. Camera Generic Configuration I/O Description

Signal Name	I/O[1]	Description	Reset
CAM.HS	I	Line trigger input signal	N/A
CAM.VS	I	Frame trigger input signal	N/A
CAM.XCLK	O	External clock for the image-sensor module	0
CAM.LCLK	I	Latch clock for parallel input data	N/A
CAM.D[9:0][2]	I	Input data bits 0 to 9[2]	N/A

Notes: 1) I = Input, O = Output

2) Applies to a 10-bit interface. For an 8-bit interface, the signal name is CAM.D[9:2].

14.2.1.3 Parallel Generic Configuration Protocol and Data Format

Because this configuration uses all BT.656 signals, plus the CAM.HS and CAM.VS signals, to recognize valid data, it is also called parallel No BT.656.

Pixel data is presented on CAM.D, where one pixel is sampled for every CAM.LCLK rising edge (or falling edge, depending on the configuration of CAM.LCLK polarity—for more information, see Section 14.4, *Camera Subsystem Functional Description*).

Active pixels are identified by a combination of two additional timing signals: CAM.HS (horizontal synchronization) and CAM.VS (vertical synchronization). During the image-sensor readout, these signals define when a row of valid data begins and ends and when a frame starts and ends.

Note:

The clock CAM.LCLK runs during blanking periods (when CAM.HS and CAM.VS are inactive).

Figure 14−6 and Figure 14−7 show the frame and data timing according to synchronization signals in the parallel No BT configuration.

Figure 14−6. Synchronization Signals and Frame Timing in No BT Protocol

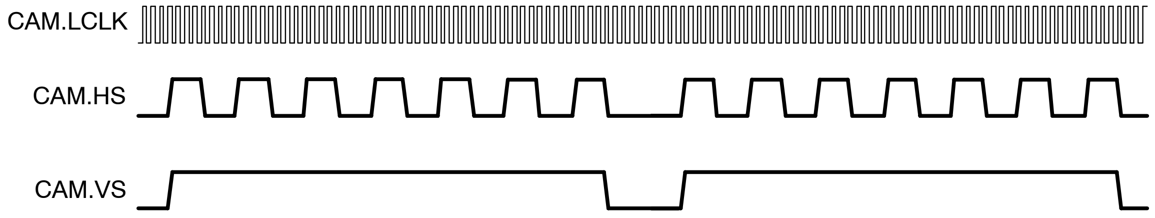
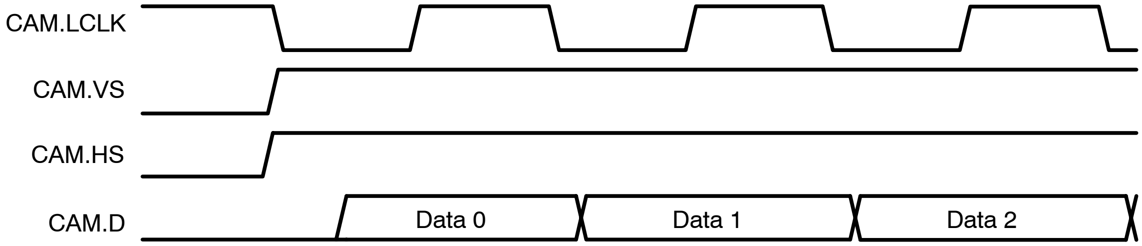


Figure 14−7. Synchronization Signals and Data Timing in No BT Protocol

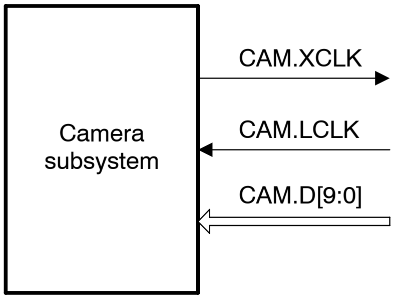


14.2.2 ITU-R BT.656 Configuration Functional Interface

14.2.2.1 Basic Camera Pins for ITU-R BT.656 Configuration

Figure 14−8 shows all possible interfaces of the camera subsystem in the ITU-R BT.656 parallel configuration.

Figure 14−8. Camera ITU-R BT.656 Configuration Interface Signals



14.2.2.2 ITU-R BT.656 Configuration Interface Description

Table 14−2 lists the input/output (I/O) interface signals for the ITU-R BT.656 configuration.

Table 14−2.I/O Description

Signal Name	I/O[1]	Description	Reset
CAM.XCLK	O	External clock for the image sensor module	0
CAM.D[9:0]	I	Input data bits 0 to 9[2]	Unknown
CAM.LCLK[2]	I	Latch clock for the parallel input data	Unknown

Notes: 1) I = Input, O = Output

2) Applies to a 10-bit interface. For an 8-bit interface, the signal name is CAM.D[9:2].

14.2.2.3 ITU-R BT.656 Protocol and Data Format

The camera interface supports data in ITU-R BT.656 format.

The ITU-R BT.656 standard specifies a method to transfer YUV422 data over an 8- or 10-bit video interface.

In BT.656, the data words in which the eight most-significant bits (MSBs) are all set to 1 or all set to 0 are reserved. Only 254 of the possible 256 8-bit word values and 1016 of the possible 1024 10-bit word values are used to represent signal values.

Data is multiplexed in the following order: Cb_0 Y_0 Cr_0 Y_1 Cb_2 Y_2 Cr_2 Y_3, etc., where the byte sequence Cb_{2n} Y_{2n} Cr_{2n} refers to cosited luminance and chroma samples and the following byte Y_{2n+1} corresponds to the next luminance sample.

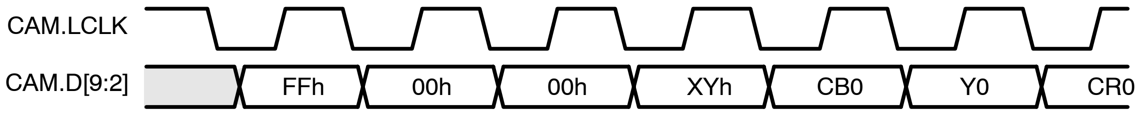
The BT.656 protocol uses a unique timing reference signal embedded in the video stream. The synchronization signals CAM.HS and CAM.VS are not needed. This reduces the number of wires required for a BT.656 video interface.

There are two timing reference codes. The start of active video (SAV) reference code precedes each video data block and the end of active video (EAV) follows each video block. Each timing reference signal consists of a 4-byte sequence in the following hexadecimal format: FF 00 00 XY. The first 3 bytes are a fixed preamble. (See the ITU-R BT.656 specification.)

The fourth byte contains information defining field identification (F), blanking (V) and SAV/EAV information (H), and four parity bits that are calculated as a function of F, V, and H.

Figure 14−12 shows data timing with SAV and EAV synchronization signals.

Figure 14−12. Data Timing in ITU-R BT, 8-Bit Case

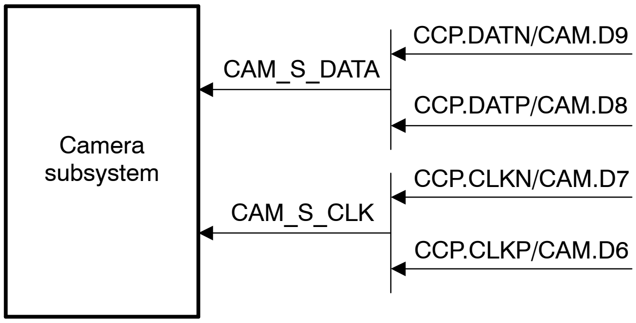


14.2.3 CCP Serial Configuration Functional Interface

14.2.3.1 Basic Camera Pins for CCP Serial Configuration

Figure 14−13 shows all possible interfaces of the camera subsystem in the CCP serial configuration.

Figure 14−13. Camera CCP Configuration Interface Signals



14.2.3.2 CCP Serial Configuration Interface Description

Table 14−3 lists the I/O interface signal names for the CCP serial configuration.

Table 14−3. I/O Description

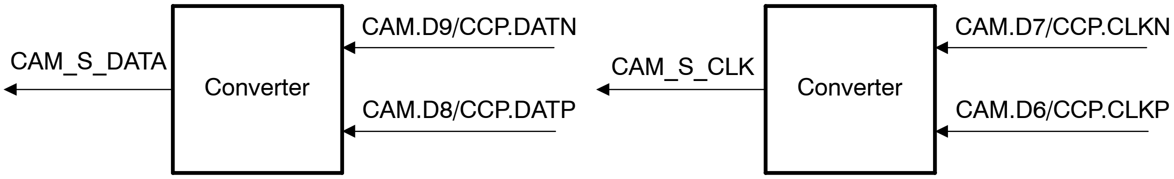
Signal Name	I/O[1]	Description	Reset
CAM_S_DATA	I	Serial data input	N/A
CAM_S_CLK	I	Serial clock input	N/A

Notes: 1) I = Input, O = Output

From the OMAP2420 side, the CCP interface comprises four differential inputs representing two effective signals: the serial data and the clock.

From the camera subsystem side, the CCP interface comprises two serial inputs (CAM_S_DATA and CAM_S_CLK) that are the result of low-voltage differential signal (LVDS) converters, as shown in Figure 14−14.

Figure 14−14. CCP Differential Signal Conversion



14.2.3.3 CCP Protocol and Data Format

The CCP is a serial interface to an image sensor. Signals from the camera are serial input data (CAM_S_DATA) and clock (CAM_S_CLK).

Reception of data from the image-sensor module uses four synchronization codes embedded in the serial bit stream:

- ☐ Frame start code (FSC): Identifies the start of a new frame.
- ☐ Line start code (LSC): Identifies the start of a new line.
- ☐ Line end code (LEC): Identifies the end of a line. It is received for every line except the one, which ends with an FEC.
- ☐ Frame end code (FEC): Identifies the end of the last line and the end of the current frame.

The CCP interface has several image data operating modes, which are summarized in Table 14−4.

Table 14−4. CCP Image Data Operating Modes

Description	Mode
YUV4:2:2 image data	YUV422
YUV4:2:0 image data	YUV420
RGB888 image data	RGB888
RGB565 image data	RGB565

Table 14−4. CCP Image Data Operating Modes (Continued)

Description	Mode
RGB444 image data	RGB444
Raw Bayer 8-bit image data	RAW8
Raw Bayer 10-bit image data	RAW10
Raw Bayer 12-bit image data	RAW12
JPEG 8-bit data	JPEG8
JPEG 8-bit data + FSP	JPEG8 FSP

Figure 14−15 through Figure 14−24 show the CCP image data operating modes.

Figure 14−15. YUV422 Big-Endian

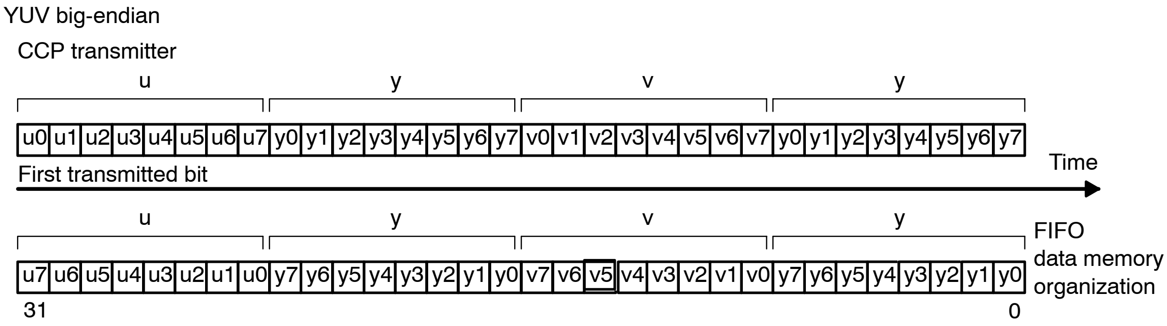

YUV big-endian
CCP transmitter
u
y
v
y
u0 u1 u2 u3 u4 u5 u6 u7 y0 y1 y2 y3 y4 y5 y6 y7 v0 v1 v2 v3 v4 v5 v6 v7 y0 y1 y2 y3 y4 y5 y6 y7
First transmitted bit
Time
u
y
v
y
u7 u6 u5 u4 u3 u2 u1 u0 y7 y6 y5 y4 y3 y2 y1 y0 v7 v6 v5 v4 v3 v2 v1 v0 y7 y6 y5 y4 y3 y2 y1 y0
FIFO data memory organization
31
0

Figure 14−16. YUV422

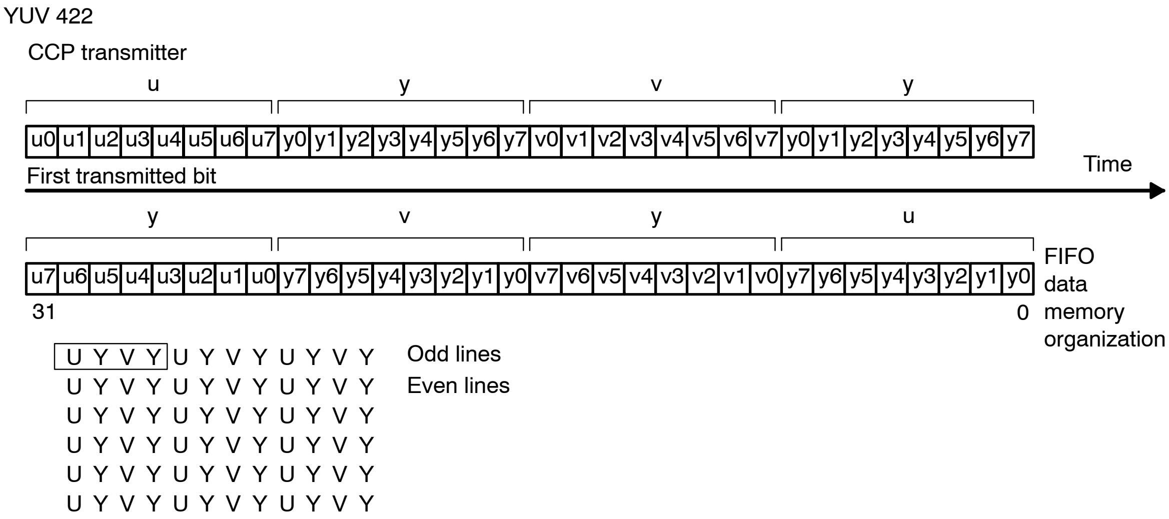

YUV 422
CCP transmitter
u
y
v
y
u0 u1 u2 u3 u4 u5 u6 u7 y0 y1 y2 y3 y4 y5 y6 y7 v0 v1 v2 v3 v4 v5 v6 v7 y0 y1 y2 y3 y4 y5 y6 y7
First transmitted bit
Time
y
v
y
u
u7 u6 u5 u4 u3 u2 u1 u0 y7 y6 y5 y4 y3 y2 y1 y0 v7 v6 v5 v4 v3 v2 v1 v0 y7 y6 y5 y4 y3 y2 y1 y0
FIFO data memory organization
31
0
U Y V Y U Y V Y U Y V Y
U Y V Y U Y V Y U Y V Y
U Y V Y U Y V Y U Y V Y
U Y V Y U Y V Y U Y V Y
U Y V Y U Y V Y U Y V Y
U Y V Y U Y V Y U Y V Y
Odd lines
Even lines

Figure 14−17. YUV420

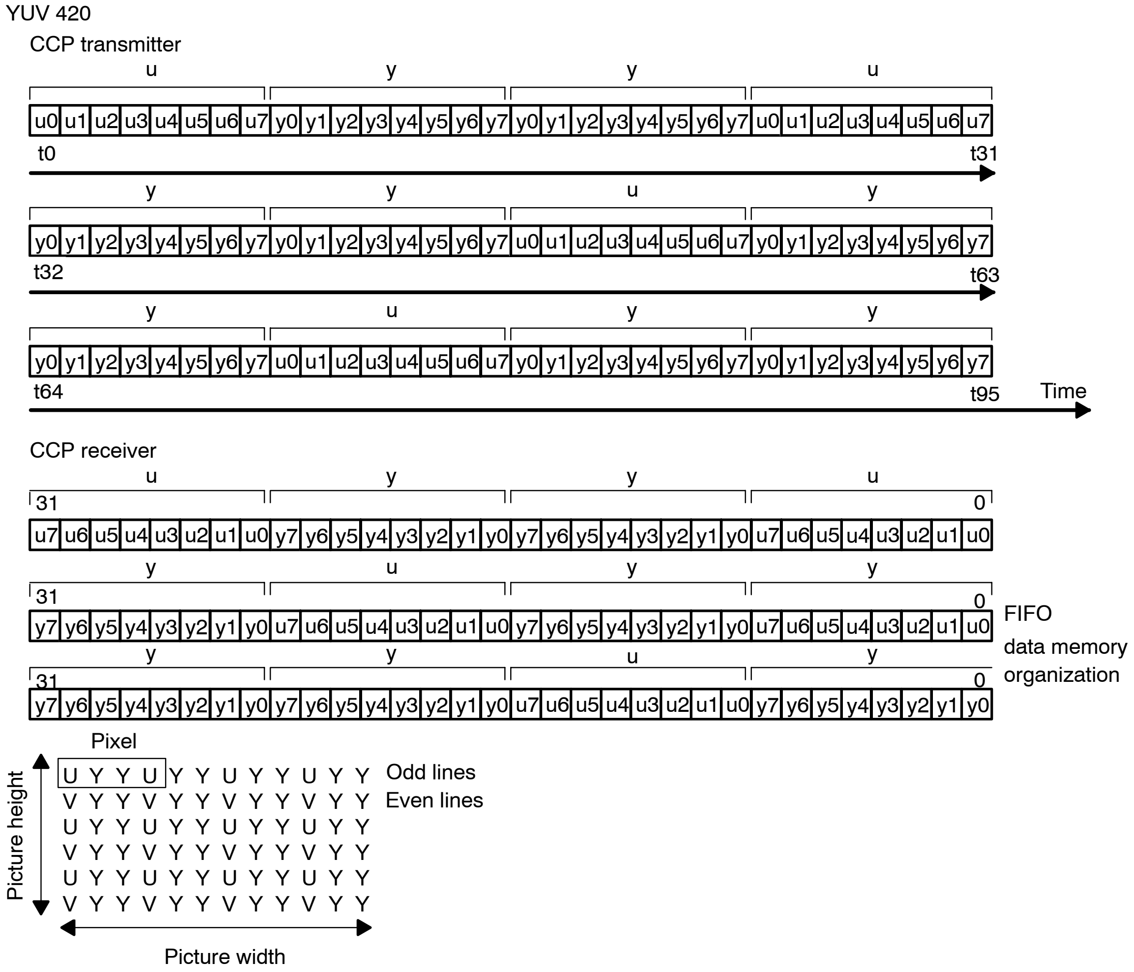

YUV 420
CCP transmitter
u y y u
u0 u1 u2 u3 u4 u5 u6 u7 y0 y1 y2 y3 y4 y5 y6 y7 y0 y1 y2 y3 y4 y5 y6 y7 u0 u1 u2 u3 u4 u5 u6 u7
t0
t31
y y u y
y0 y1 y2 y3 y4 y5 y6 y7 y0 y1 y2 y3 y4 y5 y6 y7 u0 u1 u2 u3 u4 u5 u6 u7 y0 y1 y2 y3 y4 y5 y6 y7
t32
t63
y u y y
y0 y1 y2 y3 y4 y5 y6 y7 u0 u1 u2 u3 u4 u5 u6 u7 y0 y1 y2 y3 y4 y5 y6 y7 y0 y1 y2 y3 y4 y5 y6 y7
t64
t95
Time
CCP receiver
u y y u
31
0
u7 u6 u5 u4 u3 u2 u1 u0 y7 y6 y5 y4 y3 y2 y1 y0 y7 y6 y5 y4 y3 y2 y1 y0 u7 u6 u5 u4 u3 u2 u1 u0
y u y y
31
0
y7 y6 y5 y4 y3 y2 y1 y0 u7 u6 u5 u4 u3 u2 u1 u0 y7 y6 y5 y4 y3 y2 y1 y0 u7 u6 u5 u4 u3 u2 u1 u0
y y u y
31
0
y7 y6 y5 y4 y3 y2 y1 y0 y7 y6 y5 y4 y3 y2 y1 y0 u7 u6 u5 u4 u3 u2 u1 u0 y7 y6 y5 y4 y3 y2 y1 y0
FIFO
data memory
organization
Pixel
U Y Y U Y Y U Y Y U Y Y
V Y Y V Y Y V Y Y V Y Y
U Y Y U Y Y U Y Y U Y Y
V Y Y V Y Y V Y Y V Y Y
U Y Y U Y Y U Y Y U Y Y
V Y Y V Y Y V Y Y V Y Y
Odd lines
Even lines
Picture height
Picture width

Figure 14−18. RGB888

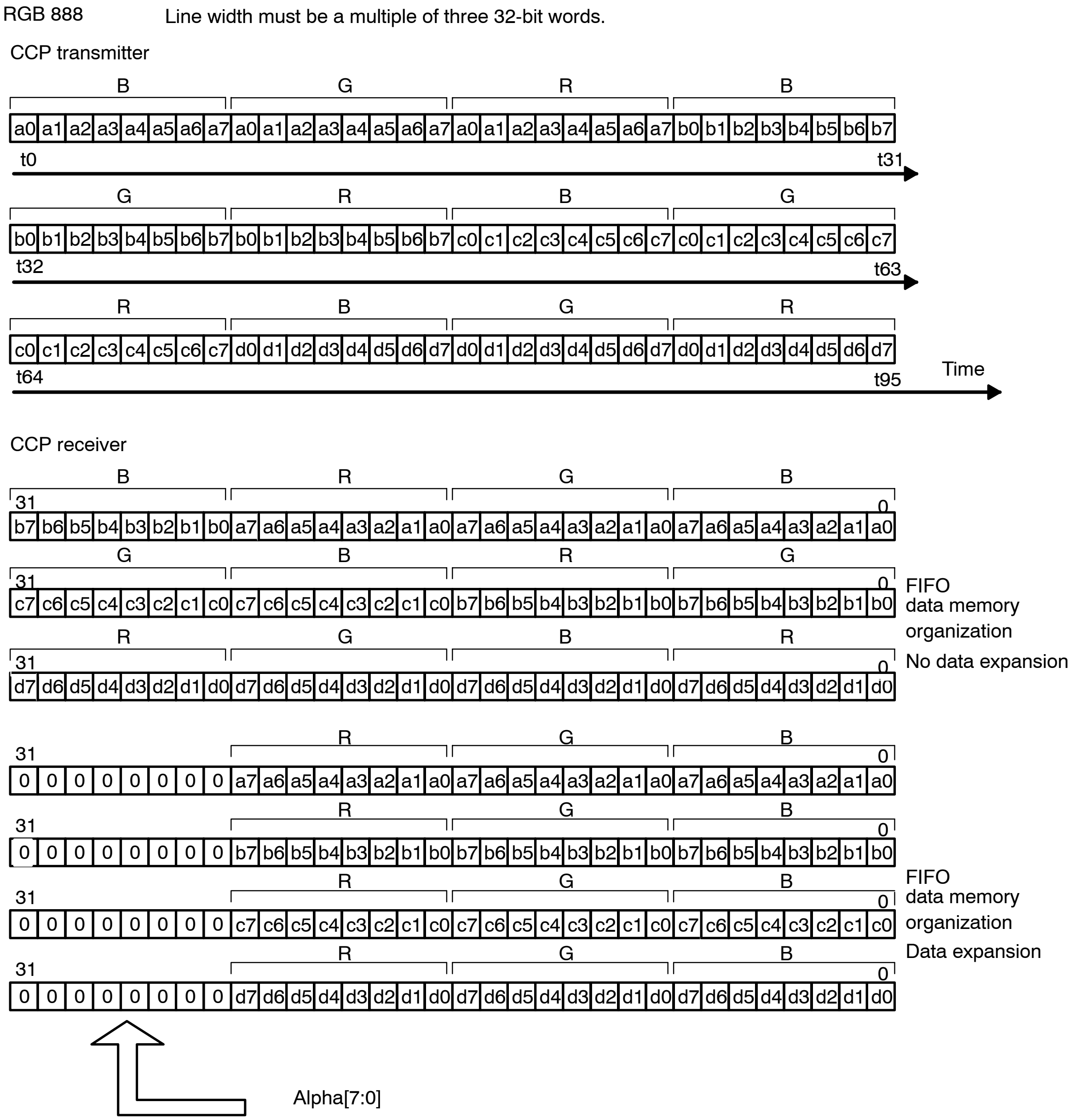

RGB 888
Line width must be a multiple of three 32-bit words.
CCP transmitter
B G R B
a0 a1 a2 a3 a4 a5 a6 a7 a0 a1 a2 a3 a4 a5 a6 a7 a0 a1 a2 a3 a4 a5 a6 a7 b0 b1 b2 b3 b4 b5 b6 b7
t0 t31
G R B G
b0 b1 b2 b3 b4 b5 b6 b7 b0 b1 b2 b3 b4 b5 b6 b7 c0 c1 c2 c3 c4 c5 c6 c7 c0 c1 c2 c3 c4 c5 c6 c7
t32 t63
R B G R
c0 c1 c2 c3 c4 c5 c6 c7 d0 d1 d2 d3 d4 d5 d6 d7 d0 d1 d2 d3 d4 d5 d6 d7 d0 d1 d2 d3 d4 d5 d6 d7
t64 t95
Time
CCP receiver
B R G B
31 0
b7 b6 b5 b4 b3 b2 b1 b0 a7 a6 a5 a4 a3 a2 a1 a0 a7 a6 a5 a4 a3 a2 a1 a0 a7 a6 a5 a4 a3 a2 a1 a0
G B R G
31 0
c7 c6 c5 c4 c3 c2 c1 c0 c7 c6 c5 c4 c3 c2 c1 c0 b7 b6 b5 b4 b3 b2 b1 b0 b7 b6 b5 b4 b3 b2 b1 b0
R G B R
31 0
d7 d6 d5 d4 d3 d2 d1 d0 d7 d6 d5 d4 d3 d2 d1 d0 d7 d6 d5 d4 d3 d2 d1 d0 d7 d6 d5 d4 d3 d2 d1 d0
FIFO
data memory
organization
No data expansion
R G B
31 0
0 0 0 0 0 0 0 0 a7 a6 a5 a4 a3 a2 a1 a0 a7 a6 a5 a4 a3 a2 a1 a0 a7 a6 a5 a4 a3 a2 a1 a0
R G B
31 0
0 0 0 0 0 0 0 0 b7 b6 b5 b4 b3 b2 b1 b0 b7 b6 b5 b4 b3 b2 b1 b0 b7 b6 b5 b4 b3 b2 b1 b0
R G B
31 0
0 0 0 0 0 0 0 0 c7 c6 c5 c4 c3 c2 c1 c0 c7 c6 c5 c4 c3 c2 c1 c0 c7 c6 c5 c4 c3 c2 c1 c0
R G B
31 0
0 0 0 0 0 0 0 0 d7 d6 d5 d4 d3 d2 d1 d0 d7 d6 d5 d4 d3 d2 d1 d0 d7 d6 d5 d4 d3 d2 d1 d0
FIFO
data memory
organization
Data expansion
Alpha[7:0]

Figure 14−19. RGB565

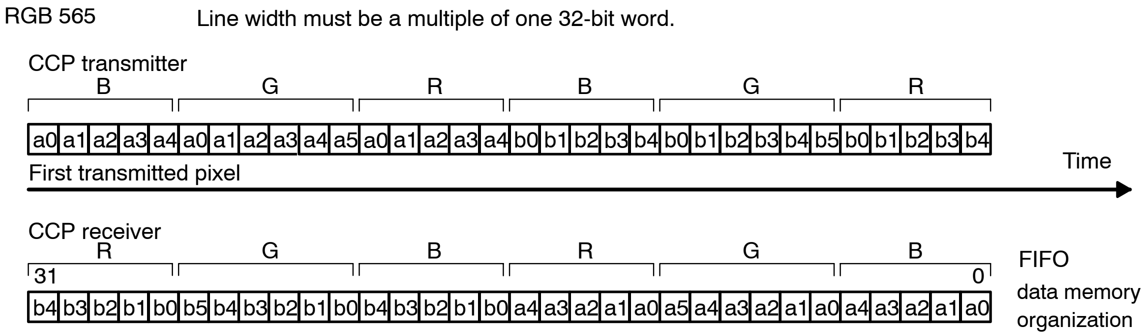

RGB 565
Line width must be a multiple of one 32-bit word.
CCP transmitter
B G R B G R
a0 a1 a2 a3 a4 a0 a1 a2 a3 a4 a5 a0 a1 a2 a3 a4 b0 b1 b2 b3 b4 b0 b1 b2 b3 b4 b5 b0 b1 b2 b3 b4
First transmitted pixel
Time
CCP receiver
R G B R G B
31
0
b4 b3 b2 b1 b0 b5 b4 b3 b2 b1 b0 b4 b3 b2 b1 b0 a4 a3 a2 a1 a0 a5 a4 a3 a2 a1 a0 a4 a3 a2 a1 a0
FIFO
data memory
organization

Figure 14−20. RGB444

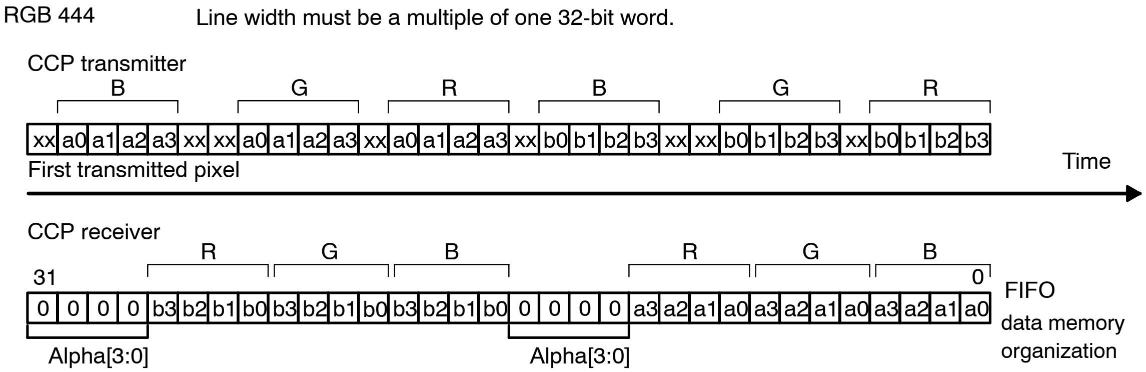

RGB 444
Line width must be a multiple of one 32-bit word.
CCP transmitter
B G R B G R
xx a0 a1 a2 a3 xx xx a0 a1 a2 a3 xx a0 a1 a2 a3 xx b0 b1 b2 b3 xx xx b0 b1 b2 b3 xx b0 b1 b2 b3
First transmitted pixel
Time
CCP receiver
R G B R G B
31
0
0 0 0 0 b3 b2 b1 b0 b3 b2 b1 b0 b3 b2 b1 b0 0 0 0 0 a3 a2 a1 a0 a3 a2 a1 a0 a3 a2 a1 a0
Alpha[3:0]
Alpha[3:0]
FIFO
data memory
organization

Figure 14−21. RAW8

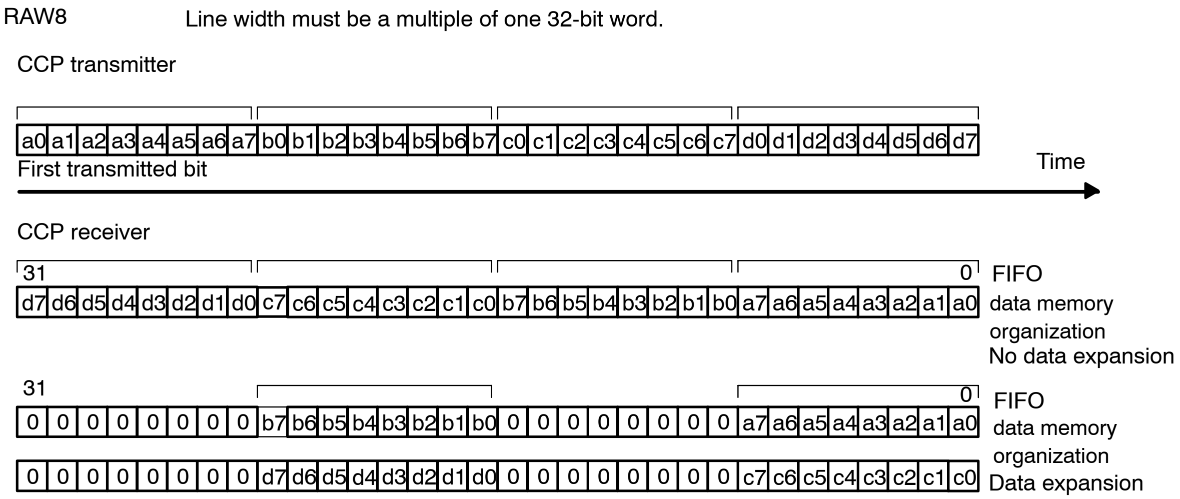
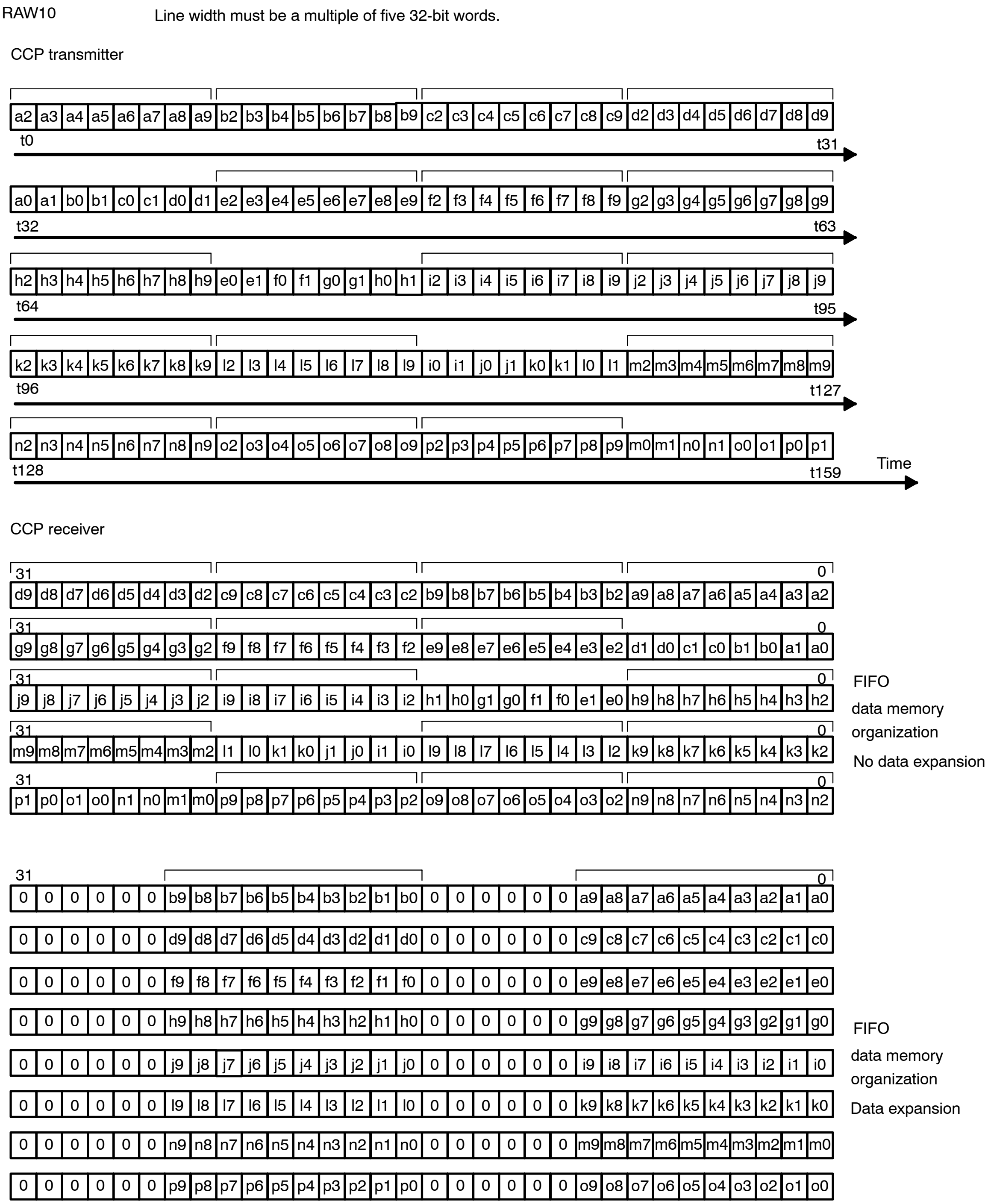

RAW8
Line width must be a multiple of one 32-bit word.
CCP transmitter
a0 a1 a2 a3 a4 a5 a6 a7 b0 b1 b2 b3 b4 b5 b6 b7 c0 c1 c2 c3 c4 c5 c6 c7 d0 d1 d2 d3 d4 d5 d6 d7
First transmitted bit
Time
CCP receiver
31
0
d7 d6 d5 d4 d3 d2 d1 d0 c7 c6 c5 c4 c3 c2 c1 c0 b7 b6 b5 b4 b3 b2 b1 b0 a7 a6 a5 a4 a3 a2 a1 a0
FIFO
data memory
organization
No data expansion
31
0
0 0 0 0 0 0 0 0 b7 b6 b5 b4 b3 b2 b1 b0 0 0 0 0 0 0 0 0 a7 a6 a5 a4 a3 a2 a1 a0
0 0 0 0 0 0 0 0 d7 d6 d5 d4 d3 d2 d1 d0 0 0 0 0 0 0 0 0 c7 c6 c5 c4 c3 c2 c1 c0
FIFO
data memory
organization
Data expansion

Figure 14−22. RAW10



RAW10 Line width must be a multiple of five 32-bit words.

CCP transmitter

a2	a3	a4	a5	a6	a7	a8	a9	b2	b3	b4	b5	b6	b7	b8	b9	c2	c3	c4	c5	c6	c7	c8	c9	d2	d3	d4	d5	d6	d7	d8	d9
a0	a1	b0	b1	c0	c1	d0	d1	e2	e3	e4	e5	e6	e7	e8	e9	f2	f3	f4	f5	f6	f7	f8	f9	g2	g3	g4	g5	g6	g7	g8	g9
h2	h3	h4	h5	h6	h7	h8	h9	e0	e1	f0	f1	g0	g1	h0	h1	i2	i3	i4	i5	i6	i7	i8	i9	j2	j3	j4	j5	j6	j7	j8	j9
k2	k3	k4	k5	k6	k7	k8	k9	l2	l3	l4	l5	l6	l7	l8	l9	i0	i1	j0	j1	k0	k1	l0	l1	m2	m3	m4	m5	m6	m7	m8	m9
n2	n3	n4	n5	n6	n7	n8	n9	o2	o3	o4	o5	o6	o7	o8	o9	p2	p3	p4	p5	p6	p7	p8	p9	m0	m1	n0	n1	o0	o1	p0	p1

CCP receiver

FIFO data memory organization No data expansion

31																															0
d9	d8	d7	d6	d5	d4	d3	d2	c9	c8	c7	c6	c5	c4	c3	c2	b9	b8	b7	b6	b5	b4	b3	b2	a9	a8	a7	a6	a5	a4	a3	a2
g9	g8	g7	g6	g5	g4	g3	g2	f9	f8	f7	f6	f5	f4	f3	f2	e9	e8	e7	e6	e5	e4	e3	e2	d1	d0	c1	c0	b1	b0	a1	a0
j9	j8	j7	j6	j5	j4	j3	j2	i9	i8	i7	i6	i5	i4	i3	i2	h1	h0	g1	g0	f1	f0	e1	e0	h9	h8	h7	h6	h5	h4	h3	h2
m9	m8	m7	m6	m5	m4	m3	m2	l1	l0	k1	k0	j1	j0	i1	i0	l9	l8	l7	l6	l5	l4	l3	l2	k9	k8	k7	k6	k5	k4	k3	k2
p1	p0	o1	o0	n1	n0	m1	m0	p9	p8	p7	p6	p5	p4	p3	p2	o9	o8	o7	o6	o5	o4	o3	o2	n9	n8	n7	n6	n5	n4	n3	n2

FIFO data memory organization Data expansion

31																															0
0	0	0	0	0	0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	0	0	0	0	0	0	a9	a8	a7	a6	a5	a4	a3	a2	a1	a0
0	0	0	0	0	0	d9	d8	d7	d6	d5	d4	d3	d2	d1	d0	0	0	0	0	0	0	c9	c8	c7	c6	c5	c4	c3	c2	c1	c0
0	0	0	0	0	0	f9	f8	f7	f6	f5	f4	f3	f2	f1	f0	0	0	0	0	0	0	e9	e8	e7	e6	e5	e4	e3	e2	e1	e0
0	0	0	0	0	0	h9	h8	h7	h6	h5	h4	h3	h2	h1	h0	0	0	0	0	0	0	g9	g8	g7	g6	g5	g4	g3	g2	g1	g0
0	0	0	0	0	0	j9	j8	j7	j6	j5	j4	j3	j2	j1	j0	0	0	0	0	0	0	i9	i8	i7	i6	i5	i4	i3	i2	i1	i0
0	0	0	0	0	0	l9	l8	l7	l6	l5	l4	l3	l2	l1	l0	0	0	0	0	0	0	k9	k8	k7	k6	k5	k4	k3	k2	k1	k0
0	0	0	0	0	0	n9	n8	n7	n6	n5	n4	n3	n2	n1	n0	0	0	0	0	0	0	m9	m8	m7	m6	m5	m4	m3	m2	m1	m0
0	0	0	0	0	0	p9	p8	p7	p6	p5	p4	p3	p2	p1	p0	0	0	0	0	0	0	o9	o8	o7	o6	o5	o4	o3	o2	o1	o0

Figure 14−23.   RAW12

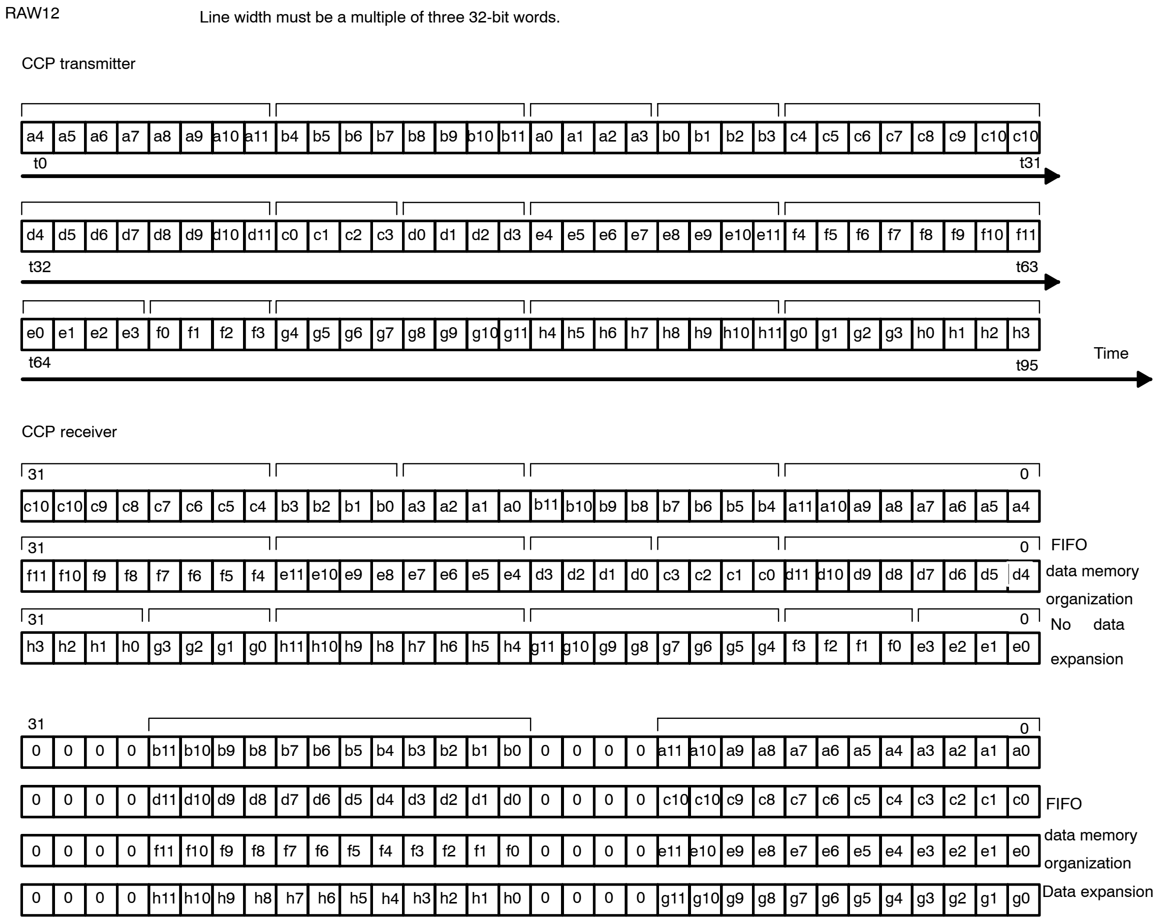
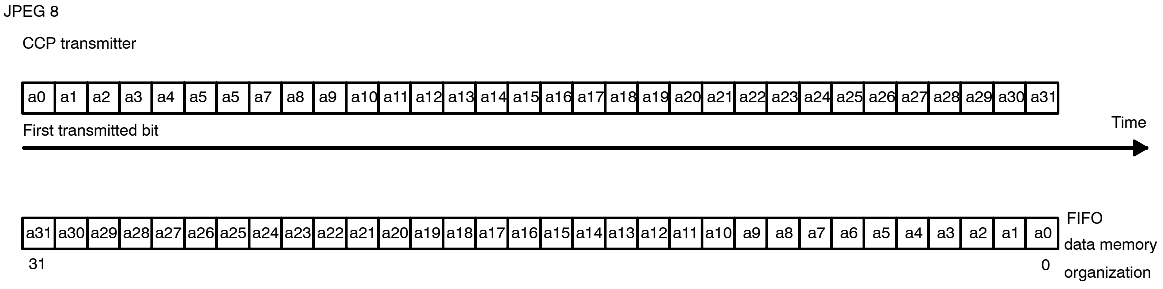

RAW12
Line width must be a multiple of three 32-bit words.
CCP transmitter
a4 a5 a6 a7 a8 a9 a10 a11 b4 b5 b6 b7 b8 b9 b10 b11 a0 a1 a2 a3 b0 b1 b2 b3 c4 c5 c6 c7 c8 c9 c10 c10
t0
t31
d4 d5 d6 d7 d8 d9 d10 d11 c0 c1 c2 c3 d0 d1 d2 d3 e4 e5 e6 e7 e8 e9 e10 e11 f4 f5 f6 f7 f8 f9 f10 f11
t32
t63
e0 e1 e2 e3 f0 f1 f2 f3 g4 g5 g6 g7 g8 g9 g10 g11 h4 h5 h6 h7 h8 h9 h10 h11 g0 g1 g2 g3 h0 h1 h2 h3
t64
t95
Time
CCP receiver
31
0
c10 c10 c9 c8 c7 c6 c5 c4 b3 b2 b1 b0 a3 a2 a1 a0 b11 b10 b9 b8 b7 b6 b5 b4 a11 a10 a9 a8 a7 a6 a5 a4
31
0
f11 f10 f9 f8 f7 f6 f5 f4 e11 e10 e9 e8 e7 e6 e5 e4 d3 d2 d1 d0 c3 c2 c1 c0 d11 d10 d9 d8 d7 d6 d5 d4
31
0
h3 h2 h1 h0 g3 g2 g1 g0 h11 h10 h9 h8 h7 h6 h5 h4 g11 g10 g9 g8 g7 g6 g5 g4 f3 f2 f1 f0 e3 e2 e1 e0
FIFO
data memory
organization
No data
expansion
31
0
0 0 0 0 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0 0 0 0 0 a11 a10 a9 a8 a7 a6 a5 a4 a3 a2 a1 a0
0 0 0 0 d11 d10 d9 d8 d7 d6 d5 d4 d3 d2 d1 d0 0 0 0 0 c10 c10 c9 c8 c7 c6 c5 c4 c3 c2 c1 c0
0 0 0 0 f11 f10 f9 f8 f7 f6 f5 f4 f3 f2 f1 f0 0 0 0 0 e11 e10 e9 e8 e7 e6 e5 e4 e3 e2 e1 e0
0 0 0 0 h11 h10 h9 h8 h7 h6 h5 h4 h3 h2 h1 h0 0 0 0 0 g11 g10 g9 g8 g7 g6 g5 g4 g3 g2 g1 g0
FIFO
data memory
organization
Data expansion

Figure 14−24.   JPEG8


JPEG 8
CCP transmitter
a0 a1 a2 a3 a4 a5 a5 a7 a8 a9 a10 a11 a12 a13 a14 a15 a16 a17 a18 a19 a20 a21 a22 a23 a24 a25 a26 a27 a28 a29 a30 a31
First transmitted bit
Time
a31 a30 a29 a28 a27 a26 a25 a24 a23 a22 a21 a20 a19 a18 a17 a16 a15 a14 a13 a12 a11 a10 a9 a8 a7 a6 a5 a4 a3 a2 a1 a0
31
0
FIFO
data memory
organization

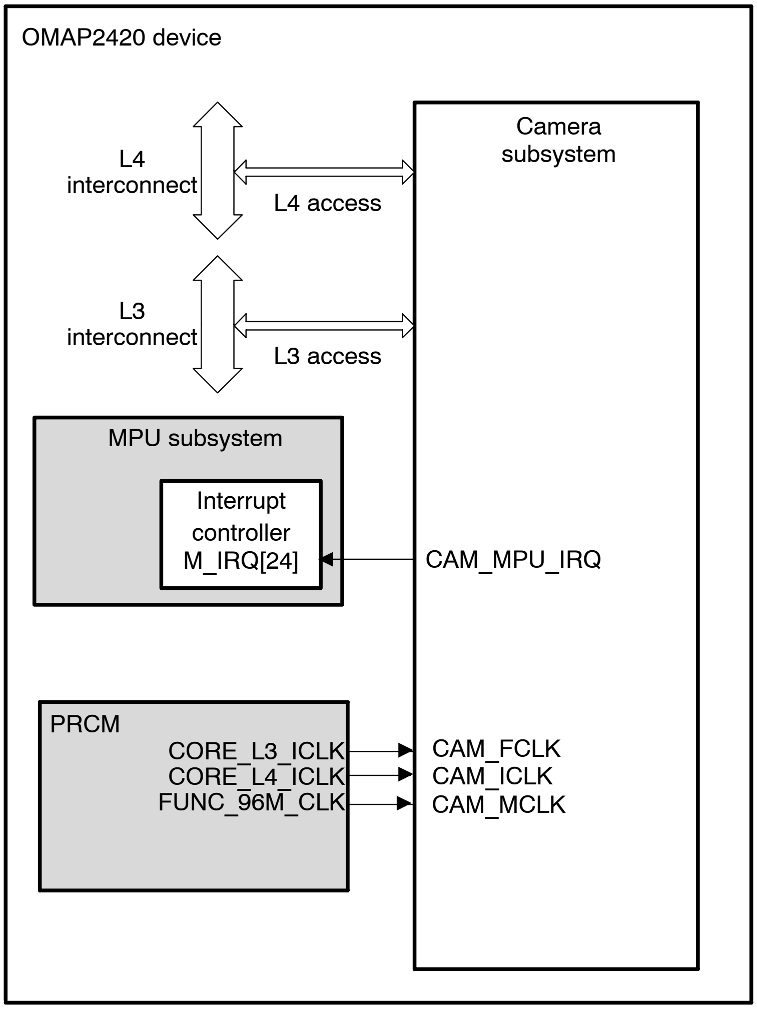
Note:

The JPEG8 FSP is a JPEG format in which FSP is a decoding performed by the receiving end on the JPEG8 format. FSP decoding removes the 0xA5 data inserted by the FSP encoder to suppress violating combinations on the data stream. The FSP decoding occurs after the FSC is received.

14.3 Camera Subsystem Integration

Figure 14−25 highlights camera subsystem integration in the OMAP2420.

Figure 14−25. Camera Subsystem Integration



14.3.1 Clocking, Reset, and Power-Management Scheme

There are four clock domains in the camera subsystem:

- ❏ Serial sensor clock domain
- ❏ Parallel sensor clock domain
- ❏ Functional clock domain
- ❏ Interface clock domain

14.3.1.1 Camera Subsystem Clocks

Serial Sensor Clock Domain

CAM_S_CLK is the serial sensor clock provided through the CAM.D7 and CAM.D6 differential clock inputs from an external serial image sensor. Its frequency can be up to 208 MHz. For details, see Section 14.2, *Camera Subsystem Environment*, and Section 14.4, *Camera Subsystem Functional Description*.

Parallel Sensor Clock Domain

CAM_P_CLK is the parallel sensor clock. It is provided through the CAM.LCLK input from an external parallel image sensor. Its frequency can be up to 96 MHz for 8-bit data and 48 MHz for 10-bit data. For details, see Section 14.2, *Camera Subsystem Environment*, and Section 14.4, *Camera Subsystem Functional Description*.

Functional Clock Domain

Two clock signals belong to the functional clock domain:

- CAM_MCLK is the 96-MHz functional clock provided by the power, reset, and clock management (PRCM) module and used in the camera subsystem to generate image-sensor clock output (CAM.XCLK). For details about CAM.XCLK generation, see Section 14.4, *Camera Subsystem Functional Description*.

  CAM_MCLK is derived from PRCM APLL 96M. When the camera subsystem does not require the CAM_MCLK, software can disable it at the PRCM level by setting the EN_CAM bit (CM_FCLKEN1_CORE[31] in the PRCM registers). For more information about CAM_MCLK, see Chapter 5, *Power, Reset, and Clock Management*.

Note:

CAM_MCLK is disabled only if other modules that receive it do not require it.

- CAM_FCLK runs at the L3 clock frequency. It is used as a functional clock on the camera subsystem L3 master interface.

  The CAM_FCLK source is CORE_l3_ICLK, provided by the PRCM module. Its frequency can be adjusted at system level using the CLKSEL_L3 bit field (CM_CLKSEL1_CORE[4:0] in the PRCM registers).

  When the camera subsystem does not require CAM_FCLK, software can disable it at the PRCM level by setting the EN_CAM bit (CM_ICLKEN1_CORE[31] in the PRCM registers). It is also possible to activate autoidle mode for this clock (by setting the PRCM AUTO_CAM bit [CM_AUTOIDLE_CORE[31]] to 1). In this case, CAM_FCLK follows the behavior of the OMAP2420 L3 clock domain. For more information, see Chapter 5, *Power, Reset, and Clock Management*.

Note:

CAM_FCLK is disabled only if other modules that receive it do not require it.

Interface Clock Domain

CAM_ICLK is the interface clock. It runs at L4 interconnect clock speed and is used to trigger access to the camera subsystem L4 interface. Its source is the PRCM CORE_L4_ICLK signal. It can run at L3 frequency or L3/2 frequency (selected using the PRCM CLKSEL_L4 bit field [CM_CLKSEL1_CORE[6:5]).

When the camera subsystem does not require the CAM_ICLK, software can disable it at the PRCM level by setting the EN_CAM bit (CM_ICLKEN1_CORE[31] in the PRCM registers). It is also possible to activate autoidle mode for this clock (by setting the PRCM AUTO_CAM bit [CM_AUTOIDLE1_CORE[31]] to 1). In this case, CAM_ICLK follows the behavior of the OMAP2420 L4 clock domain. For more information, see Chapter 5, *Power, Reset, and Clock Management*.

Note:

CAM_MCLK is disabled only if other modules that receive it do not require it.

14.3.1.2 Camera Subsystem Reset Scheme

Global reset of the module is accomplished either by activation of the CORE_RST_N signal in the CORE_RST domain (for more information, see Chapter 5, *Power, Reset, and Clock Management*) or by setting the camera subsystem CAM_SYSCONFIG register soft reset bit (CAM_SYSCONFIG[1]) to 1. Setting this bit enables active software reset functionally equivalent to hardware reset.

The camera core also provides a software reset (the camera core CC_SYSCONFIG[1] bit) that resets only the core independently of the camera subsystem.

14.3.1.3 Camera Subsystem Power Domain

The camera subsystem belongs to the CORE power domain. For information about the CORE power domain, see Chapter 5, *Power, Reset, and Clock Management*.

14.3.2 Hardware Requests

The camera subsystem can generate two interrupts:

- ❑ CAM_IRQ0 is an interrupt to the MPU subsystem interrupt controller. It is mapped on M_IRQ_24.
- ❑ CAM_IRQ1 is an interrupt to the DSP subsystem interrupt controller. It is mapped on D_L2_IRQ_29.

For details, see Section 14.3.3, *Pin List and Pad Multiplexing with Other Functions*.

14.3.3 Pin List and Pad Multiplexing With Other Functions

Table 14−5 shows the pin multiplexing functions for the camera subsystem. Gray shading indicates that camera I/Os are used. Black shading indicates that the mode is not used.

Table 14−5. Pin Multiplexing for Camera Subsystem

Interface				Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
Function n	**Description**	**DIR**	**Ball**	**Alternate Functions**					
CAM.HS	Camera horizontal synchronization	I	T3		hw.dbg1	mcbsp1.dx	gpio.55		
CAM.VS	Camera vertical synchronization	I	U2		hw.dbg0	mcbsp1.fsx	gpio.56		
CAM.XCLK	Camera clock output	O	U3		14.ext.trg	sti.clk			
CAM.D0	Camera digital image data bit 0	I	T4		hw.dbg2	sti.dout	gpio.54		
CAM.D1	Camera digital image data bit 1	I	V2		hw.dbg3	sti.din	gpio.53		
CAM.D2	Camera digital image data bit 2	I	V3		hw.dbg4	mcbsp1.clkx	gpio.52		
CAM.D3	Camera digital image data bit 3	I	U4		hw.dbg5	mcbsp1.dr	gpio.51		
CAM.D4	Camera digital image data bit 4	I	W2		hw.dbg6	mcbsp1.fsr	gpio.50		
CAM.D5	Camera digital image data bit 5	I	V4		hw.dbg7	mcbsp1.clkr	gpio.49		
CAM.D6	Camera digital image data bit 6	I	W3		hw.dbg8				
			AA8			14.ext_trig	gpio.58		
CAM.D7	Camera digital image data bit 7	I	Y2		hw.dbg9				
			AA4		usb2.tllse0		gpio.15		
CAM.D8	Camera digital image data bit 8	I	Y4		hw.dbg10		gpio.54		
			AA6		usb2.rcv	sys.ndmareq1	gpio.14		
CAM.D9	Camera digital image data bit 9	I	V6		hw.dbg11		gpio.53		
			Y5	gpio.120			gpio.120		
CAM.LCLK	Camera image data latch clock	I	V5			mcbsp.clks	gpio.57		

14.3.4 L4 Interconnect Interface

The camera subsystem interfaces with the L4 interconnect through a dedicated target agent (TA). This TA, which is part of the L4 interconnect, provides status and configuration registers as listed in Table 14−6. By default, TA register values are functional, but can be overwritten. For a complete description, see Section 6.4, *L4 Interconnect*, in Chapter 6, *Internal Interconnect*.

Table 14−6. L4 Interconnect Registers

Register Name	Type	Register Width (Bits)	Physical Address
COMPONENT	R	32	0x4805 3000
AGENT_CONTROL	R/W	32	0x4805 3020
AGENT_STATUS	R	32	0x4805 3028

14.3.5 L3 Interconnect Interface

Table 14−7 lists the L3 interconnect registers.

Table 14−7. L3 Interconnect Registers

Register Name	Type	Register Width (Bits)	Physical Address
SBIMERRLOGA	R	32	0x6800 14AB
SBIMERRLOG	R/W	32	0x6800 14B0
SBIMSTATE	R/W	32	0x6800 1590
SBTMSTATE_L	R/W	32	0x6800 1598
SBTMSTATE_H	R/W	32	0x6800 159C
SBIMCONFIG_L	R/W	32	0x6800 15A8
SBIMCONFIG_H	R/W	32	0x6800 15AC
SBID_L	R	32	0x6800 15F8
SBID_H	R	32	0x6800 15FC

14.3.6 Register Summary

Table 14−8 summarizes the camera subsystem registers.

Table 14−8. Register Summary for Camera Subsystem

Register Name	Type	Register Width (Bits)	Physical Address
		Camera Top	
CAM_REVISION	R	32	0x4805 2000
CAM_SYSCONFIG	R/W	32	0x4805 2010
CAM_SYSSTATUS	R	32	0x4805 2014
CAM_IRQSTATUS	R	32	0x4805 2018
CAM_GPO	R/W	32	0x4805 2040
CAM_GPI	R	32	0x4805 2050
		Camera Core	
CC_REVISION	R	32	0x4805 2400
CC_SYSCONFIG	R/W	32	0x4805 2410
CC_SYSSTATUS	R	32	0x4805 2414

Table 14−8. Register Summary for Camera Subsystem (Continued)

Register Name	Type	Register Width (Bits)	Physical Address
Camera Core (Continued)			
CC_IRQSTATUS	R/W	32	0x4805 2418
CC_IRQENABLE	R/W	32	0x4805 241C
CC_CTRL	R/W	32	0x4805 2440
CC_CTRL_DMA	R/W	32	0x4805 2444
CC_CTRL_XCLK	R/W	32	0x4805 2448
CC_FIFODATA	R/W	32	0x4805 244C
CC_TEST	R	32	0x4805 2450
CC_GENPAR	R	32	0x4805 2454
CC_CCPFSCR	R/W	32	0x4805 2458
CC_CCPFECR	R/W	32	0x4805 245C
CC_CCPLSCR	R/W	32	0x4805 2460
CC_CCPLECR	R/W	32	0x4805 2464
CC_CCPDFR	R/W	32	0x4805 2468
DMA			
DMA4_REVISION	R	32	0x4805 2800
DMA4_IRQSTATUS_L0	R/W	32	0x4805 2808
DMA4_IRQSTATUS_L1	R/W	32	0x4805 280C
DMA4_IRQSTATUS_L2	R/W	32	0x4805 2810
DMA4_IRQSTATUS_l3	R/W	32	0x4805 2814
DMA4_IRQENABLE_L0	R/W	32	0x4805 2818
DMA4_IRQENABLE_L1	R/W	32	0x4805 281C
DMA4_IRQENABLE_L2	R/W	32	0x4805 2820
DMA4_IRQENABLE_L3	R/W	32	0x4805 2824
DMA4_SYSSTATUS	R	32	0x4805 2828
DMA4_OCP_SYSCONFIG	R/W	32	0x4805 282C
DMA4_CAPS_0	R	32	0x4805 2864
DMA4_CAPS_2	R	32	0x4805 286C
DMA4_CAPS_3	R	32	0x4805 2870
DMA4_CAPS_4	R	32	0x4805 2874
DMA4_GCR	R/W	32	0x4805 2878
DMA4_CCR	R/W	32	0x4805 2880 + n*0x60
DMA4_CLNK_CTRL	R/W	32	0x4805 2884 + n*0x60
DMA4_CICR	R/W	32	0x4805 2888 + n*0x60
DMA4_CSR	R/W	32	0x4805 288C + n*0x60
DMA4_CSDP	R/W	32	0x4805 2890 + n*0x60

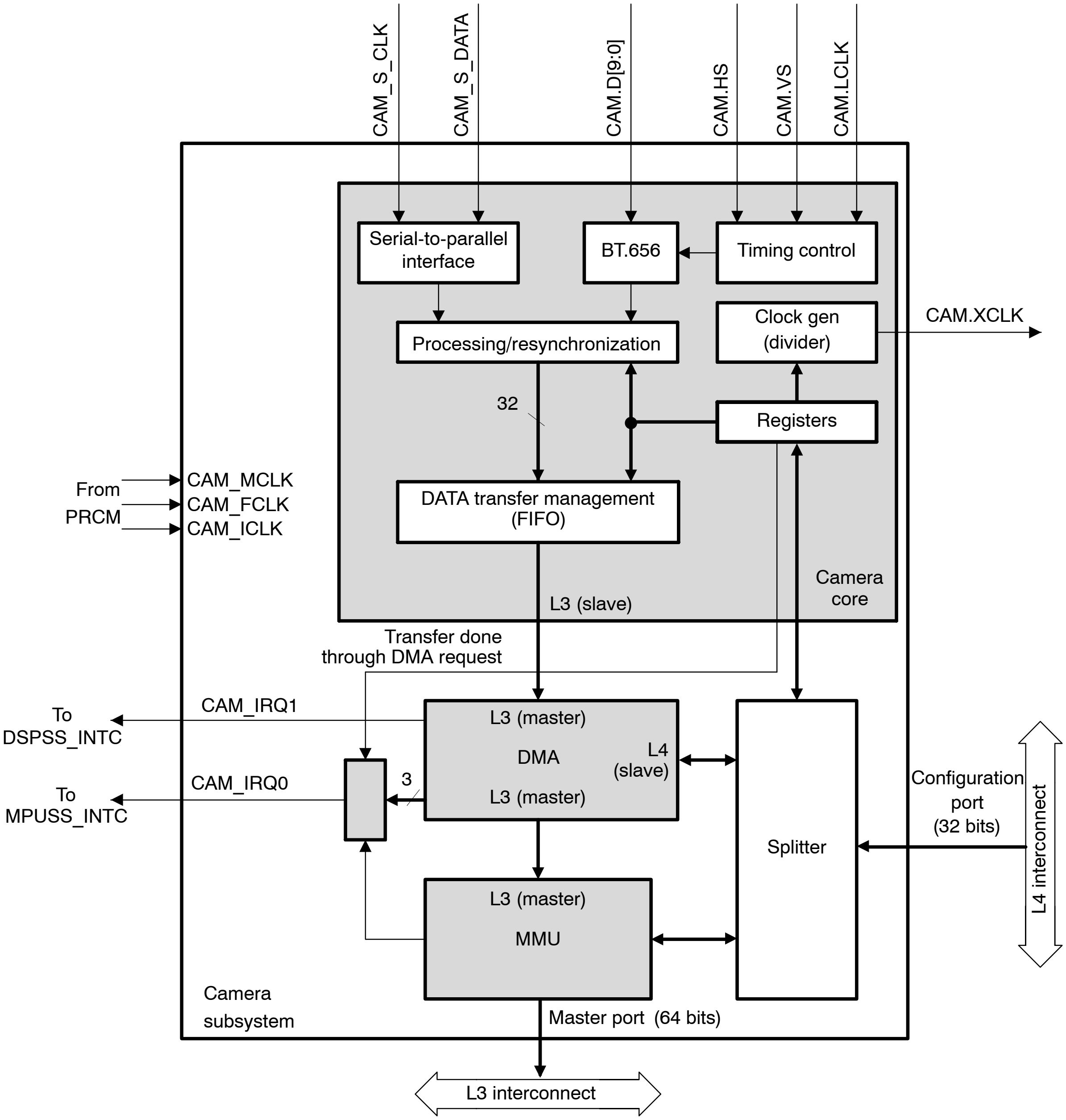
Table 14−8. Register Summary for Camera Subsystem (Continued)

Register Name	Type	Register Width (Bits)	Physical Address
DMA (Continued)			
DMA4_CEN	R/W	32	0x4805 2894 + n*0x60
DMA4_CFN	R/W	32	0x4805 2898 + n*0x60
DMA4_CSSA	R/W	32	0x4805 289C + n*0x60
DMA4_CDSA	R/W	32	0x4805 28A0 + n*0x60
DMA4_CSEI	R/W	32	0x4805 28A4 + n*0x60
DMA4_CSFI	R/W	32	0x4805 28A8 + n*0x60
DMA4_CDEI	R/W	32	0x4805 28AC + n*0x60
DMA4_CDFI	R/W	32	0x4805 28B0 + n*0x60
DMA4_CSAC	R	32	0x4805 28B4 + n*0x60
DMA4_CDAC	R	32	0x4805 28B8 + n*0x60
DMA4_CCEN	R	32	0x4805 28BC + n*0x60
DMA4_CCFN	R	32	0x4805 28C0 + n*0x60
DMA4_COLOR	R/W	32	0x4805 28C4 + n*0x60
MMU			
MMU_REVISION	R	32	0x4805 2C00
MMU_SYSCONFIG	R/W	32	0x4805 2C10
MMU_SYSSTATUS	R	32	0x4805 2C14
MMU_IRQSTATUS	R/W	32	0x4805 2C18
MMU_IRQENABLE	R/W	32	0x4805 2C1C
MMU_WALKING_ST	R/W	32	0x4805 2C40
MMU_CNTL	R/W	32	0x4805 2C44
MMU_FAULT_AD	R	32	0x4805 2C48
MMU_TTB	R/W	32	0x4805 2C4C
MMU_LOCK	R/W	32	0x4805 2C50
MMU_LD_TLB	R/W	32	0x4805 2C54
MMU_CAM	R/W	32	0x4805 2C58
MMU_RAM	R/W	32	0x4805 2C5C
MMU_GFLUSH	R/W	32	0x4805 2C60
MMU_FLUSH_ENTRY	R/W	32	0x4805 2C64
MMU_READ_CAM	R	32	0x4805 2C68
MMU_READ_RAM	R	32	0x4805 2C6C
MMU_EMU_FAULT_AD	R	32	0x4805 2C70

14.4 Camera Subsystem Functional Description

Figure 14−26 shows the camera subsystem.

Figure 14−26. Camera Subsystem



The camera core transfers data from the image sensor to the buffer (FIFO) and generates DMA requests.

The DMA splits pictures to the correct virtual address of the memory space. The MMU translates virtual address space into physical address space.

The camera subsystem can be used in several configurations to accommodate different image sensors:

- CCP configuration (serial): The compact camera port bit stream has embedded synchronization signals (frame start, line start, line end, and frame end).
- BT.656 configuration (parallel): The BT.656 block extracts the SAV, EAV, and data signals. This configuration can work with 8- and 10-bit YUV422 data.
- No BT.656 configuration (parallel): The BT.656 block uses the CAM.HS and CAM.VS signals to know when data is valid. This configuration can work with 8- and 10-bit data.

The serial and parallel interfaces cannot be active at the same time.

There are four clock domains in the camera subsystem:

- Serial sensor clock domain

  Serial interface: Frequency up to 208 MHz.

- Parallel sensor clock domain

  Parallel interface: Frequency depends on the imaging sensor type and size, its frame rate, and its blanking time. This clock domain can be as high as 96 MHz for 8-bit data and 48 MHz for 10-bit data.

- Interconnect clock

  The interconnect clock is used in the block that includes the L3 and L4 interfaces. The interconnect configuration interface can work at the same speed as the interconnect functional interface, but it can also work at any divided frequency to access the configuration registers.

  The configuration port can be accessed at any time, even if picture capture is not enabled.

- Functional clock (CAM_MCLK)

  The camera subsystem can provide a clock to the external image-sensor module (CAM.XCLK). This clock is derived from the functional clock CAM_MCLK.

14.4.1 Compact Camera Port

14.4.1.1 CCP Features

The CCP interface has several image data operating modes, as described in Table 14−4.

Each of these data formats has a constraint in terms of number of 32-bit words per line. Table 14−9 lists the constraints on the CCP data formats.

Table 14−9. Constraints on Line Length for CCP Data Formats

Data Format	Constraint on Line Length
YUV422, YUV420, RGB565, RGB444, RAW8, JPEG8, JPEG8 FSP	Line must be a multiple of one 32-bit word.
RGB888	Line must be a multiple of three 32-bit words.
RAW10	Line must be a multiple of five 32-bit words.
RAW12	Line must be a multiple of three 32-bit words.

If these constraints are not met, an error is flagged in the interrupt status register. Bit FW_ERR_IRQ in the camera core CC_IRQSTATUS register (CC_IRQSTATUS[10]) is set to 1.

The following data formats are identical for the CCP receiver:

- ☐ YUV422
- ☐ YUV420
- ☐ RGB888 (no data expansion)
- ☐ RGB565
- ☐ RAW8 (no data expansion)
- ☐ RAW10 (no data expansion)
- ☐ RAW12 (no data expansion)
- ☐ JPEG8

The following data formats require dedicated treatment by the CCP receiver:

- ☐ RGB888 (data expansion)
- ☐ RGB444
- ☐ RAW8 (data expansion)
- ☐ RAW10 (data expansion)
- ☐ RAW12 (data expansion)

The data mode is selected using the CCP data format select register. Table 14−10 summarizes the programming of the DATAFORMATSELECT field in the camera core CCP data format select register (CC_CCPDFR).

Table 14−10. DATAFORMATSELECT Programming in CC_CCPDFR

Data Format	CC_CCPDFR[3:0]
YUV422 big-endian	0000
YUV422	0001
YUV420	0010
RGB444	0100
RGB565	0101
RGB888 (no data expansion)	0110
RGB888 (data expansion)	0111
RAW8 (no data expansion)	1000
RAW8 (data expansion)	1001

Table 14−10. DATAFORMATSELECT Programming in CC_CCPDFR (Continued)

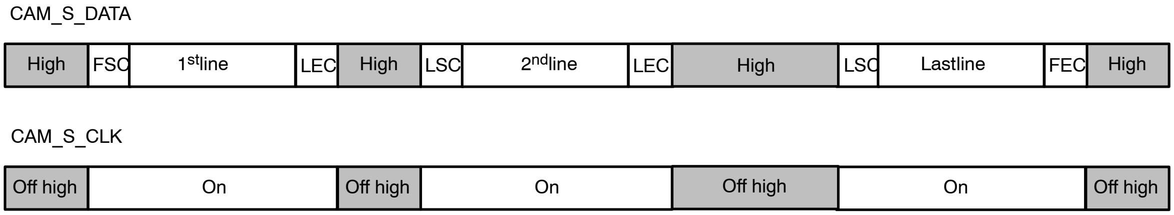
Data Format	CC_CCPDFR[3:0]
RAW10 (no data expansion)	1010
RAW10 (data expansion)	1011
RAW12 (no data expansion)	1100
RAW12 (data expansion)	1101
JPEG8 FSP	1110
JPEG8	1111

14.4.1.2 CCP Timing

Figure 14−27 shows the timing diagrams of CCP serial interface signals CAM_S_DATA and CAM_S_CLK. The CCP bit stream has embedded synchronization signals: FSC, LEC, LSC, and FEC.

CAM_S_DATA is registered at the rising edge of the clock CAM_S_CLK.

Figure 14−27. Timing Diagrams of CAM_S_DATA and CAM_S_CLK



The clock signal CAM_S_CLK can be free-running or gated. If a gated transmission clock is used, CAM_S_CLK remains high when nothing is being transferred, except in power shutdown. If a gated clock is used, the receiver reacts to the FEC immediately, because when the transmitter sends the last bit of the synchronization code, the clock signal remains high until the next frame starts. The receiver does not need additional transmission clock cycles after the FEC.

14.4.1.3 CCP Synchronization Code

Reception of data from the image-sensor module uses four synchronization codes embedded in the serial bit stream:

- FSC: Identifies the start of a new frame.
- LSC: Identifies the start of a new line.
- LEC: Identifies the end of a line. It is received for every line, except the last, which ends with an FEC.
- FEC: Identifies the end of the last line and the end of the current frame.

Every synchronization code is transmitted byte-wise least-significant bit (LSB) first. For example, the code 0xFF00:0002 transmitted from the image sensor corresponds to the following bit stream: 11111111 – 00000000 – 00000000 – 01000000.

Every default code starts with a set of eight 1s and sixteen 0s that are never received in pixel data (because having eight 1s and sixteen 0s is not allowed in pixel data).

The CCP synchronization codes are programmable through the camera core configuration registers CC_CCPFSCR, CC_CCPLECR, CC_CCPLSCR, and CC_CCPFECR.

Table 14−11 lists the CCP synchronization codes after hardware reset.

Table 14−11. CCP Synchronization Codes

Data Types	Code	Configuration Register	Reset Value
All data types	Line start	CC_CCPLSCR	0xFF00:0000
	Line end	CC_CCPLECR	0xFF00:0001
	Frame start	CC_CCPFSCR	0xFF00:0002
	Frame end	CC_CCPFECR	0xFF00:0003

Synchronization is operated bit-wise. Every line is composed of a finite number of 32-bit CCP words.

Two synchronization errors can occur:

- ☐ FALSESYNCCODE means that the detection of the synchronization code was out of order. When it occurs, the FSC_ERR_IRQ bit in the camera core interrupt status register (CC_IRQSTATUS[9]) is flagged.
- ☐ SHIFTEDSYNCCODE means that detection of a synchronization code on a non-32-bit boundary occurred. In this case, the SSC_ERR_IRQ bit in the camera core interrupt status register (CC_IRQSTATUS[8]) is flagged. This synchronization error occurs only with LECs and FECs.

14.4.2 Parallel No BT656 (No BT)

14.4.2.1 No BT Features

In this configuration, no assumptions are made about data format, but the parallel interface size is limited to 8 or 10 bits.

14.4.2.2 Description

Pixel data is presented on CAM.D. One pixel is latched every CAM.LCLK rising edge (or falling edge, depending on the configuration of CAM.LCLK polarity).

CAM.LCLK polarity is defined by the PAR_CLK_POL bit in the camera core control register (CC_CTRL[10]).

Additional pixel times between rows represent a blanking period. Active pixels are identified by a combination of two additional timing signals: CAM.HS (horizontal synchronization) and CAM.VS (vertical synchronization).

During the image-sensor readout, these signals define when a row of valid data begins and ends and when a frame starts and ends.

A bit field sets CAM.HS polarity (NOBT_HS_POL) and CAM.VS polarity (NOBT_VS_POL) in the camera core control register (CC_CTRL[9] and CC_CTRL[8], respectively).

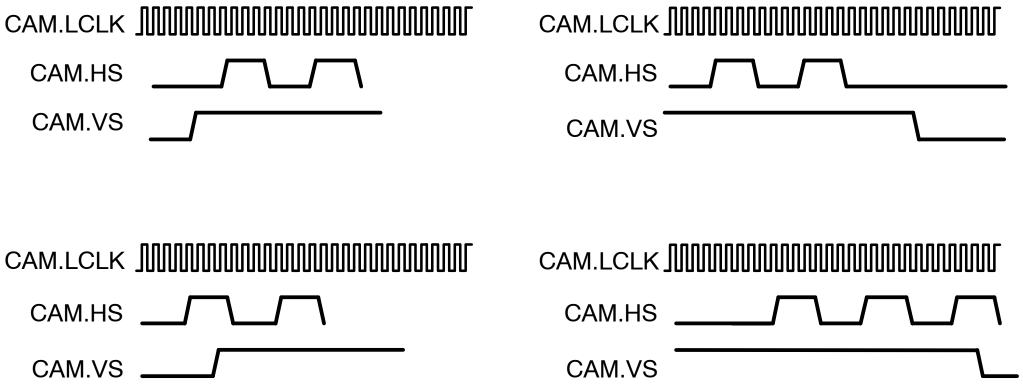
Note:

The CAM.LCLK clock runs during blanking periods (when CAM.HS and CAM.VS are inactive).

The scenarios in Figure 14−28 are allowed in No BT configuration; in other words, data is accepted while CAM.HS and CAM.VS are both active.

Figure 14−28. CAM.HS and CAM.VS Scenaros

Data is valid when both CAM.HS and CAM.VS are active (high in this example)



The camera core supports decimation from the image sensor where CAM.HS toggles between pixels. Figure 14−29 shows this scenario.

Figure 14−29. CAM.HS Toggles Between Pixels in Decimation

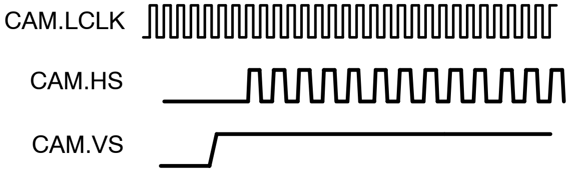
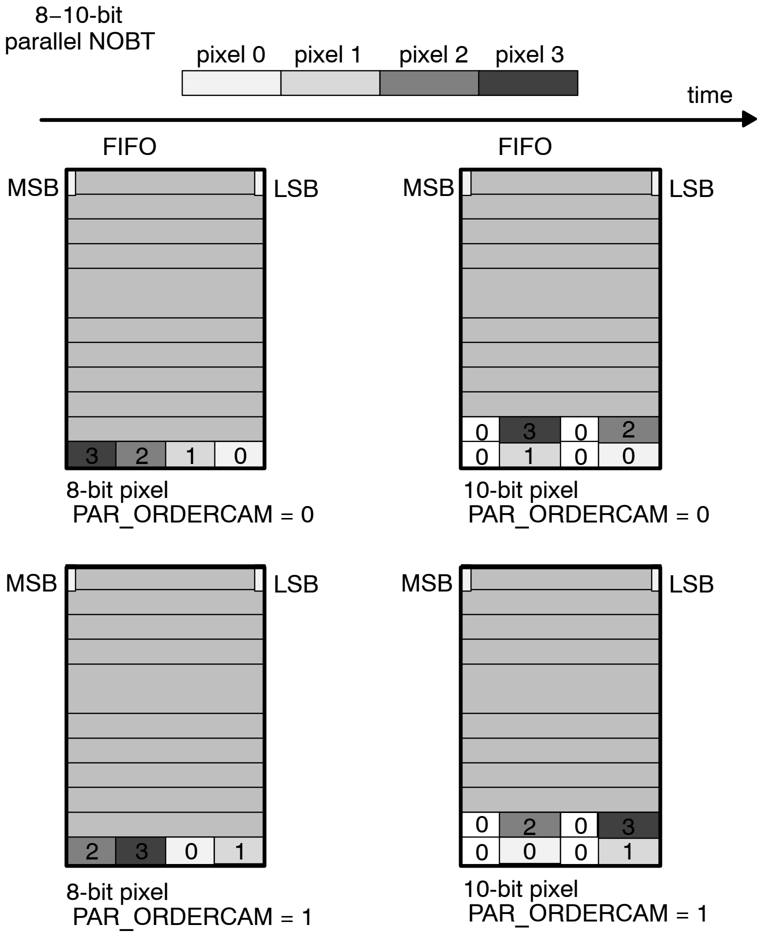


Image data is stored differently in the FIFO depending on the setting of the camera core PAR_ORDERCAM bit (CC_CTRL[11]), as shown in Figure 14−30.

Figure 14−30. FIFO Image Data for 8−10 Bits With PAR_ORDERCAM



14.4.3 BT.656

14.4.3.1 BT.656 Features

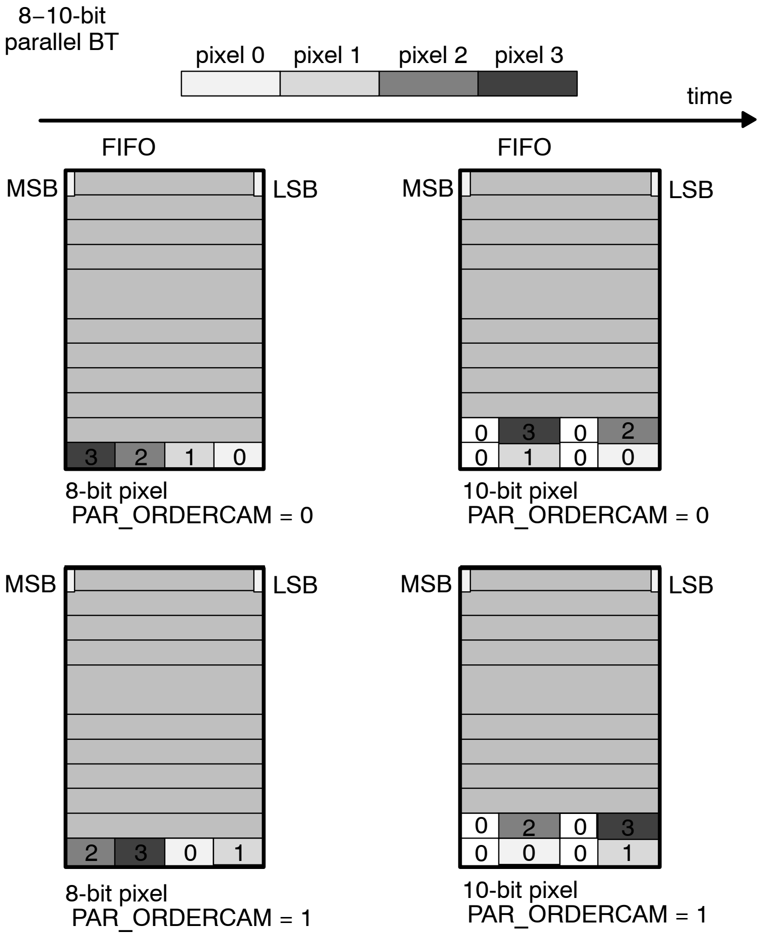
The camera interface supports data in ITU-R BT.656 format. This standard specifies a method of transferring YUV422 data stored in the FIFO over an 8- or 10-bit video interface.

14.4.3.2 Description

Error correction can be enabled or disabled by configuring the BT_CORRECT bit in the camera core control register (CC_CTRL[12]).

Image data is stored differently in the FIFO depending on the setting of camera core CC_CTRL register PAR_ORDERCAM bit (CC_CTRL[11]), as shown in Figure 14−31.

Figure 14−31. FIFO Image 8- and 10-Bit Data in BT Format



14.4.4 FIFO Transfer

The FIFO is a buffer that operates based on a first-in, first-out principle; it is synchronous with the functional clock. The FIFO receives data from either the CCP serial sensor interface or the parallel interface. Both interfaces cannot be active at the same time.

When the write FIFO counter reaches the trigger level, a DMA request is generated and remains active until the DMA transfer is complete. The next DMA request occurs if the number of remaining 32-bit words is above the threshold and the DMA has completed the transfer. A bit field (FIFO_THRESHOLD) sets the trigger level in the camera core control DMA register (CC_CTRL_DMA[6:0]).

The FIFO goes into overflow when a write is attempted to a full FIFO. The content of the full FIFO is not corrupted by a write attempt. During FIFO overflow, it is still possible to read data from the FIFO. More writes are possible when the FIFO is no longer full.

The FIFO goes into underflow when a read is attempted from an empty FIFO. All writes to the FIFO are possible. Reads are also possible when the FIFO is no longer empty.

The FIFO is reset by writing 1 to the CC_RST bit of the camera core control register (CC_CTRL[18]).

If the FIFO overflow interrupt is enabled (by setting the FIFO_OF_IRQ_EN bit of the camera core interrupt enable register [CC_IRQENABLE[1]] to 1), a FIFO overflow generates an interrupt. Writing 1 to the bit clears the interrupt. There is no need to apply CC_RST beforehand.

If the FIFO underflow interrupt is enabled (by setting the FIFO_UF_IRQ_EN bit of the camera core interrupt enable register [CC_IRQENABLE[0]] to 1), a FIFO underflow generates an interrupt. Writing 1 to the bit clears the interrupt. There is no need to apply CC_RST beforehand.

The FIFO can be accessed from L4 port I (read/write) and L4 port II (read). Table 14−12 shows how to access the FIFO by setting the PAR_MODE and CCP_MODE fields in the camera core control register (CC_CRTL[3:1] and CC_CTRL[0]) and the DMA_EN field in the camera core control DMA register (CC_CTRL_DMA[8]).

Table 14−12. FIFO Write and Read Access

FIFO Write	FIFO Read	CCP_MODE	PAR_MODE	DMA_EN
Serial sensor	DMA	1	Don’t care	1
Parallel sensor	DMA	0	000, 001, 010, 100, 101	1
L4 I (CC_FIFODATA)	DMA	0	111	1
Serial sensor	MPU (CC_FIFODATA)	1	Don’t care	0
Parallel sensor	MPU (CC_FIFODATA)	0	000, 001, 010, 100, 101	0
L4 I (CC_FIFODATA)	MPU (CC_FIFODATA)	0	111	0

14.4.5 Clock Generation

The camera subsystem can generate the external clock (CAM.XCLK) that must be provided to the image sensor. CAM.XCLK is based on one input clock: CAM.MCLK.

The generated clock is not used internally, so the camera core clock generator can be disabled if the chip can provide the required clock to the sensor.

The clock divider configuration is programmable using the camera core external clock control register (CC_CTRL_XCLK), as shown in Table 14−13.

Table 14−13. Ratio of the XCLK Frequency Generator

Ratio	XCLK Based on CAM.MCLK (96-MHz max)
0	Stable low-level divider not enabled (default)
1	Stable high-level divider not enabled
2	48 MHz
3	32 MHz
4	24 MHz
5	19.2 MHz
6	16 MHz
7	13.7 MHz
8	12 MHz
9	10.67 MHz
10	9.6 MHz
11	8.7 MHz
12	8 MHz
…	…
30	3.2 MHz
31	Bypass (CAM.XCLK = CAM.MCLK)

To generate a glitch-free signal clock, the input clock must be glitch-free before enabling the clock divider generator. The duty cycle is 50 percent.

14.4.6 Interrupt Generation

The camera subsystem provides two active-low interrupt lines, as shown in Table 14−14.

Table 14−14. Interrupt Sources

Name	I/O[1]	Description
CAM_IRQ0	O	MPU interrupt line (initiated by three potentially different DMA interrupts, one MMU interrupt, or one camera core interrupt)
CAM_IRQ1	O	DSP interrupt line (from DMA4 interrupt line 4)—event can be programmed.

Notes: 1) I = Input, O = Output

When an interrupt is generated (the line is asserted low), the CAM_IRQSTATUS register must be read to know the source of the interrupt (MMU interrupt, DMA interrupt 0, DMA interrupt 1, DMA interrupt 2, or camera core interrupt).

For more information about DMA and MMU interrupts, see Chapter 9, *Memory Management Units*, and Chapter 10, *DMA*.

For the camera core interrupt, the CAM_IRQ0 line is asserted (active low) when one of the following events occurs:

- ❏ FIFO underflow occurs—notified by FIFO_UF_IRQ (CC_IRQSTATUS[0]).
- ❏ FIFO overflow occurs—notified by FIFO_OF_IRQ (CC_IRQSTATUS[1]).
- ❏ FIFO threshold is reached—notified by FIFO_THR_IRQ (CC_IRQSTATUS[2]).
- ❏ FIFO is full—notified by FIFO_FULL_IRQ (CC_IRQSTATUS[3]).
- ❏ FIFO is not empty—notified by FIFO_NOEMPTY_IRQ (CC_IRQSTATUS[4]).
- ❏ A shifted synchronization code is detected—notified by SSC_ERR_IRQ (CC_IRQSTATUS[8]).
- ❏ A false synchronization code is detected—notified by FSC_ERR_IRQ (CC_IRQSTATUS[9]).
- ❏ A frame width error occurs—notified by FW_ERR_IRQ (CC_IRASTATUS[10]).
- ❏ An FSP decoding error occurs, indicated by FSP_ERR_IRQ (CC_IRQSTATUS[11]).
- ❏ A frame end occurs—notified by FE_IRQ (CC_IRQSTATUS[16]).
- ❏ A line start occurs—notified by LS_IRQ (CC_IRQSTATUS[17]).
- ❏ A line end occurs—notified by LE_IRQ (CC_IRQSTATUS[18]).
- ❏ A frame start occurs—notified by FS_IRQ (CC_IRQSTATUS[19]).

The camera core CC_IRQSTATUS register must be read to determine the cause of the camera core interrupt.

Table 14−15 lists the camera core interrupts operational in the CCP, parallel No BT, and parallel BT modes.

Table 14−15. Operational Interrupts in CCP, Parallel No BT, and Parallel BT Modes

	CCP	**Parallel No BT**	**Parallel BT**
FIFO_UF_IRQ	Yes	Yes	Yes
FIFO_OF_IRQ	Yes	Yes	Yes
FIFO_THR_IRQ	Yes	Yes	Yes
FIFO_FULL	Yes	Yes	Yes
FIFO_NOEMPTY	Yes	Yes	Yes
SSC_ERR_IRQ	Yes	No	No
FSC_ERR_IRQ	Yes	No	Yes
FW_ERR	Yes	No	No
FSP_ERR	Yes	No	No
FE_IRQ	Yes	Yes	Yes
LS_IRQ	Yes	No	No
LE_IRQ	Yes	No	No
FS_IRQ	Yes	No	No

The camera core CC_IRQSTATUS register is not automatically reset when read—a 1 must be written to the corresponding bit to reset the interrupt.

Each event that generates an interrupt can be individually enabled using the camera core interrupt enable register (CC_IRQENABLE).

Even if an interrupt is masked, the corresponding status bit is flagged when the event occurs; however, this does not affect the interrupt line.

14.4.7 DMA Interface

The camera core interfaces with a DMA controller (CamDMA). At system level, the interface has the advantage of discharging the CPU or system DMA (sDMA) of the data transfers. The DMA used in the camera subsystem provides four channels.

The camera core generates two DMA transmission requests over the L3 bus slave interface:

- ❑ CC_DMAREQ_N[0] transfers packet size based on the FIFO threshold level (programmable through the register bank). The DMA request is generated and remains active until the DMA transfer is complete (based on the threshold value). The DMA request hardware line is asserted when the FIFO threshold is reached. The DMA request is deasserted when the DMA controller reads a number of 32-bit words equal to the FIFO threshold.
- ❑ CC_DMAREQ_N[1] is used when the frame to transfer is not a multiple of the FIFO threshold.

  In this case, when the first DMA transfer is complete (threshold by 32-bit words transferred), some data corresponding to the end of the frame remain in the FIFO.

  To avoid waiting for data from the next picture to fulfill the threshold condition (and thus generate CC_DMAREQ_N[0]), the camera core generates a second DMA request (CC_DMAREQ_N[1]) for the remaining data in the FIFO.

  With this DMA request, the remaining data is sent to the memory unit before the next picture arrives. This ensures 1-shot picture acquisition and allows the software to start algorithms as soon as possible.

These two DMA requests are not active at the same time. CC_DMAREQ_N[1] can be generated only when CC_DMAREQ_N[0] is not active, provided two clocks cycles separate the two DMA requests.

The DMA interface can be configured using the camera core control DMA register (CC_CTRL_DMA). The DMA_EN bit (CC_CTRL_DMA[8]) enables or disables the DMA request line.

The DMA1_DISABLE bit (CC_CTRL_DMA [9]) permits the first DMA request to be generated (based only on the threshold) but not the second DMA request. In this case, pictures can be received continuously, but then they must be handled by the software. This bit is intended for test and debugging, and must be left at its default value (0).

The FIFO_THRESHOLD bit field (CC_CTRL_DMA[6:0]) sets the threshold the FIFO can reach before the DMA request is generated.

For additional information about DMA, see Chapter 10, *DMA*.

14.4.8 Module Power Saving

The camera subsystem provides an autoidle function in its interconnect clock domain.

The interconnect clock autoidle power-saving mode is enabled or disabled using the AUTOIDLE bit (CAM_SYSCONFIG[0] bit).

When this mode is enabled and there is no activity on the interconnect interface, the interconnect clock (CAM_ICLK) is disabled in the module, thus reducing power consumption. When there is new activity on the interconnect interface, the interconnect clock is restarted without a latency penalty. After a reset, this mode is disabled by default.

To save power, it is recommended to enable this mode.

The camera core provides the same autoidle facility. It works exactly as the camera subsystem autoidle mode and can be enabled or disabled using the camera core CC_SYSCONFIG[0] bit. CAM_ICLK is disabled in the camera core.

Note:

The DMA and MMU also provide autoidle functionality, which is activated by setting the CamDMA DMA4_OCP_SYSCONFIG[0] and camera MMU MMU_SYSCONFIG[0] bits. For more information about these modules, see Chapter 9, *Memory Management Units*, and Chapter 10, *DMA*.

14.4.9 System Power Management

14.4.9.1 Camera Subsystem Idle Scheme

As part of the system-wide power-management scheme, the camera subsystem can go into idle state at the request of the PRCM module (for details, see Chapter 5, *Power, Reset, and Clock Management*).

Because the DMA is the master for the exchanges performed, idle requests are managed through the DMA module. Consequently, from the PRCM perspective, the camera subsystem is seen as an initiator module.

14.4.9.2 DMA Standby Scheme

As an initiator, the DMA supports the MSTANDBY/WAIT handshake protocol with the PRCM module, which rules camera subsystem behavior.

CAUTION

The camera core does not support smart-idle mode. Consequently, the DMA must not be configured in MSTANDBY mode (CamDMA DMA4_OCP_SYSCONFIG[13:12] set to 0x2). Otherwise, the clock can be cut while the camera core is performing an acquisition, and data can be lost.

The two available modes are force-standby and no-standby (CamDMA DMA4_OCP_SYSCONFIG[13:12] set to 0x0 or 0x1, respectively).

In force-standby mode, the MSTANDBY signal to the PRCM module (used to indicate that the initiator is ready to enter standby mode) is asserted only when all DMA channels are disabled. In no-standby mode, the MSTANDBY signal is never asserted. For more information about the MSTANDBY/WAIT protocol, see Chapter 5, *Power, Reset, and Clock Management*.

14.4.9.3 System Power-Management Scheme

To save power at the system level, the PRCM module can cut a clock when it is not required at the module level.

Setting the PRCM EN_CAM bits (CM_ICLKEN1_CORE[31] for CAM_ICLK and CAM_FCLK and CM_FCLKEN1_CORE[31] for CAM_MCLK) indicates that the camera subsystem does not need its clocks. The clocks are cut only if they are not needed by other modules that receive them.

The PRCM module also provides an AUTO_CAM bit (CM_AUTOIDLE1_CORE[31]) that allows choosing camera system behavior with respect to the behavior of the entire clock domain. Setting this bit to 1 allows CAM_ICLK (or CAM_FCLK) to be cut off automatically when the L4 clock domain (or L3 clock domain) is cut.

For details, see Chapter 5, *Power, Reset, and Clock Management*.

14.4.9.4 Reset Scheme

The camera subsystem provides a general software reset (all the modules—camera core, DMA, and MMU—are reset). It is activated by setting the SOFTRESET bit of the camera subsystem CAM_SYSCONFIG to 1 (CAM_SYSCONFIG[1]).

Besides this global module reset, the camera core also provides three types of reset:

- Reset of the camera core

  The camera core can be reset by setting the SOFTRESET bit of the camera core system configuration register (CC_SYSCONFIG[1]) to 1.

- Reset of the image-sensor interface

  The internal state-machines of the image-sensor interface are reset by setting the CC_EN bit of the camera core control register (CC_CTRL[16]) to 1.

- Reset of the FIFO and DMA control

  The internal state-machines of the FIFO and DMA control circuitry are reset by setting the CC_RST bit of the camera core control register (CC_CTRL[18]) to 1.

For more information about MMU and DMA, see Chapter 9, *Memory Management Units*, and Chapter 10, *DMA*.

14.5 Camera Subsystem Programming Model

14.5.1 Camera Subsystem Reset

The camera subsystem can accept a general software reset that is propagated through all submodules. This reset is useful to initialize the module or after a specific issue.

Here are the steps for a camera subsystem soft reset:

Step 1: Set the camera subsystem SOFTRESET bit (CAM_SYSCONFIG[1]) to 1.

Step 2: Read the camera subsystem CAM_SYSSTATUS register and check that the RESETDONE bit (CAM_SYSSTATUS[0]) is set to 1.

If after 10 consecutive reads the RESETDONE bit is still 0, an error occurred during the reset stage. The application can then generate an error and flag the initial error by reading the RESETDONE information available at the camera core (CC_SYSSTATUS[0]), DMA4, or MMU levels.

14.5.2 Enable Picture/Video Acquisition

Before starting any picture or video acquisition, check that the camera subsystem is not under reset.

Assuming that reset is complete, a configuration step is required before acquisition. The module must be initialized in the following order:

Step 1: Camera subsystem register configuration

Step 2: MMU register configuration

Step 3: DMA register configuration

Step 4: Camera core register configuration

In the following subsections, an image is identified by:

- PICTURE_WIDTH (number of pixels per line)
- PICTURE_HEIGHT (number of lines in a frame)
- PIXEL_TYPE (number of bytes per pixel)
- THRESHOLD_VALUE (amount of 32-bit data to transfer from the camera core to the DMA4 for each DMA request)

PICTURE_WIDTH and PICTURE_HEIGHT must be greater than 1; PIXEL_TYPE range is {0;2} (0 for 1 byte per pixel, 1 for 2 bytes per pixel, and 2 for 4 bytes per pixel), and THRESHOLD_VALUE range is {7;54} with a recommended value of 32.

14.5.2.1 Camera Subsystem Register Configuration

Set the AUTOIDLE bit (CAM_SYSCONFIG[0]) to 1 to enable the power-saving circuitry.

14.5.2.2 MMU Register Configuration

The MMU configuration can be bypassed if address remapping or endianism conversion is not required (MMU disabled). In this case, only step 1 (below) is required.

In general, perform the following steps to configure the MMU:

Step 1: Set the camera MMU AUTOIDLE bit (MMU_SYSCONFIG[0]) to 1 to enable power-saving circuitry.

Step 2: Configure MMU interrupt generation as required using the camera MMU MMU_IRQSTATUS and MMU_IRQENABLE registers.

Step 3: Set the camera MMUENABLE bit (MMU_CNTL[1]) to 1 to enable MMU functionality or set the MMUENABLE bit (MMU_CNTL[1]) to 0 to disable MMU functionality.

For more information about MMU functionalities, see Chapter 9, *Memory Management Units*.

14.5.2.3 DMA Register Configuration

> CAUTION
>
> **It is easy to confuse a *frame* in DMA terminology with a *camera frame.* A DMA frame refers to a single line rather than to an image.**
>
> **Nevertheless, the camera probably cannot hold one complete line in the camera core FIFO to buffer the data. Thus, each line is transferred in several subsequent transfers of THRESHOLD_VALUE size. When this size is reached, a DMA request is asserted.**
>
> **This applies to source-packed synchronized transfers. The DMA transfers a subset of a DMA frame per DMA request, not an entire DMA frame.**

For best performance (interface bandwidth, power consumption) and to ensure compatibility with the camera core module, the DMA module must be used as described in the following section, *General Configuration*.

General Configuration

Step 1: Set the CamDMA AUTOIDLE bit (DMA4_OCP_SYSCONFIG[0]) to 1 to enable the power-saving circuitry.

Step 2: Configure the CamDMA DMA4_OCP_SYSCONFIG register with the desired values, depending on system requirements (for more information, see Chapter 10, *DMA*).

CAUTION

The MIDLEMODE bit field (DMA4_OCP_SYSCONFIG[13:12]) in the CamDMA DMA4_OCP_SYSCONFIG register must never be set to smart-standby mode (DMA4_OCP_SYSCONFIG[13:12] = 0x2). As described in Section 14.4.9.2, *DMA Standby Scheme,* the camera core does not support smart-standby mode. Enabling this mode at the DMA level (which is an initiator in the camera subsystem) causes the clocks to be cut off even if the camera core is handling a data reception. Only force-idle and no-idle modes are possible. The software must ensure correct configuration of this bit field.

Step 3: Set the CamDMA DMA4_GCR[7:0] bit field to 0xFF (MAX_CHANNEL_FIFO_DEPTH bits).

0x10 is the default value after reset.

Channel 0 Programming

Step 1: Use the CamDMA DMA4_IRQENABLE_L0 register to enable interrupts.

Step 2: Use the CamDMA DMA4_CNLK_CTRL0 register to define channel chaining. It is possible to chain channels 0, 1, 2, and 3 with the same configuration and then enable channel chaining if necessary (for video capture, for example).

Step 3: Set the CamDMA DMA4_GCR [7:0] bit field to 0x10 (MAX_CHANNEL_FIFO_DEPTH bits).

Step 4: Write PICTURE_HEIGHT into the CamDMA DMA4_CFN0 register to provide the number of frames in the block (number of lines per picture).

Step 5: Configure the CamDMA DMA4_CSDP0 register as follows to obtain maximum bandwidth and compatibility with the camera core module:

a. Set the DATA_TYPE field (DMA4_CSDP0[1:0]) to PIXEL_TYPE parameter value.

b. Set the RD_ADD_TRSLT field (DMA4_CSDP0[5:2]) to 0000.

c. Set the SRC_PACKED bit (DMA4_CSDP0[6]) to 1.

d. Set the SRC_BURST_EN field (DMA4_CSDP0[8:7]) to 11 (eight words of 64 bits).

e. Set the WR_ADD_TRSLT field (DMA4_CSDP0[12:9]) to 0000.

f. Set the DST_PACKED bit (DMA4_CSDP0[13]) to 1 to provide a good data flow.

g. Set the DST_BURST_EN field (DMA4_CSDP0 [15:14]) to any functional value. (Value 11, which corresponds to 8 x 64 bits, is not recommended for the OMAP2420 because of system degradation that can occur when this type of transaction crosses the interconnect.)

h. Set the WRITE_MODE field (DMA4_CSDP0[17:16]) to any value; 01 is recommended for posted-write operation.

i. Set endianness register bits according to the required data processing (pixel switching). It is recommended that these bits be fixed to 0 and this functionality left to the MMU.

Step 6: Write 0x0 to the CamDMA DMA4_CSSA0 register.

Step 7: CamDMA DMA4_CDSA0 register is fixed to 0x0. The MMU then translates the destination address.

Step 8: Write 0x0 to the CamDMA DMA4_CSEI0 register.

Step 9: Write the THRESHOLD_VALUE to the CamDMA DMA4_CSFI0 register.

Step 10: The CamDMA DMA4_CDEI0 register is fixed to 0x0. The MMU then translates the destination address.

Step 11: The CamDMA DMA4_CDFI0 register is fixed to 0x0. The MMU then translates the destination address.

Step 12: Make a write access to the CamDMA DMA4_CCR0 register, setting the following bits:

a. SYNCHRO_CONTROL field (DMA4_CCR0[4:0]) to 00001

b. FS bit (DMA4_CCR0[5]) to 1

c. SRC_AMODE field (DMA4_CCR0[13:12]) to 01

d. DST_AMODE field (DMA4_CCR[15:14]) to 01

e. BS bit (DMA4_CCR0[18]) to 1

f. SEL_SRC_DST_SYNC bit (DMA4_CCR0[24]) to 1

g. BUFFERING_DISABLE field (DMA4_CCR[25]) to 0

All other bits must remain at default values.

Step 13: Make the same write access to DMA4_CCR0 as the previous one but with the enable bit set to 1 (DMA4_CCR0[7]).

14.5.2.4 Camera Core Register Configuration

The camera core module must be configured as follows:

Step 1: Set the camera core AUTOIDLE bit (CC_SYSCONFIG[0]) to 1 to enable the power-saving circuitry.

Step 2: Select the interrupt sources in the camera core interrupt enable register (CC_IRQENABLE) by setting the corresponding bits.

Step 3: Set the FIFO threshold for DMA requests in the camera core control DMA register by setting the FIFO_THRESHOLD field to THRESHOLD_VALUE – 1 (CC_CTRL_DMA[6:0]).

Step 4: Set the camera core DMA_EN bit (CC_CTRL_DMA[8]) to 1. You can also set the DMA1_DISABLE bit (CC_CTRL_DMA[9]) to 1, but the default value is recommended.

Step 5: If external clock generation is required, it is configured using the camera core CC_CTRL_XCLK register.

CAUTION

After the end of frame occurs for the JPEG programming model, you must clear the camera core interrupt, disable the DMA channel, reconfigure the DMA, and reenable the DMA.

14.5.2.5 Picture/Video Acquisition

CCP Mode

Step 1: Set the camera core CCP_MODE bit (CC_CTRL[0]) to 1 to enable the CCP mode.

By default, the CCP mode is enabled (the CCP_MODE bit set to 1).

Step 2: Set the four camera core registers CC_CCPFSCR, CC_CCPFECR, CC_CCPLSCR, and CC_CCPLECR to suitable values if default values are not used.

Step 3: Set the desired data format in the CCP data format register by setting the DATAFORMATSELECT field (CC_CCPDFR[3:0]). (See Table 14−11 for the corresponding codes.)

Step 4: Set the camera core CC_EN bit (CC_CTRL[16]) to 1 to enable the sensor interface.

It is recommended that you set the camera core CC_FRAME_TRIG bit (CC_CTRL[17]) to 1 when the CC_EN bit is set to 1 to start the acquisition.

Note:

For 1-shot picture acquisition, the sequence must be slightly modified; do not set the CC_EN bit (CC_CTRL[16]) to 0. Instead, set the CC_ONE_SHOT bit (CC_CTRL[20]) to 1.

Parallel Mode

Step 1: Set the camera core CCP_MODE bit (CC_CTRL[0]) to 0 to disable the CCP mode.

The CCP mode is the default mode. The parallel mode (either BT656 or no BT656) can be accessed only if the CCP mode is disabled.

Step 2: Set the desired configuration in the camera core control register by setting the PAR_MODE field (CC_CTRL[3:1]).

This bit field sets the protocol mode of the camera core to parallel mode. Set 000 for 8-bit No BT, 001 for 10-bit No BT, 100 for 8-bit BT, and 101 for 10-bit BT.

Setting this bit field to 111 enables the FIFO test mode.

Step 3: Set the camera core CC_EN bit CC_CTRL[16]) to 1 to enable the sensor interface.

It is recommended to set the camera core CC_FRAME_TRIG bit (CC_CTRL[17]) to 1 when the CC_EN bit is set to 1 to start the acquisition.

Note:

For 1-shot picture acquisition, the sequence must be slightly modified; do not set the CC_EN bit (CC_CTRL[16]) to 0. Instead, set the CC_ONE_SHOT bit (CC_CTRL[20]) to 1.

14.5.3 Disable Picture/Video Acquisition

Picture acquisition is disabled through the camera core.

Set the camera core CC_EN bit (CC_CTRL[16]) to 0 and set the CC_FRAME_TRIG bit (CC_CTRL[17]) to 1 to correctly disable the frame acquisition.

In this case, the DMA4 (CamDMA) and the camera core are ready for the next picture acquisition.

14.5.4 Interrupt Handling

When an interrupt request occurs (CAM_IRQ0 asserted low), the source is identified in the camera subsystem CAM_IRQSTATUS register.

Step 1: Read the CAM_IRQSTATUS register to identify the source of the interrupt.

If the CAM_IRQSTATUS[0], CAM_IRQSTATUS[1], or CAM_IRQSTATUS[2] bit is set to 1, the interrupt source is the DMA4 module. To handle the source, see Chapter 10, *DMA*.

If the CAM_IRQSTATUS[3] bit is set to 1, the interrupt source is the MMU module. To handle the source, see Chapter 9, *Memory Management Units*.

If the CAM_IRQSTATUS[4] bit is set to 1, the interrupt source is the camera core.

If the interrupt source is the camera core, follow Step 2 through Step 3 to handle the source.

Step 2: Read the camera core CC_IRQSTATUS register to identify the source.

Step 3: Clear the source by setting the related bit to 1 in the camera core CC_IRQSTATUS register.

If the interrupt source is a FIFO overflow (the CC_IRQSTATUS[1] bit is set to 1):

> **CAUTION**
>
> **If the camera core DMA_EN bit (CC_CTRL_DMA[8]) is set to 1, the CPU can terminate the DMA transfer immediately or let it run until no DMA request remains. Be sure the desired transfers are terminated before performing Step 4.**

Step 4: Set the camera core CC_FRAME_TRIG bit (CC_CTRL[17]) to 0 so the module can be disabled immediately.

Step 5: Set the camera core CC_EN bit (CC_CTRL[16]) to 0 to disable the sensor interface.

Step 6: Clear the interrupt source by setting the FIFO_OF_IRQ bit (CC_IRQSTATUS[1]) to 1.

Step 7: Reset the FIFO pointers and camera core internal state-machines by setting the CC_RST bit (CC_CTRL[18]) to 1.

Step 8: Set the camera core CC_EN bit (CC_CTRL[16]) to 1 to reenable data flow from the image sensor.

If the source is a FIFO underflow (the CC_IRQSTATUS[0] bit is set to 1):

Step 9: Set the CC_FRAME_TRIG bit (CC_CTRL[17]) to 0 so the module can be disabled immediately.

Step 10: Set the CC_EN bit (CC_CTRL[16]) to 0 to disable the sensor interface.

Step 11: Clear the interrupt source by setting the FIFO_UF_IRQ bit (CC_IRQSTATUS[0]) to 1.

Step 12: Reset the FIFO pointers and camera core internal state-machines by setting the CC_RST bit (CC_CTRL[18]) to 1.

Step 13: Set the CC_EN bit (CC_CTRL[16]) to 1 to reenable data flow from the image sensor.

14.6 Camera Subsystem Registers

Table 14−16. Instance Summary

Module Name	Base Address	Size	Description
CAMERA	0x4805 2000	1K byte	Camera subsystem
	0x4805 2400	1K byte	Camera core
	0x4805 2800	1K byte	Camera DMA
	0x4805 2C00	1K byte	Camera MMU

14.6.1 Camera Interface Register Mapping Summary

Table 14−17. Camera Subsystem Registers Address Offset

Register Name	Type	Register Width (Bits)	Offset
CAM_REVISION	R	32	0x00
CAM_SYSCONFIG	R/W	32	0x10
CAM_SYSSTATUS	R	32	0x14
CAM_IRQSTATUS	R	32	0x18
CAM_GPO	R/W	32	0x40
CAM_GPI	R	32	0x50

Table 14−18. Camera Core Registers Address Offset

Register Name	Type	Register Width (Bits)	Offset
CC_REVISION	R	32	0x00
CC_SYSCONFIG	R/W	32	0x10
CC_SYSSTATUS	R	32	0x14
CC_IRQSTATUS	R/W	32	0x18
CC_IRQENABLE	R/W	32	0x1C
CC_CTRL	R/W	32	0x40
CC_CTRL_DMA	R/W	32	0x44
CC_CTRL_XCLK	R/W	32	0x48
CC_FIFODATA	R/W	32	0x4C
CC_TEST	R	32	0x50
CC_GENPAR	R	32	0x54
CC_CCPFSCR	R/W	32	0x58
CC_CCPFECR	R/W	32	0x5C
CC_CCPLSCR	R/W	32	0x60

Table 14−18. Camera Core Registers Address Offset (Continued)

Register Name	Type	Register Width (Bits)	Offset
CC_CCPLECR	R/W	32	0x64
CC_CCPDFR	R/W	32	0x68

Table 14−19. DMA Registers Address Offset

Register Name	Type	Register Width (Bits)	Offset
DMA4_REVISION	R	32	0x00
DMA4_IRQSTATUS_L0	R/W	32	0x08
DMA4_IRQSTATUS_L1	R/W	32	0x0C
DMA4_IRQSTATUS_L2	R/W	32	0x10
DMA4_IRQSTATUS_L3	R/W	32	0x14
DMA4_IRQENABLE_L0	R/W	32	0x18
DMA4_IRQENABLE_L1	R/W	32	0x1C
DMA4_IRQENABLE_L2	R/W	32	0x20
DMA4_IRQENABLE_L3	R/W	32	0x24
DMA4_SYSSTATUS	R	32	0x28
DMA4_OCP_SYSCONFIG	R/W	32	0x2C
DMA4_CAPS_0	R	32	0x64
DMA4_CAPS_2	R	32	0x6C
DMA4_CAPS_3	R	32	0x70
DMA4_CAPS_4	R	32	0x74
DMA4_GCR	R/W	32	0x78
DMA4_CCR	R/W	32	0x80 + n*0x60
DMA4_CLNK_CTRL	R/W	32	0x84 + n*0x60
DMA4_CICR	R/W	32	0x88 + n*0x60
DMA4_CSR	R/W	32	0x8C + n*0x60
DMA4_CSDP	R/W	32	0x90 + n*0x60
DMA4_CEN	R/W	32	0x94 + n*0x60
DMA4_CFN	R/W	32	0x98 + n*0x60
DMA4_CSSA	R/W	32	0x9C + n*0x60
DMA4_CDSA	R/W	32	0xA0 + n*0x60
DMA4_CSEI	R/W	32	0xA4 + n*0x60

Table 14−19. DMA Registers Address Offset (Continued)

Register Name	Type	Register Width (Bits)	Offset
DMA4_CSFI	R/W	32	0xA8 + n*0x60
DMA4_CDEI	R/W	32	0xAC + n*0x60
DMA4_CDFI	R/W	32	0xB0 + n*0x60
DMA4_CSAC	R	32	0xB4 + n*0x60
DMA4_CDAC	R	32	0xB8 + n*0x60
DMA4_CCEN	R	32	0xBC + n*0x60
DMA4_CCFN	R	32	0xC0 + n*0x60
DMA4_COLOR	R/W	32	0xC4 + n*0x60

Table 14−20. MMU Registers Address Offset

Register Name	Type	Register Width (Bits)	Offset
MMU_REVISION	R	32	0x00
MMU_SYSCONFIG	R/W	32	0x10
MMU_SYSSTATUS	R	32	0x14
MMU_IRQSTATUS	R/W	32	0x18
MMU_IRQENABLE	R/W	32	0x1C
MMU_WALKING_ST	R/W	32	0x40
MMU_CNTL	R/W	32	0x44
MMU_FAULT_AD	R	32	0x48
MMU_TTB	R/W	32	0x4C
MMU_LOCK	R/W	32	0x50
MMU_LD_TLB	R/W	32	0x54
MMU_CAM	R/W	32	0x58
MMU_RAM	R/W	32	0x5C
MMU_GFLUSH	R/W	32	0x60
MMU_FLUSH_ENTRY	R/W	32	0x64
MMU_READ_CAM	R	32	0x68
MMU_READ_RAM	R	32	0x6C
MMU_EMU_FAULT_AD	R	32	0x70

Section 14.6.2 and Section 14.6.3 describe only the registers related to the camera subsystem and the camera core, respectively. For DMA and MMU register descriptions, see Chapter 9, *Memory Management Units*, and Chapter 10, *DMA*.

14.6.2 Camera Top Register Descriptions

Table 14−21. CAM_REVISION

Address Offset	0x0000		
Physical Address	0x4805 2000	**Instance**	CAM1
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 Examples: 0x10 for 1.0, 0x21 for 2.1	R	

Table 14−22. CAM_SYSCONFIG

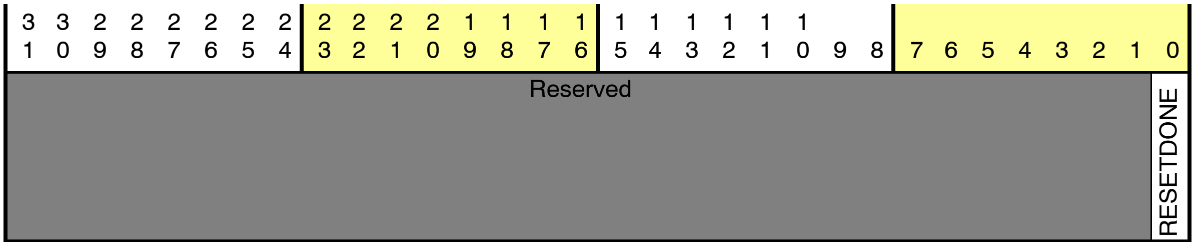
Address Offset	0x0010		
Physical Address	0x4805 2010	**Instance**	CAM1
Description	Camera subsystem configuration register		
Type	RW		

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																														SOFTRESET	AUTOIDLE

Bits	Field Name	Description	Type	Reset
31:2	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000000
1	SOFTRESET	Software reset. Set this bit to trigger a module reset. The bit is automatically reset by the hardware. Read returns 0. 0x0: Normal mode 0x1: The module is reset.	RW	0
0	AUTOIDLE	Enables power-management capability 0x0: Free-running clocks 0x1: Clocks are internally gated by the functionality.	RW	0

Table 14−23. CAM_SYSSTATUS

Address Offset	0x0014		
Physical Address	0x4805 2014	**Instance**	CAM1
Description	This register provides status information about the module.		
Type	R		



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 ongoing.		
		0x1: Reset complete[1]		

Notes: 1) During reset, the value is 0, but the value read just after the reset is 1, because ICLK/FCLK is already operating.

Table 14−24. CAM_IRQSTATUS

Address Offset	0x0018		
Physical Address	0x4805 2018	**Instance**	CAM1
Description	This register provides interrupt line status.		
Type	R		

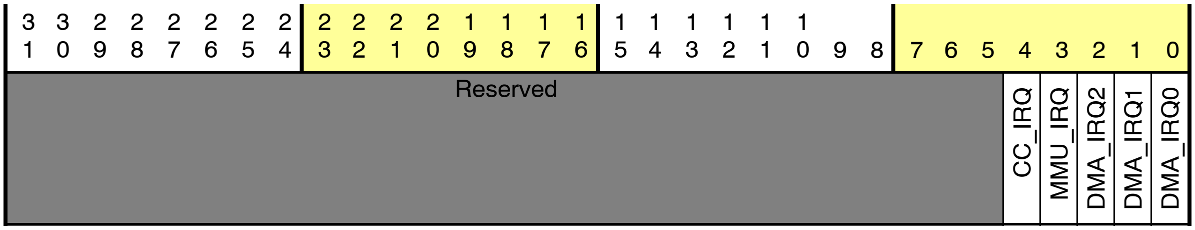


Table 14−24. CAM_IRQSTATUS (Continued)

Bits	Field Name	Description	Type	Reset
31:5	Reserved	Reserved for module-specific number of IRQ lines. Read returns 0.	R	0x0000000
4	CC_IRQ	Camera core interrupt status 0x0: Camera core interrupt inactive 0x1: Camera core interrupt active	R	0
3	MMU_IRQ	MMU interrupt status 0x0: MMU interrupt inactive 0x1: MMU interrupt active	R	0
2	DMA_IRQ2	DMA interrupt 2 status 0x0: DMA interrupt 2 inactive 0x1: DMA interrupt 2 active	R	0
1	DMA_IRQ1	DMA interrupt 1 status 0x0: DMA interrupt 1 inactive 0x1: DMA interrupt 1 active	R	0
0	DMA_IRQ0	DMA interrupt 0 status 0x0: DMA interrupt 0 inactive 0x1: DMA interrupt 0 active	R	0

Table 14−25. CAM_GPO

Address Offset	0x0040		
Physical Address	0x4805 2040	**Instance**	CAM1
Description	This register provides general-purpose output control. This register must be used for test or debug only. That does not make sense to use it during the functional activity.		
Type	RW		

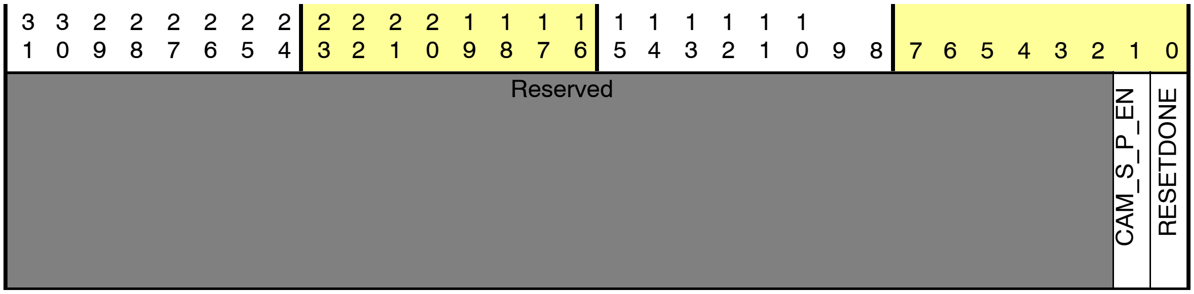


Table 14−25. CAM_GPO (Continued)

Bits	Field Name	Description	Type	Reset
31:2	Reserved	Read returns 0s, writes are ignored.	RW	0
1	CAM_S_P_EN	Camera serial/parallel protocol spy enable 0x0: CAM_S and CAM_P fields of CAM_GPI are forced to 0. 0x1: The values of CAM_S and CAM_P reflect the current status of the protocol lines.	RW	0
0	CAM_CCP_ MODE	CAM_CCP_MODE pin control 0x0: Camera buffers connected to balls V6, V4, Y2 and W3 operate in CMOS mode (suitable mode for parallel camera interface). 0x1: Camera buffers connected to balls V6, V4, Y2 and W3 operate in SubLVDS mode (suitable mode for CCP interface).	RW	0

Table 14−26. CAM_GPI

Address Offset	0x0050		
Physical Address	0x4805 2050	**Instance**	CAM1
Description	This register provides general-purpose input for internal signal observability. This register must be used for test or debug only. That does not make sense to use it during the functional activity.		
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	CC_DMA_REQ1	CC_DMA_REQ0	Reserved	CAM_MSTANDBY	CAM_WAIT	Reserved	CAM_S_DATA	CAM_S_CLK	Reserved	CAM_P_DATA	CAM_P_VS	CAM_P_HS	CAM_P_CLK

Bits	Field Name	Description	Type	Reset
31:25	Reserved	Reserved for module−specific number of GPO lines. Reads return 0	R	0x−−
24	CC_DMA_REQ1	Bits for supporting GPI capability False_r 0 Debug: Internal signal to 0 True_r 1 Debug: Internal signal to 1	R	0x−
23	CC_DMA_REQ0	DMA request 0 level False_r 0 Debug: Internal signal to 0 True_r 1 Debug: Internal signal to 1	R	0x−
22	Reserved		R	0x−

Table 14−26. CAM_GPI (Continued)

Bits	Field Name	Description	Type	Reset
21	CAM_MSTANDBY	Mstandby signal for Power Management False_r 0 Debug: Internal signal to 0 True_r 1 Debug: Internal signal to 1	R	0x−
20	CAM_WAIT	Wait Signal for Power Management False_r 0 Debug: Internal signal to 0 True_r 1 Debug: Internal signal to 1	R	0x−
19:18	Reserved		R	0x−
17	CAM_S_DATA	Seial Data(CCP) False_r 0 Debug: Internal signal to 0 True_r 1 Debug: Internal signal to 1	R	0x−
16	CAM_S_CLK	Serial Clock(CCP) False_r 0 Debug: Internal signal to 0 True_r 1 Debug: Internal signal to 1	R	0x−
15	Reserved		R	0x−
14:3	CAM_P_DATA	Parallel Data False_r 0 Debug: Internal signal to 0 True_r 1 Debug: Internal signal to 1	R	0x−−−
2	CAM_P_VS	Parallel Vertical Synchro(BT/ no BT) False_r 0 Debug: Internal signal to 0 True_r 1 Debug: Internal signal to 1	R	0x−
1	CAM_P_HS	Parallel Horizontal Synchro(BT/ no BT) False_r 0 Debug: Internal signal to 0 True_r 1 Debug: Internal signal to 1	R	0x−
0	CAM_P_CLK	Parallel Clock(BT/no BT) False_r 0 Debug: Internal signal to 0 True_r 1 Debug: Internal signal to 1	R	0x−

14.6.3 Camera Core Register Descriptions

Table 14–27. CC_REVISION

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								CC_REV_NB							

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Read returns 0.	R	0x000000
7:0	CC_REV_NB	This field contains the IP revision code.	R	

Table 14–28. CC_SYSCONFIG

Address Offset	0x0010		
Physical Address	0x4805 2410	**Instance**	CAMC1
Description	This register controls the interface parameters (CCP and parallel 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																											SIDLEMODE		RESERVED	SOFTRESET	AUTOIDLE

Bits	Field Name	Description	Type	Reset
31:5	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000000
4:3	SIDLEMODE	Slave interface power management req/ack control 0x0: An idle request is acknowledged unconditionally. 0x1: No idle. An idle request is never acknowledged. 0x2: Smart-idle mode not implemented 0x3: Do not use.	RW	0x0
2	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
1	SOFTRESET	Software reset. Set this bit to trigger a module reset. The bit is automatically reset by the hardware. Read 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 interface-gating strategy is applied based on OCP interface activity.	RW	0

Table 14−29. CC_SYSSTATUS

Address Offset	0x0014		
Physical Address	0x4805 2414	**Instance**	CAMC1
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	Reserved for module-specific status information. Read returns 0s.	R	0x000000
7:1	Reserved	Reserved for OCP socket status information. Read returns 0s.	R	0x00
0	RESETDONE	Internal reset monitoring 0x0: Internal module reset is ongoing. 0x1: Reset complete[1]	R	0

Notes: 1) During reset, the value is 0, but the value read just after the reset is 1, because ICLK/FCLK is already operating.

Table 14−30. CC_IRQSTATUS

Address Offset	0x0018		
Physical Address	0x4805 2418	**Instance**	CAMC1
Description	The interrupt status register regroups the statuses of module internal events that can generate interrupts (CCP and parallel 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												FS_IRQ	LE_IRQ	LS_IRQ	FE_IRQ	Reserved				FSP_ERR_IRQ	FW_ERR_IRQ	FSC_ERR_IRQ	SSC_ERR_IRQ	RESERVED			FIFO_NOEMPTY_IRQ	FIFO_FULL_IRQ	FIFO_THR_IRQ	FIFO_OF_IRQ	FIFO_UF_IRQ

Table 14–30. CC_IRQSTATUS (Continued)

Bits	Field Name	Description		Type	Reset
31:20	Reserved	Write 0s for future compatibility. Read returns 0.		RW	0x000
19	FS_IRQ	Frame start occurred.		RW	0
		Read 0x0:	Event false		
		Write 0x0:	Event status bit unchanged		
		Read 0x1:	Event is true (pending).		
		Write 0x1:	Event status bit is reset.		
18	LE_IRQ	Line end occurred.		RW	0
		Read 0x0:	Event false		
		Write 0x0:	Event status bit unchanged		
		Write 0x1:	Event status bit is reset.		
		Read 0x1:	Event is true (pending).		
17	LS_IRQ	Line start occurred.		RW	0
		Read 0x0:	Event false		
		Write 0x0:	Event status bit unchanged		
		Read 0x1:	Event is true (pending).		
		Write 0x1:	Event status bit is reset.		
16	FE_IRQ	Frame end occurred.		RW	0
		Read 0x0:	Event false		
		Write 0x0:	Event status bit unchanged		
		Read 0x1:	Event is true (pending).		
		Write 0x1:	Event status bit is reset.		
15:12	Reserved	Write 0s for future compatibility. Read returns 0.		RW	0x0
11	FSP_ERR_IRQ	FSP code error		RW	0
		Read 0x0:	Event false		
		Write 0x0:	Event status bit unchanged		
		Read 0x1:	Event is true (pending).		
		Write 0x1:	Event status bit is reset.		
10	FW_ERR_IRQ	Frame height error		RW	0
		Read 0x0:	Event false		
		Write 0x0:	Event status bit unchanged		
		Read 0x1:	Event is true (pending).		
		Write 0x1:	Event status bit is reset.		

Table 14–30. CC_IRQSTATUS (Continued)

Bits	Field Name	Description		Type	Reset
9	FSC_ERR_IRQ	False synchronization code		RW	0
		Read 0x0:	Event false		
		Write 0x0:	Event status bit unchanged		
		Read 0x1:	Event is true (pending).		
		Write 0x1:	Event status bit is reset.		
8	SSC_ERR_IRQ	Shifted synchronization code		RW	0
		Read 0x0:	Event false		
		Write 0x0:	Event status bit unchanged		
		Read 0x1:	Event is true (pending).		
		Write 0x1:	Event status bit is reset.		
7:5	Reserved	Write 0s for future compatibility. Read returns 0.		RW	0x0
4	FIFO_NOEMPTY_ IRQ	FIFO is not empty.		RW	0
		Read 0x0:	Event false		
		Write 0x0:	Event status bit unchanged		
		Read 0x1:	Event is true (pending).		
		Write 0x1:	Event status bit is reset.		
3	FIFO_FULL_IRQ	FIFO is full.		RW	0
		Read 0x0:	Event false		
		Write 0x0:	Event status bit unchanged		
		Read 0x1:	Event is true (pending).		
		Write 0x1:	Event status bit is reset.		
2	FIFO_THR_IRQ	FIFO threshold was reached.		RW	0
		Read 0x0:	Event false		
		Write 0x0:	Event status bit unchanged		
		Read 0x1:	Event is true (pending).		
		Write 0x1:	Event status bit is reset.		
1	FIFO_OF_IRQ	FIFO overflow occurred.		RW	0
		Read 0x0:	Event false		
		Write 0x0:	Event status bit unchanged		
		Read 0x1:	Event is true (pending).		
		Write 0x1:	Event status bit is reset.		

Table 14–30. CC_IRQSTATUS (Continued)

Bits	Field Name	Description		Type	Reset
0	FIFO_UF_IRQ	FIFO underflow occurred.		RW	0
		Read 0x0:	Event false		
		Write 0x0:	Event status bit unchanged		
		Read 0x1:	Event is true (pending).		
		Write 0x1:	Event status bit is reset.		

Table 14–31. CC_IRQENABLE

Address Offset	0x001C		
Physical Address	0x4805 241C	**Instance**	CAMC1
Description	The interrupt enable register allows enabling/disabling of the module internal sources of interrupt on an event-by-event basis (CCP and parallel 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												FS_IRQ_EN	LE_IRQ_EN	LS_IRQ_EN	FE_IRQ_EN	Reserved				FSP_ERR_IRQ_EN	FW_ERR_IRQ_EN	FSC_ERR_IRQ_EN	SSC_ERR_IRQ_EN	RESERVED			FIFO_NOEMPTY_IRQ_EN	FIFO_FULL_IRQ_EN	FIFO_THR_IRQ_EN	FIFO_OF_IRQ_EN	FIFO_UF_IRQ_EN

Bits	Field Name	Description		Type	Reset
31:20	Reserved	Write 0s for future compatibility. Read returns 0.		RW	0x000
19	FS_IRQ_EN	Frame start interrupt enable		RW	0
		0x0:	Event is masked.		
		0x1:	Event generates an interrupt when it occurs.		
18	LE_IRQ_EN	Line end interrupt enable		RW	0
		0x0:	Event is masked.		
		0x1:	Event generates an interrupt when it occurs.		
17	LS_IRQ_EN	Line start interrupt enable		RW	0
		0x0:	Event is masked.		
		0x1:	Event generates an interrupt when it occurs.		

Table 14–31. CC_IRQENABLE (Continued)

Bits	Field Name	Description		Type	Reset
16	FE_IRQ_EN	Frame end interrupt enable		RW	0
		0x0:	Event is masked.		
		0x1:	Event generates an interrupt when it occurs.		
15:12	Reserved	Write 0s for future compatibility. Read returns 0.		RW	0x0
11	FSP_ERR_ IRQ_EN	FSP code error		RW	0
		0x0:	Event is masked.		
		0x1:	Event generates an interrupt when it occurs.		
10	FW_ERR_ IRQ_EN	Frame height error interrupt enable		RW	0
		0x0:	Event is masked.		
		0x1:	Event generates an interrupt when it occurs.		
9	FSC_ERR_ IRQ_EN	False synchronization code interrupt enable		RW	0
		0x0:	Event is masked.		
		0x1:	Event generates an interrupt when it occurs.		
8	SSC_ERR_ IRQ_EN	False synchronization code interrupt enable		RW	0
		0x0:	Event is masked.		
		0x1:	Event generates an interrupt when it occurs.		
7:5	Reserved	Write 0s for future compatibility. Read returns 0.		RW	0x0
4	FIFO_ NOEMPTY_ IRQ_EN	FIFO threshold interrupt enable		RW	0
		0x0:	Event is masked.		
		0x1:	Event generates an interrupt when it occurs.		
3	FIFO_FULL_ IRQ_EN	FIFO threshold interrupt enable		RW	0
		0x0:	Event is masked.		
		0x1:	Event generates an interrupt when it occurs.		

Table 14–31. CC_IRQENABLE (Continued)

Bits	Field Name	Description	Type	Reset
2	FIFO_THR_ IRQ_EN	FIFO threshold interrupt enable 0x0: Event is masked. 0x1: Event generates an interrupt when it occurs.	RW	0
1	FIFO_OF_ IRQ_EN	FIFO overflow interrupt enable 0x0: Event is masked. 0x1: Event generates an interrupt when it occurs.	RW	0
0	FIFO_UF_ IRQ_EN	FIFO underflow interrupt enable 0x0: Event is masked. 0x1: Event generates an interrupt when it occurs.	RW	0

Table 14–32. CC_CTRL

Address Offset	0x0040		
Physical Address	0x4805 2440	**Instance**	CAMC1
Description	This register controls the DMA interface of the camera core block (CCP and parallel mode).		
Type	RW		

31–21	20	19	18	17	16	15–14	13	12	11	10	9	8	7–4	3–1	0
Reserved	CC_ONE_SHOT	Resrved	CC_RST	CC_FRAME_TRIG	CC_EN	Reserved	NOBT_SYNCHRO	BT_CORRECT	PAR_ORDERCAM	PAR_CLK_POL	NOBT_HS_POL	NOBT_VS_POL	Reserved	PAR_MODE	CCP_MODE

Bits	Field Name	Description	Type	Reset
31:21	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x000
20	CC_ONE_SHOT	One-shot capture—automatic Write 0x0: Normal mode Read 0x0: Read returns 0. Read 0x1: Undefined Write 0x1: Start 1-shot capture automatically.	RW/ dual	0

Table 14–32. CC_CTRL (Continued)

Bits	Field Name	Description	Type	Reset
19	Reserved	Read returns 0.	RW	0
18	CC_RST	Resets all he internal finite state-machines of the camera core module by writing 1 to this bit. Must be applied when CC_EN = 0. Read returns 0. 0x0: Normal mode 0x1: Resets internal camera core FSM	RW	0
17	CC_FRAME_ TRIG	Sets the modality in which CC_EN works when the sensor camera core is to be disabled 0x0: Impacts CC_EN functionality when CC_EN is cleared. Frame acquisition ends immediately. 0x1: Impacts CC_EN functionality when CC_EN is cleared. Frame acquisition ends when current frame is fully received.	RW	0
16	CC_EN	Enables the sensor interface of the camera core module 0x0: Module disabled (at the end of the frame if CC_FRAM_TRIG = 1 and immediately disabled of 0) 0x1: Module enabled	RW	0
15:14	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
13	NOBT_ SYNCHRO	Enables starting or not at the beginning of the frame in NOBT 0x0: Acquisition starts when vertical synchronization is high 0x1: Acquisition starts when vertical synchronization goes low to high (beginning of the frame)—recommended.	RW	0
12	BT_CORRECT	Enables correction within the synchronization codes in BT mode 0x0: Correction is not enabled. 0x1: Correction is enabled.	RW	1
11	PAR_ ORDERCAM	Enables swap between image and data in parallel mode 0x0: Swap is not enabled. 0x1: Swap is enabled.	RW	0

Table 14–32. CC_CTRL (Continued)

Bits	Field Name	Description	Type	Reset
10	PAR_CLK_POL	Inverts the clock coming from the sensor in parallel mode 0x0: Clock not inverted—data sampled on rising edge 0x1: Clock inverted—data sampled on falling edge	RW	0
9	NOBT_HS_POL	Sets the polarity of the synchronization signals in NOBT parallel mode 0x0: CAM_P_HS is active high. 0x1: CAM_P_HS is active low.	RW	0
8	NOBT_VS_POL	Sets the polarity of the synchronization signals in NOBT parallel mode 0x0: CAM_P_VS is active high. 0x1: CAM_P_VS is active low.	RW	0
7:4	Reserved	Write 0s for future compatibility Read returns 0.	RW	0x0
3:1	PAR_MODE	Sets the protocol mode of the camera core module in parallel mode (when CCP_MODE = 0) 0x0: Parallel NOBT 8-bit 0x1: Parallel NOBT 10-bit 0x2: Parallel NOBT 12-bit 0x3: Reserved 0x4: Parallel BT 8-bit 0x5: Parallel BT 10-bit 0x6: Reserved 0x7: FIFO test mode	RW	0x0
0	CCP_MODE	Sets the camera core in CCP mode. Read returns 0. 0x0: CCP mode not enabled 0x1: CCP mode enabled	RW	1

Table 14–33. CC_CTRL_DMA

Address Offset	0x0044		
Physical Address	0x4805 2444	**Instance**	CAMC1
Description	This register shows the status of some important variables of the camera core module (CCP and parallel 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																						DMA1_DISABLE	DMA_EN	Reserved	FIFO_THRESHOLD						

Bits	Field Name	Description	Type	Reset
31:10	Reserved	Write 0s for future compatibility Read returns 0.	RW	0x000000
9	DMA1_DISABLE	DMA1 disable feature. Use only DMA0 (threshold-based). This bit is only for testing and debugging, and must be set to 0 for correct camera DMA operation. 0x0: DMA1 interface disabled. The DMA request line stays inactive. 0x1: DMA1 interface enabled. The DMA request line is operational.	RW	0
8	DMA_EN	Sets the number of DMA request lines 0x0: DMA interface disabled. The DMA request line stays inactive. 0x1: DMA interface enabled. The DMA request line is operational.	RW	1
7	Reserved	Write 0 for future compatibility. Read returns 0.	RW	0
6:0	FIFO_ THRESHOLD	Sets the threshold of the FIFO. The assertion of the DMA request line occurs when the threshold is reached. **Note:** Only even FIFO thresholds are supported (that is, the FIFO_THRESHOLD value must be an odd number). Only threshold values equal to or below 64 are valid. 0000001: Threshold is set to 2. 0000011: Threshold is set to 4. 0111101: Threshold is set to 6. ... 0111111: Threshold is set to 64.	RW	0x07

Table 14–34. CC_CTRL_XCLK

Address Offset	0x0048		
Physical Address	0x4805 2448	**Instance**	CAMC1
Description	This register controls the value of the clock divisor used to generate the external clock (parallel 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																											XCLK_DIV				

Bits	Field Name	Description	Type	Reset
31:5	Reserved	Write 0s for future compatibility Read returns 0.	RW	0x0000000
4:0	XCLK_DIV	Sets the clock divisor value for CAM_XCLK generation based on CAM_MCK (value of CAM_MCLK is 96 MHz). Other values: Reserved 0x0: CAM_XCLK stable low-level divider not enabled 0x1: CAM_XCLK stable high-level divider not enabled 0x1F: CAM_XCLK = CAM_MCLK	RW	0x00

Table 14–35. CC_FIFODATA

Address Offset	0x004C		
Physical Address	0x4805 244C	**Instance**	CAMC1
Description	This register allows writing to the FIFO and reading from the FIFO (CCP and parallel 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
FIFO_DATA																															

Bits	Field Name	Description	Type	Reset
31:0	FIFO_DATA	Writes the 32-bit word into the FIFO	RW	0x00000000

Table 14–36. CC_TEST

Address Offset	0x0050		
Physical Address	0x4805 2450	**Instance**	CAMC1
Description	This register shows the status of some important variables of the camera core module (CCP and parallel 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
FIFO_RD_POINTER								FIFO_WR_POINTER								FIFO_LEVEL								FIFO_LEVEL_PEAK							

Bits	Field Name	Description	Type	Reset
31:24	FIFO_RD_ POINTER	FIFO READ pointer value. Expected value ranges from 0 to 127.	R	0x00
23:16	FIFO_WR_ POINTER	FIFO WRITE pointer value. Expected value ranges from 0 to 127.	R	0x00
15:8	FIFO_LEVEL	FIFO level (number of 32-bit words the FIFO contains). Can assume values from 0 to 128.	R	0x00
7:0	FIFO_LEVEL_ PEAK	FIFO level peak. This field shows the maximum value of the FIFO level and can assume values from 0 to 128.	R	0x00

Table 14–37. CC_GENPAR

Address Offset	0x0054		
Physical Address	0x4805 2454	**Instance**	CAMC1
Description	This register shows the values of the generic parameters (CCP and parallel 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																													FIFO_ DEPTH		

Bits	Field Name	Description	Type	Reset
31:3	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000000
2:0	FIFO_DEPTH	FIFO DEPTH generic parameter	R	0x6

Table 14−38. CC_CCPFSCR

Address Offset	0x0058		
Physical Address	0x4805 2458	**Instance**	CAMC1
Description	This register contains the frame start code (CCP 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
CCPFSCR																															

Bits	Field Name	Description	Type	Reset
31:0	CCPFSCR	Frame start code register	RW	0xFF000002

Table 14−39. CC_CCPFECR

Address Offset	0x005C		
Physical Address	0x4805 245C	**Instance**	CAMC1
Description	This register contains the frame end code (CCP 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
CCPFECR																															

Bits	Field Name	Description	Type	Reset
31:0	CCPFECR	Frame end code register	RW	0xFF000003

Table 14−40. CC_CCPLSCR

Address Offset	0x0060		
Physical Address	0x4805 2460	**Instance**	CAMC1
Description	This register contains the line start code (CCP 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
CCPLSCR																															

Bits	Field Name	Description	Type	Reset
31:0	CCPLSCR	Line start code register	RW	0xFF000000

Table 14−41. CC_CCPLECR

Address Offset	0x0064		
Physical Address	0x4805 2464	**Instance**	CAMC1
Description	This register contains the line-end code (CCP mode).		
Type	RW		

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="32">CCPLECR</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
31:0	CCPLECR	Line end code register	RW	0xFF000001

Table 14−42. CC_CCPDFR

Address Offset	0x0068		
Physical Address	0x4805 2468	**Instance**	CAMC1
Description	This register sets the data format and controls the receive state-machine (CCP mode).		
Type	RW		

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">Reserved</td><td colspan="8">ALPHA</td><td colspan="4">Reserved</td><td colspan="4">DATAFORMATSELECT</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000
15:8	ALPHA	Alpha parameter for RGB888 and RGB444	RW	0x00
7:4	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
3:0	DATAFORMAT-SELECT	Sets the data format in CCP mode 0x0: YUV422 big-endian 0x1: YUV422 0x2: YUV420 0x3: Reserved 0x4: RGB444 0x5: RGB565 0x6: RGB888 (no data expansion) 0x7: RGB888 (data expansion)	RW	0x0

Table 14−42. CC_CCPDFR (Continued)

Bits	Field Name	Description		Type	Reset
3:0 cont'd	DATAFORMAT-SELECT cont'd	0x8:	RAW8 (no data expansion)		
		0x9:	RAW8 (data expansion)		
		0xA:	RAW10 (no data expansion)		
		0xB:	RAW10 (data expansion)		
		0xC:	RAW12 (no data expansion)		
		0xD:	RAW12 (data expansion)		
		0xE:	JPEG8 FSP		
		0xF:	JPEG8		

Chapter 15

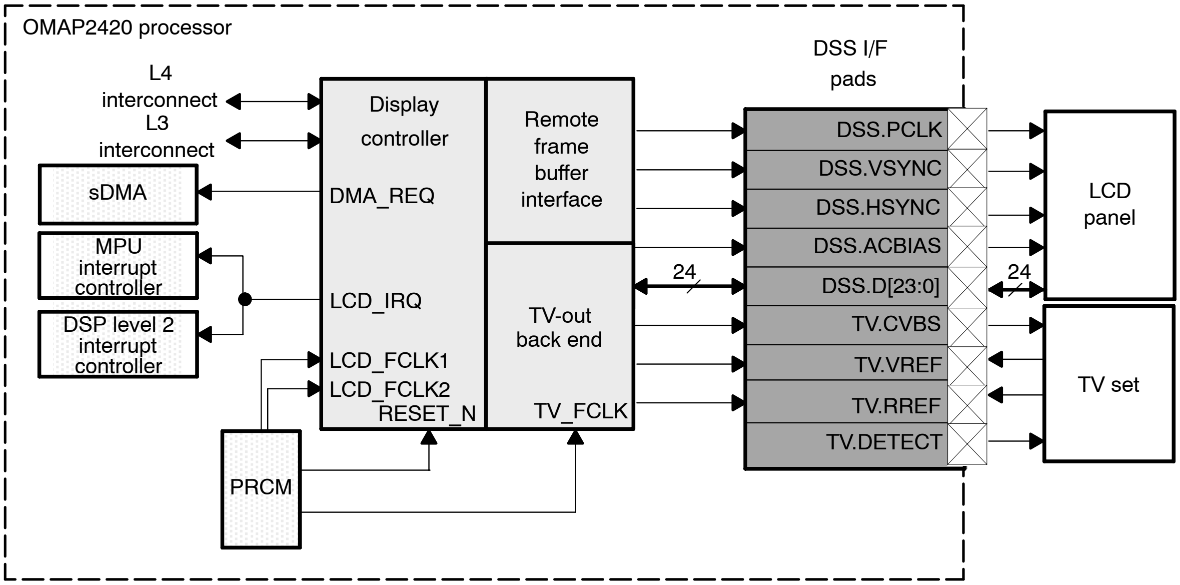
Display Subsystem

This chapter describes the display subsystem (DSS) of the OMAP2420 multimedia device.

15.1 Display Subsystem Overview

The display subsystem (DSS) provides the logic to display a video frame from external (SDRAM) or internal (SRAM) memory on a liquid crystal display (LCD) panel or on a TV set. The DSS integrates a display controller, a remote frame buffer (RFB) module, and a TV-out module. The display controller is connected to the L3 and L4 interconnects and has its own direct memory access (DMA) to read data from the system memory. The RFB and TV-out backends are connected to the L4 interconnect.

Figure 15−1. Display Subsystem Highlighted



15.1.1 Main Features

This section describes the main features of the DSS.

Display Controller

- ❑ Display modes
  - Programmable pixel display modes (1, 2, 4, 8, 12, 16, and 24 bits-per-pixel modes)
  - Programmable display size (up to 2048 [lines] x 2048 [pixels])
  - 256 x 24-bit entry palette in red, green, blue (RGB)
  - Programmable pixel rate

Note:

The panel size is programmable and the width can be any multiple of 8 pixels (line length), in the range [1:2048] pixels (in the case of the remote frame buffer interface (RFBI) mode, the minimum transfer size is 1 byte). The maximum resolution is 2048 (lines) x 2048 (pixels).

- Display support
  - Four types of displays are supported: Passive and active colors and-passive and active monochromes.
  - 4-/8-bit monochrome super twisted nematic (STN) panel interface support (15 grayscale levels supported using dithering block)
  - 8-bit color STN panel interface support (3,375 colors supported for color panel using dithering block)
  - 12-/16-/18-/24-bit thin-film transistor (TFT) panel interface support (replicated or dithered encoded pixel values)
  - RFB support through the remote frame buffer interface (RFBI) module
  - Partial display through the RFBI module
  - Second 24-bit digital output
  - Multiple cycle output format on 8-/9-/12-/16-bit interface (TDM)
- Signal processing
  - Overlay support for graphics (RGB or color look-up table [CLUT]), video1 (YCbCr 4:2:2, RGB16, and RGB24), and video2 (YCbCr 4:2:2, RGB16, and RGB24)
  - Video overlay up-sampling
  - Rotation 90°, 180°, and 270°
  - Transparency color key (source and destination)
  - Synchronized buffer update
  - Programmable video color space conversion of YCbCr 4:2:2 into RGB
  - Hardware cursor
  - Gamma curve support
  - Multiple-buffer support
  - Mirroring support
- Power modes
  - Low-power modes

Remote Frame Buffer Interface

- Access to RFB direct microprocessor unit (MPU) interface
  - Sending commands to the RFB panel
  - Sending data to the RFB panel, received data from the display controller or MPU (through the L4 port)
  - Reading data/status from the RFB to the OCP slave port
- RFBI
  - 8-/9-/12-/16-bit parallel interface
  - Two programmable configurations for two devices connected to the RFBI module
- Data formats
  - Programmable pixel modes (12-, 16-, 18-, and 24-bit-per-pixel modes in RGB format)
  - Programmable output formats on one/multiple cycles per pixel (data from display controller and L4)

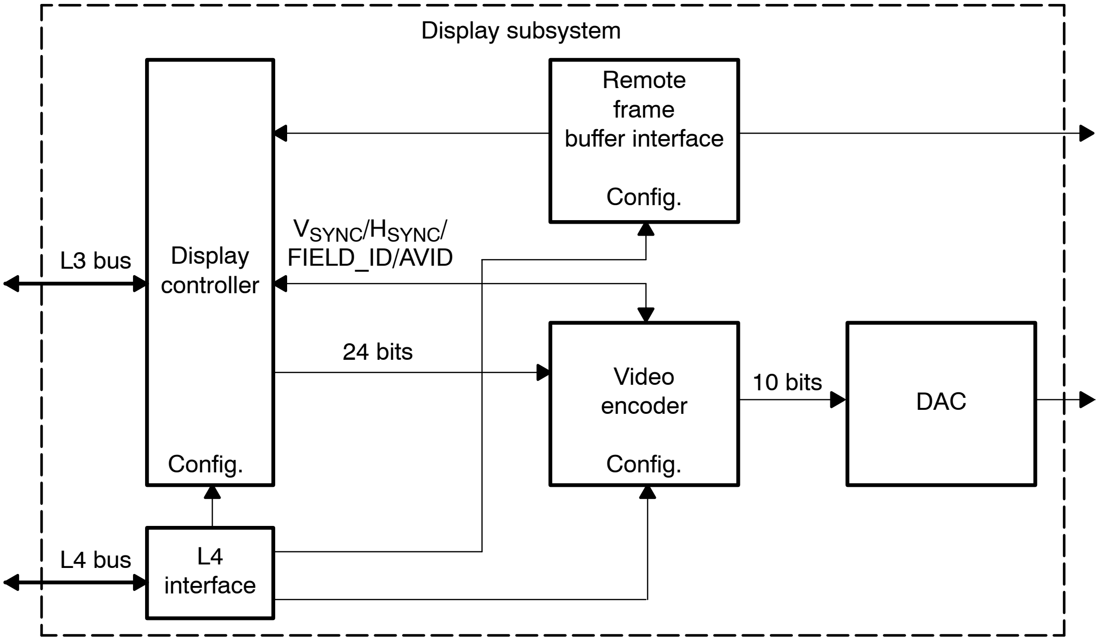
TV Encoder

- NTSC/PAL encoder outputs with the following standards:
  - NTSC-J, M, 4.43
  - PAL-B, D, G, H, I, N
  - PAL-M
  - PAL-N
  - PAL-Nc
  - PAL-60
- Input data interface compatible with the following protocols:
  - 24-bit input bus compatible with external sync
  - RGB 4:4:4
  - YCbCr 4:4:4
- Output data 10-bit interface for digital-to-analog converter (DAC)
- Master clock input 27 MHz (supports ITU-R 601 sampling for NTSC/PAL) and 54 MHz
- Programmable horizontal sync and vertical timing and waveforms
- Programmable subcarrier frequency and SCH
- 54-MHz output (4x oversampling)
- Composite video (CVBS) analog output signal

15.2 Display Subsystem Environment

The DSS handles two main functions: LCD display support and TV display support. This section describes both functions. Figure 15−2 is the DSS schematic.

Figure 15−2. DSS Schematic



15.2.1 LCD Support

The main subblocks of the LCD display path are the display controller and the RFBI.

The display controller provides the necessary control signals to interface the memory frame buffer (either SDRAM or SRAM) directly to the external displays. It is connected to the memory through the L3 interconnect and has its own DMA to read data from the system memory.

The RFB of the LCD panel is connected directly to the RFBI module. 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 it can set the DISPC registers directly to read/write the data from/to the memory in the LCD panel. The RFBI can manage two LCD panels, as shown in Figure 15−3.

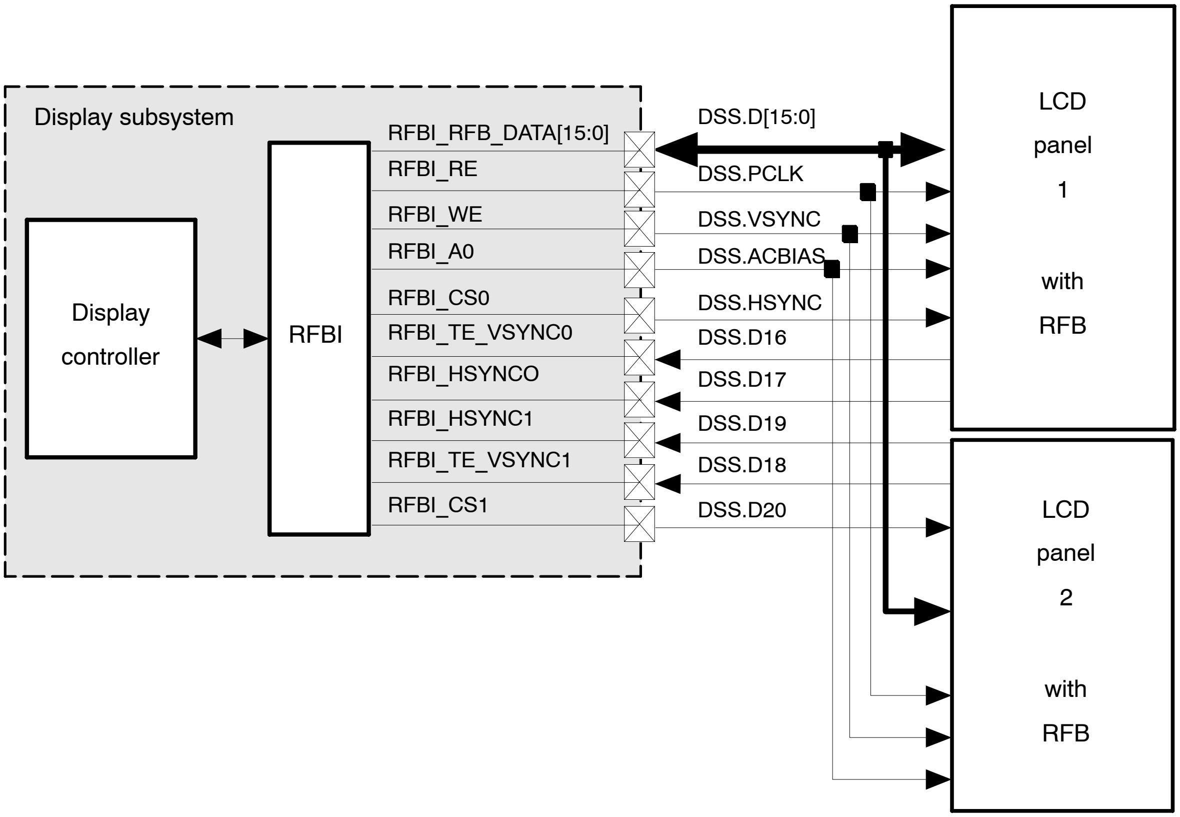
The display controller has two input/output (I/O) pad modes at the module level: RFBI mode (RFBI enabled) and bypass mode (RFBI disabled). The selection of these modules is controlled by the GPOUT[1:0] bits DISPC_CONTROL[16:15], as shown in Table 15−1.

Table 15−1.//O Pad Mode Selection

	GPOUT1 = 0	**GPOUT1 = 1**
GPOUT0 = 0	00 (RESET)	01 (INVALID)
GPOUT0 = 1	10 (RFBI mode)	11 (bypass mode)

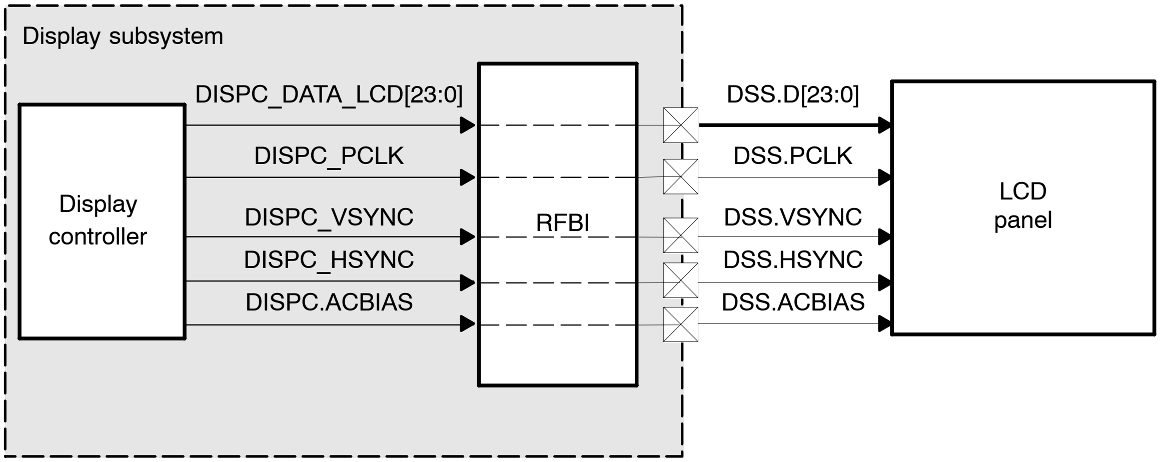
If the RFBI is enabled, the display controller must be set to use the RFBI mode, as shown in Figure 15−3.

Figure 15−3. LCD Support Interface (RFBI Mode)



If the RFBI is not enabled, the bypass mode is set and the DISPC drives the signals directly to the LCD panel, as shown in Figure 15−4.

Figure 15−4. LCD Support Interface (Bypass Mode)



In RFBI mode, two devices can be driven at the same time. Configuring RFBI CONFIGSELECT RFBI_CONTROL[3:2] makes it possible to drive signals for LCD 1 only, for LCD 2 only, or for both LCD 1 and LCD 2.

15.2.1.1 LCD Interface and Pin Description

Table 15−2 and Table 15−3 describe the interface signals to/from the LCD panel in both RFBI and bypass modes respectively.

Table 15−2. LCD Interface Signals (RFBI Mode)

Signal Name	Description Configuration RFBI	Status
RFBI_RFB_DATA[15:0]	RFBI input/output	I/O
RFBI_WE	Write access signal	O
RFBI_RE	Read access signal	O
RFBI_CS0	Chip-select signal for LCD 1	O
RFBI_CS1	Chip-select signal for LCD 2	O
RFBI_A0	Command/data selection signal	O
RFBI_TE_VSYNC0	Tearing effect synchronization signal (TE or VSYNC for the LCD panel 1)	I
RFBI_TE_VSYNC1	Tearing effect synchronization signal (TE or VSYNC for the LCD panel 2)	I
RFBI_HSYNC0	HSYNC from the LCD panel 1	I
RFBI_HSYNC1	HSYNC from the LCD panel 2	I

- ❏ RFBI_RFB_DATA: 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 any necessary commands and parameters to configure the LCD panel. The data is used as input in read operations during production test and to read the status of the registers in the LCD panel. RFBI_RFB_DATA is muxed at chip-level boundary with DSS.D[15:0].
- ❏ RFBI_RE: This signal is the read enable (RE) used to indicate when a read is ongoing from the embedded memory in the LCD panel. The RFBI regis-

ters describe the behavior of the read signal (off/on/cycle time). The polarity of the RE signal is programmable. This signal is muxed at chip-level boundary with DSS.PCLK. The read is used to get status/data information from the LCD panel.

- ☐ RFBI_A0: This 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 that have been 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 into the RFBI registers CMD/READ/STATUS/PARAM/PIXEL define the transitions of A0. This signal is muxed at chip-level boundary with DSS.ACBIAS.
- ☐ RFBI_CSx: This signal is the chip-select CSx asserted to indicate which LCD panel is selected and must be ready to receive/transmit commands and data. When RE or write enable (WE) is on, CSx must not be changed.

  x = 0 for LCD panel 1, and x = 1 for LCD panel 2.

  CS0 is muxed at the chip-level boundary with DSS.HSYNC, and CS1 is muxed at chip-level boundary with DSS.D[20].
- ☐ RFBI_WE: The WE 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 WE signal is programmable. This signal is muxed at chip-level boundary with DSS.VSYNC.
- ☐ RFBI_HSYNCx: The HSYNC pulse signals indicate to the RFBI module when the 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 tearing effect (TE) logic as a synchronization signal for sending the pixel to the LCD panel. These signals are muxed at chip-level boundary with DSS.D[17] (RFBI_HSYNC0) and DSS.D[19] (RFBI_HSYNC1).

  x = 0 for LCD panel 1, and x = 1 for LCD panel 2.
- ☐ RFBI_TE_VSYNCx: Based on the trigger mode selected, the signal is the TE pulse signal or the LCD panel vertical synchronization (VSYNC) pulse signal. RFBI_TE_VSYNC is used by TE logic as a synchronization signal for sending the pixel to the LCD panel. These signals are muxed at chip-level boundary with DSS.D[16] (RFBI_TE_VSYNC0) and DSS.D[18] (RFBI_TE_VSYNC1).

  x = 0 for LCD panel 1, and x = 1 for LCD panel 2

Table 15−3. LCD Interface Signals (Bypass Mode)

Signal Name	Description Configuration Bypass	Status
DISPC_DATA_LCD[23:0]	LCD DATA from the DISPC	O
DISPC_VSYNC	VSYNC from DISPC	O
DISPC_PCLK	Pixel CLK from DISPC	O
DISPC_HSYNC	HSYNC from DISPC	O
DISPC_ACBIAS	ACBIAS from DISPC	O

- DISPC_DATA_LCD: The panel pixel data comes directly from the DISPC. DISPC_DATA_LCD is muxed at chip-level boundary with DSS.D[23:0].
- DISPC_VSYNC: The VSYNC signal from the display controller is used. 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 have elapsed both at the beginning and at the end of each frame. This signal is muxed on chip-level boundary with DSS.VSYNC.
- DISPC_PCLK: This signal is the pixel clock that comes directly from the DISPC. This signal is muxed at chip-level boundary with DSS.PCLK.
- DISPC_HSYNC: The horizontal sync (HSYNC) signal from the display controller is used. The LCD line clock (HSYNC) toggles after all the pixels in a line are transmitted to the LCD panel and a programmable number of pixel clock wait states has elapsed both at the beginning and at the end of each line. This signal is muxed on chip-level boundary with DSS.HSYNC.
- DISPC_ACBIAS: The ac-bias signal from the display controller is used.

  In passive matrix technology, the ac bias is configured to transition each time a programmable number of line clocks occurs. To prevent dc charge in 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 (TFT panels), the ac-bias signal acts as an output enable signal to indicate when data must be latched using the pixel clock. This signal is muxed on chip-level boundary with DSS.ACBIAS.

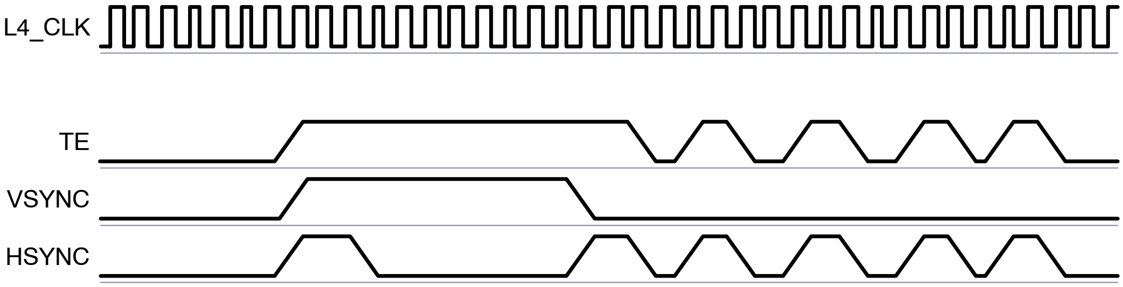
Tearing Effect Pulse Signal

The TE synchronization signal is an ORed or ANDed signal between HSYNC and VSYNC. It is generated externally. The hardware must detect the VSYNC and HSYNC pulses embedded in the received signal. The VSYNC is detected based on the minimum pulse width defined by the RFBI_VSYNC_WIDTH register and is not triggered by an inactive-to-active edge. HSYNC is detected based on the minimum pulse width defined by the RFBI_HSYNC_WIDTH register and is triggered by an inactive-to-active edge if the value RFBI_HSYNC_WIDTH.MinHSYNCWidth equals 0. 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 a bit field in the RFBI registers can be used to start the capture of the data. The polarity of the TE signals 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.

The hardware resets the line counter when the VSYNC occurs and increments it at every HSYNC. When the line counter reaches the programmable line number, the transfer to the LCD panel begins.

Figure 15−5 shows the external generation of TE based on ORed (active high) HYSNC and VSYNC.

Figure 15−5. External Generation of TE Based on ORed HSYNC and VSYNC (Active High)



VSYNC/HSYNC Pulse Signals

The VSYNC pulse signals indicate to the RFBI module when the vertical synchronizations occur. The polarity of the VSYNC signals is programmable. The VSYNC is detected based on the minimum pulse width defined by the RFBI_VSYNC_WIDTH register and is triggered by an inactive-to-active edge if the value RFBI_VSYNC_WIDTH.MinVSYNCWidth equals 0. The HSYNC is detected based on the minimum pulse width defined by the RFBI_HSYNC_WIDTH register and is triggered by an inactive-to-active edge if the value RFBI_HSYNC_WIDTH.MinHSYNCWidth equals 0.

The hardware resets the line counter when the VSYNC occurs and increments it at every HSYNC. When the line counter reaches the programmable line number, the transfer to the LCD panel begins.

15.2.1.2 LCD Output and Data Format

This section describes the pixel data bus and shows some timing diagrams of transactions and synchronizations in both RFBI and bypass modes.

Figure 15−6 through Figure 15−12 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 according to the I/O data bus width.

❏ Passive matrix technology—monochrome mode: Monochrome displays use a 4- or 8-bit interface. Each bit represents one pixel (on or off), which means that at each pixel clock, 4 or 8 pixels are sent to the LCD.

Figure 15−6. LCD Pixel Data Monochrome—4 STN

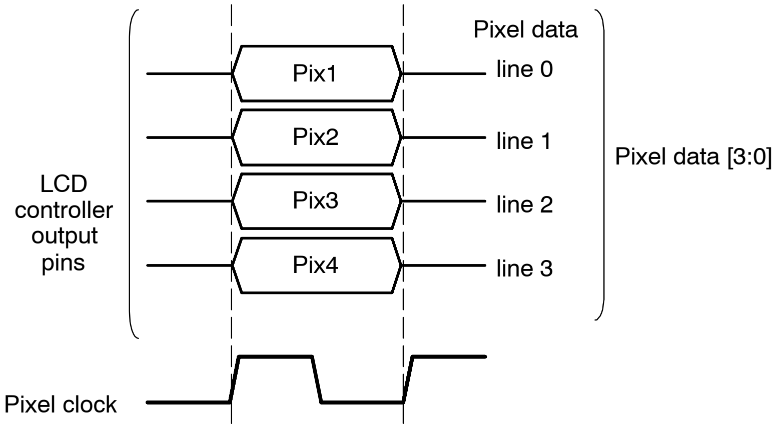
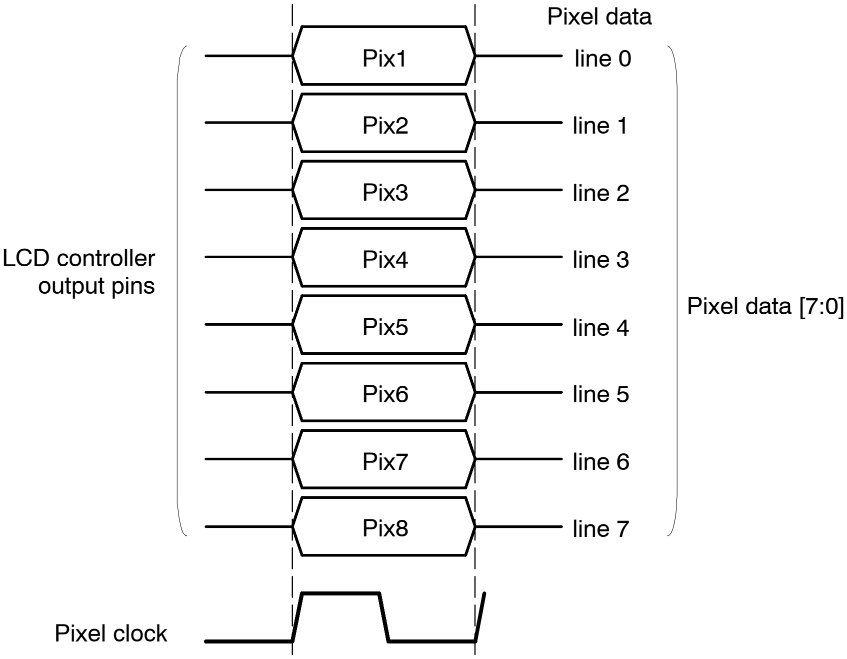
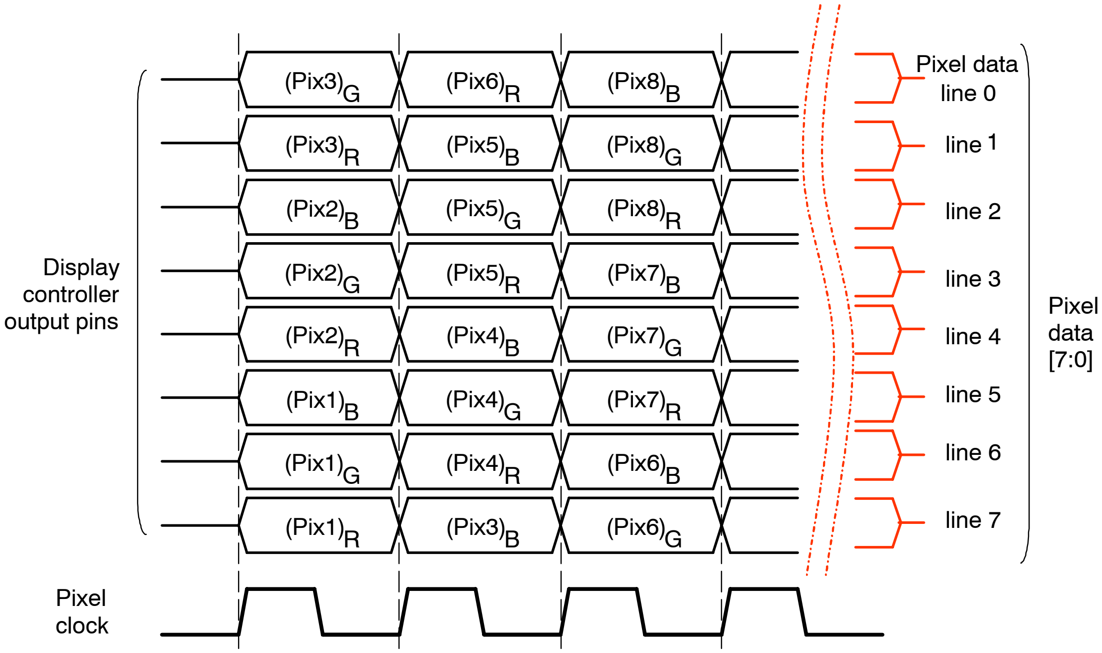


Figure 15−7. LCD Pixel Data Monochrome—8 STN



- ❑ 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 15–8. LCD Pixel Data—Color STN



- ☐ Active matrix technology—color mode: In TFT displays, the dithering logic and the output first-in first-out (FIFO) are bypassed. Each line represents one pixel.

Number of Pixels	Display
1	TFT
8/3	STN color
4	Mono 4-bit
8	Mono 8-bit

Figure 15−9. LCD Pixel Data—Color 12 TFT

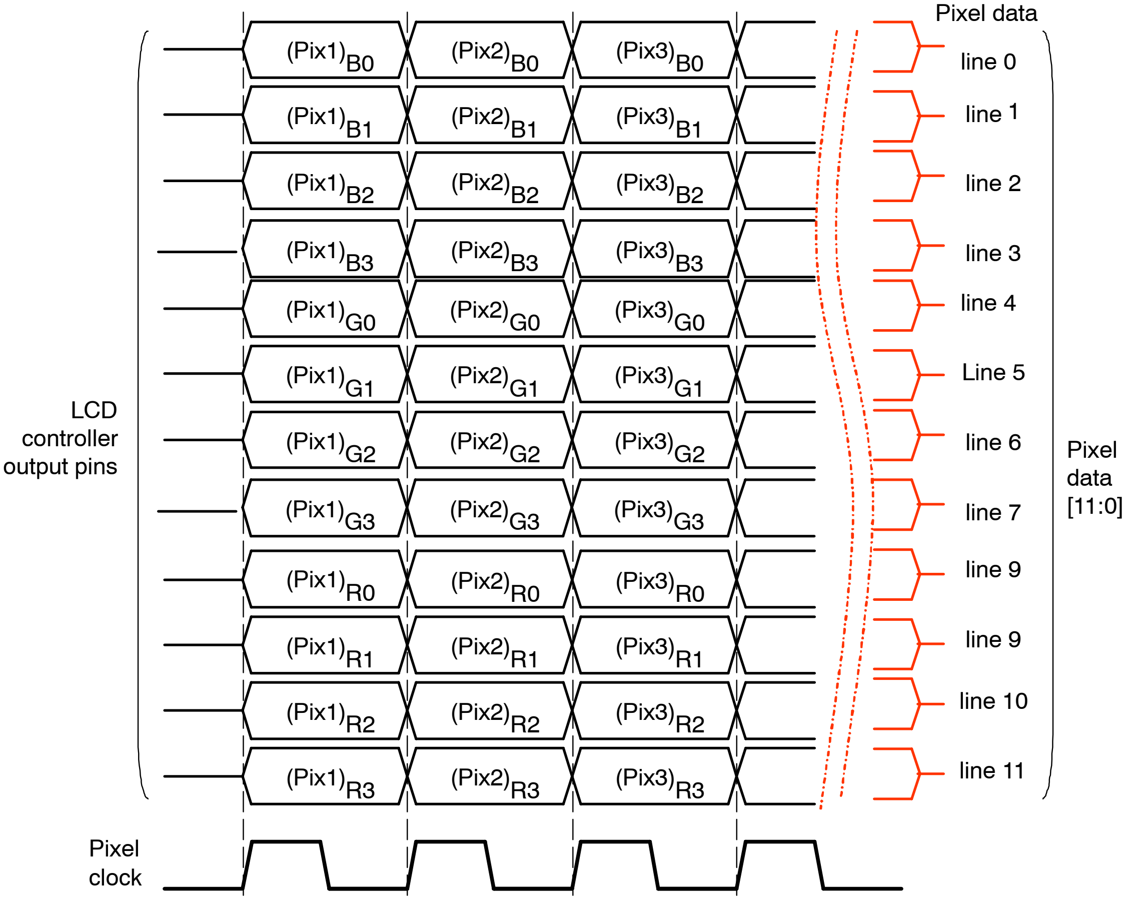

Pixel data
line 0
line 1
line 2
line 3
line 4
Line 5
line 6
line 7
line 9
line 9
line 10
line 11
Pixel data [11:0]
LCD controller output pins
(Pix1)B0 (Pix2)B0 (Pix3)B0
(Pix1)B1 (Pix2)B1 (Pix3)B1
(Pix1)B2 (Pix2)B2 (Pix3)B2
(Pix1)B3 (Pix2)B3 (Pix3)B3
(Pix1)G0 (Pix2)G0 (Pix3)G0
(Pix1)G1 (Pix2)G1 (Pix3)G1
(Pix1)G2 (Pix2)G2 (Pix3)G2
(Pix1)G3 (Pix2)G3 (Pix3)G3
(Pix1)R0 (Pix2)R0 (Pix3)R0
(Pix1)R1 (Pix2)R1 (Pix3)R1
(Pix1)R2 (Pix2)R2 (Pix3)R2
(Pix1)R3 (Pix2)R3 (Pix3)R3
Pixel clock

Figure 15–10. LCD Pixel Data—Color 16 TFT

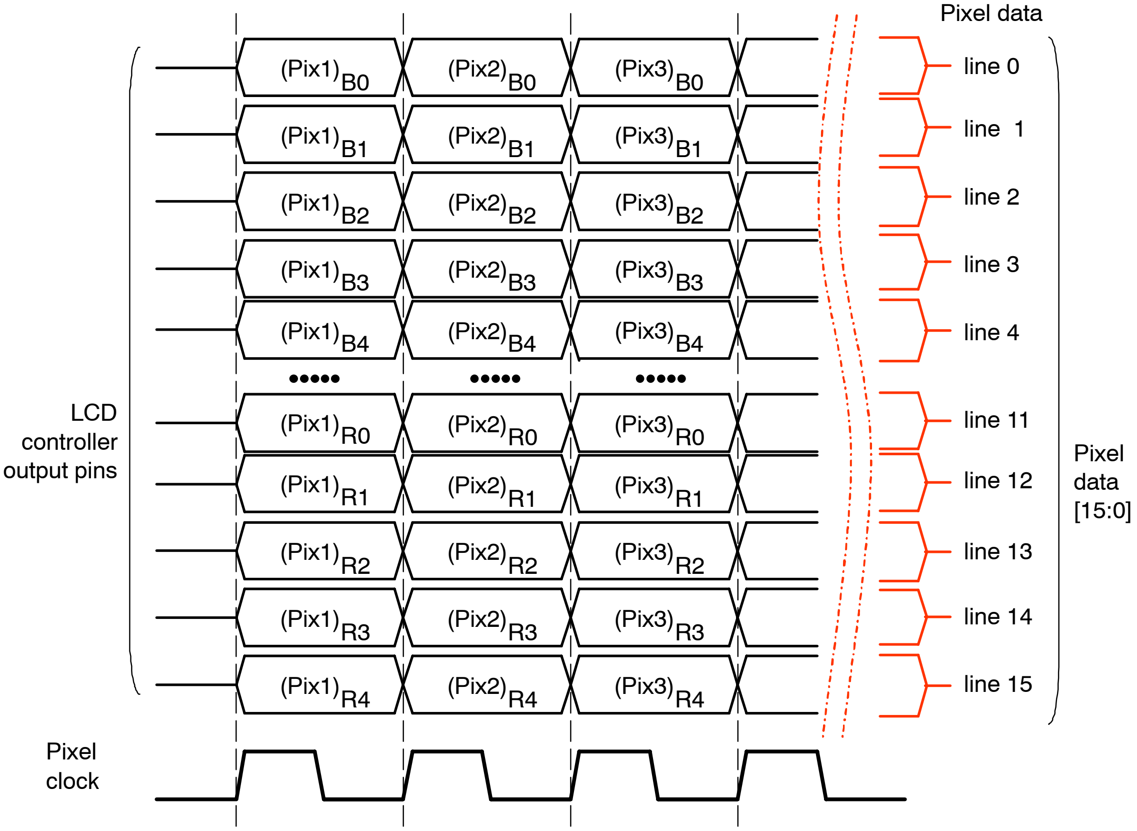


Figure 15–11. LCD Pixel Data—Color 18 TFT

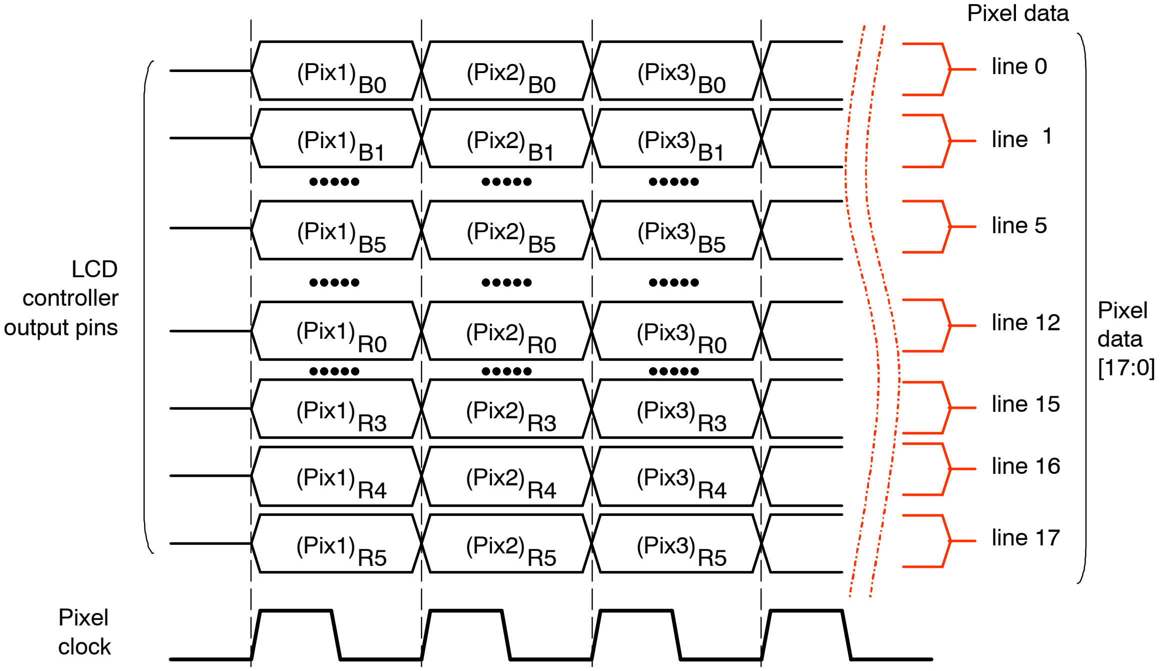


Figure 15−12. LCD Pixel Data—Color 24 TFT

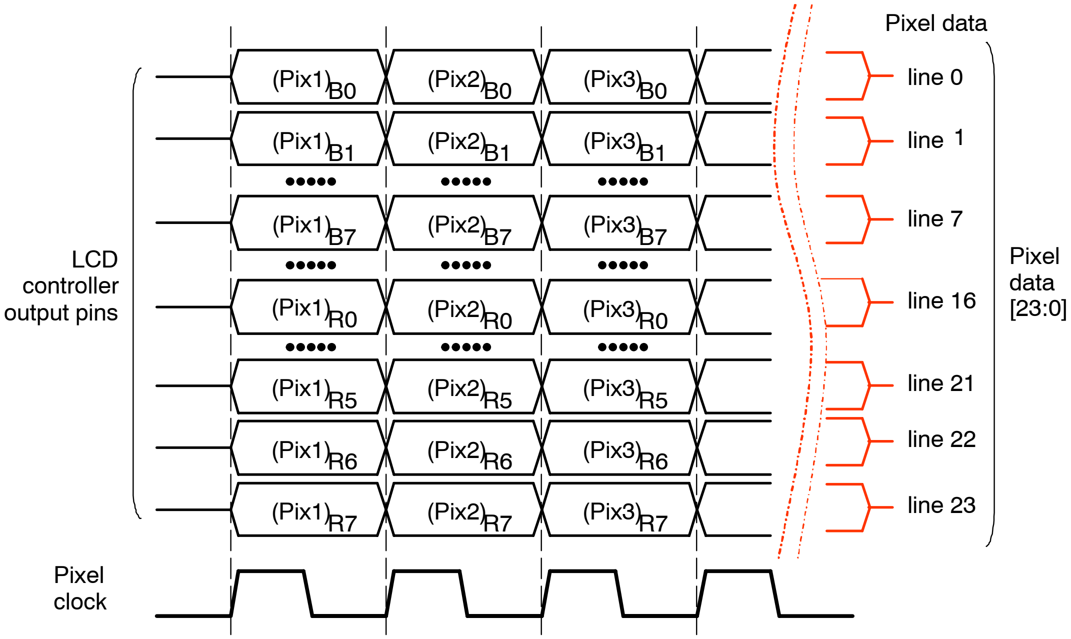


Figure 15−13 through Figure 15−15 are timing diagrams of read/write transactions to the LCD panel for RFBI mode. These diagrams show the possible situations that can be found when writing data into or reading data from the LCD panel and are related to some programmable timing fields.

The programming timing fields mentioned are:

- ☐ CSOnTime: CS assertion time from start access time
RFBI CSONTIME RFBI_ONOFF_TIMEi[3:0]
- ☐ CSOffTime: CS deassertion time from start access time
RFBI CSOFFTIME RFBI_ONOFF_TIMEi[9:4]
- ☐ WeCycleTime: The time when A0 becomes valid until write cycle completion. RFBI WECYCLETIME field RFBI_CYCLE_TIMEi[5:0].
- ☐ WeOnTime: WE assertion delay time from start access time
RFBI WEONTIME field RFBI_ONOFF_TIMEi[13:10]
- ☐ WeOffTime: WE deassertion delay time from start access time
RFBI WEOFFTIME field RFBI_ONOFF_TIMEi[19:14]
- ☐ ReCycleTime: The time when A0 becomes valid until read cycle completion. RFBI RECYCLETIME field RFBI_CYCLE_TIMEi[11:6].
- ☐ ReOnTime: RE assertion delay time from start access time
RFBI REONTIME field RFBI_ONOFF_TIMEi[23:20].
- ☐ ReOffTime: RE deassertion delay time from start access time
RFBI REOFFTIME field RFBI_ONOFF_TIMEi[29:24]
- ☐ CSPulseWidth: The time when write cycle time or read cycle time is complete. RFBI CSPULSEWIDTH field RFBI_CYCLE_TIMEi[17:12].

Figure 15−13. Command Data Write

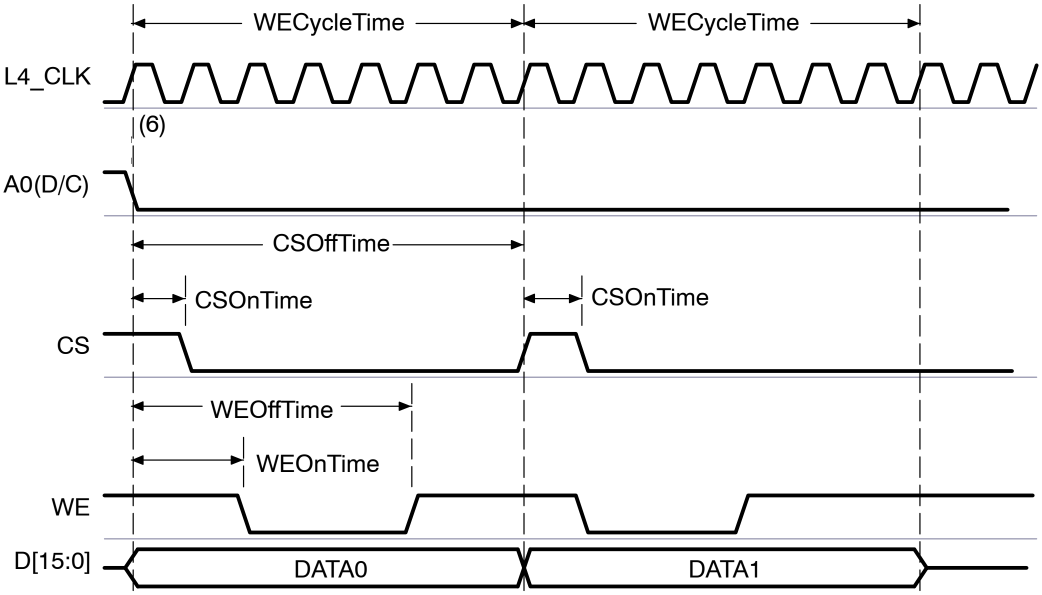


Figure 15−14. Display Data Read

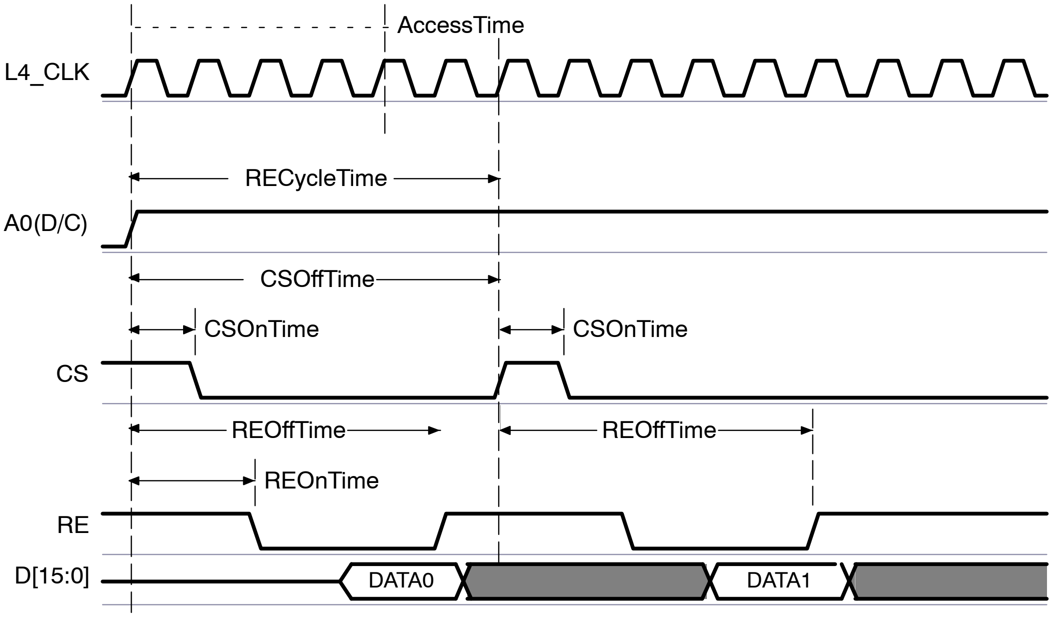


Figure 15−15. Read-to-Write and Write-to-Read

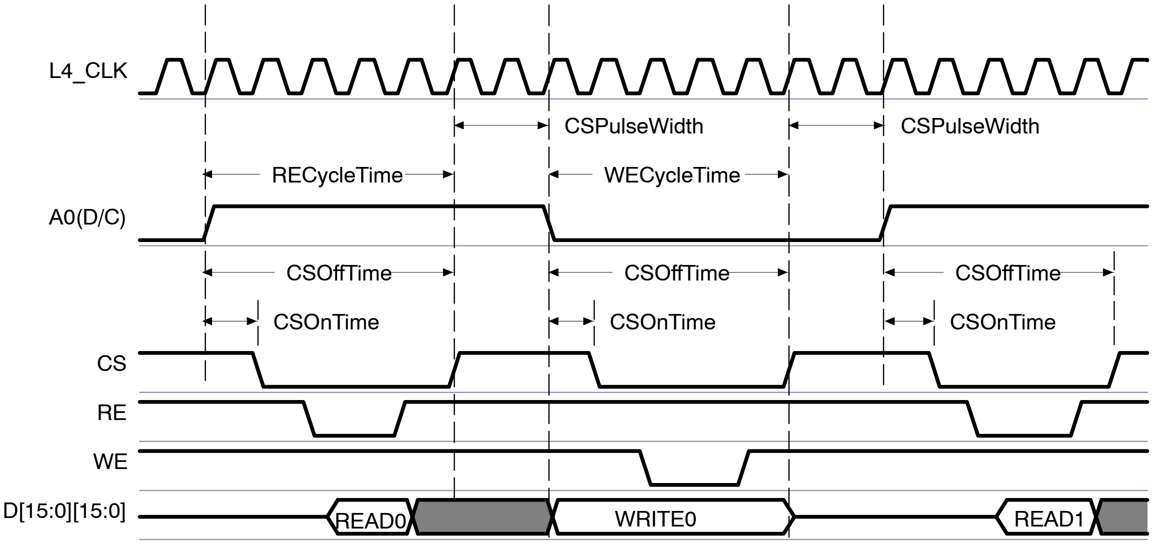


Figure 15−16 through Figure 15−18 are timing diagrams of synchronization signals and pixel clock in bypass mode for both STN and TFT panels. These signals are driven directly by the display controller and are related to the following programmable fields:

- ☐ PPL: Pixels per line. DISPC PPL DISPC_SIZE_LCD[10:0].
- ☐ LPP: Lines per panel. DISPC LPP DISPC_SIZE_LCD[26:16].
- ☐ HBP: Horizontal back porch. DISPC HBP DISPC_TIMING_H[27:20].
- ☐ HFP: Horizontal front porch. DISPC HFP DISPC_TIMING_H[15:8].
- ☐ HSW: Horizontal synchronization pulse width. DISPC HSWDISPC_TIMING_H[5:0].
- ☐ VFP: Vertical front porch. DISPC VFP DISPC_TIMING_V[15:8].
- ☐ VBP: Vertical back porch. DISPC VBP DISPC_TIMING_V[27:20].
- ☐ VSW: Vertical synchronization pulse width. DISPC VSWDISPC_TIMING_V[5:0].
- ☐ IHS: Invert HSYNC. DISPC IHS DISPCPOL_FREQ[13].
- ☐ IVS: Invert VSYNC. DISPC IVS DISPCPOL_FREQ[12].

Figure 15−16. TFT Timing Diagram

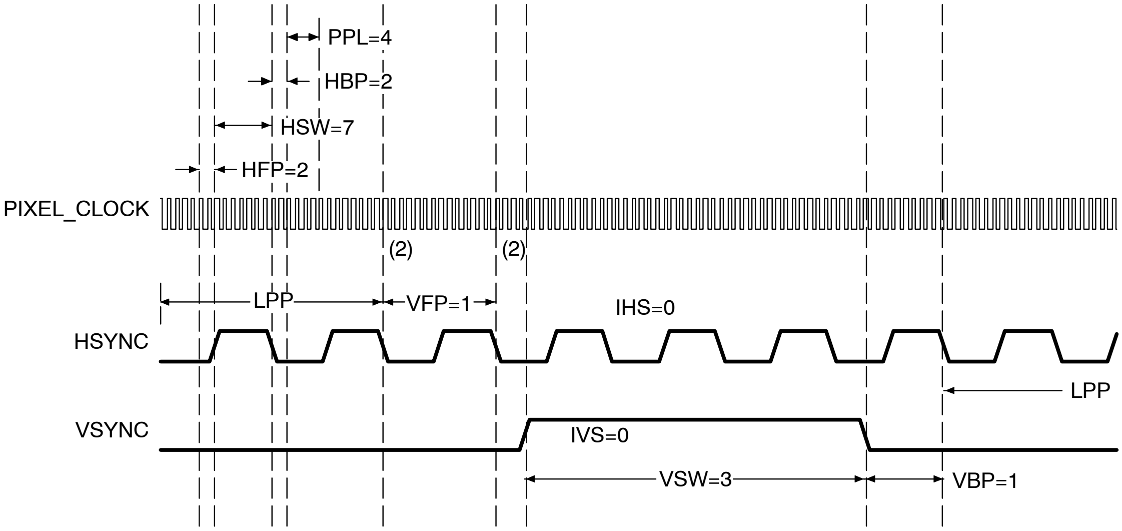


Figure 15−17. STN Timing Diagram (Beginning of Frame)

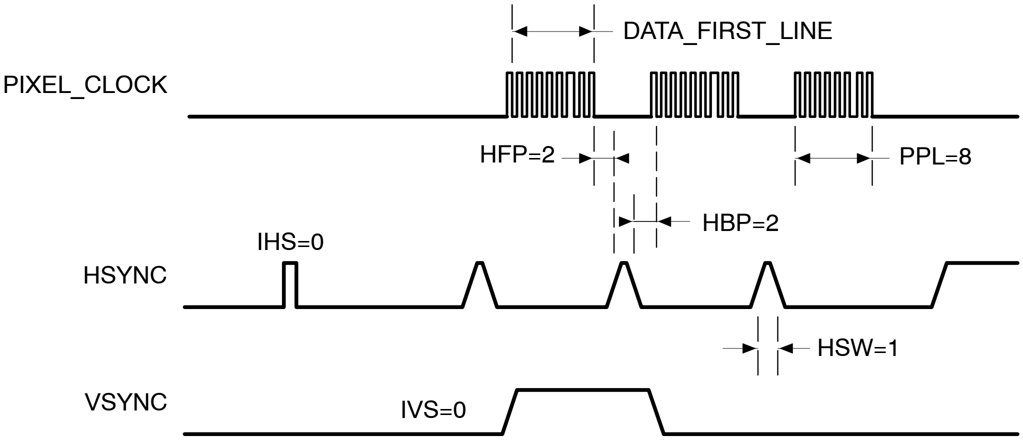
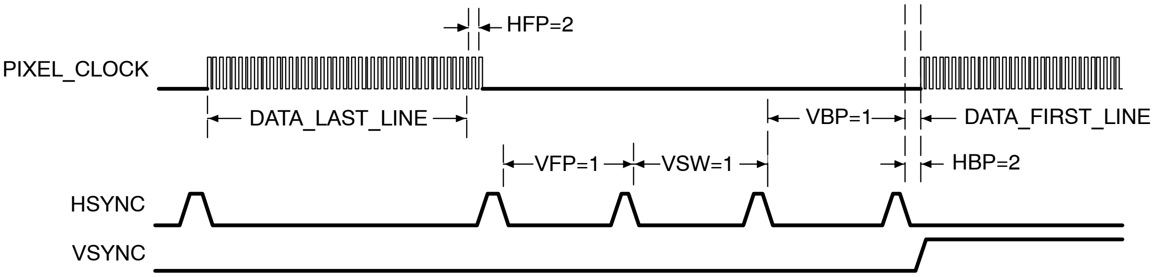


Figure 15−18. STN Timing Diagram (End of Frame)



15.2.2 TV Display Support

The main subblocks of the TV display path are the display controller, the video encoder, and the embedded DAC. The outputs of the TV path go directly to a 10-bit DAC optimized for video applications.

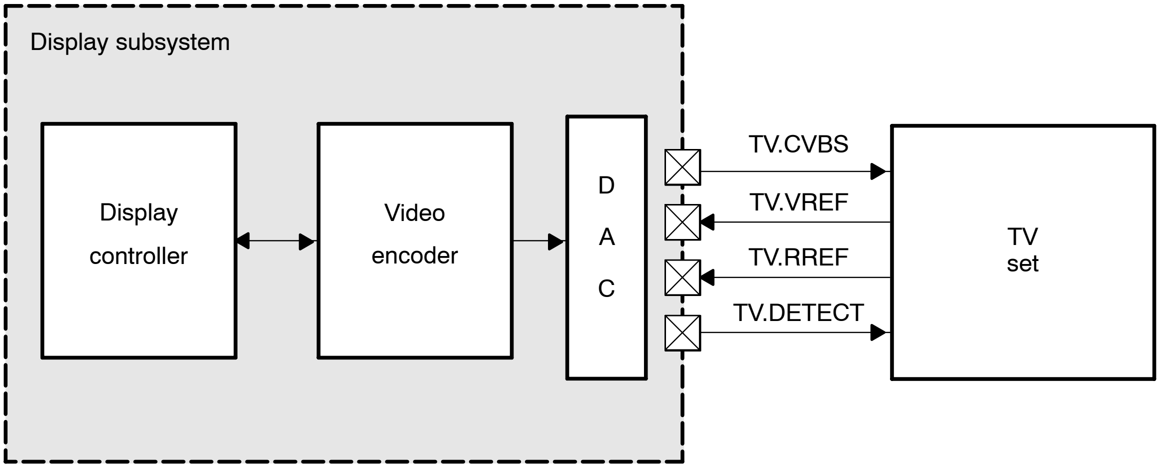
The display controller digital-output logic receives the external digital clock and, based on the VSYNC, holds the time and active video (AVID) signals from the video encoder (VENC) and outputs the pixels synchronously to the external clock. The size of the field/frame to output defines the number of pixels on each line and the number of lines.

The digital output from the DISPC is always a 24-bit RGB value based on a pixel request from the VENC.

The VENC converts RGB video signals to conform to the NTSC/PAL standard analog video. The encoder includes an integrated synchronization signal generator, one channel DAC, a data manager, luma (brightness information) stage, chroma (color information) stage, modulator, and control interface. The VENC also provides all the synchronization signals to the display controller: VSYNC, AVID, and field ID (FID).

Figure 15−19 shows the TV display support.

Figure 15−19. TV Display Support



15.2.2.1 TV Interface and Pin Description

Table 15−4 describes the interface to/from the DAC of the TV set interface.

Table 15−4. TV Interface Signals

Signal Name	Description	Status
TV.CVBS	Analog composite output	O
TV.VREF	Input voltage reference (analog)	I
TV.RREF	External resistor reference (analog, 4 kΩ)	I
TV.DETECT	Pulse detection	O

CAUTION

To avoid current leakage when TV output is not used, tie the following analog pins to ground: VDDADAC, VSSADAC, TV.VREF, and TV.RREF.

15.2.2.2 TV Output and Data Format

The output data to the TV set are the analog composite data from the DAC. The video standards supported are:

- NTSC-J, M, 4.43
- PAL-B, D, G, H, I, N
- PAL-M
- PAL-N
- PAL-Nc
- PAL-60

15.2.2.3 DAC

The DAC integrated into the device is a 10-bit current-steering DAC optimized for video applications in a deep submicron process. The DAC operates from a 1.8-V analog supply and a 1.3-V digital supply. It supports update rates up to 60 MSPS.

The main features of the DAC are:

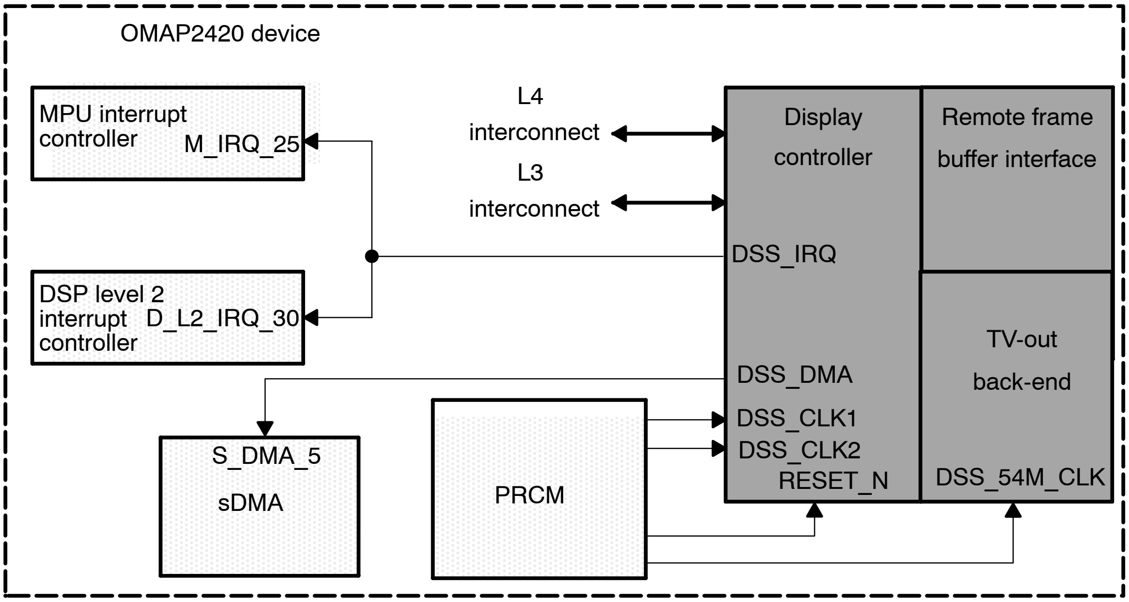
- 10-bit resolution
- DNL within ± 0.5 LSB and INL within ± 1 LSB
- Sample rates up to 60 MSPS
- Nominal full-scale current output: 1.0 mA
- Suitable for low-power wireless application. Power consumption from analog supply of less than 2.0 mA.
- Power supply: 1.0 V to 1.3 V digital, 1.8 V analog

To enhance TV color display, set the DAC_DEMEN bit of the DSS_CONTROL register.

15.3 Display Subsystem Integration

This section describes the integration of the DSS and provides details about clocks, resets, hardware requests, and power modes. Figure 15−20 shows the integration of the DSS in the device.

Figure 15−20. DSS Integration



15.3.1 Clocking, Reset, and Power-Management

15.3.1.1 Clock Domains

Table 15−5 lists the connections of clock signals between the DSS and the power, reset, and clock management (PRCM) module.

Table 15−5. Function Clocks From PRCM

Attributes	Values	DSS PORT	PRCM PORT	Comments
Functional clock	12/13/19.2 MHz up to 133 MHz	DSS_CLK1	DSS_CLK1	From PRCM: SYSCLK From PRCM: DPLL
	12/13/19.2 MHz			From PRCM: SYSCLK
	48 MHz	DSS_CLK2	DSS_CLK2	From PRCM: APLL_48
	54 MHz	DSS_54M_CLK	DSS_54M_clk	From PRCM: APLL-54
	Alt clock			From PRCM:SYS.ALTCLK
	L3 clock	DSS_L3_ICLK	DSS_L3_ICLK	FROM PRCM
	L4 clock	DSS_L4_ICLK	DSS_L4_ICLK	FROM PRCM

The clock enable bits from the PRCM are:

- DSS_CLK1: CM_FCLKEN1_CORE[0]
- DSS_CLK2: CM_FCLKEN1_CORE[1]
- DSS_54M_CLK: CM_FCLKEN1_CORE[2]
- DSS_L3_ICLK: CM_ICLKEN_CORE[0]
- DSS_L4_ICLK: CM_ICLKEN1_CORE[0]

The display subsystem has four clock domains:

- ❑ L4 clock domain
- ❑ L3 clock domain
- ❑ Encoder clock domain: Three clock frequencies based on the DSS_54M_CLK (54/27/13.5 MHz on normal mode, or up to 59/29.5/14.75 MHz for square pixel)
- ❑ Display controller clock domain (switch from DSS_CLK1 and DSS_CLK2 clock)

Encoder Functional Clock

This clock is used in the VENC, and a divided one is used in the display controller.

The DSS_54M_CLK is divided to create the three balanced clocks listed in Table 15−6, depending on the clock mode selected.

Table 15−6. Possible Digital Clock Division

Clock Output	Clock Mode 0	Clock Mode 1
VENC Clock 4x	DSS_54M_CLK	DSS_54M_CLK or 0 (Gated: Programming DSS VENC_CLOCK_4X_ENABLA DSS_CONTROL[3] (0: gated, 1: not gated)
VENC Clock 2x	DSS_54M_CLK/2	DSS_54M_CLK
VENC Clock 1x	DSS_54M_CLK/4	DSS_54M_CLK / 2

Display Controller Functional Clock

The display controller functional clock can be up to 133 MHz.

This clock can be selected from the DPLL clock (DSS_CLK1, default selection) and the APLL clock (DSS_CLK2), by writing the DPLL_APLL_CLK bit of the DSS_CONTROL register (0: DPLL; 1: APLL). The clock selection becomes active at the next inactive part of the video frame (VFP) after the GOLCD bit is set.

When switching from DSSx to DSSy functional clocks, DSSx can be cut off after the next inactive part of the video frame following the one where the clock switching is performed.

Remote Frame Buffer Functional Clock

The RFB uses the display controller functional clock and the L4 clock as the functional clocks for the RFBI.

L3 Clock

The L3 clock is used only by the display controller interface to fetch pixel data. The L3 clock can be shut down at any time, assuming that the display controller does not underflow independently of the MStandby status.

There is no ratio constraint between clocks (DSS_CLK1, DSS_CLK2, DSS_L3_ICLK, and DSS_L4_ICLK).

The interface clocks (DSS_L3_ICLK and DSS_L4_ICLK) can be stopped under conditions such as DPLL reprogramming, state change between lock and bypass, and so on.

L4 Clock

The L4 clock is used for the configuration of all submodules and as the functional clock for the RFB interface module.

15.3.1.2 Power and Reset Domains

- Power domains: DSS modules are in the core power domain, except for the DAC, which is in the analog power domain.
- Hardware reset: DSS hardware reset is connected to the PRCM CORE_RST_N signal in the CORE_RST domain.
- Software reset: The DSS can propogate a software reset through all the submodules. This reset can be done to initialize the subsystem. To apply this software reset, write the DSS SOFT_RESET DSS_SYSCONFIG[1] register (1: reset; 0: normal) and wait until the subsystem is reset. The software reset functionality is equivalent to a hardware reset.

15.3.1.3 Power Management

- Autoidle mode: The RFBI, DISPC, and L4 Interfaces can automatically gate their clocks internally to decrease power consumption. When the functionality is set, the clocks are gated if no transactions are present on the related bus. To enable this function, the following bits must be set:
  - DISPC AUTOIDLE DISPA_SYSCONFIG[0] (1: autoidle; 0: clock free-running)
  - RFBI AUTOIDLE RFBI_SYSCONFIG[0] (1: autoidle; 0: clock free-running)
  - DISS AUTOIDLE DSS_SYSCONFIG[0] (1: autoidle; 0: clock free-running)
  - FUNCGATED DISPC_CONFIG[9] must be set to 1.

Note:

To save power, set the FUNCGATED bit field on the DISPC_CONFIG register to 1.

- Standby mode: As part of the system-wide power-management scheme, the module can go into standby state.

  The module can be configured by writing the DISPC MIDLEMODE DISPC_SYSCONFIG[13:12] register (0x0: forced standby; 0x1: no standby; 0x2: smart standby) in one of the following standby modes:

- Forced mode
- No standby mode
- Smart standby mode

The conditions to enter standby mode are:

- Forced standby mode: Standby state is entered when the module is disabled.
- No standby mode: The module never goes into standby state.
- Smart standby mode: The module enters standby state when either of the following occurs:
    - The number of bytes in all the FIFOs is greater than or equal to the high thresholds for the FIFOs. Only the FIFOs of the enabled pipelines are taken into account.
    - The display controller is disabled.

When in standby state, the display controller does not generate transactions on the L3 master port. Standby is active when the PRCM module confirms this mode.

The conditions to exit standby state are:

- Forced standby mode: Standby state is exited when the display controller is enabled.
- Smart standby mode: Standby state is exited when the number of bytes in one of the FIFOs is equal to or less than the low threshold of the FIFO. Only the FIFOs of the enabled pipelines are taken into account.

When the DSS initiates a standby procedure, it also initiates an MSTANDBY/WAIT handshake protocol with the PRCM. The result of this protocol is to allow the PRCM to cut the DSS clocks. Depending on the PRCM setting, two modes are available:

- Manual mode:
    - DSS_CLK1 is shut down when the PRCM EN_DSS1 bit (CM_FCLKEN1_CORE[0]) = 0 and DSS is in standby mode.
    - DSS_CLK2 is shut down when the EN_DSS2 bit (CM_FCLKEN1_CORE[1]) = 0 and DSS is in standby mode.
    - DSS_L3_ICLK and DSS_L4_ICLK are controlled together. They are shut down when the EN_DSS bit (CM_ICLKEN1_CORE[0]) = 0 and DSS is in standby mode. This operation is optional, but recommended.

CAUTION

DSS_54M_CLK does not depend on the DSS standby state. Setting the EN_TV (CM_FCLKEN1_CORE[2]) bit stops (bit is set to 0) or activates (bit is set to 1) DSS_54M_CLK. The software must ensure the correct clock management for DSS_54M_CLK.

The clocks are reactivated when the related bits are set to 1 and DSS exits from standby. See Chapter 5, *Power, Reset, and Clock Management*, for additional information.

- Autocontrol mode:

  DSS_L3_ICLK and DSS_L4_ICLK also have an automatic mode in which they are cut when the MPU and DSS are both in standby state. This mode is activated through the AUTO_DSS bit (CM_AUTOIDLE_CORE[0]).

DSS Standby Mode, Power-Saving Use Cases

❑ Setup

1) Set DSS in smart standby mode.
2) Manually enable DSS_L3_ICLK and DSS_L4_ICLK.
3) Manually enable DSS_CLK1 or DSS_CLK2.
4) Manually enable DSS_54M_CLK if DAC is used.
5) Set the CM_CLKSTCTRL_CORE_AUTOSTAT_DSS and CM_AUTOIDLE1_CORE.AUTO_DSS bits (autocontrol mode).

❑ Low-power mode scenarios with active DSS (still picture, screen saver, and so on)

1) Set the CM_CLKSTCTRL_CORE.AUTOSTAT_DSS bit again when the MPU exits from standby mode.

❑ Shutdown DSS

1) Disable DSS.
2) Manually disable DSS_CLK1, DSS_CLK2, DSS_54M_CLK, DSS_L3_ICLK, and DSS_L4_ICLK.

15.3.2 Hardware Requests

15.3.2.1 DMA Request

One DMA request line is connected to the sDMA through the sDMA controller (S_DMA_5 input line).

The sDMA request is a synchronization signal from the DSS to the sDMA to inform that a programmable number of lines have been output to the LCD and that the system memory can be updated. This request is related to an interrupt event described in the following section.

15.3.2.2 Interrupt

One interrupt line DSS_IRQ is connected to two interrupt controllers:

❑ MPU interrupt controller (M_IRQ_25 input line)

- ☐ DSP level 2 interrupt controller (D_L2_IRQ_30 input line)

The interrupt line indicates when one or more events are detected by the hardware. Each event is independently maskable by setting the DISPC DISPC_IRQENABLE[14:0] register (see Section 15.6, *Display Subsystem Registers*).

To check when a particular interrupt event occurs and to reset a particular event, access the DISPC register DISPC_IRQSTATUS[14:0]. This register regroups 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 15.6, *Display Subsystem Registers*, for more information on checking and clearing interrupt events.

Table 15−7 lists the DSS interrupts.

Table 15−7. DSS Interrupts

Interrupt Name	**Description**
Frame Done	Active frame has completed and LCD output is disabled.
VSYNC	VSYNC interrupt has occurred at the end of the frame.
EVSYNC_EVEN	EVSYNC_EVEN interrupt has occurred at the end of the frame (EVSYNC received and the field polarity is even).
EVSYNC_ODD	EVSYNC_ODD interrupt has occurred at the end of the frame (EVSYNC received and the field polarity is odd).
AC Bias Count Status	The ac-bias transition counter has decremented to 0.
Programmed Line Number	The LCD has reached the user-programmed line number.
End of the video1 window	The screen has reached the end of the video1 window. All data for the video window have been fetched from the memory and displayed on the screen.
End of the video2 window	The screen has reached the end of the video2 window. All data for the video window have been fetched from the memory and displayed on the screen.
End of the graphics window	The screen has reached the end of the graphics window. All data for the graphics window have been fetched from the memory and displayed on the screen.
Video1 FIFO Underflow	The input video1 FIFO goes underflow.
Video2 FIFO Underflow	The input video2 FIFO goes underflow.
Graphics FIFO Underflow	The input graphics FIFO goes underflow.
Palette/Gamma Table Loading	The palette/gamma table has been loaded.
OCP Error	L3 interconnect has sent SResp = ERR.
SyncLost	Occurs when VSYNC width/front or back porches are not wide enough to load the pipe with data (LCD output)
SyncLostDigital	This interrupt is generated when the display controller is not ready to output data when a digital output request comes. This interrupt is used for the debug and informs the user when the timings of the video NTSC/PAL encoder are not set correctly.

Note:

The SyncLostDigital interrupts that occur before the first VSYNC pulse signal (from the video encoder) do not have to be taken into account.

After the first VSYNC pulse signal, the clear (by writing 0x1 in the SYNCLOSTDIGITAL bit DISPC_IRQSTATUS[15]) and the enable (writing 0x1 in the SYNCLOSTDIGITAL bit DISPC_IRQENABLE[15]) of the SyncLostDigital interrupt must be done.

The initialization sequence for the SyncLostDigital interrupt is as follows:

1) Initialize the VENC and DISPC.
2) Set the GODIGITAL bit DISPC_CONTROL[6] and the DIGITALENABLE bit DISPC_CONTROL[1] to 0x1.
3) Wait for the first receive external VSYNC pulse signal (when EVSYNC_ODD interrupt occurs, reading 0x1 from the EVSYNC_ODD bit DISPC_IRQSTATUS[3]).
4) Clear the SyncLostDigital interrupt by writing 0x1 in the SYNCLOSTDIGITAL bit DISPC_IRQSTATUS[15].
5) Enable the SyncLostDigital interrupt by writing 0x1 in the SYNCLOSTDIGITAL bit DISPC_IRQENABLE[15]).

To clear a SyncLost interrupt, ensure that the following sequence occurs:

1) SyncLost occurs.
2) Set the LCDENABLE bit DISPC_CONTROL[0] and DIGITALENABLE bit DISPC_CONTROL[1] to 0.

   Check the interrupts:

   - LCD: Check that FrameDone interrupt occurs.
   - TV: Check that EVSYNC_EVEN or EVSYNC_ODD interrupt occurs.

   All transfers are now complete, and the module can be reset properly.
3) Reset DSS (SOFTRESET bit DSS_SYSCONFIG[1] is set to 1).
4) Set the DSS registers again.

15.3.3 I/O Pad Mode Selection

The display controller has two I/O pad modes at the module level: RFBI mode (RFBI enabled) and bypass mode (RFBI disabled). Selection of these modes is controlled by the GPOUT[1:0] bit DISPC_CONTROL[16:15] (see Table 15−8).

Table 15−8. I/O Pad Mode Selection

	GPOUT1 = 0	**GPOUT1 = 1**
GPOUT0 = 0	00 (RESET)	01 (INVALID)
GPOUT0 = 1	10 (RFBI mode)	11 (bypass mode)

Table 15−9 shows the interface signals to/from the LCD panel in both RFBI and bypass modes.

Table 15−9. Interface Signals in RFBI/Bypass Modes

RFBI Mode	**Bypass Mode**	**Pin**	**DIR at Reset**	**DIR in Bypass Mode**	**DIR in RFBI Mode**
RFBI_RFB_DATA[15:0]	DISPC_DATA_LCD[15:0]	DSS.D[15:0]	I	O	I/O
Unused	DISPC_DATA_LCD[23:21]	DSS.D[23:21]	I	O	X
RFBI_WE	DISPC_VSYNC	DSS.VSYNC	I	O	O
RFBI_RE	DISPC_PCLK	DSS.PCLK	I	O	O
RFBI_CS0	DISPC_HSYNC	DSS.HSYNC	I	O	O
RFBI_CS1	DISPC_DATA_LCD[20]	DSS.DATA[20]	I	O	O
RFBI_A0	DISPC_ACBIAS	DSS.ACBIAS	I	O	O
RFB_TE_VSYNC0	DISPC_DATA_LCD[16]	DSS.D[16]	I	O	I
RFB_TE_VSYNC1	DISPC_DATA_LCD[18]	DSS.D[18]	I	O	I
RFB_HSYNC0	DISPC_DATA_LCD[17]	DSS.D[17]	I	O	I
RFB_HSYNC1	DISPC_DATA_LCD[19]	DSS.D[19]	I	O	I

15.3.4 Pin List and Pad Multiplexing

Table 15−10 provides the LCD pad multiplexing. The LCD interface signals of the DSS are multiplexed at chip boundary, depending on whether the mode is RFBI or bypass. The naming convention used is for bypass mode. Modes 4 and 5 are not used.

Table 15−10. LCD Pad Multiplexing

Display Interface	**Description**	**DIR**	**OMAP 2420 Ball**	**Alternate Functions**			
				Mode 0	**Mode 1**	**Mode 2**	**Mode 3**
DSS.PCLK	LCD pixel clock	IO	W6				
DSS.HSYNC	LCD horizontal synchronization	IO	Y6				
DSS.VSYNC	LCD vertical synchronization	IO	V7				
DSS. ACBIAS	ac-bias control (STN) or pixel data enable (TFT) output; also A0 output (command/data selection) in config 1	O	W7			mcbsp2. fsx	gpio. 48
DSS.D0	LCD pixel data bit 0	IO	Y7	dss.d0			
DSS.D1	LCD pixel data bit 1	IO	P10	dss.d1			
DSS.D2	LCD pixel data bit 2	IO	V8	dss.d2			
DSS.D3	LCD pixel data bit 3	IO	Y8	dss.d3			
DSS.D4	LCD pixel data bit 4	IO	W8	dss.d4			
DSS.D5	LCD pixel data bit 5	IO	R10	dss.d5			
DSS.D6	LCD pixel data bit 6	IO	Y9	dss.d6			
DSS.D7	LCD pixel data bit 7	IO	V9	dss.d7			
DSS.D8	LCD pixel data bit 8	IO	W9	dss.d8			gpio. 38
DSS.D9	LCD pixel data bit 9	IO	P11	dss.d9			gpio. 39
DSS.D10	LCD pixel data bit 10	IO	V10	dss.d10			gpio. 40
DSS.D11	LCD pixel data bit 11	IO	Y10	dss.d11			gpio. 41
DSS.D12	LCD pixel data bit 12	IO	W10	dss.d12			gpio. 42
DSS.D13	LCD pixel data bit 13	IO	R11	dss.d13			gpio. 43
DSS.D14	LCD pixel data bit 14	IO	V11	dss.d14			gpio. 44
DSS.D15	LCD pixel data bit 15	IO	W11	dss.d15			gpio. 45

Table 15−10. LCD Pad Multiplexing (Continued)

Display Interface	Description	DIR	OMAP 2420 Ball	Alternate Functions			
				Mode 0	Mode 1	Mode 2	Mode 3
DSS.D16	LCD pixel data bit 16	IO	P12	dss.d16			gpio.46
DSS.D17	LCD pixel data bit 17	IO	R12	dss.d17			gpio.47
DSS.D18	LCD pixel data bit 18	IO	J3	gpmc.a1	dss.d18		gpio.12
			D21	uart1.cts			gpio.32
DSS.D19	LCD pixel data bit 19	IO	H4	gpmc.a2	dss.d19		gpio.11
			H21	uart1.rts		dss.d19	gpio.8
DSS.D20	LCD pixel data bit 20	O	H3	gpmc.a3	dss.d20		gpio.10
			L20	uart1.tx		dss.d20	gpio.9
DSS.D21	LCD pixel data bit 21	O	G3	gpmc.a4	dss.d21		gpio.9
			T21	uart1.rx		dss.d21	gpio.10
DSS.D22	LCD pixel data bit 22	O	F4	gpmc.a5	dss.d22		gpio.8
			M21	mcbsp2.dr		dss.d22	gpio.11
DSS.D23	LCD pixel data bit 23	O	F3	gpmc.a6	dss.d23		gpio.7
			P21	mcbsp2.clkx		dss.d23	gpio.12

Table 15−11 presents the TV pad multiplexing. These pins are the analog I/O to/from the integrated DAC, which drives the TV set.

Table 15−11. TV Pad Multiplexing

TV Interface	Description	DIR	OMAP 2420 Ball	Alternate Functions			
				Mode 0	Mode 1	Mode 2	Mode 3
TV.CVBS	Composite video output (analog)	O	AA16				
TV.VREF	Input voltage reference (analog)	I	AA18				
TV.RREF	External resistor reference (analog 4-kΩ)	I	AA14				
TV.DETECT	Pulse detection	O	E5	gpio.6			gpio.6

15.3.5 DSS Register Summary

Table 15−12 shows the base address and address space for the module instances.

Table 15−12. Instance Summary

Instance Name	Base Address	Size	Module Name
DISS1	0x4805 0000	1K byte	DSS-specific
DISC1	0x4805 0400	1K byte	Display controller
RFBI1	0x4805 0800	1K byte	RFB interface
VENC1	0x4805 0C00	1K byte	Video encoder
L4TA10	0x4805 1000	512 bytes	DSS L4 target agent
IM3	0x6800 0800	512 bytes	DSS L3 initiator agent

Table 15−13 through Table 15−18 list the instances of the DSS registers and their physical addresses. The L4TA10 and IM3 registers are the interconnect target and initiator agents, respectively, attached to the DSS module. For a description, see Section 15.6, *Display Subsystem Registers*.

Table 15−13. DISS1 Registers

Register Name	Type	Register Width (Bits)	Physical Address
DSS_REVISIONNUMBER	R	32	0x4805 0000
DSS_SYSCONFIG	RW	32	0x4805 0010
DSS_SYSSTATUS	R	32	0x4805 0014
DSS_CONTROL	RW	32	0x4805 0040
DSS_PSA_LCD_REG_1	R	32	0x4805 0050
DSS_PSA_LCD_REG_2	R	32	0x4805 0054
DSS_PSA_VIDEO_REG	R	32	0x4805 0058
DSS_STATUS	R	32	0x4805 005C

Table 15−14. DISC1 Registers

Register Name	Type	Register Width (Bits)	Physical Address
DISPC_REVISION	R	32	0x4805 0400
DISPC_SYSCONFIG	RW	32	0x4805 0410
DISPC_SYSSTATUS	R	32	0x4805 0414
DISPC_IRQSTATUS	RW	32	0x4805 0418
DISPC_IRQENABLE	RW	32	0x4805 041C
DISPC_CONTROL	RW	32	0x4805 0440
DISPC_CONFIG	RW	32	0x4805 0444
DISPC_CAPABLE	RW	32	0x4805 0448
DISPC_DEFAULT_COLOR0 – DISPC_DEFAULT_COLOR1	RW	32	0x4805 044C– 0x4805 0450
DISPC_TRANS_COLOR0 – DISPC_TRANS_COLOR1	RW	32	0x4805 0454– 0x4805 0458
DISPC_LINE_STATUS	R	32	0x4805 045C
DISPC_LINE_NUMBER	RW	32	0x4805 0460
DISPC_TIMING_H	RW	32	0x4805 0464
DISPC_TIMING_V	RW	32	0x4805 0468
DISPC_POL_FREQ	RW	32	0x4805 046C
DISPC_DIVISOR	RW	32	0x4805 0470
DISPC_SIZE_DIG	RW	32	0x4805 0478
DISPC_SIZE_LCD	RW	32	0x4805 047C
DISPC_GFX_FIFO_THRESHOLD	RW	32	0x4805 04A4
DISPC_GFX_FIFO_SIZE_STATUS	R	32	0x4805 04A8
DISPC_GFX_ROW_INC	RW	32	0x4805 04AC
DISPC_GFX_PIXEL_INC	RW	32	0x4805 04B0
DISPC_GFX_WINDOW_SKIP	RW	32	0x4805 04B4

Table 15−14. DISC1 Registers (Continued)

Register Name	Type	Register Width (Bits)	Physical Address
DISPC_GFX_TABLE_BA	RW	32	0x4805 04B8
DISPC_VID1_BA0 – DISPC_VID1_BA1	RW	32	0x4805 04BC– 0x4805 04C0
DISPC_VID1_POSITION	RW	32	0x4805 04C4
DISPC_VID1_SIZE	RW	32	0x4805 04C8
DISPC_VID1_ATTRIBUTES	RW	32	0x4805 04CC
DISPC_VID1_FIFO_THRESHOLD	RW	32	0x4805 04D0
DISPC_VID1_FIFO_SIZE_STATUS	R	32	0x4805 04D4
DISPC_VID1_ROW_INC	RW	32	0x4805 04D8
DISPC_VID1_PIXEL_INC	RW	32	0x4805 04DC
DISPC_VID1_FIR	RW	32	0x4805 04E0
DISPC_VID1_PICTURE_SIZE	RW	32	0x4805 04E4
DISPC_VID1_ACCU0 – DISPC_VID1_ACCU1	RW	32	0x4805 04E8– 0x4805 04EC
DISPC_VID1_FIR_COEF_H0 – DISPC_VID1_FIR_COEF_H7	RW	32	0x4805 04F0– 0x4805 0528
DISPC_VID1_FIR_COEF_HV0 – DISPC_VID1_FIR_COEF_HV7	RW	32	0x4805 04F4– 0x4805 052C
DISPC_VID1_CONV_COEF0	RW	32	0x4805 0530
DISPC_VID1_CONV_COEF1	RW	32	0x4805 0534
DISPC_VID1_CONV_COEF2	RW	32	0x4805 0538
DISPC_VID1_CONV_COEF3	RW	32	0x4805 053C
DISPC_VID1_CONV_COEF4	RW	32	0x4805 0540
DISPC_VID2_BA0 – DISPC_VID2_BA1	RW	32	0x4805 054C– 0x4805 0550
DISPC_VID2_POSITION	RW	32	0x4805 0554
DISPC_VID2_SIZE	RW	32	0x4805 0558
DISPC_VID2_ATTRIBUTES	RW	32	0x4805 055C
DISPC_VID2_FIFO_THRESHOLD	RW	32	0x4805 0560
DISPC_VID2_FIFO_SIZE_STATUS	R	32	0x4805 0564
DISPC_VID2_ROW_INC	RW	32	0x4805 0568
DISPC_VID2_PIXEL_INC	RW	32	0x4805 056C
DISPC_VID2_FIR	RW	32	0x4805 0570
DISPC_VID2_PICTURE_SIZE	RW	32	0x4805 0574
DISPC_VID2_ACCU0 – DISPC_VID2_ACCU1	RW	32	0x4805 0578– 0x4805 057C

Table 15−14. DISC1 Registers (Continued)

Register Name	Type	Register Width (Bits)	Physical Address
DISPC_VID2_FIR_COEF_H0 – DISPC_VID2_FIR_COEF_H7	RW	32	0x4805 0580– 0x4805 05B8
DISPC_VID2_FIR_COEF_HV0 – DISPC_VID2_FIR_COEF_HV7	RW	32	0x4805 0584– 0x4805 05BC
DISPC_VID2_CONV_COEF0	RW	32	0x4805 05C0
DISPC_VID2_CONV_COEF1	RW	32	0x4805 05C4
DISPC_VID2_CONV_COEF2	RW	32	0x4805 05C8
DISPC_VID2_CONV_COEF3	RW	32	0x4805 05CC
DISPC_VID2_CONV_COEF4	RW	32	0x4805 05D0
DISPC_DATA_CYCLE1	RW	32	0x4805 05D4
DISPC_DATA_CYCLE2	RW	32	0x4805 05D8
DISPC_DATA_CYCLE3	RW	32	0x4805 05DC

Table 15−15. RFBI1 Registers

Register Name	Type	Register Width (Bits)	Physical Address
RFBI_REVISION	R	32	0x4805 0800
RFBI_SYSCONFIG	RW	32	0x4805 0810
RFBI_SYSSTATUS	R	32	0x4805 0814
RFBI_CONTROL	RW	32	0x4805 0840
RFBI_PIXEL_CNT	RW	32	0x4805 0844
RFBI_LINE_NUMBER	RW	32	0x4805 0848
RFBI_CMD	W	32	0x4805 084C
RFBI_PARAM	W	32	0x4805 0850
RFBI_DATA	W	32	0x4805 0854
RFBI_READ	RW	32	0x4805 0858
RFBI_STATUS	RW	32	0x4805 085C
RFBI_CONFIG0 – RFBI_CONFIG1	RW	32	0x4805 0860– 0x4805 0878
RFBI_ONOFF_TIME0 – RFBI_ONOFF_TIME1	RW	32	0x4805 0864– 0x4805 087C
RFBI_CYCLE_TIME0 – RFBI_CYCLE_TIME1	RW	32	0x4805 0868– 0x4805 0880
RFBI_DATA_CYCLE1_0 – RFBI_DATA_CYCLE1_1	RW	32	0x4805 086C– 0x4805 0884
RFBI_DATA_CYCLE2_0 – RFBI_DATA_CYCLE2_1	RW	32	0x4805 0870– 0x4805 0888
RFBI_DATA_CYCLE3_0 – RFBI_DATA_CYCLE3_1	RW	32	0x4805 0874– 0x4805 088C

Table 15−15. RFBI1 Registers (Continued)

Register Name	Type	Register Width (Bits)	Physical Address
RFBI_VSYNC_WIDTH	RW	32	0x4805 0890
RFBI_HSYNC_WIDTH	RW	32	0x4805 0894

Table 15−16. VENC1 Registers

Register Name	Type	Register Width (Bits)	Physical Address
REV_ID	R	32	0x4805 0C00
STATUS	R	32	0x4805 0C04
F_CONTROL	RW	32	0x4805 0C08
VIDOUT_CTRL	RW	32	0x4805 0C10
SYNC_CTRL	RW	32	0x4805 0C14
LLEN	RW	32	0x4805 0C1C
FLENS	RW	32	0x4805 0C20
HFLTR_CTRL	RW	32	0x4805 0C24
CC_CARR_WSS_CARR	RW	32	0x4805 0C28
C_PHASE	RW	32	0x4805 0C2C
GAIN_U	RW	32	0x4805 0C30
GAIN_V	RW	32	0x4805 0C34
GAIN_Y	RW	32	0x4805 0C38
BLACK_LEVEL	RW	32	0x4805 0C3C
BLANK_LEVEL	RW	32	0x4805 0C40
X_COLOR	RW	32	0x4805 0C44
M_CONTROL	RW	32	0x4805 0C48
BSTAMP_WSS_DATA	RW	32	0x4805 0C4C
S_CARR	RW	32	0x4805 0C50
LINE21	RW	32	0x4805 0C54
LN_SEL	RW	32	0x4805 0C58
L21__WC_CTL	RW	32	0x4805 0C5C
HTRIGGER_VTRIGGER	RW	32	0x4805 0C60
SAVID__EAVID	RW	32	0x4805 0C64
FLEN__FAL	RW	32	0x4805 0C68
LAL__PHASE_RESET	RW	32	0x4805 0C6C
HS_INT_START_STOP_X	RW	32	0x4805 0C70
HS_EXT_START_STOP_X	RW	32	0x4805 0C74
VS_INT_START_X	RW	32	0x4805 0C78
VS_INT_STOP_X__VS_INT_START_Y	RW	32	0x4805 0C7C

Table 15−16. VENC1 Registers (Continued)

Register Name	Type	Register Width (Bits)	Physical Address
VS_INT_STOP_Y__VS_EXT_START_X	RW	32	0x4805 0C80
VS_EXT_STOP_X__VS_EXT_START_Y	RW	32	0x4805 0C84
VS_EXT_STOP_Y	RW	32	0x4805 0C88
AVID_START_STOP_X	RW	32	0x4805 0C90
AVID_START_STOP_Y	RW	32	0x4805 0C94
FID_INT_START_X__FID_INT_START_Y	RW	32	0x4805 0CA0
FID_INT_OFFSET_Y__FID_EXT_START_X	RW	32	0x4805 0CA4
FID_EXT_START_Y__FID_EXT_OFFSET_Y	RW	32	0x4805 0CA8
TVDETGP_INT_START_STOP_X	RW	32	0x4805 0CB0
TVDETGP_INT_START_STOP_Y	RW	32	0x4805 0CB4
GEN_CTRL	RW	32	0x4805 0CB8
DAC_TST__DAC_A	RW	32	0x4805 0CC4
DAC_B__DAC_C	RW	32	0x4805 0CC8

Table 15−17. L4TA10 Registers

Register Name	Type	Register Width (Bits)	Physical Address
COMPONENT	R	32	0x4805 1000
AGENT_CONTROL	RW	32	0x4805 1020
AGENT_STATUS	R	32	0x4805 1028

Table 15–18. IM3 Registers

Register Name	Type	Register Width (Bits)	Physical Address
SBIMERRLOGA	R	32	0x6800 08A8
SBIMERRLOG	RW	32	0x6800 08B0
SBIMSTATE	RW	32	0x6800 0990
SBTMSTATE_L	RW	32	0x6800 0998
SBTMSTATE_H	RW	32	0x6800 099C
SBIMCONFIG_L	RW	32	0x6800 09A8
SBIMCONFIG_H	RW	32	0x6800 09AC
SBID_L	R	32	0x6800 09F8
SBID_H	R	32	0x6800 09FC

Table 15–19 lists the programming equations for the DISPC_GFX_WINDOW_SKIP overlay optimization.

Table 15–19. Overlay Optimization; DISPC_GFX_WINDOW_SKIP Programming (Equations)

Ref.	X equals or Y equals	DISPC_GFX_WINDOW_SKIP[2] equals
1a	X = DISPC_VID1.VidPosX – DISPC_GFX.GfxPosX + DISPC_VID1.VidSizeX + 1	= X * (DISPC_GFX.GfxPixelInc – 1 + Bpp[1])
1b	X = DISPC_GFX.GfxPosX – DISPC_VID1.VidPosX + DISPC_GFX.GfxSizeX + 1	
2	X = DISPC_VID1.VidSizeX + 1	= X * (DISPC_GFX.GfxPixelInc –1 + Bpp[1]) + 1
3a	Y = DISPC_VID1.VidPosY + DISPC_VID1.VidSizeY – DISPC_GFX.GfxPosY + 1	= Y * [(DISPC_GFX.GfxSizeX * (DISPC_GFX.GfxPixelInc – 1 + Bpp[1])) + (DISPC_GFX.GfxRowInc – 1 + Bpp)]
3b	Y = DISPC_VID1.VidSizeY + 1	
3c	Y = DISPC_GFX.GfxPosY + DISPC_GFX.GfxSizeY – DISPC_VID1.VidPosY + 1	

1) Bpp defines the number of bytes-per-pixel for the graphics buffer.

2) When using the overlay optimization feature, the following rules apply:

   a. Enable the overlay optimization (*DISPC_CONTROL[12]: OVERLAYOPTIMIZATION* = *1*). This feature must be enabled only when an overlay area exists between VID1 and GFX.

   b. Video1 must be enabled and configured (size, position, etc.) if the overlay optimization is turned on (that is, *DISPC_VID1_ATTRIBUTES[0]: VIDENABLE* = *1*).

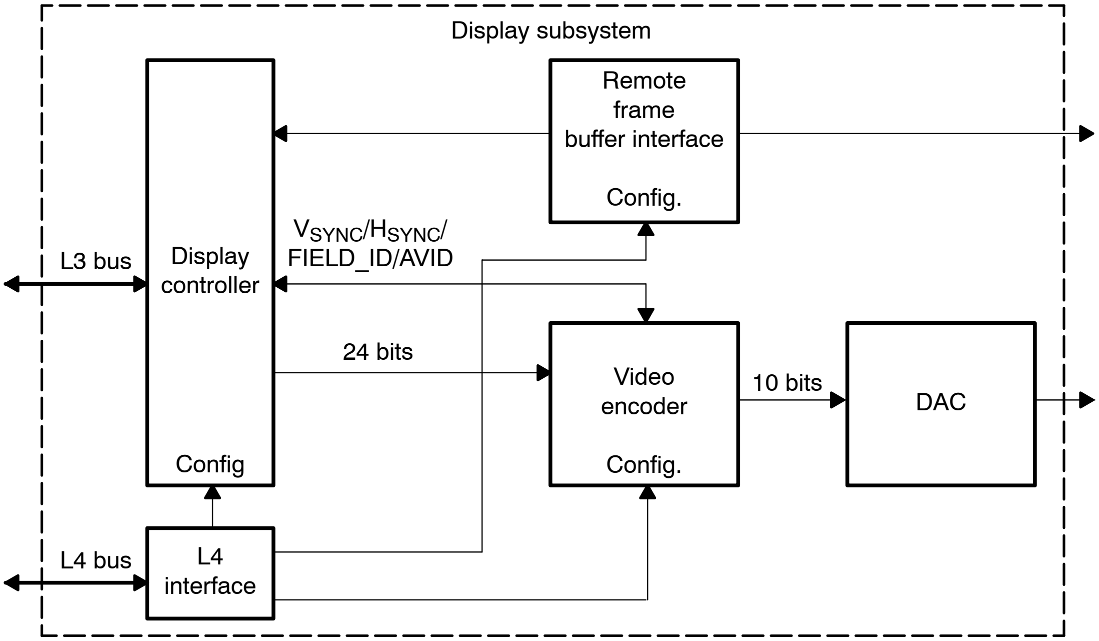
   c. The transparency color key feature must be disabled (that is, DISPC_TRANS_COLOR0 must be set to 0, and *DISPC_CONFIG[10]: TCKLCDENABLE* = *0*).

   d. Program the DISPC_GFX_WINDOW_SKIP as explained in Table 15–19.

15.4 Display Subsystem Functional Description

This section discusses the functions of the LCD and TV display supports by describing the three main subblocks: the display controller, RFBI, and VENC. The functions of the display controller are common to the LCD and TV data paths, while the RFBI functions are specific to the LCD, and the VENC functions are specific to the TV set (see Figure 15−21).

Figure 15−21. DSS Full Schematic



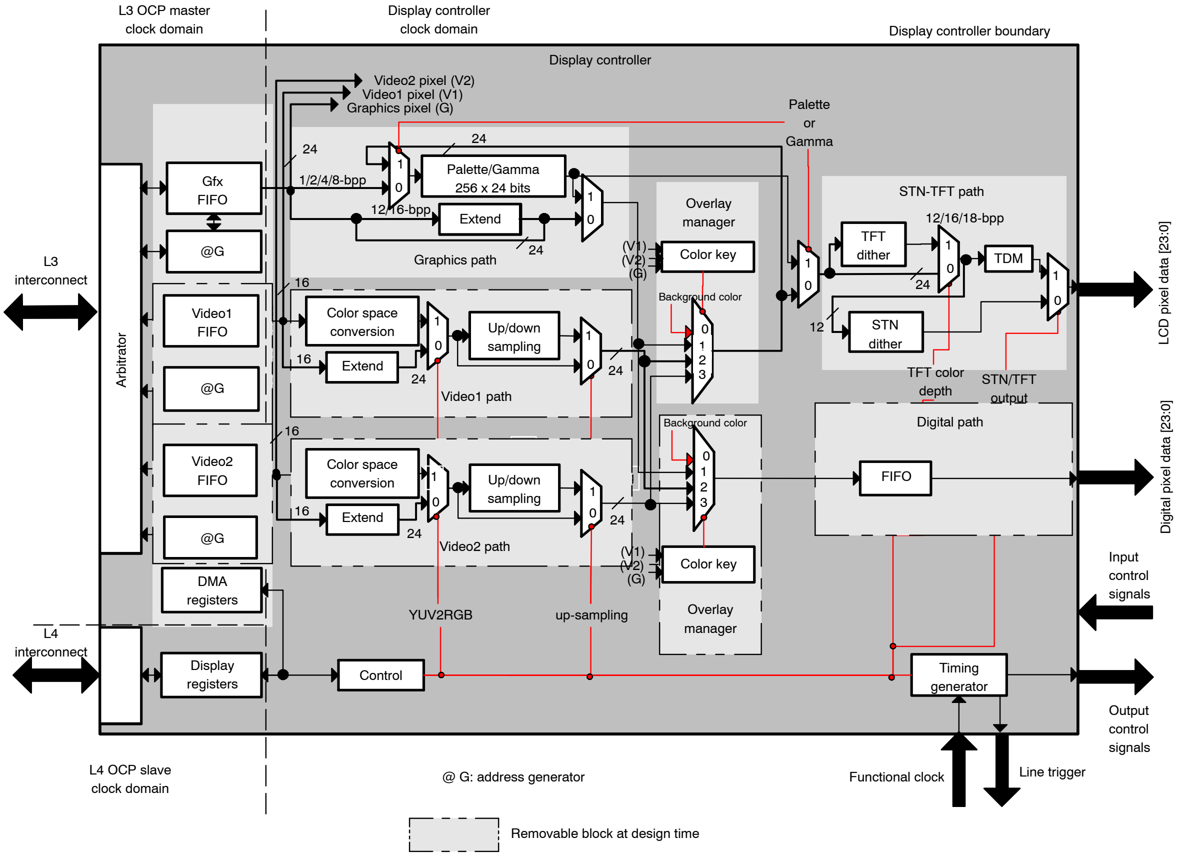
15.4.1 Display Controller Functions

The display controller can read and display the encoded pixel data stored in memory (see Figure 15−22).

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 appropriate 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 SDRAM or SRAM.

The input DMA FIFO size for graphics and video data are defined at design time, depending on system constraints.

Figure 15−22. Display Controller Architecture Overview



15.4.1.1 DMA Engine

The DMA engine requests data (encoded pixel data, palette, and gamma curve) from the L3-based interconnect according to the configuration of the display controller registers.

One FIFO is dedicated to each pipeline. The graphics FIFO is dedicated to the graphics pipeline: graphics encoded pixel data and the palette/gamma curve table. Each video FIFO is dedicated to one video pipeline: decoded video pixel data.

The burst mode is used for fetching the palette/gamma table from the system memory when only 2-, 4-, or 8-bpp format is selected for the graphics data. The hardware uses the optimum burst size between 4x32 and 16x32 to fetch the palette buffer from the system memory. The burst configuration of the graphics pipeline is used as the maximum burst size for the palette/gamma table requests.

15.4.1.2 Display Modes

❑ LCD output is used for the LCD display support functions. Two types of display technologies are supported: passive and active panels. Both monochrome and color modes are supported.

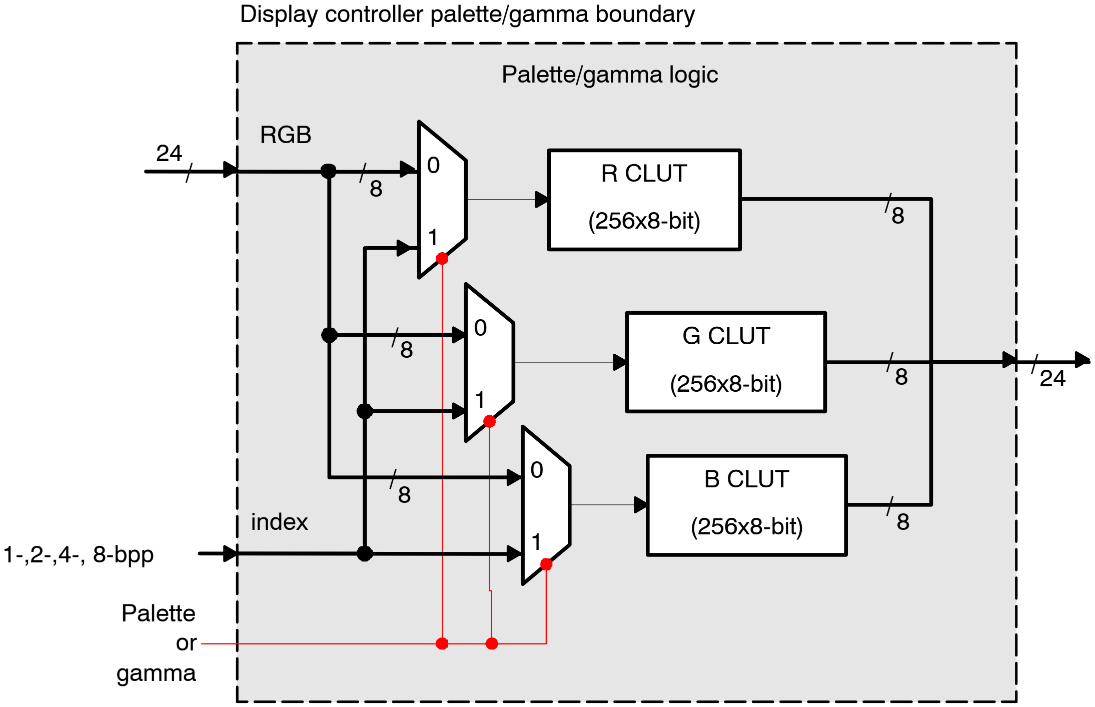
- In passive STN mode, 3,375 colors are available, which allows 16, 256, or 3,375 colors to be displayed in each frame, depending on the color depth. Fifteen grayscale levels are available for monochrome LCD.
- Regardless of color depth, active TFT mode supports 16,777,216 color display, 18 bpp supports 262,144 colors, 16 bpp supports 65,536 colors, and 12 bpp supports 4,096 colors.
- The digital output is used for the TV display support functions. The digital output is always a 24-bit RGB value based on a pixel request.

15.4.1.3 Graphics Pipeline

The graphics layer supports CLUT bitmaps (1, 2, 4, and 8 bpp) and true color bitmaps in RGB formats (12, 16, and 24 bpp in packet and nonpacket RGB24).

The graphics path supports the palette/gamma table. Figure 15−23 shows the internal architecture of the CLUT/gamma table. The palette is split into three memories of 256x8-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.

Figure 15−23. Palette/Gamma Correction Architecture



Color Look-Up Table

In palette mode, the encoded pixel values from the input graphics FIFO are used as pointers to index the 24-bit wide palette. 1-bpp pixels address two 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 STN dithering circuit or to the color time base TFT dithering circuit.

In color mode, the value in 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).

In monochrome mode, only one 8-bit value is present.

The numbers obtained after passing through the pallette are 256 grayscales and 16,777,216 colors. A redundancy introduced in the dithering logic step reduces these numbers when displaying. For STN panels, the colors are limited to 15 grayscales and 3,375 colors. For more detail, see Section 15.4.1.6, *STN Dithering Logic*.

- 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, which allows up to 2^{24} = 16,777,216 colors to be displayed.

Gamma Curve Support

In 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.

Each component of the 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 the R, G, or B component.

Graphics Memory Formats

The graphics layer supports CLUT bitmaps (1, 2, 4, and 8 bpp) and true color bitmaps in RGB formats (12, 16, and 24 bpp in packet and nonpacket RGB24).

1-bpp Data Memory Organization (Color Look-Up Table) (Little 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
P31	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0

1-bpp Data Memory Organization (Color Look-Up Table) (Little Endian + Nibble Mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P24	P25	P26	P27	P28	P29	P30	P31	P16	P17	P18	P19	P20	P21	P22	P23	P8	P9	P10	P11	P12	P13	P14	P15	P0	P1	P2	P3	P4	P5	P6	P7

1-bpp Data Memory Organization (Color Look-Up Table) (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
P7	P6	P5	P4	P3	P2	P1	P0	P15	P14	P13	P12	P11	P10	P9	P8	P23	P22	P21	P20	P19	P18	P17	P16	P31	P30	P29	P28	P27	P26	P25	P24

1-bpp Data Memory Organization (Color Look-Up Table) (Big Endian + Nibble Mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20	P21	P22	P23	P24	P25	P26	P27	P28	P29	P30	P31

2-bpp Data Memory Organization (Color Look-Up Table) (Little 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
P15		P14		P13		P12		P11		P10		P9		P8		P7		P6		P5		P4		P3		P2		P1		P0	

2-bpp Data Memory Organization (Color Look-Up Table) (Little Endian + Nibble Mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P12		P13		P14		P15		P8		P9		P10		P11		P4		P5		P6		P7		P0		P1		P2		P3	

2-bpp Data Memory Organization (Color Look-Up Table) (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
P3		P2		P1		P0		P7		P6		P5		P4		P11		P10		P9		P8		P15		P14		P13		P12	

2-bpp Data Memory Organization (Color Look-Up Table) (Big Endian + Nibble Mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P0		P1		P2		P3		P4		P5		P6		P7		P8		P9		P10		P11		P12		P13		P14		P15	

4-bpp Data Memory Organization (Color Look-Up Table) (Little 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
Pixel 7				Pixel 6				Pixel 5				Pixel 4				Pixel 3				Pixel 2				Pixel 1				Pixel 0			

4-bpp Data Memory Organization (Color Look-Up Table) (Little Endian + Nibble Mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 6				Pixel 7				Pixel 4				Pixel 5				Pixel 2				Pixel 3				Pixel 0				Pixel 1			

4-bpp Data Memory Organization (Color Look-Up Table) (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
Pixel 1				Pixel 0				Pixel 3				Pixel 2				Pixel 5				Pixel 4				Pixel 7				Pixel 6			

4-bpp Data Memory Organization (Color Look-Up Table) (Big Endian + Nibble Mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 0				Pixel 1				Pixel 2				Pixel 3				Pixel 4				Pixel 5				Pixel 6				Pixel 7			

8-bpp Data Memory Organization (Color Look-Up Table) (Little 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
Pixel 3								Pixel 2								Pixel 1								Pixel 0							

8-bpp Data Memory Organization (Color Look-Up Table) (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
Pixel 0								Pixel 1								Pixel 2								Pixel 3							

12-bpp Data Memory Organization (Little 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				R1				G1				B1				Unused				R0				G0				B0			

12-bpp Data Memory Organization (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				R0				G0				B0				Unused				R1				G1				B1			

16-bpp Data Memory Organization (Little 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
R1					G1						B1					R0					G0						B0				

16-bpp Data Memory Organization (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
R0					G0						B0					R1					G1						B1				

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								R								G								B							

24-bpp Packet 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	
B1								R0								G0								B0								+0x0
G2								B2								R1								G1								+0x4
R3								G3								B3								R2								+0x8

15.4.1.4 Video Pipeline

The video layer supports these five formats: YUV2, UYVY, RGB16, and RGB24 nonpacket and packet.

Color Space Conversion

The video path has a color space conversion module that converts the video encoded pixel values from YCbCr4:2:2 format into RGB24 format.

Up/Downsampling

The video layer also has a dedicated resizer to upsample and downsample the video encoded pixels. The format supported is RGB24. The pre-requirement to the resizing process is to convert the encoded video pixel values from YCbCr format to RGB format. Set the right size and position of the original video before resizing the upsampled/downsampled video to fit the display screen boundaries.

The filtering applies to each component independently (R,G, and B).

For horizontal up-/downsampling, the equation for the R component is:

$$Rout(n) = \left(\sum_{i=-2}^{i=2} Ci(\Phi) \times Rin(n+i) \right) >> 7$$

For vertical up-/downsampling, the equation for the R component is:

$$Rout(n) = \left(\sum_{i=-1}^{i=1} Ci(\Phi) \times Rin(n+i) \right) >> 7$$

Rout: R component output

$C_hi(\phi)$: FIR coefficients

Rin: R component input

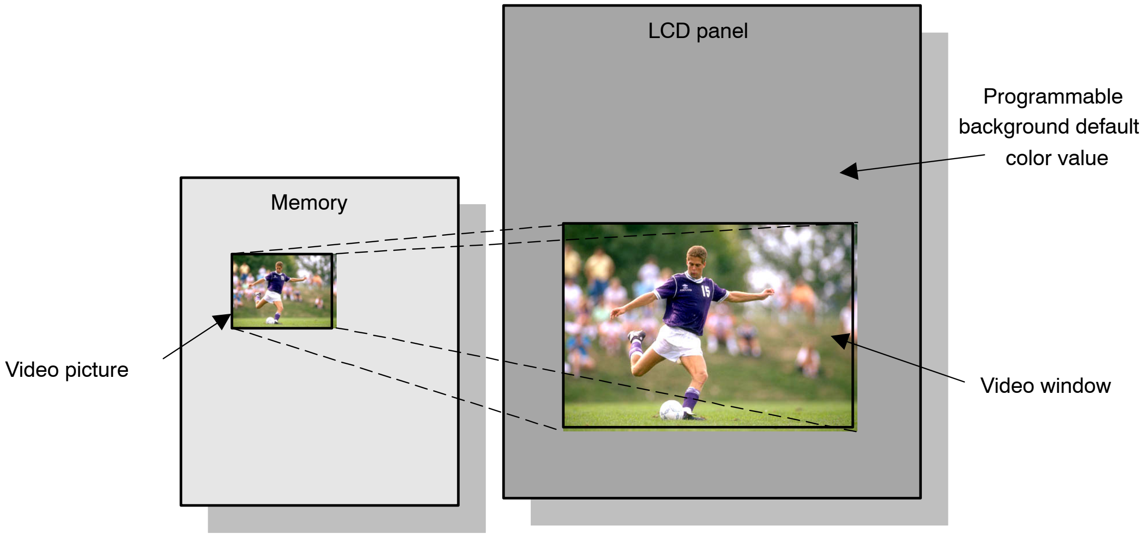
Pixel (n+1) is older than pixel (n).

Line (n+1) is older than line (n).

Note: The $Ci(\phi)$ coefficients depend on the phase between input and output pixels.

Figure 15−24 shows video upsampling.

Figure 15–24. Video Upsampling



Video Memory Formats

The video layer supports these five formats: YUV2, UYVY, RGB16, and RGB24 non-packet and packet.

16-bpp Data Memory Organization (Little 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
R1					G1						B1					R0					G0						B0				

16-bpp Data Memory Organization (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
R0					G0						B0					R1					G1						B1				

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								R								G								B							

24-bpp Packet 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	
B1								R0								G0								B0								+0x0
G2								B2								R1								G1								+0x4
R3								G3								B3								R2								+0x8

UYVY 4:2:2 Data Memory Organization (Little 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
Y1								Cr0								Y0								Cb0							

UYVY 4:2:2 Data Memory Organization (Big Endian)

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1										
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
Cb0								Y0								Cr0								Y1							

YUV2 4:2:2 Data Memory Organization (Little Endian)

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1										
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
Cr0								Y1								Cb0								Y0							

UYVY 4:2:2 Data Memory Organization (Big Endian)

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1										
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
Y0								Cb0								Y1								Cr0							

15.4.1.5 Overlay Support

The overlay mechanism consists of displaying more than one layer (graphics and video layers) using rules based on priority and transparency color keys.

The overlay managers are based on the same rules for priority and transparency color keys.

Each data pipeline is assigned to a unique overlay, and consequently to a unique display controller output.

Priority Rule

The video1 layer is always on top of the graphics layer. The video2 layer is always on top of the video1 and graphics layers. The display controller reads the data for each buffer from the system memory, and depending on the transparency color key values, displays 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 a full display screen. If there are no graphics or video encoded pixels at a specific position, the programmable solid background color is displayed.

Figure 15−25 shows the overlay manager. Figure 15−26 shows the display attributes.

Figure 15−25. Overlay Manager

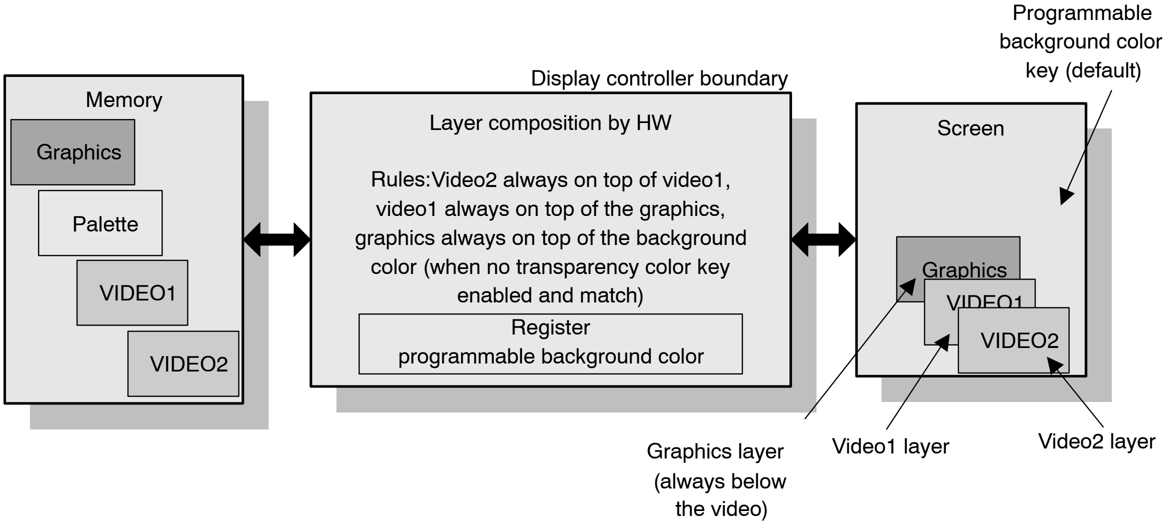
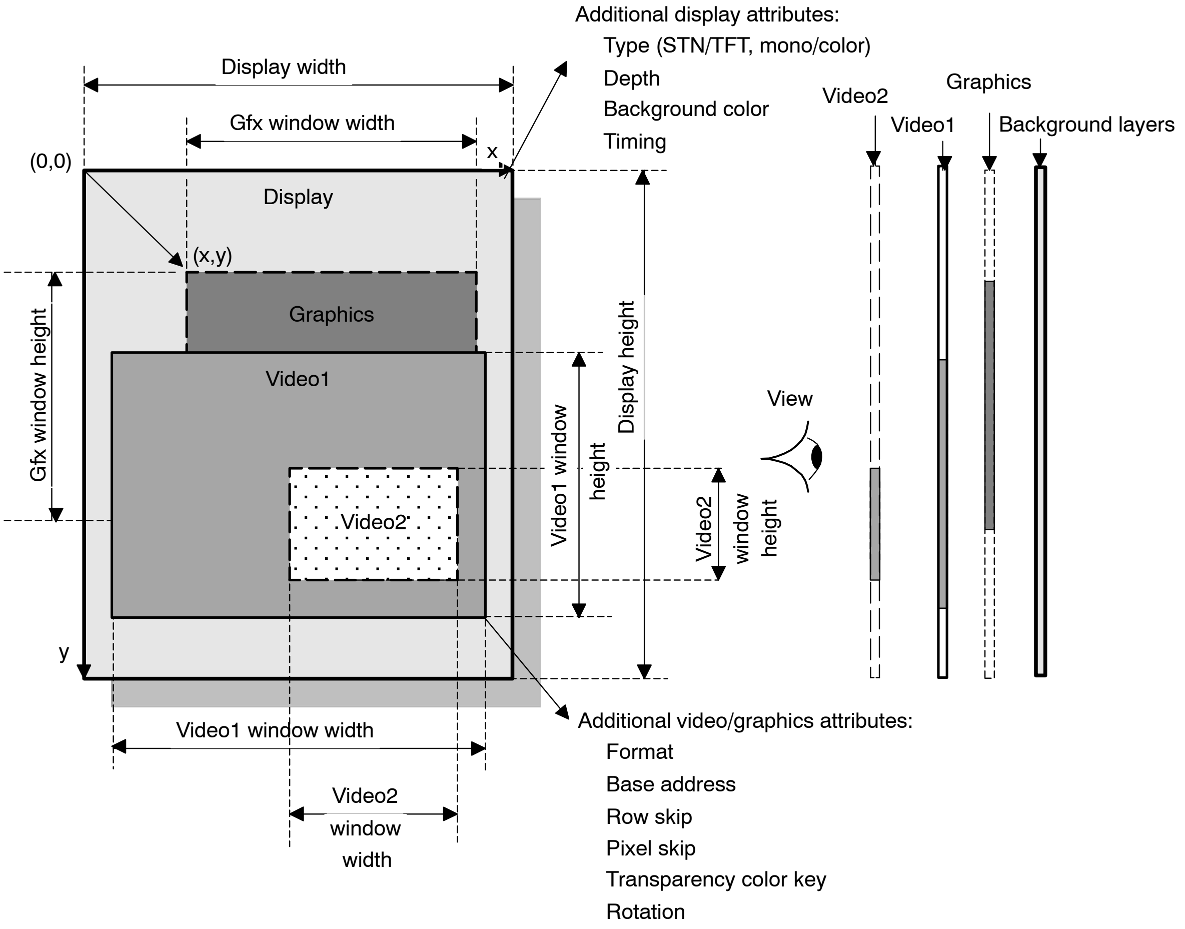

Memory
Graphics
Palette
VIDEO1
VIDEO2
Display controller boundary
Layer composition by HW
Rules:Video2 always on top of video1, video1 always on top of the graphics, graphics always on top of the background color (when no transparency color key enabled and match)
Register
programmable background color
Screen
Programmable background color key (default)
Graphics
VIDEO1
VIDEO2
Graphics layer (always below the video)
Video1 layer
Video2 layer

Figure 15−26. Display Attributes



Enabling overlay optimization (setting the DISPC_CONTROL [12]: OVERLAYOPTIMIZATION bit) if no overlay region exists (DISPC_VIDn_ATTRIBUTES [0]: VIDENABLE = 0) leads to unpredictable behavior. The overlay optimization feature must be enabled only when an overlay area exists.

Transparency Color Key

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, not to the palette value pointed out by the palette index.

Note:

The video source transparency color key and the graphics destination transparency color key cannot be active at the same time.

- ❑ Video source transparency color key: The source transparency color key value defines the encoded pixel data considered as transparent pixel. The encoded pixel values with the source color key value are not visible and the under-layer encoded pixel values or solid background color are visible. The 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 is used for handling the hardware cursor displayed by one of the video layers).
- ❑ Destination transparency color key: The 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 because pixels at the same position in the video layers are visible. 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.

15.4.1.6 STN Data Path

STN Dithering Logic

- ❑ Passive matrix technology: When 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 is encoded on 4 bits, which are the 4 most-significant bits (MSBs) of the pixel encoded component 8-bit value defined by the merging 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 that the pixel is off longer than the average time that it is on produces 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 match the visual perception of color/gray graduations.

- Active matrix technology: The STN dithering logic is always bypassed in active displays.

STN Output FIFO

- Passive matrix technology: The display controller contains a 2-entry by 8-bit-wide output FIFO that is 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 TFT mode.

15.4.1.7 TFT Data Path

- Passive matrix technology: The STN dithering logic path is used. The TFT dithering logic is not output to the LCD panel.
- Active matrix technology: The encoded pixel values are used by the TFT dithering logic to display the data in lower color depth on the LCD panel.

15.4.1.8 Multiple Cycle Output Format (TDM)

After TFT processing, the pixels are formatted on one or multiple cycles (from one to three cycles). The interface width can be 8, 9, 12, or 16 bits. On three cycles, two pixels can be concatenated and sent to the panel. When the TDM is disabled, the display controller outputs the pixels using the conventional formats: STN/TFT monochrome/color. The following is an example of an output configuration based on the interface width (8-bit) and the pixel format output (24-bit).

24-Bit Mode

DISPC TDMCYCLEFORMAT DISPC_CONTROL[24:23] = 0x2

DISPC DISPC_DATA_CYCLE1 = 0x00000008

DISPC DISPC_DATA_CYCLE2 = 0x00000008

DISPC DISPC_DATA_CYCLE3 = 0x00000008

Table 15−20 lists the 8-bit interface configuration in 24-bit mode.

Table 15−20. 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]

15.4.1.9 Rotation

For the SRAM buffer, the display controller handles the rotation in the DMA and accesses the encoded pixels in burst considering always consecutive data in memory.

The rotation engine in the SDRAM scheduler translates the addresses from virtual to physical SDRAM addresses (see Chapter 12, *Memory Subsystem*). The display controller cannot rotate encoded pixel values in 1-, 2-, and 4-bpp formats in SRAM or SDRAM.

15.4.2 Remote Frame Buffer Interface Functions

The RFBI module can capture the output pixel from the display controller and send the data to the RFB in the LCD panel.

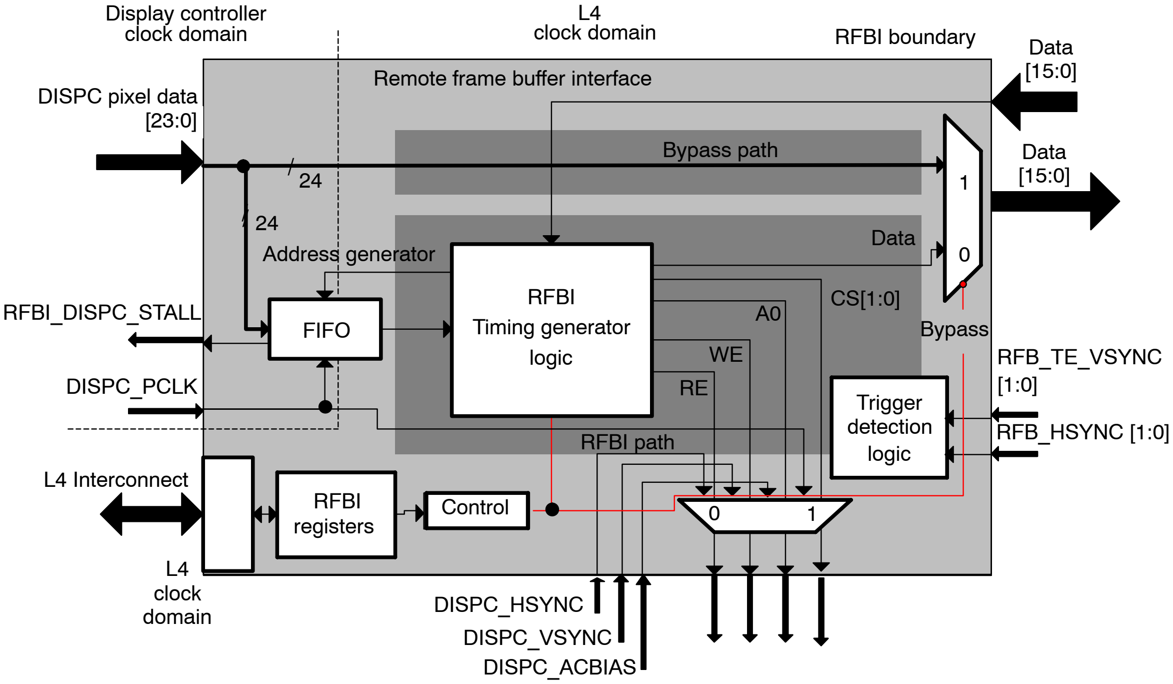
The application configures the RFBI module, sends commands, reads data, and configures the display controller to send data fetched from the system memory by the display controller DMA engine.

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 FIFO, the encoded pixel values are in the least-significant bit (LSB) aligned independently of the endianness in system memory.

Figure 15−27 is an overview of the RFBI architecture.

Figure 15−27. RFBI Architecture Overview



15.4.2.1 FIFO

The input FIFO receives data from the display controller at the pixel clock. The data in the FIFO are read by the RFBI and sent to the LCD panel. The width of the FIFO is 24 bits, and each pixel in 12-, 16-, 18-, and 24-bpp format is stored in the FIFO using one 24-bit value aligned on the 24-bit LSB. Table 15−21 is an example of output configuration based on the interface width (16-bit) and the pixel format output (24-bit).

15.4.2.2 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 according to the configuration of the output interfaces (multiple cycles).

15.4.2.3 Output Parallel Modes

The RFBI output modes are 8-, 9-, 12-, and 16-bit interfaces. Any mode of the pixel format can be selected independently. Set the correct configuration in the cycle registers to define a valid configuration for each output cycle.

24-Bit Mode

RFBI CYCLEFORMAT RFBI_CONFIGi[10:9] = 0x3
RFBI RFBI_DATAi_CYCLE1 = 0x00000010
RFBI RFBI_DATAi_CYCLE2 = 0x00080808
RFBI RFBI_DATAi_CYCLE3 = 0x00100000

Table 15−21 lists the 16-bit interface configuration in 24-bit mode.

Table 15−21. 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]

15.4.2.4 Unmodified Bits

In a cycle, if not every bit of the interface has 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).

15.4.2.5 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 module.

15.4.2.6 Send Commands

The commands are written through the L4 interconnect into the RFBI command register RFBI_CMD. When one command is sent, another one can be accepted by the module and set. If the command has not been processed, the MPU access to change the command is stalled.

15.4.2.7 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/WE/RE/CS is programmable.

Table 15−22 describes the read/write function.

Table 15−22. Read/Write Function Description

A0	WE	RE	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

Note: 1 = deasserted, 0 = asserted (all signals active low)

15.4.3 Video Encoder Functions

The following input formats are supported by the encoders:

- ☐ 24-bit 4:4:4 RGB
- ☐ 24-bit 4:4:4 YCbCr
- ☐ 16-bit 4:2:2 YCbCr
- ☐ 8-bit 4:2:2 YCbCr

The encoder output is the DAC (for details, see Section 15.2, *Display Subsystem Environment*). In the DSS, 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 decimator, which reduces the chrominance bandwidth and the amount of chrominance data by half. After the data manager, the encoder processes in 4:2:2 data path up to the 2x interpolator. A luma delay synchronizes luma to chrominance data.

15.4.3.1 Test Pattern Generation

For diagnostic purposes, the data manager can be forced to output 100/100 color bar RGB/YCbCr data by setting the CBAR register (see Table 15−23).

Table 15−23. 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

15.4.3.2 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 GAIN_Y register. Horizontal sync, vertical sync, and setup insertion are then performed. Both black level and blank level are programmable through the BLACK_LEVEL and 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 the necessary components of the luminance signal have been added, the resultant 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.

15.4.3.3 Chroma Stage

The chroma stage includes a low-pass filter, 1st stage 2x interpolator, chroma gain shaper, and 2nd stage 2x interpolator. A pair of programmable gains adjusts the time-multiplexed U/V signal. The gains for U and V are independently controlled by the GAIN_U and GAIN_V register bits.

15.4.3.4 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:

$$S\_CARR = ROUND\ ((Fsc/Fclkenc) \times 2^{32})$$

where Fsc = Frequency of the subcarrier

Fclkenc = Frequency of the internal video encoder

The S_CARR registers control the subcarrier frequency. The C_PHASE register controls the phase of the subcarrier. The phase of the color subcarrier is reset to C_PHASE. Four modes of color subcarrier reset are provided: no reset, reset every two lines, reset every two fields, or reset every eight fields. The C_PHASE register can be used to adjust SCH (subcarrier to horizontal sync phase). BSTAP[6:0] of BSTAMP register sets the amplitude of the color burst. The PAL bit of the M_CONTROL register enables phase alternation line encoding. A phase-switching subcarrier is generated to encode the chrominance signal when the 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. There are two possible alternation sequences, and the PALPHS bit of the M_CONTROL register selects one of these sequences.

15.4.3.5 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 one odd-parity bit, as shown in Figure 15−28.

Figure 15−28. Closed-Caption Data Format

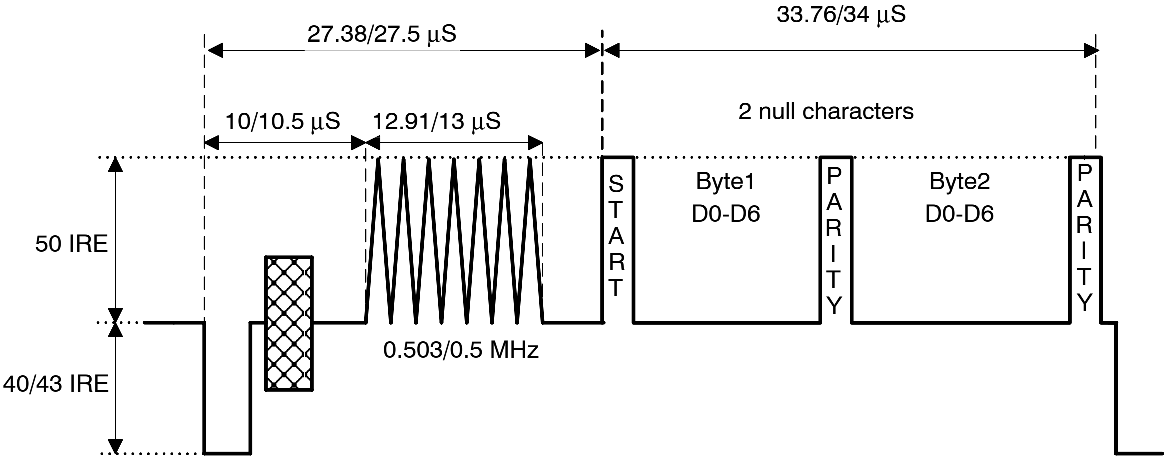
MSB							LSB
odd-parity	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Standard service encodes closed-caption in both fields, while extended service encodes closed-caption in even fields. When set to 1, L21ENA enables closed-caption encoding in odd fields, and when set to 1, L21ENB enables closed-caption encoding in even fields.

The LN_SEL register is used to program the scan line where closed-caption is to be encoded. Four closed-caption data registers contain the data to be encoded. The LINE21_O[7:0] and LINE21_O[15:8] registers contain the first and second bytes of closed-caption data to be encoded in the odd field. The LINE21_E[7:0] and LINE21_E[15:8] registers contain the first and the second bytes of data to be encoded in the even field. Immediately after the closed-caption data is written to the registers, either in the odd field or the even field, the corresponding closed-caption status bit (CCE or CCO in the STATUS register) is reset to 0 to indicate that the closed-caption data is available in the closed-caption data registers and has not yet been encoded. Immediately after the closed-caption is encoded, the CCE or CCO bit is set to 1 to indicate that the closed-caption data has been encoded and is ready to accept new data. 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 run-in clock frequency is 5034960.5 Hz (32xf_{line} of NTSC). The closed-caption data is encoded in non-return-to-zero (NRZ) format. Additionally, the data translates to IRE scale in the following manner: 0 = 0 IRE, 1 = 50 IRE. Figure 15−29 shows the parameters of closed-caption line data implemented in different standards.

Figure 15−29. 525/625-Lines 21/22 and 284/335 Closed-Caption Timing

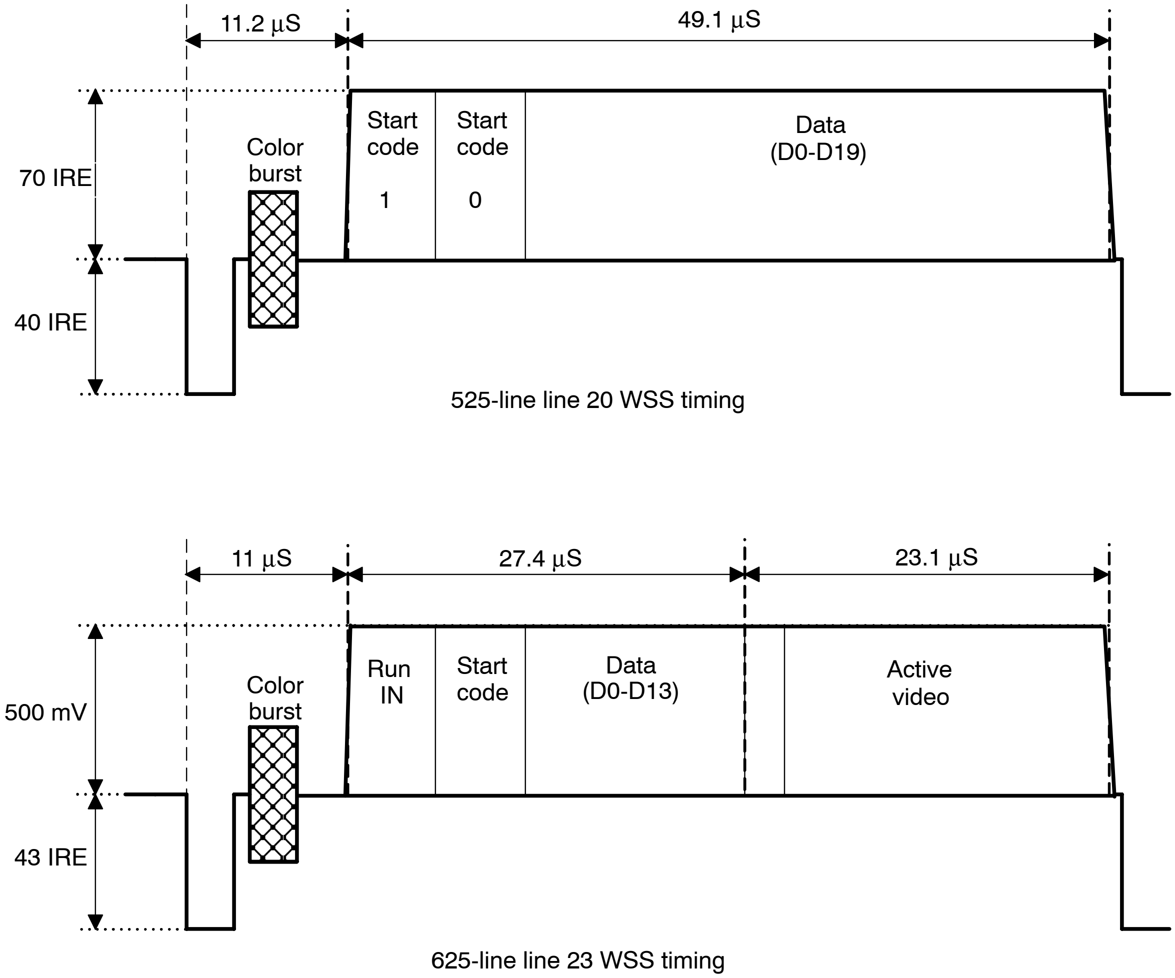


15.4.3.6 WSS Encoding

The encoder can embed data within the vertical blanking interval, encoded according to the IEC61880 and ITU-R 1119 data insertion standard.

The encoder supports WSS data insertion on line 20 of every frame in the NTSC format. WSS data insertion is enabled by activating the WSS_EN register and programming the WSS_DATA register. The WSS encoding block assumes that a full 10-bit video range is used to determine the 70% of peak-white amplitude of a logic 1 bit. The encoder also supports WSS data insertion on line 23 in PAL format. Both waveforms are shown in Figure 15−30.

Figure 15−30. WSS Timing



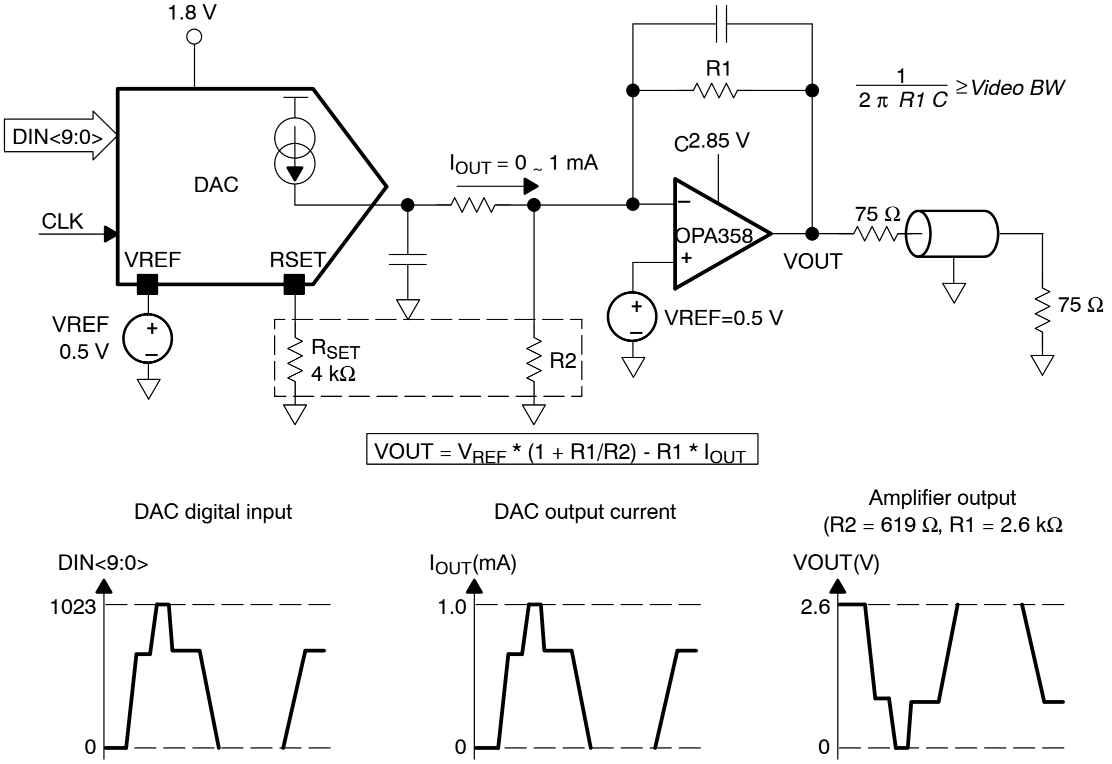
15.4.3.7 TV Detection Pulse Generation and Usage

TV Detection Pulse Generation

The TV_DET signal is used to generate a pulse that enables testing the video composite signal level. If this video composite signal level after the external amplifier and serial 75-Ω resistor is not higher than half of the blanking zone maximum voltage value, the TV is connected (load of 75 Ω). When the video composite signal level after external amplifier and serial 75-Ω resistor is close to the blanking zone maximum voltage value, the TV is disconnected (no load). This pulse is generated to ensure that the testing period of the video composite signal level occurs in the blanking zone where the video composite signal waveform is well known and predictable.

Figure 15−31 shows the video composite signal amplification proposal.

Figure 15−31. Video Composite Signal Amplification Proposal (Inverting)



CAUTION

This inverting configuration provides better linearity because the DAC output is kept at virtual ground. This configuration requires polarity inversion of the DAC digital input DIN[9:0] to obtain VOUT maximum voltage for 1023 DIN[9:0] value. DIN[9:0] polarity is inverted by setting the DAC_TST_DAC_A[3] register bit to 1 (DAC_INVT).

The following registers are used to set the TV detection pulse:

- ❑ VENC register TVDETGP_INT_START_STOP_X: Used to define which pixels are used to start and stop the pulse inside their respective line
- ❑ VENC register TVDETGP_INT_START_STOP_Y: Used to define which lines are used to start and stop the pulse
- ❑ VENC register bit GEN_CTRL[0]: Used to enable or disable the TV_DET pulse (0: disable, 1: enable)
- ❑ VENC register bit GEN_CTRL[8]: Used to set the TV_DET pulse polarity (0: active low, 1: active high)

Recommended TV Detection Pulse Waveform

Texas Instruments recommends copying the vertical synchronization pulse waveform to set the TV_DET pulse.

Using the video encoder registers, the pulse can be programmed on one of the possible video fields. The configuration is different for NTSC and PAL standards (see Figure 15−32 and Figure 15−33). In both cases, the pulse duration is 3 lines and the polarity of the pulse is programmable.

Figure 15−32. HSYNC, VSYNC, and TV_DET for NTSC (Line Numbering from ITU-R 470-4)

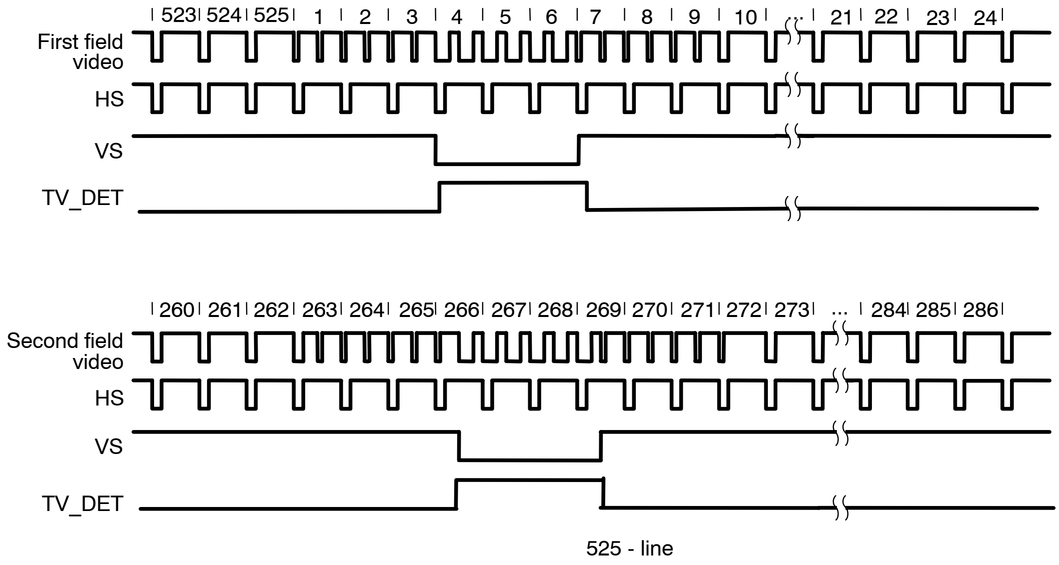
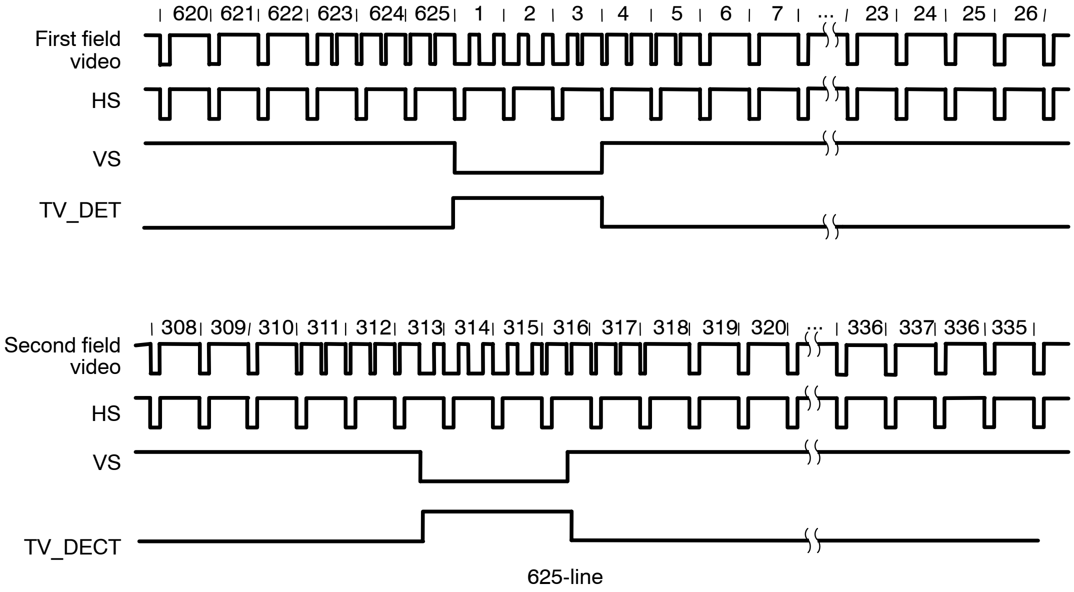


Figure 15−33. HSYNC, VSYNC, and TV_DET for PAL (Line Numbering from ITU-R 470-4)



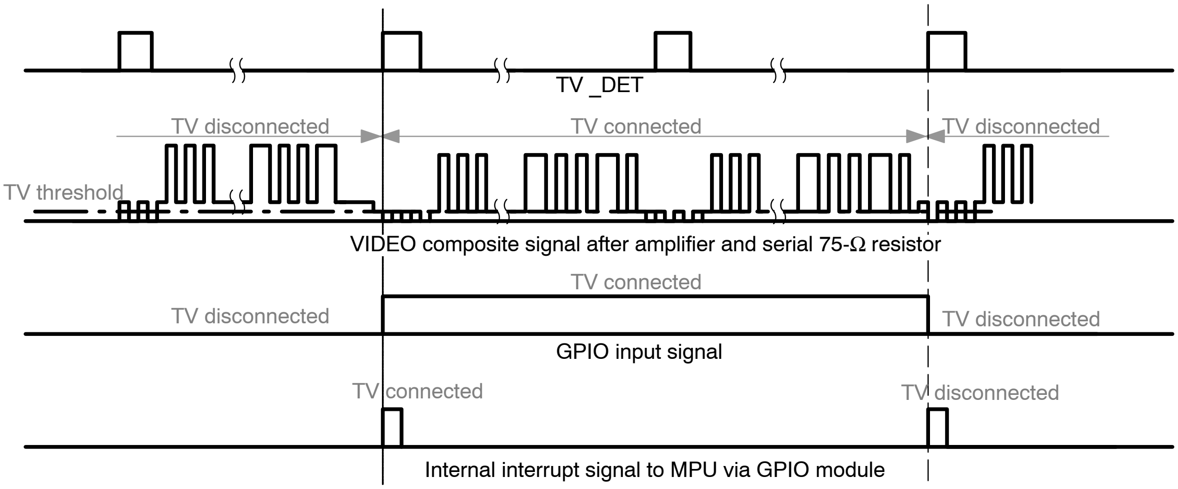
TV Detection Usage

External logic must be implemented using the video composite signal after the serial 75-Ω resistor and TV_DET signal to generate an input signal to the OMAP2420 connected to one of the GPIO pins. Then, using the GPIO signal, GPIO interrupts alert the MPU to any TV connection or disconnection.

CAUTION

To detect connection, the DAC and the video encoder must be awake. The DAC may take up to 10 μs to wake up.

Figure 15−34. GPIO Signal Waveform Proposal for TV Detection

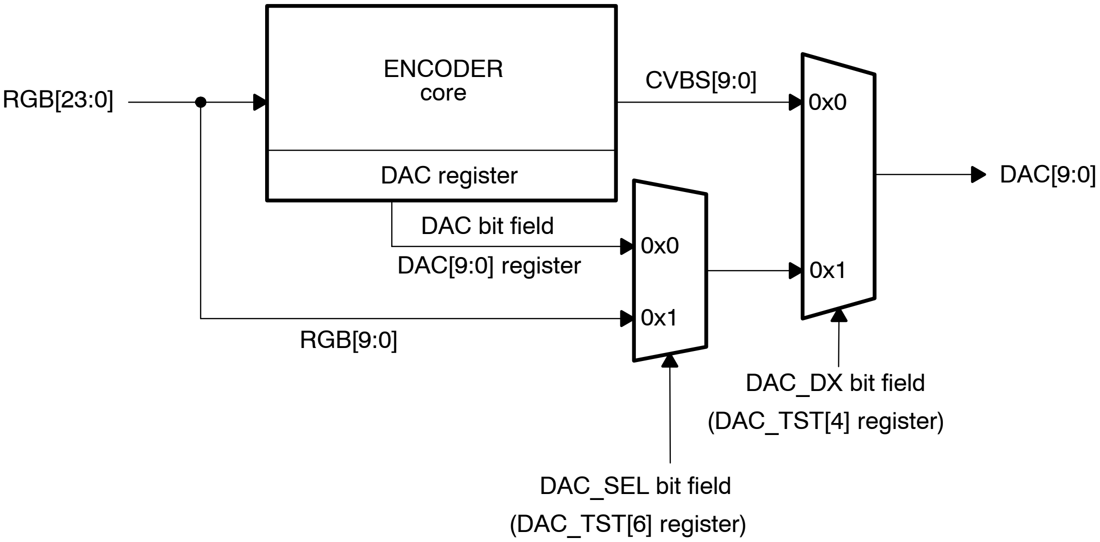


15.4.3.8 DAC_TST Mode

A DAC can be tested for debug using one of two methods: 10-bit external data or 10-bit internal register values directly connected to a DAC (see Figure 15−35).

- The DAC_DX bit field of the DAC_TST register selects whether DAC is in normal mode (0x0) or in test mode (0x1).
- The DAC_SEL bit field of the DAC_TST register selects the test mode:
  - 0x0: From the internal register DAC bit field of the DAC[9:0] register
  - 0x1: From video port G[1:0], B[7:0]

Figure 15−35. DAC Test Mode



Note:

In the external data test mode (a bypass mode), the data (G[1:0], B[7:0]) must be provided externally by the display controller. To do this, it is necessary to configure the video encoder so that it generates correct timing signals without which the display controller is unable to operate (even if the encoder core is bypassed from the data path perspective).

15.5 Display Subsystem Programming Model

This section describes how to configure the display subsystem according to the desired functions. The programming models of the three main submodules, DISPC, RFBI, and VENC, are described separately; the same method for configuring these submodules applies to the subsystem.

The following are the main configuration scenarios:

- Use of LCD panel support (bypass or RFBI mode)

  Configure the RFBI (only in RFBI mode; otherwise, leave the default values) and then the DISPC according to the desired functions to be supported.

- Use of TV-set support

  Configure the VENC and then the DISPC according to the desired functions to be supported.

- Use of both LCD panel support (bypass or RFBI mode) and TV-set support

  Configure the RFBI (only if RFBI mode; otherwise, leave the default values), the VENC, and then the DISPC according to the desired functions to be supported.

15.5.1 DSS Configuration Phase

The configuration phase is important for configuring the data flow for LCD panel use or TV use. The following is the procedure for configuring the data flow:

1) To take the DSS out of reset, the DSS-related clocks and the 54-MHz APLL clock must first be enabled:

   CM_FCLKEN1_CORE.EN_DSS1 = 0x1
   CM_FCLKEN1_CORE.EN_DSS2 = 0x1
   CM_FCLKEN1_CORE.EN_TV = 0x1
   CM_ICLKEN1_CORE.EN_DSS = 0x1
   CM_CLKEN_PLL.EN_54M_PLL = 0x3

   When the clocks are enabled, the DSS can be taken out of reset (SOFTRESET bit DSS_SYSCONFIG[1]).

2) Top-level configuration for the DISPC clock using the DPLL_APLL_CLK bit DSS_CONTROL[0] (0: DPLL, 1: APLL). This is used to set the functional clock for the DISPC (DPLL or APLL from the PRCM). The clock selection becomes active at the next inactive part of the video frame (see Section 15.3.1.1, *Clock Domains*).

3) Top-level configuration for the VENC clock using VENC_clock_mode and VENC_clock_4x_enable in DSS_CONTROL (if needed). This is used to set the VENC input clocks for TV set support.

4) Top-level configuration for DAC using DAC_DEMEN in DSS_CONTROL (setting the DAC_DEMEN bit to 1 enhances the TV color display).

5) Configuration of RFBI and (or) VENC, as needed
6) Configuration of DISPC. After a DISPC register change, the Go bit DISPC_CONTROL (GoLCD for LCD output, GoDigital for digital output) must be set. If the bit is not set, the configuration of DISPC has no effect. This setting is used to indicate that all DISPC shadow registers are programmed and the hardware can update the internal registers.
7) End

15.5.2 Display Subsystem Reset

The DSS can obtain a software reset, propagated through all submodules. This reset can be done to initialize the subsystem.

The following is a possible sequence:

1) Write DISS SOFTRESET DSS_SYSCONFIG (1: reset, 0: normal) to apply the software reset to the whole subsystem.
2) Read DISS RESETDONE DSS_SYSSTATUS to check the bit. This bit is set to 1 when the reset phase is complete.
3) If RESETDONE is set to1, the reset sequence is complete; otherwise, read DSS_SYSSTATUS again.
4) End

15.5.3 Display Controller Programming Model

Note:

Some display controller (DISPC) registers are described as shadow registers. These registers are associated with the digital output and/or the LCD output. A shadow register change has no direct effect on the configuration of DISPC without setting the Go bit DISPC_CONTROL (GOLCD for LCD output, GODIGITAL for digital output).

For digital output, after programming the shadow registers, set the GODIGITAL bit DISPC_CONTROL[6] to 1. If the GODIGITAL bit is not set, the DISPC configuration has no effect. This setting is used to indicate that all the DISPC shadow registers are programmed and the hardware can update the internal registers at the external EVSYNC.

For LCD output, after programming the shadow registers, set the GOLCD bit DISPC_CONTROL[5] to 1. If the GOLCD bit is not set, the DISPC configuration has no effect. This setting is used to indicate that all the DISPC shadow registers are programmed and the hardware can update the internal registers at the VFP start period.

Table 15−24 lists the shadow registers.

Table 15−24. Shadow Registers

Shadow Register Name	Updated on VFP Start Period (LCD Output)	Updated on External VSYNC (Digital Output)
DISPC_CONTROL	x[†]	x[†]
DISPC_CONFIG	x	x
DISPC_CAPABLE	x	
DISPC_DEFAULT_COLOR0... DISPC_DEFAULT_COLOR1	x (for i = 0)	x (for i = 1)
DISPC_TRANS_COLOR0... DISPC_TRANS_COLOR1	x (for i = 0)	x (for i = 1)
DISPC_LINE_NUMBER	x	
DISPC_TIMING_H	x	
DISPC_TIMING_V	x	
DISPC_POL_FREQ	x	
DISPC_DIVISOR	x	
DISPC_SIZE_DIG		x
DISPC_SIZE_LCD	x	
DISPC_VID1_BA0... DISPC_VID1_BA1	x	x
DISPC_VID1_POSITION	x	x
DISPC_VID1_SIZE	x	x
DISPC_VID1_ATTRIBUTES	x	x
DISPC_VID1_FIFO_THRESHOLD	x	x
DISPC_VID1_ROW_INC	x	x
DISPC_VID1_PIXEL_INC	x	x
DISPC_VID1_FIR	x	x
DISPC_VID1_PICTURE_SIZE	x	x
DISPC_VID1_ACCU0... DISPC_VID1_ACCU1	x	x
DISPC_VID1_FIR_COEF_H0... DISPC_VID1_FIR_COEF_H7	x	x
DISPC_VID1_FIR_COEF_HV0... DISPC_VID1_FIR_COEF_HV7	x	x
DISPC_VID1_CONV_COEF0 DISPC_VID1_CONV_COEF4	x	x

[†] Some bit fields of the DISPC_CONTROL register are shadow bits. For more information, see the bit field description in Table 15−48.

Table 15−24. Shadow Registers (Continued)

Shadow Register Name	Updated on VFP Start Period (LCD Output)	Updated on External VSYNC (Digital Output)
DISPC_VID2_BA0... DISPC_VID2_BA1	x	x
DISPC_VID2_POSITION	x	x
DISPC_VID2_SIZE	x	x
DISPC_VID2_ATTRIBUTES	x	x
DISPC_VID2_FIFO_THRESHOLD	x	x
DISPC_VID2_ROW_INC	x	x
DISPC_VID2_PIXEL_INC	x	x
DISPC_VID2_FIR	x	x
DISPC_VID2_PICTURE_SIZE	x	x
DISPC_VID2_ACCU0... DISPC_VID2_ACCU1	x	x
DISPC_VID2_FIR_COEF_HO... DISPC_VID2_FIR_COEF_H7	x	x
DISPC_VID2_FIR_COEF_HV0... DISPC_VID2_FIR_COEF_HV7	x	x
DISPC_VID2_CONV_COEF0 DISPC_VID2_CONV_COEF4	x	x
DISPC_DATA_CYCLE1 DISPC_DATA_CYCLE3	x	

† Some bit fields of the DISPC_CONTROL register are shadow bits. For more information, see the bit field descriptions in Table 15−48.

15.5.3.1 Display Controller Configuration

The following registers define the display controller configuration:

- ❑ DISPC_SYSCONFIG
- ❑ DISPC_SYSSTATUS
- ❑ DISPC_IRQSTATUS
- ❑ DISPC_IRQENABLE

15.5.3.2 Graphics Layer Configuration

The graphics layer configuration is common to the LCD and the TV set.

15.5.3.3 Video Layer Configuration

The video layer configuration is common to the LCD and the TV set.

Video DMA Registers

The following registers define the video DMA engine configuration:

- DISPC_CONTROL
- DISPC_VIDn_BAi
- DISPC_VIDn_ATTRIBUTES
- DISPC_VIDn_ROW_INC
- DISPC_VIDn_PIXEL_INC
- DISPC_VIDn_FIFO_THRESHOLD
- DISPC_VIDn_PICTURE_SIZE

The attributes of the video DMA engine are defined in the fields:

- Video layer enable (DISPC_VIDn_ATTRIBUTES.Enable): The video DMA engine fetches encoded pixels from the system memory only when the video layer is enabled (a valid configuration has been programmed for the video layer). The video window is present and the video pipeline is active.

- Burst size (DISPC_VIDn_ATTRIBUTES.VidBurstSize): The default burst size at reset is 4x32 bytes. The possible values are 4x32, 8x32, and 16x32 bytes. The burst size is initialized at boot time by the software and does not change when the display controller is enabled.

- Base address of the video buffer in system memory (DISPC_VIDn_BAi.VidBA): Because the base address is aligned on a pixel-size boundary, the horizontal resolution is 1 pixel. For YCbCr 4:2:2 formats, the resolution is 2 pixels. The vertical resolution is 1 line.

  The DISPC_VID_BA0 register defines the base address of the even field, and DISPC_VID_BA1 defines the base of the odd field for external synchronization based on the value and polarity of the DISPC_FID input signal.

  To improve system throughput, align the base address on a burst-size boundary.

- Video FIFO threshold (DISPC_VIDn_FIFO_THRESHOLD): The low threshold (DISPC_VIDn_FIFO_THRESHOLD.VidFifoLowThreshold) and high threshold (DISPC_VIDn_FIFO_THRESHOLD.VidFifoHigh Threshold) 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. The DMA engine waits until the low level is reached to restart the requests.

- Video buffer width (DISPC_PICTURE_SIZE.VidOrgSizeX): The buffer width in system memory is from 1 to 2048 pixels. All integer values in the range [1:2048] are allowed.

- Video buffer height (DISPC_PICTURE_SIZE.VidOrgSizeY): The buffer height in system memory is from 1 to 2048 pixels. All integer values in the range [1:2048] are allowed.

Video Rotation Configuration

- Video window row increment value (DISPC_VIDn_ROW_INC): The signed value indicates the number of bytes to increment the address at the end of the line in memory to calculate the address in system memory for the first pixel of the following line.

  The value is used in combination with DISPC_VIDn_PIXEL_INC for 2D addressing in the case of SRAM buffer.

  A value of 1 indicates that the address of the following pixel is based on the DISPC_VIDn_PIXEL_INC value only.

Note:

When the RGB24 packet format is selected, the valid values are 1 and any multiple 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.

- Video window pixel increment value (DISPC_VIDn_PIXEL_INC): The signed value indicates the number of bytes to increment the address in memory to calculate the address in system memory for the next pixel.

  The value is used in combination with DISPC_VIDn_ROW_INC for 2D addressing in the case of SRAM buffer.

  A value of 1 indicates that all the pixels on the line are contiguous in system memory (see Table 15−25).

Note:

When the RGB24 packet format is selected, the only valid value is 1.

Table 15−25. Video Rotation Register Settings

Rotation	Registers	SRAM/SDRAM Direct Access	SDRAM + Rotation Engine
0 degrees	DISPC_VIDn_BA	PBA	VBA0
	DISPC_VIDn_PIXEL_INC	1	1
	DISPC_VIDn_ROW_INC	1	(2048 – iw)*ps +1
90 degrees	DISPC_VIDn_BA	PBA + (iw * (ih−1) *ps)	VBA90
	DISPC_VIDn_PIXEL_INC	−iw * ps − (ps−1)	1
	DISPC_VIDn_ROW_INC	(iw * (ih−1)) * ps+1	(2048 – ih)*ps +1
180 degrees	DISPC_VIDn_BA	PBA + (iw * ih −1) *ps	VBA180
	DISPC_VIDn_PIXEL_INC	−ps − (ps−1)	1
	DISPC_VIDn_ROW_INC	−ps − (ps−1)	(2048 – iw)*ps +1
270 degrees	DISPC_VIDn_BA	PBA + (iw −1) *ps	VBA270
	DISPC_VIDn_PIXEL_INC	(iw−1)*ps +1	1
	DISPC_VIDn_ROW_INC	−(iw*(ih−1))*ps −ps − (ps−1)	(2048 – ih)*ps +1

PBA = Physical base address of the buffer in system memory (top-left corner of the picture)
VBAx = Virtual base address corresponding to the rotation view
iw = Number of pixels per row (original picture in system memory) for BITMAP and RGB formats and number of pixels per row divided by 2 for YUB422 formats
ih = Number of lines (original picture in system memory)
ps = Pixel size in bytes (BITMAP8: 1 byte; RGB16: 2 bytes; YUV422: 4 bytes)

Table 15−26. Video Rotation Register Settings (YUV only)

Rotation	Registers	SDRAM + Rotation Engine
0 degree	DISPC_VIDn_ATTRIBUTES.VidRotation DISPC_VIDn_ATTRIBUTES.VidRowRepeatEnable	0x0 0x0
90 degrees	DISPC_VIDn_ATTRIBUTES.VidRotation DISPC_VIDn_ATTRIBUTES.VidRowRepeatEnable	0x1 0x1
180 degrees	DISPC_VIDn_ATTRIBUTES.VidRotation DISPC_VIDn_ATTRIBUTES.VidRowRepeatEnable	0x2 0x0
270 degrees	DISPC_VIDn_ATTRIBUTES.VidRotation DISPC_VIDn_ATTRIBUTES.VidRowRepeatEnable	0x3 0x1

- ☐ Video rotation flag (DISPC_VIDn_ATTRIBUTES.VidRotation): This flag indicates 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, some post-processing must be performed on the YUV 4:2:2 data depending on the rotation.

  The bit field is set only when the video format is not RGB; otherwise, the bit field is reset to 0.

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 converting from YUV to RGB.

- For 90- and 270-degree rotation, the missing chrominance samples of 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.

Video Mirroring Configuration

Table 15−27 and Table 15−28 list the programmed values to access the buffer in memory (contiguous pixels).

Table 15−27. Register Settings for Video Rotation With Mirroring

Rotation with Mirroring	**Registers**	**SRAM/SDRAM Direct Access**	**SDRAM + Rotation Engine**
0 degree	DISPC_VIDn_BA	PBA + (iw −1)*ps	VBA180 + 2048 * (ih−1) *ps
	DISPC_VIDn_PIXEL_INC	−ps − (ps−1)	1
	DISPC_VIDn_ROW_INC	2*(iw−1)*ps+1	−(2048 + iw)*ps + 1)
90 degrees	DISPC_VIDn_BA	PBA	VBA270+ 2048 * (iw−1) *ps
	DISPC_VIDn_PIXEL_INC	(iw−1)*ps +1	1
	DISPC_VIDn_ROW_INC	(−iw*(ih−1))*ps + 1	−(2048 + ih)*ps + 1)
180 degrees	DISPC_VIDn_BA	PBA + iw*(ih−1)*ps	VBA0+ 2048 * (ih−1) *ps
	DISPC_VIDn_PIXEL_INC	1	1
	DISPC_VIDn_ROW_INC	−(2*iw−1)*ps − (ps−1)	−(2048 – iw)*ps + 1)
270 degrees	DISPC_VIDn_BA	PBA + (iw*ih−1)*ps	VBA90+ 2048 * (iw−1) *ps
	DISPC_VIDn_PIXEL_INC	−iw*ps − (ps−1)	1
	DISPC_VIDn_ROW_INC	(iw*(ih−1))*ps – ps – (ps –1)	−(2048 – ih)*ps + 1)

Note: PBA = Physical base address of the buffer in system memory (top-left corner of the picture)
VBAx = Virtual base address corresponding to the rotation view
iw = Number of pixels per row (original picture in system memory) for BITMAP and RGB formats and number of pixels per row divided by two for YUB422 formats
ih = Number of lines (original picture in system memory)
ps = Pixel size in bytes (BITMAP8: 1 byte, RGB16: 2 bytes; YUV422: 4 bytes)

Table 15−28. Register Settings for Video Rotation With Mirroring (YUV only)

Rotation	**Registers**	**SDRAM + Rotation Engine**
0 degree	DISPC_VIDn_ATTRIBUTES.VidRotation DISPC_VIDn_ATTRIBUTES.VidRowRepeatEnable	0x2 0x0
90 degrees	DISPC_VIDn_ATTRIBUTES.VidRotation DISPC_VIDn_ATTRIBUTES.VidRowRepeatEnable	0x1 0x1
180 degrees	DISPC_VIDn_ATTRIBUTES.VidRotation DISPC_VIDn_ATTRIBUTES.VidRowRepeatEnable	0x0 0x0
270 degrees	DISPC_VIDn_ATTRIBUTES.VidRotation DISPC_VIDn_ATTRIBUTES.VidRowRepeatEnable	0x3 0x1

Video Configuration Registers

The following shadow registers define the video layer #n configuration:

- DISPC_CONFIG
- DISPC_VIDn_POSITION
- DISPC_VIDn_SIZE
- DISPC_VIDn_ATTRIBUTES
- DISPC_VIDn_FIR
- DISPC_VIDn_PICTURE_SIZE
- DISPC_VIDn_FIR_COEF_Hi
- DISPC_VIDn_FIR_COEF_HVi
- DISPC_VIDn_CONV_COEFi

Video layer n is enabled/disabled by setting/resetting the DISPC_VIDn_ATTRIBUTES.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, set a valid configuration.

After a register change, the DISPC_CONTROL register GODIGITAL or GOLCD bit must be set. The software must wait for the hardware to reset this bit before setting it.

Video Window Attributes

The attributes of the video window n are defined in the fields:

- Video format (DISPC_VIDn_ATTRIBUTES.VidFormat): The video format can be RGB16, RGB24, YUV2 4:2:2 co-sited, and UYVY 4:2:2 co-sited.
- Video window X-position (DISPC_VIDn_POSITION.VidPosX): The window X-position is from 0 to 2047 columns. All integer values in the range [0:2047] are allowed.
- Video window Y-position (DISPC_VIDn_POSITION.GxfPosY): The window Y-position is from 0 to 2047 rows. All integer values in the range [0:2047] are allowed.
- Video window width (DISPC_VIDn_SIZE.VidSizeX): 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 DISPC_VIDn_ATTRIBUTES.VidBurstSize (in bytes).

Note:

When the RGB24 packet format is selected, and the DISPC_VIDn_ROW_INC is not equal to 1, the width must be a multiple of 12 bytes. When DISPC_VIDn_ROW_INC is 1, the width can be any size from 1 to 2048 pixels.

- Video window height (DISPC_VIDn_SIZE.VidSizeY): The window height is from 1 to 2048 pixels. All integer values in the range [1:2048] are allowed.
- Video picture width in system memory (DISPC_VIDn_PICTURE_SIZE.VidSizeX): The window width is from 1 to 2048 pixels. All integer values in the range [1:2048] are allowed for RGB16 and RGB24 video data. For YUV2 4:2:2 and UYVY 4:2:2 formats, the width must be a multiple of 2 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 DISPC_VIDn_ATTRIBUTES.VidBurstSize (in bytes).
- Video picture height in system memory (DISPC_VIDn_PICTURE_SIZE.VidSizeY): The window width is from 1 to 2048 pixels. All integer values in the range [1:2048] are allowed.
- Video data endianness (DISPC_VID_ATTRIBUTES.Endianness): The register indicates the endianness (little or big endian) of the video pixels.

Video Up-/Downsampling Configuration

The video horizontal up-/downsampling block for the video pipeline #n is enabled/disabled by setting/resetting the DISPC_VIDn_ATTRIBUTES.VidResizeEnable[0] field.

The video vertical up-/downsampling block for the video pipeline #n is enabled/disabled by setting/resetting the DISPC_VIDn_ATTRIBUTES.VidResizeEnable[1] field.

Before enabling the video up-/downsampling block, set a valid configuration.

After a register change, the DISPC_CONTROL.Go bit must be set. The software must wait until the hardware has reset this bit before setting the Go bit.

The configuration of the video up-/downsampling block for the video pipeline #n is defined in the fields:

- Vertical up-/downsampling increment value (DISPC_VIDn_FIR.VidFIRVInc):

  The unsigned signed integer value range is [1:2048].

  The value is calculated by the software using Equation 15−1.

Equation 15−1. Video Up-/Downsampling Increment (Vertical)

$$VidUpsampIngVInc = 1024 * \frac{VidOrgSizeY}{VidSizeY}$$

- Horizontal up-/downsampling increment value (DISPC_VIDn_FIR.VidFIRHInc):

  The unsigned signed integer value range is [1:2048].

  The value is calculated by the software using Equation 15−2.

Equation 15−2. Video Up-/Downsampling Increment (Horizontal)

$$VidUpsamplingHInc = 1024 * \frac{VidOrgSizeX}{VidSizeX}$$

- ❑ Vertical up-/downsampling accumulator value (DISPC_VIDn_ACCU.VidVerticalAccu):

  The unsigned signed integer value range is [0:1023].

  The accumulator value indicates on which phase the vertical filtering starts. The value 0 indicates that phase 0 is the first phase used by the hardware to generate the first data.

- ❑ Horizontal up-/downsampling accumulator value (DISPC_VIDn_ACCU.VidHorizontalAccu):

  The unsigned signed integer value range is [0:1023].

  The accumulator value indicates the phase on which the horizontal filtering starts. The value 0 indicates that phase 0 is the first phase used by the hardware to generate the first data (see Table 15−29).

Table 15−29. Vertical/Horizontal Accumulator Phase

Accumulator Value	**Phase (ϕ)**
0	0
128	1
256	2
384	3
512	4
640	5
768	6
896	7

- ❑ Vertical up-/downsampling coefficients (DISPC_VIDn_FIR_COEF_HVi): The three taps vertical up-/downsampling coefficients are defined in the DISPC_VIDn_FIR_COEF_HVi registers. There are eight registers for the eight phases, with three coefficients for each, for a total of 24 programmable coefficients for the vertical upsampling/downsampling block.

  Each register contains two 8-bit signed coefficients and one 8-bit unsigned coefficient (central one).

- ❑ Horizontal up-/downsampling coefficients (DISPC_VIDn_FIR_COEF_Hi and DISPC_VIDn_FIR_COEF_HVi): The five taps horizontal up-/downsampling coefficients are defined in the DISPC_VIDn_FIR_COEF_Hi and DISPC_VIDn_FIR_COEF_HVi registers.

  There are eight registers for the eight phases, with five coefficients for each, for a total of 40 programmable coefficients for the horizontal up-/downsampling block.

Each DISPC_VIDn_FIR_COEF_Hi register contains three 8-bit signed coefficients and one 8-bit unsigned coefficient (central one), and each DISPC_VIDn_FIR_COEF_HVi contains one 8-bit signed coefficient.

The programmable coefficient for the FIR up-/downsampling method must be adjusted based on application needs.

Video Color Space Conversion Configuration

The programmable color space conversion block for the video pipeline #n has nine 11-bit coefficients defined in DISPC_VIDn_COEFi (i = 0, 1, 0, 4).

Equation 15−3 and Equation 15−4 give the standard register coefficients.

Equation 15−3. YCbCr to RGB Registers (VidFullRange = 0)

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \frac{1}{256} * \begin{bmatrix} RY & RCr & RCb \\ GY & GCr & GCb \\ BY & BCr & BCb \end{bmatrix} * \begin{bmatrix} Y & 16 \\ Cr & 128 \\ Cb & 128 \end{bmatrix}$$

Equation 15−4. YCbCr to RGB Registers (VidFullRange = 1)

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \frac{1}{256} * \begin{bmatrix} RY & RCr & RCb \\ GY & GCr & GCb \\ BY & BCr & BCb \end{bmatrix} * \begin{bmatrix} Y & \\ Cr & 128 \\ Cb & 128 \end{bmatrix}$$

Table 15−30 lists the register values for color space conversion.

Table 15−30. 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

15.5.3.4 LCD-Specific Control Registers

The following registers define the LCD output configuration:

- DISPC_CONTROL
- DISPC_CONFIG
- DISPC_DEFAULT_COLOR0
- DISPC_TRANS_COLOR0
- DISPC_TIMING_H
- DISPC_TIMING_V
- DISPC_POL_FREQ
- DISPC_DIVISOR
- DISPC_SIZE_LCD

The LCD output is enabled/disabled by setting/resetting the DISPC_CONTROL.LCDEnable field.

Before enabling the LCD output, set a valid configuration.

LCD Attributes

The following fields define the attributes of the panel connected to the display controller:

- Monochrome or color panel (DISPC_CONTROL.MonoColor)
- STN or TFT panel (DISPC_CONTROL.STNTFT)
- Color depth (DISPC_CONTROL.TFTDataLines)
- Number of lines (DISPC_SIZE_LCD.LPP)
- Number of pixels per line (DISPC_SIZE_LCD.PPL)
- 4- or 8-bit interface for STN monochrome panel (DISPC_CONTROL.M8B)

LCD Timing

The following fields define the timing generation of HSYNC/VSYNC:

- Horizontal front porch (DISPC_TIMING_H.HFP)
- Horizontal back porch (DISPC_TIMING_H.HBP)
- Horizontal synchronization pulse width (DISPC_TIMING_H.HSW)
- Vertical front porch (DISPC_TIMING_V.VFP)
- Vertical back porch (DISPC_TIMING_V.VBP)
- Vertical synchronization pulse width (DISPC_TIMING_V.VSW)
- On/off control of HSYNC/VSYNC pixel clock (DISPC_POL_FREQ.OnOff)
- Program HSYNC/VSYNC rise or fall (DISPC_POL_FREQ.RF)
- Invert HSYNC (DISPC_POL_FREQ.IHS)
- Invert VSYNC (DISPC_POL_FREQ.IVS)
- HSYNC gated (DISPC_CONFIG.HSYNCGated)
- VSYNC gated (DISPC_CONFIG.VSYNCGated)

Table 15−31 lists the programming rules for the HBP, HSW, HFP, and PCD bit fields.

Table 15−31. Programming Rules

No Downsampling		**Downsampling H or V**	**Downsampling H + V**
(HBP + HSW+ HFP) * PCD	> 8	> 10	> 20

The following fields define the timing generation of ac bias (output enable in TFT mode):

- Invert output enable (DISPC_POL_FREQ.IEO)
- ac-bias pin frequency (DISPC_POL_FREQ.ACB)
- ac-bias pin transitions per interrupt (DISPC_POL_FREQ.ACBI)
- ac-bias gated (DISPC_CONFIG.ACBiasGated)

The following fields define the timing generation of the pixel clock:

- Pixel clock divisor (DISPC_DIVISOR.PCD)
- Invert pixel clock (DISPC_POL_FREQ.IPC)
- Pixel clock gated (DISPC_CONFIG.PixelClockGated)

The 8-bit pixel clock divider (DISPC_DIVISOR.PCD) field is used to select the pixel clock frequency. PCD generates a range of pixel clock frequencies from LC/2 to LC/256, where LC is the logic clock from the divided functional clock of the display controller by DISPC_DIVISOR.LCD. The refresh rate depends on the following parameters:

- Horizontal front porch (DISPC_TIMING_H.HFP)
- Horizontal back porch (DISPC_TIMING_H.HBP)
- Horizontal synchronization pulse width (DISPC_TIMING_H.HSW)
- Vertical front porch (DISPC_TIMING_V.VFP)
- Vertical back porch (DISPC_TIMING_V.VBP)
- Vertical synchronization pulse width (DISPC_TIMING_V.VSW)
- Number of lines (DISPC_SIZE.LPP)
- Number of pixels per line (DISPC_SIZE.PPL)
- 4- or 8-bit interface for STN monochrome panel (DISPC_CONTROL.M8B)

CAUTION

The DISPC_DIVISOR.PCD minimum value is:

0x2 for TFT panels

0x3 for STN panels

The following field defines the behavior of the internal blocks:

- TFT dithering enabled (DISPC_CONTROL.TFTDitherEnable)

The following field defines the security aspect:

- Security (DISPC_CONTROL.Secure)

The following field defines the clock-gating strategy:

- In TFT mode, the pixel clock is always gated or only when valid data are present (DISPC_CONFIG.PixelGated)

The following fields define the capability of the LCD output:

- TFT dithering (DISPC_CAPABLE.TFTditheringCapable)

- STN dithering (DISPC_CAPABLE.STNditheringCapable)

LCD Overlay

The following fields define the overlay attributes of the LCD output:

- Transparency color key (DISPC_TRANS_COLOR0)
- Transparency color key enable (DISPC_CONFIG.TCKLCDEnable)
- Transparency color key selection between destination graphics transparency color key, and source video transparency color key (DISPC_CONFIG.TCKLCDSelection)
- Default solid background color defined (DISPC_DEFAULT_COLOR0.DefaultColor)

Note:

The destination graphics transparency color key is available only to the overlay to which the graphics pipeline is connected. The software is responsible for setting the correct configuration of the LCD and digital overlays.

When all these fields have been set to the appropriate values, set the DISPC_CONTROL.GoLCD field to indicate that all the shadow registers of the pipelines connected to the LCD output are latched by the hardware.

15.5.3.5 TV-Set Specific Control Registers

The following registers define the digital output configuration:

- DISPC_CONTROL
- DISPC_CONFIG
- DISPC_DEFAULT_COLOR1
- DISPC_TRANS_COLOR1
- DISPC_SIZE_DIG

The digital output is enabled/disabled by setting/resetting the DISPC_CONTROL.DigitalEnable field.

Before enabling the digital output, set a valid configuration.

Digital Timings

The timing information: The following fields define the hold time:

- Data hold time (DISPC_CONTROL.HT)
- Logic clock divisor (DISPC_DIVISOR.LCD)

The 8-bit pixel clock divider (DISPC_DIVISOR.LCD) field is used to select the logic clock frequency. 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.

Digital Frame/Field Size

The following fields define the field size (frame size, if progressive mode):

- ☐ Number of lines (DISPC_SIZE_DIG.LPP)
- ☐ Number of pixels per line (DISPC_SIZE_DIG.PPL)

Digital Overlay

The following fields define the overlay attributes of the digital output:

- ☐ Transparency color key (DISPC_TRANS_COLOR1)
- ☐ Transparency color key enable (DISPC_CONFIG.TCKDigEnable)
- ☐ Transparency color key selection between destination graphics transparency color key and source video transparency color key (DISPC_CONFIG.TCKDigSelection)
- ☐ Default solid background color defined (DISPC_DEFAULT_COLOR1.DefaultColor)

Note:

The destination graphics transparency color key is available only to the overlay to which the graphics pipeline is connected. The software is responsible for setting the correct configuration of the LCD and digital overlays.

The following field defines the security aspect:

- ☐ Security (DISPC_CONTROL.Secure)

When this field is set to the appropriate values, set field DISPC_CONTROL.GoDigital to indicate that all the shadow registers of the pipelines connected to the digital output are latched by the hardware.

15.5.4 RFBI Programming Model

The RFBI programming model must be used only for LCD display support.

15.5.4.1 RFBI Control Registers

The following registers define the RFBI control registers:

- ☐ RFBI_CONTROL
- ☐ RFBI_PIXEL_CNT
- ☐ RFBI_LINE_NUMBER

Bypass Mode

By setting the RFBI_CONTROL.BypassMode bit, the output of the LCD controller is directly output to the LCD panel. By resetting the BypassMode field, the MPU module sends commands/parameters and data from the internal FIFO.

Enable

The RFBI module is enabled/disabled by setting/resetting the RFBI_CONTROL.Enable bit. The hardware resets the enable bit after all the pixels are sent to the panel. The number of pixels to be sent to the LCD panel is defined by the RFBI_PIXEL_CNT.PixelCnt value.

When the transfer is complete, the configuration that was in use can be modified.

Configuration Selection

By setting the RFBI_CONTROL.ConfigSelect bit field, the configuration number is selected (#1 or #0 if if bits are set or reset). The registers associated with the configuration are used to output the data to the LCD panel. If both CSs are selected, the configuration for the first CS is used (except for the polarity of the CS1 signal defined by the second configuration) and both devices connected to the CS signals are driven in parallel. In read mode, if both CS are set, only CS0 is asserted to read data from the device connected on the CS0. In write mode with two CSs selected, the RFBI can write to the two devices simultaneously.

ITE Bit

By setting the RFBI_CONTROL.ITE bit, the capturing of data from the display controller can begin. The ITE bit has no effect if the trigger mode is set to external.

The display controller must be configured in RFBI mode to take into account the STALL signal. If the trigger mode is set to external, the ITE bit is ignored. The corresponding CS must be selected when setting the ITE bit.

Number of Pixels to Transfer

By setting the RFBI_PIXEL_CNT.PixelCnt value, the application indicates the number of pixels to be transferred to the LCD panel. The value can be changed only when the RFBI_CONTROL.Enable bit field is reset. During the transfer, the hardware decrements the register when a pixel is sent to the RFB. If RFBI_CONTROL.Enable is set and a new value is written in the RFBI_PIXEL_CNT register when the current value in the register is a nonzero value (remaining number of pixels to transfer), the ongoing transfer is aborted.

Programmable Line Number

If the trigger mode is set to external trigger mode with HSYNC and VSYNC or TE, the hardware resets the line counter when the VSYNC occurs; after a programmable number of lines (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 (ORed between HSYNC and VSYNC).

15.5.5 RFBI Configuration

The following registers define the RFBI configuration register:

- RFBI_SYSCONFIG
- RFBI_SYSSTATUS
- RFBI_CONFIGi
- RFBI_VSYNC_WIDTH
- 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 RFBI_CONTROL. ConfigSelect).

15.5.5.1 Parallel Mode

The RFBI_CONFIG.ParallelMode field defines the width of the interface (8-, 9-, 12- or 16-bit parallel).

15.5.5.2 Trigger Mode

By setting the RFBI_CONFIG.TriggerMode field, you configure 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 ITE bit.

15.5.5.3 VSYNC Pulse Width (Minimum Value)

The RFBI_VSYNC_WIDTH.MinVSYNCWidth 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. It is also used in the VSYNC/HSYNC mode on the two separate input lines.

- VSYNC pulse width must be equal to at least two L4 cycles when HSYNC is not present.

- VSYNC pulse width must be equal to at least four L4 cycles when HSYNC is present.

15.5.5.4 HSYNC Pulse Width (Minimum Value)

The RFBI_HSYNC_WIDTH.MinHSYNCWidth field defines the minimum number of L4 clock cycles of the HSYNC pulse for detection on HSYNC. It allows the differentiation between VSYNC and HSYNC, which are ORed on the same signal. It is also used in VSYNC/HSYNC mode on the two separate input lines.

To be detected, the HSYNC pulse width must always be equal to at least two L4 cycles.

15.5.5.5 Cycle **Format**

Setting the RFBI_CONFIG.CycleFormat bit field defines which registers are to be used to format the data in the FIFO with the appropriate number of bits (starting from the LSB) and with the alignment on the interface. The following are the register selections:

- RFBI_CYCLE1 (RFBI_CONFIG.CycleFormat = 0)
- RFBI_CYCLE1 (RFBI_CONFIG.CycleFormat = 0) and RFBI_CYCLE2 (if RFBI_CONFIG.CycleFormat = 1)
- RFBI_CYCLE1 (RFBI_CONFIG.CycleFormat = 0), RFBI_CYCLE2 (if RFBI_CONFIG.CycleFormat = 1), and RFBI_CYCLE3

The data from the display controller and L4 interconnect are formatted based on the configuration of the RFBI_CYCLE registers (i = 0, 1, 2).

15.5.5.6 Unused Bits

Based on the configuration, the undefined bits for each cycle are defined with the values of the bits at the same position in the previous cycle: 0s or 1s (the unused bits can be at any position).

The field used is RFBI_CONFIG.UnusedBits.

15.5.5.7 RFBI Timings

The timing registers for one configuration can be accessed only when the configuration is not in use based on the value of RFBI_CONTROL.ConfigSelect.

Granularity is defined using the TIMEGRANULARITY bit field of the RFBI_CONFIG0/1 register (bit 4). This feature allows extension of programmable ranges of timing parameters for RFBI.

Chip Selection Assertion/Deassertion Time

A0 setup time to CS assertion is assured by programmable CS assertion time from start access time:

- CSOnTime bit field of the RFBI_ONOFF_TIMEi register
- CSOnTime can be set from 0 to 15 L4Clk cycles with a granularity of 1.
- CSOnTime can be set from 0 to 30 L4Clk cycles with a granularity of 2.

CS deassertion time from start access time is programmable:

- CSOffTime bit field of the RFBI_ONOFF_TIMEi register
- CSOffTime can be set from 0 to 63 L4Clk cycles with a granularity of 1.
- CSOffTime can be set from 0 to 126 L4Clk cycles with a granularity of 2.

CAUTION

Configuring CSOnTime = CSOff Time = 0 is not supported and must be avoided. This creates contention on the bus and will progressively damage the LCD panel.

Chip Selection Pulse Width

The total chip-select pulse width is the time when the write cycle time or read cycle time is complete; it is programmable:

- CSPulseWidth bit field of the RFBI_CYCLE_TIMEi register
- CSPulseWidth can be set from 0 to 63 L4Clk cycles with a granularity of 1.
- CSPulseWidth can be set from 0 to 126 L4Clk cycles with a granularity of 2.

It applies to the read-to-write, write-to-read, read-to-read, and write-to-write access based on:

- RREnable: Read-to-read access
- WWEnable: Write-to-write access
- RWEnable: Read-to-write access
- WREnable: Write-to-read access

By default, it applies to any access (read-to-read, read-to-write, write-to-read, and write-to-write) when the CS changes.

Access Time

The total access time is the time when A0 becomes valid until data are sampled before deasserting the RE signal; it is programmable:

- AccessTime bit field of the RFBI_CYCLE_TIMEi register
- AccessTime can be set from 0 to 63 L4Clk cycles with a granularity of 1.
- AccessTime can be set from 0 to 126 L4Clk cycles with a granularity of 2.

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 (REOffTime).

Write Enable Cycle Time

The total write enable cycle time is the time when A0 becomes valid until write cycle completion; it is programmable:

- WECycleTime bit field of the RFBI_CYCLE_TIMEi register
- WECycleTime can be set from 0 to 63 L4Clk cycles with a granularity of 1.
- WECycleTime can be set from 0 to 126 L4Clk cycles with a granularity of 2.

Write Enable Assertion/Deassertion Time

WE assertion delay time from start access time is programmable:

- WEOnTime bit field of the RFBI_ONOFF_TIMEi register
- WEOnTime can be set from 0 to 15 L4Clk cycles with a granularity of 1.
- WEOnTime can be set from 0 to 30 L4Clk cycles with a granularity of 2.

WE deassertion delay time from start access time is programmable:

- WEOffTime bit field of the RFBI_ONOFF_TIMEi register
- WEOffTime can be set from 0 to 63 L4Clk cycles with a granularity of 1.
- WEOffTime can be set from 0 to 126 L4Clk cycles with a granularity of 2.

Read Enable Cycle Time

The total read enable cycle time is the time when A0 becomes valid until read cycle completion; it is programmable:

- RECycleTime bit field of the RFBI_CYCLE_TIMEi register
- RECycleTime can be set from 0 to 63 L4Clk cycles with a granularity of 1.
- RECycleTime can be set from 0 to 126 L4Clk cycles with a granularity of 2.

Read Enable Assertion/Deassertion Time

RE assertion delay time from start access time is programmable:

- REOnTime bit field of the RFBI_ONOFF_TIMEi register
- REOnTime can be set from 0 to 15 L4Clk cycles with a granularity of 1.
- REOnTime can be set from 0 to 30 L4Clk cycles with a granularity of 2.

RE deassertion delay time from start access time is programmable:

- REOffTime bit field of the RFBI_ONOFF_TIMEi register
- REOffTime can be set from 0 to 63 L4Clk cycles with a granularity of 1.
- REOffTime can be set from 0 to 126 L4Clk cycles with a granularity of 2.

At cycle time completion (read access or write access), all control signals (CS, WE, and RE) are deasserted regardless of their deassertion time parameters values, if they are not already deasserted.

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 are not necessarily deasserted when a pipelined request to the same chip-select or to a 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 (CSPulseWidth = 0) and if assertion time parameters associated with the following access are equal to 0, the asserted control signals (CS, WE, and RE) are kept asserted. This is applicable to any read/write to read/write access combination.

15.5.5.8 RFBI State-Machine

According to Table 15−22, the signals A0, RE, and WE are asserted/deasserted based on the register that is accessed (RFBI_CMD, RFBI_PARAM, RFBI_DATA, RFBI_READ, and RFBI_STATUS).

When the RFBI_SYSSTATUS.Busy bit field is set by hardware, any access to the following registers is stalled.

Command Register

To write a command at a time, write in the RFBI_CMD register.

If the previous command has not been processed, the RFBI_SYSSTATUS.Busy bit field is set by hardware, and the access for writing a new command is stalled.

Parameter Register

To write a parameter at a time, write in the RFBI_PARAM register.

If the previous parameter has not been processed, the RFBI_SYSSTATUS.Busy bit field is set by hardware, and the access for writing a new parameter is stalled.

Data Register

To write one or two pixels at a time, write in the RFBI_DATA register (when RFBI_CONFIGi.CycleFormat = 0x3, two pixels must be written contiguously; no other access to RFBI registers except RFBI_DATA is allowed).

The pixels are formatted according to the specified cycle format. If two pixels are written into the 32-bit data register, the RFBI_CONFIGi.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 has not been processed, the RFBI_SYSSTATUS.Busy bit field is set by hardware and any access for writing new data is stalled. When the RFBI_SYSSTATUS.Busy bit field 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 RFBI_READ or RFBI_STATUS registers. In that case, the RFBI_SYSSTATUS.Busy bit field is set until the data is available in the register.

When the RFBI_SYSSTATUS.Busy bit field 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 RFBI_SYSSTATUS.Busy bit field until it is reset by hardware. To receive the data, ensure that the appropriate command/parameters are sent.

15.5.6 Video Encoder Programming Model

For the programming of the video encoder, see Table 15−32, which provides the register values to use in standard applications. Use the programming values only for TV display support.

Table 15−32. Programming Values

Registers	NTSC 601	PAL 601
F_CONTROL	0x0000 0000	0x0000 0000
VIDOUT_CTRL	0x0000 0001	0x0000 0001
SYNC_CTRL	0x0000 8040	0x0000 8040
LLEN	0x0000 0359	0x0000 035F
FLENS	0x0000 020C	0x0000 0270
HFLTR_CTRL	0x0000 0000	0x0000 0000
CC_CARR_WSS_CARR	0x043F 0000	0x2F72 0000
C_PHASE	0x0000 0000	0x0000 0000
GAIN_U	0x0000 0110	0x0000 0111
GAIN_V	0x0000 017A	0x0000 0181
GAIN_Y	0x0000 0147	0x0000 0140
BLACK_LEVEL	0x0000 0038	0x0000 003B
BLANK_LEVEL	0x0000 0038	0x0000 003B
X_COLOR	0x0000 0007	0x0000 0007
M_CONTROL	0x0000 0001	0x0000 0002
BSTAMP_WSS_DATA	0x0000 0038	0x0000 003F
S_CARR	0x21F0 7C1F	0x2A09 8ACB
LINE21	0x0000 0000	0x0000 0000
LN_SEL	0x0000 0015	0x0000 0015
L21__WC_CTL	0x0000 1400	0x0000 1400
HTRIGGER_VTRIGGER	0x0000 0000	0x0000 0000
SAVID__EAVID	0x0693 00F4	0x06A7 0108
FLEN__FAL	0x0016 020C	0x0018 0270
LAL__PHASE_RESET	0x0006 0107	0x0004 0136
HS_INT_START_STOP_X	0x008D 034E	0x0092 0358
HS_EXT_START_STOP_X	0x000F 0359	0x000F 035F
VS_INT_START_X	0x01A0 0000	0x01A7 0000
VS_INT_STOP_X__VS_INT_START_Y	0x0203 01A0	0x026F 01A7
VS_INT_STOP_Y__VS_EXT_START_X	0x01AC 0024	0x01AF 0036
VS_EXT_STOP_X__VS_EXT_START_Y	0x020D 01AC	0x0020 01AF
VS_EXT_STOP_Y	0x0000 0006	0x0000 0025
AVID_START_STOP_X	0x0348 0078	0x0353 0083
AVID_START_STOP_Y	0x0204 0024	0x0270 002F
FID_INT_START_X__FID_INT_START_Y	0x0001 008A	0x0005 008A
FID_INT_OFFSET_Y__FID_EXT_START_X	0x01AC 0106	0x002E 0138
FID_EXT_START_Y__FID_EXT_OFFSET_Y	0x0106 0006	0x0138 0055
TVDETGP_INT_START_STOP_X	0x0014 0001	0x0014 0001

Table 15−32. Programming Values (Continued)

Registers	NTSC 601	PAL 601
TVDETGP_INT_START_STOP_Y	0x0001 0001	0x0001 0001
GEN_CTRL	0x00FF 0000	0x00F9 0000
DAC_TST__DAC_A	0x0000 0002	0x0000 0002
DAC_B__DAC_C	0x0000 0000	0x0000 0000

Registers not listed in Table 15−32 use default values.

15.6 Display Subsystem Registers

This section describes the display subsystem registers.Table 15−33 lists the base address and size for each instance.

Table 15−34 through Table 15−37 list the various registers.

Table 15−33. Instance Summary

Instance Name	Base Address	Size	Module Name
DISS1	0x4805 0000	1K byte	DSS-specific
DISC1	0x4805 0400	1K byte	Display controller
RFBI1	0x4805 0800	1K byte	Remote frame buffer interface
VENC1	0x4805 0C00	1K byte	Video encoder

Table 15−34. DSS Registers

Register Name	Type	Register Width (Bits)	Offset
DSS_REVISIONNUMBER	R	32	0x0000
DSS_SYSCONFIG	RW	32	0x0010
DSS_SYSSTATUS	R	32	0x0014
DSS_CONTROL	RW	32	0x0040
DSS_PSA_LCD_REG_1	R	32	0x0050
DSS_PSA_LCD_REG_2	R	32	0x0054
DSS_PSA_VIDEO_REG	R	32	0x0058
DSS_STATUS	R	32	0x005C

Table 15−35. Display Controller Registers

Register Name	Type	Register Width (Bits)	Offset
DISPC_REVISION	R	32	0x0000
DISPC_SYSCONFIG	RW	32	0x0010
DISPC_SYSSTATUS	R	32	0x0014
DISPC_IRQSTATUS	RW	32	0x0018
DISPC_IRQENABLE	RW	32	0x001C
DISPC_CONTROL	RW	32	0x0040
DISPC_CONFIG	RW	32	0x0044
DISPC_CAPABLE	RW	32	0x0048
DISPC_DEFAULT_COLOR0 – DISPC_DEFAULT_COLOR1	RW	32	0x004C– 0x0050
DISPC_TRANS_COLOR0 – DISPC_TRANS_COLOR1	RW	32	0x0054– 0x0058
DISPC_LINE_STATUS	R	32	0x005C
DISPC_LINE_NUMBER	RW	32	0x0060
DISPC_TIMING_H	RW	32	0x0064

Table 15−35. Display Controller Registers (Continued)

Register Name	Type	Register Width (Bits)	Offset
DISPC_TIMING_V	RW	32	0x0068
DISPC_POL_FREQ	RW	32	0x006C
DISPC_DIVISOR	RW	32	0x0070
DISPC_SIZE_DIG	RW	32	0x0078
DISPC_SIZE_LCD	RW	32	0x007C
DISPC_VID1_BA0 − DISPC_VID1_BA1	RW	32	0x00BC−0x00C0
DISPC_VID1_POSITION	RW	32	0x00C4
DISPC_VID1_SIZE	RW	32	0x00C8
DISPC_VID1_ATTRIBUTES	RW	32	0x00CC
DISPC_VID1_FIFO_THRESHOLD	RW	32	0x00D0
DISPC_VID1_FIFO_SIZE_STATUS	R	32	0x00D4
DISPC_VID1_ROW_INC	RW	32	0x00D8
DISPC_VID1_PIXEL_INC	RW	32	0x00DC
DISPC_VID1_FIR	RW	32	0x00E0
DISPC_VID1_PICTURE_SIZE	RW	32	0x00E4
DISPC_VID1_ACCU0 − DISPC_VID1_ACCU1	RW	32	0x00E8−0x00EC
DISPC_VID1_FIR_COEF_H0 − DISPC_VID1_FIR_COEF_H7	RW	32	0x00F0−0x0128
DISPC_VID1_FIR_COEF_HV0 − DISPC_VID1_FIR_COEF_HV7	RW	32	0x00F4−0x012C
DISPC_VID1_CONV_COEF0	RW	32	0x0130
DISPC_VID1_CONV_COEF1	RW	32	0x0134
DISPC_VID1_CONV_COEF2	RW	32	0x0138
DISPC_VID1_CONV_COEF3	RW	32	0x013C
DISPC_VID1_CONV_COEF4	RW	32	0x0140
DISPC_VID2_BA0 − DISPC_VID2_BA1	RW	32	0x014C−0x0150
DISPC_VID2_POSITION	RW	32	0x0154
DISPC_VID2_SIZE	RW	32	0x0158
DISPC_VID2_ATTRIBUTES	RW	32	0x015C
DISPC_VID2_FIFO_THRESHOLD	RW	32	0x0160
DISPC_VID2_FIFO_SIZE_STATUS	R	32	0x0164
DISPC_VID2_ROW_INC	RW	32	0x0168
DISPC_VID2_PIXEL_INC	RW	32	0x016C
DISPC_VID2_FIR	RW	32	0x0170
DISPC_VID2_PICTURE_SIZE	RW	32	0x0174

Table 15−35. Display Controller Registers (Continued)

Register Name	Type	Register Width (Bits)	Offset
DISPC_VID2_ACCU0 − DISPC_VID2_ACCU1	RW	32	0x0178−0x017C
DISPC_VID2_FIR_COEF_H0 − DISPC_VID2_FIR_COEF_H7	RW	32	0x0180−0x01B8
DISPC_VID2_FIR_COEF_HV0 − DISPC_VID2_FIR_COEF_HV7	RW	32	0x0184−0x01BC
DISPC_VID2_CONV_COEF0	RW	32	0x01C0
DISPC_VID2_CONV_COEF1	RW	32	0x01C4
DISPC_VID2_CONV_COEF2	RW	32	0x01C8
DISPC_VID2_CONV_COEF3	RW	32	0x01CC
DISPC_VID2_CONV_COEF4	RW	32	0x01D0
DISPC_DATA_CYCLE1	RW	32	0x01D4
DISPC_DATA_CYCLE2	RW	32	0x01D8
DISPC_DATA_CYCLE3	RW	32	0x01DC

Table 15−36. Remote Frame Buffer Interface Registers

Register Name	Type	Register Width (Bits)	Offset
RFBI_REVISION	R	32	0x0000
RFBI_SYSCONFIG	RW	32	0x0010
RFBI_SYSSTATUS	R	32	0x0014
RFBI_CONTROL	RW	32	0x0040
RFBI_PIXEL_CNT	RW	32	0x0044
RFBI_LINE_NUMBER	RW	32	0x0048
RFBI_CMD	W	32	0x004C
RFBI_PARAM	W	32	0x0050
RFBI_DATA	W	32	0x0054
RFBI_READ	RW	32	0x0058
RFBI_STATUS	RW	32	0x005C
RFBI_CONFIG0 − RFBI_CONFIG1	RW	32	0x0060−0x0078
RFBI_ONOFF_TIME0 − RFBI_ONOFF_TIME1	RW	32	0x0064−0x007C
RFBI_CYCLE_TIME0 − RFBI_CYCLE_TIME1	RW	32	0x0068−0x0080
RFBI_DATA_CYCLE1_0 − RFBI_DATA_CYCLE1_1	RW	32	0x006C−0x0084
RFBI_DATA_CYCLE2_0 − RFBI_DATA_CYCLE2_1	RW	32	0x0070−0x0088

Table 15−36. Remote Frame Buffer Interface Registers (Continued)

Register Name	Type	Register Width (Bits)	Offset
RFBI_DATA_CYCLE3_0 – RFBI_DATA_CYCLE3_1	RW	32	0x0074– 0x008C
RFBI_VSYNC_WIDTH	RW	32	0x0090
RFBI_HSYNC_WIDTH	RW	32	0x0094

Table 15−37. Video Encoder Registers

Register Name	Type	Register Width (Bits)	Offset
REV_ID	R	32	0x0000
STATUS	R	32	0x0004
F_CONTROL	RW	32	0x0008
VIDOUT_CTRL	RW	32	0x0010
SYNC_CTRL	RW	32	0x0014
LLEN	RW	32	0x001C
FLENS	RW	32	0x0020
HFLTR_CTRL	RW	32	0x0024
CC_CARR_WSS_CARR	RW	32	0x0028
C_PHASE	RW	32	0x002C
GAIN_U	RW	32	0x0030
GAIN_V	RW	32	0x0034
GAIN_Y	RW	32	0x0038
BLACK_LEVEL	RW	32	0x003C
BLANK_LEVEL	RW	32	0x0040
X_COLOR	RW	32	0x0044
M_CONTROL	RW	32	0x0048
BSTAMP_WSS_DATA	RW	32	0x004C
S_CARR	RW	32	0x0050
LINE21	RW	32	0x0054
LN_SEL	RW	32	0x0058
L21__WC_CTL	RW	32	0x005C
SAVID__EAVID	RW	32	0x0064
FLEN__FAL	RW	32	0x0068
LAL__PHASE_RESET	RW	32	0x006C
HS_INT_START_STOP_X	RW	32	0x0070
HS_EXT_START_STOP_X	RW	32	0x0074
VS_INT_START_X	RW	32	0x0078
VS_INT_STOP_X__VS_INT_START_Y	RW	32	0x007C
VS_INT_STOP_Y__VS_EXT_START_X	RW	32	0x0080

Table 15−37. Video Encoder Registers (Continued)

Register Name	**Type**	**Register Width (Bits)**	**Offset**
VS_EXT_STOP_X__VS_EXT_START_Y	RW	32	0x0084
VS_EXT_STOP_Y	RW	32	0x0088
AVID_START_STOP_X	RW	32	0x0090
AVID_START_STOP_Y	RW	32	0x0094
FID_INT_START_X__FID_INT_START_Y	RW	32	0x00A0
FID_INT_OFFSET_Y__FID_EXT_START_X	RW	32	0x00A4
FID_EXT_START_Y__FID_EXT_OFFSET_Y	RW	32	0x00A8
TVDETGP_INT_START_STOP_X	RW	32	0x00B0
TVDETGP_INT_START_STOP_Y	RW	32	0x00B4
GEN_CTRL	RW	32	0x00B8
DAC_TST	RW	32	0x00C4
DAC	RW	32	0x00C8

15.6.1 DSS Registers

Table 15−38 through Table 15−42 describe the DISS register bits.

Table 15−38. DSS_REVISIONNUMBER

Address Offset	0x00		
Physical Address	0x4805 0000	**Instance**	DSS1
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	Reads return 0s.	R	0x000000
7:0	REV	Revision number [7:4] Major revision [3:0] Minor revision	R	

Table 15−39. DSS_SYSCONFIG

Address Offset	0x10		
Physical Address	0x4805 0010	**Instance**	DSS1
Description	This register controls the various parameters of the OCP 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 . Reads return 0s.	RW	0x00000000
1	SOFTRESET	Software reset. Set this bit 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	RW	0

Table 15−40. DSS_SYSSTATUS

Address Offset	0x14		
Physical Address	0x4805 0014	**Instance**	DSS1
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																								Reserved							RESETDONE

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Reserved for module-specific status information. Reads return 0s.	R	0x00000000
0	RESETDONE	Internal reset monitoring Read 0x0: Internal module reset is ongoing. Read 0x1: Reset is complete. †	R	0

† During reset the value is 0, but the value read immediately after the reset is 1 because ICLK/FCLK is already operating.

Table 15−41. DSS_CONTROL

Address Offset	0x40		
Physical Address	0x4805 0040	**Instance**	DISS1
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																						VIDEOPSAON	VIDEOPSACLR	LCDPSAON	LCDPSACLR	Reserved	DAC_DEMEN	VENC_CLOCK_4X_ENABLE	VENC_CLOCK_MODE	Reserved	DPLL_APLL_CLK

Bits	Field Name	Description	Type	Reset
31:10	Reserved	Reads return 0s.	R	0x000000
9	VIDEOPSAON	Enable the Video PSA algorithm.	RW	0
8	VIDEOPSACLR	Clear the Video PSA registers.	RW	0
7	LCDPSAON	Enable the LCD PSA algorithm.	RW	0
6	LCDPSACLR	Clear the LCD PSA registers.	RW	0

Table 15−41. DSS_CONTROL (Continued)

Bits	Field Name	Description	Type	Reset
5	Reserved	Reserved for future DAC uses	RW	0
4	DAC_DEMEN	DAC dynamic element matching enable	RW	0
3	VENC_CLOCK_4X_ENABLE	VENC clock 4x enable	RW	0
2	VENC_CLOCK_MODE	VENC clock mode 0x0: Mode 0 lets the video encoder receive a clock of 54 MHz, 49.08 MHz, or 59 MHz. 0x1: Mode 1 lets the video encoder receive a clock of 27 MHz, 24.5454 MHz, or 29.5 MHz.	RW	0
1	Reserved	Reserved	RW	0
0	DPLL_APLL_CLK	DPLL/APLL clock switch(0 for DPLL, 1 for APLL)	RW	0

Table 15−42. DSS_STATUS

Address Offset 0x5C

Physical Address 0x4805 005C **Instance** DISS1

Description This register contains the display subsystem 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																														APLL_ENABLE	DPLL_ENABLE

Bits	Field Name	Description	Type	Reset
31:2	Reserved	Reserved	R	0x00000000
1	APLL_ENABLE	APLL clock active 0x0: APLL disabled 0x1: APLL enabled	R	0
0	DPLL_ENABLE	DPLL clock active 0x0: DPLL disabled 0x1: DPLL enabled	R	1

15.6.2 Display Controller Registers

Table 15−43 through Table 15−99 describe the display controller register bits.

Table 15−43. DISPC_Revision

Address Offset	0x00000000		
Physical Address	0x4805 0400	**Instance**	DISC1
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	Reads return 0s.	R	0x000000
7:0	REV	IP revision [7:4] Major revision [3:0] Minor revision	R	

Table 15−44. DISPC_SYSCONFIG

Address Offset	0x00000010		
Physical Address	0x4805 0410	**Instance**	DISC1
Description	This register allows to control various parameters of the OCP 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							SIDLEMODE		RESERVED	SOFTRESET	AUTOIDLE

Bits	Field Name	Description	Type	Reset
31:14	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x00000
13:12	MIDLEMODE	Master interface power management, standby/wait control 0x0: Force standby: MSTANDBY is only asserted 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	RW	0x0
11:5	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x00

Table 15−44. DISPC_SYSCONFIG (Continued)

Bits	Field Name	Description	Type	Reset
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: Acknowledgement to an idle request is given based on the internal activity of the module. 0x3: Reserved	RW	0x0
2	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0
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 OCP clock-gating strategy 0x0: OCP clock is free-running. 0x1: Automatic OCP L3 and L4 clock-gating strategy is applied, based on the OCP interface activity	RW	0

Table 15−45. DISPC_SYSSTATUS

Address Offset	0x00000014		
Physical Address	0x4805 0414	**Instance**	DISC1
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

Table 15−45. DISPC_SYSSTATUS (Continued)

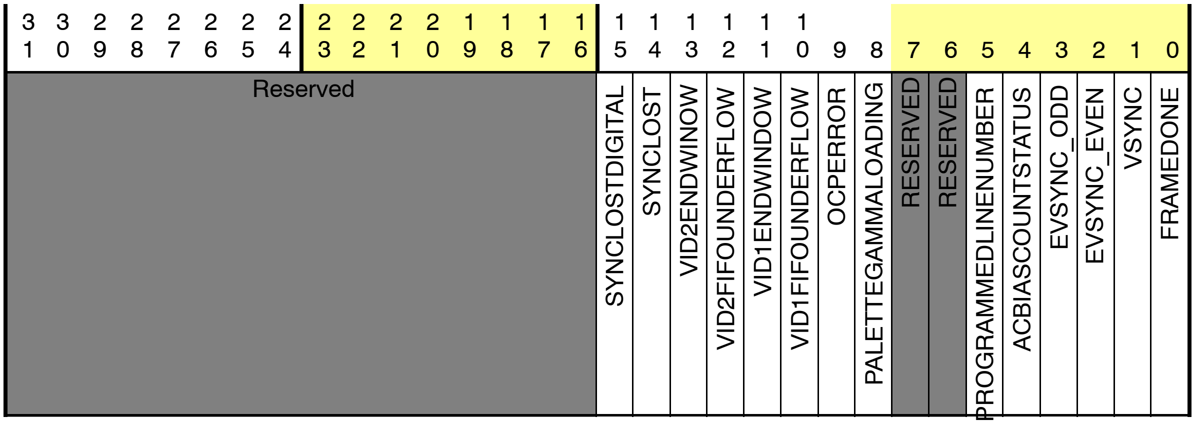
Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Write 0s for future compatibility. Reads return 0s.	R	0x000000
7:1	RESERVED	Reserved for OCP socket status information Reads return 0s.	R	0x00
0	RESETDONE †	Internal reset monitoring 0x0: Internal module reset is ongoing. 0x1: Reset completed ‡	R	0

† RESETDONE represents the status of all DISPC submodules. In other words, the RESETDONE field can report 1 only when all DISPC clocks (that is, L3 clock [DSS_L3_ICLK], L4 clock [DSS_L4_ICLK], DISPC functional clock [DSS_CLK1 or DSS_CLK2] and TV/DIGITAL clock [DSS_54M_CLK]) are provided during reset.

‡ During reset the value is 0, but the value read immediately after the reset is 1 because ICLK/FCLK is already operating.

Table 15−46. DISPC_IRQSTATUS

Address Offset	0x00000018		
Physical Address	0x4805 0418	**Instance**	DISC
Description	This register regroups the status of the module internal events that generate an interrupt. Writing 1 to a given bit resets the bit.		
Type	RW		



31:16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	SYNCLOSTDIGITAL	SYNCLOST	VID2ENDWINOW	VID2FIFOUNDERFLOW	VID1ENDWINDOW	VID1FIFOUNDERFLOW	OCPERROR	PALETTEGAMMALOADING	RESERVED	RESERVED	PROGRAMMEDLINENUMBER	ACBIASCOUNTSTATUS	EVSYNC_ODD	EVSYNC_EVEN	VSYNC	FRAMEDONE

Bits	Field Name	Description		Type	Reset
31:16	Reserved	Write 0s for future compatibility. Reads return 0s.		RW	0x00000
15	SYNCLOSTDIGITAL	SyncLostDigital			
		Read 0x0:	SyncLostDigital is false.	RW	0
		Write 0x0:	SyncLostDigital status bit is unchanged.		
		Read 0x1:	SyncLostDigital is true.		
		Write 0x1:	SynLostDigital status bit is reset.		
14	SYNCLOST	SyncLost		RW	0
		Read 0x0:	SyncLost is false.		
		Write 0x0:	SyncLost status bit is unchanged.		
		Read 0x1:	SyncLost is true.		
		Write 0x1:	SyncLost status bit is reset.		

Table 15−46. DISPC_IRQSTATUS (Continued)

Bits	Field Name	Description		Type	Reset
13	VID2ENDWINDOW	Vid2EndWindow		RW	0
		Read 0x0:	Vid2EndWindow is false.		
		Write 0x0:	Vid2EndWindow status bit is unchanged.		
		Read 0x1:	Vid2EndWindow is true.		
		Write 0x1:	Vid2EndWindow status bit is reset.		
12	VID2FIFOUNDERFLOW	Vid2FIFOUnderflow		RW	0
		Read 0x0:	Vid2FIFOUnderflow is false.		
		Write 0x0:	Vid2FIFOUnderflow status bit is unchanged		
		Read 0x1:	Vid2FIFOUnderflow is true.		
		Write 0x1:	Vid2FIFOUnderflow status bit is reset.		
11	VID1ENDWINDOW	Vid1EndWindow		RW	0
		Read 0x0:	Vid1EndWindow is false.		
		Write 0x0:	Vid1EndWindow status bit is unchanged.		
		Read 0x1:	Vid1EndWindow is true.		
		Write 0x1:	Vid1EndWindow status bit is reset.		
10	VID1FIFOUNDERFLOW	Vid1FIFOUnderflow		RW	0
		Read 0x0:	Vid1FIFOUnderflow is false.		
		Write 0x0:	Vid1FIFOUnderflow status bit is unchanged.		
		Read 0x1:	Vid1FIFOUnderflow is true.		
		Write 0x1:	Vid1FIFOUnderflow status bit is reset.		
9	OCPERROR	OCPError		RW	0
		Read 0x0:	OCPError is false.		
		Write 0x0:	OCPError status bit is unchanged.		
		Read 0x1:	OCPError is true.		
		Write 0x1:	OCPError status bit is reset .		
8	PALETTEGAMMALOADING	PaletteGammaLoading		RW	0
		Read 0x0:	PaletteGammaLoading is false.		
		Write 0x0:	PaletteGammaLoading status bit is unchanged.		
		Read 0x1:	PaletteGammaLoading is true.		
		Write 0x1:	PaletteGammaLoading status bit is reset.		

Table 15−46. DISPC_IRQSTATUS (Continued)

Bits	Field Name	Description	Type	Reset
7	Reserved	Write 0s for future compatibility. Reads return 0s.	RW	0
6	Reserved	Write 0s for future compatibility. Reads return 0s.	RW	0
5	PROGRAMMED LINENUMBER	ProgrammedLineNumber Read 0x0: ProgrammedLineNumber is false. Write 0x0: ProgrammedLineNumber status bit is unchanged. Read 0x1: ProgrammedLineNumber is true. Write 0x1: ProgrammedLineNumber status bit is reset.	RW	0
4	ACBIASCOUNT STATUS	ACBiasCountStatus Read 0x0: ACBiasCountStatus is false. Write 0x0: ACBiasCountStatus status bit is unchanged. Read 0x1: ACBiasCountStatus is true. Write 0x1: ACBiasCountStatus status bit is reset.	RW	0
3	EVSYNC_ODD	EVSYNC_ODD Read 0x0: EVSYNC_ODD is false. Write 0x0: EVSYNC_ODD status bit is unchanged. Read 0x1: EVSYNC_ODD is true. Write 0x1: EVSYNC_ODD status bit is reset.	RW	0
2	EVSYNC_EVEN	EVSYNC_EVEN Read 0x0: EVSYNC_EVEN is false. Write 0x0: EVSYNC_EVEN status bit is unchanged. Read 0x1: EVSYNC_EVEN is true. Write 0x1: EVSYNC_EVEN status bit is reset.	RW	0
1	VSYNC	VSYNC Read 0x0: VSYNC is false. Write 0x0: VSYNC status bit is unchanged. Read 0x1: VSYNC is true. Write 0x1: VSYNC status bit is reset.	RW	0
0	FRAMEDONE	FrameDone Read 0x0: FrameDone is false. Write 0x0: FrameDone status bit is unchanged. Read 0x1: FrameDone is true. Write 0x1: FrameDone status bit is reset.	RW	0

Table 15−47. DISPC_IRQENABLE

Address Offset	0x0000001C		
Physical Address	0x4805 041C	**Instance**	DISC
Description	This register allows the module internal sources of interrupt to be masked/unmasked 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																SYNCLOSTDIGITAL	SYNCLOST	VID2ENDWINDOW	VID2FIFOUNDERFLOW	ENDVID1WINDOW	VID1FIFOUNDERFLOW	OCPERROR	PALETTEGAMMAMASK	RESERVED	RESERVED	PROGRAMMEDLINENUMBER	ACBIASCOUNTSTATUS	EVSYNC_ODD	EVSYNC_EVEN	VSYNC	FRAMEMASK

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility. Reads return 0s.	RW	0x00000
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

Table 15–47. DISPC_IRQENABLE (Continued)

Bits	Field Name	Description		Type	Reset
12	VID2FIFOUNDERFLOW	Vid2FIFOUnderflow		RW	0
		0x0:	Vid2FIFOUnderflow is masked.		
		0x1:	Vid2FIFOUnderflow generates an interrupt when it occurs.		
11	ENDVID1WINDOW	EndVid1Window		RW	0
		0x0:	EndVid1Window is masked.		
		0x1:	EndVid1Window generates an interrupt when it occurs.		
10	VID1FIFOUNDERFLOW	Vid1FIFOUnderflow		RW	0
		0x0:	Vid1FIFOUnderflow is masked.		
		0x1:	Vid1FIFOUnderflow generates an interrupt when it occurs.		
9	OCPERROR	OCPError		RW	0
		0x0:	OCPError is masked.		
		0x1:	OCPError generates an interrupt when it occurs.		
8	PALETTEGAMMAMASK	PaletteGammaMask		RW	0
		0x0:	PaletteGammaMask is masked.		
		0x1:	PaletteGammaMask generates an interrupt when it occurs.		
7	Reserved	Write 0s for future compatibility. Reads return 0s.		RW	0
6	Reserved	Write 0s for future compatibility. Reads return 0s.		RW	0
5	PROGRAMMEDLINENUMBER	ProgrammedLineNumber		RW	0
		0x0:	ProgrammedLineNumber is masked.		
		0x1:	ProgrammedLineNumber generates an interrupt when it occurs.		
4	ACBIASCOUNT STATUS	ACBiasCountStatus		RW	0
		0x0:	ACBiasCountStatus is masked.		
		0x1:	ACBiasCountStatus generates an interrupt when it occurs.		

Table 15–47. DISPC_IRQENABLE (Continued)

Bits	Field Name	Description		Type	Reset
3	EVSYNC_ODD	EVSYNC_ODD		RW	0
		0x0:	EVSYNC_ODD is masked.		
		0x1:	EVSYNC_ODD generates an interrupt when it occurs.		
2	EVSYNC_EVEN	EVSYNC_EVEN		RW	0
		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 15–48. DISPC_CONTROL

Address Offset	0x00000040		
Physical Address	0x4805 0440	**Instance**	DISC1
Description	The control register allows to configure 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
Reserved					TDMUNUSEDBITS		TDMCYCLEFORMAT		TDMPARALLELMODE		TDMENABLE	HT			GPOUT1	GPOUT0	Reserved		OVERLAYOPTIMIZATION	RFBIMODE	RESERVED	TFTDATALINES		TFTDITHERENABLE	GODIGITAL	GOLCD	M8B	STNTFT	MONOCOLOR	DIGITALENABLE	LCDENABLE

Table 15−48. DISPC_CONTROL (Continued)

Bits	Field Name	Description	Type	Reset
31:27	Reserved	Write 0s for future compatibility. Reads return 0s.	RW	0x00
26:25	TDMUNUSED-BITS	State of unused bits (TDM mode only) wr: VFP 0x0: Low level (0) 0x1: High level (1) 0x2: Unchanged from previous state 0x3: Reserved	RW	0x0
24:23	TDMCYCLE FORMAT	Cycle format (TDM mode only) wr: VFP 0x0: 1 cycle for 1 pixel 0x1: 2 cycles for 1 pixel 0x2: 3 cycles for 1 pixel 0x3: 3 cycles for 2 pixels	RW	0x0
22:21	TDMPARALLEL-MODE	Output Interface width (TDM mode only) wr: VFP 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	RW	0x0
20	TDMENABLE	Enable the multiple cycle format (TDM mode only used for TFT mode with the RFBI enable bit off) wr: VFP 0x0: TDM disabled 0x1: TDM enabled	RW	0
19:17	HT	Hold time for digital output wr: EVSYNC Encoded value (from 0 to 7) holds time for digital output. The data is held for (HT+1) external digital clock periods.	RW	0x0
16	GPOUT1	General-purpose output signal 0x0: The GPout1 is reset. 0x1: The GPout1 is set.	RW	0
15	GPOUT0	General-purpose output signal 0x0: The GPout0 is reset. 0x1: The GPout0 is set.	RW	0
14:13	Reserved	Reserved	R	0x0

Table 15−48. DISPC_CONTROL (Continued)

Bits	Field Name	Description	Type	Reset
12	OVERLAY OPTIMIZATION	Overlay optimization (available when graphics format is not 1, 2, or 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	RFBIMODE	RFBI mode for the LCD output wr: VFP 0x0: Normal mode selected 0x1: RFBI 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 software must reenable the LCD output to generate a new frame.	RW	0
10	Reserved	Write 0s for future compatibility. Reads return 0s.	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	TFTDITHER ENABLE	TFT dithering enable wr: VFP 0x0: TFT dithering logic disabled 0x1: TFT dithering logic enabled	RW	0

Table 15−48. DISPC_CONTROL (Continued)

Bits	Field Name	Description		Type	Reset
6	GODIGITAL	Digital GO Command		RW	0
		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 complete.		
		0x1:	The user has 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.		
5	GOLCD	LCD GO Command		RW	0
		0x0:	The hardware has finished updating the internal shadow registers of the pipeline(s) connected with the LCD output using the user values. The hardware resets the bit when the update is complete.		
		0x1:	The user has 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.		
4	M8B	Mono 8-bit mode wr: VFP		RW	0
		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).		
3	STNTFT	LCD display type wr: VFP		RW	0
		0x0:	Passive or STN display operation is enabled. STN dither logic is enabled.		
		0x1:	Active or TFT display operation is enabled. STN dither logic and output FIFO are bypassed.		
2	MONOCOLOR	Monochrome/color wr: VFP		RW	0
		0x0:	Color operation enabled (STN mode only)		
		0x1:	Monochrome operation enabled (STN mode only)		
1	DIGITALENABLE	Digital enable		RW	0
		0x0:	Digital output disabled (at the end of the current field if interlace output when the bit is reset)		
		0x1:	Digital output enabled		

Table 15−48. DISPC_CONTROL (Continued)

Bits	Field Name	Description	Type	Reset
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

Note: The clock selection becomes active at the next inactive part of the video frame (VFP) after GOLCD is set.

Table 15−49. DISPC_CONFIG

Address Offset	0x00000044		
Physical Address	0x4805 0444	**Instance**	DISC1
Description	The control register allows to configure 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																		TCKDIGSELECTION	TCKDIGENABLE	TCKLCDSELECTION	TCKLCDENABLE	FUNCGATED	ACBIASGATED	VSYNCGATED	HSYNCGATED	PIXELCLOCKGATED	PIXELDATAGATED	PALETTEGAMMATABLE	LOADMODE		PIXELGATED

Bits	Field Name	Description	Type	Reset
31:14	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x00000
13	TCKDIG SELECTION	Transparency color key selection (digital output) wr: EVSYNC 0x0: Graphics destination transparency color key selected 0x1: Video source transparency color key selected	RW	0
12	TCKDIGENABLE	Transparency color key enabled (digital output) wr: EVSYNC 0x0: Disable the transparency color key for the digital output 0x1: Enable the transparency color key for the digital output	RW	0

Table 15−49. DISPC_CONFIG (Continued)

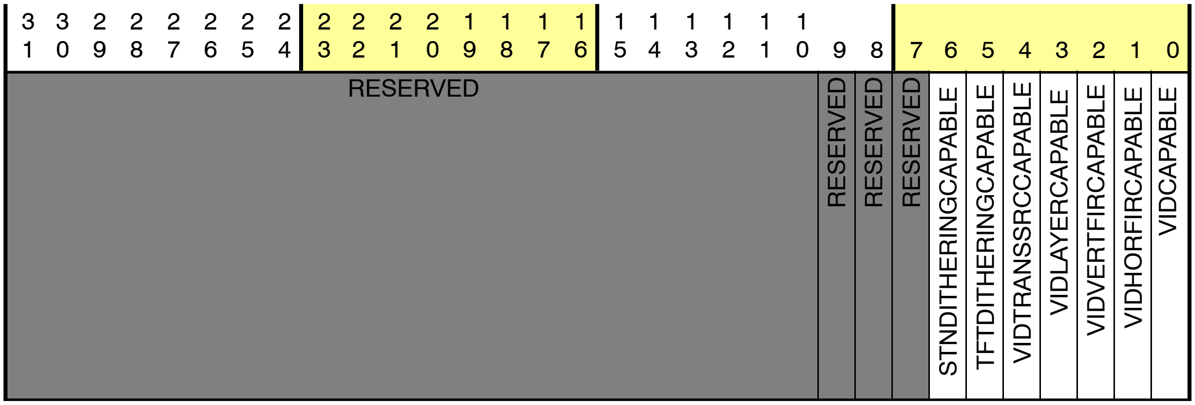
Bits	Field Name	Description	Type	Reset
11	TCKLCD SELECTION	Transparency color key selection (LCD output) wr: VFP 0x0: Graphics destination transparency color key selected 0x1: Video source transparency color key selected	RW	0
10	TCKLCDENABLE	Transparency color key enabled (LCD output) wr: VFP * 0x0: Disable the transparency color key for the LCD 0x1: Enable the transparency color key for the LCD	RW	0
9	FUNCGATED	Functional clocks autogated enabled This bit enables the autogating mechanism. It is always set independently of the clock mechanism. 0x0: Functional clocks autogated disabled 0x1: Functional clocks autogated 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	PIXELCLOCK-GATED	Pixel clock gated enabled wr: VFP 0x0: Pixel clock gated disabled 0x1: Pixel clock gated enabled	RW	0
4	PIXELDATA-GATED	Pixel data gated enabled wr: VFP 0x0: Pixel data gated disabled 0x1: Pixel data gated enabled	RW	0

Table 15−49. DISPC_CONFIG (Continued)

Bits	Field Name	Description	Type	Reset
3	PALETTE GAMMATABLE	Palette/gamma table selection wr: EVSYNC or VFP 0x0: LUT used as palette (only if graphics format is BITMAP1, 2, 4, or 8) 0x1: LUT used as gamma table (only if graphics format is not BITMAP1, 2, 4, or 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. The user sets the bit when the palette/gamma table must be loaded. Hardware resets the bit when table has been loaded. 0x2: Frame data only loaded every frame. 0x3: Palette/gamma table and frame data loaded on first frame, then switch to 10 (hardware)	RW	0x0
0	PIXELGATED	Pixel gated enable (only for TFT) wr: VFP 0x0: Pixel clock always toggles (only in TFT mode). 0x1: Pixel clock toggles only when there is valid data to display (only in TFT mode).	RW	0

Table 15−50. DISPC_CAPABLE

Address Offset	0x00000048		
Physical Address	0x4805 0448	**Instance**	DISC1
Description	The control register allows to configure the display controller capability. 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																						RESERVED	RESERVED	RESERVED	STNDITHERINGCAPABLE	TFTDITHERINGCAPABLE	VIDTRANSSRCCAPABLE	VIDLAYERCAPABLE	VIDVERTFIRCAPABLE	VIDHORFIRCAPABLE	VIDCAPABLE

Table 15−50. DISPC_CAPABLE (Continued)

Bits	Field Name	Description	Type	Reset
31:10	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x000000
9:7	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	−
6	STNDITHERING-CAPABLE	STNditheringCapable 0x0: STN dithering not capable 0x1: STN dithering capable	RW	−
5	TFTDITHERING-CAPABLE	TFTditheringCapable TFTditheringCapable 0x0: TFT dithering not capable 0x1: TFT dithering capable	RW	−
4	VIDTRANSSRC-CAPABLE	VidTransSrcCapable 0x0: Video src. transparency color key not capable 0x1: Video src. transparency color key capable	RW	−
3	VIDLAYER CAPABLE	VidLayerCapable 0x0: Video layer not capable 0x1: Video layer capable	RW	−
2	VIDVERTFIR-CAPABLE	VidVertFIRCapable 0x0: Video vertical upsampling not capable 0x1: Video vertical upsampling capable	RW	−
1	VIDHORFIR CAPABLE	VidHorFIRCapable 0x0: Video horizontal upsampling not capable 0x1: Video horizontal upsampling capable	RW	−
0	VIDCAPABLE	VidCapable 0x0: Video path not available 0x1: Video path available	RW	−

Table 15−51. DISPC_DEFAULT_COLOR0...DISPC_DEFAULT_COLOR1

Address Offset	0x0000004C−0x00000050 in 0x4-byte increments		
Physical Address	0x4805 044C−0x4805 0450	**Instance**	DISC1
Description	The control register allows to configure the default solid background color for the output #i (0 for LCD output, 1 for 24-bit digital output). Shadow register, updated on VFP start period for i = 0 and EVSYNC for i = 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																							

Table 15−51. DISPC_DEFAULT_COLOR0...DISPC_DEFAULT_COLOR1 (Continued)

Bits	Field Name	Description	Type	Reset
31:24	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x00
23:0	DEFAULTCOLOR	24-bit RGB color value specifies the default solid color to display when there is no data from the overlays.	RW	0x000000

Table 15−52. DISPC_TRANS_COLOR0...DISPC_TRANS_COLOR1

Address Offset	0x00000054−0x00000058 in 0x4 byte increments		
Physical Address	0x4805 0454−0x4805 0458	**Instance**	DISC1
Description	This register sets the transparency color value for the video/graphics overlays for the output #i (0 for LCD output, 1 for 24-bit digital output). Shadow register, updated on VFP start period for i = 0 and EVSYNC for i = 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. Reads return 0s.	RW	0x00
23:0	TRANSCOLOR-KEY	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 15−53. DISPC_LINE_STATUS

Address Offset	0x0000005C		
Physical Address	0x4805 045C	**Instance**	DISC1
Description	The control register indicates the current LCD panel display line number.		
Type	R		

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="21">RESERVED</td><td colspan="11">LINENUMBER</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
31:11	RESERVED	Write 0s for future compatibility. Reads return 0s.	R	0x000000
10:0	LINENUMBER	Current LCD panel line number Current display line number. The first active line has the value 0. The line number is not incremented during blanking lines.	R	0x7FF

Table 15−54. DISPC_LINE_NUMBER

Address Offset	0x00000060		
Physical Address	0x4805 0460	**Instance**	DISC1
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		

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="21">RESERVED</td><td colspan="11">LINENUMBER</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
31:11	RESERVED	Write 0s for future compatibility. Reads return 0s.	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 15−55. DISPC_TIMING_H

Address Offset	0x00000064		
Physical Address	0x4805 0464	**Instance**	DISC1
Description	This register configures the timing logic for the HSYNC signal. Shadow register, updated on VFP start period.		
Type	RW		

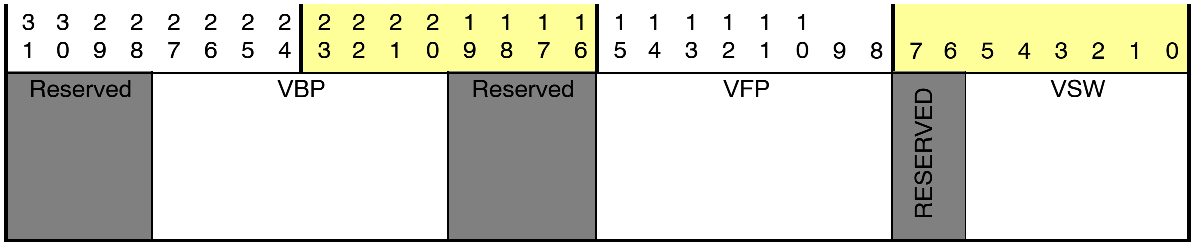
<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="4">Reserved</td><td colspan="8">HBP</td><td colspan="4">Reserved</td><td colspan="8">HFP</td><td colspan="2">RESERVED</td><td colspan="6">HSW</td></tr>
</table>

Table 15−55. DISPC_TIMING_H (Continued)

Bits	Field Name	Description	Type	Reset
31:28	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x0
27:20	HBP	Horizontal back porch Encoded value (from 1 to 256) specifies 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 one).	RW	0x00
19:16	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x0
15:8	HFP	Horizontal front porch Encoded value (from 1 to 256) specifies the number of pixel clock periods to add to the end of a line transmission before line clock is asserted (program to value minus one).	RW	0x00
7:6	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x0
5:0	HSW	Horizontal synchronization pulse width Encoded value (from 1 to 64) specifies the number of pixel clock periods to pulse the line clock at the end of each line (program to value minus one).	RW	0x00

Table 15−56. DISPC_TIMING_V

Address Offset	0x00000068		
Physical Address	0x4805 0468	**Instance**	DISC1
Description	This 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
Reserved				VBP								Reserved				VFP								RESERVED		VSW					

Bits	Field Name	Description	Type	Reset
31:28	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x0
27:20	VBP	Vertical back porch Encoded value (from 0 to 255) specifies 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
19:16	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x0
15:8	VFP	Vertical front porch Encoded value (from 0 to 255) specifies the number of line clock periods to add to the end of each frame.	RW	0x00

Table 15–56. DISPC_TIMING_V (Continued)

Bits	Field Name	Description	Type	Reset
7:6	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x0
5:0	VSW	Vertical synchronization pulse width In active mode, encoded value (from 1 to 64) specifies 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 64) specifies the number of extra line clock periods to insert after the vertical front porch (VFP) period has elapsed.	RW	0x00

Table 15–57. DISPC_POL_FREQ

Address Offset	0x0000006C		
Physical Address	0x4805 046C	**Instance**	DISC1
Description	This 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. Reads return 0s.	RW	0x0000
17	ONOFF	HSYNC/VSYNC pixel clock control on/off 0x0: HSYNC and VSYNC are driven on opposite edges of pixel clock from 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 is set to 1). 0x1: HSYNC and VSYNC are driven on rising edge of pixel clock (if bit 17 is 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

Table 15−57. DISPC_POL_FREQ (Continued)

Bits	Field Name	Description	Type	Reset
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
11:8	ACBI	ac-bias pin transitions per interrupt Value (from 0 to 15) specifies the number of ac-bias pin transitions.	RW	0x0
7:0	ACB	ac-bias pin frequency Value (from 0 to 255) specifies 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 buildup within the display.	RW	0x00

Table 15−58. DISPC_DIVISOR

Address Offset	0x00000070		
Physical Address	0x4805 0470	**Instance**	DISC1
Description	This 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. Reads return 0s.	RW	0x00
23:16	LCD	Display controller logic clock divisor Value (from 1 to 255) specifies the frequency of the display controller logic clock based on the function clock. The value 0 is invalid.	RW	0x01

Table 15−58. DISPC_DIVISOR (Continued)

Bits	Field Name	Description	Type	Reset
15:8	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x00
7:0	PCD	Pixel clock divisor Value (from 2 to 255) specifies the frequency of the pixel clock based on the logic clock, which is the functional clock divided by the LCD. The values 0 and 1 are invalid. For TFT LCD panel, minimum PCD value is 0x02. For STN LCD panel, PCD value is 0x03.	RW	0x02

Table 15−59. DISPC_SIZE_DIG

Address Offset	0x00000078		
Physical Address	0x4805 0478	**Instance**	DISC1
Description	This 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. Reads return 0s.	RW	0x00
26:16	LPP	Lines per panel Encoded value (from 1 to 2048) specifies the number of lines per panel (program to value minus one).	RW	0x000
15:11	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x00
10:0	PPL	Pixels per line Encoded value (from 1 to 2048) specifies the number of pixels contained within each line on the display (program to value minus one). The line width value must be set to a multiple of 8 pixels. (ex: PPL=0x7)	RW	0x000

Table 15−60. DISPC_SIZE_LCD

Address Offset	0x0000007C		
Physical Address	0x4805 047C	**Instance**	DISC1
Description	This 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										

Table 15−60. DISPC_SIZE_LCD (Continued)

Bits	Field Name	Description	Type	Reset
31:27	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x00
26:16	LPP	Lines per panel Encoded value (from 1 to 2048) specifies the number of lines per panel (program to value minus one).	RW	0x000
15:11	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x00
10:0	PPL	Pixels per line Encoded value (from 1 to 2048) specifies the number of pixels contained within each line on the display (program to value minus one). When running in normal mode (RFBI mode is bypassed by setting DISPC_CONTROL.RFBIMODE = 0), the line width value must be set to a multiple of 8 pixels (ex: PPL = 0x7).	RW	0x000

Table 15−61. DISPC_VID1_BA0...DISPC_VID1_BA1

Address Offset	0x000000BC−0x000000C0 in 0x4 byte increments		
Physical Address	0x4805 04BC−0x4805 04C0	**Instance**	DISC1
Description	This register configures the base address of the video buffer for the video window #n (#i 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 15−62. DISPC_VID1_POSITION

Address Offset	0x000000C4		
Physical Address	0x4805 04C4	**Instance**	DISC1
Description	This register configures the position of the 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. Reads return 0s.	RW	0x00
26:16	VIDPOSY	Y position of the video window #n Encoded value (from 0 to 2047) specifies the Y position of the video window #n.The line at the top has the Y-position 0.	RW	0x000
15:11	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x00
10:0	VIDPOSX	X position of the video window #n Encoded value (from 0 to 2047) specifies the X position of the video window #n. The first pixel on the left of the display screen has the X-position 0.	RW	0x000

Table 15−63. DISPC_VID1_SIZE

Address Offset	0x000000C8		
Physical Address	0x4805 04C8	**Instance**	DISC1
Description	This register configures the size of the 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. Reads return 0s.	RW	0x00
26:16	VIDSIZEY	Number of lines of the video #n Encoded value (from 1 to 2048) specifies the number of lines of the video window #n (program to value minus one).	RW	0x000
15:11	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x00
10:0	VIDSIZEX	Number of pixels of the video window #n Encoded value (from 1 to 2048) specifies the number of pixels of the video window #n (program to value minus one).	RW	0x000

Table 15−64. DISPC_VID1_ATTRIBUTES

Address Offset	0x000000CC		
Physical Address	0x4805 04CC	**Instance**	DISC
Description	This register configures the attributes of the video window #n such as format, video transparency color key enable, resize enable, and stride distance. 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													VIDROWREPEATENABLE	VIDENDIANESS	VIDCHANNELOUT	VIDBURSTSIZE		VIDROTATION		VIDFULLRANGE	VIDREPLICATIONENABLE	VIDCOLORCONVENABLE	VIDVRESIZECONF	VIDHRESIZECONF	VIDRESIZEENABLE		VIDFORMAT				VIDENABLE

Table 15−64. DISPC_VID1_ATTRIBUTES (Continued)

Bits	Field Name	Description	Type	Reset
31:19	Reserved	Write 0s for future compatibility. Reads return 0s.	RW	0x0000
18	VIDROW REPEATENABLE	Video row repeat (YUV case only when rotating 90 or 270 degrees) 0x0: Row of VIDn will not be read twice. 0x1: The row data are fetched twice to extract both Y components.	RW	0
17	VIDENDIANNESS	Video endianness 0x0: Little endian operation is selected. 0x1: Big endian operation is selected.	RW	0
16	VIDCHANNEL-OUT	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 0x3: Rotation by 270 degrees	RW	0x0
11	VIDFULLRANGE	VidFullRange 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.	RW	0
10	VIDREPLICA TIONENABLE	VidReplicationEnable 0x0: Disable video-replication logic. 0x1: Enable video-replication logic.	RW	0
9	VIDCOLORCON-VENABLE	VidColorConvEnable 0x0: Disable Color Space Conversion CbYCr to RGB. 0x1: Enable Color Space Conversion CbYCr to RGB.	RW	0
8	VIDVRESIZE-CONF	Video vertical resize configuration 0x0: Upsampling selected 0x1: Downsampling selected	RW	0

Table 15−64. DISPC_VID1_ATTRIBUTES (Continued)

Bits	Field Name	Description	Type	Reset
7	VIDHRESIZE-CONF	Video horizontal resize configuration 0x0: Upsampling selected 0x1: Downsampling selected	RW	0
6:5	VIDRESIZE ENABLE	Video resize enable 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.	RW	0x0
4:1	VIDFORMAT	Video format. Other enums: Reserved. 0x6: RGB 16 0x8: RGB 24 (unpack in 32-bit container) 0x9: RGB 24 (unpack in 24-bit container) 0xA: YUV2 4:2:2 co-sited 0xB: UYVY 4:2:2 co-sited	RW	0x0
0	VIDENABLE	VidEnable 0x0: Video disabled (video pipeline inactive and window not present) 0x1: Video enabled (video pipeline active and window present on the screen)	RW	0

Table 15−65. DISPC_VID1_FIFO_THRESHOLD

Address Offset	0x000000D0		
Physical Address	0x4805 04D0	**Instance**	DISC1
Description	This register configures the video FIFO associated with the 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
31:25	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x00
24:16	VIDFIFOHIGH-THRESHOLD	Video FIFO high threshold Number of bytes defining the threshold value	RW	0x–––
15:9	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x00
8:0	VIDFIFOLOW-THRESHOLD	Video FIFO high threshold Number of bytes defining the threshold value	RW	0x–––

Table 15−66. DISPC_VID1_FIFO_SIZE_STATUS

Address Offset	0x000000D4		
Physical Address	0x4805 04D4	**Instance**	DISC1
Description	This register defines the video FIFO size for the 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
31:9	RESERVED	Write 0s for future compatibility. Reads return 0s.	R	0x000000
8:0	VIDFIFOSIZE	Video FIFO size Number of bytes defining the FIFO value	R	0x−−−

Table 15−67. DISPC_VID1_ROW_INC

Address Offset	0x000000D8		
Physical Address	0x4805 04D8	**Instance**	DISC1
Description	This register configures the number of bytes to increment at the end of the row for the buffer associated with the 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 $[-2^{31}-1: 2^{31}]$ specifies 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 15−68. DISPC_VID1_PIXEL_INC

Address Offset	0x000000DC		
Physical Address	0x4805 04DC	**Instance**	DISC1
Description	This register configures the number of bytes to increment between two pixels for the buffer associated with the 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. Reads return 0s.	RW	0x0000
15:0	VIDPIXELINC	Number of bytes to increment at the end of the row. Encoded signed value $[-2^{15}-1: 2^{15}]$ specifies 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 15−69. DISPC_VID1_FIR

Address Offset	0x000000E0		
Physical Address	0x4805 04E0	**Instance**	DISC1
Description	This register configures the resize factors for horizontal and vertical up-/downsampling of the 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
31:28	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x0
27:16	VIDFIRVINC	Vertical increment of the up-/downsampling filter Encoded value (from 1 to 2048). The value 0 is invalid. Values greater than 2048 are invalid.	RW	0x000
15:12	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x0
11:0	VIDFIRHINC	Horizontal increment of the up-/downsampling filter. Encoded value (from 1 to 2048). The value 0 is invalid. Values greater than 2048 are invalid.	RW	0x000

Table 15−70. DISPC_VID1_PICTURE_SIZE

Address Offset	0x000000E4		
Physical Address	0x4805 04E4	**Instance**	DISC1
Description	This register configures the size of the video picture associated with the video layer #n before upscaling/downscaling. 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. Reads return 0s.	RW	0x00
26:16	VIDORGSIZEY	Number of lines of the video picture. Encoded value (from 1 to 2048) specifies 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. Reads return 0s.	RW	0x00
10:0	VIDORGSIZEX	Number of pixels of the video picture Encoded value (from 1 to 2048) specifies 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.	RW	0x000

Table 15−71. DISPC_VID1_ACCU0...DISPC_VID1_ACCU1

Address Offset	0x000000E8−0x000000EC in 0x4 byte increments		
Physical Address	0x4805 04E8−0x4805 04EC	**Instance**	DISC1
Description	This register configures the resize accumulator initialization values for horizontal and vertical up-/downsampling of the 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. Reads return 0s.	RW	0x00
25:16	VIDVERTICAL ACCU	Vertical initialization accumulator value Encoded value (from 0 to 1023)	RW	0x000
15:10	RESERVED	Write 0s for future compatibility Reads return 0s.	RW	0x00
9:0	VIDHORIZONTAL ACCU	Horizontal initialization accumulator value Encoded value (from 0 to 1023)	RW	0x000

Table 15−72. DISPC_VID1_FIR_COEF_H0...DISPC_VID1_FIR_COEF_H7

Address Offset	0x000000F0−0x00000128 in 0x8 byte increments		
Physical Address	0x4805 04F0−0x4805 0528	**Instance**	DISC1
Description	The bank of registers configure the up-/downscaling coefficients for the vertical and horizontal resize of the video picture associated with the video window #n for the phase i. 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-/downscaling with the phase n	RW	0x00
23:16	VIDFIRHC2	Unsigned coefficient C2 for the horizontal up-/downscaling with the phase n	RW	0x00
15:8	VIDFIRHC1	Signed coefficient C1 for the horizontal up-/downscaling with the phase n	RW	0x00
7:0	VIDFIRHC0	Signed coefficient C0 for the horizontal up-/downscaling with the phase n	RW	0x00

Table 15−73. DISPC_VID1_FIR_COEF_HV0...DISPC_VID1_FIR_COEF_HV7

Address Offset	0x000000F4−0x0000012C in 0x8 byte increments		
Physical Address	0x4805 04F4−0x4805 052C	**Instance**	DISC1
Description	The bank of registers configure the down-/up-/downscaling coefficients for the vertical and horizontal resize of the video picture associated with the video window #n for the phase i. 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-/downscaling with the phase n	RW	0x00
23:16	VIDFIRVC1	Unsigned coefficient C1 for the vertical up-/downscaling with the phase n	RW	0x00
15:8	VIDFIRVC0	Signed coefficient C0 for the vertical up-/downscaling with the phase n	RW	0x00
7:0	VIDFIRHC4	Signed coefficient C4 for the horizontal up-/downscaling with the phase n	RW	0x00

Table 15−74. DISPC_VID1_CONV_COEF0

Address Offset	0x00000130		
Physical Address	0x4805 0530	**Instance**	DISC1
Description	This register configures the color space conversion matrix coefficients for the 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. Reads return 0s.	RW	0x00
26:16	RCR	RCr coefficient Encoded signed value (from −1024 to 1023)	RW	0x000
15:11	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x00
10:0	RY	RY coefficient Encoded signed value (from −1024 to 1023)	RW	0x000

Table 15−75. DISPC_VID1_CONV_COEF1

Address Offset	0x00000134		
Physical Address	0x4805 0534	**Instance**	DISC1
Description	This register configures the color space conversion matrix coefficients for the 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. Reads return 0s.	RW	0x00
26:16	GY	GY coefficient Encoded signed value (from −1024 to 1023)	RW	0x000
15:11	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x00
10:0	RCB	RCb coefficient Encoded signed value (from −1024 to 1023)	RW	0x000

Table 15−76. DISPC_VID1_CONV_COEF2

Address Offset	0x00000138		
Physical Address	0x4805 0538	**Instance**	DISC1
Description	This register configures the color space conversion matrix coefficients for the 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. Reads return 0s.	RW	0x00
26:16	GCB	GCb coefficient Encoded signed value (from −1024 to 1023)	RW	0x000
15:11	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x00
10:0	GCR	GCr coefficient Encoded signed value (from −1024 to 1023)	RW	0x000

Table 15−77. DISPC_VID1_CONV_COEF3

Address Offset	0x0000013C		
Physical Address	0x4805 053C	**Instance**	DISC1
Description	This register configures the color space conversion matrix coefficients for the 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. Reads return 0s.	RW	0x00
26:16	BCR	BCr coefficient Encoded signed value (from −1024 to 1023)	RW	0x000
15:11	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x00
10:0	BY	BY coefficient Encoded signed value (from −1024 to 1023)	RW	0x000

Table 15−78. DISPC_VID1_CONV_COEF4

Address Offset	0x00000140		
Physical Address	0x4805 0540	**Instance**	DISC1
Description	This register configures the color space conversion matrix coefficients for the 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. Reads return 0s.	RW	0x000000
10:0	BCB	BCb coefficient Encoded signed value (from −1024 to 1023)	RW	0x000

Table 15−79. DISPC_VID2_BA0...DISPC_VID2_BA1

Address Offset	0x0000014C−0x00000150 in 0x4 byte increments		
Physical Address	0x4805 054C−0x4805 0550	**Instance**	DISC1
Description	This register configures the base address of the video buffer for the video window #n (#i 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 15−80. DISPC_VID2_POSITION

Address Offset	0x00000154		
Physical Address	0x4805 0554	**Instance**	DISC1
Description	This register configures the position of the 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. Reads return 0s.	RW	0x00
26:16	VIDPOSY	Y position of the video window #n Encoded value (from 0 to 2047) specifies the Y position of the video window #n.The line at the top has the Y-position 0.	RW	0x000
15:11	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x00
10:0	VIDPOSX	X position of the video window #n Encoded value (from 0 to 2047) specifies the X position of the video window #n. The first pixel on the left of the display screen has the X-position 0.	RW	0x000

Table 15−81. DISPC_VID2_SIZE

Address Offset	0x00000158		
Physical Address	0x4805 0558	**Instance**	DISC1
Description	This register configures the size of the 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. Reads return 0s.	RW	0x00
26:16	VIDSIZEY	Number of lines of the video #n Encoded value (from 1 to 2048) specifies the number of lines of the video window #n (program to value minus one).	RW	0x000
15:11	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x00
10:0	VIDSIZEX	Number of pixels of the video window #n Encoded value (from 1 to 2048) specifies the number of pixels of the video window #n (program to value minus one).	RW	0x000

Table 15−82. DISPC_VID2_ATTRIBUTES

Address Offset	0x0000015C		
Physical Address	0x4805 055C	**Instance**	DISC
Description	This register configures the attributes of the video window #n such as format, video transparency color key enable, resize enable and stride distance. Shadow register updated on VFP start period or EVSYNC		
Type	RW		

31–19	18	17	16	15–14	13–12	11	10	9	8	7	6–5	4–1	0
RESERVED	VIDROWREPEATENABLE	VIDENDIANESS	VIDCHANNELOUT	VIDBURSTSIZE	VIDROTATION	VIDFULLRANGE	VIDREPLICATIONENABLE	VIDCOLORCONVENABLE	VIDVRESIZECONF	VIDHRESIZECONF	VIDRESIZEENABLE	VIDFORMAT	VIDENABLE

Bits	Field Name	Description	Type	Reset
31:19	Reserved	Write 0s for future compatibility. Reads return 0s.	RW	0x0000
18	VIDROW REPEATENABLE	Video row repeat (YUV case only when rotating 90 degrees or 270 degrees) 0x0: Row of VIDn will not be read twice. 0x1: The row data are fetched twice to extract both Y components.	RW	0
17	VIDENDIANNESS	Video endianness Will be fetched twice to extract both Y components. 0x0: Little endian operation is selected. 0x1: Big endian operation is selected.	RW	0
16	VIDCHANNEL-OUT	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 0x3: Rotation by 270 degrees	RW	0x0

Table 15−82. DISPC_VID2_ATTRIBUTES (Continued)

Bits	Field Name	Description		Type	Reset
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	VIDREPLICATIONENABLE	VidReplicationEnable		RW	0
		0x0:	Disable video replication logic.		
		0x1:	Enable video replication logic.		
9	VIDCOLOR CONVENABLE	VidColorConvEnable		RW	0
		0x0:	Disable color space conversion CbYCr to RGB		
		0x1:	Enable color space conversion CbYCr to RGB		
8	VIDVRESIZE-CONF	Video vertical resize configuration		RW	0
		0x0:	Upsampling selected		
		0x1:	Downsampling selected		
7	VIDHRESIZE-CONF	Video horizontal resize configuration		RW	0
		0x0:	Upsampling selected		
		0x1:	Downsampling selected		
6:5	VIDRESIZE ENABLE	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 format. Other enums: Reserved.		RW	0x0
		0x6:	RGB 16		
		0x8:	RGB 24 (unpack in 32-bit container)		
		0x9:	RGB 24 (unpack in 24-bit container)		
		0xA:	YUV2 4:2:2 co-sited		
		0xB:	UYVY 4:2:2 co-sited		
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 15−83. DISPC_VID2_FIFO_THRESHOLD

Address Offset	0x00000160		
Physical Address	0x4805 0560	**Instance**	DISC1
Description	This register configures the video FIFO associated with the 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
31:25	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x00
24:16	VIDFIFOHIGH-THRESHOLD	Video FIFO high threshold Number of bytes defining the threshold value	RW	0x−−−
15:9	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x00
8:0	VIDFIFOLOW-THRESHOLD	Video FIFo high threshold Number of bytes defining the threshold value	RW	0x−−−

Table 15−84. DISPC_VID2_FIFO_SIZE_STATUS

Address Offset	0x00000164		
Physical Address	0x4805 0564	**Instance**	DISC1
Description	This register defines the video FIFO size for the 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
31:9	RESERVED	Write 0s for future compatibility. Reads return 0s.	R	0x000000
8:0	VIDFIFOSIZE	Video FIFO size Number of bytes defining the FIFO value	R	0x−−−

Table 15−85. DISPC_VID2_ROW_INC

Address Offset	0x00000168		
Physical Address	0x4805 0568	**Instance**	DISC1
Description	This register configures the number of bytes to increment at the end of the row for the buffer associated with the 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 $[-2^{31}-1: 2^{31}]$ specifies 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 15−86. DISPC_VID2_PIXEL_INC

Address Offset	0x0000016C		
Physical Address	0x4805 056C	**Instance**	DISC1
Description	This register configures the number of bytes to increment between two pixels for the buffer associated with the 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. Reads return 0s.	RW	0x0000
15:0	VIDPIXELINC	Number of bytes to increment at the end of the row Encoded signed value $[-2^{15}-1: 2^{15}]$ specifies 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 15−87. DISPC_VID2_FIR

Address Offset	0x00000170		
Physical Address	0x4805 0570	**Instance**	DISC1
Description	This register configures the resize factors for horizontal and vertical up-/downsampling of the 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
31:28	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x0
27:16	VIDFIRVINC	Vertical increment of the up-/downsampling filter Encoded value (from 1 to 2048). The value 0 is invalid. The values greater than 2048 are invalid.	RW	0x000
15:12	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x0
11:0	VIDFIRHINC	Horizontal increment of the up-/downsampling filter Encoded value (from 1 to 2048). The value 0 is invalid. The values greater than 2048 are invalid.	RW	0x000

Table 15−88. DISPC_VID2_PICTURE_SIZE

Address Offset	0x00000174		
Physical Address	0x4805 0574	**Instance**	DISC1
Description	This register configures the size of the video picture associated with the video layer #n before up-/downscaling. 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. Reads return 0s.	RW	0x00
26:16	VIDORGSIZEY	Number of lines of the video picture Encoded value (from 1 to 2048) specifies 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. Reads return 0s.	RW	0x00
10:0	VIDORGSIZEX	Number of pixels of the video picture Encoded value (from 1 to 2048) specifies 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.	RW	0x000

Table 15−89. DISPC_VID2_ACCU0...DISPC_VID2_ACCU1

Address Offset	0x00000178−0x0000017C in 0x4 byte increments		
Physical Address	0x4805 0578−0x4805 057C	**Instance**	DISC1
Description	This register configures the resize accumulator initialization values for horizontal and vertical up-/downsampling of the 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. Reads return 0s.	RW	0x00
25:16	VIDVERTICAL ACCU	Vertical initialization accumulator value Encoded value (from 0 to 1023)	RW	0x000
15:10	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x00
9:0	VIDHORIZONTAL ACCU	Horizontal initialization accumulator value Encoded value (from 0 to 1023)	RW	0x000

Table 15−90. DISPC_VID2_FIR_COEF_HO...DISPC_VID2_FIR_COEF_H7

Address Offset	0x00000180−0x000001B8 in 0x8 byte increments		
Physical Address	0x4805 0580−0x4805 05B8	**Instance**	DISC1
Description	The bank of registers configure the up-/downscaling coefficients for the vertical and horizontal resize of the video picture associated with the video window #n for the phase i. 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-/downscaling with the phase n	RW	0x00
23:16	VIDFIRHC2	Unsigned coefficient C2 for the horizontal up-/downscaling with the phase n	RW	0x00
15:8	VIDFIRHC1	Signed coefficient C1 for the horizontal up-/downscaling with the phase n	RW	0x00
7:0	VIDFIRHC0	Signed coefficient C0 for the horizontal up-/downscaling with the phase n	RW	0x00

Table 15−91. DISPC_VID2_FIR_COEF_HVO...DISPC_VID2_FIR_COEF_HV7

Address Offset	0x00000184−0x000001BC in 0x8 byte increments		
Physical Address	0x4805 0584−0x4805 05BC	**Instance**	DISC1
Description	The bank of registers configure the down-/up-/downscaling coefficients for the vertical and horizontal resize of the video picture associated with the video window #n for the phase i. 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-/downscaling with the phase n	RW	0x00
23:16	VIDFIRVC1	Unsigned coefficient C1 for the vertical up-/downscaling with the phase n	RW	0x00
15:8	VIDFIRVC0	Signed coefficient C0 for the vertical up-/downscaling with the phase n	RW	0x00
7:0	VIDFIRHC4	Signed coefficient C4 for the horizontal up-/downscaling with the phase n	RW	0x00

Table 15−92. DISPC_VID2_CONV_COEF0

Address Offset	0x000001C0		
Physical Address	0x4805 05C0	**Instance**	DISC1
Description	This register configures the color space conversion matrix coefficients for the 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. Reads return 0s.	RW	0x00
26:16	RCR	RCr coefficient Encoded signed value (from 1024 to 1023)	RW	0x000
15:11	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x00
10:0	RY	RY coefficient Encoded signed value (from 1024 to 1023)	RW	0x000

Table 15−93. DISPC_VID2_CONV_COEF1

Address Offset	0x000001C4		
Physical Address	0x4805 05C4	**Instance**	DISC1
Description	This register configures the color space conversion matrix coefficients for the 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. Reads return 0s.	RW	0x00
26:16	GY	GY coefficient Encoded signed value (from 1024 to 1023)	RW	0x000
15:11	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x00
10:0	RCB	RCb coefficient Encoded signed value (from 1024 to 1023)	RW	0x000

Table 15−94. DISPC_VID2_CONV_COEF2

Address Offset	0x000001C8		
Physical Address	0x4805 05C8	**Instance**	DISC1
Description	This register configures the color space conversion matrix coefficients for the 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. Reads return 0s.	RW	0x00
26:16	GCB	GCb coefficient Encoded signed value (from1024 to 1023)	RW	0x000
15:11	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x00
10:0	GCR	GCr coefficient Encoded signed value (from 1024 to 1023)	RW	0x000

Table 15−95. DISPC_VID2_CONV_COEF3

Address Offset	0x000001CC		
Physical Address	0x4805 05CC	**Instance**	DISC1
Description	This register configures the color space conversion matrix coefficients for the video pipeline #n. Shadow register, updated on VFP start period or EVSYNC		
Type	RW		

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="5">RESERVED</td><td colspan="11">BCR</td><td colspan="5">RESERVED</td><td colspan="11">BY</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
31:27	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x00
26:16	BCR	BCr coefficient Encoded signed value (from 1024 to 1023)	RW	0x000
15:11	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x00
10:0	BY	BY coefficient Encoded signed value (from 1024 to 1023)	RW	0x000

Table 15−96. DISPC_VID2_CONV_COEF4

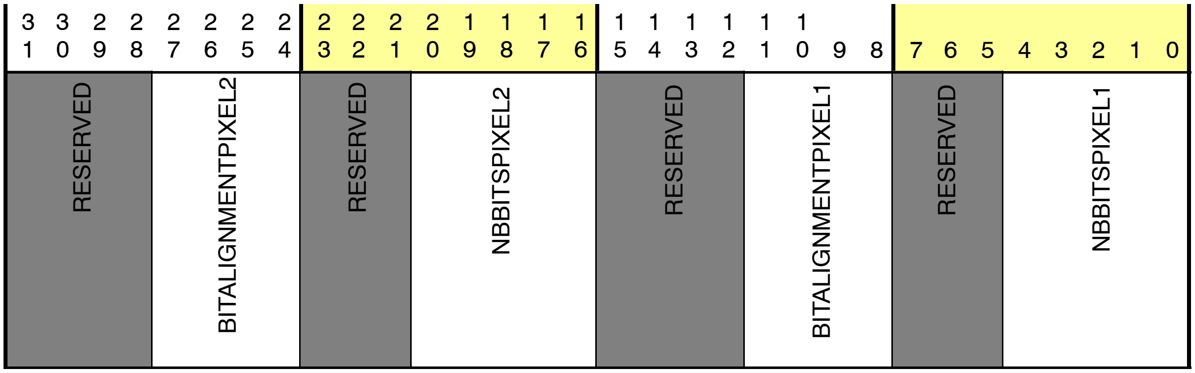
Address Offset	0x000001D0		
Physical Address	0x4805 05D0	**Instance**	DISC1
Description	This register configures the color space conversion matrix coefficients for the video pipeline #n. Shadow register, updated on VFP start period or EVSYNC		
Type	RW		

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="21">RESERVED</td><td colspan="11">BCB</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
31:11	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x000000
10:0	BCB	BCb coefficient Encoded signed value (from 1024 to 1023)	RW	0x000

Table 15−97. DISPC_DATA_CYCLE1

Address Offset	0x000001D4		
Physical Address	0x4805 05D4	**Instance**	DISC1
Description	The control register configures the output data format for 1st cycle. Shadow register, updated on VFP start period		
Type	RW		

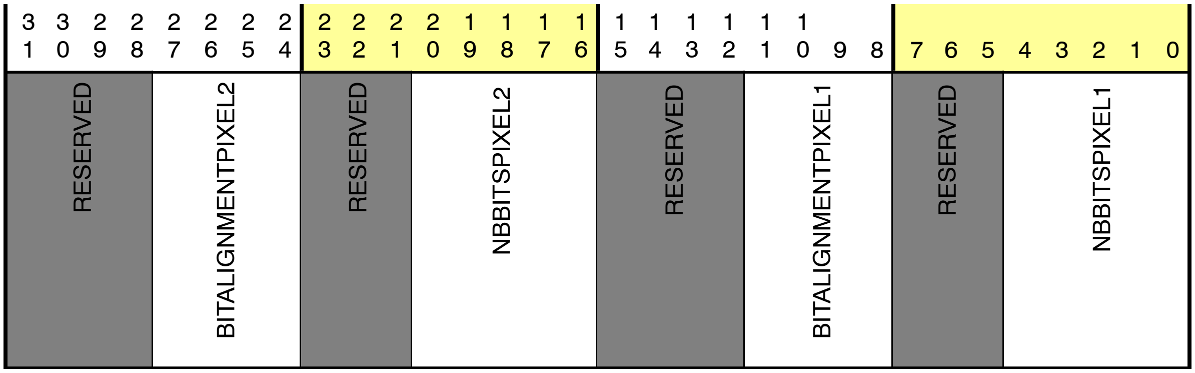


31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				BITALIGNMENTPIXEL2				RESERVED			NBBITSPIXEL2					RESERVED				BITALIGNMENTPIXEL1				RESERVED			NBBITSPIXEL1				

Bits	Field Name	Description	Type	Reset
31:28	RESERVED	Write 0s for future compatibility. Reads return 0s.	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. Reads return 0s.	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. Reads return 0s.	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. Reads return 0s.	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 15–98. DISPC_DATA_CYCLE2

Address Offset	0x000001D8		
Physical Address	0x4805 05D8	**Instance**	DISC1
Description	The control register configures the output data format for 2nd 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 Reads return 0s.	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 Reads return 0s.	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 Reads return 0s.	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 Reads return 0s.	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 15−99. DISPC_DATA_CYCLE3

Address Offset	0x000001DC		
Physical Address	0x4805 05DC	**Instance**	DISC1
Description	The control register configures the output data format for 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 Reads return 0s.	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 Reads return 0s.	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 Reads return 0s.	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 Reads return 0s.	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

15.6.3 Remote Frame Buffer Interface Registers

Table 15−100 through Table 15−118 describe the RFBI register bits.

Table 15−100. RFBI_REVISION

Address Offset	0x000		
Physical Address	0x4805 0800	**Instance**	RFBI1
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	Reads return 0s.	R	0x000000
7:0	REV	IP revision [7:4] Major revision [3:0] Minor revision	R	

Table 15−101. RFBI_SYSCONFIG

Address Offset	0x010		
Physical Address	0x4805 0810	**Instance**	RFBI1
Description	This register allows to control various parameters of the OCP 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

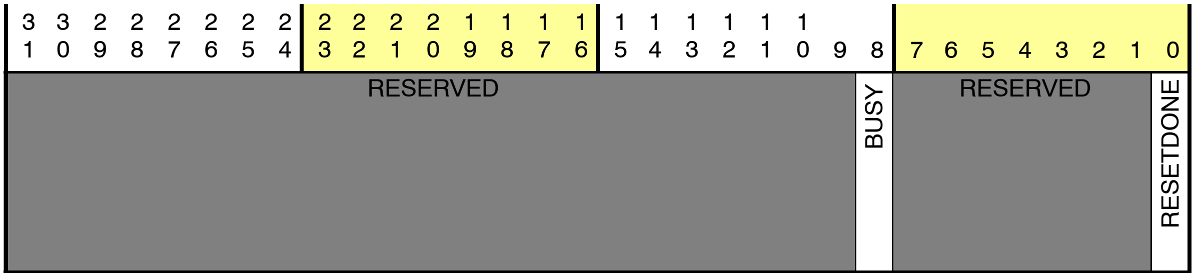
Bits	Field Name	Description	Type	Reset
31:7	RESERVED	Write 0s for future compatibility Reads return 0s.	RW	0x0000000
6	RESERVED	Write 0s for future compatibility Reads return 0.	RW	0
5	RESERVED	Write 0s for future compatibility Reads return 0.	RW	0
4:3	SIDLEMODE	Slave interface power management, idle req/ack control 00: Force-idle. An idle request is acknowledged unconditionally. 01: No-idle. An idle request is never acknowledged 10: Smart-idle. Acknowledgement to an idle request is given based on the internal activity of the module. 11: Reserved	RW	0x0

Table 15−101. RFBI_SYSCONFIG (Continued)

Bits	Field Name	Description	Type	Reset
2	RESERVED	Write 0s for future compatibility Reads return 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 (OCP L4 and display controller clock) 0: OCP L4 clock and display controller clock are free-running. 1: Automatic clock-gating strategy is applied for the OCP L4 clock and display controller clock, based on the OCP interface and internal activity.	RW	0

Table 15−102. RFBI_SYSSTATUS

Address Offset	0x014		
Physical Address	0x4805 0814	**Instance**	RFBI1
Description	This register provides status information about the module, excluding the interrupt status information.		
Type	R		



Bits	Field Name	Description	Type	Reset
31:9	RESERVED	Reserved for module-specific status information Read returns 0	R	0x000000
8	BUSY	L4 Interface busy status bit 0: Access to the following register is not stalled: RFBI_CMD, RFBI_DATA, RFBI_STATUS, RFBI_PARAM, RFBI_READ. 1: 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 for OCP socket status information Read returns 0	R	0x00
0	RESETDONE	Internal reset monitoring 0: Internal module reset is ongoing. 1: Reset completed†	R	0

† During reset the value is 0, but the value read just after the reset is 1 because ICLK/FCLK is already operating.

Table 15−103. RFBI_CONTROL

Address Offset	0x040		
Physical Address	0x4805 0840	**Instance**	RFBI1
Description	The control register allows to configure 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																											ITE	CONFIGSELECT		BYPASSMODE	ENABLE

Bits	Field Name	Description	Type	Reset
31:5	RESERVED	Write 0s for future compatibility Reads return 0s.	RW	0x0000000
4	ITE	Internal trigger 0: Hardware 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 hardware takes into account the bit, the hardware 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 is 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 15−104. RFBI_PIXEL_CNT

Address Offset	0x044		
Physical Address	0x4805 0844	**Instance**	RFBI1
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 software 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 hardware decrements the register when a pixel has been sent to the RFB.	RW	0x00000000

Table 15−105. RFBI_LINE_NUMBER

Address Offset	0x048		
Physical Address	0x4805 0848	**Instance**	RFBI1
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 Reads return 0s.	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 15−106. RFBI_CMD

Address Offset	0x04C		
Physical Address	0x4805 084C	**Instance**	RFBI1
Description	The control register configures the RFBI command.		
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
Reserve																CMD															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Write 0s for future compatibility Reads return 0s.	W	0x0000
15:0	CMD	Command Value 8-/9-/12-/16-bit value depending on the parallel mode [7:0] 8-bit DataType [8:0] 9-bit DataType [11:0] 12-bit DataType [15:0] 16-bit DataType	W	0x0000

Table 15−107. RFBI_PARAM

Address Offset	0x050		
Physical Address	0x4805 0850	**Instance**	RFBI1
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 Reads return 0s.	W	0x0000
15:0	PARAM	Param Value 8-/9-/12-/16-bit value depending on the parallel mode [7:0] 8-bit DataType [8:0] 9-bit DataType [11:0] 12-bit DataType [15:0] 16-bit DataType	W	0x0000

Table 15−108. RFBI_DATA

Address Offset	0x054		
Physical Address	0x4805 0854	**Instance**	RFBI1
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 DataType [11:0] 12-bit DataType [15:0] 16-bit DataType [17:0] 18-bit DataType [23:0] 24-bit DataType [31:0] 2x16-bit DataType	W	0x00000000

Table 15−109. RFBI_READ

Address Offset	0x058		
Physical Address	0x4805 0858	**Instance**	RFBI1
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 Reads return 0s.	RW	0x0000
15:0	READ	Read value 8-/9-/12-/16-bit value depending on the parallel mode [7:0] 8-bit DataType [8:0] 9-bit DataType [11:0] 12-bit DataType [15:0] 16-bit DataType	RW	0x0000

Table 15−110. RFBI_STATUS

Address Offset	0x05C		
Physical Address	0x4805 085C	**Instance**	RFBI1
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 Reads return 0s.	RW	0x0000
15:0	STATUS	Status value 8-/9-/12-/16-bit value depending on the parallel mode [7:0] 8-bit DataType [8:0] 9-bit DataType [11:0] 12-bit DataType [15:0] 16-bit DataType	RW	0x0000

Table 15−111. RFBI_CONFIG0...RFBI_CONFIG1

Address Offset	0x060−0x078 in 0x18 byte increments		
Physical Address	0x4805 0860−0x4805 0878	**Instance**	RFBI1
Description	The control register allows to set the 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	DATATYPE	TIMEGRANULARITY	TRIGGERMODE	PARALLELMODE

Bits	Field Name	Description	Type	Reset
31:22	RESERVED	Write 0s for future compatibility Reads return 0s.	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
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 Reads return 0s.	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

Table 15−111.   RFBI_CONFIG0...RFBI_CONFIG1 (Continued)

Bits	Field Name	Description	Type	Reset
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 pixel at the position [31:16]	RW	0x0
6:5	DATATYPE	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 2 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 (tearing effect signal) 10: External trigger mode (RFB_TE_VYSNC/ RFB_HSYNC with programmable line counter) 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 15−112. RFBI_ONOFF_TIME0...RFBI_ONOFF_TIME1

Address Offset	0x064−0x07C in 0x18 byte increments		
Physical Address	0x4805 0864−0x4805 087C	**Instance**	RFBI1
Description	The control register allows to configure the RFBI 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
RESERVED		REOFFTIME						REONTIME				WEOFFTIME						WEONTIME				CSOFFTIME						CSONTIME			

Bits	Field Name	Description	Type	Reset
31:30	RESERVED	Write 0s for future compatibility Reads return 0s.	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 15−113. RFBI_CYCLE_TIME0...RFBI_CYCLE_TIME1

Address Offset	0x068−0x080 in 0x18 byte increments		
Physical Address	0x4805 0868−0x4805 0880	**Instance**	RFBI1
Description	The control register allows to configure the RFBI 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
Reserved				ACCESSTIME						WRENABLE	WWENABLE	RRENABLE	RWENABLE	CSPULSEWIDTH						RECYCLETIME						WECYCLETIME					

Bits	Field Name	Description	Type	Reset
31:28	RESERVED	Write 0s for future compatibility Reads return 0s.	RW	0x0
27:22	ACCESSTIME	Access time Number of L4Clk cycles	RW	0x00

Table 15−113. RFBI_CYCLE_TIME0...RFBI_CYCLE_TIME1 (Continued)

Bits	Field Name	Description	Type	Reset
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 15−114. RFBI_DATA_CYCLE1_0...RFBI_DATA_CYCLE1_1

Address Offset	0x06C−0x084 in 0x18 byte increments		
Physical Address	0x4805 086C−0x4805 0884	**Instance**	RFBI1
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				BITALIGNMENT PIXEL2				RESERVED			NBBITS PIXEL2					Reserved				BITALIGNMENT PIXEL1				RESERVED			NBBITS PIXEL1				

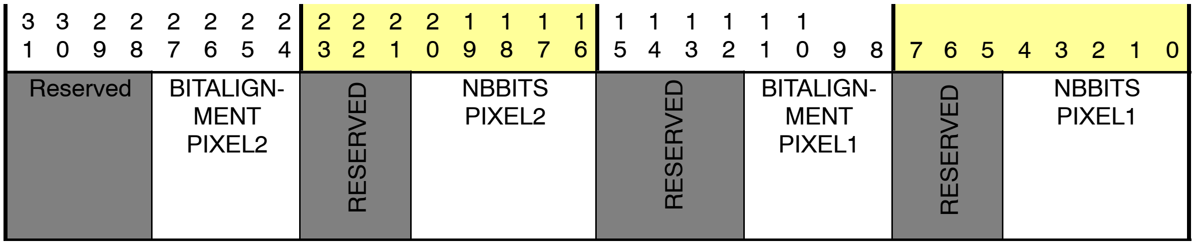
Bits	Field Name	Description	Type	Reset
31:28	RESERVED	Write 0s for future compatibility Reads return 0s.	RW	0x0
27:24	BITALIGNMENTPIXEL2	Bit alignment Alignment of the bits from pixel#2 on the output interface	RW	0x0
23:21	RESERVED	Write 0s for future compatibility Reads return 0s.	RW	0x0
20:16	NBBITSPIXEL2	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

Table 15−114. RFBI_DATA_CYCLE1_0...RFBI_DATA_CYCLE1_1 (Continued)

Bits	Field Name	Description	Type	Reset
15:12	RESERVED	Write 0s for future compatibility Reads return 0s.	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 Reads return 0s.	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 15−115. RFBI_DATA_CYCLE2_0...RFBI_DATA_CYCLE2_1

Address Offset	0x070−0x088 in 0x18 byte increments		
Physical Address	0x4805 0870−0x4805 0888	**Instance**	RFBI1
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				BITALIGN-MENT PIXEL2				RESERVED			NBBITS PIXEL2					RESERVED				BITALIGN-MENT PIXEL1				RESERVED			NBBITS PIXEL1				

Bits	Field Name	Description	Type	Reset
31:28	RESERVED	Write 0s for future compatibility Reads return 0s.	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 Reads return 0s.	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 Reads return 0s.	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 Reads return 0s.	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 15−116. RFBI_DATA_CYCLE3_0...RFBI_DATA_CYCLE3_1

Address Offset	0x074−0x08C in 0x18 byte increments		
Physical Address	0x4805 0874−0x4805 088C	**Instance**	RFBI1
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				BITALIGNMENT PIXEL2				RESERVED			NBBITS PIXEL2					Reserved				BITALIGNMENT PIXEL1				RESERVED			NBBITS PIXEL1				

Bits	Field Name	Description	Type	Reset
31:28	RESERVED	Write 0s for future compatibility Reads return 0s.	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 Reads return 0s.	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 Reads return 0s.	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 Reads return 0s.	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 15−117. RFBI_VSYNC_WIDTH

Address Offset	0x090		
Physical Address	0x4805 0890	**Instance**	RFBI1
Description	The control register configures the RFBI VSYNC minimum pulse width (on TE 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																MINVSYNCPULSEWIDTH															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Write 0s for future compatibility Reads return 0s.	RW	0x0000
15:0	MINVSYNC PULSEWIDTH	Programmable min VSYNC pulse width Minimum VSYNC pulse width from 0 to $2^{16}-1$. Number of L4 clock cycles to determine on the TE signal when VSYNC pulse occurs. The values 0 and 1 are invalid.	RW	0x0000

Table 15−118. RFBI_HSYNC_WIDTH

Address Offset	0x094		
Physical address	0x4805 0894	**Instance**	RFBI1
Description	The control register configures the RFBI HSYNC minimum pulse width.		
Type	RW		
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																MINHSYNCPULSEWIDTH															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Write 0s for future compatibility Reads return 0s.	RW	0x0000
15:0	MINHSYNC PULSEWIDTH	Programmable min HSYNC pulse width Minimum HSYNC pulse width from 0 to $2^{16}-1$. Number of L4 clock cycles to determine when HSYNC pulse occurs. The values 0 and 1 are invalid.	RW	0x0000

15.6.4 Video Encoder Registers

Table 15−119 through Table 15−160 describe the video encoder registers.

Table 15−119. REV_ID

Address Offset	0x000		
Physical Address	0x4805 0C00	**Instance**	VENC1
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		RW	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	

Table 15−120. STATUS

Address Offset	0x004		
Physical Address	0x4805 0C04	**Instance**	VENC1
Description	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		RW	0x0000000
4	CCE	Closed-caption status for even field This bit is set immediately after the data in registers LINE21_E0 and LINE21_E1 are 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 are 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. 0x0: ODD field 0x1: EVEN field	R	0x0

Table 15−121. F_CONTROL

Address Offset	0x008		
Physical Address	0x4805 0C08	**Instance**	VENC1
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		

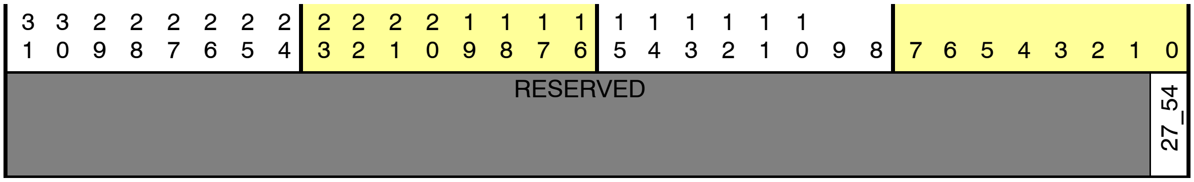
Bits	Field Name	Description	Type	Reset
31:9	RESERVED		RW	0x000000
8	RESET	Reset the encoder 0x0: No effect 0x1: Reset the encoder; after reset, this bit is automatically set to 0.	RW	0

Table 15−121. F_CONTROL (Continued)

Bits	Field Name	Description	Type	Reset
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 to 255. The input YCrCb data are in binary format with coding range 0 to 255. 0x1: The input RGB data are in binary format with coding range 16 to 235. The input YCrCb data are in binary format conforming to ITU-601 standard.	RW	0
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: Reserved 0x2: Reserved 0x3: 8-bit ITU-R 656 4:2:2	RW	0x3

Table 15−122. VIDOUT_CTRL

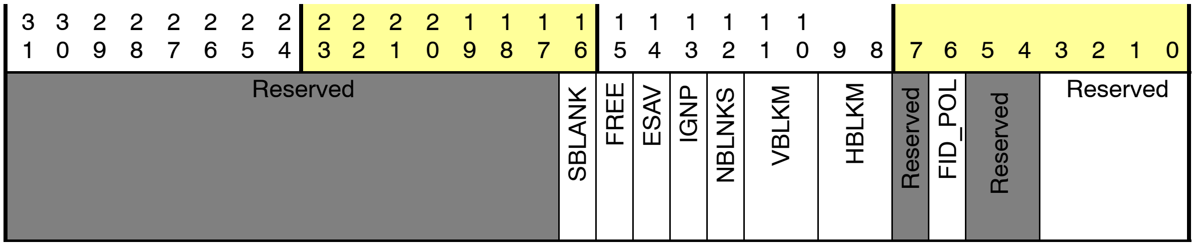
Address Offset	0x010		
Physical Address	0x4805 0C10	**Instance**	VENC1
Description	Encoder output clock		
Type	RW		



Bits	Field Name	Description		Type	Reset
31:1	RESERVED			RW	0x00000000
0	27_54	Encoder output clock		RW	0
		0x0:	54 MHz, 4x oversampling		
		0x1:	27 MHz, 2x oversampling, the last 2x oversampling filter bypassed		

Table 15−123. SYNC_CTRL

Address Offset	0x014		
Physical Address	0x4805 0C14	**Instance**	VENC1
Description			
Type	RW		



Bits	Field Name	Description		Type	Reset
31:17	Reserved	reserved_5		RW	0x0000
16	SBLANK	Data output enable		RW	0
		0x0:	No functionality		
		0x1:	Enables the output of data when SYNC_CTRL [11:10] VBLKM bits are set to 0b01		
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		

Table 15−121. F_CONTROL (Continued)

Bits	Field Name	Description		Type	Reset
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 FAL and LAL **Note:** In this mode, the LAL_PHASE_ RESET[16] SBLANK bit must be set to 0b1 to activate 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 SAVID and EAVID		
		0x2:	External blanking defined by AVID		
		0x3:	Reserved		
7	Reserved	Reserved		RW	0
6	FID_POL	FID input polarity		RW	0
		0x0:	ODD field = 0 EVEN field = 1		
		0x1:	ODD field = 1 EVEN field = 0		
5:0	Reserved			RW	0x00

Table 15−124. LLEN

Address Offset	0x01C		
Physical Address	0x4805 0C1C	**Instance**	VENC1
Description	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																					LLEN										

Bits	Field Name	Description	Type	Reset
31:11	RESERVED		RW	0x000000
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.	RW	0x359

Note: A write on the LLEN[10] is illegal.

Table 15−125. FLENS

Address Offset	0x020		
Physical Address	0x4805 0C20	**Instance**	VENC1
Description	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		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	0x03C

Table 15−126. HFLTR_CTRL

Address Offset	0x024		
Physical Address	0x4805 0C24	**Instance**	VENC1
Description	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		RW	0x00000000
2:1	CINTP	Chrominance interpolation filter control 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.	RW	0x0
0	YINTP	Luminance interpolation filter control 0x0: The luminance interpolation filter is enabled. 0x1: The luminance interpolation filter is bypassed.	RW	0

Table 15−127. CC_CARR_WSS_CARR

Address Offset	0x028		
Physical Address	0x4805 0C28	**Instance**	VENC1
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 50 Hz systems, FWSS = $2^{18} * 5*10^6/(LLEN*Fh)$ Where LLEN = Total number of pixels in a scan line Fh = Line frequency	RW	0x043F
15:0	FCC	Closed-caption run-in code frequency control For 60-Hz system, FCC = $2^{18}* 0.5035*10^6/(LLEN*Fh)$ For 50-Hz systems, FCC = $2^{18}* 0.500*10^6/(LLEN*Fh)$ Where LLEN = Total number of pixels in a scan line Fh = Line frequency	RW	0x2631

Table 15−128. C_PHASE

Address Offset	0x02C		
Physical Address	0x4805 0C2C	**Instance**	VENC1
Description	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_13																								CPHS							

Bits	Field Name	Description	Type	Reset
31:8	RESERVED_13		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 15−129. GAIN_U

Address Offset	0x030		
Physical Address	0x4805 0C30	**Instance**	VENC1
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		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 15−130. GAIN_V

Address Offset	0x034		
Physical Address	0x4805 0C34	**Instance**	VENC1
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_15																							GV								

Bits	Field Name	Description	Type	Reset
31:9	RESERVED_15		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 15−131. GAIN_Y

Address Offset	0x038		
Physical Address	0x4805 0C38	**Instance**	VENC1
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		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 15−132. BLACK_LEVEL

Address Offset	0x03C		
Physical Address	0x4805 0C3C	**Instance**	VENC1
Description	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		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 15−133. BLANK_LEVEL

Address Offset	0x040		
Physical Address	0x4805 0C40	**Instance**	VENC1
Description	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		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 15−134. X_COLOR

Address Offset	0x044		
Physical Address	0x4805 0C44	**Instance**	VENC1
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
RESERVED																									XCE	RESERVED	XCBW		LCD		

Bits	Field Name	Description	Type	Reset
31:7	RESERVED		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		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

Table 15−134. X_COLOR (Continued)

Bits	Field Name	Description	Type	Reset
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 15−135. M_CONTROL

Address Offset	0x048		
Physical Address	0x4805 0C48	**Instance**	VENC1
Description	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	RW	0x000000
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

Table 15−135. M_CONTROL (Continued)

Bits	Field Name	Description	Type	Reset
4:2	CBW	Chrominance lowpass filter bandwidth control 0x0: −6 dB at 21.8 % of encoding pixel clock frequency 0x1: −6 dB at 19.8 % of encoding pixel clock frequency 0x2: −6 dB at 18.0 % of encoding pixel clock frequency 0x3: Reserved 0x4: Reserved 0x5: −6 dB at 23.7 % of encoding pixel clock frequency 0x6: −6 dB 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	Field rate selection 0x0: 50 Hz 0x1: 60 Hz	RW	1

Table 15−136. BSTAMP_WSS_DATA

Address Offset	0x04C		
Physical Address	0x4805 0C4C	**Instance**	VENC1
Description	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_D																				SQP	BSTAP						

Table 15−136. BSTAMP_WSS_DATA (Continued)

Bits	Field Name	Description			Type	Reset
31:28	Reserved				RW	0x0
27:8	WSS_D	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. See the FFRQ bit of the M_CONTROL register (Table 15−135) 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 15−137. S_CARR

Address Offset	0x050		
Physical Address	0x4805 0C50	**Instance**	VENC1
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 are used to program color subcarrier frequency. They are determined by the following formula: S_CARR = ROUND((Fsc/Fclkenc) * 232) 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 Fh = Line frequency See Table 15−124 for the description of the LLEN registers (subaddresses 0x42 and 0x43).	RW	0x21F07C1F

Table 15−138. LINE21

Address Offset	0x054		
Physical Address	0x4805 0C54	**Instance**	VENC1
Description	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	RW	0x0000
15:0	L21O	The two bytes of the closed-caption data in the odd field	RW	0x0000

Table 15−139. LN_SEL

Address Offset	0x058		
Physical Address	0x4805 0C58	**Instance**	VENC1
Description	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																											SLINE				

Bits	Field Name	Description	Type	Reset
31:5	RESERVED		RW	0x0000000
4:0	SLINE	Selects the line where closed-caption or extended-service data are encoded	RW	0x15

Table 15−140. L21_WC_CTL

Address Offset	0x05C		
Physical Address	0x4805 0C5C	**Instance**	VENC1
Description	L21 & WC_CTL 8-bit 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	

Table 15−140. L21_WC_CTL (Continued)

Bits	Field Name	Description	Type	Reset
31:16	RESERVED		RW	0x0000
15	INV	WSS inverter 0x0: No effect 0x1: Invert WSS data	RW	0
14:13	EVEN_ODD_EN	This bit controls the WSS encoding. 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	RW	0x0
12:8	LINE	Selects the line where WSS data are encoded	RW	0x14
7:2	RESERVED		RW	0x00
1:0	L21EN	Those bits controls the Line21 closed-caption encoding according to the mode. 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	RW	0x0

Table 15−141. SAVID_EAVID

Address Offset	0x064		
Physical Address	0x4805 0C64	**Instance**	VENC1
Description	SAVID and EAVID		
Type	RW		

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="5">RESERVED</td><td colspan="11">EAVID</td><td colspan="5">RESERVED</td><td colspan="11">SAVID</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
31:27	RESERVED		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		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 15−142. FLEN_FAL

Address Offset	0x068		
Physical Address	0x4805 0C68	**Instance**	VENC1
Description	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		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		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 15−143. LAL_PHASE_RESET

Address Offset	0x06C		
Physical Address	0x4805 0C6C	**Instance**	VENC1
Description	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		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 C_Phase (subaddress 5A) on reset. 0x3: Reset every four fields. Color subcarrier phase is reset to C_Phase (subaddress 5A) on reset.	RW	0x3

Table 15−143. LAL_PHASE_RESET (Continued)

Bits	Field Name	Description	Type	Reset
16	SBLANK	Data output enable 0x0: No functionality 0x1: Enables the output of data when the SYNC_CTRL[11:10] VBLKM bits are set to 0b01	RW	0
15:9	RESERVED		RW	0x00
8:0	LAL	Last active line of field. These bits define the last active line of a field.	RW	0x107

Table 15−144. HS_INT_START_STOP_X

Address Offset	0x070		
Physical Address	0x4805 0C70	**Instance**	VENC1
Description	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		RW	0x00
25:16	HS_INT_STOP_X	HSYNC internal stop. These bits define the HSYNC internal stop pixel value.	RW	0x07E
15:10	RESERVED		RW	0x00
9:0	HS_INT_START_X	HSYNC internal start. These bits define the HSYNC internal start pixel value.	RW	0x34E

Table 15−145. HS_EXT_START_STOP_X

Address Offset	0x074		
Physical Address	0x4805 0C74	**Instance**	VENC1
Description	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		RW	0x00
25:16	HS_EXT_STOP_X	HSYNC external stop. These bits define the HSYNC external stop pixel value.	RW	0x00F
15:10	RESERVED		RW	0x00
9:0	HS_EXT_START_X	HSYNC external start. These bits define the HSYNC external start pixel value.	RW	0x359

Table 15−146. VS_INT_START_X

Address Offset	0x078		
Physical Address	0x4805 0C78	**Instance**	VENC1
Description	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		RW	0x00
25:16	VS_INT_START_X	VSYNC internal start. These bits define the VSYNC internal start pixel value.	RW	0x1A0
15:0	RESERVED		RW	0x0000

Table 15−147. VS_INT_STOP_X...VS_INT_START_Y

Address Offset	0x07C		
Physical Address	0x4805 0C7C	**Instance**	VENC1
Description	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		RW	0x00
25:16	VS_INT_START_Y	VSYNC internal start. These bits define the VSYNC internal start line value.	RW	0x209
15:10	RESERVED		RW	0x00
9:0	VS_INT_STOP_X	VSYNC internal stop. These bits define the VSYNC internal stop pixel value.	RW	0x1A0

Table 15−148. VS_INT_STOP_Y...VS_INT_START_X

Address Offset	0x080		
Physical Address	0x4805 0C80	**Instance**	VENC1
Description	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		RW	0x00
25:16	VS_EXT_START_X	VSYNC external start. These bits define the VSYNC external start pixel value.	RW	0x1AC

Table 15−148. VS_INT_STOP_Y...VS_INT_START_X (Continued)

Bits	Field Name	Description	Type	Reset
15:10	RESERVED		RW	0x00
9:0	VS_INT_STOP_Y	VSYNC internal stop. These bits define the VSYNC internal stop line value.	RW	0x022

Table 15−149. VS_EXT_STOP_X...VS_EXT_START_Y

Address Offset	0x084		
Physical Address	0x4805 0C84	**Instance**	VENC1
Description	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		RW	0x00
25:16	VS_EXT_START_Y	VSYNC external start. These bits define the VSYNC external start line value.	RW	0x20D
15:10	RESERVED		RW	0x00
9:0	VS_EXT_STOP_X	VSYNC external stop. These bits define the VSYNC external stop pixel value.	RW	0x1AC

Table 15−150. VS_EXT_STOP_Y

Address Offset	0x088		
Physical Address	0x4805 0C88	**Instance**	VENC1
Description	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		RW	0x000000
9:0	VS_EXT_STOP_Y	VSYNC external stop. These bits define the VSYNC external stop line value.	RW	0x006

Table 15−151. AVID_START_STOP_X

Address Offset	0x090		
Physical Address	0x4805 0C90	**Instance**	VENC1
Description	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		RW	0x00
25:16	AVID_STOP_X	AVID stop. These bits define the AVID stop pixel value.	RW	0x348
15:10	RESERVED		RW	0x00
9:0	AVID_START_X	AVID start. These bits define the AVID start pixel value.	RW	0x078

Table 15−152. AVID_START_STOP_Y

Address Offset	0x094		
Physical Address	0x4805 0C94	**Instance**	VENC1
Description	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		RW	0x00
25:16	AVID_STOP_Y	AVID stop. These bits define the AVID stop line value.	RW	0x206
15:10	RESERVED		RW	0x00
9:0	AVID_START_Y	AVID start. These bits define the AVID start line value.	RW	0x026

Table 15−153. FID_INT_START_X...FID_INT_START_Y

Address Offset	0x0A0		
Physical Address	0x4805 0CA0	**Instance**	VENC1
Description	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									

Table 15−153. FID_INT_START_X...FID_INT_START_Y (Continued)

Bits	Field Name	Description	Type	Reset
31:26	RESERVED		RW	0x00
25:16	FID_INT_START_Y	FID internal stop. These bits define the FID internal start line value.	RW	0x001
15:10	RESERVED		RW	0x00
9:0	FID_INT_START_X	FID internal start. These bits define the FID internal start pixel value.	RW	0x08A

Table 15−154. FID_INT_OFFSET_Y...FID_EXT_START_X

Address Offset	0x0A4		
Physical Address	0x4805 0CA4	**Instance**	VENC1
Description	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		RW	0x00
25:16	FID_EXT_START_X	FID external start. These bits define the FID external start pixel value.	RW	0x1AC
15:10	RESERVED		RW	0x00
9:0	FID_INT_OFFSET_Y	FID internal offset. These bits define the FID internal offset line value.	RW	0x106

Table 15−155. FID_EXT_START_Y...FID_EXT_OFFSET_Y

Address Offset	0x0A8		
Physical Address	0x4805 0CA8	**Instance**	VENC1
Description	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		RW	0x00
25:16	FID_EXT_OFFSET_Y	FID external offset. These bits define the FID external offset line value.	RW	0x106
15:10	RESERVED		RW	0x00
9:0	FID_EXT_START_Y	FID external start. These bits define the FID external start line value.	RW	0x006

Table 15−156. TVDETGP_INT_START_STOP_X

Address Offset	0x0B0		
Physical Address	0x4805 0CB0	**Instance**	VENC1
Description	TVDETGP_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						TVDETGP_INT_STOP_X										RESERVED						TVDETGP_INT_START_X									

Bits	Field Name	Description	Type	Reset
31:26	RESERVED		RW	0x00
25:16	TVDETGP_INT_STOP_X	TVDETGP internal stop. These bits define the TVDETGP internal stop pixel value.	RW	0x014
15:10	RESERVED		RW	0x00
9:0	TVDETGP_INT_START_X	TVDETGP internal start. These bits define the TVDETGP internal start pixel value.	RW	0x001

Table 15−157. TVDETGP_INT_START_STOP_Y

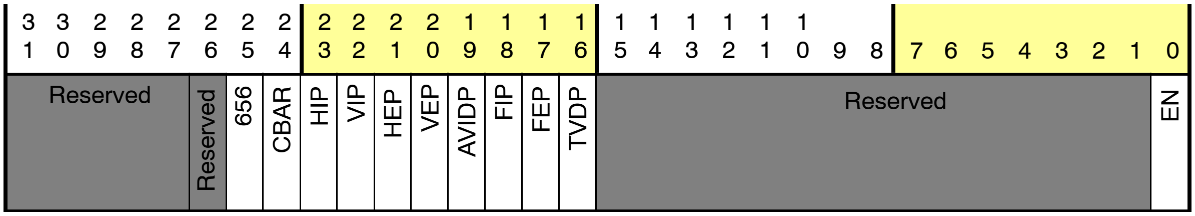
Address Offset	0x0B4		
Physical Address	0x4805 0CB4	**Instance**	VENC1
Description	TVDETGP_INT_START_STOP_Y		
Type	RW		

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						TVDETGP_INT_STOP_Y										RESERVED						TVDETGP_INT_START_Y									

Bits	Field Name	Description	Type	Reset
31:26	RESERVED		RW	0x00
25:16	TVDETGP_INT_STOP_Y	TVDETGP internal stop. These bits define the TVDETGP internal stop line value.	RW	0x001
15:10	RESERVED		RW	0x00
9:0	TVDETGP_INT_START_Y	TVDETGP internal start. These bits define the TVDETGP internal start line value.	RW	0x001

Table 15−158. GEN_CTRL

Address Offset	0x0B8		
Physical Address	0x4805 0CB8	**Instance**	VENC1
Description	TVDETGP enable and SYNC_POLARITY and UVPHASE_POL		
Type	RW		



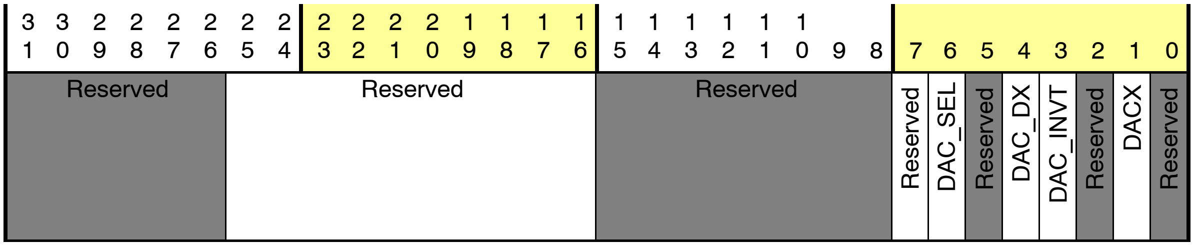
Bits	Field Name	Description	Type	Reset
31:27	Reserved	Reserved	RW	0x0000
26	Reserved	Reserved	RW	0
25	656	UVPHASE_POL 656 input mode UV phase 0x0: CbCr 0x1: CrCb	RW	0
24	CBAR	UVPHASE_POL CBAR mode UV phase 0x0: CbCr 0x1: CrCb	RW	0
23	HIP	HSYNC internal polarity 0x0: Active low 0x1: Active high	RW	1
22	VIP	VSYNC internal polarity 0x0: Active low 0x1: Active high	RW	1
21	HEP	HSYNC external polarity 0x0: Active low 0x1: Active high	RW	1
20	VEP	VSYNC external polarity 0x0: Active low 0x1: Active high	RW	1
19	AVIDP	AVID polarity 0x0: Active low 0x1: Active high	RW	1
18	FIP	FID internal polarity 0x0: Active low 0x1: Active high	RW	1

Table 15−158. GEN_CTRL (Continued)

Bits	Field Name	Description	Type	Reset
17	FEP	FID external polarity 0x0: Active low 0x1: Active high	RW	1
16	TVDP	TVDETGP polarity 0x0: Active low 0x1: Active high	RW	1
15:1	Reserved	Reserved	RW	0x00
0	EN	TVDETGP generation enable 0x0: Disabled 0x1: Enabled	RW	0

Table 15−159. DAC_TST

Address Offset	0x0C4
Physical Address	0x4805 0CC4 **Instance** VENC1
Description	DAC test controls and DAC A test 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
Reserved	Reserved	Reserved	Reserved	DAC_SEL	Reserved	DAC_DX	DAC_INVT	Reserved	DACX	Reserved

Bits	Field Name	Description	Type	Reset
31:26	Reserved	Reserved	RW	0x00
25:16	Reserved	Reserved	RW	0x000
15:8	Reserved	Reserved	RW	0x00
7	Reserved	Reserved	RW	0
6	DAC_SEL	DAC input selection (test mode) 0x0: DAC input is from internal register DAC [9:0]. 0x1: Video port G[1:0], B[7:0] is directly connected to a DAC for testing output.	RW	0
5	Reserved	Reserved	RW	0

Table 15−159. DAC_TST_DAC_A (Continued)

Bits	Field Name	Description	Type	Reset
4	DAC_DX	Direct DAC output 0x0: DACs are in normal operation. 0x1: DAC is in test mode. DACs are directly connected to either internal register or video port.	RW	0
3	DAC_INVT	DAC invert 0x0: DACs are in normal operation. 0x1: DACs inputs are inverted.	RW	1
2	Reserved	Reserved	RW	0
1	DACX	DAC_TST DACX 0x0: DAC output is disabled. 0x1: DAC output is enabled.	RW	0
0	Reserved	Reserved	RW	0

Table 15−160. DAC

Address Offset	0x0C8		
Physical Address	0x4805 0CC8	**Instance**	VENC1
Description	DAC test value		
Type	RW		

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="6">Reserved</td><td colspan="10">Reserved</td><td colspan="6">Reserved</td><td colspan="10">DAC</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
31:26	Reserved	Reserved	RW	0x00
25:16	Reserved	Reserved	RW	0x000
15:10	Reserved	Reserved	RW	0x00
9:0	DAC	In test mode, DAC input value	RW	0x000

Chapter 16

Timers

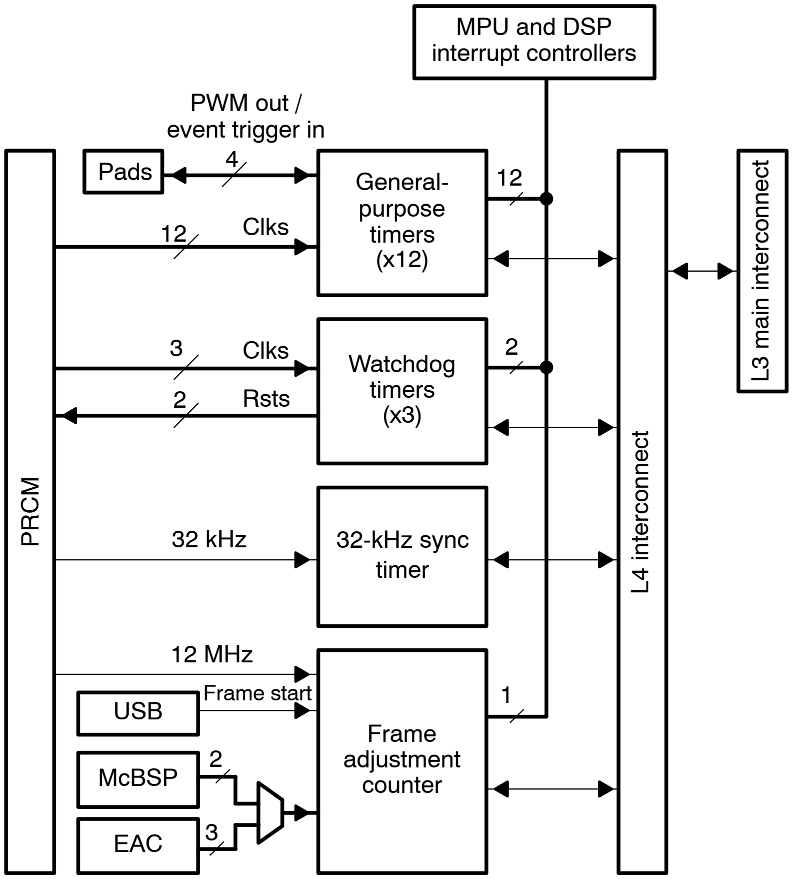
This chapter discusses several types of timers on the OMAP2420 device used by system software.

Topic **Page**

16.1 Timers Overview

The OMAP2420 device includes several different types of timers used by system software, including 12 general-purpose (GP) timers, 3 watchdog timers, a 32-kHz synchronized timer, and a specialized timer—the frame adjustment counter (FAC)—for USB/McBSP/EAC applications. Figure 16−1 highlights the counters in the OMAP2420 high-level block diagram.

Figure 16−1. OMAP2420 Timers



The three watchdog timers are divided into a single secure watchdog timer and two GP watchdog timers. The 32-kHz synchronized timer, which is only reset at power up, provides the operating system a stable timing source that stores relative time from the last power cycle of the product. The FAC provides a means for the system processor to calculate the phase difference between an audio source frame period and the universal serial bus (USB) interface 1-ms frame period, thus aligning the two frame periods with each other and eliminating a potential source of audio data dropout. Finally, twelve GP timers are included that are useful simply as basic timers, for generating time-stamp-based interrupts to the system software, or as a source of pulse width modulation (PWM) signals.

16.2 General−Purpose Timers

16.2.1 GP Timer Overview

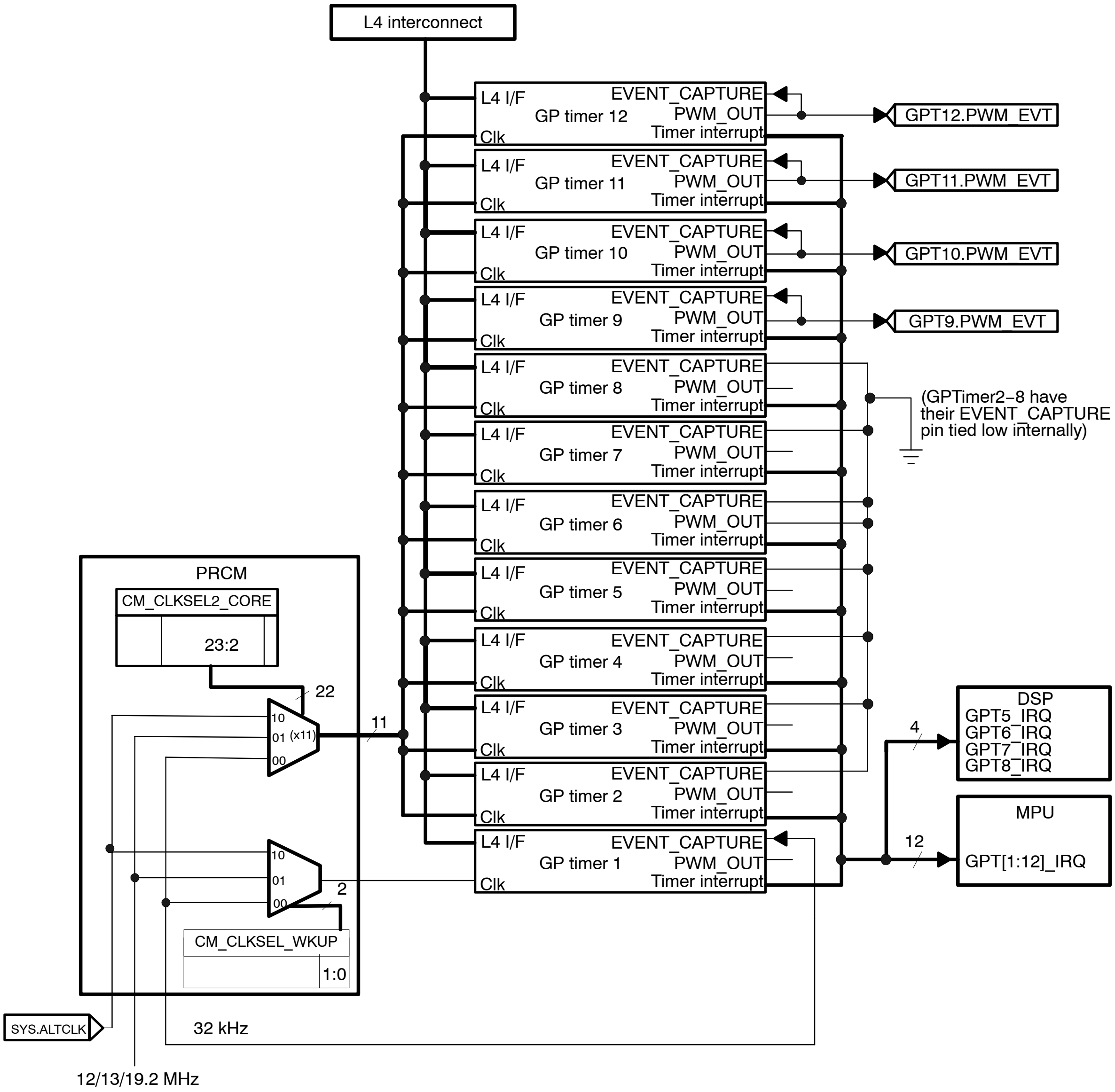
Twelve GP timers, GP timer 1 through GP timer 12, are available on the OMAP2420 device. Each timer can provide an interrupt to the microprocessor unit (MPU) subsystem. GP timers 5, 6, 7, and 8 are also connected to the interrupt register for the digital signal processor (DSP) subsystem. Each timer can be clocked from either the system clock (12/13/19.2 MHz), the 32-kHz clock, or from an alternate clock source using an input pad from the external system (SYS.ALTCLK). GP timers 9, 10, 11, and 12 can connect to external pins through their PWM output or their event capture input pin (for external timer triggering).

Note:

Some of these timers can be reserved by the operating system software; therefore, those specific timers might not be available for user application software.

Figure 16−2 shows an overview of the GP timers.

Figure 16−2. GP Timers Overview



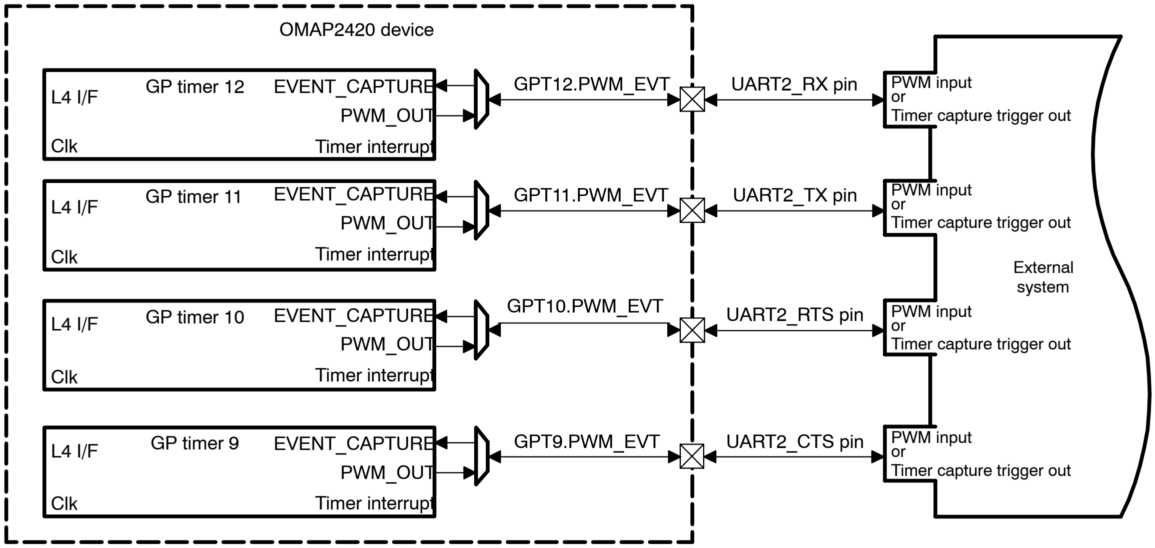
16.2.2 GP Timers Environment

16.2.2.1 GP Timers External System Interface

Four of the 12 GP timers can send or receive stimulus to or from the external (off-chip) system. GP timers 9, 10, 11, and 12 each can be configured to output a PWM pulse or to receive an external event signal that can be used as a trigger to capture the current timer count.

Figure 16−3 shows the external system interface of the GP timers.

Figure 16−3. GP Timers External System Interface



16.2.3 GP Timers Integration

16.2.3.1 Clocking, Reset, and Power-Management Scheme

Table 16−1 lists the power and reset domains and the source clocks for the GP timers present in the OMAP2420 device. For more information on power, reset, and clock control and domains, see Chapter 5, *Power, Reset, and Clock Management*.

Table 16−1. Clock, Power, and Reset Domains for GP Timers

Timer	Interface Clock	Functional Clock	Power Domain	Reset Domain
GP timer 1	WU_L4_ICLK	WU_GPT1_CLK	WKUP	WKUP_RST
GP timer 2 through GP timer 12	CORE_L4_ICLK	CORE_GPTx_ CLK (where $2 \leq x \leq 12$)	CORE	CORE_RST

Note: The 32-kHz clock drives the GP timer 1 EVENT_CAPTURE input to allow the OMAP2420 device to gauge the input frequency of the 12/13/19.2 MHz system clock.

16.2.3.2 GP Timer Interrupts

Table 16−2 lists interrupt mapping from the 12 GP timers to the 3 internal processors.

Table 16−2. Timer Interrupt Names and Processor IRQ Mapping

Timer	Interrupt Name	Mapping	Comments
GP timer 1	GPT1_IRQ	M_IRQ_37	GP timer 1 interrupt to MPU
GP timer 2	GPT2_IRQ	M_IRQ_38	GP timer 2 interrupt to MPU
GP timer 3	GPT3_IRQ	M_IRQ_39	GP timer 3 interrupt to MPU

Table 16−2. Timer Interrupt Names and Processor IRQ Mapping (Continued)

Timer	Interrupt Name	Mapping	Comments
GP timer 4	GPT4_IRQ	M_IRQ_40	GP timer 4 interrupt to MPU
GP timer 5	GPT5_IRQ	M_IRQ_41	GP timer 5 interrupt to MPU
		D_IRQ_10	GP timer 5 interrupt to DSP
		SINT22	GP timer 5 interrupt to DSP core INTC
GP timer 6	GPT6_IRQ	M_IRQ_42	GP timer 6 interrupt to MPU
		D_IRQ_11	GP timer 6 interrupt to DSP
		SINT23	GP timer 6 interrupt to DSP core INTC
GP timer 7	GPT7_IRQ	M_IRQ_43	GP timer 7 interrupt to MPU
		D_IRQ_12	GP timer 7 interrupt to DSP
GP timer 8	GPT8_IRQ	M_IRQ_44	GP timer 8 interrupt to MPU
		D_IRQ_13	GP timer 8 interrupt to DSP
GP timer 9	GPT9_IRQ	M_IRQ_45	GP timer 9 interrupt to MPU
GP timer 10	GPT10_IRQ	M_IRQ_46	GP timer 10 interrupt to MPU
GP timer 11	GPT11_IRQ	M_IRQ_47	GP timer 11 interrupt to MPU
GP timer 12	GPT12_IRQ	M_IRQ_48	GP timer 12 interrupt to MPU

16.2.3.3 GP Timer Pin List and Pad Multiplexing

Of all the OMAP2420 timers, only the FAC and four of the GP timers are capable of interfacing to the external system. Table 16−3 lists the pin locations and system control information for the GP timer I/Os.

Mux control of the configurable I/O pads on the device is accomplished by fields within specific OMAP2420 global registers found in the system control module (SCM) (see Chapter 8 and Chapter 24).

Table 16−3. GP Timer Pin Muxing

Function n	Description	DIR	Ball	Alternate Functions					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
GPT9.PWM/EVT	PWM or event for GP timer 9	IO	V19	uart2.cts	usb1.rcv		gpio.67		
			J15	i2c2.scl			gpio.99		
GPT10.PWM/EVT	PWM or event for GP timer 10	IO	R19	spi2. simo			gpio.89		
			W20	uart2.rts	usb1. txen		gpio.68		
GPT11.PWM/EVT	PWM or event for GP timer 11	IO	R20	spi2. somi			gpio.90		
			N14	uart2.tx	usb1.se0		gpio.69		
GPT12.PWM/EVT	PWM or event for GP timer 12	IO	M14	spi2. ncs0			gpio.91		
			P15	uart2.rx	usb1.dat		gpio.70		

16.2.3.4 L4 Interconnect Interface

The GP timer modules interface with the L4 interconnect through dedicated target agents (TAs) that are part of the L4 interconnect. Each TA provides status and configuration registers as listed in Table 16−4. For a complete description, see Chapter 6, *Internal Interconnect*.

Table 16−4. L4 Interconnect Target Agent Registers

Register Name	Type	Register Width (Bits)	Address Offset
COMPONENT	R	32	0x00
AGENT_CONTROL	R/W	32	0x20
AGENT_STATUS	R	32	0x28

Table 16−5 through Table 16−7 list the GP timer L4 component, agent control, and agent status physical addresses, respectively.

Table 16−5. GP Timer L4 Component Physical Addresses

Register Name	Type	Register Width (Bits)	Physical Address
GPTIMER1.COMPONENT	R	32	0x4802 9000
GPTIMER2.COMPONENT	R	32	0x4802 B000
GPTIMER3.COMPONENT	R	32	0x4807 9000
GPTIMER4.COMPONENT	R	32	0x4807 B000

Table 16−5. GP Timer L4 Component Physical Addresses (Continued)

Register Name	Type	Register Width (Bits)	Physical Address
GPTIMER5.COMPONENT	R	32	0x4807 D000
GPTIMER6.COMPONENT	R	32	0x4807 F000
GPTIMER7.COMPONENT	R	32	0x4808 1000
GPTIMER8.COMPONENT	R	32	0x4808 3000
GPTIMER9.COMPONENT	R	32	0x4808 5000
GPTIMER10.COMPONENT	R	32	0x4808 7000
GPTIMER11.COMPONENT	R	32	0x4808 9000
GPTIMER12.COMPONENT	R	32	0x4808 B000

Table 16−6. GP Timer L4 Agent Control Physical Addresses

Register Name	Type	Register Width (Bits)	Physical Address
GPTIMER1.AGENT_CONTROL	RW	32	0x4802 9020
GPTIMER2.AGENT_CONTROL	RW	32	0x4802 B020
GPTIMER3.AGENT_CONTROL	RW	32	0x4807 9020
GPTIMER4.AGENT_CONTROL	RW	32	0x4807 B020
GPTIMER5.AGENT_CONTROL	RW	32	0x4807 D020
GPTIMER6.AGENT_CONTROL	RW	32	0x4807 F020
GPTIMER7.AGENT_CONTROL	RW	32	0x4808 1020
GPTIMER8.AGENT_CONTROL	RW	32	0x4808 3020
GPTIMER9.AGENT_CONTROL	RW	32	0x4808 5020
GPTIMER10.AGENT_CONTROL	RW	32	0x4808 7020
GPTIMER11.AGENT_CONTROL	RW	32	0x4808 9020
GPTIMER12.AGENT_CONTROL	RW	32	0x4808 B020

Table 16−7. GP Timer L4 Agent Status Physical Addresses

Register Name	Type	Register Width (Bits)	Physical Address
GPTIMER1.AGENT_STATUS	R	32	0x4802 9028
GPTIMER2.AGENT_STATUS	R	32	0x4802 B028
GPTIMER3.AGENT_STATUS	R	32	0x4807 9028
GPTIMER4.AGENT_STATUS	R	32	0x4807 B028
GPTIMER5.AGENT_STATUS	R	32	0x4807 D028
GPTIMER6.AGENT_STATUS	R	32	0x4807 F028

Table 16−7. GP Timer L4 Agent Status Physical Addresses (Continued)

Register Name	Type	Register Width (Bits)	Physical Address
GPTIMER7.AGENT_STATUS	R	32	0x4808 1028
GPTIMER8.AGENT_STATUS	R	32	0x4808 3028
GPTIMER9.AGENT_STATUS	R	32	0x4808 5028
GPTIMER10.AGENT_STATUS	R	32	0x4808 7028
GPTIMER11.AGENT_STATUS	R	32	0x4808 9028
GPTIMER12.AGENT_STATUS	R	32	0x4808 B028

16.2.4 GP Timers Functional Description

16.2.4.1 GP Timers Operation

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 module includes compare logic to allow an interrupt event on 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 automatic timer counter capture and 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 TIER register for the interrupt features and a dedicated bit of TWER for the wake up.

For each GP timer implemented in the OMAP2420 device, there are three possible clock sources:

- The 32-kHz clock
- The system clock
- An external clock source

The input clock source is selected in the registers in the power, reset, and clock management (PRCM) module configuration (see Section 16.1, *Timers 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.

16.2.4.2 Timer Mode Functionality

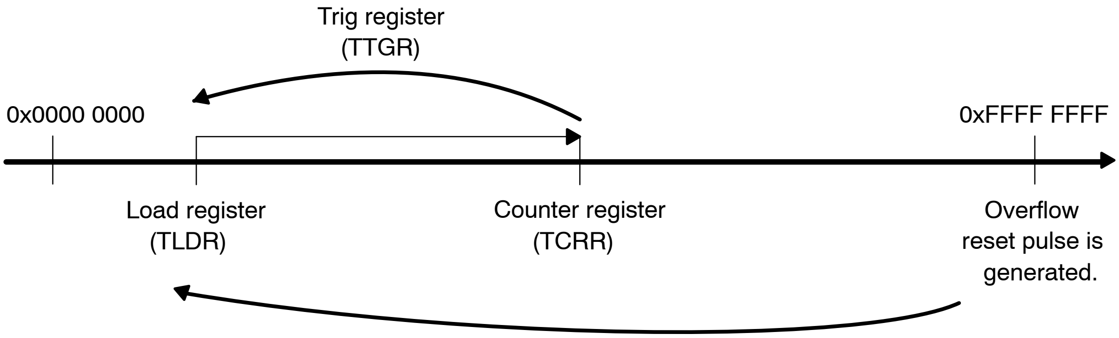
The timer is an upward counter that can be started and stopped at any time through the timer control register (GP timer register TCLR[0], ST bit). The timer counter register (TCRR) can be loaded either when it is stopped or on-the-fly (while counting). TCRR can be loaded directly by a TCRR write access with the new timer value. TCRR can also be loaded with the value held in the timer load register (TLDR) by a trig register (TTGR) write access. Loading the TCRR is done regardless of the TTGR written value. The value of TCRR can be read when stopped or captured on-the-fly by a TCRR read access. The timer is stopped and the counter value set to 0 when the module reset is asserted. The timer is maintained in stop after reset is released.

In 1-shot mode (GP timer register bit TCLR[1] = 0), the counter is stopped after counting overflow occurs (counter value remains at 0).

When autoreload mode is enabled (GP timer register bit TCLR[1] = 1), the TCRR is reloaded with the value of TLDR) after a counting overflow occurs. An interrupt can be issued on overflow if the overflow interrupt enable bit is set in the timer interrupt enable register (GP timer register bit TIER[1] =1). A dedicated output pin (timer PWM) can be programmed through GP timer register bits TCLR[12:10] (TRG and PT bits) to generate one positive pulse (prescaler duration) or to invert the current value (toggle mode) when an overflow occurs.

Figure 16−4 shows the TCRR timing value.

Figure 16−4. TCRR Timing Value



16.2.4.3 Capture Mode Functionality

The timer value in TCRR can be captured and saved in timer capture register (TCAR) when a transition is detected on the module input pin (event capture). The edge detection circuitry monitors transitions on the input pin (event capture).

Rising edge, falling edge, or both, can be selected in TCLR (TCM field) to trigger the timer counter capture. The module sets the TISR (TCAR_IT_FLAG bit) when an active edge is detected, and at the same time the counter value TCRR is stored in TCAR. If several consecutive capture events occur, the first one is saved in TCAR.

The edge detection logic is reset when a 1 is written to the TCAR_IT_FLAG bit, or when edge detection mode bits (TCM field in TCLR register) are changed from the no-capture mode detection to any other modes. The timer functional clock (input to prescaler) is used to sample the input pin (event capture). Input negative or positive pulse can be detected when 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 GP timer interrupt enable register bit TIER[12] (TCAR_IT_ENA bit).

If the TCLR CAPT_MODE field is 0, the value of the counter register is saved in the TCAR1 register on the first enabled capture event, and all following events are ignored (no update on TCAR1 and no interrupt triggering) until the detection logic is reset or the interrupt status register is cleared on TCAR's position writing a 1 to it.

If the TCLR CAPT_MODE field is 1, the counter value is saved in the TCAR1 register on the first enabled captured event and the value of the counter register is saved in the TCAR2 register on the second enabled capture event. If capture interrupt is enabled, an interrupt is generated on the second event captured. All other events are ignored (no update on TCAR1/2 and no interrupt triggering) until the detection logic is reset or the interrupt status register is cleared on TCAR's position by writing a 1 to it. This mechanism is useful for period calculation of a clock if that clock is connected to the timer EVT input pin.

16.2.4.4 Compare Mode Functionality

When the compare enable register bit TCLR[6] (CE bit) is set to 1, the value of the timer (TCRR) is continuously compared to the value held in the timer match register (TMAR). The value of TMAR can be loaded at any time (timer counting or stop). When the values of TCRR and the TMAR match, an interrupt is issued if register bit TIER[0] (MAT_IT_ENA bit) is set.

The dedicated output pin (timer PWM) can be programmed through TCLR (TRG and PT bits) to generate one positive pulse (timer clock duration) or to invert the current value (toggle mode) when an overflow or a match occurs.

16.2.4.5 Prescaler Functionality

A prescaler can be used to divide the timer counter input clock frequency. The prescaler is enabled when TCLR bit 5 is set (PRE). The PTV field in register TCLR sets the 2^n prescaler ratio. The prescaler counter is reset when the timer counter is stopped or reloaded on-the-fly.

Table 16−8 lists the prescaler/timer reload values versus context.

Table 16−8. Prescaler/Timer Reload Values Versus Contexts

Context	Prescaler	Timer Counter
Overflow (when autoreload on)	Reset	TLDR
TCRR write	Reset	TCRR
TTGR write	Reset	TLDR
Stop	Reset	Frozen

16.2.4.6 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. Either overflow alone or both overflow and match can be selected to toggle the timer PWM pin when a compare condition occurs. TCLR (the 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. The modulation is synchronously stopped when TRG is cleared and overflow has occurred. This allows fixing a deterministic state of the output pin when modulation is stopped.

In toggle mode, when TRG = 0x2 (overflow and match), the first event to make the PWM line toggle must be an overflow event. This means that if a match event occurs first, it does not toggle the PWM line.

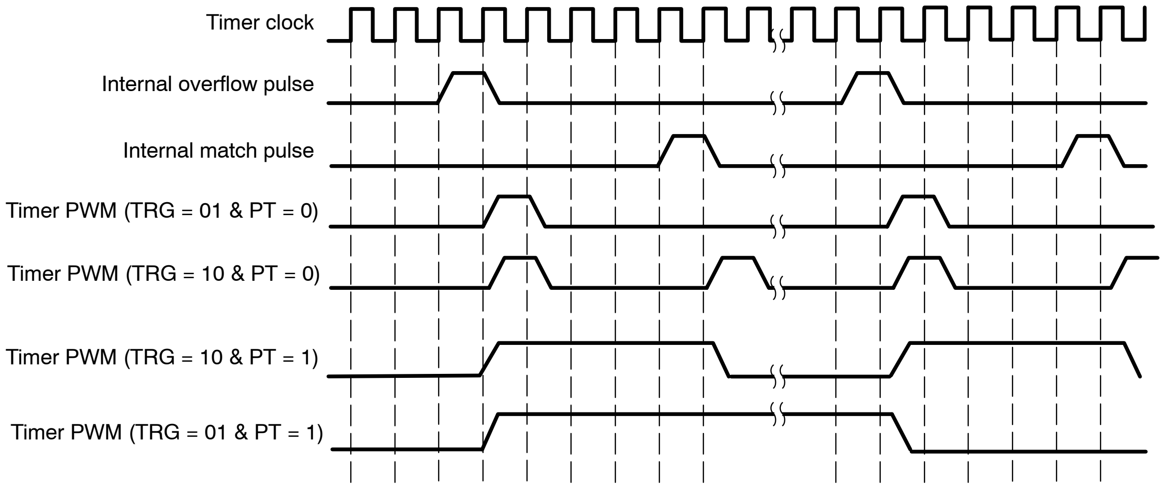
To obtain the desired waveform, users may start the counter at 0xFFFF FFFE (to ensure that an overflow occurs first) or play with the line polarity (SCPWM bit).

In Figure 16−5, the internal overflow pulse is set each time (0xFFFF FFFFF – TLDR +1) value is reached, and the internal match pulse is set when the counter reaches TMAR register value. According to TCLR (TRG and PT bits) value, the timer provides pulse or PWM on the output pin (timer PWM).

In Figure 16−5, register bit TCLR[7] (SCPWM bit) is set to 0.

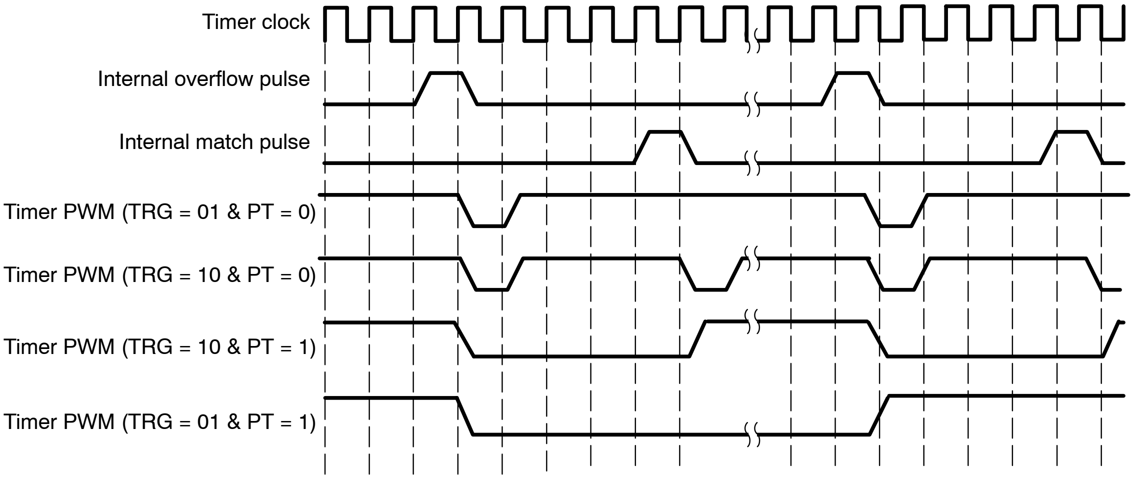
The TLDR and TMAR registers must keep their values below overflow (0xFFFFFFFF) by at least two units. If the PWM trigger events are both overflow and match, the values stored in the TMAR and TLDR registers must differ by at least two units. When a match event is used, the compare mode TCLR (CE) must be set.

Figure 16−5. Timing Diagram of PWM With SCPWM Bit = 0


Timer clock
Internal overflow pulse
Internal match pulse
Timer PWM (TRG = 01 & PT = 0)
Timer PWM (TRG = 10 & PT = 0)
Timer PWM (TRG = 10 & PT = 1)
Timer PWM (TRG = 01 & PT = 1)

In Figure 16−6, register bit TCLR[7] (SCPWM bit) is set to 1.

Figure 16−6. *Timing Diagram of PWM With SCPWM Bit = 1*


Timer clock
Internal overflow pulse
Internal match pulse
Timer PWM (TRG = 01 & PT = 0)
Timer PWM (TRG = 10 & PT = 0)
Timer PWM (TRG = 10 & PT = 1)
Timer PWM (TRG = 01 & PT = 1)

16.2.4.7 Timer Interrupt Control

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 TIER. When the interrupt event is issued, the associated interrupt status bit in the timer status register (TISR) is set. The pending interrupt event is reset when the set status bit is overwritten by 1.

16.2.4.8 Sleep Mode Request and Acknowledge

GP timer 1 through GP timer 12 support the PRCM smart-idle protocol. Each of these 12 timers responds to a PRCM sleep mode request depending on the IDLEMODE field of the system configuration register (see Table 16−25). For more detail regarding how the host processor and PRCM module issue sleep mode requests, see Chapter 5, *Power, Reset, and Clock Management.*

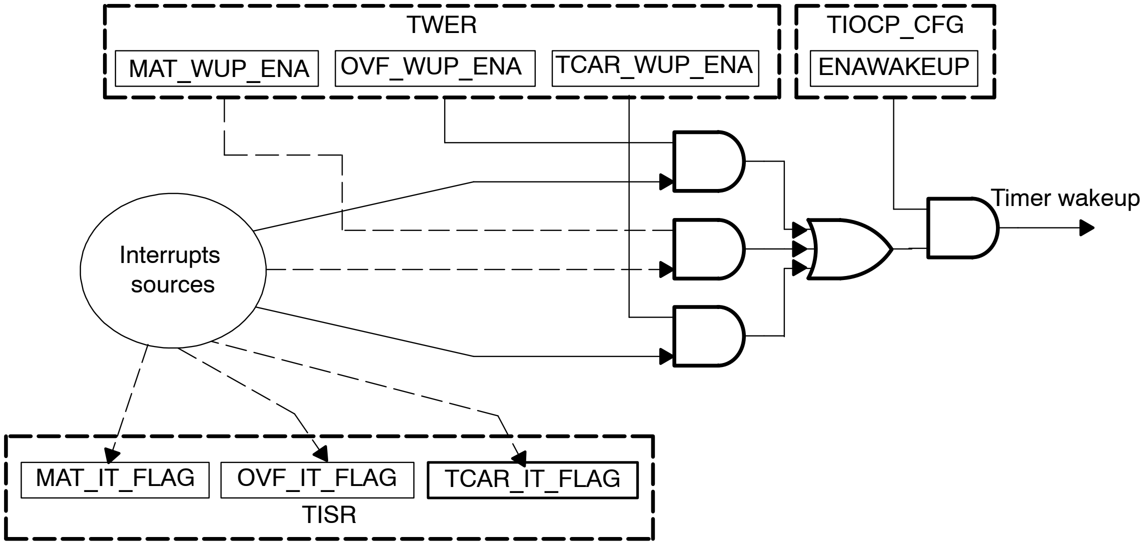
If the IDLEMODE field sets no-idle mode, the timer does not go into sleep mode and the idle acknowledge signal is never asserted.

If the IDLEMODE field sets force-idle mode, the timer goes into sleep mode independently of the internal module state and the idle acknowledge signal is unconditionally asserted.

If the IDLEMODE field sets smart-idle mode, the timer module evaluates its internal capability to have the interface clock switched off. Once 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 field ENAWAKEUP of TIOCP_CFG enables the timer wake-up capability.

Figure 16−7 shows the wake-up request generation. See Section 16.2.6, *GP timer Power Management,* for more information on GP timer clock control. if the WAKEUPENABLE bit is set to enable, the software must ensure all events in the GPTx_IRQSTATUS register are cleared before going to idle.

Figure 16−7. Wake-Up Request Generation



16.2.4.9 Wake-up Line Release

When the host processor receives a wake-up request issued by the timer peripheral, the interface clock is reactivated, the host processor deactivates the idle request signal, the timer deactivates the idle acknowledge signal, and the host can read the corresponding bit in TISR to determine which interrupt source has triggered the wake-up request. After acknowledging the wake-up request, the processor resets the status bit and releases the interrupt line by writing 1 in the corresponding bit of TISR.

16.2.4.10 Timer Counting Rate

The timer rate is defined by the following values:

- ❑ Value of the prescaler fields (PRE and PTV of the TCLR register)
- ❑ Value loaded into the timer load register (TLDR)

Table 16−9 lists the prescaler clock ratio values.

Table 16−9. Prescaler Clock Ratio Values

PRE	PTV	Divisor (PS)
0	X	1
1	0	2
1	1	4
1	2	8
1	3	16
1	4	32
1	5	64
1	6	128
1	7	256

Thus, the timer overflow-rate is expressed as:

$$\text{OVF\_Rate} = (\text{0xFFFF FFFF} - \text{GPTn.TLDR} + 1) \times (\text{timer-functional clock period}) \times \text{PS}$$

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**
>
> **Due to internal resynchronization, any write to the ST bit (GPTn.TCLR[1]) has some latency before the register is updated:**
>
> **2.5 × functional clock cycles < write_GPTn.TCLR_latency < 3.5 × functional clock cycles**
>
> **Do not forget to take this latency into account any time the timer has to be started or stopped by a software change of ST bit (GPTn.TCLR[1]).**

For example, with a timer clock input of 32 kHz and a PRE field equal to 0, the timer output period is as shown in Table 16−10.

Table 16−10. Value and Corresponding Interrupt Period

TLDR	Interrupt Period
0x0000 0000	37 h
0xFFFF 0000	2 s
0xFFFF FFF0	500 µs
0xFFFF FFFF	31.25 µs

16.2.5 Timer Under Emulation

During emulation mode, the timer continues to run according to the value of the EMUFREE bit of the timer L4 interface configuration register (TIOCP_CFG).

If EMUFREE is 1, timer execution is not stopped in emulation mode and the interrupt is still generated when overflow is reached.

If EMUFREE is 0, the prescaler and timer are frozen and both resume on exit from emulation mode. The async input pin is internally synchronized on two timer-clock rising edges.

16.2.6 GP Timer Power Management

To enable power saving at the system level, the PRCM can disable the GP timer clocks for GP timer 2 through GP timer 12.

Control of the GP timer interface and functional clocks is achieved by PRCM bits EN_GPTx (where $2 \leq x \leq 12$) in the CM_ICLKEN1_CORE and CM_FCLKEN1_CORE registers (bits CM_ICLKEN_CORE[14:4] and

CM_FCLKEN1_CORE[14:4]). Setting these bits to 0 disables the respective GP timer interface and/or functional clocks; setting these bits to 1 enables the clocks.

The PRCM bits AUTO_GPT2 through AUTO_GPT12 (CM_AUTO-IDLE1_CORE[14:4]) control GP timer interface clock behavior relative to the CORE power domain power status. Setting a bit to 1 causes the respective GP timer interface clock to disable automatically when the CORE power domain changes to a lower power state. This interface clock disabling process follows the PRCM smart-idle protocol.

For more information on power saving, see Chapter 5, *Power, Reset, and Clock Management*.

16.2.7 Accessing GP Timer Registers

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 LSB first, and the second write access must be the MSB. Write operations to the GP timer L4 registers (TIDR, TIOCP, CFG, TISR, TIER, TWER, and TSICR) can skip the MSB access if the 16 MSBs of the register does not require updating. Write operations to any functional register (TCLR, TCRR, TLDR, TTGR, and TMAR) must be complete (the MSB must be written even if the MSB data is not used).

16.2.7.1 Writing to Timer Registers

The host uses the L4 interface to write the TLDR, TCRR, TIER, TISR, TCLR, TIOCP_CFG, TWER, TTGR, TSICR, and TMAR registers synchronously with the timer interface clock.

In 16-bit access mode, the 16 LSBs must be written before writing to the 16 MSBs.

16.2.7.2 Write Posting Synchronization Mode

This mode is used if the TSICR[2] POSTED bit is set to 1.

This mode uses a posted−write scheme to update any internal register (TCLR, TCRR, TLDR, TTGR, and TMAR). This means that 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 (TWPS) register. In this mode, the software must check this status bit before 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 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

16.2.7.3 Nonwrite Posting Synchronization Mode

This mode is used if the TSICR[2] POSTED bit is set to 0.

This mode uses a nonposted write scheme to update any internal register. This means that the write transaction is not acknowledged on the L4 interface until the effective write operation occurs after the resynchronization in the timer 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. 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.

16.2.7.4 Reading From Timer Counter Registers

In 16-bit access mode, reading the 16 LSBs from the timer counter registers (TCRR, TCAR1, and TCAR2) captures the current timer counter value. This must be followed by reading the 16 MSBs.

DSP 16-bit accesses can be interleaved with MPU 32-bit accesses.

Note:

LSB/MSB accesses cannot be interleaved (that is, LSB register 1 → LSB register 2 → MSB register 1 → MSB register 2 sequence is not supported).

16.3 GP Timer Registers

16.3.1 GP Timer Register Map

16.3.1.1 Instance Summary

Table 16−11 lists the base address and block size for the GP timer module instances. All OMAP2420 timers are memory mapped to the L4 peripheral bus memory space.

Table 16−11. GP Timer Instance Summary

Module Name	Base Address	Size
GP timer 1	0x4802 8000	4K bytes
GP timer 2	0x4802 A000	4K bytes
GP timer 3	0x4807 8000	4K bytes
GP timer 4	0x4807 A000	4K bytes
GP timer 5	0x4807 C000	4K bytes
GP timer 6	0x4807 E000	4K bytes
GP timer 7	0x4808 0000	4K bytes
GP timer 8	0x4808 2000	4K bytes
GP timer 9	0x4808 4000	4K bytes
GP timer 10	0x4808 6000	4K bytes
GP timer 11	0x4808 8000	4K bytes
GP timer 12	0x4808 A000	4K bytes

16.3.2 GP Timer Register Summary

Table 16−12 through Table 16−23 list the register summary and associated addresses for the 12 GP timer internal registers. (Example: The physical address for the TCLR register of GP timer 8 is 0x4808 2024.)

Table 16−12. GPTIMER1 Register Summary

Register Name	Type	Register Width (Bits)	Physical Address
TIDR	R	32	0x4802 8000
TIOCP_CFG	RW	32	0x4802 8010
TISTAT	R	32	0x4802 8014
TISR	RW	32	0x4802 8018
TIER	RW	32	0x4802 801C
TWER	RW	32	0x4802 8020
TCLR	RW	32	0x4802 8024

Table 16−12. GPTIMER1 Register Summary (Continued)

Register Name	Type	Register Width (Bits)	Physical Address
TCRR	RW	32	0x4802 8028
TLDR	RW	32	0x4802 802C
TTGR	RW	32	0x4802 8030
TWPS	R	32	0x4802 8034
TMAR	RW	32	0x4802 8038
TCAR1	R	32	0x4802 803C
TSICR	RW	32	0x4802 8040
TCAR2	R	32	0x4802 8044

Table 16−13. GPTIMER2 Register Summary

Register Name	Type	Register Width (Bits)	Physical Address
TIDR	R	32	0x4802 A000
TIOCP_CFG	RW	32	0x4802 A010
TISTAT	R	32	0x4802 A014
TISR	RW	32	0x4802 A018
TIER	RW	32	0x4802 A01C
TWER	RW	32	0x4802 A020
TCLR	RW	32	0x4802 A024
TCRR	RW	32	0x4802 A028
TLDR	RW	32	0x4802 A02C
TTGR	RW	32	0x4802 A030
TWPS	R	32	0x4802 A034
TMAR	RW	32	0x4802 A038
TCAR1	R	32	0x4802 A03C
TSICR	RW	32	0x4802 A040
TCAR2	R	32	0x4802 A044

Table 16−14. GPTIMER3 Register Summary

Register Name	Type	Register Width (Bits)	Physical Address
TIDR	R	32	0x4807 8000
TIOCP_CFG	RW	32	0x4807 8010

Table 16−14. GPTIMER3 Register Summary (Continued)

Register Name	Type	Register Width (Bits)	Physical Address
TISTAT	R	32	0x4807 8014
TISR	RW	32	0x4807 8018
TIER	RW	32	0x4807 801C
TWER	RW	32	0x4807 8020
TCLR	RW	32	0x4807 8024
TCRR	RW	32	0x4807 8028
TLDR	RW	32	0x4807 802C
TTGR	RW	32	0x4807 8030
TWPS	R	32	0x4807 8034
TMAR	RW	32	0x4807 8038
TCAR1	R	32	0x4807 803C
TSICR	RW	32	0x4807 8040
TCAR2	R	32	0x4807 8044

Table 16−15. GPTIMER4 Register Summary

Register Name	Type	Register Width (Bits)	Physical Address
TIDR	R	32	0x4807 A000
TIOCP_CFG	RW	32	0x4807 A010
TISTAT	R	32	0x4807 A014
TISR	RW	32	0x4807 A018
TIER	RW	32	0x4807 A01C
TWER	RW	32	0x4807 A020
TCLR	RW	32	0x4807 A024
TCRR	RW	32	0x4807 A028
TLDR	RW	32	0x4807 A02C
TTGR	RW	32	0x4807 A030
TWPS	R	32	0x4807 A034
TMAR	RW	32	0x4807 A038
TCAR1	R	32	0x4807 A03C

Table 16−15. GPTIMER4 Register Summary (Continued)

Register Name	Type	Register Width (Bits)	Physical Address
TSICR	RW	32	0x4807 A040
TCAR2	R	32	0x4807 A044

Table 16−16. GPTIMER5 Register Summary

Register Name	Type	Register Width (Bits)	Physical Address
TIDR	R	32	0x4807 C000
TIOCP_CFG	RW	32	0x4807 C010
TISTAT	R	32	0x4807 C014
TISR	RW	32	0x4807 C018
TIER	RW	32	0x4807 C01C
TWER	RW	32	0x4807 C020
TCLR	RW	32	0x4807 C024
TCRR	RW	32	0x4807 C028
TLDR	RW	32	0x4807 C02C
TTGR	RW	32	0x4807 C030
TWPS	R	32	0x4807 C034
TMAR	RW	32	0x4807 C038
TCAR1	R	32	0x4807 C03C
TSICR	RW	32	0x4807 C040
TCAR2	R	32	0x4807 C044

Table 16−17. GPTIMER6 Register Summary

Register Name	Type	Register Width (Bits)	Physical Address
TIDR	R	32	0x4807 E000
TIOCP_CFG	RW	32	0x4807 E010
TISTAT	R	32	0x4807 E014
TISR	RW	32	0x4807 E018
TIER	RW	32	0x4807 E01C
TWER	RW	32	0x4807 E020
TCLR	RW	32	0x4807 E024
TCRR	RW	32	0x4807 E028

Table 16−17. GPTIMER6 Register Summary (Continued)

Register Name	Type	Register Width (Bits)	Physical Address
TLDR	RW	32	0x4807 E02C
TTGR	RW	32	0x4807 E030
TWPS	R	32	0x4807 E034
TMAR	RW	32	0x4807 E038
TCAR1	R	32	0x4807 E03C
TSICR	RW	32	0x4807 E040
TCAR2	R	32	0x4807 E044

Table 16−18. GPTIMER7 Register Summary

Register Name	Type	Register Width (Bits)	Physical Address
TIDR	R	32	0x4808 0000
TIOCP_CFG	RW	32	0x4808 0010
TISTAT	R	32	0x4808 0014
TISR	RW	32	0x4808 0018
TIER	RW	32	0x4808 001C
TWER	RW	32	0x4808 0020
TCLR	RW	32	0x4808 0024
TCRR	RW	32	0x4808 0028
TLDR	RW	32	0x4808 002C
TTGR	RW	32	0x4808 0030
TWPS	R	32	0x4808 0034
TMAR	RW	32	0x4808 0038
TCAR1	R	32	0x4808 003C
TSICR	RW	32	0x4808 0040
TCAR2	R	32	0x4808 0044

Table 16−19. GPTIMER8 Register Summary

Register Name	Type	Register Width (Bits)	Physical Address
TIDR	R	32	0x4808 2000
TIOCP_CFG	RW	32	0x4808 2010
TISTAT	R	32	0x4808 2014

Table 16−19. GPTIMER8 Register Summary (Continued)

Register Name	Type	Register Width (Bits)	Physical Address
TISR	RW	32	0x4808 2018
TIER	RW	32	0x4808 201C
TWER	RW	32	0x4808 2020
TCLR	RW	32	0x4808 2024
TCRR	RW	32	0x4808 2028
TLDR	RW	32	0x4808 202C
TTGR	RW	32	0x4808 2030
TWPS	R	32	0x4808 2034
TMAR	RW	32	0x4808 2038
TCAR1	R	32	0x4808 203C
TSICR	RW	32	0x4808 2040
TCAR2	R	32	0x4808 2044

Table 16−20. GPTIMER9 Register Summary

Register Name	Type	Register Width (Bits)	Physical Address
TIDR	R	32	0x4808 4000
TIOCP_CFG	RW	32	0x4808 4010
TISTAT	R	32	0x4808 4014
TISR	RW	32	0x4808 4018
TIER	RW	32	0x4808 401C
TWER	RW	32	0x4808 4020
TCLR	RW	32	0x4808 4024
TCRR	RW	32	0x4808 4028
TLDR	RW	32	0x4808 402C
TTGR	RW	32	0x4808 4030
TWPS	R	32	0x4808 4034
TMAR	RW	32	0x4808 4038
TCAR1	R	32	0x4808 403C
TSICR	RW	32	0x4808 4040
TCAR2	R	32	0x4808 4044

Table 16−21. GPTIMER10 Register Summary

Register Name	Type	Register Width (Bits)	Physical Address
TIDR	R	32	0x4808 6000
TIOCP_CFG	RW	32	0x4808 6010
TISTAT	R	32	0x4808 6014
TISR	RW	32	0x4808 6018
TIER	RW	32	0x4808 601C
TWER	RW	32	0x4808 6020
TCLR	RW	32	0x4808 6024
TCRR	RW	32	0x4808 6028
TLDR	RW	32	0x4808 602C
TTGR	RW	32	0x4808 6030
TWPS	R	32	0x4808 6034
TMAR	RW	32	0x4808 6038
TCAR1	R	32	0x4808 603C
TSICR	RW	32	0x4808 6040
TCAR2	R	32	0x4808 6044

Table 16−22. GPTIMER11 Register Summary

Register Name	Type	Register Width (Bits)	Physical Address
TIDR	R	32	0x4808 8000
TIOCP_CFG	RW	32	0x4808 8010
TISTAT	R	32	0x4808 8014
TISR	RW	32	0x4808 8018
TIER	RW	32	0x4808 801C
TWER	RW	32	0x4808 8020
TCLR	RW	32	0x4808 8024
TCRR	RW	32	0x4808 8028
TLDR	RW	32	0x4808 802C
TTGR	RW	32	0x4808 8030
TWPS	R	32	0x4808 8034
TMAR	RW	32	0x4808 8038
TCAR1	R	32	0x4808 803C

Table 16−22. GPTIMER11 Register Summary (Continued)

Register Name	Type	Register Width (Bits)	Physical Address
TSICR	RW	32	0x4808 8040
TCAR2	R	32	0x4808 8044

Table 16−23. GPTIMER12 Register Summary

Register Name	Type	Register Width (Bits)	Physical Address
TIDR	R	32	0x4808 A000
TIOCP_CFG	RW	32	0x4808 A010
TISTAT	R	32	0x4808 A014
TISR	RW	32	0x4808 A018
TIER	RW	32	0x4808 A01C
TWER	RW	32	0x4808 A020
TCLR	RW	32	0x4808 A024
TCRR	RW	32	0x4808 A028
TLDR	RW	32	0x4808 A02C
TTGR	RW	32	0x4808 A030
TWPS	R	32	0x4808 A034
TMAR	RW	32	0x4808 A038
TCAR1	R	32	0x4808 A03C
TSICR	RW	32	0x4808 A040
TCAR2	R	32	0x4808 A044

CAUTION

GP timer registers are limited to 32-bit and 16-bit data accesses only; 8-bit data accesses generate an error. For more details, see Section 6.2.6, *Error Management in the L3 Interconnect.*

16.3.3 GP Timer Register Descriptions

Table 16–24. GP Module Identification Register (TIDR)

Address Offset	0x000		
Physical Address	0x4802 8000	**Instance**	GPT1
	0x4802 A000		GPT2
	0x4807 8000		GPT3
	0x4807 A000		GPT4
	0x4807 C000		GPT5
	0x4807 E000		GPT6
	0x4808 0000		GPT7
	0x4808 2000		GPT8
	0x4808 4000		GPT9
	0x4808 6000		GPT10
	0x4808 8000		GPT11
	0x4808 A000		GPT12
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 0s.	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	

Table 16−25. GP Timer L4 Interface Configuration Register (TIOCP_CFG)

Address Offset	0x010		
Physical Address	0x4802 8010	**Instance**	GPT1
	0x4802 A010		GPT2
	0x4807 8010		GPT3
	0x4807 A010		GPT4
	0x4807 C010		GPT5
	0x4807 E010		GPT6
	0x4808 0010		GPT7
	0x4808 2010		GPT8
	0x4808 4010		GPT9
	0x4808 6010		GPT10
	0x4808 8010		GPT11
	0x4808 A010		GPT12
Description	This register controls the various parameters of the GP timer L4 interface.		
Type	RW		

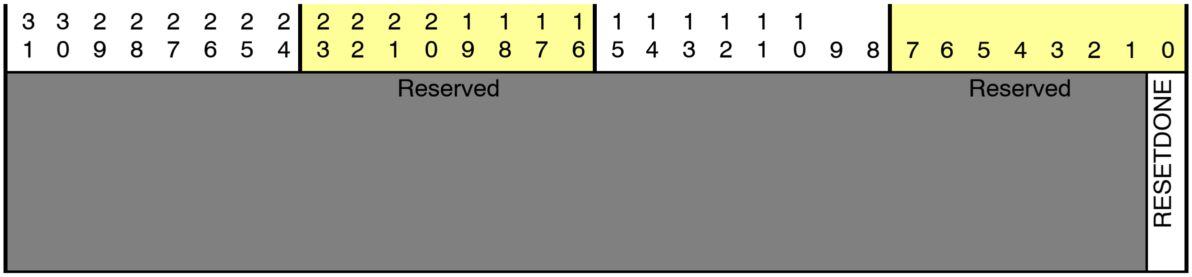
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																						CLOCKACTIVITY		RESERVED		EMUFREE	IDLEMODE		ENAWAKEUP	SOFTRESET	AUTOIDLE

Table 16−25. GP Timer L4 Interface Configuration Register (TIOCP_CFG) (Continued)

Bits	Field Name	Description	Type	Reset
31:10	Reserved	Write 0s for future compatibility. Reads return 0s.	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 can be switched off; functional clock is maintained during wake-up period. 0x2: Functional clock can be switched off; L4 interface 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 0s.	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-gaiting strategy 0x0: L4 interface clock is free-running. 0x1: Automatic L4 interface clock−gaiting strategy is applied, based on the L4 interface activity.	RW	0

Table 16–26. GP Timer System Status Register (TISTAT)

Address Offset	0x014		
Physical Address	0x4802 8014	**Instance**	GPT1
	0x4802 A014		GPT2
	0x4807 8014		GPT3
	0x4807 A014		GPT4
	0x4807 C014		GPT5
	0x4807 E014		GPT6
	0x4808 0014		GPT7
	0x4808 2014		GPT8
	0x4808 4014		GPT9
	0x4808 6014		GPT10
	0x4808 8014		GPT11
	0x4808 A014		GPT12
Description	This register provides status information about the module, excluding the interrupt status information.		
Type	R		



Bits	Field Name	Description	Type	Reset
31:8	Reserved	Reads return 0s.	R	0x000000
7:1	Reserved	Reads return 0s.	R	0x00
0	RESETDONE	Internal reset monitoring 0x0: Internal module reset is ongoing. 0x1: Reset completed	R	0

Table 16–27. GP Timer Interrupt Status Register (TISR)

Address Offset	0x018		
Physical Address	0x4802 8018	**Instance**	GPT1
	0x4802 A018		GPT2
	0x4807 8018		GPT3
	0x4807 A018		GPT4
	0x4807 C018		GPT5
	0x4807 E018		GPT6
	0x4808 0018		GPT7
	0x4808 2018		GPT8
	0x4808 4018		GPT9
	0x4808 6018		GPT10
	0x4808 8018		GPT11
	0x4808 A018		GPT12
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 0s.		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 16–28. GP Timer Interrupt Enable Register (TIER)

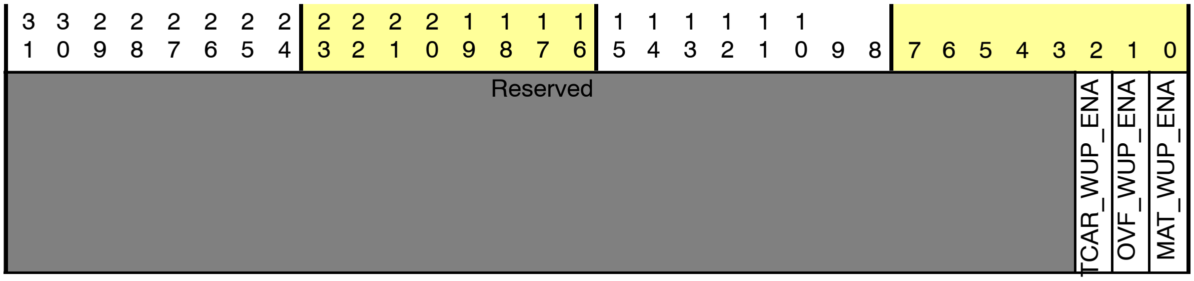
Address Offset	0x01C		
Physical Address	0x4802 801C	**Instance**	GPT1
	0x4802 A01C		GPT2
	0x4807 801C		GPT3
	0x4807 A01C		GPT4
	0x4807 C01C		GPT5
	0x4807 E01C		GPT6
	0x4808 001C		GPT7
	0x4808 201C		GPT8
	0x4808 401C		GPT9
	0x4808 601C		GPT10
	0x4808 801C		GPT11
	0x4808 A01C		GPT12
Description	This register 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 0s.	R	0x00000000
2	TCAR_IT_ENA	Enable capture interrupt 0x0: Disable capture interrupt. 0x1: Enable capture interrupt.	RW	0
1	OVF_IT_ENA	Enable overflow interrupt 0x0: Disable overflow interrupt. 0x1: Enable overflow interrupt.	RW	0
0	MAT_IT_ENA	Enable match interrupt 0x0: Disable match interrupt. 0x1: Enable match interrupt.	RW	0

Table 16–29. GP Timer Wake-up Enable Register (TWER)

Address Offset	0x020		
Physical Address	0x4802 8020	**Instance**	GPT1
	0x4802 A020		GPT2
	0x4807 8020		GPT3
	0x4807 A020		GPT4
	0x4807 C020		GPT5
	0x4807 E020		GPT6
	0x4808 0020		GPT7
	0x4808 2020		GPT8
	0x4808 4020		GPT9
	0x4808 6020		GPT10
	0x4808 8020		GPT11
	0x4808 A020		GPT12
Description	This register controls (enable/disable) the wake-up feature on specific interrupt events.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:3	Reserved	Reads return 0	R	0x00000000
2	TCAR_WUP_ENA	Enable capture wake-up 0x0: Disable capture wake-up. 0x1: Enable capture wakeup.	RW	0
1	OVF_WUP_ENA	Enable overflow wakeup 0x0: Disable overflow wakeup. 0x1: Enable overflow wakeup.	RW	0
0	MAT_WUP_ENA	Enable match wakeup 0x0: Disable match wakeup. 0x1: Enable match wakeup.	RW	0

Table 16–30. GP Timer Control Register (TCLR)

Address Offset	0x024		
Physical Address	0x4802 8024	**Instance**	GPT1
	0x4802 A024		GPT2
	0x4807 8024		GPT3
	0x4807 A024		GPT4
	0x4807 C024		GPT5
	0x4807 E024		GPT6
	0x4808 0024		GPT7
	0x4808 2024		GPT8
	0x4808 4024		GPT9
	0x4808 6024		GPT10
	0x4808 8024		GPT11
	0x4808 A024		GPT12
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 0s.	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

Table 16–30. GP Timer Control Register (TCLR) (Continued)

Bits	Field Name	Description	Type	Reset
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	PWM 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: 1-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 16–31. GP Timer Counter Register (TCRR)

Address Offset	0x028		
Physical Address	0x4802 8028	**Instance**	GPT1
	0x4802 A028		GPT2
	0x4807 8028		GPT3
	0x4807 A028		GPT4
	0x4807 C028		GPT5
	0x4807 E028		GPT6
	0x4808 0028		GPT7
	0x4808 2028		GPT8
	0x4808 4028		GPT9
	0x4808 6028		GPT10
	0x4808 8028		GPT11
	0x4808 A028		GPT12
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 16–32. GP Timer Load Register (TLDR)

Address Offset	0x02C		
Physical Address	0x4802 802C	**Instance**	GPT1
	0x4802 A02C		GPT2
	0x4807 802C		GPT3
	0x4807 A02C		GPT4
	0x4807 C02C		GPT5
	0x4807 E02C		GPT6
	0x4808 002C		GPT7
	0x4808 202C		GPT8
	0x4808 402C		GPT9
	0x4808 602C		GPT10
	0x4808 802C		GPT11
	0x4808 A02C		GPT12
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 16–33. GP Timer Trigger Register (TTGR)

Address Offset	0x030		
Physical Address	0x4802 8030	**Instance**	GPT1
	0x4802 A030		GPT2
	0x4807 8030		GPT3
	0x4807 A030		GPT4
	0x4807 C030		GPT5
	0x4807 E030		GPT6
	0x4808 0030		GPT7
	0x4808 2030		GPT8
	0x4808 4030		GPT9
	0x4808 6030		GPT10
	0x4808 8030		GPT11
	0x4808 A030		GPT12
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 16–34. Timer Write Posted Status Register (TWPS)

Address Offset	0x034		
Physical Address	0x4802 8034	**Instance**	GPT1
	0x4802 A034		GPT2
	0x4807 8034		GPT3
	0x4807 A034		GPT4
	0x4807 C034		GPT5
	0x4807 E034		GPT6
	0x4808 0034		GPT7
	0x4808 2034		GPT8
	0x4808 4034		GPT9
	0x4808 6034		GPT10
	0x4808 8034		GPT11
	0x4808 A034		GPT12
Description	This register triggers a counter reload of timer by writing any value in it.		
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_TMAR	W_PEND_TTGR	W_PEND_TLDR	W_PEND_TCRR	W_PEND_TCLR

Table 16–34. Timer Write Posted Status Register (TWPS) (Continued)

Bits	Field Name	Description	Type	Reset
31:5	Reserved	Reads return 0s.	R	0x0000000
4	W_PEND_TMAR	Write pending for register TMAR 0x0: Match register write not pending 0x1: Match register write pending	R	0
3	W_PEND_TTGR	Write pending for register TTGR 0x0: Trigger register write not pending 0x1: Trigger register write pending	R	0
2	W_PEND_TLDR	Write pending for register TLDR 0x0: Load register write not pending 0x1: Load register write pending	R	0
1	W_PEND_TCRR	Write pending for register TCRR 0x0: Counter register write not pending 0x1: Counter register write pending	R	0
0	W_PEND_TCLR	Write pending for register TCLR 0x0: Control register write not pending 0x1: Control register write pending	R	0

Table 16–35. GP Timer Match Value Register (TMAR)

Address Offset	0x038		
Physical Address	0x4802 8038	**Instance**	GPT1
	0x4802 A038		GPT2
	0x4807 8038		GPT3
	0x4807 A038		GPT4
	0x4807 C038		GPT5
	0x4807 E038		GPT6
	0x4808 0038		GPT7
	0x4808 2038		GPT8
	0x4808 4038		GPT9
	0x4808 6038		GPT10
	0x4808 8038		GPT11
	0x4808 A038		GPT12
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 16–36. GP Timer Counter 1 Capture Register (TCAR1)

Address Offset	0x03C		
Physical Address	0x4802 803C	**Instance**	GPT1
	0x4802 A03C		GPT2
	0x4807 803C		GPT3
	0x4807 A03C		GPT4
	0x4807 C03C		GPT5
	0x4807 E03C		GPT6
	0x4808 003C		GPT7
	0x4808 203C		GPT8
	0x4808 403C		GPT9
	0x4808 603C		GPT10
	0x4808 803C		GPT11
	0x4808 A03C		GPT12
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 16–37. GP Timer L4 Interface Synchronization Control Register TSICR

Address Offset	0x040		
Physical Address	0x4802 8040	**Instance**	GPT1
	0x4802 A040		GPT2
	0x4807 8040		GPT3
	0x4807 A040		GPT4
	0x4807 C040		GPT5
	0x4807 E040		GPT6
	0x4808 0040		GPT7
	0x4808 2040		GPT8
	0x4808 4040		GPT9
	0x4808 6040		GPT10
	0x4808 8040		GPT11
	0x4808 A040		GPT12
Description	This register contains the bits that control the interface between the L4 interface and functional clock domains—posted mode and functional software 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 0s.	R	0x00000000
2	POSTED	Posted mode selection 0x0: Nonposted mode selected 0x1: Posted mode selected	RW	1
1	SFT	Reset software functional registers. This bit is automatically reset by the hardware. During reads, it always returns 0. 0x0: Normal functional mode 0x1: The functional registers are reset.	RW	0
0	Reserved	Reads return 0.	R	0

Table 16–38. GP Timer Counter 2 Capture Register (TCAR2)

Address Offset	0x044		
Physical Address	0x4802 8044	**Instance**	GPT1
	0x4802 A044		GPT2
	0x4807 8044		GPT3
	0x4807 A044		GPT4
	0x4807 C044		GPT5
	0x4807 E044		GPT6
	0x4808 0044		GPT7
	0x4808 2044		GPT8
	0x4808 4044		GPT9
	0x4808 6044		GPT10
	0x4808 8044		GPT11
	0x4808 A044		GPT12
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

16.4 Watchdog Timers

16.4.1 Watchdog Timers Overview

The OMAP2420 device includes three instances of the 32-bit watchdog timer, which differ by the frequency of their respective timer clock inputs, their ability to be accessed in secure mode, and by their reset and interrupt output targets. Figure 16−8 shows how each timer is connected in the OMAP2420 device.

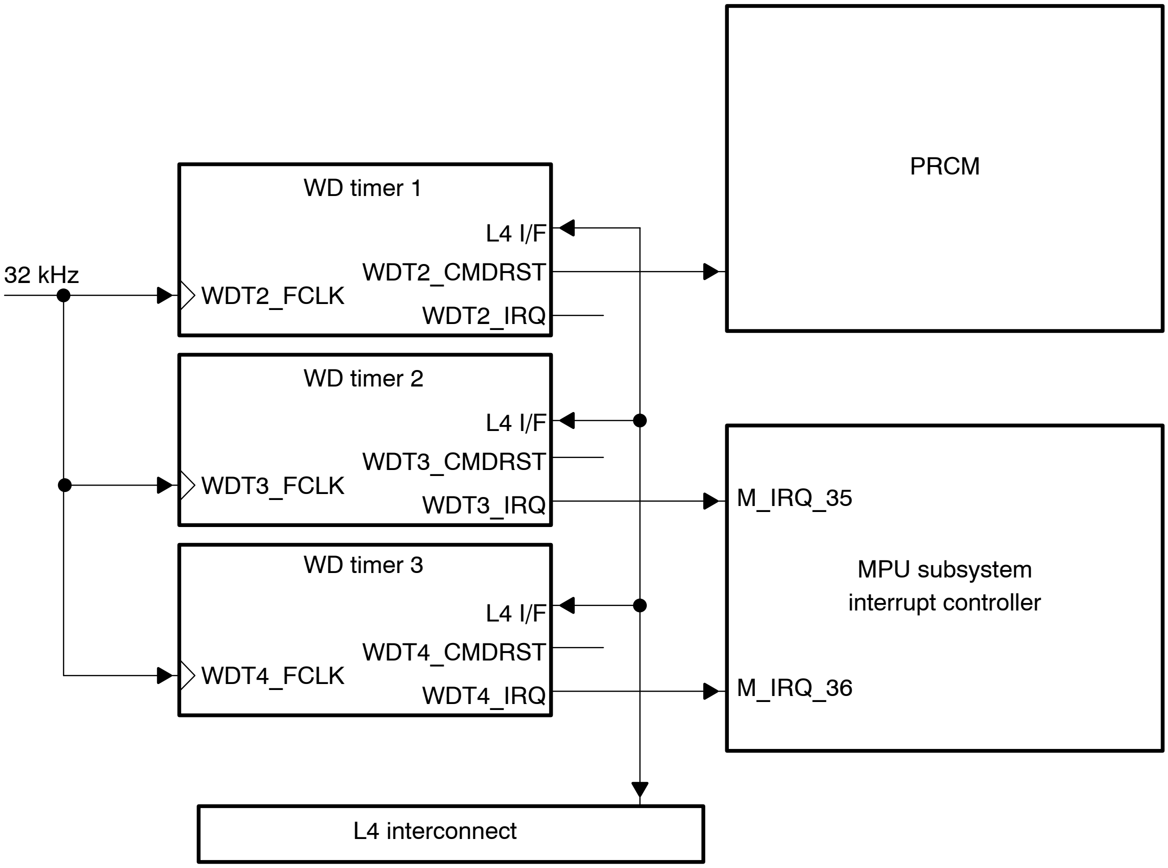
Each watchdog timer is an upward counter that can generate both a reset and an interrupt to the OMAP2420 system modules following an overflow condition. WD timer 1 and WD timer 2 serve resets to the PRCM module (their interrupt outputs are unused), and WD timer 3 serves watchdog interrupt to the MPU (its reset output is unused).

The timers can be accessed, loaded, and cleared by registers through the L4 interface. Two of the watchdog timers have the 32-kHz clock for their timer clock input, whereas the secure watchdog uses the system clock. Instances of the timer also differ by the default setting of the secure interface.

WD timer 1 and WD timer 2 generate a warm reset on overflow. WD timer 3 generates an MPU interrupt on overflow, which can be used by the application software through the PRCM to indirectly trigger a reset condition. These watchdog timers are enabled by default.

WD timer 2 and WD timer 3 connect to a single TA port on the L4 interface.

Figure 16−8. Watchdog Timers Block Diagram



16.4.2 Watchdog Timer Integration

16.4.2.1 Clocking, Reset, and Power-Management Scheme

Table 16−39 lists the power and reset domains, as well as the source clocks for each watchdog timer present in the OMAP2420 device. For more information on power, reset, and clock control and domains, see Chapter 5, *Power, Reset, and Clock Management*.

Table 16−39. Clock, Power, and Reset Domains for Watchdog Timers

Timer	Interface Clock	Functional Clock	Power Domain	Reset Domain
WD timer 1	WU_L4_ICLK	WU_32K_CLK	WKUP	WKUP_rst
WD timer 2	CORE_L4_CLK	CORE_32K_CLK	CORE	CORE_rst
WD timer 3	CORE_L4_CLK	CORE_32K_CLK	CORE	CORE_rst

16.4.2.2 Interrupts

The interrupt mapping from the three watchdog timers to the MPU is shown in Table 16−40.

Table 16−40. Watchdog Timer Interrupt Names and Processor IRQ Mapping

Timer	Interrupt Name	Mapping	Comments
WD timer 1	None		
WD timer 2	WDT2_IRQ	M_IRQ_35	WD timer 2 interrupt to MPU
WD timer 3	WDT3_IRQ	M_IRQ_36	WD timer 3 interrupt to MPU

16.4.2.3 L4 Interconnect Interface

The three watchdog internal interconnect timers interface with the L4 interconnect using dedicated TAs, which are part of the L4 interconnect. Each TA provides status and configuration registers as listed in Table 16−41. For a complete description, refer to Chapter 6, *Internal Interconnect*.

Table 16−41. L4 Interconnect Target Agent Registers

Register Name	Type	Register Width (Bits)	Address Offset
COMPONENT	R	32	0x00
AGENT_CONTROL	R/W	32	0x20
AGENT_STATUS	R	32	0x28

Table 16−42, Table 16−43, and Table 16−44 list the WD timer L4 component, agent control, and agent status physical addresses, respectively.

Table 16−42. Watchdog Timer L4 Component Physical Addresses

Register Name	Type	Register Width (Bits)	Physical Address
WDTIMER1.COMPONENT	R	32	0x4802 3000 (*)
WDTIMER2.COMPONENT	R	32	0x4802 5000
WDTIMER3.COMPONENT	R	32	0x4802 7000

* = WD timer 2 and WD timer 3 share a single TA interface to L4.

Table 16−43. Watchdog Timer L4 Agent Control Component Physical Addresses

Register Name	Type	Register Width (Bits)	Physical Address
WDTIMER1.AGENT_CONTROL	RW	32	0x4802 3020 (*)
WDTIMER2.AGENT_CONTROL	RW	32	0x4802 5020
WDTIMER3.AGENT_CONTROL	RW	32	0x4802 7020

* = WD timer 2 and WD timer 3 share a single TA interface to L4.

Table 16−44. Watchdog Timer L4 Agent Status Component Physical Addresses

Register Name	Type	Register Width (Bits)	Physical Address
WDTIMER1.AGENT_STATUS	RW	32	0x4802 3028 (*)
WDTIMER2.AGENT_STATUS	RW	32	0x4802 5028
WDTIMER3.AGENT_STATUS	RW	32	0x4802 7028

* = WD timer 2 and WD timer 3 share a single TA interface to L4.

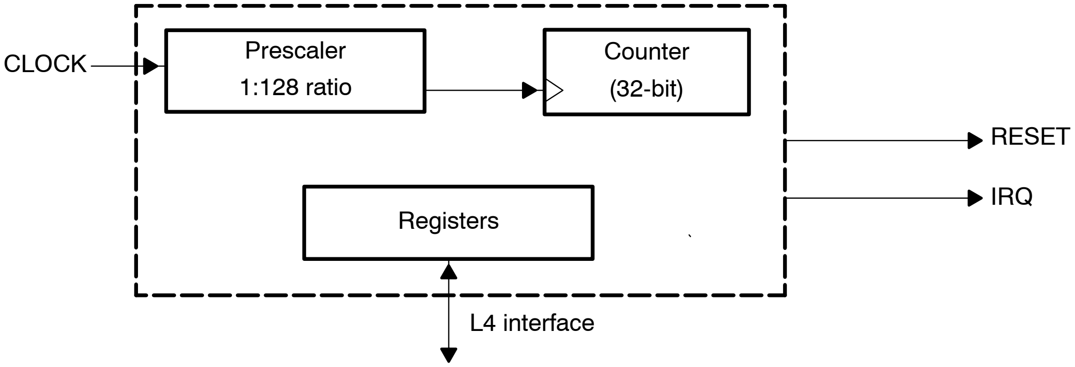
16.4.3 Watchdog Timers Functional Description

16.4.3.1 General Watchdog Timer Operation

The watchdog timers are based on an upward 32-bit counter coupled with a prescaler. The prescaler ratio can be set between 1 and 128 by accessing the PTV and PRE fields of the watchdog control register (WCLR). The current timer value can be accessed on-the-fly by reading the watchdog counter register (WCRR), modified by accessing the watchdog load register (WLDR) (no on-the-fly update), or reloaded by following a specific reload sequence on the watchdog trigger register (WTGR). A start/stop sequence applied to the watchdog start/stop register (WSPR) can start/stop the watchdog.

Figure 16−9 shows a block diagram of a 32-bit WD timer.

Figure 16−9. 32-Bit Watchdog Timer Functional Block Diagram



The watchdog timers are enabled after reset. The default reset values of the three WLDRs and prescaler ratios (PTV) are shown in Table 16−45. To get these values, the software must read the corresponding PTV fields of WCLR and the 32-bit register to retrieve the static configuration of the module.

Table 16−45. Count and Prescaler Default Reset Values

Timer	WLDR Reset Value	PTV Reset Value
WD timer 1	0xFFF00000	0
WD timer 2	0xFFF00000	0
WD timer 3	0xFFF00000	0

When the watchdog counter register (WCRR) overflows, an active-low reset pulse is generated to the PRCM. After reset generation, the counter is automatically reloaded with the value stored in WLDR and the prescaler is reset (the prescaler ratio remains unchanged). Then, after the reset pulse output has been generated, the timer counter begins to increment again.

16.4.3.2 Prescaler Setting

Each watchdog timer is composed of a prescaler stage and a timer counter.

The timer rate is defined by the following values:

- ☐ Value of the prescaler fields (PRE and PTV field of the WCLR register)
- ☐ Value loaded into the timer load register (WLDR)

Table 16−46 lists the prescaler clock ratio values.

Table 16−46. Prescaler Clock Ratios

PRE Bit (in WCLR Register)	PTV Bits (in 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 reset period can be determined by the following calculation:

The watchdog timer overflow-rate is expressed by:

OVF_Rate = (0xFFFF FFFF − WDTn.WLDR + 1) × (wd-functional clock period) × PS

Where wd-functional clock period = 1/(wd-functional clock frequency) and PS = 2(PTV).

> **CAUTION**
>
> **Due to internal resynchronization, any software write to GPTn.WSPR has some latency before WSPR is updated with the programmed value:**
>
> **1.5 × functional clock cycles < write_WSPR_latency < 2.5 × functional clock cycles**
>
> **Do not forget to take this latency into account any time the watchdog has to be started or stopped.**

Note for posted mode (WDT1_FREQRATIO=1):

If WDT1_FREQRATIO= 1 (CONTROL_DEVCONF1[19] = 1), any software action to WSPR, WTGR, WLDR, WCRR or WCLR keeps the corresponding write_pending flag asserted for the following number of clock cycles:

3.5 × functional + 3 interface < write_pending flag assertion < 4.5 × functional + 3 interface

This delay also must be taken into account when the watchdog must be started or stopped.

For example, for a 32-kHz timer clock input with a prescaler ratio value of 0x1 (clock divided by 2) and PRE field = 1 (clock divider enabled), the reset period is as shown in Table 16−47.

Table 16−47. Reset Period Examples

WLDR Value	Reset Period
0x0000 0000	74 h 56 min
0xFFFF 0000	4 s
0xFFFF FFF0	1 ms
0xFFFF FFFF	62.5 μs

Table 16−48 lists the default reset periods for the OMAP2420 watchdog timers.

Table 16−48. Default Watchdog Time Periods

Watchdog Timers	Clock Source	Default Reset Period
WD timer [1,2,3]	32 kHz	~33 s

16.4.3.3 Triggering a Timer Reload

To reload the timer counter and reset the prescaler before reaching overflow, a reload command can be executed by access to the watchdog trigger register (WTGR). This requires a specific reload sequence, which is performed whenever the written value on WTGR is different from its previous value. In this case, reload is executed in the same way as overflow autoreload but without a reset pulse generation. The timer counter is loaded with the watchdog load register (WLDR) value and the prescaler is reset.

16.4.3.4 Start/Stop Sequence for Watchdog Timers (Using WSPR Register)

To start/stop a watchdog timer, access must be made through the start/stop register (WSPR) using specific sequences.

To disable the timer, follow this sequence:

1) Write 0xXXXX AAAA in WSPR.

2) Write 0xXXXX 5555 in WSP.

To enable the timer, follow this sequence:

1) Write 0xXXXX BBBB in WSPR.

2) Write 0xXXXX 4444 in WSPR.

All other write sequences on WSPR have no effect on the start/stop feature of the module.

16.4.3.5 Modifying Timer Count/Load Values and Prescaler Setting

To modify the timer counter value (WCRR), prescaler ratio (PTV field of WCLR), or load value (WLDR), the watchdog must be disabled by using the start/stop sequence (WSPR).

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 (WTGR).

16.4.3.6 Watchdog Counter Register Access Restriction (WCRR Register)

Because the WCRR is directly related to timer counter value and is updated on the timer clock, a 32-bit shadow register is implemented to read a coherent value of the WCRR.

The shadow register is updated by a 32-bit read command or a 16-bit LSB read command. To be sure that a coherent value is read inside WCRR, use 32-bit reads, or in case of 2 × 16-bit reads, ensure the 16-bit LSB read is performed first (followed by the 16-bit MSB read).

16.5 Watchdog Timer Registers

All registers are 32 bits wide and accessible through the L4 interface with 16-bit or 32-bit access (read/write).

16.5.1 Instance Summary

Table 16−49 lists the base address and address space for the watchdog timer module instances. All OMAP2420 timers are memory mapped to the L4 peripheral bus memory space.

Table 16−49. Watchdog Timer Instance Summary

Module Name	Base Address	Size
WD timer 1	0x4802 2000	4K bytes
WD timer 2	0x4802 4000	4K bytes
WD timer 3	0x4802 6000	4K bytes

16.5.2 Watchdog Timer Register Summary

Table 16−50. WD Timer 1 Register Summary

Register Name	Type	Register Width (Bits)	Physical Address
WIDR	R	32	0x4802 2000
WD_SYSCONFIG	RW	32	0x4802 2010
WD_SYSSTATUS	R	32	0x4802 2014
WCLR	RW	32	0x4802 2024
WCRR	RW	32	0x4802 2028
WLDR	RW	32	0x4802 202C
WTGR	RW	32	0x4802 2030
WWPS	R	32	0x4802 2034
WSPR	RW	32	0x4802 2048

Table 16−51. WD Timer 2 Register Summary

Register Name	Type	Register Width (Bits)	Physical Address
WIDR	R	32	0x4802 4000
WD_SYSCONFIG	RW	32	0x4802 4010
WD_SYSSTATUS	R	32	0x4802 4014
WISR	RW	32	0x4802 4018
WIER	RW	32	0x4802 401C
WCLR	RW	32	0x4802 4024

Register Name	Type	Register Width (Bits)	Physical Address
WCRR	RW	32	0x4802 4028
WLDR	RW	32	0x4802 402C
WTGR	RW	32	0x4802 4030
WWPS	R	32	0x4802 4034
WSPR	RW	32	0x4802 4048

Table 16−52. WD Timer 3 Register Summary

Register Name	Type	Register Width (Bits)	Physical Address
WIDR	R	32	0x4802 6000
WD_SYSCONFIG	RW	32	0x4802 6010
WD_SYSSTATUS	R	32	0x4802 6014
WISR	RW	32	0x4802 6018
WIER	RW	32	0x4802 601C
WCLR	RW	32	0x4802 6024
WCRR	RW	32	0x4802 6028
WLDR	RW	32	0x4802 602C
WTGR	RW	32	0x4802 6030
WWPS	R	32	0x4802 6034
WSPR	RW	32	0x4802 6048

CAUTION

Watchdog timer registers are limited to 32-bit and 16-bit data accesses only; 8-bit data accesses generate an error. For more detail, see Section 6.2.6, *Error Management in the L3 Interconnect*.

16.5.3 Watchdog Timer Register Descriptions

Table 16–53. Watchdog Hardware Revision (ID) Register (WIDR)

Address Offset	0x000		
Physical Address	0x4802 2000 0x4802 4000 0x4802 6000	**Instance**	WDT1 WDT2 WDT3
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 0s.	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	

Table 16–54. Watchdog System Configuration Register (WD_SYSCONFIG)

Address Offset	0x010		
Physical Address	0x4802 2010 0x4802 4010 0x4802 6010	**Instance**	WDT1 WDT2 WDT3
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																										EMUFREE	RESERVED			SOFTRESET	AUTOIDLE

Bits	Field Name	Description	Type	Reset
31:6	Reserved	Write 0s for future compatibility. Reads return 0s.	R	0x0000000
5	EMUFREE	Emulation mode 0x0: Timer counter frozen in emulation 0x1: Timer counter free-running in emulation	RW	0
4:2	Reserved	Write 0s for future compatibility. Reads return 0s.	R	0x0
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 16–55. Watchdog System Status Register (WD_SYSSTATUS)

Address Offset	0x014		
Physical Address	0x4802 2014 0x4802 4014 0x4802 6014	**Instance**	WDT1 WDT2 WDT3
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 0s.	R	0x000000
7:1	Reserved	Reads return 0s.	R	0x00
0	RESETDONE	Internal reset monitoring 0x0: Internal module reset in ongoing. 0x1: Reset completed [†]	R	0

[†] During reset the value is 0, but the value read just after the reset is 1 because ICLK/FCLK is already operating.

Table 16–56. Watchdog Interrupt Event Pending (WISR)

Address Offset	0x018		
Physical Address	0x4802 4018 0x4802 6018	**Instance**	WDT2 WDT3
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 0s.	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 16–57. Watchdog Overflow Interrupt Control Register (WIER)

Address Offset	0x01C		
Physical Address	0x4802 401C 0x4802 601C	**Instance**	WDT2 WDT3
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 0s.	R	0x00000000
0	OVF_IT_ENA	Enable overflow interrupt 0x0: Disable overflow interrupt. 0x1: Enable overflow interrupt.	RW	0

Table 16–58. Watchdog Control Register (WCLR)

Address Offset	0x024		
Physical Address	0x4802 2024 0x4802 4024 0x4802 6024	**Instance**	WDT1 WDT2 WDT3
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 0s.	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: 2^{PTV} Example: PTV = 2: counter increases value is started after 4 functional clock periods.	RW	0x0
1:0	Reserved	Reads return 0s.	R	0x0

Table 16–59. Watchdog Counter Register (WCRR)

Address Offset	0x028		
Physical Address	0x4802 2028 0x4802 4028 0x4802 6028	**Instance**	WDT1 WDT2 WDT3
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 16−60. Watchdog Load Register (WLDR)

Address Offset	0x02C		
Physical Address	0x4802 202C 0x4802 402C 0x4802 602C	**Instance**	WDT1 WDT2 WDT3
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	0xFFF00000

Table 16−61. Watchdog Trigger Register (WTGR)

Address Offset	0x030		
Physical Address	0x4802 2030 0x4802 4030 0x4802 6030	**Instance**	WDT1 WDT2 WDT3
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 16−62. Watchdog Writer-Posted Status Register (WWPS)

Address Offset	0x034		
Physical Address	0x4802 2034 0x4802 4034 0x4802 6034	**Instance**	WDT1 WDT2 WDT3
Description	This register contains the write posting bits for all writable 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 0s.	R	0x0000000
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 16–63. Watchdog Start/Stop Register (WSPR)

Address Offset	0x048		
Physical Address	0x4802 2048 0x4802 4048 0x4802 6048	**Instance**	WDT1 WDT2 WDT3
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

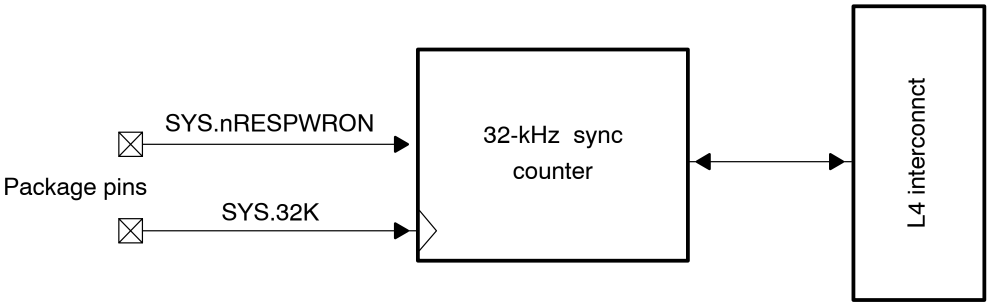
16.6 32-kHz Sync Timer

16.6.1 32-kHz Sync Timer Functional Description

The 32-kHz synchronized timer is a 32-bit counter, clocked by the falling edge of the 32-kHz system clock. It is reset with external async power-up reset (SYS.nRESPWRON). When SYS.nRESPWRON is released, 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 to 0 and starts counting again.

Figure 16−10 shows a block diagram of the sync timer.

Figure 16−10. 32-kHz Sync Timer Block Diagram



16.6.1.1 Reading the 32-kHz Sync Timer

The sync counter register is 32 bits wide; for correct count capture, it must be accessed with a 16-bit LSB access first, and with a 16-bit MSB access last. Internal synchronization logic allows reading the counter value while the counter is running. The time latency to read the counter is one L4 interconnect clock period.

16.6.2 32-kHz Sync Timer Environment

The sync timer is accessible through the L4 interface only.

16.6.3 32-kHz Sync Timer Integration

16.6.3.1 Clocking, Reset, and Power-Management Scheme

Table 16−64 lists the power and reset domain and the source clock for the 32-kHz sync timer present in the OMAP2420 device. For more information on power, reset, and clock control and domains, see Chapter 5, *Power, Reset, and Clock Management*.

Table 16−64. Clock, Power, and Reset Domains for 32-kHz Sync Timer

Timer	Source Clock	Power Domain	Reset Domain
32-kHz sync timer	SYS.32K	WKUP	Power-on

16.6.3.2 Interrupts

The 32-kHz sync timer has no interrupt outputs.

16.6.3.3 L4 Interconnect Interface

The 32-kHz sync timer interfaces with the L4 interconnect through dedicated TAs, which are part of the L4 interconnect. Each TA provides status and configuration registers as listed in Table 16−65. For a complete description, see Chapter 6, *Internal Interconnect*.

Table 16−65. L4 Interconnect Target Agent Registers

Register Name	Type	Register Width (Bits)	Offset Address
COMPONENT	R	32	0x4800 5000
AGENT_CONTROL	R/W	32	0x4800 5020
AGENT_STATUS	R	32	0x4800 5028

16.7 32-kHz Sync Timer Registers

16.7.1 32-kHz Sync Timer Register Map

16.7.1.1 Instance Summary

Table 16−66 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 16−66. 32-kHz Sync Timer Instance Summary

Module Name	Base Address	Size
32K timer	0x4800 4000	4K bytes

16.7.1.2 32-kHz Sync Timer Register Summary

CAUTION

32-kHz sync timer registers are limited to 32-bit and 16-bit data accesses only; 8-bit data accesses generate an error. For more detail, see Section 6.2.6, *Error Management in the L3 Interconnect*.

Table 16−67. Sync Timer Register Summary

Register Name	Type	Register Width (Bits)	Address Offset
32KSYNCNT_REV	R	32	0x00
CR	R	32	0x10

16.7.1.3 32-kHz Sync Timer Register Descriptions

Table 16−68. 32-kHz Sync Counter Identification Register (32KSYNCNT_REV)

Address Offset	0x0000
Physical Address	0x4800 4000
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 returns 0s.	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	

Table 16–69. Read Counter Register (CR)

Address Offset	0x0010
Physical Address	0x4800 4010
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																															

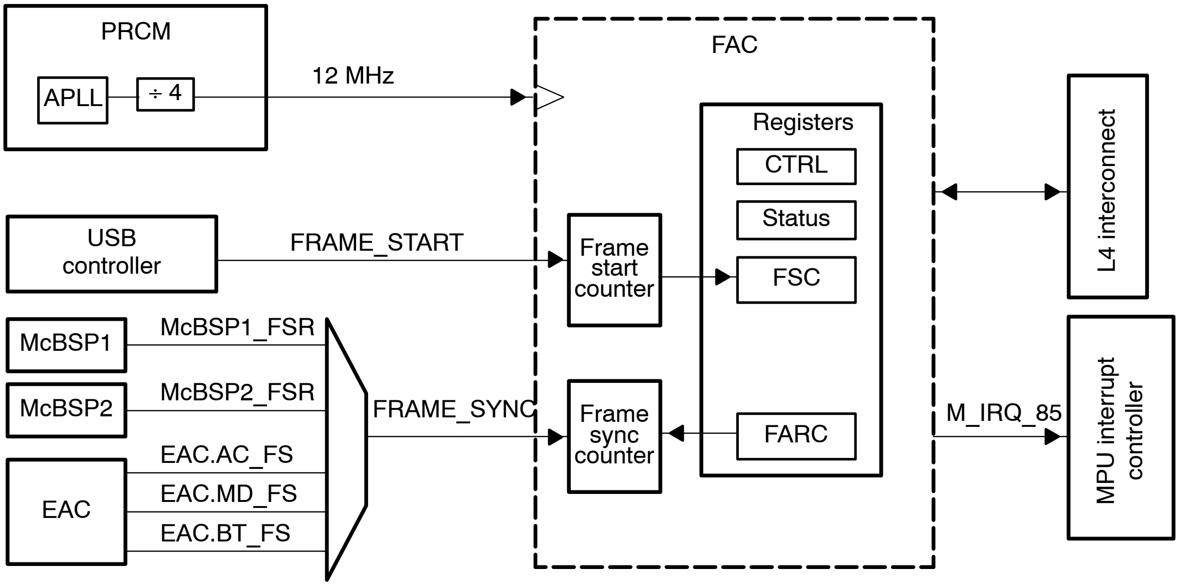
Bits	Field Name	Description	Type	Reset
31:0	COUNTER_VALUE	Counter register value	R	0x00000003

16.8 Frame Adjustment Counter

16.8.1 FAC Overview

The FAC counts the number of rising edges of one signal, FRAME_START (start of frame interrupt of the USB function), during a programmable number of rising edges of a second signal, FRAME_SYNC (receive frame synchronization input from McBSP1 or McBSP2). System-level software running on the MPU can use the ratio of these two count values to determine the frequency relationship of the two signals and then can act to modify the behavior of either the source or sink (or possibly both) frame period timing. Figure 16−11, A block diagram of the FAC, shows how the FAC interfaces to other on-chip modules.

Figure 16−11.AC Block Diagram

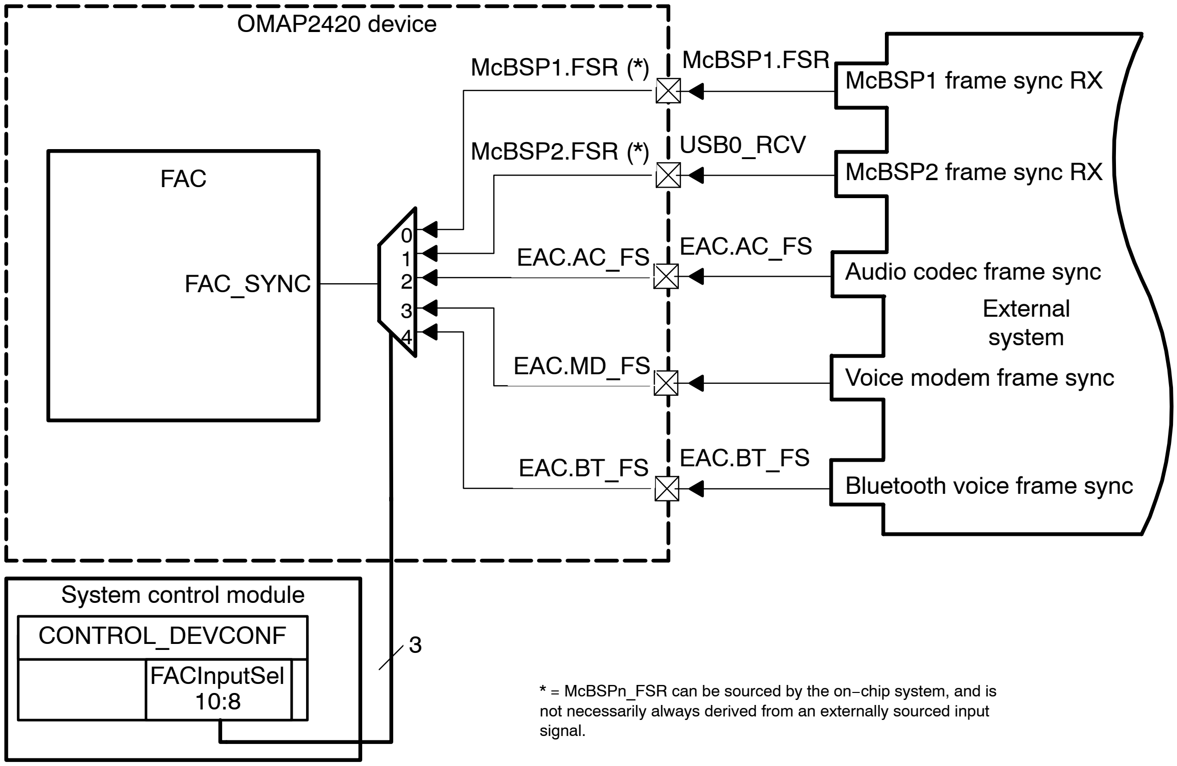


16.8.2 FAC Environment

16.8.2.1 FAC External System Interface

Figure 16−12 shows how the FAC interfaces to the external system and the various frame sync sources from which you can select.

Figure 16−12. FAC External System Interface



* = McBSPn_FSR can be sourced by the on−chip system, and is not necessarily always derived from an externally sourced input signal.

16.8.3 FAC Integration

16.8.3.1 Clocking, Reset, and Power-Management Scheme

Table 16−70 lists the power and reset domain and the source clocks for the FAC counter in the OMAP2420 device. For more information on power, reset, and clock control domains see Chapter 5, *Power, Reset, and Clock Management*.

The following clock signal names are used in Table 16−70:

Func_12M_Clk = Fixed 12-MHz clock from PRCM APLL

Table 16−70. Clock, Power, and Reset Domains for FAC

Timer	Source Clock	Power Domain	Reset Domain
FAC	Func_12M_Clk	CORE	CORE_rst

16.8.3.2 Interrupts

Table 16−71 lists the FAC Interrupt names and MPU IRQ mapping.

Table 16−71. FAC Interrupt Names and Processor IRQ Mapping

Timer	Interrupt Name	Mapping	Comments
FAC	FAC_IRQ	M_IRQ_85	FAC interrupt to MPU subsystem

16.8.3.3 FAC Pin List and Pad Multiplexing

Of all the OMAP2420 timers, only the FAC and four of the GP timers can interface with the external system. Table 16−72 lists pin locations and system control information for the possible FAC frame sync I/Os.

Mux control of the configurable I/O pads on the device is accomplished by fields within specific OMAP2420 global registers found in the system control module (see Chapter 24 for more information related to pad mux control).

Table 16−72. FAC Pin Muxing

Module Pin Name	OMAP2420		Pin Description	System Control Register	Alternate Function				
	Pin Name	Ball No.			Mode 0	Mode 1	Mode 2	Mode 3	Mode 4
McBSP1.FSR	McBSP1_FSR	P20		CONTROL_PADCONF_MCBSP1_FSR	McBSP1_FSR			GPIO_93	spi2_ncs1
	CAM_D4	W2		CONTROL_PADCONF_MCBSP1_FSR	CAM_D4	HW_DBG6	McBSP1_FSR	GPIO_50	■
McBSP2.FSR	USB0_RCV	J18		CONTROL_PADCONF_MCBSP1_FSRMCBSP1_FSR	USB0_RCV	McBSP2_FSX	■	GPIO_109	UART2_CTS
	EAC_AC_FS	R14		CONTROL_PADCONF_MCBSP1_FSRMCBSP1_FSR	EAC_AC_FS	McBSP2_FSX	■	GPIO_114	
	DSS_ACBIAS	W7		CONTROL_PADCONF_MCBSP1_FSRMCBSP1_FSR	DSS_ACBIAS	■	McBSP2_FSX	GPIO_48	
EAC.AC_FS	EAC.AC_FS	R14		CONTROL_PADCONF_EAC_AC_FSR	EAC_AC_FS	McBSP2_FSX	■	GPIO_114	■
EAC.MD_FS		Y13		CONTROL_PADCONF_SSi1_WAKE		EAC_MD_FS	■	GPIO_66	■
EAC.BT_FS	EAC.BT_FS	P9		CONTROL_PADCONF_EAC_BT_FSR	EAC_BT_FS	■	■	GPIO_72	■

16.8.3.4 L4 Interconnect Interface

The FAC module interfaces with the L4 interconnect using dedicated TAs, which are part of the L4 interconnect. Each TA provides status and configuration registers as listed in Table 16−73. For a complete description, refer to Chapter 6, *Internal Interconnect*.

Table 16−73. L4 Interconnect Target Agent Registers

Register Name	Type	Register Width (Bits)	Offset Address
COMPONENT	R	32	0x4809 3000
AGENT_CONTROL	R/W	32	0x4809 3020
AGENT_STATUS	R	32	0x4809 3028

16.8.4 FAC Functional Description

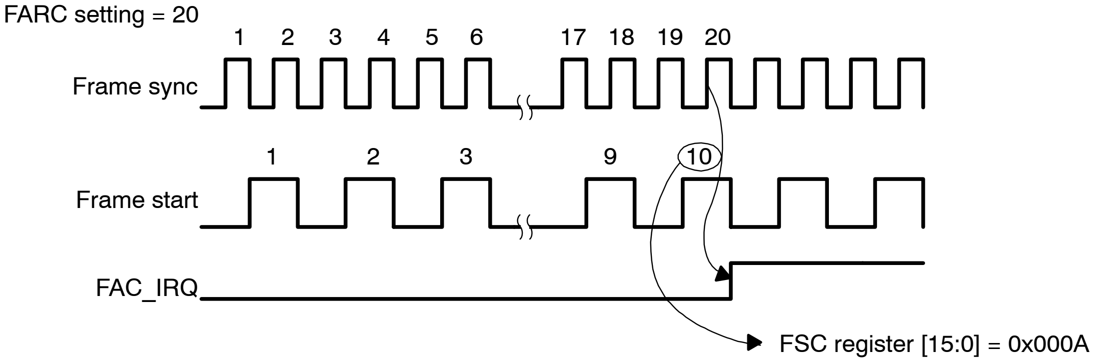
The frame-adjustment reference count (FARC) register is programmed with the number of frame sync pulses over which the frame start pulses are to be counted. The frame start count (FSC) register is updated with the number of frame start rising edges that occur during the programmable FARC period. A control and configuration (CTRL) register allows for the module to be put into either continuous or halt mode:

- In continuous mode, the FSC register is periodically updated with a new value each time the FARC register value is met, and a new count is automatically initiated; in halt mode, the FSC register is updated with a new value when the FARC register value is met, counting halts, and an interrupt is generated to the MPU.
- In halt mode, a new count is initiated when the software services the interrupt by reading the FSC register.

The RUN bit in the control and configuration register enables and disables the counters. If the RUN bit is set to 0, the counters are reset immediately. The software can use this bit as a software reset. Additionally, a status register (STATUS) containing an FSC_FULL bit indicates to the system software whether FSC has been read subsequent to the last FSC update.

Figure 16−13 shows an example of the behavior of the FAC count store registers with the frame adjustment reference count register programmed to be 20.

Figure 16−13. Frame Start Count Example

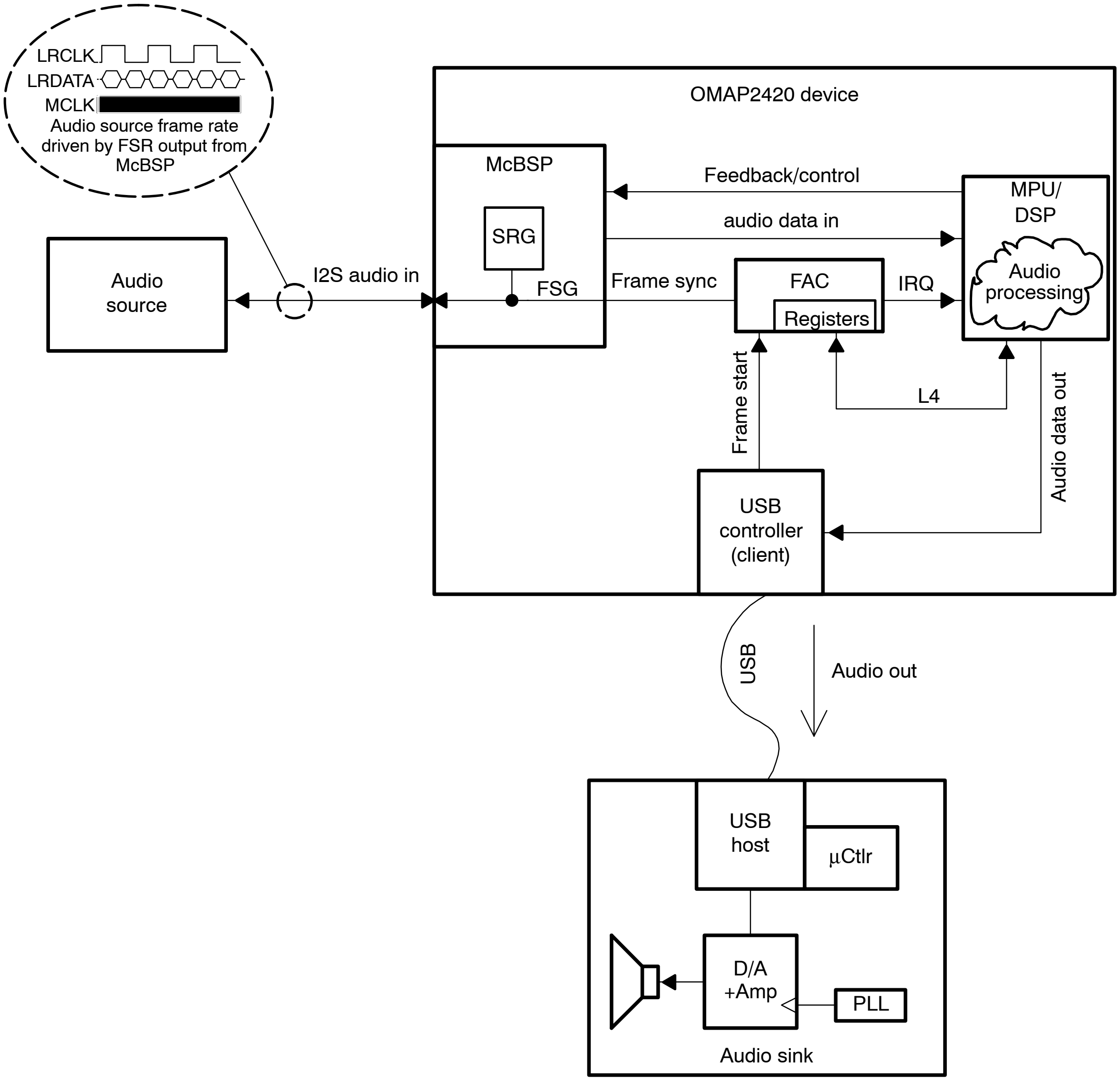


16.8.4.1 Application Example

The FAC can be used in audio applications to synchronize receiver and transmitter audio frame rates, and thus decrease or remove the possibility of audio data dropouts (loss of data). These dropouts can cause or create serious audio quality problems.

Figure 16−14 shows a system with audio data being input to the OMAP2420 device using an McBSP port, processed by the OMAP system, and output through the USB port. In this example, the audio sink device is the USB host, and thus controls the USB audio frame rate. The incoming audio data must have its frame rate synchronized to the USB host (audio sink) rate to avoid audio buffer overflow (in the case where the incoming audio rate is higher than the audio sink frame rate) or underflow (where the sink frame rate is higher than the source frame rate). By using the FAC, in conjunction with a McBSP sample rate generator (SRG in Figure 16−14), the OMAP2420 MPU can calculate the difference in the McBSP generated frame rate (the FSG signal) and the outgoing audio frame rate, and then modify the McBSP SRG rate to compensate. By periodically checking the frame rate difference and using this as feedback to the SRG control, the MPU can effectively synchronize the audio source to the audio sink frame rate.

Figure 16–14. Example FAC Audio Application



16.9 FAC Registers

16.9.1 FAC Register Map

16.9.2 Instance Summary

Table 16−74 lists the base address and block size for the FAC. It is memory mapped to the L4 peripheral bus memory space.

Table 16−74. FAC Instance Summary

Module Name	Base Address	Size
FAC	0x4809 2000	4K bytes

Table 16−75 lists the FAC register summary.

16.9.2.1 FAC Register Summary

Table 16−75. FAC Register Summary

Register Name	Type	Register Width (Bits)	Physical Address
FARC	RW	16	0x4809 2000
FSC	R	16	0x4809 2004
CTRL	RW	16	0x4809 2008
STATUS	R	16	0x4809 200C
SYNC_CNT	R	16	0x4809 2010
START_CNT	R	16	0x4809 2014
SYSCONFIG	RW	16	0x4809 2018
REVISION	R	16	0x4809 201C
SYSSTATUS	R	16	0x4809 2020

CAUTION

The FAC registers are limited to 16-bit data accesses only; 32-bit and 8-bit data accesses are not allowed and can corrupt register content.

16.9.2.2 FAC Register Descriptions

Table 16−76. Frame Adjustment Reference Count Register (FARC)

Address Offset	0x00
Physical Address	0x4809 2000
Description	This register contains the frame adjustment reference value.
Type	RW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FARC															

Bits	**Field Name**	**Description**	**Type**	**Reset**
15:0	FARC	Frame adjustment reference count value	RW	0x0000

Table 16−77. Frame Start Counter Register (FSC)

Address Offset	0x04
Physical Address	0x4809 2004
Description	This register contains the frame start counted value.
Type	R

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FSC															

Bits	**Field Name**	**Description**	**Type**	**Reset**
15:0	FSC	Frame start counted value	R	0x0000

Table 16−78. FAC Control Register (CTRL)

Address Offset	0x08
Physical Address	0x4809 2008
Description	This register controls the functional features.
Type	RW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													INT_EN	RUN	CNT

Bits	Field Name	Description	Type	Reset
15:3	Reserved	Read returns 0s.	R	0x00000000
2	INT_EN	Interrupt enable bit 0x0: Interrupt is disabled. 0x1: Interrupt is enabled.	RW	0
1	RUN	Start/stop control bit 0x0: Module is stopped. 0x1: Module is started.	RW	0
0	CNT	Functional mode control bit, continuous/halt mode select 0x0: Module is in halt mode. 0x1: Module is in continuous mode.	RW	0

Table 16−79. FAC Status Register (STATUS)

Address Offset	0x0C
Physical Address	0x4809 200C
Description	This register provides the interrupt status information.
Type	R

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															FSC_FULL

Bits	Field Name	Description	Type	Reset
15:3	Reserved	Read returns 0s.	R	0x00000000
0	FSC_FULL	Interrupt status bit 0x0: The FSC register has been read or RUN bit has been cleared. 0x1: FSC register is updated.	R	0

Table 16−80. FAC Frame Sync Count Value SYNC_CNT

Address Offset	0x10
Physical Address	0x4809 2010
Description	This register contains the current frame sync counter value.
Type	R

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SYNC_CNT															

Bits	Field Name	Description	Type	Reset
15:0	SYNC_CNT	Frame sync counter value	R	0x0000

Table 16−81. FAC Frame Start Counter Register (START_CNT)

Address Offset	0x14
Physical Address	0x4809 2014
Description	This register contains the current frame start counter value.
Type	R

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
START_CNT															

Bits	Field Name	Description	Type	Reset
15:0	START_CNT	Frame start counter value	R	0x0000

Table 16−82. FAC Configuration Register (SYSCONFIG)

Address Offset	0x18
Physical Address	0x4809 2018
Description	This register controls the various parameters of the FAC L4 interface.
Type	RW

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
15:5	Reserved	Write 0s for future compatibility Read returns 0s.	RW	0x0000000
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	Reserved	Write 0 for future compatibility. Read returns 0.	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.	R	0

Table 16–83. FAC Module Revision Register (REVISION)

Address Offset	0x1C
Physical Address	0x4809 201C
Description	This register contains the IP revision code.
Type	R

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								REV							

Bits	**Field Name**	**Description**	**Type**	**Reset**
15:8	Reserved	Reads returns 0s.	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	

Table 16–84. FAC Module Status Register (SYSSTATUS)

Address Offset	0x20
Physical Address	0x4809 2020
Description	This register provides status information about the module, excluding the interrupt status information.
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 0s.	R	0x000000
7:1	Reserved	Read returns 0s.	R	0x00
0	RESETDONE	Internal reset monitoring 0x0: Internal module reset is ongoing. 0x1: Reset completed[†]	R	0

[†] During reset the value is 0, but the value read just after the reset is 1 because ICLK/FCLK is already operating.

Chapter 17

UART/IrDA/CIR

This chapter discusses the three universal asycnchronous receiver/transmitter (UART) devices on the OMAP2420 multimedia device.

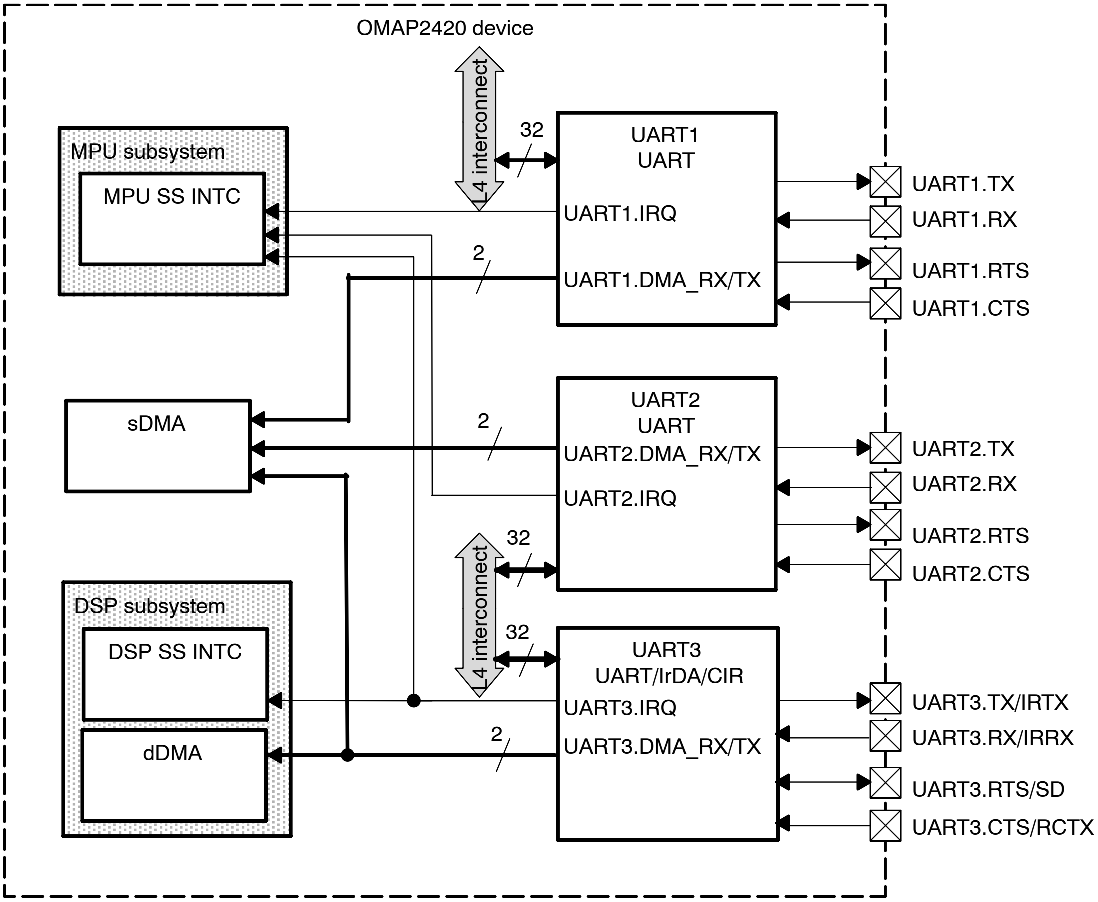
Topic **Page**

17.1 UART/IrDA/CIR Overview

The OMAP2420 device contains three universal asynchronous receiver/transceiver (UART) devices controllable by the microprocessor unit (MPU):

- ❑ Two UART modules (UART1 and UART2) are pinned-out to be usable as UART only. They must be programmed appropriately by setting the MODESELECT field in the mode definition register 1 (MDR1) to one of three UART operating modes.
- ❑ UART3 adds infrared communication support (UART/IrDA). UART3 is pinned-out to be usable as UART, IrDA, or CIR, and can be programmed to any available operating mode.

Figure 17−1. UART Module



17.1.1 UART Features

The UARTs (UART1, UART2, and UART3 when in UART mode) feature the following:

- ❑ 16C750 compatibility
- ❑ 64-byte FIFO for receiver and 64-byte FIFO for transmitter
- ❑ Programmable interrupt trigger levels for FIFOs

- ❑ Baud rate generation based on programmable divisors N (N = 1…16384) operating from a fixed functional clock of 48 MHz

  Oversampling is programmed by software as 16 or 13; therefore, the baud rate computation is either of the following:

  - 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: Software and hardware (RTS/CTS)

The UART clocks are connected in such a way to produce a baud rate up to 3.6M bits/s. Table 17−1 lists the supported baud rates, the requested divisor, and the corresponding error versus the standard baud rate.

Table 17−1. 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	104	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

17.1.2 IrDA Features

The key features of the IrDA (UART3 only) are:

- Support of IrDA 1.4 slow infrared (SIR), medium infrared (MIR), and fast infrared (FIR) communications
  - Frame formatting: Addition of variable xBOF characters and end of frame (EOF) characters
  - Uplink/downlink cyclic redundancy check (CRC) generation/detection
  - Asynchronous transparency (automatic insertion of break character)
  - 8-entry status FIFO (with selectable trigger levels) available to monitor frame length and frame errors
  - Framing error, CRC error, illegal symbol (FIR), and abort pattern (SIR, MIR) detection

Table 17−2 lists the baud rates, divisor values, and error rates for UART/IrDA mode.

Table 17−2. 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	¼	2	0
1,152,000	MIR	¼	1	0
4,000,000	FIR	4_{PPM}	1	0

17.1.3 CIR Features

Customer infrared (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 is to transmit and receive data packets according to the user-defined frame structure and packet content.

CIR mode (UART3 only) features support for the following remote control applications:

- Transmit and receive
- 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

17.2 UART/IrDA/CIR Environment

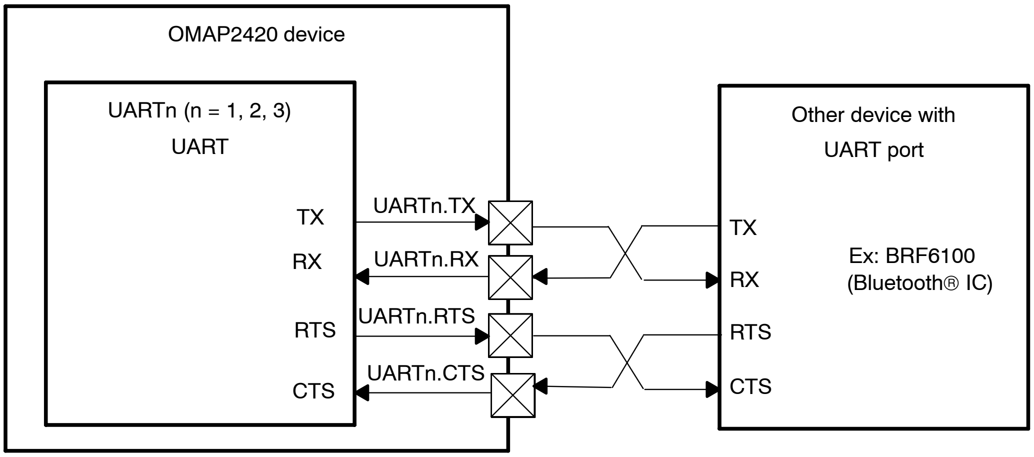
This section describes the UART/IrDA/CIR connection with external devices.

17.2.1 System Using UART Communication With Hardware Handshake

UART1, UART2, or UART3 can be easily connected to the UART port of an external IC.

Figure 17−2 is an overview of the UART mode bus system.

Figure 17−2. UART Mode Bus System Overview

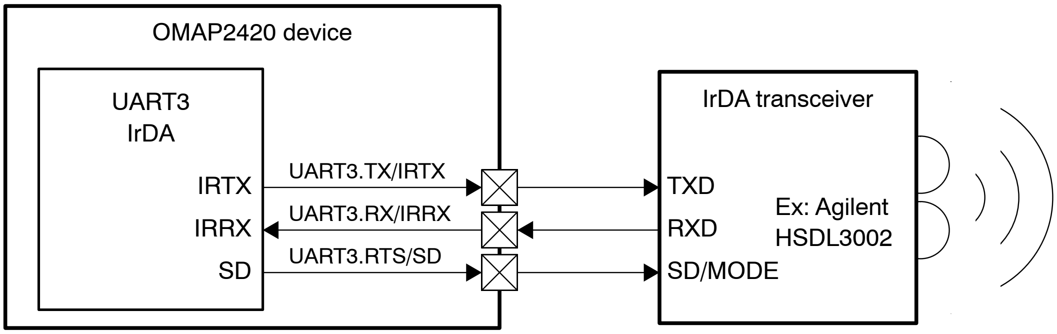


17.2.2 System Using CIR Communication Protocol With Remote Control

UART3 can be connected to an external infrared transceiver in IrDA mode (FIR, SIR, MIR).

Figure 17−3 is an overview of the infrared system.

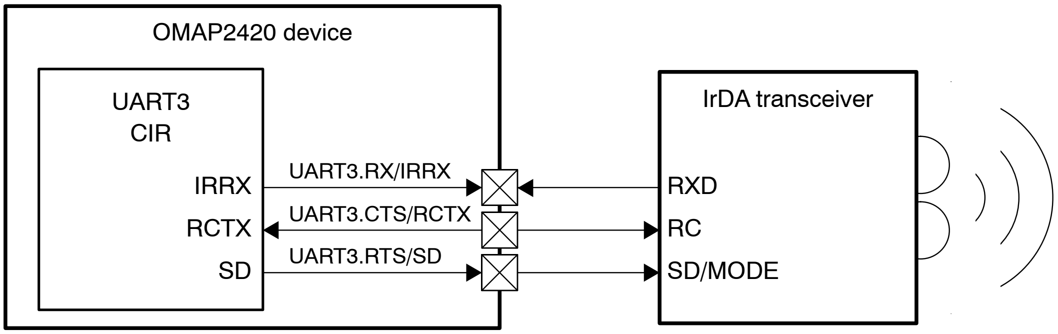
Figure 17−3. Infrared System Overview



17.2.3 System Using CIR Communication Protocol With Remote Control

UART3 can be connected to an external Infrared transceiver in CIR mode (see Figure 17−4).

Figure 17−4. CIR System Overview



17.2.4 UART Functional Interfaces

This section discusses the UART functional interfaces.

17.2.4.1 UART Interface Description

Table 17−3 describes the UART input/output (I/O) pins.

Table 17−3. UART I/O Pin Description

Signal	I/O[1]	Description	Reset
UART MODEM Signals			
RX	I	Serial data input	Unknown
TX	O	Serial data output. Because the IRTX is active high and the output is muxed, this pin is set to low on reset (when MDR1 is 111) and takes the defined inactive level of that signal corresponding to when and how MDR1 is programmed. That is, the output is 1 (inactive for modem modes) and 0 (inactive for IrDA modes).	0
nCTS	I	Clear to send Active-low modem status signal. Reading bit 4 of the modem status register checks the condition of nCTS. Reading bit 0 of that register checks a change of state of nCTS since the last read of the modem status register. nCTS is used in auto-nCTS mode to control the transmitter.	0
nRTS	O	Request to send When active (low), the module is ready to receive data. Setting modem control register bit 1 activates nRTS. It becomes inactive as a result of a module reset, loopback mode, or by clearing the MCR[1]. In auto-RTS mode, it becomes inactive as a result of the receiver threshold logic.	0

1) I = Input, O = Output

17.2.4.2 UART Protocol and Data Format

The UART device can operate in three different modes:

- ❑ UART 16x mode (≤230.4 Kbps)
- ❑ UART 16x mode with autobauding (≥1200 bps and ≤115.2 Kbps)
- ❑ UART 13x mode (≥460.8 Kbps)

CAUTION

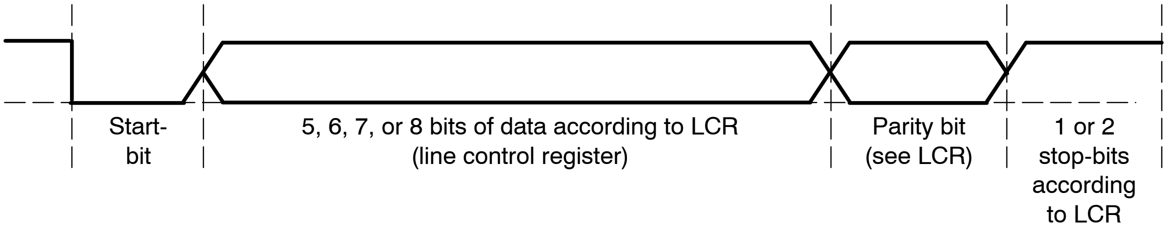
To be used as a UART, the operating mode must be programmed appropriately in MDR1, to select one of the three aforementioned modes.

The UART uses a wired interface for serial communication with a remote device.

The UART module is functionally compatible with the TL16C750 UART and with earlier designs such as the TL16C550.

Figure 17−5 shows the UART frame data format.

Figure 17−5. UART Frame Data Format


Start-bit
5, 6, 7, or 8 bits of data according to LCR (line control register)
Parity bit (see LCR)
1 or 2 stop-bits according to LCR

17.2.5 IrDA Functional Interfaces

This section discusses the IrDA functional interfaces.

17.2.5.1 UART3 Interface Description

Table 17−4 describes the UART3 I/Os.

Table 17−4. UART3 I/O Description

Signals	I/O	Description	Reset
IrDA Signals			
IRRX	I	Serial data input	Unknown
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
SD	O	SD mode is used to configure the transceivers. The SD pinout is an inverted value of ACREG[6].	1

I = Input, O = Output

17.2.5.2 IrDA Protocol and Data Format

This section discusses the IrDA protocol and data format.

SIR Mode

In SIR mode, data transfer occurs between the local host (LH) and peripheral devices at speeds of up to 115,200 baud. A SIR transmit frame starts with start flags (either a single 0xC0, multiple 0xC0, or a single 0xC0 preceded by a number of 0xFF flags), followed by frame data, CRC-16, and then 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 line control register (LCR).

BLR[6] is used to select whether 0xC0 or 0xFF start patterns are to be used when multiple start flags are required. The SIR transmit state-machine attaches start flags, CRC-16, and stop flags. It 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 is about to be transmitted, the SIR state-machine 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, removes any transparency from the incoming data, and determines the frame boundary with reception of the stop flag. It also checks for errors such as frame abort (0x7D character followed immediately by a 0xC1 stop flag, without transparency), CRC error, and frame-length error. At the end of a frame reception, the LH reads the line status register (LSR) to find possible errors of the received frame.

Note:

When the device is transmitting, data can be transferred both ways by the module. The IR RX circuitry is automatically disabled by hardware. For a description of the logical operation, see the auxiliary control register (ACREG[5]). This applies to all three modes: SIR, MIR, and FIR.

The infrared output in SIR mode can either be 1.6 μs or 3/16 encoding, selected by the PULSE_TYPE bit of the ACREG[7] register. In 1.6-μs encoding, the infrared pulse width is 1.6 μs; in 3/16 encoding, the infrared pulse width is 3/16 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.

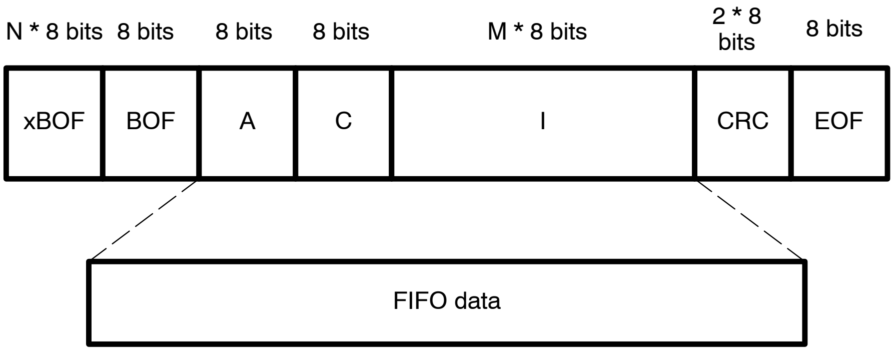
Note:

Reception supports variable-length stop-bits.

Frame Format

Figure 17−6 shows the IrDA SIR frame format.

Figure 17−6. IrDA SIR Frame Format



The CRC is applied on the address (A), control (C), and information (I) bytes.

Note:

The two words of the CRC are written in the FIFO in reception.

Asynchronous Transparency

Before transmitting a byte, the UART/IrDA controller examines each byte of the payload and the CRC field (between the beginning of frame [BOF] and EOF). For each byte equal to 0xC0 (BOF), 0xC1 (EOF), or 0x7D (control escape), it does the following.

- ❑ In transmission:
  - Insert a control escape (CE) byte before the byte.
  - Complement bit 5 of the byte (that is, exclusive OR the byte with 0x20).

The byte sent for the CRC is the initial byte written in the TX FIFO (before the XOR with 0x20).

- ❑ In reception:

  For the A, C, I, CRC field:

  - Compare the byte with the CE byte; if it is not equal to CE, send it to the CRC detector and store it in the RX FIFO.
  - If it is equal to CE, discard the CE byte.
  - Complement bit 5 of the byte following the CE byte.
  - Send the complemented byte to the CRC detector and store it in the RX FIFO.

Abort Sequence

The transmitter may decide to prematurely close a frame. The transmitter aborts by sending the 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 are received without transparency.

Pulse Shaping

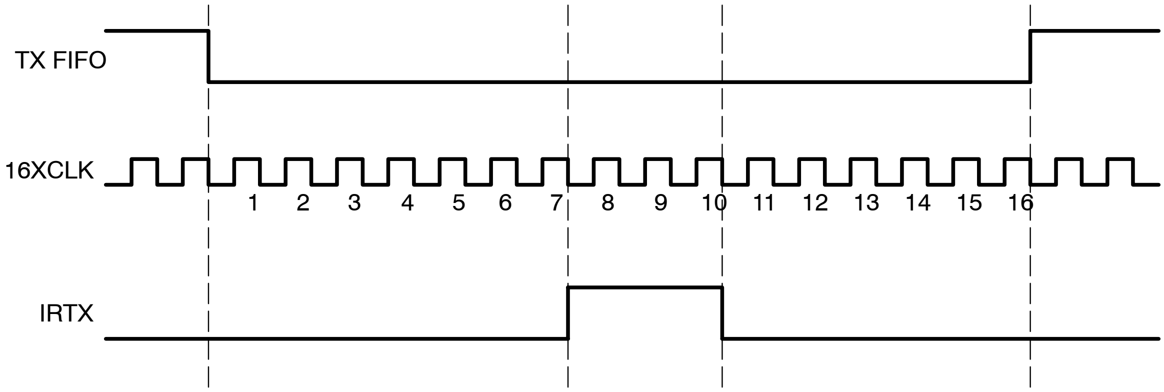
In SIR mode, both the 3/16 and the 1.6-µs pulse duration methods are supported.

Encoder

Serial data from the transmit state-machine is encoded to transmit data to the optoelectronics. While the TX FIFO output is high, the IRTX line is always low, and the counter used to form a pulse on IRTX is cleared continuously. After TXD resets to 0, IRTX rises on the falling edge of the 7th 16XCLK. On the falling edge of the 10th 16XCLK pulse, IRTX falls, creating a three-clock-wide pulse. While TXD stays low, a pulse is transmitted during the 7th to the 10th clock of each 16-clock bit cycle.

Figure 17−7 shows the IrDA SIR encoding mechanism.

Figure 17−7. IrDA SIR Encoding Mechanism

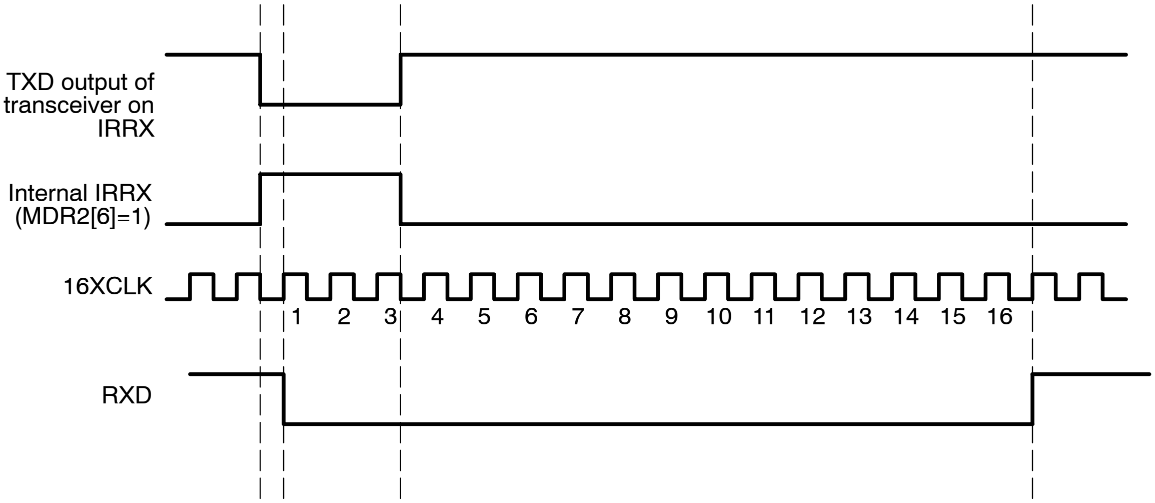


Decoder

After reset, RXD is high and the 4-bit counter is cleared. When a rising edge is detected on RX, RXD falls on the next rising edge of 16XCLK with sufficient setup time. RXD 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, RXD remains high.

Figure 17−8 shows the IrDA SIR decoding mechanism.

Figure 17−8. IrDA SIR Decoding Mechanism



Note:

When the device is transmitting, data can be transferred both ways by the module. The IRRX circuitry is automatically disabled by hardware. The operation of the IRRX input can be disabled with the DIS_IR_RX[5] bit of the ACREG register. Furthermore, the MDR2[6] bit can be used to invert the signal from the transceiver (RX output) pin to the IRRX logic inside the UART. This inversion is performed by default.

IR Address Checking

In all IR modes, if address checking is enabled, only frames intended for the device are written to the RX FIFO. This is done to avoid receiving frames not meant for this device in a multipoint IR environment. It is possible to program two frame addresses that the UART/IrDA receives with XON1/ADDR1 and XON2/ADDR2 registers.

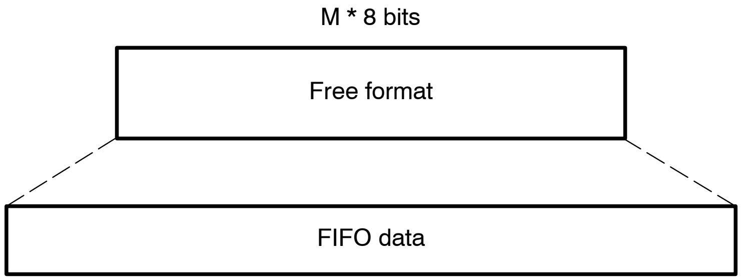
17.2.5.3 SIR Free Format Mode

To allow complete software flexibility in transmitting and receiving infrared data packets, the SIR free format mode is a subfunction of the existing SIR mode; consequently, 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. In SIR free format (FF) mode only, the data termed the FIFO data area (see Figure 17−9) is transmitted and received from the TX and RX pins.

Figure 17−9. SIR Free Format Mode

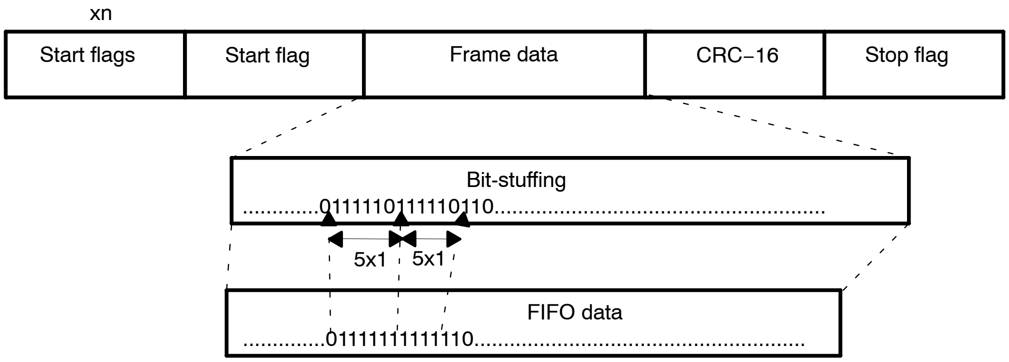


In this mode, the entire FIFO data packet is to be constructed (encoded and decoded) by the LH software.

17.2.5.4 MIR Mode

In MIR mode, data transfer occurs between the LH and peripheral devices at 0.576 or 1.152 Mbps. A MIR transmit frame starts with start flags (at least two), followed by a frame data, CRC-16, and then ends with a stop flag (see Figure 17−10).

Figure 17−10. MIR Transmit Frame Format



On transmit, the MIR state-machine attaches start flags, CRC-16, and stop flags. It also looks for five consecutive 1s in the frame data and automatically inserts 0 after five consecutive 1s (this is called bit-stuffing).

On receive, the MIR receive state-machine recovers the receive clock, removes the start flags, destuffs the incoming data, and determines frame boundary with reception of the stop flag. It also checks for errors such as frame abort, CRC error, and frame-length error. At the end of a frame reception, the LH reads the LSR to find possible errors of the received frame.

Note:

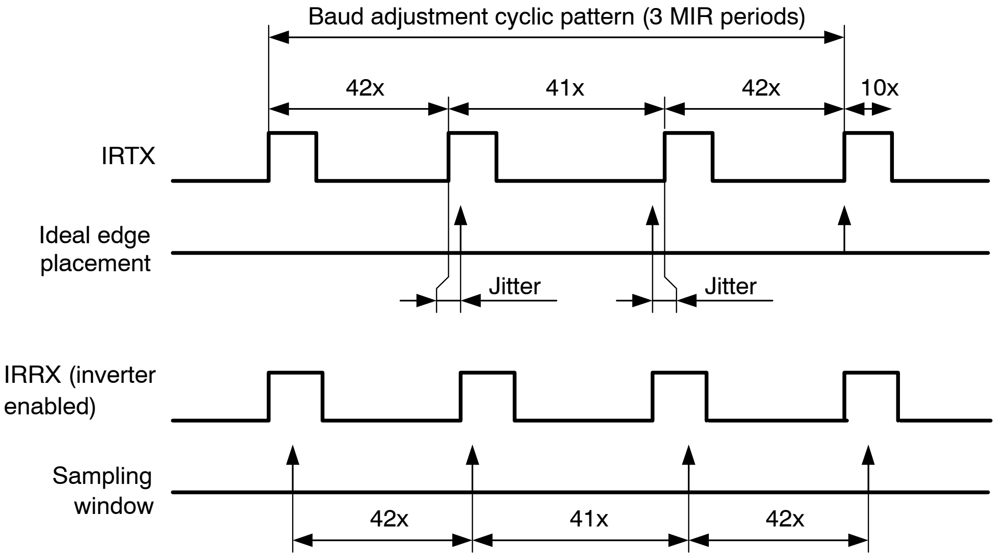
When the device is transmitting, data can be transferred both ways by the module. The IRRX circuitry is automatically disabled by hardware.

MIR Encoder/Decoder

To meet 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 1st start flag, and the 42−41−42 cyclic pattern is repeated until the stop flag is sent or detected. Figure 17−11 shows the mechanism for adjusting the MIR baud rate.

The jitter created this way is within MIR tolerances. The pulse width is not exactly 1/4, but it is within tolerances defined by the IrDA specifications.

Figure 17−11. MIR Baud Rate Adjustment Mechanism

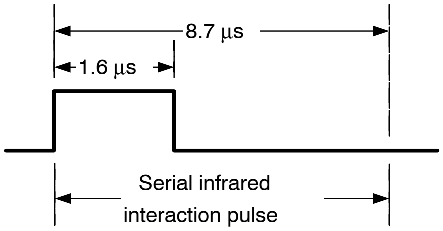


SIP Generation

In MIR and FIR operation modes, the transmitter must send a serial infrared interaction pulse (SIP) at least once every 500 ms. The purpose of the SIP is to let slow devices (operating in SIR mode) know that the medium is occupied.

Figure 17−12 shows the SIP.

Figure 17−12. SIP



17.2.5.5 FIR Mode

In FIR mode, data transfer occurs between the LH and peripheral devices at 4 Mbps. A FIR transmit frame starts with a preamble, followed by a start flag, frame data, CRC-32, and ends with a stop flag (see Figure 17−13).

Figure 17−13. FIR Transit 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. It also encodes the transmit data into 4PPM format and generates the SIP.

On receive, the FIR receive state-machine recovers the receive clock, removes the start flag, decodes the 4PPM (pulse position modulation) incoming data, and determines the frame boundary with reception of the stop flag. It also checks for errors such as illegal symbol, CRC error, and frame-length error. At the end of a frame reception, the LH reads the LSR to determine possible errors of the received frame.

Note:

When the device is transmitting, data can be transferred both ways by the module. The IRRX circuitry is automatically disabled by hardware.

17.2.6 CIR Functional Interfaces

This section describes the CIR functional interfaces.

17.2.6.1 CIR Interface Description

Table 17−5 describes the CIR I/Os.

Table 17−5. CIR I/O Description

Signal	I/O	Description	Reset
CIR Signals			
IRRX	I	Serial data input	Unknown
RCTX	O	Serial data output in CIR mode. In other modes, this pin is set to the reset value (inactive state).	0
SD	O	SD mode is used to configure the transceivers. The SD pinout is an inverted value of ACREG[6].	1

I = Input, O = Output

17.2.6.2 CIR Protocol and Data Format

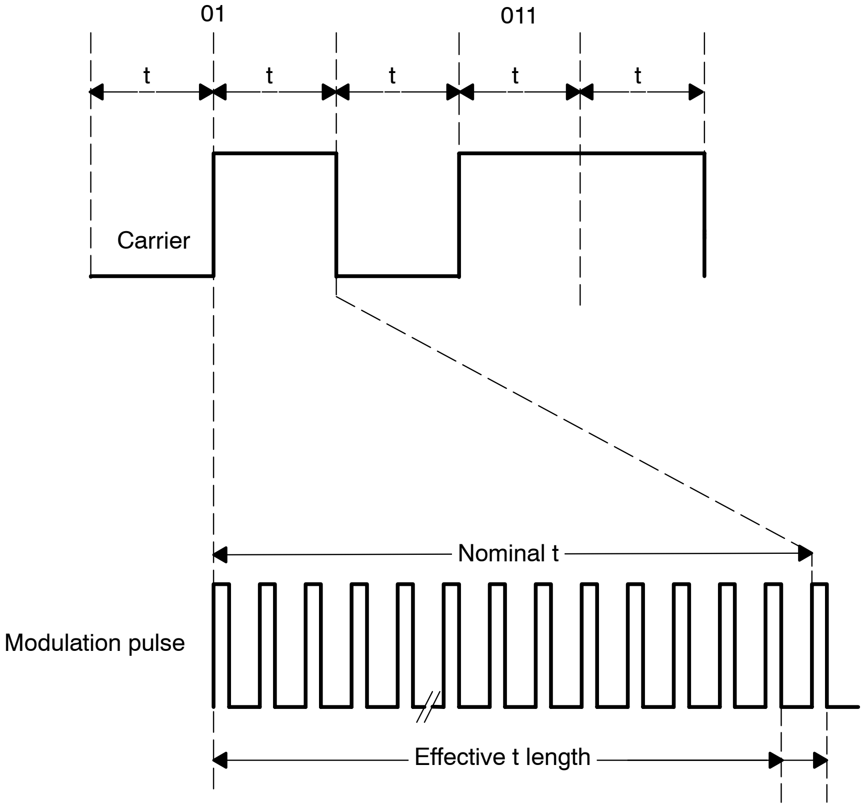
In CIR mode, the infrared operation is designed to function as a programmable (universal) remote control.

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 is to transmit and receive data packets according to the user-definable frame structure and packet content.

Carrier Modulation

Each modulated pulse that constitutes a digit is a train of on/off pulses (see Figure 17−14).

Figure 17−14. CIR Pulse Modulation

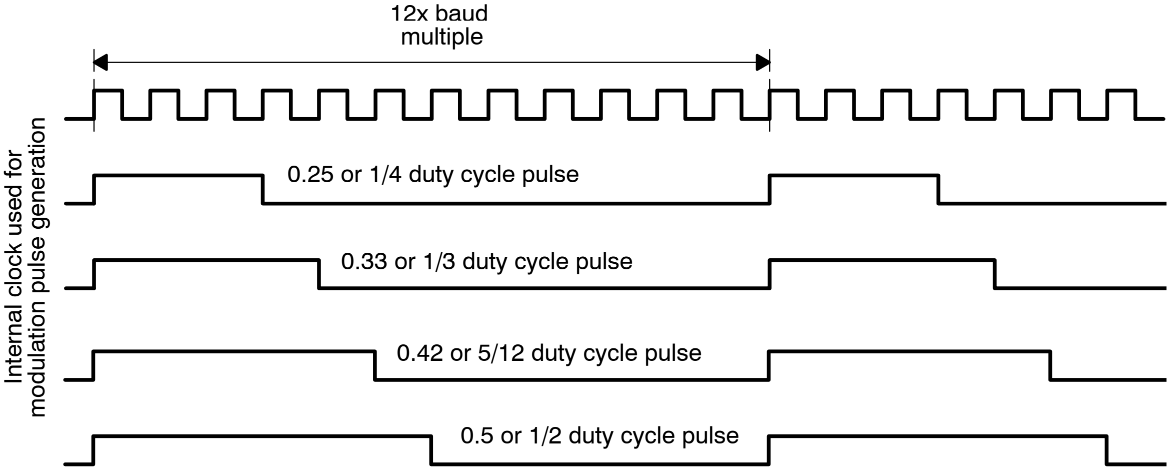


The module requires a minimum of four modulation pulses per bit.

Pulse Duty Cycle

The programmer can choose between four possible duty cycle for modulation pulses: 1/4, 1/3, 5/12, or 1/2 (see Figure 17−15).

Figure 17−15. CIR Modulation Duty Cycle


12x baud
multiple
Internal clock used for modulation pulse generation
0.25 or 1/4 duty cycle pulse
0.33 or 1/3 duty cycle pulse
0.42 or 5/12 duty cycle pulse
0.5 or 1/2 duty cycle pulse

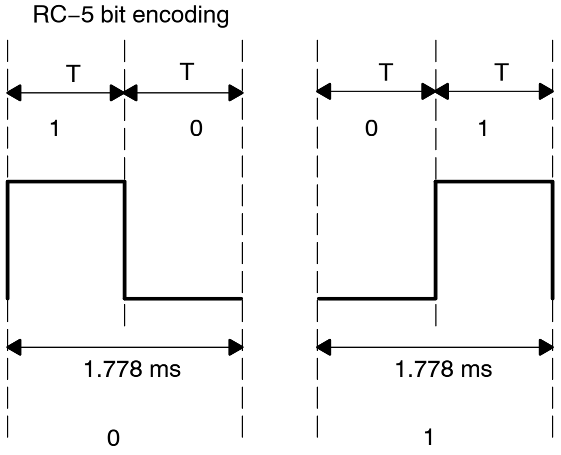
The transmission logic ensures that all pulses are transmitted completely (that is, no pulse is cut off during its transmission). Furthermore, while transmitting continuous bytes back-to-back, no delay is inserted between 2 transmitted bytes.

Consumer IR Encoding/Decoding

There are two distinct methods of encoding for remote control applications. The first method uses time-extended bit forms, that is, a variable pulse distance (or duration) whereby the difference between a logic 1 and logic 0 is the length of the pulse width. The second method uses a biphase where the encoding of the logic 0 and 1 is in the change of signal level from 1 –> 0 or 0 –> 1, respectively. Japanese manufacturers tend to favor the use of pulse duration encoding, whereas European manufacturers favor the use of biphase encoding.

CIR mode is designed to use a completely flexible free-format encoding where a 1 from the TX/RX FIFO is to be transmitted/received as a modulated pulse with duration t. Equally, a 0 is to be transmitted/received as a blank duration t. The protocol of the data is to be constructed and deciphered by the LH. 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 17−16).

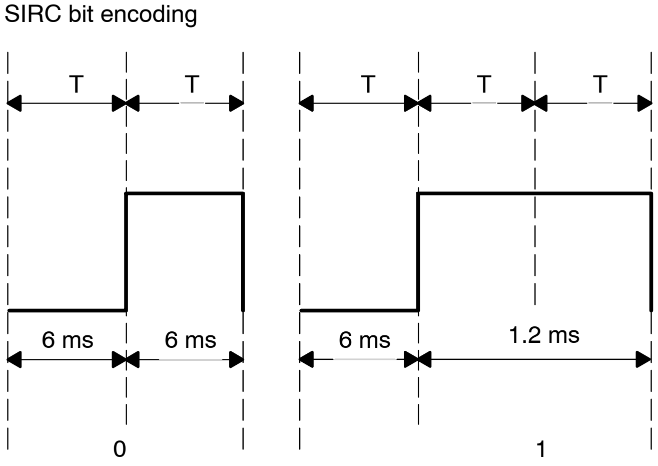
Figure 17−16. RC-5 Bit Encoding



Because the CIR mode logic does not impose a fixed format for infrared packets of data, the LH software is at liberty to define the format through the use of simple data structures that are then modulated into an industry standard such as RC5 or SIRC, and so on. To send a sequence of 0101 in RC5, the host 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 to distinguish between a 1 or a 0. Figure 17−17 shows the SIRC bit encoding.

Figure 17−17. SIRC Bit Encoding



To construct comprehensive packets that constitute remote control commands, the host software must combine a number of 8-bit data characters in a sequence that follows a universally accepted format. A standard RC5 frame is described in Figure 17−18 (the SIRC format follows this). Each field in RC-5 can be considered two t pulses (digital bits) from the TX and RX FIFOs.

Figure 17−18 shows the standard RC5 format as seen by the UART/IrDA in CIR mode.

Figure 17−18. RC-5 Standard Packet Format

S1	S2	T	A4	A3	A2	A1	A0	C5	C4	C3	C2	C1	C0

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 each time a new command is transmitted, to allow detection of pressing the same key twice (or effectively receiving the same data from the host consecutively). Because a code is being sent as long as the LH transmits characters to the UART for transmission, a brief delay in the transmission of the same command is detected by the use of the toggle bit. The address bits define the machine or device for which the infrared transmission is intended, and the command defines the operation.

To accommodate a format known as the extended RC-5 format, the S2 bit is replaced by a further command bit (C6) that allows the command range to increase to 7 bits.

The SIRC encoding uses the duration of modulation for mark and space; hence, the duration of data bits inside the standard frame length varies depending on the logic 1 content. Figure 17−19 shows the packet format and bit encoding. There is 1 start-bit of 2 ms and control codes followed by data that constitute the whole frame.

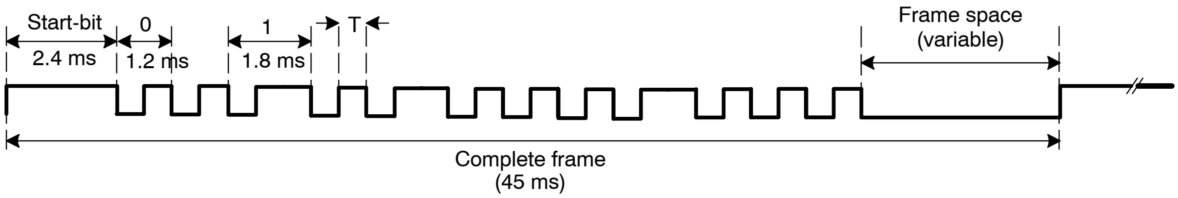
Figure 17−19. SIRC Packet Format

S	C0	C1	C2	C3	C4	C5	C6	D0	D1	D2	D3	D4

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 17−20 is an example of SIRC bit transmission.

Figure 17−20. SIRC Bit Transmission Example



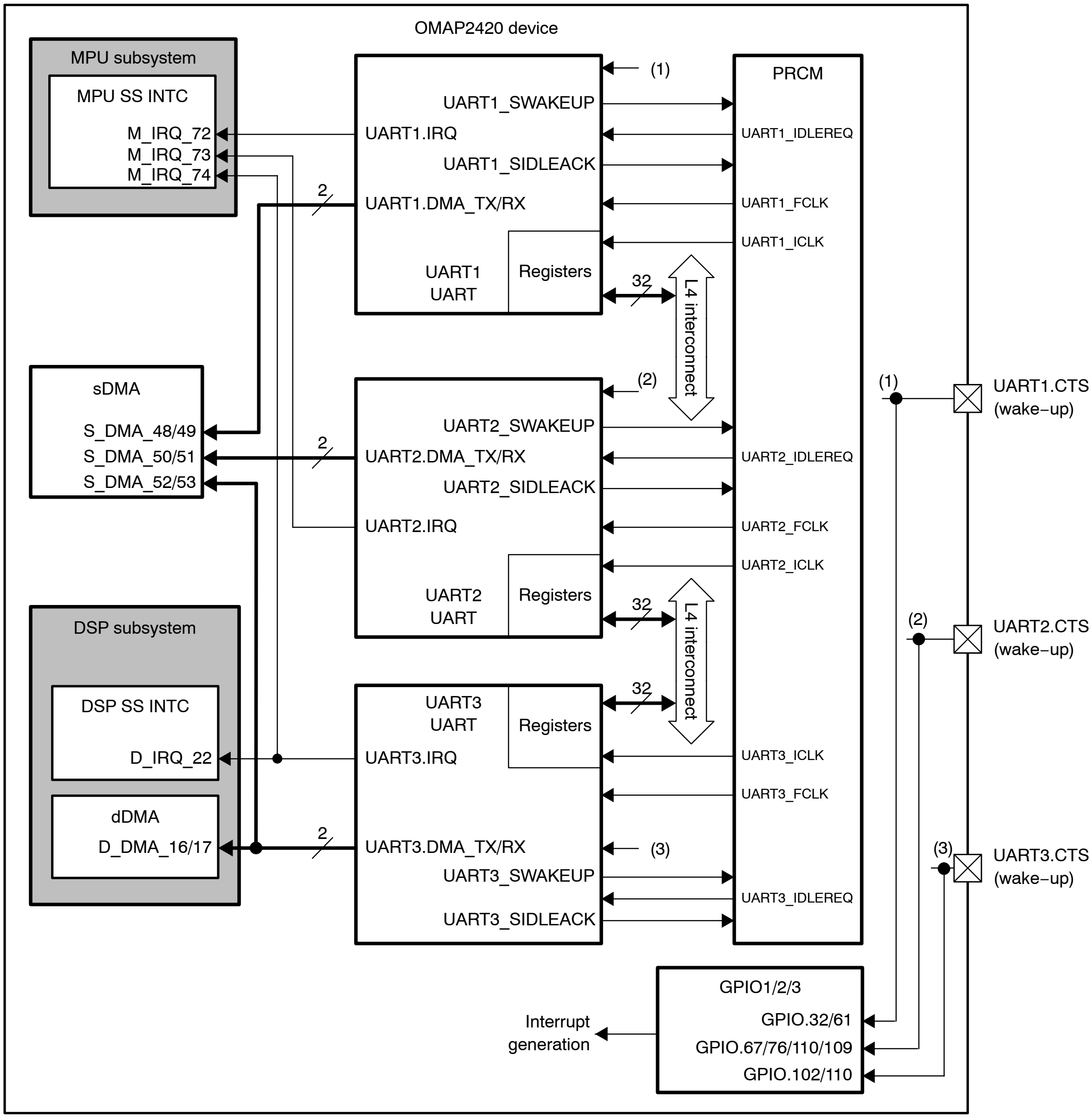
It is beyond the scope of this document to describe all encoding methods and techniques; the information in this section is provided to indicate the considerations required to employ different encoding methods for different industry standard protocols. Refer to industry standard documentation for specific methods of encoding and protocol usage.

17.3 UART/IrDA/CIR Integration

17.3.1 UART Functional Description

Figure 17−21 shows the OMAP2420 device internal connections with related modules for UART functions.

Figure 17−21. UART Functional Integration Inside OMAP2420 Devices



17.3.1.1 Clocking, Reset, and Power-Management Scheme

Clocking

Each UART uses a 48-MHz functional clock for its logic and for the generation of external interface signals. Each UART use an interface clock to interface with the L4 interconnect (register accesses) (see Table 17−6). These clocks are generated and controlled by the power, reset, and clock management (PRCM) module (see Chapter 5, *Power, Reset, and Clock Management*).

Table 17−6. UART Clocks

	Frequency	Name	Comments
UART1			
Functional clock	FUNC_48M_CLK	UART1_FCLK	Source and gating is PRCM module. CM_FCLKEN1_CORE[21] (EN_UART1)
Interface clock	CORE_L4_ICLK	UART1_ICLK	Source and gating is PRCM module. CM_ICLKEN1_CORE[21] (EN_UART1)CM_AUTOIDLE1_CORE[21] (AUTO_UART1)
UART2			
Functional clock	FUNC_48M_CLK	UART2_FCLK	Source and gating is PRCM module. CM_FCLKEN1_CORE[22] (EN_UART2)
Interface clock	CORE_L4_ICLK	UART2_ICLK	Source and gating is PRCM module. CM_ICLKEN1_CORE[22] (EN_UART2)CM_AUTOIDLE1_CORE[22] (AUTO_UART2)
UART3			
Functional clock	FUNC_48M_CLK	UART3_FCLK	Source and gating is PRCM module. CM_FCLKEN2_CORE[2] (EN_UART3)
Interface clock	CORE_L4_ICLK	UART3_ICLK	Source and gating is PRCM module. CM_ICLKEN2_CORE[2] (EN_UART3)CM_AUTOIDLE2_CORE[2] (AUTO_UART3)

Hardware Reset

Each UART receives a reset signal (the CORE_RST signal) from the PRCM module that is the reset signal to the CORE power domain. See Chapter 5, *Power, Reset, and Clock Management*, for details.

Table 17−7 lists the peripherals and reset domains.

Table 17−7. Power and Reset Domain

Peripherals	Reset Domain
UART1	CORE_RST
UART2	CORE_RST
UART3	CORE_RST

Software Reset

A software bit is available for the UART: bit [1] of the system configuration (SYSC) register. When the software reset bit is set high, it causes a full device reset. See Section 17.6, *UART/IrDA/CIR Registers*, for details.

Power Domain

Each UART connects into the CORE power domain, which can dynamically switch between low voltage and high voltage. Table 17−8 lists the peripherals in the CORE power domain.

Table 17−8. Power Domain

Peripherals	Power Domain
UART1	CORE
UART2	CORE
UART3	CORE

17.3.1.2 Hardware Requests

This section discusses hardware requests.

Interrupts

Table 17−9 describes interrupt mapping to the MPU subsystem. Table 17−10 describes the level 2 interrupt to the DSP subsystem.

Table 17−9. 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

Table 17−10. Level 2 Interrupt to DSP Subsystem

IRQ	Source	Description
D_IRQ_22	UART3_IRQ	UART module 3 interrupt to DSP

DMA Requests

Table 17−11 lists the UART DMA requests to the peripheral DMA. Table 17−12 lists the UART DMA requests to the DSP subsystem DMA.

Table 17−11. UART DMA Requests to Peripheral DMA

DMA	Source	Description
P_DMA_48	UART1_DMA_TX	UART module 1–transmit request
P_DMA_49	UART1_DMA_RX	UART module 1–receive request
P_DMA_50	UART2_DMA_TX	UART module 2–transmit request
P_DMA_51	UART2_DMA_RX	UART module 2–receive request
P_DMA_52	UART3_DMA_TX	UART module 3–transmit request
P_DMA_53	UART3_DMA_RX	UART module 3–receive request

Note: This table assumes DMA mode 1 is used.

Table 17−12. UART DMA Requests to DSP Subsystem DMA

DMA[†]	Source	Description
D_DMA_16	UART3_DMA_TX	UART module 3–transmit request
D_DMA_17	UART3_DMA_RX	UART module 3–receive request

Note: This table assumes DMA mode 1 is used.

Wake-Up Request

The UART can wake up the system on different events (see Section 17.6, *UART/IrDA/CIR Registers*).

One particularly important wake-up generation is event detection on the UARTn_CTS pin when the system is idle. UARTn_CTS 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 17−13).

Table 17−13. 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 on the CORE power domain, not the WKUP domain, which implies limitations on wake-up capability. If the CORE power domain is off, the UART cannot wake up the system, because power is not supplied. In that case, a GPIO multiplexed on the UARTn_CTS pin can be used to capture the event and wake up the system. Software must enable this strategy, if required. See Chapter 19, *Multichannel SPI*, for a description of this mechanism.

The following GPIOs are multiplexed on UARTn_CTS pins:

- ❑ UART1_CTS: GPIO.32 or GPIO.61
- ❑ UART2_CTS: GPIO.67, GPIO.76, GPIO.109, or GPIO.110

- ☐ UART3_CTS: GPIO.102 or GPIO.110

The sleep and wake-up processes use a handshake protocol between the PRCM and the UARTS. See Chapter 5, *Power, Reset, and Clock Management*, for a description of the protocol.

Table 17−14 lists the clock management requests.

Table 17−14. Clock Management Requests

Request	Description
UARTn_IDLEREQ	Idle request from the PRCM to the UARTn. If request is accepted by UARTn, UARTn_SIDLEACK is sent back to the PRCM.
UARTn_SIDLEACK	Acknowledge from the UARTn to the PRCM in answer to UARTn_IDLEREQ

17.3.1.3 Pin List and Pad Multiplexing With Other Functions

UART3 has some pins internally multiplexed with the IrDA/CIR functions (see Table 17−15). This is transparent, and this multiplexing is automatically set when the mode (UART/IrDA/CIR) is selected at module level.

Table 17−15. Pin List and Pad Multiplexing

UART1 Interface	**Description**	**DIR**	**Ball 2420**	**ALTERNATE FUNCTIONS**					
				Mode0	**Mode1**	**Mode2**	**Mode3**	**Mode4**	**Mode5**
UART1.CTS	UART1 clear to send	I	D21	▨	■	dss.d18	gpio.32	■	■
			P13		▨	usb1.txen	gpio.61	■	■
UART1.RTS	UART1 request to send	O	H21	▨	■	dss.d19	gpio.8	■	■
			V12		▨	usb1.rcv	gpio.25	■	■
UART1.RX	UART1 receive data	I	T21	▨	■	dss.d21	gpio.10	■	■
			R13	gpio.62	▨	usb1.dat	gpio.62	■	■
UART1.TX	UART1 transmit data	O	L20	▨	■	dss.d20	gpio.9	■	■
			W12		▨	usb1.se0	gpio.59	■	■

UART2 Interface	**Description**	**DIR**	**Ball 2420**	**ALTERNATE FUNCTIONS**					
				Mode0	**Mode1**	**Mode2**	**Mode3**	**Mode4**	**Mode5**
UART2.CTS	UART2 clear to send	I	V19	▨	usb1.rcv	gpt9.pwm/evt	gpio.67	■	■
			E19	mmc.dat2		▨	gpio.76	■	■
			K19	usb0.txen	uart3.cts/rctx	▨	gpio.110	■	■
			J18	usb0.rcv	mcbsp2.fsx	■	gpio.109	uart2.cts	■
UART2.RTS	UART2 request to send	O	W20	▨	usb1.txen	gpt10.pwm/evt	gpio.68	■	■
			E20	mmc.dat_dir1		▨	gpio.78	■	■
UART2.RX	UART2 receive data	I	P15	▨	usb1.dat	gpt12.pwm/evt	gpio.70	■	■
			E18	mmc.dat_dir3		▨	gpio.80	■	■
			K18	usb0.dat	uart3.rx/irrx	▨	gpio.112	uart2.tx	■
			K20	usb0.vm	mcbsp2.clkx	■	gpio.108	▨	■
			J14	usb0.se0	uart3.tx/irtx	uart2.tx	gpio.111	▨	■

Table 17−15. Pin List and Pad Multiplexing (Continued)

UART3 Interface	Description	DIR	Ball 2420	ALTERNATE FUNCTIONS					
				Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
UART2.TX	UART2 transmit data	O	N14		usb1.se0	gpt11.pwm/evt	gpio.69		
			F18	mmc.dat_dir2	ms.datu_dir		gpio.79		
			J14	usb0.se0	uart3.tx/irtx		gpio.111	uart2.rx	
			K18	usb0.dat	uart3.rx/irrx	uart2.rx	gpio.112		
UART3.CTS/RCTX	UART3 clear To send (input)	IO	L18		uart3.rx/irrx		gpio.102		
			K19	usb0.txen		uart2.cts	gpio.110		
UART3.RTS/SD	UART3 Request To Send or IrDA transceiver shutdown/mode select	O	L19		uart3.tx/irtx		gpio.103		
UART3.RX/IRRX	UART3 receive ata	I	K14				gpio.105		
			L18	uart3.cts/rctx			gpio.102		
			K18	usb0.dat		uart2.rx	gpio.112	uart2.tx	
UART3.TX/IRTX	UART3 transmit data	O	K15		uart3.rctx		gpio.104		
			L19	uart3.rts/sd			gpio.103		
			J14	usb0.se0		uart2.tx	gpio.111	uart2.rx	

17.3.1.4 UART L4 Interconnect Target Agent Registers

Each UART module connects to the peripheral L4 interconnect through a target agent (TA). See Chapter 6, *Internal Interconnect*, for the full description. Table 17−16 provides a register summary for each of the three TAs.

Table 17−16. UART Target Agent Base Addresses

UART Instance	TA Name	Base Address
UART1	L4TA19	0x4806 B000
UART2	L4TA20	0x4806 D000
UART3	L4TA21	0x4806 F000

The physical addresses in Table 17−17 are determined by replacing X in the Physical Address column with the corresponding value provided in the Base Address column of Table 17−16.

Example: The AGENT_CONTROL register physical address for UART1 is 0x4806 B020.

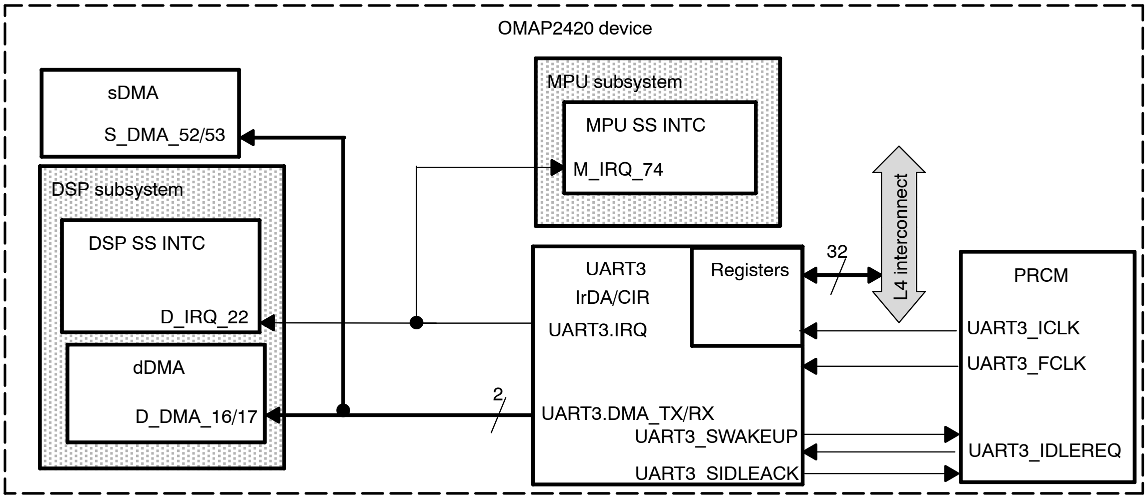
Table 17−17. UART Target Agent Register Summary

Register Name	Type	Register Width (Bits)	Physical Address
COMPONENT	R	32	0x4806 X000
AGENT_CONTROL	RW	32	0x4806 X020
AGENT_STATUS	R	32	0x4806 X028

17.3.2 IrDA and CIR Functional Description (UART3 Only)

Figure 17−22 shows the OMAP2420 device internal connections with related modules for the IrDA and CIR functions. Integration is identical for IrDA mode and CIR mode; the effect on pin usage is slightly different and is treated separately in this section.

Figure 17−22. IrDA and CIR Function Integration Inside OMAP2420 Devices



17.3.2.1 Clocking, Reset, and Power-Management Scheme

Clocking

UART3 use a 48-MHz functional clock for its logic and for the generation of external interface signals for IrDA and CIR functions. UART3 uses an interface clock to interface with the L4 interconnect (register accesses). These clocks are generated and controlled by the PRCM module (see Chapter 5, *Power, Reset, and Clock Management*, for details).

Table 17−18 lists the IrDA and CIR clocks.

Table 17−18. IrDA and CIR Clocks

	Frequency	Name	Comments
UART3			
Functional clock	FUNC_48M_CLK	UART3_FCLK	Source and gating is PRCM module. CM_FCLKEN2_CORE[2] (EN_UART3)
Interface clock	CORE_L4_ICLK	UART3_ICLK	Source and gating is PRCM module. CM_ICLKEN2_CORE[2] (EN_UART3) CM_AUTOIDLE2_CORE[2] (AUTO_UART3)

Hardware Reset

UART3 receives a reset signal (the CORE_RST signal) from the PRCM module that is the reset signal to the CORE power domain. See Chapter 5, *Power, Reset, and Clock Managment*, for details.

Software Reset

A software bit is available for UART: bit [1] of the system configuration (SYSC) register. When the software reset bit is set high, it causes a full device reset. See Section 17.6, *UART/IrDA/CIR Registers*, for details.

Power Domain

UART3 connects into the CORE power domain, which can dynamically switch between low voltage and high voltage.

17.3.2.2 Hardware Requests

This section discusses hardware requests, including interrupts, DMA requests, and wake-up requests.

Interrupts

Table 17−19 lists the interrupt mapping to the MPU subsystem. Table 17−20 lists the level 2 interrupt to the DSP subsystem.

Table 17−19. Interrupt Mapping to MPU Subsystem

IRQ	Source	Description
M_IRQ_74	UART3_IRQ	UART module 3 interrupt to MPU

Table 17−20. Level 2 Interrupt to DSP Subsystem

IRQ	Source	Description
D_IRQ_22	UART3_IRQ	UART module 3 interrupt to DSP

DMA Requests

Table 17−21 lists the UART DMA requests to the peripheral DMA. Table 17−22 lists the UART DMA request to the DSP subsystem DMA.

Table 17−21. UART DMA Requests to Peripheral DMA

DMA	Source	Description
P_DMA_52	UART3_DMA_TX	UART module 3–transmit request
P_DMA_53	UART3_DMA_RX	UART module 3–receive request

Note: This table assumes DMA mode 1 is used.

Table 17−22. UART DMA Request to DSP Subsystem DMA

DMA	Source	Description
D_DMA_16	UART3_DMA_TX	UART module 3–transmit request
D_DMA_17	UART3_DMA_RX	UART module 3–receive request

Note: This table assumes DMA mode 1 is used.

Wake-Up Request

The UART function can wake up the system on different events (see Section 17.6, *UART/IrDA/CIR Registers*). Table 17−23 lists the clock management requests.

Table 17−23. Clock Management Requests

Request	Description
UART3_IDLEREQ	Idle request from the PRCM to the UARTn. If request is accepted by UARTn, UARTn_SIDLEACK is sent back to the PRCM.
UART3_SIDLEACK	Acknowledge from the UARTn to the PRCM in answer to UARTn_IDLEREQ
UART3_SWAKEUP	UART3 system wake-up capabilities

17.3.2.3 Pin List and Pad Multiplexing With Other Functions for IrDA

Pins are multiplexed at the module level with the UART function. This is transparent. Correct pin assignment is done automatically when IrDA mode is set at the module level.

Table 17−24 provides the IrDA pin list and pad multiplexing.

Table 17−24. IrDA Pin List and Pad Multiplexing

UART3 Interface	Description	DIR	Ball 2420	ALTERNATE FUNCTIONS					
				Mode0	**Mode1**	**Mode2**	**Mode3**	**Mode4**	**Mode5**
UART3.RTS/ SD	IrDA transceiver shutdown/mode select	O	L19	▒	uart3.tx/irtx	■	gpio.103	■	■
UART3.RX/ IRRX	IrDA receive data	I	K14	▒	■	■	gpio.105	■	■
			L18	uart3.cts/ rctx	▒	■	gpio.102	■	■
			K18	usb0.dat	▒	uart2.rx	gpio.112	uart2.tx	■
UART3.TX/ IRTX	IrDA transmit data	O	K15	▒	uart3.rctx	■	gpio.104	■	■
			L19	uart3.rts/sd	▒	■	gpio.103	■	■
			J14	usb0.se0	▒	uart2.tx	gpio.111	uart2.rx	■

17.3.2.4 Pin List and Pad Multiplexing With Other Functions for CIR

Pins are multiplexed at the module level with the UART function. This is transparent. Correct pin assignment is done automatically when CIR mode is set at the module level.

Table 17−25 provides the CIR pin list and pad multiplexing.

Table 17−25. CIR Pin List and Pad Multiplexing

UART3 Interface	**Description**	DIR	Ball 2420	ALTERNATE FUNCTIONS					
				Mode0	**Mode1**	**Mode2**	**Mode3**	**Mode4**	**Mode5**
UART3.CTS/ RCTX	CIR serial transmit data output	IO	L18		uart3.rx/irrx		gpio.102		
			K19	usb0.txen		uart2.cts	gpio.110		
UART3.RCTX		O	K15	uart3.tx/irtx			gpio.104		
UART3.RTS/ SD		O	L19		uart3.tx/irtx		gpio.103		
UART3.RX/ IRRX	CIR serial input receive data	I	K14				gpio.105		
			L18	uart3.cts/ rctx			gpio.102		
			K18	usb0.dat		uart2.rx	gpio.112	uart2.tx	

17.3.2.5 IrDA/CIR L4 Interconnect Target Agent Registers

UART3 connects to the peripheral L4 interconnect through a TA (see Chapter 6, *Internal Interconnect*). Table 17−26 summarizes the UART3 TA registers.

Table 17−26. UART Target Agent Register Summary

Register Name	**Type**	**Register Width (Bits)**	**Physical Address**
COMPONENT	R	32	0x4806 F000
AGENT_CONTROL	RW	32	0x4806 F020
AGENT_STATUS	R	32	0x4806 F028

17.4 UART/IrDA/CIR Functional Description

17.4.1 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.

Mode selection is divided into two selections:

- Function mode: Routes the data to the chosen functionality (UART, IrDA, or CIR) and enables mechanism corresponding to the chosen functionality
- Register mode: Enables conditional access to registers

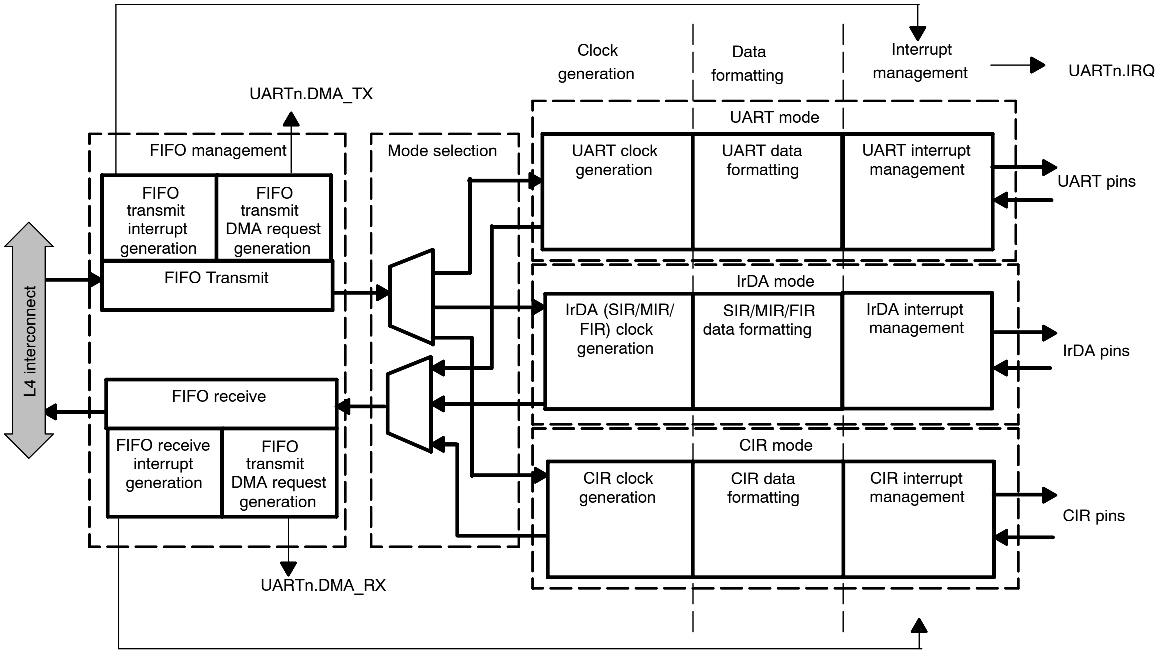
Protocol formatting can be divided into three subcategories:

- Clock generation: Required clocks are generated from the 48-MHz input clock.
- 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 17−23 is a block diagram of UART/IrDA/CIR.

Figure 17−23. UART/IrDA/CIR Block Diagram



17.4.2 FIFO Management

FIFO are accessed by reading/writing the receive holding register (RHR)/ transmit holding register (THR), and parameters are controlled using the FIFO control register (FCR) and the supplementary control register (SCR). Reading the SSR[0]:TX_FIFO_FULL register bit at 1 means the FIFO is full. The trigger level register (TLR) controls the FIFO trigger level, which enables the DMA and interrupt generation. After reset, both transmitter and receiver FIFOs are disabled; therefore, in effect, the trigger level is the default value of 1 byte. Figure 17−24 shows the FIFO management registers.

Note:

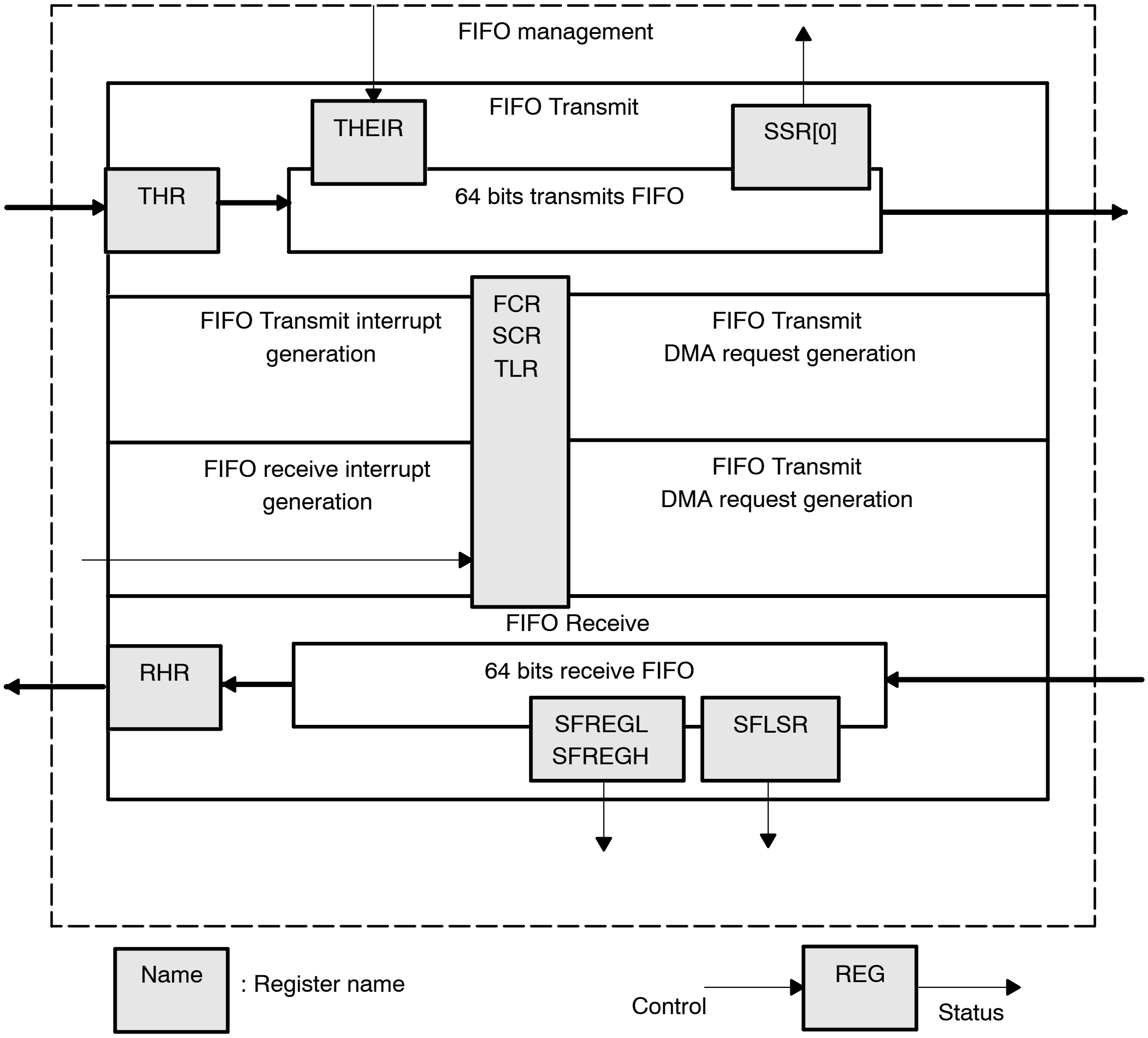
If an overflow occurs, the data in the RHR is not overwritten*.*

Note:

The status FIFO line status register (SFLSR), status FIFO register low (SFREGL) register, and status FIFO register high (SFREGH) register are used in IrDA mode only. See Section 17.4.4.4, *IrDA Mode (UART3 Only)*, subsection *IrDA Data Formatting*, for the usage.

Figure 17−24 shows the FIFO management registers.

Figure 17−24. FIFO Management Registers



17.4.2.1 FIFO Trigger

This section discusses the transmit and receive FIFO triggers.

Transmit FIFO Trigger

Table 17−27 summarizes the transmit FIFO trigger level setting.

Table 17−27. TX FIFO Trigger Level Setting Summary

SCR[6]	TLR[3:0]	TX FIFO Trigger Level
0	= 0000	Defined by FCR[5:4] (8, 16, 32, or 56 spaces)
0	≠ 0000	Defined by TLR[3:0] (from 4 to 60 spaces with a granularity of 4 spaces)
1	value	Defined by the concatenated value of TLR[3:0] and FCR[5:4] (from 1 to 63 spaces with a granularity of 1 space). Note: The combination of TLR[3:0] = 0000 and FCR[5:4] = 00 (all zeros) is not supported (minimum of 1 space required). All zeros result in unpredictable behavior.

Receive FIFO Trigger

Table 17−28 summarizes the receive FIFO trigger level setting.

Table 17−28. RX FIFO Trigger Level Setting Summary

SCR[7]	TLR[7:4]	RX FIFO Trigger Level
0	= 0000	Defined by FCR[7:6] (8, 16, 56, or 60 characters)
0	≠ 0000	Defined by TLR[7:4] (from 4 to 60 characters with a granularity of 4 characters)
1	Value	Defined by the concatenated value of TLR[7:4] and FCR[7:6] (from 1 to 63 characters with a granularity of 1 character). Note: The combination of TLR[7:4] = 0000 and FCR[7:6] = 00 (all zeros) is not supported (minimum of 1 character required). All zeros result in unpredictable behavior.

Receive threshold programming with access to the transmission control register (TCR): TCR[7:4]:RX_FIFO_TRIG_START field and TCR[3:0]: RX_FIFO_TRIG_HALT field.

- Trigger levels from 0 to 60 bytes are available with a granularity of four. (Trigger level = 4 x [4-bit register value])
- The programmer must ensure that TCR[3:0] > TCR[7:4] whenever auto-RTS is enabled to avoid improper operation of the device.
- In FIFO interrupt mode with flow control, the programmer must also ensure that the trigger level to HALT transmission exceeds or is equal to the receive FIFO trigger level (either TLR[7:4] or FCR[7:6]); otherwise, FIFO operation stalls. In FIFO DMA mode with flow control, this concept does not exist because DMA request is sent each time a byte is received.

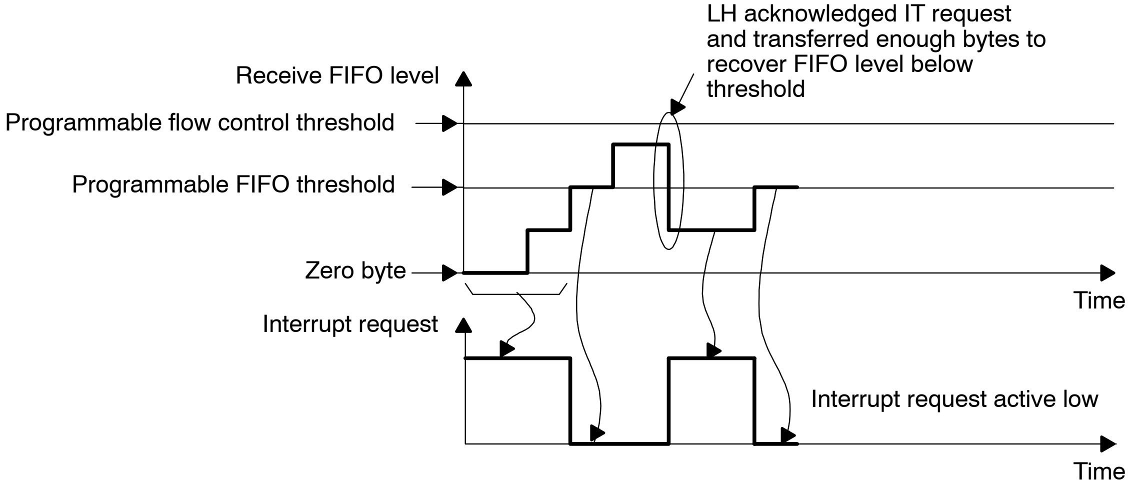
17.4.2.2 FIFO Interrupt Mode

In FIFO interrupt mode (FIFO control register FCR[0] = 1, relevant interrupts enabled using the interrupt enable register [IER]), the processor is informed of the status of the receiver and transmitter by an interrupt signal. These interrupts are raised when the receive/transmit FIFO threshold (respectively, TLR[7:4] and TLR[3:0] or FCR[7:6] and FCR[5:4]) are reached; the interrupt signals instruct the LH 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).

If the UART flow control is enabled along with the interrupt capabilities, you must ensure that the UART flow control FIFO threshold (TCR[3:0]) exceeds or is equal to the receive FIFO threshold.

Figure 17−25 shows the receive and transmit operations.

Figure 17−25. Receive FIFO Interrupt Request Generation


LH acknowledged IT request and transferred enough bytes to recover FIFO level below threshold
Receive FIFO level
Programmable flow control threshold
Programmable FIFO threshold
Zero byte
Time
Interrupt request
Interrupt request active low
Time

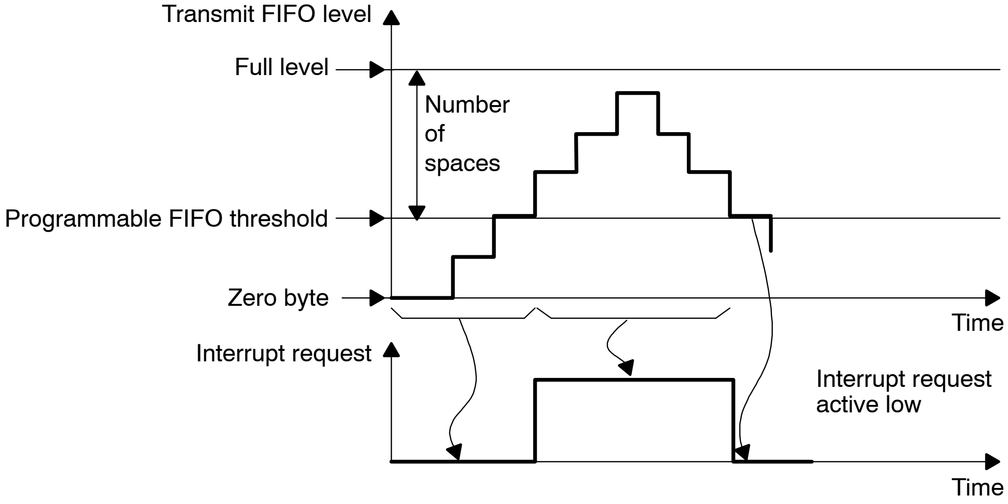
In receive, no interrupt is generated until the receive FIFO reaches its threshold. Once low, the interrupt can only be deasserted when the LH has handled enough bytes to make the FIFO level below threshold. The flow control threshold is set at a higher value than the FIFO threshold.

Receive threshold programming with access to the TCR: TCR[7:4]:RX_FIFO_ TRIG_START field and TCR[3:0]: RX_FIFO_TRIG_HALT field.

- ☐ Trigger levels from 0 to 60 bytes are available with a granularity of four. (Trigger level = 4 x [4-bit register value])
- ☐ The programmer must ensure that TCR[3:0] > TCR[7:4] whenever auto-RTS is enabled to avoid improper operation of the device.
- ☐ In FIFO interrupt mode with flow control, the programmer must also ensure that the trigger level to HALT transmission exceeds or is equal to the receive FIFO trigger level (either TLR[7:4] or FCR[7:6]); otherwise, the FIFO operation stalls. In FIFO DMA mode with flow control, this concept does not exist because DMA request is sent each time a byte is received.

Figure 17−26 shows the receive and transmit operations.

Figure 17−26. 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 have been transmitted to go below the TX FIFO threshold.

17.4.2.3 FIFO Polled Mode Operation

In FIFO polled mode (FCR [0] = 0, relevant interrupts disabled using the IER), the status of the receiver and transmitter can be checked by polling the LSR. This mode is an alternative to the FIFO interrupt mode of operation where the status of the receiver and transmitter is automatically known by means of interrupts sent to the LH.

17.4.2.4 FIFO DMA Mode Operation

There are four modes of DMA operation: DMA mode 0/1/2/3. Table 17−11, Table 17−12, Table 17−21, and Table 17−22 assume 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, while in mode 3 the remaining DMA is used for TX. In mode 2 and mode 3, the DMA requests are P_DMA_48, P_DMA_50, and P_DMA_52/D_DMA_16. DMA requests P_DMA_49, P_DMA_51, and P_DMA_53/D_DMA_17 are not used by the module in mode 2 and mode 3. The four DMA modes can be selected as follows.

When SCR[0] = 0, setting FCR[3] to 0 enables DMA mode 0. Setting FCR[3] to 1 enables DMA mode 1.

When SCR[0] = 1, SCR[2:1] determines DMA mode 0 to mode 3 according to the SCR description.

For instance:

- ❑ If no DMA operation is desired: Set SCR[0] to 1 and SCR[2:1] to 00 (FCR[3] is discarded).

- If DMA mode 1 is desired: Set either SCR[0] to 0 and FCR[3] to 1, or set SCR[0] to 1 and SCR[2:1] to 01 (FCR[3] is discarded).

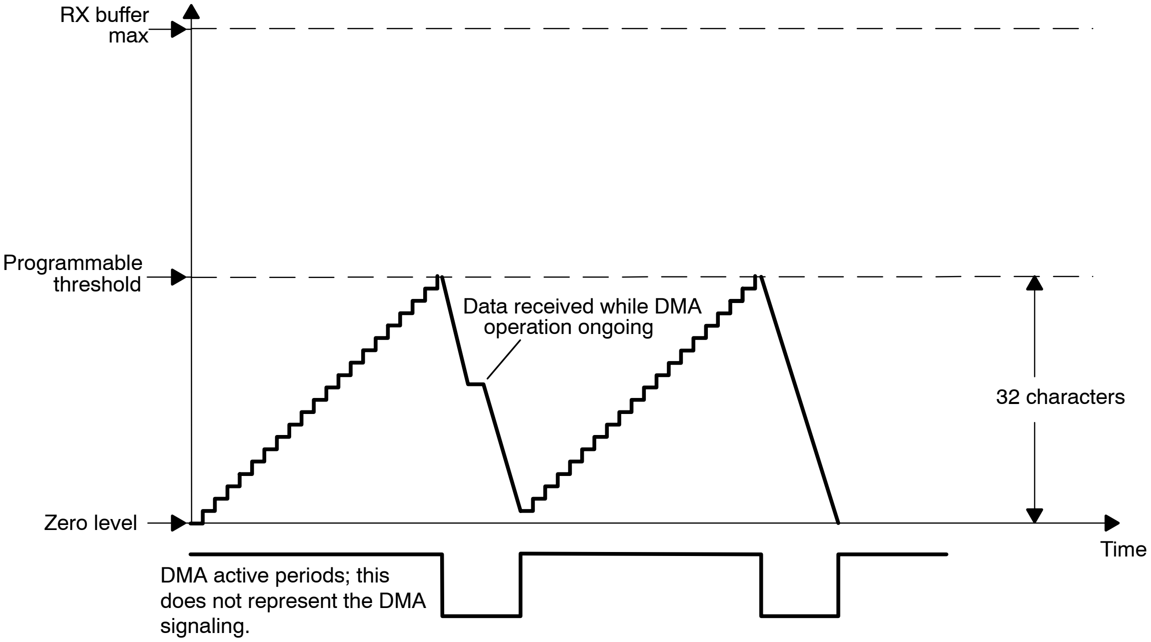
If the FIFOs are disabled (FCR[0] = 0), DMA occurs in single-character transfers.

When DMA mode 0 is programmed, the signals associated with DMA operation are not active.

DMA Transfers (DMA Mode 1, 2, or 3)

Figure 17−27 through Figure 17−30 show the supported DMA operations.

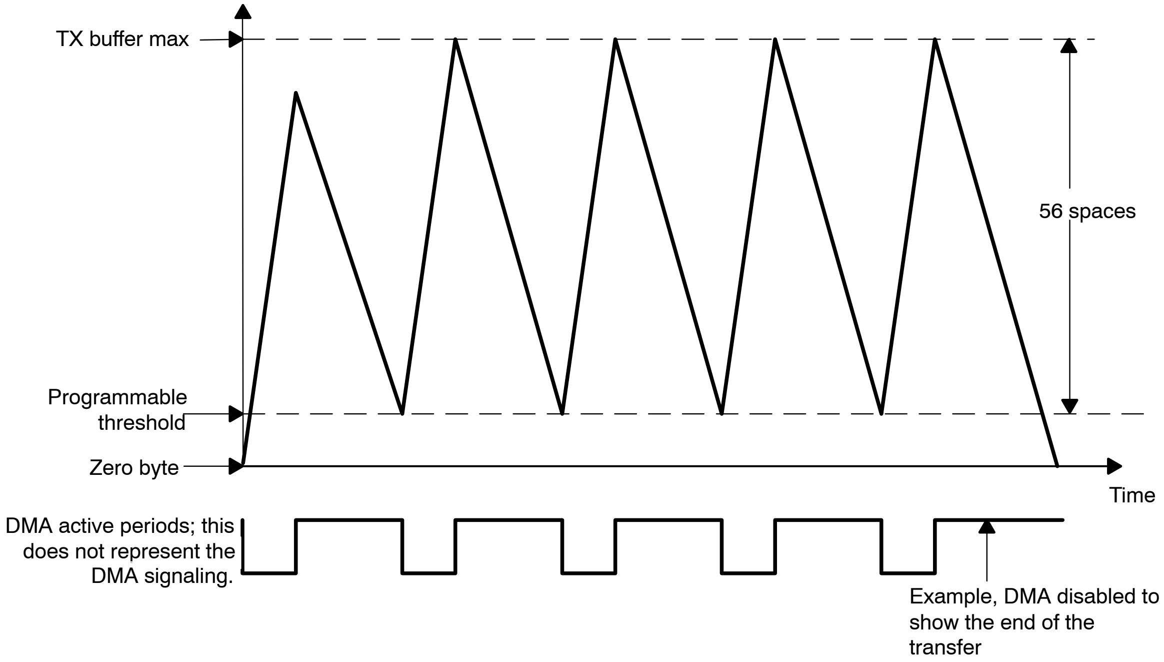
Figure 17−27. Receive FIFO DMA Request Generation (32 Characters)


RX buffer max
Programmable threshold
Zero level
Data received while DMA operation ongoing
32 characters
Time
DMA active periods; this does not represent the DMA signaling.

In receive mode, a DMA request is generated as soon as the receive FIFO reaches its threshold level defined in the TLR. 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 TLR is written by the sDMA. If an insufficient number of characters is written, the DMA request remains active.

Figure 17−28. Transmit FIFO DMA Request Generation (56 Spaces)

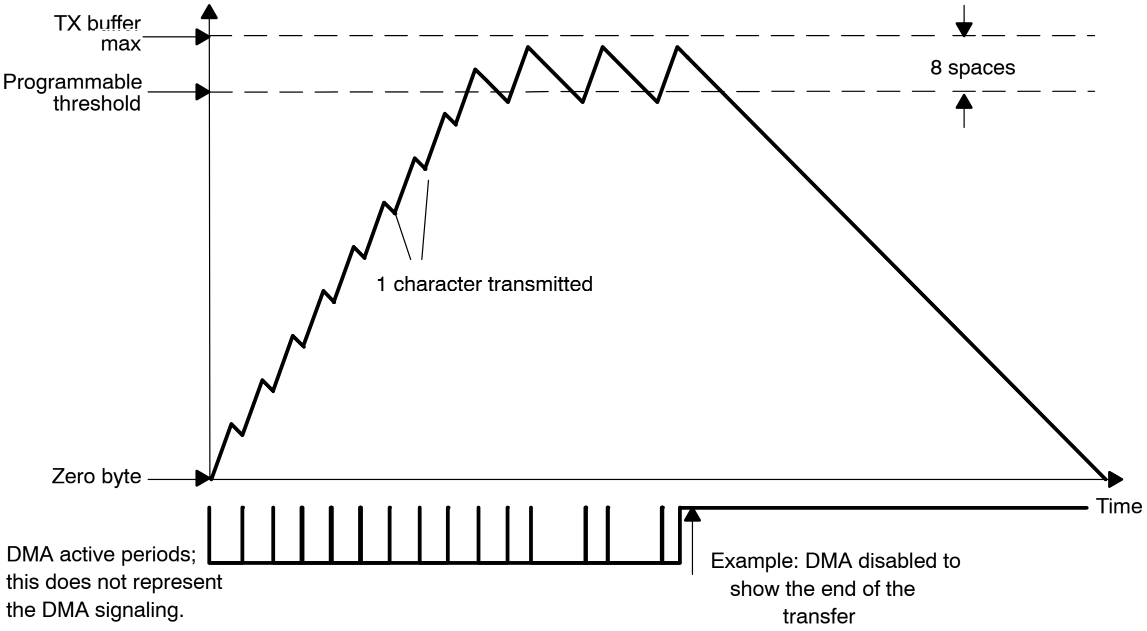

TX buffer max
Programmable threshold
Zero byte
Time
56 spaces
DMA active periods; this does not represent the DMA signaling.
Example, DMA disabled to show the end of the transfer

The DMA request is again asserted if the FIFO is able to receive the number of bytes defined by the TLR.

The threshold can be programmed in a number of ways. Figure 17−28 shows a DMA transfer that operates with a space setting of 56 that could arise from the use of the auto settings in the FCR[5:4] bit field or the use of the TLR[3:0] bit field concatenated with the FCR[5:4] bit field. The setting of 56 spaces in the UART/IrDA/CIR module must correlate with settings of the sDMA so that the buffer does not overflow (program the DMA request size of the LH controller to equal the number of spaces value in the UART/IrDA/CIR module).

Figure 17−29 shows another example with eight spaces to illustrate the buffer level crossing the space threshold. Again, the LH DMA controller settings must correspond with that of the UART/IrDA/CIR module.

Figure 17−29. Transmit FIFO DMA Request Generation (8 Spaces)

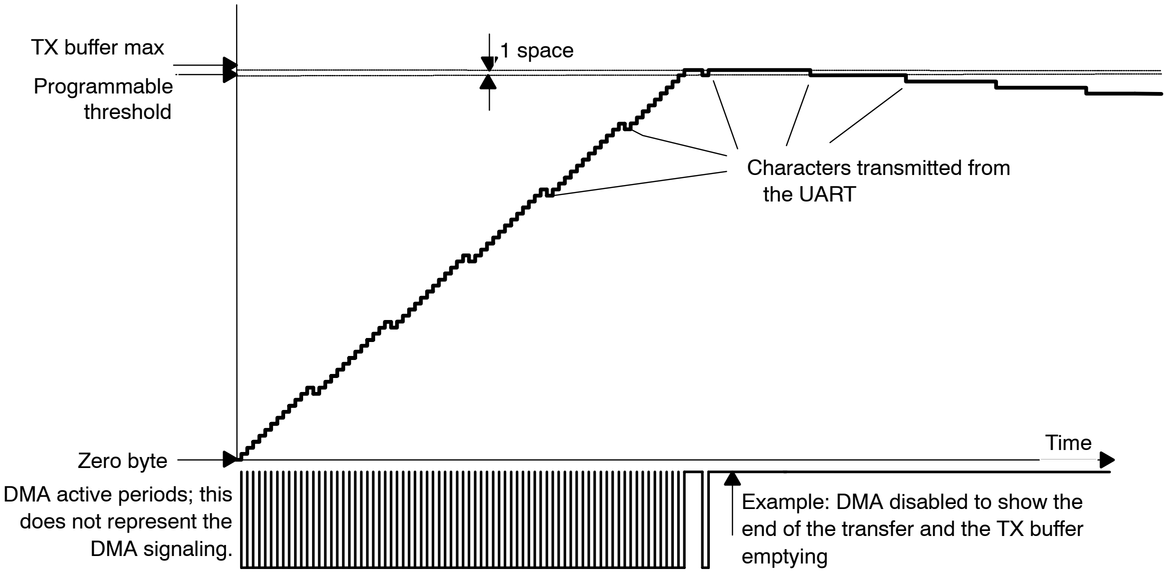

TX buffer max
Programmable threshold
8 spaces
1 character transmitted
Zero byte
Time
DMA active periods; this does not represent the DMA signaling.
Example: DMA disabled to show the end of the transfer

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 17−30). The buffer is filled at a faster rate than the baud rate transmits data to the TX pin. Eventually, the buffer is full and the DMA operations stop transferring data to the transmit buffer.

Shortly after the buffer holds the maximum number of data words, the DMA is disabled to facilitate 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 DLL and DLH registers.

Again, the DMA settings must correspond to the system LH DMA controller settings to ensure correct operation of this logic.

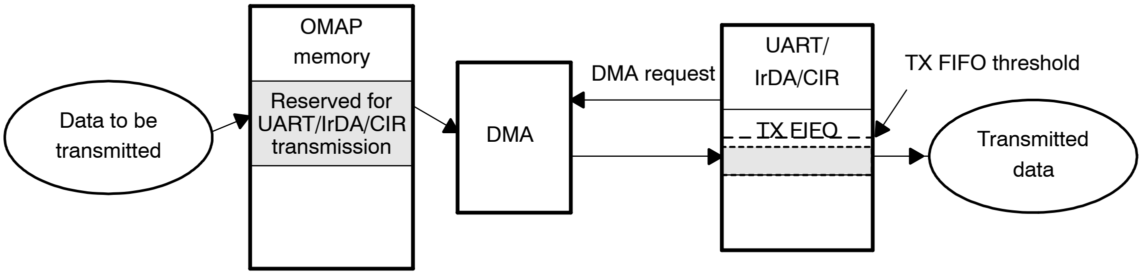
Figure 17−30. Transmit FIFO DMA Request Generation (1 Space)



DMA Transmission

Figure 17−31 shows the IrDA transmission process.

Figure 17−31. IrDA Transmission Process



The IrDA transmission process is as follows:

1) Data to be transmitted is put in the OMAP2420 memory reserved for UART/IrDA/CIR transmission by the DMA:
   a) Until the TX FIFO trigger level is reached, a DMA request is generated.
   b) An element (1 byte) is transferred from the SDRAM to the TX FIFO at each DMA request (DMA element synchronization).
2) Data in the TX FIFO is automatically transmitted.
3) The end of the transmission is signaled by the transfer holding registerTHR empty (TX FIFO empty).

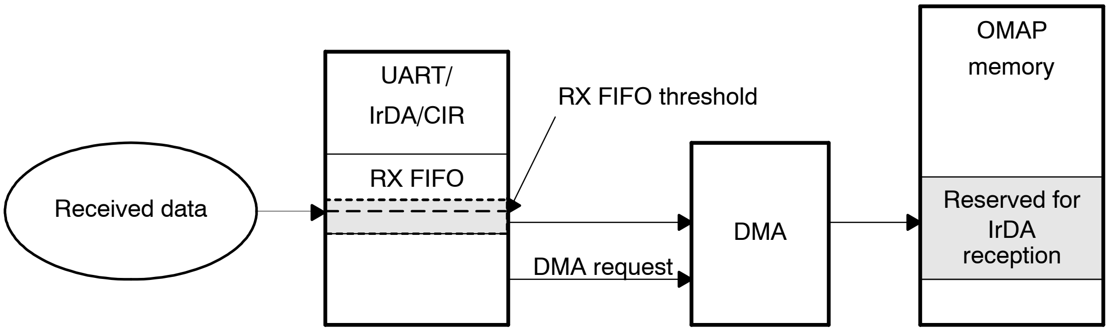
Note:

The transmission is not ended immediately after the TX FIFO empties, at which point there are still 4 bytes to be transmitted: 1 data byte, 2 CRC bytes, and 1 EOF byte. The end of transmission is, in fact, a few milliseconds after the THR empties.

DMA Reception

This section discusses the DMA reception. Figure 17−32 shows the IrDA reception process.

Figure 17−32. IrDA Reception Process



The IrDA reception process is as follows:

1) Enable the reception.

2) Received data are put in the RX FIFO.

3) Data is transferred from the RX FIFO to the OMAP2420 memory by DMA.

    a) At each received byte, the RX FIFO trigger level (one character) is reached and a DMA request is generated.

    b) An element (1 byte) is transferred from the SDRAM to the TX FIFO at each DMA request (DMA element synchronization).

4) The end of the reception is signaled by the EOF interrupt.

17.4.3 Mode Selection

This section discusses mode selection.

17.4.3.1 UART/IrDA (SIR, MIR, FIR)/CIR Mode Selection

Mode is selected by setting the MDR1[2:0]: MODE_SELECT field. Table 17−29 lists the modes and their values.

Table 17−29. 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 17.4.3.2, *Register Access Modes*).

Registers Available for UART

Only registers listed in Table 17−30 are used for UART functionality.

CAUTION

UART registers are limited to 8-bit and 16-bit data access; 32-bit data access is not allowed and can corrupt register contents.

Table 17−30. UART Mode Register Overview

Address Offset	Registers					
	configuration_mode_A		configuration_mode_B		operational_mode	
	READ	WRITE	READ	WRITE	READ	WRITE
0x000	DLL	DLL	DLL	DLL	RHR	THR
0x004	DLH	DLH	DLH	DLH	IER(UART)	IER(UART)
0x008	IIR	FCR	EFR[4]	EFR[4]	IIR(UART)	FCR(UART)
0x00C	LCR	LCR	LCR	LCR	LCR	LCR
0x010	MCR	MCR	XON1	XON1	MCR	MCR
0x014	LSR(UART)	–	XON2	XON2	LSR(UART)	–
0x018	MSR/TCR	TCR	XOFF1/TCR	XOFF1/TCR	MSR/TCR	TCR

Table 17−30. UART Mode Register Overview (Continued)

Address Offset	**Registers**					
	configuration_mode_A		**configuration_mode_B**		**operational_mode**	
	READ	**WRITE**	**READ**	**WRITE**	**READ**	**WRITE**
0x01C	TLR/SPR	TLR/SPR	TLR/XOFF2	TLR/XOFF2	TLR/SPR	TLR/SPR
0x020	MDR1	MDR1[2:0]	MDR1[2:0]	MDR1[2:0]	MDR1[2:0]	MDR1[2:0]
0x024	MDR2	MDR2	MDR2	MDR2	MDR2	MDR2
0x028	–	–	–	–	–	–
0x02C	–	–	–	–	–	–
0x030	–	–	–	–	–	–
0x034	–	–	–	–	–	–
0x038	UASR	–	UASR	–	–	–
0x03C	–	–	–	–	–	–
0x040	SCR	SCR	SCR	SCR	SCR	SCR
0x044	SSR	–	SSR	–	SSR	–
0x048	–	–	–	–	–	–
0x050	MVR	–	MVR	–	MVR	–
0x054	SYSC	SYSC	SYSC	SYSC	SYSC	SYSC
0x058	SYSS	–	SYSS	–	SYSS	–
0x05C	WER	WER	WER	WER	WER	WER
0x060	–	–	–	–	–	–

When REGISTER_NAME(UART) is applied in Table 17−30, the register exists for other functions (IrDA or CIR), but the fields have different meanings for other functions (described separately in Section 17.6, *UART/IrDA/CIR Registers*).

When REGISTER_NAME[m:n] is applied in Table 17−30, only register bits m to n apply to the UART function.

Register Available for IrDA (UART3 Only)

Only registers listed in Table 17−31 are used for UART functionality.

CAUTION

UART registers are limited to 8-bit and 16-bit data access; 32-bit access is not allowed and can corrupt register contents.

Table 17−31. IrDA Mode Register Overview

Address Offset	**Registers**					
	Configuration_Mode_A		**Configuration_Mode_B**		**Operational_Mode**	
	READ	**WRITE**	**READ**	**WRITE**	**READ**	**WRITE**
0x000	DLL	DLL	DLL	DLL	RHR	THR
0x004	DLH	DLH	DLH	DLH	IER(IrDA)	IER(IrDA)
0x008	IIR	FCR	EFR[4]	EFR[4]	IIR(IrDA)	FCR(IrDA)
0x00C	LCR[7]	LCR[7]	LCR[7]	LCR[7]	LCR[7]	LCR[7]
0x010	MCR	MCR	ADDR1	ADDR1	MCR	MCR
0x014	LSR(IrDA)	–	ADDR2	ADDR2	LSR(IrDA)	–
0x018	MSR/TCR	TCR	TCR	TCR	MSR/TCR	TCR
0x01C	TLR/SPR	TLR/SPR	TLR	TLR	TLR/SPR	TLR/SPR
0x020	MDR1	MDR1	MDR1	MDR1	MDR1	MDR1
0x024	MDR2	MDR2	MDR2	MDR2	MDR2	MDR2
0x028	SFLSR	TXFLL	SFLSR	TXFLL	SFLSR	TXFLL
0x02C	RESUME	TXFLH	RESUME	TXFLH	RESUME	TXFLH
0x030	SFREGL	RXFLL	SFREGL	RXFLL	SFREGL	RXFLL
0x034	SFREGH	RXFLH	SFREGH	RXFLH	SFREGH	RXFLH
0x038	–	–	–	–	BLR	BLR
0x03C	–	–	–	–	ACREG	ACREG
0x040	SCR	SCR	SCR	SCR	SCR	SCR
0x044	SSR	–	SSR	–	SSR	–
0x048	–	–	–	–	EBLR	EBLR
0x050	MVR	–	MVR	–	MVR	–
0x054	SYSC	SYSC	SYSC	SYSC	SYSC	SYSC
0x058	SYSS	–	SYSS	–	SYSS	–
0x05C	WER[6:4]	WER[6:4]	WER[6:4]	WER[6:4]	WER[6:4]	WER[6:4]
0x060	–	–	–	–	–	–

When REGISTER_NAME(IrDA) is applied in Table 17−31, the register exists for other functions (UART or CIR), but the fields have different meanings for other functions (described separately in Section 17.6, *UART/IrDA/CIR Registers*).

When REGISTER_NAME[m:n] is applied in Table 17−31, only register bits m to n apply to the IrDA function.

Registers Available for CIR Function (UART3 Only)

Only registers listed in Table 17−32 are used for UART functionality.

CAUTION

UART registers are limited to 8-bit and 16-bit data access; 32-bit data access is not allowed and can corrupt register contents.

Table 17−32. CIR Mode Register Overview

Address Offset	Registers					
	Configuration_Mode_A		Configuration_Mode_B		Operational_Mode	
	READ	WRITE	READ	WRITE	READ	WRITE
0x000	DLL	DLL	DLL	DLL	RHR	THR
0x004	DLH	DLH	DLH	DLH	IER(CIR)	IER(CIR)
0x008	IIR	FCR	EFR	EFR	IIR(CIR)	FCR(CIR)
0x00C	LCR	LCR[7]	LCR[7]	LCR[7]	LCR[7]	LCR[7]
0x010	MCR	MCR	–	–	MCR	MCR
0x014	LSR(IrDA)	–	–	–	LSR(IrDA)	–
0x018	MSR/TCR	TCR	TCR	TCR	MSR/TCR	TCR
0x01C	TLR/SPR	TLR/SPR	TLR	TLR	TLR/SPR	TLR/SPR
0x020	MDR1[3:0]	MDR1[3:0]	MDR1[3:0]	MDR1[3:0]	MDR1[3:0]	MDR1[3:0]
0x024	MDR2	MDR2	MDR2	MDR2	MDR2	MDR2
0x028	–	–	–	–	–	–
0x02C	RESUME	–	RESUME	–	RESUME	–
0x030	–	–	–	–	–	–
0x034	–	–	–	–	–	–
0x038	–	–	–	–	–	–
0x03C	–	–	–	–	ACREG	ACREG
0x040	SCR	SCR	SCR	SCR	SCR	SCR
0x044	SSR	–	SSR	–	SSR	–
0x048	–	–	–	–	EBLR	EBLR
0x050	MVR	–	MVR	–	MVR	–
0x054	SYSC	SYSC	SYSC	SYSC	SYSC	SYSC
0x058	SYSS	–	SYSS	–	SYSS	–
0x05C	WER[6:4]	WER[6:4]	WER[6:4]	WER[6:4]	WER[6:4]	WER[6:4]
0x060	CFPS	CFPS	CFPS	CFPS	CFPS	CFPS

When REGISTER_NAME(CIR) is applied in Table 17−32, the register exists for other functions (IrDA or UART), but the fields have different meanings for other functions (described separately in Section 17.6, *UART/IrDA/CIR Registers*).

When REGISTER_NAME[m:n] is applied in Table 17−32, only register bits m to n apply to the CIR function.

17.4.3.2 Register Access Modes

This section discusses the register access modes.

Operational Mode and Configuration Modes

Register access depends on the register access mode. These register access modes are not correlated with the functional mode selection. Three different modes are available: operational_mode, configuration_mode_A, and configuration_mode_B.

Operational_mode is selected when the function is active; serial data transfer can be performed in this mode.

Configuration_mode_A and configuration_mode_B are used during module initialization steps. These modes enable access to configuration registers, which are hidden in operational_mode. These modes are used when the module is inactive (no serial data transfer is processed) and only in the initialization step or reconfiguration of the module.

Register access mode is determined by the value of the LCR.

Table 17−33 provides the UART/IrDA/CIR register access mode programming using the LCR.

Table 17−33. UART/IrDA/CIR Register Access Mode Programming (Using LCR)

Mode Name	Condition
Configuration_Mode_A	LCR[7]=0x1 and LCR[7..0]!=0xBF
Configuration_Mode_B	LCR[7]=0x1 and LCR[7..0]=0xBF
Operational_Mode	LCR[7]=0x0

Table 17−34 lists the name of the register in each access register mode. The shaded registers are not dependent on the register access mode (available in all modes).

Table 17−34. 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	DLL	DLL	DLL	RHR	THR
0x004	DLH	DLH	DLH	DLH	IER	IER
0x008	IIR	FCR	EFR	EFR	IIR	FCR

Table 17−34. UART/IrDA/CIR Register Access Mode Overview (Continued)

Address Offset	Registers					
	Configuration_Mode_A		Configuration_Mode_B		Operational_Mode	
	READ	WRITE	READ	WRITE	READ	WRITE
0x00C	LCR	LCR	LCR	LCR	LCR	LCR
0x010	MCR	MCR	XON1_ADDR1	XON1_ADDR1	MCR	MCR
0x014	LSR	–	XON2_ADDR2	XON2_ADDR2	LSR	–
0x018	MSR/TCR	TCR	XOFF1/TCR	XOFF1/TCR	MSR/TCR	TCR
0x01C	SPR/TLR	SPR/TLR	XOFF2/TLR	XOFF2/TLR	SPR/TLR	SPR/TLR
0x020	MDR1	MDR1	MDR1	MDR1	MDR1	MDR1
0x024	MDR2	MDR2	MDR2	MDR2	MDR2	MDR2
0x028	SFLSR	TXFLL	SFLSR	TXFLL	SFLSR	TXFLL
0x02C	RESUME	TXFLH	RESUME	TXFLH	RESUME	TXFLH
0x030	SFREGL	RXFLL	SFREGL	RXFLL	SFREGL	RXFLL
0x034	SFREGH	RXFLH	SFREGH	RXFLH	SFREGH	RXFLH
0x038	UASR	–	UASR	–	BLR	BLR
0x03C	–	–	–	–	ACREG	ACREG
0x040	SCR	SCR	SCR	SCR	SCR	SCR
0x044	SSR	–	SSR	–	SSR	–
0x048	–	–	–	–	EBLR	EBLR
0x050	MVR	–	MVR	–	MVR	–
0x054	SYSC	SYSC	SYSC	SYSC	SYSC	SYSC
0x058	SYSS	–	SYSS	–	SYSS	–
0x05C	WER	WER	WER	WER	WER	WER
0x060	CFPS	CFPS	CFPS	CFPS	CFPS	CFPS

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. These registers are identified in Section 17.6, *UART/IrDA/CIR Registers*. The submodes are MSR_SPR, TCR_TLR, and XOFF.

Table 17−35 summarizes the sub-configuration_mode_A mode. Table 17−36 summarizes the sub-configuration_mode_B mode. Table 17−37 summarizes the sub-operational_mode mode.

Table 17−35. Sub-Configuration_Mode_A Mode Summary

Mode Name	Condition
MSR_SPR	(EFR[4] = 0x0 or MCR[6] = 0x0)
TCR_TLR	EFR[4] = 0x1 and MCR[6] = 0x1

Table 17−36. Sub-Configuration_Mode_B Mode Summary

Mode Name	Condition
TCR_TLR	EFR[4] = 0x1 and MCR[6] = 0x1
XOFF	(EFR[4] = 0x0 or MCR[6] = 0x0)

Table 17−37. Sub-Operational_Mode Mode Summary

Mode Name	Condition
MSR_SPR	(EFR[4] = 0x0 or MCR[6] = 0x0)
TCR_TLR	EFR[4] = 0x1 and MCR[6] = 0x1

17.4.4 Protocol Formatting

This section discusses protocol formatting.

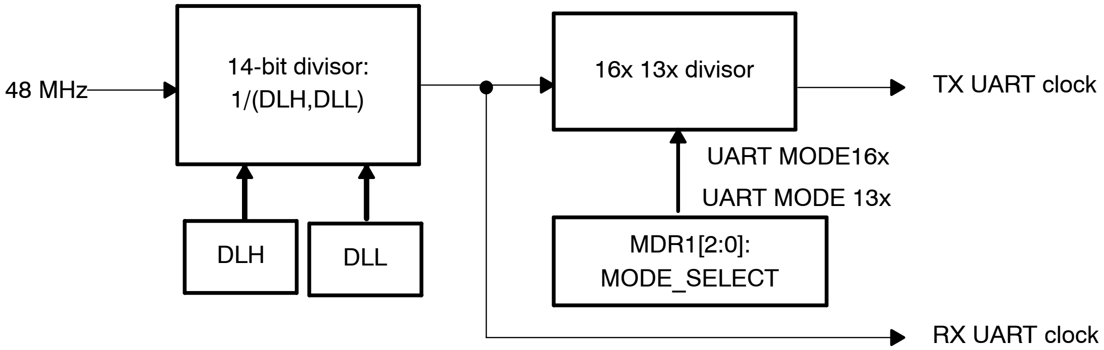
17.4.4.1 UART Mode

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 17−33 shows the baud rate generator and associated controls.

Figure 17−33. Baud Rate Generation



Before trying to initialize or modify clock parameter controls (DLH, DLL), set MODE_SELECT = DISABLE (MDR1[2:0] = 111). Failure to observe this recommendation can result in unpredictable behavior of the module.

CAUTION

Choosing the Appropriate Divisor Value

The divisor values for UART 16x mode and UART 13x mode are as follows:

- ❑ UART 16x mode: Divisor value = Operating Freq./(16x baud rate)

- ☐ UART 13x mode: Divisor value = Operating Freq./(13x baud rate)

Table 17−38 lists the UART baud rate settings for a 48-MHz clock.

Table 17−38. UART Baud Rate Settings (48-MHz Clock)

Baud Rate	Baud Multiple	DLH,DLL (Decimal)	DLH,DLL (hex)	Actual Baud Rate	Error (%)
0.3 Kbps	16 x	10000	0x27, 0x10	0.3 Kbps	0
0.6 Kbps	16 x	5000	0x13, 0x88	0.6 Kbps	0
1.2 Kbps	16 x	2500	0x09, 0xC4	1.2 Kbps	0
2.4 Kbps	16 x	1250	0x04, 0xE2	2.4 Kbps	0
4.8 Kbps	16 x	625	0x02, 0x71	4.8 Kbps	0
9.6 Kbps	16 x	312	0x01, 0x38	9.6153 Kbps	+0.16
14.4 Kbps	16x	208	0x00, 0xD0	14.423 Kbps	+0.16
19.2 Kbps	16 x	156	0x00, 0x9C	19.231 Kbps	+0.16
28.8 Kbps	16 x	104	0x00, 0x68	28.846 Kbps	+0.16
38.4 Kbps	16 x	78	0x00, 0x4E	38.462 Kbps	+0.16
57.6 Kbps	16 x	52	0x00, 0x34	57.692 Kbps	+0.16
115.2 Kbps	16 x	26	0x00, 0x1A	115.38 Kbps	+0.16
230.4 Kbps	16 x	13	0x00, 0x0D	230.77 Kbps	+0.16
460.8 Kbps	13 x	8	0x00, 0x08	461.54 Kbps	+0.16
921.6 Kbps	13 x	4	0x00, 0x04	923.08 Kbps	+0.16
1.843 Mbps	13 x	2	0x00, 0x02	1.846 Mbps	+0.16
3.6864 Mbps	13 x	1	0x00, 0x01	3.6923 Mbps	+0.16

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 using RTS output and CTS input signals to automatically control serial data flow.

The UART modem module is enhanced with the autobauding functionality. In control mode, autobauding automatically sets the speed, the number of bits per character, and the parity selected.

Frame Formatting

When autobauding functionality is not used, frame format attributes must be defined in the LCR.

Character length is specified with the LCR[1:0]: CHAR_LENGTH field.

The number of stop-bits is specified with the LCR[2]: NB_STOP bit.

Parity bit is programmed using the LCR[5:3] (PARITY_EN, PARITY_TYPE_1, and PARITY_TYPE_2) bits.

Table 17−39 lists the UART parity bit encoding.

Table 17−39. UART Parity Bit Encoding

LCR[3]	LCR[4]	LCR[5]	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

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 EFR[7:6].

With auto-CTS, nCTS must be active before the module can transmit data.

Auto-RTS activates the nRTS output only when there is enough room in the RX FIFO to receive data, and deactivates the nRTS output when the RX FIFO is sufficiently full. The HALT and RESTORE trigger levels in the TCR determine the levels at which nRTS 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 neither auto-CTS nor auto-RTS is 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 TCR. RTS is active if the RX FIFO level is below the HALT trigger level in TCR[3:0]. When the receiver FIFO HALT trigger level is reached, RTS is deasserted. The sending device (for example, another UART) may send an additional byte after the trigger level is reached because it may not recognize the deassertion of RTS until it has begun sending the additional byte. RTS is automatically reasserted once the receiver FIFO reaches the RESUME trigger level programmed by TCR[7:4]. This reassertion requests the sending device to resume transmission.

In this case, RTS is an active-low signal.

Auto-CTS

The transmitter circuitry checks CTS before sending the next data byte. When CTS is active, the transmitter sends the next byte. To stop the transmitter from sending the following byte, CTS must be deasserted before the middle of the last stop-bit that is currently being sent. The auto-CTS function reduces interrupts to the host system. When auto-CTS flow control is enabled, the CTS state changes need not 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, CTS is an active-low signal.

Software Flow Control

Software flow control is enabled through the enhanced feature register (EFR) and the modem control register (MCR). Different combinations of software flow control can be enabled by setting different combinations of EFR[3:0].

Two other enhanced features relate to software flow control:

- ☐ XON-any function (MCR[5]): Operation resumes after receiving any character following 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 (EFR[5]): Incoming data is compared to XOFF2. Detection of the special character sets the XOFF interrupt (IIR[4]) but does not halt transmission. The XOFF interrupt is cleared by a read of the interrrupt identification register (IIR). The special character is transferred to the RX FIFO.

Table 17−40. EFR[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 (See note.)
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 (See note.)

Note: 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 XON1_ADDR1[7:0] XON_WORD1 field. XON2 is defined in the XON2_ADDR2[7:0] XON_WORD2 field.

XOFF1 is defined in the XOFF1[7:0] XOFF_WORD1 field. XOFF2 is defined in the XOFF2[7:0] XOFF_WORD2 field.

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 is halted when transmission of the current character completes. XOFF detection also sets IIR[4] (if enabled by IER[5]) and causes the interrupt line to go low.

To resume transmission, a XON1/2 character must be received (in certain cases, XON1 and XON2 must be received sequentially). When the correct XON characters are received, IIR[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, EFR[1:0] = 0x2, if XON1 and XOFF1 characters are received, they are not written to the RX FIFO.

In the case where pairs of software flow characters are programmed to be received sequentially (EFR[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.

Transmit (TX)

XOFF1: Two characters are transmitted when the RX FIFO passes the trigger level programmed by TCR[3:0].

XON1: Two characters are transmitted when the RX FIFO reaches the trigger level programmed by TCR[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/XON 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 be 5, 6, or 7 characters, the 5, 6, or 7 least-significant bits (LSBs) of XOFF1/2 and XON1/2 are transmitted. The 5, 6, or 7 bits of a character are seldom transmitted, but this functionality is included to maintain compatibility with earlier designs.

It is assumed that software flow control and hardware flow control will never be enabled simultaneously.

Autobauding Modes

In autobaud mode, UART can extract transfer characteristics (speed, length, and parity) from an AT (“at”) command (ASCII code). These characteristics are used to receive data following an AT command and to send data.

Here are valid AT commands:

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. A/ and a/ are used to instruct the software to repeat the last received AT command; therefore, an a/ always comes after an AT, and transfer characteristics are not expected to change between an AT and an a/.

As soon as a valid AT is received, AT and all subsequent data are saved into RX FIFO, including the final <CR> (0x0D); the autobaud state-machine waits for next valid AT command. If an a/ (A/) is received, the a/ (A/) is saved into RX FIFO and the state-machine waits for 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 has been detected. The UART autobauding status register (UASR) reflects the correct settings for the baud rate that has been detected. The interrupt activity can continue in this fashion each time a subsequent character is received. Therefore, it is recommended that the software enable the RHR interrupt when using autobaud mode.

The following settings are detected in autobaud mode with a module clock of 48 MHz:

- Speed:
  - 115.2 kbaud
  - 57.6 kbaud
  - 38.4 kbaud
  - 28.8 kbaud
  - 19.2 kbaud
  - 14.4 kbaud
  - 9.6 kbaud
  - 4.8 kbaud
  - 2.4 kbaud
  - 1.2 kbaud
- Length: 7 or 8 bits
- Parity: Odd, even, or space

The combination of 7-bit character + space parity is not supported.

Autobauding mode is selected when MDR1[2:0] = 010. In the UART autobauding mode, DLL, DLH, LCR[5:0] settings are not used; instead, UASR is updated with the configuration detected by the autobauding logic.

UASR Autobauding Status Register Use

When UART is in autobaud mode, this register is used instead of the LCR, DLL, and DLH registers to set up transmission according to the characteristics of the previous reception.

To reset the autobauding hardware (to start a new AT detection) or to set the UART in standard mode (no autobaud), MDR1[2:0] must be set to reset state 0x7. After that, the UART can be set in autobaud mode (MDR1[2:0] = 0x2) or in standard mode (MDR1[2:0] = 0x0).

Usage limitation:

- Only 7- and 8-bit characters (5- and 6-bit not supported)
- 7-bit character with space parity not supported
- Baud rate between 1200 bps and 115,200 bps (10 possibilities)

Error Detection

When the LSR is read, LSR[4:2] reflects the error bits (break condition [BI], framing error [FE], and parity error [PE]) of the character at the top of the RX FIFO (next character to be read). Therefore, reading the LSR and then reading the RHR identifies errors in a character.

Reading RHR updates BI, FE, and PE (see Table 17−41 for the UART mode interrupts).

LSR[7] is set when there is an error anywhere in the RX FIFO and is cleared only when no more errors remain in the RX FIFO.

Note:

Reading the LSR does not cause an increment of the RX FIFO read pointer. The RX FIFO read pointer is incremented by reading the RHR.

Reading the LSR clears [OE] if set (see Table 17−41 for the UART mode interrupts).

Overrun During Receive

Overrun occurs during receive 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 LH with IIR[3] and discards the remaining portion of the frame. Overrun also sets an internal flag, which disables further reception. Before the next frame can be received the system (LH) must perform the following actions:

1) Reset the RX FIFO.
2) Read the RESUME register (this clears the internal flag).

Break and Time-Out Conditions

Time-Out Counter

An RX idle condition is detected when the receiver (RX) line has been high for a time equivalent to 4X programmed word length+12 bits. The receiver line is sampled midway through each bit.

For sleep mode, the counter is reset when there is activity on the RX line.

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 the RX line or when the RHR is read.

Break Condition

When a break condition occurs, the transmitter (TX) line is pulled low. A break condition is activated by setting LCR[6] to 1. Be aware that 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 to perform the following procedure:

1) Reset the transmit FIFO (if enabled).
2) Wait for the transmit shift register to empty (LSR[6] = 1).
3) Take a guard time according to the stop-bit definition.
4) Set LCR[6] to 1.

The preceding functionality (time-out counter and break condition) applies only to the UART modem operation and does not extend to the IrDA/CIR modes of operation.

17.4.4.2 UART Mode Interrupt Management

In UART modes, there are seven possible interrupts. These interrupts are prioritized to six different levels.

When an interrupt is generated, the IIR indicates that an interrupt is pending by bringing IIR[0] to 0 and provides the type of interrupt through IIR[5−1]. Table 17−41 summarizes the interrupt control functions.

Table 17−41. UART Mode Interrupts

IIR[5−0]	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Method
0 0 0 0 0 1	None	None	None	None
0 0 0 1 1 0	1	Receiver line status	OE, FE, PE, or BI errors occur in characters in the RX FIFO.	FE, PE, BI: Read RHR. OE: Read LSR.
0 0 1 1 0 0	2	RX time-out	Stale data in RX FIFO	Read RHR.
0 0 0 1 0 0	2	RHR interrupt	DRDY (data ready) (FIFO disable) RX FIFO above trigger level (FIFO enable)	Read RHR until interrupt condition disappears.

IIR[5–0]	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Method
0 0 0 0 1 0	3	THR interrupt	TFE (THR empty) (FIFO disable) TX FIFO below trigger level (FIFO enable)	Write to THR until interrupt condition disappears.
0 0 0 0 0 0	4	Modem status	MSR[1:0] /= 0	Read MSR.
0 1 0 0 0 0	5	XOFF interrupt/ special character interrupt	Receive XOFF characters(s)/special character	Receive XON character(s), if XOFF interrupt/read of IIR, if special character interrupt.
1 0 0 0 0 0	6	CTS, RTS	RTS pin or CTS pin change state from active (low) to inactive (high)	Read IIR.

Note:

RX_FIFO_STS bit (LSR[7]) generates the interrupt for the RX line status interrupt.

For an 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 IIR.

Wake-Up Interrupt

The wake-up interrupt is a special interrupt that is not designed in the same way as previous interrupts. This interrupt is enabled when the RX_CTS_WU_EN bit of SCR[4] is set to 1. The IIR is not modified when it occurs; SSR[1] must be checked to detect a wake-up event. When a wake-up interrupt occurs, the only way to clear it is to reset SCR[4] to 0.

17.4.4.3 UART Power Management

Module Power Saving

In UART modes, sleep mode is enabled by writing 1 to IER[4] (when EFR[4] = 1).

Sleep mode is entered when all of the following conditions exist:

- ☐ The serial data input line, RX, is idle.
- ☐ The TX FIFO and TX shift register are empty.
- ☐ The RX FIFO is empty.
- ☐ Only THR interrupts are pending.

Sleep mode is a good way to reduce the power consumption of 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, power consumption is greatly reduced. The module wakes up when any change is detected on the RX line, if data is written to the TX FIFO, when there is any change in the state of the modem input pins. An interrupt can be generated on a wake-up event by setting SCR[4] to 1. To know how to manage it, see Section 17.4.4.2, *UART Mode Interrupt Management, Wake-up Interrupt*.

Note:

There must be no writing to the divisor latches (DLL and DLH) to set the baud clock (BCLK) while in sleep mode. Disable sleep mode using IER[4] before writing to DLL or DLH.

System Power Saving

Sleep and autoidle 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 the interrupt generation mode to a wake-up generation mode for unmasked events (see SYSC[2] and the wake-up enable register [WER]).

See Chapter 5, *Power, Reset, and Clock Management*, for details.

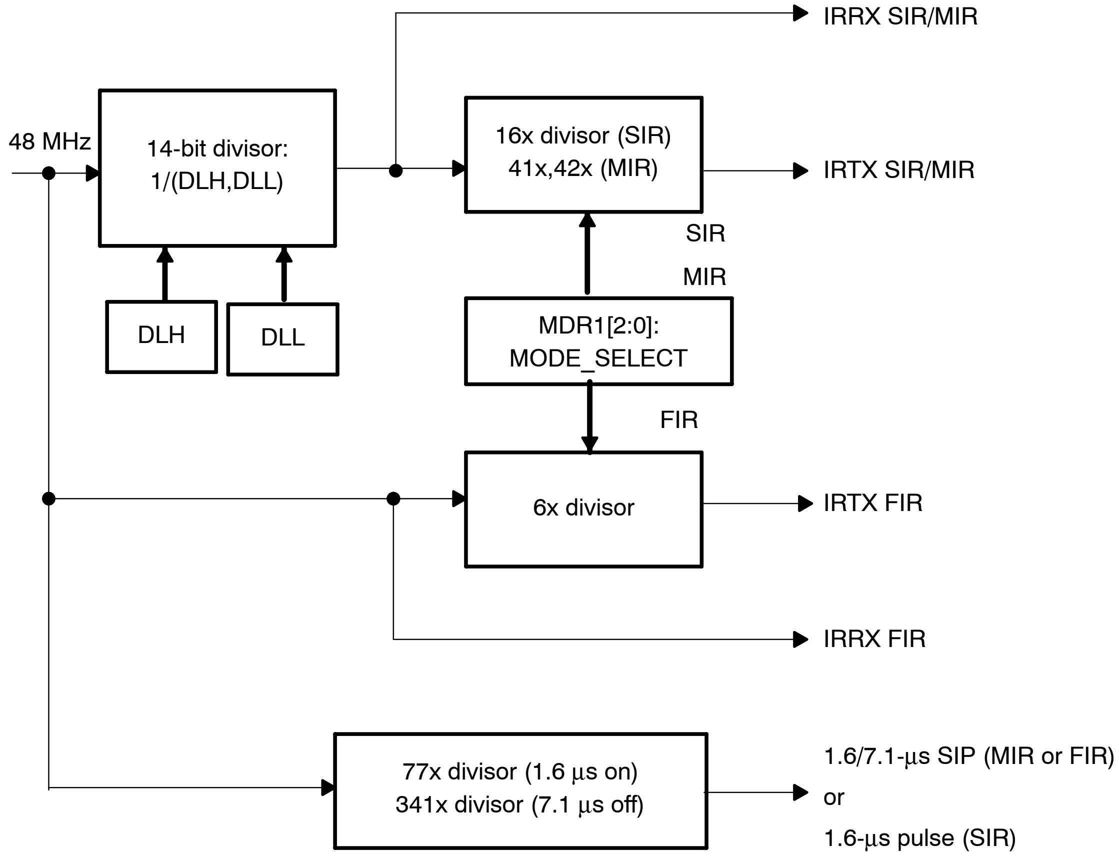
17.4.4.4 IrDA Mode (UART3 Only)

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 17–34 shows the baud rate generator and associated controls.

Figure 17–34. Baud Rate Generator



CAUTION

Before trying to initialize or modify clock parameters controls (DLH, DLL), set MODE_SELECT = DISABLE (MDR1[2:0] = 111). Failure to observe this rule can result in an unpredictable behavior of the module.

Choosing the Appropriate Divisor Value

The following are the divisor values for the SIR, MIR, and FIR modes:

- SIR mode: Divisor value = Operating Freq./(16x baud rate)
- MIR mode: Divisor value = Operating Freq./(41x/42x baud rate)
- FIR mode: Divisor value = none

IrDA Data Formatting

The methods described in this section apply to all IrDA modes: SIR, MIR, FIR.

IRRX Polarity Control

The MDR2[6] IRRXINVERT bit provides the flexibility to invert the RX pin inside the UART module to ensure that the protocol at the input of the transceiver module has the same polarity at module level. By default, the IRRX pin is inverted because most transceivers invert the IR receive pin.

IrDA Reception Control

Data can be transferred both ways by the module, but when the device is transmitting the IRRX circuitry is automatically disabled by hardware.

The operation of the IRRX input can be disabled with DIS_IR_RX bit of ACREG[5]. Furthermore, the MDR2[6] bit can be used to invert the signal from the transceiver (RX output) pin to the IRRX logic inside the UART.

IR Address Checking

In all IR modes, if address-checking is enabled, 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. It is possible to program two frame addresses that the UART/IrDA receives with the XON1/ADDR1 and XON2/ADDR2 registers.

Selecting address1 checking is done by setting EFR[0] to 1. Address2 checking is done by setting EFR[1] to 1. Setting EFR[1:0] 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 are written into the reception FIFO.

Frame Closing

There are two ways by which a transmission frame can be properly terminated:

- Frame-length method: The frame-length method is selected when MDR1[7] = 0. The LH writes the frame-length value to the transmit frame length high (TXFLH) and transmit frame length low (TXFLL) registers. The device automatically attaches end flags to the frame once the number of bytes transmitted is equal to the frame-length value.
- Set-EOT bit method: The set-EOT bit method is selected when MDR1[7] = 1. The LH writes 1 to ACREG[0] (EOT bit) just before it writes the last byte to the TX FIFO. When the LH writes the last byte to the TX FIFO, the device internally sets the tag bit for that particular 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 properly terminate the frame.

Store and Control Transmission

In SCT, the LH first starts writing data into the TX FIFO. Then, after it writes a part of a frame (for a bigger frame) or a whole frame (a small frame—that is, a supervisory frame), it writes 1 to ACREG[2] (deferred TX start) to start transmission. SCT is enabled when MDR1[5] = 1. This method of transmission is different from the normal mode, where transmission of data starts immediately after data is written to the TX FIFO. SCT is useful to send short frames without TX underrun.

Error Detection

When the LSR is read, LSR[4:2] reflect the error bits (FL, CRC, ABORT) of the frame at the top of the STATUS FIFO (next frame status to be read).

Errors are triggered by interrupts (see Table 17−42 for IrDA mode interrupts). STATUS FIFO must be read until empty (maximum of eight reads required).

Underrun During Transmission

Underrun in transmission occurs when the TX FIFO becomes 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 sets an internal flag, which disables further transmission. Before the next frame can be transmitted, the system (LH) must perform the following actions:

1) Reset the TX FIFO.
2) Read the RESUME register (this clears the internal flag).

This functionality can be disabled with ACREG[4], compensated by the extension of the stop-bit in transmission in case the TX FIFO is empty.

Overrun During Receive

Overrun occurs during receive 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 LH with IIR[3] and discards the remaining portion of the frame. Overrun also sets an internal flag, which disables further reception. Before the next frame can be received, the system (LH) must perform the following actions:

1) Reset the RX FIFO.
2) Read the RESUME register (this clears the internal flag).

Status FIFO

In IrDA modes, a status FIFO is used to record 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 into the status FIFO.

The frame length and error status can be read by reading SFREGL/SFREGH and SFLSR. Reading SFLSR causes the read pointer to be incremented. The status FIFO is eight entries deep; therefore, it can hold the status of eight frames.

The LH uses the frame-length information to locate the frame boundary in the received frame data. The LH can screen bad frames using the error status information and later request the sender to resend only the bad frames.

This status FIFO can be used effectively in DMA because the LH is interrupted only when the programmed status FIFO trigger level is reached, not every time a frame is received.

SIR Mode Data Formatting

This section gives specific instructions for SIR mode programming.

Abort Sequence

The transmitter may decide to prematurely close a frame. The transmitter aborts by sending the sequence 0x7DC1. The abort pattern closes the frame without a CRC field or an end flag.

It is possible to abort a transmission frame by programming the ABORT_EN bit of ACREG[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, are received without transparency.

CAUTION

If transmit FIFO is not empty and MDR1[5] = 1, UART/IrDA starts a new transfer with data of the previous frame as soon as an abort frame is sent. Therefore, TX FIFO must be reset before sending an abort frame.

Pulse Shaping

In SIR mode, both the 3/16th or the 1.6-µs pulse duration methods are supported. ACREG[7] selects the PWM in transmit mode.

SIR Free Format Programming

The SIR free format (FF) mode is selected by setting the module in UART mode (MDR1[2:0] = 000) and the MDR2[3] bit to 1 to allow the pulse shaping.

Because the bit format is to remain the same, some UART mode configuration registers must be set at specific values:

- ❑ LCR[1:0] = 11 (8 data bits)
- ❑ LCR[2] = 0 (1 stop-bit)
- ❑ LCR[3] = 0 (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.).

MIR and FIR Mode Data Formatting

This section gives common instructions for FIR and MIR mode programming.

At the end of a frame reception, the LH reads the LSR to determine possible errors in the received frame.

When the SIP_MODE bit of MDR1 equals 1 (MDR1[6]), the TX state-machine always sends one SIP at the end of a transmission frame. But when MDR1[6] = 0, the transmission of the SIP depends on the SEND_SIP bit of ACREG[3]. The system (LH) can set ACREG[3] at least once every 500 ms. The advantage of this approach over the default approach is that the TX state-machine does not need to send the SIP at the end of each frame, which can reduce the overhead required.

IrDA Mode Interrupt Management

IrDA Interrupts

The IrDA function generates interrupts. All interrupts can be enabled/disabled by writing to the appropriate bit in IER. The interrupt status of the device can be checked at any time by reading the IIR.

The UART, IrDA, and CIR modes have different interrupts in the UART/IrDA/CIR module and, therefore, different IER and IIR mappings according to the selected mode.

In the IrDA modes, there are eight possible interrupts. The interrupt line is activated when any of the eight interrupts is generated (there is no priority). For IIR[5], interrupt source 1 is used with interrupt reset method 1, and interrupt source 2 is used with interrupt reset method 2.

Table 17−42 lists the IrDA mode interrupts.

Table 17−42. IrDA Mode Interrupts

IIR 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 until interrupt condition disappears.
1	THR interrupt	TFE (THR empty) (FIFO disable) TX FIFO below trigger level (FIFO enable)	Write to THR 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.
3	RX overrun	Write to RHR when RX FIFO full.	Read RESUME register.
4	Status FIFO interrupt	Status FIFO triggers level reached.	Read STATUS FIFO.
5	TX Status	THR empty before EOF sent. Last bit of transmission of the IrDA frame has occurred but with an underrun error. OR Transmission of the last bit of the IrDA frame is finished successfully.	Read RESUME register. OR Read IIR.
6	Receiver line status interrupt	CRC, ABORT, or frame-length error is written into STATUS FIFO.	Read STATUS FIFO. (read until empty–maximum of 8 reads required)
7	Received EOF	Received end-of-frame	Read IIR.

Wake-Up Interrupt

The wake-up interrupt is a special interrupt that is not designed in the same way as the previous interrupts. This interrupt is enabled when the RX_CTS_WU_EN bit of SCR[4] is set to 1. The IIR is not modified when it occurs; SSR[1] must be checked to detect a wake-up event. When a wake-up interrupt occurs, the only way to clear it is to reset SCR[4] to 0.

17.4.4.5 IrDA Mode Power Management (UART3 Only)

Module Power Saving

In IrDA modes, sleep mode is enabled by writing 1 to MDR1[3].

Sleep mode is entered when all the following conditions exist:

- The serial data input line, IRRX is idle.
- The TX FIFO and TX shift register are empty.
- The RX FIFO is empty.
- Only THR interrupts are pending.

The module wakes up when any change is detected on the IRRX line or if data is written to the TX FIFO.

System Power Saving

Sleep and autoidle 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 the interrupt generation mode to a wake-up generation mode for unmasked events (see SYSC[2] and WER).

See Chapter 5, *Power, Reset, and Clock Management*, for details.

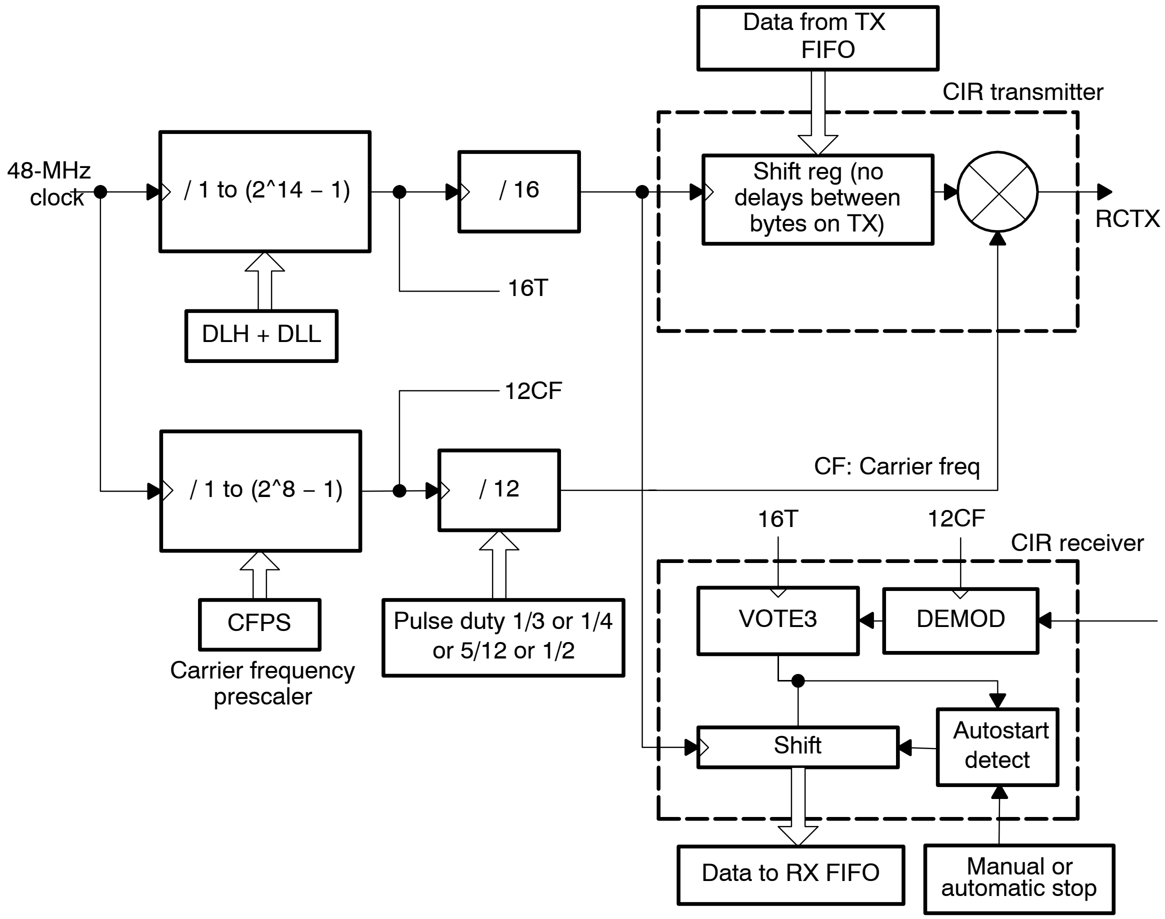
17.4.4.6 CIR Mode (UART3 Only)

This section discusses CIR mode clock generation, data formatting, and interrupt generation.

CIR Mode Clock Generation

Depending on the encoding method (variable pulse distance/biphase), the LH must develop a data structure that combines the 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 method of modulation shown in Figure 17−35.

Figure 17−35. CIR Mode Block Components



Based on the requested modulation frequency, the CFPS 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 value must be set to 111 (decimal), which provides a modulation frequency of 36.04 kHz.

Note:

The CFPS 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 using the pulse duty register bits in the MDR2 configuration register. Table 17−43 shows the duty cycle.

Table 17−43. Duty Cycle

MDR2[5:4]	Duty Cycle (High Level)
00	1/4
01	1/3
10	5/12
11	1/2

CIR Data Formatting

This section describes IRRX polarity control and CIR transmission and reception.

IRRX Polarity Control

The MDR2[6] IRRXINVERT bit provides the flexibility to invert the RX pin inside the UART module to ensure that the protocol at the input of the transceiver module has the same polarity at module level. By default, the IRRX pin is inverted because most transceivers invert the IR receive pin.

CIR Transmission

In transmission, the LH software must exercise an element of real-time control to transmit data packets; they must each be emitted at a constant delay from the start-bits of each individual packet. This means that when sending a series of packets, the packet-to-packet delay must respect a specific delay. To control this delay, two methods can be used:

- Filling the TX FIFO with a number of zero bits, which are transmitted with a t period
- Using an external system timer, which controls 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 through the configuration register MDR1[5] and ACREG[2] depending on the timer status
    - Using the TX_STATUS interrupt IIR[5] 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).

CIR Reception

There are two ways to stop reception:

- The LH can disable the reception by setting the ACREG[5] bit to 1 when it considers the reception to be finished, because a large number of 0s have been received. To receive a new frame, the ACREG[5] bit must be set to 0.
- Texas Instruments recommends using the specific mechanism for stopping reception. The reception is automatically stopped, depending on the

value set in the BOF length register (EBLR). If the value set in EBLR is different than 0, this feature is enabled and counts the number of bits received at 0. When the counter achieves the value defined in EBLR, the reception is automatically stopped and RX_STOP_IT (IIR[2]) is set. When a 1 is detected on the IRRX pin, the reception is automatically enabled.

CAUTION

If the RX_STOP interrupt occurs before a byte boundary, the remaining bits of the last byte are filled with zeros and then passed into the RX FIFO.

CIR Mode Interrupt Management

CIR Interrupts

The CIR function generates interrupts. All interrupts can be enabled/disabled by writing to the appropriate bit in IER. The interrupt status of the device can be checked at any time by reading the IIR.

The UART, IrDA, and CIR modes have different interrupts in the UART/IrDA/CIR module and, therefore, different IER and IIR mappings according to the selected mode.

Table 17−44 lists the interrupt modes that are to be maintained. In CIR mode, the IIR[5] bit has the sole purpose of indicating that the last bit of infrared data has been passed to the IR TX pin.

Table 17−44. CIR Mode Interrupts

IIR Bit No.	Interrupt Type	Interrupt Source	Interrupt Reset Method
0	RHR interrupt	DRDY (data ready) (FIFO disable) RX FIFO above trigger level (FIFO enable)	Read RHR until interrupt condition disappears.
1	THR interrupt	TFE (THR empty) (FIFO disable) TX FIFO below trigger level (FIFO enable)	Write to THR until interrupt condition disappears.
2	RX_STOP_IT	Receive stop interrupt (depending on value set in EBLR).	Read IIR.
3	RX overrun	Write to RHR when RX FIFO full.	Read RESUME register.
4	N/A for CIR	N/A for CIR	N/A for CIR
5	TX status	Transmission of the last bit of the frame is finished successfully.	Read IIR.
6	N/A for CIR	N/A for CIR	N/A for CIR
7	N/A for CIR	N/A for CIR	N/A for CIR

Wake-Up Interrupt

The wake-up interrupt is a special interrupt that does not operate in the same way as the previous interrupts. This interrupt is enabled when the RX_CTS_WU_EN bit of SCR[4] is set to 1. The IIR is not modified when it occurs; SSR[1] must be checked to detect a wake-up event. When wake-up interrupt occurs, the only way to clear it is to reset SCR[4] to 0.

17.4.4.7 CIR Mode Power Management (UART3 Only)

This section discusses power saving in CIR mode.

Module Power Saving

In CIR modes, sleep mode is enabled by writing 1 to MDR1[3].

Sleep mode is entered when all the following conditions exist:

- ☐ The serial data input line, IRRX is idle.
- ☐ The TX FIFO and TX shift register are empty.
- ☐ The RX FIFO is empty.
- ☐ Only THR interrupts are pending.

The module wakes up when any change is detected on the IRRX line or if data is written to the TX FIFO.

System Power Saving

Sleep and autoidle 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 the interrupt generation mode to a wake-up generation mode for unmasked events (see SYSC[2] and WER).

See Chapter 5, *Power, Reset, and Clock Management*, for details.

17.5 UART/IrDA/CIR Programming Model

17.5.1 UART Configuration Example

This section outlines the programming stages to operate one UART module with FIFO, interrupt, and no DMA capabilities. This 3-step procedure ensures quick start of these modules (obviously, it does not cover every UART module feature). The first step covers software reset of the module (interrupts, status, and controls). The second step deals with FIFO configuration and enable. The last step deals with the baud rate data and stop configuration. The following procedure is programming language agnostic.

17.5.2 UART Software Reset

Goal:

To clear both IER and MCR, remove UART breaks (LCR[6] = 0), and put module in reset (MDR1[2:0] = 0x07).

Procedure:

To write into both IER and MCR, EFR[4] must first be set to 1.

To access EFR, 0xBF must first be written to the LCR. Hence:

LCR = 0xBF;	First, write to the LCR.
EFR[4] = 1;	When LCR = 0xBF, enable EFR.
LCR[7] = 0;	Here, access to IER and MCR is allowed.
IER = 0x00;	Disable interrupt.
MCR = 0x00;	Force control signals inactive.
LCR[6] = 0;	Here, UART breaks are removed.
MDR1 = 0x07;	Here, UART is in reset or disabled.

Alternatiely, the SYSC[1] can be set to 1 to instigate a hardware reset from the generic synchronous reset module. The reset progress can be monitored using SYSS[0]. Once complete, the preceding sequence must ensure the UART is in the equivalent disabled mode with reference to MDR1[2:0].

17.5.3 UART FIFO Configuration

Goal:

To set trigger level for halt/restore (TCR), set the trigger level for transmit/receive (TLR) and configure FIFO (FCR).

Procedure:

To write into both the TLR and TCR, EFR[4] and MCR[6] must be set to 1. To write into the FCR, EFR[4] must be set to 1. EFR[4] = 1 was already done in the previous section; hence, a simple write to MCR[6] is necessary.

MCR[6] = 1;

Set TCR, TLR, and FCR to the desired value.

Here accesses to TCR, TLR, and FCR must be disabled to avoid any further undesired write to these registers. Hence:

LCR = 0xBF; Provides access to EFR

EFR[4] = 0;

LCR[7] = 0;

MCR[6] = 0;

17.5.4 Baud Rate Data and Stop Configuration

Goal:

To configure UART data and stop (LCR) baud rate (DLH and DLL registers) and enable UART operation. In case interrupt capability is added, configuration must be added right before UART enable.

Procedure:

Set LCR to desired value.

LCR[7] to 1; Gives access to the DLH and DLL registers

Set DLH and DLL.

LCR[7]=0; Removes access to the DLH and DLL registers

Set IER to desired value; Sets interrupts

MDR1[2:0]=0; Enables UART without autobauding

The UART module is operational.

17.6 UART/IrDA/CIR Registers

17.6.1 UART/IrDA/CIR Register Instance Summary

Table 17−45 shows the base address and address space for the UART/IrDA/CIR module instances.

Table 17−45. 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)	0x4806 E000	1K byte

17.6.2 UART/IrDA/CIR Register Mapping Summary

This section provides information on the UART/IrDA/CIR module instance within the OMAP2420 device. Each register within the module instance is described separately. This module has multiple modes. Table 17−46 provides an overview of the UART mode.

Table 17−47 through Table 17−59 summarize the registers for configuration_mode_A, configuration_mode_B, and operational_mode.

CAUTION

UART registers are limited to 8-bit and 16-bit data access; 32-bit data access is not allowed and can corrupt register contents.

Table 17−46. UART Mode Overview

Address Offset	Registers					
	Configuration_Mode_A		Configuration_Mode_B		Operational_Mode	
	READ	WRITE	READ	WRITE	READ	WRITE
0x000	DLL	DLL	DLL	DLL	RHR	THR
0x004	DLH	DLH	DLH	DLH	IER	IER
0x008	IIR	FCR	EFR	EFR	IIR	FCR
0x00C	LCR	LCR	LCR	LCR	LCR	LCR
0x010	MCR	MCR	XON1_ADDR1	XON1_ADDR1	MCR	MCR
0x014	LSR	–	XON2_ADDR2	XON2_ADDR2	LSR	–
0x018	MSR/TCR	TCR	XOFF1/TCR	XOFF1/TCR	MSR/TCR	TCR
0x01C	TLR/SPR	TLR/SPR	TLR/XOFF2	TLR/XOFF2	TLR/SPR	TLR/SPR

Table 17−46. UART Mode Overview (Continued)

Address Offset	Registers					
	Configuration_Mode_A		Configuration_Mode_B		Operational_Mode	
	READ	WRITE	READ	WRITE	READ	WRITE
0x020	MDR1	MDR1	MDR1	MDR1	MDR1	MDR1
0x024	MDR2	MDR2	MDR2	MDR2	MDR2	MDR2
0x028	SFLSR	TXFLL	SFLSR	TXFLL	SFLSR	TXFLL
0x02C	RESUME	TXFLH	RESUME	TXFLH	RESUME	TXFLH
0x030	SFREGL	RXFLL	SFREGL	RXFLL	SFREGL	RXFLL
0x034	SFREGH	RXFLH	SFREGH	RXFLH	SFREGH	RXFLH
0x038	UASR	–	UASR	–	BLR	BLR
0x03C	–	–	–	–	ACREG	ACREG
0x040	SCR	SCR	SCR	SCR	SCR	SCR
0x044	SSR	–	SSR	–	SSR	–
0x048	–	–	–	–	EBLR	EBLR
0x050	MVR	–	MVR	–	MVR	–
0x054	SYSC	SYSC	SYSC	SYSC	SYSC	SYSC
0x058	SYSS	–	SYSS	–	SYSS	–
0x05C	WER	WER	WER	WER	WER	WER
0x060	CFPS	CFPS	CFPS	CFPS	CFPS	CFPS

Table 17−47. UART1/2/3 Mode Summary

Mode Name	Condition
Configuration_Mode_A	LCR[7]=0x1 and LCR[7..0]!=0xBF
Configuration_Mode_B	LCR[7]=0x1 and LCR[7..0]=0xBF
Operational_Mode	LCR[7]=0x0

Table 17−48. UART1/2/3 Register Summary for Configuration_Mode_A Mode Active

Register Name	Type	Register Width (Bits)	Physical Address
DLL	RW	8	0x0000 0000
DLH	RW	8	0x0000 0004
IIR	R	8	0x0000 0008
FCR	W	8	0x0000 0008
LCR	RW	8	0x0000 000C
MCR	RW	8	0x0000 0010
LSR	R	8	0x0000 0014

Condition: LCR[7] = 0x1 and LCR[7..0] = 0xBF

Table 17−48. UART1/2/3 Register Summary for Configuration_Mode_A Mode Active (Continued)

Register Name	Type	Register Width (Bits)	Physical Address
See Table 17−49 and either Table 17−50 or Table 17−51.			0x0000 0018
See Table 17−49 and either Table 17−50 or Table 17−51.			0x0000 001C
MDR1	RW	8	0x0000 0020
MDR2	RW	8	0x0000 0024
SFLSR	R	8	0x0000 0028
TXFLL	W	8	0x0000 0028
RESUME	R	8	0x0000 002C
TXFLH	W	8	0x0000 002C
RXFLL	W	8	0x0000 0030
SFREGL	R	8	0x0000 0030
RXFLH	W	8	0x0000 0034
SFREGH	R	8	0x0000 0034
UASR	R	8	0x0000 0038
SCR	RW	8	0x0000 0040
SSR	R	8	0x0000 0044
MVR	R	8	0x0000 0050
SYSC	RW	8	0x0000 0054
SYSS	R	8	0x0000 0058
WER	RW	8	0x0000 005C
CFPS	RW	8	0x0000 0060

Condition: LCR[7] = 0x1 and LCR[7..0] = 0xBF

Table 17−49. UART1/2/3 Sub-Configuration_Mode_A Mode Summary

Mode Name	Condition
MSR_SPR	(EFR[4] = 0x0 or MCR[6] = 0x0)
TCR_TLR	EFR[4] = 0x1 and MCR[6] = 0x1

Table 17−50. UART1/2/3 Register Summary for Sub-Configuration_Mode_A Mode: MSR_SPR Mode Active

Register Name	Type	Register Width (Bits)	Physical Address
MSR	R	8	0x0000 0018
SPR	RW	8	0x0000 001C

Condition: (EFR[4] = 0x0 or MCR[6] = 0x0)

Table 17−51. UART1/2/3 Register Summary for Sub-Configuration_Mode_A Mode: TCR_TLR Mode Active

Register Name	Type	Register Width (Bits)	Physical Address
TCR	RW	8	0x0000 0018
TLR	RW	8	0x0000 001C

Condition: EFR[4] = 0x1 and MCR[6] = 0x1

Table 17−52. UART1/2/3 Register Summary for Configuration_Mode_B Mode Active

Register Name	Type	Register Width (Bits)	Physical Address
DLL	RW	8	0x0000 0000
DLH	RW	8	0x0000 0004
EFR	RW	8	0x0000 0008
LCR	RW	8	0x0000 000C
XON1_ADDR1	RW	8	0x0000 0010
XON2_ADDR2	RW	8	0x0000 0014
See Table 17−53 and either Table 17−54 or Table 17−55.			0x0000 0018
See Table 17−53 and either Table 17−54 or Table 17−55.			0x0000 001C
MDR1	RW	8	0x0000 0020
MDR2	RW	8	0x0000 0024
SFLSR	R	8	0x0000 0028
TXFLL	W	8	0x0000 0028
RESUME	R	8	0x0000 002C
TXFLH	W	8	0x0000 002C
RXFLL	W	8	0x0000 0030
SFREGL	R	8	0x0000 0030
RXFLH	W	8	0x0000 0034
SFREGH	R	8	0x0000 0034
UASR	R	8	0x0000 0038
SCR	RW	8	0x0000 0040
SSR	R	8	0x0000 0044
MVR	R	8	0x0000 0050
SYSC	RW	8	0x0000 0054
SYSS	R	8	0x0000 0058
WER	RW	8	0x0000 005C
CFPS	RW	8	0x0000 0060

Condition: LCR[7] = 0x1 and LCR[7..0] = 0xBF

Table 17−53. UART1/2/3 Sub-Configuration_Mode_B Mode Summary

Mode Name	Condition
TCR_TLR	EFR[4] = 0x1 and MCR[6] = 0x1
XOFF	(EFR[4] = 0x0 or MCR[6] = 0x0)

Table 17−54. UART1/2/3 Register Summary for Sub-Configuration_Mode_B Mode: TCR_TLR Mode Active

Register Name	Type	Register Width (Bits)	Physical Address
TCR	RW	8	0x0000 0018
TLR	RW	8	0x0000 001C

Condition: EFR[4] = 0x1 and MCR[6] = 0x1

Table 17−55. UART1/2/3 Register Summary for Sub-Configuration_Mode_B Mode: XOFF Mode Active

Register Name	Type	Register Width (Bits)	Physical Address
XOFF1	RW	8	0x0000 0018
XOFF2	RW	8	0x0000 001C

Condition: (EFR[4] = 0x0 or MCR[6] = 0x0)

Table 17−56. UART1/2/3 Register Summary for Operational_Mode Mode Active

Register Name	Type	Register Width (Bits)	Physical Address
RHR	R	8	0x0000 0000
THR	W	8	0x0000 0000
IER	RW	8	0x0000 0004
IIR	R	8	0x0000 0008
FCR	W	8	0x0000 0008
LCR	RW	8	0x0000 000C
MCR	RW	8	0x0000 0010
LSR	R	8	0x0000 0014
See Table 17−57 and either Table 17−58 or Table 17−59.			0x0000 0018
See Table 17−57 and either Table 17−58 or Table 17−59.			0x0000 001C
MDR1	RW	8	0x0000 0020
MDR2	RW	8	0x0000 0024
SFLSR	R	8	0x0000 0028
TXFLL	W	8	0x0000 0028
RESUME	R	8	0x0000 002C
TXFLH	W	8	0x0000 002C
RXFLL	W	8	0x0000 0030

Condition: LCR[7] = 0x0

Table 17−56. UART1/2/3 Register Summary for Operational_Mode Mode Active (Continued)

Register Name	Type	Register Width (Bits)	Physical Address
SFREGL	R	8	0x0000 0030
RXFLH	W	8	0x0000 0034
SFREGH	R	8	0x0000 0034
BLR	RW	8	0x0000 0038
ACREG	RW	8	0x0000 003C
SCR	RW	8	0x0000 0040
SSR	R	8	0x0000 0044
EBLR	RW	8	0x0000 0048
MVR	R	8	0x0000 0050
SYSC	RW	8	0x0000 0054
SYSS	R	8	0x0000 0058
WER	RW	8	0x0000 005C
CFPS	RW	8	0x0000 0060

Condition: LCR[7] = 0x0

Table 17−57. UART1/2/3 Sub-Operational_Mode Mode Summary

Mode Name	Condition
MSR_SPR	(EFR[4] = 0x0 or MCR[6] = 0x0)
TCR_TLR	EFR[4] = 0x1 and MCR[6] = 0x1

Table 17−58. UART1/2/3 Register Summary for Sub-Operational_Mode Mode: MSR_SPR Mode Active

Register Name	Type	Register Width (Bits)	Physical Address
MSR	R	8	0x0000 0018
SPR	RW	8	0x0000 001C

Condition: (EFR[4]=0x0 or MCR[6]=0x0)

Table 17−59. UART1/2/3 Register Summary for Sub-Operational_Mode Mode: TCR_TLR Mode Active

Register Name	Type	Register Width (Bits)	Physical Address
TCR	RW	8	0x0000 0018
TLR	RW	8	0x0000 001C

Condition: EFR[4]=0x1 and MCR[6]=0x1

17.6.3 UART/IrDA/CIR Register Descriptions

Table 17−60 through Table 17−113 describe the individual register bits.

Table 17−60. DLL

Address Offset	0x000		
Physical Address	0x4806 A000	**Instance**	UART1
	0x4806 C000		UART2
	0x4806 E000		UART3
Description	Divisor latches low This register, combined with DLH, stores the 14-bit divisor for generation of the baud clock in the baud rate generator. DLH stores the most significant part of the divisor. DLL stores the least significant part of the divisor. Note: DLL and DLH can only be written to before sleep mode is enabled (that is, before IER[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	Reads return 0s.	R	0x00
7:0	CLOCK_LSB	Used to store the 8-bit LSB divisor value	RW	0x00

Table 17−61. RHR

Address Offset	0x000		
Physical Address	0x4806 A000	**Instance**	UART1
	0x4806 C000		UART2
	0x4806 E000		UART3
Description	Receive holding register The receiver section consists of the receiver holding register (RHR) and the receiver shift register. The RHR 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. 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 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	Reads return 0s.	R	0x00
7:0	RHR	Receive holding register	R	0x−−

Table 17−62. THR

Address Offset	0x000		
Physical Address	0x4806 A000	**Instance**	UART1
	0x4806 C000		UART2
	0x4806 E000		UART3
Description	Transmit holding register The transmitter section consists of the transmit holding register (THR) and the transmit shift register. The transmit holding register is actually a 64-byte FIFO. The LH writes data to the THR. The data is placed into 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	Writes have no effect.	W	0x00
7:0	THR	Transmit holding register	W	0x−−

Table 17−63. IER

Address Offset	0x004		
Physical Address	0x4806 A004	**Instance**	UART1
	0x4806 C004		UART2
	0x4806 E004		UART3
Description	Interrupt enable register		
Type	RW		

Table 17−64. UART Bit Field 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

Table 17−64. UART Bit Field Details (Continued)

Bits	Field Name	Description		Type	Reset
15:8	Reserved	Reads return 0s.		RW	0x00
7	CTS_IT	Can only be written when EFR[4] = 1		RW	0
		0x0:	Disables the nCTS interrupt		
		0x1:	Enables the nCTS interrupt		
6	RTS_IT	Can only be written when EFR[4] = 1		RW	0
		0x0:	Disables the interrupt		
		0x1:	Enables the nRTS interrupt		
5	XOFF_IT	Can only be written when EFR[4] = 1		RW	0
		0x0:	Disables the XOFF interrupt		
		0x1:	Enables the XOFF interrupt		
4	SLEEP_MODE	Can only be written when EFR[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.		

Table 17−65. CIR Bit Field Details

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								RESERVED		TX_STATUS_IT	RESERVED	RX_OVERRUN_IT	RX_STOP_IT	THR_IT	RHR_IT

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reads return 0s.	RW	0x00
7:6	Reserved	Reads return 0s.	RW	0x0
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[0] = 0.	RW	0
4	Reserved	Reads return 0s.	RW	0
3	RX_OVER-RUN_IT	0x0: Disables the RX overrun interrupt 0x1: Enables the RX overrun interrupt	RW	0
2	RX_STOP_IT	RX_STOP_IT interrupt is generated based on the value set in the BOF length register (EBLR). 0x0: Disables the receive stop interrupt 0x1: Enables the receive stop interrupt	RW	0
1	THR_IT	0x0: Disables the THR interrupt 0x1: Enables the THR interrupt	RW	0
0	RHR_IT	0x0: Disables the RHR interrupt and time-out interrupt. 0x1: Enables the RHR interrupt and time-out interrupt.	RW	0

Table 17−66. IrDA Bit Field 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

Table 17−66. IrDA Bit Field Details (Continued)

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reads return 0s.	RW	0x00
7	EOF_IT	0x0: Disables the received EOF interrupt 0x1: Enables the received EOF interrupt	RW	0
6	LINE_STS_IT_I	0x0: Disables the receiver line status interrupt 0x1: Enables the receiver line status interrupt	RW	0
5	TX_STATUS_IT	TX_STATUS_IT interrupt reflects two possible conditions. The MDR2[0] must be read to determine the status in the event of this interrupt. 0x0: Disables the TX status interrupt 0x1: Enables the TX status interrupt	RW	0
4	STS_FIFO_TRIG_IT	0x0: Disables status FIFO trigger level interrupt 0x1: Enables status FIFO trigger level interrupt.	RW	0
3	RX_OVERRUN_IT	0x0: Disables the RX overrun interrupt 0x1: Enables the RX overrun interrupt	RW	0
2	LAST_RX_BYTE_IT	0x0: Disables the last byte of frame in RX FIFO interrupt 0x1: Enables the last byte of frame in RX FIFO interrupt	RW	0
1	THR_IT	0x0: Disables the THR interrupt 0x1: Enables the THR interrupt	RW	0
0	RHR_IT	0x0: Disables the RHR interrupt and time-out interrupt. 0x1: Enables the RHR interrupt and time-out interrupt.	RW	0

Table 17−67. DLH

Address Offset	0x004		
Physical Address	0x4806 A004	**Instance**	UART1
	0x4806 C004		UART2
	0x4806 E004		UART3
Description	Divisor latches high These two registers store the 14-bit divisor for generation of the baud clock in the baud rate generator. DLH stores the most significant part of the divisor. DLL stores the least significant part of the divisor. Note: DLL and DLH can only be written to before sleep mode is enabled (that is, before IER[4] is set).		
Type	RW		

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">Reserved</td><td colspan="2">Reserved</td><td colspan="6">CLOCK_MSB</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reads return 0s.	R	0x00
7:6	Reserved	Reads return 0s.	R	0x0
5:0	CLOCK_MSB	Used to store the 6-bit MSB divisor value	RW	0x00

Table 17−68. FCR

Address Offset	0x008		
Physical Address	0x4806 A008	**Instance**	UART1
	0x4806 C008		UART2
	0x4806 E008		UART3
Description	FIFO control register		
Type	W		

Table 17−69. Normal Bit Field Details: EFR[4]=0

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">Reserved</td><td colspan="2">RX_FIFO_TRIG</td><td colspan="2">RESERVED</td><td>DMA_MODE</td><td>TX_FIFO_CLEAR</td><td>RX_FIFO_CLEAR</td><td>FIFO_EN</td></tr>
</table>

Table 17−69. Normal Bit Field Details: EFR[4]=0 (Continued)

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Writes no effect.	W	0x00
7:6	RX_FIFO_TRIG	Sets the trigger level for the RX FIFO:If SCR[7] = 0 and TLR[7:4] ≠ 0000: 00: 8 characters 01: 16 characters 10: 56 characters 11: 60 characters. RX_FIFO_TRIG is not considered. If SCR[7] = 1, RX_FIFO_TRIG is 2 LSB of the trigger level (1–63 on 6 bits) with the granularity 1.	W	0x0
5:4	Reserved	Writes no effect.	W	0x0
3	DMA_MODE	This register is considered if SCR[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 zero. Returns to zero after clearing FIFO.	W	0
1	RX_FIFO_CLEAR	0x0: No change 0x1: Clears the receive FIFO and resets its counter logic to zero. Returns to zero after clearing FIFO.	W	0
0	FIFO_EN	0x0: Disables the transmit and receive FIFOs. The transmit and receive holding registers are one byte FIFOs. 0x1: Enables the transmit and receive FIFOs. The transmit and receive holding registers are 64-byte FIFOs.	W	0

Table 17−70. Enhanced Bit Field Details: EFR[4]=1

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	Writes no effect.	W	0x00
7:6	RX_FIFO_TRIG	Sets the trigger level for the RX FIFO:If SCR[7] = 0 and TLR[7:4] = 0000: If SCR[7] = 0 and TLR[7:4] != 0000, RX_FIFO_TRIG is not considered. If SCR[7]=1, RX_FIFO_TRIG is 2 LSB of the trigger level (1−63 on 6 bits) with the granularity 1. 0x0: 8 characters 0x1: 16 characters 0x2: 56 characters 0x3: 60 characters	W	0x0
5:4	TX_FIFO_TRIG	Sets the trigger level for the TX FIFO:If SCR[6] = 0 and TLR[3:0] = 0000: TX_FIFO_TRIG can be written only if EFR[4]=1 If SCR[6] = 0 and TLR[3:0] != 0000, TX_FIFO_TRIG is not considered. If SCR[6]=1, TX_FIFO_TRIG is 2 LSB of the trigger level (1−63 on 6 bits) with the granularity 1. 0x0: 8 spaces 0x1: 16 spaces 0x2: 32 spaces 0x3: 56 spaces	W	0x0
3	DMA_MODE	Can be changed only when the baud clock is not running (DLL and DLH set to 0). This register is considered if SCR[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

Table 17−70. Enhanced Bit Field Details: EFR[4]=1 (Continued)

Bits	Field Name	Description	Type	Reset
2	TX_FIFO_CLEAR	Can be changed only when the baud clock is not running (DLL and DLH set to 0). 0x0: No change 0x1: Clears the transmit FIFO and resets its counter logic to zero. Returns to zero after clearing FIFO.	W	0
1	RX_FIFO_CLEAR	Can be changed only when the baud clock is not running (DLL and DLH set to 0). 0x0: No change 0x1: Clears the receive FIFO and resets its counter logic to zero. Returns to zero after clearing FIFO.	W	0
0	FIFO_EN	Can be changed only when the baud clock is not running (DLL and DLH set to 0). 0x0: Disables the transmit and receive FIFOs. The transmit and receive holding registers are one byte FIFOs. 0x1: Enables the transmit and receive FIFOs. The transmit and receive holding registers are 64-byte FIFOs.	W	0

Table 17−71. IIR

Address Offset	0x008		
Physical Address	0x4806 A008	**Instance**	UART1
	0x4806 C008		UART2
	0x4806 E008		UART3
Description	Interrupt identification register The IIR is a read-only register, which provides the source of the interrupt in a prioritized manner. Note: An interrupt source can be flagged only if enabled in the IER register.		
Type	R		

Table 17−72. UART Bit Field Details

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								FCR_MIRROR		IT_TYPE					IT_PENDING

Table 17−72. UART Bit Field Details (Continued)

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reads return 0s.	R	0x00
7:6	FCR_MIRROR	Mirror the contents of FCR[0] on both bits.	R	0x0
5:1	IT_TYPE	Seven possible interrupt listed below in UART mode; other combination never occurred	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		R	1
		0x0: An interrupt is pending.		
		0x1: No interrupt is pending.		

Table 17−73. CIR Bit Field Details

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								RESERVED		TX_STATUS_IT	RESERVED	RX_OE_IT	RX_STOP_IT	THR_IT	RHR_IT

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reads return 0s.	R	0x00
7:6	Reserved	Reads return 0s.	R	0x0
5	TX_STATUS_IT		R	0
		0x0: TX status interrupt inactive		
		0x1: TX status interrupt active		
4	Reserved	Reads return 0s.	R	0
3	RX_OE_IT		R	0
		0x0: RX overrun interrupt inactive		
		0x1: RX overrun interrupt active		

Table 17−73. CIR Bit Field Details (Continued)

Bits	Field Name	Description		Type	Reset
2	RX_STOP_IT			R	0
		0x0:	Receive stop interrupt is inactive		
		0x1:	Receive stop interrupt is 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		

Table 17−74. IrDA Bit Field 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	Reads return 0s.		R	0x00
7	EOF_IT			R	0
		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		

Table 17−74. IrDA Bit Field Details (Continued)

Bits	Field Name	Description		Type	Reset
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		

Table 17−75. EFR

Address Offset	0x008		
Physical Address	0x4806 A008	**Instance**	UART1
	0x4806 C008		UART2
	0x4806 E008		UART3
Description	Enhanced feature register. This register enables or disables enhanced features. Most of the enhanced functions only apply to UART modes, but EFR[4] enables write accesses to FCR[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	Reserved			

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reads return 0s.	R	0x00
7	AUTO_CTS_EN	Auto-CTS enable bit (UART mode only) 0x0: Normal operation 0x1: Auto-CTS flow control is enabled, i.e., 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, i.e., nRTS pin goes high (inactive) when the receiver FIFO HALT trigger level, TCR[3:0], is reached and goes low (active) when the receiver FIFO RESTORE transmission trigger level is reached.	RW	0

Table 17−75. EFR (Continued)

Bits	Field Name	Description	Type	Reset
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 bit 4 is set to 1 to indicate that a special character has been detected.	RW	0
4	ENHANCED_EN	Enhanced functions write enable bit 0x0: Disables writing to IER bits 4−7, FCR bits 4−5, and MCR bits 5−7 0x1: Enables writing to IER bits 4−7, FCR bits 4−5, and MCR bits 5−7	RW	0
3:2	RESERVED	These bits must be kept at 0.	RW	0x0
1	ADDR2CHECK	IR address checking for Address2 (IrDA mode only) 0x0: No check on IR Address 2 0x1: Enable IR Address checking for Address1	RW	0
0	ADDR1CHECK	IR address checking for Address1 (IrDA mode only) 0x0: No check on IR Address 1 0x1: Enable IR Address checking for Address2	RW	0

Table 17−76. LCR

Address Offset	0x00C		
Physical Address	0x4806 A00C	**Instance**	UART1
	0x4806 C00C		UART2
	0x4806 E00C		UART3
Description	Line control register LCR[6:0] define parameters of the transmission and reception for UART mode. LCR[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	

Table 17−76. LCR (Continued)

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reads return 0s.	R	0x00
7	DIV_EN	0x0: Operational mode. 0x1: Divisor latch enable; put the module in Configuration_Mode_A/B. Allows to access to DLL, DLH, and other registers (refer to the registers mapping) Configuration_Mode_B: LCR[7:0] = 0xBF Else, Configuration_Mode_A	RW	0
6	BREAK_EN	Break control bit. UART mode only **Note:** As soon as LCR[6] is set to 1, the TX line is forced to 0 and remains in this state as long as LCR[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 as long as BREAK_EN = 1.	RW	0
5	PARITY_TYPE2	Selects the forced parity format (if LCR[3] = 1). UART mode only. If LCR[5] = 1and LCR[4] = 0, the parity bit is forced to 1 in the transmitted and received data. If LCR[5] = 1 and LCR[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[3] = 1). 0x1: Even parity is generated (if LCR[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-bits (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
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

Table 17−77. MCR

Address Offset	0x010		
Physical Address	0x4806 A010	**Instance**	UART1
	0x4806 C010		UART2
	0x4806 E010		UART3
Description	Modem control register (UART mode only for bits 0 to 5, bit 6 applies to UART/IrDA/CIR modes). MCR[3:0] controls the interface with the modem, data set, or peripheral device that is emulating the modem.		
Type	RW		

Table 17−78. Enhanced Bit Field Details: EFR[4]=1

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	Reserved

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0.	R	0x00
7	Reserved	Read returns 0.	RW	0
6	TCR_TLR	(UART/IrDA/CIR modes) 0x0: No action 0x1: Enables access to the TCR and TLR registers	RW	0
5	XON_EN	(UART mode only) 0x0: Disable XON any function 0x1: Enable XON any function	RW	0
4	LOOPBACK_EN	(UART mode only) 0x0: Normal operating mode 0x1: Enable local loopback mode (internal). In this mode, the MCR[3:0] signals are looped back into MSR[7:4]. The transmit output is looped back to the receive input internally	RW	0
3	CD_STS_CH	(UART mode only) 0x0: In loopback, forces IRQ outputs to inactive state. 0x1: In loopback, forces IRQ outputs to inactive state.	RW	0

Table 17−78. Enhanced Bit Field Details: EFR[4]=1 (Continued)

Bits	Field Name	Description	Type	Reset
2	RI_STS_CH	(UART mode only) 0x0: In loopback, forces nRI input high. 0x1: In loopback, forces nRI input low.	RW	0
1	RTS	In loopback, controls MSR[4].If auto-RTS is enabled, the nRTS output is controlled by hardware flow control. (UART mode only) 0x0: Force nRTS output to inactive (high). 0x1: Force nRTS output to active (low).	RW	0
0	Reserved	Reserved	RW	0

Table 17−79. Normal Bit Field Details: EFR[4]=0

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	Reserved

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0.	R	0x00
7	Reserved	Read returns 0.	R	0
6	TCR_TLR	(UART/IrDA/CIR modes) 0x0: No action 0x1: Enables access to the TCR and TLR registers	R	0
5	XON_EN	(UART mode only) 0x0: Disable ’XON any’ function 0x1: Enable ’XON any’ function	R	0
4	LOOPBACK_EN	(UART mode only) 0x0: Normal operating mode 0x1: Enable local loopback mode (internal). In this mode the MCR[3:0] signals are looped back into MSR[7:4]. The transmit output is looped back to the receive input internally	RW	0

Table 17−79. Normal Bit Field Details: EFR[4]=0 (Continued)

Bits	Field Name	Description	Type	Reset
3	CD_STS_CH	(UART mode only) 0x0: In loopback, forces IRQ outputs to inactive state. 0x1: In loopback, forces IRQ outputs to inactive state.	RW	0
2	RI_STS_CH	(UART mode only) 0x0: In loopback, forces nRI input high. 0x1: In loopback, forces nRI input low.	RW	0
1	RTS	In loop back, controls MSR[4].If auto-RTS is enabled the nRTS output is controlled by hardware flow control. (UART mode only) 0x0: Force nRTS output to inactive (high). 0x1: Force nRTS output to active (low).	RW	0
0	Reserved	Reserved	RW	0

Table 17−80. XON1_ADDR1

Address Offset	0x010		
Physical Address	0x4806 A010	**Instance**	UART1
	0x4806 C010		UART2
	0x4806 E010		UART3
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	Reads return 0s.	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

Table 17−81. LSR

Address Offset	0x014		
Physical Address	0x4806 A014	**Instance**	UART1
	0x4806 C014		UART2
	0x4806 E014		UART3
Description	Line status register		
Type	R		

Table 17−82. UART Bit Field 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	Reads return 0s.	R	0x00
7	RX_FIFO_STS	0x0: Normal operation 0x1: At least one parity error, framing error, or break indication in the RX FIFO. Bit 7 is cleared when no more errors are present in the RX FIFO.	R	0
6	TX_SR_E	0x0: Transmitter hold (TX FIFO) and shift registers are not empty. 0x1: Transmitter hold (TX FIFO) and shift registers are empty	R	1
5	TX_FIFO_E	0x0: Transmit hold register (TX FIFO) is not empty. 0x1: Transmit hold register (TX FIFO) is empty. The transmission is not necessarily completed.	R	1
4	RX_BI	0x0: No break condition 0x1: A break was detected while the data being read from the RX FIFO was being received (i.e., RX input was low for one character + 1 bit time frame).	R	0
3	RX_FE	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)	R	0
2	RX_PE	0x0: No parity error in data being read from RX FIFO 0x1: Parity error in data being read from RX FIFO	R	0

Table 17−82. UART Bit Field Details (Continued)

Bits	Field Name	Description		Type	Reset
1	RX_OE			R	0
		0x0:	No overrun error		
		0x1:	Overrun error has occurred. Set when the character held in the receive shift register is not transferred to the RX FIFO. This case can occurs only when receive FIFO is full.		
0	RX_FIFO_E			R	0
		0x0:	No data in the receive FIFO		
		0x1:	At least one data character in the RX FIFO		

Table 17−83. CIR Bit Field Details

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								THR_EMPTY	Reserved	RX_STOP	Reserved				RX_FIFO_E

Bits	Field Name	Description		Type	Reset
15:8	Reserved	Reads return 0s.		R	0x00
7	THR_EMPTY			R	1
		0x0:	Transmit holding register (TX FIFO) is not empty.		
		0x1:	Transmit hold register (TX FIFO) is empty. The transmission is not necessarily completed.		
6	Reserved				
5	RX_STOP	The RX_STOP is generated based on the value set in the BOF length register (EBLR) It is cleared on a single read of the LSR register.		R	0
		0x0:	Reception is ongoing or waiting for a new frame.		
		0x1:	Reception is completed.		
4:1	Reserved				
0	RX_FIFO_E			R	1
		0x0:	No data in the receive FIFO		
		0x1:	At least one data character in the RX FIFO		

Table 17−84. IrDA Bit Field Details

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								THR_EMPTY	STS_FIFO_FULL	RXLB	FTL	ABORT	CRC	STS_FIFO_E	RX_FIFO_E

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reads return 0s.	R	0x00
7	THR_EMPTY	0x0: Transmit holding register (TX FIFO) is not empty. 0x1: Transmit hold register (TX FIFO) is empty. The transmission is not necessarily completed.	R	1
6	STS_FIFO_FULL	0x0: Status FIFO not full 0x1: Status FIFO full	R	0
5	RX_LB	Receive last byte 0x0: The RX FIFO (RHR) does not contain the last byte of the frame to be read. 0x1: The RX FIFO (RHR) contains the last byte of the frame to be read. This bit is only set 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 register.	R	0
4	FTL	Frame too long 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 and RXFLL registers) has been received. When this error is detected, current frame reception is terminated. Reception is stopped until the next START flag is detected.	R	0

Table 17−85. Normal Bit Field Details: EFR[4]=0

Bits	Field Name	Description		Type	Reset
3	ABORT			R	0
		0x0:	No abort pattern error in frame		
		0x1:	Abort pattern is 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	1
		0x0:	Status FIFO not empty		
		0x1:	Status FIFO empty		
0	RX_FIFO_E			R	1
		0x0:	No data in the receive FIFO		
		0x1:	At least one data character in the RX FIFO		

Table 17−86. XON2_ADDR2

Address Offset	0x014		
Physical Address	0x4806 A014	**Instance**	UART1
	0x4806 C014		UART2
	0x4806 E014		UART3
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	Reads return 0s.	R	0x00
7:0	XON_WORD2	Used to store the 8-bit XON2 character in UART modes and ADDR2 address 2 for IrDA modes	RW	0x00

Table 17–87. XOFF1

Address Offset	0x018		
Physical Address	0x4806 A018	**Instance**	UART1
	0x4806 C018		UART2
	0x4806 E018		UART3
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	Reads return 0s.	R	0x00
7:0	XOFF_WORD1	Used to store the 8-bit XOFF1 character in used in UART modes	RW	0x00

Table 17–88. TCR

Address Offset	0x018		
Physical Address	0x4806 A018	**Instance**	UART1
	0x4806 C018		UART2
	0x4806 E018		UART3
Description	Transmission control register. This register is used to store 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	Reads return 0s.	R	0x00
7:4	RX_FIFO_TRIG_START	RX FIFO trigger level to RESTORE transmission (0 – 60)	RW	0x0
3:0	RX_FIFO_TRIG_HALT	RX FIFO trigger level to HALT transmission (0 – 60)	RW	0xF

Table 17−89. MSR

Address Offset	0x018		
Physical Address	0x4806 A018	**Instance**	UART1
	0x4806 C018		UART2
	0x4806 E018		UART3
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 LH. 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	Reads return 0s.	R	0x00
7	NCD_STS	In loopback mode, it is equivalent to MCR[3].	R	–
6	NRI_STS	This bit is the complement of the nRI input. In loopback mode, it is equivalent to MCR[2].	R	–
5	NDSR_STS	In loopback mode, it is equivalent to MCR[0].	R	–
4	NCTS_STS	This bit is the complement of the nCTS input. In loopback mode, it is equivalent to MCR[1].	R	–
3	DCD_STS	Indicates that MCR[3] in loopback has changed. Cleared on a read.	R	0
2	RI_STS	Indicates that nRI input (or MCR[2] in loopback) has changed state from low to high. Cleared on a read.	R	0
1	DSR_STS	0x1: Indicates that MCR[0] in loopback has changed state. Cleared on a read.	R	0
0	CTS_STS	0x1: Indicates that nCTS input (or MCR[1] in loopback) has changed state. Cleared on a read.	R	0

Table 17–90. SPR

Address Offset	0x01C		
Physical Address	0x4806 A01C	**Instance**	UART1
	0x4806 C01C		UART2
	0x4806 E01C		UART3
Description	Scratchpad register This read/write register does not control the module in anyway. It is intended as a scratchpad register to be used by the programmer to hold temporary data.		
Type	RW		

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	Reads return 0s.	R	0x00
7:0	SPR_WORD	Scratchpad register	RW	0x00

Table 17–91. XOFF2

Address Offset	0x01C		
Physical Address	0x4806 A01C	**Instance**	UART1
	0x4806 C01C		UART2
	0x4806 E01C		UART3
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	Reads return 0s.	R	0x00
7:0	XOFF_WORD2	Used to store the 8-bit XOFF2 character in used in UART modes	RW	0x00

Table 17−92. TLR

Address Offset	0x01C		
Physical Address	0x4806 A01C	**Instance**	UART1
	0x4806 C01C		UART2
	0x4806 E01C		UART3
Description	Trigger level register. This register is used to store 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				RX_FIFO_TRIG_DMA			

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reads return 0s.	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

Table 17–93. MDR1

Address Offset	0x020		
Physical Address	0x4806 A020	**Instance**	UART1
	0x4806 C020		UART2
	0x4806 E020		UART3
Description	Mode definition register 1. The mode of operation can be programmed by writing to MDR1[2:0]; therefore, the MDR1 must be programmed on start-up after configuration of the configuration registers (DLL, DLH, LCR). The value of MDR1[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_MODE	SIP_MODE	SCT	SET_IRTX	IR_SLEEP	MODE_SELECT		

Bits	Field Name	Description		Type	Reset
15:8	Reserved	Reads return 0s.		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[3].		
		0x1:	Automatic SIP mode: SIP is generated after each transmission.		
5	SCT	Store and control the transmission. IrDA only.		RW	0
		0x0:	Starts the Infrared transmission as soon as a value is written to THR		
		0x1:	Starts the Infrared transmission with the control of ACREG[2] **Note:** Before starting any transmission, there must be no reception ongoing.		

Table 17–93. MDR1 (Continued)

Bits	Field Name	Description	Type	Reset
4	SET_IRTX	Used to configure the Infrared transceiver. IrDA only. 0x0: No action 0x1: IRTX pin output is forced high.	RW	0
3	IR_SLEEP	0x0: IrDA/CIR sleep mode disabled 0x1: IrDA/CIR sleep mode enabled	RW	0
2:0	MODE_SELECT	UART-IrDA-CIR mode selection 0x0: UART 16x mode 0x1: SIR mode 0x2: UART 16x auto-baud 0x3: UART 13x mode 0x4: MIR mode 0x5: FIR mode 0x6: CIR mode 0x7: Disable (default state)	RW	0x7

Table 17–94. MDR2

Address Offset	0x024		
Physical Address	0x4806 A024	**Instance**	UART1
	0x4806 C024		UART2
	0x4806 E024		UART3
Description	Mode definition register 2. IR-IrDA and IR-CIR modes only. MDR2[0] describes the status of the interrupt in IIR[5]. The IRTX_UNDERRUN bit must be read after an IIR[5] TX_STATUS_IT interrupt has occurred. 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[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

Table 17−94. MDR2 (Continued)

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reads return 0s.	R	0x00
7	Reserved		R	0
6	IRRXINVERT	Only for IR mode (IrDA and CIR). Invert RX pin inside 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. It 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	0x0
3	UART_PULSE	UART mode only. Used to allow pulse shaping in UART mode. 0x0: Normal UART mode 0x1: UART mode with a 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	0x0
0	IRTX_ UNDERRUN	IrDA transmission status interrupt. When the IIR[5] interrupt occurs, the meaning of the interrupt is : 0x0: The last bit of the frame has been transmitted successfully without error. 0x1: An underrun has occurred. The last bit of the frame has been transmitted but with an underrun error present. The bit is reset to 0 when the RESUME register is read.	R	0

Table 17−95. TXFLL

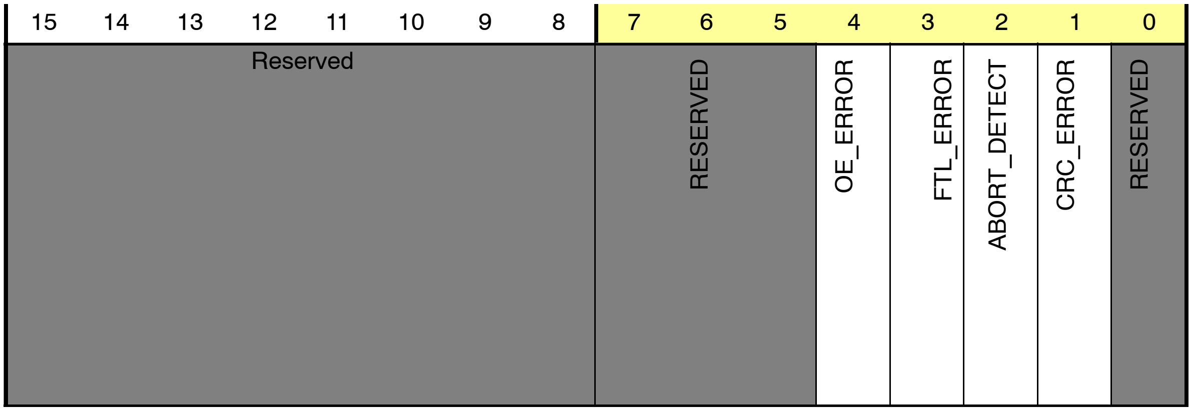
Address Offset	0x028		
Physical Address	0x4806 A028	**Instance**	UART1
	0x4806 C028		UART2
	0x4806 E028		UART3
Description	Transmit frame length register low. IrDA modes only. The registers TXFLL and TXFLH hold the 13-bit transmit frame length (expressed in bytes). TXFLL holds the least significant bits and TXFLH holds the most significant bits. 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 effect.	W	0x00
7:0	TXFLL	LSB register used to specify the frame length	W	0x00

Table 17–96. SFLSR

Address Offset	0x028		
Physical Address	0x4806 A028	**Instance**	UART1
	0x4806 C028		UART2
	0x4806 E028		UART3
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 and SFREGH must be read first).		
Type	R		



Bits	Field Name	Description		Type	Reset
15:8	Reserved	Reads return 0s.		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	Reads return 0s.		R	0

Table 17−97. RESUME

Address Offset	0x02C		
Physical Address	0x4806 A02C	**Instance**	UART1
	0x4806 C02C		UART2
	0x4806 E02C		UART3
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 reads always 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	Reads return 0s.	R	0x00
7:0	RESUME	Dummy read to restart the TX or RX	R	0x00

Table 17−98. TXFLH

Address Offset	0x02C		
Physical Address	0x4806 A02C	**Instance**	UART1
	0x4806 C02C		UART2
	0x4806 E02C		UART3
Description	Transmit frame length register low. IrDA modes only. The registers TXFLL and TXFLH hold the 13-bit transmit frame length (expressed in bytes). TXFLL holds the least significant bits and TXFLH holds the most significant bits. 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	Reads return 0s	R	0x00
7:5	Reserved		R	0x0
4:0	TXFLH	MSB register used to specify the frame length	W	0x00

Table 17−99. RXFLL

Address Offset	0x030		
Physical Address	0x4806 A030	**Instance**	UART1
	0x4806 C030		UART2
	0x4806 E030		UART3
Description	Received frame length register low. IrDA modes only. The registers RXFLL and RXFLH hold the 12-bit receive maximum frame length. RXFLL holds the least significant bits, and RXFLH holds the most significant bits. If the intended maximum receive frame length is n bytes, program RXFLL and RXFLH to be n + 3 in SIR or MIR modes and n + 6 in FIR mode (+3 and +6 are due to 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								RXFLL							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Write has no effect.	W	0x00
7:0	RXFLL	LSB register used to specify the frame length in reception	W	0x00

Table 17−100. SFREGL

Address Offset	0x030		
Physical Address	0x4806 A030	**Instance**	UART1
	0x4806 C030		UART2
	0x4806 E030		UART3
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 and SFREGH registers (these registers do not physically exist). The least significant bits are read from SFREGL, and the most significant bits are read from SFREGH. 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.		
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	Reads return 0s.	R	0x00
7:0	SFREGL	LSB part of the frame length	R	0x−−

Table 17−101. SFREGH

Address Offset	0x034		
Physical Address	0x4806 A034	**Instance**	UART1
	0x4806 C034		UART2
	0x4806 E034		UART3
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 and SFREGH registers (these registers do not physically exist). The least significant bits are read from SFREGL, and the most significant bits are read from SFREGH. 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.		
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	Reads return 0s.	R	0x00
7:4	Reserved		R	0x0
3:0	SFREGH	MSB part of the frame length	R	0x−

Table 17−102. RXFLH

Address Offset	0x034		
Physical Address	0x4806 A034	**Instance**	UART1
	0x4806 C034		UART2
	0x4806 E034		UART3
Description	Received frame length register high. IrDA modes only. The registers RXFLL and RXFLH hold the 12-bit receive maximum frame length. RXFLL holds the least significant bits, and RXFLH holds the most significant bits. If the intended maximum receive frame length is n bytes, program RXFLL and RXFLH to be n + 3 in SIR or MIR modes and n + 6 in FIR mode (+3 and +6 are due to 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 effect.	W	0x00
7:4	Reserved		W	0x0
3:0	RXFLH	MSB register used to specify the frame length in reception	W	0x0

Table 17–103. BLR

Address Offset	0x038		
Physical Address	0x4806 A038	**Instance**	UART1
	0x4806 C038		UART2
	0x4806 E038		UART3
Description	BOF control register. IrDA modes only. Note that BLR[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 will always be 0xC0. If n start flags are required, either (n−1) 0xC0 or (n−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	Reads return 0s.	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		R	0x00

Table 17−104. UASR

Address Offset	0x038		
Physical Address	0x4806 A038	**Instance**	UART1
	0x4806 C038		UART2
	0x4806 E038		UART3
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 the 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	Reads return 0s.		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 bauds		
		0x2:	57 600 bauds		
		0x3:	38 400 bauds		
		0x4:	28 800 bauds		
		0x5:	19 200 bauds		
		0x6:	14 400 bauds		
		0x7:	9 600 bauds		
		0x8:	4 800 bauds		
		0x9:	2 400 bauds		
		0xA:	1 200 bauds		

Table 17–105. ACREG

Address Offset	0x03C		
Physical Address	0x4806 A03C	**Instance**	UART1
	0x4806 C03C		UART2
	0x4806 E03C		UART3
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	Reads return 0s.	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

Table 17−105. ACREG (Continued)

Bits	Field Name	Description	Type	Reset
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[5] = 1 and the LH 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 LH 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
0	EOT_EN	EOT (end of transmission) bit. The LH 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 LH writes to the THR (TX FIFO).	RW	0

Table 17−106. SCR

Address Offset	0x040		
Physical Address	0x4806 A040	**Instance**	UART1
	0x4806 C040		UART2
	0x4806 E040		UART3
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

Table 17−106. SCR (Continued)

Bits	Field Name	Description		Type	Reset
15:8	Reserved	Reads return 0s.		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	Reserved		RW	0
4	RX_CTS_ WU_EN	RX CTS wake-up enable		RW	0
		0x0:	Disables the WAKE UP interrupt and clears SSR[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 17−41 for details on UART mode interrupts)		
		0x1:	The THR interrupt is generated when TX FIFO and TX shift register are empty.		
2:1	DMA_MODE_2	Used to specify the DMA mode valid if SCR[0] = 1		RW	0x0
		0x0:	DMA mode 0 (no DMA)		
		0x1:	DMA mode 1 (UARTn_DMA_TX, UARTn_DMA_RX)		
		0x2:	DMA mode 2 (UART2_DMA_RX)		
		0x3:	DMA mode 3 (UARTn_DMA_TX)		
0	DMA_MODE_ CTL			RW	0
		0x0:	The DMA_MODE is set with FCR[3].		
		0x1:	The DMA_MODE is set with SCR[2:1].		

Table 17−107. SSR

Address Offset	0x044		
Physical Address	0x4806 A044	**Instance**	UART1
	0x4806 C044		UART2
	0x4806 E044		UART3
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	Reads return 0s.	R	0x00
7:2	Reserved		R	0x00
1	RX_CTS_WU_STS	Pin falling edge detection: reset only when SCR[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

Table 17–108. EBLR

Address Offset	0x048		
Physical Address	0x4806 A048	**Instance**	UART1
	0x4806 C048		UART2
	0x4806 E048		UART3
Description	BOF length register. IR-IrDA and IR-CIR modes only. In IR-IrDA SIR operation, this register specifies the number of BOF + xBOFs to transmit. Value set into this register must take into account 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). In IR-CIR mode, this register specifies the number of consecutive zeros to be received before generating the RX_STOP interrupt (IIR[2]). All the received zeros are stored in the RX FIFO. When the register is set to 0, this feature is deactivated and is always in reception state, which can be disabled by setting the ACREG[5] to 1.		
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	Reads return 0s.	R	0x00
7:0	EBLR	IR-IrDA mode: This register allows to define up to 176 xBOFs, the maximum required by IrDA specification. IR-CIR mode: This register specifies the number of consecutive zeros to be received before generating the RX_STOP interrupt (IIR[2]). 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

Table 17−109. MVR

Address Offset	0x050		
Physical Address	0x4806 A050	**Instance**	UART1
	0x4806 C050		UART2
	0x4806 E050		UART3
Description	Module version register The reset value is fixed by hardware and corresponds to the RTL revision of this module. A reset has no effect on the value returned −UART/IrDA SIR only module is revision 1.x (WMU_012_1 specification). −UART/IrDA with SIR, MIR, and FIR support is revision 2.x (WMU_012_2 specification). −UART/IrDA with SIR, MIR, and FIR/CIR support is revision 3.x (this specification). For example: MVR = 0x30 => version 3.0 MVR = 0x38 => version 3.8		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								MAJOR_REV				MINOR_REV			

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reads return 0s.	R	0x00
7:4	MAJOR_REV	Major revision number of the module	R	
3:0	MINOR_REV	Minor revision number of the module	R	

Table 17−110. SYSC

Address Offset	0x054		
Physical Address	0x4806 A054	**Instance**	UART1
	0x4806 C054		UART2
	0x4806 E054		UART3
Description	System configuration register. The auto idle bit controls a power-saving technique to reduce the logic power consumption of the OCP interface. That is to say, when the feature is enabled, the clock is gated off until an OCP command for this device has been detected. 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	Reads return 0s.	R	0x00
7:5	Reserved		R	0x0
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	RW	0x0
2	ENAWAKEUP	WAKE UP FEATURE CONTROL 0x0: Wake-up is disabled. 0x1: Wake-up capability is enabled.	RW	0
1	SOFTRESET	Software reset. Set this bit to 1 to trigger a module reset. This bit is automatically reset by the hardware. During reads it always returns a 0. 0x0: Normal mode 0x1: The module is reset.	RW	0
0	AUTOIDLE	Internal OCP clock gating strategy 0x0: Clock is running. 0x1: Automatic interface clock gating strategy is applied based on the interface activity.	RW	0

Table 17−111. SYSS

Address Offset	0x058		
Physical Address	0x4806 A058	**Instance**	UART1
	0x4806 C058		UART2
	0x4806 E058		UART3
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	Reads return 0s.	R	0x00
7:1	Reserved		R	0x00
0	RESETDONE	Internal reset monitoring 0x0: Internal module reset is ongoing. 0x1: Reset completed †	R	0

† During reset the value is 0, but the value read just after the reset is 1 because ICLK/FCLK is already operating.

Table 17–112. WER

Address Offset	0x05C		
Physical Address	0x4806 A05C	**Instance**	UART1
	0x4806 C05C		UART2
	0x4806 E05C		UART3
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. The events are any activity in the logic that can cause an interrupt and/ or an activity that requires the system to wake up. It must be noted that even if the wake-up is disabled for certain events, if these events are also an interrupt to the UART, then 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	Reads return 0s.	R	0x00
7	Reserved		R	0
6	EVENT_6_RLS_INTERRUPT	Receiver line status interrupt 0x0: Event is not allowed to wake up the system. 0x1: Event can wake up the system.	RW	1
5	EVENT_5_RHR_INTERRUPT	0x0: Event is not allowed to wake up the system. 0x1: Event can wake up the system.	RW	–

Table 17−112. WER (Continued)

Bits	Field Name	Description	Type	Reset
4	EVENT_4_RX_ACTIVITY	0x0: Event is not allowed to wake up the system. 0x1: Event can wake up the system.	RW	1
3	Reserved	Reserved	RW	1
2	EVENT_2_RI_ACTIVITY	0x0: Event is not allowed to wake up the system. 0x1: Event can wake up the system.	RW	1
1	Reserved	Reserved	RW	1
0	EVENT_0_CTS_ACTIVITY	UART mode only 0x0: Event is not allowed to wake up the system. 0x1: Event can wake up the system.	RW	1

Table 17−113. CFPS

Address Offset	0x060		
Physical Address	0x4806 A060	**Instance**	UART1
	0x4806 C060		UART2
	0x4806 E060		UART3
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	Reads return 0s.	R	0x00
7:0	CFPS	Since the consumer IR works at modulation rates of 30−56.8 kHz, the 48-MHz clock must be pre scaled before the clock can drive the IR logic. This register sets the divisor rate to give a range to accommodate the 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.

Chapter 18

I^2C

This chapter describes the inter-integrated circuit (I^2C) modules on the OMAP2420 multimedia device.

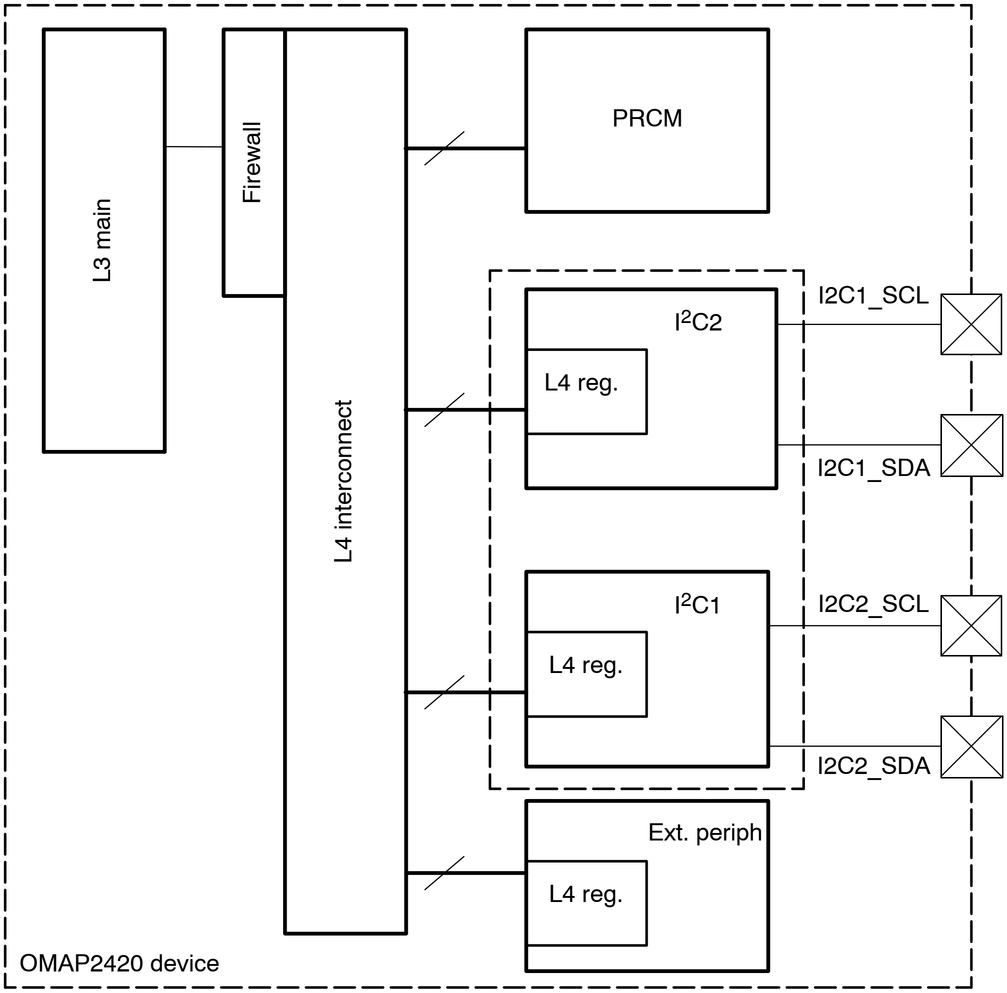
Topic **Page**

18.1 I^2C Overview

The multimaster inter-integrated circuit (I^2C) peripheral provides an interface between a local host (LH) such as a microprocessor unit (MPU), MIPS, or digital signal processor (DSP) and any I^2C-bus-compatible device that connects through the I^2C serial bus. External components attached to the I^2C bus can serially transmit/receive up to 8 bits of data to/from the LH device through the 2-wire I^2C interface.

This I^2C peripheral supports any slave or master I^2C-compatible device. Figure 18−1 shows the example of a system with multiple I^2C-compatible devices in which the I^2C serial ports are connected together for a 2-way transfer from one device to other devices. OMAP2420 devices contain two I^2C modules.

Figure 18−1. I^2C System Overview



18.1.1 Functional Overview

The I^2C bus is a multimaster bus. The I^2C controller supports the multimaster mode that allows more than one device capable of controlling the bus to be connected to it. Each I^{2}C device is recognized by a unique address and can operate as either a transmitter or receiver, according to the function of the device. In addition to being a transmitter or receiver, a device connected to the I^{2}C bus can also be considered as master or slave when performing data transfers. A master device initiates a data transfer on the bus and generates the clock signals to permit that transfer. During this transfer, any device addressed by this master is considered a slave.

18.1.2 Features

The main features of the I^{2}C controller are:

- Compliant to Philips I^{2}C specification version 2.1
- Support for standard mode (up to 100K bps) and fast mode (up to 400K bps)
- 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
- Module enable/disable capability
- Programmable clock generation
- 16-bit-wide access to maximize bus throughput
- Designed for low power
- Two direct memory access (DMA) channels
- Wide interrupt capability

The present I^{2}C does not support:

- High-speed (HS) mode for transfer up to 3.4M bps
- C-bus compatibility mode

18.2 I^2C Environment

18.2.1 I^2C Master/Slave Controller Signal Pads

Data is communicated to devices that interface with the I^2C through the serial data (SDA) line and the serial clock (SCL) line. These two wires carry information between the OMAP2420 and others connected to the I^2C bus. Both SDA and SCL are bidirectional pins. They must be connected to a positive supply voltage by using a pullup resistor. When the bus is free, both pins are high. The driver of these two pins has an open−drain to perform the required wired-AND function.

Table 18−1 lists the pins associated with the I^2C interface.

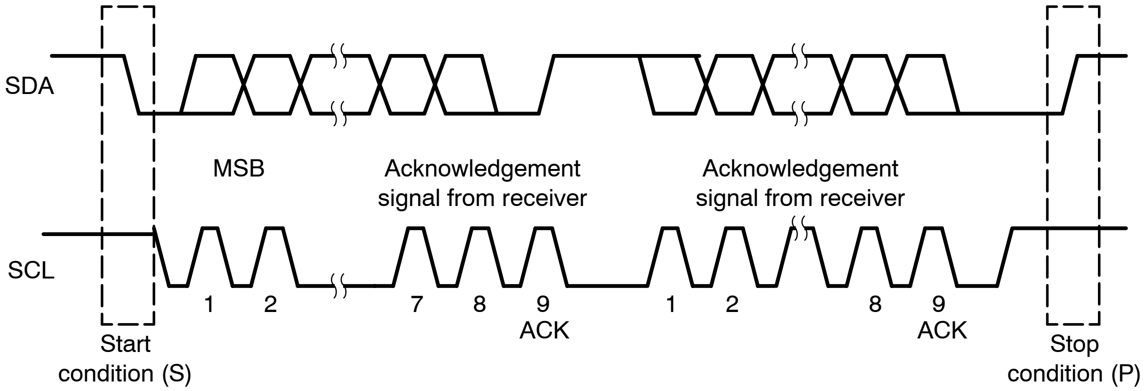
Table 18−1. Signal Pads

Name	Type	Reset Value	Description
I2C1.SCL	In/ Out(OD)	Input	Serial CLK line of I2C1 instance. Open-drain output buffer. Requires external pullup resistor (Rp).
I2C1.SDA	In/ Out(OD)	Input	Serial data line of I2C1 instance. Open-drain output buffer. Requires external pullup resistor (Rp).
I2C2.SCL	In/ Out(OD)	Input	Serial CLK line of I2C2 instance. Open-drain output buffer. Requires external pullup resistor (Rp).
I2C2.SDA	In/ Out(OD)	Input	Serial data line of I2C2 instance. Open-drain output buffer. Requires external pullup resistor (Rp).

18.2.2 Serial Data Formats

The I^2C controller operates in 16-bit word data format (byte write access supported for the last access). Each byte put on the SDA line is 8 bits long. The number of bytes that can be transmitted or received is unrestricted. The data is transferred with the most-significant bit (MSB) first. Each byte is followed by an acknowledge bit from the I^2C module if it is in receiver mode. Figure 18−2 shows a typical I^2C communication format. The I^2C controller supports endianness.

Figure 18−2. I2C Data Transfer

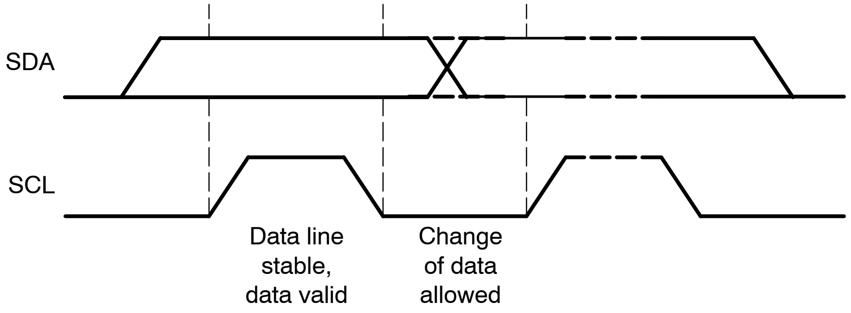

SDA
SCL
MSB
Acknowledgement signal from receiver
Acknowledgement signal from receiver
1
2
7
8
9
ACK
1
2
8
9
ACK
Start condition (S)
Stop condition (P)

18.2.3 Data Validity

The data on the SDA line must be stable during the high period of the clock. The high and low states of the data line can change only when the clock signal on the SCL line is low.

Figure 18−3 shows an example of data validity requirements.

Figure 18−3. Bit Transfer on the I2C Bus



18.2.4 Start and Stop Conditions

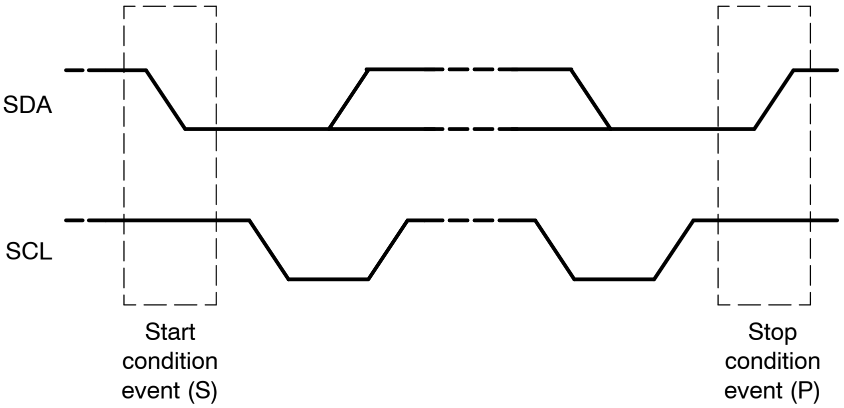
The I2C module generates start and stop conditions when it is configured as a master.

- A start condition is a high-to-low transition on the SDA line while SCL is high.
- A stop condition is a low-to-high transition on the SDA line while SCL is high.

The bus is considered busy after the start condition BUS_BUSY = 1 (BB) and free after the stop condition (BB=0).

Figure 18−4 shows the waveform during a start and stop condition.

Figure 18−4. Start and Stop Condition Events

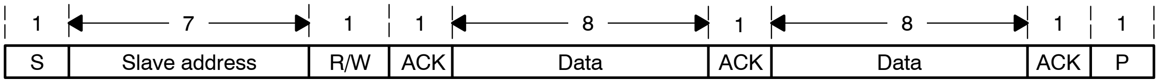


18.2.5 Addressing

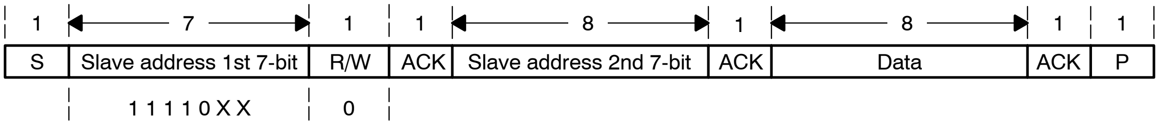
The I²C module supports two data formats, as shown in Figure 18−5.

- 7-bit/10-bit addressing format
- 7-bit/10-bit addressing format with repeated start condition

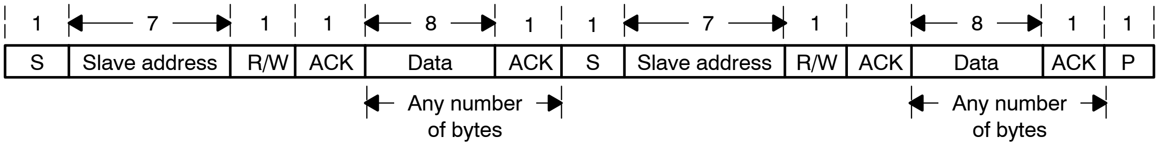
Figure 18−5. I²C Data Transfer Formats



(a) 7-bit addressing format



(b) 10-bit addressing format



(c) Addressing format with repeated start condition

The first word after a start condition (S) always consists of 8 bits. In acknowledge mode, an extra bit dedicated for acknowledgement is inserted after each byte.

In the addressing formats with 7-bit addresses, the first byte is composed by 7 MSB slave address bits and 1 least-significant bit (LSB) R/W_.

The LSB R/W_ of the address byte indicates the direction of transmission of the following data bytes. If R/W_ is 0, the master writes data into the selected slave; if R/W_ is 1, the master reads data out of the slave.

In the addressing formats with 10-bit addresses, the first byte is composed with the pattern 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, then the next byte contains the last 8 bits of the slave address. If the R/W_ bit is 1, then the next byte contains transmitted data from slave to master.

18.2.5.1 Master Transmitter

In this mode, data assembled in one of the previously described data formats is shifted out on the SDA line in synchronization with the self-generated clock pulses on the SCL line. The clock pulses are inhibited and SCL is held low when the intervention of the processor is required after a byte has been transmitted.

18.2.5.2 Master Receiver

This mode can be entered only from the master transmitter mode. With either of the address formats [see Figure 18−5 (a), (b), and (c)], the master receiver is entered after the slave address byte and the R/W_ bit have been transmitted, if R/W_ is high. Serial data bits received on bus line SDA are shifted in sync with the self-generated clock pulses on the SCL line. The clock pulses are inhibited and SCL is held low when the intervention of the processor is required after a byte has been transmitted. At the end of a transfer, it generates the stop condition.

18.2.5.3 Slave Transmitter

This mode can be entered only from the slave receiver mode. With either of the address formats [see Figure 18−5 (a), (b), and (c)], the slave transmitter is entered if the slave address byte is the same as its own address and the R/W_ bit has been transmitted, if R/W_ is high. The slave transmitter shifts the serial data out on the data line SDA in sync with the clock pulses that are generated by the master device. It does not generate the clock but it can hold clock line SCL low while intervention of the LH is required (XUDF).

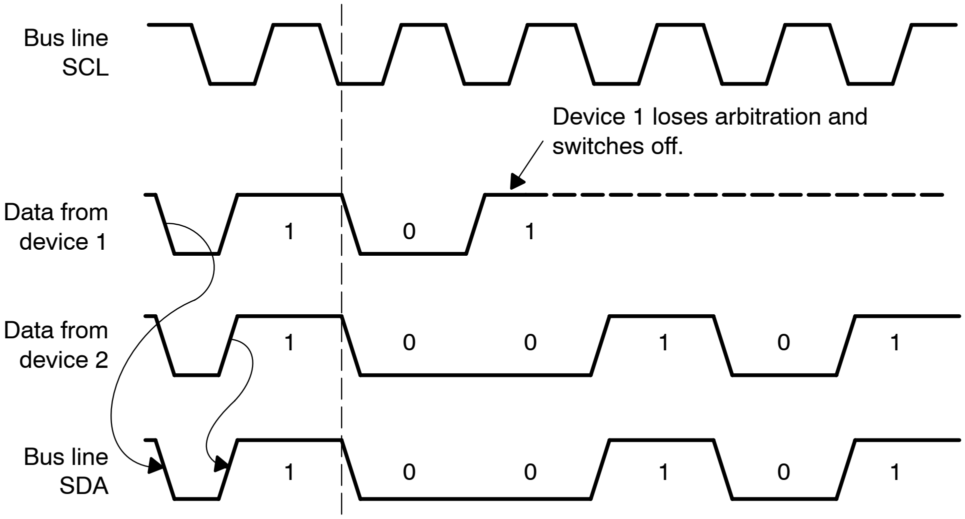
18.2.5.4 Slave Receiver

In this mode, serial data bits received on the bus line SDA are shifted-in in sync with the clock pulses on SCL that are generated by the master device. It does not generate the clock but it can hold clock line SCL low while intervention of the LH is required (ROVR) following the reception of a byte.

18.2.6 Arbitration

If two or more master transmitters start a transmission on the same bus almost simultaneously, an arbitration procedure is invoked. The 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, the transmitter switches to the slave receiver mode, sets the arbitration lost (AL) flag, and generates the AL interrupt. Figure 18−6 shows the arbitration procedure 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 18−6. Arbitration Procedure Between Two Master Transmitters

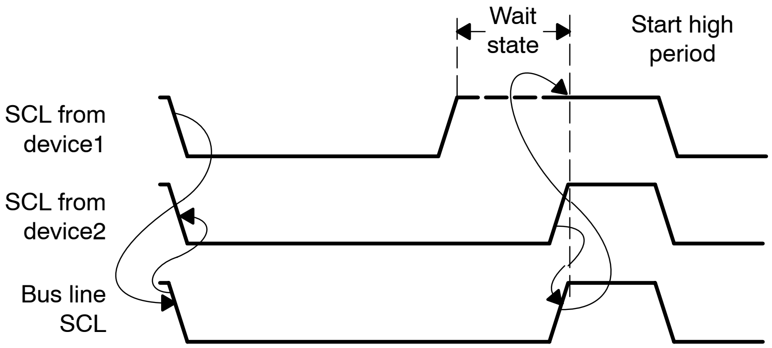


18.2.7 I²C Clock Generation and I²C Clock Synchronization

Under normal conditions, only one master device generates the clock signal (SCL). During the arbitration procedure, 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 a 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 period. 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, the result is that 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. Figure 18−7 shows the clock synchronization.

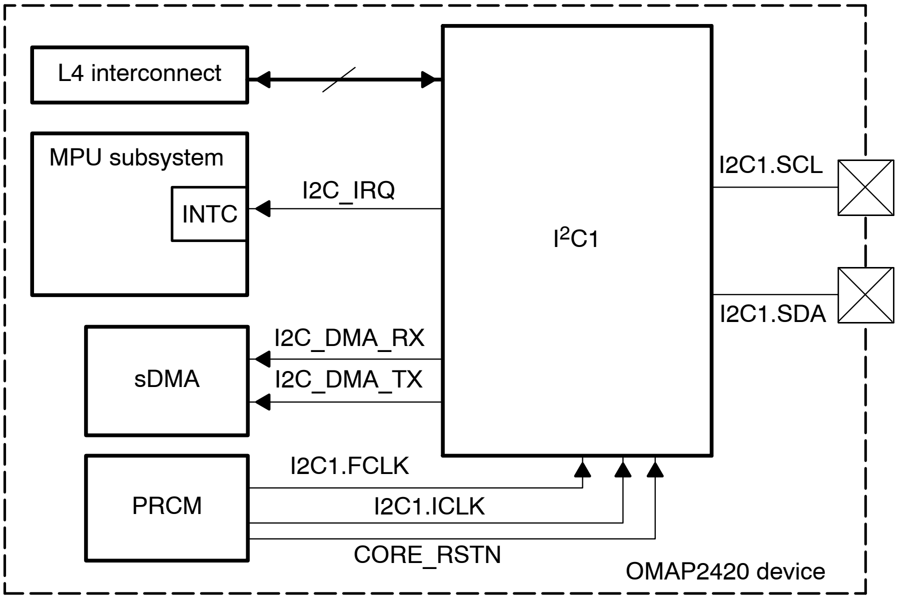
Figure 18−7. Synchronization of Two I²C Clock Generators

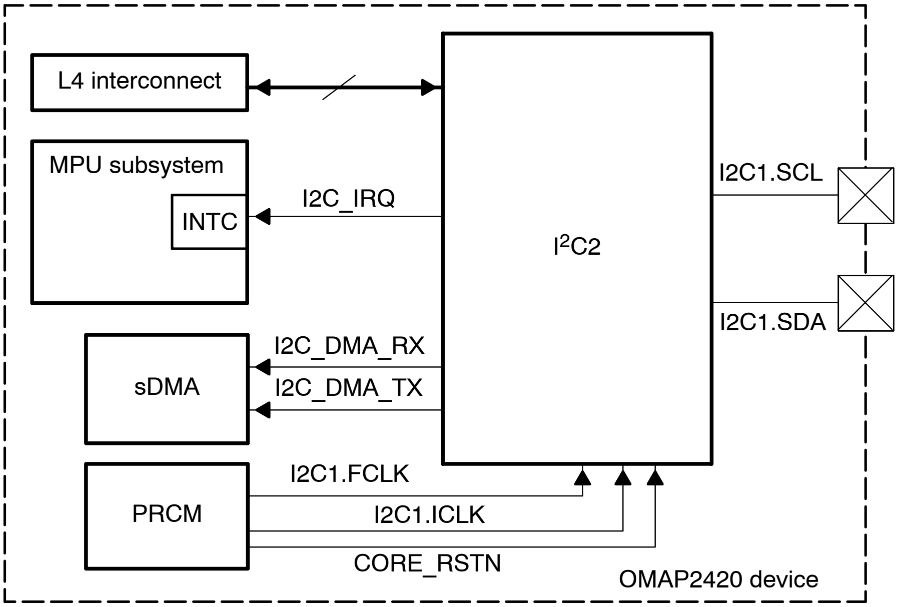


18.3 I2C Integration

18.3.1 I2C Description

Figure 18−8. Integration of I2C Modules





18.3.1.1 Clocking, Reset, and Power-Management Scheme

Each I2C module is clocked with an independent functional clock of 12 MHz (I2Cn_FCLK) and an interface clock (I2Cn_ICLK) for interfacing with L4. This clock is provided by the power, reset, and clock management (PRCM) module and is fixed at 12 MHz. The functional clock is processed by the prescaler to produce the internal sampling clock (I2Cn_ICLK). This clock is generated by the I2C prescaler block.

The prescaler consists of an 8-bit register (I2C_PSC) that is used to divide down the functional clock to obtain an internal sampling clock with a frequency value of I2Cn_FCLK/(PSC + 1).

Table 18−2 lists each I^{2}C functional clock.

Table 18−2. I^{2}C Clocks

	Frequency	Name	Comments
I2C1 functional clock	12 MHz	I2C1_FCLK	Source is PRCM module.
I2C2 functional clock	12 MHz	I2C2_FCLK	Source is PRCM module.
I2C1 interface clock	See L4 interconnect in Chapter 6, *Internal Interconnect*.	I2C1_ICLK	Source is PRCM module.
I2C2 interface clock	See L4 interconnect in Chapter 6, *Internal Interconnect*.	I2C2_ICLK	Source is PRCM module.
I2C1 interface clock	12/(PSC + 1)MHz (see note)	I2C1_internal CLK	Internal clock
I2C2 interface clock	12/(PSC + 1)MHz (see note)	I2C2_internal CLK	Internal clock

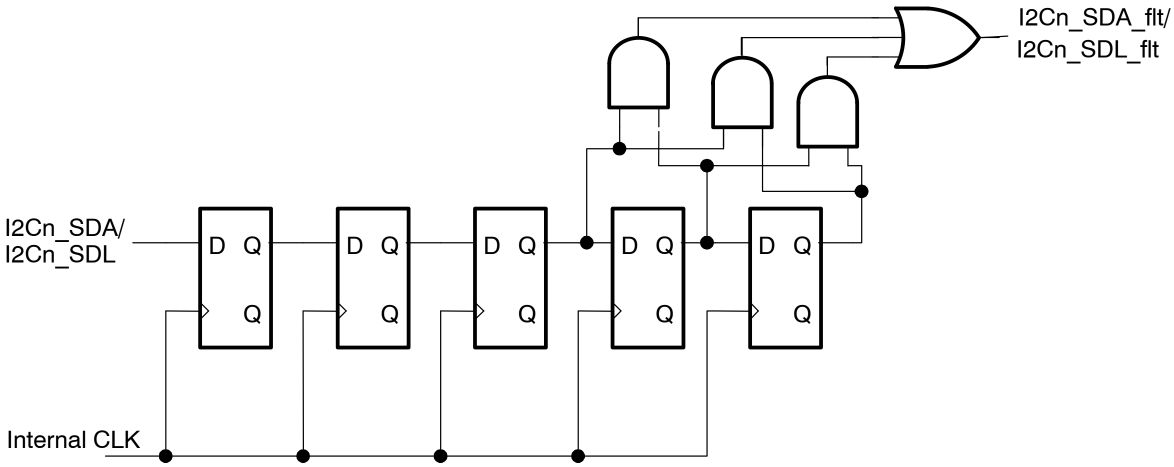
Note: PSC: Prescaler field of the I2C_PSC register

18.3.1.2 Noise Filter

The noise filter suppresses any noise 50 ns or less. The I^{2}C module contains one noise canceller circuit for each SCL and SDA line created from a 5-bit shift register and clocked from the independent sample CLK (ICLK).

Figure 18−9 shows the components of the noise filter. The first 2 bits of the shifter are employed as a meta-stable value removal to produce clean values from the input. The second part serves as a majority selector block.

Figure 18−9. I^{2}C Module Noise Filter



18.3.1.3 Power Domain

The I^{2}C module is powered by the CORE power domain.

18.3.1.4 Power Management

As part of the system-wide power-management scheme, the I^2C module can go into idle mode at the request of the PRCM module. However, the I^2C module does not support handshake protocol with the PRCM module. The I^2C 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 PRCM_AUTOIDLE1_CORE register AUTO_I2C1 and AUTO_I2C2 bits (CM_AUTOIDLE1_CORE[20:19]) are set, the I^2C module behavior follows the L4 interconnect clock domain behavior. If the whole domain is put into idle mode, the I^2C is also put into idle mode.

The software must ensure correct clock management. There is no hardware mechanism to prevent the I^2C clocks from being cut while the module performs a transfer.

18.3.1.5 Module Resets

Hardware Reset

The module is reset by the hardware when an active-low reset signal is asserted on the input pin reset from the PRCM signal CORE_RSTN.

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.

Software Reset

The I^2C register bit I2C_SYSC[1] functions as a software reset. It is triggered by writing 1 to this bit. The software reset action on the module is equivalent to a hardware reset.

The I^2C module enable I2C_EN (I^2C register bit I2C_CON[15]) can hold the I^2C module in reset after the device reset has been released. When the system bus reset is removed (RESET_ = 1), I2C_EN remains set to 0. The functional part of the I^2C module is held in reset state while I2C_EN = 0 and all configuration registers can be accessed.

18.3.1.6 Hardware Requests

The I^2C module can issue requests to the LH with interrupts and DMA events.

I^2C Interrupts

Each of the two I^2C modules owns one interrupt line. Table 18–3 lists the I^2C interrupts. The I^2C module generates six types of interrupts:

- Arbitration lost
- No-acknowledge
- General call
- Registers-ready-for-access
- Receive
- Transmit

When the interrupt signal is activated, the LH must read the I2C_STAT register to determine the type of interrupt, process the request, and then write the value 1 to the appropriate bit to clear the interrupt flag.

These six interrupts are accompanied by six interrupt masks and flags defined in the I2C_IE and I2C_STAT registers, respectively:

- ☐ Arbitration lost (AL) interrupt is generated when the I^2C arbitration procedure is lost.
- ☐ No-acknowledge (NACK) interrupt is generated when the master I^2C does not receive an acknowledgement from the receiver.
- ☐ General call (GC) interrupt is generated when the device detects the general call address. The general call address is an address of all zeros (applies to both 7-bit and 10-bit addressing mode).
- ☐ Registers-ready-for-access (ARDY) interrupt is generated by the I^2C when the previously programmed address, data, and command have been performed and the status bits have been updated. This interrupt is used to let the LH know that the I^2C registers are ready for access.
- ☐ Receive interrupt/status (RRDY) is generated when received data from the I2C_DATA register is ready to be read by the LH. The LH can poll this bit to read the received data.
- ☐ Transmit interrupt/status (XRDY) is generated when the LH must put more data in the I2C_DATA register after the transmitted data has been shifted out on the SDA pin. The LH can poll this bit to write the next transmitted data into the I2C_DATA register.

Table 18−3. I^2C Interrupts

Attributes	Values	Name	Mapping	Comments
I2C1				
Interrupt request	1	I2C1_IRQ	M_IRQ_56	Destination is MPU subsystem interrupts.
I2C2				
Interrupt request	1	I2C2_IRQ	M_IRQ_57	Destination is MPU subsystem interrupts.

I^2C DMA Events

The I^2C module can generate two DMA requests events, read (I2C_DMA_RX) and write (I2C_DMA_TX), which can be used by the DMA controller to synchronously read received data from the I2C_DATA register and write transmitted data to the I2C_DATA register. The DMA read and write requests are generated, but the bits I2C_STAT[RRDY] and I2C_STAT[XRDY] are not set.

The I^2C DMA request signals (I2C_DMA_TX and I2C_DMA_RX) are activated for every new 16-bit word to be read or written in the FIFOs.

Table 18−4 lists the DMA events for the I^2C module.

Table 18−4. I^{2}C DMA Events

Attributes	Values	Name	Mapping	Comments
I2C1				
DMA request	2	I2C1_DMA_TX	S_DMA_26	Destination: System DMA (sDMA)
		I2C1_DMA_RX	S_DMA_27	
I2C2				
DMA request	2	I2C2_DMA_TX	S_DMA_28	Destination: sDMA
		I2C2_DMA_RX	S_DMA_29	

18.3.1.7 Pin List and Pad Multiplexing With Other Functions

Table 18−5 and Table 18−6 list the pin and pad multiplexing options for the OMAP2420 device. Black cells indicate modes that are not used; gray cells indicate the pin is used for that mode; white cells indicate an alternate function.

Table 18−5. Pin Multiplexing for I^{2}C1 Module

I2C1 Interface	Description	Internal Pulldown	DIR	Ball	Alternate Functions					
					Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
I2C1.SCL	I^{2}C master serial clock. Output is open-drain.	None	IO	M19	▨	■	■	■	■	■
I2C1.SDA	I^{2}C serial bidirectional data. Output is open-drain.	None	IO	L15	▨	■	■	■	■	■

Table 18−6. Pin Multiplexing for I^{2}C2 Module

I2C1 Interface	Description	Internal Pulldown	DIR	Ball	Alternate Functions					
					Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
I2C2.SCL	I^{2}C master serial clock. Output is open-drain.	None	IO	J15	▨	■	gpt9.pwm/ evt	gpio.99	■	■
I2C2.SDA	I^{2}C serial bidirectional data. Output is open-drain.	None	IO	H19	▨	■	spi2.ncs1	gpio.100	■	■

18.3.1.8 L4 Interconnect Interface

The I^{2}C module interfaces with the L4 interconnect through a dedicated target agent (TA). The TA is part of the L4 interconnect and provides the status and configuration registers listed in Table 18−7 and Table 18−8.

By default, TA registers values are functional, but it is possible to overwrite them. See Section 18.5.1, *Main Program*, for a register field summary. For a complete description of the L4 Interconnect, see Chapter 6, *Internal Interconnect*.

Table 18−7. L4TA23 Interconnect Registers

Register Name	Type	Register Width (Bits)	Physical Address
COMPONENT	R	32	0x4807 1000
AGENT_CONTROL	RW	32	0x4807 1020
AGENT_STATUS	R	32	0x4807 1028

Table 18−8. L4TA24 Interconnect Registers

Register Name	Type	Register Width (Bits)	Physical Address
COMPONENT	R	32	0x4807 3000
AGENT_CONTROL	RW	32	0x4807 3020
AGENT_STATUS	R	32	0x4807 3028

18.3.1.9 I^{2}C Register Summary

Table 18−9 lists the base address and size for the I^{2}C module instances. Table 18−10 summarizes the I^{2}C module registers. For a detailed description of each register, see Section 18.6, *I^{2}C Registers*.

CAUTION

I^{2}C registers are limited to 16-bit data access; 32-bit and 8-bit data accesses are not allowed and can corrupt register contents.

Table 18−9. I^{2}C Module Overview

Module Name	Base Address	Size
I2C1	0x4807 0000	128 bytes
I2C2	0x4807 2000	128 bytes

Table 18−10. I^{2}C Register Overview

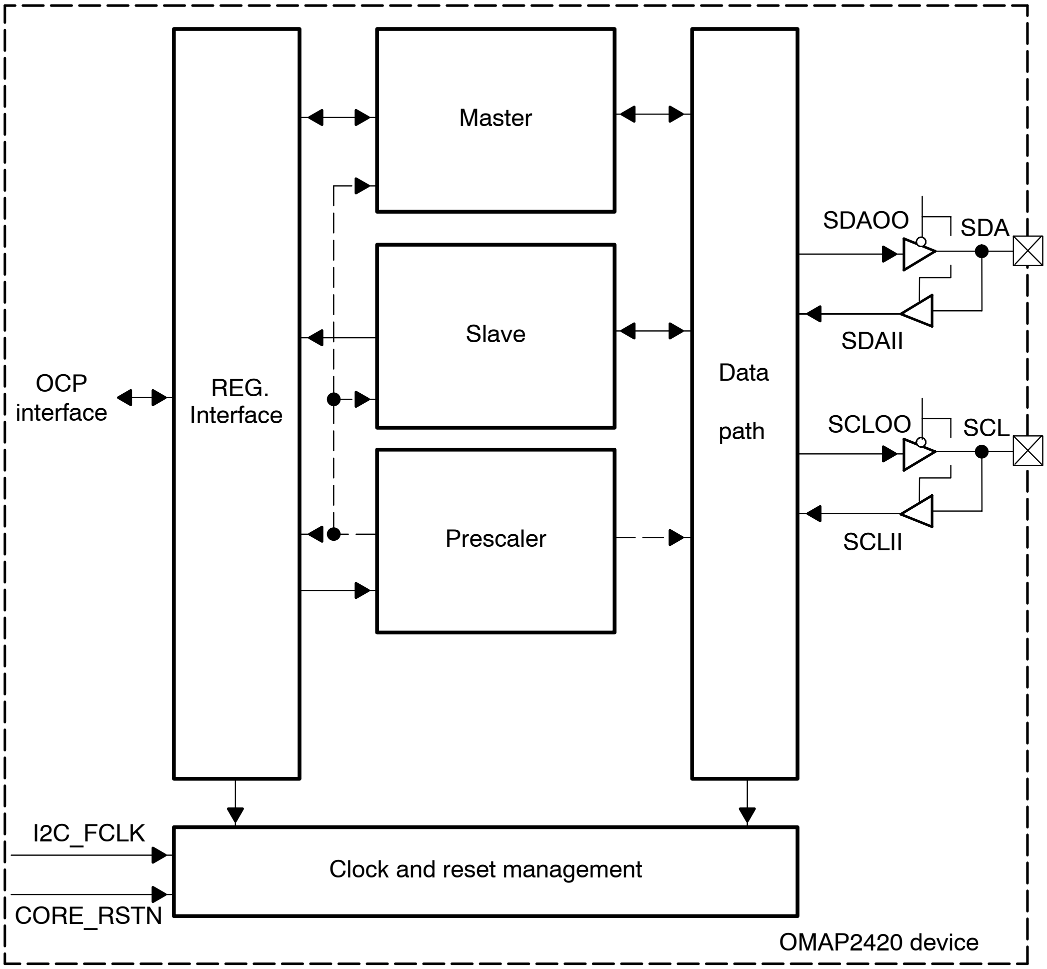
Register Name	Type	Register Width (Bits)	Physical Address
I2C_REV	RW	16	BaseAddress + 0x00
I2C_IE	RW	16	BaseAddress + 0x04
I2C_STAT	RW	16	BaseAddress + 0x08
I2C_SYSS	RW	16	BaseAddress + 0x10
I2C_BUF	RW	16	BaseAddress + 0x14
I2C_CNT	RW	16	BaseAddress + 0x18
I2C_DATA	RW	16	BaseAddress + 0x1C
I2C_SYSC	RW	16	BaseAddress + 0x20
I2C_CON	RW	16	BaseAddress + 0x24

Register Name	Type	Register Width (Bits)	Physical Address
I2C_OA	RW	16	BaseAddress + 0x28
I2C_SA	RW	16	BaseAddress + 0x2C
I2C_PSC	RW	16	BaseAddress + 0x30
I2C_SCLL	RW	16	BaseAddress + 0x34
I2C_SCLH	RW	16	BaseAddress + 0x38
I2C_SYSTEST	RW	16	BaseAddress + 0x3C

18.4 I^2C Functional Description

Figure 18−10 shows the modules included in the I^2C.

Figure 18−10. Block Diagram



18.4.1 Transmit Mode

This mode is available for either master or slave mode (both modes are configurable using the I^2C bit I2C_CON[10]) and is configured by setting the I^2C bit I2C_CON[9] to 1.

CAUTION

The I^2C register bit I2C_CON[7] must be set to 1 (MMM = 1) in multimaster mode.

The MPU subsystem uses the I2C_DATA register to fill in the FIFO with the data to transmit. The transmitter writes new data into this register while the I^2C bit I2C_STAT[4] is set to 1.

If the I2C_CNT register value is not 0, a master transmitter requests new data.

If a read is performed by the external master, a slave transmitter requests new data.

Note:

The transmitter requests 2 bytes to be written to the I2C_DATA register, even if only 1 byte is needed. In this case, the other byte must be filled with a dummy 0x00 value transmitted over the I^{2}C line.

18.4.2 Receive Mode

This mode is available for master or slave mode. It is configured by setting the I^{2}C bit I2C_CON[9] to 0.

The MPU subsystem can read new data from the I2C_DATA register when the I^{2}C bit I2C_STAT[3] is set to 1. Each time this bit is set to 1, the I^{2}C module generates an interrupt to the MPU subsystem if the interrupt is enabled (I^{2}C bit I2C_IE[3]).

Note:

In interrupt mode, the MPU subsystem must read this bit after each read to the I2C_DATA register to ensure no other data on the FIFO are waiting to be read. The I2C_STAT[3] must be cleared in the interrupt routine in order to receive a new interrupt.

The I^{2}C bit I2C_STAT[3] is not set to 1 if the DMA receive mode is enabled (bit I2C_BUF[15]). Instead, a DMA receive request is generated to the sDMA.

18.4.3 Data Format and FIFO

The FIFO size is 2*16 bits. Bytes within a word (16 bits) are stored and read in little endian format (I^{2}C bit I2C_CON[14] = 0) or big endian format (I^{2}C bit I2C_CON[14] = 1).

When read, the I2C_DATA register contains the data packet (1 or 2 bytes). This register must be accessed 16-bit-wise by the MPU subsystem. If an odd number of bytes to be read is received, the upper byte (with I^{2}C bit I2C_CON[14] = 0) or the lower byte (with I^{2}C bit I2C_CON[14] = 1) of the last access always reads as 0x00. The MPU subsystem must check the status bit I2C_STAT[15] to flush this null byte.

When written, this register contains the byte(s) value(s) to transmit over the I^{2}C data line (1or 2 bytes). This register must be accessed 16-bit-wide, except for the last byte in case of an odd number of bytes to transmit. The last byte of the data packet can be written using a byte write access or a 16-bit-wide access. In the 16-bit-wide access, only the relevant byte has been transmitted by the module based on the byte counter (I2C_CNT). This feature can be useful for DMA access, which supports only one word size per channel.

In SYSTEST loopback mode (I2C_SYSTEST[13:12] = 11), this register is also the entry/receive point for the data.

Note:

A read access when the buffer is empty returns the previous read data value. A write access when the buffer is full is ignored. In both events, the FIFO pointers are not updated and a remote access error (hardware error) is generated (access qualifier). No remote error is generated if the MPU subsystem performs a 16-bit access if the buffer contains a single byte.

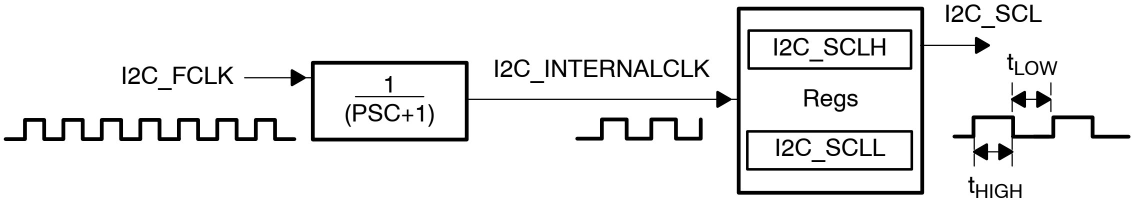
18.4.4 Clocking

The I^2C module uses a functional clock (I2C_FCLK) provided by the PRCM module. If needed, this clock is divided by a prescaler value (I2C_PSC register) to obtain a sampling clock (I2C_INTERNALCLK). This sampling clock is then used to create the SCL external clock by configuring the I2C_SCLH and I2C_SCLL registers (see Figure 18−11).

I2C_INTERNALCLK must be computed by considering the t_{high} and t_{low} values:

- ❑ I2C_INTERNALCLK time period = (t_{low} or t_{high}) /(SCLL+7) (when I2C_PSC = 0)
- ❑ I2C_INTERNALCLK = (t_{low} or t_{high})/(SCLL+6) (when I2C_PSC = 1)
- ❑ I2C_INTERNALCLK time period = (t_{low} or t_{high})/(SCLL+5) (when PSC > 1).

Figure 18−11.I^2C Clock Module



18.5 I^2C Programming Model

18.5.1 Main Program

Before enabling the module, perform the following module configuration:

Step 1: (Optional) Configure the I^2C prescaler. The I^2C FCLK is provided at 12 MHz (in most cases, this is the desirable I2C_INTERNALCLK frequency).

To change the INTERNALCLK value, program the prescaler so that I2C_PSC[PSC] + 1 = x (where x = FCLK / y and y is the desired I2C_INTERNALCLK value).

Step 2: Program the I^2C_INTERNALCLK to obtain 100 Kbps or 400 Kbps (I2C_SCLL = x and I2C_SCLH = x (where x is calculated based on the I2C_INTERNALCLK frequency).

Step 3: Configure the I^2C address (I2C_OA = x).

Step 4: Take the I^2C module out of reset (I2C_CON[15] = 1).

18.5.1.1 Initialization Procedure

Step 1: Configure the I^2C register I2C_CON (endianness, master/slave/transmit/receive).

CAUTION

The I^2C register bit I2C_CON[7] must be set to 1 (MMM = 1) in multimaster mode.

Step 2: If using an interrupt for transmit/receive data, enable interrupt masks (I2C_IE).

Step 3: If using DMA for transmit/receive data, enable the DMA (I2C_BUF) and program the DMA controller.

18.5.1.2 Configure Slave Address and Data Counter Registers

Step 1: In master mode, configure the slave address (I2C_SA = x) and the number of bytes associated with the transfer (I2C_CNT = x).

18.5.1.3 Initiate a Transfer

Step 1: Poll the bus busy (BB) bit in the I^2C status register (I2C_STAT). If it is cleared to 0 (bus not busy), configure the start/stop (I2C_CON[0]/I2C_CON[1]) condition to initiate a transfer.

18.5.1.4 Poll Receive Data

Step 1: Poll the I2Cn.I2C_STAT[3] RRDY bit, or use the RRDY interrupt (the I2Cn.I2C_IE[3] RRDY_IE bit must be set to 1) or the DMA RX channel (the I2Cn.I2C_BUF[15] RDMA_EN must be set to 1) to read the receive data in the I2Cn.I2C_DATA register.

18.5.1.5 Poll Transmit Data

Step 1: Poll the I2Cn.I2C_STAT[4] XRDY bit, or use the XRDY interrupt (the I2Cn.I2C_IE[4] XRDY_IE bit must be set to 1) or the DMA TX channel (the I2Cn.I2C_BUF[7] XDMA_EN bit must be set to 1) to write data into the I2Cn.I2C_DATA register.

18.5.2 Interrupt Subroutines Sequence

Step 1: Test for arbitration lost and resolve accordingly.

Step 2: Test for no-acknowledge and resolve accordingly.

Step 3: Test for register-access-ready and resolve accordingly.

Step 4: Test for receive data and resolve accordingly.

Step 5: Test for transmit data and resolve accordingly.

The six interrupt flag bits of the I2Cn.I2C_STAT register are updated even if the corresponding interrupt is not enabled. Thus it is possible to poll the interrupt flag bits without using interrupt generation.

Moreover, the six I^2C interrupt sources are level sensitive. From a software point of view, this means the user must ensure that any interrupt flag bit of the I2C_STAT register is set before enabling an interrupt (by setting the I2Cn_I2C_IE register). Enabling an interrupt whose flag bit is already set can generate the corresponding interrupt.

18.5.3 Programming Flow Diagrams

Figure 18−12 through Figure 18−20 are diagrams to help in programming the I²C modes.

Figure 18−12. Setup Procedure

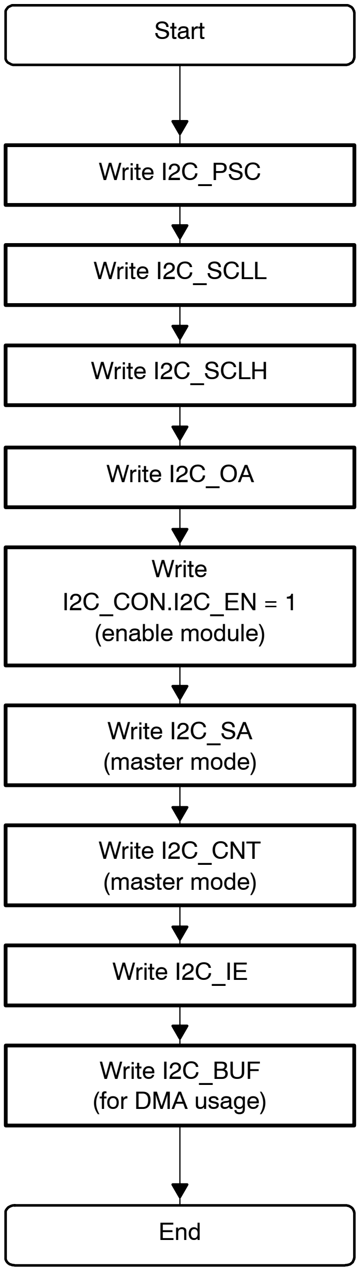
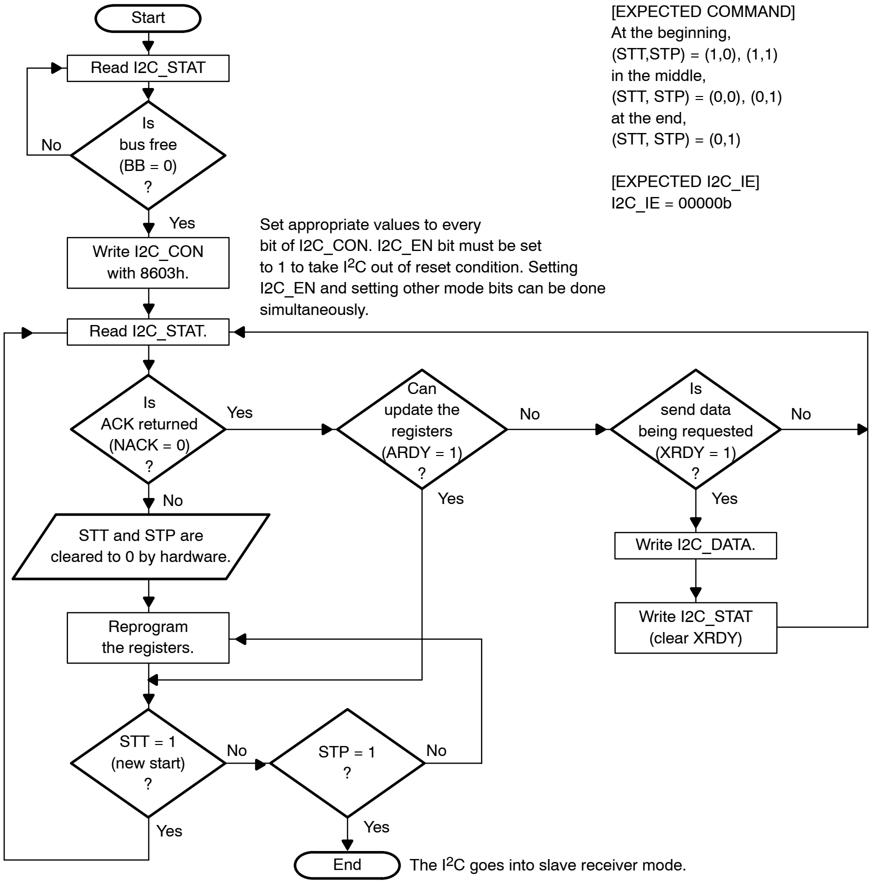

Start
Write I2C_PSC
Write I2C_SCLL
Write I2C_SCLH
Write I2C_OA
Write I2C_CON.I2C_EN = 1 (enable module)
Write I2C_SA (master mode)
Write I2C_CNT (master mode)
Write I2C_IE
Write I2C_BUF (for DMA usage)
End

Figure 18−13. Master Transmitter Mode, Polling


Start
Read I2C_STAT
No
Is bus free (BB = 0) ?
Yes
Write I2C_CON with 8603h.
Read I2C_STAT.
Is ACK returned (NACK = 0) ?
Yes
Can update the registers (ARDY = 1) ?
No
Is send data being requested (XRDY = 1) ?
No
Yes
Write I2C_DATA.
Write I2C_STAT (clear XRDY)
Yes
No
STT and STP are cleared to 0 by hardware.
Reprogram the registers.
STT = 1 (new start) ?
No
STP = 1 ?
No
Yes
Yes
End

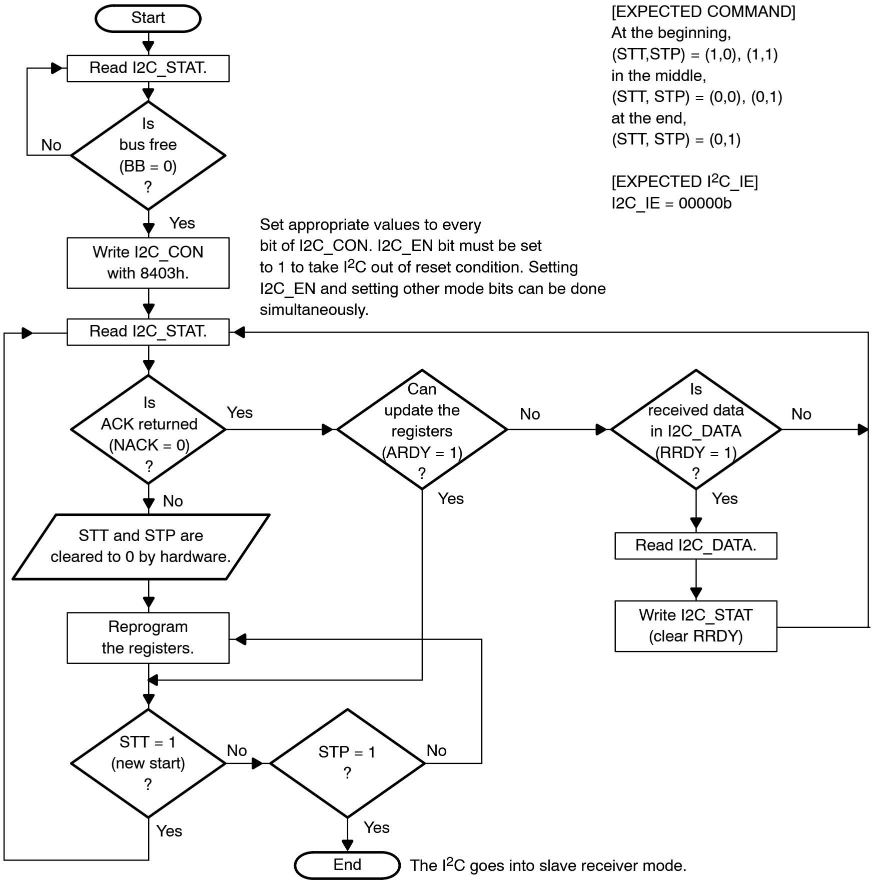
[EXPECTED COMMAND]
At the beginning,
(STT,STP) = (1,0), (1,1)
in the middle,
(STT, STP) = (0,0), (0,1)
at the end,
(STT, STP) = (0,1)

[EXPECTED I2C_IE]
I2C_IE = 00000b

Set appropriate values to every bit of I2C_CON. I2C_EN bit must be set to 1 to take I^2C out of reset condition. Setting I2C_EN and setting other mode bits can be done simultaneously.

The I^2C goes into slave receiver mode.

Figure 18−14. Master Receiver Mode, Polling


Start
Read I2C_STAT.
Is bus free (BB = 0) ?
No
Yes
Write I2C_CON with 8403h.
Read I2C_STAT.
Is ACK returned (NACK = 0) ?
Yes
No
Can update the registers (ARDY = 1) ?
No
Yes
Is received data in I2C_DATA (RRDY = 1) ?
No
Yes
Read I2C_DATA.
Write I2C_STAT (clear RRDY)
STT and STP are cleared to 0 by hardware.
Reprogram the registers.
STT = 1 (new start) ?
No
Yes
STP = 1 ?
No
Yes
End

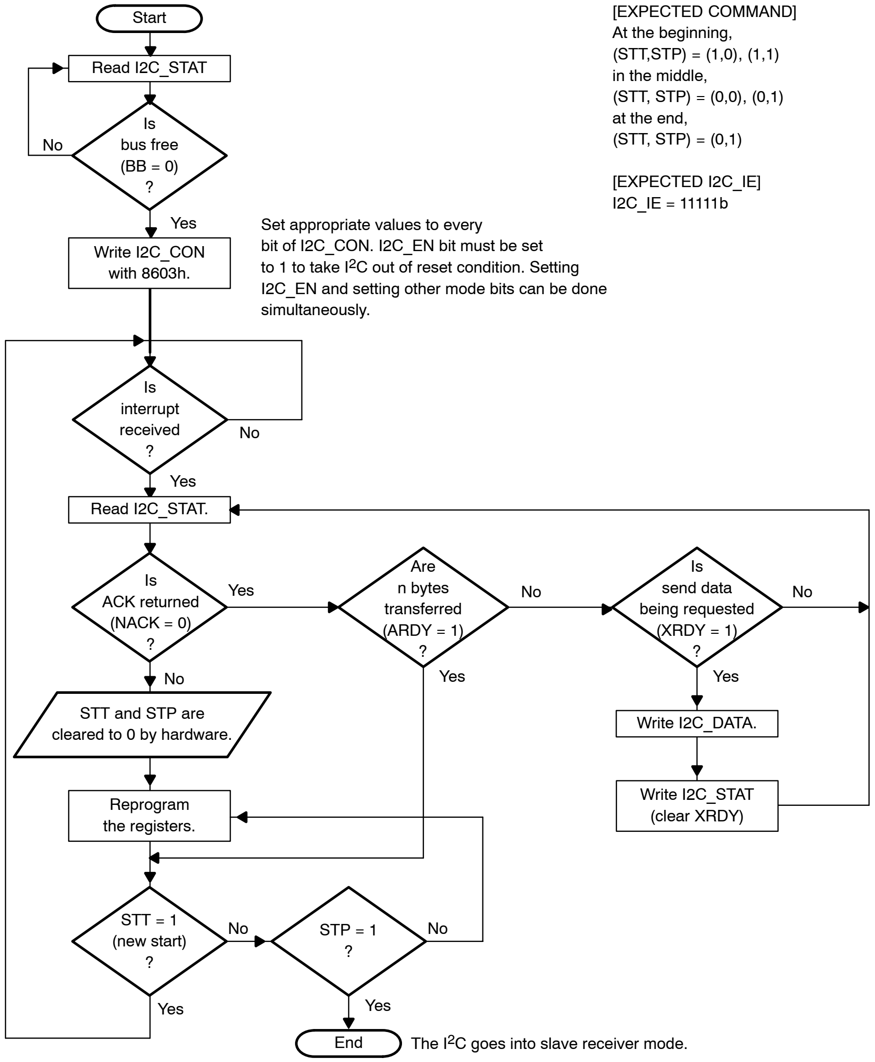
[EXPECTED COMMAND]
At the beginning,
(STT,STP) = (1,0), (1,1)
in the middle,
(STT, STP) = (0,0), (0,1)
at the end,
(STT, STP) = (0,1)

[EXPECTED I^{2}C_IE]
I2C_IE = 00000b

Set appropriate values to every bit of I2C_CON. I2C_EN bit must be set to 1 to take I^{2}C out of reset condition. Setting I2C_EN and setting other mode bits can be done simultaneously.

The I^{2}C goes into slave receiver mode.

Figure 18−15. Master Transmitter Mode, Interrupt


Start
Read I2C_STAT
Is bus free (BB = 0) ?
No
Yes
Write I2C_CON with 8603h.
Is interrupt received ?
No
Yes
Read I2C_STAT.
Is ACK returned (NACK = 0) ?
Yes
No
Are n bytes transferred (ARDY = 1) ?
No
Yes
Is send data being requested (XRDY = 1) ?
No
Yes
Write I2C_DATA.
Write I2C_STAT (clear XRDY)
STT and STP are cleared to 0 by hardware.
Reprogram the registers.
STT = 1 (new start) ?
No
Yes
STP = 1 ?
No
Yes
End

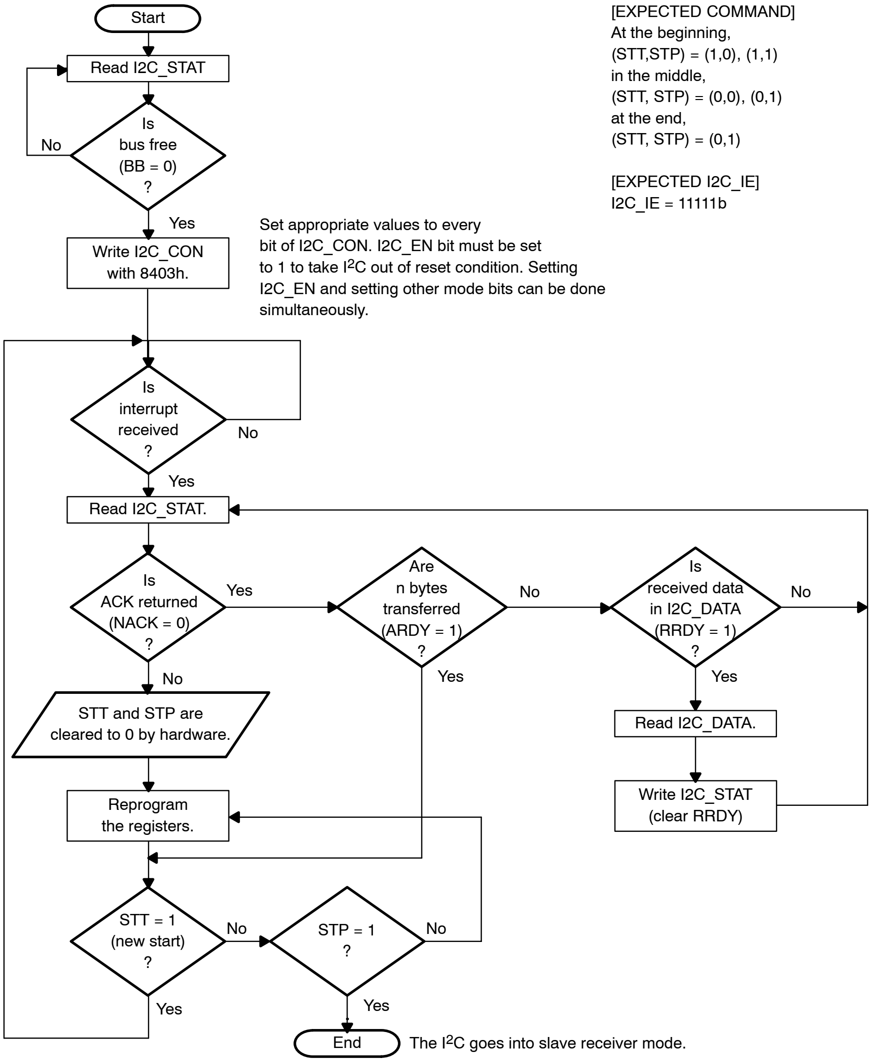
[EXPECTED COMMAND]
At the beginning,
(STT,STP) = (1,0), (1,1)
in the middle,
(STT, STP) = (0,0), (0,1)
at the end,
(STT, STP) = (0,1)

[EXPECTED I2C_IE]
I2C_IE = 11111b

Set appropriate values to every bit of I2C_CON. I2C_EN bit must be set to 1 to take I²C out of reset condition. Setting I2C_EN and setting other mode bits can be done simultaneously.

The I²C goes into slave receiver mode.

Figure 18−16. Master Receiver Mode, Interrupt


Start
Read I2C_STAT
Is bus free (BB = 0) ?
No
Yes
Write I2C_CON with 8403h.
Is interrupt received ?
No
Yes
Read I2C_STAT.
Is ACK returned (NACK = 0) ?
Yes
No
Are n bytes transferred (ARDY = 1) ?
No
Yes
Is received data in I2C_DATA (RRDY = 1) ?
No
Yes
Read I2C_DATA.
Write I2C_STAT (clear RRDY)
STT and STP are cleared to 0 by hardware.
Reprogram the registers.
STT = 1 (new start) ?
No
Yes
STP = 1 ?
No
Yes
End

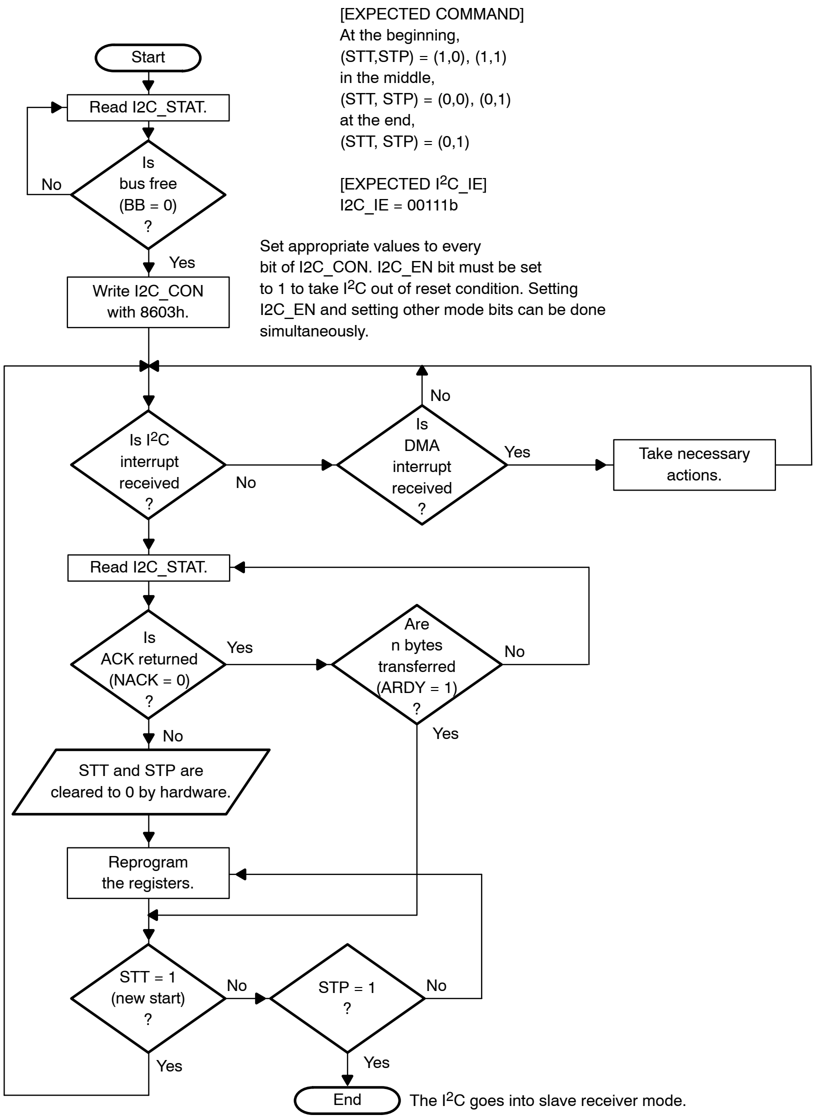
[EXPECTED COMMAND]
At the beginning,
(STT,STP) = (1,0), (1,1)
in the middle,
(STT, STP) = (0,0), (0,1)
at the end,
(STT, STP) = (0,1)

[EXPECTED I2C_IE]
I2C_IE = 11111b

Set appropriate values to every bit of I2C_CON. I2C_EN bit must be set to 1 to take I^2C out of reset condition. Setting I2C_EN and setting other mode bits can be done simultaneously.

The I^2C goes into slave receiver mode.

Figure 18−17. Master Transmitter Mode, DMA


Start
Read I2C_STAT.
Is bus free (BB = 0) ?
No
Yes
Write I2C_CON with 8603h.
Is I^{2}C interrupt received ?
No
Is DMA interrupt received ?
No
Yes
Take necessary actions.
Read I2C_STAT.
Is ACK returned (NACK = 0) ?
Yes
Are n bytes transferred (ARDY = 1) ?
No
Yes
No
STT and STP are cleared to 0 by hardware.
Reprogram the registers.
STT = 1 (new start) ?
No
Yes
STP = 1 ?
No
Yes
End

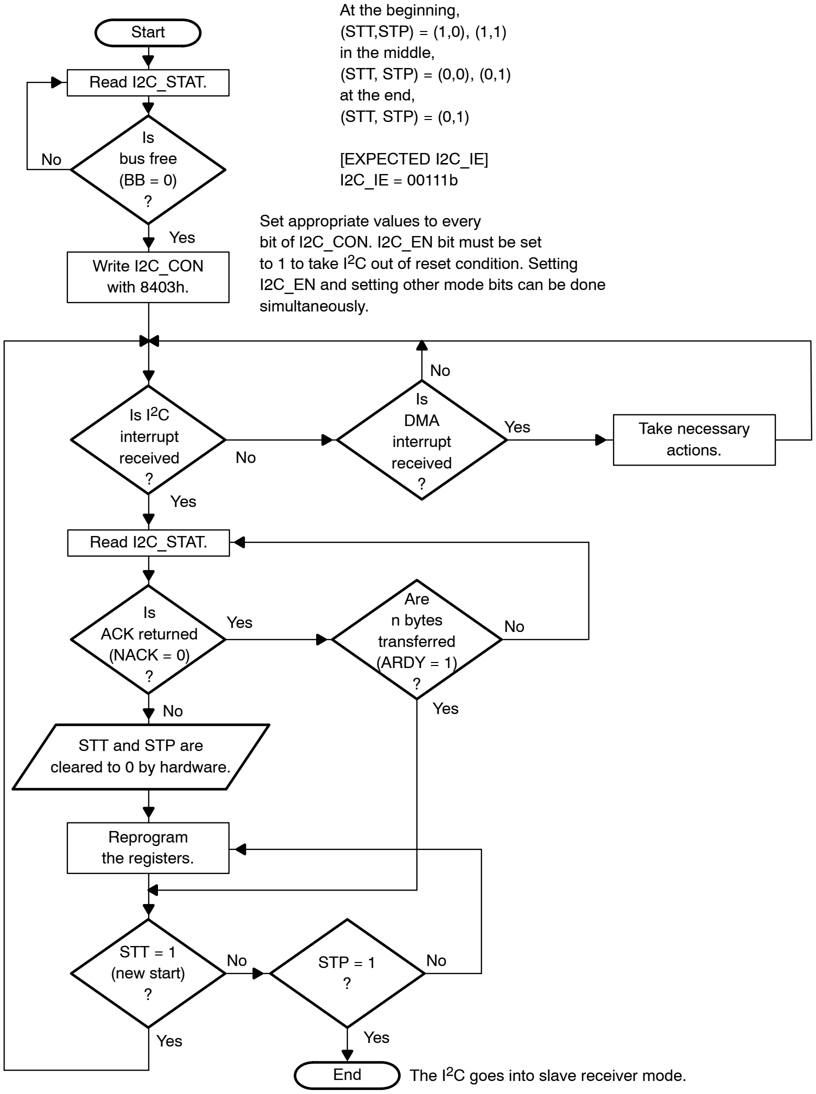
[EXPECTED COMMAND]
At the beginning,
(STT,STP) = (1,0), (1,1)
in the middle,
(STT, STP) = (0,0), (0,1)
at the end,
(STT, STP) = (0,1)

[EXPECTED I^{2}C_IE]
I2C_IE = 00111b

Set appropriate values to every bit of I2C_CON. I2C_EN bit must be set to 1 to take I^{2}C out of reset condition. Setting I2C_EN and setting other mode bits can be done simultaneously.

The I^{2}C goes into slave receiver mode.

Figure 18−18. Master Receiver Mode, DMA


Start
Read I2C_STAT.
Is bus free (BB = 0) ?
No
Yes
Write I2C_CON with 8403h.
Is I^2C interrupt received ?
No
Yes
Is DMA interrupt received ?
No
Yes
Take necessary actions.
Read I2C_STAT.
Is ACK returned (NACK = 0) ?
Yes
No
Are n bytes transferred (ARDY = 1) ?
No
Yes
STT and STP are cleared to 0 by hardware.
Reprogram the registers.
STT = 1 (new start) ?
No
Yes
STP = 1 ?
No
Yes
End

[EXPECTED COMMAND]
At the beginning,
(STT,STP) = (1,0), (1,1)
in the middle,
(STT, STP) = (0,0), (0,1)
at the end,
(STT, STP) = (0,1)

[EXPECTED I2C_IE]
I2C_IE = 00111b

Set appropriate values to every bit of I2C_CON. I2C_EN bit must be set to 1 to take I^2C out of reset condition. Setting I2C_EN and setting other mode bits can be done simultaneously.

The I^2C goes into slave receiver mode.

Figure 18−19. Slave Transmitter/Receiver Mode, Polling

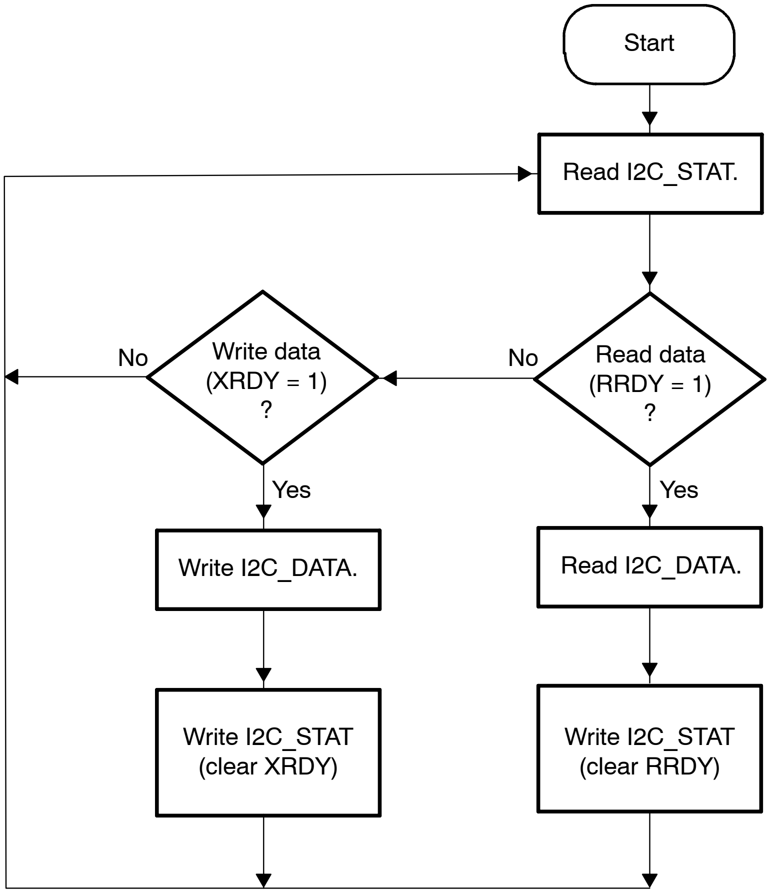
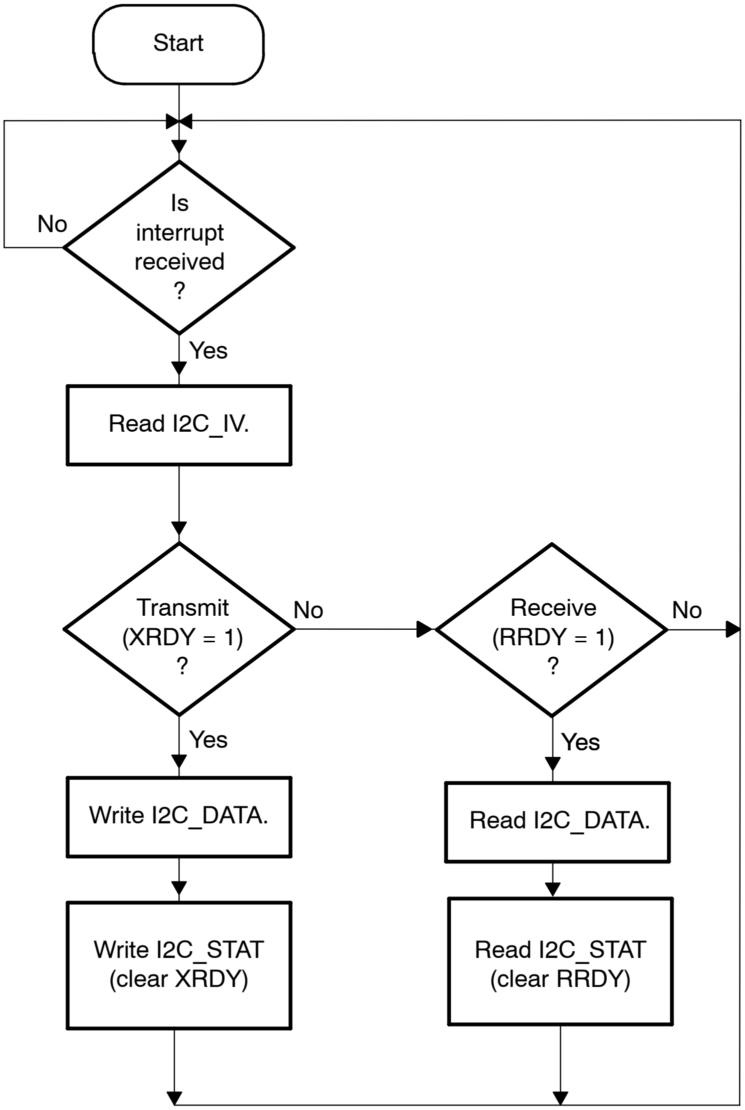


Figure 18−20. Slave Transmitter/Receiver Mode, Interrupt



18.6 I^{2}C Registers

CAUTION

I^{2}C registers are limited to 16-bit access; 32-bit and 8-bit data accesses can corrupt register content and are not allowed.

18.6.1 Register Mapping Summary

This section provides information on the I^{2}C module instance in the OMAP2420 device. Each register within the module instance is described separately in this section.

Table 18−11 provides the base address and size for the instances in the I^{2}C module.

Table 18−12 summarizes the I^{2}C offset register mapping.

Table 18−11. Instance Summary

Module Name	Base Address	Size
I2C1	0x4807 0000	4K bytes
I2C2	0x4807 2000	4K bytes

Table 18−12. I^{2}C Offset Register Mapping Summary

Register Name	Type	Register Width (Bits)	Physical Address
I2C_REV	RW	16	0x00
I2C_IE	RW	16	0x04
I2C_STAT	RW	16	0x08
I2C_SYSS	RW	16	0x10
I2C_BUF	RW	16	0x14
I2C_CNT	RW	16	0x18
I2C_DATA	RW	16	0x1C
I2C_SYSC	RW	16	0x20
I2C_CON	RW	16	0x24
I2C_OA	RW	16	0x28
I2C_SA	RW	16	0x2C
I2C_PSC	RW	16	0x30
I2C_SCLL	RW	16	0x34
I2C_SCLH	RW	16	0x38
I2C_SYSTEST	RW	16	0x3C

Note: All bits defined as reserved must be set to 0 to preserve future compatibility. When read, any reserved bit returns 0. Also, it is good software practice to use complete mask patterns when setting or testing bit fields individually within a register.

18.6.2 Register Descriptions

Table 18−13 through Table 18−27 describe the individual register bits.

Table 18−13. I2C_REV

Address Offset	0x00		
Physical Address	0x4807 0000	**Instance**	I2C1
	0x4807 2000		I2C2
Description	Module revision register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								REV							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reads return 0.	R	0x00
7:0	REV	IP revision [7:4] Major revision [3:0] Minor revision Examples: 0x10 for 1.0, 0x21 for 2.1	R	0x−−

Table 18−14. I2C_IE

Address Offset	0x04		
Physical Address	0x4807 0004	**Instance**	I2C1
	0x4807 2004		I2C2
Description	I²C Interrupt enable register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										GC_IE	XRDY_IE	RRDY_IE	ARDY_IE	NACK_IE	AL_IE

Table 18−14. I2C_IE (Continued)

Bits	Field Name	Description	Type	Reset
15:6	Reserved	Write 0x for future compatibility. Reads return 0.	R	0x000
5	GC_IE	General call_interrupt enable. Mask or unmask the interrupt signaled by 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 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 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 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 bit in I2C_STAT[NACK]. 0x0: Not acknowledge interrupt disabled 0x1: Not acknowledge interrupt enabled	RW	0
0	AL_IE	Arbitration lost interrupt enable. Mask or unmask the interrupt signaled by bit in I2C_STAT[AL]. 0x0: Arbitration lost interrupt disabled 0x1: Arbitration lost interrupt enabled		

Table 18−15. I2C_STAT

Address Offset	0x08		
Physical Address	0x4807 0008	**Instance**	I2C1
	0x4807 2008		I2C2
Description	I^2C status register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SBD	Reserved		BB	ROVER	XUDF	AAS	Reserved			GC	XRDY	RRDY	ARDY	NACK	AL

Table 18−15. I2C_STAT (Continued)

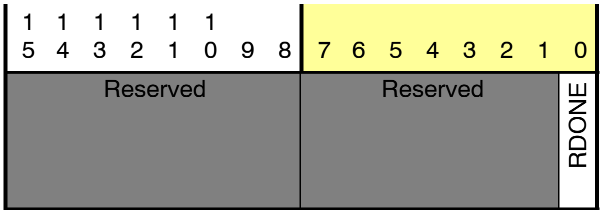
Bits	Field Name	Description	Type	Reset
15	SBD	Single byte data status 0x0: No action 0x1: Single valid byte in last 16-bit data read	R	0
14:13	Reserved	Write 0s for future compatibility. Read returns 0.	R	0x0
12	BB	Bus busy status 0x0: Bus is free. 0x1: Bus is occupied.	R	0
11	ROVR	Receive overrun status 0x0: Normal operation 0x1: Receiver overrun	R	0
10	XUDF	Transmit underflow status 0x0: Normal operation 0x1: Transmit underflow	R	0
9	AAS	Address recognized as slave status 0x0: No action 0x1: Address recognized	R	0
8:6	Reserved	Write 0s for future compatibility. Read returns 0.	R	0x0
5	GC	General call status. Set to 1 by core when general call address detected and interrupt signaled to MPUSS. Write 1 to clear. 0x0: No general call detected 0x1: General call address detected	RW	0
4	XRDY	Transmit data ready status. Set to 1 by core in transmitter mode when new data is requested. When set to 1 by core, an interrupt is signaled to MPUSS. 0x0: Transmission ongoing 0x1: Transmit data ready	RW	0
3	RRDY	Receive data ready status. Set to 1 by the core in receiver mode when new data can be read. When set to 1 by the core, an interrupt is signaled to MPUSS. 0x0: No data available 0x1: Receive data available	RW	0
2	ARDY	Register access ready status. When set to 1, ARDY indicates that access has been performed and registers are ready to be accessed. An interrupt is signaled to MPUSS. 0x0: Module busy 0x1: Access ready	RW	0

Table 18−15. I2C_STAT (Continued)

Bits	Field Name	Description	Type	Reset
1	NACK	No acknowledgement status. Bit is set when NACK is received; an interrupt is signaled to MPUSS. Write 1 to clear this bit. 0x0: Normal operation 0x1: Not Acknowledge detected	RW	0
0	AL	Arbitration lost status. This bit is automatically set by the hardware when it loses the arbitration in master transmit mode; an interrupt is signaled to MPUSS. During reads, it always returns 0. 0x0: Normal operation 0x1: Arbitration lost detected	RW	0

Table 18−16. I2C_SYSS

Address Offset	0x10		
Physical Address	0x4807 0010	**Instance**	I2C1
	0x4807 2010		I2C2
Description	This register provides status information about the module, excluding the interrupt status information.		
Type	R		



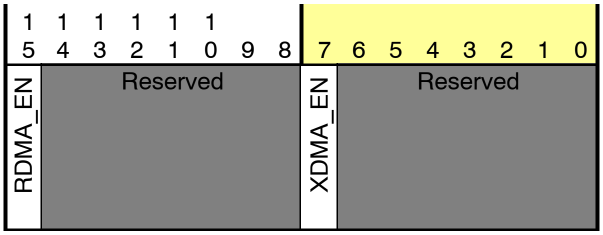
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								Reserved							RDONE

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0.	R	0x00
7:1	Reserved	Read returns 0. Reserved for OCP socket status information	R	0x00
0	RDONE	Internal reset monitoring 0x0: Internal module reset is ongoing. 0x1: Reset completed (see note)	R	0

Note: During reset the value is 0, but the value read just after the reset is 1 because ICLK/FCLK is already operating.

Table 18−17. I2C_BUF

Address Offset	0x14		
Physical Address	0x4807 0014	**Instance**	I2C1
	0x4807 2014		I2C2
Description	Receive DMA channel disabled		
Type	RW		

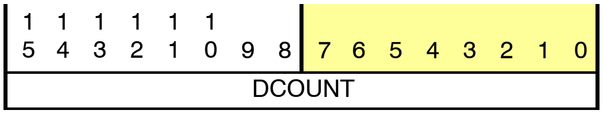


15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDMA_EN	Reserved							XDMA_EN	Reserved						

Bits	Field Name	Description	Type	Reset
15	RDMA_EN	Receive DMA channel enable 0x0: Receive DMA channel disabled 0x1: Receive DMA channel enabled	RW	0
14:8	Reserved	Read returns 0	R	0x00
7	XDMA_EN	Transmit DMA channel enable 0x0: Transmit DMA channel disabled 0x1: Transmit DMA channel enabled	RW	0
6:0	Reserved	Read returns 0.	R	0x00

Table 18−18. I2C_CNT

Address Offset	0x18		
Physical Address	0x4807 0018	**Instance**	I2C1
	0x4807 2018		I2C2
Description	This read/write register is used to control the numbers of bytes in the I^{2}C data payload.		
Type	RW		

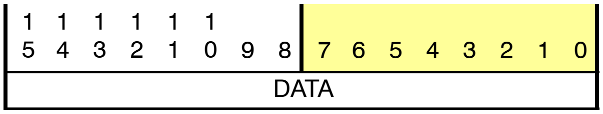


15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCOUNT															

Bits	Field Name	Description	Type	Reset
15:0	DCOUNT	Data count	RW	0x0000

Table 18−19. I2C_DATA

Address Offset	0x1C		
Physical Address	0x4807 001C	**Instance**	I2C1
	0x4807 201C		I2C2
Description	This register is the entry point for the local host to read data from or write data to the FIFO buffer.		
Type	RW		

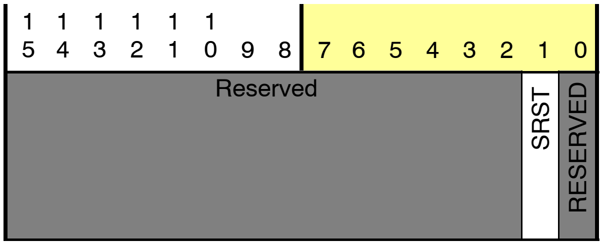


Bits	Field Name	Description	Type	Reset
15:0	DATA	Transmit/receive FIFO data	RW	†

† Attempting to read the I2C_DATA register after reset generates the hardware remote error, *access to buffer empty*.

Table 18−20. I2C_SYSC

Address Offset	0x20		
Physical Address	0x4807 0020	**Instance**	I2C1
	0x4807 2020		I2C2
Description	This register controls the various parameters of the OCP interface.		
Type	RW		



Bits	Field Name	Description	Type	Reset
15:2	Reserved	Write 0s for future compatibility. Read returns 0.	R	0x0000
1	SRST	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	Reserved	Write 0s for future compatibility. Read returns 0.	R	0

Table 18−21. I2C_CON

Address Offset	0x24		
Physical Address	0x4807 0024	**Instance**	I2C1
	0x4807 2024		I2C2
Description	This register controls the functional features. **Caution:** This register must not be modified during an active transfer phase (STT is set to 1). Modification can result in unpredictable behavior.		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I2C_EN	BE	Reserved		STB	MST	TRX	XA	MMM	Reserved					STP	STT

Bits	Field Name	Description	Type	Reset
15	I2C_EN	Module enable 0x0: Controller in reset. FIFO is cleared and status bits are set to their default value. 0x1: Module enabled	RW	0
14	BE	Big endian mode 0x0: Little endian mode. LS byte is transmitted first in transmit mode. In receive mode, the first byte is stored in the LS byte position. 0x1: Big endian mode	RW	0
13:12	Reserved	Read returns 0.	R	0x0
11	STB	Start byte mode (master mode only) 0x0: Normal mode 0x1: Start byte mode	RW	0
10	MST	Master/slave mode 0x0: Slave mode 0x1: Master mode	RW	0
9	TRX	Transmitter/receiver mode (master mode only) 0x0: Receiver mode 0x1: Transmitter mode	RW	0
8	XA	Expand address 0x0: 7-bit address mode 0x1: 10-bit address mode	RW	0
7	MMM	Multimaster mode 0x0: No action 0x1: Enables the multimaster mode	RW	0

Table 18−21. I2C_CON (Continued)

Bits	Field Name	Description	Type	Reset
6:2	Reserved	Read returns 0.	R	0x00
1	STP	Stop condition (master mode only) 0x0: No action or stop condition detected 0x1: Stop condition queried	RW	0
0	STT	Start condition (master mode only)	RW	0

Table 18−22. I2C_OA

Address Offset	0x28		
Physical Address	0x4807 0028	**Instance**	I2C1
	0x4807 2028		I2C2
Description	This register is used to specify the module I^{2}C 7-bit or 10-bit address.		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						OA									

Bits	Field Name	Description	Type	Reset
15:10	Reserved	Write 0s for future compatibility Read returns 0.	R	0x00
9:0	OA	Own address	RW	0x000

Table 18−23. I2C_SA

Address Offset	0x2C		
Physical Address	0x4807 002C	**Instance**	I2C1
	0x4807 202C		I2C2
Description	This register is used to specify the addressed I^{2}C module 7-bit or 10-bit address.		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						SA									

Bits	Field Name	Description	Type	Reset
15:10	Reserved	Write 0s for future compatibility. Read returns 0.	R	0x00
9:0	SA	Slave address	RW	0x3FF

Table 18−24. I2C_PSC

Address Offset	0x30		
Physical Address	0x4807 0030	**Instance**	I2C1
	0x4807 2030		I2C2
Description	This register is used to specify the internal clocking of the I^{2}C peripheral core. The core logic is sampled at the clock rate of the system clock for the module divided by (PSC+1).		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								PSC							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Write 0s for future compatibility. Read returns 0.	R	0x00
7:0	PSC	Fast/standard mode prescale-sampling clock divider value	RW	0x00

Table 18−25. I2C_SCLL

Address Offset	0x34		
Physical Address	0x4807 0034	**Instance**	I2C1
	0x4807 2034		I2C2
Description	This register determines the SCL low time value when the peripheral is in master mode.		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								SCLL							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Write 0s for future compatibility. Read returns 0.	R	0x00
7:0	SCLL	Fast/standard mode SCL low time	RW	0x00

Table 18−26. I2C_SCLH

Address Offset	0x38		
Physical Address	0x4807 0038	**Instance**	I2C1
	0x4807 2038		I2C2
Description	This register determines the SCL high time value when the peripheral is in master mode.		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								SCLH							

Table 18–26. I2C_SCLH (Continued)

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Write 0s for future compatibility. Read returns 0.	R	0x00
7:0	SCLH	Fast/standard mode SCL high time	RW	0x00

Table 18–27. I2C_SYSTEST

Address Offset	0x3C		
Physical Address	0x4807 003C	**Instance**	I2C1
	0x4807 203C		I2C2
Description	This register facilitates system-level tests by overriding some standard functional features of the peripheral.		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ST_EN	FREE	TMODE		SSB	Reserved							SCL_I	SCL_O	SDA_I	SDA_O

Bits	Field Name	Description	Type	Reset
15	ST_EN	System test enable 0x0: Normal mode 0x1: System test enabled. Permit other system test register bits to be set.	RW	0
14	FREE	Free-running mode 0x0: Stop mode (on breakpoint condition). In 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	RW	0
13:12	TMODE	Test-mode select 0x0: Functional mode (default) 0x1: Reserved 0x2: Test of SCL counters (SCLL, SCLH, PSC). SCL provides a permanent clock with master mode. 0x3: Loopback mode select + SDA/SCL IO mode select	RW	0x0
11	SSB	Set status bits 0x0: No action 0x1: Set all interrupt status bits to 1.	RW	0
10:4	Reserved	Write 0s for future compatibility. Read returns 0.	R	0x00

Table 18−27. I2C_SYSTEST (Continued)

Bits	Field Name	Description		Type	Reset
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-sense output value		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-sense output value		RW	0
		0x0:	Write 0 to SDA line.		
		0x1:	Write 1 to SDA line.		

Chapter 19

Multichannel SPI

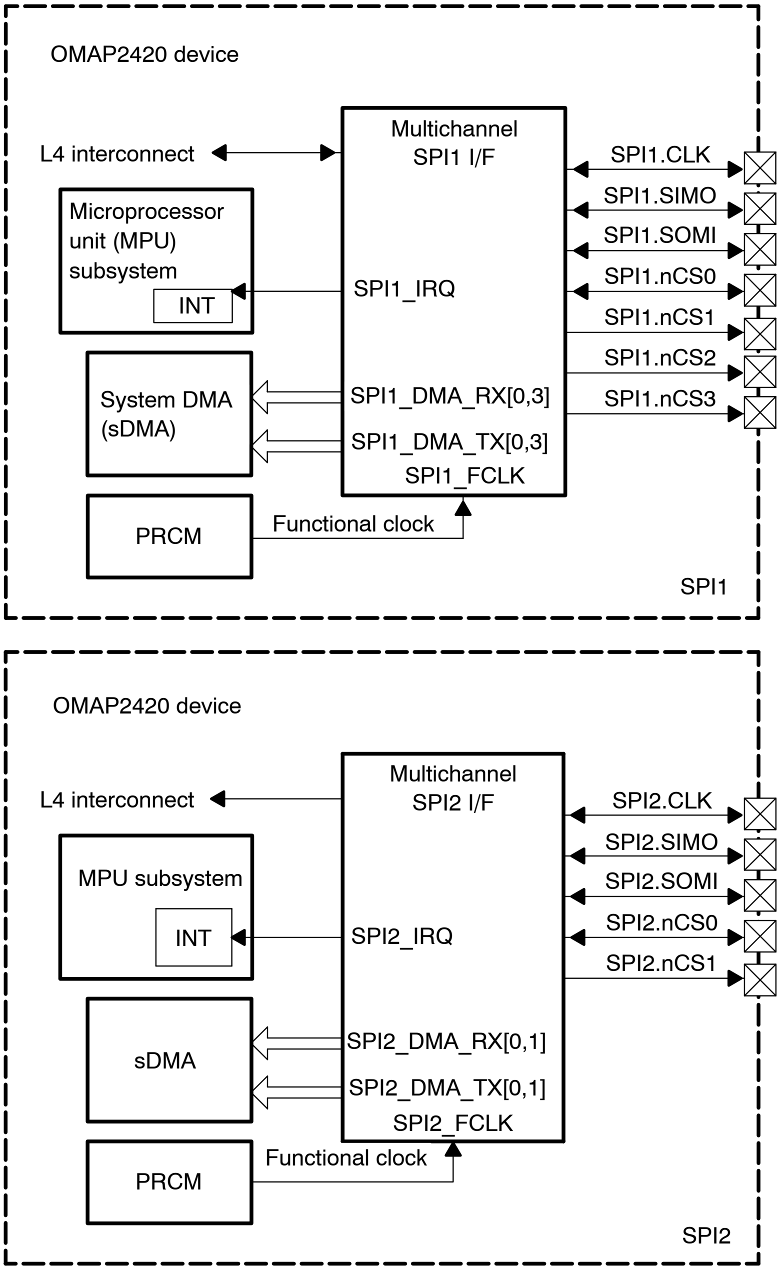
This chapter describes the two multichannel SPI (McSPI) modules in the OMAP2420 device.

Topic **Page**

19.1 McSPI Overview

The multichannel SPI (McSPI) is a master/slave synchronous serial bus. There are two separate McSPI modules, SPI1 and SPI2, in the OMAP2420 device (see Figure 19−1). SPI1 supports up to four peripherals and SPI2 supports up to two peripherals. Figure 19−1 shows the McSPI modules.

Figure 19−1. Multichannel Modules SPI1 and SPI2



McSPI modules have the following features:

- Serial clock with programmable frequency, polarity, and phase for each channel
- Wide selection of SPI word lengths ranging from 4 to 32 bits
- Up to four master channels or a single channel in slave mode
- Master multichannel mode:
  - Full duplex
  - Transmit only/receive only/transmit and receive modes
  - Flexible input/output (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

19.2 McSPI Environment

Figure 19−2 through Figure 19−7 show master mode and slave mode configurations. Each mode can be wired on single or full-duplex configuration.

19.2.1 SPI Interface in Master Mode

Figure 19−2 and Figure 19−3 show two cases in master mode.

Figure 19−2. McSPI Master Mode (Full-Duplex)

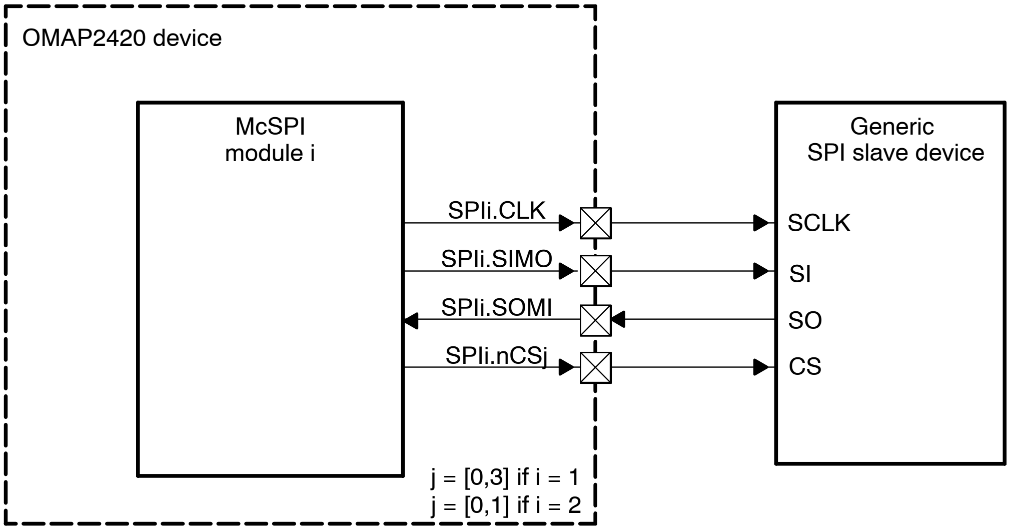
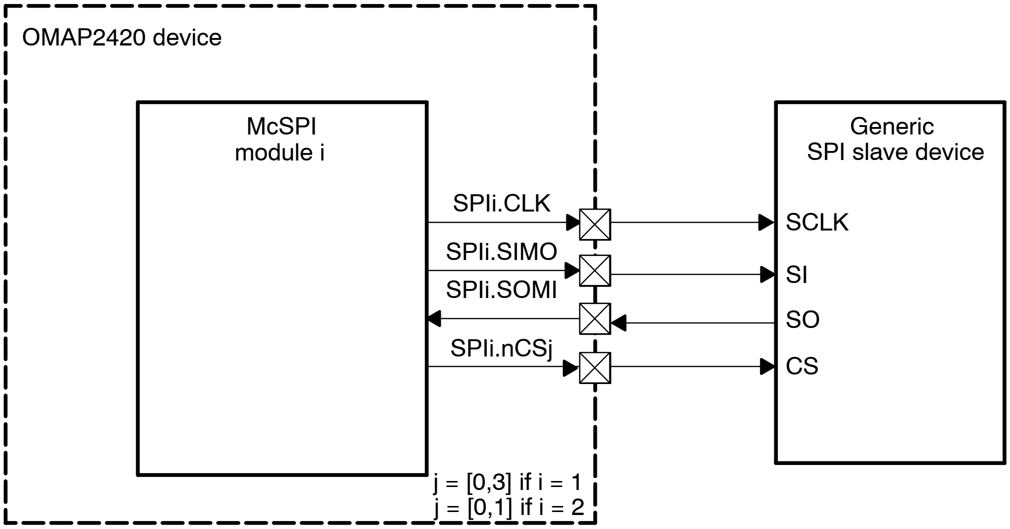


Figure 19−3 shows the master single mode, which can also be configured in receive-only mode.

Figure 19−3. McSPI Master Single Mode (Receive-Only)



19.2.2 SPI Interface in Slave Mode

Figure 19−4 and Figure 19−5 show two cases in slave mode.

Note:

Only channel 0 can be configured as slave, but the chip-enable signal can be connected to any of the SPIi.nCSj pins and then rerouted internally to channel 0 (the MCSPIn_CHCONF0 register).

Figure 19−4. McSPI Slave Mode (Full Duplex)

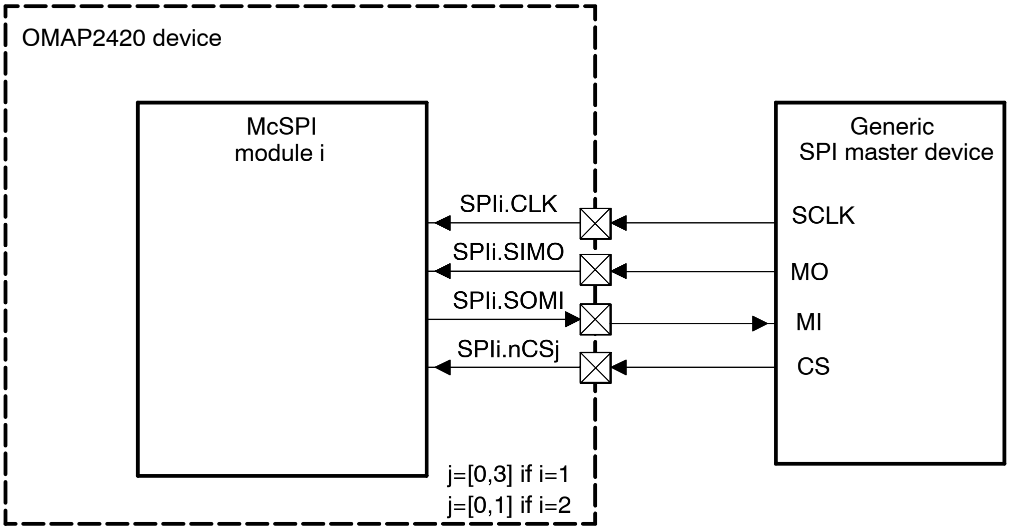
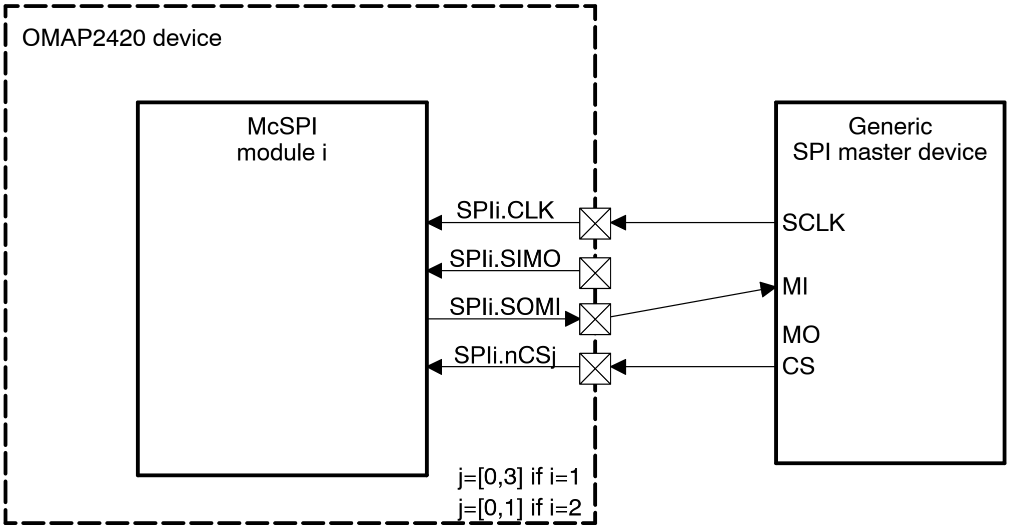


Figure 19−5 shows the slave single mode, which can also be configured in transmit-only mode.

Figure 19−5. McSPI Slave Single Mode (Transmit Only)



19.3 McSPI Functional Interface

19.3.1 Basic McSPI Pins for Master Mode

Figure 19−6 shows all possible interfaces of McSPI in master mode.

Figure 19−6. McSPI Interface Signals in Master Mode

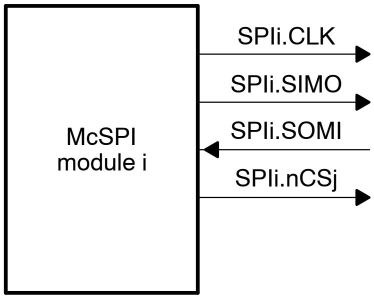


Table 19−1 lists McSPI master I/Os.

Table 19−1. McSPI I/O Description (Master Mode)

Signal Name	I/O[1]	Description	Reset
SPIi.CLK	O	SPIi serial clock	Unknown
SPIi.SIMO	O	SPIi serial data master out	Unknown
SPIi.SOMI	I	SPIi serial data master in	
SPIi.nCSj	O	SPIi chip-select i output	Low

1) I = Input, O = Output

19.3.1.1 Basic McSPI Pins for Slave Mode

Figure 19−7 shows all possible interfaces of McSPI in slave mode.

Figure 19−7. McSPI Interface Signals in Slave Mode

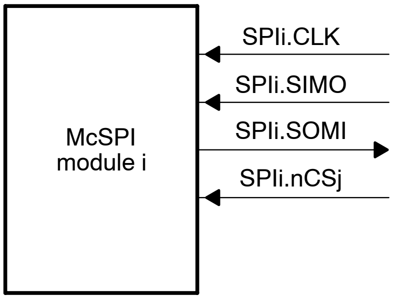


Table 19−2 lists McSPI slave I/Os.

Table 19−2. McSPI I/O Description (Slave Mode)

Signal Name	I/O[1]	Description	Reset
SPIi.CLK	I	SPIi serial clock	Unknown
SPIi.SIMO	I	SPIi serial data slave in	Unknown

Signal Name	I/O[1]	Description	Reset
SPIi.SOMI	O	SPIi serial data slave out	
SPIi.nCSj	I	SPIi chip-select i input	Low

1) I = Input, O = Output

19.3.1.2 McSPI Protocol and Data Format

The SPI protocol is a synchronous protocol that allows a master device to initiate serial data transfers to a slave device. A slave select line (SPIi.nCSj) allows selection of an individual slave SPI device. Slave devices that are not selected do not interfere with SPI bus activities.

McSPI offers flexibility to modify the following parameters to adapt to device features:

- ☐ Word length

  McSPI supports any SPI word ranging from 4 to 32 bits long (bits MCSPI_CHCONFi[11:7]).

  An SPI word can be changed between transmissions to let the master device communicate with peripheral slaves that have different requirements.

- ☐ SPI enable (SPIi.nCSj)

  The polarity of the SPI enable signals is programmable (bit MCSPI_CHCONFi[6]). The SPIi.nCSj signals can be active high or low.

  The assertion of the SPIi.nCSj signals is programmable. It can be manually or automatically asserted. Manual assertion mode is available in single master mode only. SPIi.nCSj can be kept active between words with MCSPI_CHCONFi[20].

  Two consecutive words for two different slave devices can go along with active SPIi.nCSj signals with different polarity (see the example in Section 19.6, *McSPI Initialization Sequence Example*).

- ☐ Programmable SPI clock

  - ■ Bit rate

    In master mode, the baud rate of the SPI serial clock is programmable using the 48-MHz reference clock SPIi_FCLK (internal clock).

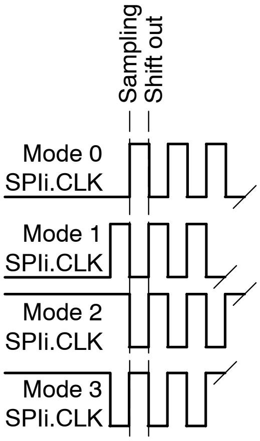
  - ■ Polarity and phase

    The polarity (MCSPI_CHCONFi[1]) and the phase (MCSPI_CHCONFi[0]) of the SPI serial clock (SPIi.CLK) are configurable to offer four combinations. Software selects the correct combination for the device. See Table 19−3 and Figure 19−8.

Table 19−3. Phase and Polarity Combinations

Polarity (POL)	Phase (PHA)	SPI Mode	Comments
0	0	Mode 0	SPIi.CLK *a*ctive high; sampling occurs on the rising edge.
0	1	Mode 1	SPIi.CLK active high; sampling occurs on the falling edge.
1	0	Mode 2	SPIi.CLK active low; sampling occurs on the falling edge.
1	1	Mode 3	SPIi.CLK active low; sampling occurs on the rising edge.

Figure 19−8. Phase and Polarity Combinations



Transfer Format

In both master and slave modes, McSPI drives the data lines when SPIi.nCSj is asserted.

This section explains two cases of data transmission determined by the clock phase (PHA): transmission in mode 0 and mode 2 (PHA = 0) and transmission in mode 1 and mode 3 (PHA = 1).

Each data frame is transmitted starting with the most-significant bit (MSB).

❑ Transmission in mode 0 and mode 2 (PHA = 0)

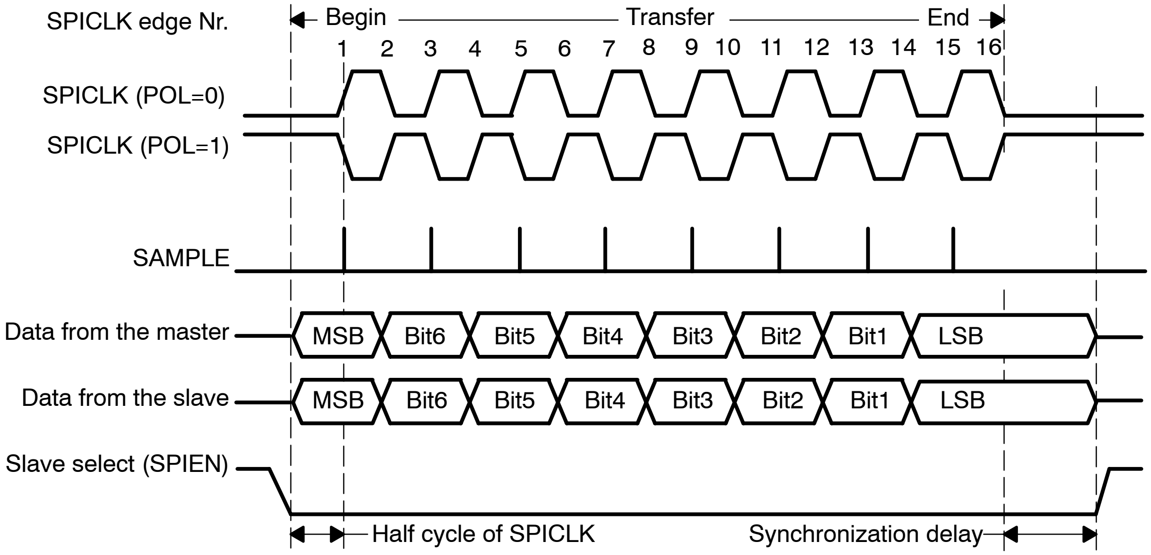
When PHA = 0, the first bit of the SPI word to transmit (on the master or slave data output pin) is valid one half cycle of SPIi.CLK after SPIi.nCSj assertion.

Therefore, the first edge of the SPIi.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 SPIi.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 total of pulses on the SPIi.CLK line defined by the SPI word length programmed in the master device, with data latched on odd-numbered edges and shifted on even-numbered edges. See Figure 19−9.

Figure 19−9. Full-Duplex Transfer Format With PHA = 0



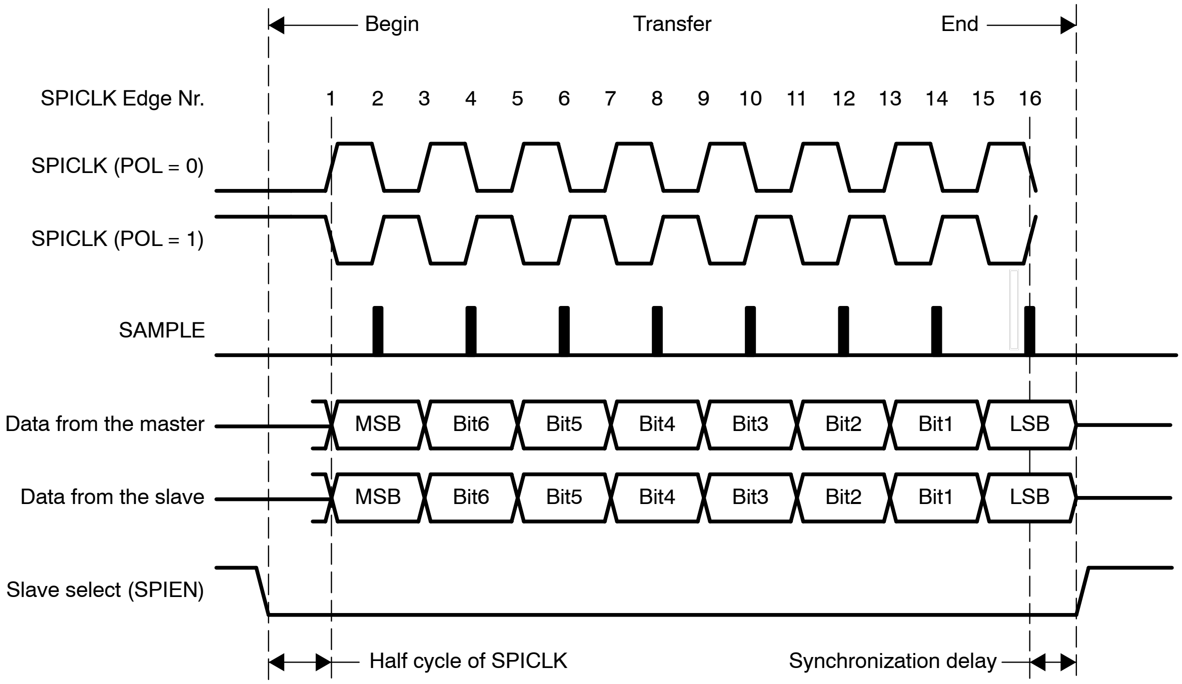
- ❑ Transmission in mode 1 and mode 3 (PHA = 1)

  When PHA = 1, the first bit of the SPI word to transmit (on the master or slave data output pin) is valid on the following SPIi.CLK edge (one half cycle later). This is the sampling edge for the master and slave. A synchronization delay is added between SPIi.nCSj activation and the first SPIi.CLK edge.

  The received data bit is shifted into the shift register on the third SPIi.CLK edge.

  This process continues for a number of pulses on the SPIi.CLK line defined by the word length programmed in the master device, with data latched on even-numbered edges and shifted on odd-numbered edges. See Figure 19−10.

Figure 19−10. Full-Duplex Transfer Format With PHA = 1



Note:

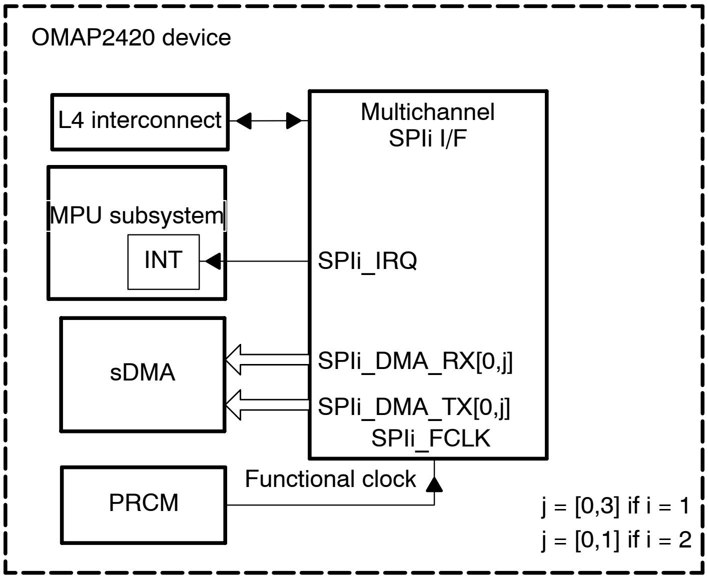
The minimum synchronization delay is one cycle of SPIi.CLK if the SPIi.CLK frequency equals the SPIi_FCLK (McSPIi functional clock) frequency in master mode. The minimum synchronization delay is one half cycle of SPIi.CLK if the SPIi.CLK frequency is lower than the SPIi_FCLK frequency in both master and slave modes.

19.4 McSPI Integration

19.4.1 McSPI Description

Figure 19−11 shows McSPI module integration in the OMAP2420 device.

Figure 19−11. McSPI Integration



19.4.2 Clocking, Reset, and Power-Management Scheme

19.4.2.1 Clocking

Each McSPI module is clocked with an independent functional clock of 48 MHz (see Chapter 5, *Power, Reset, and Clock Management*). Table 19−4 lists the two functional clocks.

Table 19−4. McSPI Functional Clocks

Clock	Frequency	Name	Comments
SPI1 functional clock	48 MHz	SPI1_FCLK	Source is PRCM module.
SPI2 functional clock	48 MHz	SPI2_FCLK	Source is PRCM module.

19.4.2.2 Hardware Reset

The module is reset by the hardware when an active-low reset signal is asserted on input pin RESETN.

This hardware reset signal has a global reset effect on the module. All configuration registers and all state-machines are reset in all clock domains. Table 19−5 describes the McSPI hardware reset.

Table 19−5. McSPI Hardware Reset

Peripherals	Name	Comments
McSPI module	RESETN	Same as global reset

19.4.2.3 Software Reset

The soft reset bit MCSPI_SYSCONFIG[1] controls the software reset of the SPI interface. Writing 1 to this bit enables active software reset functionality, which is equivalent to hardware reset.

19.4.2.4 Power Domain

The McSPI module operates in the CORE power domain (see Table 19−6).

Table 19−6. McSPI in the CORE Power Domain

Peripherals	Power Domain
McSPI module	CORE

19.4.2.5 Hardware Requests

DMA Requests

Each channel includes two DMA requests, one for reception and one for transmission. Table 19−7 lists the DMA requests and their characteristics.

Table 19−7. McSPI DMA Requests

Attributes	Values	Name	Mapping	Comments
SPI1				
DMA request	8	SPI1_DMA_RX[3:0] SPI1_DMA_TX[3:0]	S_DMA_34−S_DMA_41	Destination is system DMA (sDMA).
SPI2				
DMA request	4	SPI2_DMA_RX[1:0] SPI2_DMA_TX[1:0]	S_DMA_42−S_DMA_45	Destination is sDMA.

Interrupt Requests

Each module McSPI handles one interrupt line. Table 19−8 lists the interrupt lines.

Table 19−8. McSPI Interrupt Request Lines

Attributes	Values	Name	Mapping	Comments
SPI1				
Interrupt request	1	SPI1_IRQ	M_IRQ_65	Destination is MPU subsystem interrupts.
SPI2				
Interrupt request	1	SPI2_IRQ	M_IRQ_66	Destination is MPU subsystem interrupts.

Wake Requests

Table 19−9 lists the wake requests.

Table 19−9. Wake Requests

Attributes	Values	Name	Mapping	Comments
SPI1				
Wake source	1	SPI1.CS0	SPI1_ SWAKEUP	Destination is PRCM module.
SPI2				
Wake source	1	SPI2.CS0	SPI2_ SWAKEUP	Destination is PRCM module.

19.4.2.6 Pin List and Pad Multiplexing With Other Functions

In Table 19−10 and Table 19−11, a black cell indicates that a mode is not used, gray shading indicates that a mode is used, and no shading indicates an alternate function.

Table 19−10. Pin Multiplexing for McSPI1

Function n	Description	DIR	Ball	Alternate Functions					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
SPI1.CLK	SPI clock	IO	U18	░	■	■	gpio.81	■	■
SPI1.SIMO	Slave data input, master data output	IO	V20	░	■	■	gpio.82	■	■
SPI1.SOMI	Slave data output, master data input	IO	T18	░	■	■	gpio.83	■	■
SPI1.nCS0	SPI enable 0	IO	U19	░	■	■	gpio.84	■	■
SPI1.nCS1	SPI enable 1	O	N15	░	■	■	gpio.85	■	■
SPI1.nCS2	SPI enable 2	O	R18	░	■	■	gpio.86	■	■
SPI1.nCS3	SPI enable 3	O	U21	░	■	■	gpio.87	■	■

Table 19−11. Pin Multiplexing for McSPI2

SPI2 Interface	Description	DIR	Ball	Alternate Functions					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
SPI2.CLK	SPI clock	IO	T19	░	■	■	gpio.88	■	■
SPI2.SIMO	Slave data input, master data output	IO	R19	░	gpt10. pwm/evt	■	gpio.89	■	■
SPI2.SOMI	Slave data output, master data input	IO	R20	░	gpt11. pwm/evt	■	gpio.90	■	■
SPI2.nCS0	SPI enable 0	IO	M14	░	gpt12. pwm/evt	■	gpio.91	■	■
SPI2.nCS1	SPI enable 1	O	H19	i2c2.sda	■	░	gpio.100	■	■
			P20	mcbsp1.fsr			gpio.93	░	■

19.4.3 L4 Interconnect Interface

The McSPI has an interface with the L4 interconnect through a dedicated target agent (TA). This TA, which is part of the L4 interconnect, provides status and configuration registers, as listed in Table 19−12 and Table 19−13. By default, TA registers values are functional, but they can be overwritten. After changing the values of TA registers, the changes take effect without having to reset the system. (For a complete description, see Chapter 6, *Internal Interconnect*).

Table 19−12. L4 Interconnect Registers (McSPI1)

Register Name	Type	Register Width (Bits)	Physical Address
COMPONENT	R	32	0x4809 9000
AGENT_CONTROL	RW	32	0x4809 9020
AGENT_STATUS	R	32	0x4809 9028

Table 19−13. L4 Interconnect Registers (McSPI2)

Register Name	Type	Register Width (Bits)	Physical Address
COMPONENT	R	32	0x4809 B000
AGENT_CONTROL	RW	32	0x4809 B020
AGENT_STATUS	R	32	0x4809 B028

19.4.4 SPI Interface Register Summary

Table 19−14 and Table 19−15 list the McSPI registers.

Table 19−14. MCSPI1 Register Summary

Register Name	Type	Register Width (Bits)	Physical Address
MCSPI_REVISION	R	32	BaseAddrSPI1 +0x00
MCSPI_SYSCONFIG	RW	32	BaseAddrSPI1 +0x10
MCSPI_SYSSTATUS	R	32	BaseAddrSPI1 +0x14
MCSPI_IRQSTATUS	RW	32	BaseAddrSPI1 +0x18
MCSPI_IRQENABLE	RW	32	BaseAddrSPI1 +0x1C
MCSPI_SYST	RW	32	BaseAddrSPI1 +0x24
MCSPI_MODULCTRL	RW	32	BaseAddrSPI1 +0x28
MCSPI_CHCONF0 – MCSPI_CHCONF3	RW	32	BaseAddrSPI1 +0x2C– BaseAddrSPI1 +0x68
MCSPI_CHSTAT0 – MCSPI_CHSTAT3	RW	32	BaseAddrSPI1 +0x30– BaseAddrSPI1 +0x6C
MCSPI_CHCTRL0 – MCSPI_CHCTRL3	RW	32	BaseAddrSPI1 +0x34– BaseAddrSPI1 +0x70
MCSPI_TX0 – MCSPI_TX3	RW	32	BaseAddrSPI1 +0x38– BaseAddrSPI1 +0x74
MCSPI_RX0 – MCSPI_RX3	RW	32	BaseAddrSPI1 +0x3C– BaseAddrSPI1 +0x78

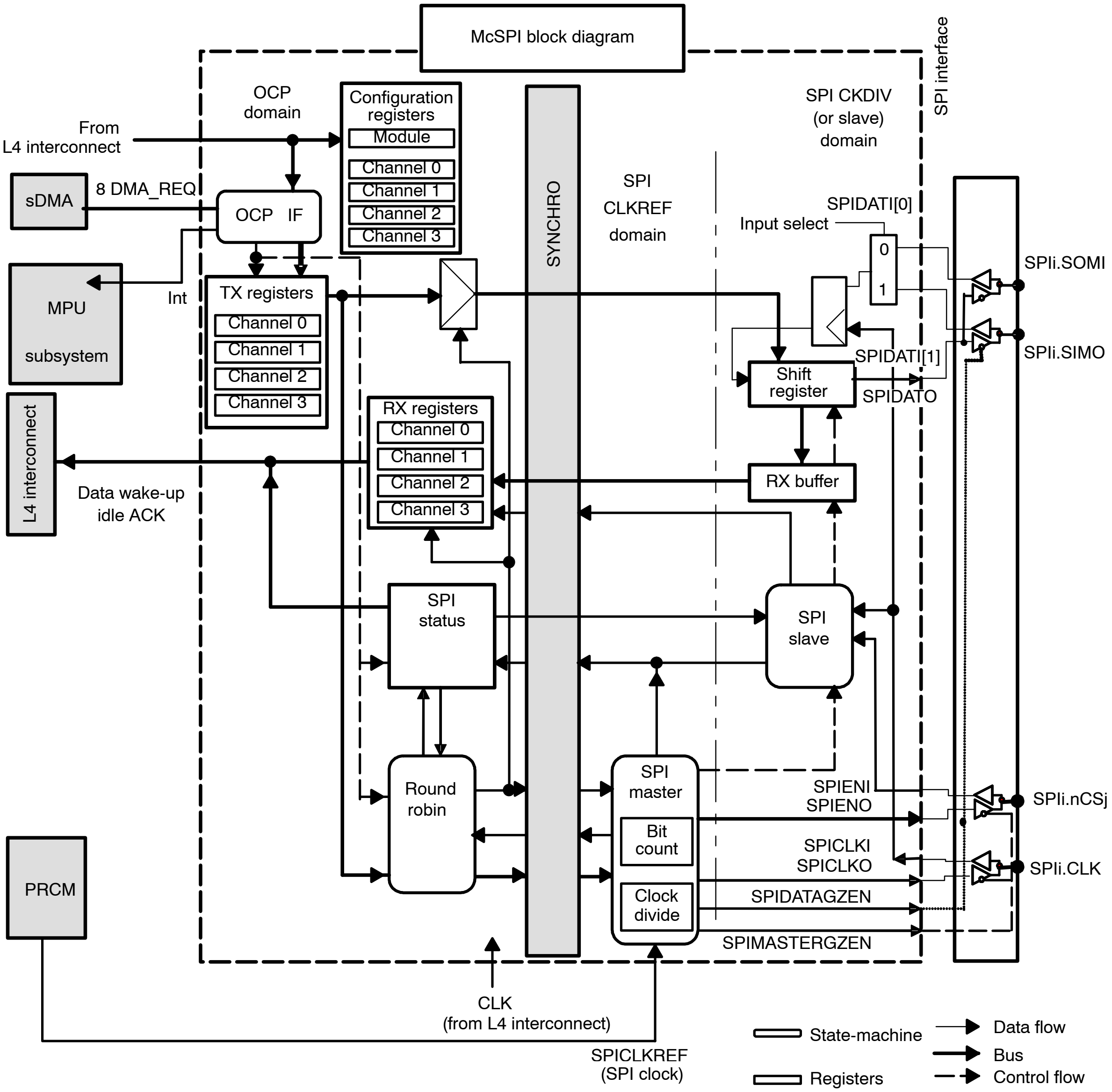
Table 19−15. MCSPI2 Register Summary

Register Name	Type	Register Width (Bits)	Physical Address
MCSPI_REVISION	R	32	BaseAddrSPI2 +0x00
MCSPI_SYSCONFIG	RW	32	BaseAddrSPI2 +0x10
MCSPI_SYSSTATUS	R	32	BaseAddrSPI2 +0x14
MCSPI_IRQSTATUS	RW	32	BaseAddrSPI2 +0x18
MCSPI_IRQENABLE	RW	32	BaseAddrSPI2 +0x1C
MCSPI_SYST	RW	32	BaseAddrSPI2 +0x24
MCSPI_MODULCTRL	RW	32	BaseAddrSPI2 +0x28
MCSPI_CHCONF0 − MCSPI_CHCONF3	RW	32	BaseAddrSPI2 +0x2C− BaseAddrSPI2 +0x68
MCSPI_CHSTAT0 − MCSPI_CHSTAT3	RW	32	BaseAddrSPI2 +0x30− BaseAddrSPI2 +0x6C
MCSPI_CHCTRL0 − MCSPI_CHCTRL3	RW	32	BaseAddrSPI2 +0x34− BaseAddrSPI2 +0x70
MCSPI_TX0 − MCSPI_TX3	RW	32	BaseAddrSPI2 +0x38− BaseAddrSPI2 +0x74
MCSPI_RX0 − MCSPI_RX3	RW	32	BaseAddrSPI2 +0x3C− BaseAddrSPI2 +0x78

19.5 McSPI Functional Description

Figure 19−12 shows the McSPI (module n).

Figure 19−12. McSPI (Module n) Block Diagram



19.5.1 Master Mode

19.5.1.1 Master Mode Features

McSPI master mode supports multichannel communication with up to four independent SPI communication channel contexts. McSPI initiates data transfer on the data lines (SPIi.SIMO and SPIi.SOMI) and generates clock (SPIi.CLK) and control signals (SPIi.nCSj).

Connected to multiple external devices, 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)
- Single-data-pin interface mode (recommended for half-duplex transmission)

Two DMA request events (read and write) allow accesses of the DMA controller that are synchronized with McSPI activity.

Three interrupt events can be used for data transmission and reception in master mode (see Section 19.5.4.1, *Interrupt Events in Master Mode*).

19.5.1.2 Master Transmit and Receive Mode (Full Duplex)

In full-duplex transmission, data is transmitted (shifted out serially on SPIi.SIMO) and received (shifted in serially on SPIi.SOMI) simultaneously on separate data lines.

Master transmit and receive mode is programmable per channel (using the TRM bits of the register MCSPI_CHCONFi).

Channel access (for transmission/reception) to the shift registers is based on the statuses of the MCSPI_TXi transmitter register and the MCSPI_RXi receiver register and on round-robin arbitration.

The channel that meets the following rules is included in the round-robin list of active channels scheduled for transmission and/or reception. The arbiter skips a channel that does not meet the rules and searches for the next enabled channel, in rotation.

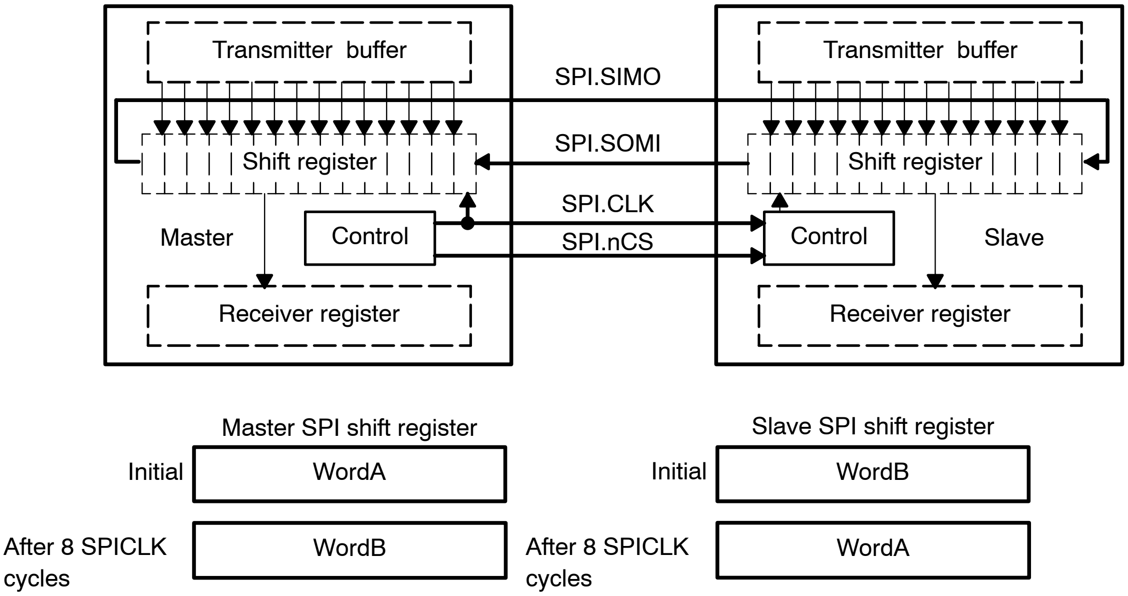
- Rule 1: Only enabled channels (EN bit MCSPI_CHCTRLi[0]), can be scheduled for transmission and/or reception.
- Rule 2: If its transmitter register is not empty (bit TXS of the register MCSPI_CHSTATi), an enabled channel can be scheduled at the time of shift register assignment. If the transmitter register is empty at the time of shift register assignment, the TX_UNDERFLOW event is activated and the next enabled channel with new data to transmit is scheduled. (See also transmit-only mode).
- Rule 3: An enabled channel can be scheduled if its receive register is not full (RXS bit of the MCSPI_CHSTATi register) at the time of shift register assignment. (See also receive-only mode). Therefore, the receiver register cannot be overwritten. In this mode, the RX_OVERFLOW bit in the MCSPI_IRQSTATUS register is never set.

When the SPI word transfer is complete (the MCSPI_CHSTATi[2] bit is set), the updated transmitter register of the next scheduled channel is loaded into the shift register. Serialization (transmit and receive) starts according to the channel communication configuration. On serialization completion, the received data is transferred to the channel receive register.

The serial clock (SPIi.CLK) synchronizes shifting and sampling of the information on the two serial data lines (SPIi.SIMO and SPIi.SOMI). Each time a bit is transferred out from the master, 1 bit is transferred in from the slave.

Figure 19−13 shows an example of a full-duplex system with a master device on the left and a slave device on the right. After eight cycles of the serial clock SPIi.CLK, WordA is transferred from the master to the slave. At the same time, WordB is transferred from the slave to the master.

Figure 19−13. SPI Full-Duplex Transmission (Example)



Master Transmit-Only Mode

Master transmit-only mode prevents the local host (LH) from reading the receiver register (minimizing data movement) when only transmission is meaningful.

The master transmit-only mode is programmable per channel (using the TRM bits of the register MCSPI_CHCONFi).

In master transmit-only mode, transmission starts only after data is loaded into the transmitter register.

Rule 1 and Rule 2, defined above, apply in this mode.

Rule 3, defined above, does not apply.

In master transmit-only mode, the receiver register state FULL does not prevent transmission, and the receiver register is always overwritten with the new SPI word. This event is not significant when only transmission is meaningful. Therefore, the RX_OVERFLOW bit in the MCSPI_IRQSTATUS register is never set in this mode.

The hardware automatically disables RX_FULL interrupt and DMA read requests.

The status of serialization is given by MCSPI_CHSTATi[2].

Master Receive-Only Mode

Master receive-only mode prevents the LH from refilling the transmitter register (minimizing data movement) when only reception is meaningful.

The master receive mode is programmable per channel (using the TRM bits of the MCSPI_CHCONFi register).

The master receive-only mode enables channel scheduling only on empty state of the receiver register.

Rule 1 and Rule 3, defined above, apply in this mode.

Rule 2, defined above, does not apply.

In master receive-only mode, the software must write dummy data to the transmit register; the transmitter register state is kept full. The content of the transmitter register is always loaded into the shift register at the time of shift register assignment. So, after writing the dummy data to the transmit register, the bits TX_EMPTY and TX_UNDERFLOW in the MCSPI_IRQSTATUS register are never set in this mode.

The status of serialization is given by MCSPI_CHSTATi[2]. The bit MCSPI_IRQSTATUS[RXi_FULL] is set when received data is loaded from the shift register to the receiver register.

19.5.2 Single-Channel Master Mode

When McSPI is configured as a master device with a single enabled channel, the assertion of the SPIi.nCSj can be controlled in two ways:

- SPIi.nCSj assertion/deassertion after each SPI word, automatically controlled by McSPI module (see subsections of Section 19.5.1, *Master Mode*).
- SPIi.nCSj assertion/deassertion controlled by software (see Section 19.5.2.1, *Programming Aid for Switching to Another Channel*).

19.5.2.1 Programming Aid for Switching to Another Channel

When a single channel is enabled, and a data transfer is ongoing,

- Wait for completion of SPI word transfer (MCSPI_CHSTATi[2] is set to 0) before disabling the current channel and enabling a different channel.
- Enable the other channel while the current channel is still enabled, then disable the previous channel.

19.5.2.2 Keep SPI.nCS Active Mode (MCSPI_CHCONFi[20])

Continuous transfers are manually allowed by keeping the SPI.nCS signal active for successive SPI words. Several sequences (configuration/enable/disable of the channel) can be run without deactivating the SPIi.nCSj line. This mode is supported by all the channels and any master sequence can be used (transmit-receive, transmit-only, receive-only).

Keep SPI.nCS active mode is authorized when:

- A single channel is to be used (MCSPI_MODULCTRL[0] = 1)

- ☐ The parameters of the transfer are loaded in the configuration register (MCSPI_CHCONFi) in the appropriate channel

  The state of the SPIi.nCSj signal is programmable:

  - Writing 1 to the MCSPI_CHCONFi[20] bit drives the SPIi.nCSj line high when MCSPI_CHCONFi[6] = 0. SPIi.nCSj is driven low when MCSPI_CHCONFi[6] = 1.
  - Writing 0 to the MCSPI_CHCONFi[20] bit drives the SPIi.nCSj line low when MCSPI_CHCONFi[6] = 0. SPIi.nCSj is driven high when MCSPI_CHCONFi[6] = 1.

- ☐ A single channel is enabled (MCSPI_CHCTRLi[0] = 1). The first enabled channel activates the SPIi.nCSj line.

When the channel is enabled, the SPIi.nCSj signal is activated with the programmed polarity. As in multichannel master mode, the start of the transfer depends on the status of the transmitter register, the status of the receiver register, and the mode defined by the TRM bits in the configuration register (transmit only, receive only, or transmit and receive) of the enabled channel.

The status of the serialization completion of each SPI word is given by the MCSPI_CHSTATi[2] bit. The MCSPI_IRQSTATUS[14] bit is set when received data is loaded from the shift register to the receiver register.

A change in the configuration parameters is directly propagated on the SPI interface. If the SPIi.nCSj signal is activated, ensure that the configuration is changed only between SPI words, to avoid corrupting the current transfer.

Note:

The SPIi.nCSj polarity, the SPICLK phase, and the SPICLK polarity must not be modified when the SPIi.nCSj signal is activated.

A delay between SPI words that requires the connected SPI slave device to switch from one configuration to another (for instance, from transmit-only to receive-only) must be handled by software.

At the end of the last SPI word, the channel must be deactivated (MCSPI_CHCTRLi[0] = 0) and the SPIi.nCSj signal can be forced to its inactive state (MCSPI_CHCONFi[20].

Figure 19−14 and Figure 19−15 show successive transfers with the SPIi.nCSj signal kept active low with a different configuration for each SPI word in single-data-pin and dual-data-pin interface modes, respectively.

Figure 19−14. Continuous Transfers With SPIi.nCSj Kept Active (Single-Data-Pin Interface Mode)

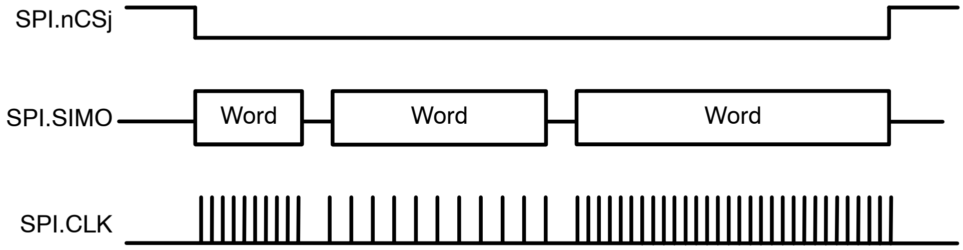
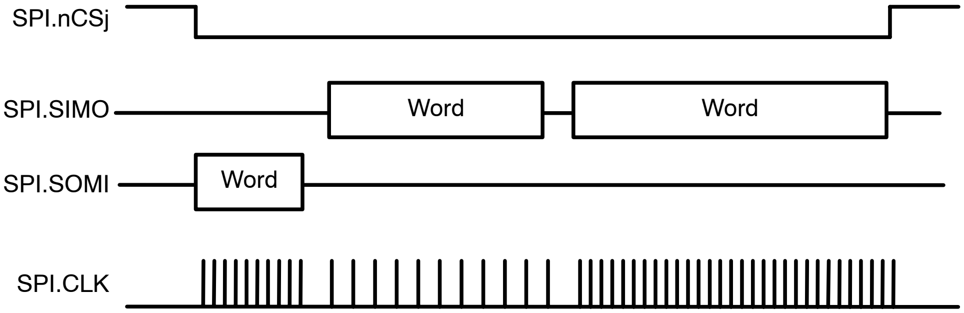


Figure 19−15. Continuous Transfers With SPIi.nCSj Kept Active (Dual-Data-Pin Interface Mode)



Note:

The turbo mode described in Section 19.5.2.3, *Turbo Mode*, is also supported to keep SPI.nCSj in active mode when the following conditions are met:

−A single channel is explicitly used (the MCSPI_MODULCTRL[0] bit is set to 1).

−Turbo mode is enabled in the configuration of the channel (MCSPI_CHCONFi[19]).

19.5.2.3 Turbo Mode

Turbo mode improves the throughput of the SPI interface when a single channel is enabled, by allowing transfers until the shift register and the receiver register are full. Turbo mode is useful (gain of time) when a transfer exceeds two words.

This mode is programmable per channel (using the MCSPI_CHCONFi[19] bit).

When several channels are enabled, MCSPI_CHCONFi[19] has no effect, and the channel access to the shift registers remains as described in Section 19.5.1.2, *Master Transmit and Receive Mode (Full Duplex).*

In turbo mode, Rule 1 and Rule 2 defined in Section 19.5.1.2, *Master Transmit and Receive Mode (Full Duplex)*, apply, but Rule 3 does not apply. An enabled channel can be scheduled if its receive register is full (MCSPI_CHSTATi[0]) at the time of shift register assignment until the shift register is full.

The receiver register cannot be overwritten in turbo mode. Thus, the receive overflow bit (MCSPI_IRQSTATUS[3]) is never set in turbo mode.

19.5.3 Slave Mode

When the MCSPI_MODULCTRL[2] bit is set, the McSPI is in slave mode.

The McSPI can be connected to up to four external SPI master devices, but it handles transactions with only one SPI master device at a time.

In slave mode, the McSPI initiates data transfer on the data lines (SPIi.SIMO and SPIi.SOMI) when it is selected by an active control signal (SPIi.nCSj) and receives an SPI clock (SPIi.CLK) from the external SPI master device. Only channel 0 can be configured as slave, but the chip enable signal can be connected to any SPIi.nCSj, and it is rerouted internally to channel 0 (the MCSPIn_CHCONF0 register).

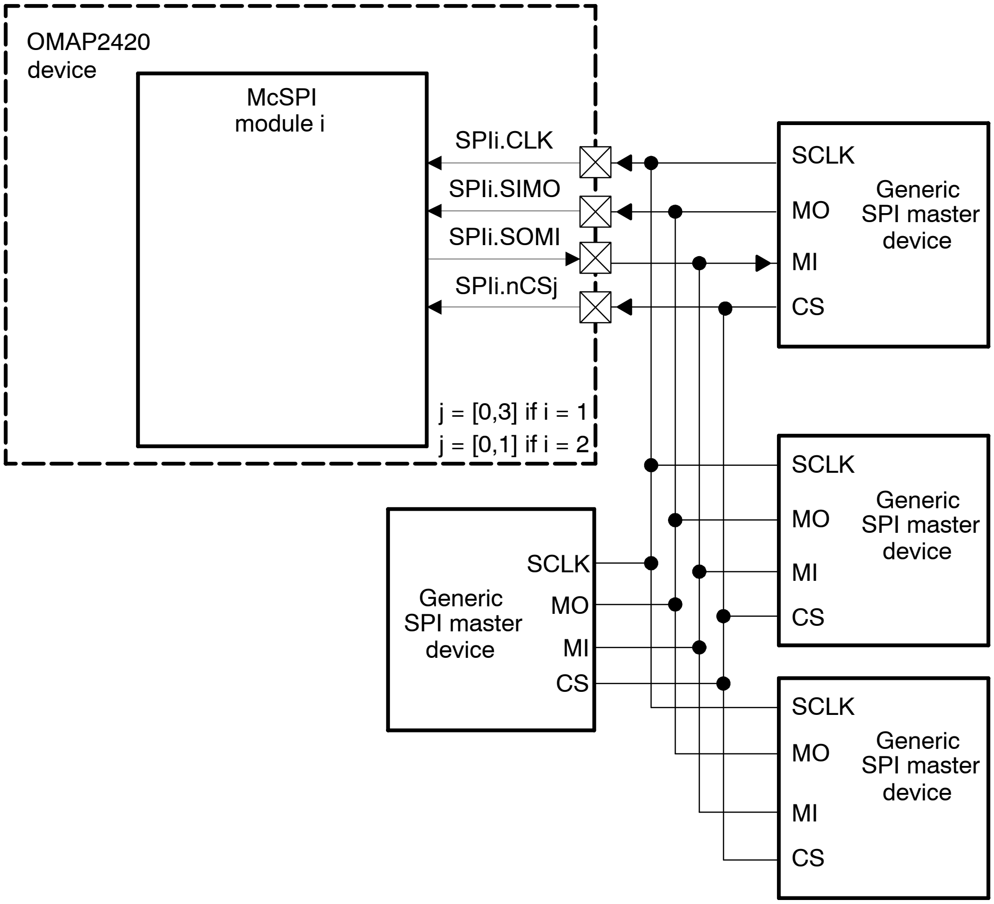
The McSPI does not support SPIi.nCSj activity between SPI words. It uses the edge to detect word length.

19.5.3.1 Dedicated Resources

Only channel 0 can be enabled in slave mode.

Figure 19−16 shows an example of four masters wired on channel 0 slave.

Figure 19−16. MCSPI Slave with Multi devices Master on Channel 0 (Example)



Channel 0, in slave mode has the following resources:

- Its own channel enable, programmable with the EN bit of the MCSPIn_CHCTRL0 register. This channel must be enabled before transmission and reception.
- For this mode, the slave select signal can be detected on any SPIi.nCSj port. This is programmable with the SPIENSLV bits of the MCSPIn_CHCONF0 register.
- Its own transmitter register, MCSPI_TX, on top of the common transmit shift register. If the transmitter register is empty, the status bit TXS of the MCSPIn_CHSTAT0 register is set. If McSPI is selected by an external master (active signal on the SPIi.nCSj port assigned to channel 0), the transmitter register content of channel 0 is always loaded into the shift register, regardless of whether its content is updated. The transmitter register must be loaded before McSPI is selected by a master.
- Its own receiver register, MCSPI_RX, on top of the common receive shift register. If the receiver register is full, the status bit RXS of the MCSPIn_CHSTAT0 register is set.

Note:

The transmitter and receiver registers of the other channels are not used. Reads from or writes to a channel register other than channel 0 has no effect.

- Its own communication configuration with the following parameters through the MCSPIn_CHCONF0 register:
  - Transmit/receive modes, programmable with the TRM bit
  - Interface mode (two data pins or one data pin) and data pin assignment, both programmable with the IS and DPE bits
  - SPI word length, programmable with the WL bit
  - SPIi.nCSj polarity, programmable with the EPOL bit
  - SPIi.CLK polarity, programmable with the POL bit
  - SPIi.CLK phase, programmable with the PHA bit

  The SPIi.CLK frequency of a transfer is controlled by the external SPI master connected to McSPI. The CLKD0 bits of the MCSPIn_CHCONF0 register are not used in slave mode.

Note:

The configuration of the channel can be loaded in the MCSPIn_CHCONF0 register only when the channel is disabled.

- Two DMA request events, read and write, to synchronize read/write accesses of the DMA controller with the activity of McSPI. The DMA requests are enabled with the DMAR and DMAW bits of the MCSPIn_CHCONF0 register.
- Four interrupt events (see Section 19.5.4.2, *Interrupt Events in Slave Mode*)

19.5.3.2 Slave Transmit-and-Receive Mode

Slave transmit-and-receive mode is the only available mode when McSPI is configured as slave (the MCSPI_CHCONF0[13:12] bits are unused in slave mode).

In slave transmit-and-receive mode, the transmitter 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.

Transmitter register content is always loaded in the shift register, regardless whether it is updated. The event TX_UNDERFLOW is activated accordingly, and does not prevent transmission.

When the SPI word transfer is complete (the MCSPI_CHSTAT0[2] bit is set), the received data is transferred to the channel receive register.

To use McSPI as a slave transmit-only device, disable the RX_FULL and RX_OVERFLOW interrupts and the DMA read requests.

19.5.3.3 Receive-Only Slave

Slave receive mode is programmable (set the MCSPI_CONF0[13:12] bits to 01).

In receive-only mode, the transmitter register must be loaded before the McSPI is selected by an external SPI master device. The transmitter register content is always loaded into the shift register, regardless of whether it is updated. The TX_UNDERFLOW event is activated accordingly and does not prevent transmission.

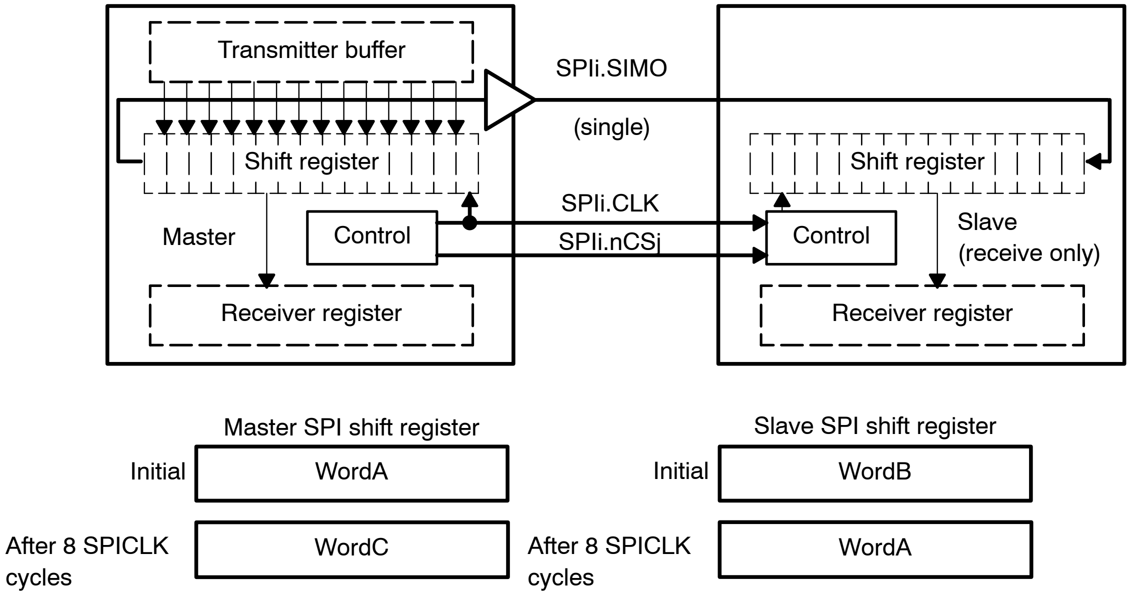
When the SPI word transfer is complete (the MCSPI_CHSTAT0[2] bit is set), the received data is transferred to the channel receive register.

To use the McSPI as a slave receive-only device, disable the TX_EMPTY and TX_UNDERFLOW interrupts and the DMA write requests.

For a full-duplex transmission, the serial clock (SPIi.CLK) synchronizes shifting and sampling of the information on the single serial data line.

Figure 19−17 shows a half-duplex system with a master device on the left and a receive-only slave device on the right.

Figure 19−17. SPI Half-Duplex Transmission (Receive-Only Slave)



When a bit is transferred out from the master, 1 bit is transferred in the slave.

After eight cycles of the serial clock SPIi.CLK, WordA is transferred from the master to the slave.

19.5.3.4 Transmit-Only Slave

Slave transmit-only mode is programmable (set the MCSPI_CHCONF0[13:12] bit to 10). This mode avoids the requirement of the LH to read the receiver register (minimizing data movement) when only transmission is meaningful.

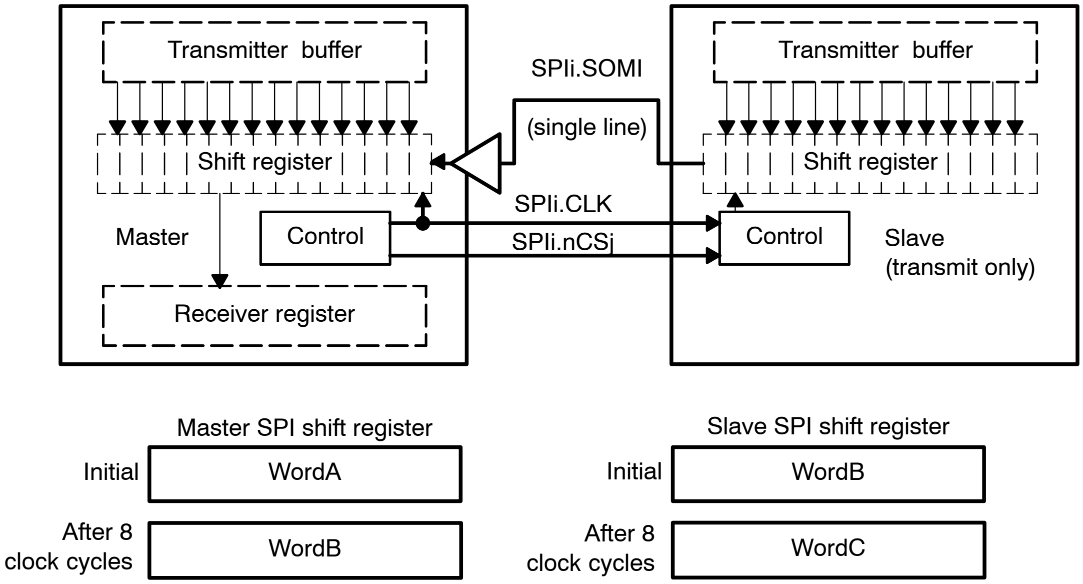
To use the McSPI as a slave transmit-only device with MCSPI_CHCONF0[13:12] = 00, inhibit the RX_FULL and RX_OVERFLOW interrupts and the DMA read requests.

When the SPI word transfer is complete, the MCSPI_CHSTAT0[2] bit is set.

19.5.3.5 Example with a Transmit-Only Slave

Figure 19−18 shows a half-duplex system with a master device on the left and a transmit-only slave device on the right. When a bit is transferred out from the slave device, 1 bit is transferred in the master. After eight cycles of the serial clock SPIi.CLK, WordA is transferred from the slave to the master.

Figure 19−18. SPI Half-Duplex Transmission (Transmit-Only Slave)



19.5.4 Interrupts

Each channel can issue interrupt events.

Each interrupt event has a status bit in the MCSPI_IRQSTATUS register that indicates that service is required, and an interrupt enable bit in the MCSPI_IRQENABLE register that enables the status to generate hardware interrupt requests.

When an interrupt occurs and later a mask is applied on it (IRQENABLE), the interrupt line is not asserted again, even if the interrupt source is not serviced.

McSPI supports interrupt-driven operation and polling.

19.5.4.1 Interrupt Events in Master Mode

In master mode, the interrupt events related to the transmitter register state are TX_EMPTY and TX_UNDERFLOW. The interrupt event related to the receiver register state is RX_FULL.

TX_EMPTY

The TX_EMPTY event is activated when a channel is enabled and its transmitter register is empty (transient event). Enabling a channel automatically triggers this event, except in master receive-only mode (see *Master Receive-Only Mode* in Section 19.5.1.2, *Master Transmit-and-Receive Mode [Full Duplex]*).

The transmitter register must be loaded with data to remove the source of the interrupt, and the TX_EMPTY interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as an interrupt source).

TX_UNDERFLOW

The TX_UNDERFLOW event is activated when a channel is enabled and its transmitter register is empty (not updated with new data) at the time of shift register assignment.

To avoid a TX_UNDERFLOW event at the beginning of a transmission, the TX_UNDERFLOW event is not activated when no data is loaded into the transmitter register.

TX_UNDERFLOW interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as an interrupt source).

TX_UNDERFLOW is a harmless warning in master mode.

RX_ FULL

The RX_FULL event is activated when the channel is enabled and the receiver register is full (transient event).

The receiver register must be read to remove the source of the interrupt, and the RX_FULL interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as an interrupt source).

19.5.4.2 Interrupt Events in Slave Mode

The interrupt events related to the transmitter register state are TX_EMPTY and TX_UNDERFLOW. The interrupt events related to the receiver register state are RX_FULL and RX_OVERFLOW.

TX_EMPTY

The TX_EMPTY event is activated when the channel is enabled and its transmitter register is empty. Enabling the channel automatically rises this event.

The transmitter register must not be empty when the source of the interrupt is removed, and the TX_EMPTY interrupt status bit must be cleared for interrupt line deassertion.

TX_UNDERFLOW

The TX_UNDERFLOW event is activated when a channel is enabled and if its transmitter register is empty (not updated with new data) when an external master device starts a data transfer with McSPI (transmit and receive).

To avoid having a TX_UNDERFLOW event at the beginning of a transmission, the event TX_UNDERFLOW is not activated when no data is loaded into the transmitter register because the channel is enabled.

The TX_UNDERFLOW interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as an interrupt source).

RX_FULL

The RX_FULL event is activated when the channel is enabled and the receiver register is being filled (transient event).

The receiver register must be read to remove the source of the interrupt, and the RX_FULL interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as an interrupt source).

RX_OVERFLOW

The RX_OVERFLOW event is activated when a channel is enabled and its receiver register is full at the time of a new SPI word reception. The receiver register is always overwritten with the new SPI word.

The RX_OVERFLOW interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as an interrupt source).

19.5.4.3 Interrupt-Driven Operation

An interrupt enable bit in the 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 LH must:

- Read the MCSPI_IRQSTATUS register to identify which event occurred
- Read the receiver register that corresponds to the event to remove the source of an RX_FULL event, or write into the transmitter register that corresponds to the event to remove the source of a TX_EMPTY event. No action is necessary to remove the source of the events WAKE_UP, TX_UNDERFLOW, and RX_OVERFLOW.
- Write 1 to the corresponding bit of the MCSPI_IRQSTATUS register to clear the interrupt status and release the interrupt line

The interrupt status bit must always be reset after the channel is enabled and before events are enabled as interrupt sources.

19.5.4.4 Polling

When the interrupt capability of an event is disabled in the MCSPI_IRQENABLE register, the interrupt line is not asserted.

- The status bits in the MCSPI_IRQSTATUS register can be polled by software to detect when the corresponding event occurs.
- When the expected event occurs, the LH must read the receiver register that corresponds to the event to remove the source of an RX_FULL event, or write into the transmitter register that corresponds to the event to remove the source of a TX_EMPTY event. No action is necessary to remove the source of the events WAKE_UP, TX_UNDERFLOW, and RX_OVERFLOW.
- To clear an interrupt, set the corresponding status bit of the MCSPI_IRQSTATUS register to 1. This does not affect the interrupt line state.

19.5.5 DMA Requests

DMA accesses are managed by the sDMA controller module. This helps the LH with data transfers.

If enabled, each channel can issue DMA requests.

There are two DMA request lines per channel (one for read and one for write).

The DMA read request line is asserted when the channel is enabled and new data is available in the receive register of the channel. A DMA read request can be individually masked with the DMAR bit of the MCSPI_CHCONFi register. The DMA read request line is deasserted on read completion of the receive register of the channel.

The DMA write request line is asserted when the channel is enabled and the transmitter register of the channel is empty. A DMA write request can be individually masked with the DMAW bit of the MCSPI_CHCONFi register. The DMA write request line is deasserted on load completion of the transmitter register of the channel.

19.5.6 Programmable SPI Clock (SPIi.CLK)

In master mode, the baud rate of the SPI serial clock is programmable.

An internal reference clock, SPIi_FCLK, is used as an input of a programmable divider (the CLKD bits of the MCSPI_MODULCTRL register) to generate the bit rate of the serial output clock SPIi.CLK (see Table 19−16).

Table 19−16. SPI Master Clock Rates

Divisor	Bit Rate (Peak)	Divisor	Bit Rate (Peak)
1	48 Mbps	128	375 Kbps
2	24 Mbps	256	~187 Kbps
4	12 Mbps	512	~93.7 Kbps
8	6 Mbps	1024	~46.8 Kbps
16	3 Mbps	2048	~23.4 Kbps
32	1.5 Mbps	4096	~11.7 Kbps
64	750 Kbps		

19.5.7 Power Management

Power consumption can be optimized by switching off internal clocks (OCP clock and SPI clock) when there is no activity.

From a power-management perspective, two modes of operation are defined:

- ❏ Normal mode
- ❏ Idle mode

The McSPI complies with the profile IdleReq/SIdleAck/SWakeUp in idle mode.

19.5.7.1 Normal Mode

In normal mode, internal SPI module clocks are automatically switched off (autogated) when there is no activity in slave or master mode.

The module OCP clock and SPI clock are autogated under the following conditions:

- The AUTOIDLE bit of the register MCSPI_SYSCONFIG is set.
- In master mode, there is no data to transmit or receive in all channels.
- In slave mode, McSPI is not selected by the external master and there are no OCP accesses.

The autogating of the module OCP clock and SPI clock stops under the following conditions:

- In master mode, an internal access occurs.
- In slave mode, an internal access occurs or McSPI is selected by the external master.

When the SIDLEMODE bit of the MCSPI_SYSCONFIG register is set to 01 (no-idle mode), any access to the module generates an error as long as the OCP clock is alive and the idle request from the host (IdleReq) is asserted.

19.5.7.2 Idle Mode

The OCP clock and SPI clock provided to the McSPI can be switched off on a system power-manager request and switched on again on a module request.

The McSPI complies with the power-management handshaking protocol: idle request from the system power manager, idle acknowledge from the McSPI, and wake-up request from the McSPI are supported.

The idle acknowledge in response to an idle request from the system power manager varies according to a mode programmed in the MCSPI_SYSCONFIG register: no-idle mode, force-idle mode, and smart-idle mode.

- When programmed for no-idle mode (MCSPI_SYSCONFIG[4:3] are set to 01), the module ignores the system power-manager request and behaves normally, as if the request is not asserted.
- When programmed for smart-idle mode (MCSPI_SYSCONFIG[4:3] are set to 10), the module acknowledges the system power-manager request according to its internal state.
- When programmed for force-idle mode (MCSPI_SYSCONFIG[4:3] are set to 00), the module acknowledges the system power-manager request unconditionally.

The OCP clock can be switched off during the smart-idle mode period, if MCSPI_SYSCONFIG[8] is set to 0.

The SPI clock can be switched off during the smart-idle mode period, if MCSPI_SYSCONFIG[9] is set to 0.

The McSPI assumes that both clocks can be switched off, regardless of the value set in MCSPI_SYSCONFIG[9:8].

Transitions From Normal Mode to Smart-Idle Mode

The module detects an idle request when the synchronous signal IDLEREQ is asserted. When IDLREQ is asserted, any access to the module generates an error while the OCP clock is alive.

When configured as a slave device, the McSPI responds to the idle request signal by asserting the SIDLEACK (idle acknowledge) signal only after completion of the current transfer (SPIi.nCSj slave selection signal deasserted by the external master) and if interrupt or DMA request lines are not asserted.

When configured as a master device, the McSPI responds to the idle request signal by asserting the SIDLEACK (idle acknowledgment) signal only after the current channel data transfer is complete and if DMA request lines are not asserted.

While SIDLEACK is not asserted, if an event occurs, the module can still generate an interrupt or a DMA request after IDLEREQ assertion. In this case, the module ignores the idle request and SIDLEACK is not asserted. The system power manager aborts the power mode transition procedure. The system must then deassert IDLEREQ before trying to access the module.

When SIDLE ACK is asserted, the module does not assert any new interrupt or DMA requests.

Wake-up Event in Smart-Idle Mode

The module wake-up feature is enabled when the MCSPI_SYSCONFIG[2] and MCSPI_WAKEUPENABLE[0] bits are both set. Wake-up capability is relevant only when the module is configured in slave mode.

When SIDLEACK is asserted, the module generates an asynchronous wake-up request to the system power manager to switch back the OCP clock and SPI clocks.

A wakeup is requested when channel 0 is enabled and an asynchronous selection occurs on the SPIi.nCS0 port associated with channel 0 (see the definition for bits [22:21] in Table 19−29, the MCSPI_CHCONF0 register description table).

After an McSPI wake-up request, the system power manager must reactivate the OCP clock at the following times:

- ❑ 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 register MCSPI_RX0 before the second SPI word serialization is complete.

The supported cases in wake-up mode are listed in Table 19−17.

Table 19−17. Smart-Idle Mode and Wake-Up Capabilities

Mode	OCP Clock	SPI Clock Ref	Functionality	Wake-Up Event
Master	Must be maintained	Must be maintained	Full functionality, but the module does not generate any new interrupt or DMA request until the system exits wake-up mode	No 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 sends asynchronously a wake-up request on detection of an event on the SPI.nCS port associated with channel 0.

In wake-up mode, interrupt and DMA request lines are not asserted.

An access to the module in wake-up mode generates an error while the OCP clock is alive.

Transitions From Smart-Idle Mode to Normal Mode

The McSPI detects the end of the wake-up period when the idle request signal (IDLEREQ) is deasserted.

The interrupt status register (the MCSPI_IRQSTATUS[16] bit) is updated with the event that caused the wakeup, and the wake-up event at the origin of the transition to normal mode is converted to its corresponding interrupt request (the WKS bit of the MCSPI_IRQSTATUS register when enabled by the WKE bit of the MCSPI_IRQENABLE register) or DMA request.

Interrupts and wake-up events have independent enable/disable controls, accessible through the MCSPI_IRQENABLE and MCSPI_WAKEUPENABLE registers, respectively. The software must ensure overall consistency.

If wakeup is requested, SWAKEUP is deasserted immediately after IDLEREQ deassertion. The interrupt status register MCSPI_IRQSTATUS is updated with the event that caused the wakeup, and the wake-up event at the origin of the transition to normal mode is converted to its corresponding interrupt request or DMA request.

Upon IDLEREQ deassertion, the module switches back to normal mode and deasserts the SIDLEACK signal. The module is fully operational.

19.5.7.3 Force-Idle Mode

Force-idle mode is enabled and exited in the following manner.

- ❑ Force-idle mode is enabled when the MCSPI_SYSCONFIG[4:3] bits are set to 00 (force idle).

  In force-idle mode, McSPI responds unconditionally to the idle request by asserting the SIDLEACK signal and by deasserting unconditionally the interrupt and DMA request lines if they are asserted. In addition, the wakeup capability is totally inhibited even if the control bits MCSPI_SYSCONFIG[2] and MCSPI_WAKEUPENABLE[0] are set.

  The transition from normal mode to idle mode does not affect the interrupt event bits of the MCSPI_IRQSTATUS register.

  In force-idle mode, the module must be disabled so the interrupt and DMA request lines are likewise deasserted. The OCP 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.

  An access to the module in force-idle mode generates an error while the OCP clock is alive and IDLEREQ is asserted.

- ❑ The module exits force-idle mode when IDLEREQ is deasserted.

  On IDLEREQ deassertion, the module switches back to normal mode and deasserts SIDLEACK. The module is fully operational. The interrupt and DMA request lines can be asserted one clock cycle later.

19.6 McSPI Programming Model

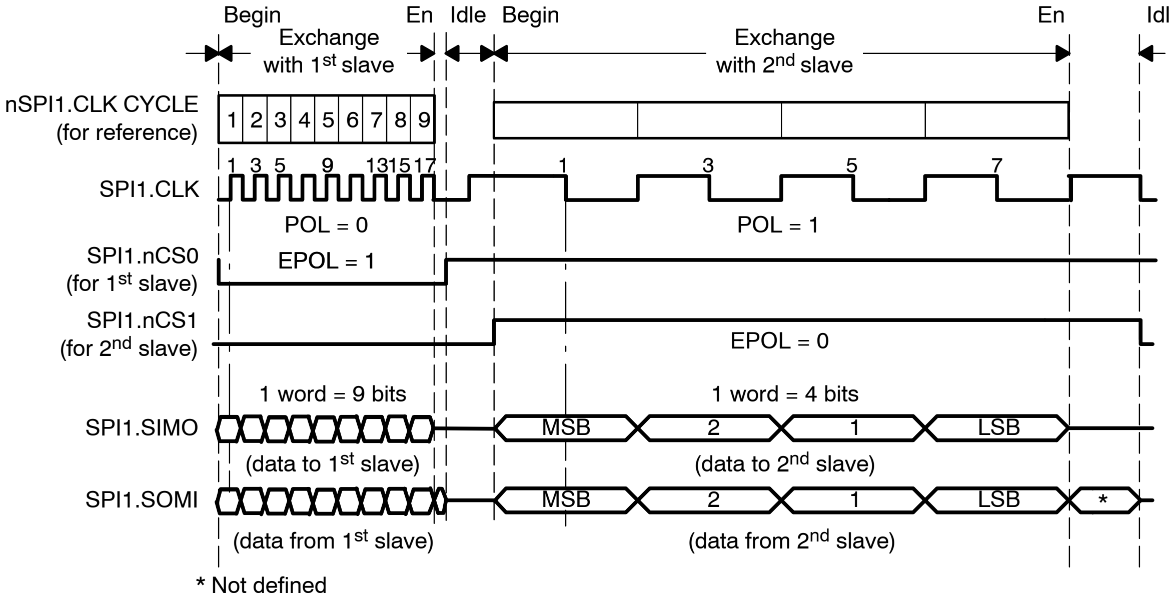
See Section 19.6.2, *Programming Aids,* for the steps required to configure and use the McSPI modes.

Figure 19−19 shows an implementation of two successive transfers between two devices (slave) and the McSPI1 (master) in transmit-and-receive mode.

McSPI is the master and SPI1.SIMO is used to shift the data to the external slaves. SPI1.SOMI is connected to the data output port of two slave devices. McSPI controls the first device with the SPI1.nCS0 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.nCS1 control signal (active high) for the exchange of 4-bit words synchronized with an SPI clock (active low) with a slower frequency than for the first slave device.

Figure 19−19. Two SPI Transfers With PHA = 0 (Flexibility of McSPI)



Section 19.6.1, *McSPI Initialization Sequence Example*, describes the steps required for this type of transmission.

19.6.1 McSPI Initialization Sequence Example

As shown in Figure 19−20, the McSPI module must first reset all registers and all state-machines.

For a software reset:

Step 1: Read the MCSPI_SYSTATUS[0] bit and check that it is set to 1.

Step 2: Set the MCSPI_SYSCONFIG[1] bit to 1.

19.6.1.1 Operations for the First Slave (on Channel 0)

Programming in Polling Mode

Mode Selection

The MCSPI_CHCONF0 register allows configuration of slave mode:

Step 1: Set the MCSPI_CHCONF0[18] bit to 0 for the SPI1.SOMI pin on reception mode.

Step 2: Set the MCSPI_CHCONF0[17:16] bit field to 1 for the SPI1.SIMO pin on transmission mode.

Step 3: Set the MCSPI_CHCONF0[13:12] bit field to 00 for transmit-and-receive mode.

Step 4: Write 01000 in the MCSPI_CHCONF0[11:7] bit field for 9-bit word length.

Step 5: Set the MCSPI_CHCONF0[6] bit to 1 for SPI1.nCS0 activated low during active state.

Step 6: Set the MCSPI_CHCONF0[1] bit to 0 for SPI1.CLK held low during active state.

Step 7: Set the MCSPI_CHCONF0[0] bit to 0 for data latched on the odd-numbered edges of the SPI clock.

Clock Initialization and SPI1.nCS0 Enable

In master mode, the SPI must provide the clock and enable the channel:

Step 1: Set the MCSPI_MODULCTRL[2] bit to 0 to provide the clock.

Step 2: Set the MCSPI_CHCTRL0[0] bit to 1 to enable channel 0.

Write Operation

Step 1: Write the command/address or data value in the MCSPI_TX0 register to transmit the value.

Step 2: Read the MCSPI_CHSTAT0[1] bit and return to Step 1 if it set to 1 (polling method).

Note:

Write and read operations can occur simultaneously.

Read Operation

Step 1: Read the MCSPI_CHSTAT0[0] bit and go to Step 2 if it set to 1.

Step 2: Read the received data value in the MCSPI_RX0 register (polling method).

Note:

Write and read operations can occur simultaneously.

19.6.1.2 Programming in Interrupt Mode

This section follows the flow of Figure 19−20.

Step 1: Initialize the software variable: Write count = 0 and read count = 0.

Step 2: Initialize the interrupt: Write 1 in the MCSPI_IRQSTATUS[3:0] bit field and set the MCSPI_IRQENABLE[3:0] bit to 1.

Step 3: Follow the steps in the *Mode Selection* subsection of Section 19.6.1.3, *Operations for the Second Slave (On Channel 1)*.

This interrupt routine follows the flow of Table 19−18 and Figure 19−20.

Step 4: Read the MCSPI_IRQSTATUS[6:4] bits.

Step 5: If MCSPI_IRQSTATUS[4] = 1, then:

- Write the command/address or data value in MCSPI_TX1.
- WRITE_COUNT = 1
- Write MCSPI_IRQSTATUS[4] = 1.

If MCSPI_IRQSTATUS[6] = 1, then:

- Read MCSPI_RX1
- READ_COUNT = 1
- Write MCSPI_IRQSTATUS[6] = 1

Step 6: Return

Step 7: If WRITE_COUNT = 1, write MCSPI_CHCTRL1 = 0 to stop the channel.

19.6.1.3 Operations for the Second Slave (On Channel 1)

Programming in Polling Mode

Mode Selection

The MCSPI_CHCONF1 register allows configuration of the mode.

Step 1: Set the MCSPI_CHCONF0[18] bit to 0 for the SPI1.SOMI pin on reception mode.

Step 2: Set the MCSPI_CHCONF0[17:16] bits to 01 for the SPI1.SIMO pin on transmission mode.

Step 3: Transmit-and-receive mode is already selected at reset (the MCSPI_CHCONF0[13:12] bits).

Step 4: Write 00011 in the MCSPI_CHCONF0[11:7] bits for 4-bit word length.

Step 5: Set the MCSPI_CHCONF0[6] bit to 0 for SPI1.nCS1 and for SPI1.CLK if activated high during active state.

Step 6: Set the MCSPI_CHCONF0[1] bit to 1 for SPI1.CLK held high during active state.

Step 7: Set the MCSPI_CHCONF0[0] bit to 1 for data latched on odd-numbered edges of SPI1.CLK.

Clock Initialization and SPI1.nCS1 Enable

In master mode, the SPI must provide the clock and enable the channel:

Step 1: The MCSPI_MODULCTRL[2] bit is previously set to 0 (providing the clock).

Step 2: Disable channel 0 by setting the MCSPI_CHCTRL0[0] bit to 0 and set the MCSPI_CHCTRL1[0] bit to 1 to enable channel 1.

Note:

Read and write operations for the second slave are identical to the first slave.

Write Operation

Step 1: Write the command/address or data value in the MCSPI_TX1 register to transmit the value.

Step 2: Read the MCSPI_CHSTAT1[1] bit and return to Step 1 if it is set to 1 (polling method).

Note:

Write and read operations can occur simultaneously.

Read Operation

Step 1: Read the MCSPI_CHSTAT1[0] bit and go to Step 2 if RX1S is set to 1.

Step 2: Read the received data value in the MCSPI_RX1 register (polling method).

Note:

Write and read operations can occur simultaneously.

19.6.1.4 Programming in Interrupt Mode

This section follows the flow of Table 19−18 and Figure 19−20.

Step 1: Initialize the software variable: Write count = 0 and read count = 0.

Step 2: Initialize the interrupt: Write 1 in the MCSPI_IRQSTATUS[6:4] bit field and set the MCSPI_IRQENABLE[6:4] bits to 1.

Step 3: Follow the steps in the *Mode Selection* and *Clock Initialization and SPI1.nCS1 Enable Operation* subsections of Section 19.6.1.3, *Operations for the Second Slave (On Channel 1)*.

Step 4: Write the command/address or data value in the MCSPI_TX1 register to transmit the value.

This interrupt routine follows the flow of Table 19−18 and Figure 19−20.

Step 5: Read the MCSPI_IRQSTATUS[6:4] bits.

Step 6: If MCSPI_IRQSTATUS[4] = 1, then:

- Write the command/address or data value in MCSPI_TX1.
- WRITE_COUNT = 1
- Write MCSPI_IRQSTATUS[4] = 1

If MCSPI_IRQSTATUS[6] = 1, then:

- Read MCSPI_RX1
- READ_COUNT = 1
- Write MCSPI_IRQSTATUS[6] = 1

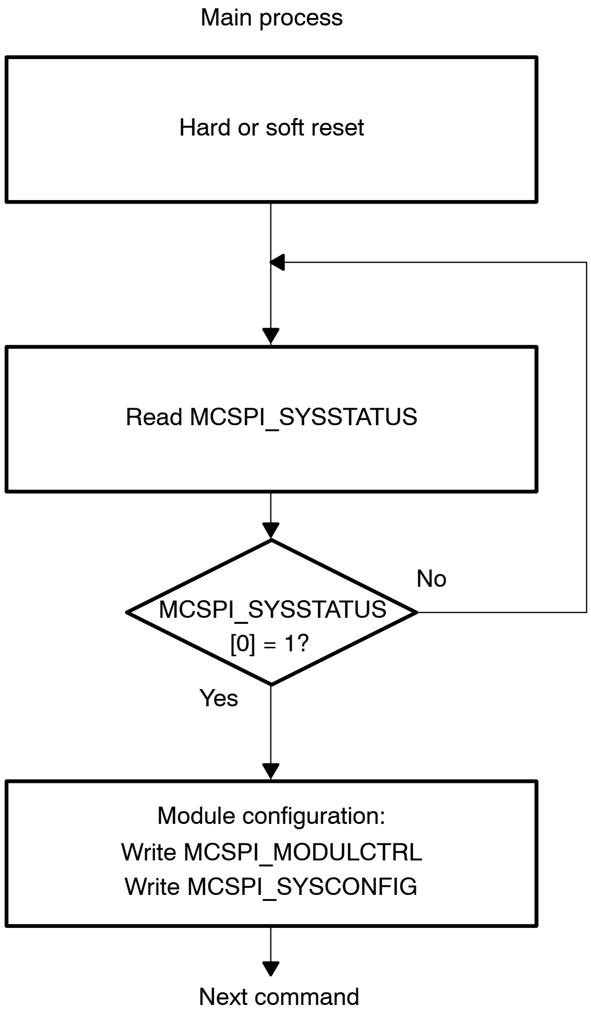
Step 7: Return

Step 8: If WRITE_COUNT = 1, write MCSPI_CHCTRL1 = 0 to stop the channel.

19.6.2 Programming Aids

Figure 19−20 and Figure 19−21 show the steps for configuring McSPI modes.

Figure 19−20. Module Initialization Flow



Note:

Before the MCSPI_SYSSTATUS[0] bit is set, the clocks CLK and CLKSPIREF must be provided to the module.

To avoid unpredictable and potentially catastrophic behavior, reset the module before changing from master mode to slave mode, or vice versa.

19.6.2.1 Common Transfer Sequence

The McSPI module allows the transfer of one or more words, according to different modes:

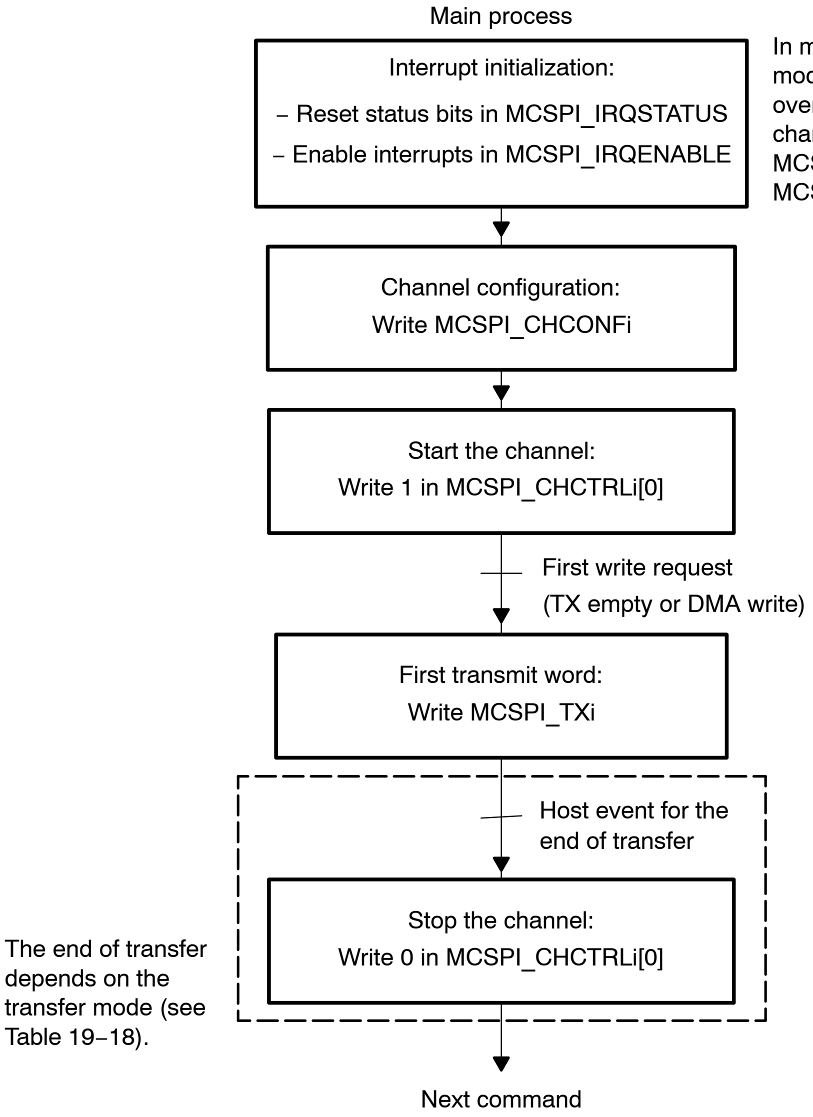
- ❏ Master normal, master turbo, slave
- ❏ Transmit-and-receive, transmit-only, receive-only

- ☐ Write and read requests: Interrupts, DMA
- ☐ SPIi.nCSj line assertion/deassertion: Automatic, manual

For all of 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.

Figure 19−21 shows the main sequence common to all transfers. In multichannel master mode, the flows of different channels can be run simultaneously.

Figure 19−21. Common Transfer Sequence


Main process
Interrupt initialization:
– Reset status bits in MCSPI_IRQSTATUS
– Enable interrupts in MCSPI_IRQENABLE
In multichannel master mode, do not overwrite the bits of other channels when initializing MCSPI_IRQSTATUS and MCSPI_IRQENABLE.
Channel configuration:
Write MCSPI_CHCONFi
Start the channel:
Write 1 in MCSPI_CHCTRLi[0]
First write request
(TX empty or DMA write)
First transmit word:
Write MCSPI_TXi
Host event for the end of transfer
Stop the channel:
Write 0 in MCSPI_CHCTRLi[0]
The end of transfer depends on the transfer mode (see Table 19−18).
Next command

]

19.6.2.2 End-of-Transfer Sequences

For transfers carried out with DMA or interrupt mode, the end of transfer must be achieved by following the steps depicted in the flow charts that correspond to Table 19−18, which summarizes the end-of-transfer types per transfer mode and provides cross-references for further information.

Table 19−18. End-of-Transfer Sequences

		Transmit Receive		Transmit Only		Receive Only	
		Interrupt	**DMA**	**Interrupt**	**DMA**	**Interrupt**	**DMA**
Master normal	End-of-transfer sequence	See Section 19.6.2.3.		See. Section 19.6.2.4	See Section 19.6.2.4.	See Section 19.6.2.5.	See Section 19.6.2.5.
	Minimum number of words	1	1	1	1	1	2
	DMA transfer size		n[1]		n[1]		n[1] – 1
Master turbo	End-of-transfer sequence	See Section 19.6.2.3.		See Section 19.6.2.4.	See Section 19.6.2.4.	See Section 19.6.2.6.	See Section 19.6.2.6.
	Minimum number of words	1	1	1	1	2	3
	DMA transfer size		n[1]		n[1]		n[1] – 2
Slave	End-of-transfer sequence	See Section 19.6.2.3.		See Section 19.6.2.4.	See Section 19.6.2.4.	See Section 19.6.2.7.	
	Minimum number of words	1	1	1	1	1	1
	DMA transfer size		n[1]		n[1]	n[1]	n[1]

1) n = number of words to transfer

The 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): Indicates that the channel is enabled
- 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

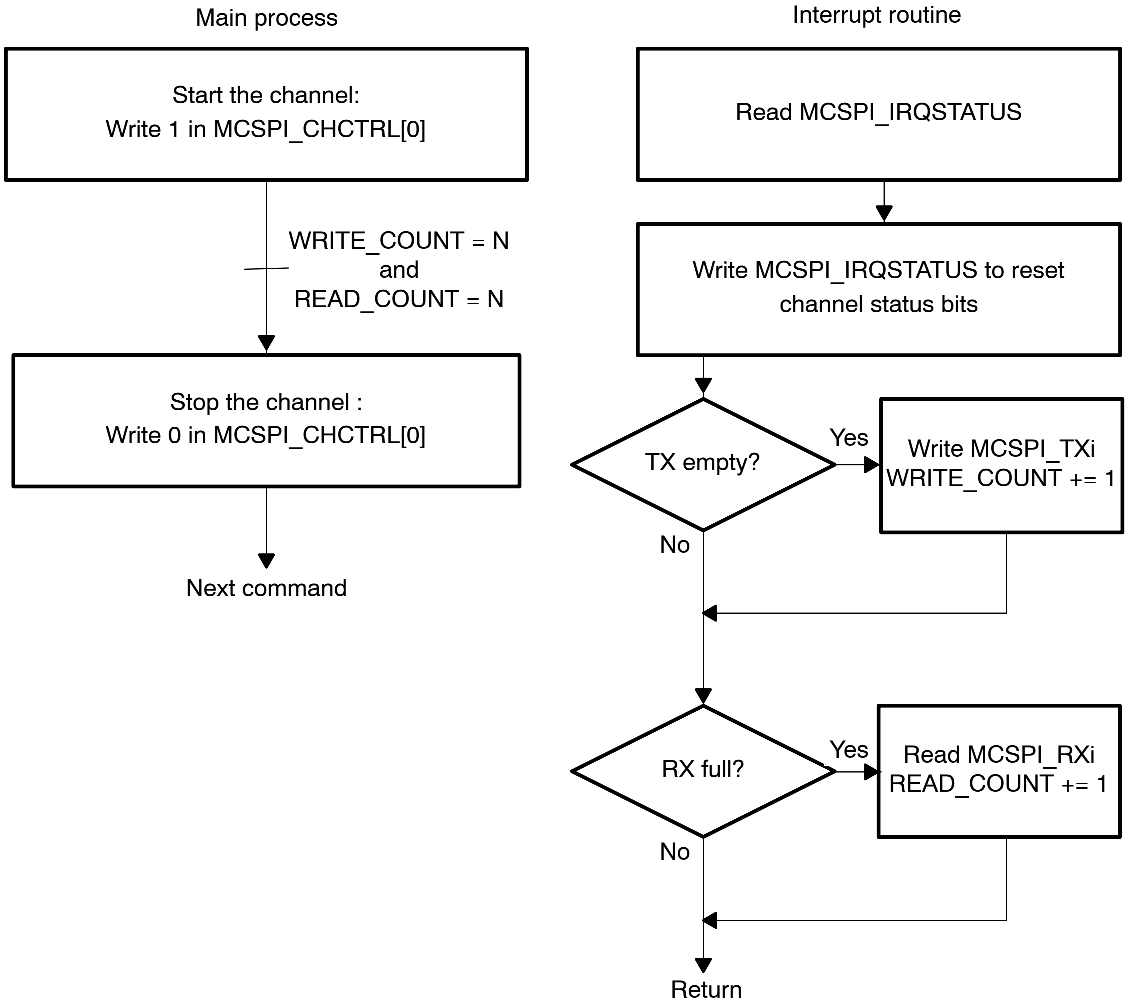
All variables are initialized before starting the channel.

19.6.2.3 Transmit and Receive (Master and Slave)

Figure 19−22 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 when all words are received for this mode.

If the requests are configured in DMA, WRITE_COUNT and READ_COUNT are assigned the value N (see Figure 19−22) when the DMA handlers have completed N OCP accesses.

Figure 19−22. Transmit and Receive (Master and Slave)

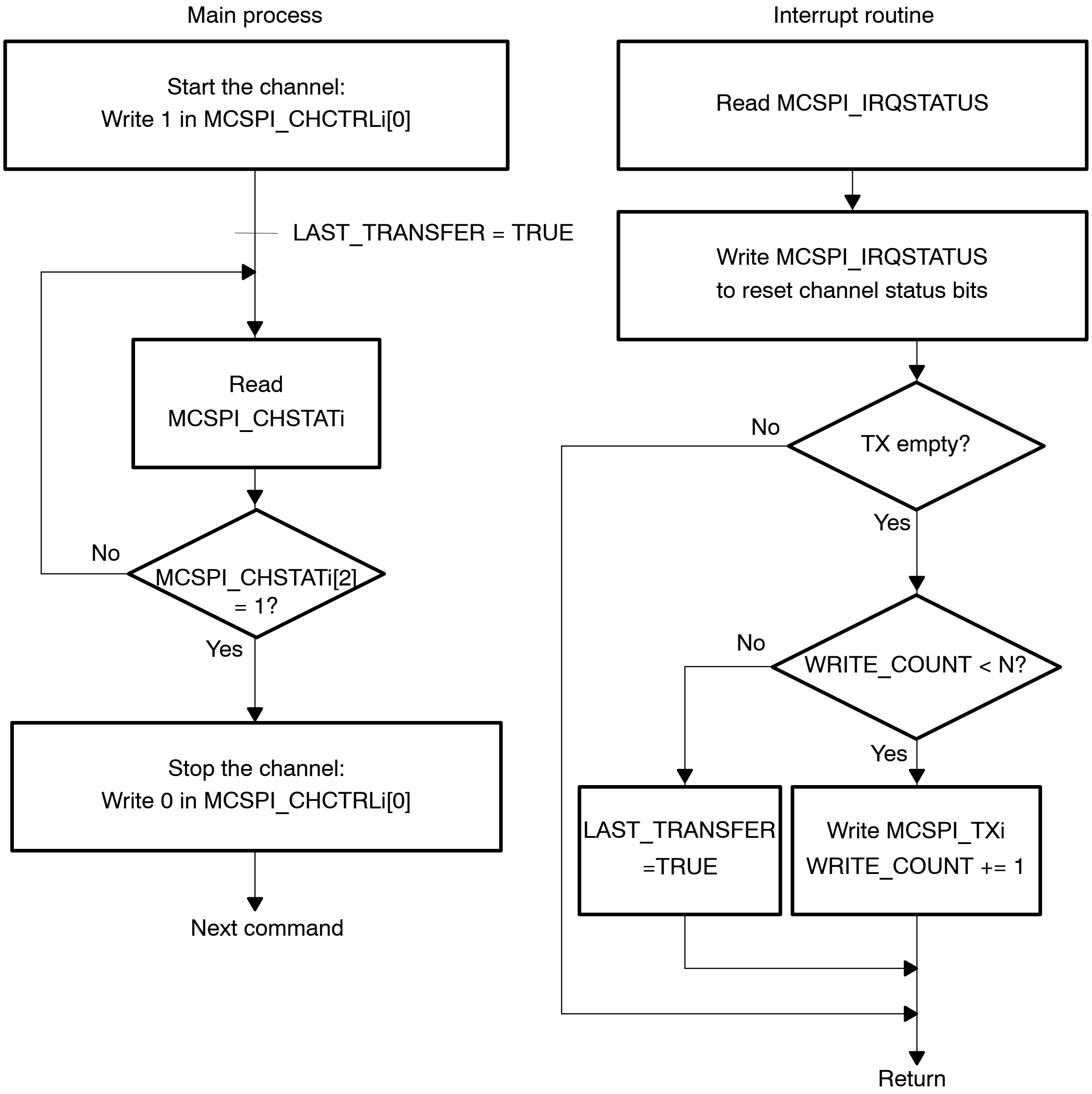


19.6.2.4 Transmit Only (Master and Slave)

Based on Interrupt Requests

Figure 19−23 shows the handling procedure for words transmitted by interrupt in transmit-only mode. The main process flow shows how the end of the transfer must be done when all words are received for this mode.

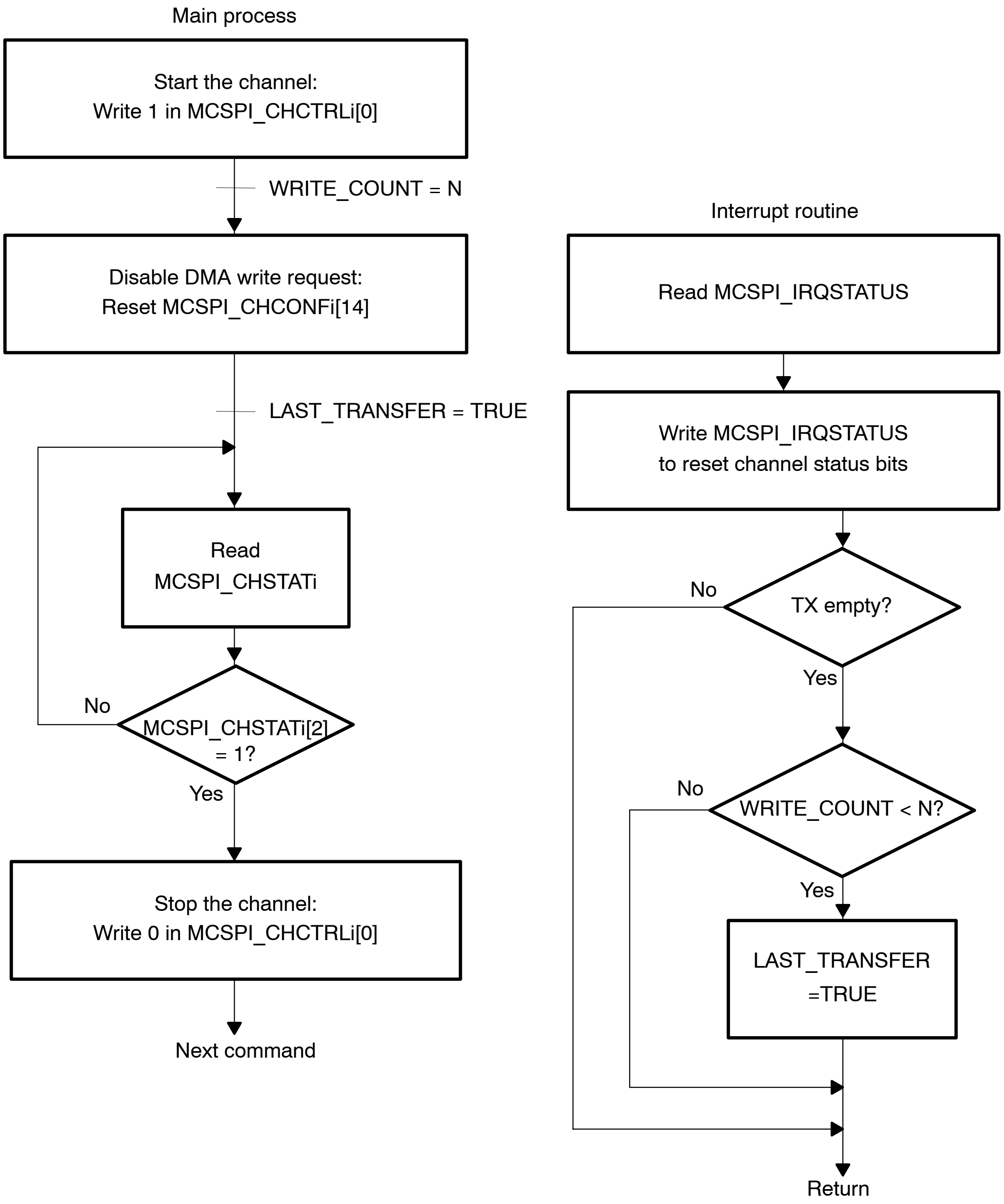
Figure 19−23. Transmit Only With Interrupts (Master and Slave)



Based on DMA Write Requests

In Figure 19−24, the main process shows completion of the transfer of words in transmit-only mode with DMA write requests. When the DMA handler completes N OCP accesses, WRITE_COUNT is assigned the value N.

Figure 19−24. Transmit Only With DMA (Master and Slave)

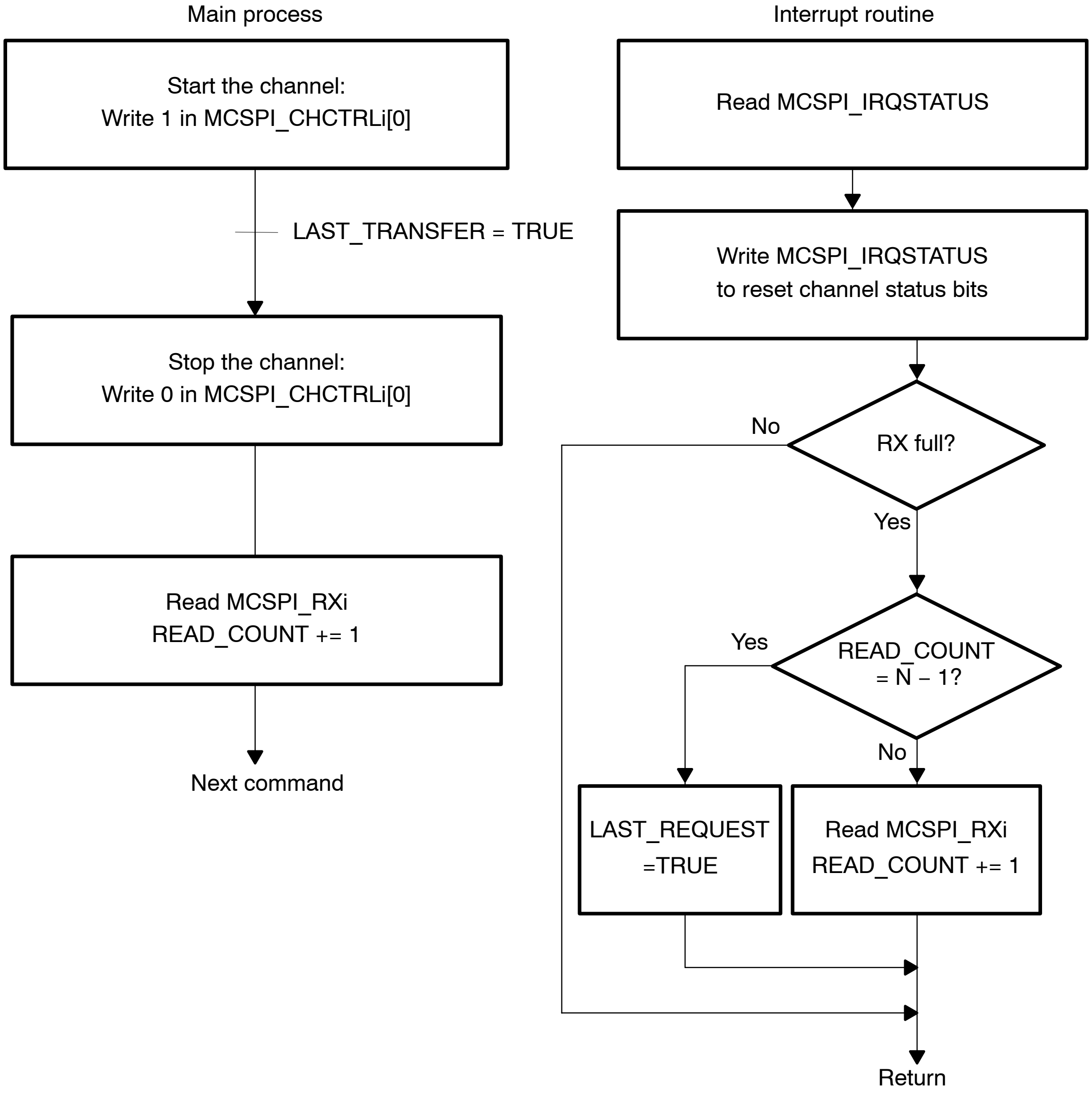


19.6.2.5 Master Normal Receive Only

Based on Interrupt Requests

Figure 19−25 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 when all words are received for this mode.

Figure 19−25. Receive Only With Interrupt (Master Normal)

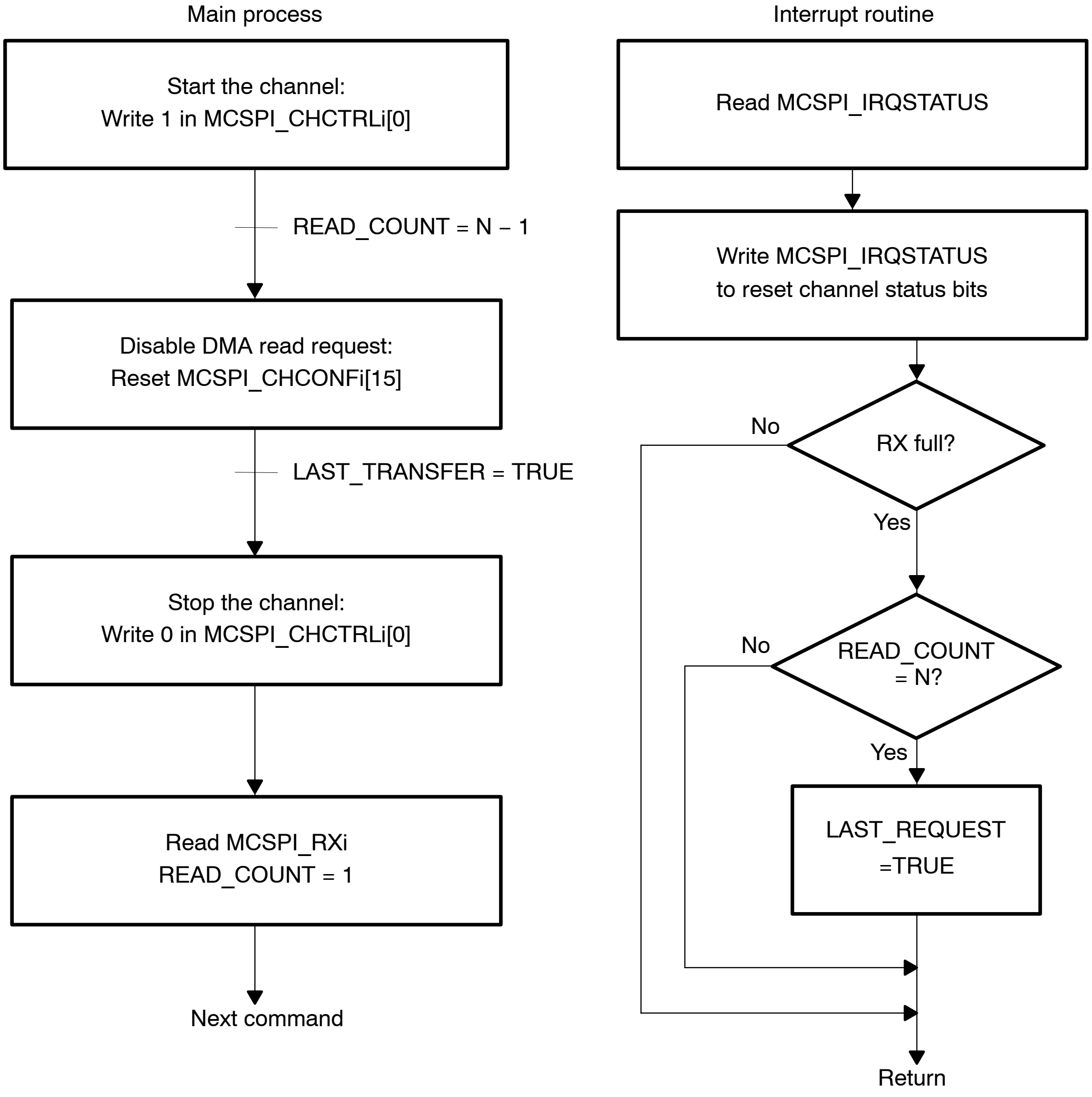


Based on DMA Read Requests

In Figure 19−26, the main process shows completion of word transfer in transmit-only mode with DMA read requests.

When the DMA handler completes N − 1 OCP accesses, READ_COUNT is assigned with N − 1.

Figure 19−26. Receive Only With DMA (Master Normal)

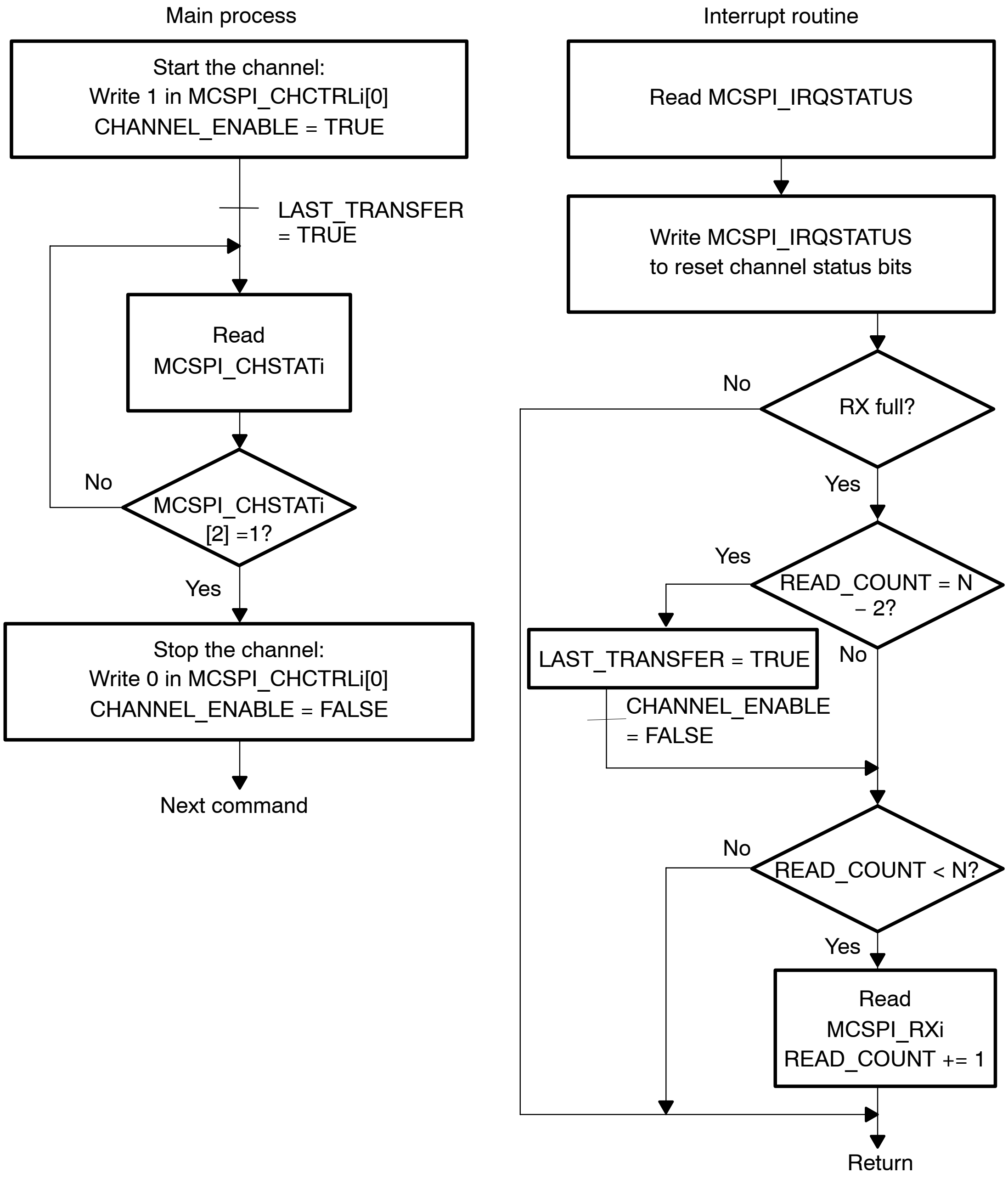


19.6.2.6 Master Turbo Receive Only

Based on Interrupt Requests

Figure 19−27 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 when all words are received for this mode.

Figure 19−27. Receive Only With Interrupt (Master Turbo)

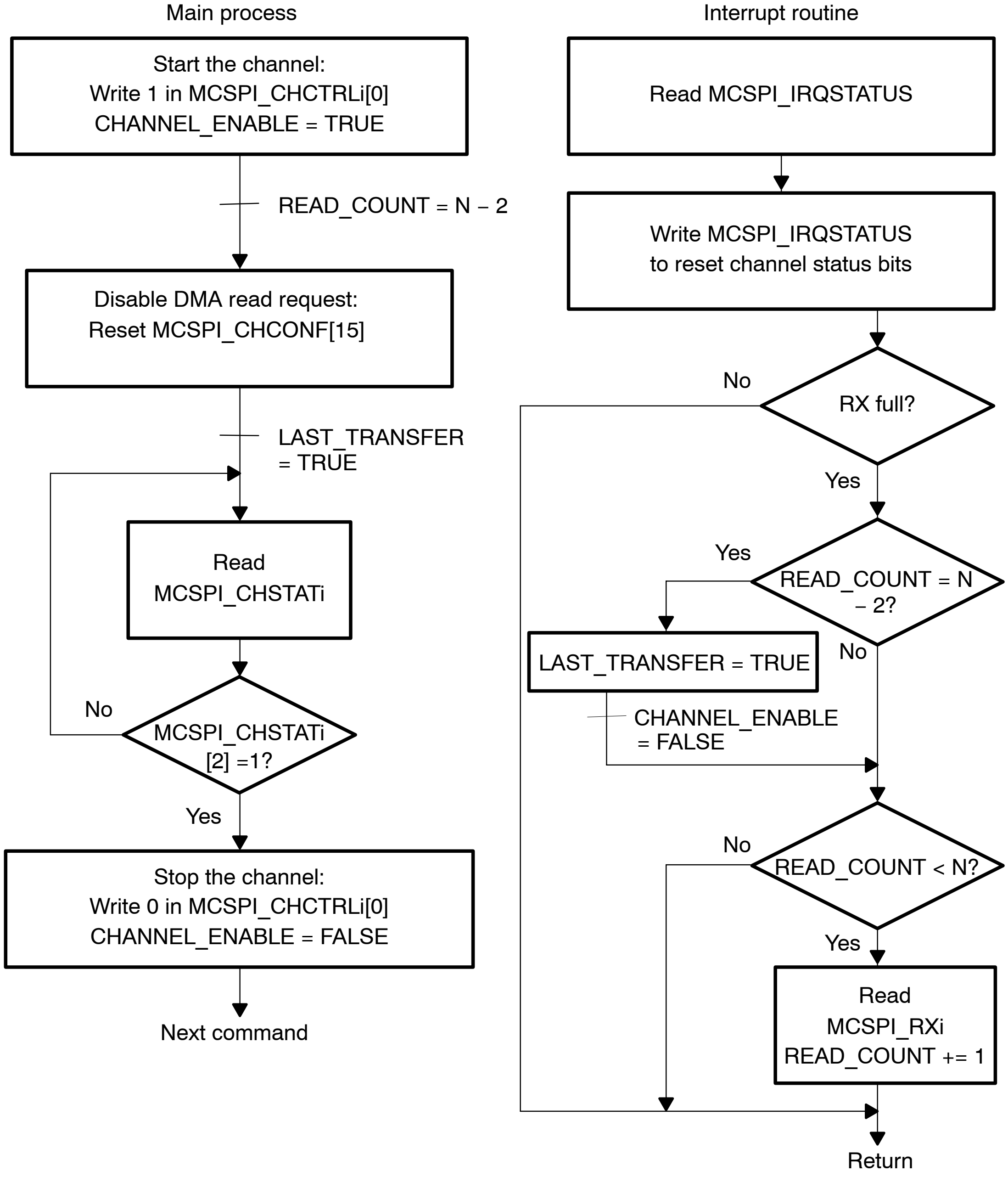


Based on DMA Read Requests

In Figure 19−28, the main process shows completion of word reception in master turbo receive-only mode with DMA write requests.

When the DMA handler completes N − 2 OCP accesses, READ_COUNT is assigned the value N − 2.

Figure 19−28. Receive Only With DMA (Master Turbo)

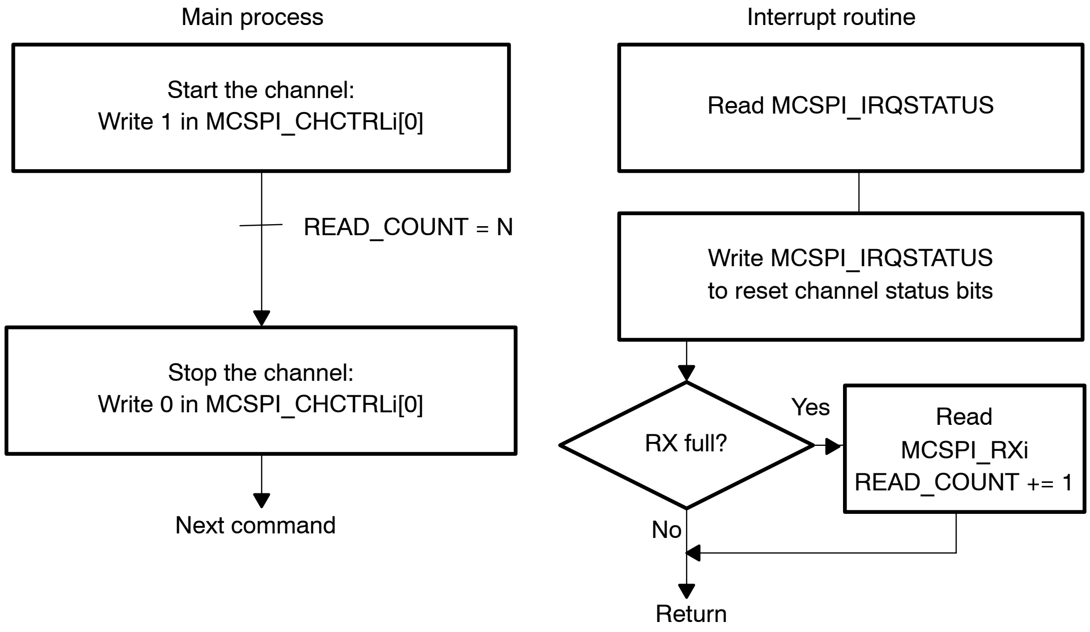


19.6.2.7 Slave Receive Only

Figure 19−29 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 when all words are received for this mode.

If the requests are configured in DMA, READ_COUNT is assigned the value N when the DMA handler completes N OCP processes.

Figure 19−29. Receive Only (Slave)



19.7 McSPI Registers

19.7.1 Instance Summary

Table 19−19 shows the base address and address space for the MPU module instances. Table 19−20 summarizes the McSPI registers.

Table 19−19. Instance Summary

Module Name	Base Address	Size
MCSPI1	0x4809 8000	256 bytes
MCSPI2	0x4809 A000	256 bytes

19.7.2 McSPI Offset Register Mapping Summary

Table 19−20. MCSPI1 Register Summary

Register Name	Type	Register Width (Bits)	Physical Address
MCSPI_REVISION	R	32	0x00
MCSPI_SYSCONFIG	RW	32	0x10
MCSPI_SYSSTATUS	R	32	0x14
MCSPI_IRQSTATUS	RW	32	0x18
MCSPI_IRQENABLE	RW	32	0x1C
MCSPI_WAKEUPENABLE	RW	32	0x20
MCSPI_SYST	RW	32	0x24
MCSPI_MODULCTRL	RW	32	0x28
MCSPI_CHCONF0 − MCSPI_CHCONF3	RW	32	0x2C− 0x68
MCSPI_CHSTAT0 − MCSPI_CHSTAT3	RW	32	0x30− 0x6C
MCSPI_CHCTRL0 − MCSPI_CHCTRL3	RW	32	0x34− 0x70
MCSPI_TX0 − MCSPI_TX3	RW	32	0x38− 0x74
MCSPI_RX0 − MCSPI_RX3	RW	32	0x3C− 0x78

19.7.3 Register Descriptions

Each SPI module has a set of registers (see the corresponding base address module). Table 19−21 through Table 19−33 contain the details of these registers.

Table 19−21. MCSPI_REVISION

Address Offset	0x00
Description	This register contains the hard-coded RTL 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.	RW	0x000000
7:0	REV	IP revision [7:4] indicates a major revision. [3:0] indicates a minor revision. Examples: 0x10 for 1.0, 0x21 for 2.1	R	

Table 19−22. MCSPI_SYSCONFIG

Address Offset	0x10
Description	This register allows control of the OCP interface 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																						CLOCKACTIVITY		RESERVED			SIDELMODE		ENAWAKEUP	SOFTRESET	AUTOIDLE

Bits	Field Name	Description	Type	Reset
31:10	Reserved	Read returns 0.	RW	0x000000
9:8	CLOCKACTIVITY	Clock activity during wake-up mode period 0x0: Interface clock and functional clock can be switched off. 0x1: Interface clock is maintained. Functional clock can be switched off. 0x2: Functional clock is maintained. Interface clock can be switched off. 0x3: Interface and functional clocks are maintained.	RW	0x0

Table 19−22. MCSPI_SYSCONFIG (Continued)

Bits	Field Name	Description	Type	Reset
7:5	Reserved	Read returns 0.	RW	0x0
4:3	SIDLEMODE	Power management 0x0: If an idle request is detected, the McSPI goes into inactive mode. Interrupt, DMA requests, and wake-up lines are unconditionally deasserted, 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. The module does not switch to wake-up mode, and continues to behave normally. 0x2: If an idle request is detected, the module switches 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.	RW	0x0
2	ENAWAKEUP	Wake-up feature control 0x0: Wake-up capability is disabled. 0x1: Wake-up capability is enabled.	RW	0
1	SOFTRESET	Software reset 0x0: Normal mode 0x1: The module is reset. Set this bit to 1 to trigger a module reset. The bit is automatically reset by the hardware. During reset, it always returns 0.	RW	0
0	AUTOIDLE	Internal OCP clock-gating strategy 0x0: OCP clock is free-running 0x1: Automatic OCP clock-gating strategy is applied, based on OCP interface activity.	RW	0

Table 19–23. MCSPI_SYSSTATUS

Address Offset	0x14
Description	This register provides status information about the module, except for 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	Read returns 0.	RW	0x00000000
0	RESETDONE	Internal reset monitoring 0x0: Internal module reset is ongoing. 0x1: Reset complete[1]	R	0

[1] During reset, the value is 0, but the value read just after the reset is 1, because ICLK/FCLK is already operating.

Table 19−24. MCSPI_IRQSTATUS

Address Offset	0x18
Description	The interrupt status regroups the statuses of module internal events that can generate 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															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
31:17	Reserved	Read returns 0.	RW	0x0000
16	WKS	Wake-up event in slave mode only. Active control signal detected on the SPI.nCS line programmed in the MCSPI_CHCONF0[SPIENSLV] field. Read 0x0: R: Event is false. Write 0x0: W: Event status bit is unchanged. Read 0x1: R: Event is pending. Write 0x1: W: Event status bit is reset.	RW	0
15	Reserved	Reads return 0	RW	0
14	RX3_FULL	Receiver register is full. Only when channel 3 is enabled. Read 0x0: R: Event is false. Write 0x0: W: Event status bit is unchanged. Read 0x1: R: Event is pending. Write 0x1: W: Event status bit is reset.	RW	0
13	TX3_UNDERFLOW	Transmitter register underflow. Only when channel 3 is enabled. The transmitter 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 is loaded into the transmitter register, because the channel is enabled. Read 0x0: R: Event is false. Write 0x0: W: Event status bit is unchanged. Read 0x1: R: Event is pending. Write 0x1: W: Event status bit is reset.	RW	0

Table 19−24. MCSPI_IRQSTATUS (Continued)

Bits	Field Name	Description		Type	Reset
12	TX3_EMPTY	Transmitter register is empty. **Note**: Enabling the channel automatically causes this event.		RW	0
		Read 0x0:	R: Event is false.		
		Write 0x0:	W: Event status bit is unchanged.		
		Read 0x1:	R: Event is pending.		
		Write 0x1:	W: Event status bit is reset.		
11	Reserved			RW	0
10	RX2_FULL	Receiver register full. Channel 2.		RW	0
		Read 0x0:	R: Event is false.		
		Write 0x0:	W: Event status bit is unchanged.		
		Read 0x1:	R: Event is pending.		
		Write 0x1:	W: Event status bit is reset.		
9	TX2_ UNDERFLOW	Transmitter register underflow. Channel 2.		RW	0
		Read 0x0:	R: Event is false.		
		Write 0x0:	W: Event status bit is unchanged.		
		Read 0x1:	R: Event is pending.		
		Write 0x1:	W: Event status bit is reset.		
8	TX2_EMPTY	Transmitter register empty. Channel 2.		RW	0
		Read 0x0:	R: Event is false.		
		Write 0x0:	W: Event status bit is unchanged.		
		Read 0x1:	R: Event is pending.		
		Write 0x1:	W: Event status bit is reset.		
7	Reserved	Reads return 0		RW	0
6	RX1_FULL	Receiver register full. Channel 1		RW	0
		Read 0x0:	R: Event false		
		Write 0x0:	W: Event status bit unchanged		
		Read 0x1:	R: Event is pending		
		Write 0x1:	W: Event status bit is reset		

Table 19−24. MCSPI_IRQSTATUS (Continued)

Bits	Field Name	Description		Type	Reset
5	TX1_ UNDERFLOW	Transmitter register underflow. Channel 1.		RW	0
		Read 0x0:	R: Event is false.		
		Write 0x0:	W: Event status bit is unchanged.		
		Read 0x1:	R: Event is pending.		
		Write 0x1:	W: Event status bit is reset.		
4	TX1_EMPTY	Transmitter register empty. Channel 1.		RW	0
		Write 0x0:	W: Event status bit is unchanged.		
		Read 0x0:	R: Event is false.		
		Read 0x1:	R: Event is pending.		
		Write 0x1:	W: Event status bit is reset.		
3	RX0_ OVERFLOW	Receiver register overflow (slave mode only). Channel 0.		RW	0
		Read 0x0:	R: Event is false.		
		Write 0x0:	W: Event status bit is unchanged.		
		Read 0x1:	R: Event is pending.		
		Write 0x1:	W: Event status bit is reset.		
2	RX0_FULL	Receiver register full. Channel 0.		RW	0
		Read 0x0:	R: Event is false.		
		Write 0x0:	W: Event status bit is unchanged.		
		Read 0x1:	R: Event is pending.		
		Write 0x1:	W: Event status bit is reset.		
1	TX0_ UNDERFLOW	Transmitter register underflow. Channel 0.		RW	0
		Read 0x0:	R: Event is false.		
		Write 0x0:	W: Event status bit is unchanged.		
		Read 0x1:	R: Event is pending.		
		Write 0x1:	W: Event status bit is reset.		
0	TX0_EMPTY	Transmitter register empty. Channel 0.		RW	0
		Read 0x0:	R: Event is false.		
		Write 0x0:	W: Event status bit is unchanged.		
		Read 0x1:	R: Event is pending.		
		Write 0x1:	W: Event status bit is reset.		

Table 19−25. MCSPI_IRQENABLE

Address Offset	0x1C
Description	This register allows enabling/disabling of the module internal sources of interrupt, 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															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
31:17	Reserved	Read returns 0.	RW	0x0000
16	WKE	Wake-up event interrupt enable in slave mode when an active control signal is detected on the SPI.nCS line programmed in the MCSPI_CHCONF0[SPIENSLV] field. 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0
15	Reserved	Receiver register overflow interrupt enable	RW	0
14	RX3_FULL_ENABLE	Receiver register full interrupt enable. Channel 3. 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0
13	TX3_UNDERFLOW_ENABLE	Transmitter register underflow interrupt enable. Channel 3. 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0
12	TX3_EMPTY_ENABLE	Transmitter register empty interrupt enable. Channel 3. 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0
11	Reserved	Reads return 0	RW	0
10	RX2_FULL_ENABLE	Receiver register full interrupt enable. Channel 2. 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0

Table 19−25. MCSPI_IRQENABLE (Continued)

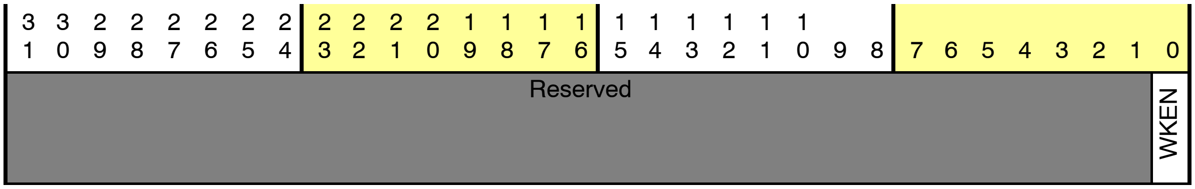
Bits	Field Name	Description	Type	Reset
9	TX2_UNDERFLOW_ENABLE	Transmitter register underflow interrupt enable. Channel 2. 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0
8	TX2_EMPTY_ENABLE	Transmitter register empty interrupt enable. Channel 2. 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0
7	Reserved	Reads return 0	RW	0
6	RX1_FULL_ENABLE	Receiver register full interrupt enable. Channel 1. 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0
5	TX1_UNDERFLOW_ENABLE	Transmitter register underflow interrupt enable. Channel 1. 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0
4	TX1_EMPTY_ENABLE	Transmitter register empty interrupt enable. Channel 1. 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0
3	RX0_OVERFLOW_ENABLE	Receiver register overflow interrupt enable. Channel 0. 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0
2	RX0_FULL_ENABLE	Receiver register full interrupt enable. Channel 0. 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0

Table 19–25. MCSPI_IRQENABLE (Continued)

Bits	Field Name	Description	Type	Reset
1	TX0_ UNDERFLOW_ ENABLE	Transmitter register underflow interrupt enable. Channel 0. 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0
0	TX0_EMPTY_ ENABLE	Transmitter register empty interrupt enable. Channel 0. 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0

Table 19–26. MCSPI_WAKEUPENABLE

Address Offset	0x20		
Physical Address	0x0000 0020	**Instance**	MCSPI
Description	The wake-up enable register allows enabling/disabling the module internal sources of wakeup on an event-by-event basis.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:1	Reserved	Read returns 0.	RW	0x00000000
0	WKEN	Wake-up functionality in slave mode when an active control signal is detected on the SPI.nCS line programmed in the MCSPI_CHCONF0[SPIENSLV] field 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

This register is implemented for future compatibility when more than one source of wakeup is defined, to improve power management.

Table 19–27. MCSPI_SYST

Address Offset	0x24		
Physical Address	0x0000 0024	**Instance**	MCSPI
Description	This register is used to check the correctness of the system interconnect either internally to peripheral bus, or externally to the device I/O pads, when the module is configured in system test (SYSTEST) 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																				SSB	SPIENDIR	SPIDATDIR1	SPIDATDIR0	WAKD	SPICLK	SPIDAT_1	SPIDAT_0	SPIEN_3	SPIEN_2	SPIEN_1	SPIEN_0

Bits	Field Name	Description	Type	Reset
31:12	Reserved	Read returns 0.	RW	0x000000
11	SSB	Set the status bit. 0x0: No action. Writing 0 does not clear already set status bits; This bit must be cleared before trying 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 in the <MCSPI_IRQSTATUS> register.	RW	0
10	SPIENDIR	Set the direction of the SPI.nCS[3:0] lines and SPI.CLK line. 0x0: Output (as in master mode) 0x1: Input (as in slave mode)	RW	0
9	SPIDATDIR1	Set the direction of the SPI.SOMI. 0x0: Output 0x1: Input	RW	0
8	SPIDATDIR0	Set the direction of the SPI.SIMO. 0x0: Output 0x1: Input	RW	0

Table 19−27. MCSPI_SYST (Continued)

Bits	Field Name	Description	Type	Reset
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	SPI.CLK line (signal data value) If MCSPI_SYST[SPIENDIR] = 0 (input mode direction), this bit returns the value on the SPI.CLK line (high or low), and a write into this bit has no effect. If MCSPI_SYST[SPIENDIR] = 1 (output mode direction), the CLK.SPI line is driven high or low according to the value written into this register.	RW	0
5	SPIDAT_1	SPI.SIMO line (signal data value) If MCSPI_SYST[SPIDATDIR] = 0 (input mode direction), this bit returns the value on the SPI.SOMI line (high or low), and a write to this bit has no effect. If MCSPI_SYST[SPIDATDIR] = 1 (output mode direction), the SPIDAT[1] line is driven high or low according to the value written into this register. A write to this bit has no effect.	RW	0
4	SPIDAT_0	SPI.SOMI line (signal data value) If MCSPI_SYST[SPIDATDIR] = 0 (input mode direction), this bit returns the value on the SPI.SIMO line (high or low), and a write into this bit has no effect. If MCSPI_SYST[SPIDATDIR] = 1 (output mode direction), the SPIDAT[0] line is driven high or low according to the value written into this register. A write to this bit has no effect.	RW	0
3	SPIEN_3	SPI.nCS[3] line (signal data value) If MCSPI_SYST[SPIENDIR] = 0 (input mode direction), this bit returns the value on the SPI.nCS[3] line (high or low), and a write into this bit has no effect. If MCSPI_SYST[SPIENDIR] = 1 (output mode direction), the SPI.nCS[3] line is driven high or low according to the value written into this register. A write to this bit has no effect.	RW	0
2	SPIEN_2	SPI.nCS[2] line (signal data value) If MCSPI_SYST[SPIENDIR] = 0 (input mode direction), this bit returns the value on the SPI.nCS[2] line (high or low), and a write to this bit has no effect. If MCSPI_SYST[SPIENDIR] = 1 (output mode direction), the SPI.nCS[2] line is driven high or low according to the value written into this register. A write to this bit has no effect.	RW	0

Table 19−27. MCSPI_SYST (Continued)

Bits	Field Name	Description	Type	Reset
1	SPIEN_1	SPI.nCS[1] line (signal data value) If MCSPI_SYST[SPIENDIR] = 0 (input mode direction), this bit returns the value on the SPI.nCS[1] line (high or low), and a write to this bit has no effect. If MCSPI_SYST[SPIENDIR] = 1 (output mode direction), the SPI.nCS[1] line is driven high or low according to the value written into this register. A write to this bit has no effect.	RW	0
0	SPIEN_0	SPI.nCS[0] line (signal data value) If MCSPI_SYST[SPIENDIR] = 0 (input mode direction), this bit returns the value on the SPI.nCS[0] line (high or low), and a write to this bit has no effect. If MCSPI_SYST[SPIENDIR] = 1 (output mode direction), the SPI.nCS[0] line is driven high or low according to the value written into this register. A write to this bit has no effect.	RW	0

Table 19−28. MCSPI_MODULCTRL

Address Offset	0x28
Description	This register is dedicated to the configuration of the serial port 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																												SYSTEM_TEST	MS	RESERVED	SINGLE

Bits	Field Name	Description		Type	Reset
31:4	Reserved	Read 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 SPI.CLK and SPI.nCS[3:0].		
		0x1:	Slave − The module receives the SPI.CLK and SPI.nCS[3:0].		
1	Reserved	Returns 0 after writing 0; returns 1 after writing 1		RW	0

Table 19–28. MCSPI_MODULCTRL (Continued)

Bits	Field Name	Description	Type	Reset
0	Single	Read returns 0. 0x0: More than one channel is used in master mode. 0x1: Only one channel is used in master mode. This bit must be set in force-SPI.nCS mode.	RW	0

Table 19–29. MCSPI_CHCONF0–MCSPI_CHCONF3

Address Offset	0x2C–0x68 in 0x14 byte increments		
Physical Address	0x0000 002C–0x0000 0068	**Instance**	MCSPI
Description	This register is dedicated to the configuration of channel i i={0,1,2,3}.		
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									SPIENSLV		FORCE	TURBO	IS	DPE1	DPE0	DMAR	DMAW	TRM		WL					EPOL	CLKD				POL	PHA

Bits	Field Name	Description	Type	Reset
31:23	Reserved	Read returns 0.	RW	0x000
22:21	SPIENSLV	Channel 0 only and slave mode only: SPI slave select signal detection. Reserved bits for other cases. 0x0: Detection enabled only on SPI.nCS0 0x1: Detection enabled only on SPI.nCS1 0x2: Detection enabled only on SPI.nCS2 0x3: Detection enabled only on SPI.nCS3	RW	0x0
20	FORCE	Manual SPI.nCS assertion to keep SPI.nCS active between SPI words (single-channel master mode only). 0x0: Writing 0 to this bit drives low the SPI.nCS line when MCSPI_CHCONFi[EPOL] = 0 and drives it high when MCSPI_CHCONFi[EPOL] = 1. 0x1: Writing 1 to this bit drives high the SPI.nCS line when MCSPI_CHCONFi[EPOL] = 0 and drives it low when MCSPI_CHCONFi[EPOL] = 1.	RW	0x0

Table 19−29. MCSPI_CHCONF0−MCSPI_CHCONF3 (Continued)

Bits	Field Name	Description	Type	Reset
19	TURBO	Turbo mode 0x0: Turbo is deactivated (recommended for single SPI word transfer). 0x1: Turbo is activated to maximize throughput for multi-SPI-word transfers.	RW	0
18	IS	Input select 0x0: Data line 0 (SPI.SOMI) selected for reception 0x1: Data line 1 (SPI.SIMO) selected for reception	RW	1
17	DPE1	Transmission enable for data line 1 (SPIDATA GZEN[1]) 0x0: Data line1 (SPI.SIMO) selected for transmission 0x1: No transmission on data line 1 (SPI.SIMO)	RW	1
16	DPE0	Transmission enable for data line 0 (SPIDATA GZEN[0]) 0x0: Data line 0 (SPI.SOMI selected for transmission 0x1: No transmission on data line 0 (SPI.SOMI)	RW	0
15	DMAR	DMA read request The DMA read request line is asserted when the channel is enabled and 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	0
14	DMAW	DMA write request The DMA write request line is asserted when the channel is enabled and the transmitter register of the channel is empty. The DMA write request line is deasserted on load completion of the transmitter register of the channel. 0x0: DMA write request disabled 0x1: DMA write request enabled	RW	0

Table 19−29. MCSPI_CHCONF0−MCSPI_CHCONF3 (Continued)

Bits	Field Name	Description	Type	Reset
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 bits long. 0x4: The SPI word is 5 bits long. 0x5: The SPI word is 6 bits long. 0x6: The SPI word is 7 bits long. 0x7: The SPI word is 8 bits long. 0x8: The SPI word is 9 bits long. 0x9: The SPI word is 10 bits long. 0xA: The SPI word is 11 bits long. 0xB: The SPI word is 12 bits long. 0xC: The SPI word is 13 bits long. 0xD: The SPI word is 14 bits long. 0xE: The SPI word is 15 bits long. 0xF: The SPI word is 16 bits long. 0x10: The SPI word is 17 bits long. 0x11: The SPI word is 18 bits long. 0x12: The SPI word is 19 bits long. 0x13: The SPI word is 20 bits long. 0x14: The SPI word is 21 bits long. 0x15: The SPI word is 22 bits long. 0x16: The SPI word is 23 bits long. 0x17: The SPI word is 24 bits long. 0x18: The SPI word is 25 bits long. 0x19: The SPI word is 26 bits long.	RW	0x00

Table 19−29. MCSPI_CHCONF0−MCSPI_CHCONF3 (Continued)

Bits	Field Name	Description		Type	Reset
		0x1A:	The SPI word is 27 bits long.		
		0x1B:	The SPI word is 28 bits long.		
		0x1C:	The SPI word is 29 bits long.		
		0x1D:	The SPI word is 30 bits long.		
		0x1E:	The SPI word is 31 bits long.		
		0x1F:	The SPI word is 32 bits long.		
6	EPOL	SPI.nCS polarity		RW	0
		0x0:	SPI.nCS is held high during the active state.		
		0x1:	SPI.nCS is held low during the active state.		
5:2	CLKD	Frequency divider for SPI.CLK (only when the module is a master SPI device). A programmable clock divider divides the SPI reference clock (SPI.FCLK) with a 4-bit value and results in a new clock SPI.CLK 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	SPI.CLK polarity		RW	0
		0x0:	SPI.CLK is held high during the active state.		
		0x1:	SPI.CLK is held low during the active state.		

Table 19–29. MCSPI_CHCONF0–MCSPI_CHCONF3 (Continued)

Bits	Field Name	Description	Type	Reset
0	PHA	SPI.CLK phase 0x0: Data are latched on odd-numbered edges of SPI.CLK. 0x1: Data are latched on even-numbered edges of SPI.CLK.	RW	0

Table 19–30. MCSPI_CHSTAT0–MCSPI_CHSTAT3

Address Offset	0x30–0x6C in 0x14 byte increments		
Physical Address	0x0000 0030–0x0000 0000	**Instance**	MCSPI
Description	This register provides status information about transmitter and receiver registers of channel i i={0,1,2,3}.		
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																													EOT	TXS	RXS

Bits	Field Name	Description	Type	Reset
31:3	Reserved	Read returns 0.	RW	0x00000000
2	EOT	Channel i end-of-transfer status. The definitions of beginning and end of transfer vary with master versus slave and the transfer format (transmit/receive modes, turbo mode). 0x0: This flag is automatically cleared when the shift register is loaded with data from the transmitter register (beginning of transfer). 0x1: This flag is automatically set to 1 at the end of an SPI transfer.	R	0
1	TXS	Channel i transmitter register status 0x0: Register is full. 0x1: Register is empty.	R	0
0	RXS	Channel i receiver register status 0x0: Register is empty. 0x1: Register is full.	R	0

Table 19−31. MCSPI_CHCTRL0−MCSPI_CHCTRL3

Address Offset	0x34−0x70 in 0x14 byte increments
Description	This register is dedicated to enable channel i i={0,1,2,3}.
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																															EN

Bits	**Field Name**	**Description**	**Type**	**Reset**
31:1	Reserved	Read returns 0.	RW	0x00000000
0	EN	Channel enable 0x0: Channel i is not active. 0x1: Channel i is active.	RW	0

Table 19−32. MCSPI_TX0−MCSPI_TX3

Address Offset	0x38−0x74 in 0x14 byte increments
Description	This register contains a single SPI word to transmit on the serial link, regardless of SPI word length.
Type	RW

31	30	29	28	27	26	25	24	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 i data to transmit	RW	0x00000000

Table 19−33. MCSPI_RX0−MCSPI_RX3

Address Offset	0x3C−0x78 in 0x14 byte increments
Description	This register contains a single SPI word received through the serial link, regardless of SPI word length.
Type	RW

31	30	29	28	27	26	25	24	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 i received data	R	0x00000000

Chapter 20

HDQ/1-Wire

This chapter describes the features and functions of the HDQ/1-Wire module on the OMAP2420 device.

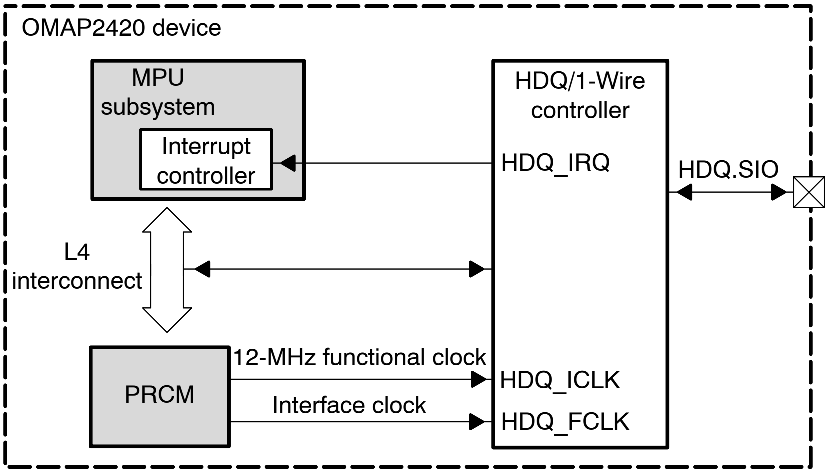
Topic **Page**

20.1 HDQ/1-Wire Overview

This module implements the hardware protocol of the master functions of the Benchmark 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 20−1 shows the HDQ/1-Wire controller module.

Figure 20−1. HDQ/1-Wire Highlight



The HDQ/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 or 13-MHz functional clock provided by the power, reset, and clock management (PRCM) module.

Only the MPU subsystem uses the HDQ/1-Wire module.

The main features of the HDQ/1-Wire module are:

- Supports Benchmark HDQ protocol
- Supports Dallas Semiconductor 1-Wire protocol
- Supports power-down mode

The HDQ/1-Wire module provides a communication rate of 5K bits/s over an address space of 128 bytes.

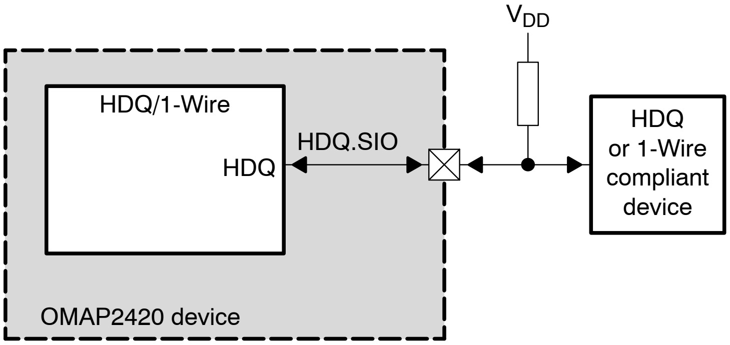
A typical application of the HDQ/1-Wire module is communication with battery monitor (gas gauge) integrated circuits.

20.2 HDQ/1-Wire Environment

20.2.1 HDQ/1-Wire Functional Interface

Figure 20−2 shows a typical application using an HDQ/1-Wire connection.

Figure 20−2. HDQ/1-Wire Typical Application System Overview



As the two protocols use a return-to-1 mechanism (that is, after any command, the line is pulled-up to a logical high level), an external pullup is required.

The HDQ/1-Wire module operates according to a command structure that is programmed into transmit command registers (as described in the programming model).

Table 20−1. I/O Description

Signal Name	I/O	Description	Value at Reset
HDQ.SIO	Bidirectional	Serial data input/output	0

20.2.2 HDQ and 1-Wire (SDQ) Protocols

20.2.2.1 HDQ Protocol Initialization

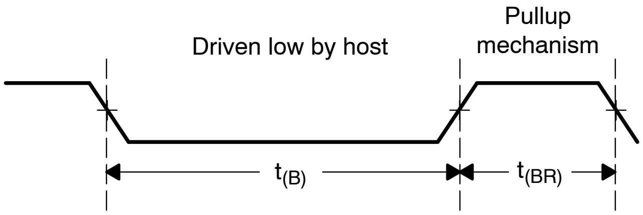
The HDQ is a command-based protocol, where the host sends a command byte to the slave. The command directs the slave either to store the next eight bits of data received in a register specified by the command byte (write operation) or to output the eight bits of data from a register specified by the command byte (read operation).

Command and data bytes consist of a stream of eight bits with a maximum transmission rate of 5K bits/s. The least-significant bit 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 a 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 resending the command.

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 20−3 shows this behavior.

Figure 20−3. HDQ BREAK-Pulse Timing Diagram

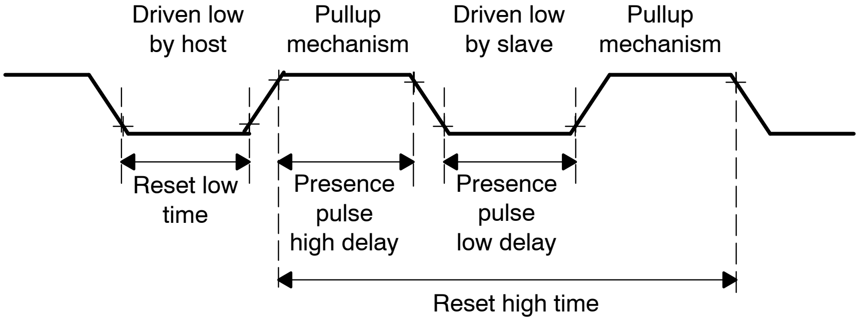


20.2.2.2 1-Wire (SDQ) Protocol Initialization

In the 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 with the initialization pulse, the presence pulse is a low-going edge on the line initiated by the slave. Figure 20−4 shows the 1-Wire (SDQ) reset sequence.

Figure 20−4. 1-Wire (SDQ) Reset Timing Diagram



The host drives the line to a logic-low state for at least reset low time. Once the slave has detected this pulse, it must drive the line to a logic-low state within the presence pulse high delay for at least a period of presence pulse low.

If the slave does not respond within this time interval, 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.

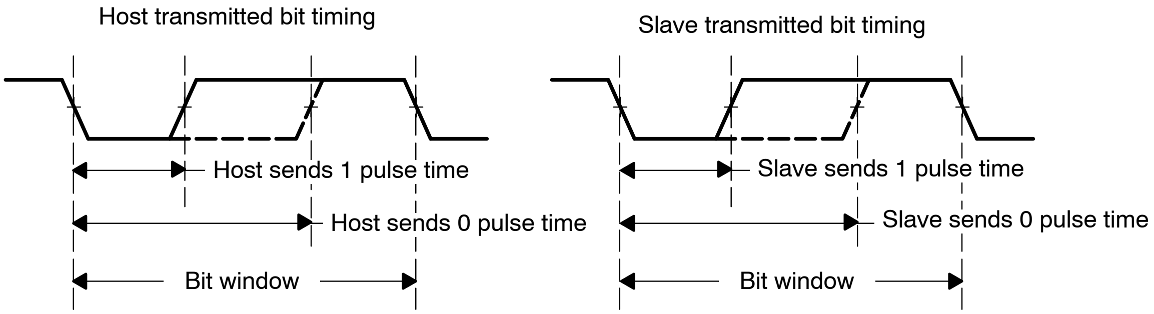
On the other hand, if the slave sends back its presence pulse in the specified time interval, the communication can be enabled after the reset high time.

20.2.2.3 Communication Sequence (Both HDQ and 1-Wire Protocols)

This section describes the common part of the two protocols.

After a successful BREAK-pulse (HDQ mode) or initialization sequence (1-Wire protocol), the host and the 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 20–5.

Figure 20–5. HDQ/1-Wire Transmitted Bit Timing



The return-to-one data-bit frame consists of three distinct sections. The first section starts the transmission by either the slave or the host taking the line to a logic-low state.

The next section is the actual data transmission, where the data must be valid by a specified period of time after the negative edge that starts 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 least-significant bit 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 of the memory location to read from or write to. The command byte values are shown in Table 20–2.

Table 20–2. HDQ/1-Wire Command Byte

Command Byte							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
R/W	AD6	AD5	AD4	AD3	AD2	AD1	AD0

R/W

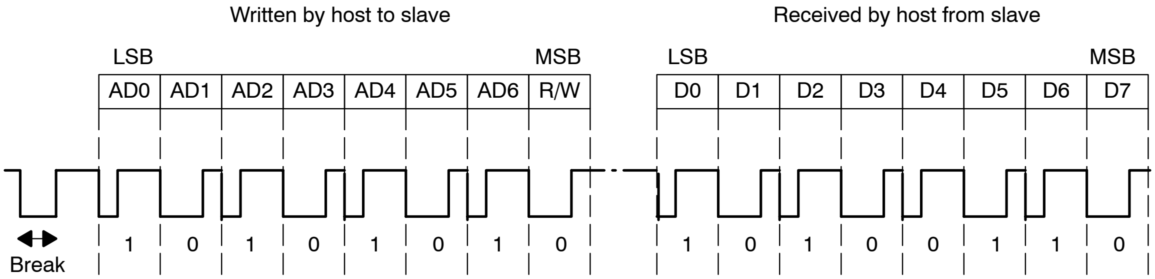
Indicates whether the command byte is a read or a write. A 1 indicates a write command, and the following eight bits must be written to the register specified by the address field of the command byte. A 0 indicates that the command is a read. On a read command, the slave outputs the requested register contents.

AD6−AD0

The seven bits labeled AD6−AD0 contain the address portion of the register to be accessed.

Figure 20−6 shows a read command at address 55h; received data is at address 65h.

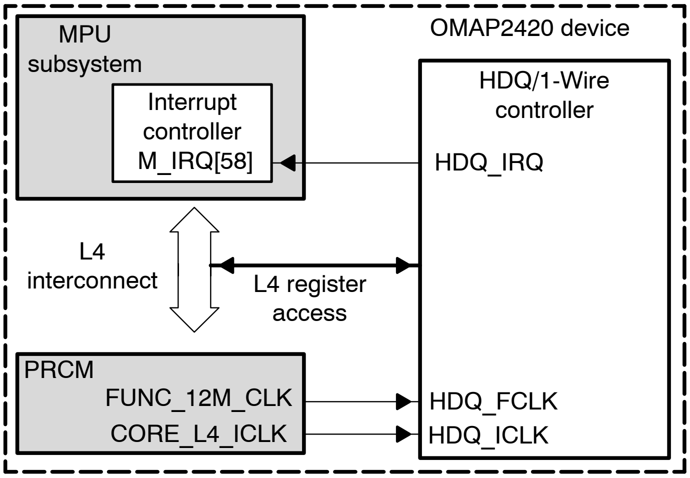
Figure 20−6. HDQ/1-Wire Communication Sequence



20.3 HDQ/1-Wire Integration

Figure 20−7 shows the HDQ/1-Wire module integration in the OMAP2420 device.

Figure 20−7. HDQ/1-Wire Module Integration



20.3.1 Clocking, Reset, and Power Management Scheme

20.3.1.1 HDQ/1-Wire Clocks

There are two clock domains in the HDQ/1-Wire module: The functional clock domain and the interface clock domain.

- 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 FUNC_12M_CLK PRCM output, derived from the FUNC_48M_CLK signal (a 1/4 ratio is applied). The FUNC_48_CLK source can be either an external source or the 96M APLL; selection occurs at the PRCM level using bit 48M_SOURCE (CM_CLKSEL1_PLL[3]) of register PRCM CM_CLKSEL1_PLL.

  When the HDQ/1-Wire module no longer requires HDQ_FCLK, the software can disable it at the PRCM level by setting the EN_HDQ bit (CM_FCLKEN1_CORE[23]) in the PRCM registers. The clock is only cut if it is not needed by other modules.

  For details about the PRCM register settings and APLL96M configuration, see Chapter 5, *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 can run at L3 or L3/2 frequency. Selection occurs using CLKSEL_L4 bit field (CM_CLKSEL1_CORE[6:5]) of register PRCM CM_CLKSEL1_CORE. When the HDQ/1-Wire module no longer requires the HDQ_ICLK, software can disable it at the PRCM level by setting the EN_HDQ bit (CM_ICLKEN1_CORE[23]) in the PRCM registers. The clock is cut only if it is not needed by other modules.

The autoidle mode can also be activated for this clock (PRCM CM_AUTOIDLE1_CORE register AUTO_HDQ bit CM_AUTOIDLE1_CORE[23] set to 1). In this case, HDQ_ICLK follows the OMAP2420 L4 clock domain behavior.

For more information about the HDQ_ICLK, see Chapter 5, *Power, Reset, and Clock Management*.

20.3.1.2 HDQ/1-Wire Reset Scheme

Global reset of the module is accomplished either by activation of the CORE_RST_N signal in the CORE_RST domain (see Chapter 5, *Power, Reset, and Clock Management,* for details) or by setting the HDQ/1-Wire HDQ_SYSCONFIG register soft reset bit (HDQ_SYSCONFIG[1]) to 1. Setting the soft reset bit enables active software reset functionality equivalent to a hardware reset.

20.3.1.3 HDQ/1-Wire Power Domain

The HDQ/1-Wire module belongs to the CORE power domain. For more information about the CORE power domain, see Chapter 5, *Power, Reset, and Clock Management*.

20.3.2 Hardware Requests

The HDQ/1-Wire module can generate one interrupt. HDQ_IRQ goes to the MPU subsystem interrupt controller. It is mapped on M_IRQ_58. For more details, see Table 20−3.

Table 20−3. Pin Multiplexing for HDQ/1-Wire Module

HDQ_ 1Wire Interface	**Description**	**DIR**	**Ball 2420**	**Alternate Functions**					
				Mode0	**Mode1**	**Mode2**	**Mode3**	**Mode4**	**Mode5**
HDQ.SIO	Bidir HDQ 1-Wire control and data interface. Output is open drain.	I/O	N18		USB2. TLLSE0	SYS. ALTCLK	GPIO. 101		

20.3.3 L4 Interconnect Interface

The HDQ/1-Wire module interface with the L4 interconnect through a dedicated target agent (TA). The TA is part of the L4 interconnect and provides status and configuration registers as listed in Table 20−4. By default, TA register values are functional but can be overwritten, if necessary. For a complete description, see Chapter 6, *Internal Interconnect*.

Table 20−4 lists the L4 interconnect registers.

Table 20−4. L4 Interconnect Registers

Register Name	Type	Register Width (Bits)	Physical Address
COMPONENT	R	32	0x480B 3000
AGENT_CONTROL	R/W	32	0x480B 3020
AGENT_STATUS	R	32	0x480B 3028

20.3.4 HDQ/1-Wire Register Summary

Table 20−5 lists the HDQ/1-Wire registers. See Section 20.6.2, *Register Descriptions*, for a detailed description of the registers.

Table 20−5. Register Summary for HDQ/1-Wire Module

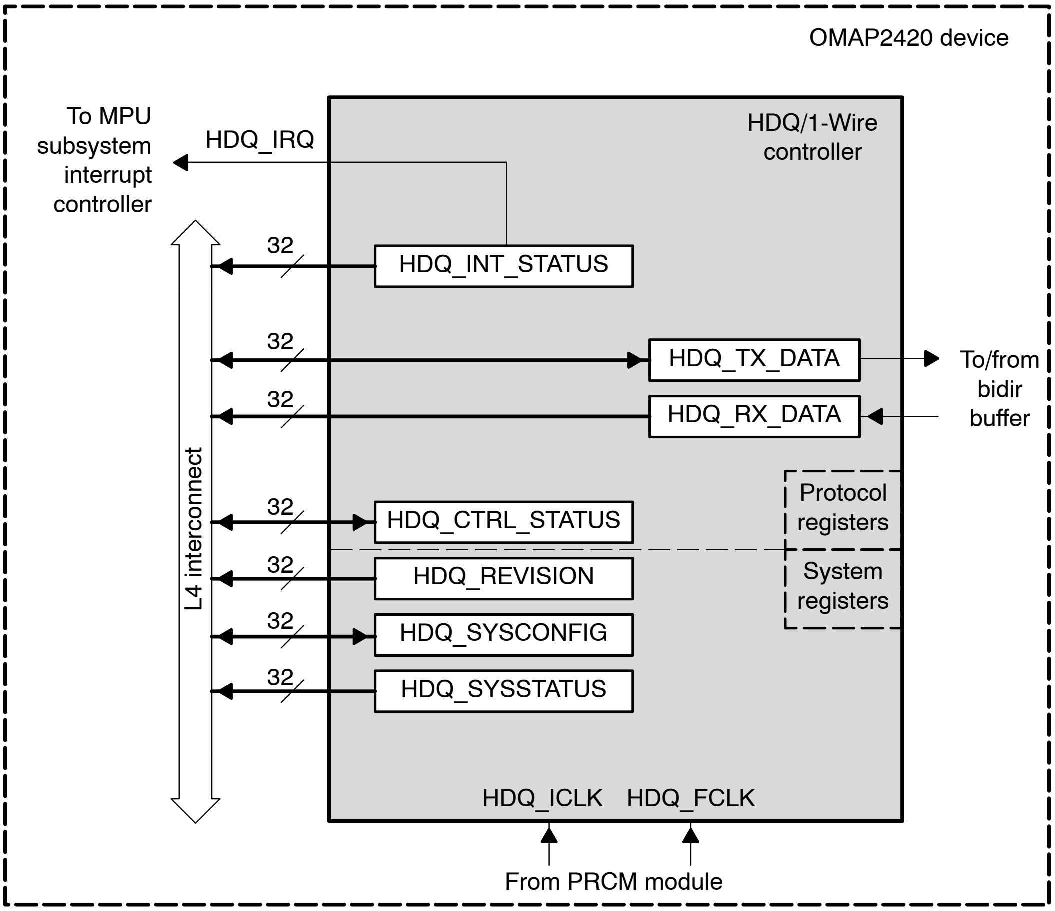
Register Name	Type	Register Width (Bits)	Physical Address
HDQ_REVISION	R	32	0x480B 2000
HDQ_TX_DATA	R/W	32	0x480B 2004
HDQ_RX_DATA	R	32	0x480B 2008
HDQ_CTRL_STATUS	R/W	32	0x480B 200C
HDQ_INT_STATUS	R	32	0x480B 2010
HDQ_SYSCONFIG	R/W	32	0x480B 2014
HDQ_SYSSTATUS	R	32	0x480B 2018

20.4 HDQ/1-Wire Functional Description

The HEQ/1-Wire module is intended to work with both the HDQ and 1-Wire protocols. The protocols use a single wire to establish communication between the master and the slave. Both use a return-to-1 mechanism; that is, after any command driven, the line is pulled up to a high level. This mechanism requires an external pullup.

Figure 20−8 shows the HDQ/1-Wire block diagram.

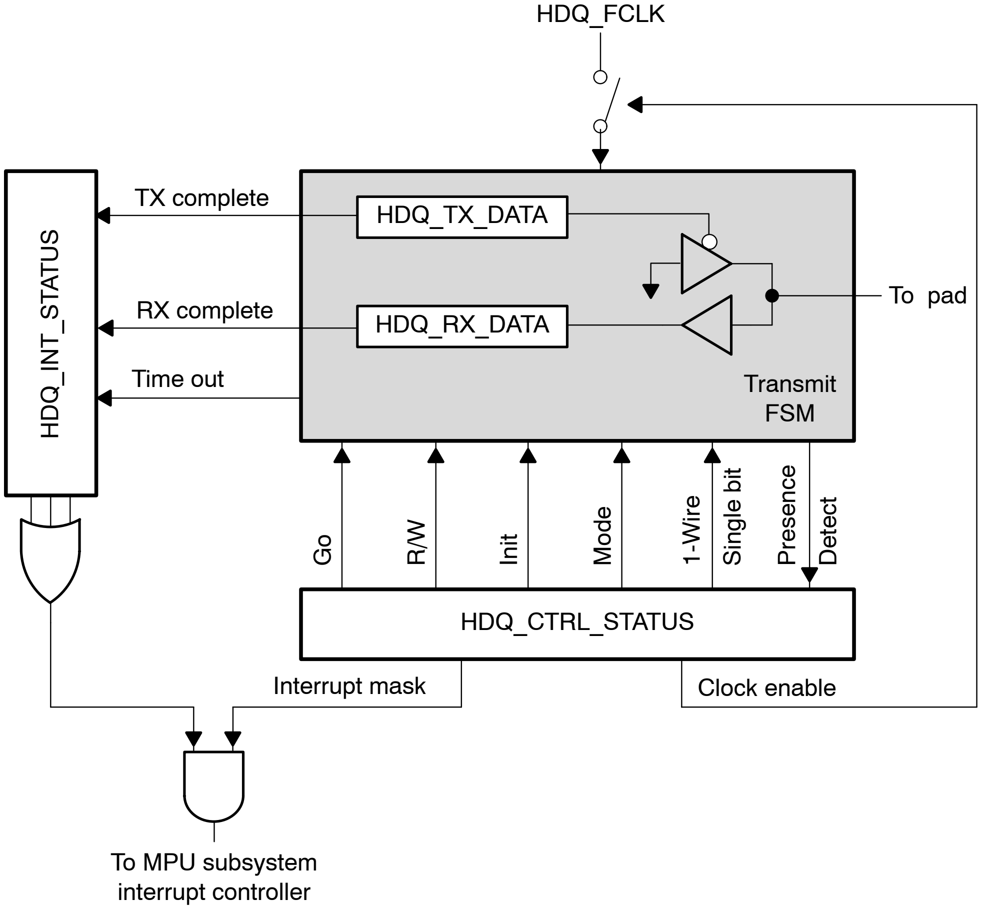
Figure 20−8. HDQ/1-Wire Block Diagram



A control bit in the HDQ/1-Wire control and status register (HDQ_CTRL_STATUS[0]) allows selection between the HDQ and 1-Wire protocols. For design purposes, the bit is assumed static. By default, the configuration is in HDQ mode.

Figure 20−9 shows the protocol-dedicated register scheme.

Figure 20−9. Protocol Registers



Once in the OMAP device, the module is implemented and clocked with a single clock, whether it runs HDQ or 1-Wire protocol. For 1-Wire mode, the data exchange is slower than the capabilities of this mode while still meeting the timing requirements and practical considerations. That is. the 1-Wire protocol runs at the slower HDG protocol speed (5K bits/s). But the timing parameters and protocol are different in the two modes.

20.4.1 HDQ Mode (Default)

20.4.1.1 HDQ Mode Features

- ❑ Supports Benchmark HDQ protocol
- ❑ Power-down mode

20.4.1.2 Description

In HDQ mode, there is no need for the host to create an initialization pulse to the slave. However, the host can reset the slave using an initialization pulse (also known as BREAK pulse). Setting the INITIALIZATION bit in the HDQ/1-Wire control and status register (HDQ_CTRL_STATUS[2]) creates this pulse by pulling the line down). When the slave receives it, it is ready for communication but does not respond with any presence pulse.

In a typical write operation, two bytes are sent to the slave. The first one corresponds to the command/address byte and the second one to the data to be written.

In a typical read operation, the host sends a command/address byte and the slave sends back a byte of data.

The master is implemented to send and receive bytes. The firmware sends command/address and data. The master only provides a single data TX register.

HDQ uses the 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 3-state line in the device is brought up to a logical high level by an external pullup.

During 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 any data within a specified time interval (that is, the slave does not drive the line low within this interval), a time-out status bit (HDQ_INT_STATUS[0]) is set in the HDQ/1-Wire HDQ_INT_STATUS register. It indicates a read failure. The time-out bit remains set until the host reads the interrupt status register (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_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 exits to enable or disable the interrupts. Interrupts cannot be individually masked.

20.4.1.3 Single-Bit Mode

In HDQ mode, the single-bit mode (HDQ_CTRL_STATUS[7] set to 1) has no effect, as the HDQ protocol only supports byte transfers.

20.4.1.4 Interrupt Conditions

The HDQ/1-Wire module provides three options for interrupt status:

1) Transmitter complete:

   A write operation of one byte was completed. Successful or failed completion is not indicated, as there is no acknowledge from the slave in HDQ/1-Wire protocol. This interrupt condition is cleared by reading the interrupt status register (HDQ_INT_STATUS).

2) Read complete:

   In HDQ mode, it indicates that a byte is successfully read. This interrupt condition is cleared by reading the interrupt status register (HDQ_INT_STATUS).

3) Presence detect/time-out:

   In HDQ mode, it indicates that the slave did not pull the line low within the specified time after a read command initiated by the host. This interrupt condition is cleared by reading the interrupt status register (HDQ_INT_STATUS).

Only one interrupt is generated to the MPU, based upon any of the above interrupt conditions. A read operation on the interrupt status register clears all the interrupt status bits that were previously set.

20.4.2 1-Wire Mode

20.4.2.1 1-Wire Mode Features

- ☐ Supports the Dallas Semiconductor 1-Wire protocol
- ☐ Power-down mode
- ☐ Supports single-bit mode

20.4.2.2 Description

The 1-Wire mode requires that an initialization pulse be sent to the slave(s) connected to the interface. If a slave is present, it responds with a presence pulse.

The initialization pulse is sent after the INITIALIZATION bit is set in the HDQ/1-Wire control and status register (HDQ_CTRL_STATUS[2]). The firmware sends the initialization pulse when the value of this bit is 1.

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 presence detect bit in the HDQ_CTRL_STATUS register (HDQ_CTRL_STATUS[3]).

In the same way, if a presence pulse is not received from the slave after an initialization pulse is sent, the presence detect bit remains cleared.

Whether a presence pulse is detected or not after an initialization pulse is sent, the time-out bit is set in the HDQ/1-Wire HDQ_INT_STATUS register (HDQ_INT_STATUS[0]) and an interrupt condition is generated.

In 1-Wire mode it means that the maximum time allowed for receiving the response has elapsed, so the software must check the presence detect bit to determine whether there was a presence pulse.

The INITIALIZATION bit is cleared at the end of the initialization pulse at the same time as the time-out bit is set. The time-out bit is cleared when the interrupt status register (HDQ_INT_STATUS) is read.

For read operations, 1-Wire is a bit-by-bit protocol. That means that for each bit of the byte to read, the slave must be clocked by the host.

The line is pulled up at the end of the command/address byte. On first read, the host creates a low-going edge to initiate a bit read. The line is then pulled up (pulled to 3-state mode 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 transmits 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 MSB in the receive HDQ/1-Wire data register (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 all the interrupt conditions previously set up. As in the HDQ mode, only one interrupt signal is sent to the MPU. Only an overall mask bit exists to enable or disable the interrupt (the interrupt conditions cannot be masked individually).

20.4.2.3 1-Wire Single-Bit Mode Operation

A single-bit mode can be entered by setting the appropriate bit in the control and status register (HDQ_CTRL_STATUS[7]). In this mode, only one bit of data is transferred each time 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 is updated each time a bit is received from the slave, whereas bit 0 of the TX register contains the bit to be sent.

20.4.2.4 Interrupt Conditions

The HDQ/1-Wire module provides three options for interrupt status:

1) Transmission complete:

   A write operation was completed. Successful or failed completion is not indicated, as there is no acknowledge from the slave in 1-Wire protocol. This interrupt condition is cleared by reading the interrupt status register (HDQ_INT_STATUS).

2) Read complete:

   In 1-Wire mode, it indicates that a byte wad successfully read. This interrupt condition is cleared by reading the interrupt status register (HDQ_INT_STATUS).

3) Presence detect/time-out:

   In 1-Wire mode, it indicates that it is now valid to check the presence detect received bit. This interrupt condition is cleared by reading the interrupt status register (HDQ_INT_STATUS).

   Only one interrupt is generated to the MPU, based upon any of the above interrupt conditions. A read operation on the interrupt status register clears all the interrupt status bits that were previously set.

20.4.2.5 Status Flags

The presence-condition-detected status flag is contained in the HDQ/1-Wire control and status register PRESENCEDETECT bit (HDQ_CTRL_STATUS[3]). This is valid only in 1-Wire mode. The flag is updated when the time-out bit is set. Thus, it shows the correct value 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.

20.4.3 Module Power Saving

20.4.3.1 Autoidle Mode

The HDQ/1-Wire module provides an autoidle facility in its interconnect clock domain.

The interconnect clock autoidle power-saving mode is enabled or disabled through the AUTOIDLE bit (HDQ_SYSCONFIG[0]) of the HDQ/1-Wire SYSCONFIG register. When this mode is enabled and there is no activity on the interconnect interface, the interconnect clock (HDQ_ICLK) is disabled inside the module, thus reducing power consumption. When there is new activity on the interconnect interface, the interconnect clock is restarted with no latency penalty. After a reset, this mode is disabled by default.

For reduced power consumption, it is recommended to enable this mode.

20.4.3.2 Power Down Mode

The HDQ/1-Wire module also provides a power-saving facility in its functional clock domain.

Setting the clock enable bit (HDQ_CTRL_STATUS[5]) in the HDQ/1-Wire control and status register to 0 shuts off the functional clock (HDQ_FCLK) to the state-machine. The state-machine is reset when the functional clock is disabled, and any ongoing transaction is aborted into the reset state.

The register values are not affected by disabling the functional clock.

There should be no access to the module registers after the software has put the module in power-down mode, other than a write to the clock enable bit to take it out of power-down mode.

20.4.4 System Power Management

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 details, see Chapter 5, *Power, Reset, and Clock Management*).

The HDQ/1-Wire module does not support handshake protocol with the PRCM. The HDQ/1-Wire module can only go into idle mode as part of the L4 interconnect clock domain (both CORE_L4_ICLK and FUNC_12M_CLK described in Section 20.3.1.1, *Clocking, Reset, and Power-Management Scheme*, belong to the L4 interconnect clock domain).

When the PRCM CM_AUTOIDLE1_CORE register AUTO_HDQ bit (CM_AUTOIDLE1_CORE[23]) 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.

The software must ensure correct clock management.

CAUTION

There is no hardware mechanism to prevent cutting the HDQ/1-Wire clocks while the module is performing a transfer.

See Section 20.5, *HDQ/1-Wire Programming Model*, for more information.

20.5 HDQ/1-Wire Programming Model

Because the module implements only the hardware interface layer for both HDQ and 1-Wire protocols, it can be considered a simple byte-engine. The firmware performs the correct sequencing, as described in this section.

20.5.1 Module Initialization Sequence

20.5.1.1 Mode Selection

The MODE bit in the HDQ/1-Wire control and status register (HDQ_CTRL_STATUS[0]) allows selection between HDQ and 1-Wire protocols. When set to 0, the protocol is HDQ; when set to 1, the protocol is 1-Wire. For design purposes, the bit is assumed static. 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.

20.5.1.2 Reset/Initialization

In HDQ mode, no slave presence test is required. However, the slave can be reset by setting the INITIALIZATION bit (HDQ_CTRL_STATUS[2]) in the HDQ_CTRL_STATUS register. 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 sends back a presence pulse when it receives the initialization pulse. The protocol for initialization is as follows:

Step 1: Set HDQ_CTRL_STATUS[2] to 1 to send the pulse. When the pulse is sent, the bit is cleared in the register.

Step 2: Wait for the presence-detect flag (HDQ_INT_STATUS[0]) to generate an interrupt. This flag is set when the time allowed for the slave to respond has elapsed, whether it has sent a pulse or not.

Step 3: Read the HDQ_CTRL_STATUS register to check whether the presence pulse is received before starting any transfer.

20.5.2 HDQ Protocol Programming Model

20.5.2.1 Write Operation

Step 1: Write the command/address or data value to the HDQ/1-Wire TX-write register (HDQ_TX_DATA).

Step 2: Set the DIR bit of the HDQ/1-Wire control and status register to 0 (HDQ_CTRL_STATUS[1]) to indicate a write.

Step 3: Set the GO bit (HDQ_CTRL_STATUS[4]) of the HDQ/1-Wire control and status register to 1 to start the transmission:

- The hardware sends the byte from the TX-write register.

- In a write operation, the time-out bit is always cleared, as there is no acknowledge mechanism from the slave.
- When the write operation is complete, the TX-complete flag (HDQ_INT_STATUS[2]) is set in the HDQ/1-Wire interrupt status register. If interrupts are masked (that is, the corresponding bit has been previously set in the control and status register), no interrupt signal is generated.

Note:

Steps 2 and 3 can be performed simultaneously.

Step 4: The software must read the interrupt status register to clear the interrupt.

Step 5: Repeat the above steps for each successive byte to write.

The GO bit of the control and status register is cleared at the end of a write operation.

20.5.2.2 Read Operation

Step 1: Write the command value to the HDQ/1-Wire TX-write register (HDQ_TX_DATA).

Step 2: In the HDQ/1-Wire control and status register, set the DIR bit (HDQ_CTRL_STATUS[1]) to 0, the GO bit (HDQ_CTRL_STATUS[4]) to 1, and wait for the TX-complete interrupt.

Step 3: Set the DIR bit (HDQ_CTRL_STATUS[1]) of the HDQ/1-Wire control and status register to 1 to indicate a read.

Step 4: Write (HDQ_CTRL_STATUS[4]) 1 to the GO bit of the HDQ/1-Wire control and status register to initiate the read.

- The hardware detects a low-going edge on the line (generated by the slave) and receives 8 bits of data in the HDQ/1-Wire RX receive buffer register (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 the GO bit. However, an RX-complete interrupt is generated only when the software writes the GO bit.
- If a time-out occurs, the time-out bit is set in the HDQ/1-Wire control and status register (HDQ_CTRL_STATUS).
- The completion of the operation is indicated by setting the RX-complete flag (HDQ_INT_STATUS[1]) in the HDQ/1-Wire interrupt status register. If interrupts are masked (that is, the corresponding bit has been previously set in the control and status register), no interrupt signal is generated.
- The GO bit is cleared at the end of the read operation It is also cleared if a time-out is detected.

Note:

Steps 3 and 4 can be performed simultaneously.

Step 5: The software must read the HDQ/1-Wire interrupt status register (HDQ_INT_STATUS) to determine whether an RX was successfully completed or a time-out occurred.

Step 6: The software reads the HDQ/1-Wire RX receive buffer register (HDQ_RX_DATA) to retrieve the read data from the slave.

Step 7: Repeat the above steps for each successive byte.

Note:

In HDQ16 mode, the address/command is only written 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.

20.5.3 1-Wire Mode (SDQ) Programming Model

20.5.3.1 Write Operation

Step 1: Write the ID, command, or data value to the HDQ/1-Wire TX-write register (HDQ_TX_DATA).

Step 2: Set the DIR bit (HDQ_CTRL_STATUS[1]) of the control and status register to 0 to indicate a write operation.

Note:

Steps 2 and 3 can be performed simultaneously.

Step 3: Set the GO bit (HDQ_CTRL_STATUS[4]) of the control and status register to 1 to start the transmission.

- The hardware sends the byte stored in the TX-write register.
- In a write operation, the time-out bit is always cleared.
- When the operation is completed, the TX-complete flag is set in the interrupt status register. If interrupts are masked (that is, the corresponding bit has been previously set in the control and status register), no interrupt signal is generated.
- The GO bit of the control and status register is cleared at the end of a write operation.

Step 4: The software must read the interrupt status register to clear the interrupt.

Step 5: Repeat the above steps for each successive byte to write.

20.5.3.2 Read Operation

Step 1: Write the address to the HDQ/1-Wire TX-write register (HDQ_TX_DATA).

Step 2: Set the DIR bit to 0 (HDQ_CTRL_STATUS[1]) and the GO bit (HDQ_CTRL_STATUS[4]) to 1 and wait for the TX-complete interrupt flag.

Step 3: Write the command value to the TX-write register.

Step 4: Set the DIR bit to 0 (HDQ_CTRL_STATUS[1]) and the GO bit (HDQ_CTRL_STATUS[4]) to 1 and wait for the TX-complete interrupt flag.

Step 5: Set the DIR bit (HDQ_CTRL_STATUS[1]) of the control and status register to 1 to indicate a read.

Step 6: Set the GO bit (HDQ_CTRL_STATUS[4]) of the control and status register to 1 to start the transmission.

- The hardware detects a low-going edge on the line (generated by the slave) and clocks 8 bits of data into the RX receive register. The first bit received is the LSB, and the last bit is the MSB.
- Once the operation is complete, the RX-complete flag is set in the interrupt status register. If interrupts are masked (that is, the corresponding bit has been previously set in the control and status register), no interrupt signal is generated.
- The GO bit is cleared at the end of the read. It is also cleared if a time-out occurs.

Note:

Steps 5 and 6 can be performed simultaneously.

Step 7: The software must read the interrupt status register to determine whether a RX was successfully completed or a time-out occurred.

Step 8: The software reads the RX receive buffer register (HDQ_RX_DATA) to retrieve the read data from the slave.

Step 9: Repeat the above steps for each successive byte.

20.5.3.3 1-Wire Bit Mode Operation

Select the single-bit mode by setting the 1-WIRE_SINGLE_BIT bit (HDQ_CTRL_STATUS[7]) to 1 in the control and status register. In this mode, only one bit of data is transferred at a time between the master and the slave. After the bit is transferred, the corresponding interrupt flag is set. 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 is updated each time a bit is received, while bit 0 of the TX register contains the bit of data to be sent.

20.5.4 Power Management

The software has independent control of the two clock domains (interconnect clock HDQ_ICLK and functional clock HDQ_FCLK). As there is no acknowledge mechanism from the HDQ/1-Wire module to an idle request, the software ensures that a clock is not shut off while a transfer is processed (the data would be lost).

If the autoidle function (HDQ_SYSCONFIG[0] set to 1) provides transfer security (the module wakes up the HDQ_ICLK as soon as a transfer is initiated), the power-down mode and the PRCM idle requests (throughout the whole L4 clock domain idle request) must be handled carefully.

The following subsections describe the steps to follow when using the power-down and idle modes.

20.5.4.1 Module Power-Down Mode

Step 1: Before shutting off the HDQ_FCLK, wait for a RX or TX-complete interrupt.

- In a read operation, the transfer is completed when the RX-complete flag (HDQ_INT_STATUS[1]) generates an interrupt.
- In a write operation, the transfer is completed when the TX-complete flag (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, or else the data is not written to the slave.

The HDQ_INT_STATUS must be read to clear the interrupt condition.

Step 2: Set the HDQ/1-Wire CLOCKENABLE bit (HDQ_CTRL_STATUS[5]) in the HDQ_CTRL_STATUS register to 0 to disable the clock.

There should be no access to the module registers after the software has put the module in power-down mode, other than a write to the clock enable bit to take it out of power-down mode.

20.5.4.2 System-Idle Mode

This subsection describes the steps to follow at the module level before enabling the idle mode at the system level (for more information concerning the system power management scheme, see Chapter 5, *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 PRCM CM_FCLKEN1_CORE register EN_HDQ bit (CM_FCLKEN1_CORE[23]) is set to 0 and no other modules must receive FUNC_12M_CLK. The software must verify that no transfer is in progress.

In a read operation:

Step 1: Wait for an RX-complete interrupt.

- In a read operation, the transfer is completed when the RX-complete flag (HDQ_INT_STATUS[1]) generates an interrupt.

Step 2: Read the HDQ_INT_STATUS to clear the read-complete interrupt flag.

Step 3: Read the HDQ_RX_DATA to retrieve the read data.

Step 4: The HDQ_ICLK can be shut off by entering the system-idle mode.

In a write operation:

Step 1: Wait for a TX-complete interrupt.

- In a write operation, the transfer is completed when the TX-complete flag (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 or else the data is not written to the slave.

Step 2: Read the HDQ/1-Wire HDQ_INT_STATUS to clear the TX-complete interrupt flag.

Step 3: The HDQ_ICLK can be shut off by entering the system-idle mode.

Two cases appear in HDQ_ICLK:

- ☐ The clock is no longer required and the EN_HDQ bit (CM_ ICLKEN1_CORE[23]) bit in the PRCM CM_ICLKEN1_CORE register is set by software. In this case, the clock is cut, provided no other module needs it. Before setting the EN_HDQ bit, the software must follow the previous steps.
- ☐ The AUTO_HDQ bit (CM_AUTOIDLE_CORE[23]) in the PRCM CM_ AUTOIDLE1_CORE register is set. In this case, the software must ensure that all the HDQ/1-Wire transfers are complete before enabling the L4 interconnect clock domain idle mode. Otherwise, the HDQ/1-Wire module has no way of preventing the clock from being cut (there are no hardware mechanisms for this purpose). The previous steps must be verified before putting the L4 clock domain into idle state.

20.6 HDQ/1-Wire Registers

Table 20−6. Instance Summary

Module Name	Base Address	Size
HDQ/1-Wire	0x480B 2000	4K bytes

20.6.1 HDQ/1-Wire Register Mapping Summary

Table 20−7. HDQ/1-Wire Register Offset Address

Register Name	Type	Register Width (Bits)	Offset
HDQ_REVISION	R	32	0x00
HDQ_TX_DATA	R/W	32	0x04
HDQ_RX_DATA	R	32	0x08
HDQ_CTRL_STATUS	R/W	32	0x0C
HDQ_INT_STATUS	R	32	0x10
HDQ_SYSCONFIG	R/W	32	0x14
HDQ_SYSSTATUS	R	32	0x18

20.6.2 Register Descriptions

Table 20−8. HDQ_REVISION

Address Offset	0x000		
Physical Address	0x480B2000	**Instance**	HDQW1
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																								HDQ_REVISION							

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Reads return 0.	R	0x000000
7:0	HDQ_REVISION	Revision number of the current module The 4 LSB indicate a minor revision. The 4 MSB indicate a major revision. Ex: “0x21”: Revision 2.1	R	

Table 20−9. HDQ_TX_DATA

Address Offset	0x004		
Physical Address	0x480B2004	**Instance**	HDQW1
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 0.	R	0x000000
7:0	TX DATA	Transmit data (used in both HDQ and 1-Wire modes)	R/W	0x00

Table 20−10. HDQ_RX_DATA

Address Offset	0x008		
Physical Address	0x480B2008	**Instance**	HDQW1
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 0, writes ignored	R	0x000000
7:0	RX DATA	Read data (used in both HDQ and 1-Wire modes)	R	0x00

Table 20−11. HDQ_CTRL_STATUS

Address Offset	0x00C		
Physical Address	0x480B200C	**Instance**	HDQW1
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

Table 20−11. HDQ_CTRL_STATUS (Continued)

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Reads return 0	RW	0x000000
7	1_WIRE_ SINGLE_BIT	Single-bit mode for 1-Wire 0x0: Disabled 0x1: Enabled	RW	0
6	INTERRUPT-MASK	Interrupt masking bit 0x0: Disable interrupts 0x1: Enable interrupts	RW	0
5	CLOCKENABLE	Power-down mode bit 0x0: Disable clocks 0x1: Enable clocks	RW	0
4	GO	GO bit Write 1 to send the appropriate commands. Bit returns to 0 after the command is complete.	RW	0
3	PRESENCE DETECT	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 whether next command is read or write. 0x0: Read 0x1: Write	RW	0
0	MODE	Mode selection bit 0x0: HDQ mode 0x1: 1-Wire mode	RW	0

Table 20−12. HDQ_INT_STATUS

Address Offset	0x010		
Physical address	0x480B2010	**Instance**	HDQW1
Description	This register controls interrupts status.		
Type	R		

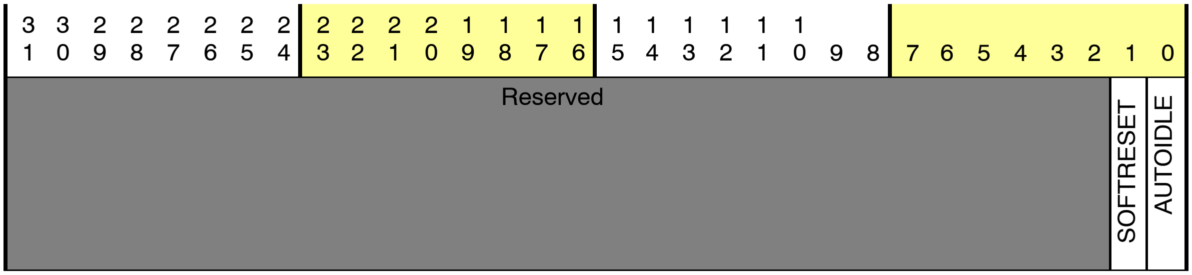
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													TXCOMPLETE	RXCOMPLETE	TIMEOUT

Table 20−12. HDQ_INT_STATUS (Continued)

Bits	Field Name	Description	Type	Reset
31:3	Reserved	Reads return 0	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/time-out interrupt flag In 1-Wire .mode, set to 1 if slave presence detected. In HDQ mode, set to 1 if time-out on read occurs. Set to 0 when register read.	R	0

Table 20−13. HDQ_SYSCONFIG

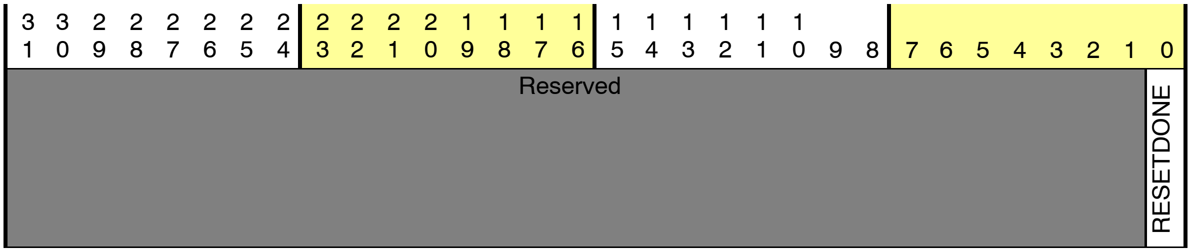
Address Offset	0x014		
Physical Address	0x480B2014	**Instance**	HDQW1
Description	This register controls various bits.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:2	Reserved	Reads return 0.	R/W	0
1	SOFTRESET	Start soft reset sequence 0x0: Disabled 0x1: Enabled	R/W	0
0	AUTOIDLE	Interconnect idle 0x0: Module clock is free-running. 0x1: Module is in power-saving mode.	R/W	0

Table 20−14. HDQ_SYSSTATUS

Address Offset	0x018		
Physical Address	0x480B2018	**Instance**	HDQW1
Description	This register monitors the reset sequence.		
Type	R		



Bits	Field Name	Description	Type	Reset
31:1	Reserved	Reads return 0.	R	0x00000000
0	RESETDONE	Reset monitoring 0x0: The module is currently performing its reset. 0x1: The module has finished its reset.	R	1

Chapter 21

Multichannel Buffered Serial Port

This chapter discusses the multichannel buffered serial port (McBSP) on the OMAP2420 device.

Topic **Page**

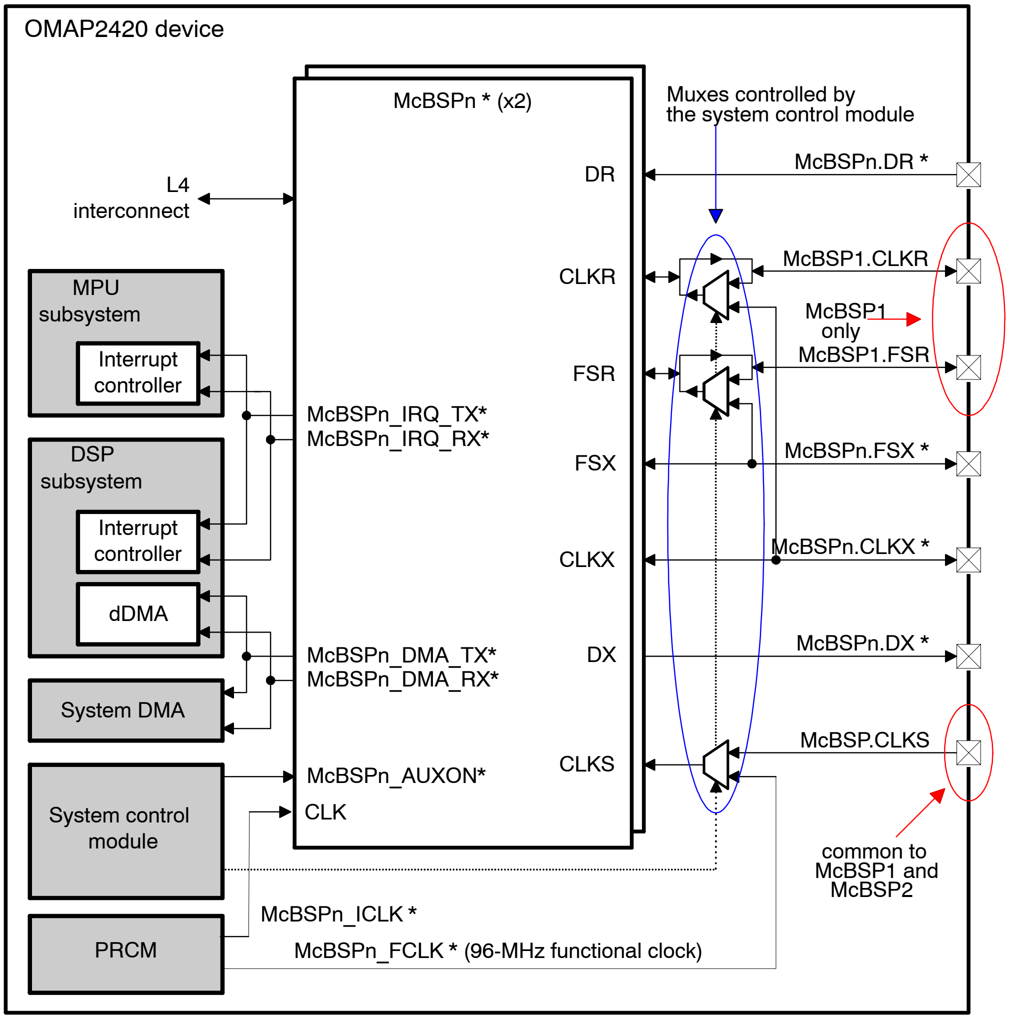
21.1 McBSP Overview

The multichannel buffered serial port (McBSP) provides a full-duplex, direct serial interface between the OMAP2420 device and other devices in systems such as application chips and codecs. It can accommodate a wide range of peripherals and protocols because of its high level of versatility.

The OMAP2420 device provides two identical instantiations of the McBSP module, called McBSP1 and McBSP2.

Figure 21−1 illustrates the McBSP modules in the OMAP2420 device.

Figure 21−1. McBSP Highlight



* n = 1 for McBSP1; n = 2 for McBSP2

The main features of the McBSP modules are:

❑ Full-duplex communication

❑ Double-buffered transmission and triple-buffered reception that allows a continuous data stream

❑ Independent clocking and framing for reception and transmission

- Capability to send interrupts to the CPU and to send DMA events to the DMA controller
- 128 channels for transmission and for reception
- Multichannel selection modes that allow enabling or blocking transfers in each of the channels
- Direct interface to industry-standard codecs, analog interface chips (AICs), and other serially connected A/D and D/A devices (SPI devices and IIS compliant devices, for example)
- Support for external generation of clock signals and frame-synchronization (frame-sync) signals
- A programmable sample rate generator (SRG) for internal generation and control of clock signals and frame-sync signals (FSGs)
- Programmable polarity for frame-sync pulses and clock signals
- A wide selection of data sizes: 8, 12, 16, 20, 24, and 32 bits
- μ-law and A-law companding
- The option of transmitting/receiving 8-bit data with the least-significan bit (LSB) first
- Status bits for flagging exception/error conditions

21.2 McBSP Environment

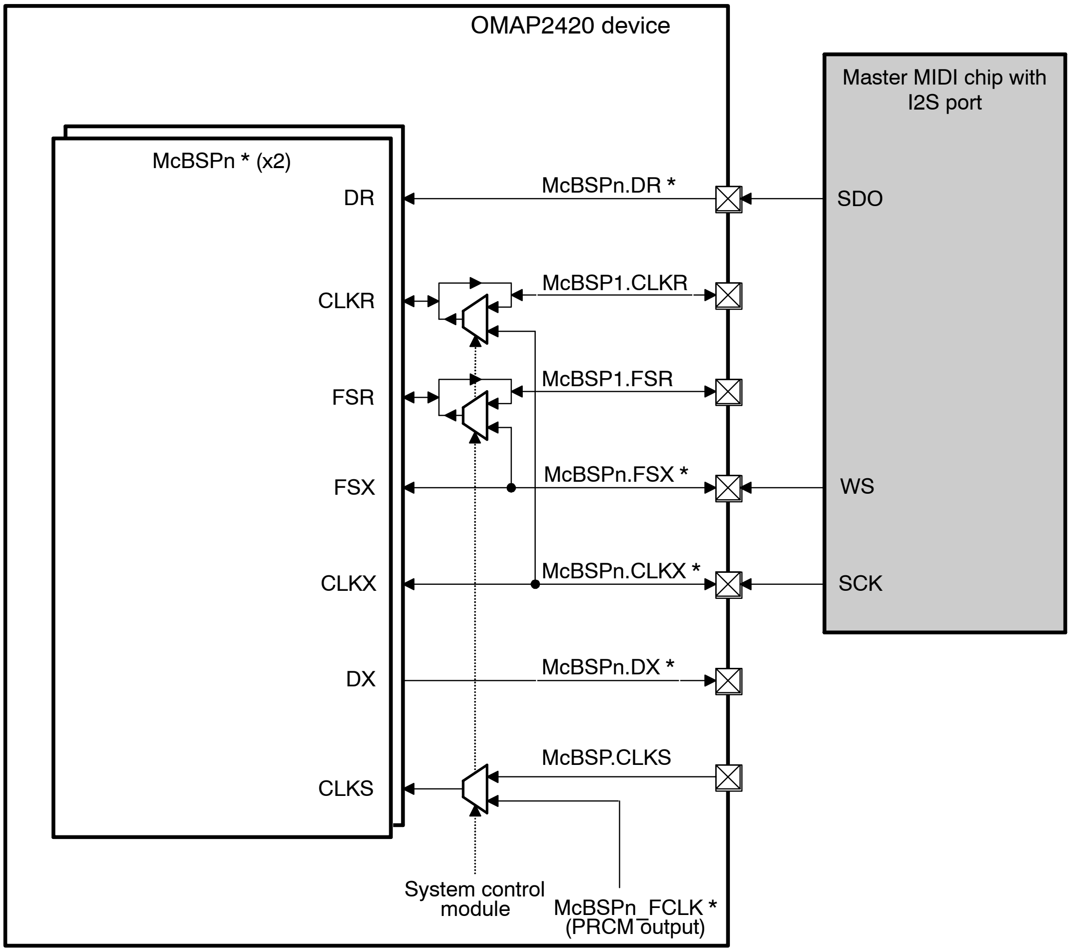
21.2.1 McBSP Functional Interface

21.2.1.1 Overview

The McBSP consists of a data-flow path and a control path connected to external devices. The internal loopback capabilities between CLKX/CLKR and FSX/FSR allow the module to interface with I2S bidirectional systems. Note that the related multiplexers are controlled through OMAP2420 system control global registers (see Section 21.2.1.2, *Functional Interface Control*, for more information).

Figure 21−2 shows a typical application using the McBSP interface connected to a MIDI chip through its I2S port.

Figure 21−2. McBSP Typical Application System Overview



* n = 1 for McBSP1; n = 2 for McBSP2

Table 21−1 describes the McBSP module interface.

Table 21−1. McBSP I/O Description

Signal Name	**I/O**	**Description**	**Reset Value**
McBSP.CLKS	I	External clock (common to McBSP1 and McBSP2)	N/A
McBSP1			
McBSP1.DR	I	Received serial data	N/A
McBSP1.CLKR	IO	Receive clock	0
McBSP1.FSR	IO	Receive frame-sync	0
McBSP1.DX	O	Transmitted serial data	0
McBSP1.CLKX	IO	Transmit clock	0
McBSP1.FSX	IO	Transmit frame-sync	0
McBSP2			
McBSP2.DR	I	Received serial data	N/A
McBSP2.DX	O	Transmitted serial data	0
McBSP2.CLKX	IO	Clock	0
McBSP2.FSX	IO	Frame-sync	0

Data transmitted to devices interfaced with the McBSPs is emitted through the McBSPn.DX data transmit pins, while data from those devices is received on the McBSPn.DR data receive pins.

Control information (that is, clocking and frame-sync) is communicated through the following pins: McBSPn.CLKX (transmit clock), McBSPn.CLKR (receive clock), McBSPn.FSX (FSX), and McBSPn.FSR (FSR).

The OMAP2420 CPU and DMA controller communicate with the McBSP through 16-bit-wide registers accessible through the L4 interface.

The CPU or the DMA controller writes the data to be transmitted into the McBSP data transmit registers (MCBSP_DXR1_REG, MCBSP_DXR2_REG). Data is then shifted out to McBSPn.DX pins through transmit shift registers.

Data received on the McBSPn.DR pins is shifted into receive shift registers and first copied into receive buffers. It is then copied to the McBSP data receive registers (MCBSP_DRR1_REG and MCBSP_DRR2_REG) and read by the CPU or DMA controller.

Note that MCBSP_DXR2_REG and MCBSP_DRR2_REG are only used if the serial word length is greater than 16 bits to handle the most-significant bits (MSBs).

21.2.1.2 Functional Interface Control

The OMAP2420 system control module (see Chapter 8, *System Control Module*) allows control of various parameters of the McBSP1 and McBSP2 interfaces.

McBSP1 Interface Control

- Bit MCBSP1_FSR (CONTROL_DEVCONF[4]) of system control module register CONTROL_DEVCONF is used to select the McBSP1 module FSR signal source.
  - When set to 1, the FSR source comes from the McBSP1.FSX pin.
  - When set to 0, the FSR source comes from the McBSP1.FSR pin.
- Bit MCBSP1_CLKR (CONTROL_DEVCONF[3]) of system control module register CONTROL_DEVCONF is used to select the McBSP1 module CLKR signal source.
  - When set to 1, the CLKR source comes from the McBSP1.CLKX pin.
  - When set to 0, the CLKR source comes from the McBSP1.CLKR pin.
- Bit MCBSP1_CLKS (CONTROL_DEVCONF[2]) of system control module register CONTROL_DEVCONF is used to select the McBSP1 module CLKS signal source.
  - When set to 1, the CLKS source comes from the McBSP.CLKS pin.
  - When set to 0, the CLKS source comes from the internal functional clock provided by the PRCM (McBSP1_FCLK).

McBSP2 Interface Control

- Bit MCBSP2_CLKS (CONTROL_DEVCONF[6]) of system control module register CONTROL_DEVCONF is used to select the McBSP2 module CLKS signal source.
  - When set to 1, CLKS source comes from McBSP.CLKS pin.
  - When set to 0, CLKS source comes from the internal functional clock provided by the PRCM (McBSP2_FCLK).

CAUTION

There are no McBSP2.CLKR and McBSP2.FSR external pins. The McBSP2 module CLKR and FSR signal sources are the McBSP2.CLKX and McBSP2.FSX pins, respectively. Consequently, there is a light restriction on the McBSP2 module when used in full-duplex mode: Reception and transmission use the same clock signal and the same FSG.

CAUTION

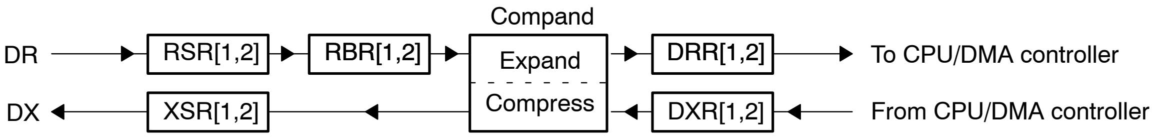
The CONTROL_DEVCONF[6] and CONTROL_DEVCONF[2] bits select between the internal functional clock McBSPn_FCLK coming from the PRCM and the McBSP.CLKS pin. The signal coming out of this multiplexer is provided to the McBSP and is called CLKS. Inside the McBSP, use the CLKSM bit to choose between this signal and the internal interface clock McBSPn_ICLK available on the CLK signal for feeding the SRG.

21.2.2 McBSP Protocols

21.2.2.1 McBSP Data Transfer Process

Figure 21−3 shows the McBSP data transfer paths. McBSP receive operations are triple-buffered and transmit operations are double-buffered. Register use depends on the serial word length with regards to 16-bit words. A single register can handle 16-bit words or smaller; words greater than 16 bits require additional registers.

Figure 21−3. McBSP Data Transfer Paths



Data Transfer Process for 8/12/16-Bit Long Words

If the word length is 16 bits or smaller, only one 16-bit register is required at each stage of the data transfer paths.

Receive data arrives at the McBSPn.DR pin and is shifted into receive shift register 1. When a full word is received, the content of the shift register is copied into receive buffer register 1 if it is not full with previous data. Receive buffer register 1 is then copied to data receive register 1 (MCBSP_DRR1_REG) if the content of MCBSP_DRR1_REG has previously been read by the CPU or the DMA controller.

If the companding feature of the McBSP is implemented (see the following sections for more details), the required word length is 8 bits and receive data is expanded into the appropriate format before being passed from receive buffer 1 to the MCBSP_DRR1_REG register.

Transmit data is written by the CPU or the DMA controller to data transmit register 1 (MCBSP_DXR1_REG). If there is no previous data in transmit shift register 1, the value in MCBSP_DXR1_REG is copied to transmit shift register1; otherwise, the content of MCBSP_DXR1_REG is copied to transmit shift register 1 when the last bit of the previous data is shifted out on the McBSPn.DX pin.

If selected, the companding module compresses 16-bit data into the appropriate 8-bit format before writing it to transmit shift register 1. After FSX, the transmitter begins shifting bits from transmit shift register 1 to the McBSPn.DX pin.

Data Transfer Process for 20/24/32-Bit Long Words

If the word length is larger than 16 bits, two 16-bit registers are required at each stage of the data transfer paths.

Receive data arrives on the McBSPn.DR pin and is shifted into receive shift register 2 first (MSBs) and then into receive shift register 1 (LSBs). Once the full word is received, the contents of receive shift registers 2 and 1 are copied to receive buffers 2 and 1, respectively, if receive buffer 1 is not full. Then, if the content of MCBSP_DRR1_REG has been previously read by the CPU or the DMA controller, the contents of receive buffers 2 and 1 are copied to MCBSP_DRR2_REG and MCBSP_DRR1_REG, respectively. The CPU or the DMA controller must read data from MCBSP_DRR2_REG first and then from MCBSP_DRR1_REG. When MCBSP_DRR1_REG is read, the next receive buffer to data receive register copy occurs. For more information about reception, see Section 21.4, *McBSP Functional Description*.

For transmission, the CPU or the DMA controller must write data to MCBSP_DXR2_REG first and then to MCBSP_DXR1_REG. If there is no previous data in transmit shift register 1 when new data arrives in MCBSP_DXR1_REG, the contents of MCBSP_DXR2_REG and MCBSP_DXR1_REG are copied to transmit shift register 2 and 1, respectively; otherwise, the contents of the data transmit registers are copied to the transmit shift registers when the last bit of the previous data is shifted out on the McBSPn.DX pin. After FSX, the transmitter begins shifting bits from the transmit shift registers to the McBSPn.DX pin.

21.2.2.2 Companding (Compressing and Expanding) Data

Companding hardware allows data compression and expansion in either μ-law or A-law format. The companding standard used in the United States and Japan is μ-law, whereas the companding standard in Europe is referred to as A-law.

The specifications for μ-law and A-law log PCM are part of the CCITT G.711 recommendation.

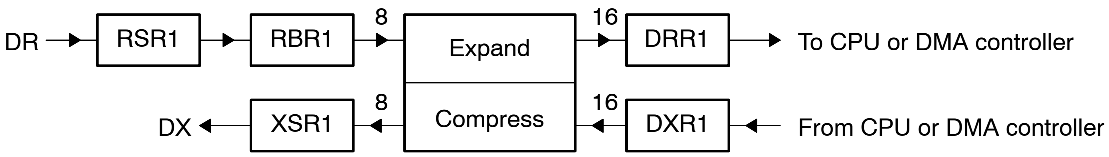
A-law and μ-law allow 13 and 14 bits of dynamic range, respectively. Values outside this range are set to the most positive or most negative value. Thus, for companding to work best, the data transferred to and from the McBSP through the CPU or DMA controller must be at least 16 bits wide. The μ-law and A-law formats encode data into 8-bit code words.

As companded data is always 8 bits wide, the serial data stream length must be set to 8 bits when companding is enabled. If a word length in a frame is not 8 bits, the companding process continues as if the word length were 8 bits.

Figure 21−4 shows the companding processes. In the transmit operation, compression occurs during the process of copying data from the MCBSP_DXR1_REG register to transmit shift register 1. The transmit data is encoded according to the specified companding law (A-law or μ-law).

In the receive operation, expansion occurs during the process of copying data from receive buffer 1 to the MCBSP_DRR1_REG register. The receive data is decoded to twos-complement format.

Figure 21−4. Companding Process



Companding Formats

For reception, the 8-bit compressed data in the receive buffer 1 is expanded to left-justified 16-bit data in MCBSP_DRR1_REG register.

For transmission using μ-law compression, ensure that the 14 data bits are left-justified in the MCBSP_DXR1_REG register, with the remaining two low-order bits filled with 0s as shown in Figure 21−5.

Figure 21−5. μ-Law Transmit Data Companding Format

	15−2	1−0
μ-law format in DXR1	Value	00

For transmission using A-law compression, ensure that the 13 data bits are left-justified in the MCBSP_DXR1_REG register, with the remaining three low-order bits filled with 0s as shown in Figure 21−6.

Figure 21−6. A-Law Transmit Data Companding Format

	15−3	2−0
A-law format in DXR1	Value	000

Capability to Compand Internal Data

If the McBSP is otherwise unused (the serial port transmit and receive sections are reset), the companding hardware can compand internal data. This can be used to:

- Convert linear to the appropriate μ-law or A-law format
- Convert μ-law or A-law to the linear format
- Observe the quantization effects in companding by transmitting linear data, and compressing and reexpanding this data. This is useful only if the XCOMPAND (MCBSP_XCR2_REG[4:3]) bit field and the RCOMPAND bit field (MCBSP_RCR2_REG[4:3]) enable the same companding format.

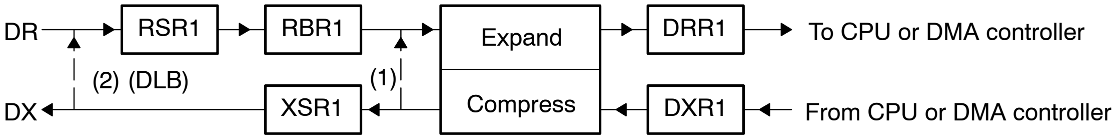
Figure 21−7 shows two methods by which the McBSP can compand internal data. The data paths for these two methods are used to do the following.

- When the transmit and receive sections of the serial port are reset, MCBSP_DRR1_REG and MCBSP_DXR1_REG are connected internally through the companding logic. Values from MCBSP_DXR1_REG are compressed, as selected by the XCOMPAND bit field, and then expanded, as selected by the RCOMPAND bit field. Note that the RRDY and XRDY bits (MCBSP_SPCR1_REG[1] and MCBSP_SPCR2_REG[1], respectively) are not set. However, data is available in MCBSP_DRR1_REG within four functional clock cycles after being written to

MCBSP_DXR1_REG. The advantage of this method is its speed. The disadvantage is that there is no synchronization available to the CPU and DMA to control the flow. Note that MCBSP_DRR1_REG and MCBSP_DXR1_REG are internally connected if the (X/R)COMPAND bits are set to 10b or 11b (compand using µ-law or A-law).

- The McBSP is enabled in digital loopback mode with companding appropriately enabled by the RCOMPAND and XCOMPAND bit fields. Receive and transmit interrupts (RINT when RINTM bit field (MCBSP_SPCR1_REG[5:4]) = 0, XINT when the XINTM bit field (MCBSP_SPCR2_REG[5:4]) = 0) or synchronization events (REVT and XEVT) allow synchronization of the CPU or DMA to these conversions, respectively. Here, the time for this companding depends on the serial bit rate selected.

Figure 21−7. Two Methods of Companding Internal Data by the McBSP



Reversing Bit Order: Option to Transfer LSB First

Generally, the McBSP transmits or receives all data with the MSB first. However, certain 8-bit data protocols (that do not use companded data) require the LSB to be transferred first.

Setting XCOMPAND = 01b in MCBSP_XCR2_REG[4:3] reverses the bit order of 8-bit words (LSB first) before being sent from the serial port. Setting RCOMPAND = 01b in MCBSP_RCR2_REG[4:3] reverses the bit order of 8-bit words during reception.

Similar to companding, this feature is enabled only if the appropriate word length bits are set to 0, indicating that 8-bit words are to be transferred serially. If either phase of the frame does not have an 8-bit word length, the McBSP assumes the word length is eight bits, and LSB-first ordering is done.

21.2.2.3 Clocking and Framing Data

This section explains basic concepts and terminology important for understanding how McBSP data transfers are timed and delimited.

Clocking

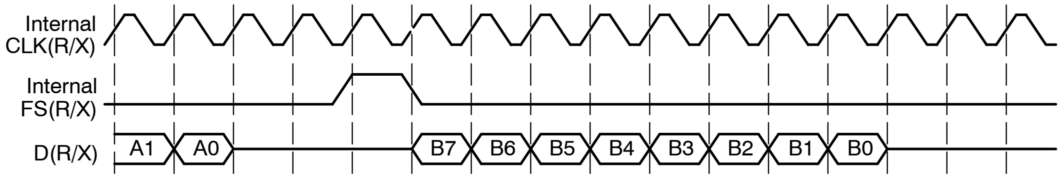
Data is shifted one bit at a time from the McBSPn.DR pin to the receive shift registers, or from the transmit shift registers to the McBSPn.DX pin. The time for each bit transfer is controlled by the rising or falling edge of a clock signal.

The receive clock signal (CLKR) controls bit transfers from the McBSPn.DR pin to the receive shift registers. The transmit clock signal (CLKX) controls bit transfers from the transmit shift registers to the McBSPn.DX pin. CLKR or CLKX can be derived from a pin at the boundary of the McBSP

(McBSPn.CLKR and McBSPn.CLKX respectively) or derived from within the McBSP. The polarities of CLKR and CLKX are programmable.

Figure 21−8 gives an example in which the clock signal controls the timing of each bit transfer on the pin.

Figure 21−8. Clock Signal Control of Bit Transfer Timing



Note:

The McBSP cannot operate at a frequency higher than 1/2 the CPU clock frequency. When driving CLKX or CLKR at the pin, choose an appropriate input clock frequency. When using the internal SRG for CLKX and/or CLKR, choose an appropriate input clock frequency and divide down value (CLKDV bit field (MCBSP_SRGR1_REG[7:0]).

Serial Words

Bits traveling between a shift register (RSR or XSR) and a data pin (McBSPn.DR or McBSPn.DX) are transferred as a group called a serial word. You can define how many bits are in a word.

Bits coming in on the McBSPn.DR pin are held in RSR until it holds a full serial word. Only then is the word passed to RBR (and ultimately to the MCBSP_DRR[1,2]_REG registers).

During transmission, XSR does not accept new data from the MCBSP_DXR[1,2]_REG registers until a full serial word has been passed from XSR to the McBSPn.DX pin.

In the example in Figure 8, an 8-bit word size is defined (see bits 7 through 0 of word B being transferred).

Frames and Frame Sync

One or more words are transferred as a group called a frame. The number of words in a frame can be defined.

All the words in a frame are sent in a continuous stream. However, there can be pauses between frame transfers. The McBSP uses FSR signals to determine when each frame is received/transmitted. When a pulse occurs on a FSR signal, the McBSP begins receiving/transmitting a frame of data. When the next pulse occurs, the McBSP receives/transmits the next frame, and so on.

Pulses on the FSR signal initiate frame transfers on McBSPn.DR. Pulses on the transmit frame-synchronization (FSX) signal initiate frame transfers on

McBSPn.DX. FSR or FSX can be derived from a pin at the boundary of the McBSP (McBSPn.FSR and McBSPn.FSX, respectively) or derived from inside the McBSP, as determined by FSXM and FSRM bits (MCBSP_PCR_REG[11] and MCBSP_PCR_REG[10], respectively).

In the example in Figure 21−8, a one-word frame is transferred when a frame-sync pulse occurs.

In McBSP operation, the inactive-to-active transition of the FSG indicates the start of the next frame. For this reason, the FSG may be high for an arbitrary number of clock cycles. The next frame-sync occurs only after the signal is recognized to have gone inactive, and then active again.

Detecting Frame-Sync Pulses, Even in Reset State

The McBSP can send receive and transmit interrupts to the CPU to indicate specific events in the McBSP. To facilitate detection of frame-sync, these interrupts can be sent in response to frame-sync pulses (see Section 21.5 for 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 RINT when the receiver is in reset). In this case, FSRM/FSXM bits (MCBSP_PCR_REG[10]/MCBSP_PCR_REG[11], respectively) and FSRP/FSXP bits (MCBSP_PCR_REG[2]/MCBSP_PCR_REG[3], respectively) still select the appropriate source and polarity of frame-sync. Thus, even when the serial port is in the reset state, these signals are synchronized to the CPU clock and then sent to the CPU in the form of RINT and XINT at the point at which they feed the receiver and transmitter of the serial port. Consequently, a new frame-sync pulse can be detected, and after this occurs the CPU can take the serial port out of reset safely.

Ignoring Frame-Sync Pulses

The McBSP can be configured to ignore transmit and/or FSR pulses. To have the receiver or transmitter recognize frame-sync pulses, clear the appropriate frame-sync ignore bit (RFIG = 0 for the receiver, XFIG = 0 for the transmitter). To have the receiver or transmitter ignore frame-sync pulses until the desired frame length or number of words is reached, set the appropriate frame-sync ignore bit (RFIG = 1 for the receiver, XFIG = 1 for the transmitter). For more details on unexpected frame-sync pulses, see one of the following topics:

- *Unexpected FSR Pulse* (Section 21.4.2.3)
- *Unexpected FSX Pulse* (Section 21.4.2.6)

The frame-sync ignore function can be used for data packing (for more information, see Section 21.5.6, *Data Packing Examples*).

Frame Frequency

The frame frequency is determined by the period between frame-sync pulses and is defined as shown in the following equation:

Frame frequency = Clock frequency / (Number of clock cycles between frame-sync pulses)

The frame frequency can be increased by decreasing the time between frame-sync 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.

Maximum Frame Frequency

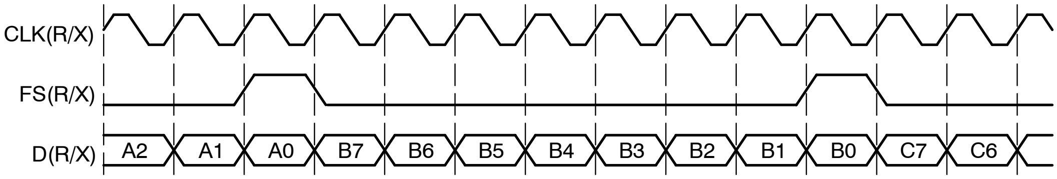
The minimum number of clock cycles between frame-sync pulses is equal to the number of bits transferred per frame. The maximum frame frequency is defined as shown in the following equation:

Equation 21−1. McBSP Maximum Frame Frequency

$$\text{Maximum Frame Frequency} = \frac{\text{Clock Frequency}}{\text{Number of Bits Per Frame}}$$

Figure 21−9 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 21−9. McBSP Operating at Maximum Packet Frequency



If there is a 1-bit data delay as shown in this figure, the frame-sync pulse overlaps the last bit transmitted in the previous frame. Effectively, this permits a continuous stream of data, making frame-sync pulses redundant. Theoretically, only an initial frame-sync pulse is required to initiate a multi-packet transfer.

The McBSP supports operation of the serial port in this fashion by ignoring the successive frame-sync pulses. Data is clocked into the receiver or clocked out of the transmitter during every clock cycle.

Note:

For MCBSP_XCR_REG[1:0] register XDATDLY bit field = 0x0 (0-bit data delay), the first bit of data is transmitted asynchronously to the internal transmit clock signal (CLKX). For more information, see Section 21.5.4.12, *Set the Transmit Data Delay*.

21.2.2.4 Frame Phases

The McBSP allows you to configure each frame to contain one or two phases. The number of words per frame and 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, you might define a frame as consisting of one phase containing two words of 16 bits each, followed by a second phase consisting of 10 words of 8 bits each. This configuration lets you compose frames for custom applications or, in general, maximize the efficiency of data transfers.

Number of Phases, Words, and Bits per Frame

Table 21−2 shows which bit fields in the receive control registers (MCBSP_RCR1_REG and MCBSP_RCR2_REG) and transmit control registers (MCBSP_XCR1_REG and MCBSP_XCR2_REG) determine the number of phases per frame, the number of words per frame, and number of bits per word for each phase, for the receiver and transmitter.

The maximum number of words per frame is 128 for a single-phase frame and 256 for a dual-phase frame. The number of bits per word can be 8, 12, 16, 20, 24, or 32 bits.

Table 21−2. Phases, Words, and Bits per Frame Control Bits

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 **and** RFRLEN2	RWDLEN1 for phase 1 RWDLEN2 for phase 2
Transmission	1 (XPHASE = 0)	XFRLEN1	XWDLEN1
Transmission	2 (XPHASE = 1)	XFRLEN1 **and** XFRLEN2	XWDLEN1 for phase 1 XWDLEN2 for phase 2

Note: RPHASE = MCBSP_RCR2_REG[15]
XPHASE = MCBSP_XCR2_REG[15]
RFRLEN1 = MCBSP_RCR1_REG[14:8]
RFRLEN2 = MCBSP_RCR2_REG[14:8]
XFRLEN1 = MCBSP_XCR1_REG[14:8]
XFRLEN2 = MCBSP_XCR2_REG[14:8]
RWDLEN1 = MCBSP_RCR1_REG[7:5]
RWDLEN2 = MCBSP_RCR2_REG[7:5]
XWDLEN1 = MCBSP_XCR1_REG[7:5]
XWDLEN2 = MCBSP_XCR2_REG[7:5]

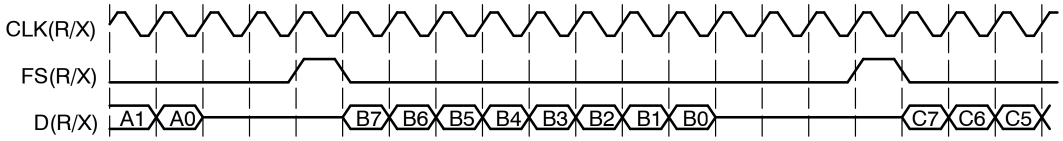
Single-Phase Frame Example

Figure 21−10 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 McBSPn.DX and McBSPn.DR pins are available one clock cycle after FS(R/X) goes active. The figure makes the following assumptions:

- ☐ RPHASE (MCBSP_RCR2_REG[15]) and XPHASE (MCBSP_XCR2_REG[15]) = 0: Single-phase frame
- ☐ RFRLEN1 (MCBSP_RCR1_REG[14:8]) and XFRLEN1 (MCBSP_XCR1_REG[14:8]) = 0x0: 1 word per frame
- ☐ RWDLEN1 (MCBSP_RCR1_REG[7:5]) and XWDLEN1 (MCBSP_XCR1_REG[7:5]) = 0x0: 8-bit word length
- ☐ RFRLEN2 (MCBSP_RCR2_REG[14:8]) and XWDLEN2 (MCBSP_XCR2_REG[14:8]) are ignored

- CLKRP (MCBSP_PCR_REG[0]) and CLKXP (MCBSP_PCR_REG[1]) = 0: Receive data clocked on falling edge; transmit data clocked on rising edge
- FSRP (MCBSP_PCR_REG[2]) and FSXP (MCBSP_PCR_REG[3]) = 0: Active-high FSGs
- RDATDLY (MCBSP_RCR2_REG[1:0]) and XDARDLY (MCBSP_XCR2_REG[1:0]) = 01b: 1-bit data delay

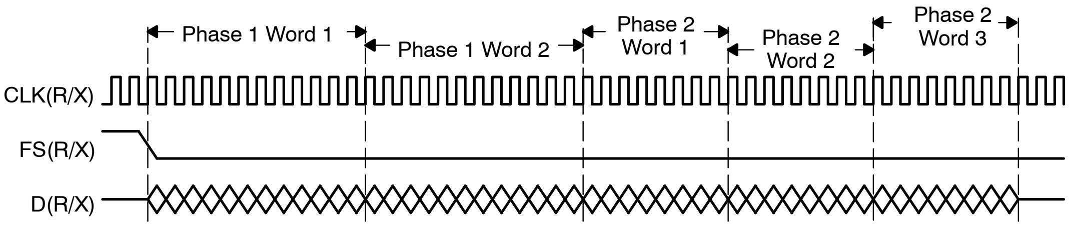
Figure 21−10. Single-Phase Frame for a McBSP Data Transfer



Dual-Phase Frame Example

Figure 21−11 shows an example of a frame where the first phase consists of two words of 12 bits each, followed by a second phase of three words of 8 bits each. The entire bit stream in the frame is contiguous. There are no gaps either between words or between phases.

Figure 21−11. Dual-Phase Frame for a McBSP Data Transfer



21.2.2.5 McBSP Reception

This section explains the fundamental process of reception in the McBSP. For details about how to program the McBSP receiver, see Section 21.5.3, *Receiver Configuration*.

Figure 21−12 and Figure 21−13 show how reception occurs in the McBSP. Figure 21−12 shows the physical path for the data. Figure 21−13 is a timing diagram that shows signal activity for one possible reception scenario. A description of the process follows the figures.

Figure 21−12. McBSP Reception Physical Data Path

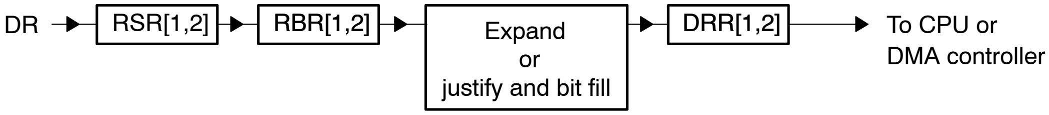
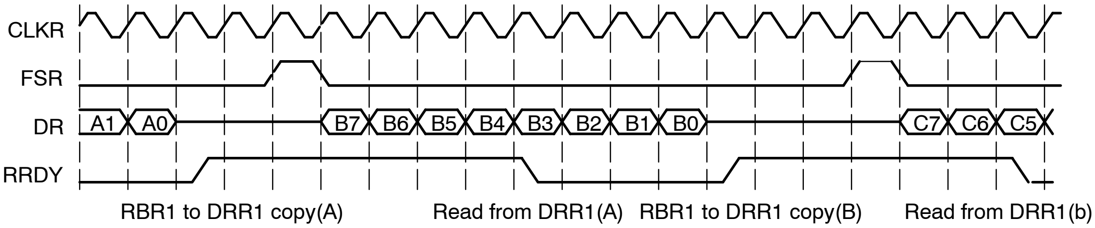


Figure 21−13. McBSP Reception Signal Activity



The following process describes how data travels from the McBSPn.DR pin to the CPU or to the DMA controller:

Step 1: The McBSP waits for a FSR pulse on internal FSR.

Step 2: When the pulse arrives, the McBSP inserts the appropriate data delay that is selected with the RDATDLY bits of the MCBSP_RCR2_REG[1:0] register. In the preceding timing diagram (Figure 21−13), a 1-bit data delay is selected.

Step 3: The McBSP accepts data bits on the McBSPn.DR pin and shifts them into the receive shift register(s). If the word length is 16 bits or smaller, only RSR1 is used. If the word length is larger than 16 bits, RSR2 and RSR1 are used and RSR2 contains the MSBs. For details on choosing a word length, see Section 21.5.3.8, *Set the Receive Word Length(s)*.

Step 4: When a full word is received, the McBSP copies the contents of the receive shift register(s) to the receive buffer register(s), provided that RBR1 is not full with previous data. If the word length is 16 bits or smaller, only RBR1 is used. If the word length is larger than 16 bits, RBR2 and RBR1 are used, and RBR2 contains the MSBs.

Step 5: The McBSP copies the contents of the receive buffer register(s) into the data receive register(s), provided that MCBSP_DRR1_REG is not full with previous data. When MCBSP_DRR1_REG receives new data, the receiver ready bit (RRDY) is set in MCBSP_SPCR1_REG. This indicates that receive data is ready to be read by the CPU or the DMA controller.

If the word length is 16 bits or smaller, only MCBSP_DRR1_REG is used. If the word length is larger than 16 bits, MCBSP_DRR2_REG and MCBSP_DRR1_REG are used, and MCBSP_DRR2_REG contains the MSBs.

If companding is used during the copy (RCOMPAND = 10b or 11b in MCBSP_RCR2_REG[4:3] register), the 8-bit compressed data in RBR1 is expanded to a left-justified 16-bit value in

MCBSP_DRR1_REG. If companding is disabled, the data copied from RBR[1,2] to MCBSP_DRR[1,2]_REG is justified and bit filled according to the RJUST bits (MCBSP_S{CR1_REG[14:13]).

Step 6: The CPU or the DMA controller reads the data from the data receive register(s). When MCBSP_DRR1_REG is read, RRDY bit (MCBSP_SPCR1_REG[1]) is cleared and the next RBR-to-DRR copy is initiated.

Note:

If both MCBSP_DRR1_REG and MCBSP_DDR2_REG are required (word length larger than 16 bits), the CPU or the DMA controller must read from MCBSP_DRR2_REG first and then from MCBSP_DRR1_REG. As soon as MCBSP_DRR1_REG is read, the next RBR−to−DRR copy is initiated. If MCBSP_DRR2_REG is not read first, the data in MCBSP_DRR2_REG is lost.

When activity is not properly timed, errors can occur. See the following for details:

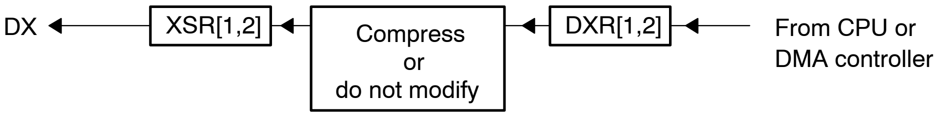
- *Overrun in the Receive*r (Section 21.4.2.2)
- *Unexpected FSR Pulse* (Section 21.4.2.3)

21.2.2.6 McBSP Transmission

This section explains the fundamental process of transmission in the McBSP. For details about how to program the McBSP transmitter, see Section 21.5.4, *Transmitter Configuration*.

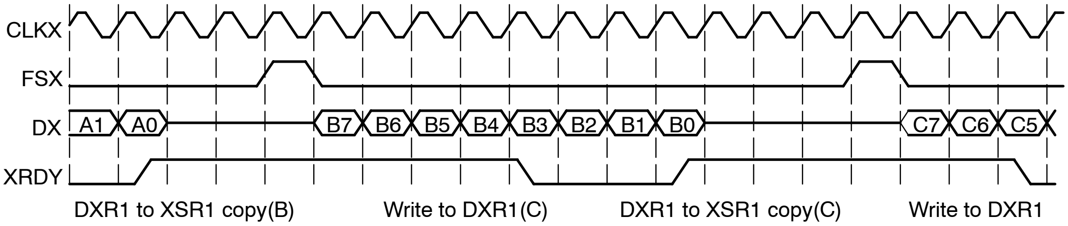
Figure 21−14 and Figure 21−15 show how transmission occurs in the McBSP. Figure 21−14 shows the physical path for the data. Figure 21−15 shows signal activity for one possible transmission scenario. A description of the process follows the figures.

Figure 21−14. McBSP Transmission Physical Data Path



XSR[1,2]: Transmit shift registers 1 and 2 DXR[1,2]: Data transmit registers 1 and 2

Figure 21−15. McBSP Transmission Signal Activity



CLKX: Internal transmit clock
FSX: Internal FSX
DX: Data on DX pin
XRDY: Status of transmitter ready bit (high is 1)

Step 1: The CPU or the DMA controller writes data to the data transmit register(s). When MCBSP_DXR1_REG is loaded, the transmitter ready bit (XRDY) is cleared in MCBSP_SPCR2_REG[1] register to indicate that the transmitter is not ready for new data.

If the word length is 16 bits or smaller, only MCBSP_DXR1_REG is used. If the word length is larger than 16 bits, MCBSP_DXR2_REG and MCBSP_DXR1_REG are used and MCBSP_DXR2_REG contains the MSBs. For details on choosing a word length, see Section 21.5.4.8, *Set the Transmit Word Length(s)*.

Note:

If both MCBSP_DXRn_REG registers are required (word length larger than 16 bits), the CPU or the DMA controller must load MCBSP_DXR2_REG first and then MCBSP_DXR1_REG. As soon as MCBSP_DXR1_REG is loaded, the contents of both MCBSP_DXRn_REG are copied to the transmit shift registers (XSRs), as described in the next step. If MCBSP_DXR2_REG is not loaded first, the previous content of MCBSP_DXR2_REG is passed to the XSR2.

Step 2: When new data arrives in MCBSP_DXR1_REG register, the McBSP copies the content of the data transmit register(s) to the transmit shift register(s). In addition, the transmit ready bit (XRDY) is set. This indicates that the transmitter is ready to accept new data from the CPU or the DMA controller.

If the word length is 16 bits or smaller, only XSR1 is used. If the word length is larger than 16 bits, XSR2 and XSR1 are used and XSR2 contains the MSBs.

If companding is used during the transfer (XCOMPAND = 10b or 11b in MCBSP_XCR2_REG[4:3]), the McBSP compresses the 16-bit data in MCBSP_DXR1_REG to 8-bit data in the μ-law or A-law format in XSR1. If companding is disabled, the McBSP passes data from the MCBSP_DXRn_REG registers to the XSR(s) without modification.

Step 3: The McBSP waits for a FSX pulse on internal FSX.

Step 4: When the pulse arrives, the McBSP inserts the appropriate data delay that is selected with the MCBSP_XCR2_REG[1:0] register XDATDLY bits.

In the preceding timing diagram (Figure 21−15), a 1-bit data delay is selected.

Step 5: The McBSP shifts data bits from the transmit shift register(s) to the McBSPn.DX pin.

When activity is not properly timed, errors can occur. See the following topics for more details:

☐ *Overwrite in the Transmitter* (Section 21.4.2.4)

☐ *Underflow in the Transmitter* (Section 21.4.2.5)

☐ *Unexpected FSX Pulse* (Section 21.4.2.6)

21.3 McBSP Integration

Figure 21−16 and Figure 21−17 show the integration of the McBSP1 and McBSP2 modules, respectively, in the OMAP2420 device.

Figure 21−16. McBSP1 Integration

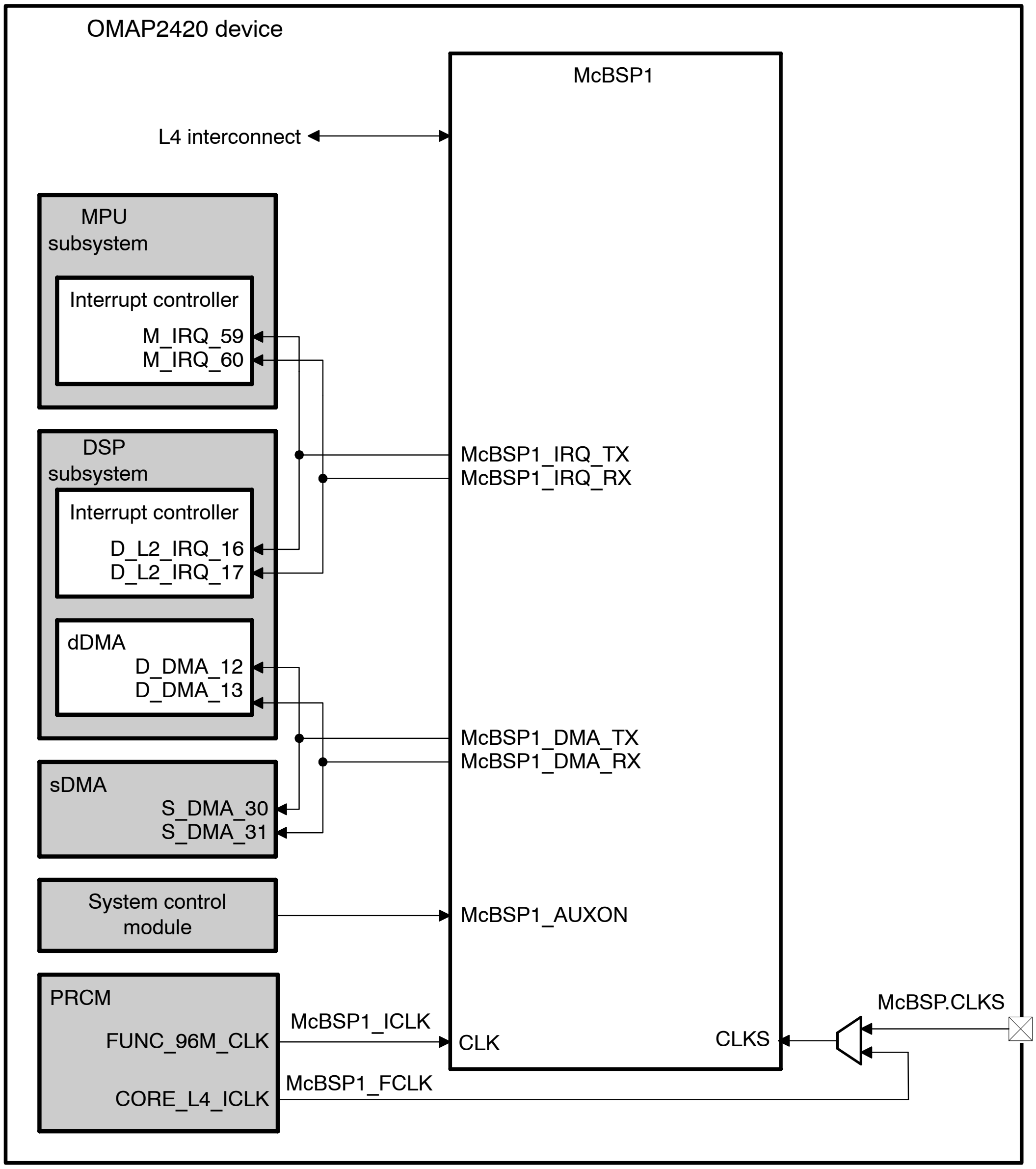
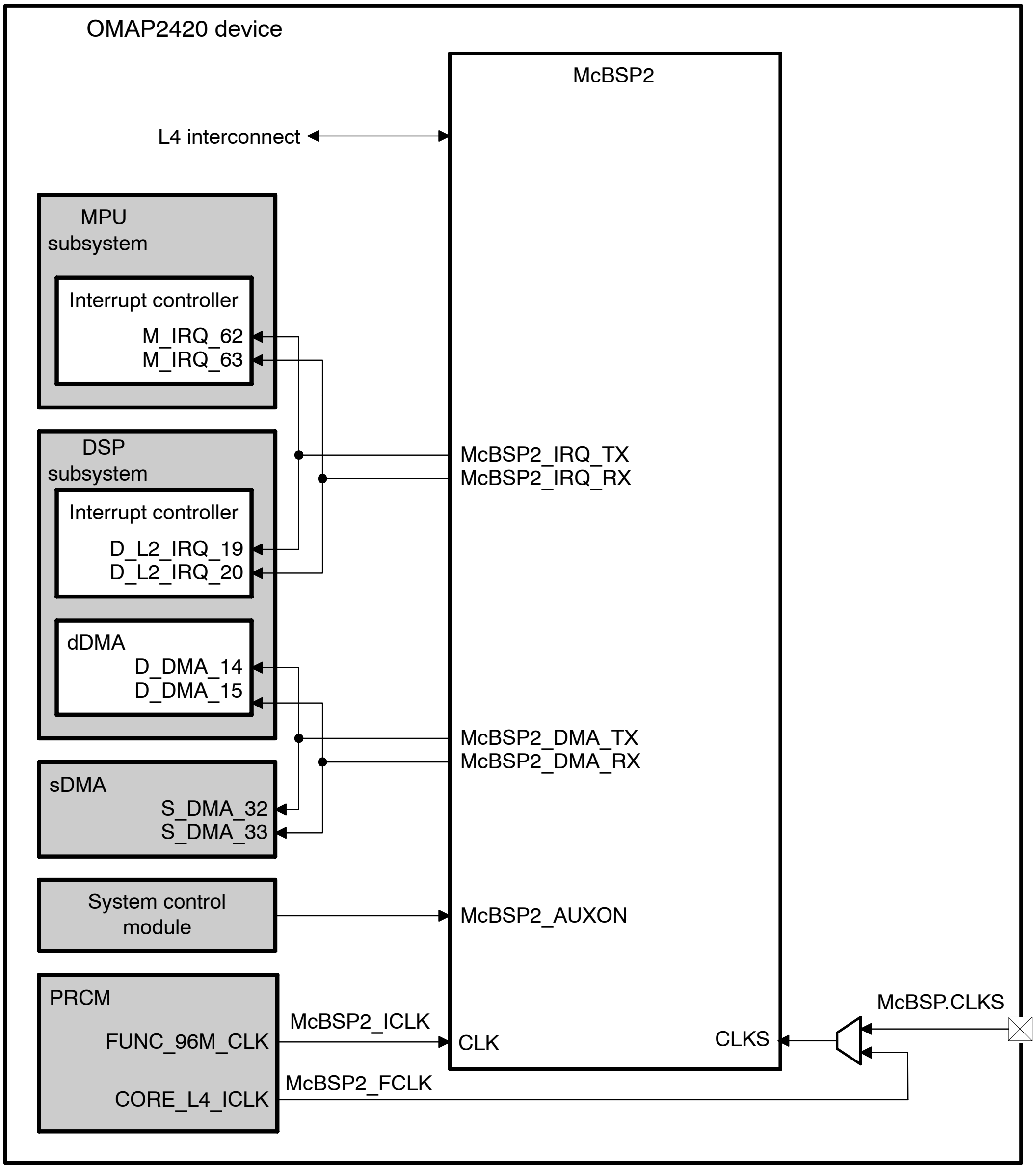


Figure 21−17. McBSP2 Integration



As described in Section 21.2, there are no external pins MCBSP2.CLKR and MCBSP2. FSR. The McBSP2 module CLKR and FSR signal sources are MCBSP2.CLKX and MCBSP2.FSX pins, respectively. Consequently, there is a light restriction on the McBSP2 module when used in full-duplex mode: Receive and transmit use the same clock and FSGs.

21.3.1 Clocking, Reset, and Power Management Scheme

21.3.1.1 McBSP1 Clocks

The McBSP1 module has two clock domains: The functional clock domain and the interface clock domain.

- McBSP1_FCLK belongs to the functional clock domain. It is a fixed 96-MHz clock provided by the PRCM. It is used to clock the internal module logic.

  MCBSP1_FCLK source is the FUNC_96M_CLK PRCM output. When the McBSP1 module does not require it anymore, the software can disable it at PRCM level, setting the EN_McBSP1 bit (CM_FCLKEN1_CORE[15]) in the PRCM registers. Note that the clock is effectively cut provided the other modules that receive it do not require it either.

  For details about PRCM registers settings and the way APLL 96M can be configured, see Chapter 5, *Power, Reset, and Clock Management*.

- McBSP1_ICLK is the interface clock. It runs at L4 interconnect clock speed and is used to trigger access to the McBSP1 L4 interface. Its source is the PRCM CORE_L4_ICLK signal. It can run at L3 frequency or L3/2 frequency (selection is made through the PRCM CM_CLKSEL1_CORE register CLKSEL_L4 bit field: CM_CLKSEL1_CORE[6:5]).

  When the McBSP1 module does not require it anymore, the software can disable it at PRCM level, setting the EN_McBSP1 bit (CM_ICLKEN1_CORE[15]) in the PRCM registers. Note that the clock is effectively cut, provided the other modules that receive it do not require it either.

  It is also possible to activate an auto-idle mode for this clock (PRCM CM_AUTOIDLE1_CORE register AUTO_McBSP1 bit set to 1: CM_AUTOIDLE1_CORE[15] = 1). In this case, McBSP1_ICLK follows the OMAP2420 L4 clock domain behavior. For more information, see Chapter 5, *Power, Reset, and Clock Management*.

Note:

The MCBSP1_AUXON bit (CONTROL_DEVCONF[5]) of system control module register CONTROL_DEVCONF allows local gating of the McBSP1 module interface clock. Setting this bit to 1 deactivates the interface clock.

CAUTION

The software must ensure correct clock management and avoid setting the CONTROL_DEVCONF[5] bit during an ongoing transaction. Setting this bit during an ongoing transaction causes loss of data.

21.3.1.2 McBSP2 Clocks

The McBSP2 module has two clock domains: The functional clock domain and the interface clock domain.

- McBSP2_FCLK belongs to the functional clock domain. It is a fixed 96-MHz clock provided by the PRCM. It is used to clock the internal module logic.

  The MCBSP2_FCLK source is the FUNC_96M_CLK PRCM output. When the McBSP2 module does not require it anymore, the software can disable it at PRCM level, setting the EN_McBSP2 bit (CM_FCLKEN1_CORE[16]) in the PRCM registers. Note that the clock is effectively cut, provided the other modules that receive it do not require it either.

  For details about PRCM registers settings and the way APLL 96M can be configured, see Chapter 5, *Power, Reset, and Clock Management*.

- McBSP2_ICLK is the interface clock. It runs at L4 interconnect clock speed and is used to trigger access to the McBSP2 L4 interface. Its source is the PRCM Core_L4_ICLK signal. It can run at L3 frequency or L3/2 frequency (selection is made through the PRCM CM_CLKSEL1_CORE register CLKSEL_L4 bit field: CM_CLKSEL1_CORE[6:5]).

  When the McBSP2 module does not require it anymore, the software can disable it at PRCM level, setting the EN_McBSP2 bit (CM_ICLKEN1_CORE[16]) in the PRCM registers. Note that the clock is effectively cut, provided the other modules that receive it do not require it either.

  It is also possible to activate an auto-idle mode for this clock (PRCM CM_AUTOIDLE1_CORE register AUTO_McBSP2 bit set to 1: CM_AUTOIDLE1_CORE[16] = 1). In this case, McBSP2_ICLK follows the OMAP2420 L4 clock domain behavior. For more information, see Chapter 5, *Power, Reset, and Clock Management*.

21.3.1.3 McBSP Reset Scheme

Both McBSP1 and McBSP2 belong to the CORE_RST domain. Activating the CORE_RST_N signal in the CORE_RST domain resets both of the modules. For more information, see Chapter 5, *Power, Reset, and Clock Management*.

McBSP register sets also provide several bits to reset parts of the McBSP modules.

21.3.1.4 McBSP Power Domain

The McBSPs belong to the CORE power domain. For more information about the CORE power domain, see Chapter 5, *Power, Reset, and Clock Management*.

21.3.2 Hardware Requests

21.3.2.1 McBSP1 Hardware Requests

The McBSP1 can generate two interrupts:

- McBSP1.IRQ_TX is an interrupt to the MPU and DSP subsystem interrupt controllers. It is mapped on M_IRQ_59 and D_L2_IRQ_16, respectively.
- McBSP1.IRQ_RX is an interrupt to the MPU and DSP subsystem interrupt controllers. It is mapped on M_IRQ_60 and D_L2_IRQ_17, respectively.

The McBSP1 can also generate two DMA events:

- McBSP1.DMA_TX is a DMA request to the DSP subsystem DMA controller (dDMA) and to the system DMA (sDMA). It is mapped on D_DMA_12 and S_DMA_30, respectively.
- McBSP1.DMA_RX is a DMA request to the DSP subsystem DMA controller (dDMA) and to the system DMA (sDMA). It is mapped on D_DMA_13 and S_DMA_31, respectively.

21.3.2.2 McBSP2 Hardware Requests

The McBSP2 can generate two interrupts:

- McBSP2.IRQ_TX is an interrupt to both the MPU subsystem and the DSP subsystem interrupt controllers. It is mapped on M_IRQ_62 and D_L2_IRQ_19, respectively.
- McBSP2.IRQ_RX is an interrupt to both the MPU subsystem and the DSP subsystem interrupt controllers. It is mapped on M_IRQ_63 and D_L2_IRQ_20, respectively.

The McBSP2 can also generate two DMA events:

- McBSP2.DMA_TX is a DMA request to both the DSP subsystem DMA controller (dDMA) and the system DMA (sDMA). It is mapped on D_DMA_14 and S_DMA_32, respectively.
- McBSP2.DMA_RX is a DMA request to both the DSP subsystem DMA controller (dDMA) and the system DMA (sDMA). It is mapped on D_DMA_15 and S_DMA_33, respectively.

21.3.3 Pin List and Pad Multiplexing With Other Functions

Table 21−3 gives the pin multiplexing for the McBSP1 and McBSP2 modules.

Table 21−3. Pin Multiplexing for McBSP1 and McBSP2 Modules

Function n	Description	DIR	Ball	Alternate Functions					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
McBSP.CLKS	External Clock Input (shared by both McBSPs)	I	M18				gpio.96		
			V5	cam.lclk			gpio.57		
McBSP1.DR	Received Serial Data	I	P18				gpio.95		
			U4	cam.d3	hw.dbg5		gpio.51		
McBSP1.CLKR	Receive Clock	IO	M15				gpio.92		
			V4	cam.d5	hw.dbg7		gpio.49		

Table 21−3. Pin Multiplexing for McBSP1 and McBSP2 Modules (Continued)

Function n	Description	DIR	Ball 2420	Alternate Functions					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
McBSP1.FSR	FSR	IO	P20	▒			gpio.93	spi2.ncs1	■
			W2	cam.d4	hw.dbg6	▒	gpio.50	■	■
McBSP1.DX	Transmitted Serial Data	O	P19	▒			gpio.94	■	■
			T3	cam.hs	hw.dbg1	▒	gpio.55	■	■
McBSP1.CLKX	Transmit Clock	IO	N19	▒			gpio.98	■	■
			V3	cam.d2	hw.dbg4	▒	gpio.52	■	■
McBSP1.FSX	FSX	IO	L14	▒		■	gpio.97	■	■
			U2	cam.vs	hw.dbg0	▒	gpio.56	■	■
McBSP2.DR	Received Serial Data	I	M21	▒	■	dss.d22	gpio.11	■	■
			J19	usb0.vp	▒	■	gpio.107	■	■
			W15	eac.ac_din	▒	■	gpio.115	■	■
McBSP2.DX	TranSmitted Serial Data	O	J20	usb0.puen	▒	■	gpio.106	■	■
			V15	eac.ac_dout	▒	■	gpio.116	■	■
McBSP2.CLKX	Combined Serial Clock	IO	P21	▒	■	dss.d23	gpio.12	■	■
			K20	usb0.vm	▒	■	gpio.108	uart2.rx	■
			Y15	eac.ac_sclk	▒	■	gpio.113	■	■
McBSP2.FSX	Combined frame-sync	IO	J18	usb0.rcv	▒	■	gpio.109	uart2.cts	■
			R14	eac.ac_fs	▒	■	gpio.114	■	■
			W7	dss.acbias	■	mcbsp2.fsx	gpio.48	■	■

21.3.4 L4 Interconnect Interface

Both McBSP1 and McBSP2 modules have an interface with the L4 interconnect through a dedicated target agent (TA). This TA, which is part of the L4 interconnect, provides status and configuration registers as listed in Table 21−4 and Table 21−5. By default, TA registers values are functional, but it is possible to overwrite them if required. For a complete description, see Chapter 6, *Internal Interconnect*.

Table 21−4. McBSP1 L4 Interconnect Registers

Register Name	Type	Register Width (Bits)	Physical Address
COMPONENT	R	32	0x4807 5000
AGENT_CONTROL	RW	32	0x4807 5020
AGENT_STATUS	R	32	0x4807 5028

Table 21−5. McBSP2 L4 Interconnect Registers

Register Name	Type	Register Width (Bits)	Physical Address
COMPONENT	R	32	0x4807 7000
AGENT_CONTROL	RW	32	0x4807 7020
AGENT_STATUS	R	32	0x4807 7028

21.3.5 Register Summary

Table 21−6 and Table 21−7 provide a register summary for McBSP1 and McBSP2 modules.

CAUTION

MCBSP registers are limited to 16-bit data access. 32-bit and 8-bit data access are not allowed and can corrupt register contents.

Table 21−6. Register Summary for McBSP1 Module

Register Name	Type	Register Width (Bits)	Physical Address
MCBSP_DRR2_REG	RW	16	0x4807 4000
MCBSP_DRR1_REG	RW	16	0x4807 4004
MCBSP_DXR2_REG	RW	16	0x4807 4008
MCBSP_DXR1_REG	RW	16	0x4807 400C
MCBSP_SPCR2_REG	RW	16	0x4807 4010
MCBSP_SPCR1_REG	RW	16	0x4807 4014
MCBSP_RCR2_REG	RW	16	0x4807 4018
MCBSP_RCR1_REG	RW	16	0x4807 401C
MCBSP_XCR2_REG	RW	16	0x4807 4020
MCBSP_XCR1_REG	RW	16	0x4807 4024
MCBSP_SRGR2_REG	RW	16	0x4807 4028
MCBSP_SRGR1_REG	RW	16	0x4807 402C
MCBSP_MCR2_REG	RW	16	0x4807 4030
MCBSP_MCR1_REG	RW	16	0x4807 4034
MCBSP_RCERA_REG	RW	16	0x4807 4038
MCBSP_RCERB_REG	RW	16	0x4807 403C

Table 21−6. Register Summary for McBSP1 Module (Continued)

Register Name	Type	Register Width (Bits)	Physical Address
MCBSP_XCERA_REG	RW	16	0x4807 4040
MCBSP_XCERB_REG	RW	16	0x4807 4044
MCBSP_PCR_REG	RW	16	0x4807 4048
MCBSP_RCERC_REG	RW	16	0x4807 404C
MCBSP_RCERD_REG	RW	16	0x4807 4050
MCBSP_XCERC_REG	RW	16	0x4807 4054
MCBSP_XCERD_REG	RW	16	0x4807 4058
MCBSP_RCERE_REG	RW	16	0x4807 405C
MCBSP_RCERF_REG	RW	16	0x4807 4060
MCBSP_XCERE_REG	RW	16	0x4807 4064
MCBSP_XCERF_REG	RW	16	0x4807 4068
MCBSP_RCERG_REG	RW	16	0x4807 406C
MCBSP_RCERH_REG	RW	16	0x4807 4070
MCBSP_XCERG_REG	RW	16	0x4807 4074
MCBSP_XCERH_REG	RW	16	0x4807 4078
MCBSP_REV_REG	RW	16	0x4807 407C

Table 21−7. Register Summary for McBSP2 Module

Register Name	Type	Register Width (Bits)	Physical Address
MCBSP_DRR2_REG	RW	16	0x4807 6000
MCBSP_DRR1_REG	RW	16	0x4807 6004
MCBSP_DXR2_REG	RW	16	0x4807 6008
MCBSP_DXR1_REG	RW	16	0x4807 600C
MCBSP_SPCR2_REG	RW	16	0x4807 6010
MCBSP_SPCR1_REG	RW	16	0x4807 6014
MCBSP_RCR2_REG	RW	16	0x4807 6018
MCBSP_RCR1_REG	RW	16	0x4807 601C
MCBSP_XCR2_REG	RW	16	0x4807 6020
MCBSP_XCR1_REG	RW	16	0x4807 6024
MCBSP_SRGR2_REG	RW	16	0x4807 6028
MCBSP_SRGR1_REG	RW	16	0x4807 602C
MCBSP_MCR2_REG	RW	16	0x4807 6030
MCBSP_MCR1_REG	RW	16	0x4807 6034
MCBSP_RCERA_REG	RW	16	0x4807 6038
MCBSP_RCERB_REG	RW	16	0x4807 603C
MCBSP_XCERA_REG	RW	16	0x4807 6040
MCBSP_XCERB_REG	RW	16	0x4807 6044

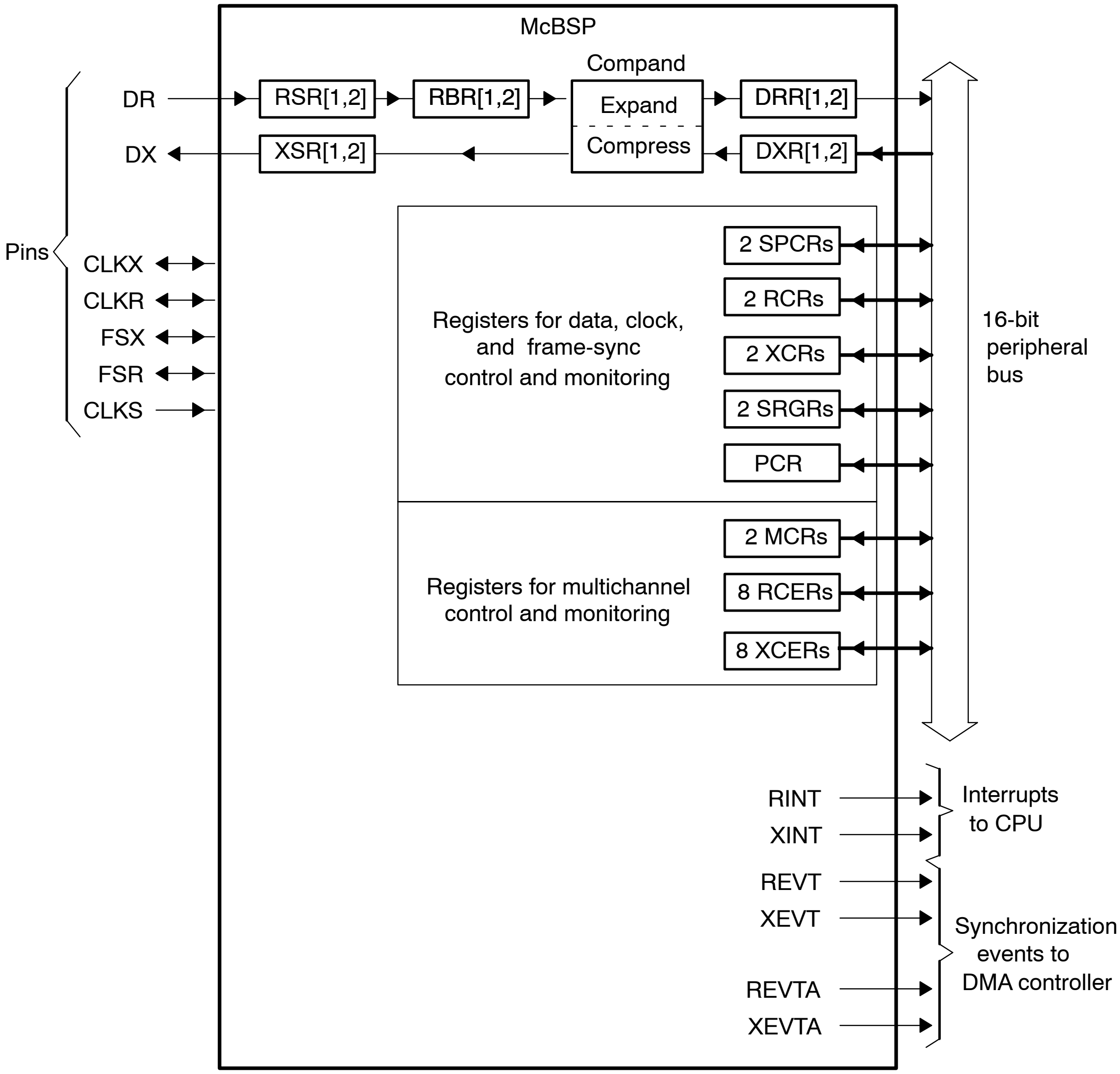
Table 21−7. Register Summary for McBSP2 Module (Continued)

Register Name	Type	Register Width (Bits)	Physical Address
MCBSP_PCR_REG	RW	16	0x4807 6048
MCBSP_RCERC_REG	RW	16	0x4807 604C
MCBSP_RCERD_REG	RW	16	0x4807 6050
MCBSP_XCERC_REG	RW	16	0x4807 6054
MCBSP_XCERD_REG	RW	16	0x4807 6058
MCBSP_RCERE_REG	RW	16	0x4807 605C
MCBSP_RCERF_REG	RW	16	0x4807 6060
MCBSP_XCERE_REG	RW	16	0x4807 6064
MCBSP_XCERF_REG	RW	16	0x4807 6068
MCBSP_RCERG_REG	RW	16	0x4807 606C
MCBSP_RCERH_REG	RW	16	0x4807 6070
MCBSP_XCERG_REG	RW	16	0x4807 6074
MCBSP_XCERH_REG	RW	16	0x4807 6078
MCBSP_REV_REG	RW	16	0x4807 607C

21.4 McBSP Functional Description

Figure 21−18 shows the McBSP, conceptually.

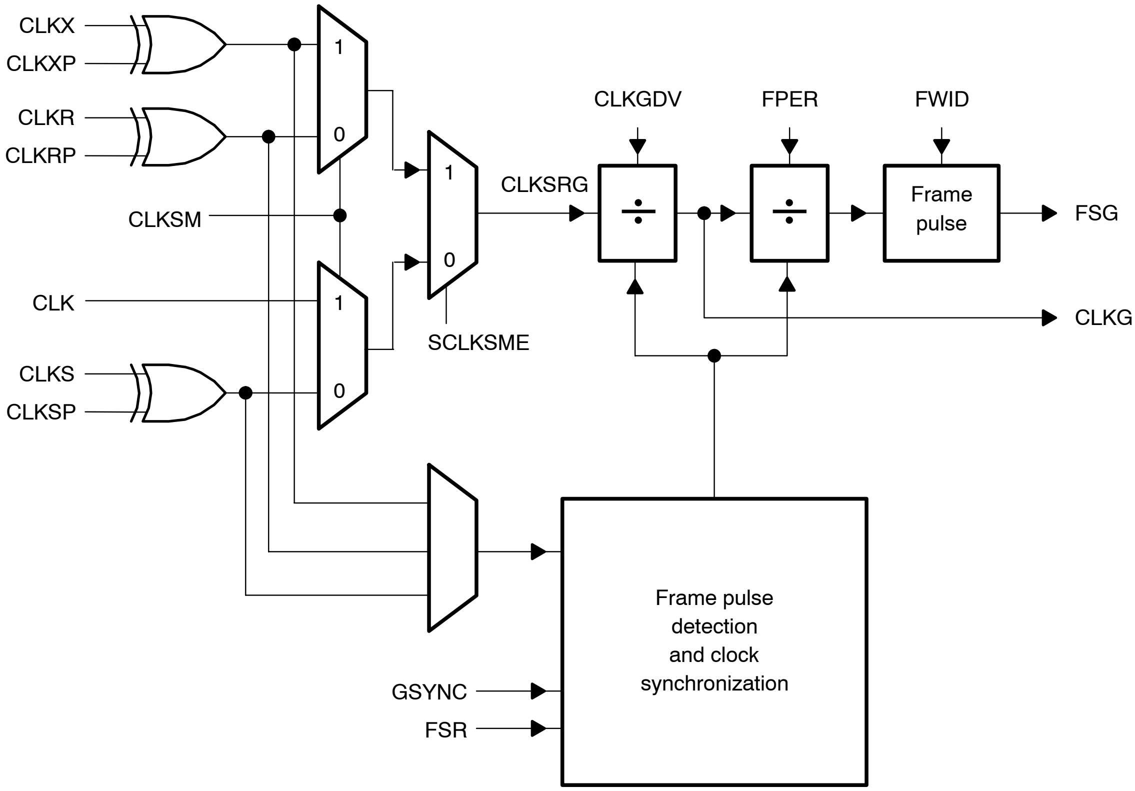
Figure 21−18. McBSP Conceptual Block Diagram



21.4.1 McBSP Sample Rate Generator

Each McBSP contains an SRG that can be used to generate an internal data clock (CLKG) and an internal FSG. CLKG can be used for bit shifting on the data receive (McBSPn.DR) pin and/or the data transmit (McBSPn.DX) pin. FSG can be used to initiate frame transfers on McBSPn.DR and/or McBSPn.DX. Figure 21−19 is a conceptual block diagram of the SRG.

Figure 21−19. Conceptual Block Diagram of the SRG



The source clock for the SRG (labeled CLKSRG in the diagram) can be supplied by either the CLK input pin of the module, by an external pin (McBSPn.CLKX or McBSPn.CLKR), or by the CLKS pin of the McBSP module, which can be connected either to the McBSPn_FCLK or the McBSP.CLKS pin. The source is selected with the SCLKME bit of the MCBSP_PCR_REG[7] register, the CLKSM bit of the MCBSP_SRGR2_REG[13] register, and the CONTROL_DEVCONF[2] or CONTROL_DEVCONF[6] bits.

If a pin is used, the polarity of the incoming signal can be inverted with the appropriate polarity bit (CLKSP bit of MCBSP_SRGR2_REG[14] register, CLKXP bit of MCBSP_PCR_REG register, or CLKRP bit of the MCBSP_PCR_REG[0] register).

The SRG has a three-stage clock divider that gives CLKG and FSG programmability.

The three stages provide:

- Clock divide-down. The source clock is divided according to the MCBSP_SRGR1_REG[7:0] register CLKGDV bits to produce CLKG.
- Frame period divide-down. CLKG is divided according to the MCBSP_SRGR2_REG[11:0] register FPER bit field to control the period from the start of a frame-pulse to the start of the next pulse.
- Frame-sync pulse-width countdown. CLKG cycles are counted according to the MCBSP_SRGR1_REG[15:8] register FWID bits to control the width of each frame-sync pulse.

In addition to the three-stage clock divider, the SRG has a frame-sync pulse detection and clock synchronization module that allows synchronization of the clock divide-down with an incoming frame-sync pulse on the McBSPn.FSR pin. This feature is enabled or disabled with the MCBSP_SRGR2_REG[15] register GSYNC bit.

For details on getting the SRG ready for operation, see Section 21.5, *McBSP Programming Model*.

21.4.1.1 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 (CLKRM and CLKXM) in the pin control register (MCBSP_PCR_REG[8] and MCBSP_PCR_REG[9], respectively).

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).

The effects of CLKRM = 1 and CLKXM = 1 on the McBSP are partially affected by the use of the digital loopback mode and the clock stop (SPI) mode, respectively, as described in Table 21−8. The digital loopback mode (described in Section 21.5.3.4) is selected with the MCBSP_SPCR1_REG[15] register DLB bit. The clock stop mode is selected with the CLKSTP bits of the MCBSP_SPCR1_REG register (MCBSP_ SPCR1_REG[12:11]).

When using the SRG as a clock source, make sure the SRG is enabled (GRST = 1).

Table 21−8. Effects of DLB and CLKSTP Bits on Clock Modes

Mode Bit Settings		**Effect**
CLKRM = 1	DLB = 0 (Digital loopback mode disabled)	CLKR is an output pin driven by the SRG output clock (CLKG).
	DLB = 1 (Digital loopback mode enabled)	CLKR is an output pin driven by internal CLKX. The source for CLKX depends on the CLKXM bit.
CLKXM = 1	CLKSTP = 00b or 01b (Clock stop (SPI) mode disabled)	CLKX is an output pin driven by the SRG output clock (CLKG).
	CLKSTP = 10b or 11b (Clock stop (SPI) mode enabled)	The McBSP is a master in an SPI system. Internal CLKX drives internal CLKR and the shift clocks of any SPI-compliant slave devices in the system. CLKX is driven by the internal SRG.

21.4.1.2 Frame Sync Generation in the SRG

The SRG can produce an FSG for use by the receiver, the transmitter, or both.

If you want the receiver to use FSG for frame-sync, make sure MCBSP_PCR_REG[10] register FSRM bit = 1. (When FSRM = 0, FSR is supplied through the McBSPn.FSR pin.)

If you want the transmitter to use FSG for frame-sync, you must set:

- FSXM = 1 in MCBSP_[11]_REG register: This indicates that FSX is supplied by the McBSP itself rather than from the McBSPn.FSX pin.
- FSGM = 1 in MCBSP_SRGR2_REG[12] register: This indicates that when FSXM = 1, FSX is supplied by the SRG. (When FSGM = 0 and FSXM = 1, the transmitter uses frame-sync pulses generated every time data is transferred from MCBSP_DXR[1,2]_REG registers to XSR[1,2].)

In either case, the SRG must be enabled (MCBSP_SPCR2_REG[6] register GRST bit = 1) and the frame-sync logic in the SRG must be enabled (MCBSP_SPCR2_REG[7] register FRST bit = 0).

Choosing the Width of the Frame-Sync Pulse

Each pulse on FSG has a programmable width. You program the MCBSP_SRGR1_REG[15:8] register FWID bits, and the resulting pulse width is (FWID + 1) CLKG cycles, where CLKG is the output clock of the SRG.

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 MCBSP_SRGR2_REG[15] register GSYNC bit = 1, FSG pulses in response to an inactive-to-active transition on the McBSPn.FSR pin. Thus, the frame-sync period is controlled by an external device.
- Otherwise, program the FPER bits of MCBSP_SRGR2_REG[11:0] register, and the resulting frame-sync period is (FPER + 1) CLKG cycles, where CLKG is the output clock of the SRG.

Keeping FSG Synchronized to an External Clock

When an external signal is selected to drive the SRG, the GSYNC bit of MCBSP_SRGR2_REG15] register and the McBSPn.FSR pin can be used to configure the timing of FSG pulses.

GSYNC = 1 ensures that the McBSP and an external device divide down the input clock with the same phase relationship. If GSYNC = 1, an inactive-to-active transition on the McBSPn.FSR pin triggers a resynchronization of CLKG and generation of FSG.

21.4.1.3 Synchronizing SRG Outputs to an External Clock

The SRG can produce a clock signal (CLKG) and an FSG signal based on an input clock signal that is either the CLK pin of the McBSP module, a signal coming from an external pin (McBSPn.CLKX or McBSPn.CLKR), or a signal coming from the CLKS pin of the McBSP module, which can be connected either to the McBSPn_FCLK or the McBSP.CLKS pin. The source is selected with the SCLKME bit of the MCBSP_PCR_REG[7] register, the CLKSM bit of the

MCBSP_SRGR2_REG[13] register, and the CONTROL_DEVCONF[2] or CONTROL_DEVCONF[6] bits.

Make GSYNC = 1 when you want the McBSP and an external device to divide down the input clock with the same phase relationship. If GSYNC = 1:

- An inactive-to-active transition on the McBSPn.FSR pin triggers a resynchronization of CLKG and a pulsing of FSG.
- CLKG always begins with a high state after synchronization.
- FSR is always detected at the same edge of the input clock signal that generates CLKG, no matter how long the FSR pulse is.
- The MCBSP_SRGR2_REG[11:0] register FPER bits are ignored because the frame-sync period on FSG is determined by the arrival of the next frame-sync pulse on the McBSPn.FSR pin.

If GSYNC = 0, CLKG runs freely and is not resynchronized, and the frame-sync period on FSG is determined by FPER.

Operating the Transmitter Synchronously With the Receiver

When GSYNC = 1, the transmitter can operate synchronously with the receiver provided that:

- FSX is programmed to be driven by FSG (FSGM = 1 in MCBSP_SRGR2_REG[12] register and FSXM = 1 in MCBSP_PCR_REG[11] register). If the input FSR has appropriate timing so that it can be sampled by the falling edge of CLKG, it can be used, instead, by setting FSXM = 0 and connecting FSR to FSX externally.
- The SRG clock drives the transmit and receive clocking (MCBSP_PCR_REG[8] CLKRM bit and MCBSP_PCR_REG[9] CLKXM bit are set to 1). Therefore, the CLK(R/X) pin must not be driven by any other driving source.

Synchronization Examples

Figure 21−20 and Figure 21−21 show the clock and frame-sync operation with various polarities of CLKS (the chosen input clock) and FSR. These figures assume FWID = 0x0 in MCBSP_SRGR1_REG[15:8] register, for an FSG pulse that is one CLKG cycle wide. The FPER bits of MCBSP_ SRGR2_REG[11:0] register are not programmed; the period from the start of a frame-sync pulse to the start of the next pulse is determined by the arrival of the next inactive-to-active transition on the McBSPn.FSR pin. Each of the figures shows what happens to CLKG when it is initially synchronized and GSYNC = 1, and when it is not initially synchronized and GSYNC = 1. Figure 21−21 shows a slower CLKG frequency (it has a larger divide-down value in the CLKGDV bits of MCBSP_SRGR1_REG[7:0] register).

Figure 21−20. CLKG Synchronization and FSG Generation (GSYNC = 1 and CLKGDV = 1)

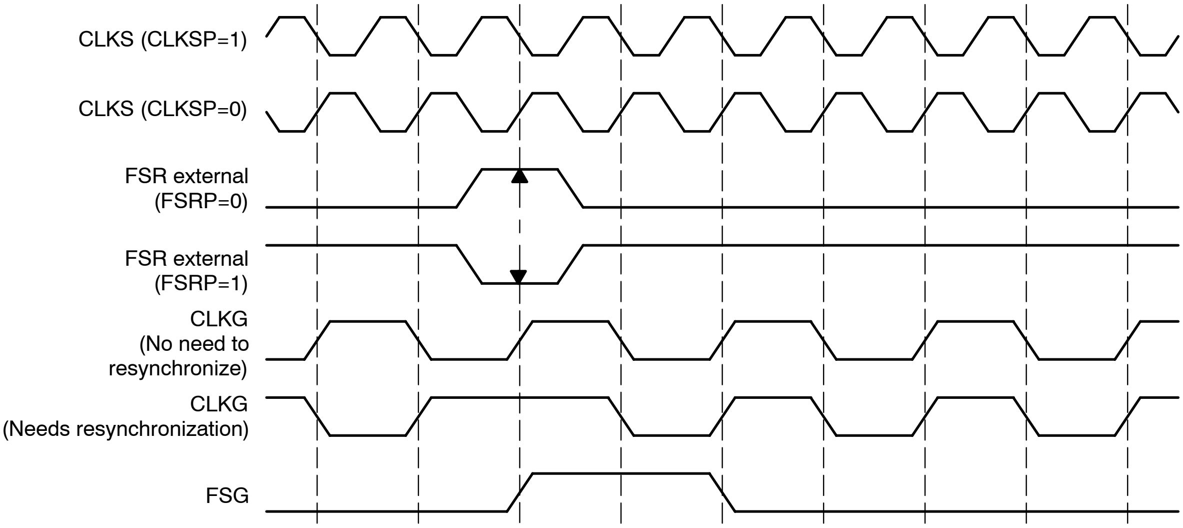
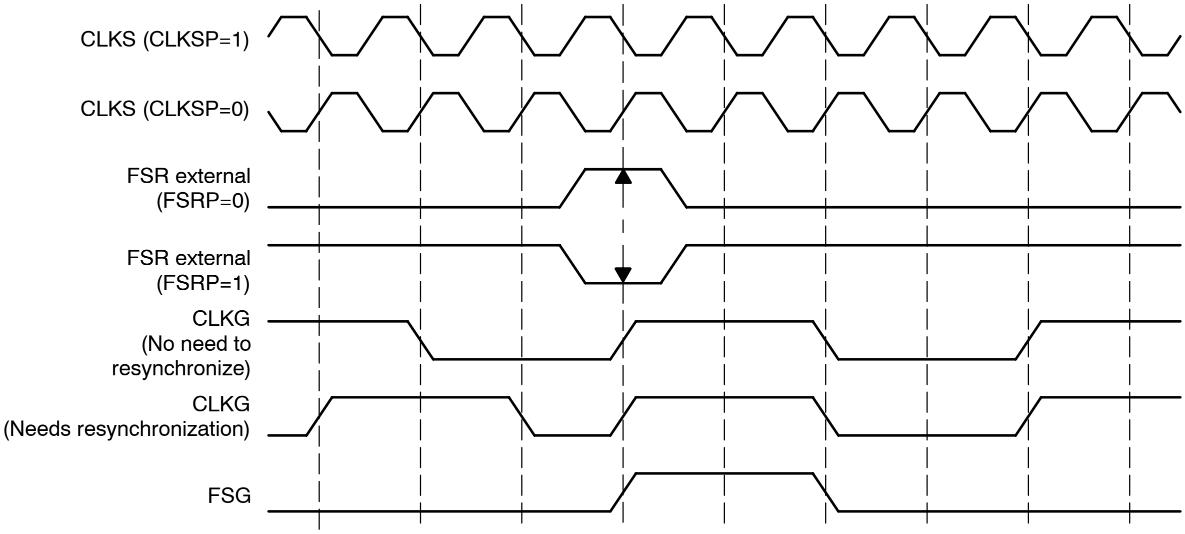


Figure 21−21. CLKG Synchronization and FSG Generation (GSYNC = 1 and CLKGDV = 3)



21.4.2 McBSP Exception/Error Conditions

21.4.2.1 Introduction

There are five serial port events that can constitute a system error:

- Receiver overrun: RFULL = 1 (MCBSP_SPCR1_REG[2] = 1):

  This occurs when MCBSP_DRR1_REG register has not been read since the last RBR-to-DRR copy. Consequently, the receiver does not copy a new word from the RBR(s) to the MCBSP_DRR[1,2]_REG registers and the RSR(s) are now full with another new word shifted in from McBSPn.DR. Therefore, RFULL = 1 indicates an error condition wherein any new data that arrives at this time on McBSPn.DR replaces the contents of the RSR(s), and the previous word is lost. The RSRs continue to be overwritten as long as new data arrives on McBSPn.DR and MCBSP_DRR1_REG register is not read. For more information about overrun in the receiver, see Section 21.4.2.2, *Overrun in the Receiver*.

- Unexpected FSR pulse: RSYNCERR = 1 (MCBSP_SPCR1_REG[3] = 1):

  This occurs during reception when RFIG = 0 and an unexpected frame-sync pulse occurs. An unexpected frame-sync pulse is one that begins the next frame transfer before all the bits of the current frame have been received. Such a pulse causes data reception to abort and restart. If new data has been copied into the RBR(s) from the RSR(s) since the last RBR-to-DRR copy, this new data in the RBR(s) is lost. This is because no RBR-to-DRR copy occurs; the reception is restarted. For more information about FSR errors, see Section 21.4.2.3, *Unexpected FSR Pulse*.

- Transmitter data overwrite:

  This occurs when the CPU or DMA controller overwrites data in the MCBSP_DXR[1,2]_REG registers before the data is copied to the XSR(s) transmit shift registers. The overwritten data never reaches the McBSPn.DX pin. For more information about overwrite in the transmitter, see Section 21.4.2.4, *Overwrite in the Transmitter*.

- Transmitter underflow: XEMPTY = 0 (MCBSP_SPCR2_REG[2] = 0):

  If a new FSG arrives before new data is loaded into MCBSP_DXR1_REG register, the previous data in the MCBSP_DXR[1,2]_REG registers is sent again. This procedure continues for every new frame-sync pulse that arrives until MCBSP_DXR1_REG register is loaded with new data. For more information about underflow in the transmitter, see Section 21.4.2.5, *Underflow in the Transmitter*.

- Unexpected FSX pulse: XSYNCERR = 1 (MCBSP_SPCR2_REG[3] = 0):

  This occurs during transmission when XFIG = 0 and an unexpected frame-sync pulse occurs. An unexpected frame- synchronization pulse is one that begins the next frame transfer before all the bits of the current frame have been transferred. Such a pulse causes the current data transmission to abort and restart. If new data has been written to the MCBSP_DXR[1,2]_REG registers since the last DXR-to-XSR copy, the current value in the XSR(s) is lost. For more information about FSX errors, see Section 21.4.2.6, *Unexpected FSX Pulse*.

21.4.2.2 Overrun in the Receiver

RFULL = 1 (MCBSP_SPCR1_REG[2] = 1) indicates that the receiver has experienced overrun and is in an error condition. RFULL bit is set when all of the following conditions are met:

1) MCBSP_DRR1_REG has not been read since the last RBR-to-DRR copy (MCBSP_SPCR1_REG[1] register RRDY bit = 1).
2) RBR1 is full and an RBR-to-DRR copy has not occurred.
3) RSR1 is full and an RSR1-to-RBR copy has not occurred.

As previously described, data arriving on McBSPn.DR is continuously shifted into RSR1 (for word length of 16 bits or smaller) or RSR2 and RSR1 (for word length larger than 16 bits). Once a complete word is shifted into the RSR(s), an RSR-to-RBR copy can occur only if the previous data in RBR1 has been copied to MCBSP_DRR1_REG register. The MCBSP_SPCR1_REG[1] register RRDY bit is set when new data arrives in MCBSP_DRR1_REG and is cleared when that data is read from MCBSP_DRR1_REG. Until RRDY = 0, the next RBR-to-DRR copy does not take place, and the data is held in the RSR(s). New data arriving on the McBSPn.DR pin is shifted into RSR(s), and the previous content of the RSR(s) is lost.

You can prevent the loss of data if MCBSP_DRR1_REG is read no later than 2.5 cycles before the end of the third word is shifted into the RSR1.

Note:

If both MCBSP_DRR[1,2]_REG registers are required (word length larger than 16 bits), the CPU or the DMA controller must read from MCBSP_DRR2_REG first and then from MCBSP_DRR1_REG. As soon as MCBSP_DRR1_REG is read, the next RBR−to−DRR copy is initiated. If MCBSP_DRR2_REG is not read first, the data in MCBSP_DRR2_REG is lost.

After the receiver starts running from reset, a minimum of three words must be received before the RFULL bit is set. Either of the following events clears the RFULL bit and allows subsequent transfers to be read properly:

- ☐ The CPU or DMA controller reads MCBSP_DRR1_REG.
- ☐ The receiver is reset individually (MCBSP_SPCR1_REG[0] register RRST bit = 0) or as part of an OMAP2420 reset.

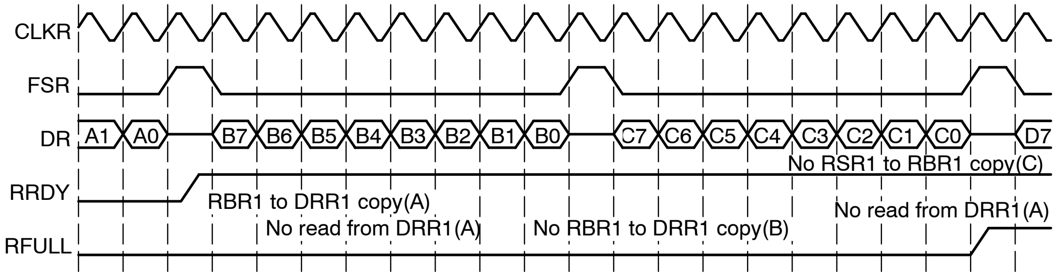
Another frame-sync pulse is required to restart the receiver.

Example of the Overrun Condition

Figure 21−22 shows the receive overrun condition. Because serial word A is not read from MCBSP_DRR1_REG register before serial word B arrives in

RBR1, B is not transferred to MCBSP_DRR1_REG yet. Another new word (C) arrives and RSR1 is full with this data. MCBSP_DRR1_REG is finally read, but not earlier than 2.5 cycles before the end of word C. Therefore, new data (D) overwrites word C in RSR1. If MCBSP_DRR1_REG is not read in time, the next word can overwrite D.

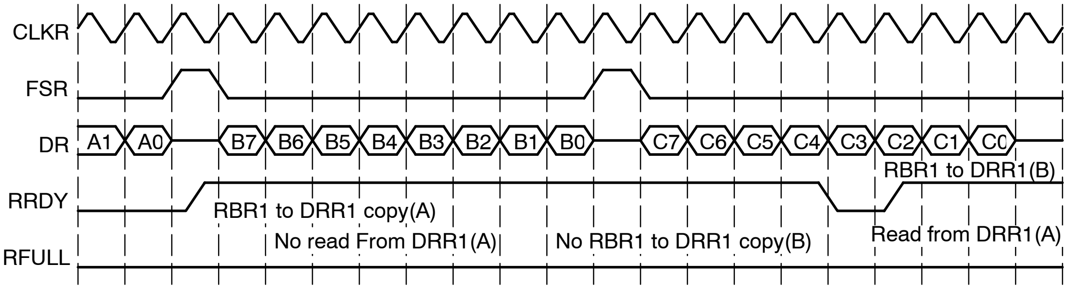
Figure 21−22. Overrun in the McBSP Receiver



Example of Preventing the Overrun Condition

Figure 21−23 shows the case where RFULL is set, but the overrun condition is prevented by a read from MCBSP_DRR1_REG at least 2.5 cycles before the next serial word (C) is completely shifted into RSR1. This ensures that an RBR1-to-DRR1 copy of word B occurs before the receiver attempts to transfer word C from RSR1 to RBR1.

Figure 21−23. Overrun Prevented in the McBSP Receiver



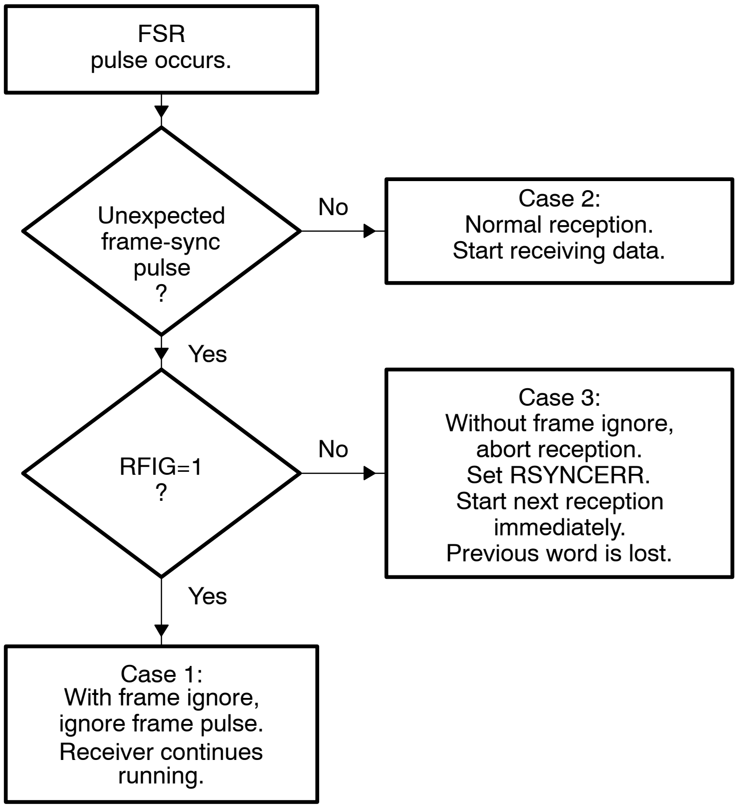
21.4.2.3 Unexpected FSR Pulse

The subsections in this section show how the McBSP responds to any FSR pulses, including an unexpected pulse, an example of a frame-sync error, and an example of how to prevent such an error.

Possible Responses to Receive Frame-Sync Pulses

Figure 21−24 shows the decision tree that the receiver uses to handle all incoming frame-sync pulses. The figure assumes that the receiver has been started (RRST = 1 in MCBSP_SPCR1_REG[0] register). Case 3 in the figure is the case in which an error occurs.

Figure 21−24. Possible Responses to Receive Frame-Sync Pulses


FSR
pulse occurs.
Unexpected frame-sync pulse ?
No
Case 2:
Normal reception.
Start receiving data.
Yes
RFIG=1 ?
No
Case 3:
Without frame ignore, abort reception.
Set RSYNCERR.
Start next reception immediately.
Previous word is lost.
Yes
Case 1:
With frame ignore, ignore frame pulse.
Receiver continues running.

Any one of the three following cases can occur:

- **Case 1:** Unexpected internal FSR pulses with MCBSP_RCR2_REG[2] register RFIG bit = 1. FSR pulses are ignored, and the reception continues.
- **Case 2:** Normal serial port reception. Reception continues normally because the frame-sync pulse is not unexpected. There are three possible reasons why a receive operation can not be in progress when the pulse occurs:
  - The FSR pulse is the first after the receiver is enabled (MCBSP_SPCR1_REG[0] register RRST bit = 1).
  - The FSR pulse is the first after MCBSP_DRR[1,2]_REG register is read, clearing a receiver full condition (MCBSP_SPCR1_REG[2] register RFULL bit = 1).
  - The serial port is in the interpacket intervals. The programmed data delay for reception (programmed with the MCBSP_RCR2_REG[1:0] register RDATDLY bit field) may start during these inter-packet intervals for the first bit of the next word to be received. Thus, at maximum frame frequency, frame-sync can still be received 0 to 2 clock cycles before the first bit of the synchronized frame.
- **Case 3:** Unexpected FSR with MCBSP_RCR2_REG[2] register RFIG bit = 0 (frame-sync pulses not ignored). Unexpected frame-sync pulses can originate from an external source or from the internal SRG.

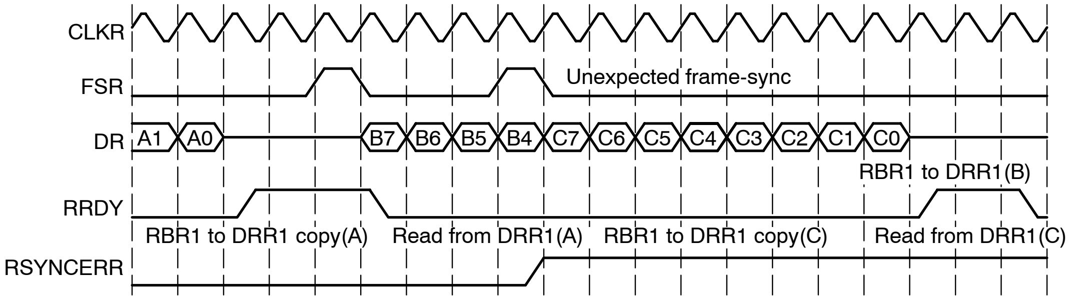
  If a frame-sync pulse starts the transfer of a new frame before the current frame is fully received, this pulse is treated as an unexpected frame-sync pulse, and the receiver sets the FSR error bit (RSYNCERR) in MCBSP_SPCR1_REG[3] register. RSYNCERR can be cleared only by a receiver reset or by writing 0 to this bit.

  If the McBSP is to notify the CPU of FSR errors, set a special receive interrupt mode with the MCBSP_SPCR1_REG[5:4] register RINTM bits. When RINTM = 11b, the McBSP sends a receive interrupt (RINT) request to the CPU each time that RSYNCERR is set.

Example of an Unexpected Receive Frame-Sync Pulse

Figure 21−25 shows an unexpected FSR pulse during normal operation of the serial port, with time intervals between data packets. When the unexpected frame-sync pulse occurs, the RSYNCERR bit is set, the reception of data B is aborted, and the reception of data C begins. In addition, if RINTM = 11b, the McBSP sends a receive interrupt (RINT) request to the CPU.

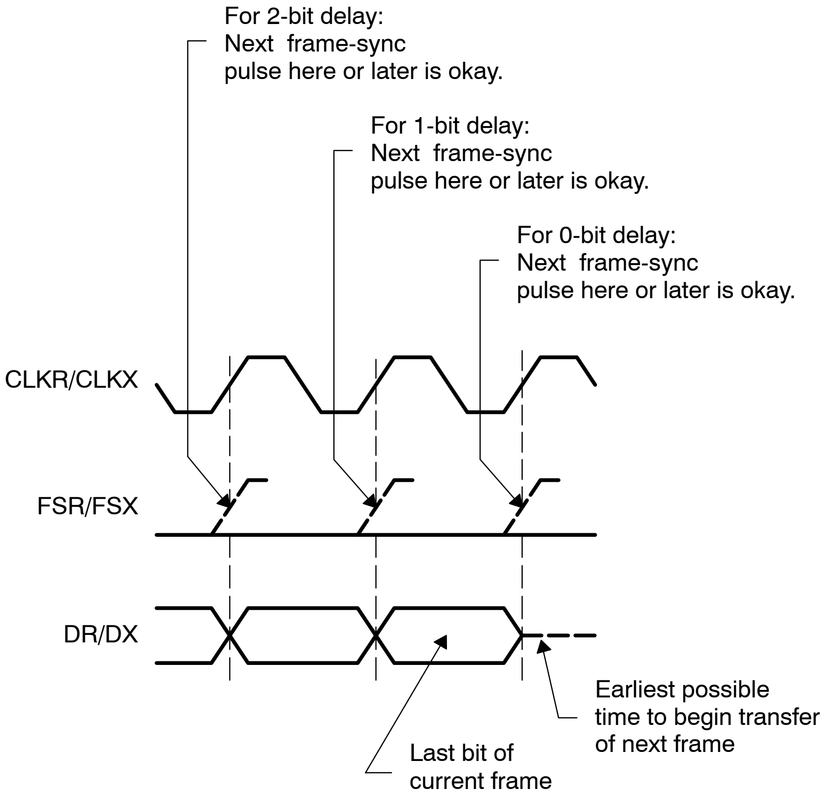
Figure 21−25. Unexpected Frame-Sync Pulse During a McBSP Reception



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 MCBSP_RCR2_[1:0] RDATDLY bits. For each possible data delay, Figure 21−26 shows when a new frame-sync pulse on FSR can safely occur relative to the last bit of the current frame.

Figure 21−26. Proper Positioning of Frame-Sync Pulses



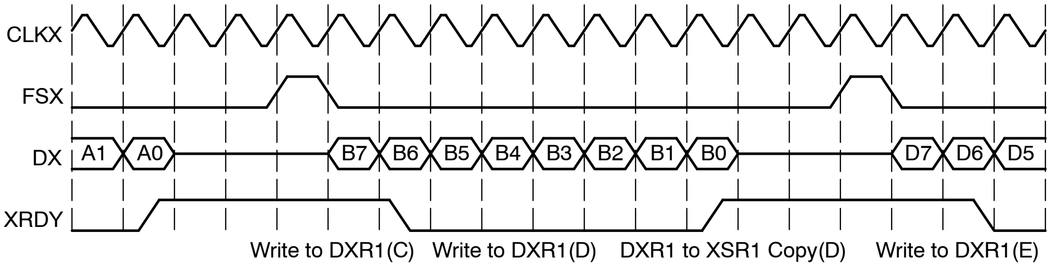
21.4.2.4 Overwrite in the Transmitter

As described in the McBSP transmission section, the transmitter must copy the data previously written to the MCBSP_DXR[1,2]_REG registers by the CPU or DMA controller into the XSR(s) and then shift each bit from the XSR(s) to the McBSPn.DX pin. If new data is written to the MCBSP_DXR[1,2]_REG registers before the previous data is copied to the XSR(s), the previous data in the MCBSP_DXR[1,2]_REG registers is overwritten and thus lost.

Example of Overwrite Condition

Figure 21−27 shows what happens if the data in MCBSP_DXR1_REG register is overwritten before being transmitted. Initially, MCBSP_DXR1_REG is loaded with data C. A subsequent write to MCBSP_DXR1_REG overwrites C with D before C is copied to XSR1. Thus, C is never transmitted on McBSPn.DX.

Figure 21−27. Overwritten Data in the McBSP Transmitter



Preventing Overwrites

You can prevent CPU overwrites by making the CPU:

- ❑ Poll for XRDY = 1 in MCBSP_SPCR2_REG[1] register before writing to the MCBSP_DXR[1,2]_REG registers. XRDY is set when data is copied from MCBSP_DXR1_REG to XSR1 and is cleared when new data is written to MCBSP_DXR1_REG.
- ❑ Wait for a transmit interrupt (XINT) before writing to the MCBSP_DXR[1,2]_REG registers. When XINTM = 00b in MCBSP_SPCR2_REG[5:4] register, the transmitter sends XINT to the CPU each time XRDY is set.

You can prevent DMA overwrites by synchronizing DMA transfers to the transmit synchronization event XEVT. The transmitter sends an XEVT signal each time XRDY is set.

21.4.2.5 Underflow in the Transmitter

The McBSP indicates a transmitter empty (or underflow) condition by clearing the XEMPTY bit in MCBSP_SPCR2_REG[2] register. Either of the following events activates XEMPTY (XEMPTY = 0):

- ❑ MCBSP_DXR1_REG has not been loaded since the last DXR-to-XSR copy, and all bits of the data word in the XSR(s) have been shifted out on the McBSPn.DX pin.
- ❑ The transmitter is reset (by forcing XRST = 0 in MCBSP_SPCR2_REG[0] register, or by an OMAP2420 reset) and is then restarted.

In the underflow condition, the transmitter continues to transmit the old data that is in the MCBSP_DXR[1,2]_REG registers for every new FSX signal until a new value is loaded into MCBSP_DXR1_REG by the CPU or the DMA controller.

Note:

If both MCBSP_DXR[1,2]_REG registers are required (word length larger than 16 bits), the CPU or the DMA controller must load MCBSP_DXR2_REG first and then load MCBSP_DXR1_REG. As soon as MCBSP_DXR1_REG is loaded, the contents of both MCBSP_DXR[1,2]_REG registers are copied to the transmit shift registers (XSRs). If MCBSP_DXR2_REG is not loaded first, the previous content of MCBSP_DXR2_REG is passed to the XSR2.

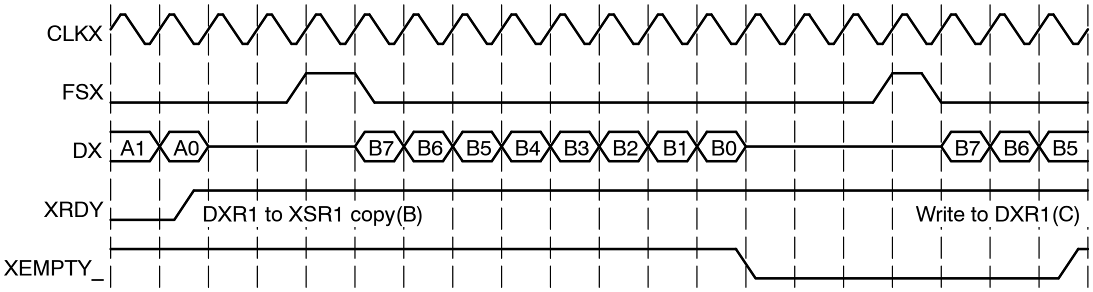
XEMPTY is deactivated (XEMPTY = 1) when a new word in MCBSP_DXR1_REG is transferred to XSR1. If FSXM = 1 in MCBSP_PCR_REG[11] and FSGM = 0 in MCBSP_SRGR2_REG[12], the transmitter generates a single internal FSX pulse in response to a DXR-to-XSR copy. Otherwise, the transmitter waits for the next frame-sync pulse before sending out the next frame on McBSPn.DX.

When the transmitter is taken out of reset (XRST = 1), it is in a transmitter ready (MCBSP_SPCR2_REG[1] register XRDY bit = 1) and transmitter empty (XEMPTY = 0) state. If MCBSP_DXR1_REG is loaded by the CPU or the DMA controller before internal FSX goes active high, a valid DXR-to-XSR transfer occurs. This allows for the first word of the first frame to be valid even before the FSX pulse is generated or detected. Alternatively, if a FSX pulse is detected before MCBSP_DXR1_REG is loaded, zeros are output on McBSPn.DX.

Example of the Underflow Condition

Figure 21−28 shows an underflow condition. After B is transmitted, MCBSP_DXR1_REG is not reloaded before the subsequent frame-sync pulse. Thus, B is again transmitted on McBSPn.DX.

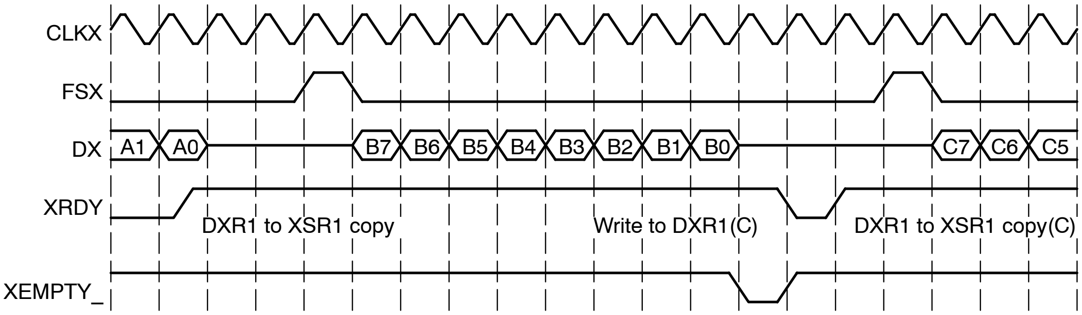
Figure 21−28. Underflow During McBSP Transmission



Example of Preventing the Underflow Condition

Figure 21−29 shows the case of writing to MCBSP_DXR1_REG register just before an underflow condition would otherwise occur. After B is transmitted, C is written to MCBSP_DXR1_REG before the next frame-sync pulse. As a result, there is no underflow; B is not transmitted twice.

Figure 21−29. Underflow Prevented in the McBSP Transmitter

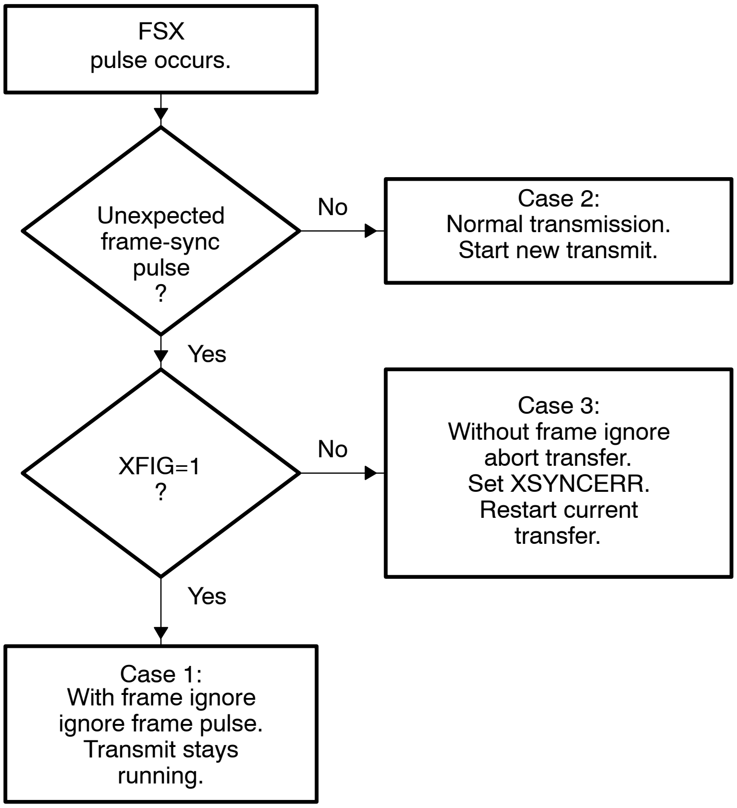


21.4.2.6 Unexpected FSX Pulse

Possible Responses to Transmit Frame-Sync Pulses

Figure 21−30 shows the decision tree that the transmitter uses to handle all incoming frame-sync pulses. The figure assumes that the transmitter is started (XRST = 1 in MCBSP_SPCR2_REG[0]). Case 3 in the figure is the case in which an error occurs.

Figure 21−30. Possible Responses to FSX Pulses


FSX pulse occurs.
Unexpected frame-sync pulse ?
No
Case 2: Normal transmission. Start new transmit.
Yes
XFIG=1 ?
No
Case 3: Without frame ignore abort transfer. Set XSYNCERR. Restart current transfer.
Yes
Case 1: With frame ignore ignore frame pulse. Transmit stays running.

Any one of the three following cases can occur:

- **Case 1:** Unexpected internal FSX pulses with MCBSP_XCR2_REG[2] register XFIG bit = 1. FSX pulses are ignored, and the transmission continues.

- ❑ **Case 2:** Normal serial port transmission. Transmission continues normally because the frame-sync pulse is not unexpected. There are two possible reasons why a transmit operation can not be in progress when the pulse occurs:
  - ■ This FSX pulse is the first after the transmitter is enabled (XRST = 1).
  - ■ The serial port is in the interpacket intervals. The programmed data delay for transmission (programmed with the XDATDLY bits of MCBSP_XCR2_REG[1:0]) can start during these interpacket intervals before the first bit of the previous word is transmitted. Thus, at maximum packet frequency, frame-sync can still be received 0 to 2 clock cycles before the first bit of the synchronized frame.
- ❑ **Case 3:** Unexpected FSX with XFIG = 0 (frame-sync pulses not ignored). Unexpected frame-sync pulses can originate from an external source or from the internal SRG.

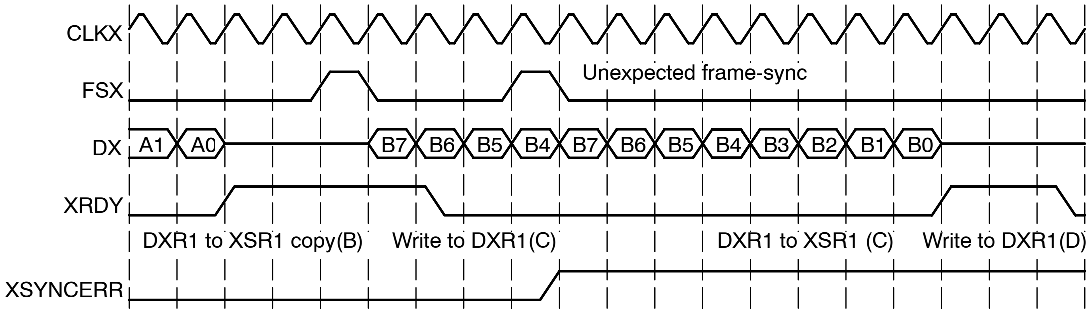
  If a frame-sync pulse starts the transfer of a new frame before the current frame is fully transmitted, this pulse is treated as an unexpected frame-sync pulse, and the transmitter sets the FSX error bit (XSYNCERR) in MCBSP_SPCR2_REG[3]. XSYNCERR can be cleared only by a transmitter reset or by a write of 0 to this bit.

If you want the McBSP to notify the CPU of frame-sync errors, you can set a special transmit interrupt mode with the MCBSP_SPCR2_REG[5:4] register XINTM bits. When XINTM = 11b, the McBSP sends a transmit interrupt (XINT) request to the CPU each time that XSYNCERR is set.

Example of Unexpected FSX Pulse

Figure 21−31 shows an unexpected FSX pulse during normal operation of the serial port with intervals between the data packets. When the unexpected frame-sync pulse occurs, the XSYNCERR bit is set and the transmission of data B is restarted because no new data has been passed to XSR1 yet. In addition, if XINTM = 11b, the McBSP sends a transmit interrupt (XINT) request to the CPU.

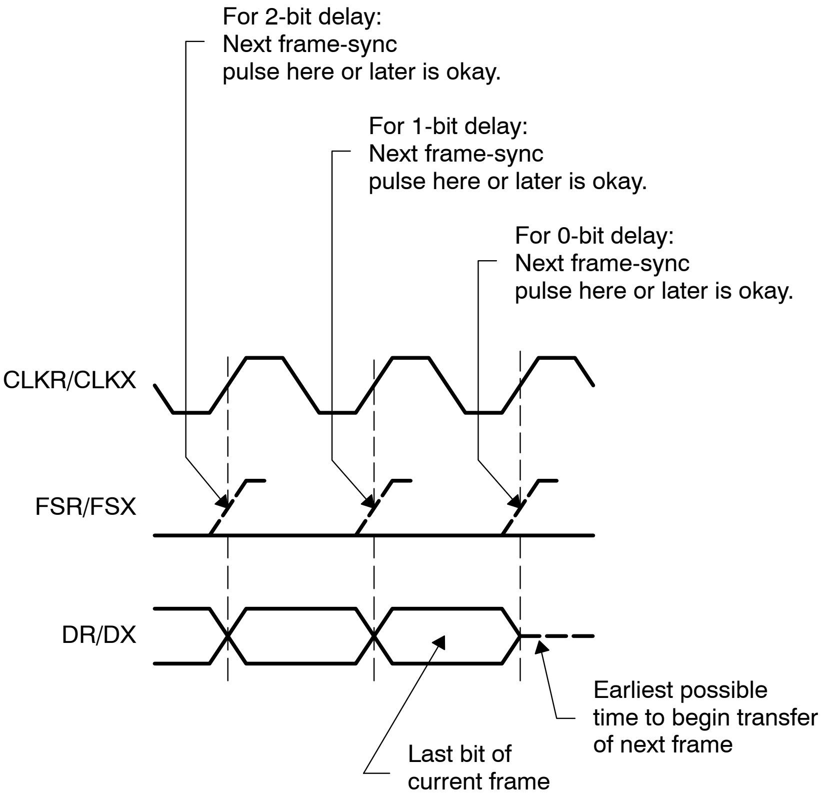
Figure 21−31. Unexpected Frame-Sync Pulse During a McBSP Transmission



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 MCBSP_XCR2_REG[1:0] register XDATDLY bits. For each possible data delay, Figure 21−32 shows when a new frame-sync pulse on FSX can safely occur relative to the last bit of the current frame.

Figure 21−32. Proper Positioning of Frame-Sync Pulses



21.4.3 Multichannel Selection Modes

21.4.3.1 Channels, Blocks, and Partitions

A McBSP channel is a time slot for shifting in/out the bits of one serial word. Each McBSP supports up to 128 channels for reception and 128 channels for transmission.

In the receiver and in the transmitter, the 128 available channels are divided into eight blocks that each contain 16 contiguous channels:

Block 0: Channels 0–15
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 (see Section 21.4.3.4), 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 eight-partition mode (see Section 21.4.3.5), blocks 0 through 7 are automatically assigned to partitions, A through H, respectively.

The number of partitions for reception and the number of partitions for transmission are independent. For example, it is possible to use two receive partitions (A and B) and eight transmit partitions (A–H).

21.4.3.2 Multichannel Selection

When a McBSP uses a time-division multiplexed (TDM) data stream while communicating with other McBSPs or serial devices, the McBSP may need to receive and/or transmit on only a few channels. To save memory and bus bandwidth, you can use a multichannel selection mode to prevent data flow in some of the channels.

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 has one receive multichannel selection mode (see Section 21.4.3.6) and three transmit multichannel selection modes (see Section 21.4.3.7).

21.4.3.3 Configuring a Frame for Multichannel Selection

Before you enable a multichannel selection mode, make sure you properly configure the data frame:

- ☐ Select a single-phase frame (MCBSP_RCR2_REG[15] register RPHASE bit and MCBSP_XCR2_REG[15] register XPHASE bit = 0). Each frame represents a TDM data stream.
- ☐ Set a frame length (in MCBSP_RCR1_REG[14:8] register RFRLEN1 bit field and in MCBSP_XCR1_REG[14:8] register XFRLEN1 bit 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 XFRLEN1 = 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.

21.4.3.4 Using Two Partitions

For multichannel selection operation in the receiver and/or the transmitter, you can use two partitions or eight partitions. If you choose the two-partition mode (RMCME = 0 for reception, XMCME = 0 for transmission), McBSP channels are activated using an alternating scheme. In response to a frame-sync pulse, the receiver or transmitter begins with the channels in partition A and then alternates between partitions B and A until the complete frame is transferred. When the next frame-sync pulse occurs, the next frame is transferred beginning with the channels in partition A.

Assigning Blocks to Partitions A and B

For reception, any two of the eight receive-channel blocks can be assigned to receive partitions A and B, which means up to 32 receive channels can be enabled at any given time. 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

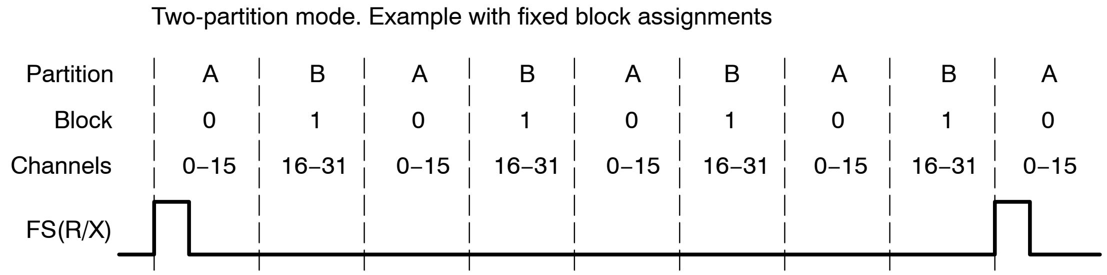
- Assign an even-numbered channel block (0, 2, 4, or 6) to receive partition A by writing to the RPABLK bits. In the receive multichannel-selection mode, the channels in this partition are controlled by receive channel enable register A (MCBSP_RCERA_REG).
- Assign an odd-numbered block (1, 3, 5, or 7) to receive partition B with the RPBBLK bits. In the receive multichannel selection mode, the channels in this partition are controlled by receive channel enable register B (MCBSP_RCERB_REG).

For Transmission

- Assign an even-numbered channel block (0, 2, 4, or 6) to transmit partition A by writing to the XPABLK bits. In one of the transmit multichannel selection modes (described in Section 21.4.3.7), the channels in this partition are controlled by transmit channel-enable register A (MCBSP_XCERA_REG).
- Assign an odd-numbered block (1, 3, 5, or 7) to transmit partition B with the XPBBLK bits. In one of the transmit multichannel selection modes, the channels in this partition are controlled by transmit channel-enable register B (MCBSP_XCERB_REG).

Figure 21−33 shows an example of alternating between the channels of partition A and the channels of partition B. Channels 0–15 have been assigned to partition A, and channels 16–31 have been assigned to partition B. In response to a frame-sync pulse, the McBSP begins a frame transfer with partition A and then alternates between partitions B and A until the complete frame is transferred.

Figure 21−33. Alternating Between Partitions A and B Channels



Reassigning Blocks During Reception/Transmission

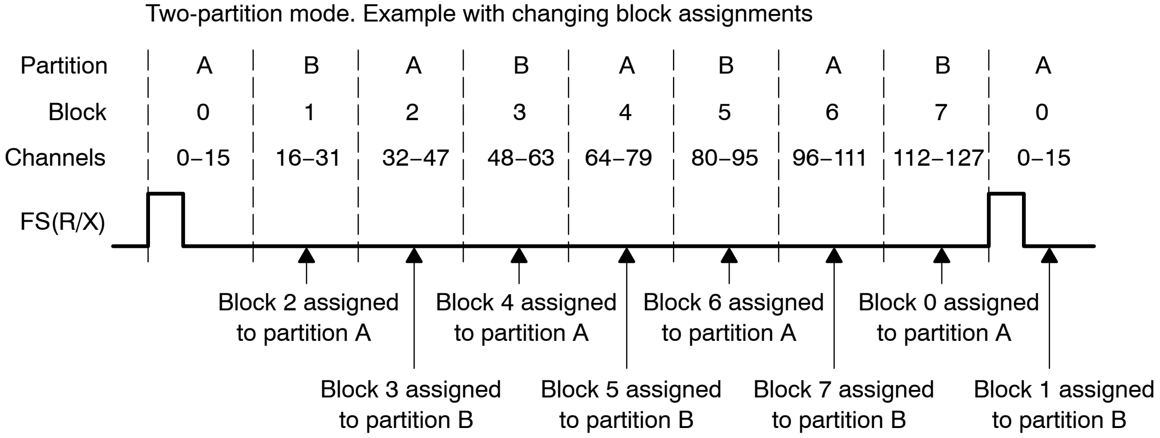
If you want to use more than 32 channels, you can change which channel blocks are assigned to partitions A and B during the course of a data transfer.

However, these changes must be carefully timed. While a partition is transferred, its associated block assignment bits cannot be modified and its associated channel enable register cannot be modified. For example, if block 3 is transferred and assigned to partition A, you can modify neither (R/X)PABLK to assign different channels to partition A nor (R/X)CERA to change the channel configuration for partition A. Several features of the McBSP help you time the reassignment:

- The block of channels currently involved in reception/transmission (the current block) is reflected in the RCBLK/XCBLK bits. Your program can poll these bits to determine which partition is active. When a partition is not active, it is safe to change its block assignment and channel configuration.
- At the end of every block (at the boundary of two partitions), an interrupt can be sent to the CPU. In response to the interrupt, the CPU can then check the RCBLK/XCBLK bits and update the inactive partition. For more information, see Section 21.4.3.8, *Using Interrupts Between Block Transfers*.

Figure 21−34 shows an example of reassigning channels throughout a data transfer. In response to a frame-sync pulse, the McBSP alternates between partitions A and B. Whenever partition B is active, the CPU changes the block assignment for partition A. Whenever partition A is active, the CPU changes the block assignment for partition B.

Figure 21−34. Reassigning Channel Blocks Throughout a McBSP Data Transfer


Two-partition mode. Example with changing block assignments
Partition
A
B
A
B
A
B
A
B
A
Block
0
1
2
3
4
5
6
7
0
Channels
0−15
16−31
32−47
48−63
64−79
80−95
96−111
112−127
0−15
FS(R/X)
Block 2 assigned to partition A
Block 4 assigned to partition A
Block 6 assigned to partition A
Block 0 assigned to partition A
Block 3 assigned to partition B
Block 5 assigned to partition B
Block 7 assigned to partition B
Block 1 assigned to partition B

21.4.3.5 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 eight-partition mode (RMCME = 1 for reception, 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-sync pulse, the receiver or transmitter begins with the channels in partition A and then continues with the other partitions, in order, until the complete frame is transferred. When the next frame-sync pulse occurs, the next frame is transferred, beginning with the channels in partition A.

In the eight-partition mode, the (R/X)PABLK and (R/X)PBBLK bits are ignored and the 16-channel blocks are assigned to the partitions as shown in Table 21−9 and Table 21−10. These assignments cannot be changed. The tables also show the registers used to control the channels in the partitions.

Table 21−9. Eight Partitions—Receive Channel Assignment and Control

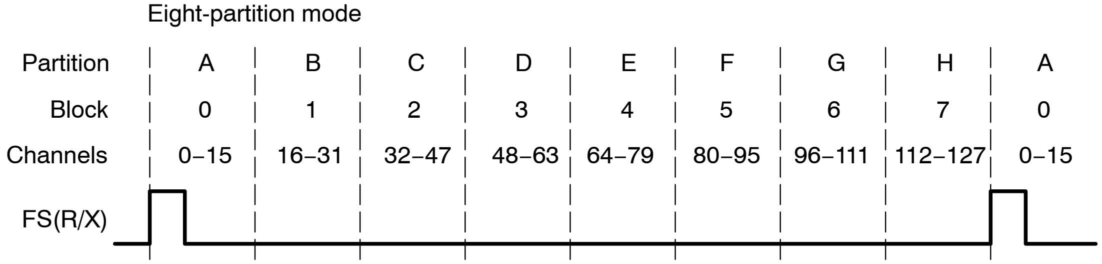
Receive Partition	Assigned Block of Receive Channels	Register Used for Channel Control
A	Block 0: Channels 0 through 15	MCBSP_RCERA_REG
B	Block 1: Channels 16 through 31	MCBSP_RCERB_REG
C	Block 2: Channels 32 through 47	MCBSP_RCERC_REG
D	Block 3: Channels 48 through 63	MCBSP_RCERD_REG
E	Block 4: Channels 64 through 79	MCBSP_RCERE_REG
F	Block 5: Channels 80 through 95	MCBSP_RCERF_REG
G	Block 6: Channels 96 through 111	MCBSP_RCERG_REG
H	Block 7: Channels 112 through 127	MCBSP_RCERH_REG

Table 21−10. Eight Partitions—Transmit Channel Assignment and Control

Transmit Partition	Assigned Block of Transmit Channels	Register Used for Channel Control
A	Block 0: Channels 0 through 15	MCBSP_XCERA_REG
B	Block 1: Channels 16 through 31	MCBSP_XCERB_REG
C	Block 2: Channels 32 through 47	MCBSP_XCERC_REG
D	Block 3: Channels 48 through 63	MCBSP_XCERD_REG
E	Block 4: Channels 64 through 79	MCBSP_XCERE_REG
F	Block 5: Channels 80 through 95	MCBSP_XCERF_REG
G	Block 6: Channels 96 through 111	MCBSP_XCERG_REG
H	Block 7: Channels 112 through 127	MCBSP_XCERH_REG

Figure 21−35 is an example of the McBSP using the eight-partition mode. In response to a frame-sync pulse, the McBSP begins a frame transfer with partition A and then activates B, C, D, E, F, G, and H to complete a 128-word frame.

Figure 21−35. McBSP Data Transfer in the Eight-Partition Mode



21.4.3.6 Receive Multichannel Selection Mode

The MCBSP_MCR1_REG[0] register 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 only channels enabled are those selected in the appropriate receive channel enable registers (MCBSP_RCER[A,H]_REG). The way channels are assigned to the MCBSP_RCER[A,H]_REG registers depends on the number of receive channel partitions (2 or 8), as defined by the MCBSP_MCR1_REG[9] register RMCME bit.
- ❏ If a receive channel is disabled, any bits received in that channel are passed only as far as the receive buffer register(s) (RBR(s)). The receiver does not copy the content of the RBR(s) to the MCBSP_DRR[1,2]_REG registers, and as a result, does not set the receiver ready bit (RRDY). Therefore, no DMA synchronization event (REVT) is generated and, if the receiver interrupt mode depends on RRDY (RINTM = 00b), no interrupt is generated.

As an example of how the McBSP behaves in the receive multichannel selection mode, suppose you enable only channels 0, 15, and 39 and that the frame length is 40. The McBSP:

1) Accepts bits shifted in from the McBSPn.DR pin in channel 0

2) Ignores bits received in channels 1–14

3) Accepts bits shifted in from the McBSPn.DR pin in channel 15

4) Ignores bits received in channels 16–38

5) Accepts bits shifted in from the McBSPn.DR pin in channel 39

21.4.3.7 Transmit Multichannel Selection Modes

The MCBSP_MCR2_REG[1:0] register XMCM bits determine whether all channels or only selected channels are enabled and unmasked for transmission. The McBSP has three transmit multichannel selection modes (XMCM = 01b, XMCM = 10b, and XMCM = 11b), which are described in Table 21−11.

Table 21−11. Selecting a Transmit Multichannel Selection Mode With the XMCM Bit Field

XMCM	Transmit Multichannel Selection Mode
00b	No transmit multichannel selection mode is on. All channels are enabled and unmasked. No channels can be disabled or masked.
01b	All channels are disabled unless they are selected in the appropriate transmit channel enable registers (MCBSP_XCER[A,H]_REG). If enabled, a channel in this mode is also unmasked. The MCBSP_MCR2_REG[9] register XMCME bit determines whether 32 channels or 128 channels are selectable in the MCBSP_XCER[A,H]_REG registers.

Table 21−11. Selecting a Transmit Multichannel Selection Mode With the XMCM Bit Field (Continued)

XMCM	Transmit Multichannel Selection Mode
10b	All channels are enabled, but they are masked unless they are selected in the appropriate transmit channel enable registers (MCBSP_XCER[A,H]_REG). The MCBSP_MCR2_REG[9] register XMCME bit determines whether 32 channels or 128 channels are selectable in the MCBSP_XCER[A,H]_REG registers.
11b	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 (MCBSP_RCER[A,H]_REG). Once enabled, they are masked unless they are also selected in the appropriate transmit channel enable registers (MCBSP_XCER[A,H]_REG). The MCBSP_MCR2_REG[9] register XMCME bit determines whether 32 channels or 128 channels are selectable in MCBSP_RCER[A,H]_REG registers and MCBSP_XCER[A,H]_REG registers.

As an example of how the McBSP behaves in a transmit multichannel selection mode, suppose that XMCM = 01b (all channels disabled unless individually enabled) and that you have enabled only channels 0, 15, and 39. Suppose also that the frame length is 40. The McBSP:

1) Shifts data to the McBSPn.DX pin in channel 0
2) Places the McBSPn.DX pin in the high impedance state in channels 1−14
3) Shifts data to the McBSPn.DX pin in channel 15
4) Places the McBSPn.DX pin in the high impedance state in channels 16−38
5) Shifts data to the McBSPn.DX pin in channel 39

Disabling/Enabling Versus Masking/Unmasking

For transmission, a channel can be:

- ☐ Enabled and unmasked (transmission can begin and can be completed)
- ☐ Enabled but masked (transmission can begin but cannot be completed)
- ☐ Disabled (transmission cannot occur)

The following definitions explain the channel control options:

Enabled Channel A channel that can begin transmission by passing data from the data transmit register(s) (MCBSP_DXR[1,2]_REG) to the transmit shift registers (XSR(s)).

Masked Channel A channel that cannot complete transmission. The McBSPn.DX pin is held in the high impedance state; data cannot be shifted out on the McBSPn.DX pin.
In systems where symmetric transmit and receive provides software benefits, this feature allows transmit channels to be disabled on a shared serial bus. A similar feature is not needed for reception because multiple receptions cannot cause serial bus contention.

Disabled Channel	A channel that is not enabled. A disabled channel is also masked. Because no DXR-to-XSR copy occurs, the MCBSP_SPCR2_REG[1] register XRDY bit is not set. Therefore, no DMA synchronization event (XEVT) is generated, and if the transmit interrupt mode depends on XRDY (XINTM = 00b in MCBSP_SPCR2_REG[5:4]), no interrupt is generated. The MCBSP_SPCR2_REG[2] register XEMPTY bit is not affected.
Unmasked Channel	A channel that is not masked. Data in the XSR(s) is shifted out on the McBSPn.DX pin.

Activity on McBSP Pins for Different Values of XMCM

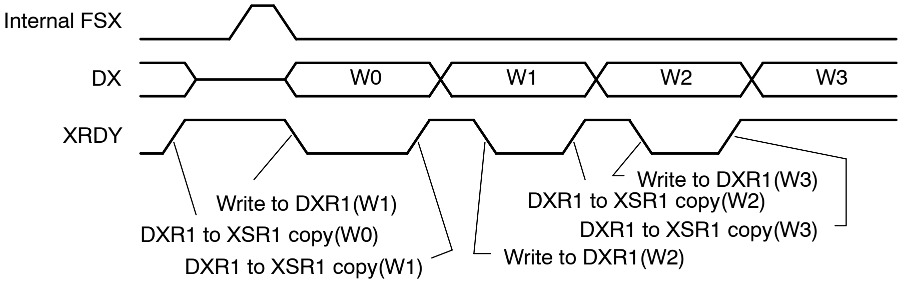
Figure 21−36 shows the activity on the McBSP pins for the various XMCM frame is configured as follows:

- XPHASE = 0: Single-phase frame (required for multichannel selection modes)
- XFRLEN1 = 0000011b: 4 words per frame. (i.e., 4 channels per frame)
- XWDLEN1 = 000b: 8 bits per word
- XMCME = 0: Two-partition mode (only partitions A and B used)

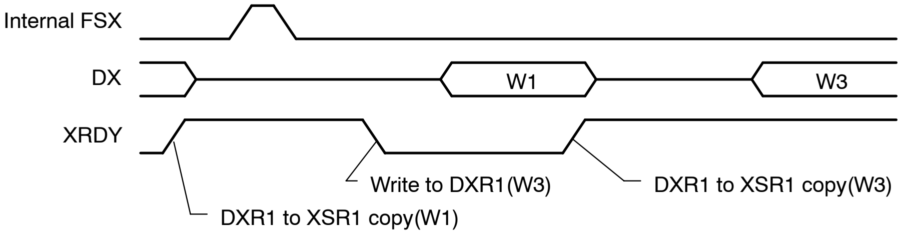
In the case where XMCM = 11b, transmission and reception are symmetric, which means the corresponding bits for the receiver (RPHASE, RFRLEN1, RWDLEN1, and RMCME) must have the same values as XPHASE, XFRLEN1, and XWDLEN1, respectively.

Figure 21−36. Activity on McBSP Pins for the Possible Values of XMCM

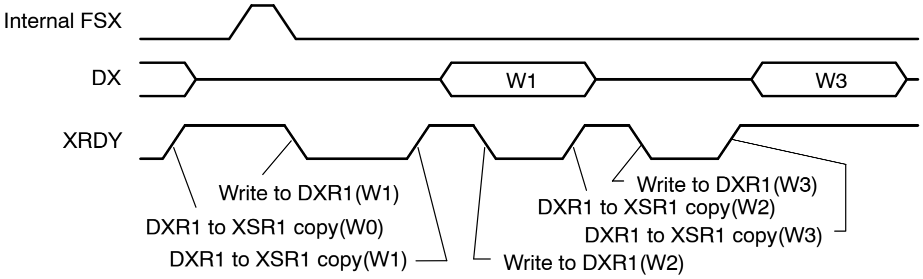
(a) XMCM = 00b: All channels enabled and unmasked



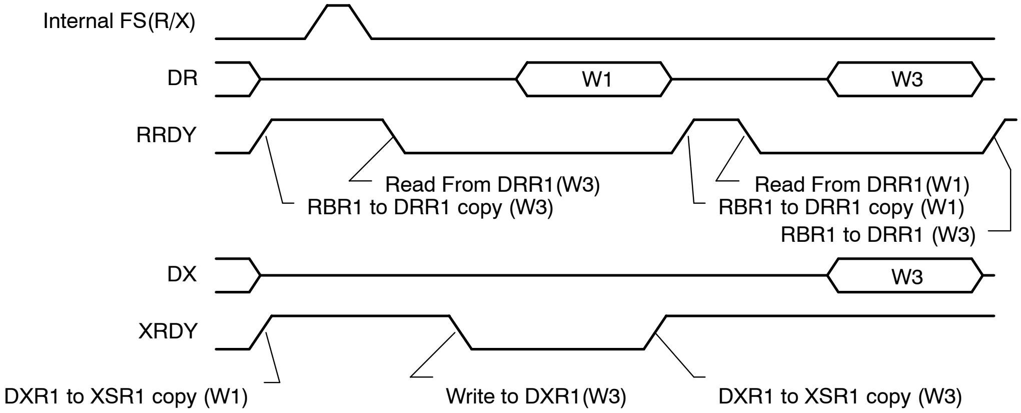
(b) XMCM = 01b, XPABLK = 00b, XCERA = 1010b: Only channels 1 and 3 enabled and unmasked



(c) XMCM = 10b, XPABLK = 00b, XCERA = 1010b: All channels enabled, only 1 and 3 unmasked



(d) XMCM = 11b, RPABLK = 00b, XPABLK = X, RCERA = 1010b, XCERA = 1000b:
Receive channels: 1 and 3 enabled; transmit channels: 1 and 3 enabled, but only 3 unmasked



21.4.3.8 Using Interrupts Between Block Transfers

When a multichannel selection mode is used, an interrupt request can be sent to the CPU at the end of every 16-channel block (at the boundary between partitions and at the end of the frame). In the receive multichannel selection mode, a receive interrupt (RINT) request is generated at the end of each block transfer if RINTM = 01b. In any of the transmit multichannel selection modes, a transmit interrupt (XINT) request is generated at the end of each block transfer if XINTM = 01b. When RINTM/XINTM = 01b, no interrupt is generated unless a multichannel selection mode is on.

These interrupt pulses are active high and last for two clock cycles.

This type of interrupt is especially helpful if you are using the two-partition mode and you want to know when you can assign a different block of channels to partition A or B.

21.4.4 SPI Operation Using the Clock Stop Mode

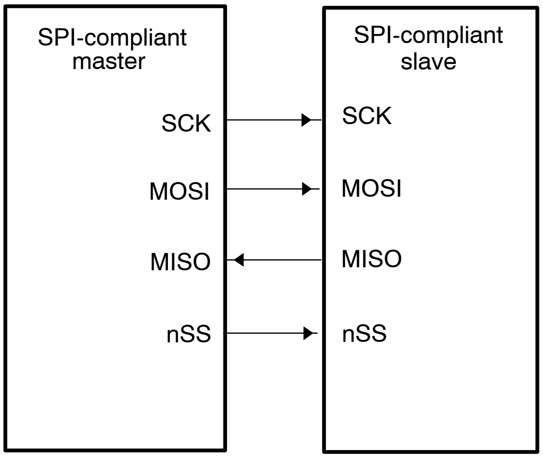
21.4.4.1 SPI Protocol

The SPI protocol is a master/slave configuration with one master device and one or more slave devices. The interface consists of the following four signals:

- ☐ Serial data input (also referred to as master in/slave out, or MISO)
- ☐ Serial data output (also referred to as master out/slave in, or MOSI)
- ☐ Serial clock (also referred to as SCK)
- ☐ Slave-enable signal (also referred to as SS signal)

A typical SPI interface with a single slave device is shown in Figure 21−37.

Figure 21−37. Typical SPI Interface



The master device controls the flow of communication by providing shift-clock and slave-enable signals. The slave-enable signal is an optional active-low signal that enables the serial data input and output of the slave device (device not sending out the clock).

In the absence of a dedicated slave-enable signal, communication between the master and slave is determined by the presence or absence of an active shift-clock. When the McBSP is operating in SPI master mode and the SS signal is not used by the slave SPI port, the slave device must remain enabled at all times, and multiple slaves cannot be used.

21.4.4.2 Clock Stop Mode

The clock stop mode of the McBSP provides compatibility with the SPI protocol. When the McBSP is configured in clock stop mode, the transmitter and receiver are internally synchronized so that the McBSP functions as an SPI master or slave device. The transmit clock signal (CLKX) corresponds to the SCK signal of the SPI protocol, while the FSX signal is used as the SS signal.

The receive clock (CLKR) and FSR signals are not used in the clock stop mode because these signals are internally connected to their transmit counterparts, CLKX and FSX.

21.4.4.3 Bits Used to Enable and Configure the Clock Stop Mode

The bits required to configure the McBSP as an SPI device are introduced in Table 21−12 and Table 21−13. The tables show how the various combinations of the CLKSTP bit and the polarity bits CLKXP and CLKRP create four possible clock stop mode configurations. The timing diagrams in the following section show the effects of CLKSTP, CLKXP, and CLKRP.

Table 21−12. Bits Used to Enable and Configure the Clock Stop Mode

Bit Field	Description
CLKSTP (MCBSP_SPCR1_REG[12:11])	Use these bits to enable the clock stop mode and to select one of two timing variations.
CLKXP (MCBSP_PCR_REG[1])	This bit determines the polarity of the CLKX signal.
CLKRP (MCBSP_PCR_REG[0])	This bit determines the polarity of the CLKR signal.
CLKXM (MCBSP_PCR_REG[9])	This bit determines whether CLKX is an input signal (McBSP as slave) or an output signal (McBSP as master).
XPHASE (MCBSP_XCR2_REG[15])	Use a single-phase transmit frame (XPHASE = 0).
RPHASE (MCBSP_RCR2_REG[15])	Use a single-phase receive frame (RPHASE = 0).
XFRLEN1 (MCBSP_XCR1_REG[14:8])	Use a transmit frame length of one serial word (XFRLEN1 = 0).
RFRLEN1 (MCBSP_RCR1_REG[14:8])	Use a receive frame length of one serial word (RFRLEN1 = 0).
XWDLEN1 (MCBSP_XCR1_REG[7:5])	The XWDLEN1 bits determine the transmit packet length. XWDLEN1 must be equal to RWDLEN1 because in the clock stop mode. The McBSP transmit and receive circuits are synchronized to a single clock.
RWDLEN1 (MCBSP_RCR1_REG[7:5])	The RWDLEN1 bits determine the receive packet length. RWDLEN1 must be equal to XWDLEN1 because in the clock stop mode. The McBSP transmit and receive circuits are synchronized to a single clock.

Table 21−13. Effects of CLKSTP, CLKXP and CLKRP on the Clock Scheme

Bit Settings	Clock Scheme

CLKSTP = 00b or 01b
CLKXP = 0 or 1
CLKRP = 0 or 1
	Clock stop mode disabled. Clock enabled for non-SPI mode.

CLKSTP = 10b
CLKXP = 0
CLKRP = 0
	Low inactive state without delay: The McBSP transmits data on the rising edge of CLKX and receives data on the falling edge of CLKR.

CLKSTP = 11b
CLKXP = 0
CLKRP = 1
	Low inactive state with delay: The McBSP transmits data one-half cycle ahead of the rising edge of CLKX and receives data on the rising edge of CLKR.

CLKSTP = 10b
CLKXP = 1
CLKRP = 0
	High inactive state without delay: The McBSP transmits data on the falling edge of CLKX and receives data on the rising edge of CLKR.

CLKSTP = 11b
CLKXP = 1
CLKRP = 1
	High inactive state with delay: The McBSP transmits data one-half cycle ahead of the falling edge of CLKX and receives data on the falling edge of CLKR.

21.4.4.4 Clock Stop Mode Timing Diagrams

The timing diagrams for the four possible clock stop mode configurations are shown in Figure 21−38 through Figure 21−41. Notice that the FSG used in clock stop mode is active throughout the entire transmission as a slave-enable signal. Although the timing diagrams show 8-bit transfers, the packet length can be set to 8, 12, 16, 20, 24, or 32 bits per packet. The receive packet length is selected with the MCBSP_RCR1_REG[7:5] register RWDLEN1 bits, and the transmit packet length is selected with the MCBSP_XCR1_REG[7:5] register XWDLEN1 bits. For clock stop mode, the values of RWDLEN1 and XWDLEN1 must be the same because the McBSP transmit and receive circuits are synchronized to a single clock.

Note:

Even if multiple words are consecutively transferred, the CLKX signal is always stopped, and the FSX signal returns to the inactive state after a packet transfer. When consecutive packet transfers are performed, this leads to a minimum idle time of two bit periods between each packet transfer.

Figure 21−38. SPI Transfer With CLKSTP = 10b, CLKXP = 0 and CLKRP = 0

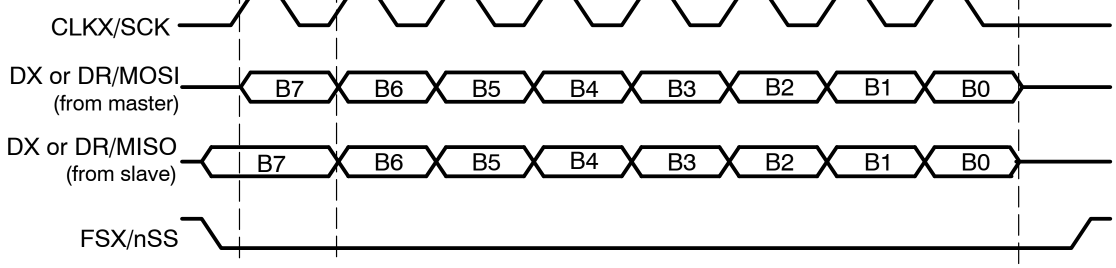


Figure 21−39. SPI Transfer With CLKSTP = 11b, CLKXP = 0 and CLKRP = 1

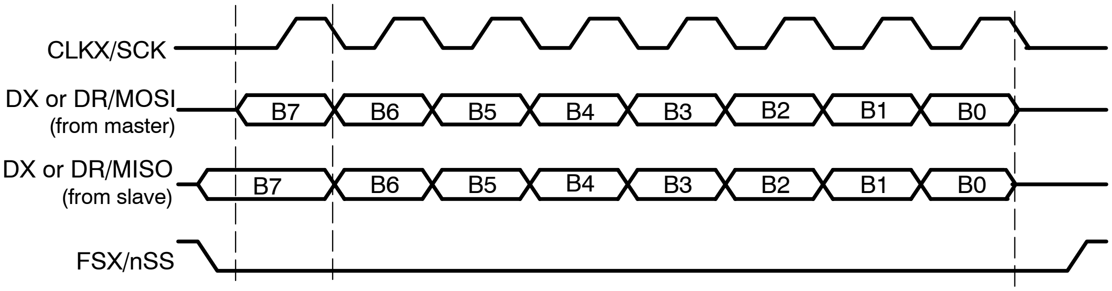


Figure 21−40. SPI Transfer With CLKSTP = 10b, CLKXP = 1 and CLKRP = 0

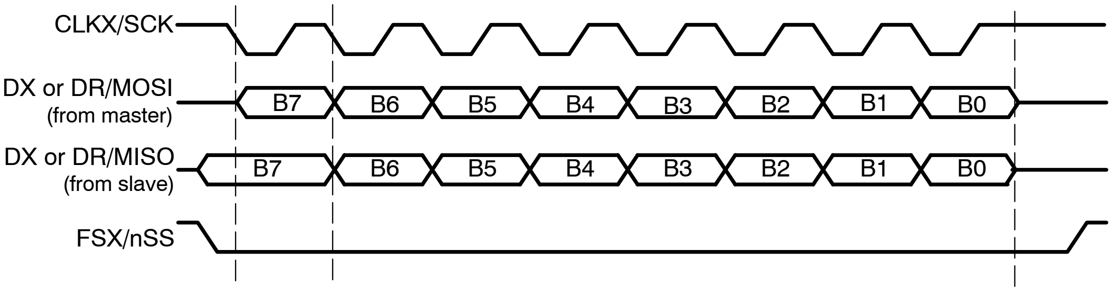
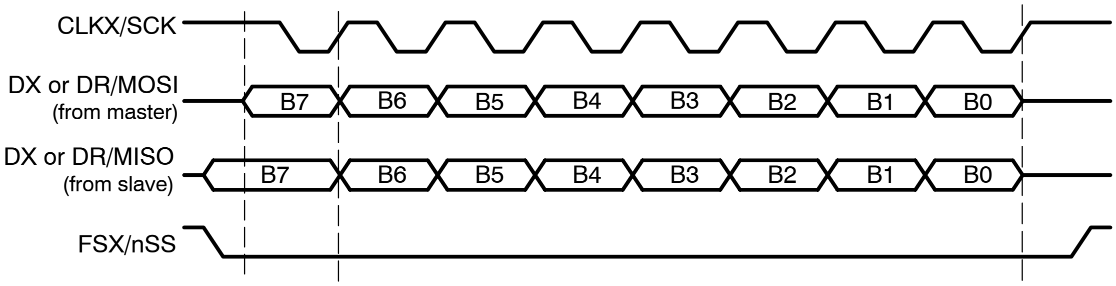


Figure 21−41. SPI Transfer With CLKSTP = 11b, CLKXP = 1 and CLKRP = 1



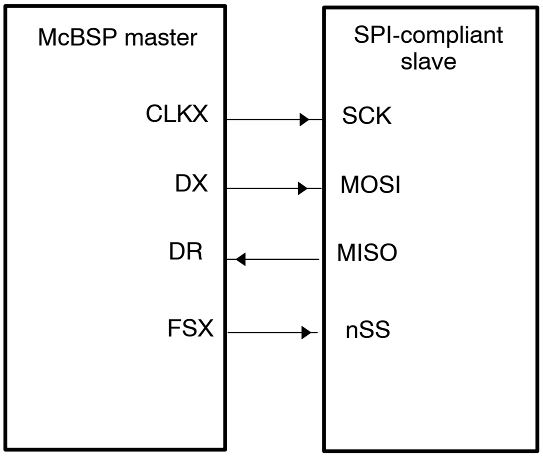
Notes:

1) If the McBSP is the SPI master (CLKXM = 1), MOSI = McBSPn.DX. If the McBSP is the SPI slave (CLKMX = 0), MOSI = McBSPn.DR.
2) If the McBSP is the SPI master (CLKXM = 1), MISO = McBSPn.DR. If the McBSP is the SPI slave (CLKXM = 0), MISO = McBSPn.DX.

21.4.4.5 McBSP as the SPI Master

An SPI interface with the McBSP used as the master is shown in Figure 21−42. When the McBSP is configured as a master, the transmit output signal (McBSPn.DX) is used as the MOSI signal of the SPI protocol and the receive input signal (McBSPn.DR) is used as the MISO signal.

Figure 21−42. McBSP as the SPI Master



The register bit values required to configure the McBSP as a master are listed in Table 21−14. Following the table are more details about the configuration requirements.

Table 21−14. Bits Values Required to Configure the McBSP as an SPI Master

Required Bit Setting	**Description**
CLKSTP = 10b or 11b	The clock stop mode (without or with a clock delay) is selected.
CLKXP = 0 or 1	The polarity of CLKX as seen on the McBSPn.CLKX pin is positive (CLKXP = 0) or negative (CLKXP = 1).
CLKRP = 0 or 1	The polarity of CLKR as seen on the McBSPn.CLKR pin is positive (CLKRP = 0) or negative (CLKRP = 1).
CLKXM = 1	The McBSPn.CLKX pin is an output pin driven by the internal SRG. Because CLKSTP is equal to 10b or 11b, CLKR is driven internally by CLKX.

SCLKME = 0
CLKSM = 1
	The clock (CLKG) generated by the SRG is derived from the CLK input from the McBSP module.

CLKGDV is a value from 0 to 255	CLKGDV defines the divide down value for CLKG.
FSXM = 1	The McBSPn.FSX pin is an output pin driven according to the FSGM bit.
FSGM = 0	The transmitter drives a frame-sync pulse on the McBSPn.FSX pin every time data is transferred from MCBSP_DXR1_REG to XSR1.
FSXP = 1	The McBSPn.FSX pin is active low.

XDATDLY = 01b
RDATDLY = 01b
	This setting provides the correct setup time on the FSX signal.

When the McBSP functions as the SPI master, it controls the transmission of data by producing the serial clock signal. The clock signal on the McBSPn.CLKX pin is enabled only during packet transfers. When packets are not transferred, the McBSPn.CLKX pin remains high or low depending on the polarity used.

For SPI master operation, the McBSPn.CLKX pin must be configured as an output. The SRG is then used to derive the CLKX signal from the CORE_L4_ICLK clock. The clock stop mode internally connects the McBSPn.CLKX pin to the CLKR signal so that no external signal connection is required on the McBSPn.CLKR pin and both the transmit and receive circuits are clocked by the master clock (CLKX).

The data delay parameters of the McBSP (XDATDLY and RDATDLY) must be set to 1 for proper SPI master operation. A data delay value of 0 or 2 is undefined in the clock stop mode.

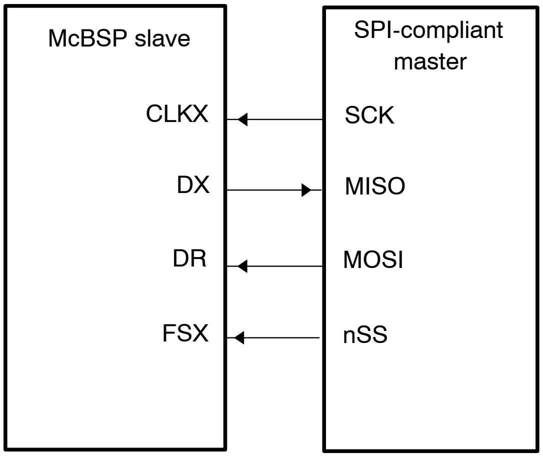
The McBSP can also provide a slave-enable signal (SS_) on the McBSPn.FSX pin. If a slave-enable signal is required, the McBSPn.FSX pin must be configured as an output and the transmitter must be configured so that a frame-sync pulse is generated automatically each time a packet is transmitted (FSGM = 0). The polarity of the McBSPn.FSX pin is programmable high or low; however, in most cases the pin must be configured active low.

When the McBSP is configured as described for SPI-master operation, the bit fields for frame-sync pulse width (FWID) and frame-sync period (FPER) are overridden, and custom frame-sync waveforms are not allowed. To see the resulting waveform produced on the McBSPn.FSX pin, see the timing diagrams in Section 21.4.4.4, *Clock Stop Mode Timing Diagrams*. The signal becomes active before the first bit of a packet transfer, and remains active until the last bit of the packet is transferred. After the packet transfer is complete, the FSX signal returns to the inactive state.

21.4.4.6 McBSP as an SPI Slave

An SPI interface with the McBSP used as a slave is shown in Figure 21−43. When the McBSP is configured as a slave, McBSPn.DX is used as the MISO signal and DR is used as the MOSI signal.

Figure 21−43. McBSP as an SPI Slave



The register bit values required to configure the McBSP as a slave are listed in Table 21−15. Following the table are more details about configuration requirements.

Table 21−15. Bits Values Required to Configure the McBSP as an SPI Slave

Required Bit Setting	Description
CLKSTP = 10b or 11b	The clock stop mode (without or with a clock delay) is selected.
CLKXP = 0 or 1	The polarity of CLKX as seen on the McBSPn.CLKX pin is positive (CLKXP = 0) or negative (CLKXP = 1).
CLKRP = 0 or 1	The polarity of CLKR as seen on the McBSPn.CLKR pin is positive (CLKRP = 0) or negative (CLKRP = 1).
CLKXM = 0	The McBSPn.CLKX pin is an input pin, so that it can be driven by the SPI master. Because CLKSTP = 10b or 11b, CLKR is driven internally by CLKX.

SCLKME = 0
CLKSM = 1
	The clock generated by the SRG (CLKG) is derived from the CLK input from the McBSP module (the SRG is used to synchronize the McBSP logic with the externally-generated master clock).

CLKGDV = 1	The SRG divides the McBSPn_FCLK clock by 2 before generating CLKG.
FSXM = 0	The McBSPn.FSX pin is an input pin, so that it can be driven by the SPI master.
FSXP = 1	The McBSPn.FSX pin is active low.

XDATDLY = 00b
RDATDLY = 00b
	These bits must be 0s for SPI slave operation.

When the McBSP is used as an SPI slave, the master clock and slave-enable signals are generated externally by a master device. Accordingly, the McBSPn.CLKX and McBSPn.FSX pins must be configured as inputs. The McBSPn.CLKX pin is internally connected to the CLKR signal so that both the transmit and receive circuits of the McBSP are clocked by the external master clock. The McBSPn.FSX pin is also internally connected to the FSR signal, and no external signal connections are required on the McBSPn.CLKR and McBSPn.FSR pins.

Although the CLKX signal is generated externally by the master and is asynchronous to the McBSP, the SRG of the McBSP must be enabled for proper SPI slave operation. The SRG must be programmed to its maximum rate of half the CPU clock rate. The internal sample rate clock is then used to synchronize the McBSP logic to the external master clock and slave-enable signals.

The McBSP requires an active edge of the slave-enable signal on the FSX input for each transfer. This means that the master device must assert the slave-enable signal at the beginning of each transfer, and deassert the signal after the completion of each packet transfer; the slave-enable signal cannot remain active between transfers.

The data delay parameters of the McBSP must be set to 0 for proper SPI slave operation. A value of 1 or 2 is undefined in the clock stop mode.

21.5 McBSP Programming Model

21.5.1 McBSP Initialization Procedure

The serial port initialization procedure is as follows:

Step 1: Set MCBSP_SPCR1_REG[0] register RRST bit, MCBSP_SPCR2_REG[7] register FRST bit and MCBSP_SPCR2_REG[0] register XRST bit to 0. If coming out of an OMAP2420 reset, this step is not required.

Step 2: While the serial port is in the reset state, program only the McBSP configuration registers (not the data registers) as required.

Step 3: Wait for two clock cycles. This ensures proper internal synchronization.

Step 4: Set up data acquisition as required (such as writing to MCBSP_DXR1_REG or MCBSP_DXR2_REG registers).

Step 5: Set MCBSP_SPCR1_REG[0] register RRST bit and MCBSP_SPCR2_REG[0] register XRST bit to 1 to enable the serial port. Make sure that as you set these reset bits you do not modify any of the other bits in the MCBSP_SPCR1_REG and MCBSP_SPCR2_REG registers. Otherwise, you would change the configuration set in step 2.

Step 6: Set the MCBSP_SPCR2_REG[7] register FRST bit to 1 if internally generated frame-sync is required.

Step 7: Wait for two clock cycles for the receiver and transmitter to become active.

Alternatively, on write (steps 1 or 5), the transmitter and receiver can be placed in or taken out of reset by modifying the desired bit.

The above procedure for reset/initialization can be applied in general when the receiver or transmitter must be reset during its normal operation, and also when the SRG is not used for either operation.

Note:

1. Change muxes to deselect McBSP.CLKR/CLKX for the pins. For example, GPIO in input or protected mode.
2. Feed CLKR/CLKX from internal clock generator and reset receiver/transmitter.
3. Change muxes to select McBSP.CLKR/CLKX.

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 (MCBSP_SPCR1_REG, MCBSP_PCR_REG, MCBSP_RCR1_REG, MCBSP_RCR2_REG, MCBSP_XCR1_REG, MCBSP_XCR2_REG, MCBSP_SRGR1_REG, and MCBSP_SRGR2_REG) must only be modified when the affected portion of the serial port is in its reset state.

In most cases, the data transmit registers (MCBSP_DXR1_REG and MCBSP_DXR2_REG) must be loaded by the CPU or the DMA controller only when the transmitter is enabled (XRST = 1). An exception to this rule is when these registers are used for companding internal data.

The bits of the channel control registers (MCBSP_MCR1_REG, MCBSP_MCR2_REG, MCBSP_RCER{A−H}_REG, and MCBSP_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 MCBSP_SPCR2_REG[6] register GRST bit to 0.

21.5.2 Reset and Initialization Procedure for the SRG

To reset and initialize the SRG:

1) Place the McBSP/SRG in reset.

   During an OMAP2420 reset, the SRG, the receiver, and the transmitter reset bits (GRST, RRST, and XRST) are automatically forced to 0. Otherwise, during normal operation, the SRG can be reset by making GRST = 0 in MCBSP_SPCR2_REG[6] register, provided that CLKG and/or FSG is not used by any portion of the McBSP. Depending on the particular system, it may be desirable to reset the receiver (RRST = 0 in MCBSP_SPCR1_REG[0]) and reset the transmitter (MCBSP_SPCR2_REG[0] register XRST bit = 0).

   If GRST = 0 because of an OMAP2420 reset, CLKG is driven by the FUNC_96M_CLK clock divided by 2, and FSG is driven inactive low. If GRST = 0 because of program code, CLKG and FSG are driven low (inactive).

2) Program the registers that affect the SRG.

   Program the SRG registers (MCBSP_SPCR1_REG and MCBSP_SPCR2_REG) as required for your application. If necessary, other control registers can be loaded with desired values provided the respective portion of the McBSP (the receiver or transmitter) is in reset.

   After the SRG registers are programmed, wait 2 CLKSRG cycles. This ensures proper synchronization internally.

3) Enable the SRG (take it out of reset).

In MCBSP_SPCR2_REG[6], set GRST bit to 1 to enable the SRG.

After the SRG is enabled, wait 2 CLKG cycles for the SRG logic to stabilize.

On the next rising edge of CLKSRG, CLKG transitions to 1 and starts clocking with a frequency equal to (input clock frequency)/(CLKGDV + 1) where the input clock is selected with the SCLKME bit of MCBSP_PCR_REG[7] register and the MCBSP_SRGR2_REG[13] register CLKSM bit in one of the following configurations (see Table 21−16).

Table 21−16. Input Clock Selection for SRG

SCLKME Bit	CLKSM bit	Input Clock for SRG
0	0	Signal on CLKS pin of McBSP module
0	1	Signal on CLK pin of McBSP module
1	0	Signal on McBSPn.CLKR pin
1	1	Signal on McBSPn.CLKX pin

1) If necessary, enable the receiver and/or the transmitter.

   If necessary, remove the receiver and/or transmitter from reset by setting RRST and/or XRST = 1.

2) If necessary, enable the frame-sync logic of the SRG.

   After the required data acquisition setup is done (MCBSP_DXR[1,2]_REG is loaded with data), set GRST = 1 in MCBSP_SPCR2_REG[6] if an internally generated frame-sync pulse is required. FSG is generated with an active-high edge after the programmed number of CLKG clocks (FPER + 1) have elapsed.

21.5.3 Receiver Configuration

To configure the McBSP receiver, perform the following procedure:

1) Place the McBSP/receiver in reset (see Section 21.5.3.2).
2) Program the McBSP registers for the desired receiver operation (see Section 21.5.3.1).
3) Take the receiver out of reset (see Section 21.5.3.2).

21.5.3.1 Programming the McBSP Registers for the Desired Receiver Configuration

The following is a list of important tasks to be performed when you are configuring the McBSP receiver. Each task corresponds to one or more McBSP register bit fields.

- Global behavior:
  - Set the receiver pins to operate as McBSP pins.
  - Enable/disable the digital loopback mode.
  - Enable/disable the clock stop mode.

  - Enable/disable the receive multichannel selection mode.

- Data behavior:
  - Choose 1 or 2 phases for the receive frame.
  - Set the receive word length(s).
  - Set the receive frame length.
  - Enable/disable the FSR ignore function.
  - Set the receive companding mode.
  - Set the receive data delay.
  - Set the receive sign-extension and justification mode.
  - Set the receive interrupt mode.

- Frame-sync behavior:
  - Set the FSR mode.
  - Set the FSR polarity.
  - Set the SRG (SRG) frame-sync period and pulse width.

- Clock behavior:
  - Set the receive clock mode.
  - Set the receive clock polarity.
  - Set the SRG clock divide-down value.
  - Set the SRG clock synchronization mode.
  - Set the SRG clock mode (choose an input clock).
  - Set the SRG input clock polarity.

21.5.3.2 Resetting and Enabling 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 two ways:

Step 1: An OMAP2420 reset places the receiver, transmitter, and SRG in reset. When the device reset is removed, GRST, FRST, RRST, and XRST bits = 0, keeping the entire serial port in the reset state.

Step 2: The serial port transmitter and receiver can be reset directly using the RRST and XRST bits in the MCBSP_SPCR1_REG register. The SRG can be reset directly using the GRST bit in the MCBSP_SPCR2_REG register.

21.5.3.3 Set the Receiver Pins to Operate as McBSP Pins

The RIOEN bit (MCBSP_PCR_REG[12]) determines whether the receiver pins are McBSP pins or general-purpose I/O pins.

See Section 21.6, *McBSP Registers*, for more information.

21.5.3.4 Enable/Disable the Digital Loopback Mode

The DLB bit (MCBSP_SPCR1_REG[15]) 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. This mode allows testing of serial port code; the McBSP receives the data it transmits.

21.5.3.5 Enable/Disable the Clock Stop Mode

The CLKSTP bits (MCBSP_SPCR1_REG[12:11]) determine whether the clock stop mode is on or off.

The clock stop mode supports the SPI master-slave protocol. If you do not plan to use the SPI protocol, you can clear CLKSTP to disable the clock stop mode.

In the clock stop mode, the clock stops at the end of each data transfer. At the beginning of each data transfer, the clock starts immediately (CLKSTP = 10b) or after a half-cycle delay (CLKSTP = 11b).

The CLKXP bit (MCBSP_PCR_REG[1]) determines whether the starting edge of the clock on the McBSPn.CLKX pin is rising or falling.

The CLKRP bit (MCBSP_PCR_REG[0]) determines whether receive data is sampled on the rising or falling edge of the clock shown on the McBSPn.CLKR pin.

Table 21−17 summarizes the impact of CLKSTP, CLKXP, and CLKRP on serial port operation. In the clock stop mode, the receive clock is tied internally to the transmit clock, and the FSR signal is tied internally to the FSX signal.

Table 21−17. Effects of CLKSTP, CLKXP, and CLKRP on the Clock Scheme

Bit Settings	**Clock Scheme**

CLKSTP = 00b or 01b
CLKXP = 0 or 1
CLKRP = 0 or 1
	Clock stop mode disabled. Clock enabled for non-SPI mode.

CLKSTP = 10b
CLKXP = 0
CLKRP = 0
	Low inactive state without delay. The McBSP transmits data on the rising edge of CLKX and receives data on the falling edge of CLKR.

CLKSTP = 11b
CLKXP = 0
CLKRP = 1
	Low inactive state with delay. The McBSP transmits data one half cycle ahead of the rising edge of CLKX and receives data on the rising edge of CLKR.

CLKSTP = 10b
CLKXP = 1
CLKRP = 0
	High inactive state without delay. The McBSP transmits data on the falling edge of CLKX and receives data on the rising edge of CLKR.

CLKSTP = 11b
CLKXP = 1
CLKRP = 1
	High inactive state with delay. The McBSP transmits data one half cycle ahead of the falling edge of CLKX and receives data on the falling edge of CLKR.

21.5.3.6 Enable/Disable the Receive Multichannel Selection Mode

The RMCM bit (MCBSP_MCR1_REG[0]) determines whether the receive multichannel selection mode is on or off.

For more information, see Section 21.4.3.6, *Receive Multichannel Selection Mode*.

21.5.3.7 Choose 1 or 2 Phases for the Receive Frame

The RPHASE bit (MCBSP_RCR2_REG[15]) determines whether the receive data frame has one or two phases.

21.5.3.8 Set the Receive Word Length(s)

The RWDLEN1 (MCBSP_RCR1_REG[7:5]) and RWDLEN2 (MCBSP_RCR2_REG[7:5]) bit fields determine how many bits are in each serial word in phase 1 and in phase 2, respectively, of the receive data frame.

Each frame can have one or two phases, depending on the value that you load into the RPHASE bit. 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 determines the length of the serial words in phase 1 of the frame, and RWDLEN2 determines the word length in phase 2 of the frame.

21.5.3.9 Set the Receive Frame Length

The RFRLEN1 (MCBSP_RCR1_REG[14:8]) and RFRLEN2 (MCBSP_RCR2_REG[14:8]) bit fields determine how many serial words are in phase 1 and in phase 2, respectively, of the receive data frame.

The receive frame length is the number of serial words in the receive frame. Each frame can have one or two phases, depending on the value loaded into the RPHASE bit.

If a single-phase frame is selected (RPHASE = 0), the frame length is equal to the length of phase 1. If a dual-phase frame is selected (RPHASE = 1), the frame length is the length of phase 1 plus the length of phase 2.

The 7-bit RFRLEN fields allow up to 128 words per phase. See Table 21−18 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-sync pulse.

Program the RFRLEN fields with [*w minus 1*], where *w* represents the number of words per phase. For example, if you want a phase length of 128 words in phase 1, load 127 into RFRLEN1.

Table 21−18. How to Calculate the Length of the Receive Frame

RPHASE	RFRLEN1	RFRLEN2	Frame Length
0	0 = RFRLEN1 = 127	Don’t care	(RFRLEN1 + 1) words
1	0 = RFRLEN1 = 127	0 = RFRLEN2 = 127	(RFRLEN1 + 1) + (RFRLEN2 + 1) words

21.5.3.10 Enable/Disable the Receive Frame-Sync Ignore Function

The RFIG bit (MCBSP_RCR2_REG[2]) controls the FSR ignore function.

If a frame-sync pulse starts the transfer of a new frame before the current frame is fully received, this pulse is treated as an unexpected frame-sync pulse.

When RFIG = 1, reception continues, ignoring the unexpected frame-sync pulses.

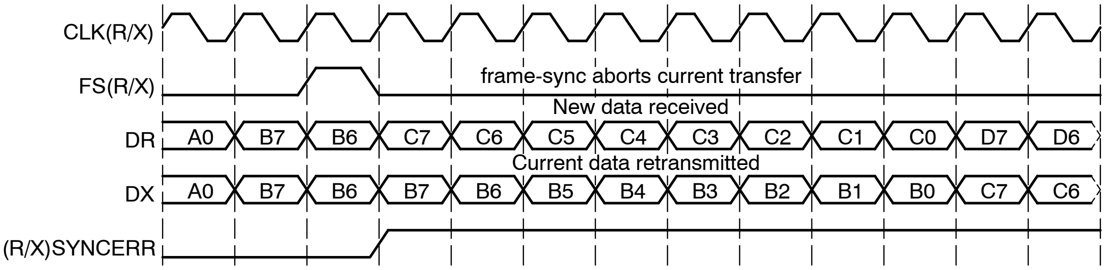
When RFIG = 0, an unexpected FSR pulse causes the McBSP to discard the contents of RSR[1,2] in favor of the new incoming data. Therefore, if RFIG = 0 and an unexpected frame-sync pulse occurs, the serial port:

- Aborts the current data transfer
- Sets MCBSP_SPCR1_REG[3] register RSYNCERR bit to 1
- Begins the transfer of a new data word

For more details about the frame-sync error condition, see Section 21.4.2.3*, Unexpected FSR Pulse.*

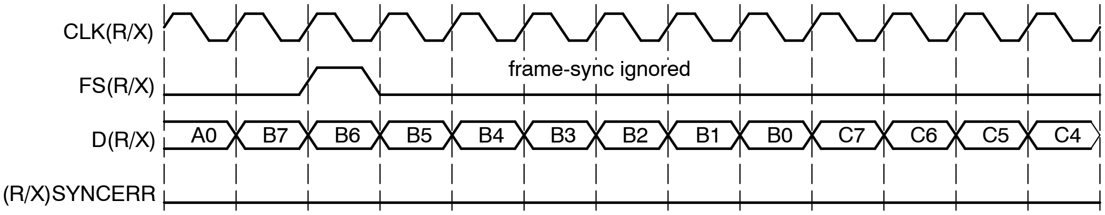
Figure 21−44 shows an example in which word B is interrupted by an unexpected frame-sync pulse when (R/X)FIG = 0. for reception, the reception of B is aborted (B is lost), and a new data word (C in this example) is received after the appropriate data delay. This condition is a receive synchronization error, which sets the RSYNCERR bit.

Figure 21−44. Unexpected Frame-Sync Pulse With (R/X)FIG = 0



In contrast to Figure 21−44, Figure 21−45 shows McBSP operation when unexpected FSG signals are ignored (when (R/X)FIG = 1). Here, the transfer of word B is not affected by an unexpected pulse.

Figure 21−45. Unexpected Frame-Sync Pulse With (R/X)FIG = 1



21.5.3.11 Set the Receive Companding Mode

The RCOMPAND bit field (MCBSP_RCR2_REG[4:3]) determines whether companding or another data transfer option is chosen for McBSP reception.

For more information about companding, see Section 21.2, McBSP Environment.

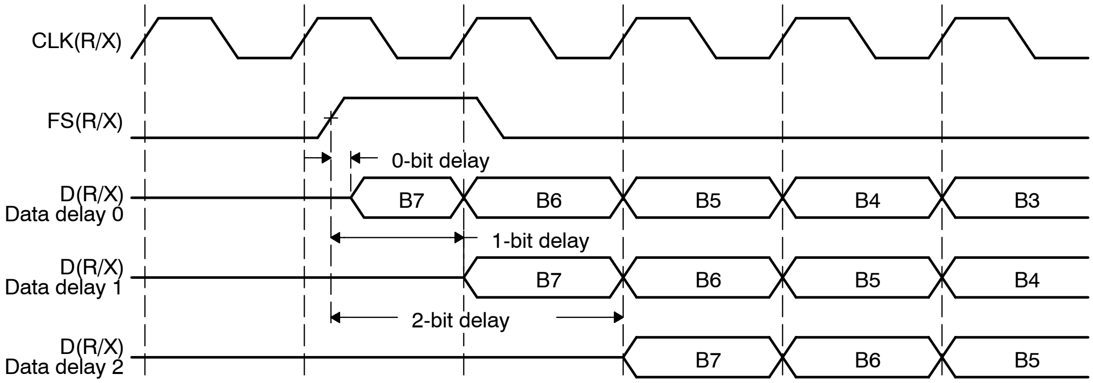
21.5.3.12 Set the Receive Data Delay

The RDATDLY bit field (MCBSP_RCR2_REG[1:0]) 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-sync is found to be active. The beginning of actual data reception or transmission with respect to the start of the frame can be delayed if required. This delay is called data delay.

RDATDLY specifies the data delay for reception. The range of programmable data delay is zero to two bit-clocks (RDATDLY = 00b–10b), as shown in Figure 21−46. 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-sync pulse.

Figure 21−46. Range of Programmable Data Delay



0-Bit Data Delay

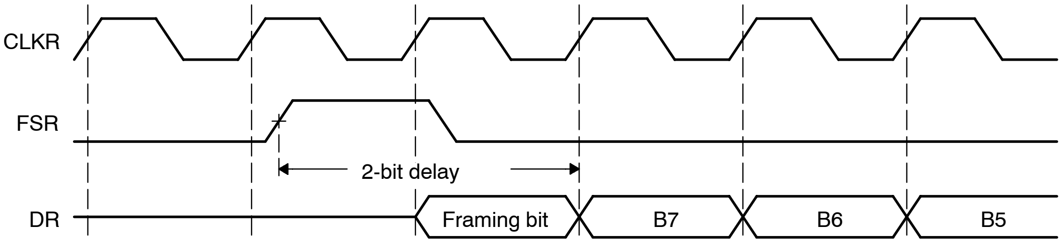
Normally, a frame-sync pulse is detected or sampled with respect to an edge of internal serial clock CLK(R/X). Thus, on the following cycle or later (depending on the data delay value), data can be received or transmitted. However, for 0-bit data delay, the data must be ready for reception and/or transmission 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-sync. Therefore, the first data bit is assumed to be present in XSR1, and thus on McBSPn.DX. The transmitter then asynchronously detects the FSX signal going active high and starts driving the first bit to be transmitted on the McBSPn.DX pin.

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 21−47. In this figure, the data transferred is an 8-bit value with bits labeled B7, B6, B5, and so on.

Figure 21−47. 2−Bit Data Delay Used to Skip a Framing Bit



21.5.3.13 Set the Receive Sign-Extension and Justification Mode

The RJUST bit field (MCBSP_SPCR1_REG[14:13]) determines whether data received by the McBSP is sign-extended or not, and how it is justified.

RJUST bit field selects whether data in RBR[1,2] is right- or left-justified (with respect to the MSB) in MCBSP_DRR[1,2]_REG registers and whether unused bits in MCBSP_DRR[1,2]_REG are filled with zeros or with sign bits.

Table 21−19 and Table 21−20 show the effects of various RJUST values. Table 21−19 shows the effect on an example 12-bit receive-data value 0xABC. Table 21−20 shows the effect on an example 20-bit receive-data value 0xABCDE.

Table 21−19. Example: Use of RJUST Bit Field With 12-Bit Data Value 0xABC

RJUST	Justification	Extension	Value in DRR2	Value in DRR1
00b	Right	Zero fill MSBs	0x0000	0x0ABC
01b	Right	Sign extend data into MSBs	0xFFFF	0xFABC
10b	Left	Zero fill LSBs	0x0000	0xABC0
11b	Reserved	Reserved	Reserved	Reserved

Table 21−20. Example: Use of RJUST Bit Field With 20-Bit Data Value 0xABCDE

RJUST	Justification	Extension	Value in DRR2	Value in DRR1
00b	Right	Zero fill MSBs	0x000A	0xBCDE
01b	Right	Sign extend data into MSBs	0xFFFA	0xBCDE
10b	Left	Zero fill LSBs	0xABCD	0xE000
11b	Reserved	Reserved	Reserved	Reserved

21.5.3.14 Set the Receive Interrupt Mode

The RINTM bit field (MCBSP_SPCR1_REG[5:4]) determines which event generates a receive interrupt request to the CPU.

The receive interrupt (RINT) informs the CPU of changes to the serial port status. Four options exist for configuring this interrupt:

- ☐ RINTM = 00: RINT is generated when the RRDY bit (MCBSP_SPCR1_REG[1]) changes from 0 to 1. Interrupt on every serial word by tracking the RRDY bit in the MCBSP_SPCR1_REG register. Regardless of the value of RINTM, RRDY can be read to detect the RRDY = 1 condition. The interruption is deasserted when the read of DDR1 register is performed.
- ☐ RINTM = 01: RINT is generated by an end-of-block or end-of-frame condition in the receive multichannel selection mode. In the multichannel selection mode, interrupt after every 16-channel block boundary is crossed within a frame and at the end of the frame. In any other serial transfer case, this setting is not applicable and, therefore, no interrupts are generated. The software must deassert the interruption at the end of the interrupt routine by performing a read access to RINTN register.
- ☐ RINTM = 10: RINT is generated by a new FSR pulse. Interrupt on detection of FSR pulses. This generates an interrupt even when the receiver is in its reset state. This is done by synchronizing the incoming frame-sync pulse to the CPU clock and sending it to the CPU through RINT. The software must deassert the interruption at the end of the interrupt routine by performing a read access to RINTN register.
- ☐ RINTM = 11: RINT is generated when RSYNCERR is set. Interrupt on frame-sync error. Regardless of the value of RINTM, RSYNCERR can be read to detect this condition. For information on using RSYNCERR, see Section 21.4.2.3, *Unexpected FSR Pulse*. The interruption is deasserted when a write of 0 to McBSP_SPCR1_REG[3] RSYNCERR bit is performed.

21.5.3.15 Set the Receive Frame-Sync Mode

The FSRM bit (MCBSP_PCR_REG[10]), GSYNC bit (MCBSP_SRGR2_REG[15]), DLB bit (MCBSP_SPCR1_REG[15]), and CLKSTP bit field (MCBSP_SPCR1_REG[12:11]) are used to determine the source for FSR and the function of the McBSPn.FSR pin.

Table 21−21 shows how you can select various sources to provide the FSR signal and the effect on the McBSPn.FSR pin. The polarity of the signal on the McBSPn.FSR pin is determined by the FSRP bit.

In digital loopback mode (DLB = 1), the FSX signal is used as the FSR signal.

Also in the clock stop mode, the internal CLKR and FSR signals are internally connected to their transmit counterparts, CLKX and FSX.

Table 21−21. DLB, FSRM, and GSYNC Effects on FSG and McBSPn.FSR Pin

DLB	FSRM	GSYNC	Source of FSR	McBSPn.FSR Pin Status
0	0	0 or 1	An external FSG signal enters the McBSP through the McBSPn.FSR pin. The signal is then inverted as determined by FSRP before being used as internal FSR.	Input
0	1	0	Internal FSR is driven by the SRG FSG.	Output. FSG is inverted as determined by FSRP before being driven out on the McBSPn.FSR pin.
0	1	1	Internal FSR is driven by the SRG FSG.	Input. The external frame-sync input on the McBSPn.FSR pin is used to synchronize CLKG and generate FSG pulses.
1	0	0	Internal FSX drives internal FSR.	High impedance
1	0 or 1	1	Internal FSX drives internal FSR.	Input. If the SRG is running, external FSR is used to synchronize CLKG and generate FSG pulses.
1	1	0	Internal FSX drives internal FSR.	Output. Receive (same as FSX is inverted as determined by FSRP before being driven out on the McBSPn.FSR pin.

21.5.3.16 Set the FSR Polarity

The FSRP bit (MCBSP_PCR_REG[2]) determines whether frame-sync pulses are active high or active low on the McBSPn.FSR pin.

FSR pulses can be generated internally by the SRG or driven by an external source. The source of frame-sync is selected by programming the MCBSP_PCR_REG[10] register FSRM mode bit. FSR is also affected by the MCBSP_SRGR2_REG[15] register GSYNC bit. For information about the effects of FSRM and GSYNC, see Section 21.5.3.15. Likewise, receive clocks can be selected to be inputs or outputs by programming the MCBSP_PCR_REG[8] register CLKRM mode bit (see Section 21.5.3.18, *Set the Receive Clock Mode*).

When FSR and FSX are inputs (FSXM = FSRM= 0, external frame-sync pulses), the McBSP detects them on the internal falling edge of clock, internal CLKR, and internal CLKX, respectively. The receive data arriving at the McBSPn.DR pin is also sampled on the falling edge of internal CLKR. These internal clock signals are either derived from an external source through CLK(R/X) pins or driven by the SRG clock (CLKG) internal to the McBSP.

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 McBSPn.DX pin is output on the rising edge of internal CLKX.

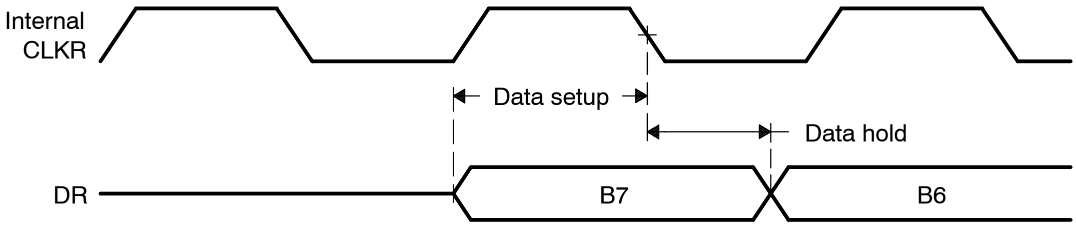
FSRP, FSXP, CLKRP, and CLKXP bits in the pin control register (MCBSP_PCR_REG) configure the polarities of the FSR, FSX, CLKR, and CLKX signals, respectively. FSR and FSX, which are internal to the serial port, are active high. If the serial port is configured for external frame-sync (FSR/FSX are inputs to McBSP), and FSRP = FSXP = 1, the external active-low FSGs 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 FSGs are inverted, if the polarity bit FS(R/X)P = 1, before being sent to the FS(R/X) pin.

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 McBSPn.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 McBSPn.CLKR pin.

Note that 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. Figure 21−48 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 21−48. Data Externally Clocked on a Rising Edge and Sampled on a Falling Edge



21.5.3.17 Set the SRG Frame-Sync Period and Pulse Width

The FPER bit field (MCBSP_SRGR2_REG[11:0]) is used to set the SRG frame-sync period, and FWID bit field (MCBSP_SRGR1_REG[15:8]) is used to set the SRG pulse width.

The SRG can produce a clock signal, CLKG, and an FSG signal. If the SRG is supplying FSR or FSX, bit fields FPER and FWID must be programmed.

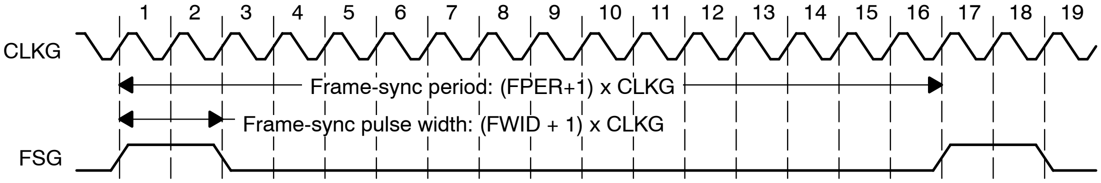
On FSG, the period from the start of a frame-sync pulse to the start of the next pulse is (FPER + 1) CLKG cycles. The 12 bits of FPER allow a frame-sync period of 1 to 4096 CLKG cycles, which allows up to 4096 data bits per frame. When GSYNC = 1, FPER is a don't care value.

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 21−49 shows a frame-sync period of 16 CLKG periods (FPER = 15 or 00001111b) and a frame-sync pulse with an active width of 2 CLKG periods (FWID = 1).

Figure 21−49. Frame of Period 16 CLKG Periods and Active Width of 2 CLKG Periods


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
CLKG
Frame-sync period: (FPER+1) x CLKG
Frame-sync pulse width: (FWID + 1) x CLKG
FSG

When the SRG comes out of reset, FSG is in its inactive state. Then, when GRST = 1 and FSGM = 1, a frame-sync 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.

21.5.3.18 Set the Receive Clock Mode

CLKRM bit (MCBSP_PCR_REG[8]), DLB bit (MCBSP_SPCR1_REG[15]) and CLKSTP bit field (MCBSP_SPCR1_REG[12:11]) are used to set the receive clock mode.

Table 21−22 shows how you can select various sources to provide the receive clock signal and affect the McBSPn.CLKR pin. The polarity of the signal on the McBSPn.CLKR pin is determined by the CLKRP bit.

In the digital loopback mode (DLB = 1), the transmit clock signal is used as the receive clock signal.

Also, in the clock stop mode, the internal receive clock signal (CLKR) and the internal FSR signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX.

Table 21−22. DLB and CLKRM Effect on Receive Clock Signal and McBSPn.CLKR Pin

DLB	CLKRM	Source of Receive Clock	McBSPn.CLKR Pin Status
0	0	The McBSPn.CLKR pin is an input driven by an external clock. The external clock signal is inverted as determined by CLKRP before being used.	Input
0	1	The SRG clock (CLKG) drives internal CLKR.	Output. CLKG, inverted as determined by CLKRP, is driven out on the McBSPn.CLKR pin.
1	0	Internal CLKX drives internal CLKR. To configure CLKX, see Section 21.5.4.18.	High impedance
1	1	Internal CLKX drives internal CLKR. To configure CLKX, see Section 21.5.4.18.	Output. Internal CLKR (same as internal CLKX) is inverted as determined by CLKRP before being driven out on the McBSPn.CLKR pin.

21.5.3.19 Set the Receive Clock Polarity

CLKRP bit (MCBSP_PCR_REG[0]) is used to set the receive clock polarity.

FSR pulses can be generated internally by the SRG or driven by an external source. The source of frame-sync is selected by programming the MCBSP_PCR_REG[10] register FSRM mode bit. FSR is also affected by the GSYNC bit in MCBSP_SRGR2_REG[15]. For information about the effects of FSRM and GSYNC, see Section 21.5.3.15, *Set the Receive Frame-Sync Mode*. Similarly, receive clocks can be selected to be inputs or outputs by programming the MCBSP_PCR_REG[8] register CLKRM mode bit.

When FSR and FSX are inputs (FSXM = FSRM= 0, external frame-sync pulses), the McBSP detects them on the internal falling edge of clock, internal CLKR, and internal CLKX, respectively. The receive data arriving at the McBSPn.DR pin is also sampled on the falling edge of internal CLKR. These internal clock signals are either derived from external source through CLK(R/X) pins or driven by the SRG clock (CLKG) internal to the McBSP.

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 McBSPn.DX pin is output on the rising edge of internal CLKX.

FSRP, FSXP, CLKRP, and CLKXP in the pin control register (MCBSP_PCR_REG) configure the polarities of the FSR, FSX, CLKR, and CLKX signals, respectively. FSR and FSX, which are internal to the serial port, are active high. If the serial port is configured for external frame-sync (FSR/FSX are inputs to McBSP) and FSRP = FSXP = 1, the external active-low FSGs 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 FSGs are inverted, if the polarity bit FS(R/X)P = 1, before being sent to the FS(R/X) pin.

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 McBSPn.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 McBSPn.CLKR pin.

Note that 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.

21.5.3.20 Set the SRG Clock Divide-Down Value

CLKGDV bit field (MCBSP_SRGR1_REG[7:0]) 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 uses 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. When CLKGDV is odd or equal to 0, the CLKG duty cycle is 50%. When CLKGDV is an even value, 2p, representing an odd divide-down, the high-state duration is p + 1 cycles, and the low-state duration is p cycles.

21.5.3.21 Set the SRG Clock Synchronization Mode

GSYNC bit (MCBSP_SRGR2_REG[15]) is used to set the SRG clock-synchronization mode.

For more information about the clock-synchronization feature, see Section 21.4.1, *McBSP Sample Rate Generator*.

21.5.3.22 Set the SRG Clock Mode (Choose an Input Clock)

SCLKME bit (MCBSP_PCR_REG[7]) and CLKSM bit (MCBSP_SRGR2_REG[13]) 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 more information about the clock-synchronization feature, see Section 21.4.1, *McBSP Sample Rate Generator*.

21.5.3.23 Set the SRG Input Clock Polarity

CLKSP bit (MCBSP_SRGR2_REG[14]), CLKXP bit (MCBSP_PCR_REG[1]) and CLKRP bit (MCBSP_PCR_REG[0]) are used to set the SRG input clock polarity.

The SRG can produce a clock signal (CLKG) and an 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 either the CLK pin of the McBSP module, a signal coming from an external pin (McBSPn.CLKX or McBSPn.CLKR), or a signal coming from the CLKS pin of the McBSP module, which can be connected either to the McBSPn_FCLK clock or the McBSP.CLKS OMAP pin.

The source is selected with the SCLKME bit of the MCBSP_PCR_REG[7] register, the CLKSM bit of the MCBSP_SRGR2_REG[13] register, and the CONTROL_DEVCONF[2] or CONTROL_DEVCONF[6] bits.

If a pin is used, choose a polarity for that pin by using the appropriate polarity bit (CLKSP for the McBSP.CLKS pin, CLKXP for the McBSPn.CLKX pin, CLKRP for the McBSPn.CLKR pin). The polarity determines whether the rising or falling edge of the input clock generates transitions on CLKG and FSG.

21.5.4 Transmitter Configuration

To configure the McBSP transmitter, perform the following procedure:

Step 1: Place the McBSP transmitter in reset.

Step 2: Program the McBSP registers for the desired transmitter operation.

Step 3: Take the transmitter out of reset.

These three steps are detailed in the sections below.

21.5.4.1 Programming the McBSP Registers for the Desired Transmitter Operation

The following is a list of important tasks to be performed when you are configuring the McBSP transmitter. Each task corresponds to one or more McBSP register bit fields.

- Global behavior:
  - Set the transmitter pins to operate as McBSP pins.
  - Enable/disable the digital loopback mode.
  - Enable/disable the clock stop mode.
  - Enable/disable transmit multichannel selection.
- Data behavior:
  - Choose 1 or 2 phases for the transmit frame.
  - Set the transmit word length(s).
  - Set the transmit frame length.
  - Enable/disable the FSX ignore function.
  - Set the transmit companding mode.
  - Set the transmit data delay.
  - Set the transmit DXENA mode.
  - Set the transmit interrupt mode.
- Frame-sync behavior:
  - Set the FSX mode.
  - Set the FSX polarity.
  - Set the SRG frame-sync period and pulse width.
- Clock behavior:
  - Set the transmit clock mode.
  - Set the transmit clock polarity.
  - Set the SRG clock divide-down value.
  - Set the SRG clock synchronization mode.
  - Set the SRG clock mode (choose an input clock).
  - Set the SRG input clock polarity.

21.5.4.2 Resetting and Enabling 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 two ways:

1) An OMAP2420 reset places the receiver, transmitter, and SRG in reset. When the device reset is removed, GRST, FRST, RRST and XRST bits = 0, keeping the entire serial port in the reset state.

2) The serial port transmitter and receiver can be reset directly using the RRST and XRST bits in the MCBSP_SPCR1_REG register. The SRG can be reset directly using the GRST bit in MCBSP_SPCR2_REG register.

21.5.4.3 Set the Transmitter Pins to Operate as McBSP Pins

The XIOEN bit (MCBSP_PCR_REG[13]) determines whether the transmitter pins are McBSP pins or general-purpose I/O pins.

For more information, see Section 21.6, *McBSP Registers*.

21.5.4.4 Enable/Disable the Digital Loopback Mode

See Section 21.5.3.4, *Enable/Disable the Digital Loopback Mode*.

21.5.4.5 Enable/Disable the Clock Stop Mode

See Section 21.5.3.5, *Enable/Disable the Clock Stop Mode*.

21.5.4.6 Enable/Disable the Transmit Multichannel Selection

The XMCM bit field (MCBSP_MCR2_REG[1:0]) determines whether the transmit multichannel selection mode is on or off.

For more information, see Section 21.4.3.7, *Transmit Multichannel Selection Modes*.

21.5.4.7 Choose 1 or 2 Phases for the Transmit Frame

The XPHASE bit (MCBSP_XCR2_REG[15]) determines whether the transmit data frame has one or two phases.

21.5.4.8 Set the Transmit Word Length(s)

The XWDLEN1 (MCBSP_XCR1_REG[7:5]) and XWDLEN2 (MCBSP_XCR2_REG[7:5]) bit fields determine how many bits are in each serial word in phase 1 and phase 2, respectively, of the transmit data frame.

Each frame can have one or two phases, depending on the value that you load into the XPHASE bit. 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 determines the length of the serial words in phase 1 of the frame, and XWDLEN2 determines the word length in phase 2 of the frame.

21.5.4.9 Set the Transmit Frame Length

The XFRLEN1 (MCBSP_XCR1_REG[14:8]) and XFRLEN2 (MCBSP_XCR2_REG[14:8]) bit fields determine how many serial words are in phase 1 and phase 2, respectively, of the transmit data frame.

The transmit frame length is the number of serial words in the transmit frame. Each frame can have one or two phases, depending on value that you load into the XPHASE bit.

If a single-phase frame is selected (XPHASE = 0), the frame length is equal to the length of phase 1. If a dual-phase frame is selected (XPHASE = 1), the frame length is the length of phase 1 plus the length of phase 2.

The 7-bit XFRLEN fields allow up to 128 words per phase. See Table 21−23 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-sync 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 21−23. How to Calculate the Length of the Transmit Frame

XPHASE	XFRLEN1	XFRLEN2	Frame Length
0	0 = XFRLEN1 = 127	Don’t care	(XFRLEN1 + 1) words
1	0 = XFRLEN1 = 127	0 = XFRLEN2 = 127	(XFRLEN1 + 1) + (XFRLEN2 + 1) words

21.5.4.10 Enable/Disable the Transmit Frame-Sync Ignore Function

The XFIG bit (MCBSP_XCR2_REG[2]) controls the FSX ignore function.

If a frame-sync pulse starts the transfer of a new frame before the current frame is fully transmitted, this pulse is treated as an unexpected frame-sync pulse.

When XFIG = 1, transmission continues, ignoring the unexpected frame-sync pulses.

When XFIG = 0 and an unexpected frame-sync pulse occurs, the serial port:

1) Aborts the current data transfer

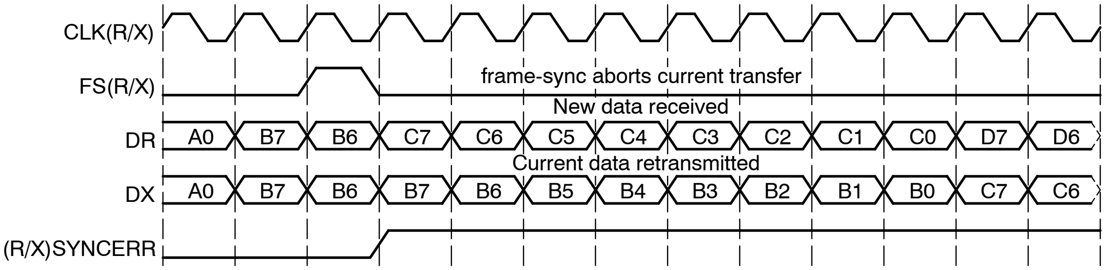
2) Sets MCBSP_SPCR2_REG[3] register XSYNCERR bit to 1

3) Reinitiates transmission of the aborted word

For more information about the frame-sync error condition, see Section 21.4.2, McBSP Exception/Error Conditions.

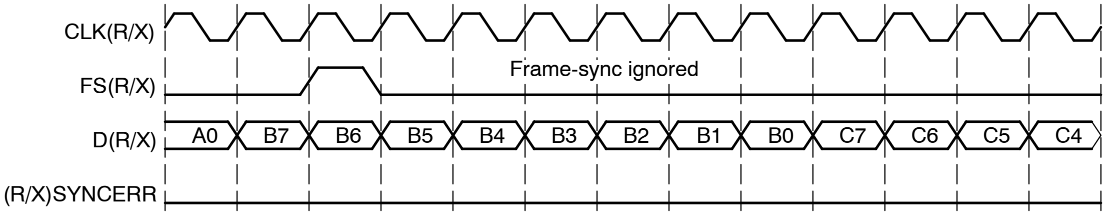
Figure 21−50 shows an example in which word B is interrupted by an unexpected frame-sync pulse when (R/X)FIG = 0. For transmission, the transmission of B is aborted (B is lost). This condition is a transmit synchronization error, which sets the XSYNCERR bit. No new data is written to the MCBSP_DXR[1,2]_REG registers, and therefore the McBSP transmits B again.

Figure 21−50. Unexpected Frame-Sync Pulse With (R/X)FIG = 0


CLK(R/X)
FS(R/X)
frame-sync aborts current transfer
New data received
DR
A0 B7 B6 C7 C6 C5 C4 C3 C2 C1 C0 D7 D6
Current data retransmitted
DX
A0 B7 B6 B7 B6 B5 B4 B3 B2 B1 B0 C7 C6
(R/X)SYNCERR

In contrast to Figure 21−50, Figure 21−51 shows McBSP operation when unexpected FSGs are ignored (when (R/X)FIG = 1). Here, the transfer of word B is not affected by an unexpected pulse.

Figure 21−51. Unexpected Frame-Sync Pulse With (R/X)FIG = 1



21.5.4.11 Set the Transmit Companding Mode

The XCOMPAND bit field (MCBSP_XCR2_REG[4:3]) determines whether companding or another data transfer option is chosen for McBSP reception.

For more information about companding, see Section 21.2, *McBSP Environment*.

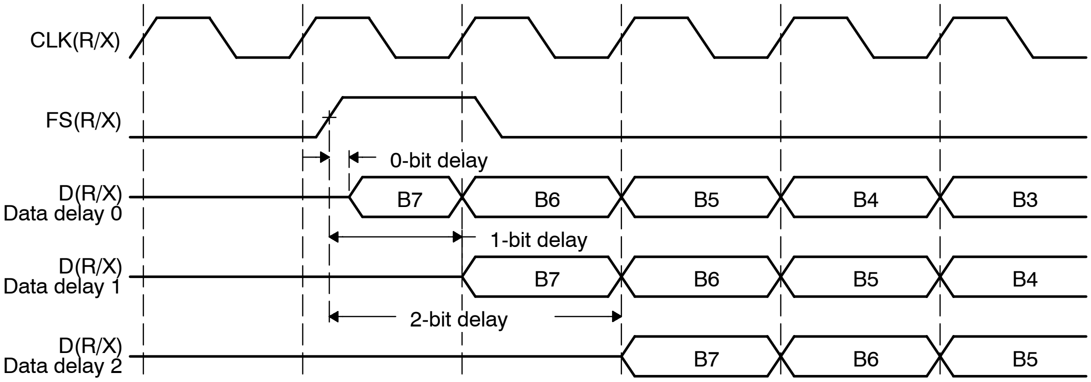
21.5.4.12 Set the Transmit Data Delay

The XDATDLY bit field (MCBSP_XCR2_REG[1:0]) 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-sync is found to be 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.

XDATDLY specifies the data delay for reception. The range of programmable data delay is 0 to 2 bit clocks (XDATDLY = 00b–10b), as shown in Figure 21−52. 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-sync pulse.

Figure 21−52. Range of Programmable Data Delay



0-Bit Data Delay

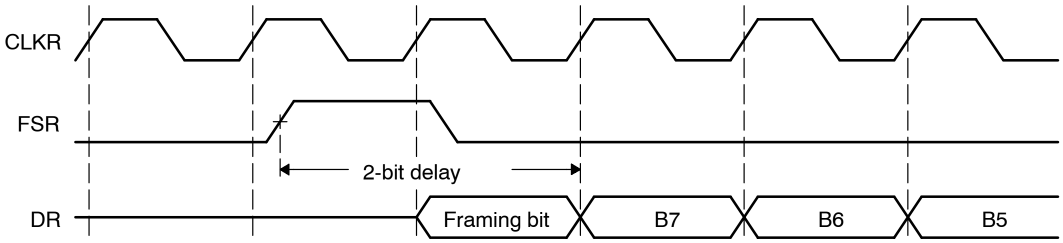
Normally, a frame-sync pulse is detected or sampled with respect to an edge of the internal serial clock CLK(R/X). Thus, on the following cycle or later (depending on the data delay value), data can be received or transmitted. However, for 0-bit data delay, the data must be ready for reception and/or 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-sync. Therefore, the first data bit is assumed to be present in XSR1, and thus on McBSPn.DX. The transmitter then asynchronously detects theFSX signal going active-high and immediately starts driving the first bit to be transmitted on the McBSPn.DX pin.

2-Bit Data Delay

A data delay of two bit periods allows the serial port to interface with 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 21−53. In this figure, the data transferred is an 8-bit value with bits labeled B7, B6, B5, and so on.

Figure 21−53. 2-Bit Data Delay Used to Skip a Framing Bit



21.5.4.13 Set the Transmit DXENA Mode

DXENA bit (MCBSP_SPRC1_REG[7]) is used to set the transmit DXENA (DX delay enabler) mode.

The DXENA bit controls the delay enabler on the McBSPn.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.

If you tie together the McBSPn.DX pins of multiple McBSPs, make sure DXENA = 1 to avoid having more than one McBSP transmit on the data line at one time.

21.5.4.14 Set the Transmit Interrupt Mode

The XINTM bit field (MCBSP_SPCRX_REG[5:4]) determines which event generates a transmit interrupt request to the CPU.

The transmit interrupt (RINT) informs the CPU of changes to the serial port status. Four options exist for configuring this interrupt:

- ☐ XINTM = 00: XINT is generated when the XRDY bit (MCBSP_SPCR2_REG[1]) changes from 0 to 1. Interrupt on every serial word by tracking the XRDY bit in the MCBSP_SPCR2_REG register. Regardless of the value of XINTM, XRDY can be read to detect the XRDY = 1 condition. The interruption is deasserted when the read of DXR1 register is performed.
- ☐ XINTM = 01: XINT is generated by an end-of-block or end-of-frame condition in the transmit multichannel selection mode. In the multichannel selection mode, interrupt after every 16-channel block boundary is crossed within a frame and at the end of the frame. In any other serial transfer case, this setting is not applicable and, therefore, no interrupts are generated. The software must deassert the interruption at the end of the interrupt routine by performing a read access to XINTN register.
- ☐ XINTM = 10: XINT is generated by a new FSX pulse. Interrupt on detection of FSX pulses. This generates an interrupt even when the transmitter is in its reset state. This is done by synchronizing the incoming frame-sync pulse to the CPU clock and sending it to the CPU through XINT. The software must deassert the interruption at the end of the interrupt routine by performing a read access to XINTN register.
- ☐ XINTM = 11: XINT is generated when XSYNCERR is set. Interrupt on frame-sync error. Regardless of the value of XINTM, XSYNCERR can be read to detect this condition. For more information about using XSYNCERR, see Section 21.4.2.6, *Unexpected FSX Pulse*. The interruption id deasserted when a write of 0 to McBSP_SPCR2_REG[3] XSYNCERR bit is performed.

21.5.4.15 Set the Transmit Frame-Sync Mode

FSXM bit (MCBSP_PCR_REG[11]) and FSGM bit (MCBSP_SRGR2_REG[12]) are used to set the FSX mode.

Table 21−24 shows how FSXM and FSGM select the source of FSX pulses. The three choices are:

1) External frame-sync input

2) Sample rate generator FSG

3) Internal signal that indicates a DXR-to-XSR copy is made

Table 21−24 shows the effect of each bit setting on the McBSPn.FSX pin. The polarity of the signal on the McBSPn.FSX pin is determined by the FSXP bit.

Table 21−24. How FSXM and FSGM Select the Source of FSX Pulses

FSXM	FSGM	Source of FSX	McBSPn.FSX Pin Status
0	0 or 1	An external FSG enters the McBSP through the McBSPn.FSX pin. The signal is then inverted by FSXP before being used as internal FSX.	Input
1	1	Internal FSX is driven by the SRG FSG.	Output. FSG is inverted by FSXP before being driven out on McBSPn.FSX pin.
1	0	A DXR-to-XSR copy causes the McBSP to generate a FSX pulse that is 1 cycle wide.	Output. The generated frame-sync pulse is inverted as determined by FSXP before being driven out on McBSPn.FSX pin.

If the SRG creates an FSG signal that is derived from an external input clock, the GSYNC bit determines whether FSG is kept synchronized with pulses on the McBSPn.FSR pin. For more information, see Section 21.4.1.3, Synchronizing SRG Outputs to an External Clock.

In the clock stop mode (CLKSTP = 10b or 11b), the McBSP can act as a master or as a slave in the SPI protocol. If the McBSP is a master and must provide a slave-enable signal (SS) on the McBSPn.FSX pin, make sure that FSXM = 1 and FSGM = 0 so that FSX is an output and is driven active for the duration of each transmission. If the McBSP is a slave, make sure that FSXM = 0 so that the McBSP can receive the slave-enable signal on the McBSPn.FSX pin.

21.5.4.16 Set the FSX Polarity

The FSXP bit (MCBSP_PCR_REG[3]) determines whether frame-sync pulses are active high or active low on the McBSPn.FSX pin.

FSX pulses can be generated internally by the SRG or driven by an external source. The source of frame-sync is selected by programming the MCBSP_PCR_REG[11] register FSXM mode bit. FSX is also affected by the FSGM bit (MCBSP_SRGR2_REG[12]). For information about the effects of FSXM and FSGM, see Section 21.5.4.15, *Set the FSX Mode*. Similarly, transmit clocks can be selected to be inputs or outputs by programming the MCBSP_PCR_REG[9] register CLKXM mode bit (see Section 21.5.4.18, *Set the Transmit Clock Mode*).

When FSR and FSX are inputs (FSXM = FSRM= 0, external frame-sync pulses), the McBSP detects them on the internal falling edge of clock, internal CLKR, and internal CLKX, respectively. The receive data arriving at the McBSPn.DR pin is also sampled on the falling edge of internal CLKR. These internal clock signals are either derived from external source through CLK(R/X) pins or driven by the SRG clock (CLKG) internal to the McBSP.

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 McBSPn.DX pin is output on the rising edge of internal CLKX.

FSRP, FSXP, CLKRP, and CLKXP in the pin control register (MCBSP_PCR_REG) configure the polarities of the FSR, FSX, CLKR, and CLKX signals, respectively. FSR and FSX, which are internal to the serial port, are active high. If the serial port is configured for external frame-sync (FSR/FSX are inputs to McBSP) and FSRP = FSXP = 1, the external active-low FSGs 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 FSGs are inverted before being sent to the FS(R/X) pin.

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 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 McBSPn.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 McBSPn.CLKR pin.

Note that 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.

21.5.4.17 Set the SRG Frame-Sync Period and Pulse Width

This section is identical to Section 21.5.3.17, *Set the SRG Frame-Sync Period and Pulse Width*.

21.5.4.18 Set the Transmit Clock Mode

CLKXM bit (MCBSP_PCR_REG[9]) is used to set the transmit clock mode.

Table 21−25 shows how the CLKXM bit selects the transmit clock and the corresponding status of the McBSPn.CLKX pin. The polarity of the signal on the McBSPn.CLKX pin is determined by the CLKXP bit.

Table 21−25. CLKXM Bit Effect on Transmit Clock and McBSPn.CLKX Pin

CLKXM	Source of Transmit Clock	McBSPn.CLKX Pin Status
0	Internal CLKX is driven by an external clock on the McBSPn.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 McBSPn.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 McBSPn.FSR pin.

In the clock stop mode (CLKSTP = 10b or 11b), the McBSP can act as a master or as a slave in the SPI protocol. If the McBSP is a master, make sure that CLKXM = 1 so that CLKX is an output to supply the master clock to any slave devices. If the McBSP is a slave, make sure that CLKXM = 0 so that CLKX is an input to accept the master clock signal.

21.5.4.19 Set the Transmit Clock Polarity

CLKXP bit (MCBSP_PCR_REG[1]) is used to set the transmit clock polarity.

FSX pulses can be either generated internally by the SRG or driven by an external source. The source of frame-sync is selected by programming the MCBSP_PCR_REG[11] register FSXM mode bit. FSX is also affected by the FSGM bit (MCBSP_SRGR2_REG[12]). Similarly, transmit clocks can be selected to be inputs or outputs by programming the MCBSP_PCR_REG[9] register CLKXM mode bit.

When FSR and FSX are inputs (FSXM = FSRM= 0, external frame-sync pulses), the McBSP detects them on the internal falling edge of clock, internal CLKR, and internal CLKX, respectively. The receive data arriving at the McBSPn.DR pin is also sampled on the falling edge of internal CLKR. These internal clock signals are either derived from external source through CLK(R/X) pins or driven by the SRG clock (CLKG) internal to the McBSP.

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 McBSPn.DX pin is output on the rising edge of internal CLKX.

FSRP, FSXP, CLKRP, and CLKXP in the pin control register (MCBSP_PCR_REG) configure the polarities of the FSR, FSX, CLKR, and CLKX signals, respectively. FSR and FSX, which are internal to the serial port, are active high. If the serial port is configured for external frame-sync (FSR/FSX are inputs to McBSP), and FSRP = FSXP = 1, the external active-low FSGs 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 FSGs are inverted, if the polarity bit FS(R/X)P = 1, before being sent to the FS(R/X) pin.

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 McBSPn.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 McBSPn.CLKR pin.

Note that 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.

21.5.4.20 Set the SRG Clock Divide-Down Value

This section is identical to Section 21.5.3.20, *Set the SRG Clock Divide-Down Value*.

21.5.4.21 Set the SRG Clock Synchronization Mode

This section is identical to Section 21.5.3.21, *Set the SRG Clock Synchronization Mode*.

21.5.4.22 Set the SRG Clock Mode (Choose an Input Clock)

This section is identical to Section 21.5.3.22, *Set the SRG Clock Mode (Choose and Input Clock)*.

21.5.4.23 Set the SRG Input Clock Polarity

This section is identical to Section 21.5.3.23, *Set the SRG Input Clock Polarity*.

21.5.5 General-Purpose I/O on McBSP Pins

Table 21−26 summarizes how to use the McBSP pins as general-purpose I/O pins. All the bits mentioned in the table, except XRST and RRST, are in the pin control register (MCBSP_PCR_REG). XRST and RRST are in the serial port control registers (MCBSP_SPCR2_REG[0] and MCBSP_SPCR1_ REG[0], respectively).

To use receiver pins McBSPn.CLKR, McBSPn.FSR, and McBSPn.DR as general-purpose I/O pins rather than as serial port pins, you must set two conditions:

- The receiver of the serial port is in reset (MCBSP_SPCR1_REG[0] RRST bit = 0).
- General-purpose I/O is enabled for the serial port receiver (RIOEN = 1 in MCBSP_PCR_REG[12]).

The McBSPn.CLKR and McBSPn.FSR pins can be individually configured as either input or output pins with the CLKRM and FSRM bits, respectively. The McBSPn.DR pin can only be an input pin. Table 21−26 shows which bits in MCBSP_PCR_REG are used to read from/write to these pins.

For the transmitter pins McBSPn.CLKX, McBSPn.FSX, and McBSPn.DX, you must meet two conditions:

- The transmitter of the serial port is in reset (XRST = 0 in MCBSP_SPCR2_REG[0]).
- General-purpose I/O is enabled for the serial port transmitter (XIOEN = 1 in MCBSP_PCR_REG[12]).

The McBSPn.CLKX and McBSPn.FSX pins can be individually configured as input or output pins with the CLKXM and FSXM bits, respectively. The McBSPn.DX pin can only be an output pin. Table 21−26 shows which bits in MCBSP_PCR_REG are used to read from/write to these pins.

For the McBSP.CLKS pin, all of the reset and I/O enable conditions must be met:

- Both the receiver and transmitter of the serial port are in reset (RRST = 0 and XRST = 0).
- 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 CLKS_STAT bit in MCBSP_PCR_REG.

Table 21−26. 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
McBSPn.CLKX	XRST = 0 XIOEN = 1	CLKXM = 1	CLKXP	CLKXM = 0	CLKXP
McBSPn.FSX	XRST = 0 XIOEN = 1	FSXM = 1	FSXP	FSXM = 0	FSXP
McBSPn.DX	XRST = 0 XIOEN = 1	Always	DX_STAT	Never	Does not apply
McBSPn.CLKR	RRST = 0 RIOEN = 1	CLKRM = 1	CLKRP	CLKRM = 0	CLKRP
McBSPn.FSR	RRST = 0 RIOEN = 1	FSRM = 1	FSRP	FSRM = 0	FSRP
McBSPn.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

21.5.6 Data Packing Examples

This section shows two ways to implement data packing in the McBSP.

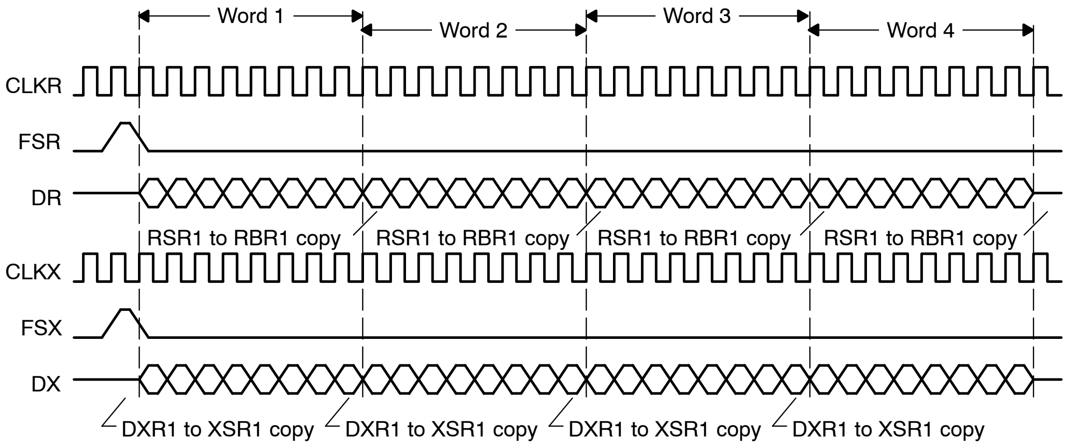
21.5.6.1 Data Packing Using Frame Length and Word Length

Frame length and word length can be manipulated to pack data effectively. For example, consider a situation where four 8-bit words are transferred in a single-phase frame as shown in Figure 21−54. In this case:

- (R/X)PHASE = 0: Single-phase frame
- (R/X)FRLEN1 = 0000011b: 4-word frame
- (R/X)WDLEN1 = 000b: 8-bit words

Four 8-bit data words are transferred to and from the McBSP by the CPU or by the DMA controller. Thus, four reads from MCBSP_DRR1_REG and four writes to MCBSP_DXR1_REG are necessary for each frame.

Figure 21−54. Four 8-Bit Data Words Transferred To/From McBSP



This data can also be treated as a single-phase frame consisting of one 32-bit data word, as shown in Figure 21−55. In this case:

☐ (R/X)PHASE = 0: Single-phase frame

☐ (R/X)FRLEN1 = 0000000b: 1-word frame

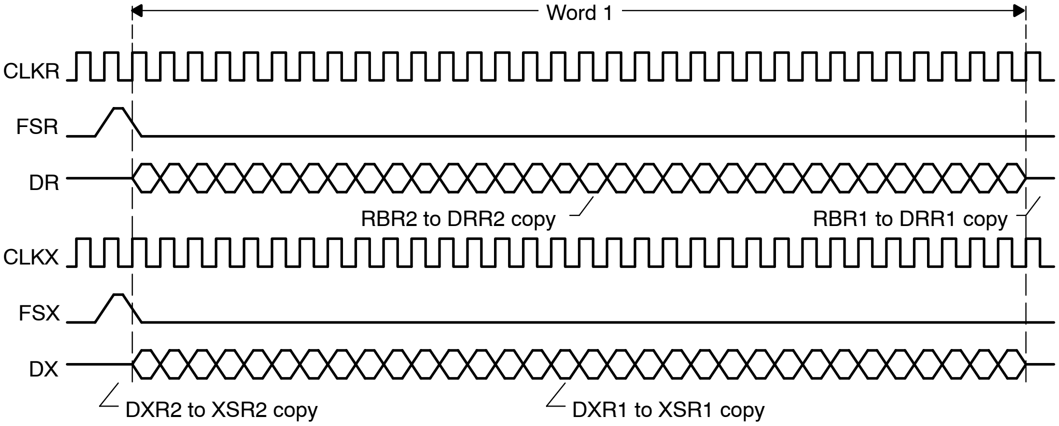
☐ (R/X)WDLEN1 = 101b: 32-bit word

Two 16-bit data words are transferred to and from the McBSP by the CPU or DMA controller. Thus, two reads, from MCBSP_DRR2_REG and MCBSP_DRR1_REG, and two writes, to MCBSP_DXR2_REG and MCBSP_DXR1_REG, are necessary for each frame. This results in only half the number of transfers compared to the previous case. This manipulation reduces the percentage of bus time required for serial port data movement.

Note:

When the word length is larger than 16 bits, make sure you access MCBSP_DRR2_REG and MCBSP_DXR2_REG registers before you access MCBSP_DRR1_REG and MCBSP_DXR1_REG registers. McBSP activity is tied to accesses of MCBSP_DRR1_REG and MCBSP_DXR1_REG. During the reception of 24-bit or 32-bit words, read MCBSP_DRR2_REG and then read MCBSP_DRR1_REG. Otherwise, the next RBR[1,2]−to−DRR[1,2] copy occurs before MCBSP_DRR2_REG is read. Similarly, during the transmission of 24-bit or 32-bit words, write to MCBSP_DXR2_REG and then write to MCBSP_DXR1_REG. Otherwise, the next DXR[1,2]−to−XSR[1,2] copy occurs before MCBSP_DXR2_REG is loaded with new data.

Figure 21−55. One 32-Bit Data Word Transferred To/From the McBSP



21.5.6.2 Data Packing Using Word Length and the Frame-Sync Ignore Function

When there are multiple words per frame, you can implement data packing by increasing the word length (defining a serial word with more bits) and by ignoring frame-sync pulses. First, consider Figure 21−56 below, which shows the McBSP operating at the maximum packet frequency. Here, each frame only has a single 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 21−56. 8-Bit Data Words Transferred at Maximum Packet Frequency

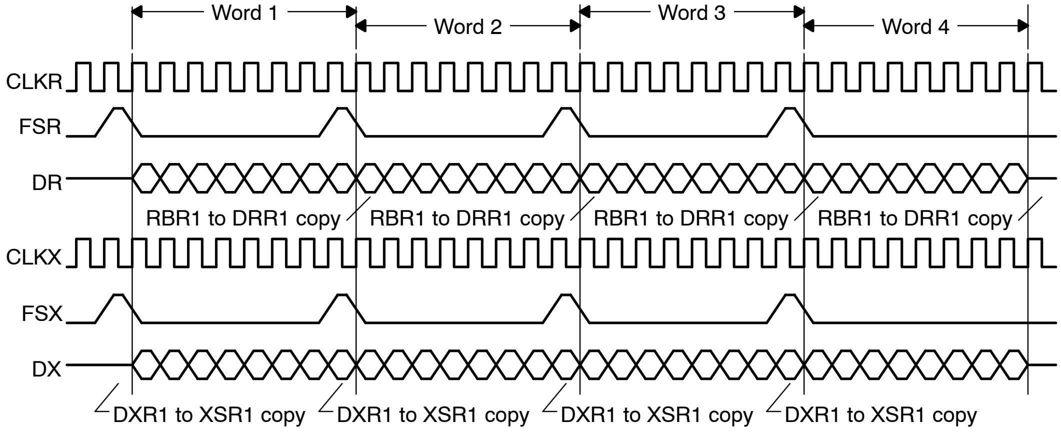
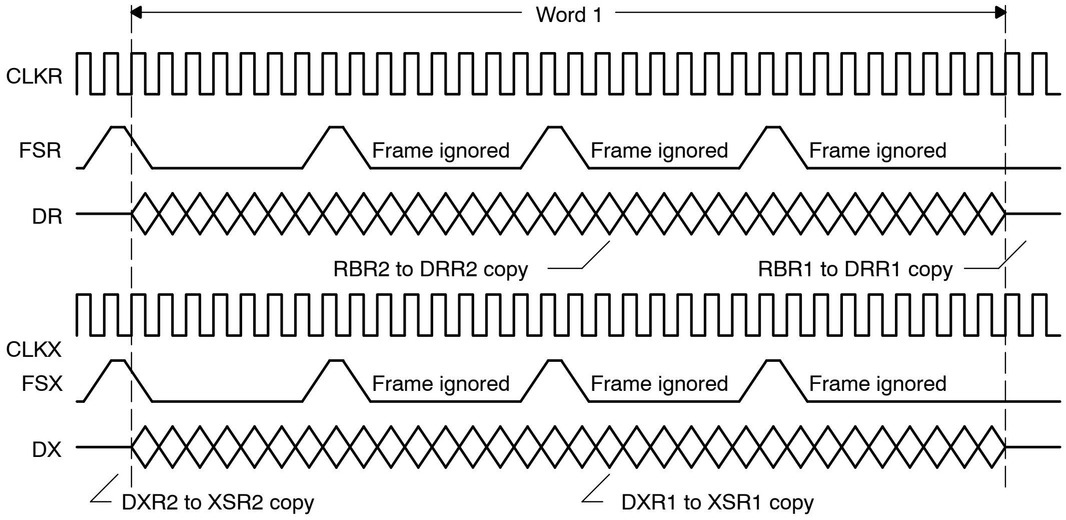


Figure 21−57 shows the McBSP configured to treat this data stream as a continuous 32-bit word. In this example, the McBSP responds to an initial frame-sync pulse. However, (R/X)FIG = 1 so that the McBSP ignores subsequent pulses. Only two read transfers or two write transfers are needed every 32 bits. This configuration effectively reduces the required bus bandwidth to half the bandwidth needed to transfer four 8-bit words.

Figure 21−57. Configuring the Data Stream as a Continuous 32-Bit Word



21.5.7 Procedure for Configuring the McBSP for SPI Operation

To configure the McBSP for SPI master or slave operation:

Step 1: Place the transmitter and receiver in reset.

Clear the transmitter reset bit (MCBSP_SPCR2_REG[0] register XRST bit = 0) to reset the transmitter. Clear the receiver reset bit (MCBSP_SPCR1_REG[0] register RRST bit = 0) to reset the receiver.

Step 2: Place the SRG in reset.

Clear the SRG reset bit (MCBSP_SPCR2_REG[6] register GRST bit = 0) to reset the SRG.

Step 3: Program registers that affect SPI operation.

Program the appropriate McBSP registers to configure the McBSP for proper operation as an SPI master or an SPI slave. For a list of important bits settings, see one of the following topics:

McBSP as the SPI Master (Section 21.4.4.5)

McBSP as an SPI Slave (Section 21.4.4.6)

Step 4: Enable the SRG.

To release the SRG from reset, set the SRG reset bit (MCBSP_SPCR2_REG[6] register GRST bit = 1).

Make sure that during the write to MCBSP_SPCR2_REG register, you only modify the GRST bit. Otherwise, you modify the McBSP configuration you selected in the previous step.

Step 5: Enable the transmitter and receiver.

After the SRG is released from reset, wait two SRG clock periods for the McBSP logic to stabilize.

If the CPU services the McBSP transmit and receive buffers, then you can immediately enable the transmitter (MCBSP_SPCR2_REG[0] register XRST bit = 1) and enable the receiver (MCBSP_SPCR1_REG[0] register RRST bit = 1).

If the DMA controller services the McBSP transmit and receive buffers, then you must first configure the DMA controller (this includes enabling the channels that service the McBSP buffers). When the DMA controller is ready, make XRST = 1 and RRST = 1.

In either case, make sure you only change XRST and RRST when you write to MCBSP_SPCR2_REG[0] register and MCBSP_SPCR1_REG[0] register. Otherwise, you modify the bit settings you selected earlier in this procedure.

After the transmitter and receiver are released from reset, wait two SRG clock periods for the McBSP logic to stabilize.

Step 6: If necessary, enable the frame-sync logic of the SRG.

After the required data acquisition setup is done (MCBSP_DXR[1,2]_REG is loaded with data), set FRST = 1 if an internally generated frame-sync pulse is required (that is, if the McBSP is the SPI master).

21.6 McBSP Registers

Table 21−27. Instance Summary

Module Name	Base Address	Size
MCBSP1	0x4807 4000	128 bytes
MCBSP2	0x4807 6000	128 bytes

21.6.1 McBSP Register Mapping Summary

Table 21−28. McBSP Register Offset Address

Register Name	Type	Register Width (Bits)	Address Offset	McBSP1 Physical Address	McBSP2 Physical Address
MCBSP_DRR2_REG	RW	16	0x00	0x4807 4000	0x4807 6000
MCBSP_DRR1_REG	RW	16	0x04	0x4807 4004	0x4807 6004
MCBSP_DXR2_REG	RW	16	0x08	0x4807 4008	0x4807 6008
MCBSP_DXR1_REG	RW	16	0x0C	0x4807 400C	0x4807 600C
MCBSP_SPCR2_REG	RW	16	0x10	0x4807 4010	0x4807 6010
MCBSP_SPCR1_REG	RW	16	0x14	0x4807 4014	0x4807 6014
MCBSP_RCR2_REG	RW	16	0x18	0x4807 4018	0x4807 6018
MCBSP_RCR1_REG	RW	16	0x1C	0x4807 401C	0x4807 601C
MCBSP_XCR2_REG	RW	16	0x20	0x4807 4020	0x4807 6020
MCBSP_XCR1_REG	RW	16	0x24	0x4807 4024	0x4807 6024
MCBSP_SRGR2_REG	RW	16	0x28	0x4807 4028	0x4807 6028
MCBSP_SRGR1_REG	RW	16	0x2C	0x4807 402C	0x4807 602C
MCBSP_MCR2_REG	RW	16	0x30	0x4807 4030	0x4807 6030
MCBSP_MCR1_REG	RW	16	0x34	0x4807 4034	0x4807 6034
MCBSP_RCERA_REG	RW	16	0x38	0x4807 4038	0x4807 6038
MCBSP_RCERB_REG	RW	16	0x3C	0x4807 403C	0x4807 603C
MCBSP_XCERA_REG	RW	16	0x40	0x4807 4040	0x4807 6040
MCBSP_XCERB_REG	RW	16	0x44	0x4807 4044	0x4807 6044
MCBSP_PCR_REG	RW	16	0x48	0x4807 4048	0x4807 6048
MCBSP_RCERC_REG	RW	16	0x4C	0x4807 404C	0x4807 604C
MCBSP_RCERD_REG	RW	16	0x50	0x4807 4050	0x4807 6050
MCBSP_XCERC_REG	RW	16	0x54	0x4807 4054	0x4807 6054
MCBSP_XCERD_REG	RW	16	0x58	0x4807 4058	0x4807 6058
MCBSP_RCERE_REG	RW	16	0x5C	0x4807 405C	0x4807 605C
MCBSP_RCERF_REG	RW	16	0x60	0x4807 4060	0x4807 6060
MCBSP_XCERE_REG	RW	16	0x64	0x4807 4064	0x4807 6064
MCBSP_XCERF_REG	RW	16	0x68	0x4807 4068	0x4807 6068
MCBSP_RCERG_REG	RW	16	0x6C	0x4807 406C	0x4807 606C

Table 21−28. McBSP Register Offset Address (Continued)

Register Name	Type	Register Width (Bits)	Address Offset	McBSP1 Physical Address	McBSP2 Physical Address
MCBSP_RCERH_REG	RW	16	0x70	0x4807 4070	0x4807 6070
MCBSP_XCERG_REG	RW	16	0x74	0x4807 4074	0x4807 6074
MCBSP_XCERH_REG	RW	16	0x78	0x4807 4078	0x4807 6078
MCBSP_REV_REG	RW	16	0x7C	0x4807 407C	0x4807 607C
MCBSP_RINTN	R	16	0x80	0x4807 4080	0x4807 6080
MCBSP_XINTN	R	16	0x84	0x4807 4084	0x4807 6084

21.6.2 Register Descriptions

Table 21−29. MCBSP_DRR2_REG

Address Offset	0x00		
Physical Address	0x4807 4000	**Instance**	MBSP1
	0x4807 6000		MBSP2
Description	McBSP data receive register 2		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DRR2															

Bits	Field Name	Description	Type	Reset
15:0	DRR2	Data receive register 2 (high part of receive data)	R	0x0000

Table 21−30. MCBSP_DRR1_REG

Address Offset	0x04		
Physical Address	0x4807 4004	**Instance**	MBSP1
	0x4807 6004		MBSP2
Description	McBSP data receive register 1		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DRR1															

Bits	Field Name	Description	Type	Reset
15:0	DRR1	Data receive register 1 (low part of the receive data)	R	0x0000

Table 21−31. MCBSP_DXR2_REG

Address Offset	0x08		
Physical Address	0x4807 4008	**Instance**	MBSP1
	0x4807 6008		MBSP2
Description	McBSP data transmit register 2		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DXR2															

Bits	Field Name	Description	Type	Reset
15:0	DXR2	Data transmit register 2 (high part of transmit data)	W	0x0000

Table 21−32. MCBSP_DXR1_REG

Address Offset	0x0C		
Physical Address	0x4807 400C	**Instance**	MBSP1
	0x4807 600C		MBSP2
Description	McBSP data transmit register 1		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DXR1															

Bits	Field Name	Description	Type	Reset
15:0	DXR1	Data transmit register 1 (low part of transmit data)	W	0x0000

Table 21−33. MCBSP_SPCR2_REG

Address Offset	0x10		
Physical Address	0x4807 4010	**Instance**	MBSP1
	0x4807 6010		MBSP2
Description	McBSP serial port control register 2		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD_UQ0						FREE	SOFT	FRST	GRST	XINTM		XSYNCERR	XEMPTY	XRDY	XRST

Table 21–33. MCBSP_SPCR2_REG (Continued)

Bits	Field Name	Description	Type	Reset
15:10	RSVD_UQ0	Reserved	R	0x00
9	FREE	Free-running mode 0x0: Free-running mode is disabled. 0x1: Free-running mode is enabled.	RW	0
8	SOFT	Soft bit 0x0: SOFT mode is disabled. 0x1: SOFT mode is enabled.	RW	0
7	FRST	Frame-sync generator reset 0x0: Frame-sync logic is reset. FSG is not generated by the SRG. 0x1: FSG is generated after (FPER+1) number of CLKG clocks; that is, all frame counters are loaded with their programmed values.	RW	0
6	GRST	SRG 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	0
5:4	XINTM	Transmit interrupt mode 0x0: XINT is driven by XRDY (that is, end of word) and end of frame in A-bis mode. 0x1: XINT generated by end-of-block or end-of-frame in multichannel operation 0x2: XINT generated by a new frame-sync 0x3: XINT generated by XSYNCERR	RW	0x0
3	XSYNCERR	Transmit synchronization error 0x0: No synchronization error 0x1: Synchronization error detected by McBSP	RW	0
2	XEMPTY	Transmit shift register (XSR[1,2]) empty 0x0: XSR[1,2] is empty. 0x1: XSR[1,2] is not empty.	R	0
1	XRDY	Transmitter ready 0x0: Transmitter is not ready. 0x1: Transmitter is ready for new data in DXR[1,2].	R	0
0	XRST	Transmitter reset. This reset and enables the transmitter. 0x0: The serial port transmitter is disabled and in reset state. 0x1: The serial port transmitter is enabled.	RW	0

Table 21–34. MCBSP_SPCR1_REG

Address Offset	0x14		
Physical Address	0x4807 4014	**Instance**	MBSP1
	0x4807 6014		MBSP2
Description	McBSP serial port control register 1		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DLB	RJUST		CLKSTP		RSVD_UQ1			DXENA	RESERVED	RINTM		RSYNCERR	RFULL	RRDY	RRST

Bits	Field Name	Description		Type	Reset
15	DLB	Digital loopback mode		RW	0
		0x0:	Digital loopback mode disabled		
		0x1:	Digital loopback mode enabled		
14:13	RJUST	Receive sign-extension and justification mode		RW	0x0
		0x0:	Right-justify and zero-fill MSBs in DRR[1,2]		
		0x1:	Right-justify and sign-extend MSBs in DRR[1,2]		
		0x2:	Left-justify and zero-fill LSBs in DRR[1,2]		
		0x3:	Reserved		
12:11	CLKSTP	Clock stop mode		RW	0x0
		0x0:	Clock stop mode disabled. Normal clocking for non-SPI mode		
		0x1:	Clock stop mode disabled. Normal clocking for non-SPI mode		
		0x2:	Clock stop mode, without clock delay. CLKXP = 1: Clock starts with falling edge without delay. CLKXP = 0: Clock starts with rising edge without delay.		
		0x3:	Clock stop mode, with half-cycle clock delay. CLKXP = 1: Clock starts with falling edge with delay. CLKXP = 0: Clock starts with rising edge with delay.		
10:8	RSVD_UQ1	Reserved		R	0x0
7	DXENA	DX enabler		RW	0
		0x0:	DX enabler is off.		
		0x1:	DX enabler is on.		

Table 21–34. MCBSP_SPCR1_REG (Continued)

Bits	Field Name	Description	Type	Reset
6	Reserved	Reserved for future compatibility	RW	0
5:4	RINTM	Receive interrupt mode 0x0: RINT driven by RRDY (that is, end of word) and end of frame in A-bis mode 0x1: RINT generated by end-of-block or end-of-frame in multichannel operation 0x2: RINT generated by a new frame-sync 0x3: RINT generated by RSYNCERR	RW	0x0
3	RSYNCERR	Receive sync error 0x0: No sync error 0x1: Sync error detected by McBSP	RW	0
2	RFULL	Receive shift register (RSR[1,2]) full 0x0: RBR[1,2] is not in overrun condition. 0x1: DRR[1,2] is not read, RBR[1,2] is full and RSR[1,2] is also full with new word.	R	0
1	RRDY	Receiver ready 0x0: Receiver is not ready. 0x1: Receiver is ready with data to be read from DRR[1,2].	R	0
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	0

Table 21–35. MCBSP_RCR2_REG

Address Offset	0x18		
Physical Address	0x4807 4018	**Instance**	MBSP1
	0x4807 6018		MBSP2
Description	McBSP receive control register 2		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RPHASE	RFRLEN2							RWDLEN2			RCOMPAND		RFIG	RDATDLY	

Table 21–35. MCBSP_RCR2_REG (Continued)

Bits	Field Name	Description	Type	Reset
15	RPHASE	Receive phases 0x0: Single-phase frame 0x1: Dual-phase frame	RW	0

14:8	RFRLEN2	Receive frame length 2
		RFRLEN2= 000 0000 – 1 word per frame
		RFRLEN2= 000 0001 – 2 words per frame
		RFRLEN2= 111 1111 – 128 words per frame
			RW	0x00

7:5	RWDLEN2	Receive word length 2 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
4:3	RCOMPAND	Receive companding mode. Modes other than 00b are only enabled when the appropriate RWDLEN is 000b, indicating 8-bit data. 0x0: No companding, data transfer starts with MSB first. 0x1: No companding, 8-bit data, transfer starts with LSB first. 0x2: Compand using µ-law for receive data 0x3: Compand using A-law for receive data	RW	0x0
2	RFIG	Receive frame ignore 0x0: FSR pulses after the first restarts the transfer. 0x1: FSR pulses after the first are ignored.	RW	0
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 21–36. MCBSP_RCR1_REG

Address Offset	0x1C		
Physical Address	0x4807 401C	**Instance**	MBSP1
	0x4807 601C		MBSP2
Description	McBSP receive control register 1		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD1	RFRLEN1							RWDLEN1			RSVD2				

Bits	Field Name	Description	Type	Reset
15	RSVD1	Reserved	R	0

14:8	RFRLEN1	Receive frame length 1
		RFRLEN1=000 0000 – 1 word per frame
		RFRLEN1=000 0001 – 2 words per frame
		RFRLEN1=111 1111 – 128 words per frame
			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	RSVD2	Reserved	R	0x00

Table 21–37. MCBSP_XCR2_REG

Address Offset	0x20		
Physical Address	0x4807 4020	**Instance**	MBSP1
	0x4807 6020		MBSP2
Description	McBSP transmit control register 2		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XPHASE	XFRLEN2							XWDLEN2			XCOMPAND		XFIG	XDATDLY	

Bits	Field Name	Description	Type	Reset
15	XPHASE	Transmit phases 0x0: Single-phase frame 0x1: Dual-phase frame	RW	0

14:8	XFRLEN2	Transmit frame length 2
		XFRLEN2=000 0000 – 1 word per frame
		XFRLEN2=000 0001 – 2 words per frame
		XFRLEN2=111 1111 – 128 words per frame
			RW	0x00

7:5	XWDLEN2	Transmit word length 2 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
4:3	XCOMPAND	Transmit companding mode. Modes other than 00b are only enabled when the appropriate XWDLEN is 000b, indicating 8-bit data. 0x0: No companding, data transfer starts with MSB first. 0x1: No companding, 8-bit data, transfer starts with LSB first. 0x2: Compand using µ-law for transmit data. 0x3: Compand using A–law for transmit data.	RW	0x0

Table 21−37. MCBSP_XCR2_REG (Continued)

Bits	Field Name	Description	Type	Reset
2	XFIG	Transmit frame ignore 0x0: FSX pulses after the first restarts the transfer. 0x1: FSX pulses after the first are ignored.	RW	0
1:0	XDATDLY	Transmit data delay 0x0: 0-bit data delay 0x1: 1-bit data delay 0x2: 2-bit data delay 0x3: Reserved	RW	0x0

Table 21−38. MCBSP_XCR1_REG

Address Offset	0x24		
Physical Address	0x4807 4024	**Instance**	MBSP1
	0x4807 6024		MBSP2
Description	McBSP transmit control register 1		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD1	XFRLEN1							XWDLEN1			RSVD2				

Bits	Field Name	Description	Type	Reset
15	RSVD1	Reserved	R	0

14:8	XFRLEN1	Transmit frame length 1
		XFRLEN1=000 0000 − 1 word per frame
		XFRLEN1=000 0001 − 2 words per frame
		XFRLEN1=111 1111 − 128 words per frame
			RW	0x00

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	RSVD2	Reserved	R	0x00

Table 21–39. MCBSP_SRGR2_REG

Address Offset	0x28		
Physical Address	0x4807 4028	**Instance**	MBSP1
	0x4807 6028		MBSP2
Description	McBSP SRG register 2		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GSYNC	CLKSP	CLKSM	FSGM	FPER											

Bits	Field Name	Description	Type	Reset
15	GSYNC	SRG 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 FSG is generated only after detecting the FSR signal (FSR). Also, frame period, FPER, is a don't care because the period is dictated by the external frame-sync pulse.	RW	0
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	0
13	CLKSM	McBSP SRG clock mode 0x0: SCLKME = 0: SRG clock is derived from the CLKS pin of the McBSP module. SCLKME = 1: SRG clock is derived from the CLKRI pin 0x1: SCLKME = 0: SRG clock is derived from the CLK pin of the McBSP module SCLKME = 1: SRG clock is derived form the CLKXI clock	RW	1

Table 21−39. MCBSP_SRGR2_REG (Continued)

Bits	Field Name	Description	Type	Reset
12	FSGM	SRG FSX mode Used when FSXM=1 in the PCR. 0x0: FSX signal due to DXR[1,2]−to−XSR[1,2] copy. When FSGM=0, FPR and FWID are ignored. 0x1: FSX signal driven by the SRG FSG.	RW	0
11:0	FPER	Frame period. This field plus 1 determines when the next FSG becomes active. Range: 1 to 4096 CLKG periods	RW	0x000

Table 21−40. MCBSP_SRGR1_REG

Address Offset	0x2C		
Physical Address	0x4807 402C	**Instance**	MBSP1
	0x4807 602C		MBSP2
Description	McBSP SRG register 1		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FWID								CLKGDV							

Bits	Field Name	Description	Type	Reset
15:8	FWID	Frame width. This field plus 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	SRG 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 21−41. MCBSP_MCR2_REG

Address Offset	0x30		
Physical Address	0x4807 4030	**Instance**	MBSP1
	0x4807 6030		MBSP2
Description	McBSP multichannel register 2		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD_UQ2						XMCME	XPBBLK		XPABLK		XCBLK			XMCM	

Bits	Field Name	Description	Type	Reset
15:10	RSVD_UQ2	Reserved	R	0x00
9	XMCME	XMCME transmit multichannel partition mode bit 0 XMCME is only applicable if channels can be individually disabled/enabled or masked/unmasked for transmission (XMCM is nonzero). XMCME determines whether only 32 channels or all 128 channels are to be individually selectable. 0x0: Two-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	RW	0

Table 21−41. MCBSP_MCR2_REG (Continued)

Bits	Field Name	Description		Type	Reset
9 cont'd	XMCME cont'd	0x1:	Eight-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		
8:7	XPBBLK	Transmit partition B block		RW	0x0
		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		
6:5	XPABLK	Transmit partition A block		RW	0x0
		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		
4:2	XCBLK	Transmit current block		R	0x0
		0x0:	Block 0. Channel 0 to channel 15		
		0x1:	Block 1. Channel 16 to channel 31		
		0x2:	Block 2. Channel 32 to channel 47		
		0x3:	Block 3. Channel 48 to channel 63		
		0x4:	Block 4. Channel 64 to channel 79		
		0x5:	Block 5. Channel 80 to channel 95		
		0x6:	Block 6. Channel 96 to channel 111		
		0x7:	Block 7. Channel 112 to channel 127		
1:0	XMCM	Transmit multichannel selection enable		RW	0x0
		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; therefore, DX is always driven.		

Table 21−41. MCBSP_MCR2_REG (Continued)

Bits	Field Name	Description	Type	Reset
1:0 cont'd	XMCM cont'd	0x2: All channels enabled, but masked. Selected channels enabled through 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.		

Table 21−42. MCBSP_MCR1_REG

Address Offset	0x34		
Physical Address	0x4807 4034	**Instance**	MBSP1
	0x4807 6034		MBSP2
Description	McBSP multichannel register 1		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD1						RMCME	RPBBLK		RPABLK		RCBLK			RSVD2	RMCM

Bits	Field Name	Description	Type	Reset
15:10	RSVD1	Reserved	R	0x00
9	RMCME	Multichannel buffered serial port (McBSP) RMCME receive multichannel partition mode bit 0 RMCME is only applicable if channels can be individually enabled or disabled for reception (RMCM = 1). RMCME determines whether only 32 channels or all 128 channels are to be individually selectable. 0x0: Two-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	RW	0

Table 21−42. MCBSP_MCR1_REG (Continued)

Bits	Field Name	Description		Type	Reset
9 cont'd	RMCME cont'd	0x1:	Eight-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		
8:7	RPBBLK	Receive partition B block		RW	0x0
		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		
6:5	RPABLK	Receive partition A block		RW	0x0
		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		
4:2	RCBLK	Receive current block		R	0x0
		0x0:	Block 0. Channel 0 to channel 15		
		0x1:	Block 1. Channel 16 to channel 31		
		0x2:	Block 2. Channel 32 to channel 47		
		0x3:	Block 3. Channel 48 to channel 63		
		0x4:	Block 4. Channel 64 to channel 79		
		0x5:	Block 5. Channel 80 to channel 95		
		0x6:	Block 6. Channel 96 to channel 111		
		0x7:	Block 7. Channel 112 to channel 127		
1	RSVD2	Reserved		R	0
0	RMCM	Receive multichannel selection enable		RW	0
		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		

Table 21−43. MCBSP_RCERA_REG

Address Offset	0x38		
Physical Address	0x4807 4038	**Instance**	MBSP1
	0x4807 6038		MBSP2
Description	McBSP receive channel enable register partition A		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCERA															

Bits	Field Name	Description	Type	Reset
15:0	RCERA	Receive channel enable RCERA n=0: Disables reception of nth channel in an even-numbered block in partition A RCERA n=1: Enables reception of nth channel in an even-numbered block in partition A	RW	0x0000

Table 21−44. MCBSP_RCERB_REG

Address Offset	0x3C		
Physical Address	0x4807 403C	**Instance**	MBSP1
	0x4807 603C		MBSP2
Description	McBSP receive channel enable register partition B		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCERB															

Bits	Field Name	Description	Type	Reset
15:0	RCERB	Receive channel enable RCERB n=0: Disables reception of nth channel in an even-numbered block in partition B RCERB n=1: Enables reception of nth channel in an even-numbered block in partition B	RW	0x0000

Table 21−45. MCBSP_XCERA_REG

Address Offset	0x40		
Physical Address	0x4807 4040	**Instance**	MBSP1
	0x4807 6040		MBSP2
Description	McBSP transmit channel enable register partition A		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XCERA															

Bits	Field Name	Description	Type	Reset
15:0	XCERA	Transmit channel enable XCERA n=0: Disables transmission of nth channel in an event−numbered block in partition A XCERA n=1: Enables transmission of nth channel in an event−numbered block in partition A	RW	0x0000

Table 21−46. MCBSP_XCERB_REG

Address Offset	0x44		
Physical Address	0x4807 4044	**Instance**	MBSP1
	0x4807 6044		MBSP2
Description	McBSP transmit channel enable register partition B		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XCERB															

Bits	Field Name	Description	Type	Reset
15:0	XCERB	Transmit channel enable XCERB n=0: Disables transmission of nth channel in an even-numbered block in partition B XCERB n=1: Enables transmission of nth channel in an even-numbered block in partition B	RW	0x0000

Table 21–47. MCBSP_PCR_REG

Address Offset	0x48		
Physical Address	0x4807 4048	**Instance**	MBSP1
	0x4807 6048		MBSP2
Description	McBSP pin control register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD_UQ3	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
15	RSVD_UQ3	Reserved	R	0
14	IDLE_EN	Idle enable. This bit allows to stop all the clocks in the McBSP. 0x0: The McBSP is running. 0x1: The clocks in the McBSP are shut off when both IDLE_EN=1 and peripheral domain is in idle mode.	RW	0
13	XIOEN	Transmit general-purpose I/O mode only when XRST=0 in SPCR[1,2] 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	0
12	RIOEN	Receive general-purpose I/O mode when RRST=0 in SPCR[1,2] 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	0
11	FSXM	FSX mode 0x0: FSG derived from an external source 0x1: Frame-sync is determined by the SRG frame-sync mode bit FSGM in SRGR2.	RW	0

Table 21−47. MCBSP_PCR_REG (Continued)

Bits	Field Name	Description	Type	Reset
10	FSRM	FSR mode 0x0: Frame-sync pulses generated by an external device. FSR is an input pin. 0x1: Frame-sync generated internally by SRG. FSR is an output pin except when GSYNC=1 in SRGR.	RW	0
9	CLKXM	Transmitter clock mode During SPI mode (when CLKSTP is a non-zero value): CLKM=0: McBSP is a slave, and clock (CLKX) is driven by the SPI master in the system. CLKR is internally driven by CLKX. CLKM=1:McBSP is a master and generates the clock (CLKX) to drive its receive clock (CLKR) and the shift clock of the SPI-compliant slaves in the system. 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	0
8	CLKRM	Receiver clock mode 0x0: Case 1: Digital loop-back mode not set (DLB=0) in SPCR1: Receive clock (CLKR) is an input driven by an external clock. Case 2: Digital loop-back mode set (DLB=1) in SPCR1: Receive clock (not the CLKR pin) is driven by transmit clock (CLKX) which is based on the CLKXM bit in the PCR. CLKR pin is in high impedance. 0x1: Case 1: Digital loop-back mode not set (DLB=0) in SPCR1: CLKR is an output pin and is driven by the internal SRG. Case 2: Digital loop-back mode set (DLB=1) in SPCR1: CLKR is an output pin and is driven by the transmit clock. The transmit clock is derived based on the CLKRM bit in the PCR.	RW	0

Table 21−47. MCBSP_PCR_REG (Continued)

Bits	Field Name	Description	Type	Reset
7	SCLKME	SCLKME sample-rate-generator input clock mode bit 0 The SRG can produce a clock signal, CLKG. 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 of the McBSP module CLKSM = 1: Signal on CLK pin of the McBSP module 0x1: CLKSM = 0: Signal on CLKR pin CLKSM = 1: Signal on CLKX pin	RW	0
6	CLKS_STAT	CLKS pin status. Reflects value on CLKS pin when selected as a general-purpose input. 0x0: The signal on the CLKS pin is low. 0x1: The signal on the CLKS pin is high.	R	0
5	DX_STAT	DX pin status. Reflects value driven on to DX pin when selected as a general-purpose output. 0x0: Drive the signal on the DX pin low 0x1: Drive the signal on the DX pin high	RW	0
4	DR_STAT	DR pin status. Reflects value on DR pin when selected as a general-purpose input. 0x0: The signal on DR pin is low. 0x1: The signal on DR pin is high.	R	0
3	FSXP	FSX polarity 0x0: Frame-sync pulse FSX is active high. 0x1: Frame-sync pulse FSX is active low.	RW	0
2	FSRP	FSR Polarity 0x0: Frame-sync pulse FSR is active high. 0x1: Frame-sync pulse FSR is active low.	RW	0
1	CLKXP	Transmit clock polarity 0x0: Transmit data driven on rising edge of CLKX 0x1: Transmit data driven on falling edge of CLKX	RW	0
0	CLKRP	Receive clock polarity 0x0: Receive data sampled on falling edge of CLKR 0x1: Receive data sampled on rising edge of CLKR	RW	0

Table 21−48. MCBSP_RCERC_REG

Address Offset	0x4C		
Physical Address	0x4807 404C	**Instance**	MBSP1

	0x4807 604C		MBSP2
Description	McBSP receive channel enable register partition C		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCERC															

Bits	Field Name	Description	Type	Reset
15:0	RCERC	Receive channel enable RCERC n=0: Disables reception of nth channel in an even-numbered block in partition C RCERC n=1: Enables reception of nth channel in an even-numbered block in partition C	RW	0x0000

Table 21–49. MCBSP_RCERD_REG

Address Offset	0x50		
Physical Address	0x4807 4050	**Instance**	MBSP1
	0x4807 6050		MBSP2
Description	McBSP receive channel enable register partition D		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCERD															

Bits	Field Name	Description	Type	Reset
15:0	RCERD	Receive channel enable RCERD n=0: Disables reception of nth channel in an even-numbered block in partition D RCERD n=1: Enables reception of nth channel in an even-numbered block in partition D	RW	0x0000

Table 21−50. MCBSP_XCERC_REG

Address Offset	0x54		
Physical Address	0x4807 4054	**Instance**	MBSP1
	0x4807 6054		MBSP2
Description	McBSP transmit channel enable register partition C		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XCERC															

Bits	Field Name	Description	Type	Reset
15:0	XCERC	Transmit channel enable XCERC n=0: Disables transmission of nth channel in an even-numbered block in partition C XCERC n=1: Enables transmission of nth channel in an even-numbered block in partition C	RW	0x0000

Table 21−51. MCBSP_XCERD_REG

Address Offset	0x58		
Physical Address	0x4807 4058	**Instance**	MBSP1
	0x4807 6058		MBSP2
Description	McBSP transmit channel enable register partition D		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XCERD															

Bits	Field Name	Description	Type	Reset
15:0	XCERD	Transmit channel enable XCERD n=0: Disables transmission of nth channel in an even-numbered block in partition D XCERD n=1: Enables transmission of nth channel in an even-numbered block in partition D	RW	0x0000

Table 21−52. MCBSP_RCERE_REG

Address Offset	0x5C		
Physical Address	0x4807 405C	**Instance**	MBSP1
	0x4807 605C		MBSP2
Description	McBSP receive channel enable register partition E		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCERE															

Bits	Field Name	Description	Type	Reset
15:0	RCERE	Receive channel enable RCERE n=0: Disables reception of nth channel in an even-numbered block in partition E RCERE n=1: Enables reception of nth channel in an even-numbered block in partition E	RW	0x0000

Table 21−53. MCBSP_RCERF_REG

Address Offset	0x60		
Physical Address	0x4807 4060	**Instance**	MBSP1
	0x4807 6060		MBSP2
Description	McBSP receive channel enable register partition F		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCERF															

Bits	Field Name	Description	Type	Reset
15:0	RCERF	Receive channel enable RCERF n=0: Disables reception of nth channel in an even-numbered block in partition F RCERF n=1: Enables reception of nth channel in an even-numbered block in partition F	RW	0x0000

Table 21−54. MCBSP_XCERE_REG

Address Offset	0x64		
Physical Address	0x4807 4064	**Instance**	MBSP1
	0x4807 6064		MBSP2
Description	McBSP transmit channel enable register partition E		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XCERE															

Bits	Field Name	Description	Type	Reset
15:0	XCERE	Transmit channel enable XCERE n=0: Disables transmission of nth channel in an even-numbered block in partition E XCERE n=1: Enables transmission of nth channel in an even-numbered block in partition E	RW	0x0000

Table 21−55. MCBSP_XCERF_REG

Address Offset	0x68		
Physical Address	0x4807 4068	**Instance**	MBSP1
	0x4807 6068		MBSP2
Description	McBSP transmit channel enable register partition F		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XCERF															

Bits	Field Name	Description	Type	Reset
15:0	XCERF	Transmit channel enable XCERF n=0: Disables transmission of nth channel in an even-numbered block in partition F XCERF n=1: Enables transmission of nth channel in an even-numbered block in partition F	RW	0x0000

Table 21−56. MCBSP_RCERG_REG

Address Offset	0x6C		
Physical Address	0x4807 406C	**Instance**	MBSP1
	0x4807 606C		MBSP2
Description	McBSP receive channel enable register partition G		
Type	RW		

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">RCERG</td></tr>
</table>

Bits	**Field Name**	**Description**	**Type**	**Reset**
15:0	RCERG	Receive channel enable RCERG n=0: Disables reception of nth channel in an even-numbered block in partition G RCERG n=1: Enables reception of nth channel in an even-numbered block in partition G	RW	0x0000

Table 21−57. MCBSP_RCERH_REG

Address Offset	0x70		
Physical Address	0x4807 4070	**Instance**	MBSP1
	0x4807 6070		MBSP2
Description	McBSP receive channel enable register partition H		
Type	RW		

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">RCERH</td></tr>
</table>

Bits	**Field Name**	**Description**	**Type**	**Reset**
15:0	RCERH	Receive channel enable RCERH n=0: Disables reception of nth channel in an even-numbered block in partition H RCERH n=1: Enables reception of nth channel in an even-numbered block in partition H	RW	0x0000

Table 21−58. MCBSP_XCERG_REG

Address Offset	0x74		
Physical Address	0x4807 4074	**Instance**	MBSP1
	0x4807 6074		MBSP2
Description	McBSP transmit channel enable register partition G		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XCERG															

Bits	Field Name	Description	Type	Reset
15:0	XCERG	Transmit channel enable XCERG n=0: Disables transmission of nth channel in an even-numbered block in partition G XCERG n=1: Enables transmission of nth channel in an even-numbered block in partition G	RW	0x0000

Table 21−59. MCBSP_XCERH_REG

Address Offset	0x78		
Physical Address	0x4807 4078	**Instance**	MBSP1
	0x4807 6078		MBSP2
Description	McBSP transmit channel enable register partition H		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XCERH															

Bits	Field Name	Description	Type	Reset
15:0	XCERH	Transmit channel enable XCERH n=0: Disables transmission of nth channel in an even-numbered block in partition H XCERH n=1: Enables transmission of nth channel in an even-numbered block in partition H	RW	0x0000

Table 21−60. MCBSP_REV_REG

Address Offset	0x7C		
Physical Address	0x4807 407C	**Instance**	MBSP1
	0x4807 607C		MBSP2
Description	MCBSP revision number register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD_UQ4								REV							

Bits	Field Name	Description	Type	Reset
15:8	RSVD_UQ4	Reserved	R	0x00
7:0	REV	Revision number	R	

Table 21−61. MCBSP_RINTN

Address Offset	0x80		
Physical Address	0x4807 4080	**Instance**	MBSP1
	0x4807 6080		MBSP2
Description	Register RINTN		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														RINTN	

Bits	Field Name	Description	Type	Reset
1:0	RINTN	A read access to this field must be performed to deassert the interrupt at the end-of-interrupt routine when receive interrupt mode is 01 or 10. The value read is uncared.	R	0x−−

Table 21−62. MCBSP_XINTN

Address Offset	0x84		
Physical Address	0x4807 4084	**Instance**	MBSP1
	0x4807 6084		MBSP2
Description	Register XINTN		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														XINTN	

Bits	Field Name	Description	Type	Reset
1:0	XINTN	A read access to this field must be performed to deassert the interrupt at the end-of-interrupt routine when transmit interrupt mode is 01 or 10. The value read is uncared.	R	0x−−

Chapter 22

Universal Serial Bus

This chapter describes the three universal serial bus (USB) ports and several varieties of USB functionality on the OMAP2420 processor.

Topic **Page**

22.1 USB Controller Overview

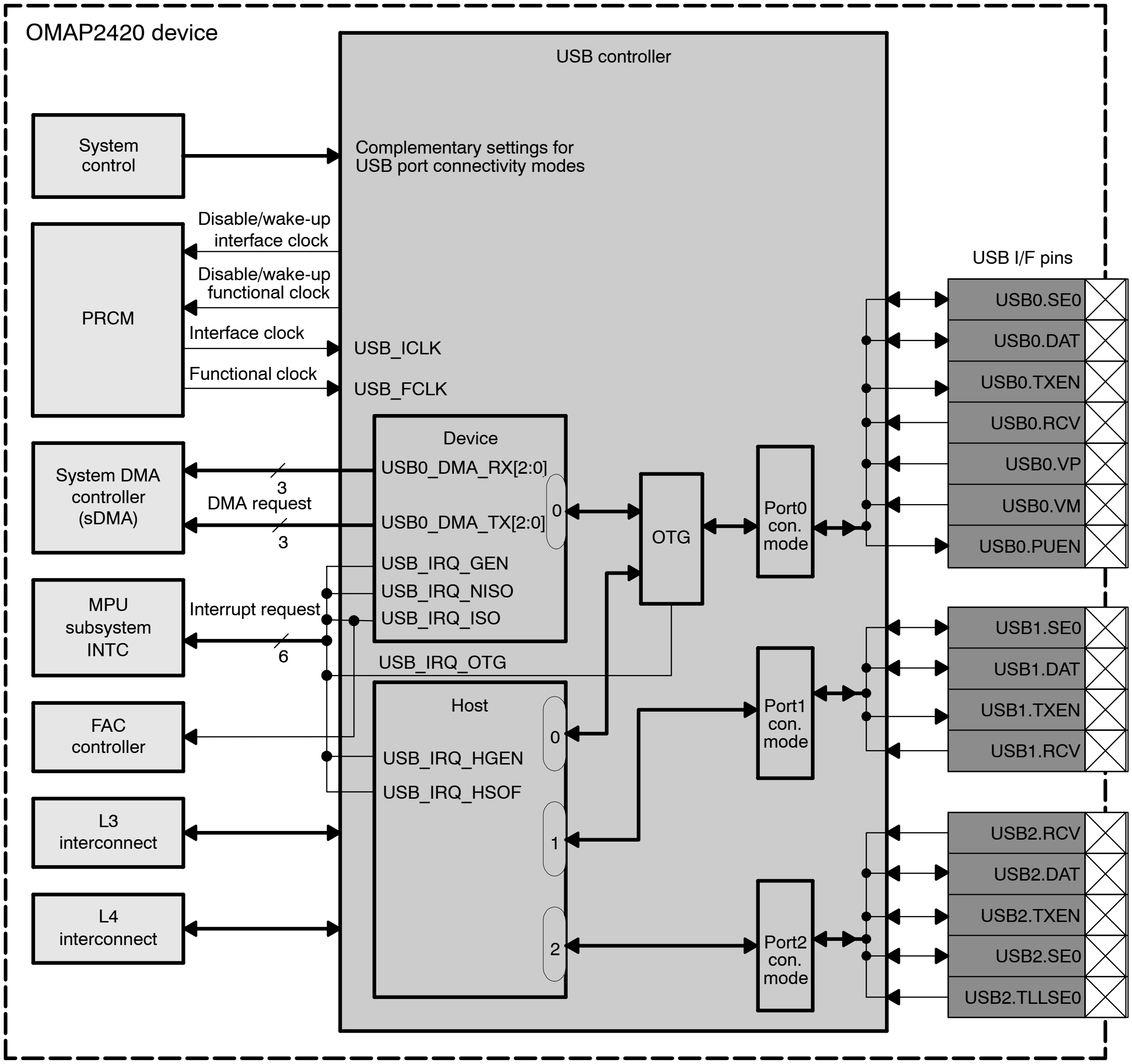
The OMAP2420 device provides three universal serial bus (USB) ports and several varieties of USB functionality, including:

- ❑ One USB device port: The OMAP2420 device provides a full-speed USB device compatible with the *Universal Serial Bus Revision 2.0 Specification* and the *Universal Serial Bus Revision 1.1 Specification* (hereafter called the USB 2.0 specification and the USB 1.1 specification, respectively).
- ❑ Three USB host ports: The OMAP2420 device provides a 3-port USB 2.0 specification-compatible host controller, which is based on the *Open HCI—Open Host Controller Interface Specification for USB,* Release 1.0 (hereafter called the OHCI specification).
- ❑ One USB On-The-Go (OTG) dual-role device (DRD) port compatible with the *On-The-Go Supplement to the USB 2.0 Specification*, (hereafter called the OTG specification supplement): The OMAP2420 device acts as an OTG DRD; the USB device functionality and one port of the USB host controller act in concert to provide an OTG port.

Each port can connect either to a USB transceiver external to the OMAP™ device or directly to an external IC in a transceiverless link (TLL) manner with the assumption that the other IC supports the same TLL protocol.

This is an L3 and L4 interconnect interface with an independent functional clock clocked at 48 MHz. The L4 interconnect is used by the USB device and other block controls, whereas the L3 interconnect is used by the USB host for transmit and receive data (see Figure 22−1).

Figure 22−1. USB Controller Highlight



22.1.1 Main Features

The main features of the multiport USB controller are as follows:

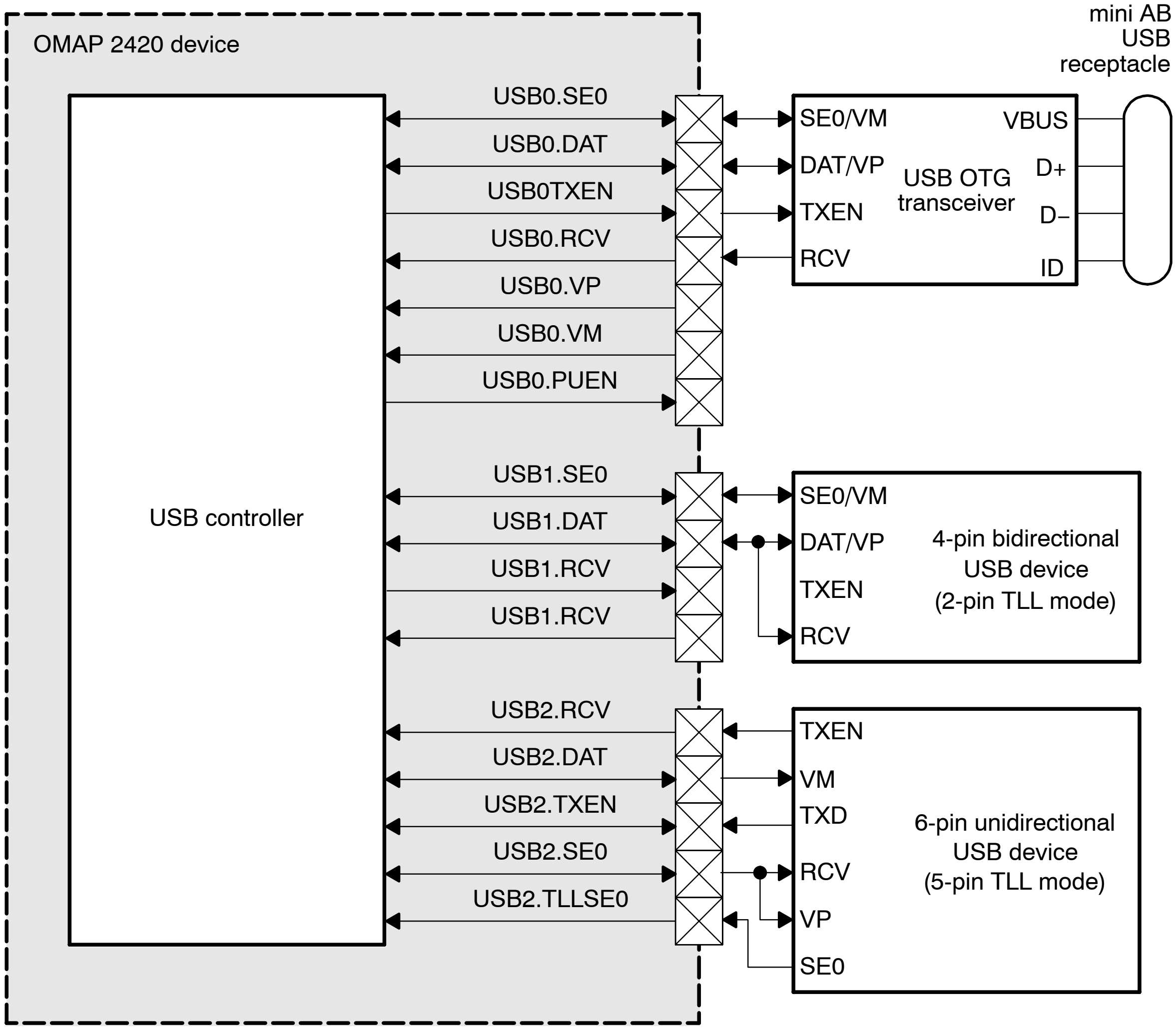
- ❑ USB device features
  - USB 2.0 full-speed (12 Mbps) operations
  - One control and up to 15 IN endpoints and 15 OUT software-configurable endpoints
  - Each endpoint can be configured for types (interrupt, bulk, or isochronous), for size, and for double-buffering operation within a limit of 2K bytes of data.
  - Supports suspend/resume and remote wakeup
  - Six DMA channels (3 IN and 3 OUT), three interrupt lines

- USB host features:
  - Compatible with the OHCI specification
  - USB 2.0 low-speed (1.5 Mbps) and full-speed (12 Mbps) operations
  - Three upstream ports to interface with up to three downstream hubs or devices
  - Two interrupt lines
- USB OTG features:
  - Supports the OTG specification supplement
  - Supports host negotiation protocol (HNP) and session request protocol (SRP)
  - One interrupt line
- USB port signal pins interface supporting:
  - External USB transceivers
    - 3-pin mode (DAT/SE0 bidirectional mode)
    - 4-pin mode (VP/VM bidirectional mode)
    - 6-pin mode (unidirectional mode – on port 0 only)
    - Optional pullup enable control signal on port 0
  - TLL modes enable glueless interconnect to another device USB device port without a transceiver.
    - 2-pin mode (DAT/SE0 and DP/DM bidirectional modes)
    - 5-pin mode (DAT/SE0 and DP/DM unidirectional modes—on port 2 only)

22.2 USB Controller Environment

Figure 22−2 shows a typical application using the USB controller.

Figure 22−2. USB Controller Typical Application System Overview



The USB controller physically connects the OMAP2420 device to external USB transceivers.

An external USB transceiver is required for each USB port used in the system. It converts between signaling that is appropriate for the OMAP2420 USB controllers and signaling that is appropriate for the USB wire. OMAP2420 USB functionality includes support for several types of USB transceivers. The OMAP2420 device provides signaling to up to three external USB transceivers and/or three external USB-OTG-capable transceivers.

TLL modes enable glueless interconnect to another device USB device port without a transceiver.

22.2.1 Selecting and Configuring USB Connectivity

The process of selecting desired USB connectivity and configuring the OMAP2420 device for that connectivity can be done using the following steps.

22.2.1.1 Select Desired USB Functionality

1) Choose the desired USB functionality. The OMAP2420 device can bring up to three USB ports to device pins. The options include the following:
   - One USB device port (available ony on port 0)
   - One USB OTG port (available only on port 0)
   - Up to three USB host ports (ports 1 and 2 are USB host ports only)

2) Choose a maximum of three of the above. Because the USB OTG port must use the single USB device controller, it is not possible to configure both a USB device port and a USB OTG port. Similarly, a USB OTG port uses one of the three USB host controller ports. Some possible configurations are as follows:
   - One USB device port
   - One OTG port and one host port
   - Two host ports and one USB device port

   Some illegal combinations are as follows:
   - One USB device port and one OTG port (only one USB device controller is available; it can be used as part of the OTG functionality or it can be used as a standard USB device port, but not both simultaneously)
   - One OTG port and three host ports (violates the maximum of three USB ports brought to the OMAP2420 pins; also violates the limitation of three available host ports where one is used by OTG)
   - Three host ports and one USB device port (violates the maximum of three USB ports brought to the OMAP2420 pins)

Table 22−1 lists the possible USB functionality in the OMAP2420 device.

Table 22−1. USB Functionality

	Port 0	**Port 1**	**Port 2**
Device (full speed)	✓		
Host (full or low speed)	✓	✓	✓
OTG (full speed)	✓		

22.2.1.2 Select How USB Functionality Is Multiplexed to OMAP2420 Pins

The OMAP2420 device provides pin interfaces for up to three USB ports. The USB functional ports can be mapped to the different OMAP2420 USB ports.

All three USB-related ports are associated with standard CMOS input and output pins. USB connectivity that uses these ports must use USB external transceivers. The OMAP2420 top-level pin multiplexing allows a wide variety of functionality to be provided using these pins, so selecting these pins for use as USB functionality limits the ability to use those pins for other (non-USB) functionality.

The selection procedure is as follows:

1) Given the required USB functionality, the USB port descriptions, and any non-USB use of the pins in the port descriptions, choose a USB signal multiplexing mode from Table 22−2.
2) Find a value of HMC_MODE where all of the required USB functionality is available from the three USB multiplexing port columns.
3) Examine the OMAP2420 pin use shown under the port columns and find one HMC_MODE value that meets your needs for both USB and non-USB functionality.
4) Carefully note the information in the row that determines how the top-level pin multiplexing and HMC_MODE value must be configured (see Table 22−2).

USB controller ports that are not specifically mentioned for a row of Table 22−2 are held in a disabled state. When a USB host controller port is disabled, the port appears to the controller as a disconnected port. When the USB device controller is not available to a port and OTG is not enabled, the USB device controller sees pin signaling equivalent to USB_RESET.

When an HMC_MODE is selected that can connect a USB multiplexing port to OMAP2420 device pins, but all of the pins associated with that port are set for non-USB functionality, the associated USB multiplexing port sees 6-wire transceiver signaling, which implies that both D+ and D– are low. If that port is associated with a host controller port, that host controller port appears as a disconnected USB link. If that port is associated with the device controller, the device controller sees USB reset signaling.

Functionality is undefined when top-level pin multiplexing selects non-USB signaling of one or more of the pins that are required by the selected HMC_MODE and transceiver signaling type. For example, if an HMC_MODE value selects USB port 0 as a USB host connection and a 4-pin transceiver interface is selected, but pin UART2.CTS/USB0.RCV top-level pin multiplexing selects signal UART2.CTS, the associated USB host controller port behavior is undefined.

Table 22−2. USB Configurations

HMC_MODE Bits OTG_SYSCON_2[5:0]	OTG_EN Bit OTG_SYSCON_2[31]	Port 0	Port 1	Port 2
0x0	0	Device	Disabled	Disabled
0x1	0	Host 0	Host 1	Host 2
0x2	0	Unsupported mode		
0x3	0	Unsupported mode		
0x4	0	Device	Host 1	Host 2
0x5	0	Host 0	Disabled	Host 2

Table 22−2. USB Configurations (Continued)

HMC_MODE Bits OTG_SYSCON_2[5:0]	**OTG_EN Bit OTG_SYSCON_2[31]**	**Port 0**	**Port 1**	**Port 2**
0x6	0	Device	Disabled	Host 2
0x7	0	Unsupported mode		
0x8	0	Unsupported mode		
0x9	0	Host 0	Host 1	Host 2 + TLL
0xA	0	Unsupported mode		
0xB	0	Unsupported mode		
0xC	0	Device	Host 1	Host 2 + TLL
0xD	0	Unsupported mode		
0xE	0	Device	Disabled	Host 2 + TLL
0xF	0	Unsupported mode		
0x10	0	Host 0	Disabled	Disabled
0x11	0	Host 0	Host 1	Disabled
0x12	0	Unsupported mode		
0x13	0	Unsupported mode		
0x14	0	Device	Host 1	Disabled
0x15	0	Unsupported mode		
0x16	0	Disabled	Disabled	Disabled
0x17	0	Unsupported mode		
0x18	0	Unsupported mode		
0x19	0	Unsupported mode		
Others	0	Disabled	Disabled	Disabled
0x0 or 0x10	1	OTG	Disabled	Disabled
0x1 or 0x4	1	OTG	Host 1	Host 2
0x2	1	Unsupported mode		
0x3	1	Unsupported mode		
0x5 or 0x6	1	OTG	Disabled	Host 2
0x7	1	Unsupported mode		
0x8	1	Unsupported mode		
0x9 or 0xC	1	OTG	Host 1	Host 2 + TLL
0xA	1	Unsupported mode		
0xB	1	Unsupported mode		
0xD	1	Unsupported mode		
0xE	1	Unsupported mode		
0xF	1	Unsupported mode		
0x11 or 0x14	1	OTG	Host 1	Disabled
0x12	1	Unsupported mode		

Table 22−2. USB Configurations (Continued)

HMC_MODE Bits OTG_SYSCON_2[5:0]	OTG_EN Bit OTG_SYSCON_2[31]	Port 0	Port 1	Port 2
0x13	1	Unsupported mode		
0x15	1	Unsupported mode		
0x16	1	Disabled	Disabled	Disabled
0x17 or 0x19	1	Unsupported mode		
0x18	1	Unsupported mode		
Others	1	Disabled	Disabled	Disabled

Select USB and/or USB OTG Transceiver Type

USB connectivity on OMAP2420 USB port 0, USB port 1, and USB port 2 require external components, except for port 1 and port 2 when they are configured for a TLL mode. Several different types of external transceiver signaling are supported. Signaling between the OMAP2420 USB controller and the external USB transceiver for monitoring and controlling the differential USB signal can be by a 6-pin, 4-pin, or 3-pin signaling interface, with two or more additional control signals provided by additional signals or by an inter-integrated circuit (I^2C) link. OTG transceivers can support the 6-pin, 4-pin, and/or 3-pin basic signaling, but generally provide an interrupt output and an I^2C link for the additional control and status reporting required by OTG functionality.

Carefully examine the options for the HMC_MODE you have identified.

Consider which OMAP2420 pins are required in each diagram and determine if any particular choice (3-pin, 4-pin, or 6-pin) is advantageous in terms of other non-USB functionality that is available for each port. Choose a transceiver type.

A USB OTG transceiver can generally be used in place of a USB transceiver; using an OTG transceiver with an I^2C interface can reduce the number of OMAP2420 pins that are required. This can help make non-USB functionality available on the USB ports.

Determine System Control Module Control Register Settings

The system control module (SCM) controls the settings for the USB ports connectivity modes through its configuration register CONTROL_DEVCONF:

- ❑ USBT0WRMODEI bits CONTROL_DEVCONF[23:22]: USB port 0 top level transceiver interface mode control (00: Unidir; 01: Unidir TLL; 10: Bidir; 11: Bidir TLL)
- ❑ USBT1WRMODEI bits CONTROL_DEVCONF[21:20]: USB port 1 top level transceiver interface mode control (00: Unidir; 01: Unidir TLL; 10: Bidir; 11: Bidir TLL)
- ❑ USBT2WRMODEI bits (CONTROL_DEVCONF[19:18]: USB port 2 top level transceiver interface mode control (00: Unidir; 01: Unidir TLL; 10: Bidir; 11: Bidir TLL)

- ☐ USBT2TLL5PI bit CONTROL_DEVCONF[17]: 5-pin TLL mode enable for USB port 2 (0: disabled; 1: enabled)
- ☐ USB0PUENACTLOI bit CONTROL_DEVCONF[16]: USB port 0 pullup enable polarity control (0: pullup enable active high; 1: pullup enable active low).

Determine OTG Module Control Register Settings

The OTG controller provides registers that control several aspects of the USB pin signaling. These registers must be properly configured to allow proper USB operation, even when the OTG feature is not used.

The configuration of a port is the combination of several parameters. They must be set according to Table 22−4 through Table 22−6 to obtain the desired mode.

- ☐ USB0_TRX_MODE bits OTG_SYSCON_1[18:16]: USB port 0 transceiver mode. These bits are used to select the USB port 0 transceiver interface type. The transceiver signaling type depends on the signaling mode used by the transceiver. The transceiver signaling types supported on the OMAP2420 are 3-pin bidirectional (DAT_SE0) mode transceiver signaling (0x2), 4-pin bidirectional (VP/VM) mode transceiver signaling (0x1), and 6-pin unidirectional transceiver signaling (0x3). When USB port 0 is not connected to a transceiver, those pins must be configured for non-USB operation and the USB0_TRX_MODE bits OTG_SYSCON_1[18:16] must be set to 0x0. See Table 22−4.
- ☐ USB1_TRX_MODE bits OTG_SYSCON_1[22:20]: USB port 1 transceiver mode. These bits are used to select the USB port 1 transceiver interface type. Transceiver signaling type depends on the signaling mode used by the transceiver. The transceiver signaling types supported on the OMAP2420 are 3-pin bidirectional (DAT_SE0) mode transceiver signaling (0x2) and 4-pin bidirectional (VP/VM) mode transceiver signaling (0x1). When USB port 1 is not connected to a transceiver, those pins must be configured for non-USB operation and the USB1_TRX_MODE bits OTG_SYSCON_1[22:20] must be set to 0x0. See Table 22−5.
- ☐ USB2_TRX_MODE bits OTG_SYSCON_1[26:24]: USB port 2 transceiver mode. These bits are used to select the USB port 2 transceiver interface type. Transceiver signaling type depends on the signaling mode used by the transceiver. The OMAP2420-supported transceiver signaling types are 3-pin bidirectional (DAT_SE0) mode transceiver signaling (0x2) and 4-pin bidirectional (VP/VM) mode transceiver signaling (0x1). When USB port 2 is not connected to a transceiver, those pins must be configured for non-USB operation and the USB2_TRX_MODE bits OTG_SYSCON_1[26:24] must be set to 0x0. See Table 22−6.
- ☐ USBx_SYNCHRO bit OTG_SYSCON_2[30] (0: USB signals not synchronized; 1: USB signals synchronized): This bit enables synchronizing (or not) the USB output signals (on all USB ports). When the functionality is activated, all USB output signals are synchronized. It must be set to 1 to allow proper timing on OMAP2420 pins.

- OTG_PADEN bits OTG_SYSCON_2[10]: OTG transceiver control and status information selector. This bit must be set to 0 whenever OTG functionality is required. Writing 1 to this bit prevents proper operation of the OTG_CTRL register. When OTG_PADEN is 0 and OTG_EN is 0, software monitors the VBUS_DETECT signal (GPIO) and updates the BSESSVLD bit to OTG_CTRL[18] with the value from GPIO.

  OTG_PADEN = 1 is not a supported configuration in the OMAP2420.

Table 22−3. OTG_PADEN Source Status

OTG_PADEN bit OTG_SYSCON_2[10]	**Value**	**Source**
0	ID	ID bit OTG_CTRL[16]
	VBUSVLD	VBUSVLD bit OTG_CTRL[17]
	BSESSEND	BSESSEND bit OTG_CTRL[19]
	ASESSVLD	ASESSVLD bit OTG_CTRL[20]
	BSESSVLD	BSESSVLD bit OTG_CTRL[18]
	VBUS value to USB device controller	

- HMC_PADEN bit OTG_SYSCON_2[9]: Must be set to 0 because HMC data comes from internal registers in the OMAP2420 device. It concerns the UHOST_EN, HMC_MODE, HMC_TLLATTACH, and HMC_TLLSPEED bits in the OTG_SYSCON_2 register.
- HMC_MODE bits OTG_SYSCON_2[5:0]: The HMC_MODE bits OTG_SYSCON_2[5:0] control the OMAP USB signal multiplexing selection (see Table 22−5).
- OTG_EN bit OTG_SYSCON_2[31] (0: OTG controller not activated; 1: OTG controller activated): The OTG enable bit selects the OTG controller functionality.
- HMC_MODE and OTG_EN set the port multiplexing—that is, the nature of each port (host, device, and OTG).

Table 22−4. Configuration and Pin Usage for USB Port 0

Mode Name	**6-Pin Unidirectional**	**3-Pin Bidirectional (DAT/SE0)**	**4-Pin Bidirectional (VP/VM)**
Control Signals/Mode			
HMC_MODE bits OTG_SYSCON_2[5:0]	Device, OTG, host	Device, OTG, host	Device, OTG, host
USB0_TRX_MODE bits OTG_SYSCON_1[18:16]	6-pin (0x3)	3-pin (0x2)	4-pin (0x1)
USBT0WRMODEI bits CONTRL_DEVCONF[23:22]	Unidirectional (0b00)	Bidirectional (0b10)	Bidirectional (0b10)
IC pin count	6	3	4

Table 22−5. Configuration and Pin Usage for USB Port 1

Mode Name	**3-Pin Bidirectional (DAT/SE0)**	**4-Pin Bidirectional (VP/VM)**	**2-Pin Bidirectional TLL (DAT/SE0)**	**2-Pin Bidirectional TLL (DP/DM)**
Control Signals/Mode				
HMC_MODE bits OTG_SYSCON_2[5:0]	Host	Host	Host	Host
USB1_TRX_MODE bits OTG_SYSCON_1[22:20]	3-pin (0x2)	4-pin (0x1)	3-pin (0x2)	4-pin (0x1)
USBT1WRMODEI bits CONTRL_DEVCONF[21:20]	Bidirectional (0b10)	Bidirectional (0b10)	Bidirectional (0b10)	Bidirectional TLL (0b11)
IC pin count	3	4	2	2

Table 22−6. Configuration and Pin Usage for USB Port 2

Mode Name	**3-Pin Bidirectional (DAT/SE0)**	**4-Pin Bidirectional (VP/VM)**	**5-Pin Unidirectional TLL (DAT/SE0)**	**5-Pin Unidirectional TLL (DP/DM)**	**2-Pin Bidirectional TLL (DAT/SE0)**	**2-Pin Bidirectional TLL (DP/DM)**
Control Signals/Mode						
HMC_MODE bits OTG_SYSCON_2 [5:0]	Host 2	Host 2	Host 2 + TLL	Host 2 + TLL	Host 2	Host 2
USB2_TRX_MODE bits OTG_SYSCON_1 [26:24]	3-pin (0x2)	4-pin (0x1)	6-pin (0x3)	6-pin (0x3)	3-pin (0x2)	4-pin (0x1)
USBT2WRMODEI bit CONTRL_DEVCONF[19:18]	Bidirectional (0b10)	Bidirectional (0b10)	Unidirectional (0b00)	Unidirectional TLL (0b01)	Bidirectional (0b10)	Bidirectional TLL (0b11)
USBT2TLL5PI bit CONTRL_DEVCONF[17]	0 (don't care)	0 (don't care)	1	1	0 (don't care)	0
IC pin count	3	4	5	5	2	2

Table 22−7 summarizes the previous information.

Table 22−7. USB Connectivity Modes

	Port 0	**Port 1**	**Port 2**
3-pin bidirectional (DAT/SE0)	✓	✓	✓
4-pin bidirectional (VP/VM)	✓	✓	✓
6-pin unidirectional	✓		
TLL 2-pin bidirectional (DAT/SE0)		✓	✓
TLL 2-pin bidirectional (DP/DM)		✓	✓
TLL 5-pin unidirectional (DAT/SE0)			✓
TLL 5-pin unidirectional (DP/DM)			✓

- ☐ OTG_CTRL bit fields: This read/write register reflects the status of some USB OTG control bits. System software uses these register bits to convey OTG transceiver control and status between the transceiver and the OTG controller. The OTG_PU_ID, OTG_PU_VBUS, OTG_PD_VBUS, OTG_DRV_VBUS, OTG_PU, and OTG_PD bits are controlled by the OTG controller and determine how software configures the OTG transceiver. Whenever one of these bits is changed by the OTG controller, an OPRT_CHG interrupt is generated (if enabled). It is best for system software to implement an interrupt controller that updates the OTG transceiver control registers when the OPRT_CHG interrupt occurs.

  The ID, VBUSVLD, BSESSVLD, BSESSEND, and ASESSVLD bits must be updated to reflect the OTG transceiver status. Typically, the OTG transceiver provides an interrupt that can be configured to assert when the OTG transceiver status changes. This interrupt output is generally connected to an OMAP2420 GPIO input that is configured as a microprocessor unit (MPU) interrupt source. The interrupt service routine (ISR) for the transceiver interrupt must query the OTG transceiver interrupt and appropriate status bits by using I^{2}C operations and update the OTG controller registers as appropriate.

  The OTG_BUSDROP, B_BUSREQ, B_HNPEN, A_BUSREQ, and A_SETB_HNPEN bits control the OTG controller state-machines and provide functionality defined in the OTG specification supplement. System software must manage these bits.

  The DRIVER_SEL bit shows whether the USB host controller or the USB device controller has control of the OTG link.

22.2.2 Transceiver Signaling Types

USB Transceiver 6-Pin Unidirectional Signaling

When a USB or USB OTG transceiver is connected to the OMAP2420 device and is used in 6-pin unidirectional DAT/SE0 signaling mode, the signaling shown in Table 22−8 is used.

Table 22−8. Signaling Between USB Controller and 6-Pin Unidirectional USB Transceiver Using DAT/SE0 Signaling

<table>
<tr><th>Logical Signal Name</th><th>OMAP2420 Pin Direction</th><th>Transceiver Pin Direction</th><th>Description</th></tr>
<tr><td>TXEN</td><td>Output</td><td>Input</td><td>When low, USB transceiver drives D+ and D–.</td></tr>
<tr><td>DAT and SE0</td><td>Output</td><td>Input</td><td>Controls the values output by the USB transceiver on D+ and D– when TXEN is low. Ignored when TXEN is high.
<table>
<tr><td>TXEN</td><td>DAT</td><td>SE0</td><td>D+</td><td>D–</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td></td><td>1</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td></td><td>X</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>X</td><td>X</td><td>Undriven</td><td>Undriven</td></tr>
</table></td></tr>
<tr><td>RCV</td><td>Input</td><td>Output</td><td>Output from transceiver differential receiver
<table>
<tr><td>D+</td><td>D–</td><td>RCV</td></tr>
<tr><td>0</td><td>0</td><td>X</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>X</td></tr>
</table></td></tr>
<tr><td>VP</td><td>Input</td><td>Output</td><td>Output from transceiver single-ended D+ signal receiver
<table>
<tr><td>D+</td><td>VP</td></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
</table></td></tr>
<tr><td>VM</td><td>Input</td><td>Output</td><td>Output from transceiver single-ended D– signal receiver
<table>
<tr><td>D–</td><td>VM</td></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
</table></td></tr>
</table>

The OMAP2420 device does not support unidirectional transceivers that implement VPO/VMO signaling.

USB Transceiver 3-Pin Bidirectional Signaling

When a USB or USB OTG transceiver is connected to the OMAP2420 device and is used in 3-pin bidirectional DAT/SE0 signaling mode, the signaling shown in Table 22−9 is used.

Table 22−9. Signaling Between USB Controller and 3-Pin Bidirectional USB Transceiver Using DAT/SE0 Signaling

Logical Signal Name	OMAP2420 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 OMAP2420 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

The OMAP2420 device does not support 3-pin bidirectional signaling using VP/VM signals.

USB Transceiver 4-Pin Bidirectional Signaling

When a USB or USB OTG transceiver is connected to the OMAP2420 device and is used in 4-pin bidirectional VP/VM signaling mode, the signaling shown in Table 22−10 is used.

Table 22−10. Signaling Between USB Controller and 4-Pin Bidirectional USB Transceiver Using VP/VM Signaling

Logical Signal Name	OMAP2420 Pin Direction	Transceiver Pin Direction	Description		
TXEN	Output	Input	When low, USB transceiver drives D+ and D–.		
VM			Value driven to or received from D–		
	Output	Input	TXEN	VM	D–
			0	0	0
			0	1	1
	Input	Output	TXEN	D–	VM
			1	0	0
			1	1	1

Table 22−10. Signaling Between USB Controller and 4-Pin Bidirectional USB Transceiver Using VP/VM Signaling (Continued)

Logical Signal Name	**OMAP2420 Pin Direction**	**Transceiver Pin Direction**	**Description**		
VP			Value driven to or received from D+		
	Output	Input	TXEN	VP	D+
			0	0	0
			0	1	1
	Input	Output	TXEN	D+	VP
			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

The OMAP2420 device does not support 4-pin bidirectional signaling using DAT/SE0 signals.

22.2.3 USB OTG External Connectivity

To provide USB OTG connectivity, a DRD system that provides a USB OTG controller must implement certain features:

- USB mini-AB receptacle
- VBUS monitoring and control
- Transient suppression
- ID monitoring
- Controllable D+ and D– series
- Pullup and pulldown resistors
- D+ and D– driver/receiver

USB OTG transceivers that implement many of these features are available commercially.

Some USB OTG transceivers implement a 6-pin connection to the OTG controller, whereas others require only 4 pins or 3 pins. The OMAP2420 device must be properly configured to support the signaling required by the attached transceiver (see Section 22.2.1, *Selecting and Configuring USB Connectivity).*

Many OTG transceiver control and status functions are provided using I^2C registers. This reduces the number of connections between the OTG transceiver and the OTG controller. OTG transceiver I^2C registers control a variety of functions, including D+ and D– pullups, D+ and D– pulldowns, and the VBUS output. OTG transceiver I^2C registers provide transceiver status, including status of the VBUS and ID pins and interrupt conditions.

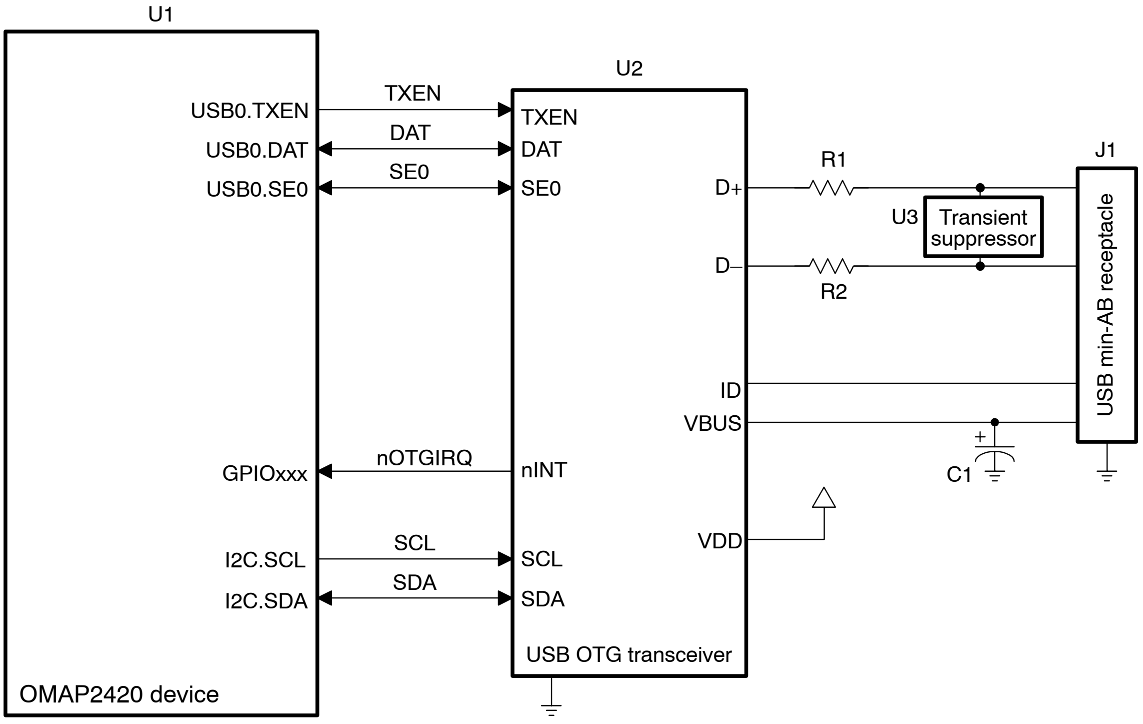
The typical OTG transceiver provides an interrupt output to the processor. This interrupt informs the processor when the VBUS state changes, when the ID state changes, or when any number of other transceiver-dependent conditions occur. When OTG transceiver interrupts occur, system software must query the OTG transceiver status by using I^2C and then update the OTG controller registers as appropriate.

Some system applications require that the USB mini-AB receptacle can be connected to downstream non-OTG USB devices, such as mice, keyboards, or printers. In such cases, the OTG specification supplement allows use of a converter cable with a mini-A plug on one end and a standard type-A receptacle on the other end. The USB OTG transceiver, which can supply the amount of VBUS current required by the OTG specification supplement, may not be able to meet the VBUS current supply requirement of the USB specification. In such cases, it is necessary to provide some alternate VBUS source that meets the USB specification requirements.

OTG on USB Port 0 Using 3-Pin OTG Transceiver

Figure 22−3 shows the OTG on USB port 0 using a 3-pin OTG transceiver.

Figure 22−3. OTG on USB Port 0 Using 3-Pin OTG Transceiver



R1, R2	Value depends on transceiver
C1	VBUS capacitance must meet OTG spec C_{DRD_VBUS}
U1	OMAP2420 device
U2	USB OTG transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
J1	USB mini-AB receptacle

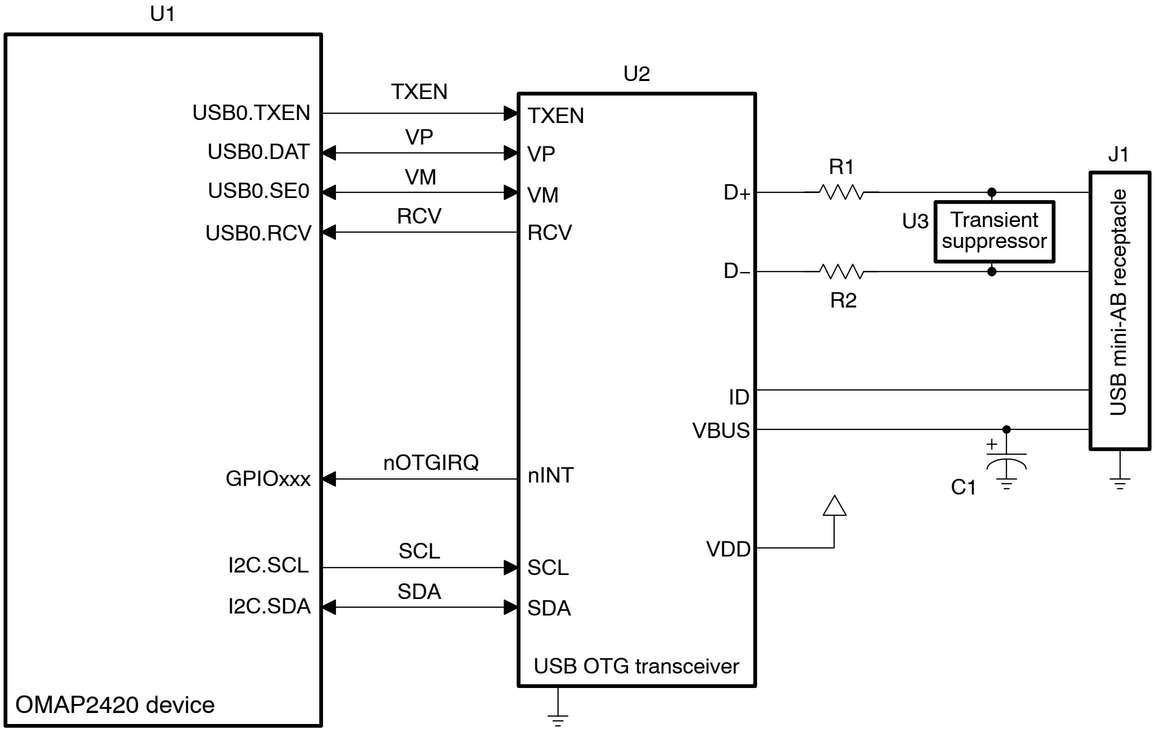
HMC_MODE must be set to a value that provides OTG functionality on USB port 0 (see Table 22−2). The USB0_TRX_MODE bits OTG_

SYSCON_1[18:16] must be set to 0x2 to allow proper bidirectional operation of the 3-pin USB transceiver. The system control module USBT0WRMODEI bits CONTRL_DEVCONF[23:22] must be set to 0b10 to allow bidirectional transceiver interface mode.

OTG on USB Port 0 Using 4-Pin OTG Transceiver

Figure 22−4 shows OTG on USB port 0 using a 4-pin OTG transceiver.

Figure 22−4. OTG on USB Port 0 Using 4-Pin OTG Transceiver



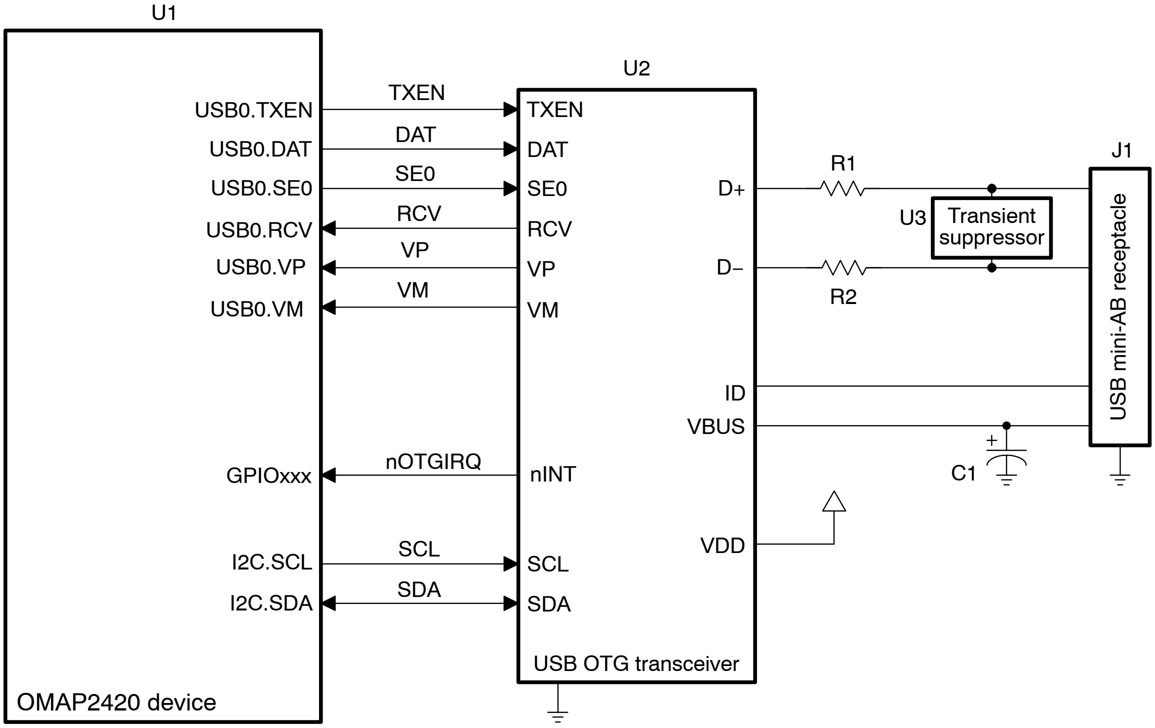
R1, R2	Value depends on transceiver
C1	VBUS capacitance must meet OTG spec C DRD_VBUS
U1	OMAP2420 device
U2	USB OTG transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
J1	USB mini-AB receptacle

HMC_MODE must be set to a value that provides OTG functionality on USB port 0 (see Table 22−2). The USB0_TRX_MODE bits OTG_SYSCON_1[18:16] must be set to 0x1 to allow proper bidirectional operation of the 4-pin USB transceiver. The system control module USBT0WRMODEI bits CONTRL_DEVCONF[23:22] must be set to 0b10 to allow bidirectional transceiver interface mode.

OTG on USB Port 0 Using 6-Pin OTG Transceiver

Figure 22−5 shows OTG on USB port 0 using a 6-pin OTG transceiver.

Figure 22−5. OTG on USB Port 0 Using 6-Pin OTG Transceiver



R1, R2	Value depends on transceiver
C1	VBUS capacitance must meet OTG spec C DRD_VBUS
U1	OMAP2420 device
U2	USB OTG transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
J1	USB mini-AB receptacle

HMC_MODE must be set to a value that provides OTG functionality on USB port 0 (see Table 22−2). The USB0_TRX_MODE bits OTG_SYSCON_1[18:16] must be set to 0x3 to allow proper unidirectional operation of the 6-pin USB transceiver. The system control module USBT0WRMODEI bits CONTRL_DEVCONF[23:22] must be set to 0b00 to allow unidirectional transceiver interface mode.

22.2.4 Host Controller Connectivity With USB Transceivers

To provide a robust USB solution, a system that provides a USB host controller must implement certain features:

- ❑ Type-A receptacle
- ❑ Power on the VBUS signal (can be switched or unswitched power)
- ❑ Transient suppression
- ❑ Pulldown resistors
- ❑ USB-compatible downstream port transceiver

Because the OMAP2420 device does not provide a pin that connects to the USB host controller port power control registers, some other mechanism must be used if VBUS switching is required. Similarly, the OMAP2420 device does not provide any pins that connect to the USB host controller overcurrent status bits, so some other mechanism must be used if overcurrent sensing is required.

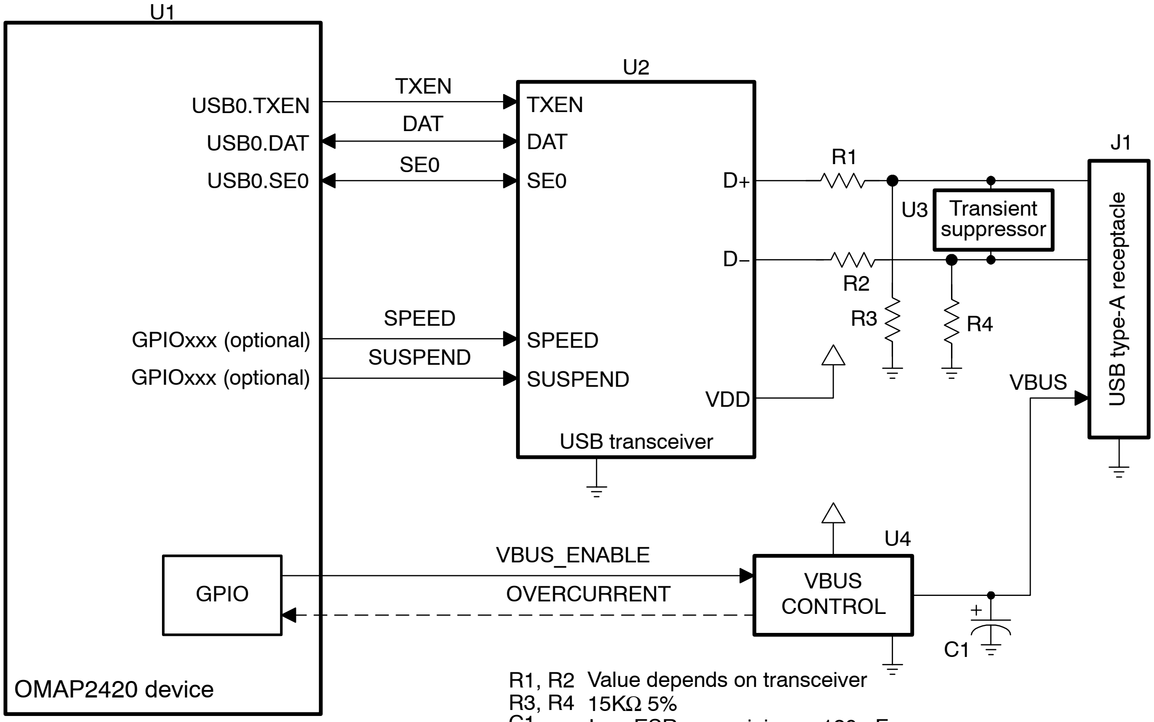
Some USB transceivers implement a 6-pin connection to the host controller, whereas others require only 4 pins or 3 pins. The OMAP2420 device must be properly configured to support the signaling required by the attached transceiver (see Section 22.2.1, *Selecting and Configuring USB Connectivity*).

It is possible to use OTG transceiver connectivity to provide a USB host-only port. The OTG transceiver meets all of the requirements of a USB upstream transceiver with the exception of VBUS current sourcing. OTG transceiver VBUS current is typically limited to a maximum value that is lower than the minimum required by the USB specification for a type-A receptacle. Systems that use an OTG transceiver to control a host-only type-A receptacle must provide some VBUS source that is capable of meeting USB specification current requirements. Systems that implement a standard type-A USB receptacle and use an OTG transceiver must ground the OTG transceiver ID input. If the OTG transceiver is used to monitor VBUS for a standard type-A USB receptacle, a series resistor is recommended between the OTG transceiver VBUS pin and the connector so that the OTG transceiver cannot drive any significant current onto VBUS.

Figure 22−6 through Figure 22−8 show the signal connectivity options for the external 3-, 4-, and -6 pin USB transceivers on the OMAP2420 USB port 0.

Host on USB Port 0 Using 3-Pin USB Transceiver

Figure 22−6. Host on USB Port 0 Using 3-Pin USBTransceiver

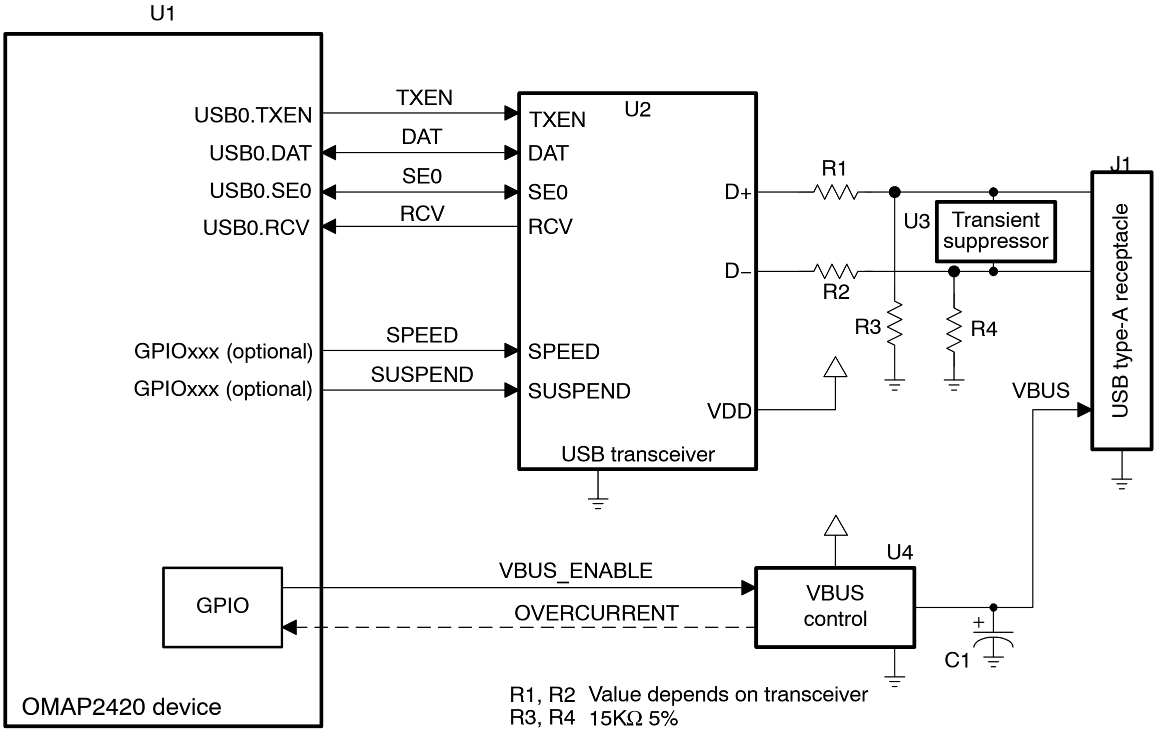


R1, R2	Value depends on transceiver
R3, R4	15KΩ 5%
C1	Low ESR cap, minimum 120 μF
U1	OMAP2420 device
U2	USB transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
U4	Power switch, like TPS2014 or TPS2015
J1	USB type-A receptacle

HMC_MODE must be set to a value that routes a host controller port to USB port 0 (see Table 22−2). The USB0_TRX_MODE bits OTG_SYSCON_1[18:16] must be set to 0x2 to allow proper bidirectional operation of the 3-pin USB transceiver. The USBx_SYNCHRO bit OTG_SYSCON_2[30] must be set to allow proper bidirectional signaling on pins USB0.DAT and USB0.SE0. The system control module USBT0WRMODEI bits CONTRL_DEVCONF[23:22] must be set to 0b10 to allow bidirectional transceiver interface mode.

Host on USB Port 0 Using 4-Pin USB Transceiver

Figure 22−7. Host on USB Port 0 Using 4-Pin USB Transceiver

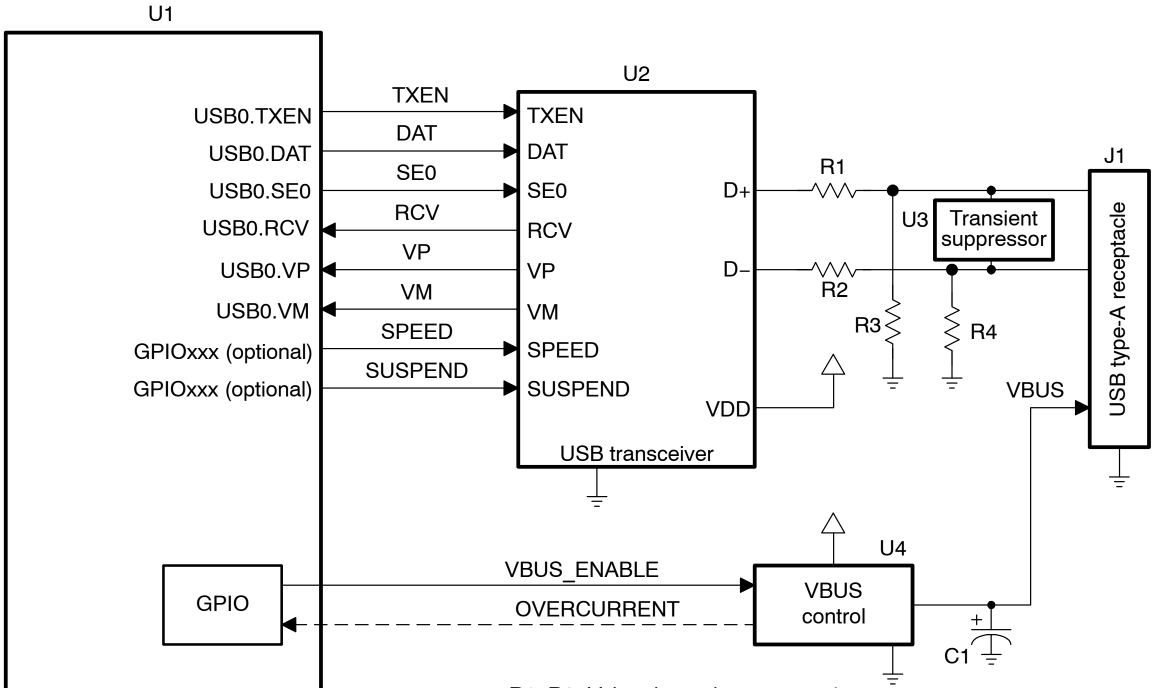


R1, R2	Value depends on transceiver
R3, R4	15KΩ 5%
C1	Low ESR cap, minimum 120 µF
U1	OMAP2420 device
U2	USB transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
U4	Power switch, like TPS2014 or TPS2015
J1	USB type-A receptacle

HMC_MODE must be set to a value that routes a host controller port to USB port 0 (see Table 22−2). The USB0_TRX_MODE bits OTG_SYSCON_1[18:16] must be set to 0x1 to allow proper bidirectional operation of the 4-pin USB transceiver. The USBx_SYNCHRO bit OTG_SYSCON_0[30] must be set to allow proper bidirectional signaling on pins USB0.DAT and USB0.SE0. The system control module USBT0WRMODEI bits CONTRL_DEVCONF[23:22] must be set to 0b10 to allow bidirectional transceiver interface mode.

Host on USB Port 0 Using 6-Pin USB Transceiver

Figure 22−8. Host on USB Port 0 Using 6-Pin USB Transceiver



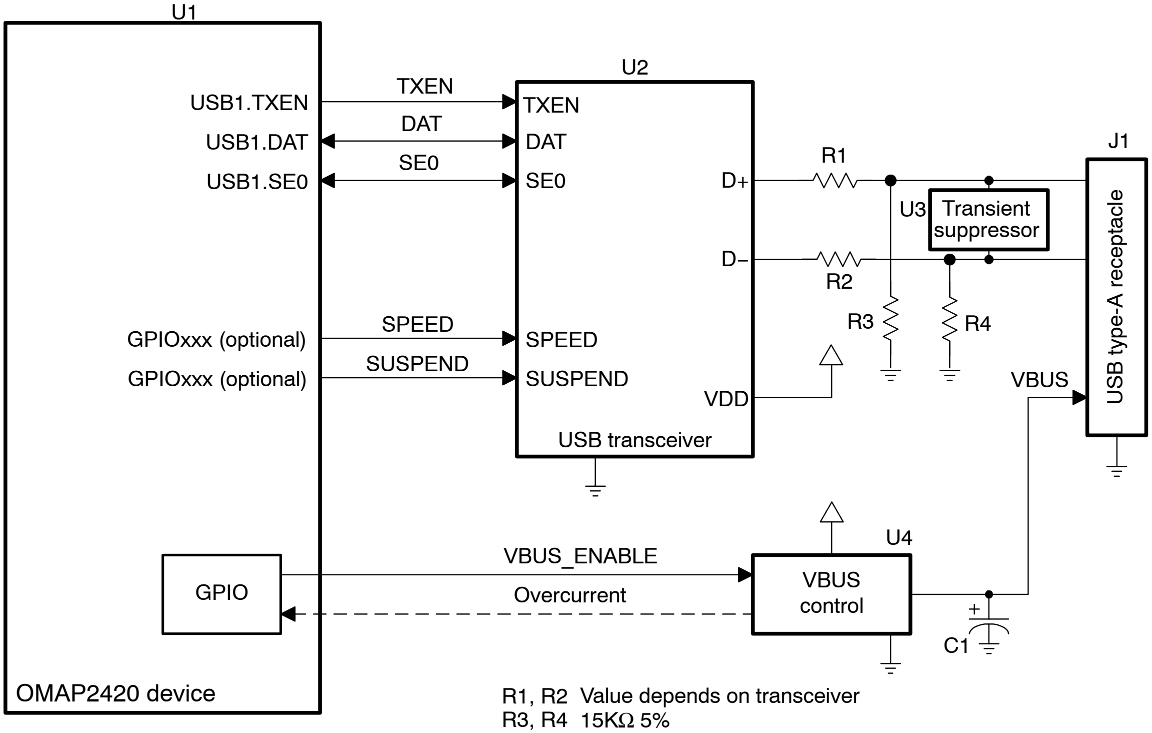
R1, R2	Value depends on transceiver
R3, R4	15KΩ 5%
C1	Low ESR cap, minimum 120 µF
U1	OMAP2420 device
U2	USB transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
U4	Power switch, like TPS2014 or TPS2015
J1	USB type-A receptacle

HMC_MODE must be set to a value that routes a host controller port to USB port 0 (see Table 22−2). The USB0_TRX_MODE bits OTG_ SYSCON_1[18:16] must be set to 0x3 to allow proper unidirectional operation of the 6-pin USB transceiver. The system control module USBT0WRMODEI bits CONTRL_DEVCONF[23:22] must be set to 0b00 to allow unidirectional transceiver interface mode.

Figure 22−9 and Figure 22−10 show the signal connectivity options for the external 3- and 4-pin USB transceivers on the OMAP2420 USB port 1.

Host on USB Port 1 Using 3-Pin USB Transceiver

Figure 22−9. Host on USB Port 1 Using 3-Pin USBTransceiver

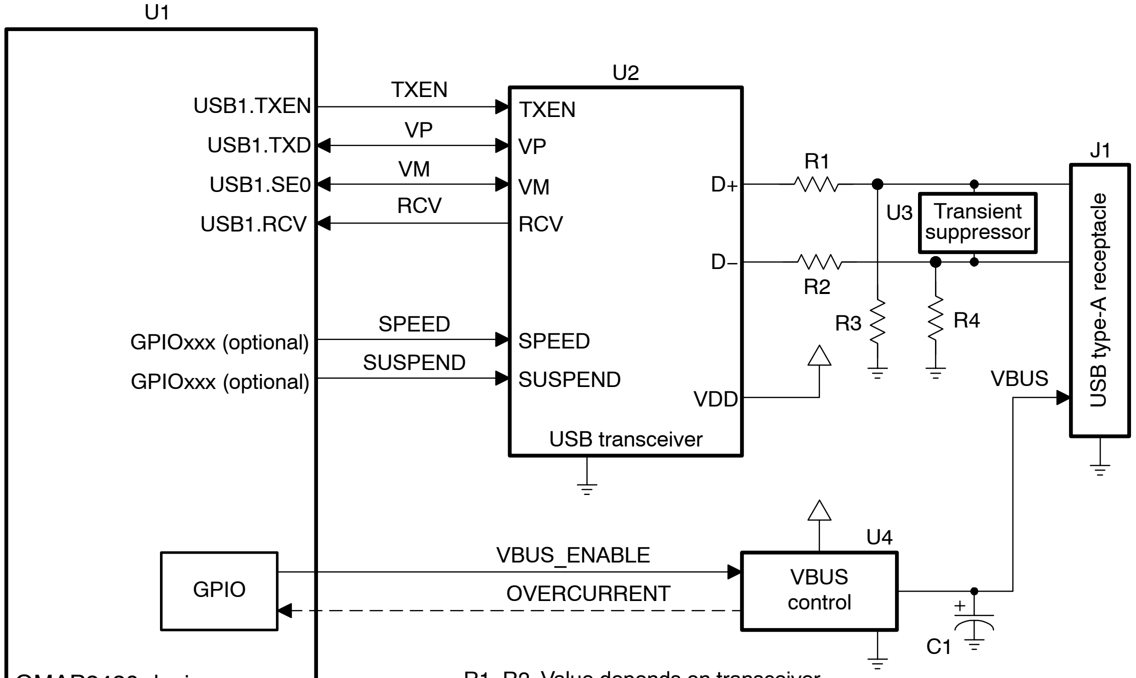


R1, R2	Value depends on transceiver
R3, R4	15KΩ 5%
C1	Low ESR Cap, minimum 120 µF
U1	OMAP2420 device
U2	USB transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
U4	Power switch, like TPS2014 or TPS2015
J1	USB type-A receptacle

HMC_MODE must be set to a value that routes a host controller port to USB port 1 (see Table 22−2). The USB1_TRX_MODE bits OTG_SYSCON_1[22:20] must be set to 0x2 to allow proper bidirectional operation of the 3-pin USB transceiver. The USBx_SYNCHRO bit OTG_SYSCON_2[30] must be set to allow proper bidirectional signaling on pins USB1.DAT and USB1.SE0. The system control module USBT1WRMODEI bits CONTRL_DEVCONF[21:20] must be set to 0b10 to allow bidirectional transceiver interface mode.

Host on USB Port 1 Using 4-Pin USB Transceiver

Figure 22−10. Host on USB Port 1 Using 4-Pin USB Transceiver



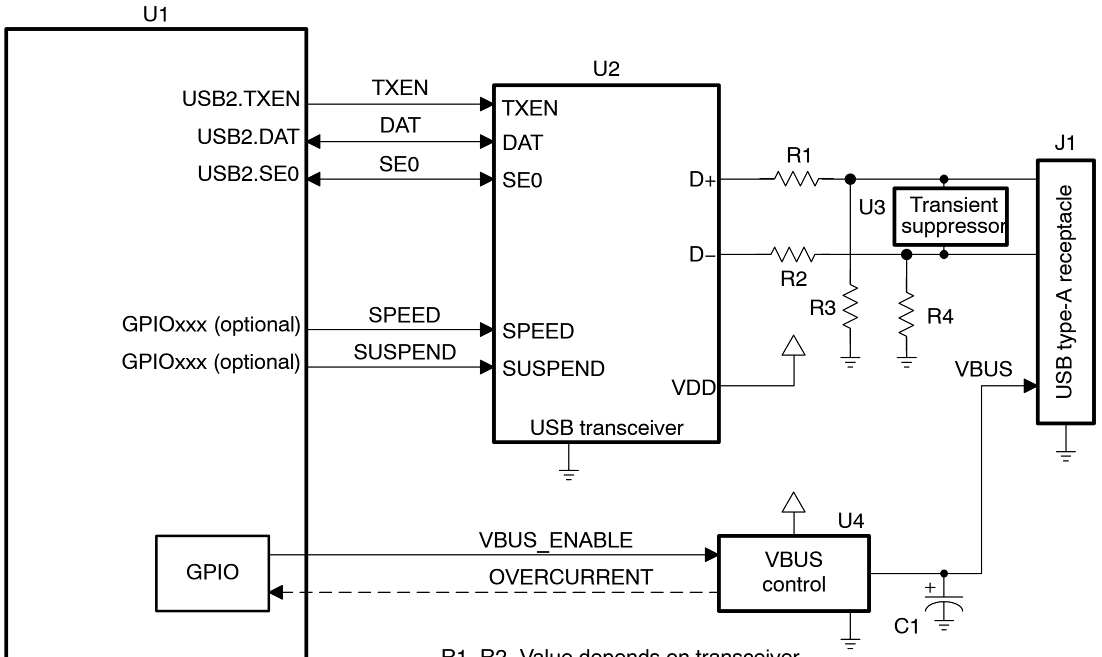
R1, R2	Value depends on transceiver
R3, R4	15KΩ 5%
C1	Low ESR cap, minimum 120 µF
U1	OMAP2420 device
U2	USB transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
U4	Power switch, like TPS2014 or TPS2015
J1	USB type-A receptacle

HMC_MODE must be set to a value that routes a host controller port to USB port 1 (see Table 22−2). The USB1_TRX_MODE bits OTG_SYSCON_1[22:20] must be set to 0x1 to allow proper bidirectional operation of the 4-pin USB transceiver. The USBx_SYNCHRO bit OTG_SYSCON_2[30] must be set to allow proper bidirectional signaling on pins USB1.DAT and USB1.SE0. The system control module USBT1WRMODEI bits CONTRL_DEVCONF[21:20] must be set to 0b10 to allow bidirectional transceiver interface mode.

Figure 22−11 and Figure 22−12 show the signal connectivity options for the external 3- and 4-pin USB transceivers on the OMAP2420 USB port 2.

Host on USB Port 2 Using 3-Pin USB Transceiver

Figure 22−11. Host on USB Port 2 Using 3-Pin USB Transceiver

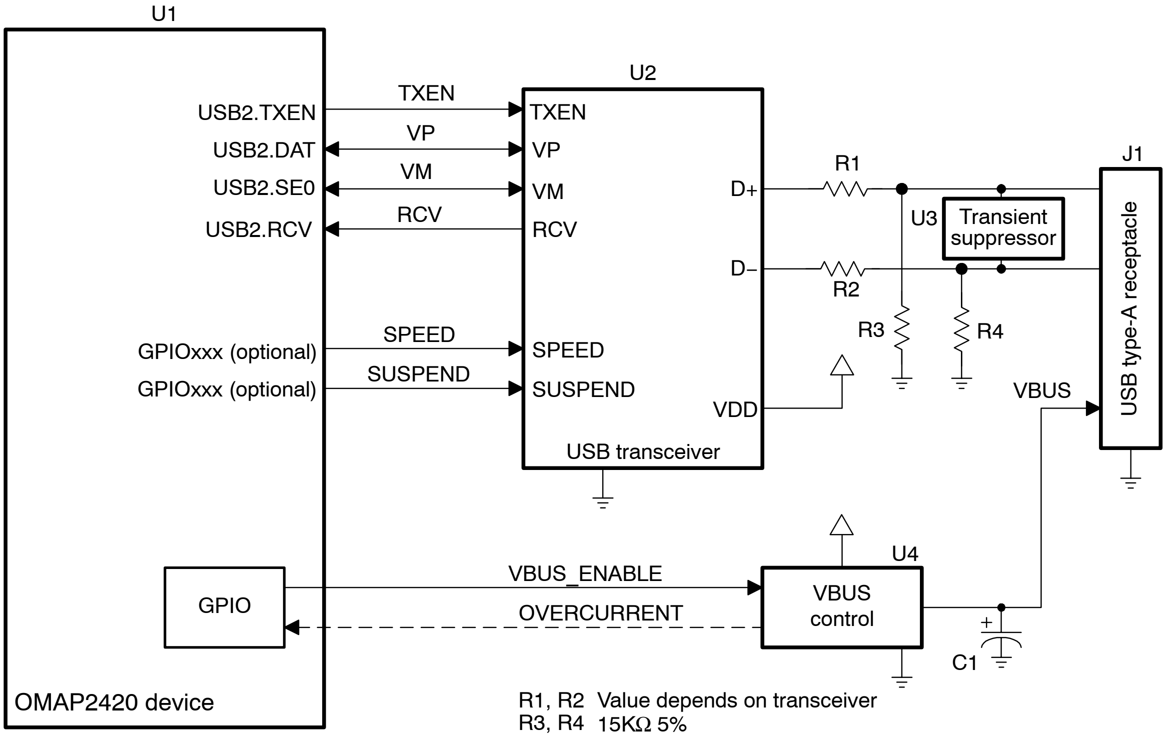


R1, R2	Value depends on transceiver
R3, R4	15KΩ 5%
C1	Low ESR cap, minimum 120 µF
U1	OMAP2420 device
U2	USB transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
U4	Power switch, like TPS2014 or TPS2015
J1	USB type-A receptacle

HMC_MODE must be set to a value that routes a host controller port to USB port 2 (see Table 22−2). The USB2_TRX_MODE bits OTG_SYSCON_1[26:24] must be set to 0x2 to allow proper bidirectional operation of the 3-pin USB transceiver. The USBx_SYNCHRO bit OTG_SYSCON_2[30] must be set to allow proper bidirectional signaling on pins USB2.DAT and USB2.SE0. The system control module USBT2WRMODEI bits CONTRL_DEVCONF[19:18] must be set to 0b10 to allow bidirectional transceiver interface mode.

Host on USB Port 2 Using 4-Pin USB Transceiver

Figure 22−12. Host on USB Port 2 Using 4-Pin USB Transceiver



R1, R2	Value depends on transceiver
R3, R4	15KΩ 5%
C1	Low ESR cap, minimum 120 µF
U1	OMAP2420 device
U2	USB transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
U4	Power switch, like TPS2014 or TPS2015
J1	USB type-A receptacle

HMC_MODE must be set to a value that routes a host controller port to USB port 2 (see Table 22−2). The USB2_TRX_MODE bits OTG_SYSCON_1[26:24] must be set to 1 to allow proper bidirectional operation of the 4-pin USB transceiver. The USBx_SYNCHRO bit OTG_SYSCON_2[30] must be set to allow proper bidirectional signaling on pins USB2.DAT and USB2.SE0. The system control module USBT2WRMODEI bits CONTRL_DEVCONF[19:18] must be set to 0b10 to allow bidirectional transceiver interface mode.

22.2.5 USB Device Controller Connectivity With USB Transceivers

To provide a robust USB solution, a system that provides a non-OTG USB device controller port must implement certain features:

- USB type-B or mini-B receptacle
- VBUS power detection
- Transient suppression
- Controllable pullup resistor to the D+ or D– line
- USB-compatible upstream port transceiver

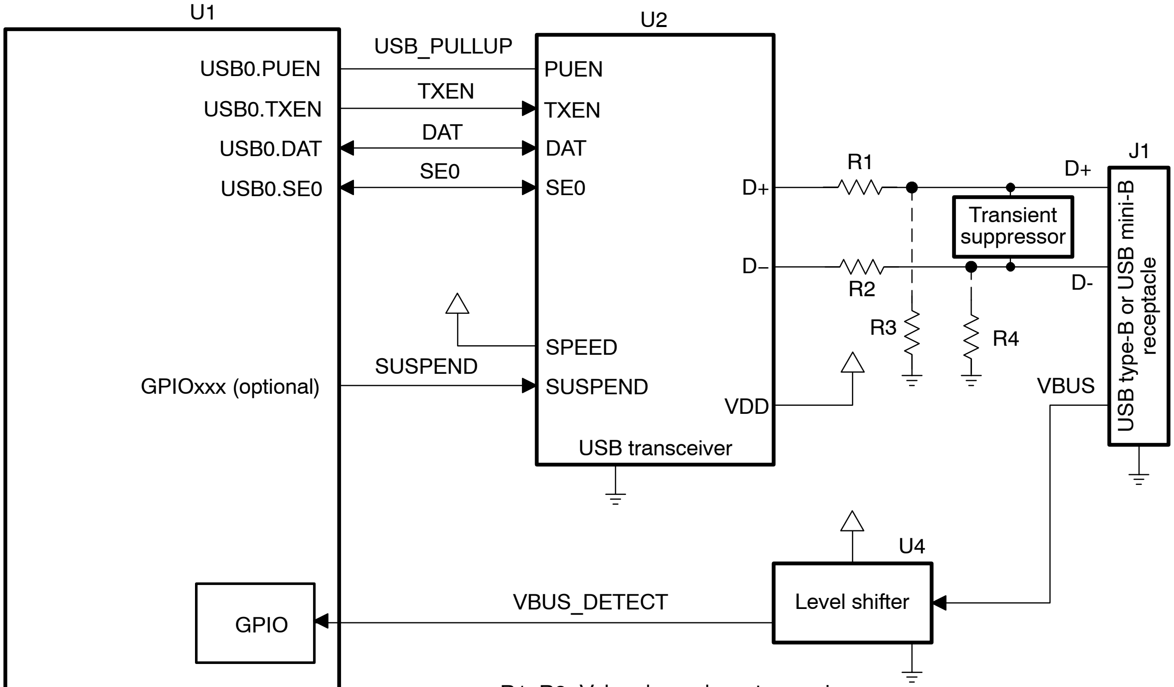
Optional weak pulldown resistors can be used to hold the D+ and D– signals at voltages below the USB transceiver VIL level when there is no USB host connected to the USB type B connector. By keeping D+ and D– voltages below VIL, the USB transceiver IDDQ can be reduced. The choice of value for the pulldown resistors must be made carefully so that the circuit meets the requirements of the USB specification.

The OMAP2420 USB device controller only supports implementation as a full-speed USB device. As such, the pullup resistor must be connected to the D+ signal to indicate implementation of a full-speed USB device.

Some USB transceivers implement a 6-pin connection to the device controller, whereas others require only 4 pins or 3 pins. Figure 22−13 through Figure 22−15 show the signal connectivity options for the external 3-, 4-, and 6-pin USB transceivers on the OMAP2420 USB port 0.

Device on USB Port 0 Using 3-Pin USB Transceiver

Figure 22−13. Device on USB Port 0 Using 3-Pin USB Transceiver



R1, R2	Value depends on transceiver
R3, R4	Optional weak pulldown (see text)
R5	1.5KΩ 5%
U1	OMAP2420 device
U2	USB transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
U4	Level shifter
J1	USB type-B receptacle or USB mini-B receptacle

HMC_MODE must be set to a value that routes the USB device controller to USB port 0 (see Table 22−2). The USB0_TRX_MODE bits OTG_SYSCON_1[18:16] must be set to 0x2 to allow proper bidirectional operation of the 3-pin USB transceiver. The system control module USBT0WRMODEI bits CONTRL_DEVCONF[23:22] must be set to 0b10 to allow bidirectional transceiver interface mode.

A USB OTG transceiver can provide USB device-only functionality instead of a USB transceiver, with appropriate I^{2}C and interrupt connectivity. The OTG_PADEN bit OTG_SYSCON_2[10] must be 0, and software passes VBUS validity information from the USB OTG transceiver to the device controller by reading the VBUS status via the I^{2}C link and updating the value sent to the device controller by using the BSESSVLD bit OTG_CTRL[18]. The OTG transceiver ID pin must be left unconnected, and OTG functionality is not available.

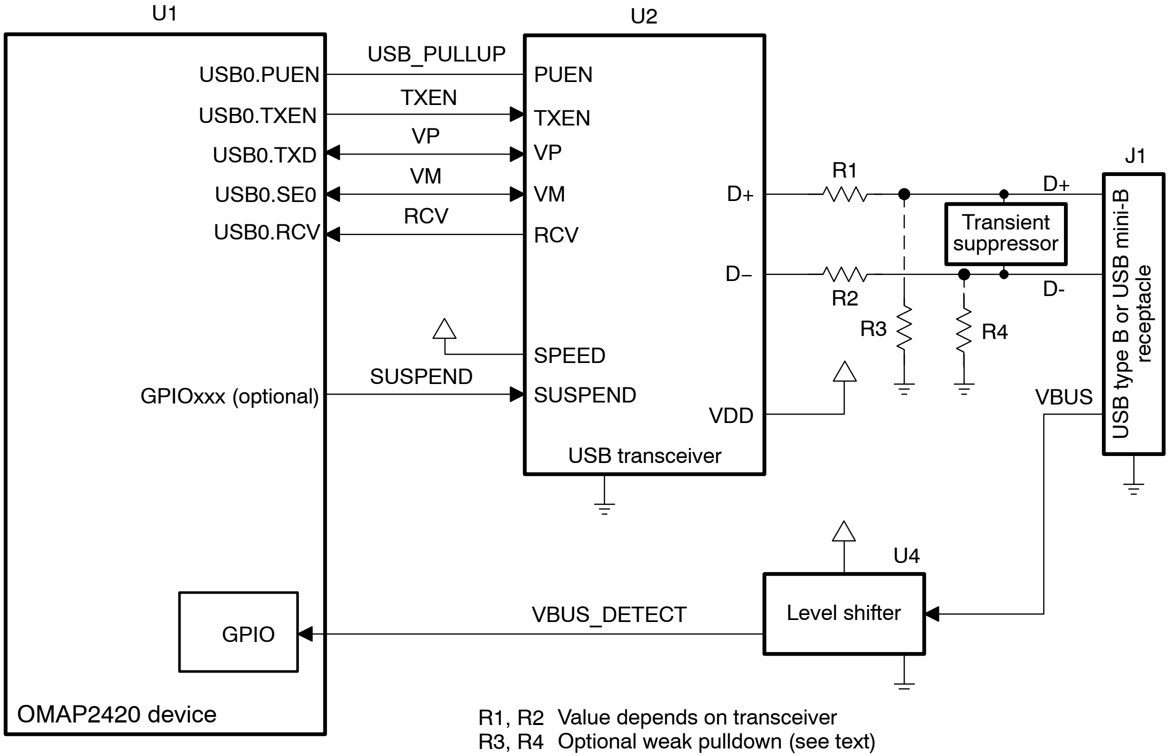
Because the OTG transceiver provides integrated D+ pullup capability, suspend mode control, and D+/D– pulldown controls, the hardware connections shown can be removed when an OTG transceiver is used. System software must then control those features using the OTG transceiver I^{2}C register set.

When OTG_PADEN bit OTG_SYSCON_2[10] is 0, the USB device controller sees the VBUS status only from the software-controlled BSESSVLD bit OTG_CTRL[18]. In this case, software must monitor the VBUS_DETECT signal (using GPIO if wired as shown), and update the BSESSVLD bit OTG_CTRL[18] whenever the VBUS_DETECT signal changes.

Figure 22−14 and Figure 22−15 show the signal connectivity options for the external 4- and 6-pin USB transceivers on OMAP2420 USB port 0.

Device on USB Port 0 Using 4-Pin USB Transceiver

Figure 22−14. USB Device on USB Port 0 Using 4-Pin USB Transceiver



R1, R2	Value depends on transceiver
R3, R4	Optional weak pulldown (see text)
R5	1.5KΩ 5%
U1	OMAP2420 device
U2	USB transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
U4	Level shifter
J1	USB type-B receptacle or USB mini-B receptacle

HMC_MODE must be set to a value that routes the USB device controller to USB port 0 (see Table 22−2). The USB0_TRX_MODE bits OTG_SYSCON_1[18:16] must be set to 0x1 to allow proper bidirectional operation of the 4-pin USB transceiver. The system control module USBT0WRMODEI bits CONTRL_DEVCONF[23:22] must be set to 0b10 to allow bidirectional transceiver interface mode.

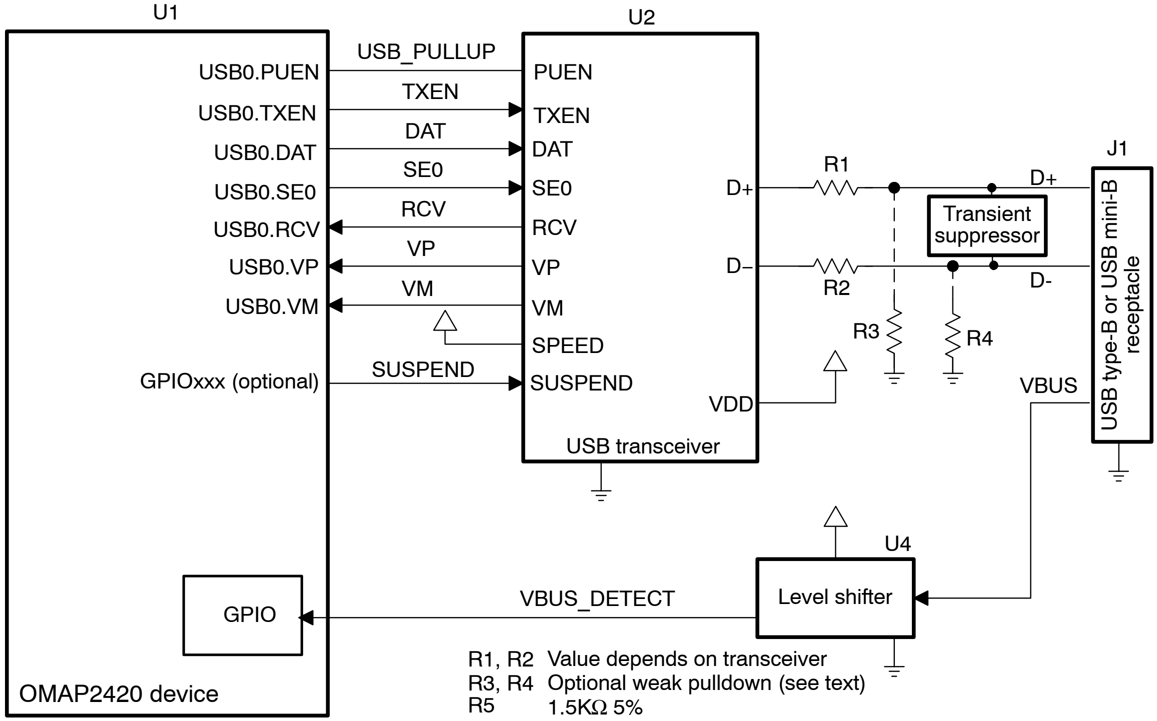
A USB OTG transceiver can provide USB device-only functionality instead of a USB transceiver, with appropriate I^2C and interrupt connectivity. The OTG_PADEN bir OTG_SYSCON_2[10] must be 0, and software passes VBUS validity information from the USB OTG transceiver to the device controller by reading VBUS status via the I^{2}C link and updating the value sent to the device controller by using the BSESSVLD bit OTG_CTRL[18]. The OTG transceiver ID pin is left unconnected, and OTG functionality is not available.

Because the OTG transceiver provides integrated D+ pullup capability, suspend mode control, and D+/D− pulldown controls, the hardware features shown can be removed when an OTG transceiver is used. System software must control those features through the OTG transceiver I^{2}C register set.

When the OTG_PADEN bit OTG_SYSCON_2[10] is 0, the USB device controller sees the VBUS status only from the software-controlled BSESSVLD bit OTG_CTRL[18]. In this case, software must monitor the VBUS_DETECT signal (using GPIO if wired as shown), and update the BSESSVLD bit OTG_CTRL[18] whenever the VBUS_DETECT signal changes.

Device on USB Port 0 Using 6-Pin USB Transceiver

Figure 22−15. Device on USB Port 0 Using 6-Pin USB Transceiver



R1, R2	Value depends on transceiver
R3, R4	Optional weak pulldown (see text)
R5	1.5KΩ 5%
U1	OMAP2420 device
U2	USB transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
U4	Level shifter
J1	USB type-B receptacle or USB mini-B receptacle

HMC_MODE must be set to a value that routes the USB device controller to USB port 0 (see Table 22−2). The USB0_TRX_MODE bits OTG_SYSCON_1[18:16] must be set to 0x3 to allow proper unidirectional operation of the 6-pin USB transceiver. The system control module USBT0WRMODEI bits CONTRL_DEVCONF[23:22] must be set to 0b00 to allow unidirectional transceiver interface mode.

A USB OTG transceiver can provide USB device-only functionality instead of a USB transceiver, with appropriate I^{2}C and interrupt connectivity. The OTG_PADEN bit OTG_SYSCON_2[10] must be 0, and software passes VBUS validity information from the USB OTG transceiver to the device controller (USB_OTG) by reading VBUS status via the I^{2}C link and updating the value sent to the device controller by using the BSESSVLD bit OTG_CTRL[18]. The OTG transceiver ID pin is left unconnected, and OTG functionality is not available.

Because the OTG transceiver provides integrated D+ pullup capability, suspend mode control, and D+/D– pulldown controls, the hardware features shown can be removed when an OTG transceiver is used. System software must control those features through the OTG transceiver I^{2}C register set.

When the OTG_PADEN bit OTG_SYSCON_2[10] is 0, the USB device controller sees the VBUS status only from the software-controlled BSESSVLD bit OTG_CTRL[18]. In this case, software must monitor the VBUS_DETECT signal (using GPIO if wired as shown), and update the BSESSVLD bit OTG_CTRL[18] whenever the VBUS_DETECT signal changes.

22.2.6 Onboard Transceiverless Connection Using OMAP2420 Transceiverless Link

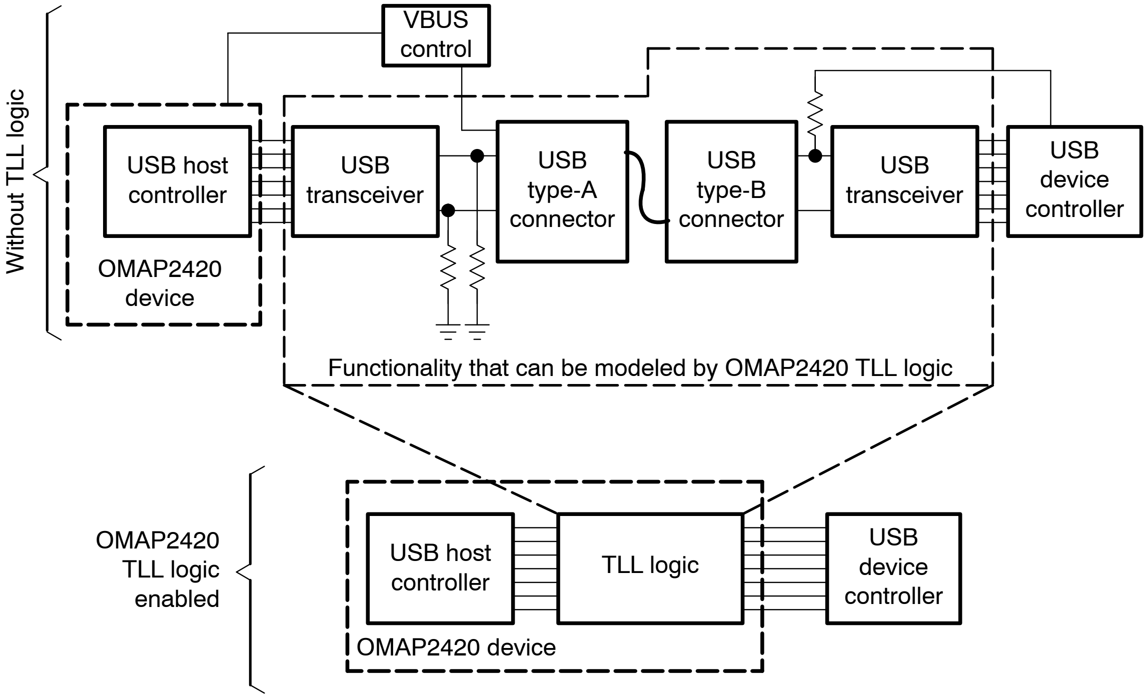
The TLL logic feature of the OMAP2420 USB signal multiplexing enables connection of the OMAP2420 USB host controller without the use of USB transceivers or associated circuitry. When TLL logic is used, both USB transceivers, the series resistors, pullup and pulldown resistors, VBUS switching components, and USB connectors and cables that typically are used between a USB host controller and the downstream USB device controller are removed. The OMAP2420 TLL logic feature does not support OTG session request protocol or OTG host negotiation protocol, so it cannot be used for a transceiverless OTG link.

The TLL logic signaling system is not suitable for use across a cable. It is intended only for use when the OMAP2420 device is used with an external USB integrated circuit that is on the same board.

When using the TLL logic, signals of the external USB integrated circuit pins that typically connect to a USB transceiver connect instead directly to OMAP2420 device pins. Signaling on these pins uses CMOS levels. TLL logic can be used with external devices that support 6-, 4-, and 3-pin transceiver connectivity.

Figure 22–16 shows how TLL logic can be compared to a typical USB implementation. It shows the OMAP2420 device used as a USB host controller, with the top portion of the diagram showing a transceiver-based solution, and the bottom portion showing a transceiverless solution using the OMAP2420 TLL logic.

Figure 22−16. OMAP2420 USB Host Controller Connection—With and Without OMAP2420 TLL Logic



The TLL logic function in the OMAP2420 device interprets the transmit control signals from the external USB integrated circuit and similar signals from the OMAP2420 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 integrated circuit and to the OMAP2420 USB host controller.

Two control bits help determine the proper differential pair state. The HMC_TLLATTACH bit OTG_SYSCON_2[6] and the HMC_TLLSPEED bit OTG_SYSCON_2[7] provide these inputs. HMC_TLLATTACH is used to control when the differential pair models a pullup resistor as is implemented in a transceiver-based USB device. HMC_TLLSPEED is used solely to determine whether the modeled pullup resistor is modeled on the internal version of D+ (for a full-speed TLL) or D– (for a low-speed TLL).

The TLL logic cannot be used as part of an OTG connection to an on-board OTG device. An OTG connection always requires an OTG transceiver. Some modes of operation simultaneously support a transceiver-based OTG connection and an additional TLL logic connection to an on-board downstream USB device. The OMAP2420 device does not provide any modes where both a TLL logic connection to an on-board USB host and a transceiver-based OTG connection are simultaneously available.

22.2.6.1 Bidirectional TLL Modes

Bidirectional TLL modes are implemented using existing bidirectional modes. Those TLL modes do not use the TLL functions of the USB core set by the HMC mode. Their goal is to reduce pin count for 6- and 5-pin TLL modes.

Pullup/pulldown resistors on the bidirectional lines are now compulsory, because the lines are not constantly driven from one or the other side of the interface. In this case, the pullup on one of the two lines triggers the connect signal from the device to the host. The pullup can be controlled (by the USB device side) or hardwired: in the first case, the USB host sees the external device only if the pullup is active; in the second case, the USB host sees the device as soon as the software enables the port in question.

The general rule is that the transceiver interface must reflect the following USB bus data line states:

- ☐ SE0 when the device is disconnected
- ☐ J (idle) when the device is connected. The value of J depends on the USB speed: differential 1 (D+/D– = (1/0)) in full-speed; differential 0 (D+/D– = (0/1)) in low-speed.

Note:

There is always a gap between two driven times (no collisions). Driven-to-nondriven transitions always occur in the idle electrical state (J) because all packets must end with a bit of driven idle (J) before going back to nondriven idle. Therefore, the pullup must be only strong enough to keep the default value on the line; the pullup/down is not required to be fast (that is, strong) enough to drive transitions within USB timing constraints.

Figure 22–17 shows the host on USB port 1 using DP/DM encoding with a standard 4-pin bidirectional USB device.

Host on USB Port 1 Using Bidirectional 2-Pin TLLMode

Figure 22−17. Host on USB Port 1 Using DP/DM Encoding,With Standard 4-Pin Bidirectional USB Device

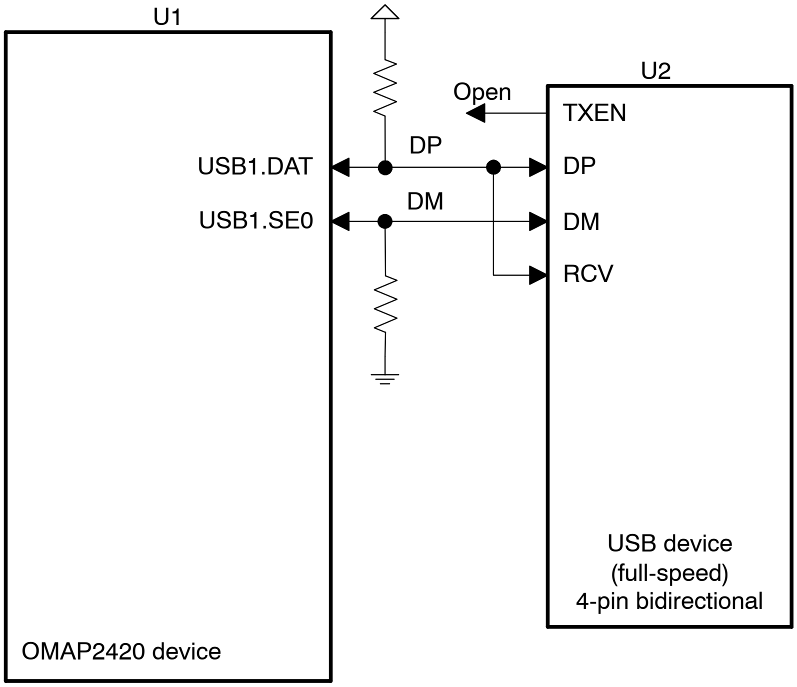


Table 22−11. Pullup/Pulldown Configuration for DP/DM Encoding

	Nonconnected Device (any speed)	**Connected Low-Speed Device**	**Connected Full-Speed Device**
DP	Pulldown	Pulldown	Pullup
DM	Pulldown	Pullup	Pulldown

HMC_MODE must be set to a value that routes a host controller to USB port 1 (see Table 22−2.) The USB1_TRX_MODE bits OTG_SYSCON_1[22:20] must be set to 0x1 to allow proper bidirectional operation of the 4-pin USB transceiver. The system control module USBT1WRMODEI bits CONTROL_DEVCONF[21:20] must be set to 0b11 to allow bidirectional TLL transceiver interface mode.

Figure 22−18 shows the host on USB port 1 using DAT/SE0 encoding with a standard 3-pin bidirectional USB device.

Figure 22−18. Host on USB Port 1 Using DAT/SE0 Encoding, With Standard 3-Pin Bidirectional USB Device

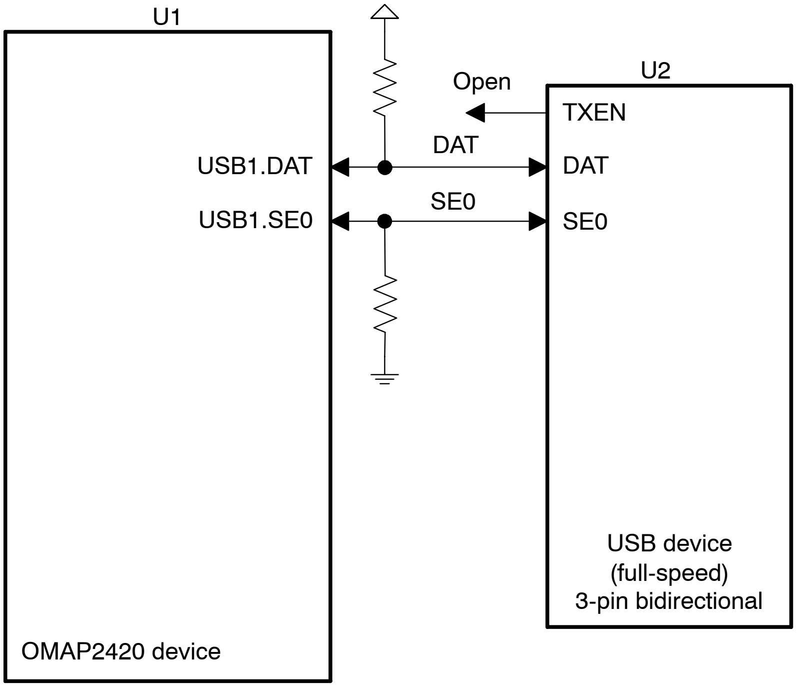


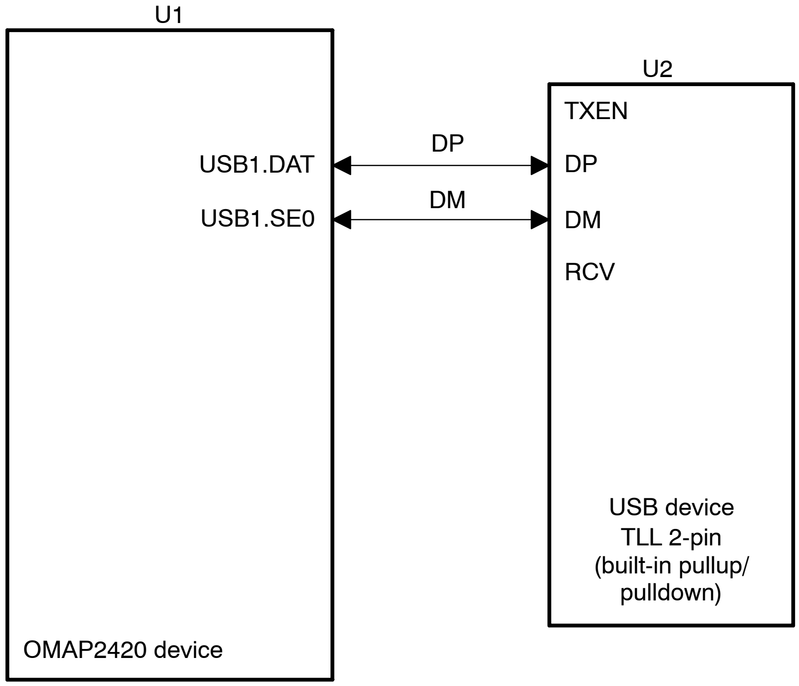
Table 22−12. Pullup/Pulldown Configuration for DAT/SE0 Encoding

	Nonconnected Device (any speed)	Connected Low-Speed Device	Connected Full-Speed Device
DAT	Pulldown	Pulldown	Pullup
SEO	Pullup	Pulldown	Pulldown

HMC_MODE must be set to a value that routes a host controller to USB port 1 (see Table 22−2). The USB1_TRX_MODE bits OTG_SYSCON_1[22:20] must be set to 0x2 to allow proper bidirectional operation of the 3-pin USB transceiver. The system control module USBT1WRMODEI bits CONTROL_DEVCONF[21:20] must be set to 0b10 to allow bidirectional transceiver interface mode.

It is also possible to implement a specific 2-pin TLL USB device with no other pins than the two bidirectional pins. Figure 22−19 shows DP/DM mode with built-in pullup/pulldown resistors.

Figure 22−19. Host on USB Port 1 Using DP/DM Encoding, With Specific 2-Pin Bidirectional USB Device



Host on USB Port 2 Using Bidirectional 2-Pin TLL Mode

Figure 22−20. Host on USB Port 2 Using DP/DM Encoding, With Standard 4-Pin Bidirectional USB Device

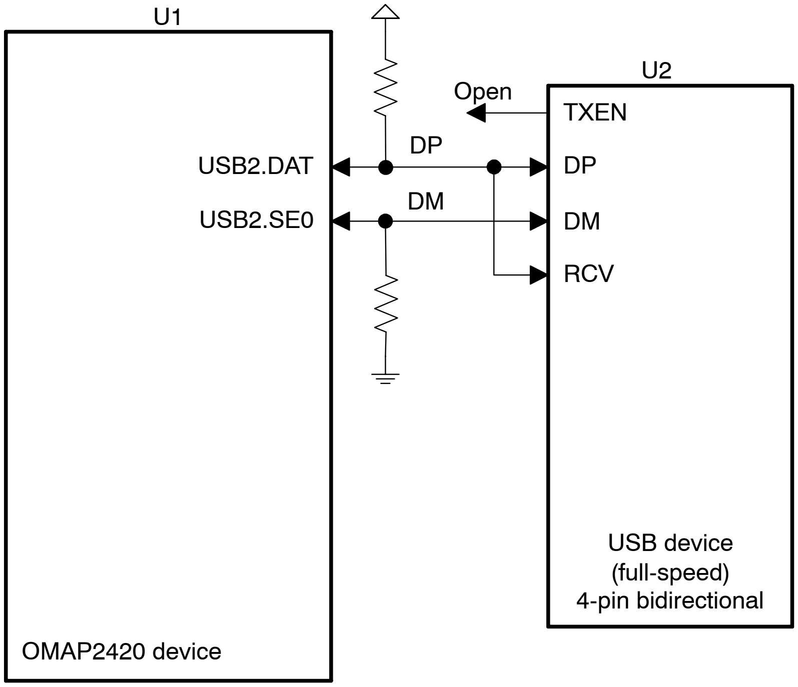


Table 22−13. Pullup/Pulldown Configuration for DP/DM Encoding

	Nonconnected Device (any speed)	**Connected Low-Speed Device**	**Connected Full-Speed Device**
DP	Pulldown	Pulldown	Pullup
DM	Pulldown	Pullup	Pulldown

HMC_MODE must be set to a value that routes a host controller to USB port 2 (see Table 22−2). The USB2_TRX_MODE bits OTG_SYSCON_1[26:24] must be set to 0x1 to allow proper bidirectional operation of the 4-pin USB transceiver. The system control module USBT2WRMODEI bits CONTROL_DEVCONF[19:18] must be set to 0b11 to allow bidirectional TLL transceiver interface mode. The system control module USBT2TLL5PI bit CONTROL_DEVCONF[17] must be set to 0 to disable 5-pin TLL mode.

Figure 22−21 shows the host on USB port 2 using DAT/SE0 encoding and a standard 3-pin bidirectional USB device.

Figure 22−21. Host on USB Port 2 Using DAT/SE0 Encoding, With Standard 3-Pin Bidirectional USB Device

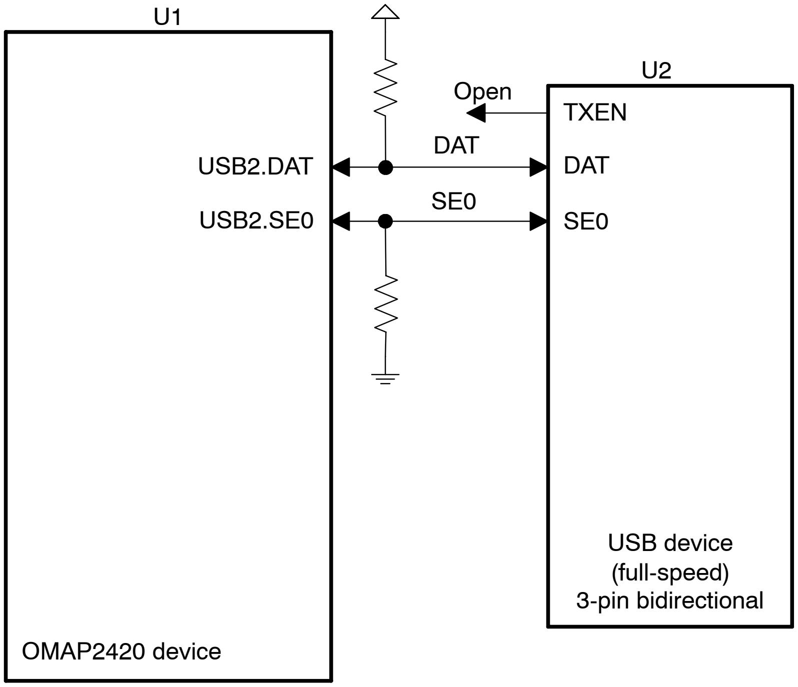


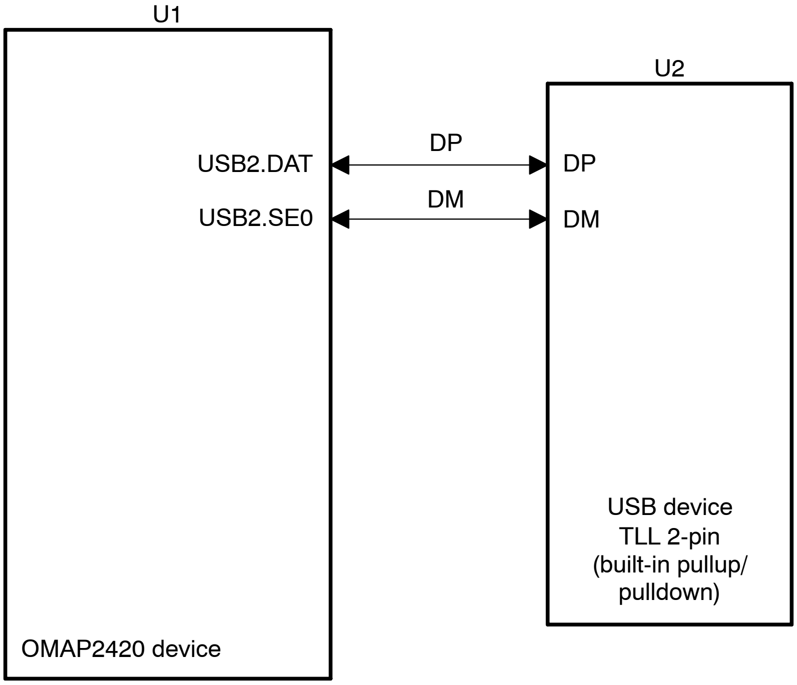
Table 22−14. Pullup/Pulldown Configuration for DAT/SE0 Encoding

	Nonconnected Device (any speed)	**Connected Low-Speed Device**	**Connected Full-Speed Device**
DAT	Pulldown	Pulldown	Pullup
SEO	Pullup	Pulldown	Pulldown

HMC_MODE must be set to a value that routes a host controller to USB port 2 (see Table 22−2). The USB2_TRX_MODE bits OTG_SYSCON_1[26:24] must be set to 0x2 to allow proper bidirectional operation of the 3-pin USB transceiver. The system control module USBT2WRMODEI bits CONTROL_DEVCONF[19:18] must be set to 0b10 to allow bidirectional transceiver interface mode. The system control module USBT2TLL5PI bit CONTROL_DEVCONF[17] must be set to 0 to disable 5-pin TLL mode.

It is also possible to implement a specific 2-pin TLL USB device with no other pins than the two bidirectional pins. Figure 22−22 shows the DP/DM mode with built-in pullup/pulldown resistors.

Figure 22−22. Host on USB Port 2 Using DP/DM Encoding, With Specific 2-Pin Bidirectional USB Device



22.2.6.2 Unidirectional TLL Mode

Unidirectional 5-pin TLL modes are implemented inside the USB controller, and are restricted to USB port 2 (host-only). They are used to connect directly to an external USB device with the appropriate unidirectional interface. The TLL is selected by choosing the appropriate mode in the HMC_MODE bits OTG_SYSCON_2[5:0] (see Table 22−2). The 5-pin TLL mode, only available for USB port 2, is enabled by the system control module USBT2TLL5PI bit CONTROL_DEVCONF[17].

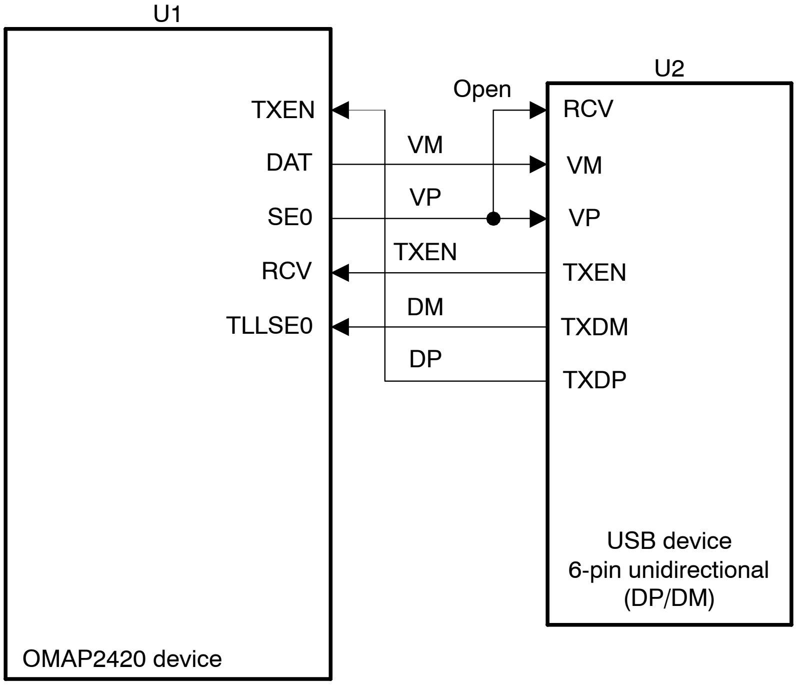
In unidirectional 5-pin TLL mode, the attachment and the speed of the device (pullup) can be emulated by the glue logic in the USB controller. This glue logic is set by the HMC_TLLSPEED bit OTG_SYSCON_2[7] and the HMC_TLLATTACH bit OTG_SYSCON_2[6].

Two possible 5-pin TLL modes exist, depending on the TX data encoding used by the external device; TX encoding can be selected to be DP/DM or DAT/SE0 (see Figure 22−23 and Figure 22−24, respectively).

The selection between the two is done by setting the system control module USBT2WRMODEI bits CONTROL_DEVCONF[19:18] to either Unidir (0b00) or Unidir TLL (0b01).

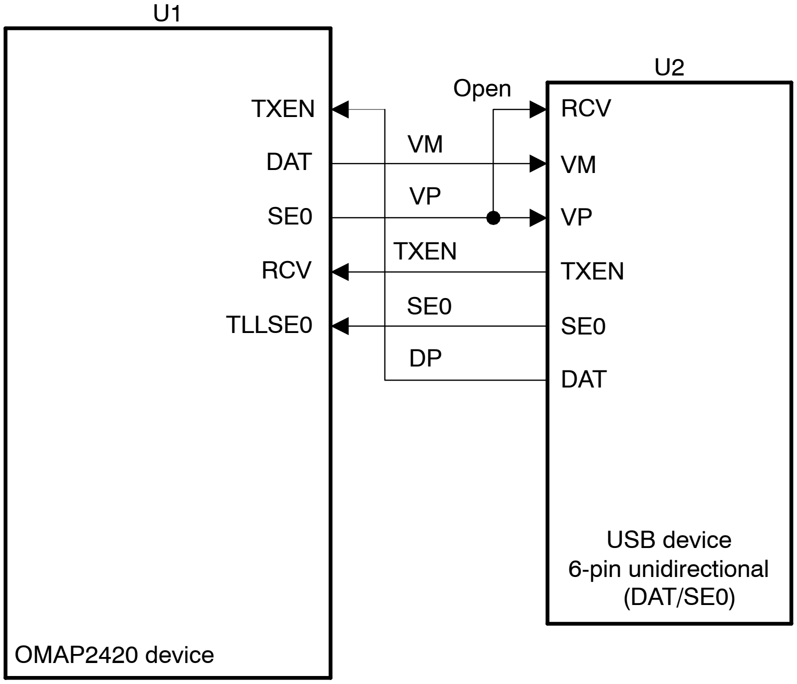
Host on USB Port 2 Using Unidirectional 5-Pin TLL Mode

Figure 22−23. Host on USB Port 2 Using DP/DM Encoding, With Standard 6-Pin Unidirectional USB Device



HMC_MODE must be set to a value that routes a host controller to USB port 2 (see Table 22−2). The USB2_TRX_MODE bits OTG_SYSCON_1[26:24] must be set to 0x3 to allow proper bidirectional operation of the 6-pin USB transceiver. The system control module USBT2WRMODEI bits CONTROL_DEVCONF[19:18] must be set to 0b01 to allow unidirectional TLL transceiver interface mode. The system control module USBT2TLL5PI bit CONTROL_DEVCONF[17] must be set to enable 5-pin TLL mode.

Figure 22−24. Host on USB Port 2 Using DAT/SE0 Encoding, With Standard 6-Pin Unidirectional USB Device



HMC_MODE must be set to a value that routes a host controller to USB port 2 (see Table 22−2). The USB2_TRX_MODE bits OTG_SYSCON_1[26:24] must be set to 0x3 to allow proper bidirectional operation of the 6-pin USB transceiver. The system control module USBT2WRMODEI bits CONTROL_DEVCONF[19:18] must be set to 0b00 to allow unidirectional transceiver interface mode. The system control module USBT2TLL5PI bit CONTROL_DEVCONF[17] must be set to enable 5-pin TLL mode.

22.2.7 Conflicts Between USB Signal Multiplexing and Top-Level Multiplexing

When OMAP2420 top-level signal multiplexing selects non-USB functionality for a pin but USB signal multiplexing is set to use that pin as an output, the signal from the USB signal multiplexing is ignored and the source selected by the OMAP2420 top-level signal multiplexing is used.

When OMAP2420 top-level signal multiplexing selects non-USB functionality for a pin but USB signal multiplexing is set to use that pin as an input, the OMAP2420 top-level signal multiplexing presents a low level to the USB signal multiplexer.

It may be useful to select an HMC_MODE value that brings some USB signals to the OMAP2420 top-level signal multiplexing, but then set the top-level signal multiplexing to ignore those USB signals.

22.2.8 OMAP2420 USB Hardware Considerations

22.2.8.1 VBUS Power Switching for USB Type A Host Receptacles

The USB specification places several VBUS requirements on USB hosts, including current capability, droop, and other important characteristics. Circuits that meet the USB specification requirements can be implemented using Texas Instruments devices, such as the TPS2014 and TSP2015 power-distribution switch devices. For further information, see the TI website.

Although an OTG transceiver can power VBUS within the OTG specification supplement limits for an OTG DRD, it may not have sufficient current sourcing capability to meet the USB 1.1 specification, which requires far greater output current on VBUS. If an OTG downstream port must also support a downstream non-OTG USB device (using a standard-A receptacle to a mini-A plug as defined in the OTG specification supplement), a VBUS switch as mentioned previously can be used to meet those requirements.

22.2.8.2 Transient Suppression for USB Connectors

It is important to provide transient suppression for USB connectors. Electrostatic discharge that occurs when you connect or disconnect a USB cable can have a dramatic effect on a system if not suppressed. TI offers several devices for transient suppression on USB connections, such as the SN65220, SN65240, and SN75240 USB port transient suppressor devices. For further information, see the TI website.

22.2.8.3 VBUS Monitoring for USB Device Controller

A USB device controller must be able to monitor the VBUS voltage provided by the upstream USB host controller. An external signal level shifter is required to convert the VBUS signal range to a range suitable for the OMAP2420 device.

22.2.8.4 USB D+, D– Pulldown for USB Device Controller

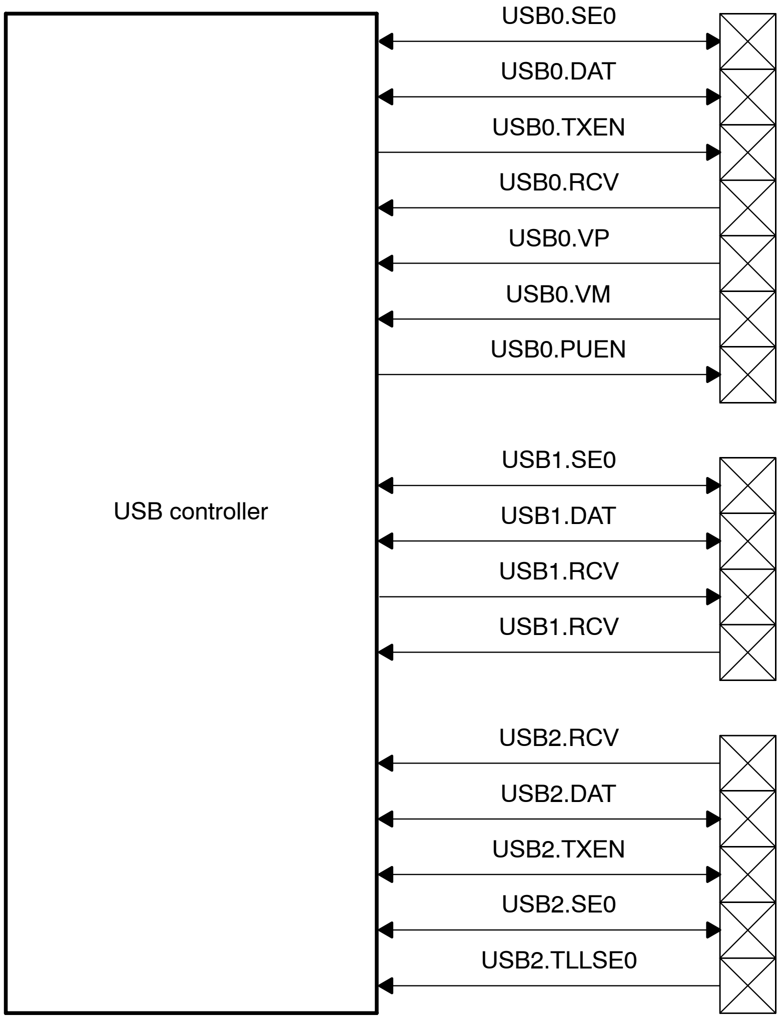
System implementations that use the OMAP2420 USB device controller and are sensitive to supply current requirements can implement weak pulldown resistors on the USB D+ and D– signals associated with the USB type-B receptacle. When there is no host controller attached upstream of the USB type-B receptacle, the undriven D+ and D–wires can float to voltages that cause excessive current consumption by the USB transceiver. Weak pulldowns can help prevent this problem. Selection of pulldown resistors depends on transceiver characteristics, D+ pullup resistor implementation, and board layout, and must be designed to meet USB D+ and D– signal requirements.

22.2.9 USB Controller Functional Interfaces

22.2.9.1 Basic USB Controller Pins

Figure 22−25 shows the USB controller functional interface.

Figure 22−25. USB Controller Functional Interface Signals



22.2.9.2 USB Controller Interface Description

Table 22−15 provides the I/O description.

Table 22−15. I/O Description

Signal Name	I/O	Description	Value at Reset
USB port 0 (device, host or OTG)			
Interface operating modes: 3- or 4-pin bidirectional or 6-pin unidirectional			
USB0.SE0	I/O	SE0 function in 3-pin bidirectional DAT/SE0 mode	Unknown
	I/O	VM function in 4-pin bidirectional VP/VM mode	
	O	SE0 output in 6-pin unidirectional mode	
USB0.DAT	I/O	DAT function in 3-pin bidirectional DAT/SE0 mode	Unknown
	I/O	VP function in 4-pin bidirectional VP/VM mode	
	O	Transmit data in 6-pin unidirectional mode	
USB0.TXEN	O	Transmit enable	1
USB0.RCV	I	Differential receiver signal input (Not used in the 3-pin bidirectional DAT/SE0 mode)	Unknown
USB0.VP	I	Single-ended DP receiver signal input (Not used in the 3-pin bidirectional DAT/SE0 or 4-pin bidirectional VP/VM modes)	Unknown
USB0.VM	I	Single-ended DM receiver signal input (Not used in the 3-pin bidirectional DAT/SE0 or 4-pin bidirectional VP/VM modes)	Unknown
USB0.PUEN	O	USB device pullup enable control	USB0PUENACTLOI
USB port 1 (host only)			
Interface operating modes: 3- or 4-pin bidirectional or TLL in 2-/4-pin (bidirectional)			
USB1.SE0	I/O	SE0 function in 3-pin bidirectional DAT/SE0 mode	0
		VM function in 4-pin bidirectional VP/VM mode	
		DP-TLL/VM in 2-/4-pin TLL mode	
USB1.DAT	I/O	DAT function in 3-pin bidirectional DAT/SE0 mode	0
		VP function in 4-pin bidirectional VP/VM mode	
		DM-TLL/VP in 2-/4-pin TLL mode	
USB1.TXEN	O	Transmit enable in the 3-pin bidirectional DAT/SE0 or 4-pin bidirectional VP/VM modes (Not used in 2-/4-pin TLL mode)	1
USB1.RCV	I	Differential receiver signal input (Not used in the 3-pin bidirectional DAT/SE0 mode or 2-/4-pin TLL mode)	Unknown

I = Input, O = Output

Table 22−15. I/O Description (Continued)

Signal Name	I/O	Description	Value at Reset
USB port 2 (host only)			
Interface operating modes: 3- or 4-pin bidirectional or TLL in 2-/4-pin (bidirectional) or TLL 5-pin (unidirectional)			
USB2.SE0	I/O	SE0 function in 3-pin bidirectional DAT/SE0 mode	0
	I/O	VM function in 4-pin bidirectional VP/VM mode	
	I/O	DP-TLL/VP in 2-/4-pin TLL mode	
	O	VP-TLL/RCV-TLL output in 5-pin TLL mode	
USB2.DAT	I/O	DAT function in 3-pin bidirectional DAT/SE0 mode	0
	I/O	VP function in 4-pin bidirectional VP/VM mode	
	I/O	DM-TLL/VP in 2-/4-pin TLL mode	
	O	VM-TLL output in 5-pin TLL mode	
USB2.TXEN	O	Transmit enable in the 3-pin bidirectional DAT/SE0 or 4-pin bidirectional VP/VM modes (not used in 2-/4-pin TLL mode)	1
	I	DAT-TLL input in 5-pin TLL mode	
USB2.RCV	I	Differential receiver signal input (Not used in the 3-pin bidirectional DAT/SE0 mode or 2-/4-pin TLL mode)	Unknown
	I	TXEN/TLL input in 5-pin TLL mode	
USB2.TLLSE0	I	SE0-TLL input in 5-pin TLL mode (not used in the 3-pin bidirectional DAT/SE0, 4-pin bidirectional VP/VM or 2-/4-pin TLL modes)	Unknown

I = Input, O = Output

22.3 USB Controller Integration

22.3.1 Description

The USB controller module (USB) is connected to the L3 interconnect initiator and target interfaces, and the L4 interconnect. The USB host controller uses the L3 interconnect to generate data traffic within the OMAP2420 device. The USB device controller uses the L4 interconnect, which is a configuration port for register setting.

Six interrupts, M_IRQ[75:80] (USB_IRQ), and six DMA requests, S_DMA[54:59] (USB0_DMA), are connected to the system (see Figure 22−26).

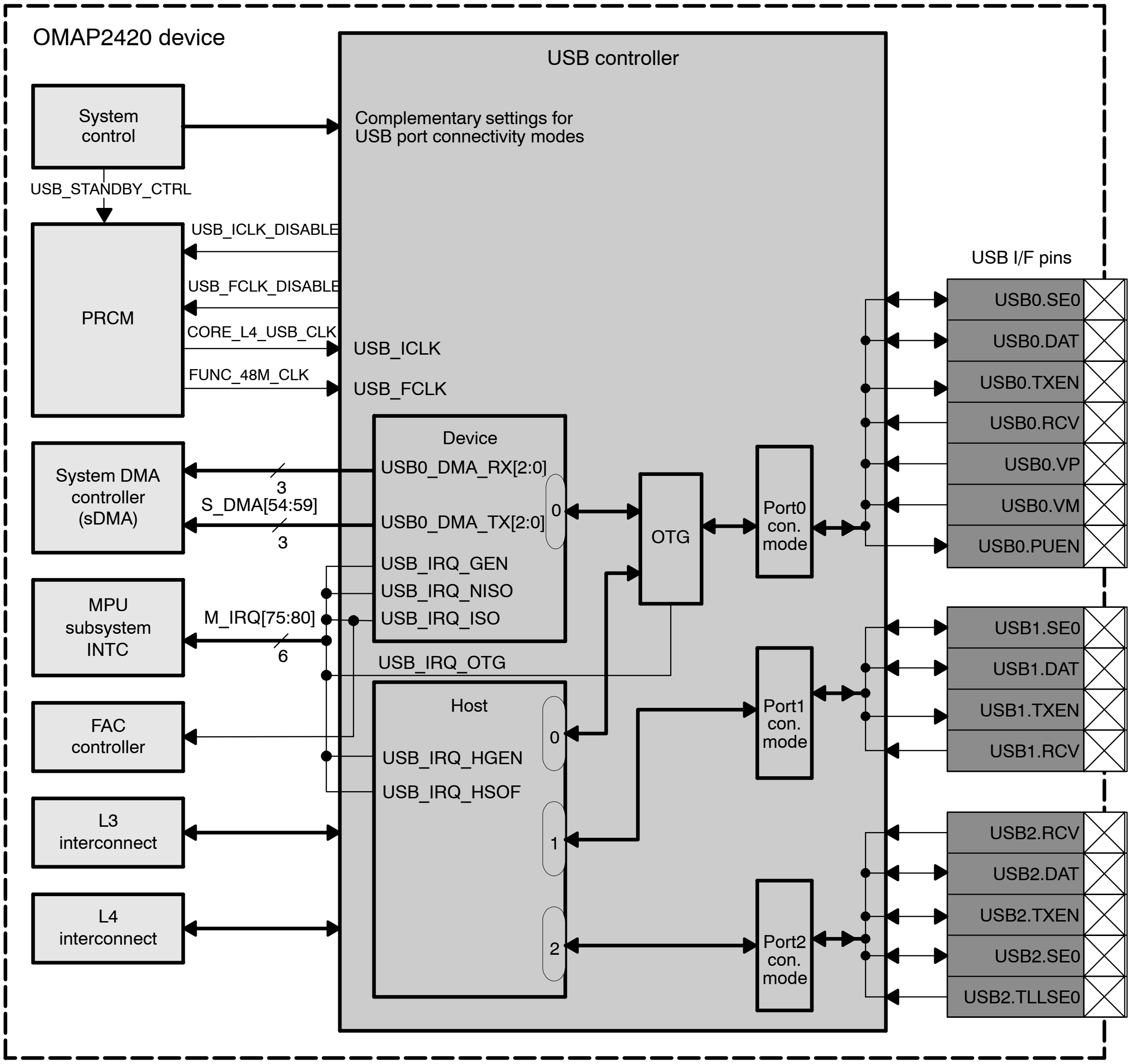
Also, one of the USB interrupts is sent to the frame adjustment counter (FAC) (see Chapter 16, *Timers*). It is considered by the FAC as the frame start signal (start-of-frame [SOF] interrupt of the USB function).

The system control module offers complementary settings for USB port connectivity modes.

When the USB controller is powered down, wake-up events are monitored by the GPIO module connected to the USB input pins.

Figure 22−26 shows the USB controller integration in the OMAP2420 device.

Figure 22−26. USB Controller Typical Application System Overview



22.3.1.1 Clocking, Reset, and Power-Management Scheme

Clocking

Two clocks are provided to the USB controller, as listed in Table 22−16.

Table 22−16. USB Clocks

Attributes	Frequency	Name	Mapping	Comments
Functional clock	48 MHz	USB_FCLK	FUNC_48M_CLK	Source is PRCM module.
Interface clock	Depending on PRCM register settings	USB_ICLK	CORE_L4_USB_CLK	Source is PRCM module.

Functional Clock

The functional clock (48 MHz), USB_FCLK, comes from the power, reset, and clock managment (PRCM) module. The PRCM module provides this clock to the USB OTG controller, USB device controller, and USB host controller. This clock is controlled by the PRCM register bit CM_FCLKEN2_CORE[0] (0: disabled; 1: enabled).

Interface Clock

The interface clock (USB_ICLK) comes from the PRCM module. This clock is controlled by the PRCM register bits CM_ICLKEN2_CORE[0] (0: disabled; 1: enabled) and CM_AUTOIDLE2_CORE[0] (enables/disables automatic control of the interface clock) (see Table 22−17).

Table 22−17. USB Interface Clock

CM_AUTOIDLE2_CORE[0]	CM_ICLKEN2_CORE[0]	Interface Clock
0	0	Disabled
0	1	Enabled
1	0	Disabled
1	1	Automatic enabling/disabling

The USB interface clock frequency is selected by writing in the PRCM register bits CM_CLKSEL1_CORE[27:25] (see Table 22−18).

Table 22−18. USB Interface Clock Selection

CM_CLKSEL1_CORE[27:25]	Selects USB interface clock
000	Reserved
001	USB interface clock is L3 interconnect clock/1 (boot mode only).
010	USB interface clock is L3 interconnect clock/2.
100	USB interface clock is L3 interconnect clock/4.
Others	Reserved

Reset

The USB controller reset can be done using a hardware reset. The USB controller is attached to the OMAP2420 CORE_RST reset domain. The CORE_RST_N signal resets the module (see Chapter 5, *Power, Reset, and Clock Management*).

The OTG controller provides a software reset mechanism through the SOFT_RESET bit OTG_SYSCON_1[1]. This field controls the software reset (set to 1 to reset USB controller, automatically reset to 0 by the hardware) for the USB OTG controller, the USB device controller, and the USB host controller.

Software must monitor the RESET_DONE bit OTG_SYSCON_1[2] (0: internal reset is ongoing; 1: reset completed) that reflects the reset status of the controller (hardware and software reset, internal reset monitoring of the USB device controller and the USB OTG controller sections). This register bit does not reflect the reset status of the USB host controller.

The USB host controller is held in reset whenever the OMAP2420 CORE_RST reset domain is under reset or the SOFT_RESET bit OTG_SYSCON_1[1] is 1. When held in reset, the USB host controller does not generate any USB activity on its USB ports. The USB host controller can transition out of reset whenever the clock is enabled, the UHOST_EN bit OTG_SYSCON_2[8] is set and the SOFT_RESET bit OTG_SYSCON_1[1] is 0.

The USB host controller completes its reset within about 72 cycles of the functional clock (48 MHz) after the host controller clock is transitioned from disabled to enabled (using the UHOST_EN bit OTG_SYSCON_2[8]), and the host controller reset is removed. After system software turns on the clock to the USB host controller and removes it from reset, it is necessary to wait until the USB host controller internal reset completes. To ensure that the USB host controller has completely reset, system software must wait until reads of both the HCREVISION and HCHCCA registers return the correct reset default values.

The OHCI specification provides the HCR bit HCCOMMANDSTATUS[0] (0: no effect; 1: software reset of the USB host controller), which resets most USB host controller OHCI registers. This reset can be used to reset the OHCI functionality and has no effect on the USB host controller L3 interconnect and L4 interconnect interfaces. The OHCI reset does not affect the USB host controller clock.

Power Domain

The USB controller is attached to the CORE power domain (see Chapter 5, *Power, Reset, and Clock Management*).

Clock and Reset Requirements

- Input conditions

  The HMC_PADEN configuration bit OTG_SYSCON_2[9] must be kept passive low (0) at all times.

  The power management circuitry is based on these register bits:

  - The OTG_IDLE_EN bit OTG_SYSCON_1[15] (OTG controller clock gating control), HST_IDLE_EN bit OTG_SYSCON_1[14] (host controller clock gating control), and DEV_IDLE_EN bit OTG_SYSCON[13] (device controller clock-gating control) are used to enable the power management circuitry or not (0: functional clock is not gated; 1: functional clock is disabled).
  - The SOFF_DIS bit SYSCON1[1]: When the shutoff disable bit is set, it disables the power shutoff circuitry (0: enabled; 1: disabled).
  - The OTG_EN bit OTG_SYSCON_2[31] is used to ensure the activity of the clocks for the OTG controller. The OTG_EN bit (0: OTG controller circuitry not activated; 1: OTG controller circuitry activated) selects the OTG controller functionality. When this bit is cleared, the USB

    device controller and the USB host controller can be used, but an OTG link cannot be implemented.

  - The UHOST_EN bit OTG_SYSCON_2[8]: Information about USB host controller use concerning USB host clock activity (0: USB host controller not clocked and held in hardware reset; 1: USB host controller not held inactive and USB host-controller hardware reset not forced).

    Power management of the OMAP2420 USB host controller is limited to disabling the clock to the USB host by clearing the UHOST_EN bit. When the UHOST_EN bit is 0, the USB signal multiplexing controlled by HMC_MODE is not affected.
  - The HMC_MODE bits OTG_SYSCON_2[5:0]: Information about USB port used. The HMC_MODE configuration bits control the OMAP USB signal multiplexing.
  - OTG_CTRL[29-21] register bits force wakeup as a function of the logic levels detected on the D+/D– lines of the USB ports (0,1, or 2) used.

- Output conditions (clock requests)

  To request the external clocks, two signals are generated externally to the PRCM module:

  - USB_ICLK_DISABLE is low to request the interface clock (1: interface clock can be shut off; 0: interface clock must be activated).
  - USB_FCLK_DISABLE is high to enable shutoff of the functional (48 MHz) clock (0: functional clock must be free-running; 1: functional clock is shut off).

Note:

The USB_ICLK_DISABLE and USB_FCLK_DISABLE output pins can be used to shut off or wake up the clocks.

  For proper USB controller clock management, power management functions in the following manner.

  In all cases the OTG_IDLE_EN, HST_IDLE_EN, and DEV_IDLE_EN bits must be set to 1, and the SOFF_DIS bit SYSCON1[1] must be set to 0 to enable the power consumption circuitry. If they are not set, USB_ICLK_DISABLE and USB_FCLK_DISABLE always require the clocks, which are internally free-running.

  - OTG used (host and device)

    Set the OTG_EN bit OTG_SYSCON_2[31] to 1 for OTG functionality, set the UHOST_EN bit OTG_SYSCON_2[8] to 1 to enable the USB host functionality, and set the SOFF_DIS bit SYSCON1[1] to enable internal power consumption for the USB device.

    In that case, USB_ICLK_DISABLE is active 0 (interface clock required), and USB_FCLK_DISABLE is passive 0 (functional clock required).

    The interface and functional clocks must be free-running for full functionality.

In any case, internal clock-gating is used to ensure power saving and module functionality.

- Host and device used (not as OTG; OTG controller not required)

  Clear the OTG_EN bit to 0 (OTG controller not used), set the UHOST_EN bit OTG_SYSCON_2[8] to 1 to enable the USB host, and set the SOFF_DIS bit SYSCON1[1] to enable internal power consumption for the USB device.

  In that case, the USB_ICLK_DISABLE is active 0 (interface clock required) and the USB_FCLK_DISABLE is passive 0 (functional clock required).

  The interface and functional clocks must be free-running for full functionality.

  In any case, internal clock gating is used to ensure power saving and module functionality.

- Host only

  Clear the OTG_EN bit OTG_SYSCON_2[31] to 0 (OTG controller not used), set the UHOST_EN bit OTG_SYSCON_2[8] to 1 to enable the USB host, and set the SOFF_DIS bit SYSCON1[1] to enable internal power consumption for USB device.

  In that case, USB_ICLK_DISABLE is active 0 (interface clock required), and USB_FCLK_DISABLE is passive 0 (functional clock required).

  The interface clock must be free-running to ensure L3 interconnect accesses.

  The functional clock must be free-running to ensure the host is cleared and the host is functional.

  In any case, internal clock-gating is used to ensure power saving and module functionality.

- Device only

  Clear the OTG_EN bit OTG_SYSCON_2[31] to 0 (OTG controller not used), and clear the UHOST_EN bit OTG_SYSCON_2[8] to 0 to disable the USB host.

  The USBx_EN bits must be set to 0 (OTG_CTRL[29] = 0; OTG_CTRL[26] = 0; OTG_CTRL[23] = 0). In that case, USB_ICLK_DISABLE and USB_FCLK_DISABLE change as a function of the USB line (VBUS, suspend…), but the clock is internally gated.

- Low power consumption (as low as possible): No use of the module but can access it

  Enable the low-power circuitry by setting the following register bits of OTG_SYSCON_1 to 1: OTG_IDLE_EN, HST_IDLE_EN, and DEV_IDLE_EN.

  Clear the OTG_EN bit OTG_SYSCON_2[31] to 0, clear the UHOST_EN bit OTG_SYSCON_2[8] to 0 to disable the USB host, and select the HMC_MODE bits OTG_SYSCON_2[5:0] to 0x16.

In that case, USB_ICLK_DISABLE and USB_FCLK_DISABLE remain inactive 1 (interface and functional clocks are disabled).

- ☐ Clock requirement for register accesses

  Any access to the USB host registers (address from 0x4805 E000 to 0x4805 E1FF) requires that the UHOST_EN bit OTG_SYSCON_2[8] is set to enable the USB host clock and the PRCM functional clock (48 MHz). In that case, the USB_ICLK_DISABLE and USB_FCLK_DISABLE output pins always require the interface and functional clocks.

  Any access to the USB device registers (address from 0x4805 E200 to 0x4805 E2FF) requires the interface clock. In some particular cases, USB_ICLK_DISABLE can be inactive 1 (USB suspend or USB cable disconnected) and the system can access the module only by generating the clock of the bus interface.

  Any access to the USB OTG registers (address from 0x4805 E300 to 0x4805 E3FF) requires only the interface clock (for a USB device, USB_ICLK_DISABLE can also be inactive 1). In any case, two registers (OTG_SYSCON_2 and OTG_CTRL) cannot be accessed if the system does not enable the functional clock (48 MHz). If these addresses are accessed, the system is not stalled if the functional clock is not available, because the USB_FCLK_DISABLE output pin is inactive 0 during such accesses.

  The OTG controller uses the same functional clock (48 MHz) input from the PRCM module to provide clocks to the USB device controller and the USB host controller.

  When the PRCM functional clock (48 MHz) to the OTG controller is disabled, the USB host controller registers cannot be accessed and the USB host controller cannot respond to any downstream USB bus activity.

  The OTG controller can disable the USB device controller clock using the DEV_IDLE_EN bit OTG_SYSCON_1[13]. When the PRCM functional clock to the OTG controller is disabled, or when the DEV_IDLE_EN bit OTG_SYSCON_1[13] is 1, the USB device controller registers cannot be accessed; however, the USB device controller can respond to USB bus resume signaling by issuing a USB device controller general interrupt.

  System software can read USB device controller registers when the DEV_IDLE_EN bit OTG_SYSCON_1[13] is 1, but must enable the functional clock to the USB device controller before writing to the USB device controller registers.

- ☐ Reset requirement for register accesses

  When the USB host controller is in hardware reset, read or write accesses to its registers have no effect. It is recommended that USB host controller software read the HCREVISION and HCHCCA registers after deasserting reset to verify the proper reset values. If the read values for HCREVISION and HCHCCA are not correct, reset the values and continue reading until the proper reset values are seen.

  When the UHOST_EN bit is cleared, it also holds the USB host controller in a hardware reset. While the USB host controller is in reset, reads from the USB host controller registers do not return valid data, and writes to the USB host controller registers have no effect. When the UHOST_EN bit is cleared, all USB host controller internal state information is lost.

Software that initializes the USB host controller must ensure that the reset is turned off, the SOFT_RESET bit OTG_SYSCON_1[1] is 0, the PRCM functional clock (48 MHz) for the USB host controller is enabled, and the UHOST_EN bit OTG_SYSCON_2[8] is set. It must then wait until reads of both the HCREVISION and HCHCCA registers return their correct reset default values.

When held in hardware reset, the USB OTG controller cannot perform USB On-The-Go SRP or HNP protocols, and its registers cannot be accessed.

No known specific behavior for USB device controller.

☐ Clock requirement for system memory access

The PRCM interface clock must be active to allow USB host controller access to system memory.

22.3.1.2 Hardware Requests

DMA Requests

Table 22−19 lists the DMA channels that are driven out from the USB device controller to the system DMA (sDMA) to support streaming USB data.

Table 22−19. USB DMA

Name	Mapping	Comments
USB device (port 0) – DMA request for IN transactions (transmit) channel 0		
USB0_DMA_TX0	S_DMA_54	Destination is sDMA.
USB device (port 0) – DMA request for OUT transactions (receive) channel 0		
USB0_DMA_RX0	S_DMA_55	Destination is sDMA.
USB device (port 0)– DMA request for IN transactions (transmit) channel 1		
USB0_DMA_TX1	S_DMA_56	Destination is sDMA.
USB device (port 0) – DMA request for OUT transactions (receive) channel 1		
USB0_DMA_RX1	S_DMA_57	Destination is sDMA.
USB device (port 0) – DMA request for IN transactions (transmit) channel 2		
USB0_DMA_TX2	S_DMA_58	Destination is sDMA.
USB device (port 0) – DMA request for OUT transactions (receive) channel 2		
USB0_DMA_RX2	S_DMA_59	Destination is sDMA.

When an RX DMA request is active (0) for a channel (only one active at a given time), data must be read into the DMA FIFO data register (DATA_DMA).

When a TX DMA request is active (0) for a channel (only one active at a given time), data must be written into the DMA FIFO data register (DATA_DMA) to be transmitted.

Interrupts Requests

Table 22−20 lists the interrupt lines that are driven out from the USB controller to the MPU subsystem interrupt controller.

Table 22−20. USB Interrupts

Name	Mapping	Comments
USB device general interrupt		
USB_IRQ_GEN	M_IRQ_75	Destination is MPU subsystem interrupt controller.
USB device non-ISO		
USB_IRQ_NISO	M_IRQ_76	Destination is MPU subsystem interrupt controller.
USB device ISO		
USB_IRQ_ISO	M_IRQ_77	Destination is MPU subsystem interrupt controller.
USB host general interrupt		
USB_IRQ_HGEN	M_IRQ_78	Destination is MPU subsystem interrupt controller.
USB host start of frame		
USB_IRQ_HSOF	M_IRQ_79	Destination is MPU subsystem interrupt controller.
USB OTG		
USB_IRQ_OTG	M_IRQ_80	Destination is MPU subsystem interrupt controller.

Idle Mode Request/Acknowledge Feature

The USB controller is an L3 interconnect initiator and target module.

When set to 1, the USB_FCLK_DISABLE and USB_ICLK_DISABLE output signals (the functional and interface clocks can be shut off; see Section 22.3.1.1, *Clocking, Reset, and Power-Management Scheme*) from the USB controller to the PRCM module, and the USB_STANDBY_CTRL signal (via the system control module bit CONTROL_DEVCONF[15]; see Chapter 8, *System Control Module*), which is the software control of the USB standby signal from the system control module to the PRCM module, allow the USB controller to enter idle mode and save power (see Table 22−21).

Table 22−21. Idle Mode Request/Acknowledge Signals

Name	Source	Destination
USB_FCLK_DISABLE	USB controller	PRCM module
USB_ICLK_DISABLE	USB controller	PRCM module
USB_STANDBY_CTRL	System control module	PRCM module

The USB controller idle state can be checked by the PRCM module register bit CM_IDLEST2_CORE[0] (0: USB cannot be accessed; 1: USB can be accessed).

The USB controller wake-up status can be checked by the PRCM module register bit PM_WKST2_CORE[0] (read 0: no wakeup occurred; read 1: wake-up occurred; write 1: status bit reset).

Wakeup Generation

- ❑ Internal wakeup

The PRCM register bit PM_WKEN2_CORE[0] (0: USB wakeup is disabled; 1: USB wakeup is enabled) lets you enable or disable the core domain on a USB controller wake-up event. When the USB controller is in idle mode and the L3 clocks shut down, a USB internal wake-up event can occur. Releasing the USB wake-up output signals (USB_FCLK_DISABLE or USB_ICLK_DISABLE) (0: functional and interface clock must be activated; see Section 22.3.1.1, *Clocking, Reset, and Power-Management Scheme*) activates the L3 clocks.

- ❑ External wakeup

When disabled, the USB controller also has an external wake-up event through the GPIO channels MUXed on its pins that are configured as wake-up sources. USB wakeup can also be forced by the OTG_CTRL[29:21] register bits as a function of the programmed values of USBx_DP and USBx_DM and the logic level detected on the D+/D– line of the USB port (0, 1, or 2) (see Table 22–22).

Table 22–22. OTG_CTRL: OTG Control Register

Address Offset:: 0x30C **OTC_CTRL:** OTG Control **Read-Write**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		USB2_EN	USB2_DP	USB2_DM	USB1_EN	USB1_DP	USB1_DM	USB0_EN	USB0_DP	USB0_DM	ASESSVLD	BSESSEND	BSESSVLD	VBUSVLD	ID	DRIVER_SEL	Reserved		A_SETB_HNPEN	A_BUSREQ	Reserved	B_HNPEN	B_BUSREQ	OTG_BUSDROP	Reserved	OTG_PD	OTG_PU	OTG_DRV_VBUS	OTG_PD_VBUS	OTG_PU_VBUS	OTG_PU_ID
Reset																															
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
31–30	Reserved	
29	USB2_EN	USB port 2 – Enable information
28	USB2_DP	USB port 2 – D+ information
27	USB2_DM	USB port 2 – D– information
26	USB1_EN	USB port 1 – Enable information
25	USB1_DP	USB port 1 – D+ information
24	USB1_DM	USB port 1 – D– information
23	USB0_EN	USB port 0 – Enable information
22	USB0_DP	USB port 0 – D+ information
21	USB0_DM	USB port 0 – D– information

Table 22−22. OTG_CTRL: OTG Control Register (Continued)

Bit	**Name**	**Description**
20	ASESSVLD	ASESSVLD information
19	BSESSEND	BSESSEND information
18	BSESSVLD	BSESSVLD information
17	VBUSVLD	VBUSVLD information
16	ID	ID information
15	DRIVER_SEL	OTG driver selection
14–13	Reserved	NA
12	A_SETB_HNPEN	A-device information, B-device HNP enabled
11	A_BUSREQ	A-device information, A-device bus request
10	Reserved	NA
9	B_HNPEN	B-device information, B-device HNP enabled
8	B_BUSREQ	B-device information, B-device bus request
7	OTG_BUSDROP	OTG BUS DROP information
6	Reserved	NA
5	OTG_PD	USBxPDENON pin status information
4	OTG_PU	USBxPUENON pin status information
3	OTG_DRV_VBUS	USBxDRIVEVBUSO pin status information
2	OTG_PD_VBUS	OTGPDVBUSON pin status information
1	OTG_PU_VBUS	OTGPUVBUSON pin status information
0	OTG_PU_ID	OTGPUIDON pin status information

22.3.1.3 Pin List and Pad Multiplexing With Other Functions

The OMAP2420 USB signal multiplexing determines which USB functionality is available at which OMAP2420 pins. The OMAP2420 device includes three pin groups that can be configured to provide up to three USB ports. These ports can be configured as a USB OTG port, USB device port, and/or up to three USB host ports. Each USB port used in the system requires an external USB transceiver (see Section 22.2, *USB Controller Environment*).

Table 22−23 describes pin and pad multiplexing options for the USB controller. USB port 2 is available only with the OMAP2420 device. Mode 5 is not used.

Table 22−23. Pin Multiplexing for the USB Controller

Function n	Description	DIR	Ball	Alternate Functions					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
USB0.SE0	Single-ended zero. Used as VM in 4-pin VP_VM mode.	IO	J14		uart3.tx/irtx	uart2.tx	gpio.111	uart2.rx	
USB0.DAT	USB data.Used as VP in 4-pin VP_VM mode.	IO	K18		uart3.rx/irrx	uart2.rx	gpio.112	uart2.tx	
USB0.TXEN	Transmit enable	O	K19		uart3.cts/rctx	uart2.cts	gpio.110		
USB0.RCV	Differential receiver signal input (not used in 3-pin mode)	I	J18		mcbsp2.fsx		gpio.109	uart2.cts	
USB0.VP	Vplus receive data (not used in 3- or 4-pin configurations)	I	J19		mcbsp2.dr		gpio.107		
USB0.VM	Vminus receive data (not used in 3 or 4 pin configurations)	I	K20		mcbsp2.clkx		gpio.108	uart2.rx	
USB0.PUEN	USB pullup enable	O	J20		mcbsp2.dx		gpio.106		
USB1.SE0	Single-ended zero. Used as VM in 4-pin VP_VM mode, Also used as DP_TLL in 2- or 4-pin TLL mode.	IO	N14	uart2.tx		gpt11.pwm/evt	gpio.69		
			W12		uart1.tx		gpio.59		
USB1.DAT	USB data. Used as VP in 4-pin VP_VM mode and as DN_TLL in 2- or 4-pin TLL.	IO	P15	uart2.rx		gpt12.pwm/evt	gpio.70		
			R13	gpio.62	uart1.rx		gpio.62		
USB1.TXEN	Transmit enable (not used in 2- or 3-pin TLL configurations)	O	W20	uart2.rts		gpt10.pwm/evt	gpio.68		
			P13		uart1.cts		gpio.61		
USB1.RCV	Differential receiver signal Input (not used in 2-pin configuration)	I	V19	uart2.cts		gpt9.pwm/evt	gpio.67		
			V12		uart1.rts		gpio.25		

Table 22−23. Pin Multiplexing for the USB Controller (Continued)

Function n	Description	DIR	Ball	Alternate Functions					
Interface				Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
USB2.SE0	Single-ended zero. Used as VM in 4-pin VP_VM mode, Used as DP_TLL in 2- or 4-pin TLL mode.	IO	AA10			sys.ndma-req0	gpio.13		
USB2.DAT	USB data. Used as VP in 4-pin VP_VM mode and as DN_TLL in 2-pin TLL mode.	IO	Y11			sys.clkout2	gpio.16		
USB2.TXEN	Transmit enable (not used in 2-pin TLL configuration).	IO	AA12				gpio.17		
USB2.RCV	Differential receiver signal input. Also used as output for RCV_TLL/VP_TLL in 5-pin TLL mode.	I	AA6			sys.ndma-req1	gpio.14	cam_d8	
USB2.TLLSE0	VM_TLL data in 5-pin TLL mode (not used in 2-, 3-, or 4-pin configurations)	I	AA4				gpio.15	cam_d7	

22.3.1.4 USB Controller Register Summary

Table 22−24 lists the base address and address space for the USB controller.

Table 22−25 through Table 22−27 list the USB controller registers and their physical addresses. For a detailed description, see Section 22.7, *USB Registers*.

Table 22−24. Instance Summary

Device Name	Description	Base Address	Size
USB controller	USB module	0x4805 E000	4K bytes
	L4 interconnect target	0x4805 F000	512 bytes
	L3 interconnect initiator and target	0x6800 1200	512 bytes

CAUTION

USB host controller (HC) registers (0x4805E000 to 0x4805E0EC addresses) are limited to 32-bit and 16-bit data access. 8-bit data access generates an error. See Section 22.7.1, *Register Descriptions*, for more information.

USB device controller registers (0x4805E200 to 0x4805E2FC addresses) are limited to 32-bit data access; 16-bit and 8-bit data access generates an error. See Section 22.7.1, *Register Descriptions*, for more information.

Table 22−25. USB Module Registers

Register Name	Type	Register Width (Bits)	Physical Address
HCREVISION	R	32	0x4805 E000
HCCONTROL	RW	32	0x4805 E004
HCCOMMANDSTATUS	RW	32	0x4805 E008
HCINTERRUPTSTATUS	RW	32	0x4805 E00C
HCINTERRUPTENABLE	RW	32	0x4805 E010
HCINTERRUPTDISABLE	RW	32	0x4805 E014
HCHCCA	RW	32	0x4805 E018
HCPERIODCURRENTED	R	32	0x4805 E01C
HCCONTROLHEADED	RW	32	0x4805 E020
HCCONTROLCURRENTED	RW	32	0x4805 E024
HCBULKHEADED	RW	32	0x4805 E028
HCBULKCURRENTED	RW	32	0x4805 E02C
HCDONEHEAD	R	32	0x4805 E030
HCFMINTERVAL	RW	32	0x4805 E034
HCFMREMAINING	R	32	0x4805 E038
HCFMNUMBER	R	32	0x4805 E03C
HCPERIODICSTART	RW	32	0x4805 E040
HCLSTHRESHOLD	RW	32	0x4805 E044
HCRHDESCRIPTORA	RW	32	0x4805 E048
HCRHDESCRIPTORB	RW	32	0x4805 E04C
HCRHSTATUS	RW	32	0x4805 E050
HCRHPORTSTATUS_1	RW	32	0x4805 E054
HCRHPORTSTATUS_2	RW	32	0x4805 E058
HCRHPORTSTATUS_3	RW	32	0x4805 E05C
HOSTUEADDR	R	32	0x4805 E0E0
HOSTUESTATUS	R	32	0x4805 E0E4
HOSTTIMEOUTCTRL	RW	32	0x4805 E0E8
HOSTREVISION	R	32	0x4805 E0EC
WHM_REVID_REGISTER	R	32	0x4805 E0F4
WHM_TEST_OBSV	R	32	0x4805 E0F8
WHM_TEST_CTL	RW	32	0x4805 E0FC
HHC_TEST_CFG	RW	32	0x4805 E100
HHC_TEST_CTL	RW	32	0x4805 E104
HHC_TEST_OBSV	R	32	0x4805 E108
REVDEV	R	16	0x4805 E200
EP_NUM	RW	16	0x4805 E204

Table 22−25. USB Module Registers (Continued)

Register Name	Type	Register Width (Bits)	Physical Address
DATA	RW	16	0x4805 E208
CTRL	RW	16	0x4805 E20C
STAT_FLG	R	16	0x4805 E210
RXFSTAT	R	16	0x4805 E214
SYSCON1	RW	16	0x4805 E218
SYSCON2	RW	16	0x4805 E21C
DEVSTAT	R	16	0x4805 E220
SOFREG	R	16	0x4805 E224
IRQ_EN	RW	16	0x4805 E228
DMA_IRQ_EN	RW	16	0x4805 E22C
IRQ_SRC	RW	16	0x4805 E230
EPN_STAT	R	16	0x4805 E234
DMAN_STAT	R	16	0x4805 E238
RXDMA_CFG	RW	16	0x4805 E240
TXDMA_CFG	RW	16	0x4805 E244
DATA_DMA	RW	16	0x4805 E248
TXDMA0 − TXDMA2	RW	16	0x4805 E250−0x4805 E258
RXDMA0 − RXDMA2	RW	16	0x4805 E260−0x4805 E268
EP0	RW	16	0x4805 E280
EP_RX1 − EP_RX15	RW	16	0x4805 E284−0x4805 E2BC
EP_TX1 − EP_TX15	RW	16	0x4805 E2C4−0x4805 E2FC
OTG_REV	R	32	0x4805 E300
OTG_SYSCON_1	RW	32	0x4805 E304
OTG_SYSCON_2	RW	32	0x4805 E308
OTG_CTRL	RW	32	0x4805 E30C
OTG_IRQ_EN	RW	16	0x4805 E310
OTG_IRQ_SRC	RW	16	0x4805 E314
OTG_OUTCTRL	RW	16	0x4805 E318
OTG_TEST	RW	16	0x4805 E320
OTG_VC	R	32	0x4805 E3FC

Table 22−26. L4 Interconnect Target Registers

Register Name	Type	Register Width (Bits)	Physical Address
COMPONENT	R	32	0x4805 F000
AGENT_CONTROL	RW	32	0x4805 F020
AGENT_STATUS	R	32	0x4805 F028

Table 22−27. L3 Interconnect Initiator and Target Registers

Register Name	Type	Register Width (Bits)	Physical Address
SBIMERRLOGA	R	32	0x6800 12A8
SBIMERRLOG	RW	32	0x6800 12B0
SBIMSTATE	RW	32	0x6800 1390
SBTMSTATE_L	RW	32	0x6800 1398
SBTMSTATE_H	RW	32	0x6800 139C
SBIMCONFIG_L	RW	32	0x6800 13A8
SBIMCONFIG_H	RW	32	0x6800 13AC
SBID_L	R	32	0x6800 13F8
SBID_H	R	32	0x6800 13FC

22.4 USB Host Controller

The OMAP2420 USB host controller is a 3-port controller that communicates with USB devices at USB low-speed (1.5 Mbps maximum) and full-speed (12 Mbps maximum) data rates. It is compatible with the USB 2.0 specification and the OTG specification supplement, available through the Compaq Computer Corporation website. It is assumed that users of the OMAP2420 USB host controller are familiar with the USB specification and the OHCI specification.

The OMAP2420 OTG controller can use one of the USB host controller ports as part of a USB OTG-capable connection. When used for an OTG connection, the host controller port acts as the upstream device when the OMAP2420 device controls the OTG link. The USB device controller acts as the downstream device when OMAP2420 acts as an OTG downstream device. The OMAP2420 OTG controller is described in Section 22.6, *USB OTG Controller*.

The OMAP2420 USB host controller implements the register set and uses the memory data structures defined in the OHCI specification. These registers and data structures are the mechanism by which a USB host controller driver software package can control the OMAP2420 USB host controller. The OHCI specification also defines how the USB host controller implementation must interact with those registers and data structures in system memory. The OMAP2420 MPU subsystem accesses these registers through the OMAP2420 L4 interconnect.

To reduce processor software and interrupt overhead, the USB host controller generates USB traffic based on data structures and data buffers stored in system memory. The OMAP2420 USB host controller accesses these data structures without direct intervention by the processor using the OMAP2420 L3 interconnect. These data structures and data buffers can be in internal or external system RAM.

The USB host controller provides two interrupts to the MPU subsystem interrupt controller to signal certain hardware events to the host controller driver software.

22.4.1 USB Open Host Controller Interface Functionality

22.4.1.1 OHCI Controller Overview

The OHCI specification defines a set of registers and data structures stored in system memory that define how a USB host controller interfaces to system software. This specification, in conjunction with the USB 2.0 specification, defines most of the USB functionality that the OMAP2420 USB host controller provides.

The OHCI specification focuses on two main aspects of the hardware implementation of a USB host controller: its register set and the memory data structures that define the activity to appear on the USB bus. It also discusses issues such as interrupt generation, USB host controller state, USB frame management, and the methods the hardware must use to process the lists of data structures in system memory.

This document does not duplicate the information presented in the OHCI specification or the USB specification. Refer to these specifications for detailed discussions of USB requirements and OHCI controller operation.

22.4.2 USB Host Controller Differences From OHCI Specification

The OMAP2420 USB host controller implementation does not implement every aspect of the functionality defined in the OHCI specification. The differences focus on power switching, overcurrent reporting, and the OHCI ownership change interrupt. Other restrictions are imposed by the effects of OMAP2420 pin multiplexing options.

22.4.2.1 Power Switching Output Pins Not Supported

The OMAP2420 device does not provide pins that can be controlled directly by the OHCI port power-control features in the USB host controller. The OHCI HCRHPORTSTATUS register port power-control bits can be programmed by the USB host controller driver software, but this does not directly affect any VBUS switching implemented on the board.

To control VBUS switching, use software to control GPIO pins or other implementation-specific control mechanisms.

22.4.2.2 Overcurrent Protection Input Pins Not Supported

The OMAP2420 device does not provide any pins that allow the OHCI HCRHPORTSTATUS overcurrent-protection status bits in the USB host controller to be directly controlled by external hardware.

To report port overcurrent information to the USB host controller driver, use software to monitor GPIO pins or other implementation-specific control mechanisms.

22.4.2.3 HMC_MODE and Top-Level Pin Multiplexing and OHCI Registers

The USB signal multiplexing modes selected by HMC_MODE provide selections where 0, 1, 2, or 3 USB host controller ports can be brought to OMAP2420 pins. The OHCI HCRHDESCRIPTORA register always reports three available USB host ports, regardless of the HMC_MODE setting or top-level pin multiplexing settings. When the HMC_MODE setting disables a USB host controller port, the USB host sees the port as unattached.

When OMAP2420 top-level pin multiplexing configures a pin for functionality other than the USB, the USB host controller is disconnected from that pin and that pin does not affect the USB host controller.

22.4.2.4 No Ownership Change Interrupt

The OMAP2420 USB host controller does not implement the OHCI ownership change interrupt.

22.4.3 OMAP2420 Implementation of OHCI Specification

22.4.3.1 Isochronous Transmit Descriptor (TD) OFFSETX/PSWX Values

The OMAP2420 USB host controller implements the optional feature in the OHCI specification for checking isochronous OFFSETX/PSWX values. If either OFFSETX or OFFSET(X+1) does not have a condition code of Not Accessed, or if the OFFSET(X+1) value is not greater than or equal to OFFSETX, then an unrecoverable error is reported. Unrecoverable errors issued for these reasons do not cause an update of the HOSTUEADDR, HOSTUESTATUS, or HOSTTIMEOUTCTRL registers.

22.4.3.2 OMAP2420 USB Host Controller Endpoint Descriptor (ED) List Head Pointers

The OHCI specification provides a specific sequence of operations for the host controller driver to perform when setting up the host controller. Failure to follow that sequence can result in malfunction. As a specific example, the HCCONTROLHEADED and HCBULKHEADED pointer registers and the 32 HCCAINTERRUPTTABLE pointers must all point to valid physical addresses of valid endpoint descriptors.

The OMAP2420 USB host controller does not check HCCONTROLHEADED registers, HCBULKHEADED registers, or the values in the 32 HCCAINTERRUPTTABLE pointers before using them to access EDs. If any of these pointers are NULL when the corresponding list enable bit is set, the OMAP2420 USB host controller tries to access using the physical address of 0, which causes an unrecoverable error to be signaled. In this case, the HOSTUEADDR, HOSTUESTATUS, and HOSTTIMEOUTCTRL registers are updated.

22.4.3.3 OHCI USB Suspend State

The OMAP2420 USB host controller ignores upstream traffic from downstream devices for about 3 ms after the host controller state (HCFS bits HCCONTROL[7:6]) changes from USB resume state to USB operational state. If any TDs cause generation of downstream packets during that time, the downstream packets are sent, but any response provided by the downstream device is ignored. Any such TDs are aborted with completion codes marked as Device Not Responding. TDs on any of the lists (periodic, control, bulk, and isochronous) can cause such an occurrence.

The USB specification requires system software to provide a 10-ms resume recovery time (T_{RSMRCY}) after a bus segment transitions from resume signaling to normal operational mode. During that time, only start-of-frame (SOF) packets are to be sent on the bus segment. It is recommended that system software disable all list enable bits (PLE bit HCCONTROL[2], IE bit HCCONTROL[3], CLE bit HCCONTROL[4], and BLE bit HCCONTROL[5]), and then wait at least 1 ms before setting the host controller into USB suspend state (using HCFS bits HCCONTROL[7:6]). When restoring from suspend state, system software must set the host controller into USB resume state, and wait for the host controller to transition into USB operational state. System software must then wait 10 ms before enabling the host controller list enable bits.

When the host controller is placed into the USB suspend state under software control but is brought out by a remote wakeup, system software must monitor the PSS_SPS bit HCRHPORTSTATUS_x[2] and the PSSC bit HCRHPORTSTATUS_x[18] (where x is 1, 2, or 3). The PSS_SPS bit HCRHPORT STATUS_x[2] changes to 0 only after completion of resume signaling on the bus segment and completion of the 3-ms period where packets from downstream devices are ignored.

When using port-specific suspend, it is not necessary to disable the host controller lists as long as there are no active EDs and TDs directed toward devices that are downstream of the suspended port. For port-specific suspend operations, the host controller does not issue a root hub status change interrupt with the PSSC bit HCRHPORTSTATUS_N[18] = 1 and the PSS_SPS bit HCRHPORTSTATUS_N[2] = 0 until after the approximately 3-ms delay after resume signaling completes.

When using port-specific suspend, system software must ensure that there are no active EDs for devices that are downstream of the suspended port before setting the port into suspend state. While the port is in suspend or is being resumed, system software must not enable any EDs for any devices downstream of the suspended port. When the root hub status change interrupt occurs because the suspended port PSS_SPS bit changes to 0, EDs can be enabled for devices downstream of the port that is now operational.

22.4.4 OHCI Interrupts

The OMAP2420 USB host controller provides two interrupt outputs to the MPU subsystem interrupt controller. The USB host controller general interrupt (USB_IRQ_HGEN) is a level-sensitive interrupt signal on its M_IRQ_78 interrupt input. The USB host SOF indication (USB_IRQ_HSOF) is a pulse-sensitive interrupt signal on its M_IRQ_79 interrupt input.

22.4.4.1 OHCI Scheduling Overrun Interrupt

The OHCI scheduling overrun interrupt is supported as described in the OHCI specification.

When enabled, the OHCI scheduling overrun interrupt is propagated by the USB host controller general interrupt (USB_IRQ_HGEN) interrupt signal to the M_IRQ_78 interrupt input.

22.4.4.2 OHCI HCDONEHEAD Writeback Interrupt

The OHCI HCDONEHEAD writeback interrupt is supported as described in the OHCI specification.

When enabled, the OHCI HCDONEHEAD writeback interrupt is propagated by the USB host controller general interrupt (USB_IRQ_HGEN) interrupt signal to the M_IRQ_78 interrupt input.

22.4.4.3 OHCI SOF Interrupt

The OHCI SOF interrupt is supported as described in the OHCI specification.

When enabled, the OHCI SOF interrupt is propagated by the USB host SOF indication (USB_IRQ_HSOF) interrupt signal to the M_IRQ_79 interrupt input.

22.4.4.4 OHCI Resume Detect Interrupt

The OHCI resume detect interrupt is supported as described in the OHCI specification.

When enabled, the OHCI resume detect interrupt is propagated by the USB host controller general interrupt (USB_IRQ_HGEN) interrupt signal to the M_IRQ_78 interrupt input.

22.4.4.5 OHCI Unrecoverable Error Interrupt

The OHCI unrecoverable error interrupt is supported as described in the OHCI specification. This interrupt occurs if the USB host controller cannot complete an L3 interconnect read or L3 interconnect write within 4096 interface clocks when the USB host L3 interconnect time-out feature is enabled. When an L3 interconnect time-out causes an unrecoverable error, the HOSTUEADDR and HOSTUESTATUS registers are updated. When an isochronous TD is processed with an OFFSET/PSW field that is not set for Not Accessed, an unrecoverable error interrupt is generated, but HOSTUEADDR and HOSTUESTATUS are not updated.

When enabled, the OHCI unrecoverable error interrupt is propagated by the USB host controller general interrupt (USB_IRQ_HGEN) interrupt signal to the M_IRQ_78 interrupt input.

22.4.4.6 OHCI Frame Number Overflow

The OHCI frame number overflow interrupt is supported as described in the OHCI specification.

When enabled, the OHCI frame number overflow interrupt is propagated by the USB host controller general interrupt (USB_IRQ_HGEN) interrupt signal to the M_IRQ_78 interrupt input.

22.4.4.7 OHCI Root Hub Status Change

The OHCI root hub status change interrupt is supported as described in the OHCI specification. The OMAP2420 device does not provide a connection between the USB host controller and USB port overcurrent detection hardware, so the root hub status change interrupt does not occur because of a port overcurrent event.

When enabled, the OHCI root hub status change interrupt is propagated by the USB host controller general interrupt (USB_IRQ_HGEN) interrupt signal to the M_IRQ_78 interrupt input.

22.4.4.8 OHCI Ownership Change Interrupt

The optional OHCI ownership change interrupt is not supported.

22.4.5 USB Host Controller Access to System Memory

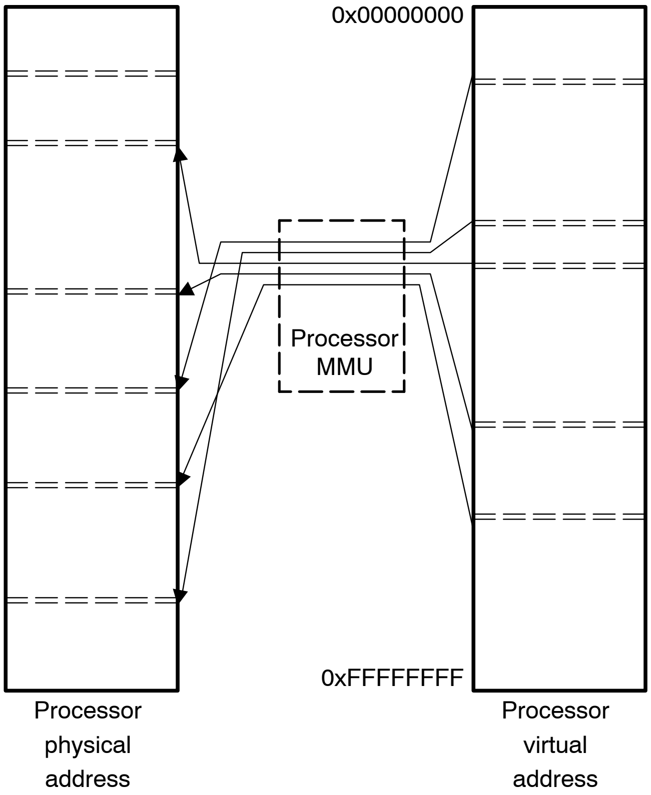
The USB host controller must have access to system memory to read and write the OHCI data structures and data buffers associated with USB traffic. The OMAP2420 L3 interconnect lets the USB host controller access OMAP2420 system memory.

22.4.6 Physical Addressing

Because transactions on the OMAP2420 L3 interconnect use physical addresses, all system memory accesses initiated by the USB host controller must use physical addresses. The OMAP2420 MPU subsystem can be configured to use virtual addressing. In this case, MPU software manipulates virtual addresses that may or may not be identical to physical addresses. When virtual addressing is used, system software must perform conversions from the appropriate virtual address to physical address and physical address to virtual address when manipulating the USB host controllers data structures and pointers to those data structures.

Figure 22−27 shows the MPU virtual address-to-physical address conversion.

Figure 22−27. Relationships Between Virtual Address/Physical Address



22.4.7 Cache Coherency in OHCI Data Structures and Data Buffers

The OMAP2420 traffic controller does not provide mechanisms to flush (or writeback) the MPU cache when a DMA controller or L3 interconnect access to system memory occurs. Because there is no forced coherency mechanism, the system implementation must ensure that the OMAP2420 USB host controller can access the correct data from system memory and that the MPU subsystem accesses that same data. This requires that any system memory accessed by the USB host controller be allocated in noncached system memory.

If the OHCI data structures and/or data buffers are allocated in cached portions of system memory, a cache coherency problem can exist because the MPU subsystem can read from, and if in writeback mode, write to the cache; however, the USB host controller accesses are always directly to the physical system memory. If the data structures are in a cached portion of system memory and writeback mode is enabled, it is possible that the USB host controller can read stale data that has not been updated by a cache writeback.

Similarly, if the data structure is in memory that is currently in the MPU cache (either writeback or writethrough mode) and the OHCI controller modifies the information in physical memory, the MPU subsystem can read stale data from the cache.

Cache coherency problems can be avoided by allocating the OHCI data structures (HCCA, EDs, and TDs) and the USB data buffers in noncacheable system memory. In this case, every MPU subsystem access directly accesses physical memory, so there is no coherency issue. Configuration of cacheable portions of the MPU virtual address space is provided by the MPU memory management unit (see Chapter 9, *Memory Management Units*).

22.4.8 L3 Interconnect Addressing and OHCI Data Structure Pointers

The USB host controller OHCI registers that point to the HCCA and the ED lists must be programmed with values that correspond to the physical addresses of the particular data structure. System software must use physical addresses and not virtual addresses when manipulating the OHCI control registers that point to the HCCA and to the ED and TD lists. System software must also use physical addresses for the ED and TD pointers that are stored in ED and TD entries.

The USB host controller driver software must also be able to examine the list of completed transfer descriptors that the host controller creates as it retires transfer descriptors. The HCDONEHEAD register, which points to this list, contains a physical address that points to the most recent transfer descriptor that is retired. This requires that the host controller driver software must be able to convert from the physical address back to an MPU virtual address.

Several address conversion functions are helpful to enable proper addressing by the MPU software and the USB host controller. The functionality required to convert an address properly depends on the settings used in the MPU memory management unit (MMU); therefore, it is system-dependent. The conversion functions are described in general terms in the following subsections.

22.4.8.1 MPUVA to PA()—MPU Virtual Address-to-Physical Address Conversion Function

This function converts an MPU virtual address to the equivalent system physical address. This function must understand the way that the system software has configured the MPU MMU. This function must provide a conversion that has a result that is identical to the conversion done in hardware by the MPU MMU.

22.4.8.2 PA to MPUVA()—Physical Address-to-MPU Virtual Address Conversion Function

This function is a reverse version of the MPU virtual address-to-physical address conversion. It accepts a physical address as an argument and returns the equivalent MPU virtual address.

It is possible to program the MPU MMU to allow two (or more) different MPU virtual addresses to map to the same physical address. This is especially common in systems that use linear addressing within a task. System software implementers must be careful to avoid that situation or to perform the conversion in a way that understands the task-specific conversion requirements.

22.4.9 NULL Pointers

The OHCI specification uses NULL pointers to indicate the end of a list. The OMAP USB host controller compares the ED and TD pointers against the value 0x00000000 to determine if the pointer is a NULL pointer. Address conversion routines must understand this usage.

22.4.10 L3 Interconnect and the USB Host Controller

The OMAP2420 L3 interconnect provides a mechanism whereby OMAP2420 peripheral modules can access system memory. Unlike peripherals that use the DMA controller, peripherals connected to the L3 interconnect can generate the address, byte enables, and read/write indication for each L3 interconnect access. This functionality lets the USB host controller implement a scatter-gather engine to access the OHCI data structures from system memory in an efficient manner.

22.4.11 L3 Interconnect Registers

The USB host controller connects to an open-core protocol interface (OCPI) arbiter, which then connects to the transfer controller OCPI port. There are no OCPI arbiter register settings that affect USB host controller access to system memory, but the transfer controller OCPI port must be properly configured to allow access to system memory.

Because the USB host must be able to access system memory, the OMAP2420 security features must be properly configured to allow USB host controller access to system memory.

22.5 USB Device Controller

The USB device controller supports the implementation of a full-speed device compatible with the USB 2.0 specification and the USB 1.1 specification.

It provides an interface between the MPU subsystem and the USB wire and handles USB transactions with minimal MPU software intervention.

The USB device controller module supports one control endpoint (EP0), up to 15 IN endpoints, and up to 15 OUT endpoints. The exact endpoint configuration is software programmable. The specific items of a configuration for each endpoint are the size (in bytes), the direction (IN, OUT), the type (bulk/interrupt or ISO), and the associated number.

The USB device controller module also supports three DMA channels for IN endpoints and three DMA channels for OUT endpoints for either bulk/interrupt or ISO transactions.

22.5.1 USB Device Transactions

There is an interrupt to the MPU subsystem at the end of a USB transaction if the MPU subsystem has actions to perform. Isochronous transactions are an exception because isochronous interrupt information is available at SOF interrupts. The MPU ISR code determines which endpoint and direction caused the interrupt, and then acts appropriately. The following subsections detail the activities surrounding USB transactions that are not part of a DMA transfer. Cases where a transaction occurs before the previous one is handled by the MPU subsystem are not considered here. The information is organized so that each section deals with one type and direction of transaction, such as:

☐ Nonisochronous, nonsetup OUT transactions
☐ Nonisochronous IN transactions
☐ Isochronous OUT transactions
☐ Isochronous IN transactions

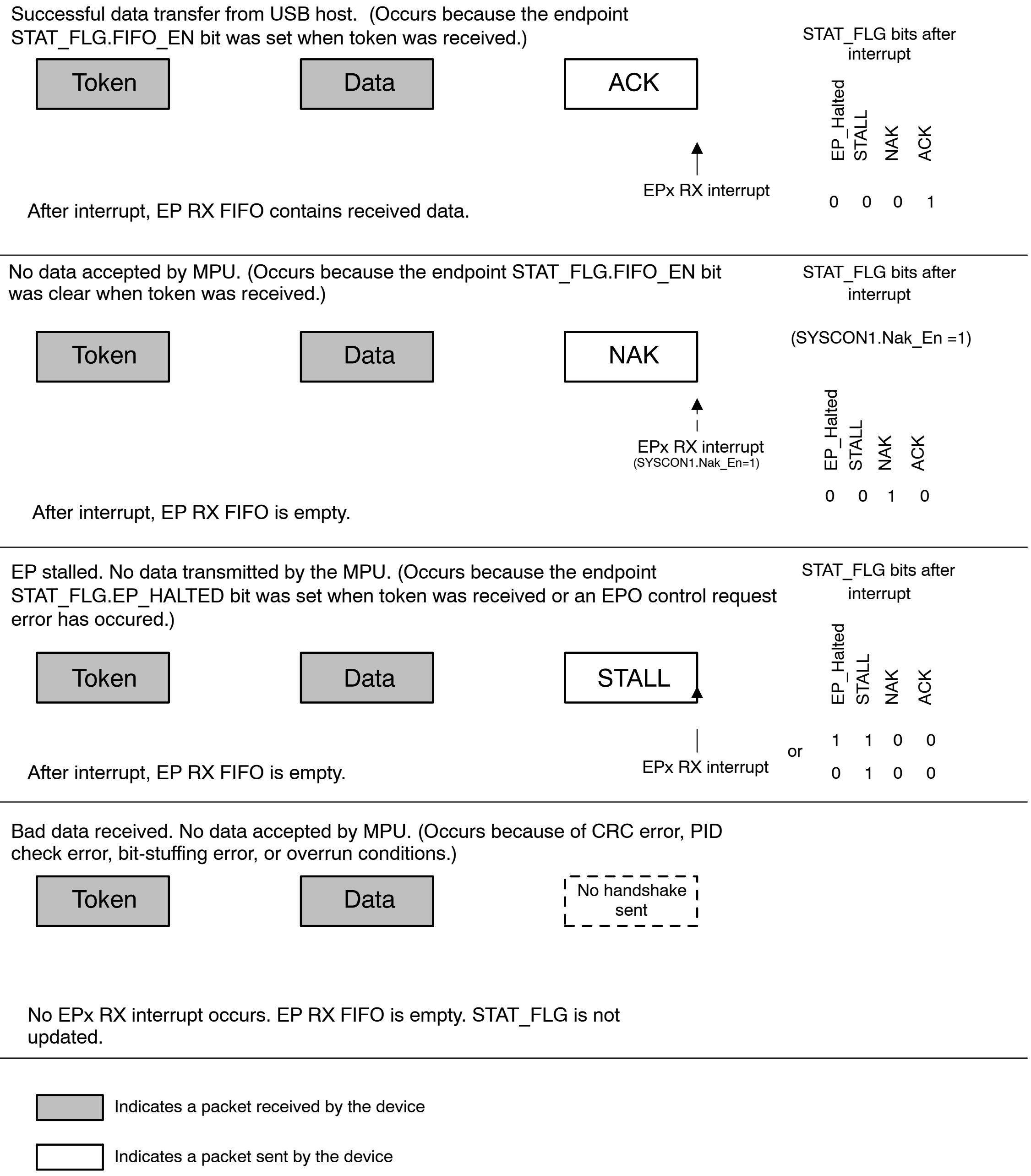
This lets each section focus on a specific style of transaction without adding the confusion of special cases related to other styles.

22.5.2 Nonisochronous, Nonsetup OUT (USB HOST -> MPU) Transactions

Nonisochronous, nonsetup OUT transactions are USB transactions in which data is moved from the USB host to the MPU subsystem, the USB handshaking protocols are in effect, and data transmission is ensured. These types of transactions apply to all OUT transactions on bulk and interrupt endpoint types and to nonsetup transactions on control endpoints.

Figure 22−28 shows the various USB protocol conditions that can occur during nonisochronous, nonsetup OUT transactions. It also shows the three phases that can occur in an OUT transaction, the direction of information flow for each phase, when endpoint interrupts are generated, and the resulting STAT_FLG bits for the endpoint. The top three cases show the normal USB handshaking modes: acknowledge ACK (good data received), NAK (device not ready to receive data), and STALL (device in a condition where the endpoint cannot handle OUT transactions). The fourth case shows an abnormal instance where the token packet or the data packet was received with errors. The RX FIFO contains valid receive data under the first (ACK) case.

Figure 22−28. Nonisochronous, Noncontrol OUT Transaction Phases and Interrupts


Successful data transfer from USB host. (Occurs because the endpoint STAT_FLG.FIFO_EN bit was set when token was received.)
STAT_FLG bits after interrupt
Token
Data
ACK
EP_Halted
STALL
NAK
ACK
EPx RX interrupt
0 0 0 1
After interrupt, EP RX FIFO contains received data.
No data accepted by MPU. (Occurs because the endpoint STAT_FLG.FIFO_EN bit was clear when token was received.)
STAT_FLG bits after interrupt
(SYSCON1.Nak_En =1)
Token
Data
NAK
EPx RX interrupt
(SYSCON1.Nak_En=1)
EP_Halted
STALL
NAK
ACK
0 0 1 0
After interrupt, EP RX FIFO is empty.
EP stalled. No data transmitted by the MPU. (Occurs because the endpoint STAT_FLG.EP_HALTED bit was set when token was received or an EPO control request error has occured.)
STAT_FLG bits after interrupt
Token
Data
STALL
EP_Halted
STALL
NAK
ACK
1 1 0 0
or
0 1 0 0
EPx RX interrupt
After interrupt, EP RX FIFO is empty.
Bad data received. No data accepted by MPU. (Occurs because of CRC error, PID check error, bit-stuffing error, or overrun conditions.)
Token
Data
No handshake sent
No EPx RX interrupt occurs. EP RX FIFO is empty. STAT_FLG is not updated.
Indicates a packet received by the device
Indicates a packet sent by the device

22.5.2.1 Nonisochronous, Noncontrol OUT Endpoint Handshaking Conditions

An endpoint SET_FIFO_EN bit CTRL[2] provides the main control that allows the endpoint to receive OUT transaction data sucessfully. If at the beginning of an OUT transaction to an endpoint, the endpoint FIFO_EN bit STAT_FLG[2] is set to 1, the USB module is allowed to accept the OUT transaction data to the RX FIFO, and when the transaction completes, the USB module can return ACK to the PC to indicate that the data was received correctly (see the first case in Figure 22−28). If, however, the endpoint FIFO_EN bit STAT_FLG[2] is set to 0 at the beginning of an OUT transaction to the endpoint, the USB module returns NAK during the handshake phase to indicate that the endpoint did not accept the data (see the second case in Figure 22−28).

The USB host need not send a whole RX FIFO worth of data to the endpoint during an OUT transaction. In this case, the RX FIFO is not full when the endpoint RX interrupt is generated. The MPU code must be careful not to read too much data. MPU code must read the RXF_COUNT value RXFSTAT[9:0] before reading data from the RX FIFO.

At the end of a USB OUT transaction to an endpoint where the data is accepted (ACKed), the hardware clears the endpoint FIFO_EN bit STAT_FLG[2]. Once the MPU software has dealt with the OUT transaction data in the endpoint RX FIFO, it must reenable the endpoint OUT transaction reception by setting the endpoint SET_FIFO_EN bit CTRL[2]. MPU software can use the endpoint SET_FIFO_EN bit CTRL[2] as a receive flow control mechanism.

Acknowledged (ACK) Transactions

When the OUT transaction to an endpoint completes, the USB module issues an endpoint-specific interrupt to the MPU subsystem and STAT_FLG is updated. In response to the endpoint interrupt, the MPU subsystem must read the EPN_STAT register to identify the endpoint causing the interrupt, and then write 1 to the interrupt bit to clear it. The MPU subsystem must then set the EP_NUM bits EP_NUM[3:0] to the endpoint number and the EP_SEL bit EP_NUM[5] to 1, and then read the endpoint status from STAT_FLG. The ACK bit STAT_FLG[3] is set to indicate that the endpoint received a transaction to which the USB module signaled ACK handshaking.

If the NON_ISO_FIFO_EMPTY bit STAT_FLG[1] is cleared, the transaction sends1 or more bytes of data (but less than or equal to the physical size of the endpoint RX FIFO), and the data is in the endpoint RX FIFO. The MPU subsystem determines the number of bytes to read from RX FIFO by reading the RXF_COUNT value RXFSTAT[9:0]. The MPU subsystem can then read RX data from the DATA register. Once the MPU subsystem reads the data from the FIFO, it sets the SET_FIFO_EN bit CTRL[2] to allow the next USB OUT transaction to the endpoint to be placed into the RX FIFO, and then clears the EP_SEL bit EP_NUM[5]. This clears the ACK bit STAT_FLG[3] for this endpoint to allow the next transaction status to be written into the STAT_FLG register.

Nonacknowledged (NAK) Transactions

The device can be configured by the NAK_EN bit SYSCON1[4] either to inform the MPU subsystem of a NAKed transaction or not. If the NAK_EN bit

SYSCON1[4] is cleared, no interrupt is asserted to the MPU subsystem if an OUT transaction completes with a NAK handshake and the NAK bit STAT_FLG[4] is not set. If the NAK_EN bit SYSCON1[4] is set, the USB module issues an endpoint-specific interrupt to the MPU subsystem when an OUT transaction to an endpoint completes, and the NAK bit STAT_FLG[4] is set. In response to the endpoint interrupt, the MPU subsystem must read the EPN_STAT register to identify the endpoint causing the interrupt, and then write 1 to the interrupt bit to clear it. The MPU subsystem must then set the EP_NUM bits EP_NUM[3:0] to the endpoint number and the EP_SEL bit EP_NUM[5] to 1, and then read the endpoint status from STAT_FLG. The NAK bit STAT_FLG[4] is set to indicate that the endpoint received a transaction to which the USB module signaled NAK handshaking.

The MPU subsystem must set the SET_FIFO_EN bit CTRL[2] to allow the next USB OUT transaction to the endpoint to be placed into the RX FIFO, and then clear the EP_SEL bit EP_NUM[5]. This clears the NAK bit STAT_FLG[4] for this endpoint to allow the next transaction status to be written into the STAT_FLG register.

22.5.2.2 Nonisochronous, Noncontrol OUT Transaction Error Conditions

STALLed Transactions

The USB module responds to an endpoint OUT transaction with a STALL handshake to indicate an error condition on the endpoint if either the endpoint EP_HALTED bit STAT_FLG[6] is set or a request error occurs (control transactions only). When an endpoint OUT transaction is given a STALL handshake, the endpoint STALL bit STAT_FLG[5] is set and an endpoint-specific interrupt is generated for the endpoint. The FIFO_EN bit STAT_FLG[2] is of lower priority than the EP_HALTED bit STAT_FLG[6]; when the EP_HALTED bit STAT_FLG[6] is set, transactions to the RX endpoint are stalled, regardless of the FIFO_EN value STAT_FLG[2]. If the FIFO_EN bit STAT_FLG[2] is set, the FIFO_EN bit STAT_FLG[2] is automatically cleared at the end of the STALLed transaction and RX FIFO is cleared.

In response to the endpoint interrupt, the MPU subsystem must read the EPN_STAT register to identify the endpoint causing the interrupt, and then write 1 to the interrupt bit to clear it. The MPU subsystem must then set the EP_NUM bits EP_NUM[3:0] to the endpoint number and the EP_SEL bit EP_NUM[5] to 1, and then read the endpoint status from STAT_FLG. The STALL bit STAT_FLG[5] is set to indicate that the endpoint received a transaction to which the USB module signaled STALL handshaking.

If the EP_HALTED bit STAT_FLG[6] is set by the MPU subsystem and can be removed, the MPU subsystem must set the CLR_HALT bit CTRL[7] to clear the condition, and then set the SET_FIFO_EN bit CTRL[2] to allow the next USB OUT transaction to the endpoint to be placed into the RX FIFO.

If the EP_HALTED bit STAT_FLG[6] is set in response to a SET_FEATURE request sent by the USB host, or if the bit is cleared (control transaction only), the MPU subsystem has no action to perform and must clear the EP_SEL bit

EP_NUM[5]. This clears the STALL bit STAT_FLG[5] for this endpoint and allows the next transaction status to be written into the STAT_FLG register.

Packet Errors

If a receive data error occurs during an endpoint OUT transaction (token or data packet), the USB module does not provide a handshake during the handshake phase of the transaction and no interrupt is asserted to the MPU subsystem (the fourth case shown in Figure 22−28).

Additionally, the endpoint RX FIFO is not filled, and the FIFO_EN bit STAT_FLG[2] is not cleared. If the MPU subsystem clears the RX FIFO during the data packet of an OUT transaction, no handshake is returned to the USB host to signal an error.

Sequence Bit Errors

If the core does not receive an expected DATA protocol identifier (PID) during an OUT transaction, the module automatically returns an ACK handshake to the USB host, regardless of the FIFO_EN bit STAT_FLG[2] (per the USB specification). Data is ignored, and no interrupt is asserted to the MPU subsystem.

This error occurs if the ACK handshake from a previous OUT transaction is received corrupted by the USB host.

22.5.2.3 Nonisochronous, Noncontrol OUT Endpoint FIFO Error Conditions

If the USB host tries to fill more data into an endpoint RX FIFO than the FIFO can hold, a FIFO overrun occurs. The USB module does not provide a handshake during the handshake phase of the transaction, and no interrupt is asserted to the MPU subsystem. Additionally, the endpoint RX FIFO is not filled, and the FIFO_EN bit STAT_FLG[2] bit is not cleared.

The MPU subsystem must not read more data from RX FIFO than the value indicated by the RXF_COUNT bits RXFSTAT[9:0].

22.5.3 Nonisochronous IN (MPU->USB HOST) Transactions

Nonisochronous IN transactions are USB transactions in which data is moved from the MPU subsystem to the USB host, the USB handshaking protocols are in effect, and data transmission is ensured. These transactions are the IN transactions that occur on control, bulk, and interrupt endpoints. These transactions do not ensure USB bandwidth.

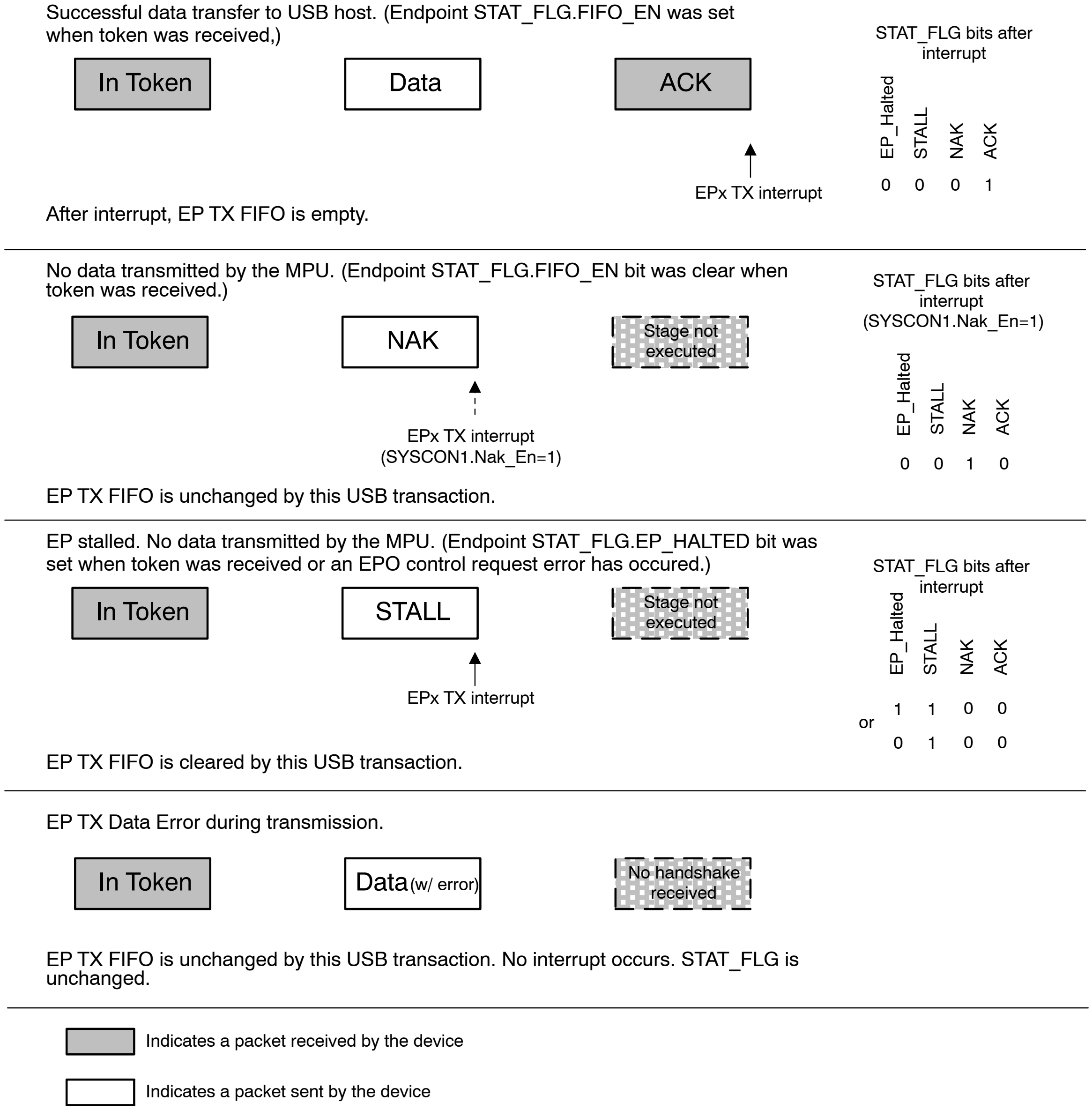
To provide data for an endpoint IN transaction, the MPU code writes the transmit data into the endpoint transmit FIFO. MPU code must first wait until the USB is done with any previous TX data for the endpoint (if data had previously been written to the TX FIFO). This must be done by proper response to endpoint-specific transmit interrupts. When an IN transaction to the endpoint occurs, if the endpoint FIFO_EN bit STAT_FLG[2] bit is set, the USB module sends any data that is in the endpoint TX FIFO during the data phase. If the

TX FIFO is empty and the FIFO_EN bit STAT_FLG[2] is set when an IN transaction to the endpoint occurs, a 0-byte data packet is sent.

After the endpoint previous transmit activity is taken care of, the MPU code gains access to the endpoint FIFO and status by setting the EP_SEL bit EP_NUM[5]. Then the MPU subsystem can write the new transmit data to the endpoint TX FIFO via the DATA register (being careful not to overflow the FIFO). Once all of the transmit data is written to the endpoint FIFO, MPU code sets the SET_FIFO_EN bit CTRL[2] to allow the USB to use the endpoint TX FIFO, and then clears the EP_SEL bit EP_NUM[5]. The data in the endpoint TX FIFO is sent to the USB host the next time an IN transaction to the endpoint occurs.

Figure 22−29 shows the various USB protocol conditions that can occur during nonisochronous IN transactions. It diagrams the three phases of the IN transaction, the direction of information flow for each phase, when endpoint-specific interrupts are generated, and the resulting STAT_FLG bits for the endpoint. The top three cases show the normal USB handshaking modes: ACK (data sent by USB module and received properly by the USB host), NAK (device not ready to send data to USB host), and STALL (device in a condition where the endpoint cannot handle IN transactions). The fourth case shows an abnormal case in which there is an error either in the token packet received by the core or in the data packet received by the USB host.

Figure 22−29. Nonisochronous IN Transaction Phases and Interrupts


Successful data transfer to USB host. (Endpoint STAT_FLG.FIFO_EN was set when token was received,)
In Token
Data
ACK
EPx TX interrupt
STAT_FLG bits after interrupt
EP_Halted
STALL
NAK
ACK
0 0 0 1
After interrupt, EP TX FIFO is empty.
No data transmitted by the MPU. (Endpoint STAT_FLG.FIFO_EN bit was clear when token was received.)
In Token
NAK
Stage not executed
EPx TX interrupt (SYSCON1.Nak_En=1)
STAT_FLG bits after interrupt (SYSCON1.Nak_En=1)
EP_Halted
STALL
NAK
ACK
0 0 1 0
EP TX FIFO is unchanged by this USB transaction.
EP stalled. No data transmitted by the MPU. (Endpoint STAT_FLG.EP_HALTED bit was set when token was received or an EPO control request error has occured.)
In Token
STALL
Stage not executed
EPx TX interrupt
STAT_FLG bits after interrupt
EP_Halted
STALL
NAK
ACK
1 1 0 0
or
0 1 0 0
EP TX FIFO is cleared by this USB transaction.
EP TX Data Error during transmission.
In Token
Data (w/ error)
No handshake received
EP TX FIFO is unchanged by this USB transaction. No interrupt occurs. STAT_FLG is unchanged.
Indicates a packet received by the device
Indicates a packet sent by the device

22.5.3.1 Nonisochronous IN Endpoint Handshaking

Per USB specifications for IN transactions, the USB host can provide only one of two handshakes to the USB function during the handshake phase: ACK, or no handshake at all. The first indicates successful transfer (see the first case in Figure 22−29), and the second indicates that the host received a garbled data packet (see the last case in Figure 22−29).

Acknowledged (ACK) Transactions

When the endpoint IN transaction completes on the USB bus with an ACK handshake, the endpoint generates an endpoint-specific interrupt to the MPU subsystem (see the first case in Figure 22−29). In response to the endpoint interrupt, the MPU subsystem must read the EPN_STAT register to identify the endpoint causing the interrupt, and then write 1 to the interrupt bit to clear it. The MPU subsystem must then set the EP_NUM bits EP_NUM[3:0] to the endpoint number, set the EP_DIR bit EP_NUM[4] to 1 (to signal an IN endpoint), set the EP_SEL bit EP_NUM[5] to 1, and then read the endpoint status from STAT_FLG. The ACK bit STAT_FLG[3] is set to indicate that the endpoint received an ACK handshake from the USB host, and the TX FIFO is empty (because any data that was in the TX FIFO was transmitted during the IN transaction).

If the MPU subsystem has more data to transmit to the USB host, it must fill the TX FIFO following the process outlined in the previous paragraph. It must then clear the EP_SEL bit EP_NUM[5]. This clears the ACK bit STAT_FLG[3] for this endpoint to allow the next transaction status to be written into the STAT_FLG register.

Nonacknowledged (NAK) Transactions

For the case in which the MPU subsystem is not ready to provide transmit data for transactions to an IN endpoint, the core provides a NAK handshake to the host for any USB IN transaction to that endpoint. Readiness to transmit data is signaled through the endpoint FIFO_EN bit STAT_FLG[2]; when set to 1, this bit indicates that data in the TX FIFO can be sent to the USB host. When the endpoint FIFO_EN bit STAT_FLG[2] bit is set to 0 and an IN transaction to the endpoint occurs, a NAK handshake is sent, indicating that the MPU subsystem is not ready to handle the request.

If the NAK_EN bit SYSCON1[4] is cleared, when the NAK handshake is sent in the data packet portion of the transaction to the IN endpoint, STAT_FLG is not updated and no endpoint-specific interrupt to the MPU subsystem is generated. If the NAK_EN bit SYSCON1[4] is set, when the NAK handshake is sent in the data packet portion of the transaction to the IN endpoint, the NAK bit STAT_FLG[4] is set and an endpoint-specific interrupt to the MPU subsystem is generated.

In response to the endpoint interrupt, the MPU subsystem must read the EPN_STAT register to identify the endpoint causing the interrupt, and then write 1 to the interrupt bit to clear it. The MPU subsystem must then set the EP_NUM bits EP_NUM[3:0] to the endpoint number, set the EP_DIR bit EP_NUM[4] to 1 (to signal an IN endpoint), set the EP_SEL bit EP_NUM[5] to 1, and then read the endpoint status from STAT_FLG. The NAK bit STAT_FLG[4] is set to indicate that the endpoint sent a NAK handshake to the USB host.

If the MPU subsystem has data to transmit to the USB host, it must fill the TX FIFO following the process explained in the previous paragraph. The MPU subsystem must then clear the EP_SEL bit EP_NUM[5], which clears the NAK bit STAT_FLG[4] for this endpoint to allow the next transaction status to be written into the STAT_FLG register. Signaling NAK does not cause the endpoint TX FIFO to be cleared (because the MPU subsystem still retains control of the FIFO).

Signaling a NAK handshake for several endpoint transactions in a row can cause the PC host to discard the transaction; therefore, NAK is not necessarily a good mechanism in cases where the MPU subsystem cannot service a request for a long period of time.

22.5.3.2 Nonisochronous IN Transaction Error Conditions

STALLed Transactions

The USB module sends a STALL handshake to the USB host during the data phase of the transaction to the IN endpoint either if the endpoint EP_HALTED flag STAT_FLG[6] is set or a request error occurs (control transaction only). A USB STALL handshake indicates that the device endpoint is in a condition in which it cannot transfer data and it instructs the USB host not to retry the transaction. The device typically requires intervention by some other mechanism to clear the condition, usually a control transfer via endpoint 0. The MPU subsystem can set the endpoint EP_HALTED bit STAT_FLG[6] by selecting the endpoint by writing the appropriate value in the EP_NUM register, and then setting the endpoint SET_HALT bit CTRL[6]; it can clear the endpoint by selecting the endpoint and then setting the endpoint CLR_HALT bit CTRL[7]. When the endpoint EP_HALTED bit STAT_FLG[6] is set, the endpoint signals STALL for its IN transactions until the HALT condition is cleared. When the STALL handshake is sent in response to a transaction to the endpoint, the STALL bit STAT_FLG[5] is set, and an endpoint-specific interrupt to the MPU subsystem is generated.

In response to the endpoint interrupt, the MPU subsystem must read the EPN_STAT register to identify the endpoint that is causing the interrupt, and then write 1 to the interrupt bit to clear it. The MPU subsystem must then set the EP_NUM bits EP_NUM[3:0] to the endpoint number, set the EPDIR bit EP_NUM[4] to 1 (to signal an IN endpoint), set the EP_SEL bit EP_NUM[5] to 1, and then read the endpoint status from STAT_FLG. The STALL bit STAT_FLG[5] is set to indicate that the endpoint sent a STALL handshake to the USB host. The MPU subsystem must then clear the EP_SEL bit EP_NUM[5]. This clears the STALL bit STAT_FLG[5] for this endpoint and allows the next transaction status to be written into the STAT_FLG register.

Except for control endpoint 0, separate endpoint halt bits are defined for each direction; therefore, for a given endpoint number, TX can be halted when RX is not.

Packet Errors

If an error (cyclic redundancy check [CRC], bit-stuffing, or PID check) occurs during the token packet of a USB IN transaction to a nonisochronous endpoint, the USB block ignores the transaction. No endpoint-specific interrupt to the MPU subsystem occurs for transactions with corrupted packets. If the MPU subsystem clears the TX FIFO during the data packet of an IN transaction, a bit-stuffing error is forced.

If the USB host returns no handshake after an IN transaction (in case of an error during transmission), the USB device controller detects after a time-out that an error has occurred. The data to transmit is still in the TX FIFO to be resent during the next IN transaction, the FIFO_EN bit STAT_FLG[2] is not cleared, and no interrupt is asserted to the MPU subsystem.

22.5.3.3 Nonisochronous IN Endpoint FIFO Error Conditions

The MPU subsystem cannot write more data into the TX FIFO than the configured FIFO size.

22.5.4 Isochronous OUT (USB Host-> MPU) Transactions

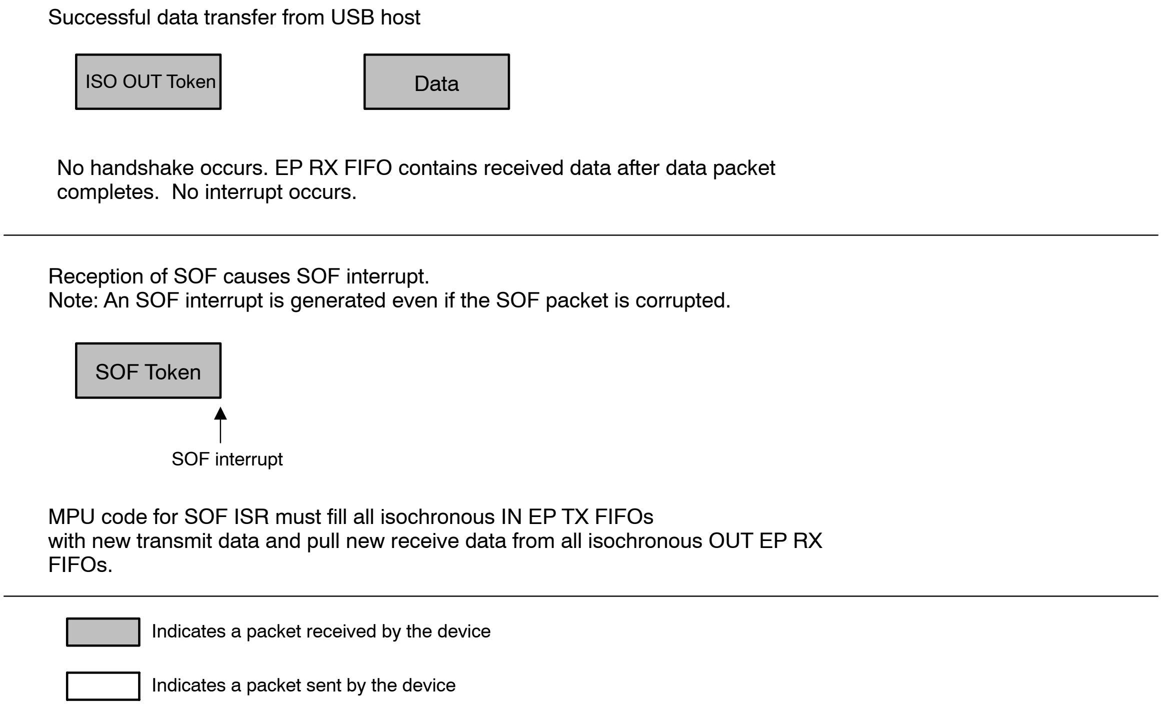
Isochronous OUT transactions are USB transactions in which a given amount of data is transferred from the USB host to the USB device controller module device every 1-ms USB frame. No USB handshaking is provided, and no endpoint-specific interrupt to the MPU subsystem is generated when an isochronous OUT transaction completes. The MPU subsystem must handle isochronous OUT data at each SOF interrupt.

At every SOF interrupt, for each isochronous OUT endpoint, MPU code must select the endpoint by writing the appropriate value in the EP_NUM register and check the ISO_FIFO_EMPTY bit STAT_FLG[9]. If the RX FIFO contains data, code must read the RXF_COUNT value RXFSTAT[9:0] (if the number of bytes to read from RX FIFO is not known), read all the bytes from RX FIFO via the data register, and then clear the EP_SEL bit EP_NUM[5].

Because the USB transaction for the isochronous endpoint can occur at any time during the USB 1-ms frame, the USB interface implements a double-buffering of the endpoint receive data FIFO. The endpoint includes two FIFOs, each of which is the length of the configured isochronous endpoint. At all times, one of the two FIFOs is foreground and the other is background. The USB interface side of the USB module is allowed to write to the background RX FIFO, and the MPU subsystem is allowed to read from the foreground RX FIFO. The designations foreground and background are swapped at each SOF. Isochronous endpoint FIFOs in the background are always enabled to the USB, whereas FIFOs in the foreground are enabled to the MPU subsystem.

Figure 22−30 shows the two phases (ISO OUT token and data) of an isochronous OUT data transfer in the top portion of the figure. No endpoint-specific interrupt to the MPU subsystem is generated for the isochronous OUT transaction. The data for isochronous endpoints are instead handled by the MPU subsystem at each SOF interrupt, which is shown as the second case in Figure 22−30.

Figure 22−30. Isochronous OUT Transaction Phases and Interrupts



Successful data transfer from USB host

No handshake occurs. EP RX FIFO contains received data after data packet completes. No interrupt occurs.

Reception of SOF causes SOF interrupt.
Note: An SOF interrupt is generated even if the SOF packet is corrupted.

MPU code for SOF ISR must fill all isochronous IN EP TX FIFOs with new transmit data and pull new receive data from all isochronous OUT EP RX FIFOs.

Indicates a packet received by the device
Indicates a packet sent by the device

22.5.4.1 Isochronous OUT Endpoint Handshaking

Because isochronous endpoint transactions have no handshake packets, the STALL bit STAT_FLG[5], the NAK bit STAT_FLG[4], and the ACK bit STAT_FLG[3] for isochronous endpoints always return 0. Because there is no handshake, the endpoint-specific interrupt for isochronous endpoints is not used.

22.5.4.2 Isochronous OUT Transaction Error Conditions

If the MPU subsystem fails to read all of the data in the ISO OUT endpoint foreground FIFO by the time the foreground and background FIFOs are switched (at the next SOF), the endpoint FIFO that is being switched to the background is flushed, and the DATA_FLUSH bit STAT_FLG[13] is asserted for the duration of the next frame.

There is no special indication for the case in which the USB host does not provide a transaction to an ISO OUT endpoint during a frame, but once the FIFO that was background in that frame becomes foreground, the FIFO is empty. (A 0-length data ISO OUT transaction also results in an empty FIFO and cannot be distinguished from a missed ISO OUT transaction.)

If an ISO OUT transaction occurs with a data error (CRC, PID check, or bitstuffing), the RX FIFO is empty at the next SOF interrupt, and the ISO_ERR bit STAT_FLG[12] is asserted for the duration of the next frame.

22.5.4.3 Isochronous OUT Endpoint FIFO Error Conditions

The MPU subsystem must never read more data than the value given by the RXF_COUNT bits RXFSTAT[9:0].

If the USB host sends more data than the FIFO can contain, the FIFO is cleared and the ISO_ERR bit STAT_FLG[12] is set at the next SOF interrupt. A properly configured USB system does not do this.

Note:

Both foreground and background isochronous FIFOs are cleared when the CLR_EP bit CTRL[1] bit is set.

22.5.5 Isochronous IN (MPU->USB Host) Transactions

Isochronous IN transactions are USB transactions in which a given amount of data is transferred from the USB device controller module device to the USB host every 1-ms USB frame. No handshaking is provided.

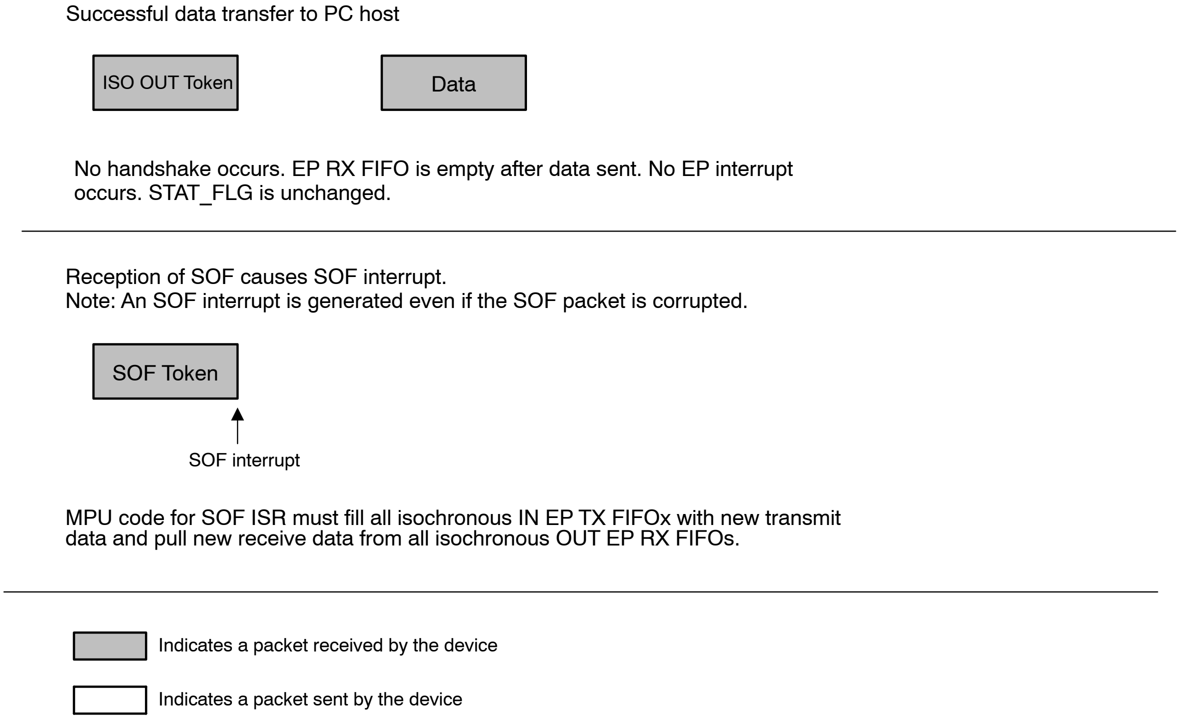
The USB module provides double-buffering of data for ISO IN endpoints. The background FIFO is the source of data for IN transactions to the ISO endpoint, and the foreground FIFO can be written to by the MPU subsystem. When an IN transaction to an ISO endpoint occurs, the USB module sends all data found in the endpoint background TX FIFO. The MPU subsystem provides new data to the isochronous IN endpoint foreground TX FIFO at each SOF interrupt.

In response to the SOF interrupt, for each isochronous IN endpoint, MPU code selects the endpoint (using the EP_NUM register), and then fills the endpoint TX FIFO (through the DATA register). When all transmit data are written to the FIFO, the MPU code must clear the EP_SEL bit EP_NUM[5].

Because the USB transaction for the isochronous endpoint can occur at any time during the USB 1-ms frame, the USB interface implements a double-buffering of the endpoint transmit data FIFO. The endpoint includes two FIFOs, each of which is the length of the configured isochronous endpoint. At all times, one of the two FIFOs is foreground and the other is background. The USB interface side of the USB module is allowed to read from the background TX FIFO, and the MPU subsystem is allowed to write to the foreground TX FIFO. The designations foreground and background are swapped, and the new background TX FIFO is cleared at each SOF. Because ISO endpoints implement double-buffering, ISO endpoints do not control access to the FIFOs through the SET_FIFO_EN bit CTRL[2]; the SET_FIFO_EN bit CTRL[2] and the FIFO_EN bit STAT_FLG[2] are not implemented for ISO IN endpoints.

Figure 22−31 shows the transaction phases associated with isochronous IN transactions and the SOF transaction. No endpoint-specific interrupt to the MPU subsystem is generated as a result of an isochronous IN transaction, and there is no handshake phase. The SOF transaction causes an SOF interrupt to the MPU subsystem; it is assumed that the MPU subsystem refills the isochronous IN endpoint transmit FIFO at each SOF interrupt.

Figure 22−31. Isochronous IN Transaction Phases and Interrupts


Successful data transfer to PC host
ISO OUT Token
Data
No handshake occurs. EP RX FIFO is empty after data sent. No EP interrupt occurs. STAT_FLG is unchanged.
Reception of SOF causes SOF interrupt.
Note: An SOF interrupt is generated even if the SOF packet is corrupted.
SOF Token
SOF interrupt
MPU code for SOF ISR must fill all isochronous IN EP TX FIFOx with new transmit data and pull new receive data from all isochronous OUT EP RX FIFOs.
Indicates a packet received by the device
Indicates a packet sent by the device

22.5.5.1 Isochronous IN Endpoint Handshaking

Because isochronous endpoint transactions have no handshake packets, the STALL bit STAT_FLG[5], the NAK bit STAT_FLG[4], and the ACK bit STAT_FLG[3] for isochronous endpoints always return 0. Because there is no handshake, there is no endpoint-specific interrupt to the MPU subsystem to report handshake results for isochronous endpoints.

22.5.5.2 Isochronous IN Transaction Error Conditions

If the USB host failed to complete an ISO IN transaction in the previous frame, and if data are present in TX FIFO to be sent at the IN transaction, the MISS_IN bit STAT_FLG[14] is asserted for the duration of the following frame. If the ISO IN endpoint is cleared in the middle of a USB transaction to the background FIFO, the macro forces a bit-stuffing error for the ISO transaction.

22.5.5.3 Isochronous IN Endpoint FIFO Error Conditions

If the MPU subsystem tries to overfill the configured endpoint FIFO, any data written to the DATA register after the TX FIFO is full is lost; however, any data that was successfully put into the FIFO is transmitted when that FIFO is the background FIFO, and an IN transaction for that endpoint occurs. Because an ISO TX FIFO is cleared automatically on the toggle from background to foreground, there is no reason to clear the FIFO. However, if the MPU subsystem does not wish to send the data it wrote, clearing the endpoint is the only way to do this.

22.5.6 Control Transfers on Endpoint 0

Control transfers on endpoint 0 include control write and control read transfers. Control write and control read transfers are each composed of two or more transactions to endpoint 0. Additionally, the USB device controller module can autodecode some control write and control read transfers. These operations are summarized in Figure 22−32 and Figure 22−33. An IN or an OUT transaction is received out of a control request. This transaction is automatically stalled by the core.

Figure 22−32. Stages and Transaction Phases of Autodecoded Control Transfers

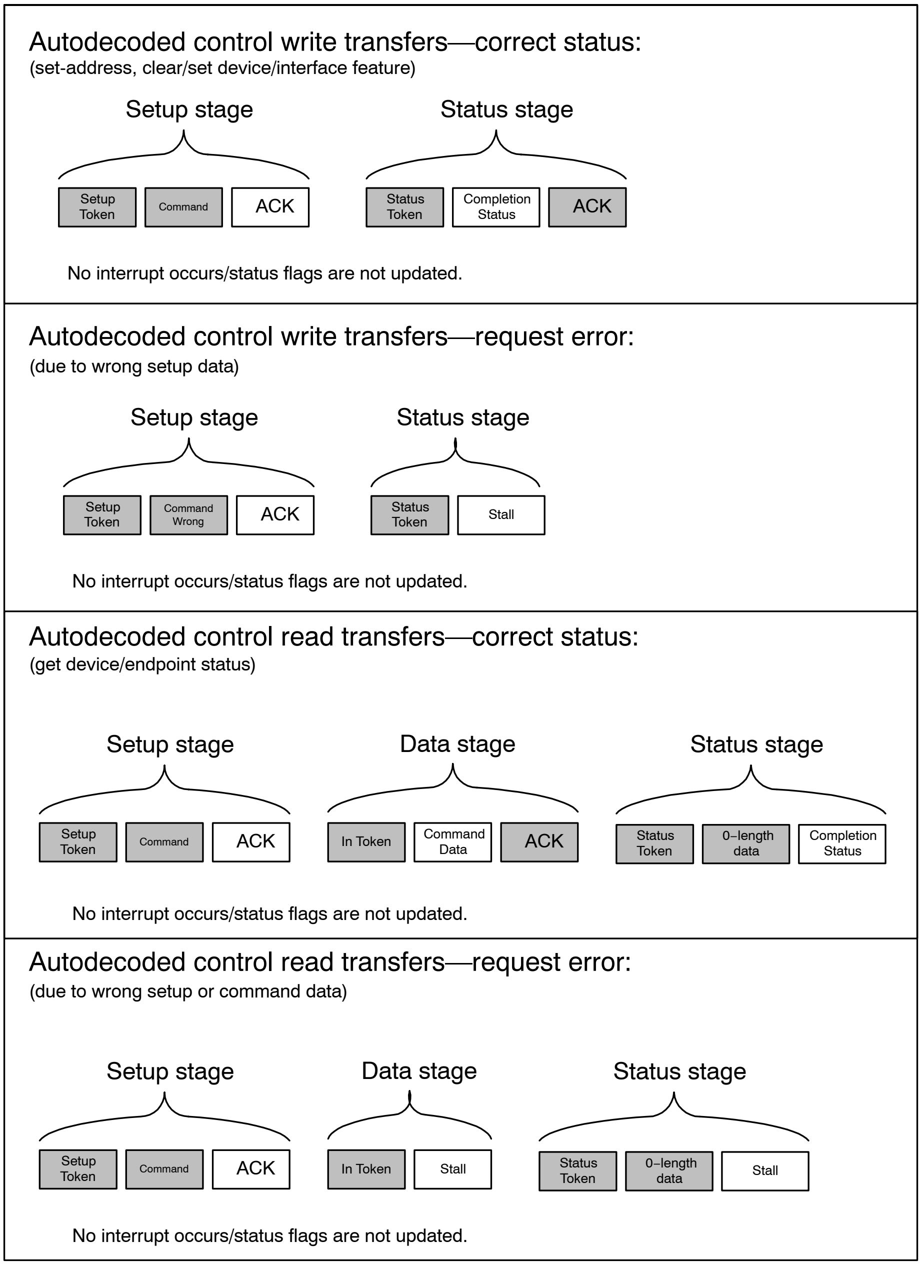
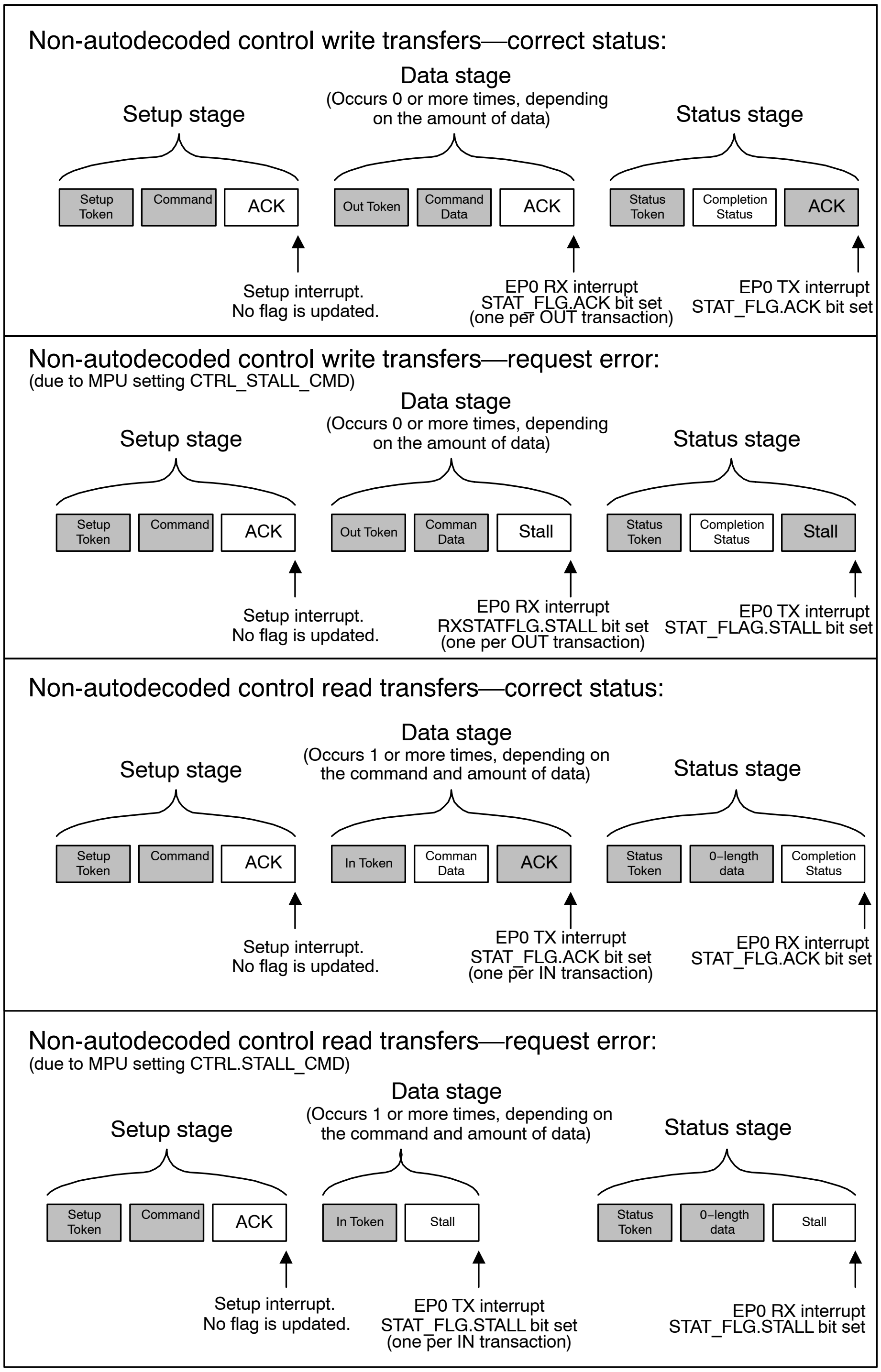

Autodecoded control write transfers—correct status:
(set-address, clear/set device/interface feature)
Setup stage
Status stage
Setup Token
Command
ACK
Status Token
Completion Status
ACK
No interrupt occurs/status flags are not updated.
Autodecoded control write transfers—request error:
(due to wrong setup data)
Setup stage
Status stage
Setup Token
Command Wrong
ACK
Status Token
Stall
No interrupt occurs/status flags are not updated.
Autodecoded control read transfers—correct status:
(get device/endpoint status)
Setup stage
Data stage
Status stage
Setup Token
Command
ACK
In Token
Command Data
ACK
Status Token
0−length data
Completion Status
No interrupt occurs/status flags are not updated.
Autodecoded control read transfers—request error:
(due to wrong setup or command data)
Setup stage
Data stage
Status stage
Setup Token
Command
ACK
In Token
Stall
Status Token
0−length data
Stall
No interrupt occurs/status flags are not updated.

Figure 22−33. Stages and Transaction Phases of Non-Autodecoded Control Transfers


Non-autodecoded control write transfers—correct status:
Setup stage
Data stage
(Occurs 0 or more times, depending on the amount of data)
Status stage
Setup Token
Command
ACK
Out Token
Command Data
ACK
Status Token
Completion Status
ACK
Setup interrupt. No flag is updated.
EP0 RX interrupt STAT_FLG.ACK bit set (one per OUT transaction)
EP0 TX interrupt STAT_FLG.ACK bit set
Non-autodecoded control write transfers—request error:
(due to MPU setting CTRL_STALL_CMD)
Setup stage
Data stage
(Occurs 0 or more times, depending on the amount of data)
Status stage
Setup Token
Command
ACK
Out Token
Comman Data
Stall
Status Token
Completion Status
Stall
Setup interrupt. No flag is updated.
EP0 RX interrupt RXSTATFLG.STALL bit set (one per OUT transaction)
EP0 TX interrupt STAT_FLAG.STALL bit set
Non-autodecoded control read transfers—correct status:
Setup stage
Data stage
(Occurs 1 or more times, depending on the command and amount of data)
Status stage
Setup Token
Command
ACK
In Token
Comman Data
ACK
Status Token
0−length data
Completion Status
Setup interrupt. No flag is updated.
EP0 TX interrupt STAT_FLG.ACK bit set (one per IN transaction)
EP0 RX interrupt STAT_FLG.ACK bit set
Non-autodecoded control read transfers—request error:
(due to MPU setting CTRL.STALL_CMD)
Setup stage
Data stage
(Occurs 1 or more times, depending on the command and amount of data)
Status stage
Setup Token
Command
ACK
In Token
Stall
Status Token
0−length data
Stall
Setup interrupt. No flag is updated.
EP0 TX interrupt STAT_FLG.STALL bit set (one per IN transaction)
EP0 RX interrupt STAT_FLG.STALL bit set

Non-autodecoded control read and control write transfers are sets of transactions that occur on endpoint 0 and have specific USB protocol meaning; they are not, however, handled automatically by the core. The USB device controller block automatically provides an ACK handshake for the setup stage transaction, but the data and status stage transaction handshaking is accomplished using the usual RX and TX control bits that affect transaction handshaking.

A general USB interrupt to the MPU subsystem occurs at the end of each transaction of each stage of a control transfer. The MPU subsystem must perform the following actions to act on non-autodecoded control transfers:

- ☐ Process the setup phase setup USB interrupt. The MPU subsystem reads the control transfer command from the setup FIFO and decodes the command. For control reads, the MPU subsystem fetches the requested read data and places it (or as much of the read data as fits) into the endpoint 0 FIFO, and then enables the endpoint 0 FIFO. For control writes, the MPU code enables only the endpoint 0 FIFO. MPU code also sets any flags required to process endpoint 0 USB interrupts during the control transfer.
- ☐ Process the data phase endpoint 0 general USB interrupt(s). For control reads, the data phase general USB endpoint 0 TX interrupt indicates that the previously provided transmit data is sent. Any additional data must be written to the endpoint 0 FIFO. For control writes, the write data must be pulled from the endpoint 0 FIFO, and when all control write data is available, the MPU subsystem interprets the write data and acts on the write request. After handling the last data phase interrupt, the MPU subsystem must set the endpoint 0 control bits to signal the desired status to the host.
- ☐ Process the status stage endpoint 0 general USB interrupt. The MPU subsystem provides its completion status to the USB host during this stage, either through the status in the data phase of the transaction (for control write transfers) or by the handshake phase of the transaction (for control read transfers).

Autodecoded control read and control write transfers are sets of transactions that occur on endpoint 0 that have specific USB protocol meaning and are handled automatically by the USB device controller block without any intervention by the MPU subsystem. The USB device controller block handles all handshaking automatically and without regard to the endpoint 0 control bits that affect normal (noncontrol transfer) transaction handshaking. No interrupt is asserted to the MPU subsystem during autodecoded control transfers.

If no request defined by the USB1.1 specification is associated with the data of the setup phase, the request is stalled by the core and the MPU subsystem is not informed of its occurrence (autodecoded).

When a setup token is identified, the USB decode module must monitor the setup stage data packet, decode it, and determine whether it is an autodecoded or a non-autodecoded transfer and whether it is a control read or a control write. If it is a valid non-autodecoded request, the setup FIFO is immediately cleared and control of the FIFO is immediately taken away from the MPU subsystem (if the MPU subsystem had control of the FIFO). New setup data are placed into the setup FIFO, and the setup interrupt flag is set (SETUP bit IRQ_SRC[2]).

In response to the setup interrupt, the MPU subsystem must select the setup FIFO by setting the SETUP_SEL bit EP_NUM[6]. This clears the SETUP flag IRQ_SRC[2]. The MPU subsystem must then read 8 bytes from the setup FIFO, clear the SETUP_SEL bit EP_NUM[6], and confirm that the SETUP bit

IRQ_SRC[2] has not been reset by a new setup transaction. If the SETUP flag IRQ_SRC[2] is asserted, the MPU subsystem must discard the previously read data and handle the new setup packet as previously explained. Thus, the MPU subsystem never misses a new occurring setup transaction (per the USB 1.1 specification).

22.5.6.1 Autodecoded Control Write Transfers

For set address control write transfers, the USB address provided in the setup token is captured to the USB module device address register. If the new address is not 0, the device moves into addressed state (ADD bit DEVSTAT[2] set), if it was not already addressed.

For set and clear feature control writes, the appropriate feature information bit is set or cleared. When a set or clear feature transfer occurs to set or clear the device remote wake-up feature, the R_WK_OK bit DEVSTAT[6] is set or cleared, as appropriate. If a set or clear interface feature occurs, the request is automatically stalled by the core because no feature is defined for interface (see the USB 1.1 specification).

According to the USB 1.1 specification, a SET_ADDRESS request is effective after the status stage of the request, even if the status stage does not end with an ACK handshake. SET/CLEAR_FEATURE requests are effective after the setup stage, even if no status stage occurs.

Autodecoded Control Write Transfer Handshaking

The USB device controller module automatically provides ACK handshaking for all transactions of all stages of autodecoded control write transfers, unless a corrupted packet is received; the USB module ignores corrupted packets. The SET_FIFO_EN bit CTRL[2] and the STALL_CMD bit SYSCON2[5] have no effect on handshaking.

Autodecoded Control Write Transfer Error Conditions

If the token packet or the data packet of a setup stage transaction has an error (bad CRC, PID check, or bit-stuffing error), the USB block ignores the transaction. The USB block does not provide ACK handshaking in this case.

22.5.6.2 Autodecoded Control Read Transfers

Autodecoded control reads include the standard device requests GET_ENDPOINT and DEVICE_STATUS. Because these control read transfers access information that is kept in registers inside the USB module, MPU code is not involved in filling the read data into the TX FIFO.

The USB module returns the currently selected appropriate status information (depending on the wIndex value in the setup stage data packet) during the data phase of the single IN transaction of the data stage, and provides ACK as the handshake for the status stage handshake phase. The MPU subsystem receives no interrupt.

Autodecoded Control Read Transfer Handshaking

The USB device controller module automatically provides ACK handshaking for all transactions of all stages of autodecoded control read transfers, unless a corrupted token packet is received; the USB module ignores corrupted packets. The SET_FIFO_EN bit CTRL[2] and the STALL_CMD bit SYSCON2[5] have no effect on the handshaking. If the status packet has a DATA0 PID instead of a DATA1 PID, status is STALLed and no interrupt is asserted to the MPU subsystem. If the setup packet has a DATA1 PID instead of a DATA0 PID, setup transaction is ignored (error).

Autodecoded Control Read Transfer Error Conditions

If the token phase or the data phase of a setup stage transaction has an error (bad CRC, PID check, or bit-stuffing error), the USB block ignores the transaction. The USB block does not provide ACK handshaking in this case.

Data errors during the data stage of autodecoded control write transfers are handled in the standard way; any data stage transaction from the host where a data error occurs is ignored.

It is possible that the USB host sends a GET_ENDPOINT/DEVICE_STATUS request with a bad parameter. If the autodecode mechanism senses a bad parameter in the setup stage data phase, the autodecode mechanism causes a STALL handshake to be signaled during the data phase of the data stage and during the status stage.

22.5.6.3 Non-Autodecoded Control Write Transfers

Non-autodecoded control write transfers include the SET_/CLEAR_ ENDPOINT feature, SET_CONFIGURATION, SET_INTERFACE, SET_ DESCRIPTOR, and class- or vendor-specific control write transfers. Non-autodecoded control write transfers consist of two or three stages (setup, data [optional], and status).

The setup stage of a valid non-autodecoded control write transfer consists of one SETUP transaction from USB host to USB device. At the end of the setup stage handshake, the USB module generates an MPU general USB interrupt with the SETUP flag IRQ_SRC[2] set. The MPU subsystem must respond to this general USB interrupt by setting the SETUP_SEL bit EP_NUM[6], which clears the setup interrupt flag. The MPU subsystem must then read 8 bytes from the setup FIFO via the DATA register, clear the EP_SEL bit EP_NUM[5], and check the SETUP flag IRQ_SRC[2]. If the SETUP flag IRQ_SRC[2] is set, the MPU subsystem must discard the setup data it has just read and handle the new setup data packet following the same scheme. If the SETUP flag IRQ_SRC[2] is cleared, the MPU code interprets this request information and performs any application-specific activity required because of the setup stage request. If there is one or more data stage for the transfer, the MPU subsystem must set the SET_FIFO_EN bit CTRL[2] for endpoint 0 to allow the core to accept RX data from the coming OUT transaction.

The data stage for non-autodecoded control writes consists of zero or more OUT transactions. Transaction handshaking and interrupt generation applies for nonisochronous, noncontrol OUT endpoints. The MPU subsystem can cause NAK, STALL, or ACK signaling for the data stage transactions. If ACK was signaled on a given general USB interrupt, the MPU subsystem must respond by reading the data from the endpoint 0 RX FIFO and saving it for processing.

When the data stage completes, a status stage IN transaction occurs. The USB module provides handshaking to the USB host based on the endpoint 0 handshaking control FIFO_EN bit STAT_FLG[2]. The MPU subsystem can delay signaling completion of the control write transfer by forcing NAK handshaking to the host during the status stage (by holding the FIFO_EN bit STAT_FLG[2] 0), or causing ACK handshaking by setting the SET_FIFO_EN bit CTRL[2] with an empty endpoint 0 FIFO. An endpoint 0 TX general USB interrupt is sent to the MPU subsystem when the status stage completes.

After a SET_CONFIGURATION request, the device goes into addressed or configured state as soon as the MPU subsystem sets either the DEV_CFG bit SYSCON2[3] or the CLR_CFG bit SYSCON2[2].

Specific MPU-Required Actions

If the device receives a valid set endpoint halt feature request, it must set the appropriate SET_HALT control bit CTRL[6].

If the device receives a valid CLEAR_ENDPOINT halt feature request, it must set the appropriate RESET_EP bit CTRL[0] to clear the halt condition, clear the FIFO flags, and reset the data PID to DATA0 for the endpoint. If the specified endpoint number is 0, the MPU subsystem is only required to set the CLR_HALT bit CTRL[7] to clear the halt condition.

If the device receives a valid SET_CONFIGURATION request, it must reset all endpoints by setting the RESET_EP control bit CTRL[0], set the SELF_PWR bit SYSCON1[2] to the appropriate value, and then set halt conditions for endpoints not used by the default interface set for the configuration. If the device was addressed when the SET_CONFIGURATION was received, the MPU subsystem must write 1 to the DEV_CFG bit SYSCON2[3] to allow the device to enter the configured state (CFG bit DEVSTAT[3] set). If the device was configured when the SET_CONFIGURATION was received, and the new configuration value is 0, the MPU subsystem must write 1 to the CLR_CFG bit SYSCON2[2] to allow the device to return to the addressed state (CFG bit DEVSTAT[3] cleared).

If the device receives a valid set interface request, it must set the RESET_EP control bit CTRL[0] to reset all endpoints used by the interface set, and then set halt conditions for endpoints not used by this interface.

Other MPU-required actions are specific to the request and are not detailed in this document.

Non-Autodecoded Control Write Transfer Handshaking

Setup stage transactions that are valid are signaled ACK. Transactions with an invalid setup stage token or data packets are ignored and receive no handshake packet from the USB module, and no interrupt is generated.

Data stage handshaking for non-autodecoded control write transfers depends on the endpoint 0 FIFO_EN bit STAT_FLG[2], the EP_HALTED bit

STAT_FLG[6], and the STALL_CMD bit SYSCON2[5]. The MPU subsystem can delay completion of any transaction of the data stage by signaling NAK (by not setting SET_FIFO_EN bit CTRL[2]). The USB specification requires that once STALL is signaled in a control transfer, it must be signaled on that endpoint until the next setup token is received. Either the STALL_CMD bit SYSCON2[5] or the SET_HALT bit CTRL[6] (reflected in the EP_HALTED register bit STAT_FLG[6]) provides this functionality. The EP_HALTED bit STAT_FLG[6] does not reflect the forced STALL caused by the STALL_CMD bit SYSCON2[5]; it retains its previous value.

Status stage handshaking is controlled by the endpoint 0 FIFO_EN bit STAT_FLG[2] and the STALL_CMD bit SYSCON2[5]. Successful completion of a non-autodecoded control write transfer is indicated by the USB device controller module returning a 0-length data payload for the data phase of the status stage and an ACK handshake from the host for the handshake phase of the status stage. Although NAK handshaking can be used to indicate delays in completion of the requested control write, the USB host can choose to abort the control write after some number of NAKs.

Non-Autodecoded Control Write Transfer Error Conditions

If an error occurs during the control write, which the MPU subsystem itself cannot handle, it must signal STALL to the USB host for all subsequent transactions until a new setup token to endpoint 0 occurs. This is true for both data stage and status stage transactions. This is most conveniently done by setting endpoint 0 STALL_CMD bit SYSCON2[5], which stalls the remaining transactions of the remaining stages of a non-autodecoded control transfer, up to the reception of the next valid SETUP command.

Error conditions are handled in the same way as BULK/INTERRUPT transactions. If a packet is received corrupted, the core ignores the transaction and no interrupt is asserted.

22.5.6.4 Non-Autodecoded Control Read Transfers

Non-autodecoded control read transfers include the GET_INTERFACE_STATUS, GET_CONFIGURATION, GET_INTERFACE, GET_DESCRIPTOR, SYNCH_FRAME, and class- or vendor-specific control read transfers. Non-autodecoded control read transfers consist of three stages: setup, data, and status.

The setup stage of a valid non-autodecoded control read transfer consists of one SETUP transaction from the USB host to the USB device. At the end of the setup stage handshake, the USB module generates an MPU general USB interrupt with the SETUP flag IRQ_SRC[2] set. The MPU subsystem must respond to this general USB interrupt by setting the SETUP_SEL bit EP_NUM[6], which clears the setup interrupt flag. The MPU subsystem must then read 8 bytes from the setup FIFO using the DATA register, clear the EP_SEL bit EP_NUM[5], and check the SETUP flag IRQ_SRC[2]. If the SETUP flag IRQ_SRC[2] is set, the MPU subsystem must discard the setup data it has just read and handle the new setup data packet following the same scheme. If the SETUP flag IRQ_SRC[2] is cleared, the MPU code interprets this request information and then prepares data for the IN transaction that follows. This includes placing the data requested (or the first few bytes, if more than one FIFO worth of data is returned) into the endpoint 0 FIFO, and setting the SET_FIFO_EN bit CTRL[2].

The data stage of a control read transfer consists of one or more IN transactions. Transaction handshaking and interrupt generation applies for nonisochronous, noncontrol IN endpoints; the MPU subsystem can cause NAK, STALL, or ACK signaling for the data stage transactions. At endpoint 0 TX general USB interrupts, MPU code must move more data to the endpoint 0 FIFO, until the last bytes of the requested data are provided. Although SETUP packets have a defined payload length, the USB host can cancel the transaction at any time, without the status stage, and resend another SETUP command. The MPU code must be able to operate correctly in this situation.

When the data stage completes, a status stage OUT transaction occurs. The USB host sends a 0-length data packet, and the MPU code must return its completion status for the control read standard request through standard handshaking mechanisms.

Note:

When exactly what the host requested is returned, and that request is a multiple of the maximum packet size, no 0-length packet is required. A 0-length packet is required only when the amount of data the device must return is less than the amount requested by the host and the amount returned is a multiple of the maximum packet size (source: USB Forum).

Non-Autodecoded Control Read Transfer Handshaking

Handshaking for the setup stage of non-autodecoded control read transfers is forced by the USB module to always be ACK, unless there is a data error in the packet, in which case the USB module ignores the transaction. If the setup packet has a DATA1 PID instead of a DATA0 PID, the setup transaction is ignored (error).

Data stage handshaking for non-autodecoded control read transfers depends on the endpoint 0 FIFO_EN bit STAT_FLG[2], the EP_HALTED bit STAT_FLG[6], and the STALL_CMD bit SYSCON2[5]. The handshaking information is used during the data phase of the data stage transaction. The USB specification requires that once STALL is signaled in a control transfer, it must be signaled until the next setup token is received. The STALL_CMD bit SYSCON2[5] and the SET_HALT bit CTRL[6] (reflected through the EP_HALTED register bit STAT_FLG[6]) provide this functionality. The EP_HALTED bit STAT_FLG[6] does not reflect the forced STALL caused by the STALL_CMD bit SYSCON2[5]; it retains its previous value.

The status stage is controlled by the FIFO_EN bit STAT_FLG[2] and the STALL_CMD bit SYSCON2[5].

Successful completion of non-autodecoded control read transfers is indicated by the host sending an OUT token followed by an empty packet and the USB device controller responding with ACK. If the data packet sent by the USB host during the status stage of a control read request is not empty, the OUT transaction is accepted by the core, but OUT data is not put into the endpoint 0 RX FIFO. If the status packet has a DATA0 PID instead of a DATA1 PID, a STALLed is returned by the core and an interrupt is asserted.

Non-Autodecoded Control Read Transfer Error Conditions

If an error occurs during the control read, which the MPU subsystem itself cannot handle, it must signal STALL to the USB host for all subsequent transactions until a new setup token to endpoint 0 occurs. This is true for both data stage and status stage transactions. This is most conveniently done by setting endpoint 0 STALL_CMD bit SYSCON2[5], which stalls the remaining transactions of the remaining stages of a non-autodecoded control transfer, up to the reception of the next valid SETUP command.

Error conditions are handled in the same way as BULK/INTERRUPT transactions. The USB device controller module responds to control read status stage transactions that have a bad token or bad data by not sending a handshake packet. In both cases, the transaction is ignored and no general USB interrupt is generated to the MPU subsystem.

22.5.6.5 Autodecoded Versus Non-Autodecoded Control Requests

Table 22−28 lists the autodecoded versus the non-autodecoded control requests.

Table 22−28. Autodecoded Versus Non-Autodecoded Control Requests

Request	**Recipient**	**Status**	**MPU-Required Action**	**Device Behavior if Device Is Not Configured**
GET_ STATUS	Device	Autodecoded (Function of AUTODEC_ DIS register bit)	None	Device status is returned (SYSCON1.SELF_PWR and DEVSTAT.R_WK_OK bits).
	Interface	Non-autodecoded	The MPU must stall the command (via SYSCON2.STALL_ CMD bit) if interface number is not correct. No feature is defined for interface.	Command is passed to the MPU.

Notes:
1) Transactions on endpoints other than 0 are ignored if the device is not configured (addressed state).
2) If some endpoints are not used by the interface currently set, transactions on these endpoints are not ignored; the MPU must set the halt feature for the endpoint. This does not happen if the USB host works correctly.
3) If endpoint 0 is halted, per the USB 1.1 specification (see 9.4.5: Get_Status), all requests are stalled except GET_STATUS, CLEAR_FEATURE, and SET_FEATURE requests.
4) Requests are handled with respect to the USB 1.1 specification when specified as such, but many device reactions are not included in the specification.
5) During a SET_ADDRESS autodecoded command, only the 7 LSBs are significant for the new address (decimal value from 0 to 127); all others are discarded.

Table 22−28. Autodecoded Versus Non-Autodecoded Control Requests (Continued)

Request	Recipient	Status	MPU-Required Action	Device Behavior if Device Is Not Configured
	Endpoint	Autodecoded (Function of AUTODEC_DIS register bit)	None	The core automatically stalls the command if endpoint number is not 0.
CLEAR/SET FEATURE	Device	Autodecoded (Function of AUTODEC_DIS register bit)	None. (DS_CHG IT is asserted to the MPU after any DEVSTAT.R_WK_OK bit modification.)	The core handles the request.
	Interface	Autodecoded (Function of AUTODEC_DIS register bit)	None. (No feature is defined in USB 1.1 specification for interface. These requests are stalled.)	Command is stalled in any case.
	Endpoint	Non-autodecoded	The MPU must stall the command (through the SYSCON2.STALL_CMD bit) if the endpoint number/type/direction is not correct. The MPU must reset the EP after having handled the pending transactions (if CLEAR) or set halt condition (if SET). For EP 0, MPU must only clear or set halt condition; FIFO and data PID is always correct for next setup.	Command is passed to the MPU.
SET_ADDRESS	Device	Autodecoded	None (see Note 5). (Whether the device is addressed or not is available in DEVSTAT register. A valid SET_ADDRESS request with address number from 0 generates a DS_CHG interrupt to the MPU.)	Default: Device moves in the addressed state if address number is not 0. Addressed: Device takes the new address value or moves in default state if address number is 0. Configured: Request is STALLed.

Notes:
1) Transactions on endpoints other than 0 are ignored if the device is not configured (addressed state).
2) If some endpoints are not used by the interface currently set, transactions on these endpoints are not ignored; the MPU must set the halt feature for the endpoint. This does not happen if the USB host works correctly.
3) If endpoint 0 is halted, per the USB 1.1 specification (see 9.4.5: Get_Status), all requests are stalled except GET_STATUS, CLEAR_FEATURE, and SET_FEATURE requests.
4) Requests are handled with respect to the USB 1.1 specification when specified as such, but many device reactions are not included in the specification.
5) During a SET_ADDRESS autodecoded command, only the 7 LSBs are significant for the new address (decimal value from 0 to 127); all others are discarded.

Table 22−28. Autodecoded Versus Non-Autodecoded Control Requests (Continued)

Request	**Recipient**	**Status**	**MPU-Required Action**	**Device Behavior if Device Is Not Configured**
GET_ DESCRIPTOR	All	Non-autodecoded	The MPU must write descriptor data into endpoint 0 FIFO.	Command is passed to the MPU.
SET_ DESCRIPTOR	All	Non-autodecoded	The MPU must stall the command (through the SYSCON2.STALL_CMD bit) if it does not support set descriptor requests.	Command is passed to the MPU.
GET/SET CONFIGURATION	Device	Non-autodecoded	The MPU must stall the command (through the SYSCON2.STALL_CMD bit) if configuration number is not correct. If the request is SET_CONFIG, the MPU must reset all endpoints, halt endpoints not used by the default interface setting, set SYSCON1.SELF_PWR value if device is self-powered for the configuration set, and then set SYSCON2.DEV_CFG bit (if configuration is not 0), or set SYSCON2.CLR_CFG bit (if configuration is 0) before allowing status stage to complete. The device moves to configured state (if DEV_CFG is set), or moves to addressed state (if CLR_CFG is set) and a DS_CHG interrupt is asserted to the MPU.	Command is passed to the MPU.

Notes:
1) Transactions on endpoints other than 0 are ignored if the device is not configured (addressed state).
2) If some endpoints are not used by the interface currently set, transactions on these endpoints are not ignored; the MPU must set the halt feature for the endpoint. This does not happen if the USB host works correctly.
3) If endpoint 0 is halted, per the USB 1.1 specification (see 9.4.5: Get_Status), all requests are stalled except GET_STATUS, CLEAR_FEATURE, and SET_FEATURE requests.
4) Requests are handled with respect to the USB 1.1 specification when specified as such, but many device reactions are not included in the specification.
5) During a SET_ADDRESS autodecoded command, only the 7 LSBs are significant for the new address (decimal value from 0 to 127); all others are discarded.

Table 22−28. Autodecoded Versus Non-Autodecoded Control Requests (Continued)

Request	Recipient	Status	MPU-Required Action	Device Behavior if Device Is Not Configured
GET/SET INTERFACE	Interface	Non-autodecoded	The MPU must stall the command (through the SYSCON2.STALL_CMD bit) if interface/setting number is not correct. If the request is SET_ INTERFACE, the MPU must reset endpoints used by the interface and halt endpoints not used by the interface setting before allowing status stage to complete.	Command is passed to the MPU.
SYNCH_ FRAME	Endpoint	Non-autodecoded	The MPU must stall the command if it does not support SYNCH_FRAME request, else write requested data in the endpoint 0 FIFO.	Command is passed to the MPU.

Notes:
1) Transactions on endpoints other than 0 are ignored if the device is not configured (addressed state).
2) If some endpoints are not used by the interface currently set, transactions on these endpoints are not ignored; the MPU must set the halt feature for the endpoint. This does not happen if the USB host works correctly.
3) If endpoint 0 is halted, per the USB 1.1 specification (see 9.4.5: Get_Status), all requests are stalled except GET_STATUS, CLEAR_FEATURE, and SET_FEATURE requests.
4) Requests are handled with respect to the USB 1.1 specification when specified as such, but many device reactions are not included in the specification.
5) During a SET_ADDRESS autodecoded command, only the 7 LSBs are significant for the new address (decimal value from 0 to 127); all others are discarded.

22.5.6.6 Control Transfers Data Stage Length

The control transfer data stage length is indicated in the setup data packet.

During control reads, if the USB host requests more data than indicated in the setup packet, the unexpected IN transaction is STALLed, thus causing a STALL handshake for the remaining transactions of the transfer until the next SETUP. If the USB host requires less data than indicated in the setup packet, the transfer is not STALLed. However, if the host moves to status stage earlier than expected for a non-autodecoded request, the OUT status stage is NAKed because the MPU subsystem has not enabled the RX FIFO.

During control writes, if the USB host sends more bytes than indicated in the setup packet, the transfer is STALLed. If the USB host sends fewer bytes than were expected, the request is accepted. However, if the USB host moves to status stage earlier than expected for a non-autodecoded request, the IN status stage is NAKed because the MPU subsystem has not enabled the TX FIFO.

22.5.7 USB Device Initialization

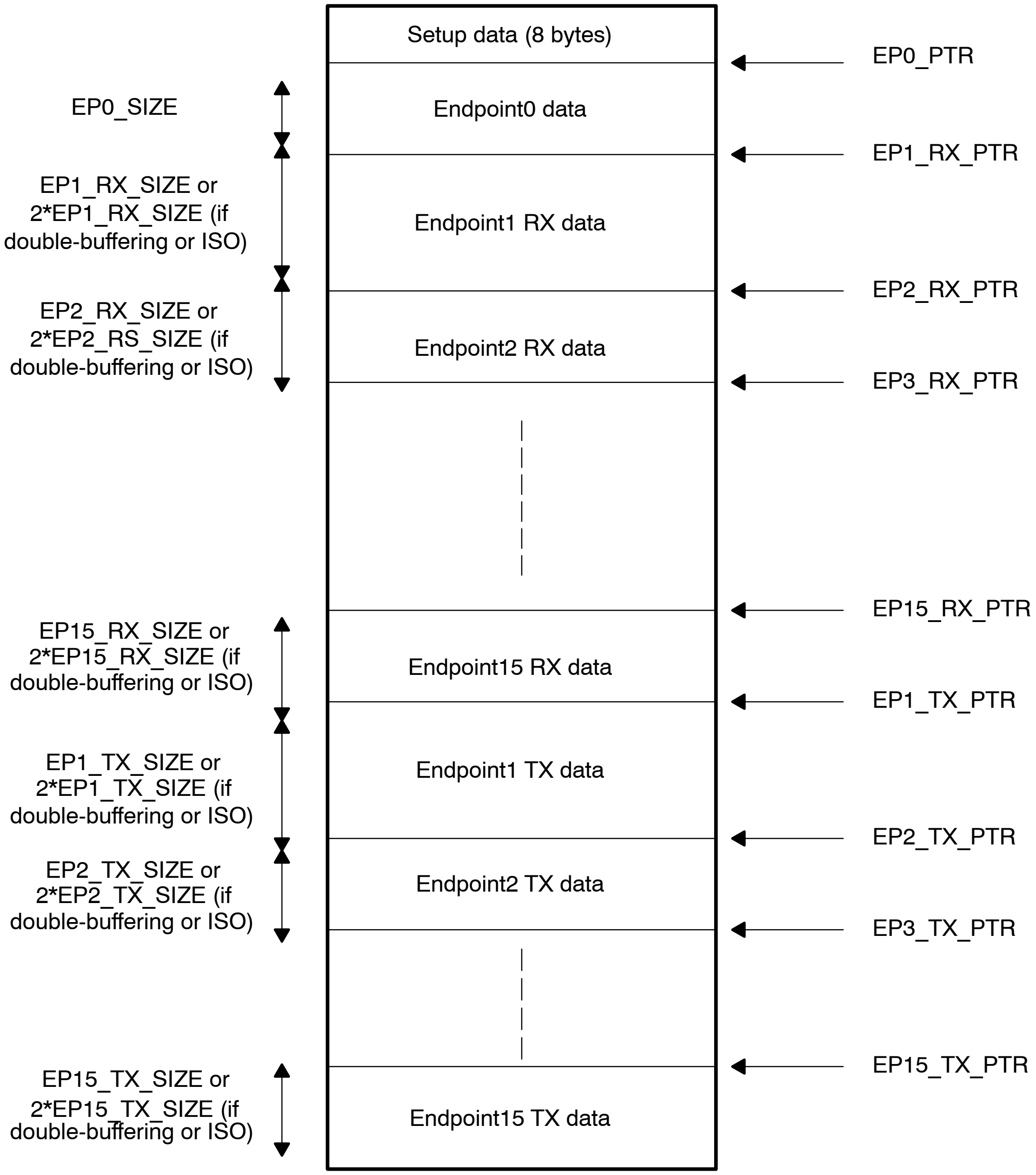
To allow communication between the device and a USB host, the MPU subsystem must configure the device by filling the configuration registers.

For each endpoint, the MPU subsystem must write the following on the dedicated register:

- Endpoint size
- Whether or not double-buffering is allowed for the endpoint
- Endpoint type (ISO or non-ISO)
- Address of the pointer

System software must choose how to allocate the 2,040 available bytes of USB device controller RAM to the USB endpoints. The receive endpoint size and type are configured using the EP1_RX through EP15_RX registers. The transmit endpoint size and type are configured using the EP1_TX through EP15_TX registers. Figure 22−34 shows an example of the RAM organization, obtained by following the flowchart shown in Figure 22−35.

Figure 22−34. Example of RAM Organization


Setup data (8 bytes)
EP0_PTR
EP0_SIZE
Endpoint0 data
EP1_RX_PTR
EP1_RX_SIZE or 2*EP1_RX_SIZE (if double-buffering or ISO)
Endpoint1 RX data
EP2_RX_PTR
EP2_RX_SIZE or 2*EP2_RS_SIZE (if double-buffering or ISO)
Endpoint2 RX data
EP3_RX_PTR
EP15_RX_PTR
EP15_RX_SIZE or 2*EP15_RX_SIZE (if double-buffering or ISO)
Endpoint15 RX data
EP1_TX_PTR
EP1_TX_SIZE or 2*EP1_TX_SIZE (if double-buffering or ISO)
Endpoint1 TX data
EP2_TX_PTR
EP2_TX_SIZE or 2*EP2_TX_SIZE (if double-buffering or ISO)
Endpoint2 TX data
EP3_TX_PTR
EP15_TX_PTR
EP15_TX_SIZE or 2*EP15_TX_SIZE (if double-buffering or ISO)
Endpoint15 TX data

Once the endpoints are configured, the MPU subsystem must set the CFG_LOCK bit SYSCON1[8]. If this bit is not set, all transactions are ignored by the core. Then, when the MPU subsystem is ready to communicate with the USB host, it must set the PULLUP_EN bit SYSCON1[0].

The MPU subsystem can wait until the DS_CHG attach interrupt is detected and handled before setting the PULLUP_EN bit SYSCON1[0]. The USB host cannot detect the device until this bit is set.

Figure 22−35 and Figure 22−36 are flowcharts for the configuration phase.

Figure 22−35. Device Configuration Routine

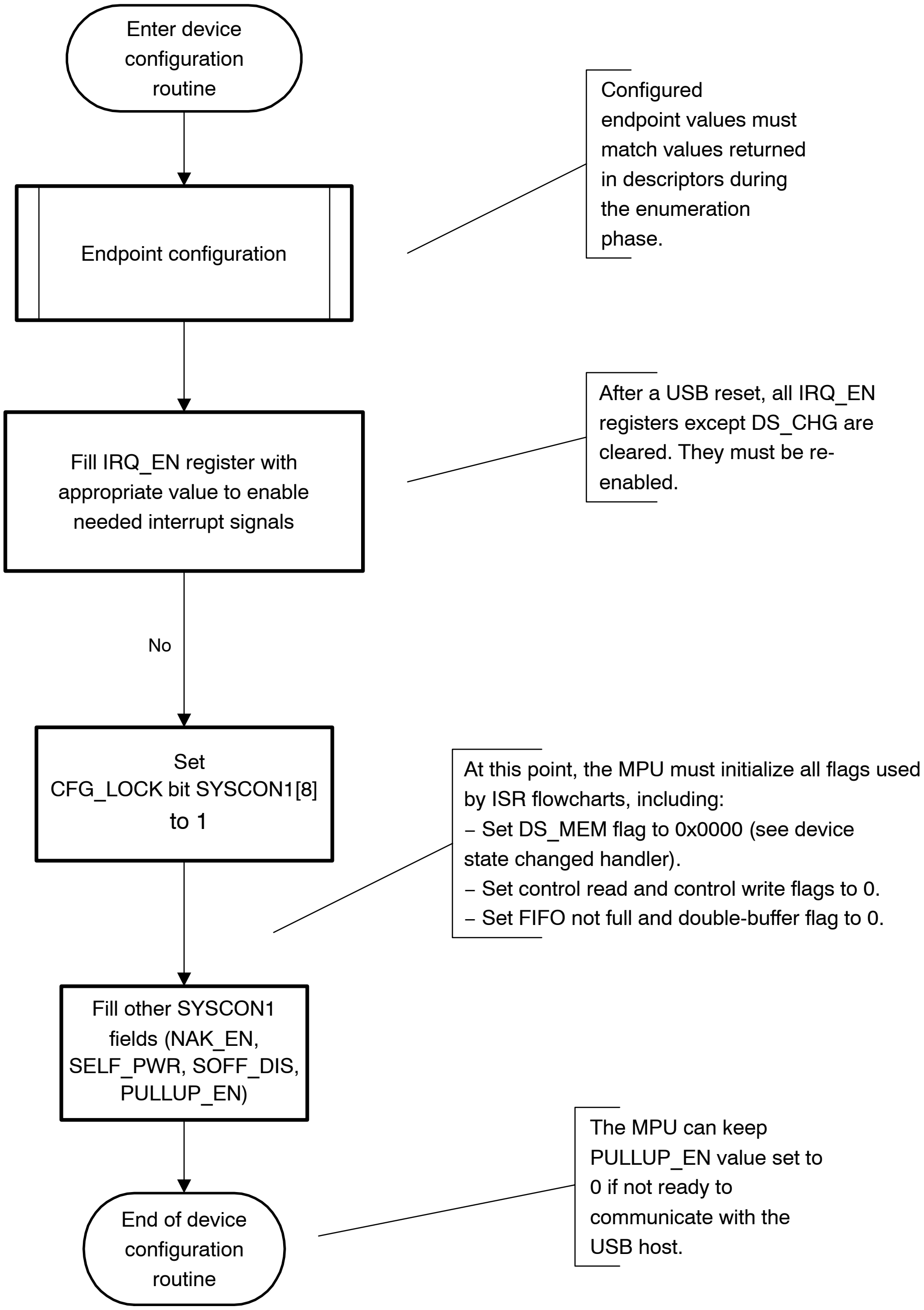
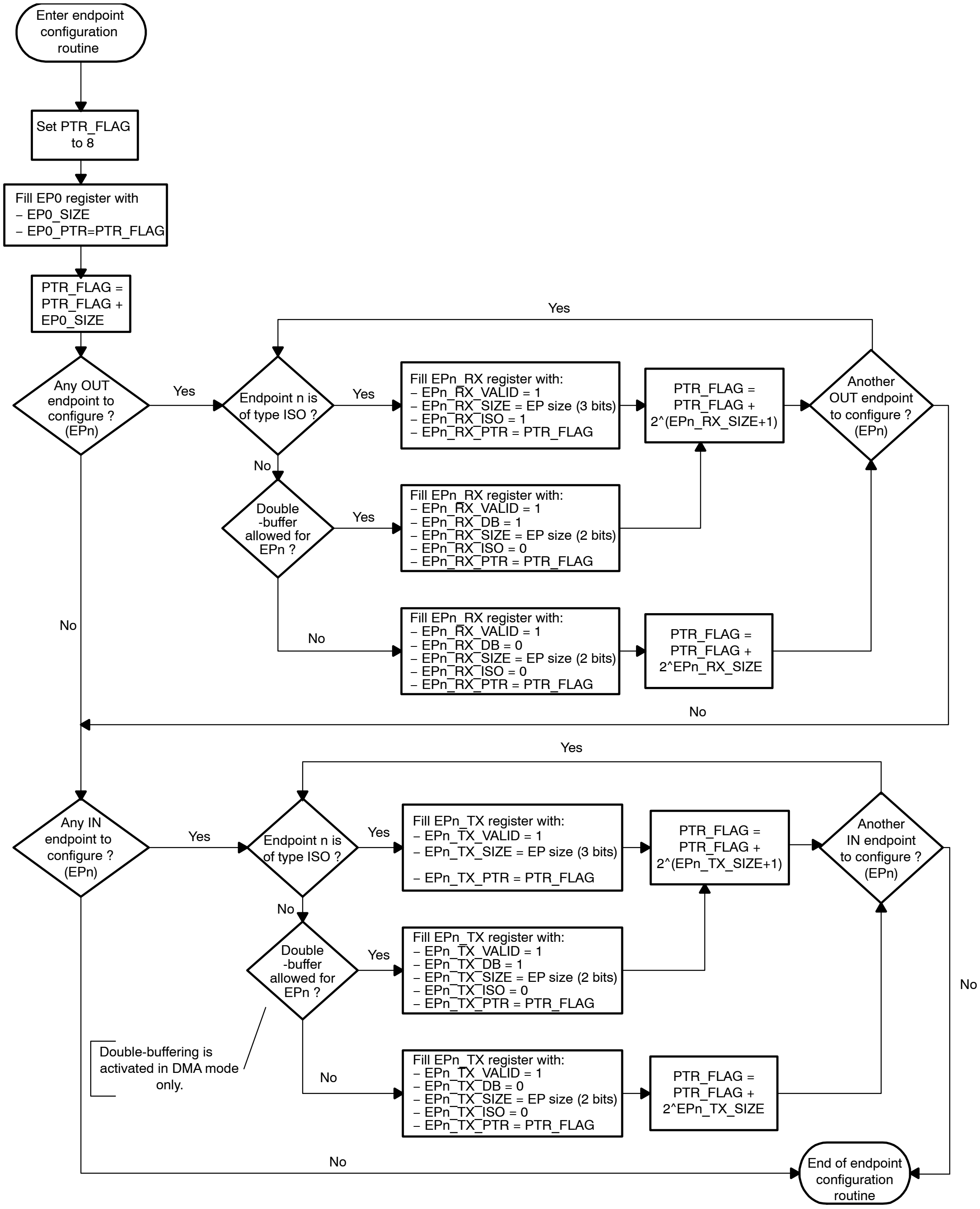

Enter device configuration routine
Endpoint configuration
Configured endpoint values must match values returned in descriptors during the enumeration phase.
Fill IRQ_EN register with appropriate value to enable needed interrupt signals
After a USB reset, all IRQ_EN registers except DS_CHG are cleared. They must be re-enabled.
No
Set CFG_LOCK bit SYSCON1[8] to 1
At this point, the MPU must initialize all flags used by ISR flowcharts, including:
− Set DS_MEM flag to 0x0000 (see device state changed handler).
− Set control read and control write flags to 0.
− Set FIFO not full and double-buffer flag to 0.
Fill other SYSCON1 fields (NAK_EN, SELF_PWR, SOFF_DIS, PULLUP_EN)
The MPU can keep PULLUP_EN value set to 0 if not ready to communicate with the USB host.
End of device configuration routine

Figure 22−36. Endpoint Configuration Routine


Enter endpoint configuration routine
Set PTR_FLAG to 8
Fill EP0 register with
− EP0_SIZE
− EP0_PTR=PTR_FLAG
PTR_FLAG = PTR_FLAG + EP0_SIZE
Any OUT endpoint to configure ? (EPn)
Yes
No
Endpoint n is of type ISO ?
Yes
No
Fill EPn_RX register with:
− EPn_RX_VALID = 1
− EPn_RX_SIZE = EP size (3 bits)
− EPn_RX_ISO = 1
− EPn_RX_PTR = PTR_FLAG
PTR_FLAG = PTR_FLAG + 2^(EPn_RX_SIZE+1)
Another OUT endpoint to configure ? (EPn)
Yes
No
Double -buffer allowed for EPn ?
Yes
No
Fill EPn_RX register with:
− EPn_RX_VALID = 1
− EPn_RX_DB = 1
− EPn_RX_SIZE = EP size (2 bits)
− EPn_RX_ISO = 0
− EPn_RX_PTR = PTR_FLAG
Fill EPn_RX register with:
− EPn_RX_VALID = 1
− EPn_RX_DB = 0
− EPn_RX_SIZE = EP size (2 bits)
− EPn_RX_ISO = 0
− EPn_RX_PTR = PTR_FLAG
PTR_FLAG = PTR_FLAG + 2^EPn_RX_SIZE
Any IN endpoint to configure ? (EPn)
Yes
No
Endpoint n is of type ISO ?
Yes
No
Fill EPn_TX register with:
− EPn_TX_VALID = 1
− EPn_TX_SIZE = EP size (3 bits)
− EPn_TX_PTR = PTR_FLAG
PTR_FLAG = PTR_FLAG + 2^(EPn_TX_SIZE+1)
Another IN endpoint to configure ? (EPn)
Yes
No
Double -buffer allowed for EPn ?
Yes
No
Fill EPn_TX register with:
− EPn_TX_VALID = 1
− EPn_TX_DB = 1
− EPn_TX_SIZE = EP size (2 bits)
− EPn_TX_ISO = 0
− EPn_TX_PTR = PTR_FLAG
Double-buffering is activated in DMA mode only.
Fill EPn_TX register with:
− EPn_TX_VALID = 1
− EPn_TX_DB = 0
− EPn_TX_SIZE = EP size (2 bits)
− EPn_TX_ISO = 0
− EPn_TX_PTR = PTR_FLAG
PTR_FLAG = PTR_FLAG + 2^EPn_TX_SIZE
End of endpoint configuration routine

Note: The local host must fill these fields during an initial endpoint configuration before setting CFG_LOCK bit SYSCON1[8] and must not change the values once CFG_LOCK bit SYSCON1[8] is set. If an endpoint is no longer used and is cleared by software, you can proceed with a new configuration of the endpoint.

22.5.8 Preparing for Transfers

To avoid NAK handshakes for the first transaction on an endpoint, the MPU subsystem must prepare the endpoint FIFO to receive or transfer data. After the first transaction, the FIFO is enabled during the interrupt handling (see Section 22.5.9, *USB Device ISR Flowcharts*).

For receive endpoints, this phase consists of enabling the FIFO to receive data from the USB host. If double-buffering is allowed for the endpoint, setting the SET_FIFO_EN bit CTRL[2] enables both FIFOs. Therefore, it is not possible to allow a single transaction when double-buffering is used.

The MPU subsystem enters the prepare-for-USB-RX-transfers routine, presented once after the enumeration phase, and then properly reacts to EP interrupts. Whether or not double-buffering is allowed is transparent to the MPU subsystem, unless both FIFOs are cleared through the CLR_EP bit CTRL[1] or RESET_EP bit CTRL[0]. In that case, and in the case where the MPU subsystem finishes to handle an interrupt without having set the SET_FIFO_EN bit CTRL[2], the MPU subsystem must reenter the prepare-for-USB-RX-transfers routine.

For transmit endpoints, the MPU subsystem enters the prepare-for-endpoint-n-TX-transfer routine, presented each time a new file must be transmitted from endpoint n to the USB host. The MPU subsystem must not enter this routine until data written into TX FIFO from the previous transfer have all been received successfully by the USB host (ACK interrupt received), unless TX FIFO is cleared through the CLR_EP bit CTRL[1] or RESET_EP bit CTRL[0] (see Figure 22−37 and Figure 22−38).

Note:

This does not apply to endpoint 0, which is not used before a setup interrupt occurs. At setup interrupt, the MPU subsystem reacts appropriately, and enables EP0 FIFO only if necessary.

To ensure proper use of the module, data cannot be prepared on different endpoints at the same time. A routine can be entered once the routine is not used by another endpoint (no parallelism); the EP_NUM register can be accessed using different routines.

Figure 22–37. Prepare-for-USB-RX-Transfers Routine

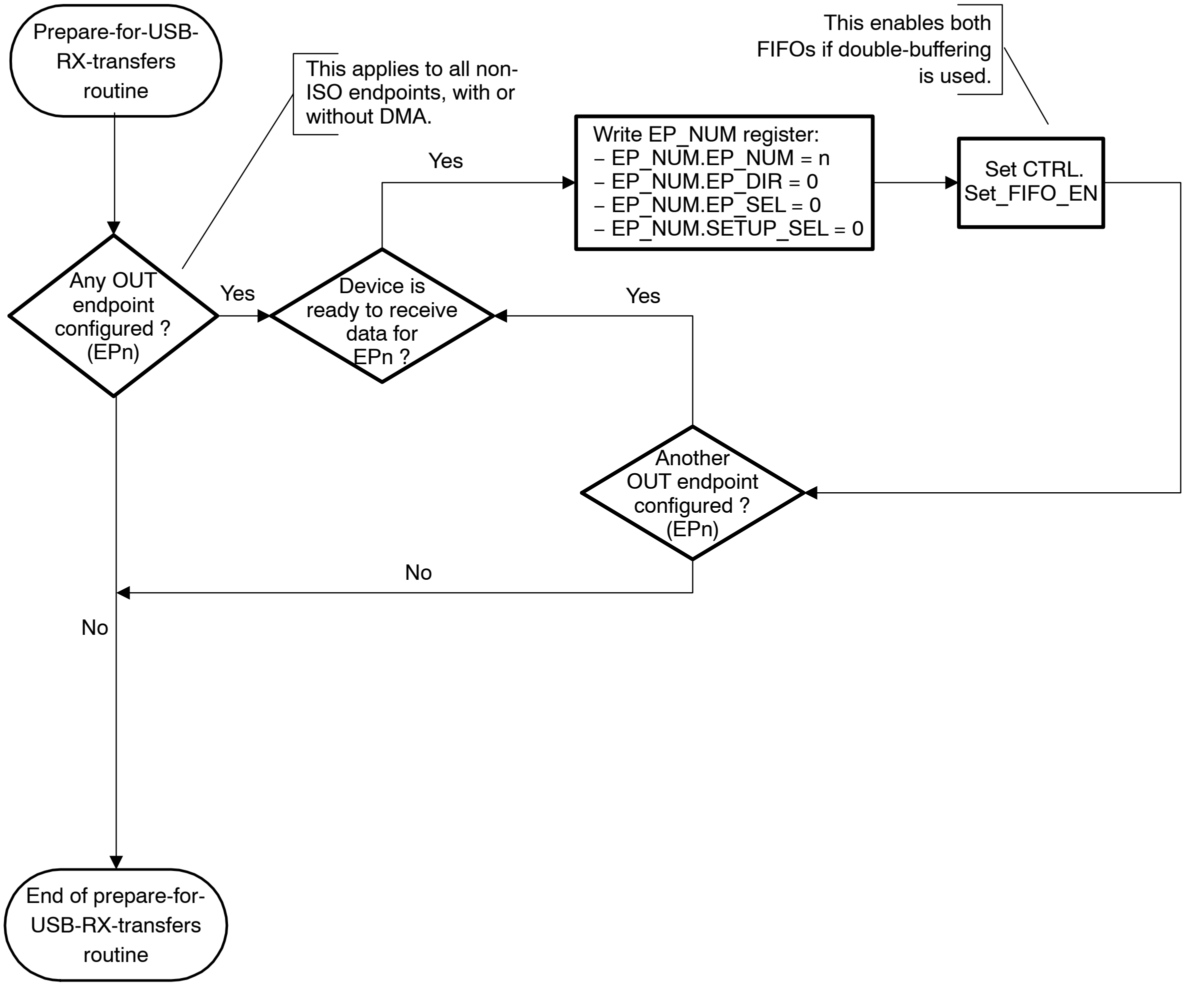
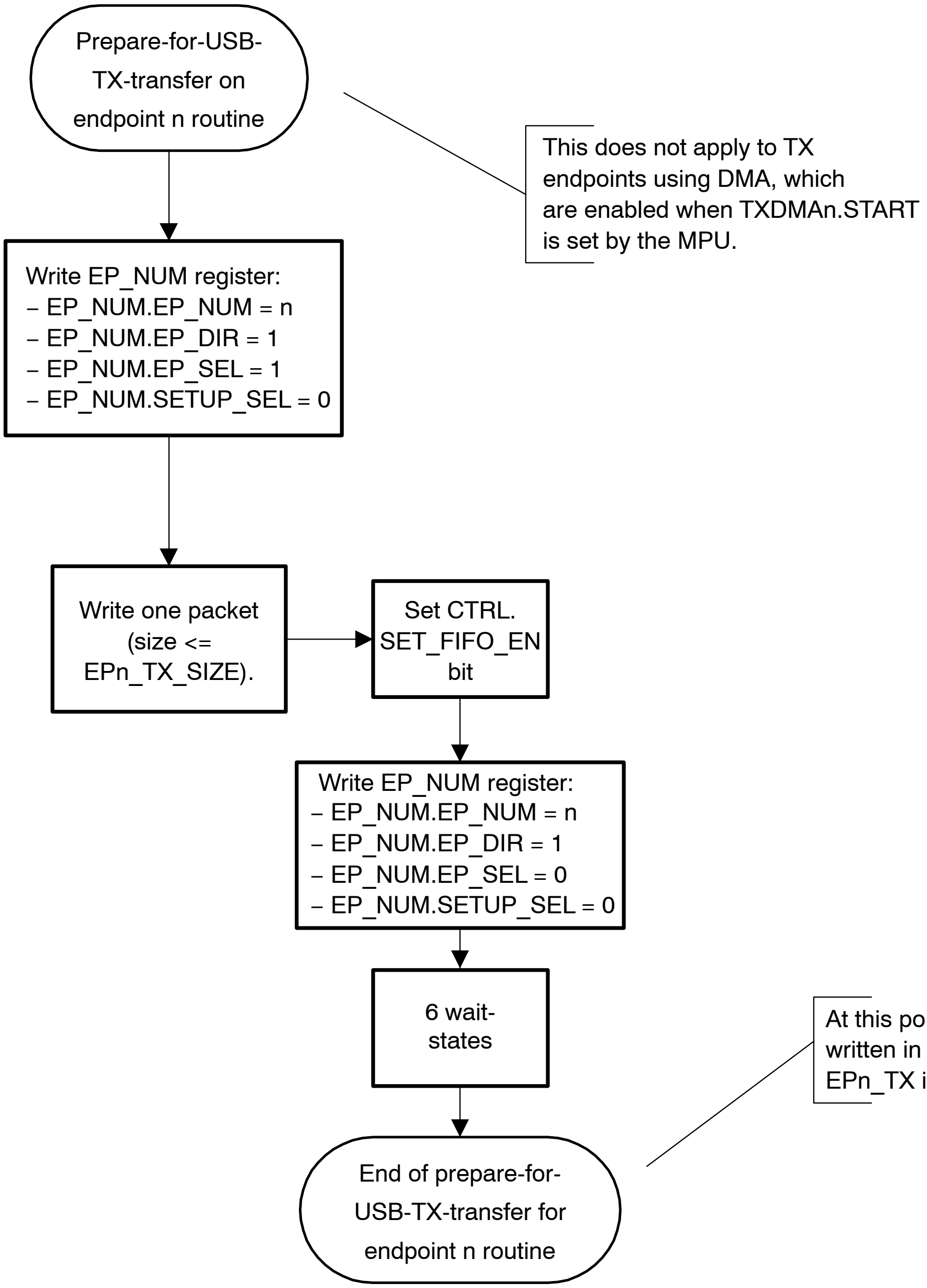

Prepare-for-USB-RX-transfers routine
This applies to all non-ISO endpoints, with or without DMA.
This enables both FIFOs if double-buffering is used.
Yes
Write EP_NUM register:
– EP_NUM.EP_NUM = n
– EP_NUM.EP_DIR = 0
– EP_NUM.EP_SEL = 0
– EP_NUM.SETUP_SEL = 0
Set CTRL.Set_FIFO_EN
Any OUT endpoint configured ? (EPn)
Yes
Device is ready to receive data for EPn ?
Yes
Another OUT endpoint configured ? (EPn)
No
No
End of prepare-for-USB-RX-transfers routine

Figure 22–38. Prepare-for-Endpoint n-TX-Transfer Routine



This does not apply to TX endpoints using DMA, which are enabled when TXDMAn.START is set by the MPU.

At this point, TX data is written in response to EPn_TX interrupts.

22.5.9 USB Device ISR Flowcharts

The flowcharts in this section give general operational guidelines for USB device ISR processing. System-architecture-specific details are contained in the MPU and USB host code. One USB-specific interrupt register (IRQ_SRC) is provided, including:

- ❑ General USB interrupts (USB_IRQ_GEN), including endpoint 0, DMA and device states interrupts, on M_IRQ_75
- ❑ Non-ISO endpoint-specific interrupt (USB_IRQ_NISO) on M_IRQ_76
- ❑ SOF interrupt for ISO transactions (USB_IRQ_ISO) on M_IRQ_77

The general USB interrupt ISR must handle non-autodecoded control transfers on endpoint 0 and some specialty interrupts generated because of USB device state modifications or DMA transfers. The ISR for the endpoint-specific interrupt must handle interrupts from the USB module that are generated because of USB activity for nonisochronous endpoints. The SOF ISR handles isochronous endpoints, and if required by the application, tracks the USB frame number. Many flowcharts show guidelines for how to handle the interrupts related to the USB device controller module. The flowcharts in this section assume that the NAK_EN bit SYSCON1[4] is cleared.

Note:

A key assumption behind the flowcharts presented in this section is that the application provides separate buffers for each direction of an endpoint, except for endpoint 0. The flowcharts read from these application buffers for IN transactions on TX endpoints and write to these application buffers for OUT transactions on RX endpoints.

The USB device controller does not support reentrant interrupts. Each USB device controller interrupt must be handled completely before handling another USB device controller interrupt. This restriction occurs because there is only one EP_NUM register, so endpoint control operations must be completed before working with another endpoint or endpoint direction.

22.5.10 USB Device Interrupts

When an endpoint interrupt is asserted, the MPU subsystem writes the EP_NUM register with the EP_SEL bit EP_NUM[5] set to 1. The MPU subsystem must finish the interrupt handling before clearing the EP_SEL bit EP_NUM[5], because clearing this bit clears the corresponding status bit in the STAT_FLG register (ACK, NAK, STALL). When an interrupt is pending on an endpoint, the MPU subsystem must not select and then deselect the endpoint without handling the interrupt, because this clears the pending transaction status flags. The MPU subsystem is not required to set the EP_SEL bit EP_NUM[5] to 1 when setting the SET_FIFO_EN bit CTRL[2], the SET_HALT bit CTRL[6], and the CLR_HALT bit CTRL[7].

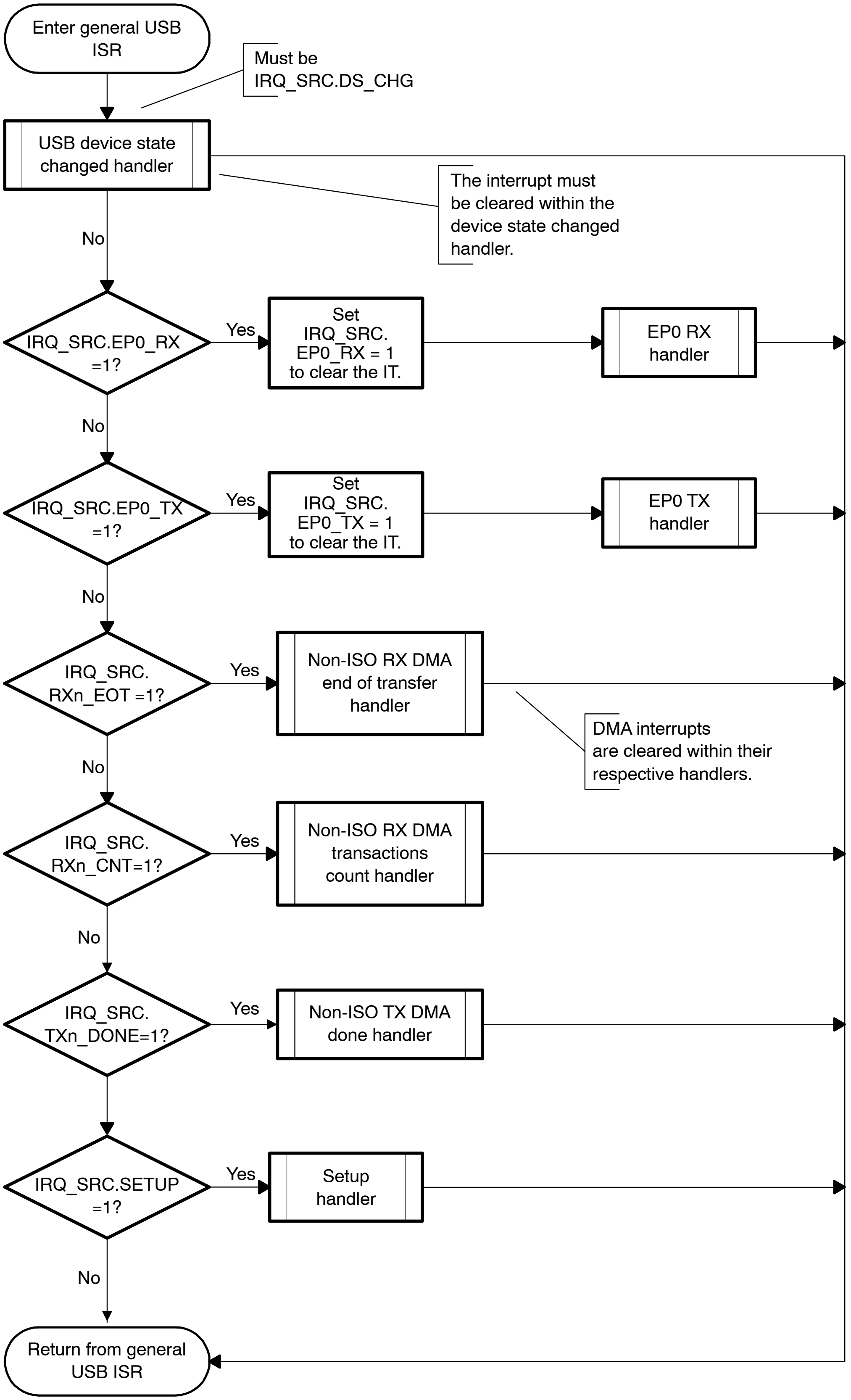
The endpoint status (STAT_FLG register) is updated at the end of each USB transaction if the previous transaction is handled. If a pending interrupt has not been handled when a new nontransparent transaction occurs, status flags are not updated (and NAK is returned, even if FIFO was enabled, or STALLed if the EP halt feature was set); therefore, the MPU subsystem never misses an ACKed transaction. If double-buffering is used for an endpoint, STAT_FLG is updated if there is zero or one interrupt pending for the endpoint; it is not updated if there are two interrupts pending on the endpoint.

The MPU subsystem is not required to set the NAK_EN bit SYSCON1[4] during normal operation. For debugging process, however, this bit can be set when the MPU subsystem finishes handling an EP interrupt without having set the corresponding SET_FIFO_EN bit CTRL[2]. During TX transaction, if the NAK_EN bit SYSCON1[4] is set, the MPU subsystem must wait for a NAK interrupt to write the TX data, to avoid a possible conflict caused by the NAK interrupt received while the MPU subsystem was writing the TX data.

22.5.11 Parsing General USB Device Interrupt

The USB_IRQ_GEN general USB interrupt (M_IRQ_75) ISR must parse the interrupt identifier register IRQ_SRC to determine the types of general USB interrupts that are active. These include interrupts relating to USB device state modifications (USB reset, suspend/resume, during enumeration phase) and control transfers on endpoint 0 or non-ISO DMA transfers in either receive or transmit mode. Multiple interrupts can be active at any time, and all must be handled by the ISR before returning from the ISR. Figure 22−39 is the flowchart for parsing the general USB interrupts.

Figure 22−39. General USB Interrupt ISR Source Parsing Flowchart


Enter general USB ISR
Must be IRQ_SRC.DS_CHG
USB device state changed handler
The interrupt must be cleared within the device state changed handler.
No
IRQ_SRC.EP0_RX =1?
Yes
Set IRQ_SRC. EP0_RX = 1 to clear the IT.
EP0 RX handler
No
IRQ_SRC.EP0_TX =1?
Yes
Set IRQ_SRC. EP0_TX = 1 to clear the IT.
EP0 TX handler
No
IRQ_SRC. RXn_EOT =1?
Yes
Non-ISO RX DMA end of transfer handler
DMA interrupts are cleared within their respective handlers.
No
IRQ_SRC. RXn_CNT=1?
Yes
Non-ISO RX DMA transactions count handler
No
IRQ_SRC. TXn_DONE=1?
Yes
Non-ISO TX DMA done handler
IRQ_SRC.SETUP =1?
Yes
Setup handler
No
Return from general USB ISR

22.5.12 Setup Interrupt Handler

A separate interrupt flag exists for setup transactions so that the MPU subsystem cannot miss a setup transaction, even if it occurs during the data or status phase of another transfer (case of an aborted transfer). The setup parsing function captures the control transfer request information, which is used to determine which USB bus activity is required and to control how the MPU subsystem must generate or respond to the control transfer. This information includes the following:

- bmRequestType
- bmRequest
- wValue
- wIndex
- wLength

The setup interrupt handler shown in Figure 22−40 processes setup transactions that occur on endpoint 0. It calls the routine that parses the control transfer request information (see Figure 22−41) to set flags that the rest of the ISR code can use to control proper response to control transfers. The following two flags, which are used by the endpoint 0 interrupt handlers, are set by the setup interrupt handler:

- Control read flag
- Control write flag

Figure 22−40. Setup Interrupt Handler

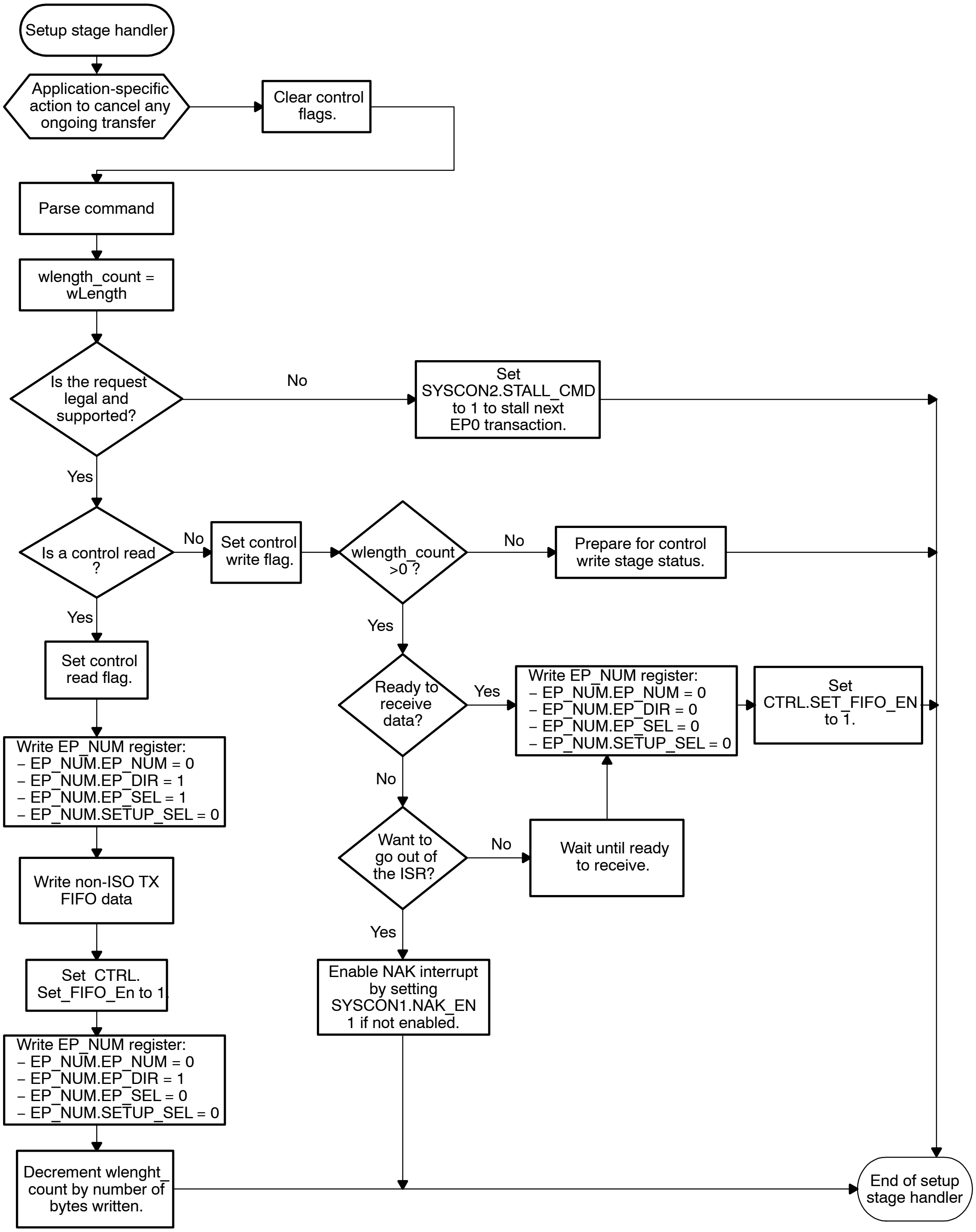
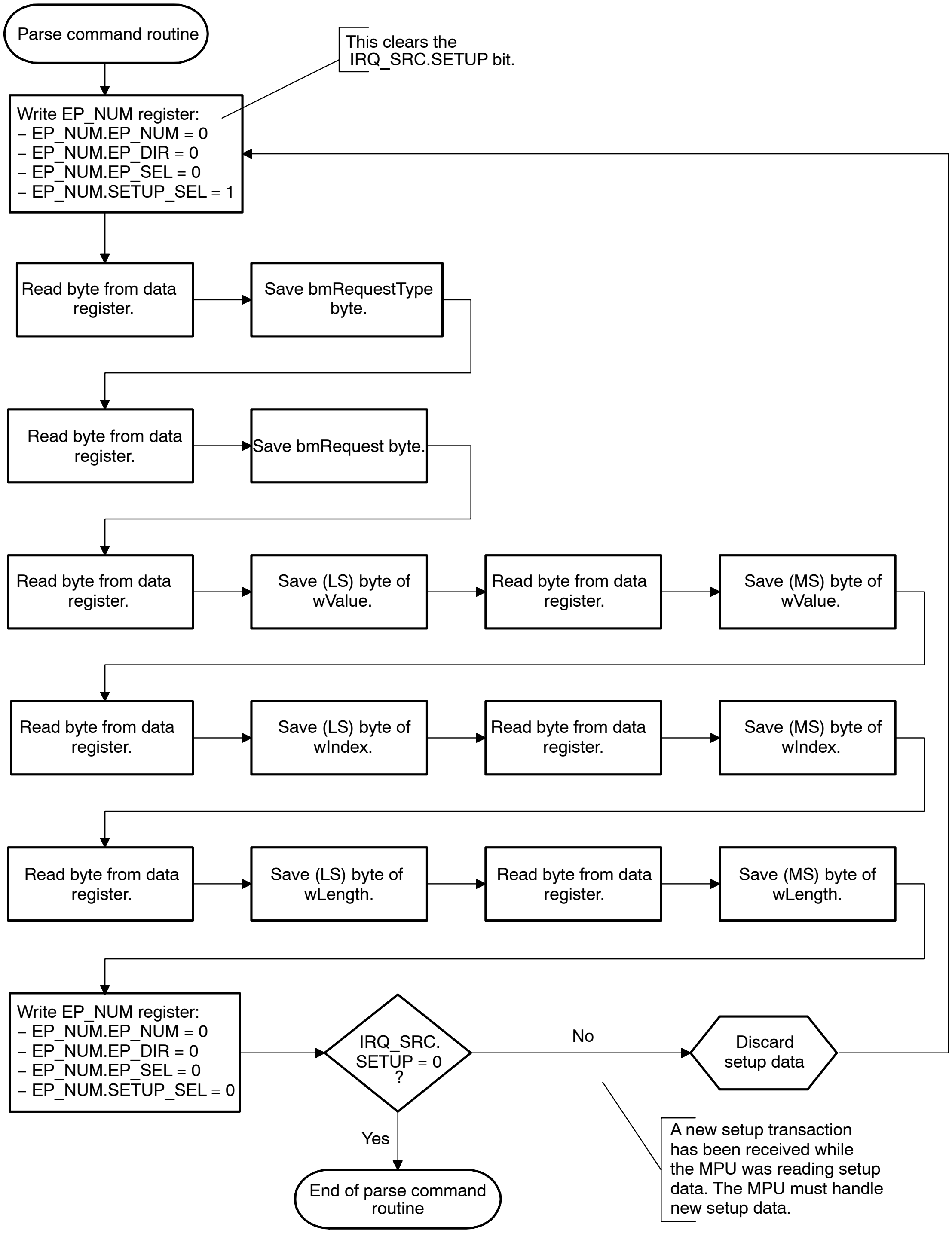


Figure 22−41. Parse Command Routine (Setup Stage Control Transfer Request)


Parse command routine
This clears the IRQ_SRC.SETUP bit.
Write EP_NUM register:
– EP_NUM.EP_NUM = 0
– EP_NUM.EP_DIR = 0
– EP_NUM.EP_SEL = 0
– EP_NUM.SETUP_SEL = 1
Read byte from data register.
Save bmRequestType byte.
Read byte from data register.
Save bmRequest byte.
Read byte from data register.
Save (LS) byte of wValue.
Read byte from data register.
Save (MS) byte of wValue.
Read byte from data register.
Save (LS) byte of wIndex.
Read byte from data register.
Save (MS) byte of wIndex.
Read byte from data register.
Save (LS) byte of wLength.
Read byte from data register.
Save (MS) byte of wLength.
Write EP_NUM register:
– EP_NUM.EP_NUM = 0
– EP_NUM.EP_DIR = 0
– EP_NUM.EP_SEL = 0
– EP_NUM.SETUP_SEL = 0
IRQ_SRC. SETUP = 0 ?
No
Discard setup data
Yes
End of parse command routine
A new setup transaction has been received while the MPU was reading setup data. The MPU must handle new setup data.

22.5.13 Endpoint 0 RX Interrupt Handler

Figure 22−42 shows the endpoint 0 RX portion of the general USB interrupt handler, which handles general USB interrupts related to control OUT transactions on endpoint 0. No EP0 interrupt is generated for autodecoded control transfers.

Figure 22−42. Endpoint 0 RX Interrupt Handler

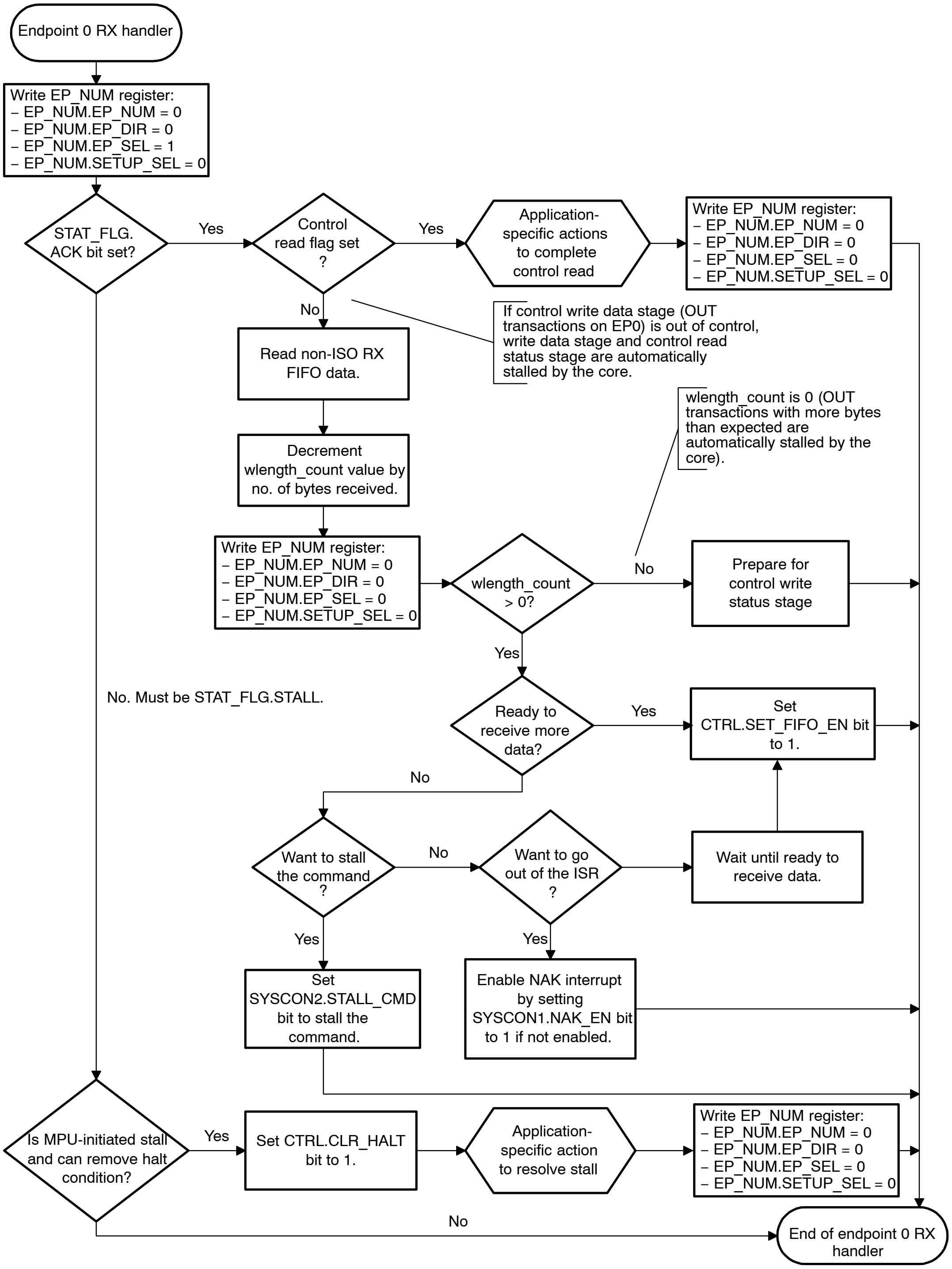
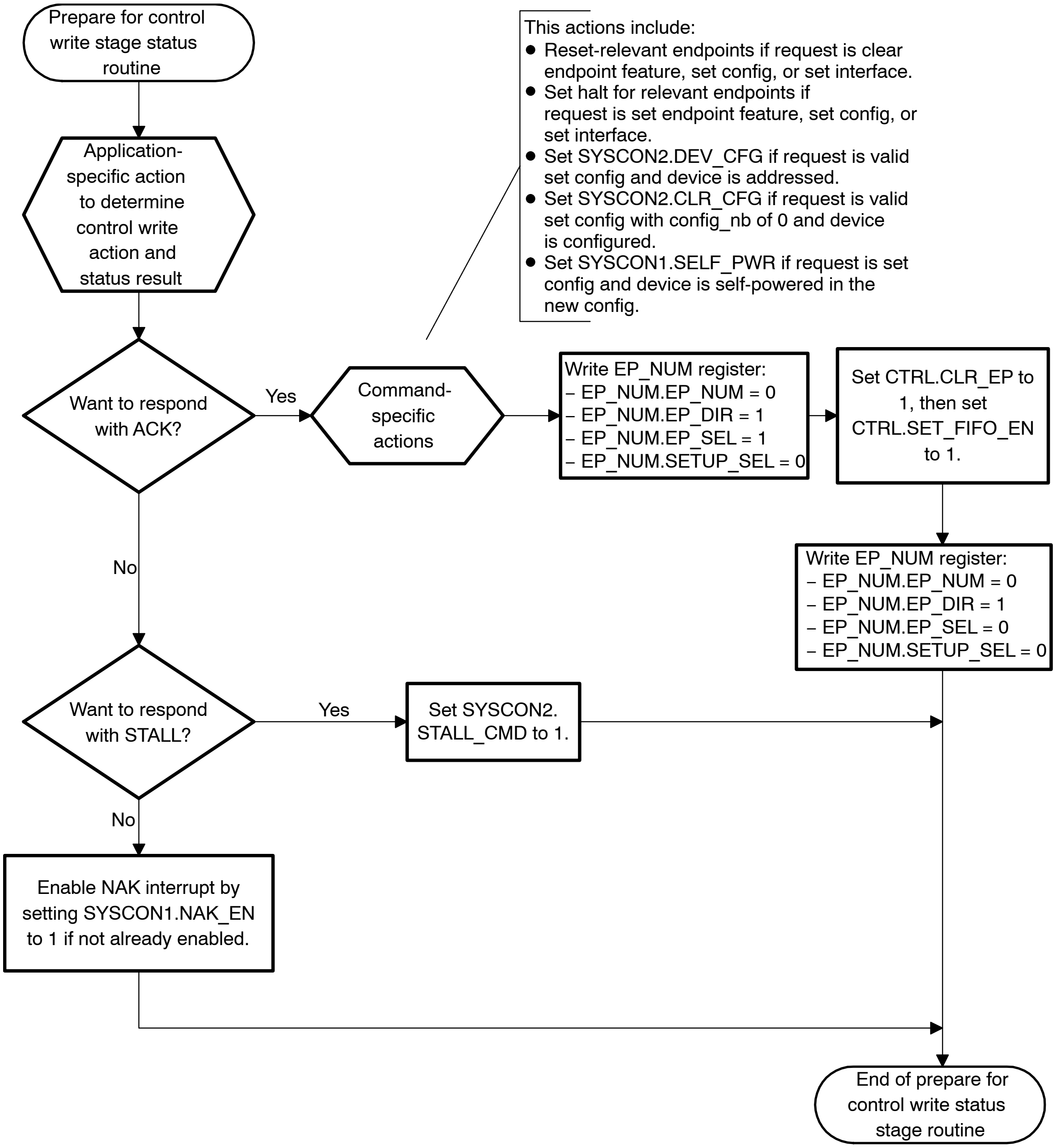

Endpoint 0 RX handler
Write EP_NUM register:
– EP_NUM.EP_NUM = 0
– EP_NUM.EP_DIR = 0
– EP_NUM.EP_SEL = 1
– EP_NUM.SETUP_SEL = 0
STAT_FLG. ACK bit set?
Yes
Control read flag set ?
Yes
Application-specific actions to complete control read
Write EP_NUM register:
– EP_NUM.EP_NUM = 0
– EP_NUM.EP_DIR = 0
– EP_NUM.EP_SEL = 0
– EP_NUM.SETUP_SEL = 0
No
Read non-ISO RX FIFO data.
If control write data stage (OUT transactions on EP0) is out of control, write data stage and control read status stage are automatically stalled by the core.
wlength_count is 0 (OUT transactions with more bytes than expected are automatically stalled by the core).
Decrement wlength_count value by no. of bytes received.
Write EP_NUM register:
– EP_NUM.EP_NUM = 0
– EP_NUM.EP_DIR = 0
– EP_NUM.EP_SEL = 0
– EP_NUM.SETUP_SEL = 0
wlength_count > 0?
No
Prepare for control write status stage
Yes
No. Must be STAT_FLG.STALL.
Ready to receive more data?
Yes
Set CTRL.SET_FIFO_EN bit to 1.
No
Want to stall the command ?
No
Want to go out of the ISR ?
Wait until ready to receive data.
Yes
Yes
Set SYSCON2.STALL_CMD bit to stall the command.
Enable NAK interrupt by setting SYSCON1.NAK_EN bit to 1 if not enabled.
Is MPU-initiated stall and can remove halt condition?
Yes
Set CTRL.CLR_HALT bit to 1.
Application-specific action to resolve stall
Write EP_NUM register:
– EP_NUM.EP_NUM = 0
– EP_NUM.EP_DIR = 0
– EP_NUM.EP_SEL = 0
– EP_NUM.SETUP_SEL = 0
No
End of endpoint 0 RX handler

Figure 22−43. Prepare for Control Write Status Stage Routine


Prepare for control write stage status routine
Application-specific action to determine control write action and status result
This actions include:
● Reset-relevant endpoints if request is clear endpoint feature, set config, or set interface.
● Set halt for relevant endpoints if request is set endpoint feature, set config, or set interface.
● Set SYSCON2.DEV_CFG if request is valid set config and device is addressed.
● Set SYSCON2.CLR_CFG if request is valid set config with config_nb of 0 and device is configured.
● Set SYSCON1.SELF_PWR if request is set config and device is self-powered in the new config.
Want to respond with ACK?
Yes
Command-specific actions
Write EP_NUM register:
− EP_NUM.EP_NUM = 0
− EP_NUM.EP_DIR = 1
− EP_NUM.EP_SEL = 1
− EP_NUM.SETUP_SEL = 0
Set CTRL.CLR_EP to 1, then set CTRL.SET_FIFO_EN to 1.
Write EP_NUM register:
− EP_NUM.EP_NUM = 0
− EP_NUM.EP_DIR = 1
− EP_NUM.EP_SEL = 0
− EP_NUM.SETUP_SEL = 0
No
Want to respond with STALL?
Yes
Set SYSCON2. STALL_CMD to 1.
No
Enable NAK interrupt by setting SYSCON1.NAK_EN to 1 if not already enabled.
End of prepare for control write status stage routine

To support the SET_DESCRIPTOR command, when in communication-specific actions, perform the following actions:

1) Reset all endpoints by selecting the endpoint, and then clear it.
2) Unlock the configuration.
3) Charge the new configuration.
4) Lock the configuration.
5) Go into prepare-for-transfers.
6) Enable all interrupts. (See Figure 22−43.)

22.5.14 Endpoint 0 TX Interrupt Handler

Figure 22−44 shows the TX portion of the general USB interrupt handler, which handles general USB interrupts related to control IN transactions on endpoint 0.

The endpoint 0 TX interrupt handler must be able to move data into the endpoint 0 TX FIFO when the application buffer for endpoint 0 TX data is not empty and an endpoint 0 TX interrupt signals an ACKed non-autodecoded endpoint 0 IN transaction. This data can be control read data stage information or control write status stage handshaking information (see Figure 22−45).

Figure 22−44. Endpoint 0 TX Interrupt Handler

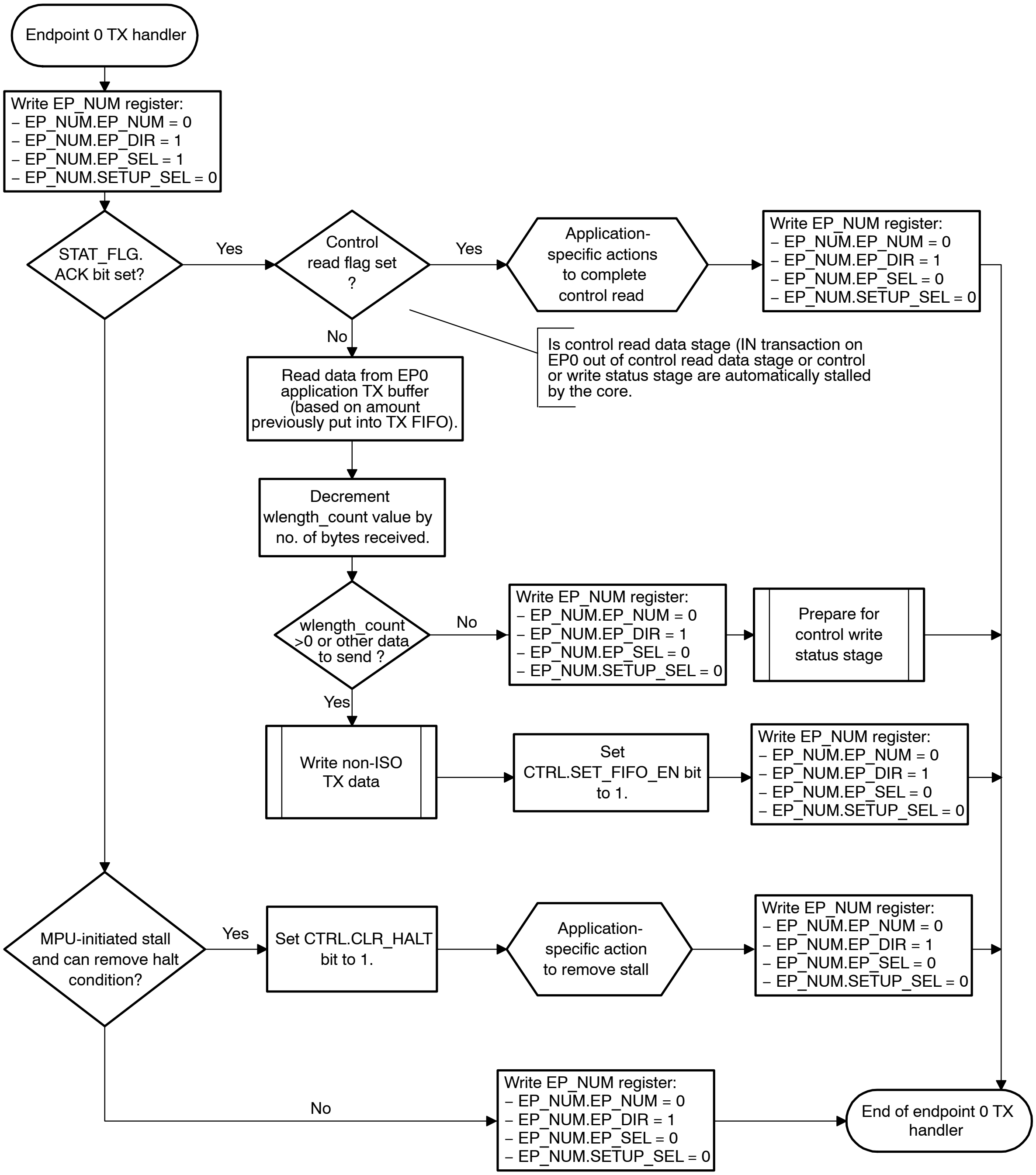
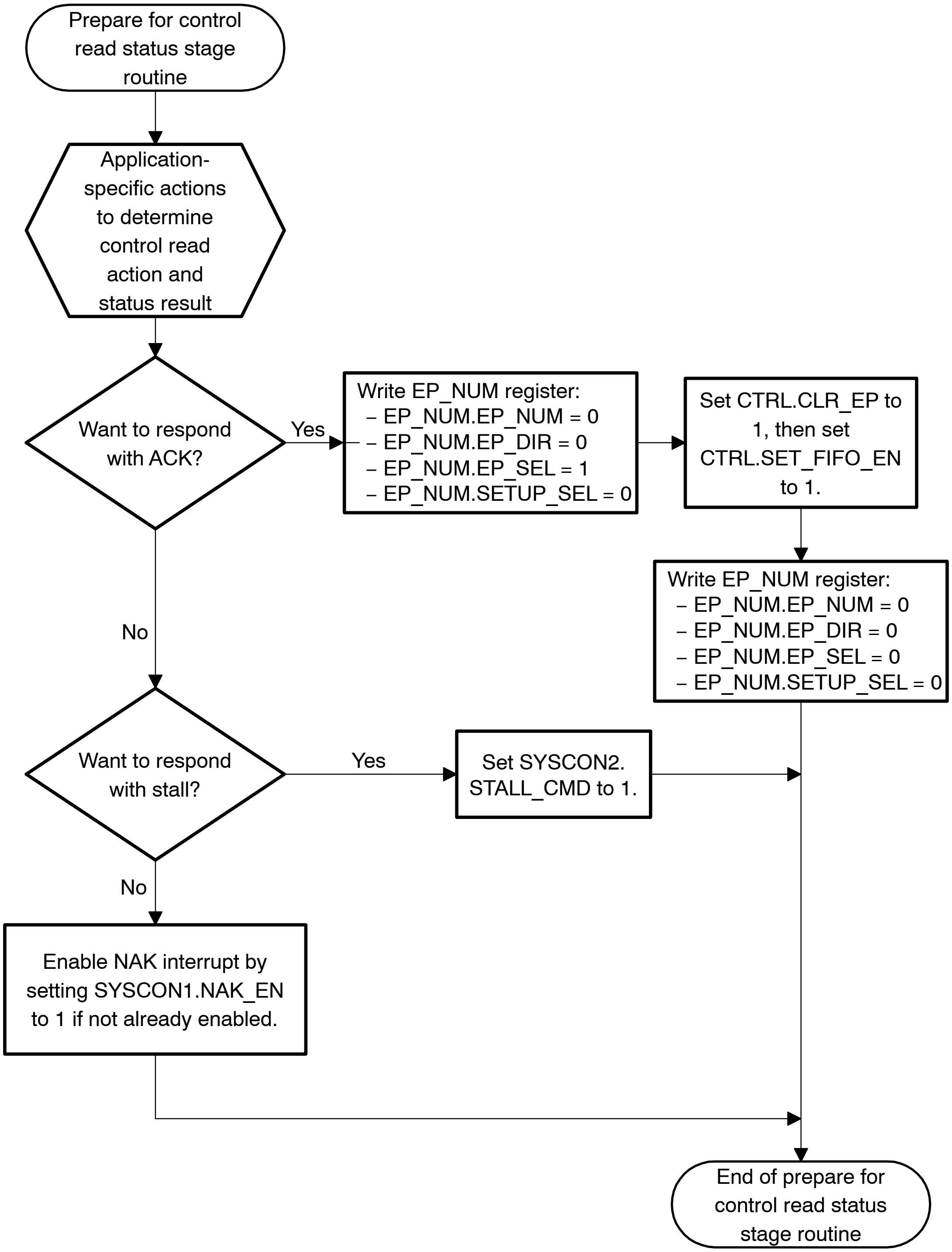


Figure 22−45. Prepare for Control Read Status Stage Routine


Prepare for control read status stage routine
Application-specific actions to determine control read action and status result
Want to respond with ACK?
Yes
Write EP_NUM register:
– EP_NUM.EP_NUM = 0
– EP_NUM.EP_DIR = 0
– EP_NUM.EP_SEL = 1
– EP_NUM.SETUP_SEL = 0
Set CTRL.CLR_EP to 1, then set CTRL.SET_FIFO_EN to 1.
Write EP_NUM register:
– EP_NUM.EP_NUM = 0
– EP_NUM.EP_DIR = 0
– EP_NUM.EP_SEL = 0
– EP_NUM.SETUP_SEL = 0
No
Want to respond with stall?
Yes
Set SYSCON2. STALL_CMD to 1.
No
Enable NAK interrupt by setting SYSCON1.NAK_EN to 1 if not already enabled.
End of prepare for control read status stage routine

22.5.15 Device States Change Handler

This section describes how USB device states and transitions states are decoded by the USB device controller and how they can be handled.

The state-machine (see Figure 22−46) shows how the USB device controller device moves from one state to another with respect to the USB1.1 specification. The attach/unattach transition is not shown in the transition flow.

Because SET_CONFIGURATION is not decoded by the core, the MPU subsystem must distinguish a SET_CONFIGURATION with a nonvalid configuration value from other SET_CONFIGURATION requests, and set the DEV_CFG bit SYSCON2[3] only if the configuration value is valid (value 0 is nonvalid) when the device is in addressed state. When the device is in configured state, the MPU subsystem must set the CLR_CFG bit SYSCON2[2] if the configuration number is 0 so that the device moves to addressed state.

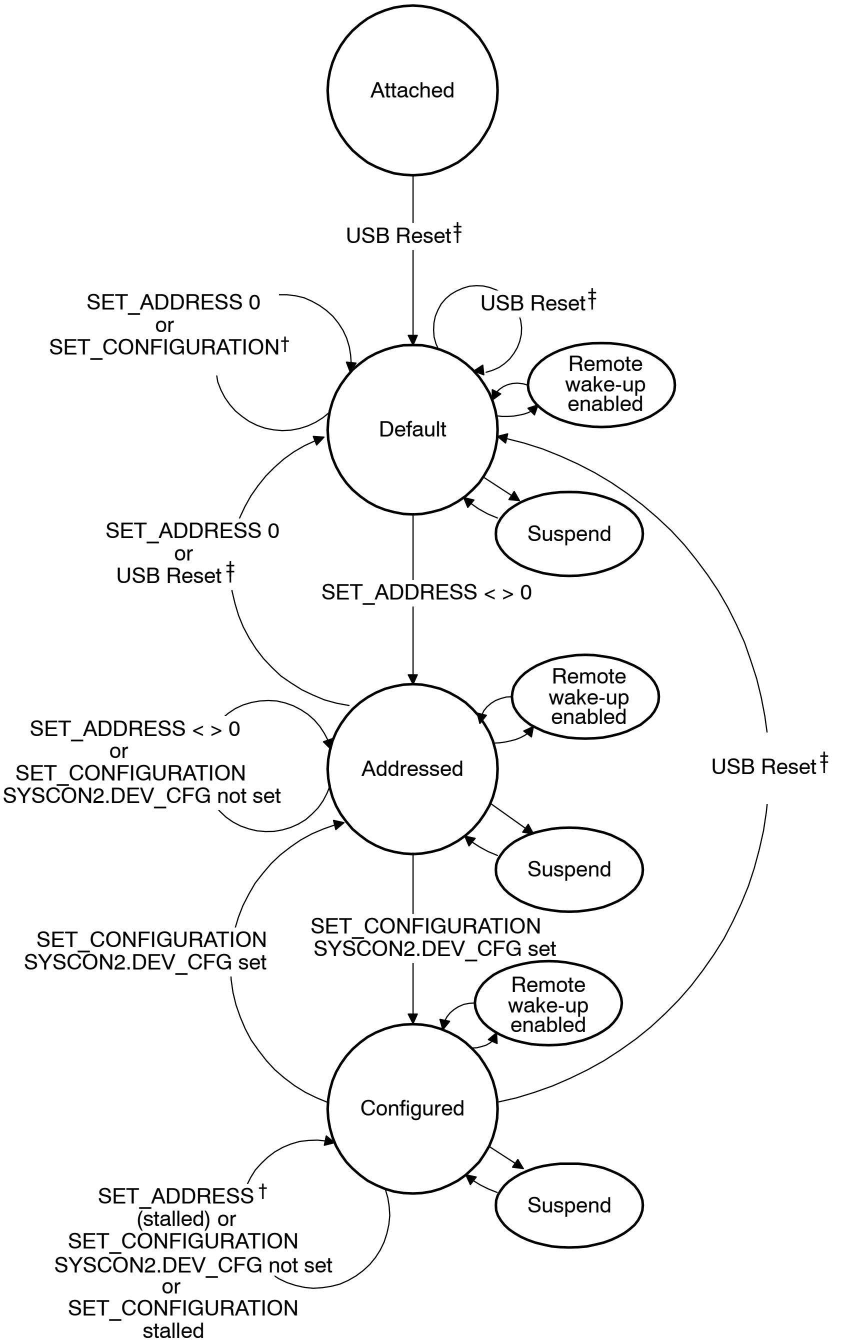
Device states are visible in the DEVSTAT register and are decoded as follows:

- Attached (ATT[0]): The device is attached to the USB and powered.
- Default (DEF[1]): The device is attached to the USB and is powered and reset.
- Addressed (ADD[2]): The device is attached to the USB and is powered and reset with an address assigned. The device moves into the addressed state after a SET_ADDRESS request with an address number other than 0.
- Configured (CFG[3]): The device is attached to the USB, is powered and reset with an address other than 0, and is configured. The device enters into the configured state after a valid SET_CONFIGURATION request, only if the MPU subsystem has set the DEV_CFG bit SYSCON2[3] (meaning the configuration is valid).
- Suspended (SUS[4]): The device is at minimum default and has not had bus activity for 5 ms.
- Reset (USB_RESET[5]): When set, the device is receiving a valid USB host reset.
- Remote wakeup granted (R_WK_OK[5]): This bit is set (cleared) automatically after a valid SET_DEVICE_FEATURE (CLEAR_DEVICE_FEATURE) request from the USB host.

Any change in the DEVSTAT register bits triggers a device change interrupt (DS_CHG bit IRQ_SRC[3]), if enabled.

The device moves to addressed state after the status stage of a valid SET_ADDRESS, even if the status stage ACK handshake is received corrupted or not sent by the USB host. A SET_DEVICE_FEATURE or a CLEAR_DEVICE_FEATURE is effective after setup transaction, even if no status stage occurs. A SET_CONFIGURATION request is effective before status stage, when the MPU subsystem sets either the CLR_CFG bit SYSCON2[2] or the DEV_CFG bit SYSCON2[3] (see Figure 22−47).

Figure 22−46. USB Device Controller Device State Transitions

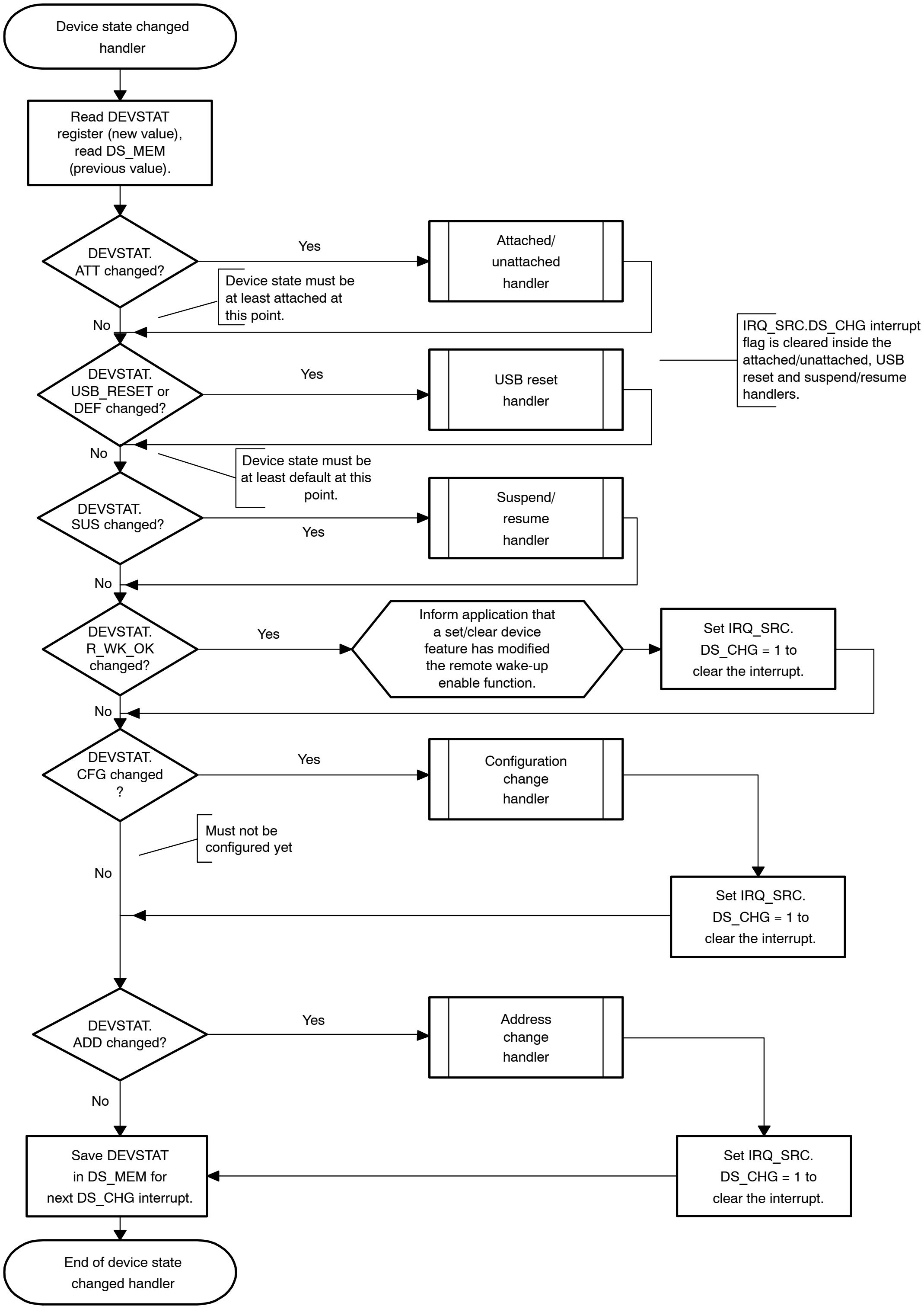

Attached
USB Reset‡
SET_ADDRESS 0 or SET_CONFIGURATION†
USB Reset‡
Remote wake-up enabled
Default
Suspend
SET_ADDRESS 0 or USB Reset‡
SET_ADDRESS < > 0
Remote wake-up enabled
SET_ADDRESS < > 0 or SET_CONFIGURATION SYSCON2.DEV_CFG not set
Addressed
USB Reset‡
Suspend
SET_CONFIGURATION SYSCON2.DEV_CFG set
SET_CONFIGURATION SYSCON2.DEV_CFG set
Remote wake-up enabled
Configured
Suspend
SET_ADDRESS† (stalled) or SET_CONFIGURATION SYSCON2.DEV_CFG not set or SET_CONFIGURATION stalled

† Behavior not specified by USB 1.1 specification (see Chapter 9)

‡ USB reset generates two interrupts (when USB reset is asserted and then when USB reset completes).

No interrupt is asserted by the core for tansitions shown with dashed lines.

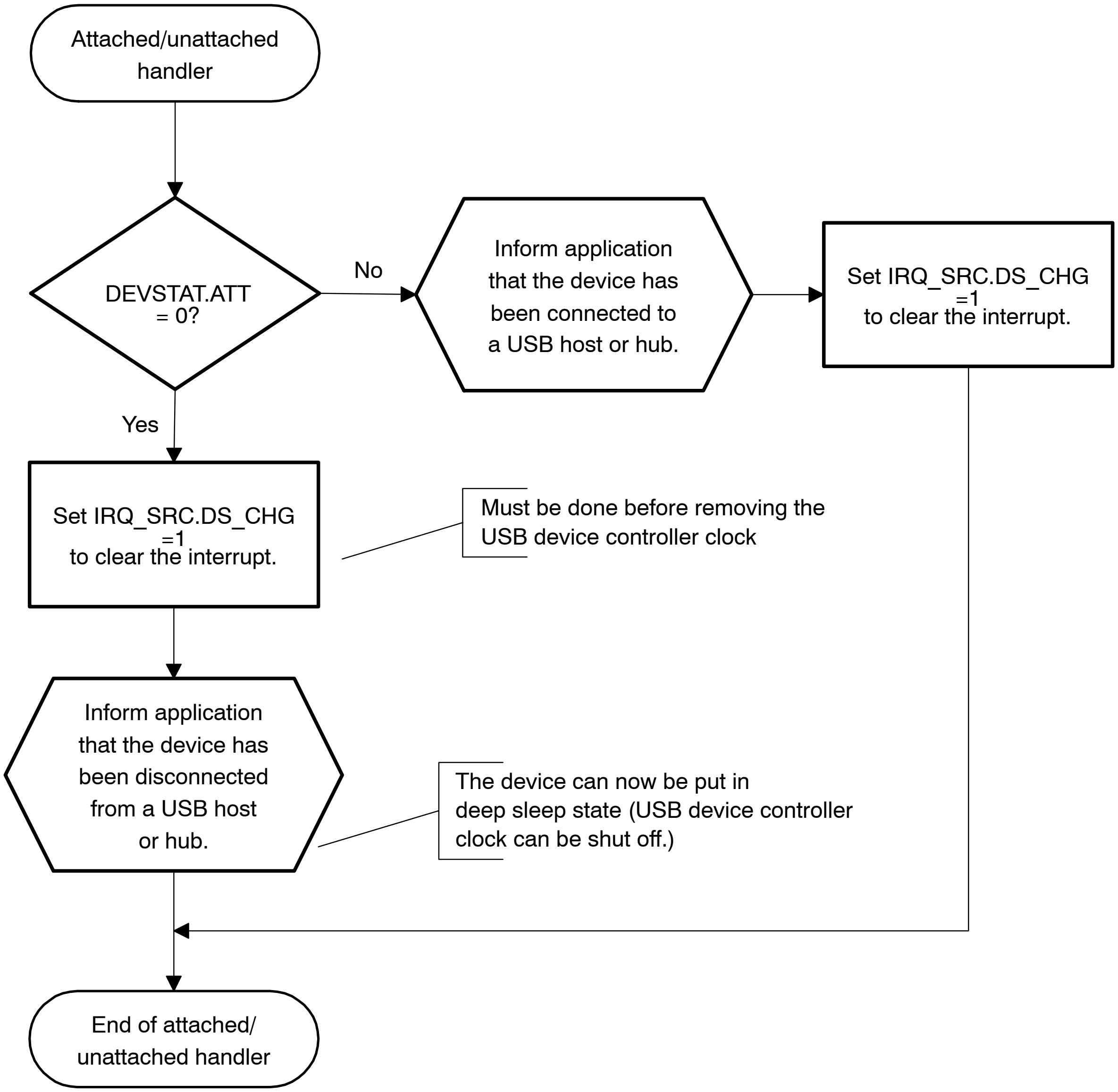
Figure 22−47. Typical Operation for USB Device State Changed Interrupt Handler


Device state changed handler
Read DEVSTAT register (new value), read DS_MEM (previous value).
DEVSTAT. ATT changed?
Yes
Attached/ unattached handler
Device state must be at least attached at this point.
No
IRQ_SRC.DS_CHG interrupt flag is cleared inside the attached/unattached, USB reset and suspend/resume handlers.
DEVSTAT. USB_RESET or DEF changed?
Yes
USB reset handler
No
Device state must be at least default at this point.
DEVSTAT. SUS changed?
Yes
Suspend/ resume handler
No
DEVSTAT. R_WK_OK changed?
Yes
Inform application that a set/clear device feature has modified the remote wake-up enable function.
Set IRQ_SRC. DS_CHG = 1 to clear the interrupt.
No
DEVSTAT. CFG changed ?
Yes
Configuration change handler
Must not be configured yet
No
Set IRQ_SRC. DS_CHG = 1 to clear the interrupt.
DEVSTAT. ADD changed?
Yes
Address change handler
No
Save DEVSTAT in DS_MEM for next DS_CHG interrupt.
Set IRQ_SRC. DS_CHG = 1 to clear the interrupt.
End of device state changed handler

22.5.15.1 Device States Attached/Unattached Handler

Device attached/unattached interrupts occur when the device detects that its VBUS has changed. System software can disable the USB device controller clock when the DS_CHG bit IRQ_SRC[3] is cleared after servicing an unattached interrupt. Disabling the USB device controller clock before the DS_CHG bit IRQ_SRC[3] is cleared can result in improper functionality for future USB device controller interrupts (see Figure 22−48).

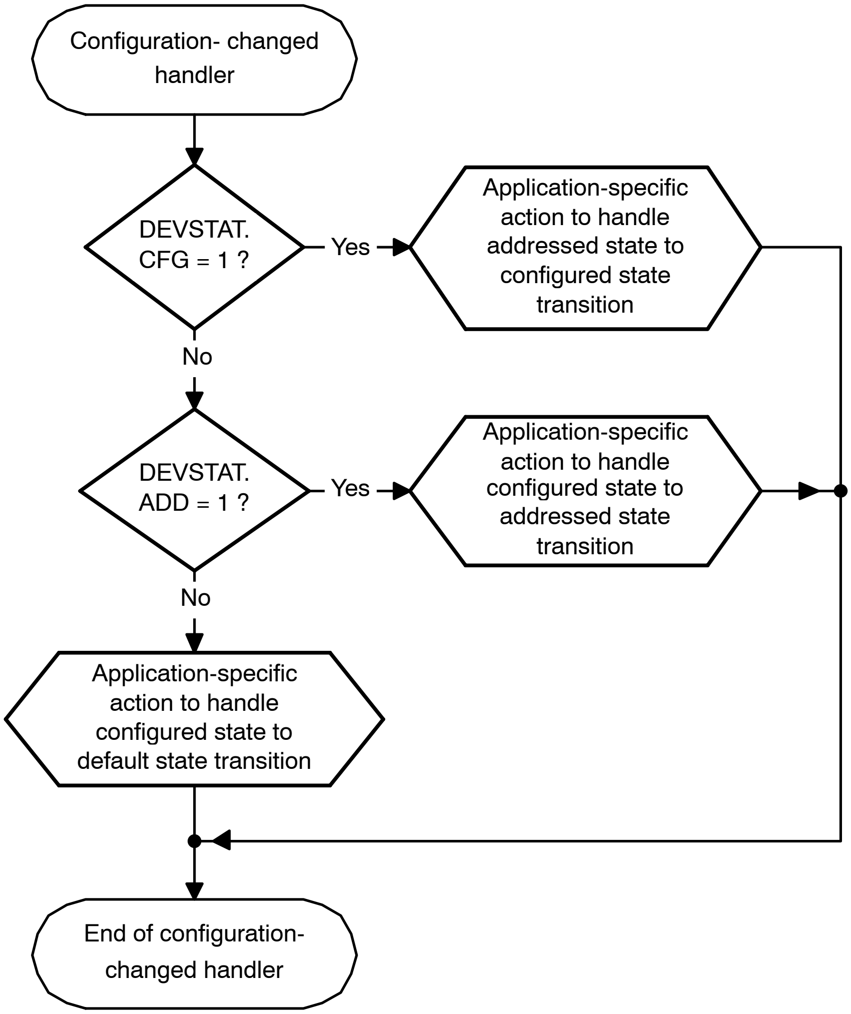
Figure 22−48. Attached/Unattached Handler


Attached/unattached handler
DEVSTAT.ATT = 0?
No
Inform application that the device has been connected to a USB host or hub.
Set IRQ_SRC.DS_CHG =1 to clear the interrupt.
Yes
Set IRQ_SRC.DS_CHG =1 to clear the interrupt.
Must be done before removing the USB device controller clock
Inform application that the device has been disconnected from a USB host or hub.
The device can now be put in deep sleep state (USB device controller clock can be shut off.)
End of attached/ unattached handler

22.5.15.2 Device State Configuration-Changed Handler

When a configuration-changed interrupt occurs, the USB device has received a set configuration operation. When this occurs, the configuration-changed handler performs the operations shown in Figure 22−49.

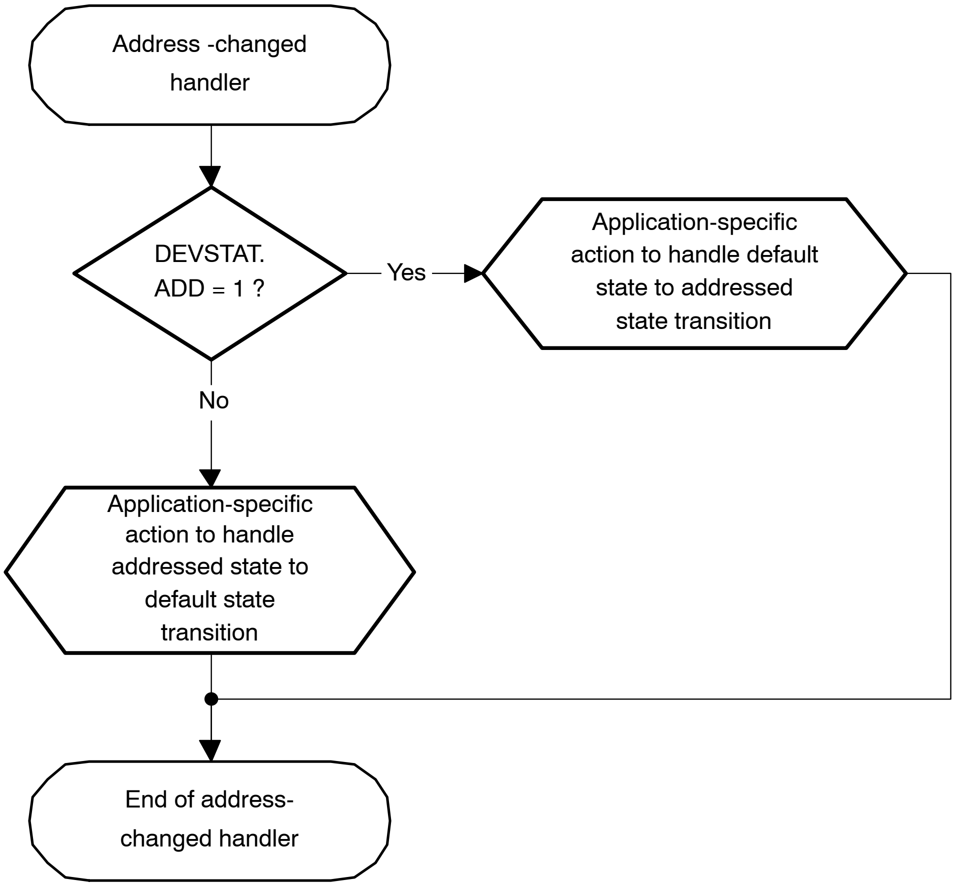
Figure 22−49. Configuration-Changed Handler


Configuration- changed handler
DEVSTAT. CFG = 1 ?
Yes
Application-specific action to handle addressed state to configured state transition
No
DEVSTAT. ADD = 1 ?
Yes
Application-specific action to handle configured state to addressed state transition
No
Application-specific action to handle configured state to default state transition
End of configuration-changed handler

22.5.15.3 Device State Address-Changed Handler

When an address-changed interrupt occurs, the USB device has received a set address operation. When this occurs, the address-changed handler performs the operations shown in Figure 22−50.

Figure 22−50. Address-Changed Handler

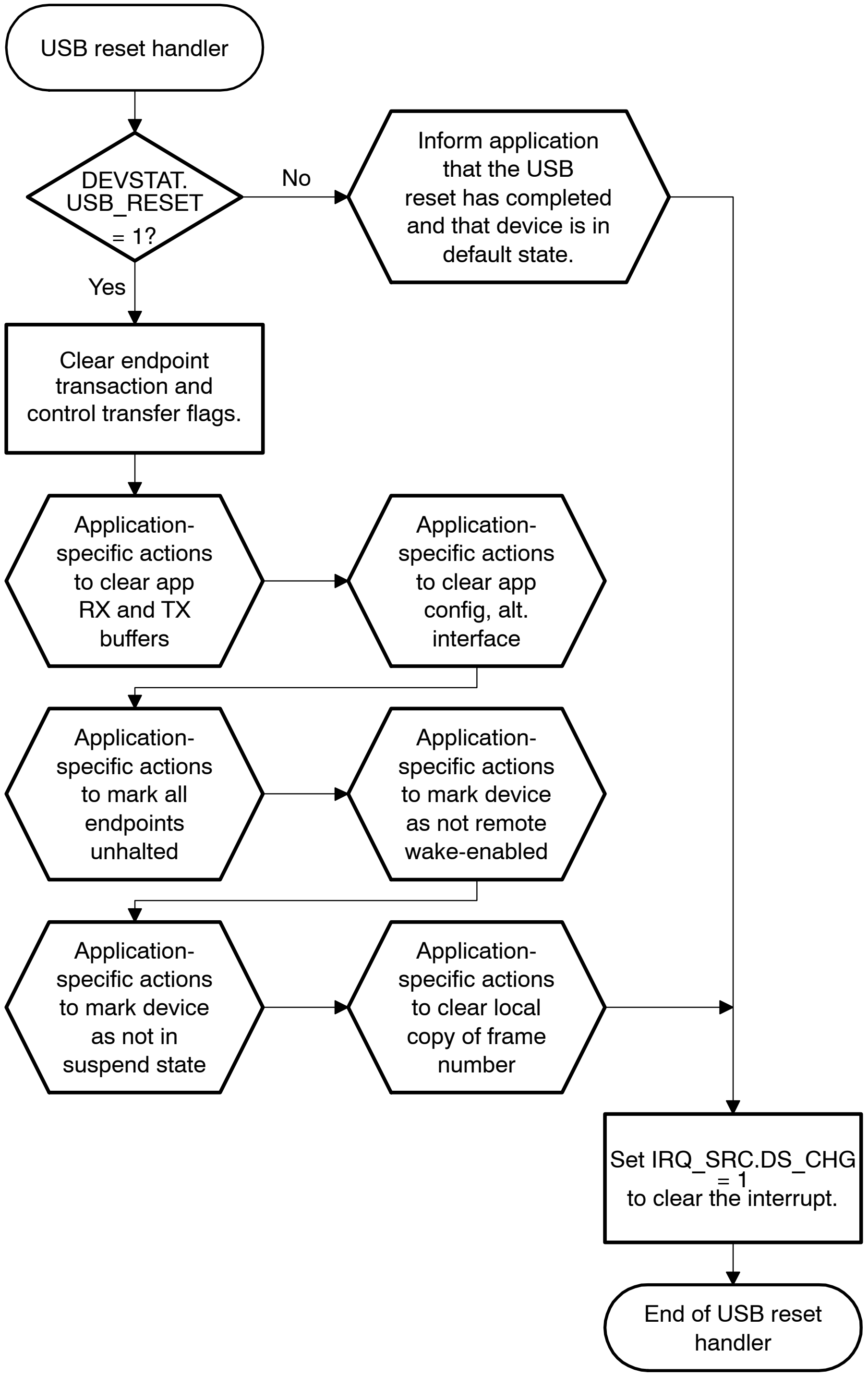

Address -changed handler
DEVSTAT. ADD = 1 ?
Yes
Application-specific action to handle default state to addressed state transition
No
Application-specific action to handle addressed state to default state transition
End of address-changed handler

22.5.15.4 USB Device Reset Interrupt Handler

When a USB reset occurs, the USB module generates a general USB interrupt to the MPU subsystem (see Figure 22−51). The MPU subsystem responds to this interrupt by performing the following operations:

- ❑ Cancels any ongoing USB transaction and/or control transfer handling
- ❑ Clears any copies that the application has of configuration number or alternate interface numbers
- ❑ Clears any application-specific information relating to halted endpoints
- ❑ Clears any application-specific information relating to the remote wake-enable flag
- ❑ Clears any application-specific information relating to the suspend state flag
- ❑ Clears any application-specific copy of the frame number

Figure 22–51. USB Device Reset Handler Flowchart


USB reset handler
DEVSTAT. USB_RESET = 1?
No
Inform application that the USB reset has completed and that device is in default state.
Yes
Clear endpoint transaction and control transfer flags.
Application-specific actions to clear app RX and TX buffers
Application-specific actions to clear app config, alt. interface
Application-specific actions to mark all endpoints unhalted
Application-specific actions to mark device as not remote wake-enabled
Application-specific actions to mark device as not in suspend state
Application-specific actions to clear local copy of frame number
Set IRQ_SRC.DS_CHG = 1 to clear the interrupt.
End of USB reset handler

22.5.15.5 Suspend/Resume Interrupt Handler

When a USB device suspend/resume general USB interrupt occurs, the USB module has either entered or left suspend state. The MPU code must determine which has occurred and react appropriately (see Figure 22−52).

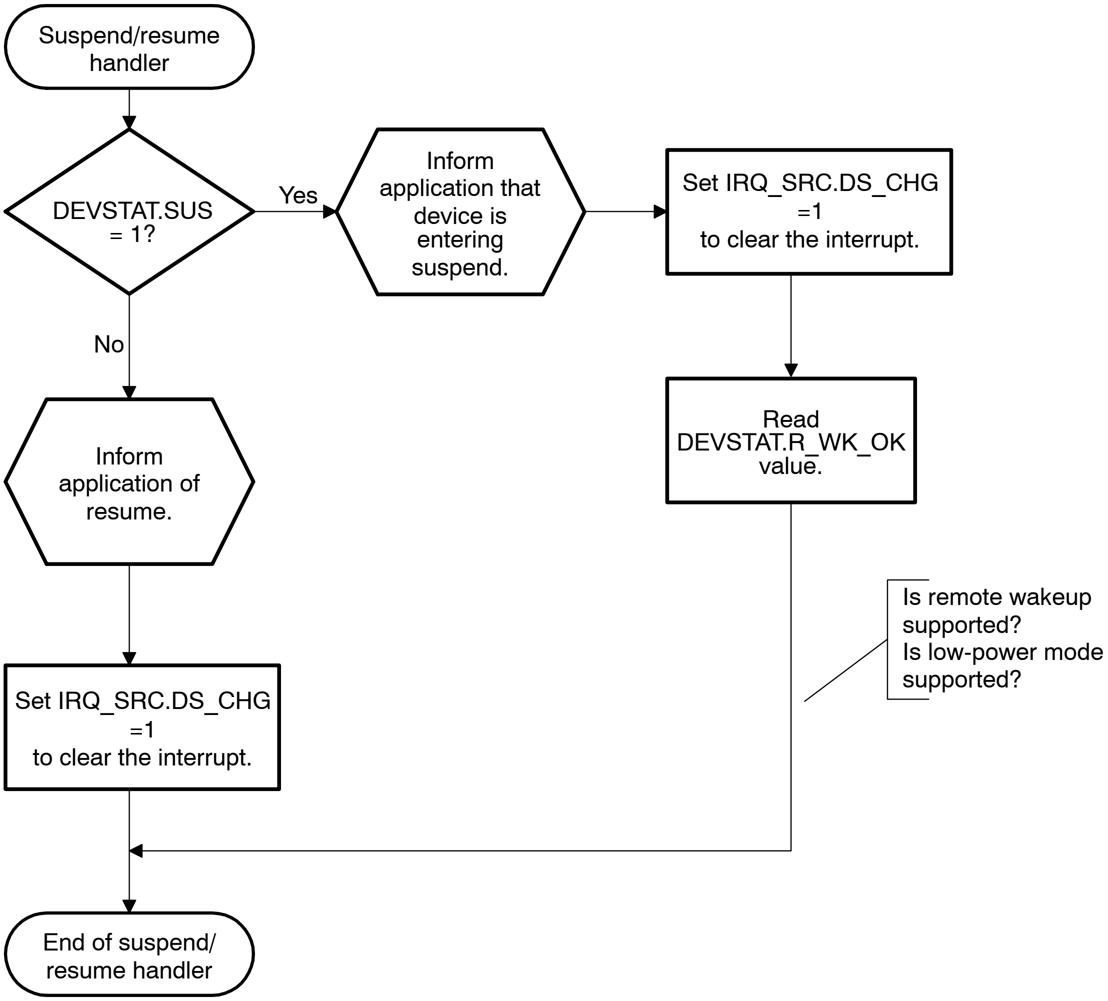
The suspend sense hardware is implemented to trigger only after 5 ms of bus idle. This forces compliance with the USB 1.1 specification T_{WTRSM} timing parameter (3 ms of IDLE to identify suspend, 2 ms before the remote device can signal resume).

If the MPU subsystem wants to wake the device from suspend state and remote wake-up enable is set (DEVSTAT.R_WK_OK = 1), it must first turn on its clock (if stopped), and then set the RMT_WKP bit SYSCON2[6]. The device then drives resume.

If shutoff is enabled (SYSCON1.SOFF_DIS = 0), the 48-MHz clock is automatically shut off at suspend and turned on at resume driven by the USB host or MPU subsystem. Setting or not setting the SOFF_DIS bit SYSCON1[1] is part of the device configuration. However, the MPU subsystem can modify its value at suspend interrupt time, if necessary.

A USB reset is also a valid way to exit suspend state. The suspend/resume handler and the USB reset handler do not have to consider this, however, because three interrupts are generated in that case (one for resume, one for reset, and one for end of reset).

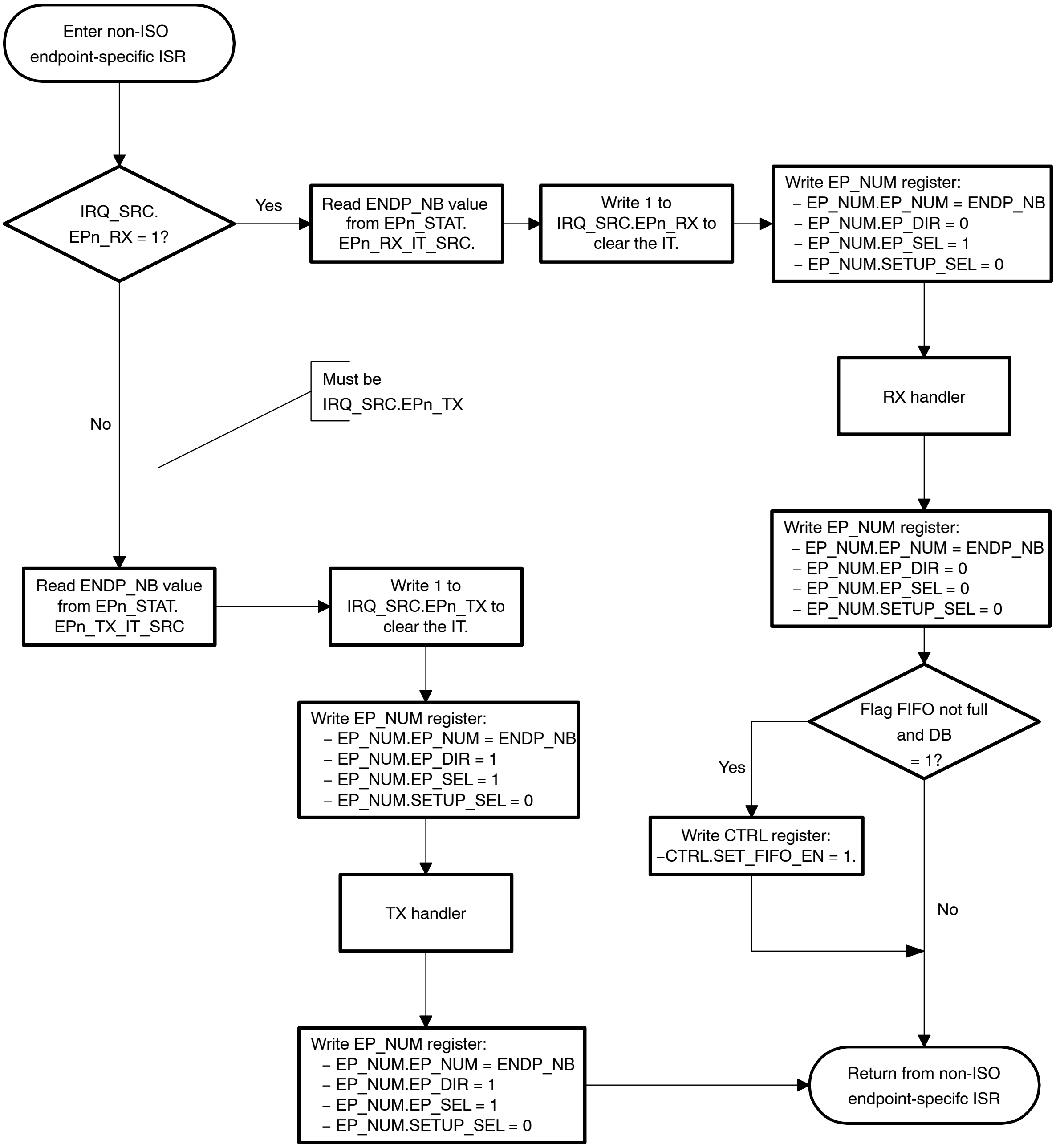
Figure 22–52. Typical Operation for USB Suspend/Resume General USB Interrupt Handler


Suspend/resume handler
DEVSTAT.SUS = 1?
Yes
Inform application that device is entering suspend.
Set IRQ_SRC.DS_CHG =1 to clear the interrupt.
No
Read DEVSTAT.R_WK_OK value.
Inform application of resume.
Is remote wakeup supported?
Is low-power mode supported?
Set IRQ_SRC.DS_CHG =1 to clear the interrupt.
End of suspend/ resume handler

22.5.16 Parsing Non-ISO Endpoint-Specific Interrupt

The USB_IRQ_NISO endpoint-specific interrupt ISR (also known as the non-isochronous interrupt, on M_IRQ_76) must parse the interrupt identifier register IRQ_SRC to determine the interrupts that are active (EPN_RX bit IRQ_SRC[5], EPN_TX bit IRQ_SRC[4], or both). The two interrupts can be active at any time. The ISR must then read the EPN_STAT register to determine the endpoint causing the interrupt. For each direction, only one endpoint interrupt can be active at a time (see Figure 22−53).

Figure 22−53. Non-ISO Endpoint-Specific (Except EP0) ISR Flowchart


Enter non-ISO endpoint-specific ISR
IRQ_SRC. EPn_RX = 1?
Yes
Read ENDP_NB value from EPn_STAT. EPn_RX_IT_SRC.
Write 1 to IRQ_SRC.EPn_RX to clear the IT.
Write EP_NUM register:
– EP_NUM.EP_NUM = ENDP_NB
– EP_NUM.EP_DIR = 0
– EP_NUM.EP_SEL = 1
– EP_NUM.SETUP_SEL = 0
RX handler
Write EP_NUM register:
– EP_NUM.EP_NUM = ENDP_NB
– EP_NUM.EP_DIR = 0
– EP_NUM.EP_SEL = 0
– EP_NUM.SETUP_SEL = 0
Flag FIFO not full and DB = 1?
Yes
Write CTRL register: –CTRL.SET_FIFO_EN = 1.
No
No
Must be IRQ_SRC.EPn_TX
Read ENDP_NB value from EPn_STAT. EPn_TX_IT_SRC
Write 1 to IRQ_SRC.EPn_TX to clear the IT.
Write EP_NUM register:
– EP_NUM.EP_NUM = ENDP_NB
– EP_NUM.EP_DIR = 1
– EP_NUM.EP_SEL = 1
– EP_NUM.SETUP_SEL = 0
TX handler
Write EP_NUM register:
– EP_NUM.EP_NUM = ENDP_NB
– EP_NUM.EP_DIR = 1
– EP_NUM.EP_SEL = 1
– EP_NUM.SETUP_SEL = 0
Return from non-ISO endpoint-specifc ISR

22.5.17 Non-ISO, Noncontrol OUT Endpoint Receive Interrupt Handler

Figure 22−54 shows the operations required to handle non-ISO, noncontrol OUT endpoint-specific receive interrupts. This flowchart shows two RX transaction handshaking interrupts. A third interrupt handshaking is possible when NAK interrupts are enabled, which is not shown in the figure. Depending on the application-specific actions required for various endpoints in the system, it is possible to use one routine that is common to all non-ISO, noncontrol receive endpoints, where the only differences are in the EP_NUM register value and the selection of the proper application RX data buffer in the read non-ISO RX FIFO data routine (see Figure 22−55).

This flowchart does not show control endpoint 0 receive interrupts, which are discussed separately because of the more complex 3-stage transfer mechanism used for control writes.

Figure 22−54. Nonisochronous, Noncontrol Endpoint Receive Interrupt Handler

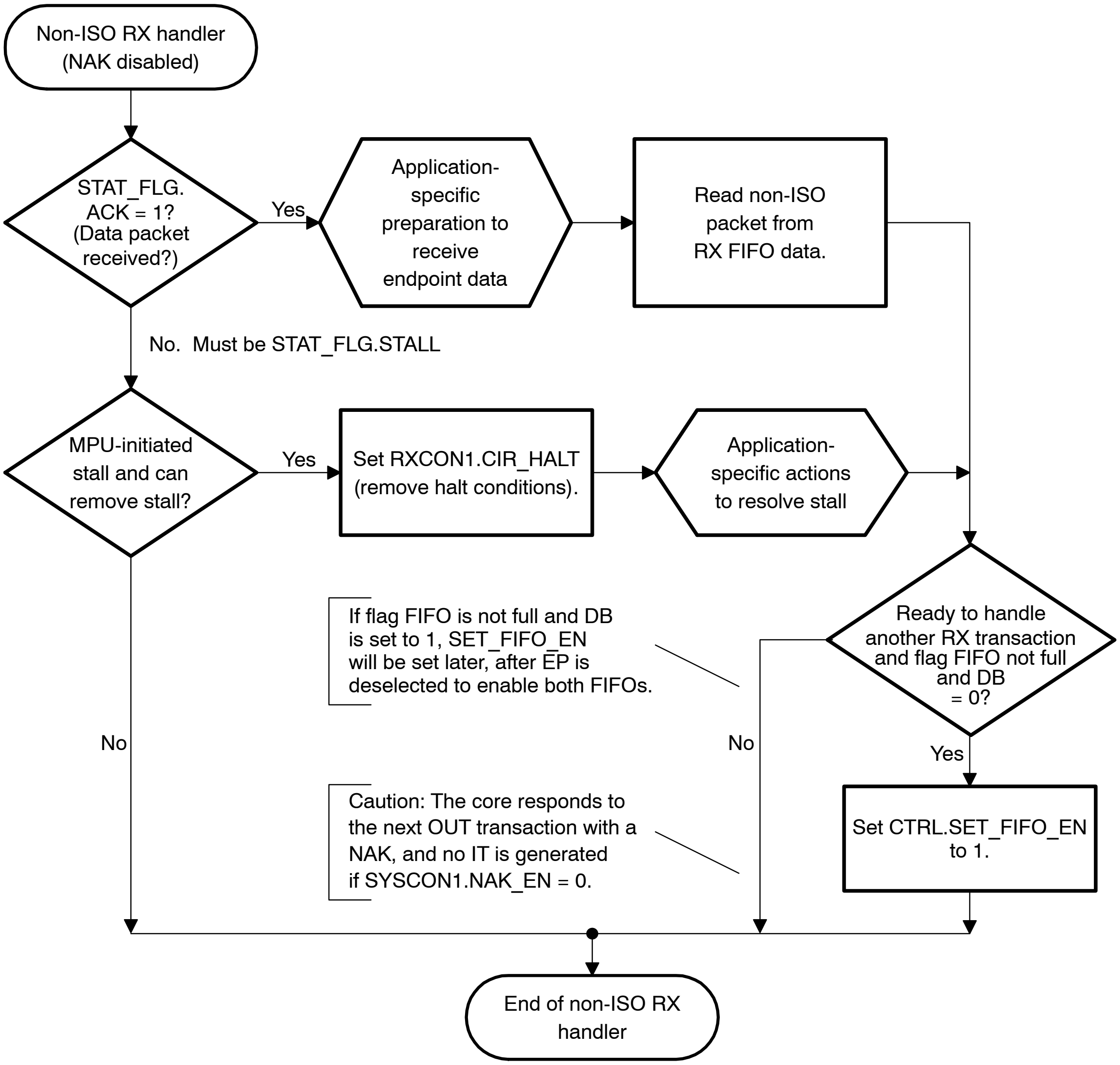
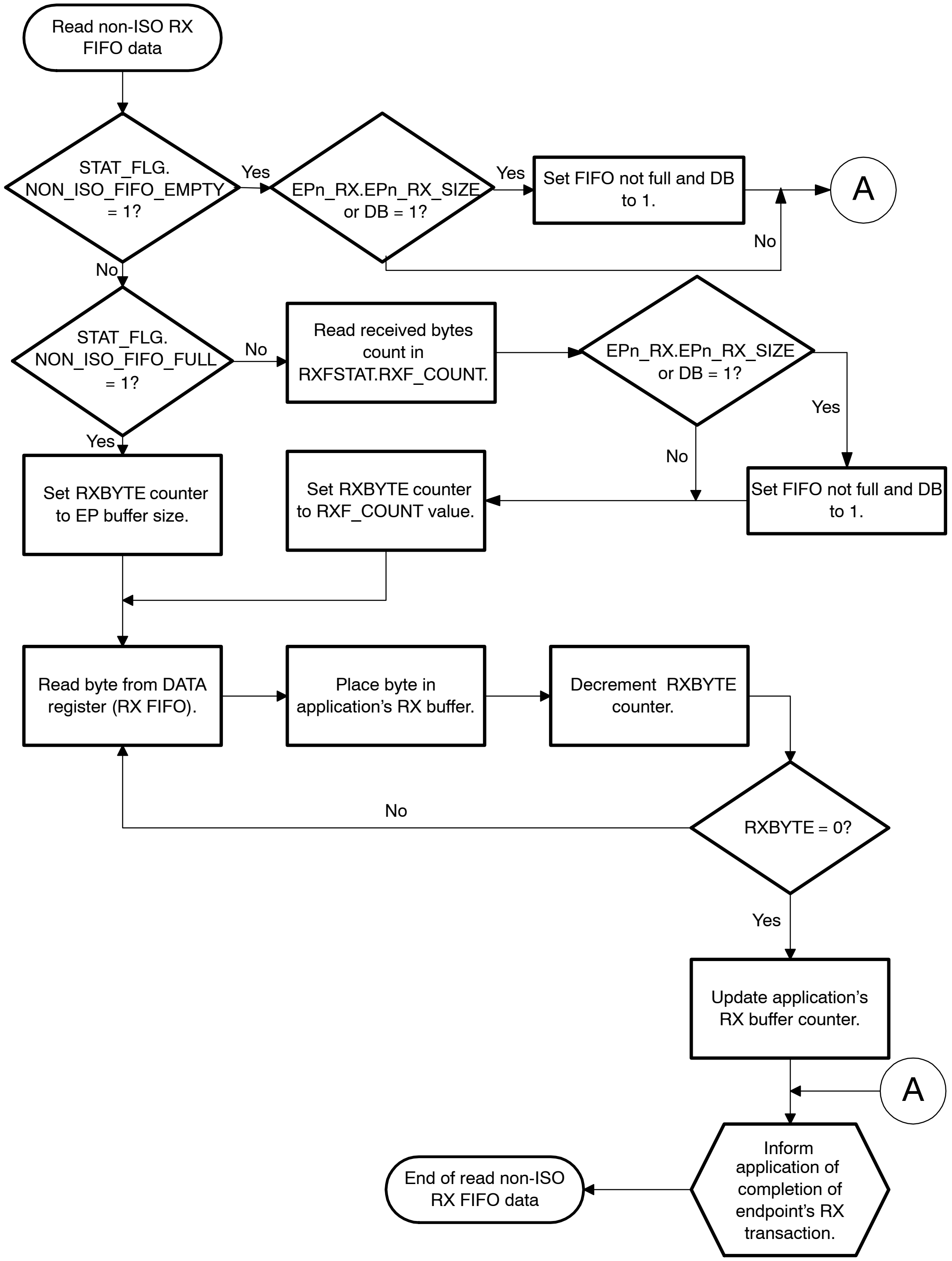

Non-ISO RX handler (NAK disabled)
STAT_FLG. ACK = 1? (Data packet received?)
Yes
Application-specific preparation to receive endpoint data
Read non-ISO packet from RX FIFO data.
No. Must be STAT_FLG.STALL
MPU-initiated stall and can remove stall?
Yes
Set RXCON1.CIR_HALT (remove halt conditions).
Application-specific actions to resolve stall
If flag FIFO is not full and DB is set to 1, SET_FIFO_EN will be set later, after EP is deselected to enable both FIFOs.
Ready to handle another RX transaction and flag FIFO not full and DB = 0?
No
No
Yes
Caution: The core responds to the next OUT transaction with a NAK, and no IT is generated if SYSCON1.NAK_EN = 0.
Set CTRL.SET_FIFO_EN to 1.
End of non-ISO RX handler

Figure 22−55. Read Nonisochronous RX FIFO Data Flowchart


Read non-ISO RX FIFO data
STAT_FLG. NON_ISO_FIFO_EMPTY = 1?
Yes
EPn_RX.EPn_RX_SIZE or DB = 1?
Yes
Set FIFO not full and DB to 1.
A
No
No
STAT_FLG. NON_ISO_FIFO_FULL = 1?
No
Read received bytes count in RXFSTAT.RXF_COUNT.
EPn_RX.EPn_RX_SIZE or DB = 1?
Yes
No
Yes
Set RXBYTE counter to EP buffer size.
Set RXBYTE counter to RXF_COUNT value.
Set FIFO not full and DB to 1.
Read byte from DATA register (RX FIFO).
Place byte in application's RX buffer.
Decrement RXBYTE counter.
No
RXBYTE = 0?
Yes
Update application's RX buffer counter.
A
End of read non-ISO RX FIFO data
Inform application of completion of endpoint's RX transaction.

22.5.18 Non-ISO, Noncontrol IN Endpoint Transmit Interrupt Handler

Figure 22−56 shows the operations required to handle non-ISO, noncontrol IN endpoint-specific transmit interrupts. This flowchart shows two TX transaction handshaking interrupts. A third interrupt handshaking is possible when NAK interrupts are enabled, which is not shown in this figure. Depending on the application-specific actions required for various endpoints in the system, it is possible to use one routine that is common to all non-ISO, noncontrol transmit endpoints, where the only differences are in the EP_NUM register value and in the routine selection of the application TX buffer (see Figure 22−57).

This flowchart does not show control endpoint 0 transmit interrupts, which are discussed separately because of the more complex 3-stage transfer mechanism used for control reads.

Figure 22−56. Nonisochronous, Noncontrol Endpoint Transmit Interrupt Handler

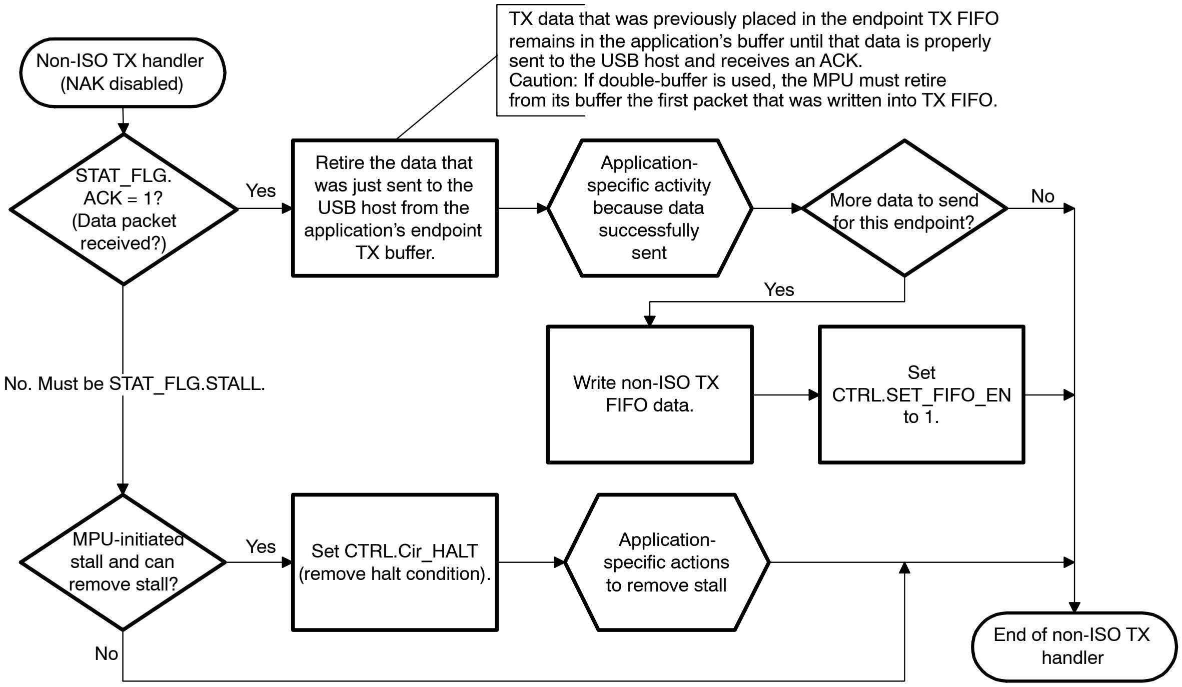
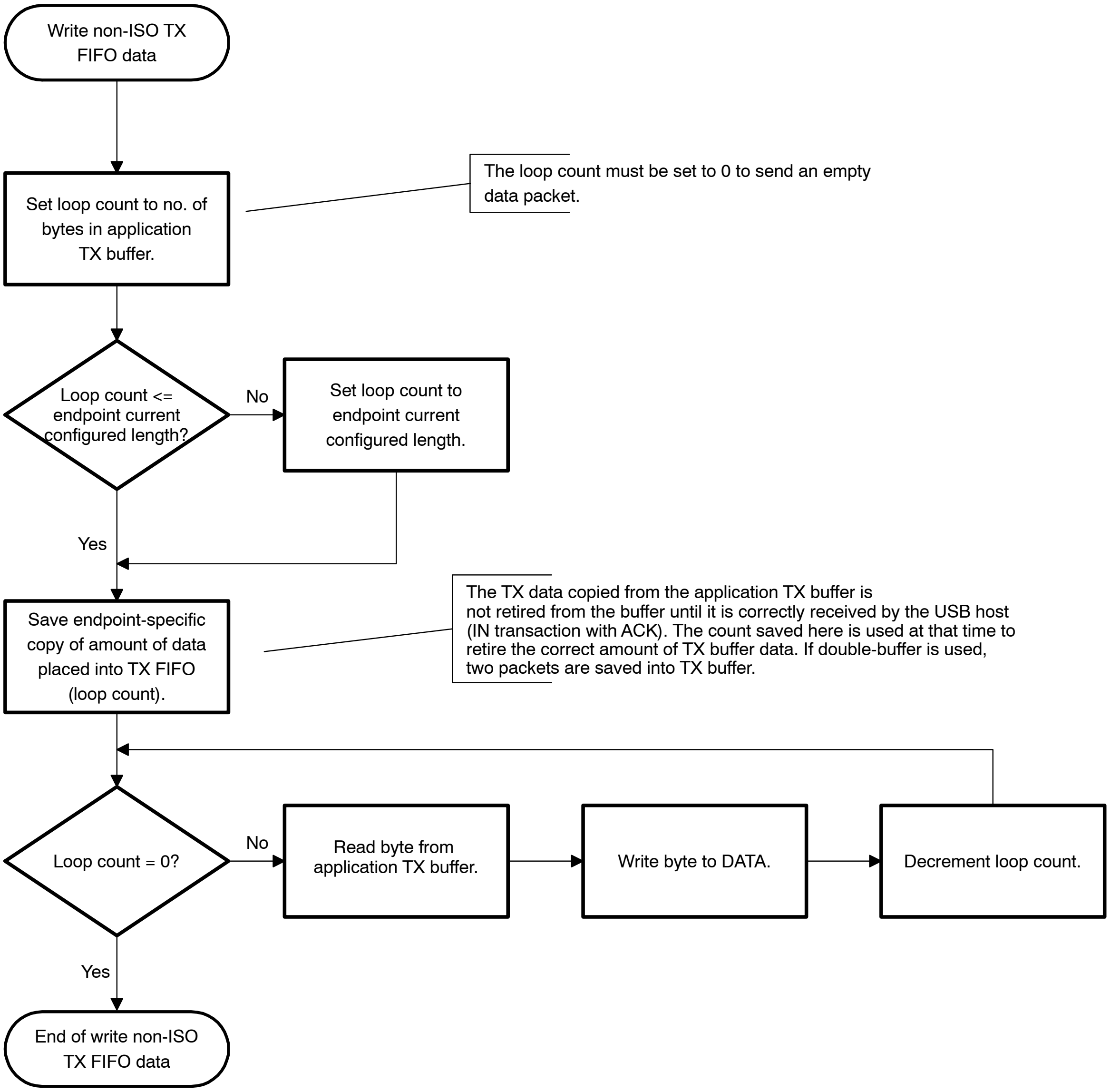

TX data that was previously placed in the endpoint TX FIFO remains in the application's buffer until that data is properly sent to the USB host and receives an ACK.
Caution: If double-buffer is used, the MPU must retire from its buffer the first packet that was written into TX FIFO.
Non-ISO TX handler (NAK disabled)
STAT_FLG. ACK = 1? (Data packet received?)
Yes
Retire the data that was just sent to the USB host from the application's endpoint TX buffer.
Application-specific activity because data successfully sent
More data to send for this endpoint?
No
Yes
No. Must be STAT_FLG.STALL.
Write non-ISO TX FIFO data.
Set CTRL.SET_FIFO_EN to 1.
MPU-initiated stall and can remove stall?
Yes
Set CTRL.Cir_HALT (remove halt condition).
Application-specific actions to remove stall
No
End of non-ISO TX handler

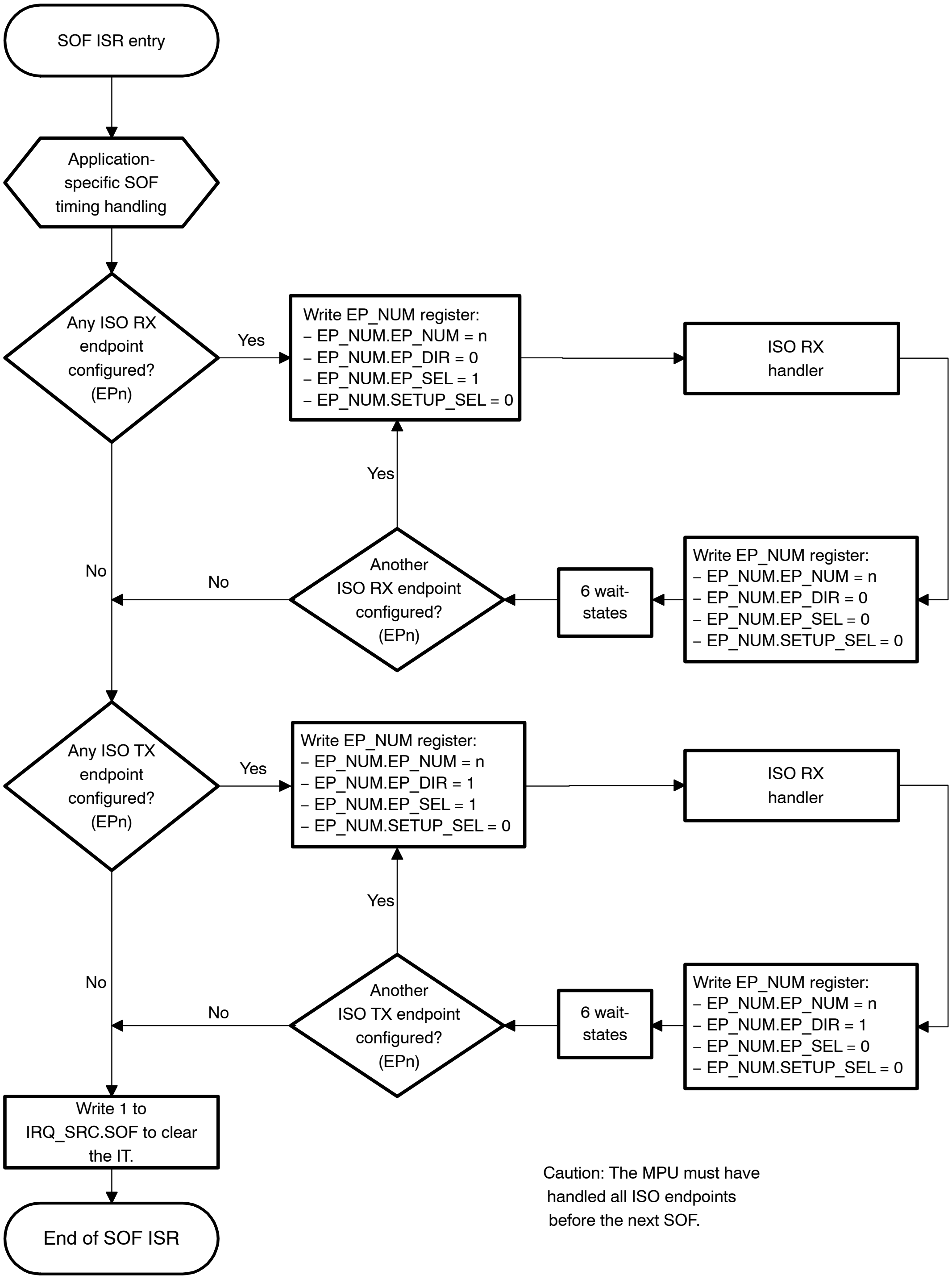
Figure 22−57. Write Nonisochronous TX FIFO Data Flowchart


Write non-ISO TX FIFO data
Set loop count to no. of bytes in application TX buffer.
The loop count must be set to 0 to send an empty data packet.
Loop count <= endpoint current configured length?
No
Set loop count to endpoint current configured length.
Yes
Save endpoint-specific copy of amount of data placed into TX FIFO (loop count).
The TX data copied from the application TX buffer is not retired from the buffer until it is correctly received by the USB host (IN transaction with ACK). The count saved here is used at that time to retire the correct amount of TX buffer data. If double-buffer is used, two packets are saved into TX buffer.
Loop count = 0?
No
Read byte from application TX buffer.
Write byte to DATA.
Decrement loop count.
Yes
End of write non-ISO TX FIFO data

22.5.19 SOF Interrupt Handler

The USB_IRQ_ISO SOF interrupts to the MPU subsystem (also known as isochronous interrupts, on M_IRQ_77) occur once per USB frame. The MPU subsystem must handle data traffic for the isochronous endpoints at each SOF interrupt. In addition, the SOF ISR can handle any application-specific activity related to the implicit timing of the SOF interrupt. Figure 22−58 shows the SOF ISR flowchart. The read ISO RX FIFO data and write ISO TX FIFO data procedures are shown in Figure 22−59 and Figure 22−60, respectively.

Figure 22–58. SOF Interrupt Handler Flowchart



Caution: The MPU must have handled all ISO endpoints before the next SOF.

Figure 22−59. Read Isochronous RX FIFO Data Flowchart

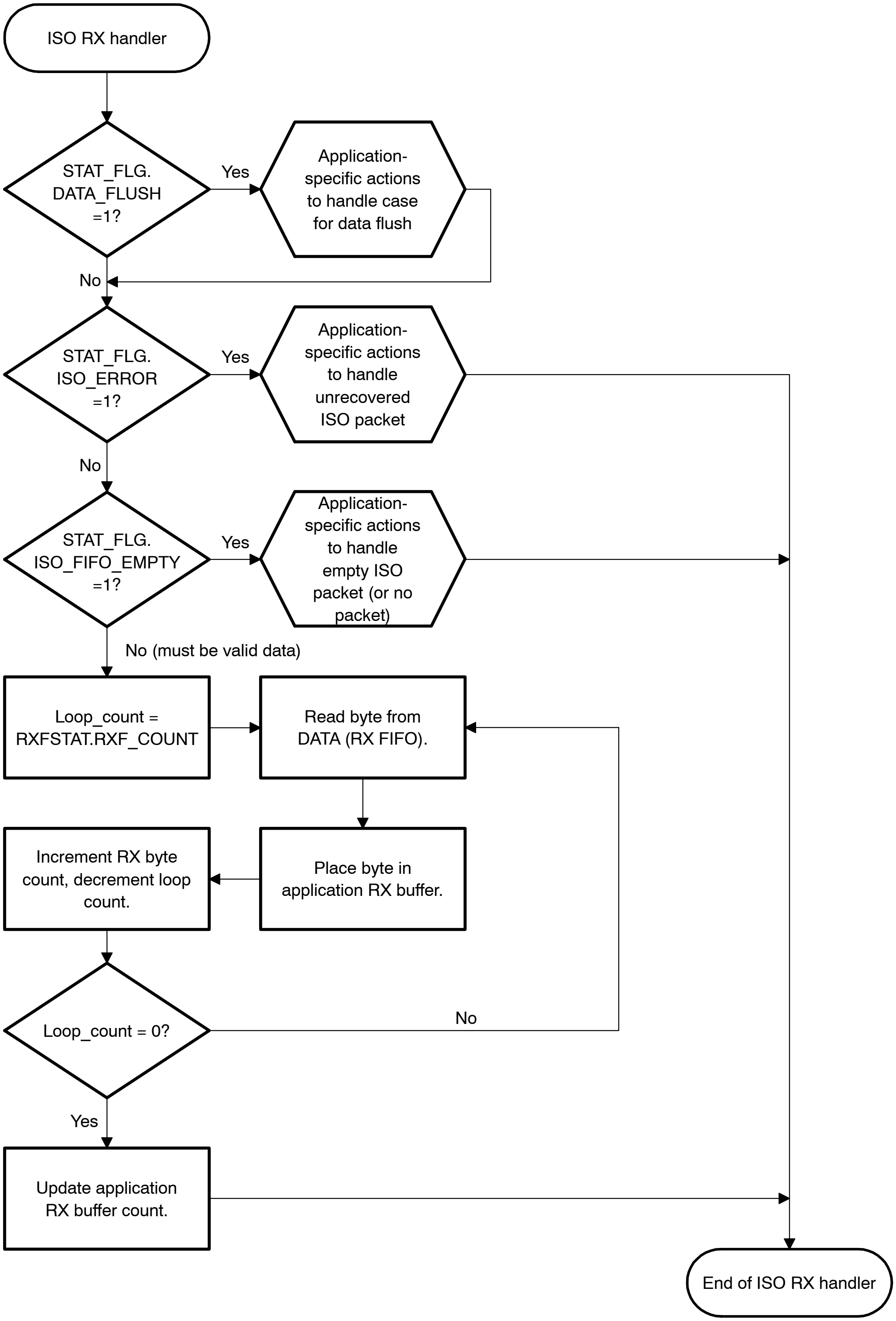
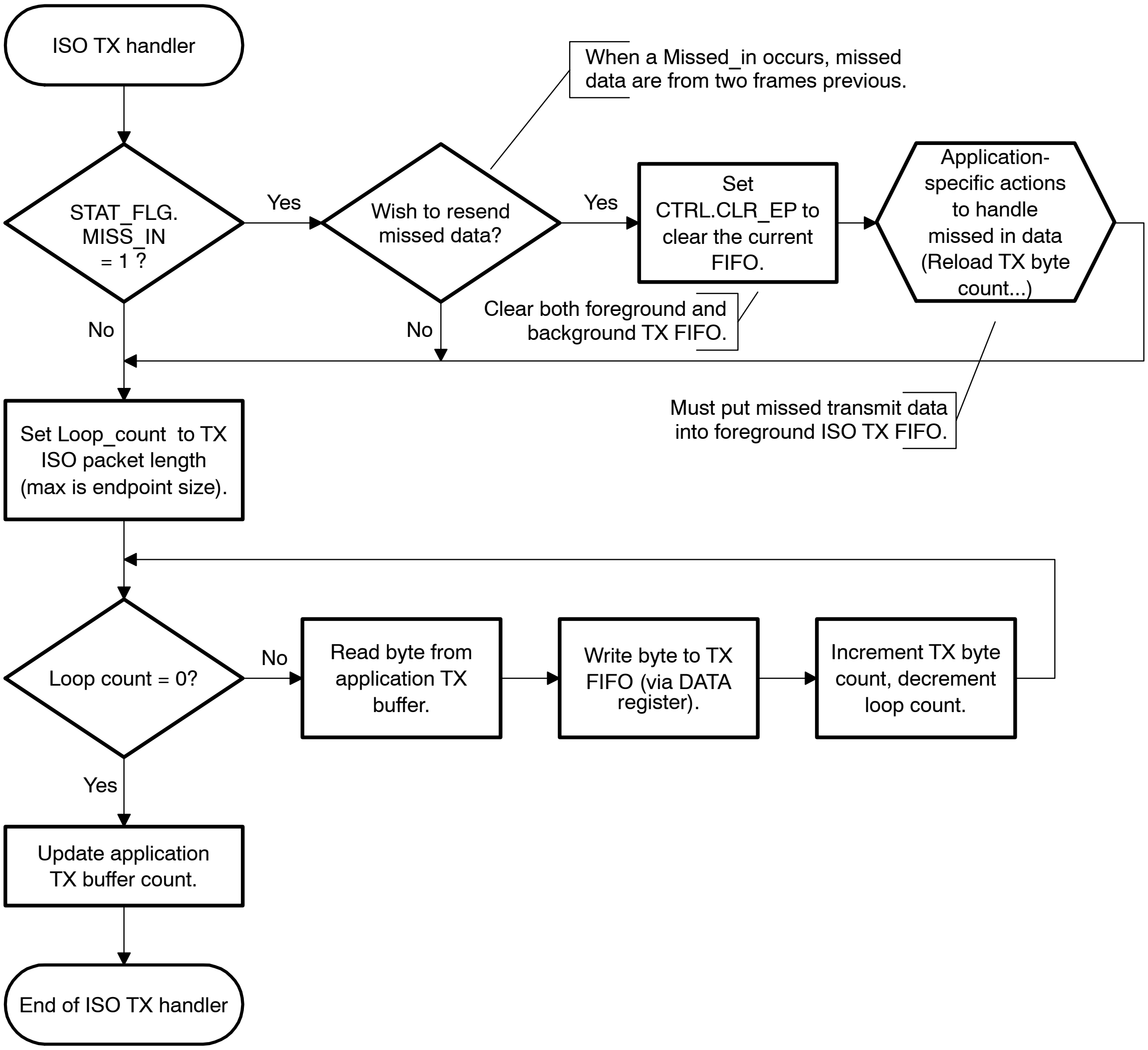

ISO RX handler
STAT_FLG.DATA_FLUSH =1?
Yes
Application-specific actions to handle case for data flush
No
STAT_FLG.ISO_ERROR =1?
Yes
Application-specific actions to handle unrecovered ISO packet
No
STAT_FLG.ISO_FIFO_EMPTY =1?
Yes
Application-specific actions to handle empty ISO packet (or no packet)
No (must be valid data)
Loop_count = RXFSTAT.RXF_COUNT
Read byte from DATA (RX FIFO).
Place byte in application RX buffer.
Increment RX byte count, decrement loop count.
Loop_count = 0?
No
Yes
Update application RX buffer count.
End of ISO RX handler

Figure 22−60. Write Isochronous TX FIFO Data Flowchart


ISO TX handler
When a Missed_in occurs, missed data are from two frames previous.
STAT_FLG. MISS_IN = 1 ?
Yes
Wish to resend missed data?
Yes
Set CTRL.CLR_EP to clear the current FIFO.
Application-specific actions to handle missed in data (Reload TX byte count...)
No
No
Clear both foreground and background TX FIFO.
Must put missed transmit data into foreground ISO TX FIFO.
Set Loop_count to TX ISO packet length (max is endpoint size).
Loop count = 0?
No
Read byte from application TX buffer.
Write byte to TX FIFO (via DATA register).
Increment TX byte count, decrement loop count.
Yes
Update application TX buffer count.
End of ISO TX handler

22.5.20 Summary of USB Device Controller Interrupts

Table 22−29 lists the interrupt types by endpoint types.

Table 22−29. USB Device Controller Interrupt Type by Endpoint Type

Interrupt Type	**USB_IRQ_GEN** **General USB IRQs (M_IRQ_75)**				**USB_IRQ_NISO** **EP-Specific IRQs (Non-ISO Interrupt on M_IRQ_76)**		**USB_IRQ_ISO** **SOF ISO IRQs (M_IRQ_77)**
	Setup (EP0)	**Control (EP0) Out**	**Control (EP0) In**	**Other**	**Bulk or Interrupt** **Out**	**Bulk or Interrupt** **In**	**(Isochronous)** **SOF**
Transaction ACKed		X	X		X	X	
Transaction NAKed (if enabled)		X	X		X	X	
Transaction STALLed		X	X		X	X	
Setup	X						
SOF							X
Device state changed				X			
RX DMA EOT (non-ISO)				X			
RX DMA trans count (non-ISO)				X			
TX DMA done (non-ISO)				X			

22.5.21 DMA Operation

The USB device controller provides support for six DMA channels: three receive DMA channels reserved to OUT transfers (ISO or non-ISO), and three transmit DMA channels reserved to IN transfers (ISO or non-ISO). It is not possible to operate DMA transactions on control EP0.

CAUTION

The MPU subsystem must not access an endpoint used in a DMA transfer through the EP_NUM, CTRL, or STAT_FLG registers (in DMA, this remark applies after the MPU subsystem has set the SET_FIFO_EN bit CTRL[2] to enable the RX DMA transfer). In particular, the MPU subsystem must not set the halt feature while the endpoint is selected in the RXDMA_CFG register.

Note:

To use the DMA channels properly, set the DMA configuration during the address state interrupt (DS_CHANGE).

The parameters used for DMA transactions (FIFO size, ISO endpoint, double-buffering, and pointers) are those defined for the associated endpoint.

Receive DMA Channels Overview

Receive DMA channels are programmed using the three RXDMA control registers. Each channel is assigned to a given endpoint number by assigning a non-zero value in the RXDMAn_EP fields RXDMA_CFG (a 0 value means the DMA channel is deselected). Received OUT data must be read through the register DATA_DMA when an RX DMA request is active. The RX FIFO accessed is that of the endpoint for which the DMA request is active (only one RX DMA request is active at a time).

The USB device controller receive DMA channels 0 through 2 are connected to OMAP2420 DMA controller requests S_DMA_55, S_DMA_57, and S_DMA_59, respectively. These DMA requests are made to the sDMA.

Nonisochronous OUT (USB Host –> MPU) DMA Transactions

During non-ISO transfers to a DMA-operated OUT endpoint, a request to the MPU DMA controller is generated when data are placed into endpoint FIFO and must be read. ACK and NAK interrupts are always disabled automatically by the core for DMA-operated endpoints.

There are two dedicated maskable interrupts per DMA channel to control non-ISO OUT transfers: end of transfer (EOT) and transaction count.

End of Transfer Interrupt (RXn_EOT Bit IRQ_SRC[8])

This interrupt signals that the core has detected an end-of-transfer (EOT). EOT occurs in the two following cases:

- When the last valid transaction to the endpoint is either an empty packet (ACK and buffer empty) or a packet whose size is less than the physical endpoint buffer size (ACK and buffer not full)
- When the number of received transactions has reached the programmed value in the RXn_TC field RXDMAn[7:0], if the RXn_STOP bit RXDMAn[15] is set by the MPU subsystem

After an EOT interrupt, the MPU subsystem must set the SET_FIFO_EN bit CTRL[2] for the endpoint to reenable the channel.

The MPU subsystem must not initiate a new RX DMA transfer until it receives an EOT interrupt.

Transaction Count Interrupt (RXn_CNT bit IRQ_SRC[9])

This interrupt provides watermark control. It can be used by the MPU subsystem to monitor the file size of incoming transfers and take appropriate actions if, for example, the received file exceeds an expected size.

A transaction count interrupt does not disable the ongoing DMA transfer.

A transaction count interrupt occurs each time the number of received transactions (not bytes) reaches the programmed value in the receive transaction counter for the DMA channel. One transaction has a size equal to the buffer size of the selected non-ISO endpoint. The RXn_COUNT interrupt is asserted even if the RXn_STOP bit RXDMAn[15] is set; in that case, both the RXn_COUNT and RXn_EOT interrupts are asserted (see Figure 22−61 through Figure 22−64).

The transaction count watermark is programmed in the RXDMAn register.

Figure 22−61. Non-ISO RX DMA Transaction Example (RX_TC=2)

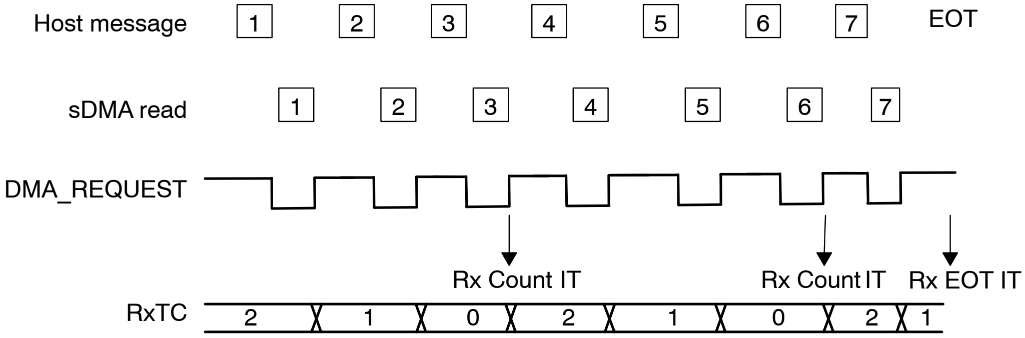


Figure 22−62. Non-ISO RX DMA Start Routine

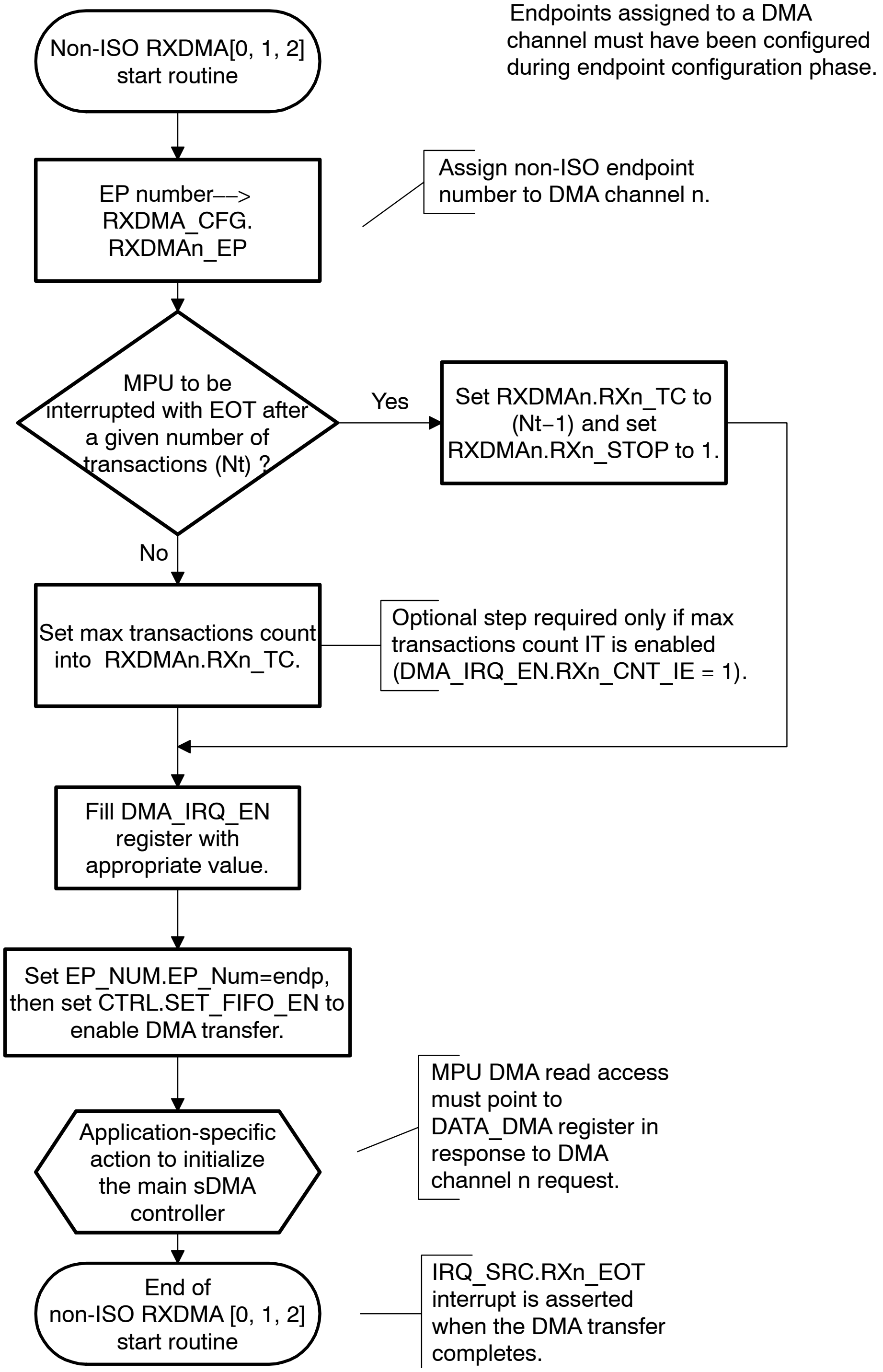

Endpoints assigned to a DMA channel must have been configured during endpoint configuration phase.
Non-ISO RXDMA[0, 1, 2] start routine
EP number−−> RXDMA_CFG. RXDMAn_EP
Assign non-ISO endpoint number to DMA channel n.
MPU to be interrupted with EOT after a given number of transactions (Nt) ?
Yes
Set RXDMAn.RXn_TC to (Nt−1) and set RXDMAn.RXn_STOP to 1.
No
Set max transactions count into RXDMAn.RXn_TC.
Optional step required only if max transactions count IT is enabled (DMA_IRQ_EN.RXn_CNT_IE = 1).
Fill DMA_IRQ_EN register with appropriate value.
Set EP_NUM.EP_Num=endp, then set CTRL.SET_FIFO_EN to enable DMA transfer.
Application-specific action to initialize the main sDMA controller
MPU DMA read access must point to DATA_DMA register in response to DMA channel n request.
End of non-ISO RXDMA [0, 1, 2] start routine
IRQ_SRC.RXn_EOT interrupt is asserted when the DMA transfer completes.

Figure 22−63. Non-ISO RX DMA EOT Interrupt Handler

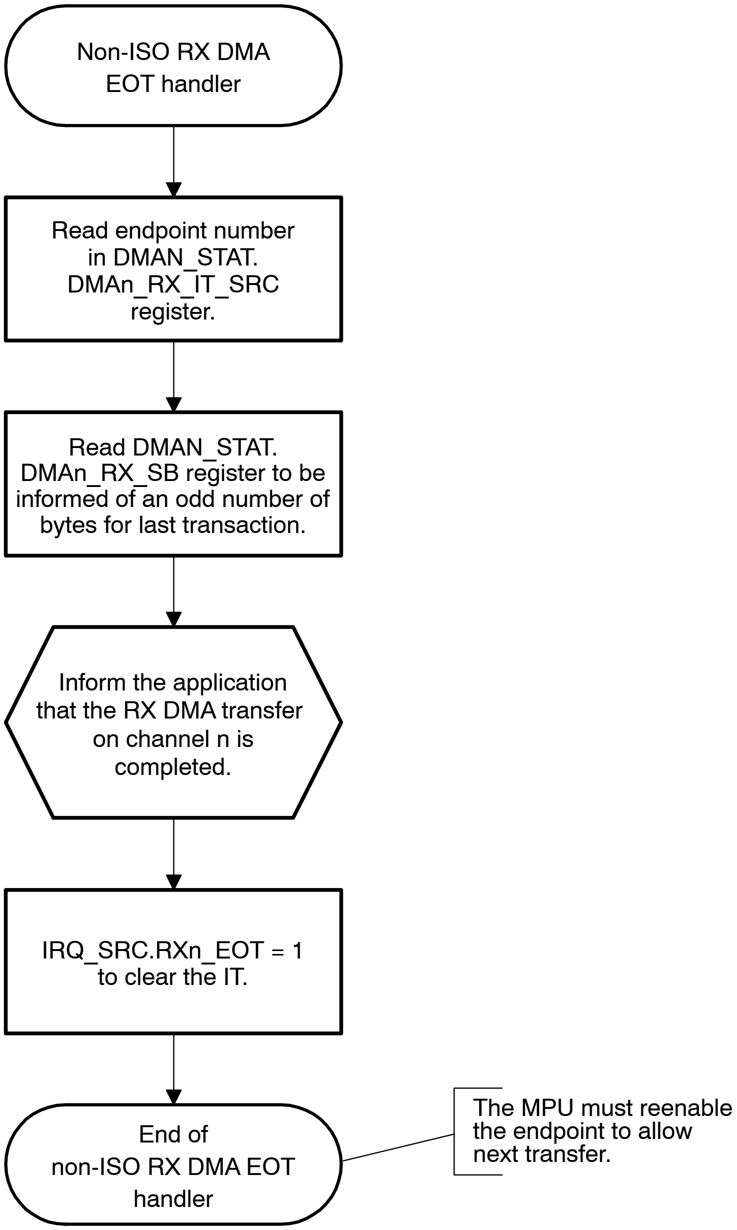
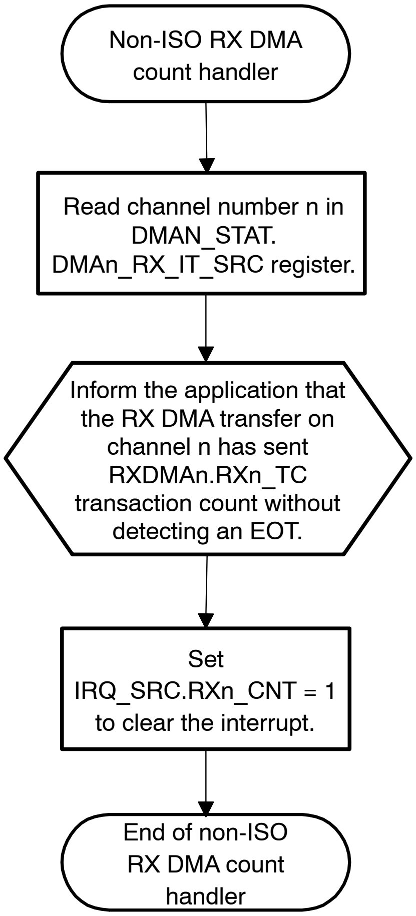

Non-ISO RX DMA EOT handler
Read endpoint number in DMAN_STAT. DMAn_RX_IT_SRC register.
Read DMAN_STAT. DMAn_RX_SB register to be informed of an odd number of bytes for last transaction.
Inform the application that the RX DMA transfer on channel n is completed.
IRQ_SRC.RXn_EOT = 1 to clear the IT.
End of non-ISO RX DMA EOT handler
The MPU must reenable the endpoint to allow next transfer.

Figure 22−64. Non-ISO RX DMA Transactions Count Interrupt Handler


Non-ISO RX DMA count handler
Read channel number n in DMAN_STAT. DMAn_RX_IT_SRC register.
Inform the application that the RX DMA transfer on channel n has sent RXDMAn.RXn_TC transaction count without detecting an EOT.
Set IRQ_SRC.RXn_CNT = 1 to clear the interrupt.
End of non-ISO RX DMA count handler

Isochronous OUT (USB Host –> MPU) DMA Transactions

During ISO transfers to a DMA-operated OUT endpoint, a request to the MPU DMA controller is generated every 1-ms frame when an isochronous data packet is received with no error. No interrupt is associated with DMA transfer to ISO OUT endpoints (see Figure 22−65 and Figure 22−66).

Figure 22−65. ISO RX DMA Transaction

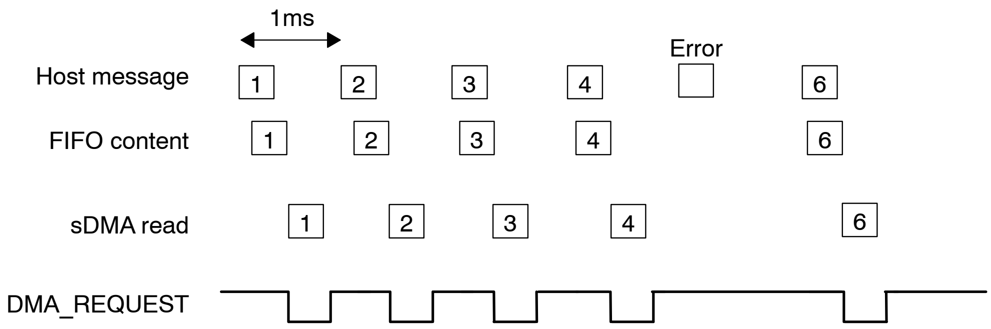
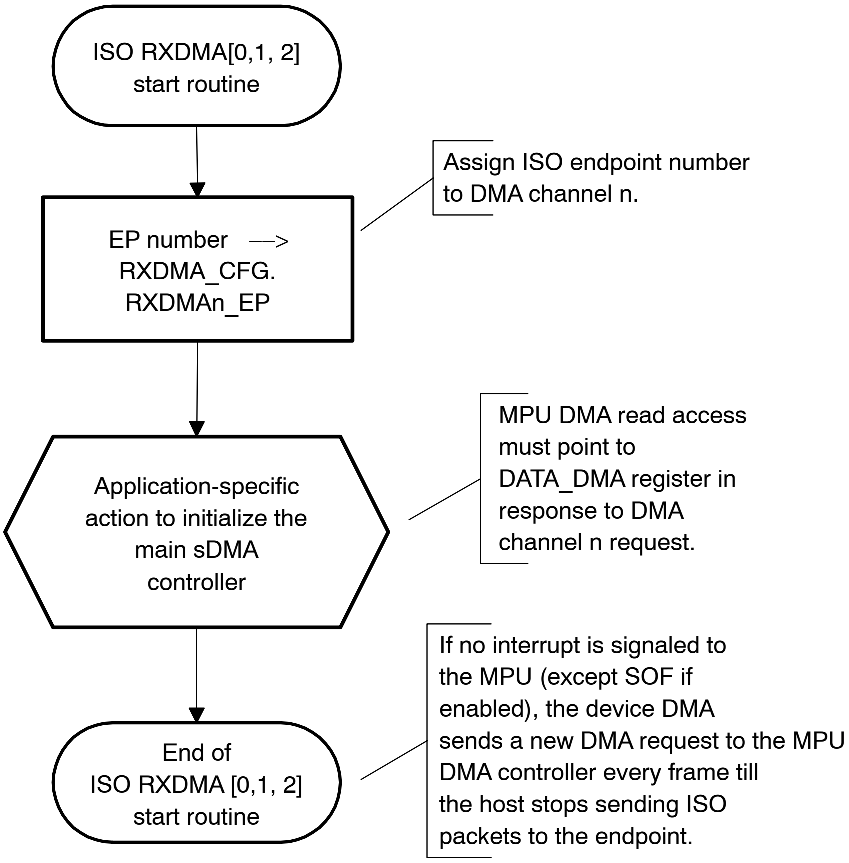


Figure 22−66. ISO RX DMA Start Routine


ISO RXDMA[0,1, 2] start routine
EP number —> RXDMA_CFG. RXDMAn_EP
Assign ISO endpoint number to DMA channel n.
Application-specific action to initialize the main sDMA controller
MPU DMA read access must point to DATA_DMA register in response to DMA channel n request.
End of ISO RXDMA [0,1, 2] start routine
If no interrupt is signaled to the MPU (except SOF if enabled), the device DMA sends a new DMA request to the MPU DMA controller every frame till the host stops sending ISO packets to the endpoint.

Transmit DMA Channels Overview

Transmit DMA channels are programmed using the three TXDMA control registers. Each channel is assigned to a given endpoint number by assigning a nonzero value in the TXDMAn_EP bits TXDMA_CFG (a 0 value means the DMA channel is deselected). The other three control registers (TXDMA0, TXDMA1, and TXDMA2) operate in a different manner for ISO and non-ISO endpoints. Transmitted data must be written into the DATA_DMA when a TX DMA request is active. They are written into the TX FIFO of the endpoint associated with active request (only one TX DMA request active at a given time).

The USB device controller transmit DMA channels 0 through 2 are connected to OMAP2420 DMA controller requests S_DMA_54, S_DMA_56, and S_DMA_58, respectively. These DMA requests are made to the sDMA.

Nonisochronous IN (MPU –> USB Host) DMA Transactions

Non-ISO (bulk) TX DMA file transfers are virtually unlimited in size. The flowcharts in Figure 22−68 and Figure 22−69 show how to handle small, medium, or large file transfers.

The TXDMA0, TXDMA1, and TXDMA2 registers operate for non-ISO endpoints in the following manner. The transfer size counter (TXN_TSC bits TXDMAn[9:0]) corresponds to either the number of bytes to transmit (EOT bit set) or the number of buffers to transmit (EOT bit cleared). The buffer size corresponds to the programmed size of the TX endpoint.

A request to the MPU main DMA controller is generated when the endpoint buffer is empty (initially after the START bit is set), and then each time there is space free in TX FIFO for other TX packets to be written, until the TXn_TSC bits TXDMAn[9:0] count down to zero. The request is removed when the buffer is full or when there are no more bytes of data to be sent.

A DMA transmit transfer done interrupt is signaled to the MPU subsystem after the last IN transaction completes successfully. This is after the START bit is set and after the TXn_TSC bits TXDMAn[9:0] equal 0 for the selected DMA channel.

The MPU subsystem must not initiate a new TX DMA transfer until it receives a TX_DONE interrupt.

A small file transfer less than 1024 bytes can be achieved in a single-pass DMA signaled by a single interrupt completion. A file size equal to or greater than 1024 bytes requires two or more DMA passes, signaled by an interrupt completion after each pass (see Example 22−1).

The file transfer size (FTZ) can be conceptualized as a concatenation of three arguments, as shown in Figure 22−67.

Figure 22−67. File Transfer Size

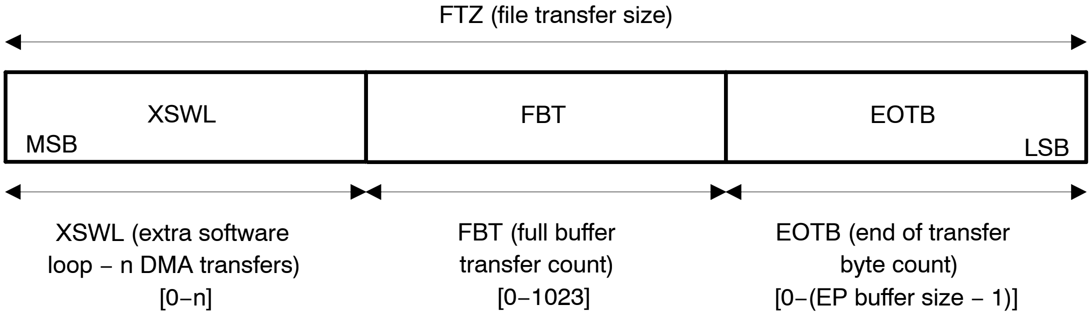

FTZ (file transfer size)
XSWL
MSB
FBT
EOTB
LSB
XSWL (extra software loop − n DMA transfers) [0−n]
FBT (full buffer transfer count) [0−1023]
EOTB (end of transfer byte count) [0−(EP buffer size − 1)]

Figure 22−68 shows the required steps to prepare and permit a TX DMA transfer of any size. It also effectively starts the initial DMA transfer. The completion of this DMA task is signaled to the MPU subsystem through a DONE interrupt, whose handler is shown in Figure 22−69. The start routine and the associated interrupt handler are tightly coupled.

Figure 22−68. Non-ISO TX DMA Start Routine

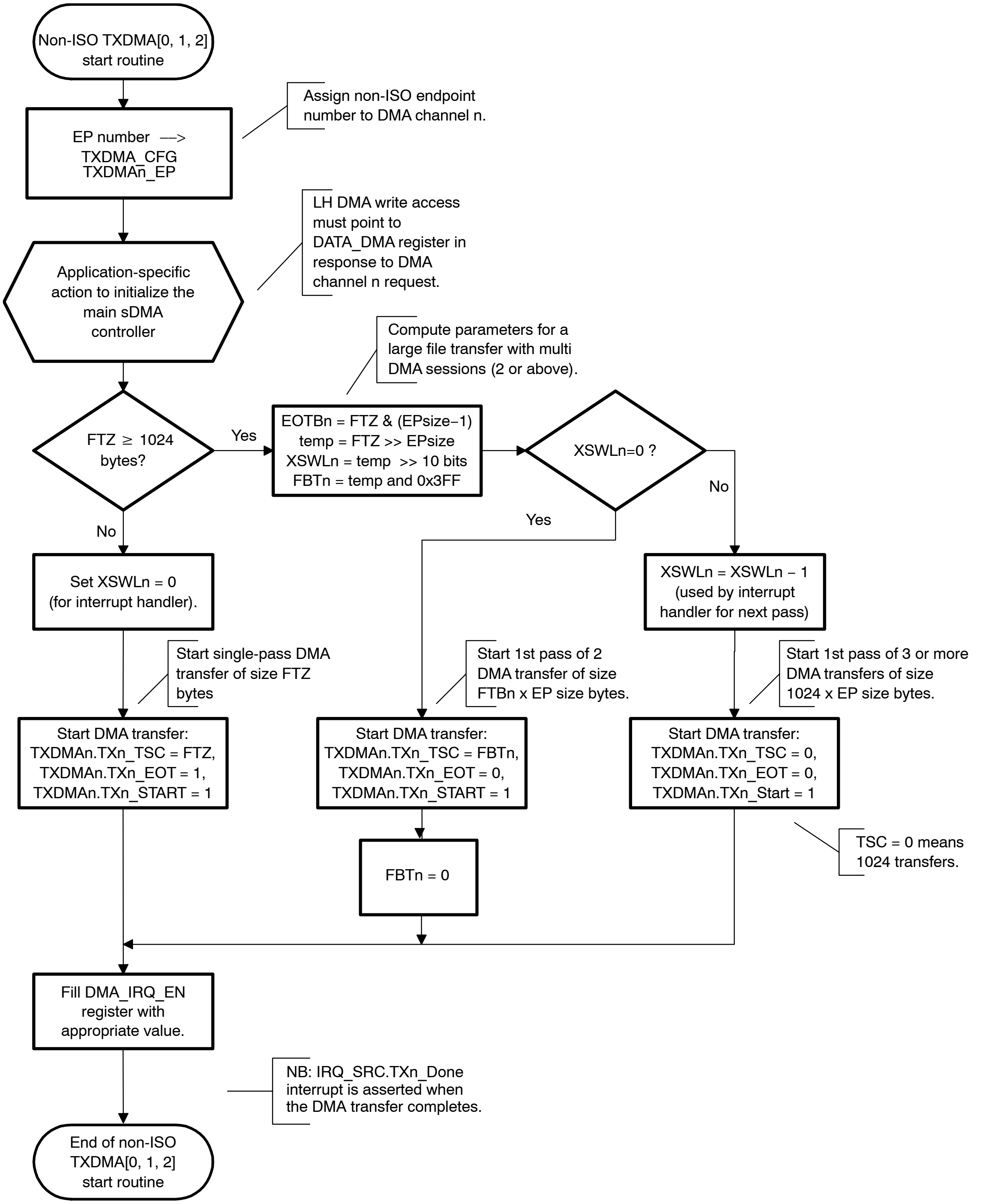
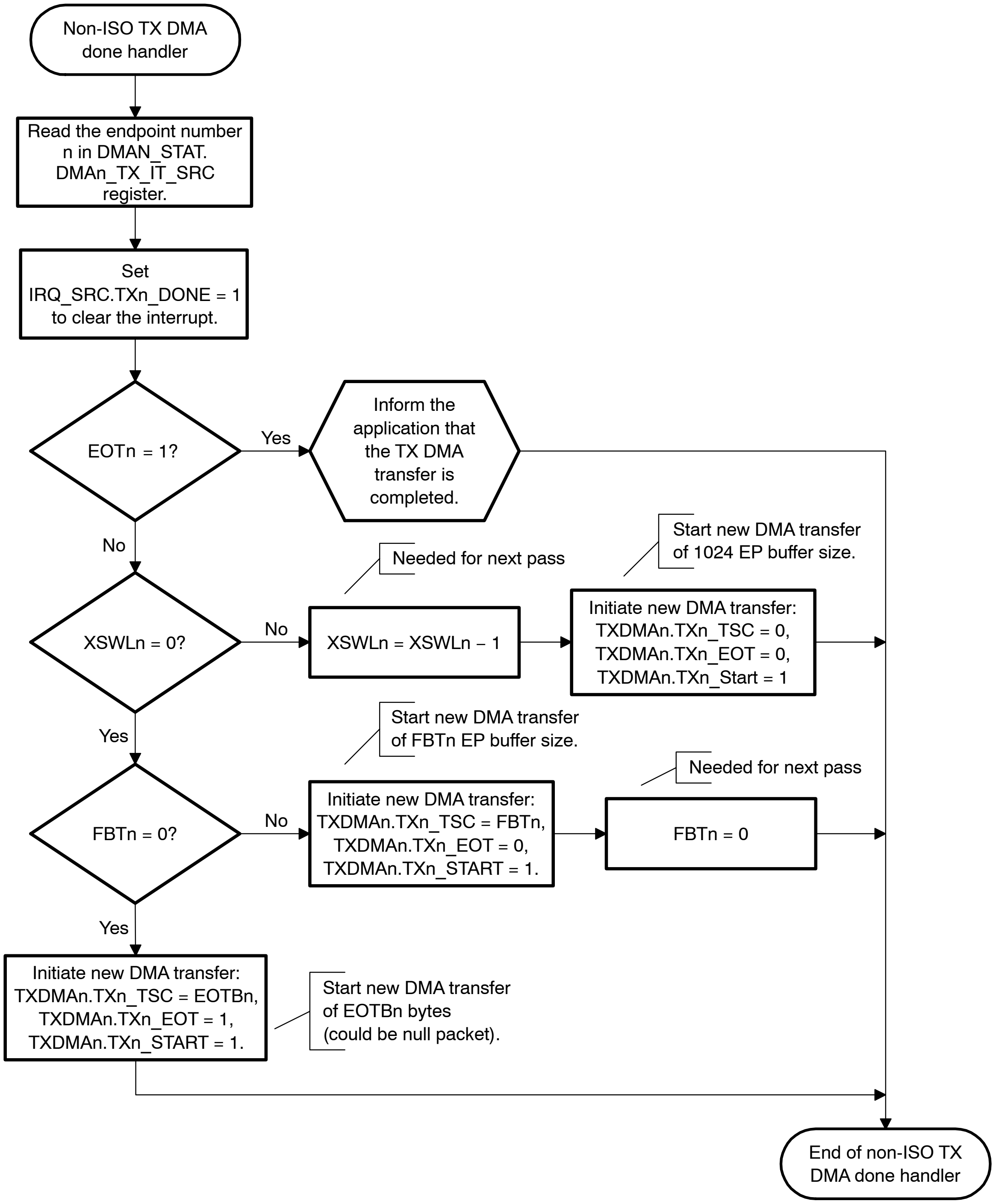

Non-ISO TXDMA[0, 1, 2] start routine
EP number —> TXDMA_CFG TXDMAn_EP
Assign non-ISO endpoint number to DMA channel n.
Application-specific action to initialize the main sDMA controller
LH DMA write access must point to DATA_DMA register in response to DMA channel n request.
Compute parameters for a large file transfer with multi DMA sessions (2 or above).
FTZ ≥ 1024 bytes?
Yes
EOTBn = FTZ & (EPsize−1)
temp = FTZ >> EPsize
XSWLn = temp >> 10 bits
FBTn = temp and 0x3FF
XSWLn=0 ?
No
Yes
No
Set XSWLn = 0 (for interrupt handler).
XSWLn = XSWLn − 1 (used by interrupt handler for next pass)
Start single-pass DMA transfer of size FTZ bytes
Start 1st pass of 2 DMA transfer of size FTBn x EP size bytes.
Start 1st pass of 3 or more DMA transfers of size 1024 x EP size bytes.
Start DMA transfer: TXDMAn.TXn_TSC = FTZ, TXDMAn.TXn_EOT = 1, TXDMAn.TXn_START = 1
Start DMA transfer: TXDMAn.TXn_TSC = FBTn, TXDMAn.TXn_EOT = 0, TXDMAn.TXn_START = 1
Start DMA transfer: TXDMAn.TXn_TSC = 0, TXDMAn.TXn_EOT = 0, TXDMAn.TXn_Start = 1
TSC = 0 means 1024 transfers.
FBTn = 0
Fill DMA_IRQ_EN register with appropriate value.
NB: IRQ_SRC.TXn_Done interrupt is asserted when the DMA transfer completes.
End of non-ISO TXDMA[0, 1, 2] start routine

Figure 22−69. Non-ISO TX DMA Done Interrupt Handler


Non-ISO TX DMA done handler
Read the endpoint number n in DMAN_STAT. DMAn_TX_IT_SRC register.
Set IRQ_SRC.TXn_DONE = 1 to clear the interrupt.
EOTn = 1?
Yes
Inform the application that the TX DMA transfer is completed.
No
XSWLn = 0?
No
Needed for next pass
XSWLn = XSWLn − 1
Start new DMA transfer of 1024 EP buffer size.
Initiate new DMA transfer: TXDMAn.TXn_TSC = 0, TXDMAn.TXn_EOT = 0, TXDMAn.TXn_Start = 1
Yes
FBTn = 0?
No
Start new DMA transfer of FBTn EP buffer size.
Initiate new DMA transfer: TXDMAn.TXn_TSC = FBTn, TXDMAn.TXn_EOT = 0, TXDMAn.TXn_START = 1.
Needed for next pass
FBTn = 0
Yes
Initiate new DMA transfer: TXDMAn.TXn_TSC = EOTBn, TXDMAn.TXn_EOT = 1, TXDMAn.TXn_START = 1.
Start new DMA transfer of EOTBn bytes (could be null packet).
End of non-ISO TX DMA done handler

Example 22−1. 100,603 Bytes to Transfer via 32 Bytes IN Bulk Endpoint

This gives XSWL = 0x3, FBT = 0x47, EOTB = 0x1b,

which means five passes of DMA transfer, signaled by five TXn_DONE interrupts, are required:

1) EOT=0, FBT=0, loop=3 . . . . 32,768 bytes transferred (1024 x 32 bytes)

2) EOT=0, FBT=0, loop=2 32,768 bytes

3) EOT=0, FBT=0, loop=1 32,768 bytes

4) EOT=0, FBT=0x47, loop=0 2,272 bytes (71 x 32 bytes)

5) EOT=1, FBT=0x1B, loop=0 27 bytes

Isochronous IN (USB Host –> MPU) DMA Transactions

For ISO endpoints, the transfer size counter (TXn_TSC bits TXDMAn[9:0]) corresponds to the number of bytes to transmit. The programmed size must not exceed the programmed buffer size of the endpoint. Otherwise, the result is unpredictable .

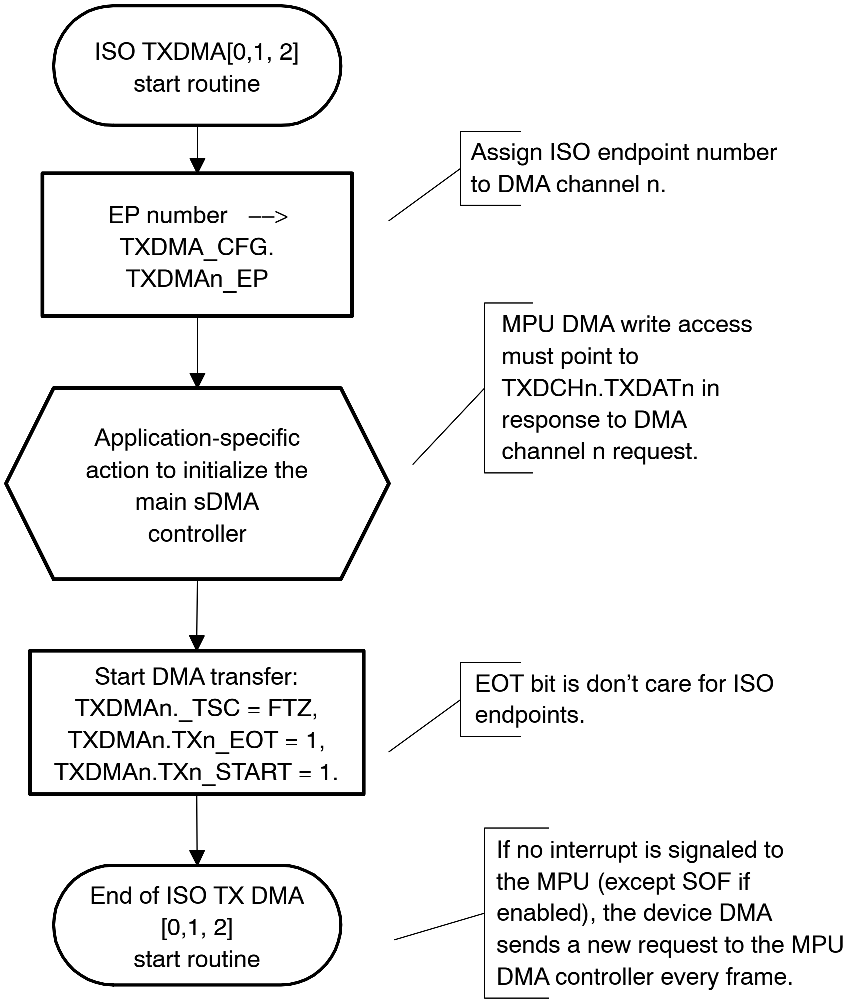
A request to the MPU main DMA controller is generated when the endpoint buffer is empty (initially after the START bit is set), and then after each SOF (every 1 ms). The request is removed when the number of bytes written in the buffer matches the TXn_TSC bits TXDMAn[9:0] value.

During ISO transfers to a DMA operated IN endpoint, a request to the MPU sDMA controller is generated every 1-ms frame when an isochronous data packet is received with no error. There is no special interrupt associated with the DMA transfer.

No interrupt is signaled to the MPU subsystem during DMA operation to ISO IN endpoints.

Figure 22−70 shows the ISO TX DMA start routine.

Figure 22−70. ISO TX DMA Start Routine


ISO TXDMA[0,1, 2]
start routine
EP number —>
TXDMA_CFG.
TXDMAn_EP
Assign ISO endpoint number to DMA channel n.
Application-specific action to initialize the main sDMA controller
MPU DMA write access must point to TXDCHn.TXDATn in response to DMA channel n request.
Start DMA transfer:
TXDMAn._TSC = FTZ,
TXDMAn.TXn_EOT = 1,
TXDMAn.TXn_START = 1.
EOT bit is don't care for ISO endpoints.
End of ISO TX DMA [0,1, 2] start routine
If no interrupt is signaled to the MPU (except SOF if enabled), the device DMA sends a new request to the MPU DMA controller every frame.

DMA Requests

For each direction, only one DMA request can be active at any time. A request must then be serviced to allow the next pending request on the same direction to be asserted. In particular, a TX DMA request is asserted at each SOF if a TX DMA channel is configured for an isochronous endpoint; it is imperative that this request be serviced.

DMA Channel Deconfiguration

It is recommended that the MPU subsystem wait for an EOT (RX) or a DONE (TX) interrupt before disabling the channel by writing 0 in the TX/RXDMA_CFG register. However, if the MPU subsystem is needed by the application, it can deselect the endpoint number in the TX/RXDMA_CFG register during a DMA transfer. The resulting behavior is as follows:

- For RX transfer:
  - If anRX DMA request is active for the endpoint when the endpoint is unselected, deconfiguration is effective only at the end of the

RX DMA request (that is, after all the DMA data are read). When double-buffering is used, the deconfiguration is effective after both buffers are read (if both buffers are not empty at deselection). An EOT interrupt is asserted if an EOT is detected.

  - If the RX DMA request is not active when deselection occurs, the effect is immediate.

- For TX transfer:
  - If the request is active when the endpoint is unselected, deconfiguration is effective after the TX DMA request is handled, and the TX data are sent through an IN transaction (both buffers in case of double-buffering with both buffers full). No TX_DONE interrupt is asserted even if the TXn_TSC bits TXDMAn[9:0] is 0 after the transaction.
  - If the TX DMA request is inactive when the endpoint is unselected, deconfiguration is effective when all data available in the TX buffer(s) are sent through IN transaction(s). If the TXn_TSC bits TXDMAn[9:0] is 0 at this point, no TX_DONE interrupt is asserted. If a TX_DONE interrupt is already asserted when the endpoint is deselected, configuration is effective only after TX_DONE interrupt handling.

The TX/RXDMAn_EP bits TX/RXDMA_CFG reflect the endpoint value until deconfiguration is effective. The MPU subsystem must read this register to know whether the channel is disabled. It must wait until the read value is 0 before performing other actions to the endpoint. After effective deconfiguration, all transactions to the endpoint generate an endpoint-specific interrupt (if non-transparent).

If the selected endpoint is isochronous, deconfiguration is effective after the TX/RX request is serviced, and the subsequent isochronous transactions are handled at the SOF interrupt through the endpoint registers (EP_NUM and STAT_FLG).

22.5.22 Power Management

Figure 22−71 shows the values assigned to the USB device controller signals concerned with power management, in the function of the device state. These signals are:

- PUEN_O: Pullup enable signal, always reflects the SYSCON1.PULLUP_EN register bit
- USB_FCLK_DISABLE: Power circuitry shutoff signal, active high to shut down the functional clock, controlled by the core and the SOFF_DIS bit FSUSB.SYSCON1[1]
- USB_ICLK_DISABLE: Asserted low to request the interface clock
- SUSPEND_O: Suspend signal, asserted high when the device is in suspend state

Figure 22−71. Power Management Signal Values

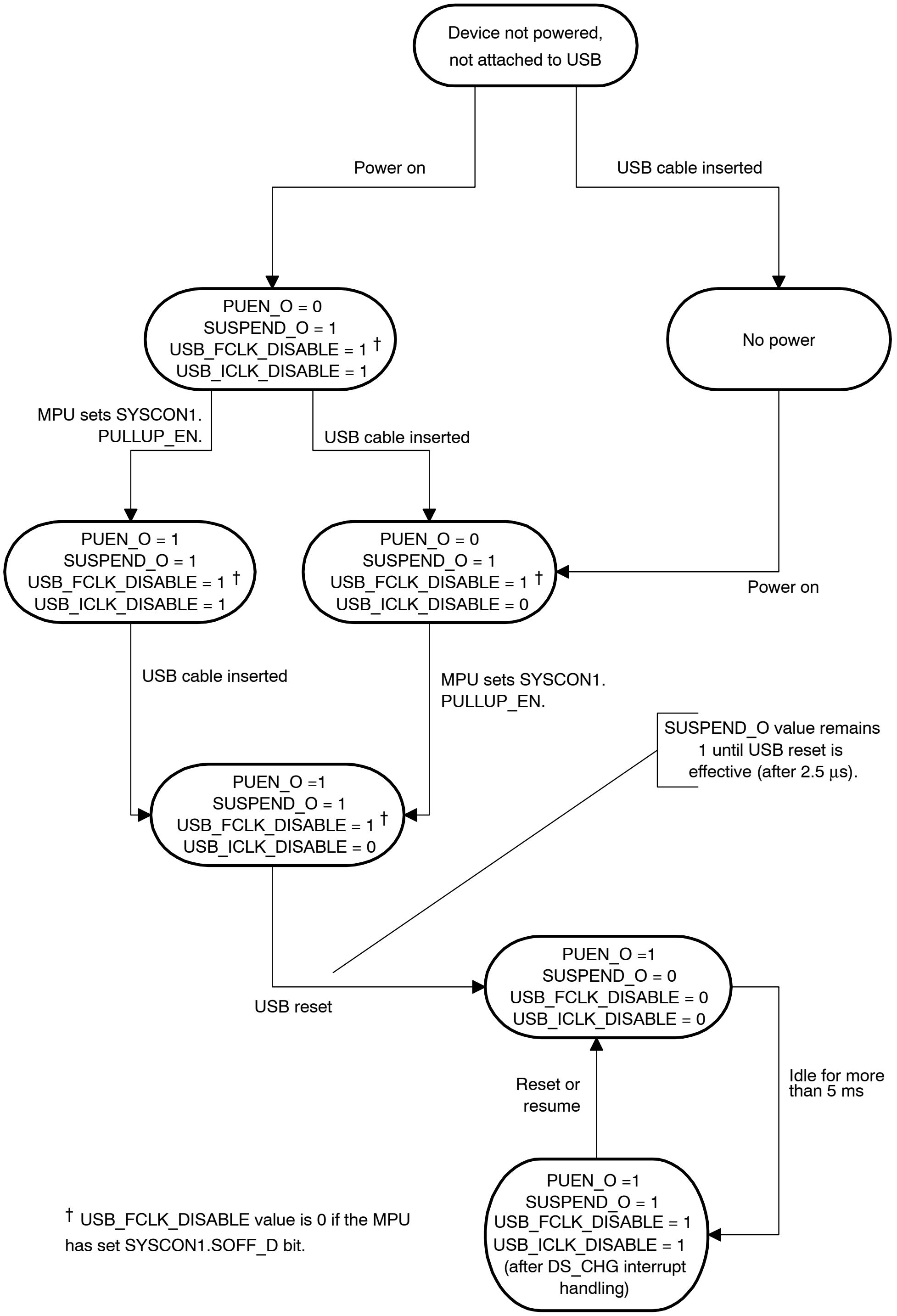
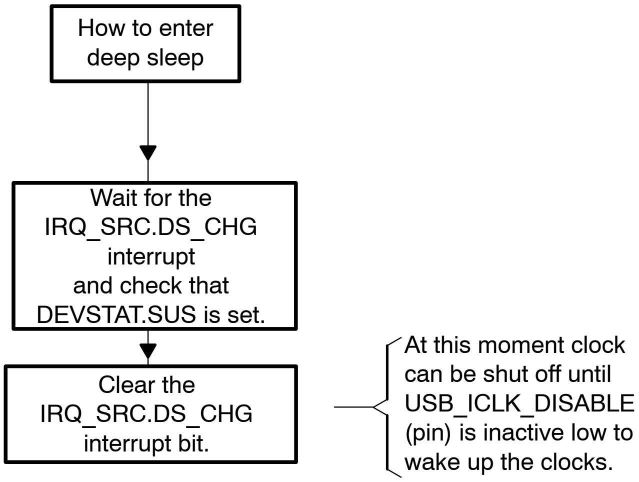


Figure 22−72 shows the reaction from a software point of view. This flowchart does not need to be implemented; it only reflects the way the module can enter deep sleep.

Figure 22−72. Power Management Flowchart


How to enter deep sleep
Wait for the IRQ_SRC.DS_CHG interrupt and check that DEVSTAT.SUS is set.
Clear the IRQ_SRC.DS_CHG interrupt bit.
At this moment clock can be shut off until USB_ICLK_DISABLE (pin) is inactive low to wake up the clocks.

22.6 USB OTG Controller

The OMAP2420 USB OTG controller works with the OMAP2420 USB device controller and one port of the OMAP2420 host controller to provide USB OTG functionality. The OMAP2420 OTG controller includes various control and status logic that switches between the two controllers as required by the USB OTG protocol. The combination allows the OMAP2420 device to act as a USB OTG DRD. The OMAP2420 OTG implementation simultaneously allows one USB OTG port and up to two additional USB host ports.

The USB OTG controller provides a DRD capability that is compatible with the to the OTG specification supplement. This DRD can provide a 12-Mbps connection between the OMAP2420 device and other USB OTG dual-role devices. The OMAP2420 OTG solution does not support 48-Mbps OTG connectivity.

In addition, the USB OTG controller provides control and status information for various aspects of the OMAP2420 USB host controller and the OMAP2420 USB device controller operation when OTG functionality is not enabled.

22.6.1 OTG Controller Features

The main features of the OMAP2420 USB OTG controller include:

- Compatability with the USB 2.0 specification
- When acting as an OTG A-peripheral or an OTG B-peripheral:
  - 12-Mbps communication link with configurable data transfer type
  - Data buffer size
  - Single- or double-buffering for each endpoint
  - Up to three IN endpoints using DMA and up to three OUT endpoints using DMA to support streaming USB data
- When acting as an OTG A-host or an OTG B-host: An OHCI specification-compatible USB host controller capable of USB full-speed (12 Mbps) and USB low-speed (1.5 Mbps) communication

These are basic features of the OMAP2420 USB device controller and the OMAP2420 USB host controller. Additional OTG-specific features provided by the OMAP2420 OTG controller include:

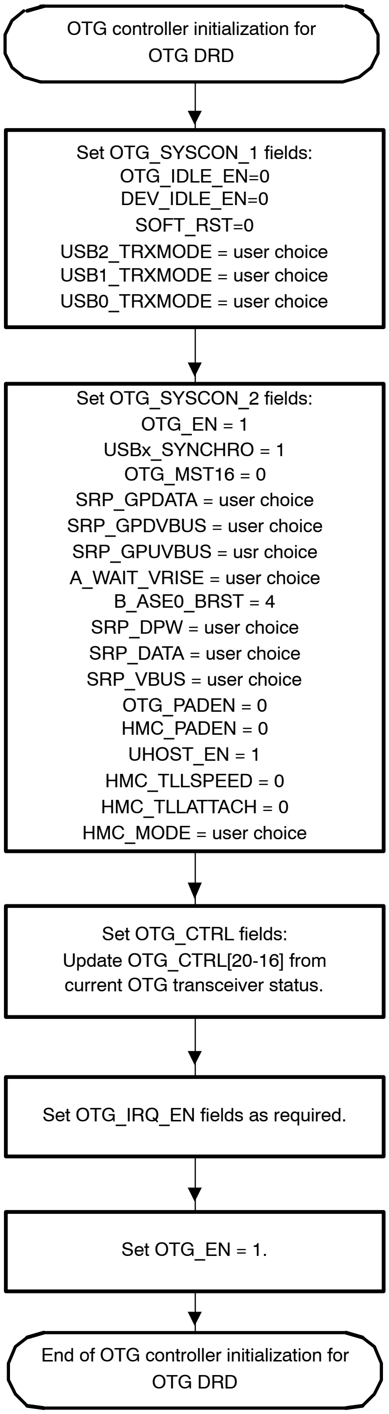
- Control and status logic that allows implementation of an OTG DRD
- Ability to implement an OTG peripheral-only device (that is, one that cannot act as an OTG A-device)
- Support for generation and detection of the OTG HNP and both types of OTG session request protocol
- Support for OTG transceivers that are compatible with the OTG working group *OTG Transceiver Interface Specification* (hereafter called the OTG transceiver specification)

22.6.1.1 OTG Controller Initialization

OTG controller initialization operations depend on whether or not the system implements an OTG DRD.

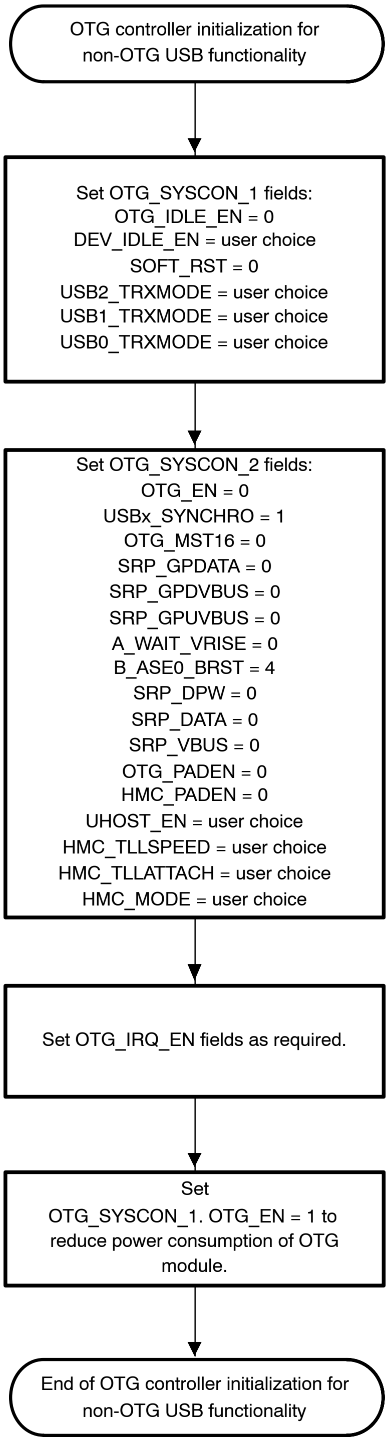
If the application implements an OTG DRD, follow the initialization process shown in Figure 22−73.

Figure 22−73. OTG Controller Initialization When Implementing an OTG DRD



Systems that implement USB functionality without implementing an OTG DRD should follow the initialization process shown in Figure 22−74.

Figure 22−74. OTG Controller Initialization When Not Implementing an OTG DRD


OTG controller initialization for non-OTG USB functionality
Set OTG_SYSCON_1 fields:
OTG_IDLE_EN = 0
DEV_IDLE_EN = user choice
SOFT_RST = 0
USB2_TRXMODE = user choice
USB1_TRXMODE = user choice
USB0_TRXMODE = user choice
Set OTG_SYSCON_2 fields:
OTG_EN = 0
USBx_SYNCHRO = 1
OTG_MST16 = 0
SRP_GPDATA = 0
SRP_GPDVBUS = 0
SRP_GPUVBUS = 0
A_WAIT_VRISE = 0
B_ASE0_BRST = 4
SRP_DPW = 0
SRP_DATA = 0
SRP_VBUS = 0
OTG_PADEN = 0
HMC_PADEN = 0
UHOST_EN = user choice
HMC_TLLSPEED = user choice
HMC_TLLATTACH = user choice
HMC_MODE = user choice
Set OTG_IRQ_EN fields as required.
Set OTG_SYSCON_1. OTG_EN = 1 to reduce power consumption of OTG module.
End of OTG controller initialization for non-OTG USB functionality

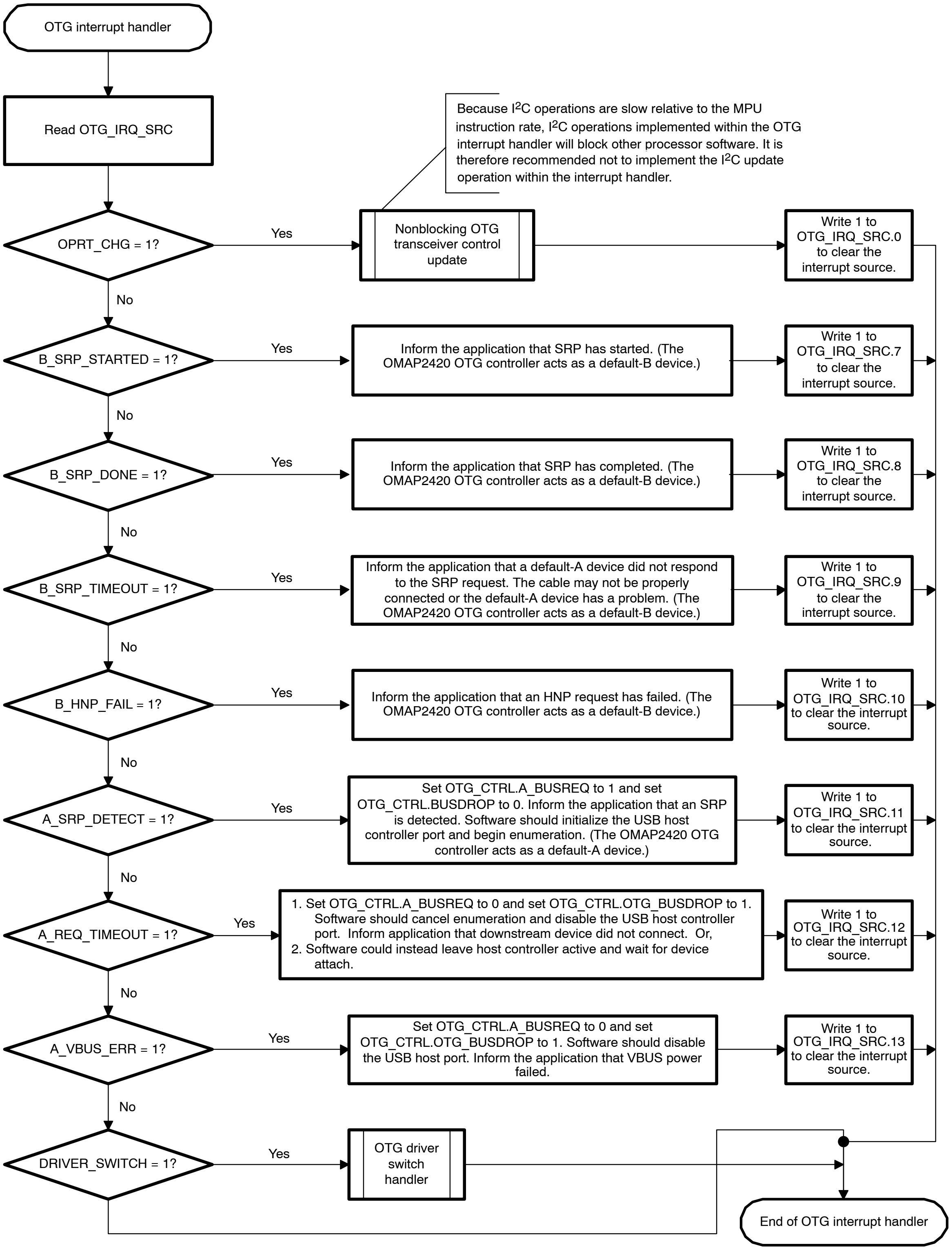
The user choice values in Figure 22−73 and Figure 22−74 depend on the system implementation, including the transceiver types used, top-level pin multiplexing choice, and connector type. Pin multiplexing options and transceiver types are discussed in Section 22.2, *USB Controller Environment*.

22.6.1.2 OTG Controller Interrupt Handler and Related Software

The OTG controller implements several interrupt sources that are separately enabled by bits in OTG_IRQ_EN. The OTG controller provides a single interrupt output (USB_IRQ_OTG) that provides OTG controller interrupts to the MPU subsystem interrupt controller M_IRQ_80 input.

Figure 22−75 shows the operations required in the USB OTG controller interrupt handler. This flowchart references several other operations, which are described in flowcharts later in this section.

Figure 22−75. OTG Interrupt Handler

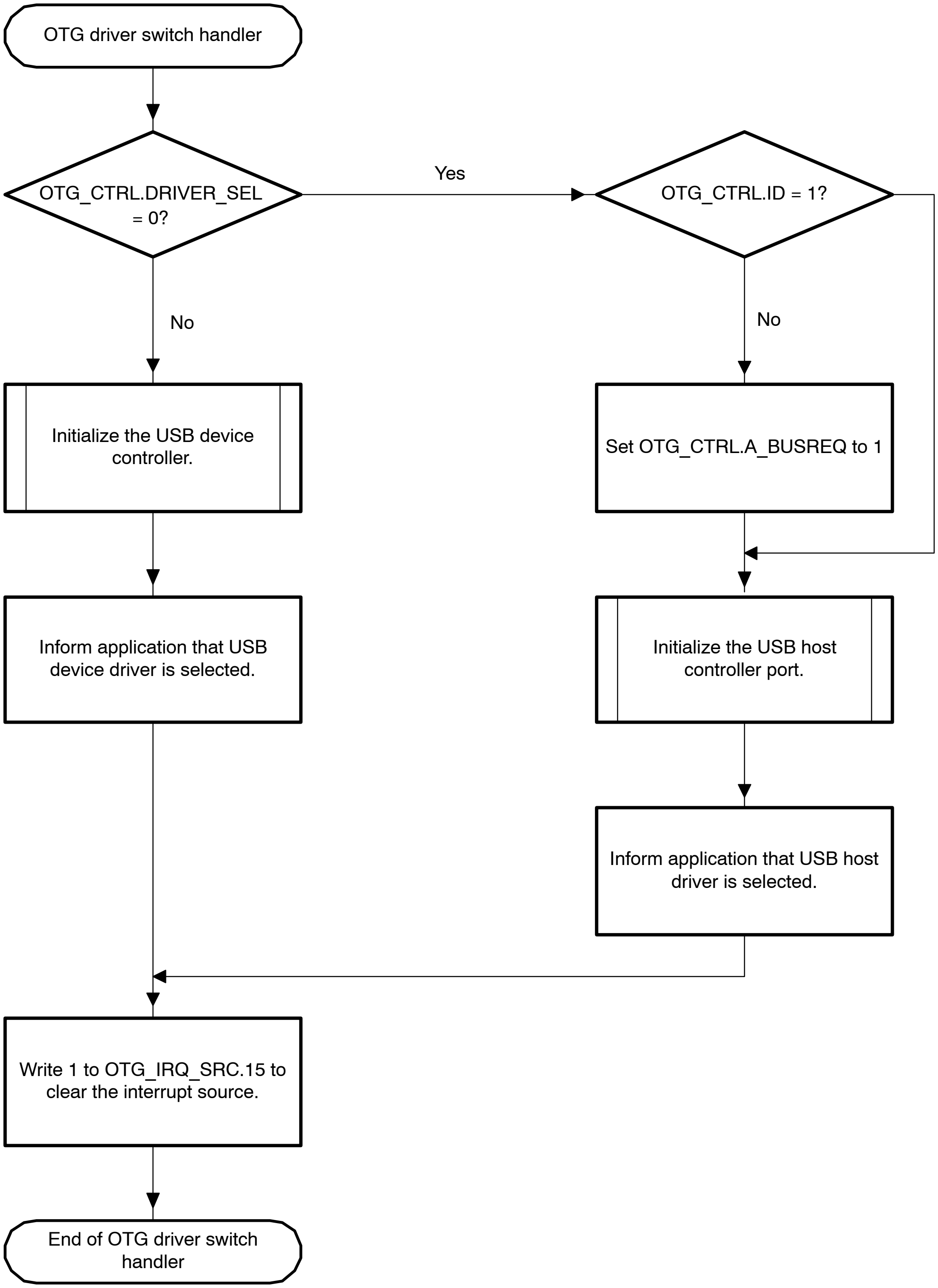


Because I²C operations are slow relative to the MPU instruction rate, I²C operations implemented within the OTG interrupt handler will block other processor software. It is therefore recommended not to implement the I²C update operation within the interrupt handler.

OTG interrupt handler
Read OTG_IRQ_SRC
OPRT_CHG = 1?
Yes
Nonblocking OTG transceiver control update
Write 1 to OTG_IRQ_SRC.0 to clear the interrupt source.
No
B_SRP_STARTED = 1?
Yes
Inform the application that SRP has started. (The OMAP2420 OTG controller acts as a default-B device.)
Write 1 to OTG_IRQ_SRC.7 to clear the interrupt source.
No
B_SRP_DONE = 1?
Yes
Inform the application that SRP has completed. (The OMAP2420 OTG controller acts as a default-B device.)
Write 1 to OTG_IRQ_SRC.8 to clear the interrupt source.
No
B_SRP_TIMEOUT = 1?
Yes
Inform the application that a default-A device did not respond to the SRP request. The cable may not be properly connected or the default-A device has a problem. (The OMAP2420 OTG controller acts as a default-B device.)
Write 1 to OTG_IRQ_SRC.9 to clear the interrupt source.
No
B_HNP_FAIL = 1?
Yes
Inform the application that an HNP request has failed. (The OMAP2420 OTG controller acts as a default-B device.)
Write 1 to OTG_IRQ_SRC.10 to clear the interrupt source.
No
A_SRP_DETECT = 1?
Yes
Set OTG_CTRL.A_BUSREQ to 1 and set OTG_CTRL.BUSDROP to 0. Inform the application that an SRP is detected. Software should initialize the USB host controller port and begin enumeration. (The OMAP2420 OTG controller acts as a default-A device.)
Write 1 to OTG_IRQ_SRC.11 to clear the interrupt source.
No
A_REQ_TIMEOUT = 1?
Yes
1. Set OTG_CTRL.A_BUSREQ to 0 and set OTG_CTRL.OTG_BUSDROP to 1. Software should cancel enumeration and disable the USB host controller port. Inform application that downstream device did not connect. Or,
2. Software could instead leave host controller active and wait for device attach.
Write 1 to OTG_IRQ_SRC.12 to clear the interrupt source.
No
A_VBUS_ERR = 1?
Yes
Set OTG_CTRL.A_BUSREQ to 0 and set OTG_CTRL.OTG_BUSDROP to 1. Software should disable the USB host port. Inform the application that VBUS power failed.
Write 1 to OTG_IRQ_SRC.13 to clear the interrupt source.
No
DRIVER_SWITCH = 1?
Yes
OTG driver switch handler
End of OTG interrupt handler

Figure 22−76 shows the OTG driver switch handler. It is responsible for switching control of the OTG link between the USB host controller and the USB device controller. This switch occurs mainly because of HNP operations, but it can also occur when the OTG_EN bit OTG_SYSCON_2[31] is set to 1.

Figure 22−76. OTG Driver Switch Handler


OTG driver switch handler
OTG_CTRL.DRIVER_SEL = 0?
Yes
OTG_CTRL.ID = 1?
No
No
Initialize the USB device controller.
Set OTG_CTRL.A_BUSREQ to 1
Yes
Inform application that USB device driver is selected.
Initialize the USB host controller port.
Inform application that USB host driver is selected.
Write 1 to OTG_IRQ_SRC.15 to clear the interrupt source.
End of OTG driver switch handler

The effects of OTG functionality on OMAP2420 USB device controller operation are documented in the USB device controller flowcharts. Because USB host controller operations are mainly defined in the OHCI specification, only some OTG functionality issues are discussed here.

Most OMAP2420 USB host controller software is not affected by USB OTG functionality. A host controller driver written for non-OTG functionality needs few modifications to support OTG functionality. In general, OTG events precede USB host controller driver operations, so the OTG event can cause the OTG driver software to pass the appropriate message to the USB host controller driver. It is never necessary for the USB host controller driver to access OTG controller registers.

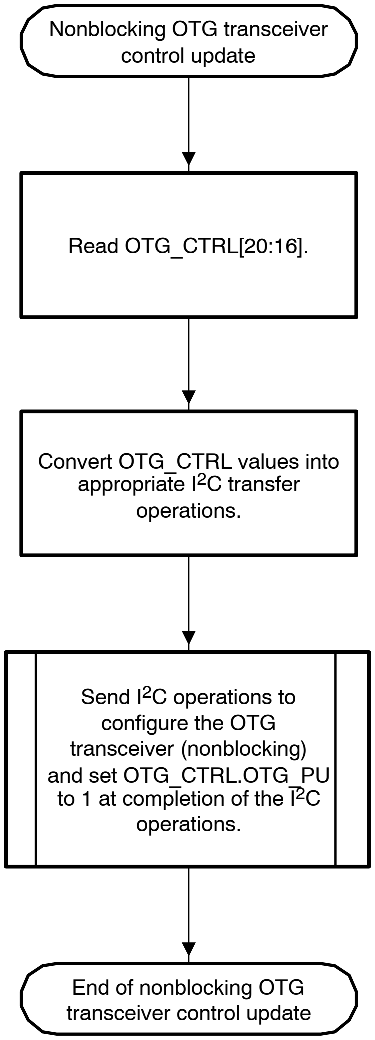
For an OTG connection, the SRP and HNP operations can be considered to replace the attach, detach, suspend, resume, and remote wake functions. This means that it can be beneficial for system software to control the USB host controller driver differently for an OTG link than for a typical USB host port.

The USB host controller driver must implement a mechanism where the application can specify that the OTG port is no longer needed. The host controller driver software must respond to this message by suspending the port associated with the OTG link and cancelling all EDs and TDs associated with the OTG connection. These ED and TD removal activities are usually associated with a disconnect event, so the host controller driver already has these functions available. The application typically sends this message and then informs the OTG controller driver that its use of the host is complete and that the OTG controller can now release the bus for a possible HNP.

When the USB device controller owns the USB OTG link, the USB host controller port does not have to remain active. If no other USB host controller ports are used, it is possible to disable the USB host controller clock for the duration while the USB device controller owns the OTG link. System software must reenable the USB host controller clock; if the host clock is dynamically disabled, the sofware must reinitialize the USB host on an OTG driver switch interrupt to USB host controller control of the OTG link.

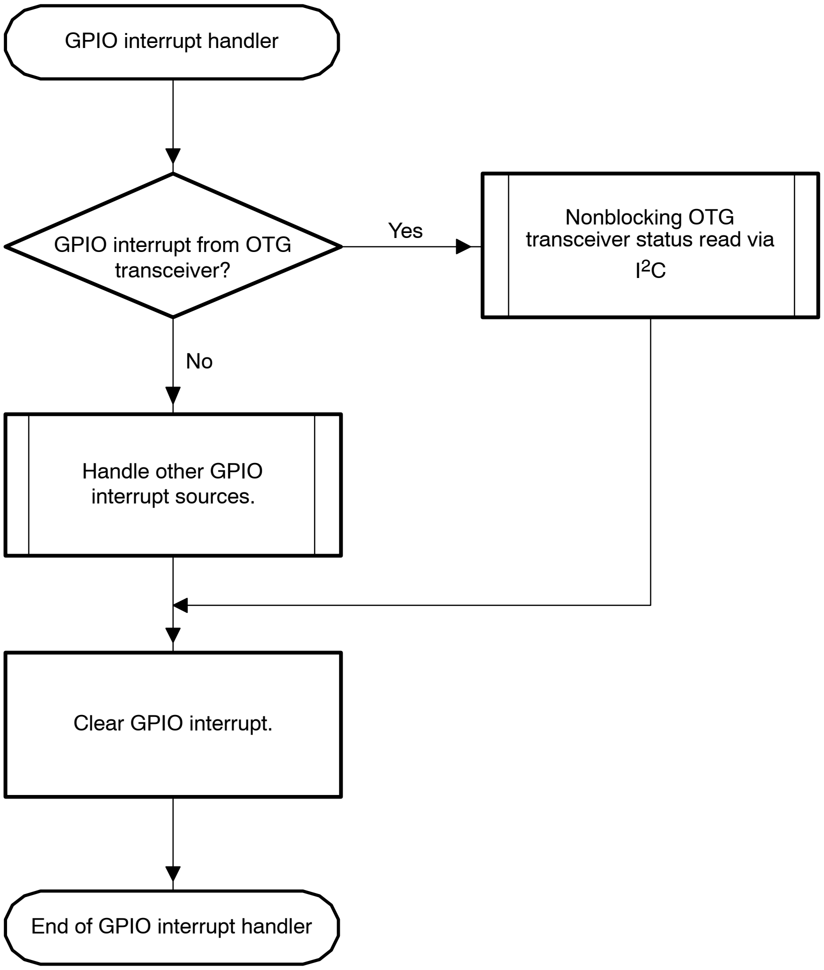
Software to support communication between the USB OTG controller and the external OTG transceiver using I^2C is described in Figure 22−77. This software must handle control information from the OTG controller to the external OTG. These software operations apply directly to the OTG controller OPRT_CHG interrupt. Because I^2C operations are slow (relative to the MPU instruction speed), it is best if these functions are not implemented as part of an interrupt handler; interrupt handlers block other processor software operations until the handler completes. System software must implement a non-blocking mechanism, such as a message-based system, to allow the I^2C operations to be completed outside of the OTG interrupt handler, but it must then allow updating of the OTG controller OTG_PU bit OTG_CTRL[4] when the I^2C operations complete.

Figure 22−77. OTG Transceiver I^{2}C Control Handler


Nonblocking OTG transceiver control update
Read OTG_CTRL[20:16].
Convert OTG_CTRL values into appropriate I2C transfer operations.
Send I2C operations to configure the OTG transceiver (nonblocking) and set OTG_CTRL.OTG_PU to 1 at completion of the I2C operations.
End of nonblocking OTG transceiver control update

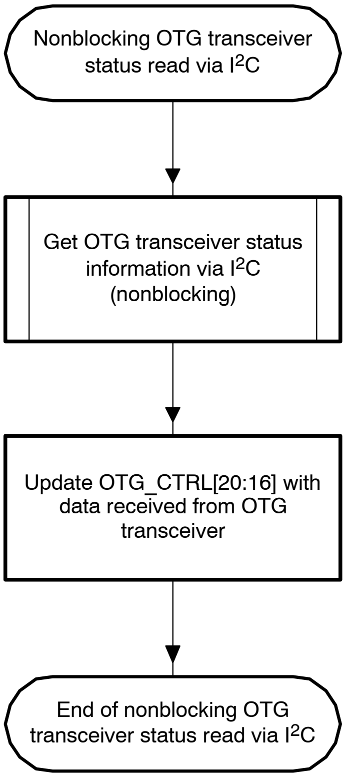
The OTG transceiver signals the need for a status transfer from the OTG transceiver to the OTG controller, using an interrupt output signal. This interrupt is generally connected to an OMAP2420 GPIO input pin, which is configured to provide an interrupt to the MPU subsystem interrupt controller. Figure 22−78 shows the basic GPIO interrupt handler functionality.

Figure 22−78. GPIO Interrupt Handler Support for OTG Transceiver Interrupt Input


GPIO interrupt handler
GPIO interrupt from OTG transceiver?
Yes
Nonblocking OTG transceiver status read via I2C
No
Handle other GPIO interrupt sources.
Clear GPIO interrupt.
End of GPIO interrupt handler

The operations in Figure 22−79 show the nonblocking OTG transceiver status read and update to the OTG controller registers. To reduce the effect on system performance, it is important that the I^{2}C read operations are nonblocking.

Figure 22−79. OTG Transceiver Status Read


Nonblocking OTG transceiver status read via I2C
Get OTG transceiver status information via I2C (nonblocking)
Update OTG_CTRL[20:16] with data received from OTG transceiver
End of nonblocking OTG transceiver status read via I2C

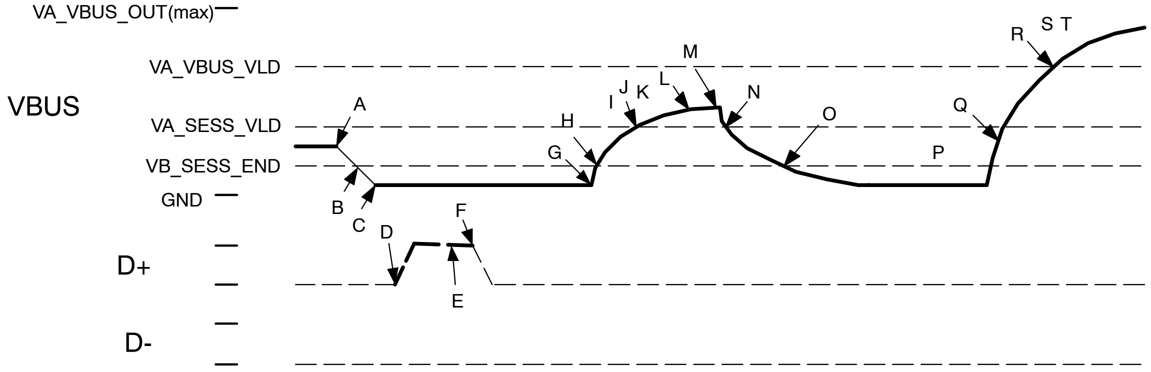
22.6.1.3 Typical SRP and HNP Events

This section describes the typical sequence of events for SRP and HNP events with the OMAP2420 OTG controller.

Careful system software design can optimize system performance by managing interrupt source enables at both the OTG controller and the OTG transceiver. System software developers must carefully weigh the advantages brought by disabling unneeded OTG transceiver interrupt sources versus the performance cost associated with the I^2C activity to enable and disable the transceiver interrupt sources.

Figure 22−80 shows the typical events that occur when the OMAP2420 device acts as an OTG default-A DRD and the default-B device issues an SRP. Figure 22−80 shows that the default-B device pulses its D+ pullup resistor. The OMAP2420 OTG controller accepts D– pullup pulses and the D+ pullup pulse shown as SRP data line.

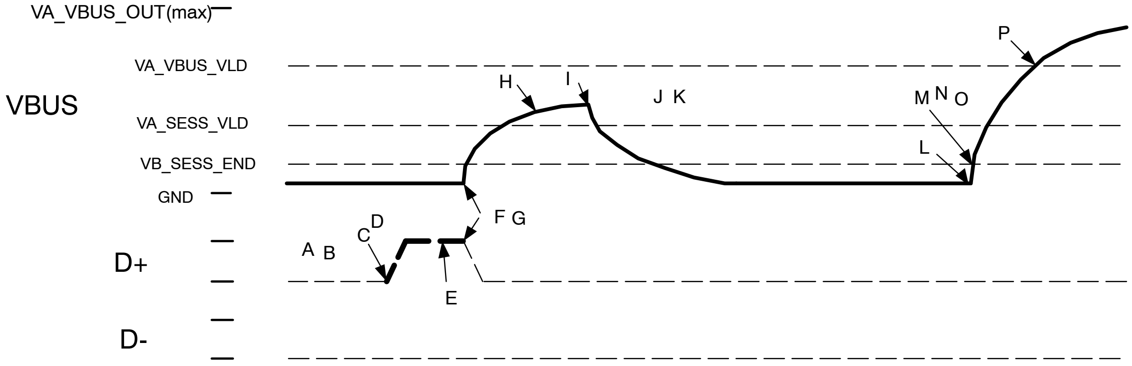
Figure 22−80. OMAP2420 OTG Controller Response to SRP When Acting as an OTG Default-A DRD



A. OTG default-B DRD begins discharging VBUS (optional).
B. VBUS at default-B device falls below VB_SESS_END.
C. OTG default-B DRD stops discharging VBUS (optional).
D. Default-B device enables SRP D+ pullup.
E. OMAP2420 OTG controller recognizes D+ pulse SRP request if D+ pulse is longer than the duration programmed in OTG_SYSCON_2.SRP_DPW and if OTG_SYSCON_2.SRP_DATA =1. If recognized, OTG_CTRL.OTG_DRV_VBUS is set to 1 and OPRT_CHG and SRP_DETECT interrupts are generated.
F. Default-B device disables D+ pullup (to end SRP data line pulse).
G. Initial conditions for SRP are satisfied. Default-B device enables VBUS SRP pulse.
H. VBUS voltage crosses VB_SESS_END at default-B.
I. VBUS voltage crosses VA_SESS_VLD at OMAP2420 OTG transceiver. OTG transceiver issues an interrupt.
J. OMAP2420 GPIO interrupt handler identifies OTG transceiver interrupt. Handler initiates I^{2}C operation to get transceiver interrupt source information.
K. OMAP2420 I^{2}C operations to get transceiver interrupt source information completes. System software sees VBUS > VA_SESS_VLD and sets OTG_CTRL.ASESSVLD to 1.
L. OMAP2420 OTG controller recognizes VBUS pulse SRP request if OTG_CTRL.ASESSVLD is 1 for longer than the duration programmed in OTG_SYSCON_2.SRP_DPW and if OTG_SYSCON_2.SRP_VBUS=1. If recognized, OTG_CTRL.OTG_DRV_VBUS is set to 1 and OPRT_CHG and SRP_DETECT interrupts are generated.
M. Default-B device stops driving VBUS pulse.
N. VBUS voltage discharges through various leakage sources.
O. VBUS voltage drops below VB_SESS_END at default-B device.
P. OMAP2420 OTG Interrupt handler sees OPRT_CHG interrupt and sees OTG_CTRL.OTG_DRV_VBUS set to 1 and initiates I^{2}C operations to configure OTG transceiver to drive VBUS. OTG interrupt handler also sees SRP_DETECT interrupt and begins initialization of OTG session as a default-A DRD.
Q. OMAP2420 I^{2}C operations to configure OTG transceiver to drive VBUS complete. System software writes 1 to OTG_CTRL.OTG_PU to indicate that the I^{2}C operation has completed.
R. VBUS rises above VA_VBUS_VLD. OTG transceiver issues interrupt.
S. OMAP2420 GPIO interrupt handler sees OTG transceiver interrupt and initiates I^{2}C operations to get OTG interrupt source information.
T. OMAP2420 I^{2}C operations complete. System software sees OTG status showing that VBUS voltage is above VA_SESS_VLD and sets OTG_CTRL.ASESSVLD to 1. OMAP2420 OTG DRD is now ready for default-B device to enable its pullup.

Figure 22−81 shows the typical events that occur when the OMAP2420 device acts as an OTG default-B DRD and issues an SRP to the default-A device. If the OMAP2420 device completes its SRP request but the default-A device does not drive VBUS within 5.5 seconds, the OMAP2420 OTG controller issues a B_SRP_TIMEOUT interrupt. System software must provide a message that the SRP request failed and that the cable and the A-device must be checked. The OMAP2420 OTG controller implements only the D+ data line SRP pulses and does not request a D– pulse as a data line SRP pulse.

Figure 22−81. OMAP2420 OTG Controller SRP Generation When Acting as an OTG Default-B DRD



A. OMAP2420 is acting as a default-B device. OTG_CTRL.BSESSEND = 1. Application decides that it wants to communicate with the default-B DRD. System software sets OTG_CTRL.B_BUSREQ.

B. OMAP2420 OTG controller sees write 1 to OTG_CTRL.B_BUSREQ and sees OTG_CTRL.BSESSEND = 1. OTG controller waits until USB bus has been SE0 for the minimum time and then begins SRP process by setting OTG_CTRL.OTG_PU and issues the OPRT_CHG interrupt.

C. OMAP2420 OTG interrupt handler sees OPRT_CHG interrupt and sees OTG_PU = 1. Handler initiates I^{2}C operation to enable OTG transceiver D+ pullup.

D. OMAP2420 I^{2}C operations to enable the D+ pullup complete. OMAP2420 software sets OTG_CTRL.OTG_PU to 1 to indicate that the D+ pullup enable operation is complete.

E. OMAP2420 OTG controller changes OTG_CTRL.OTG_PU to 0 approximately 9 ms after it sets it to 1. OTG controller also sets OTG_CTRL.OTG_VBUS_PU at the same time. OTG controller issues OPRT_CHG interrupt.

F. OMAP2420 OTG interrupt handler sees OPRT_CHG interrupt and sees OTG_PU = 0 and sees OTG_PU_VBUS = 1. Handler initiates I^{2}C operation to disable OTG transceiver D+ pullup and enable OTG transceiver VBUS pullup.

G. OMAP2420 I^{2}C operations to disable the D+ pullup and enable the VBUS pullup complete. OMAP2420 software sets OTG_CTRL.OTG_PU to 1 to indicate that the D+ pullup disable operation is complete.

H. OMAP2420 OTG controller changes OTG_CTRL.OTG_PU_VBUS to 0 after a time specified by OTG_SYSCON_2.SRP_GPUVBUS from the time the controller sets OTG_PU_VBUS to 1. If OTG_SYSCON_2.SRP_GPDVBUS = 1, OTG controller sets OTG_CTRL.OTG_PD_VBUS to 1. OTG controller issues OPRT_CHG interrupt.

I. OMAP2420 OTG interrupt handler sees the OPRT_CHG interrupt, OTG_CTRL.OTG_PU_VBUS = 0, and the current value in OTG_CTRL.OTG_PD_VBUS. Handler initiates I^{2}C operations to disable the OTG transceiver VBUS pullup and to enable or disable the OTG transceiver VBUS pulldown, depending on the value of OTG_PD_VBUS.

J. OMAP2420 OTG controller waits for the time specified by OTG_SYSCON_2.SRP_GPUVBUS from the time the controller had set OTG_PU_VBUS to 0. The OTG controller issues a B_SRP_DONE interrupt. If OTG_CTRL.OTG_PD_VBUS = 1, the controller sets OTG_PD_VBUS to 0 and issues an OPRT_CHG interrupt.

K. OMAP2420 OTG interrupt handler sees OTG interrupt. If OPRT_CHG interrupt is active, interrupt handler sees that OTG_CTRL.OTG_PD_VBUS is 0 and initiates I^{2}C operations to disable the OTG transceiver VBUS pulldown.

L. Some time later, the default-A device responds to the SRP request by driving VBUS active.

M. The OTG transceiver sees VBUS rise above VB_SESS_VLD and issues an interrupt.

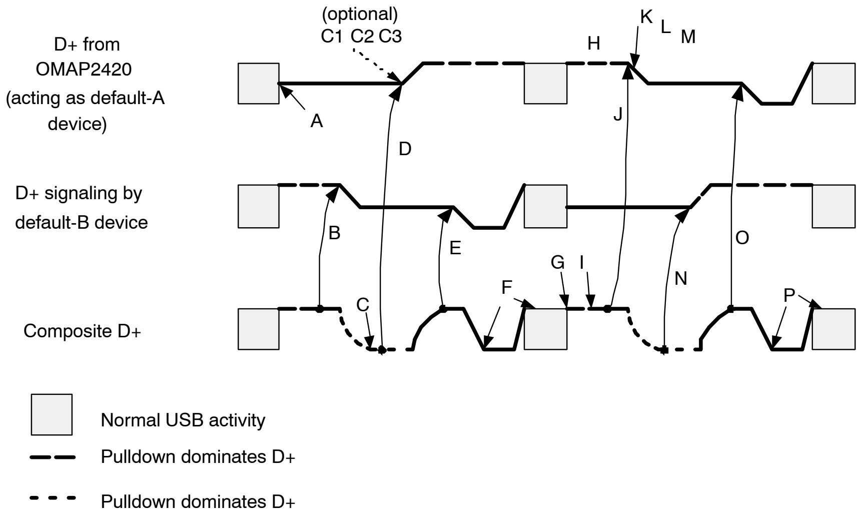
N. The OMAP2420 GPIO interrupt handler sees the OTG transceiver interrupt and initiates the I^{2}C operations, which query the OTG transceiver interrupt status.

O. The I^{2}C operation completes, and OMAP2420 system software updates OTG_CTRL.ASESSVLD, OTG_CTRL.BSESSEND, OTG_CTRL.BSESSVLD, OTG_CTRL.VBUSVLD, and OTG_CTRL.ID. If BSESSVLD is 1, then the OMAP2420 USB device controller sees VBUS go active and can begin preparations for enumeration.

P. The default-A device sees VBUS rise above VA_VBUS_VLD and can begin USB reset and enumerating the OMAP2420 default-B device.

Figure 22−82 shows the typical events that occur when the OMAP2420 device acts as an OTG default-A DRD and transitions from acting as an A-host to acting as an A-peripheral via HNP, and then transitions back to acting as an A-host via HNP. The figure shows the optional activities associated with the OTG transceiver autoconnect feature.

Figure 22−82. OMAP2420 OTG Controller HNP Events When Acting as an OTG . . . . Default-A DRD



A: OMAP2420 application has no more traffic for default-B device. System software suspends the USB host port used for the OTG link, and sets OTG_CTRL.A_BUSREQ to 0. If the OTG transceiver supports the autoconnect feature, system software should initiate the appropriate I^{2}C operations to enable it before suspending the OTG link.

B: Default-B device sees OTG link suspended by default-A device. Because the default-B device is enabled for HNP, and desires to issue HNP, it disables its D+ pullup, allowing D+ to float. Default-B controller waits for OMAP2420 OTG controller to enable its D+ pullup.

C: OMAP2420 OTG controller sees OTG_CTRL.A_SETB_HNPEN = 1 and sees SE0 on the bus. OTG controller sets OTG_CTRL.OTG_PU to 0 and issues both the OPRT_CHG interrupt and the driver switch interrupt.

C1. If the OTG transceiver autoconnect feature was enabled in step A, the OTG transceiver automatically enables its D+ pullup upon seeing SE0 on the bus, and issues an interrupt.

C2. OMAP2420 GPIO interrupt handler sees an OTG transceiver interrupt and issues I^{2}C operations to query the transceiver interrupt source.

C3. OMAP2420 I^{2}C operation completes and the interrupt status shows autoconnect occurred. It is not necessary to update any OTG controller registers when auto-HNP occurs.

D. OMAP2420 OTG interrupt handler sees OPRT_CHG interrupt and OTG_PU set to 1, and (if auto-HNP is not used) causes I^{2}C operation to enable OTG transceiver D+ pullup. OMAP2420 OTG interrupt handler sees driver switch interrupt and begins initialization of USB device controller.

E. Default-B device sees D+ pullup as HNP operation and begins acting as a USB host. It signals USB reset and enumerates the OMAP2420 DRD as a peripheral, and begins normal USB bus activity.

F. OMAP2420 USB device controller sees and responds appropriately to USB reset, enumeration, and normal USB bus activity.

G. Default-B device finishes its bus activity and suspends the USB link.

H. OMAP2420 application decides that it wants to communicate with the default-B device. System software sets OTG_CTRL.A_BUSREQ.

I. OMAP2420 OTG controller sees suspend signaled on the OTG link, sets OTG_CTRL.OTG_PU to 0, and issues OPRT_CHG interrupt.

J. OMAP2420 OTG interrupt handler sees the OPRT_CHG interrupt and OTG_CTRL.OTG_PU set to 0, and issues I^{2}C operations to disable the OTG transceiver D+ pullup.

K. OMAP2420 I^{2}C operations to disable the D+ pullup complete. OMAP2420 software sets OTG_CTRL.OTG_PU to 1 to indicate that the D+ pullup disable operation is complete.

L. OMAP2420 OTG controller sees write of 1 to OTG_CTRL.OTG_PU and issues driver switch interrupt.

M. OMAP2420 OTG interrupt handler sees driver switch interrupt, and begins initialization of host port.

N. Default-B device sees SE0 on the OTG link, and turns on its D+ pullup.

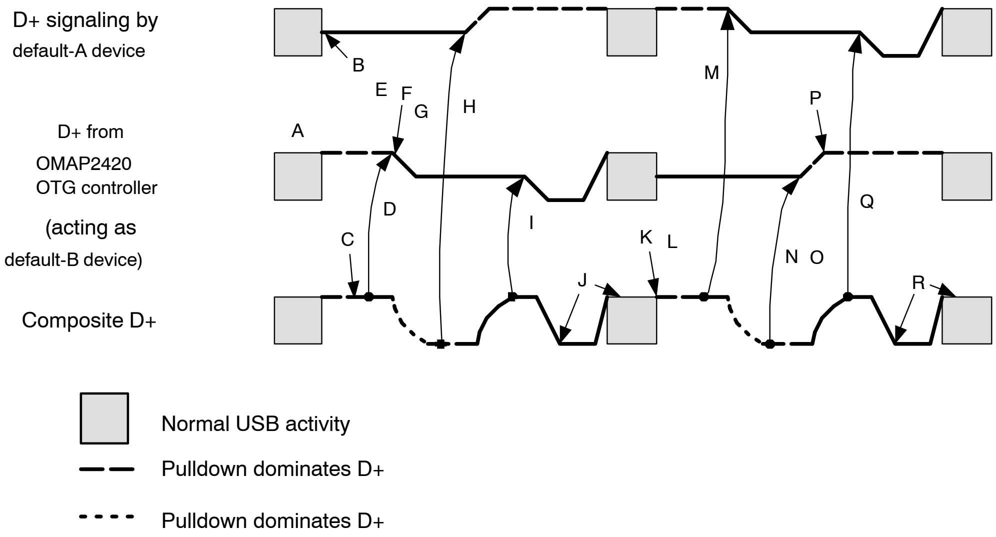
O. OMAP2420 OTG controller and USB host controller see D+ pullup as HNP operation. OMAP2420 USB host controller signals USB reset, sends the set feature signal with B_HNPEN feature enabled, enumerates the default-B DRD as a peripheral, and begins normal USB bus activity.

P. Default-B device sees and responds appropriately to USB reset, the set feature signal, enumeration, and normal USB bus activity.

Figure 22−83 shows the typical events that occur when the OMAP2420 device acts as an OTG default-B DRD and transitions from acting as a B-peripheral to acting as a B-host via HNP, and then transitions back to acting as a B-peripheral via HNP.

If the default-A device does not enable its pullup resistor within the required time, if the default-A device issues a USB resume during the HNP process, or if the VBUS voltage falls below the VB_SESS_VLD threshold during the HNP process, the OMAP2420 OTG controller issues a B_HNP_FAIL interrupt.

Figure 22−83. OMAP2420 OTG Controller HNP Events When Acting as an OTG Default-B DRD



A: OMAP2420 application decides to communicate with the default-A device. System software sets OTG_CTRL.B_BUSREQ.
B: Default-A device has no more traffic for the default-B device (OMAP2420 OTG controller). Default-A device suspends the OTG link.
C: OMAP2420 OTG controller (acting as the OTG default-B DRD) sees OTG link suspended, sees OTG_CTRL.B_HNPEN set to 1, sees OTG_CTRL.B_BUSREQ set to 1. OTG controller sets OTG_CTRL.OTG_PU to 0 and issues OPRT_CHG interrupt.
D. OMAP2420 OTG interrupt handler sees OPRT_CHG interrupt and sees OTG_PU = 0. Handler initiates I^{2}C operation to disable OTG transceiver D+ pullup.
E. OMAP2420 I^{2}C operations to disable the D+ pullup complete. OMAP2420 software sets OTG_CTRL.OTG_PU to 1 to indicate that the D+ pullup disable operation is complete.
F. OMAP2420 OTG controller sees write 1 to OTG_CTRL.OTG_PU and issues driver switch interrupt.
G. OMAP2420 OTG interrupt handler sees driver switch interrupt, and begins initialization of host port.
H. Default-A device sees SE0 on the OTG link, and turns on its D+ pullup.
I. OMAP2420 OTG controller sees D+ pullup as HNP operation, and OMAP2420 USB host controller sees D+ pullup as an attach. OMAP2420 USB host controller signals USB reset, enumerates the default-A DRD as a peripheral, and begins normal USB bus activity.
J. Default-A device sees and responds appropriately to USB reset, enumeration, and normal USB bus activity.
K. OMAP2420 application decides it has no more information to communicate to the default-A device. System software suspends the USB host port used for the OTG link, and sets OTG_CTRL.A_BUSREQ to 0.
L. OMAP2420 OTG controller sees OTG_CTRL.A_BUSREQ set to 0. OTG controller sets OTG_CTRL.OTG_PU to 1 and issues both OPRT_CHG and driver switch interrupts.
M. Default-A device sees OTG link signal USB Suspend. Default-A device disables its D+ pullup.
N. OMAP2420 OTG interrupt handler sees OPRT_CHG interrupt, sees OTG_CTRL.OTG_PU set to 1, and initiates I^{2}C operations to enable OTG transceiver D+ pullup.
O. OMAP2420 OTG interrupt handler sees driver switch and begins initialization of OMAP2420 USB device controller.
P. OMAP2420 I^{2}C operations to enable OTG transceiver D+ pullup complete.
Q. Default-A device sees D+ pullup, signals USB reset, issues Set Feature of the B_HNP_ENABLE feature, enumerates the OMAP2420 default-B DRD as a peripheral, and begins normal USB bus activity.
R. OMAP2420 USB device controller sees and responds appropriately to USB reset, set feature of the B_HNP_ENABLE feature, enumeration, and normal USB bus activity.

22.6.1.4 System-Level OTG Considerations

The OMAP2420 USB OTG implementation allows up to two separate USB host controller ports to function in parallel with an OTG link. The OTG specification supplement states that an OTG DRD has only one USB connector. If a system implements an OTG connector, the system can only use the other two USB host controller ports for on-board connectivity.

An OTG implementation using an OTG transceiver that is based on the OTG transceiver specification can use the transceiver autoconnect feature. Enabling this OTG transceiver feature when the OMAP2420 device is acting as an OTG default-A DRD eases the timing requirements on system software during an HNP transition from the OMAP2420 OTG controller acting as an A-host to acting as an A-peripheral.

A system that implements an OTG controller has a USB mini-AB receptacle. This receptacle is used with an OTG cable to connect the system to another OTG system. A non-OTG USB device can be connected to the OMAP2420-based OTG system mini-AB receptacle using an adapter cable as defined in the OTG specification supplement. If the system supports this option, it may be necessary to provide a VBUS source with larger current source capability than is typically available through an OTG transceiver.

22.7 USB Registers

This section provides a summary of the hardware interface for the USB functionality on the OMAP2420 processor.

Table 22−30 lists the base address and address space for the USB module.

Table 22−31 lists the USB module registers.

Table 22−32 through Table 22−97 describe the individual register bits.

Table 22−30. Instance Summary

Module Name	Base Address	Size
USBOT1	0x4805 E000	4K bytes

CAUTION

USB host controller (HC) registers (0x000 to 0x0EC offsets) are limited to 32-bit and 16-bit data access; 8-bit data access generates an error. See Section 22.7.1 for more information.

USB device controller registers (0x200 to 0x2FC offsets) are not limited to 32-bit data access (16-bit or 8-bit data can be used) as long as the address of the access is aligned on a 32-bit boundary.

Table 22−31. USB Module Registers

Register Name	Type	Register Width (Bits)	Offset
HCREVISION	R	32	0x000
HCCONTROL	RW	32	0x004
HCCOMMANDSTATUS	RW	32	0x008
HCINTERRUPTSTATUS	RW	32	0x00C
HCINTERRUPTENABLE	RW	32	0x010
HCINTERRUPTDISABLE	RW	32	0x014
HCHCCA	RW	32	0x018
HCPERIODCURRENTED	R	32	0x01C
HCCONTROLHEADED	RW	32	0x020
HCCONTROLCURRENTED	RW	32	0x024
HCBULKHEADED	RW	32	0x028
HCBULKCURRENTED	RW	32	0x02C
HCDONEHEAD	R	32	0x030
HCFMINTERVAL	RW	32	0x034
HCFMREMAINING	R	32	0x038
HCFMNUMBER	R	32	0x03C
HCPERIODICSTART	RW	32	0x040
HCLSTHRESHOLD	RW	32	0x044
HCRHDESCRIPTORA	RW	32	0x048

Table 22−31. USB Module Registers (Continued)

Register Name	**Type**	**Register Width (Bits)**	**Offset**
HCRHDESCRIPTORB	RW	32	0x04C
HCRHSTATUS	RW	32	0x050
HCRHPORTSTATUS_1	RW	32	0x054
HCRHPORTSTATUS_2	RW	32	0x058
HCRHPORTSTATUS_3	RW	32	0x05C
HOSTUEADDR	R	32	0x0E0
HOSTUESTATUS	R	32	0x0E4
HOSTTIMEOUTCTRL	RW	32	0x0E8
HOSTREVISION	R	32	0x0EC
WHM_REVID_REGISTER	R	32	0x0F4
WHM_TEST_OBSV	R	32	0x0F8
WHM_TEST_CTL	RW	32	0x0FC
HHC_TEST_CFG	RW	32	0x100
HHC_TEST_CTL	RW	32	0x104
HHC_TEST_OBSV	R	32	0x108
REVDEV	R	16	0x200
EP_NUM	RW	16	0x204
DATA	RW	16	0x208
CTRL	RW	16	0x20C
STAT_FLG	R	16	0x210
RXFSTAT	R	16	0x214
SYSCON1	RW	16	0x218
SYSCON2	RW	16	0x21C
DEVSTAT	R	16	0x220
SOFREG	R	16	0x224
IRQ_EN	RW	16	0x228
DMA_IRQ_EN	RW	16	0x22C
IRQ_SRC	RW	16	0x230
EPN_STAT	R	16	0x234
DMAN_STAT	R	16	0x238
RXDMA_CFG	RW	16	0x240
TXDMA_CFG	RW	16	0x244
DATA_DMA	RW	16	0x248
TXDMA0–TXDMA2	RW	16	0x250-0x258
RXDMA0–RXDMA2	RW	16	0x260-0x268

Table 22–31. USB Module Registers (Continued)

Register Name	Type	Register Width (Bits)	Offset
EP0	RW	16	0x280
EP_RX1–EP_RX15	RW	16	0x284- 0x2BC
EP_TX1–EP_TX15	RW	16	0x2C4- 0x2FC
OTG_REV	R	32	0x300
OTG_SYSCON_1	RW	32	0x304
OTG_SYSCON_2	RW	32	0x308
OTG_CTRL	RW	32	0x30C
OTG_IRQ_EN	RW	16	0x310
OTG_IRQ_SRC	RW	16	0x314
OTG_OUTCTRL	RW	16	0x318
OTG_TEST	RW	16	0x320
OTG_VC	R	32	0x3FC

22.7.1 Register Descriptions

Table 22–32. HCREVISION

Address Offset	0x000		
Physical Address	0x4805 E000	**Instance**	USBOT1
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	0x------
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	

Table 22–33. HCCONTROL

Address Offset	0x004		
Physical Address	0x4805 E004	**Instance**	USBOT1
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	

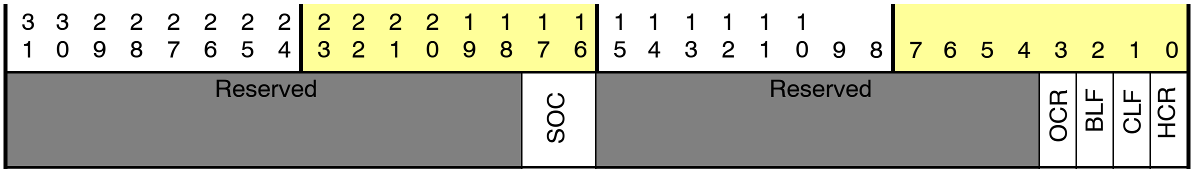
Bits	Field Name	Description	Type	Reset
31:11	Reserved	Reserved	RW	0x------
10	RWE	Remote wake-up enable	RW	0
9	RWC	Remote wake-up connected	RW	0
8	IR	Interrupt routing	RW	0
7:6	HCFS	Host controller functional state 0x0: HCFS: USB reset 0x1: HCFS: USB resume 0x2: HCFS: USB operational 0x3: HCFS: USB suspend	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	0
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	0
3	IE	Isochronous ED processing enabled by host controller driver. 0x0: Isochronous EDs are not processed. 0x1: Enables processing of isochronous EDs	RW	0
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	0

Table 22–33. HCCONTROL (Continued)

Bits	Field Name	Description	Type	Reset
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

Table 22–34. HCCOMMANDSTATUS

Address Offset	0x008		
Physical Address	0x4805 E008	**Instance**	USBOT1
Description	HC command and status		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:18	Reserved	Reserved	RW	0x----
17:16	SOC	Scheduling overrun count	R	0x0
15:4	Reserved	Reserved	RW	0x---
3	OCR	Ownership change request. The OMAP2420 USB host controller does not implement ownership change interrupts.	RW	0
2	BLF	Bulk list filled	RW	0
1	CLF	Control list filled	RW	0
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	0

Table 22–35. HCINTERRUPTSTATUS

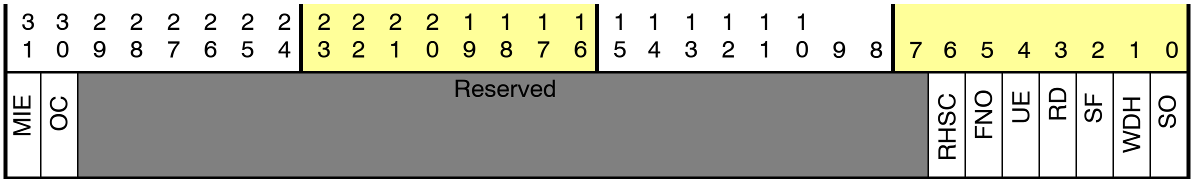
Address Offset	0x00C		
Physical Address	0x4805 E00C	**Instance**	USBOT1
Description	HC interrupt status		
Type	RW		

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	RW	-
30	OC	Ownership change The OMAP2420 USB host controller does not implement ownership change interrupts.	RW	0
29:7	Reserved	Reserved	RW	0x------
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	0
5	FNO	Frame number overflow. When 0x1: A frame number overflow has occurred. Write 0x0: No effect Write 0x1: Clears this bit	RW	0
4	UE	Unrecoverable error. When 0x1: An unrecoverable error has occurred. Write 0x0: No effect Write 0x1: Clears this bit	RW	0
3	RD	Resume detected. When 0x1: A downstream device has issued a resume request. Write 0x0: No effect Write 0x1: Clears this bit	RW	0
2	SF	Start of frame. When 0x1:A SOF is issued. Write 0x0: No effect Write 0x1:Clears this bit	RW	0
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	0
0	SO	Scheduling overrun. When 0x1: A scheduling overrun has occurred. Write 0x0: No effect Write 0x1: Clears this bit	RW	0

Table 22–36. HCINTERRUPTENABLE

Address Offset	0x010		
Physical address	0x4805 E010	**Instance**	USBOT1
Description	HC interrupt enable		
Type	RW		



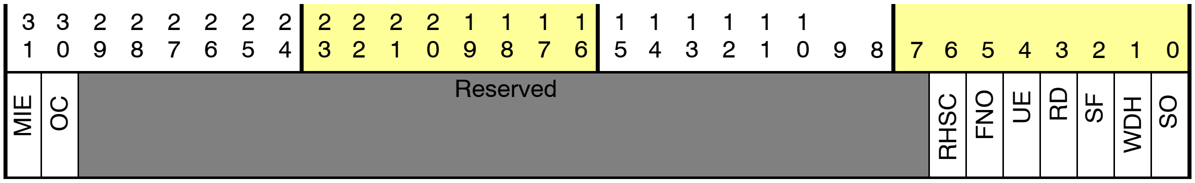
Bits	Field Name	Description	Type	Reset
31	MIE	Master interrupt enable. When 0x1: Allows other enabled OHCI interrupt sources to propagate to the device interrupt controller When 0x0: OHCI interrupt sources are ignored. Write 0x0: No effect Write 0x1: Sets this bit	RW	0
30	OC	Ownership change This bit has no effect on OMAP2420.	RW	0
29:7	Reserved	Reserved	RW	0x------
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	0
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	0
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	0
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	0

Table 22–36. HCINTERRUPTENABLE (Continued)

Bits	Field Name	Description	Type	Reset
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	0
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	0
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	0

Table 22–37. HCINTERRUPTDISABLE

Address Offset	0x014		
Physical Address	0x4805 E014	**Instance**	USBOT1
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	0
30	OC	Ownership change This bit has no effect on OMAP2420.	RW	0
29:7	Reserved	Reserved	RW	0x------
6	RHSC	Root hub status change. Always reads 0x0. Write 0x0: No effect Write 0x1: Clears the HCINTERRUPTENABLE RHSC bit	RW	0

Table 22–37. HCINTERRUPTDISABLE (Continued)

Bits	Field Name	Description	Type	Reset
5	FNO	Frame number overflow. Always reads 0x0. Write 0x0: No effect Write 0x1: Clears the HCINTERRUPTENABLE FNO bit	RW	0
4	UE	Unrecoverable error. Always reads 0x0. Write 0x0: No effect Write 0x1: Clears the HCINTERRUPTENABLE UE bit	RW	0
3	RD	Resume detected. Always reads 0x0. Write 0x0: No effect Write 0x1: Clears the HCINTERRUPTENABLE RD bit	RW	0
2	SF	Start of frame. Always reads 0x0. Write 0x0: No effect Write 0x1: Clears the HCINTERRUPTENABLE SF bit	RW	0
1	WDH	Write done head. Always reads 0x0. Write 0x0: No effect Write 0x1: Clears the HCINTERRUPTENABLE WDH bit	RW	0
0	SO	Scheduling overrun. Always reads 0x0. Write 0x0: No effect Write 0x1: Clears the HCINTERRUPTENABLE SO bit	RW	0

Table 22–38. HCHCCA

Address Offset	0x018		
Physical Address	0x4805 E018	**Instance**	USBOT1
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	RW	0x--

Table 22–39. HCPERIODCURRENTED

Address Offset	0x01C		
Physical Address	0x4805 E01C	**Instance**	USBOT1
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	0x-

Table 22–40. HCCONTROLHEADED

Address Offset	0x020		
Physical Address	0x4805 E020	**Instance**	USBOT1
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	RW	0x-

Table 22–41. HCCONTROLCURRENTED

Address Offset	0x024		
Physical Address	0x4805 E024	**Instance**	USBOT1
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	RW	0x-

Table 22−42. HCBULKHEADED

Address Offset	0x028		
Physical Address	0x4805 E028	**Instance**	USBOT1
Description	HC head bulk register		
Type	RW		

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1										
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	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	RW	0x-

Table 22−43. HCBULKCURRENTED

Address Offset	0x02C		
Physical Address	0x4805 E02C	**Instance**	USBOT1
Description	HC current bulk register		
Type	RW		

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1										
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	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	RW	0x-

Table 22−44. HCDONEHEAD

Address Offset	0x030		
Physical Address	0x4805 E030	**Instance**	USBOT1
Description	HC head done register		
Type	R		

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0
1	0	9	8	7	6	5	4	3	2	1	0	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	0x-

Table 22–45. HCFMINTERVAL

Address Offset	0x034		
Physical Address	0x4805 E034	**Instance**	USBOT1
Description	HC frame interval register		
Type	RW		

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>RT</td><td colspan="15">FSMPS</td><td colspan="2">Reserved</td><td colspan="14">FI</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
31	FIT	Frame interval toggle	RW	0
30:16	FSMPS	Largest data packet size for full-speed packets, bit times	RW	0x----
15:14	Reserved	Reserved	RW	0x-
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

Table 22–46. HCFMREMAINING

Address Offset	0x038		
Physical Address	0x4805 E038	**Instance**	USBOT1
Description	HC frame remaining register		
Type	R		

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>FRT</td><td colspan="17">Reserved</td><td colspan="14">FR</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
31	FRT	Frame remaining toggle	R	0
30:14	Reserved	Reserved	R	0x-----
13:0	FR	Frame remaining	R	0x0000

Table 22−47. HCFMNUMBER

Address Offset	0x03C		
Physical Address	0x4805 E03C	**Instance**	USBOT1
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	0x----
15:0	FN	Frame number	R	0x0000

Table 22−48. HCPERIODICSTART

Address Offset	0x040		
Physical Address	0x4805 E040	**Instance**	USBOT1
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	RW	0x-----
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

Table 22−49. HCLSTHRESHOLD

Address Offset	0x044		
Physical Address	0x4805 E044	**Instance**	USBOT1
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	0x-----
11:0	LST	Low-speed threshold	RW	0x628

Table 22–50. HCRHDESCRIPTORA

Address Offset	0x048		
Physical Address	0x4805 E048	**Instance**	USBOT1
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											NOCP	OCPM	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
23:13	Reserved	Reserved	RW	0x---
12	NOCP	No overcurrent protection. 0x0: Overcurrent status is reported collectively for all downstream ports. 0x1: The USB host controller does not implement overcurrent protection inputs.	RW	1
11	OCPM	Overcurrent protection mode	RW	0
10	DT	Device type. Always reads 0x0: Indicates that the USB host controller implemented is not a compound device.	R	0
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	1
8	PSM	Power switching mode 0x0: Indicates that all ports are powered at the same time 0x1: Individual-port power switching is supported	RW	0
7:0	NDP	Number of downstream ports	R	0x03

Table 22–51. HCRHDESCRIPTORB

Address Offset	0x04C		
Physical Address	0x4805 E04C	**Instance**	USBOT1
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 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

Table 22–52. HCRHSTATUS

Address Offset	0x050		
Physical Address	0x4805 E050	**Instance**	USBOT1
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													OCIC	LPSC	DRWE	Reserved													OCI	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	-
30:18	Reserved	Reserved	RW	0x----
17	OCIC	Overcurrent indication change. This bit is automatically set when the overcurrent indicator bit changes. Write 0x0: No effect Write 0x1: Clears this bit	RW	0

Table 22–52. HCRHSTATUS (Continued)

Bits	Field Name	Description	Type	Reset
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	0
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	0
14:2	Reserved	Reserved	RW	0x----
1	OCI	Overcurrent indicator. Reports global overcurrent indication if global overcurrent reporting is selected. If per-port overcurrent protection is implemented, this bit is always 0. 0x0: All power operations are normal. 0x1: An overcurrent condition exists.	R	0
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	0

Table 22−53. HCRHPORTSTATUS_1

Address Offset	0x054		
Physical Address	0x4805 E054	**Instance**	USBOT1
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	OCIC	PSSC	PESC	CSC	Reserved						LSDA_CPP	PPS_SPP	Reserved			PRS_SPR	POCI_CSS	PSS_SPS	PES_SPE	CCS_CPE

Bits	Field Name	Description	Type	Reset
31:21	Reserved	Reserved	RW	0x---
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	0
19	OCIC	Port 1 overcurrent indicator change. This bit is set when the Port 1 port overcurrent indicator has changed. Write 0x0: No effect Write 0x1: Clears this bit	RW	0
18	PSSC	Port 1 suspend status change. This bit is set when the Port 1 port suspend status has changed. Write 0x0: No effect Write 0x1: Clears this bit	RW	0
17	PESC	Port 1 enable status change. This bit is set when the Port 1 port enable status has changed. Write 0x0: No effect Write 0x1: Clears this bit	RW	0
16	CSC	Port 1 connect status change. This bit is set when the Port 1 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	0
15:10	Reserved	Reserved	RW	0x--

Table 22−53. HCRHPORTSTATUS_1 (Continued)

Bits	Field Name	Description	Type	Reset
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	-
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	1
7:5	Reserved	Reserved	RW	0x-
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	0
3	POCI_CSS	Port 1 port overcurrent indicator/clear suspend status Read 0x0: No port 1 port overcurrent condition has occurred. Read 0x1: A port 1 port overcurrent condition has occurred. Write 0x0: No effect Write 0x1: When port 1 port suspend status is 1, causes resume signaling on port 1. When port 1 port suspend status is 0, has no effect.	RW	0
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	0

Table 22−53. HCRHPORTSTATUS_1 (Continued)

Bits	Field Name	Description	Type	Reset
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	0
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	0

Table 22−54. HCRHPORTSTATUS_2

Address Offset	0x058		
Physical Address	0x4805 E058	**Instance**	USBOT1
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	OCIC	PSSC	PESC	CSC	Reserved						LSDA_CPP	PPS_SPP	RESERVED			PRS_SPR	POCI_CSS	PSS_SPS	PES_SPE	CCS_CPE

Bits	Field Name	Description	Type	Reset
31:21	Reserved	Reserved	RW	0x---
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	0
19	OCIC	Port 2 overcurrent indicator change. This bit is set when the port 2 port overcurrent indicator has changed. Write 0x0: No effect Write 0x1: Clears this bit	RW	0

Table 22−54. HCRHPORTSTATUS_2 (Continued)

Bits	Field Name	Description	Type	Reset
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	0
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	0
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	0
15:10	Reserved	Reserved	RW	0x--
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	-
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	1
7:5	Reserved	Reserved	RW	0x-
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	0
3	POCI_CSS	Port 2 port overcurrent indicator/clear suspend status. Read 0x0: No port 2 port overcurrent condition has occurred. Read 0x1: A port 2 port overcurrent condition has occurred. Write 0x0: No effect Write 0x1: When port 2 port suspend status is 1, causes resume signaling on port 2. When port 2 port suspend status is 0, has no effect.	RW	0

Table 22−54. HCRHPORTSTATUS_2 (Continued)

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	0
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	0
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	0

Table 22−55. HCRHPORTSTATUS_3

Address Offset	0x05C		
Physical Address	0x4805 E05C	**Instance**	USBOT1
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	OCIC	PSSC	PESC	CSC	Reserved						LSDA_CPP	PPS_SPP	RESERVED			PRS_SPR	POCI_CSS	PSS_SPS	PES_SPE	CCS_CPE

Table 22−55. HCRHPORTSTATUS_3 (Continued)

Bits	Field Name	Description	Type	Reset
31:21	Reserved	Reserved	RW	0x---
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	0
19	OCIC	Port 3 overcurrent indicator change. This bit is set when the port 3 port overcurrent indicator has changed. Write 0x0 No effect Write 0x1: Clears this bit	RW	0
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	0
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	0
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	0
15:10	Reserved	Reserved	RW	0x--
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	-
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	1
7:5	Reserved	Reserved	RW	0x-
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	0

Table 22−55. HCRHPORTSTATUS_3 (Continued)

Bits	Field Name	Description	Type	Reset
3	POCI_CSS	Port 3 overcurrent indicator/clear suspend status. Read 0x0: No port 3 port overcurrent condition has occurred. Read 0x1: A port 3 port overcurrent condition has occurred. Write 0x0: No effect Write 0x1: When port 3 port suspend status is 1, causes resume signaling on port 3. When port 3 port suspend status is 0, has no effect.	RW	0
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	0
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	0
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	0

Table 22–56. HOSTUEADDR

Address Offset	0x0E0		
Physical Address	0x4805 E0E0	**Instance**	USBOT1
Description	Host UE address 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
UE_ADDR																															

Bits	Field Name	Description	Type	Reset
31:0	UE_ADDR	Unrecoverable error address. This register captures the physical address of any OCPI bus operation that is started by the USB host controller that encounters an unrecoverable error condition.	R	0x00000000

Table 22–57. HOSTUESTATUS

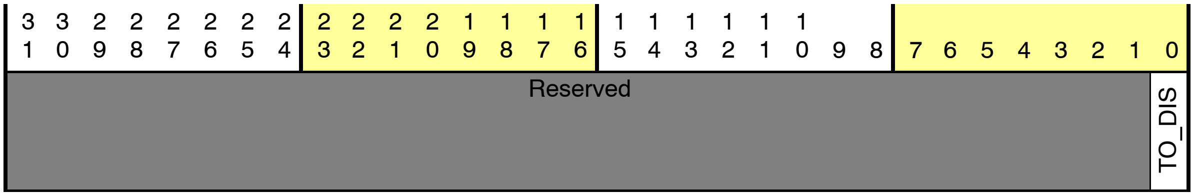
Address Offset	0x0E4		
Physical Address	0x4805 E0E4	**Instance**	USBOT1
Description	Host UE status register		
Type	R		

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																															UEACCESS

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Reserved	R	0x--------
0	UEACCESS	Access type when unrecoverable error occurred. This bit has no meaning before an unrecoverable error occurs. 0x0: An unrecoverable error occurs because of time-out of a OCPI bus read. 0x1: An unrecoverable error occurs because of time-out of a OCPI bus write.	R	0

Table 22−58. HOSTTIMEOUTCTRL

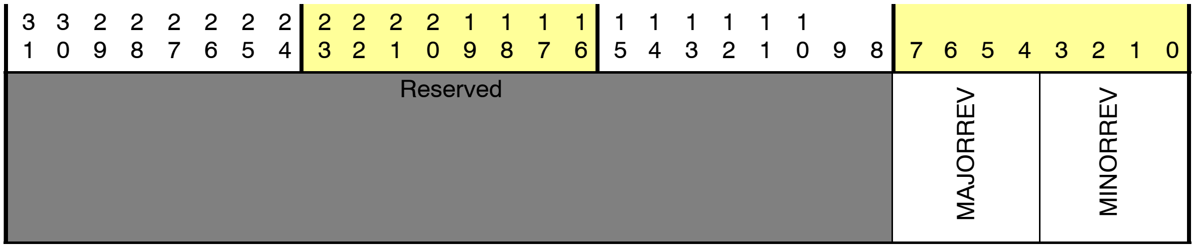
Address Offset	0x0E8		
Physical Address	0x4805 E0E8	**Instance**	USBOT1
Description	Host time-out control register		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:1	Reserved	Reserved	RW	0x--------
0	TO_DIS	OCPI bus time-out disable 0x0: The USB host controller waits indefinitely to access system memory. When cleared (the default state), the USB host controller waits no more than 4096 OCPI bus clocks for completion of an OCPI bus access to system memory. If the OCPI bus cycle does not complete in that time, the USB host controller signals an unrecoverable error 0x1: The USB host controller OCPI bus time-out counter is disabled and the host controller waits indefinitely for completion of a USB host controller access to system memory.	RW	0

Table 22−59. HOSTREVISION

Address Offset	0x0EC		
Physical Address	0x4805 E0EC	**Instance**	USBOT1
Description	Host revision register		
Type	R		



Bits	Field Name	Description	Type	Reset
31:8	Reserved	Reserved	R	0x------
7:4	MAJORREV	Major revision number of the USB host controller	R	
3:0	MINORREV	Minor revision number of the USB host controller	R	

Table 22−60. WHM_REVID_REGISTER

Address Offset	0x0F4		
Physical Address	0x4805 E0F4	**Instance**	USBOT1
Description	The WHM RevID register provides an indication of the USB WHM subchip 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																								MAJORREV				MINORREV			

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Reserved	R	0x------
7:4	MAJOR_REVISION	Major revision	R	0x-
3:0	MINOR_REVISION	Minor revision	R	0x-

Table 22−61. WHM_TEST_OBSV

Address Offset	0x0F8		
Physical Address	0x4805 E0F8	**Instance**	USBOT1
Description	Provides S/W visibility of certain signals		
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																					UART_CTS_O	UART_RX_O	PRT_RCVDPLS_2	PRT_RCVDPLS_1	PRT_RCVDPLS_0	PRT_RCVDMNS_2	PRT_RCVDMNS_1	PRT_RCVDMNS_0	PRT_RCVDATA_2	PRT_RCVDATA_1	PRT_RCVDATA_0

Bits	Field Name	Description	Type	Reset
31:11	Reserved	Reserved	R	0x------
10	UART_CTS_O	Current value of UART_CTS_O when WHM_TEST_CTL.TEST_UNLOCK = 1; else, returns 0	R	0
9	UART_RX_O	Current value of UART_RX_O when WHM_TEST_CTL.TEST_UNLOCK = 1; else, returns 0	R	0
8	PRT_RCVDPLS_2	Current value of PRT_RCVDPLS(2) when WHM_TEST_CTL.TEST_UNLOCK = 1; else, returns 0	R	0
7	PRT_RCVDPLS_1	Current value of PRT_RCVDPLS(1) when WHM_TEST_CTL.TEST_UNLOCK = 1; else, returns 0	R	0
6	PRT_RCVDPLS_0	Current value of PRT_RCVDPLS(0) when WHM_TEST_CTL.TEST_UNLOCK = 1; else, returns 0	R	0

Table 22−61. WHM_TEST_OBSV (Continued)

Bits	Field Name	Description	Type	Reset
5	PRT_RCVDMNS_2	Current value of PRT_RCVDMNS(2) when WHM_TEST_CTL.TEST_UNLOCK = 1; else, returns 0	R	0
4	PRT_RCVDMNS_1	Current value of PRT_RCVDMNS(1) when WHM_TEST_CTL.TEST_UNLOCK = 1; else, returns 0	R	0
3	PRT_RCVDMNS_0	Current value of PRT_RCVDMNS(1) when WHM_TEST_CTL.TEST_UNLOCK = 1; else, returns 0	R	0
2	PRT_RCVDATA_2	Current value of PRT_RCVDATA(2) when WHM_TEST_CTL.TEST_UNLOCK = 1; else, returns 0	R	0
1	PRT_RCVDATA_1	Current value of PRT_RCVDATA(1) when WHM_TEST_CTL.TEST_UNLOCK = 1; else, returns 0	R	0
0	PRT_RCVDATA_0	Current value of PRT_RCVDATA(0) when WHM_TEST_CTL.TEST_UNLOCK = 1; else, returns 0	R	0

Table 22−62. WHM_TEST_CTL

Address Offset	0x0FC		
Physical Address	0x4805 E0FC	**Instance**	USBOT1
Description	WHM test 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
TEST_UNLOCK	Reserved										CTL_ENABLE	UART_TX_I	TLL_DETACH	RCFG_SPEED_2	RCFG_SPEED_1	RCFG_SPEED_0	RCFG_TXSE0_2	RCFG_TXSE0_1	RCFG_TXSE0_0	RCFG_TXEN_2	RCFG_TXEN-1	RCFG_TXEN_0	RCFG_TXDPLS_2	RCFG_TXDPLS_1	RCFG_TXDPLS_0	RCFG_TXDMNS_2	RCFG_TXDMNS_1	RCFG_TXDMNS_0	RCFG_SUSPEND_2	RCFG_SUSPEND_1	RCFG_SUSPEND_0

Bits	Field Name	Description	Type	Reset
31	TEST_UNLOCK	Unlock the testability registers	RW	0
30:21	Reserved	Reserved	R	0x---
20	CTL_ENABLE	Enables USB_HMC to use WHM_TEST_CTL register bits rather than USB_WHC and UART signals 0x0: USB_HMC connects to USB_WHC and UART without any controllability 0x1: USB_HMC use WHM_TEST_CTL	RW	0

Table 22−62. WHM_TEST_CTL (Continued)

Bits	Field Name	Description	Type	Reset
19	UART_TX_I	Used by USB_HMC instead of the UARTs TX signal	RW	0
18	TLL_DETACH	Inverted value used by USB_HMC TLL for the TLL_ATTACH signal	RW	0
17	RCFG_SPEED_2	Used by USB_HMC instead of the USB_WHCs RCFG_SPEED(2) signal	RW	0
16	RCFG_SPEED_1	Used by USB_HMC instead of the USB_WHCs RCFG_SPEED(1) signal	RW	0
15	RCFG_SPEED_0	Used by USB_HMC instead of the USB_WHCs RCFG_SPEED(0) signal	RW	0
14	RCFG_TXSE0_2	Used by USB_HMC instead of the USB_WHCs RCFG_TXSE0(2) signal	RW	0
13	RCFG_TXSE0_1	Used by USB_HMC instead of the USB_WHCs RCFG_TXSE0(1) signal	RW	0
12	RCFG_TXSE0_0	Used by USB_HMC instead of the USB_WHCs RCFG_TXSE0(0) signal	RW	0
11	RCFG_TXEN_2	Used by USB_HMC instead of the USB_WHCs RCFG_TXEN(2) signal	RW	0
10	RCFG_TXEN_1	Used by USB_HMC instead of the USB_WHCs RCFG_TXEN(1) signal	RW	0
9	RCFG_TXEN_0	Used by USB_HMC instead of the USB_WHCs RCFG_TXEN(0) signal	RW	0
8	RCFG_TXDPLS_2	Used by USB_HMC instead of the USB_WHCs RCFG_TXDPLS(0) signal	RW	0
7	RCFG_TXDPLS_1	Used by USB_HMC instead of the USB_WHCs RCFG_TXDPLS(1) signal	RW	0
6	RCFG_TXDPLS_0	Used by USB_HMC instead of the USB_WHCs RCFG_TXDPLS(2) signal	RW	0
5	RCFG_TXDMNS_2	Used by USB_HMC instead of the USB_WHCs RCFG_TXDMNS(2) signal	RW	0
4	RCFG_TXDMNS_1	Used by USB_HMC instead of the USB_WHCs RCFG_TXDMNS(1) signal	RW	0
3	RCFG_TXDMNS_0	Used by USB_HMC instead of the USB_WHCs RCFG_TXDMNS(0) signal	RW	0
2	RCFG_SUSPEND_2	Used by USB_HMC instead of the USB_WHCs RCFG_SUSPEND(2) signal	RW	0
1	RCFG_SUSPEND_1	Used by USB_HMC instead of the USB_WHCs RCFG_SUSPEND(1) signal	RW	0
0	RCFG_SUSPEND_0	Used by USB_HMC instead of the USB_WHCs RCFG_SUSPEND(0) signal	RW	0

Table 22–63. HHC_TEST_CFG

Address Offset	0x100		
Physical Address	0x4805 E100	**Instance**	USBOT1
Description	HC test 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																												HHCSHORTRST	HHCCLKSYNC	HHCCONTROLENABLE	HHCTESTENABLE

Bits	**Field Name**	**Description**	**Type**	**Reset**
31:4	Reserved	Reserved	RW	0x-------
3	HHCSHORTRST	Short reset enable 0x0: Normal count of 12-MHz clock used for reset, resume, etc. 0x1: Short count of 12-MHz clock used for reset, resume, etc.	RW	0
2	HHCCLKSYNC	Enables mechanism to hold USB_WHCs data recovery PLL in reset 0x0: Data recovery PLL is in functional mode. 0x1: Data recovery PLL is in reset.	RW	0
1	HHCCONTROL ENABLE	Enables HHC-level controllability 0x0: Disabled HHC level controllability 0x1: Enabled HHC level controllability	RW	0
0	HHCTESTENABLE	Enables HHC-level testability 0x0: Disabled HHC level testability 0x1: Enabled HHC level testability	RW	0

Table 22−64. HHC_TEST_CTL

Address Offset	0x104		
Physical Address	0x4805 E104	**Instance**	USBOT1
Description	HHC test 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																											BISTEN	BISTLY_TM	BISTRSTMEM	IRQ	SOF

Bits	Field Name	Description	Type	Reset
31:5	Reserved	Reserved	RW	0x-------
4	BISTEN	This bit value is placed on HHCs HCI_MBIST_EN_I input port.	RW	0
3	BISTLV_TM	This bit value is placed on HHCs HCI_MBIST_LV_TM_I input port.	RW	0
2	BISTRSTMEM	This bit value is placed on HHCs HCI_MBIST_RST_MEM_I output port.	RW	0
1	IRQ	This bit value is placed on HHCs HCI_MIRQ_ON output port.	RW	0
0	SOF	This bit value is placed on HHCs HCI_MSOF_ON output port.	RW	0

Table 22–65. HHC_TEST_OBSV

Address Offset	0x108		
Physical Address	0x4805 E108	**Instance**	USBOT1
Description	Provides S/W visibility of certain signals		
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																			RDCOUNTOFDWORDS			BISTGO	BISTDONE	BISTCOMPSTAT	SYSERR	RDFIFOEMPTY	RDFIFOFIRST	WRFIFOFULL	WRFIFOLAST	FOFORDCOMP	LBACCESSCOMP

Bits	Field Name	Description	Type	Reset
31:13	Reserved	Reserved	R	0x-----
12:10	RDCOUNTOFDWORDS	Count of DWORDs read from LB slave during last master read request 0x0: 0 DWORD received 0x1: 1 DWORD received 0x2: 2 DWORD received 0x3: 3 DWORD received 0x4: 4 DWORD received	R	0x0
9	BISTGO	Current value of MBIST_GO	R	0
8	BISTDONE	Current value of MBIST_DONE	R	0
7	BISTCOMPSTAT	Current value of MBIST_COMP_STAT	R	0
6	SYSERR	System error has occurred on LB when 0.	R	0
5	RDFIFOEMPTY	Read FIFO is empty when 0.	R	0
4	RDFIFOFIRST	Read FIFO has at least one DWORD of data when 0.	R	0
3	WRFIFOFULL	Write FIFO is full when 0.	R	0
2	WRFIFOLAST	Write FIFO is one word from full when 0.	R	0
1	FIFORDCOMP	FifoRead is completed when 0.	R	0
0	LBACCESSCOMP	Local bus access is completed when 0.	R	0

Table 22−66. REVDEV

Address Offset	0x200		
Physical Address	0x4805 E200	**Instance**	USBOT1
Description	Revision register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								REV_NB							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	0x--
7:0	REV_NB	Revision number of current USB device controller. Examples: 0x01 for 0.1, 0x21 for 2.1	R	

Table 22−67. EP_NUM

Address Offset	0x204		
Physical Address	0x4805 E204	**Instance**	USBOT1
Description	Endpoint selection register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									SETUP_SEL	EP_SEL	EP_DIR	EP_NUM			

Bits	Field Name	Description	Type	Reset
15:7	Reserved	Reserved	RW	0x---
6	SETUP_SEL	The setup FIFO select bit. Set by the local host in response to a setup general USB interrupt, to access the EP0 read-only setup FIFO when reading DATA register. Setting this bit has for effect to clear IRQ_SRC.SETUP interrupt bit. When this bit is set, other EP_NUM register bits value must be 0. 0x0: No access 0x1: Access permitted	RW	0

Table 22−67. EP_NUM (Continued)

Bits	Field Name	Description	Type	Reset
5	EP_SEL	Set by the local host to access the status (STAT_FLG, RXFSTAT) and data (DATA) registers for the endpoint selected. After each access to an endpoint during IT handling, the local host must ABSOLUTELY clear this bit. **Caution:** When the local host sets this bit, it must set SETUP_SEL bit to 0. After having accessed the Endpoint _s FIFO either for read or write access the local host must clear this bit (write 0x0). 0x0: No access 0x1: Access permitted	RW	0
4	EP_DIR	Endpoint direction bit. Gives the direction associated with the endpoint number selected in EP_NUM.EP_NUM. 0x0: OUT endpoint 0x1: IN endoint	RW	0
3:0	EP_NUM	Endpoint number binary. Encoded in these four bits, associated to the direction given by EP_NUM.EP_DIR bit, is the current endpoint selected. All reads and writes to the endpoint status, control and data locations are for this endpoint. 0x0: EP0 0x1: EP1 0x2: EP2 0x3: EP3 0x4: EP4 0x5: EP5 0x6: EP6 0x7: EP7 0x8: EP8 0x9: EP9 0xA: EP10 0xB: EP11 0xC: EP12 0xD: EP13 0xE: EP14 0xF: EP15	RW	0x0

Table 22–68. DATA

Address Offset	0x208		
Physical Address	0x4805 E208	**Instance**	USBOT1
Description	Data register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA															

Bits	Field Name	Description	Type	Reset
15:0	DATA	Transmit/receive FIFO data EP_NUM.EP_DIR = 1: This register contains the data written by the local host to be sent to the USB host during the next IN transaction. Data can only be written successfully if the EP_NUM.EP_SEL bit is asserted. EP_NUM.EP_DIR = 0: This register contains the data received by the USB core from USB host OUT or SETUP transactions. Data can only be read successfully if the EP_NUM.EP_SEL bit is asserted, or if EP_NUM.SETUP_SEL bit is asserted (for setup data). **Note:** A write into DATA register when EP_NUM.EP_DIR=0 and a read from DATA register when EP_NUM.EP_DIR=1 are denied.	RW	0x----

Table 22–69. CTRL

Address Offset	0x20C		
Physical Address	0x4805 E20C	**Instance**	USBOT1
Description	Control register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CLR_HALT	SET_HALT	RESERVED		CLR_DATA_TOGGLE	SET_FIFO_EN	CLR_EP	RESET_EP

Table 22−69. CTRL (Continued)

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	RW	0x--
7	CLR_HALT	Clear halt endpoint (non-ISO) bit. EP_NUM.EP_DIR = 1: This register contains the data written by the local host to be sent to the USB host during the next IN transaction. Data can only be written successfully if the EP_NUM.EP_SEL bit is asserted. EP_NUM.EP_DIR = 0: This register contains the data received by the USB core from USB host OUT or SETUP transactions. Data can only be read successfully if the EP_NUM.EP_SEL bit is asserted, or if EP_NUM.SETUP_SEL bit is asserted (for setup data). **Note:** A write into DATA register when EP_NUM.EP_DIR=0 and a read from DATA register when EP_NUM.EP_DIR=1 are denied. 0x0: No action 0x1: Clear halt condition	RW	0
6	SET_HALT	Set halt endpoint (non-ISO). Concerns non-ISO endpoints only. Used by the local host to halt the selected endpoint. **Caution:** If the endpoint to halt is used by a DMA channel, the local host must disable the DMA channel before setting halt condition for this endpoint. Always reads 0x0. 0x0: No action 0x1: Clear halt condition	RW	0
5:4	Reserved	Reserved	RW	0x-
3	CLR_DATA_ TOGGLE	Clear data toggle bit of endpoint. Concerns non-ISO endpoints only. This bit is set by the local host to force Data PID to DATA0 for an interrupt or bulk endpoint. Always reads 0. 0x0: No action 0x1: Reset Data PID to DATA0	RW	0

Table 22−69. CTRL (Continued)

Bits	Field Name	Description	Type	Reset
2	SET_FIFO_EN	Set FIFO enable (non-ISO) bit. Only concerns non-ISO Endpoints. If the selected endpoint direction is IN, this bit is used by the local host to enable the USB device to transmit data from the FIFO at the next valid IN token. If the selected endpoint direction is OUT, this bit is used by the local host to enable the USB device to receive data from the USB host at the next valid OUT transaction. If not set the device returns a NAK handshake. **Caution:** The local host must never enable endpoint 0 FIFO out of control transfers. For bulk and interrupt endpoints, FIFO must never be enabled when halt feature is set. FIFO must not be enabled either when RX FIFO is not empty. Setting this bit when RX FIFO is not empty will have no effect. Furthermore, during EP interrupts handling, the local host must have cleared the interrupt bit before setting CTRL.SET_FIFO_EN bit (to avoid masked ACK interrupts). Always reads 0x0. 0x0: No action 0x1: FIFO enabled	RW	0
1	CLR_EP	Clear endpoint. Set by the local host to clear the selected endpoint FIFO pointers and flags. It also clears previous transaction handshake status. For ISO endpoints or non-ISO double-buffered endpoints, both foreground and background FIFO are cleared. Always reads 0x0. 0x0: No action 0x1: Clear endpoint	RW	0
0	RESET_EP	Endpoint reset (non-control) bit. Concerns non-ISO endpoints only. Set by the local host to reset the selected endpoint. It clears a halt condition, clears FIFO and previous transactions handshake status. For an ISO endpoint, it only clears the FIFO (both foreground and background). Always reads 0x0. 0x0: No action 0x1: Reset endpoint	RW	0

Table 22−70. STAT_FLG

Address Offset	0x210		
Physical Address	0x4805 E210	**Instance**	USBOT1
Description	Status register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NO_RXRACKET	MISS_IN	DATA_FLUSH	ISO_ERR	RESERVED		ISO_FIFO_EMPTY	ISO_FIFO_FULL	RESERVED	EP_HALTED	STALL	NAK	ACK	FIFO_EN	NON_ISO_FIFO_EMPTY	NON_ISO_FIFO_FULL

Bits	Field Name	Description	Type	Reset
15	NO_RXPACKET	Isochronous no packet received (ISO OUT) bit. Only concerns ISO OUT endpoints. Notify the local host that the core has not received any isochronous packet on the endpoint. Updated on a SOF (start of frame). 0x0: Endpoint received a packet on the previous frame. 0x1: Endpoint did not receive a packet on the previous frame.	R	1
14	MISS_IN	Isochronous missed IN token for previous frame (ISO IN) bit. Only concerns ISO IN endpoints. Notify the local host that the core missed a valid ISO IN token during previous frame and that TX data were flushed from the FIFO instead of being transmitted to the USB host. Updated on a SOF (start of frame). 0x0: Endpoint received an IN token the previous frame. 0x1: Endpoint did not receive an IN token the previous frame and TX data were flushed.	R	0
13	DATA_FLUSH	Isochronous receive data flush (ISO OUT) bit. Only concerns ISO OUT endpoints. When set, indicates that data were flushed from the isochronous FIFO that was moved from the foreground to the background (the local host does not read all of the data from the foreground FIFO in a frame). Updated every frame.	R	0

Note: The updates for non-ISO transactions are done at the end of each non-transparent and valid transaction to a given endpoint, if no non-handled interrupt is pending on the endpoint.

The definition of a non-transparent, non-ISO IN transaction is one that responds with an ACK handshake or a STALL handshake, or optionally a NAK handshake if the NAK_EN bit SYSCON1[4] is asserted to 1. An ERR handshake or a NAK handshake when the NAK_EN bit SYSCON1[4] is 0 is considered transparent.

Table 22−70. STAT_FLG (Continued)

Bits	Field Name	Description	Type	Reset
13 (cont'd)	DATA_FLUSH (cont'd)	0x0: Not significant 0x1: ISO packet received with errors		
12	ISO_ERR	Isynchronous receive data error (ISO_OUT) bit. Concerns ISO OUT endpoints. When set, indicates that the ISO data packet is received correctly; that is, the core detects an error in the data packet (CRC, bit stuffing, PID check) or an overrun condition occurs in the FIFO. When this bit is set, the FIFO contents are automatically flushed by the core and the FIFO status is empty. Updated every frame. 0x0: Not significant 0x1: ISO packet received with errors	R	0
11:10	Reserved	Reserved	R	0x−
9	ISO_FIFO_EMPTY	ISO FIFO empty bit concerns ISO endpoints only. Set when the FIFO or selected ISO endpoint is empty, either via an appropriate CTRL_CLR_EP bit or CTRL_RESET_EP bit or after successful reads from the selected FIFO. 0x0: ISO FIFO is not empty. 0x1: ISO FIFO is empty.	R	0
8	ISO_FIFO_FULL	ISO FIFO full bit concerns ISO endpoints only. Set when the FIFO for the selected ISO endpoints is full. This condition is cleared by setting the CTRL_CLR_EP bit or CTRL_RESET_EP bit or after one successful read (by the local or USB host). 0x0: ISO FIFO is not full. 0x1: ISO FIFO is full.	R	0
7	Reserved		R	0
6	EP_HALTED	Endpoint halted flag (non-ISO) bit. Only concerns non-ISO endpoints. When set, indicates the selected endpoint is halted. The endpoint can be put into the halt state only by the local host writing the endpoint halt control bit (in response to a SET_FEATURE request for instance). 0x0: The selected endpoint is not halted 0x1: The selected endpoint is halted	R	0

Note: The updates for non-ISO transactions are done at the end of each non-transparent and valid transaction to a given endpoint, if no non-handled interrupt is pending on the endpoint.

The definition of a non-transparent, non-ISO IN transaction is one that responds with an ACK handshake or a STALL handshake, or optionally a NAK handshake if the NAK_EN bit SYSCON1[4] is asserted to 1. An ERR handshake or a NAK handshake when the NAK_EN bit SYSCON1[4] is 0 is considered transparent.

Table 22−70. STAT_FLG (Continued)

Bits	Field Name	Description	Type	Reset
5	STALL	Transaction stall (non-ISO)bit. Only concerns non-ISO endpoints. Set at the end of a transaction if a STALL handshake packet was returned to the USB host, and if no non-handled interrupt is pending on current buffer. The core automatically returns a STALL packet if a valid IN token is received by a halted TX endpoint, or if a valid OUT transaction is received by a halted RX endpoint, or if there is a request error (endpoint 0). Cleared when the local host has finished to handle corresponding interrupt (at EP_NUM.EP_Sel bit deselection). 0x0: No STALL handshake was returned. 0x1: STALL handshake was returned.	R	0
4	NAK	Transaction non-acknowledge (non-ISO) bit. Set at the end of a transaction if a NAK handshake is returned to the USB host, and if no non-handled interrupt is pending on current buffer. The USB core automatically returns a NAK handshake to the USB host if a valid IN token is received by a TX endpoint or if a valid OUT transaction is received by an RX endpoint, and the STAT_FLG.FIFO_En bit is not set for the endpoint. Cleared when the local host has finished to handle corresponding interrupt (at EP_NUM.EP_Sel bit deselection). 0x0: No NAK handshake was returned. 0x1: NAK handshake was returned.	R	0
3	ACK	Transaction acknowledge (non-ISO) bit. Set at the end of a non-transparent valid IN transaction if the data packet was sent successfully to the USB host, and the ACK handshake was received, or at the end of a non-transparent valid OUT transaction if the data packet was received successfully by the USB device, and the ACK handshake was returned. Cleared when the local host has finished to handle corresponding interrupt (at EP_NUM.EP_Sel bit deselection). 0x0: No ACK handshake was returned. 0x1: ACK handshake was returned.	R	0
2	FIFO_EN	FIFO enable status (non-ISO) bit. This bit is asserted when CTRL.SET_FIFO_EN is set to 1 and cleared automatically after a transaction completes with an ACK, STALL. 0x0: Non-ISO endpoint FIFO is disabled. 0x1: Non-ISO endpoint FIFO is enabled.	R	0

Note: The updates for non-ISO transactions are done at the end of each non-transparent and valid transaction to a given endpoint, if no non-handled interrupt is pending on the endpoint.

The definition of a non-transparent, non-ISO IN transaction is one that responds with an ACK handshake or a STALL handshake, or optionally a NAK handshake if the NAK_EN bit SYSCON1[4] is asserted to 1. An ERR handshake or a NAK handshake when the NAK_EN bit SYSCON1[4] is 0 is considered transparent.

Table 22−70. STAT_FLG (Continued)

Bits	Field Name	Description	Type	Reset
1	NON_ISO_FIFO_ EMPTY	Non-ISO FIFO empty bit. Set when the FIFO for the selected non-ISO endpoint is empty, either via an appropriate CTRL.CLR_EP bit or CTRL.RESET_EP bit or after successful reads from the selected FIFO. 0x0: Non-ISO endpoint FIFO is not empty. 0x1: Non-ISO endpoint FIFO is empty.	R	1
0	NON_ISO_FIFO_ FULL	Non-ISO FIFO full bit. Only concerns non-ISO endpoints. Set when the FIFO for the selected non-ISO endpoint is full. This condition is cleared by setting the CTRL.CLR_EP bit or CTRL.RESET_EP bit, or after one successful read (by the local host or the USB host). 0x0: Non-ISO endpoint FIFO is not full. 0x1: Non-ISO endpoint FIFO is full.	R	0

Note: The updates for non-ISO transactions are done at the end of each non-transparent and valid transaction to a given endpoint, if no non-handled interrupt is pending on the endpoint.

The definition of a non-transparent, non-ISO IN transaction is one that responds with an ACK handshake or a STALL handshake, or optionally a NAK handshake if the NAK_EN bit SYSCON1[4] is asserted to 1. An ERR handshake or a NAK handshake when the NAK_EN bit SYSCON1[4] is 0 is considered transparent.

Table 22−71. RXFSTAT

Address Offset	0x214		
Physical Address	0x4805 E214	**Instance**	USBOT1
Description	Receive FIFO status register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						RXF_COUNT									

Bits	Field Name	Description	Type	Reset
15:10	Reserved	Reserved	R	0x--
9:0	RXF_COUNT	Receive FIFO byte count 10-bit field. Indicates the number of bytes currently in the receive FIFO.	R	0x000

Table 22−72. SYSCON1

Address Offset	0x218		
Physical Address	0x4805 E218	**Instance**	USBOT1
Description	System configuration register 1		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							CFG_LOCK	DATA_ENDIAN	DMA_ENDIAN	RESERVED	NAK_EN	AUTODEC_DIS	SELF_PWR	SOFF_DIS	PULLUP_EN

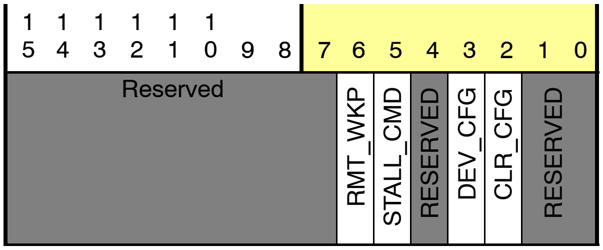
Bits	Field Name	Description	Type	Reset
15:9	Reserved	Reserved	RW	0x00
8	CFG_LOCK	Device configuration locked bit. After the local host enters the device configuration (registers 0x20 to 0x3F), it must set this bit so that the device can be used. When the device configuration is not locked, the device is not ready to be used. 0x0: Device configuration is not locked. Device is not ready. 0x1: Device configuration is locked.	RW	0
7	DATA_ENDIAN	Data endian bit. Can be set by local host to select little or big endian format on data access (read or write). 0x0: Little endian 0x1: Big endian	RW	0
6	DMA_ENDIAN	DMA data endian bit. Can be set by local host to select little or big endian format on data access (read or write). 0x0: Little endian 0x1: Big endian	RW	0
5	Reserved	Reserved	RW	-
4	NAK_EN	NAK enable bit. Can be set by the local host to be signaled for NAK transaction handshake response. Note: If the local host sets this bit, it must wait for a NAK interrupt before selecting TX endpoint to write TX data. 0x0: NAK is disabled. 0x1: NAK is enabled.	RW	0
3	AUTODEC_DIS	Autodecode process disabled. This function can be set to force all EP0 transactions (except SET_ADDRESS) to be handled by software. 0x0: Autodecode process is activated. 0x1: Autodecode process is deactivated.	RW	0

Table 22−72. SYSCON1 (Continued)

Bits	Field Name	Description	Type	Reset
2	SELF_PWR	Self-powered bit. Indicates to the USB host if the device is bus-powered or self-powered. This is needed for a GET_DEVICE_STATUS auto-decoded request. 0x0: Bus powered 0x1: Self power	RW	0
1	SOFF_DIS	Shutoff disable bit 0x0: Power shutoff circuitry is enabled. 0x1: Power shutoff circuitry is disabled.	RW	0
0	PULLUP_EN	External pullup enable bit. Allows the device to disconnect itself from the USB bus, forcing the host to reset and reconfigure the device. This bit can be used to prevent from USB traffic when the device is not ready. 0x0: Pullup is disabled. USB host cannot detect the device. In this mode, the 48-MHz USB clock is forced off. 0x1: Pullup is enabled.	RW	0

Table 22−73. SYSCON2

Address Offset	0x21C		
Physical Address	0x4805 E21C	**Instance**	USBOT1
Description	System configuration register 2		
Type	RW		



Bits	Field Name	Description	Type	Reset
15:7	Reserved	Reserved	RW	0x---
6	RMT_WKP	Set-only remote wake-up bit. This set-only bit when written with a 1 initiates the remote wake-up sequence even if DEVSTAT.R_WK_OK bit was not previously set to 1 by the USB host. Then to generate a resume, the software itself must check the Remote Wake-Up Enable value before initiating any wake-up sequence. Always reads 0x0. 0x0: No action 0x1: Initiates remote wake-up sequence	RW	0

Table 22−73. SYSCON2 (Continued)

Bits	Field Name	Description	Type	Reset
5	STALL_CMD	Set-only stall command bit. Only concerns non-autodecoded requests on control endpoint (EP0). This is asserted in response to a USB command where either the command itself or its data is invalid. Asserting this bit forces the non-autodecoded command to complete with a STALL handshake. It has no effect for autodecoded requests. Always reads 0x0. 0x0: No action 0x1: Stall current USB command	RW	0
4	Reserved	Reserved	R	-
3	DEV_CFG	USB device controller. If the local host receives a SET_CONFIGURATION with a valid configuration value, and the device is in addressed state, it must write 1 to this bit to inform the command decode that the device moves to configured state. The DEVSTAT.CFG is set to 1 by the core. **Note:** If the device is already configured when the SET_CONFIGURATION request is received, the local host does not have to set this bit. It must set the SYSCON2.CLR_CFG bit if the new configuration value is 0 to move to addressed state. Always reads 0x0: no effect Write 0x0: no effect 0x0: No action 0x1: Allows DEVSTAT.CFG to be set	RW	0
2	CLR_CFG	Clear configured. If the local host receives a SET_CONFIGURATION with a configuration value of 0, and if device is configured it must write 1 to this bit to inform the command decode that the device becomes deconfigured (moves to addressed state). This register bit has no effect unless the local host receives a SET_CONFIGURATION. Always reads 0x0. 0x0: No action 0x1: Allows DEVSTAT.CFG to be cleared	RW	0
1:0	Reserved	Reserved	RW	0x-

Table 22−74. DEVSTAT

Address Offset	0x220		
Physical Address	0x4805 E220	**Instance**	USBOT1
Description	Device status register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						B_HNP_ENABLE	A_HNP_SUPPORT	A_ALT_HNP_SUPPORT	R_WK_OK	USB_RESET	SUS	CFG	ADD	DEF	ATT

Bits	Field Name	Description	Type	Reset
15:10	Reserved	Reserved	R	0x--
9	B_HNP_ENABLE	The HNP enable for B-device bit is used for on-the-go feature selection purposes. 0x0: Not significant 0x1: HNP enable for B-device	R	0
8	A_HNP_SUPPORT	The B-device is directly connected to an A-device port that supports HNP. 0x0: Not significant 0x1: B-device connection to HNP-capable A-device	R	0
7	A_ALT_HNP_SUPPORT	B-device is connected to an A-device port that is not capable of HNP. 0x0: Not significant 0x1: B-device connection to not-HNP-capable A-device	R	0
6	R_WK_OK	Remote wake-up granted bit. Automatically set and cleared when the core receives a valid set/clear device feature request from the USB host. It returns a 1 when the USB host has granted the function the ability to assert remote wake-up. 0x0: Not significant 0x1: Remote wake-up granted	R	0

Note: This register is double buffered. If the DS_CHG_IE bit IRQ_EN[3] is set (interrupt enabled), the background register is moved into foreground position only after clearing any pending DS_CHG interrupt. Therefore, if there is a state change and a pending DS_CHG interrupt has not been serviced yet, then the recent state change is not visible, because the background register was only updated and not moved into the foreground position.

Table 22−74. DEVSTAT (Continued)

Bits	Field Name	Description	Type	Reset
5	USB_RESET	USB reset signaling is active. A valid USB reset effectively resets all endpoints FIFOS, all the others control registers bits except CFG_LOCK bit SYSCON1[8]. and associated configuration (registers 0x20 to 0x3F), IRQ_EN.DS_CHG_IE and IRQ_SRC.DS_Chg, and for DEVSTAT forces the device to the default state. This bit is cleared at the end of reset. 0x0: Device is not being reset by USB host. 0x1: Device is being reset by USB host.	R	0
4	SUS	Suspended state bit. At minimum, device is attached to the USB, is powered and reset by the USB host, and has had no bus activity for 5 ms. It can also have a unique address and be configured for use. Because the device is suspended, the host cannot use the device function. 0x0: Not suspended 0x1: Suspended	R	0
3	CFG	Configured state bit. Device is attached to the USB, is powered and reset, has a unique address, and is configured. The host can use the function provided by the device. Returns 1 when the USB device is configured after a set SYSCON2.DEV_CFG=1. Remains set to 1 until the device becomes deconfigured. Cleared when the core receives a valid SET_CONFIGURATION request and the local host sets SYSCON2.CLR_CFG bit. While not set to 1, any transactions not for EP0 control are ignored. A GET_ENDPOINT_STATUS to a non-control endpoint is stalled. 0x0: Not configured 0x1: Configured	R	0
2	ADD	Addressed state bit. Device is attached to the USB, is powered and reset, and has a unique device address assigned. Returns 1 after a SET_ADDRESS standard request. Remains set to 1 until the device becomes de-addressed. 0x0: Not addressed 0x1: Addressed	R	0
1	DEF	Default state bit. Returns 1 when the USB device is attached to the USB and is powered and reset. Remains set to 1 until the device becomes de-powered. Device moves into default state as soon as the USB reset is effective. 0x0: Not in default 0x1: Default	R	0

Note: This register is double buffered. If the DS_CHG_IE bit IRQ_EN[3] is set (interrupt enabled), the background register is moved into foreground position only after clearing any pending DS_CHG interrupt. Therefore, if there is a state change and a pending DS_CHG interrupt has not been serviced yet, then the recent state change is not visible, because the background register was only updated and not moved into the foreground position.

Table 22−74. DEVSTAT (Continued)

Bits	Field Name	Description	Type	Reset
0	ATT	Attached state bit. Returns 1 when the device is attached to the USB and powered. Remains set to 1 until the device becomes de-powered. 0x0: Not attached 0x1: Attached	R	0

Note: This register is double buffered. If the DS_CHG_IE bit IRQ_EN[3] is set (interrupt enabled), the background register is moved into foreground position only after clearing any pending DS_CHG interrupt. Therefore, if there is a state change and a pending DS_CHG interrupt has not been serviced yet, then the recent state change is not visible, because the background register was only updated and not moved into the foreground position.

Table 22−75. SOFREG

Address Offset	0x224		
Physical Address	0x4805 E224	**Instance**	USBOT1
Description	Start of frame register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED			FT_LOCK	TS_OK	TS										

Bits	Field Name	Description	Type	Reset
15:13	Reserved	Reserved	R	0x0
12	FT_LOCK	Frame timer locked bit. The USB host sends out a start of frame (SOF) packet every millisecond. When a SOF packet is not received by the device because of a bus error, a local start of frame is generated, used for ISO FIFO switch. Once the core receives two valid SOFs separated by TF (time frame), it sets SOF.FT_LOCK to 1, only if TF is in frame interval IF allowed by USB 1.1 Specification (IF = [11964:12036] USB bit time). If TF is out of this interval, SOF.FT_LOCK value remains 0, and a local SOF is generated by the core. When SOF.FT_LOCK is set, and the frame timer is locked to the timing TF, a local SOF is generated if no valid SOF is received in an interval of TF because of the last valid SOF. The SOF.FT_LOCK bit is cleared if a valid SOF is received out of the interval IF. If the core receives a valid SOF in this interval, the frame timer locks to the new frame time. If the core does not receive a valid SOF, the frame timer remains lock to TF. When SOF.FT_LOCK is cleared, a local SOF is generated after 12036 USB bit time if no valid SOF is received, and SOF.FT_LOCK remains 0. **Note** : Each time a valid SOF is received by the core out of allowed interval IF, a local SOF is generated and ISO FIFO switch.	R	0

Table 22−75. SOFREG (Continued)

Bits	Field Name	Description	Type	Reset
12 (cont'd)	FT_LOCK (cont'd)	0x0: Frame timer is not locked. 0x1: Frame timer is locked.		
11	TS_OK	Timestamp OK bit. This bit indicates that the timestamp in SOF.TS is valid for the current frame. 0x0: Timestamp is invalid. 0x1: Timestamp is valid. A valid SOF packet was received from the USB host.	R	0
10:0	TS	Timestamp number field .This field returns the timestamp from last USB host valid SOF packet. The frame number is valid if SOF.TS_OK is 1. In case of a SOF miss, this value is not updated and TS_OK is cleared.	R	0x000

Table 22−76. IRQ_EN

Address Offset	0x228		
Physical Address	0x4805 E228	**Instance**	USBOT1
Description	Interrupt enable register (enables all non-DMA interrupts)		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								SOF_IE	RESERVED	EPN_RX_IE	EPN_TX_IE	DS_CHG_IE	RESERVED		EP0_IE

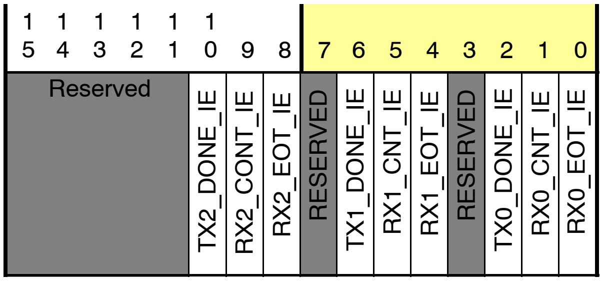
Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	RW	0x--
7	SOF_IE	Start-of-frame interrupt enable 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0
6	Reserved	Reserved	RW	-
5	EPN_RX_IE	Receive endpoint n interrupt enable (non-ISO) 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0
4	EPN_TX_IE	Transmit endpoint n interrupt enable (non-ISO) 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0
3	DS_CHG_IE	Device state changed interrupt enable 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0

Table 22−76. IRQ_EN (Continued)

Bits	Field Name	Description	Type	Reset
2:1	Reserved	Reserved	RW	0x-
0	EP0_IE	EP0 transactions interrupt enable 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0

Table 22−77. DMA_IRQ_EN

Address Offset	0x22C		
Physical Address	0x4805 E22C	**Instance**	USBOT1
Description	DMA interrupt enable register (enables all DMA interrupts)		
Type	RW		



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					TX2_DONE_IE	RX2_CONT_IE	RX2_EOT_IE	RESERVED	TX1_DONE_IE	RX1_CNT_IE	RX1_EOT_IE	RESERVED	TX0_DONE_IE	RX0_CNT_IE	RX0_EOT_IE

Bits	Field Name	Description	Type	Reset
15:11	Reserved	Reserved	RW	0x--
10	TX2_DONE_IE	Transmit DMA channel 2 done interrupt enable 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0
9	RX2_CNTT_IE	Receive DMA channel 2 transactions count interrupt enable. 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0
8	RX2_EOT_IE	Receive DMA channel 2 end of transfer interrupt enable 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0
7	Reserved	Reserved	RW	-
6	TX1_DONE_IE	Transmit DMA channel 1 done interrupt enable 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0
5	RX1_CNT_IE	Receive DMA channel 1 transactions count interrupt enable 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0

Table 22−77. DMA_IRQ_EN (Continued)

Bits	Field Name	Description	Type	Reset
4	RX1_EOT_IE	Receive DMA channel 1 end of transfer interrupt enable 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0
3	Reserved	Reserved	RW	-
2	TX0_DONE_IE	Transmit DMA channel 0 done interrupt enable 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0
1	RX0_CNT_IE	Receive DMA channel 0 transactions count interrupt enable 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0
0	RX0_EOT_IE	Receive DMA channel 0 end of transfer interrupt enable 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0

Table 22−78. IRQ_SRC

Address Offset	0x230		
Physical Address	0x4805 E230	**Instance**	USBOT1
Description	Interrupt source register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					TXN_DONE	RXN_CNT	RXN_EOT	SOF	RESERVED	EPN_RX	EPN_TX	DS_CHG	SETUP	EP0_RX	EP0_TX

Bits	Field Name	Description	Type	Reset
15:11	Reserved	Reserved	RW	0x--
10	TXN_DONE	Transmit DMA channel n done interrupt flag (non-ISO) bit. Only for non-ISO DMA transfer. Never set for ISO DMA transfer. Set automatically by the core when a transmit DMA channel has completed the programmed transfer by servicing the last IN transaction from the USB host (TXDMAn.TXn_TSC (transfer size counter) equals 0 and the last IN transaction completes with an ACK). When asserted, the local host must read DMAN_STAT register to identify the endpoint number for which the transfer completed. **Note:** The endpoint interrupt IRQ_SRC.EPn_TX is never set for the assigned endpoint to TX DMA Channel n. 0x0: No action	RW	0

Table 22−78. IRQ_SRC (Continued)

Bits	Field Name	Description	Type	Reset
10 (cont'd)	TXN_DONE (cont'd)	0x1: Non-ISO transmit DMA transfer for a channel has ended.		
9	RXN_CNT	Receive DMA channel n transactions count interrupt flag (non-ISO) bit. Only for non-ISO DMA transfer. Never set for ISO DMA transfer and never set if the interrupt enable bit associated is not set. It means that no poll operation is possible on the DMA. Set automatically by the core during an active receive DMA transfer on each time RXDMAn.RXn_TC equals 0 after an OUT transaction with ACK status. This bit is set after RX DMA data have been read (end of DMA request). When this bit is asserted, the local host must read DMAN_STAT register to identify the endpoint number for which the transfer completed. Note that an RXn_CNT IT is asserted also if RXDMAn.RXn_STOP is set; in this case, both IRQ_SRC.RXn_EOT and IRQ_SRC.RXn_CNT are asserted. 0x0: No action 0x1: Non-ISO receive DMA transfer for a channel has reached transactions count level.	RW	0
8	RXN_EOT	Receive DMA channel n end of transfer interrupt flag (non-ISO) bit. Only for non-ISO DMA transfer. Never set for ISO DMA transfer and never set if the interrupt enable bit associated is not set. It means that no poll operation is possible on the DMA. Set automatically by the core when a receive DMA channel has detected an end-of-transfer (EOT) packet during the last OUT transaction from the USB host. This bit is set after RX DMA data have been read (end of DMA request). When this happens, the DMA assigned endpoint FIFO is kept disabled (STAT_FLG.FIFO_En=0) to avoid receiving a new packet data from the USB host. The local host can grant again DMA transfer to the same endpoint by just enabling the FIFO again (STAT_FLG.FIFO_EN=1). An EOT is detected when the core receives a data packet whose size is less than configured endpoint FIFO size (or empty), or when RXDMAn.RXn_TC equals 0 after an OUT transaction with ACK status, and RXDMAn.RXn_STOP bit is set. When this bit is asserted, the local host must read DMAN_STAT.DMAn_RX_IT_SRC to identify the endpoint number for which the transfer completed, and DMAN_STAT.DMAn_RX_SB to be informed of an odd number of bytes received during the last transaction (useful for 16-bit read access from DATA_DMA register). **Note:** The endpoint interrupt IRQ_SRC.EPn_RX is never set for the assigned endpoint to RX DMA channel. 0x0: No action 0x1: Non-ISO receive DMA transfer for a channel has ended.	RW	0

Table 22−78. IRQ_SRC (Continued)

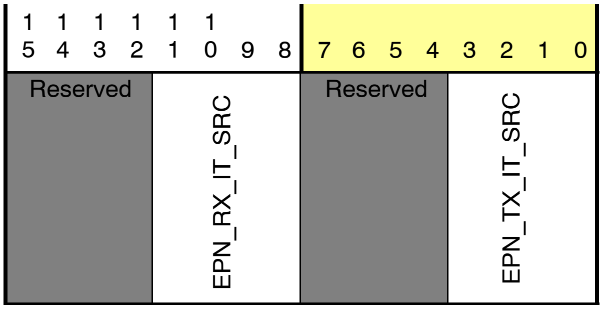
Bits	Field Name	Description	Type	Reset
7	SOF	Start-of-frame interrupt flag bit. Every 1 ms, the USB host outputs a start-of-frame packet to the functions. The SOF bit reflects when a new SOF is received. 0x0: No action 0x1: Start-of-frame packet received (or internal SOF)	RW	0
6	Reserved	Reserved	RW	-
5	EPN_RX	EPn OUT transactions interrupt flag bit. Only concerns non-ISO endpoints. Automatically set by the core when a handshake sequence occurs for an OUT transaction to an interrupt of bulk endpoint (NAK with SYSCON1.NAK_EN set, ACK or STALL). The local host must read EPN_STAT register to identify the endpoint causing the interrupt. 0x0: No action 0x1: OUT transaction detected on an endpoint	RW	0
4	EPN_TX	EPn IN transactions interrupt flag bit. Only concerns non-ISO endpoints. Automatically set by the core when a handshake sequence occurs for an IN transaction to an interrupt of bulk endpoint (NAK with SYSCON1.NAK_EN set, ACK or STALL). The local host must read EPN_STAT register to identify the endpoint causing the interrupt. 0x0: No action 0x1: IN transaction detected on an endpoint value after MPU or USB reset is low	RW	0
3	DS_CHG	Device state changed interrupt flag bit. Automatically set by the core when the state of the device changes (when the core modifies any of the bits present in the DEVSTAT register). When cleared, the background DEVSTAT register moves into foreground position. 0x0: No action 0x1: Device state change detected	RW	0
2	SETUP	Setup transaction interrupt flag bit. Automatically set by the core when a valid setup transaction completes on control endpoint for a non-autodecoded control request. Cleared automatically by the core when the local host sets EP_NUM.SETUP_SEL bit when reading setup data. 0x0: No action 0x1: Valid setup transaction occurred on endpoint 0.	RW	0
1	EP0_RX	EP0 OUT transactions interrupt flag bit. Set automatically by the core when a handshake sequence occurs for a non auto-decoded OUT transaction to control endpoint (NAK with SYSCON1.NAK_EN set, ACK or STALL). 0x0: No action 0x1: OUT transaction on EP0	RW	0

Table 22−78. IRQ_SRC (Continued)

Bits	Field Name	Description	Type	Reset
0	EP0_TX	EP0 IN transactions interrupt flag bit. Set automatically by the core when a handshake sequence occurs for a non auto-decoded IN transaction to control endpoint (NAK with SYSCON1.NAK_EN set, ACK or STALL). 0x0: No action 0x1: IN transaction on EP0	RW	0

Table 22−79. EPN_STAT

Address Offset	0x234		
Physical Address	0x4805 E234	**Instance**	USBOT1
Description	Non-ISO endpoint interrupt status register		
Type	R		



Bits	Field Name	Description	Type	Reset
15:12	Reserved	Reserved	R	0x-
11:8	EPN_RX_IT_SRC	Receive endpoint interrupt source (non-ISO) bit. Only concerns non-ISO endpoints. When IRQ_SRC.EPn_RX flag is set, the endpoint causing the interrupt condition is encoded in these four register bits. When IRQ_SRC.EPn_RX flag is cleared, the four bits read as 0. 0x0: No receive endpoint interrupt is pending. 0x1: EP1 0x2: EP2 0x3: EP3 0x4: EP4 0x5: EP5 0x6: EP6 0x7: EP7 0x8: EP8	R	0x0

Note: If a nontransparent transaction occurs before a previous one on another endpoint in the same direction, the second interrupt is asserted only after the USB device controller clears the first one and EPN_STAT is updated with the corresponding interrupt assertion.

Table 22−79. EPN_STAT (Continued)

Bits	Field Name	Description		Type	Reset
11:8	EPN_RX_IT_SRC	0x9:	EP9		
(cont'd)	(cont'd)	0xA:	EP10		
		0xB:	EP11		
		0xC:	EP12		
		0xD:	EP13		
		0xE:	EP14		
		0xF:	EP15		
7:4	Reserved	Reserved		R	0x-
3:0	EPN_TX_IT_SRC	Transmit endpoint interrupt source (non-ISO) bit. Only concerns non-ISO endpoints. When IRQ_SRC.EPn_TX flag is set, the endpoint causing this flag to be set is encoded in these four register bits. When IRQ_SRC.EPn_TX flag is cleared, the four bits read as 0.		R	0x0
		0x0:	No transmit endpoint interrupt is pending.		
		0x1:	EP1		
		0x2:	EP2		
		0x3:	EP3		
		0x4:	EP4		
		0x5:	EP5		
		0x6:	EP6		
		0x7:	EP7		
		0x8:	EP8		
		0x9:	EP9		
		0xA:	EP10		
		0xB:	EP11		
		0xC:	EP12		
		0xD:	EP13		
		0xE:	EP14		
		0xF:	EP15		

Note: If a nontransparent transaction occurs before a previous one on another endpoint in the same direction, the second interrupt is asserted only after the USB device controller clears the first one and EPN_STAT is updated with the corresponding interrupt assertion.

Table 22–80. DMAN_STAT

Address Offset	0x238		
Physical Address	0x4805 E238	**Instance**	USBOT1
Description	Non-ISO DMA interrupt status register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED			DMAN_RX_SB	DMAN_RX_IT_SRC				Reserved				DMAN_TX_IT_SRC			

Bits	Field Name	Description	Type	Reset
15:13	Reserved	Reserved	R	0x-
12	DMAN_RX_SB	DMA receive single byte. Concerns non-ISO endpoints (ISO endpoints receives a constant number of bytes). Set when an IRQ_SRC.RXn_EOT interrupt is asserted and the core received an odd number of bytes during the last transaction. This is used to know the exact number of bytes received in case of a 16-bit read access from DATA_DMA register. When IRQ_SRC.RXn_EOT flag is cleared, this bit read as 0. 0x0: No EOT DMA interrupt is pending or the core received an even number of bytes during the last transaction. 0x1: EOT DMA interrupt is pending and the core received an even number of bytes during the last transaction.	R	0
11:8	DMAN_RX_IT_ SRC	DMA receive interrupt source. Only concerns non-ISO endpoints. When IRQ_SRC.RXn_EOT flag is set, the endpoint causing this flag to be set is encoded in these four register bits. When IRQ_SRC.RXn_EOT flag is cleared, the four bits read as 0. 0x0: No receive DMA interrupt is pending. 0x1: EP1 0x2: EP2 0x3: EP3 0x4: EP4 0x5: EP5 0x6: EP6	R	0x0

Note: If a DMA interrupt occurs before a previous one on another endpoint in the same direction, the second interrupt is asserted only after the USB device controller clears the first one. DMAn_STAT is updated when the corresponding interrupt is asserted.

Table 22−80. DMAN_STAT (Continued)

Bits	Field Name	Description	Type	Reset
11:8 (cont'd)	DMAN_RX_IT_SRC (cont'd)	0x7: EP7 0x8: EP8 0x9: EP9 0xA: EP10 0xB: EP11 0xC: EP12 0xD: EP13 0xE: EP14 0xF: EP15		
7:4	Reserved	Reserved	R	0x-
3:0	DMAN_TX_IT_SRC	DMA transmit interrupt source. Only concerns non-ISO endpoints. When IRQ_SRC.TXn_DONE flag is set, the endpoint causing this flag to be set is encoded in these four register bits. When IRQ_SRC.TXn_DONE flag is cleared, the four bits read as 0. 0x0: No transmit DMA interrupt is pending. 0x1: EP1 0x2: EP2 0x3: EP3 0x4: EP4 0x5: EP5 0x6: EP6 0x7: EP7 0x8: EP8 0x9: EP9 0xA: EP10 0xB: EP11 0xC: EP12 0xD: EP13 0xE: EP14 0xF: EP15	R	0x0

Note: If a DMA interrupt occurs before a previous one on another endpoint in the same direction, the second interrupt is asserted only after the USB device controller clears the first one. DMAn_STAT is updated when the corresponding interrupt is asserted.

Table 22–81. RXDMA_CFG

Address Offset	0x240		
Physical Address	0x4805 E240	**Instance**	USBOT1
Description	DMA receive channels configuration register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED			RX_REQ	RXDMA2_EP				RXDMS1_EP				RXDMA0_EP			

Bits	Field Name	Description	Type	Reset
15:13	Reserved	Reserved	RW	0x-
12	RX_REQ	The RX DMA request active. Allows the RXDMA request to be configurable level or pulse sensitive. When pulse sensitive, the request is active during 2 cycles. Note: The system DMA requests are hard-coded to level sensitive during integration in OMAP2420. This behavior corresponds to RX DMA request configured in pulse sensitive. The RX_REQ bit must be set to 1 to make the RX DMA request acknowledged by the system DMA. Setting this bit to 0 is not compliant with system DMA integration and does not work 0x0: RX DMA request active level – does not work in OMAP2420 0x1: RX DMA request active pulse	RW	0
11:8	RXDMA2_EP	Receive endpoint number for DMA channel 2. The endpoint number binary encoded in these four bits is the current receive endpoint selected for DMA channel 2. A zero value indicates that the DMA channel 2 is deactivated. Any other value automatically enables receive DMA transfer for the selected endpoint. 0x0: Receive DMA channel 2 is deactivated. 0x1: EP1 0x2: EP2 0x3: EP3 0x4: EP4 0x5: EP5 0x6: EP6 0x7: EP7 0x8: EP8 0x9: EP9 0xA: EP10 0xB: EP11 0xC: EP12 0xD: EP13 0xE: EP14 0xF: EP15	RW	0x0

Caution: System software must not program RXDMAx_EP to endpoint numbers that are not configured as DMA endpoints.

Table 22−81. RXDMA_CFG (Continued)

Bits	Field Name	Description	Type	Reset
7:4	RXDMA1_EP	Receive endpoint number for DMA channel 1. The endpoint number binary encoded in these four bits is the current receive endpoint selected for DMA channel 1. A zero value indicates that the DMA channel 1 is de-activated. Any other value automatically enables receive DMA transfer for the selected endpoint.	RW	0x0
		0x0: Receive DMA channel 1 is deactivated.		
		0x1: EP1		
		0x2: EP2		
		0x3: EP3		
		0x4: EP4		
		0x5: EP5	RW	0x0
		0x6: EP6		
		0x7: EP7		
		0x8: EP8		
		0x9: EP9		
		0xA: EP10		
		0xB: EP11		
		0xC: EP12		
		0xD: EP13		
		0xE: EP14		
		0xF: EP15		
3:0	RXDMA0_EP	Receive endpoint number for DMA channel 0.The endpoint number binary encoded in these four bits is the current receive endpoint selected for DMA channel 0. A zero value indicates that the DMA channel 0 is de-activated. Any other value automatically enables receive DMA transfer for the selected endpoint.	RW	0x0
		0x0: Receive DMA channel 0 is deactivated.		
		0x1: EP1		
		0x2: EP2		
		0x3: EP3		
		0x4: EP4		
		0x5: EP5		
		0x6: EP6		
		0x7: EP7		
		0x8: EP8		
		0x9: EP9		
		0xA: EP10		

Caution: System software must not program RXDMAx_EP to endpoint numbers that are not configured as DMA endpoints.

Table 22−81. RXDMA_CFG (Continued)

Bits	Field Name	Description	Type	Reset
3:0 (cont'd)	RXDMA0_EP (cont'd)	0xB: EP11 0xC: EP12 0xD: EP13 0xE: EP14 0xF: EP15		

Caution: System software must not program RXDMAx_EP to endpoint numbers that are not configured as DMA endpoints.

Table 22−82. TXDMA_CFG

Address Offset	0x244		
Physical Address	0x4805 E244	**Instance**	USBOT1
Description	DMA transmit channels configuration register		
Type	RW		

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="3">RESERVED</td><td>TX_REQ</td><td colspan="4">TXDMA2_EP</td><td colspan="4">TXDMS1_EP</td><td colspan="4">TXDMA0_EP</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
15:13	Reserved	Reserved	R	0x-
12	TX_REQ	The TX DMA request active. Allows the TXDMA request to be configurable level or pulse sensitive. When pulse sensitive, the request is active during 2 cycles. Note: The system DMA requests are hard-coded to level sensitive during integration in OMAP2420. This behavior corresponds to TX DMA request configured in pulse sensitive. The TX_REQ bit must be set to 1 to make the TX DMA request acknowledged by the system DMA. Setting this bit to 0 is not compliant with system DMA integration and does not work. 0x0: TX DMA request active level − does not work in OMAP2420 0x1: TX DMA request active pulse	RW	0
11:8	TXDMA2_EP	Transmit endpoint number for DMA channel 2. The endpoint number binary encoded in these four bits is the current transmit endpoint selected for DMA channel 2. A zero value indicates that the DMA channel 2 is de-activated. Any other value automatically enables transmit DMA transfer for the selected endpoint. 0x0: Transmit DMA channel 2 is deactivated. 0x1: EP1 0x2: EP2 0x3: EP3 0x4: EP4 0x5: EP5 0x6: EP6 0x7: EP7	RW	0x0

Table 22−82. TXDMA_CFG (Continued)

Bits	Field Name	Description	Type	Reset
11:8 (cont;d)	TXDMA2_EP (cont;d)	0x8: EP8 0x9: EP9 0xA: EP10 0xB: EP11 0xC: EP12 0xD: EP13 0xE: EP14 0xF: EP15		
7:4	TXDMA1_EP	Transmit endpoint number for DMA channel 1. The endpoint number binary encoded in these four bits is the current transmit endpoint selected for DMA channel 1. A zero value indicates that the DMA channel 1 is de-activated. Any other value automatically enables transmit DMA transfer for the selected endpoint. 0x0: Transmit DMA channel 1 is deactivated. 0x1: EP1 0x2: EP2 0x3: EP3 0x4: EP4 0x5: EP5 0x6: EP6 0x7: EP7 0x8: EP8 0x9: EP9 0xA: EP10 0xB: EP11 0xC: EP12 0xD: EP13 0xE: EP14 0xF: EP15	RW	0x0
3:0	TXDMA0_EP	Transmit endpoint number for DMA channel 0. The endpoint number binary encoded in these four bits is the current transmit endpoint selected for DMA channel 0. A zero value indicates that the DMA channel 0 is de-activated. Any other value automatically enables transmit DMA transfer for the selected endpoint. 0x0: Transmit DMA channel 0 is deactivated. 0x1: EP1 0x2: EP2 0x3: EP3	RW	0x0

Table 22–82. TXDMA_CFG (Continued)

3:0	TXDMA0_EP			RW	0x0
(cont;d)	(cont;d)	0x4:	EP4		
		0x5:	EP5		
		0x6:	EP6		
		0x7:	EP7		
		0x8:	EP8		
		0x9:	EP9		
		0xA:	EP10		
		0xB:	EP11		
		0xC:	EP12		
		0xD:	EP13		
		0xE:	EP14		
		0xF:	EP15		

Table 22–83. DATA_DMA

Address Offset	0x248		
Physical Address	0x4805 E248	**Instance**	USBOT1
Description	DMA FIFO data register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA_DMA															

Bits	Field Name	Description	Type	Reset
15:0	DATA_DMA	DMA FIFO data. **Warning:** It is possible for both an RX DMA request and a TX DMA request to be active at the same time. In this case, the main DMA controller engine can access both transmit endpoint and receive endpoint FIFO. A read access to DATA_DMA register affects the endpoint that caused the RX DMA request to be active, and a write access affects the endpoint that caused the TX DMA request to be active. **Caution:** The local host must not access this register directly; however, there is no hardware mechanism that would protect from such access. No access into this register must be done out of DMA requests handling.	RW	0x----

Table 22−84. TXDMA0...TXDMA2

Address Offset	0x250-0x258 in 0x4 byte increments		
Physical Address	0x4805 E250-0x4805 E258	**Instance**	USBOT1
Description	Transmit DMA control register n		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXN_EOT	TXN_START	Reserved				TXN_TSC									

Bits	Field Name	Description	Type	Reset
15	TXN_EOT	Transmit DMA channel n end of transfer. Can be either 0 or 1 for BULK DMA transfer. When set by the local host, it signals to the core that the transfer size set in TXDMAn.TXn_TSC is in bytes. A TX done interrupt (IRQ_SRC.TXn_Done) is asserted with the last IN transaction. Note that if the number of bytes set in TXDMAn.TXn_TSC is a multiple of the endpoint _s buffer size, the TX done interrupt is asserted only after an IN transaction with an empty data packet. When cleared, the transfer size set in TXn_TSC is in full buffer size for the endpoint selected (BULK only). A TX done interrupt is asserted when the last buffer is sent with the last IN transaction. This mode is to be used for a partial bulk transfer of a large file exceeding 1023 bytes. This register bit is reset after all data is transmitted to the USB host. 0x0: DMA transfer size is in buffers. 0x1: DMA transfer size is in bytes.	RW	0
14	TXN_START	Transmit DMA channel n start. Set by the local host to tell the device that the main DMA system is ready to transmit the number of bytes or buffers. Once set, the DMA transfer cannot be interrupted, except if the local host clears endpoint in TXDMA_CFG register. The IRQ_SRC.TXn_DONE interrupt bit is asserted when the DMA transfer ends. Always reads 0x0. 0x0: No action 0x1: DMA transfer start	RW	0

Table 22−84. TXDMA0...TXDMA2 (Continued)

Bits	Field Name	Description	Type	Reset
13:10	Reserved	Reserved	RW	0x-
9:0	TXN_TSC	Transmit DMA channel n transfer size counter. The binary encoded value from 0 to 1023, written by the local host into this register, corresponds to the number of bytes or number of buffer transfers (function of TXDMAn.TXn_EOT) which is transmitted by the transmit DMA channel n. When read, the register reflects the number of bytes/buffers the USB device has still to transmit. This read mode is only provided for software debug purpose. **Caution:** For ISO transfer, you must verify that the set value does not exceed the ISO FIFO size for the endpoint. There is no hardware mechanism to protect from this situation. If it happens, results are unpredictable. For bulk transfer, when TXDMAn.TXn_EOT = 0, a set value of TXDMAn.TXn_TSC = 0 means 1024 buffers and not 0. The counter then operates in the following way: 000, 3FF, 3FE, &001, 000, stop. When TXDMAn.TXn_EOT = 1, a set value of TXDMAn.TXn_TSC = 0 a NULL packet is sent in response to the next IN token.	RW	0x000

Table 22−85. RXDMA0...RXDMA2

Address Offset	0x260-0x268 in 0x4 byte increments		
Physical Address	0x4805 E260-0x4805 E268	**Instance**	USBOT1
Description	Receive DMA control register n		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXN_STOP	Reserved							RXN_TC							

Table 22−85. RXDMA0...RXDMA2 (Continued)

Bits	Field Name	Description	Type	Reset
15	RXN_STOP	Receive DMA channel n transfer stop. When set, an RXn_EOT interrupt is asserted to the local host after n OUT transactions where n is the encoded binary value + 1 programmed into RXDMAn.RXn_TC field. This register is used when no smaller than buffer size packet is received at an end-of-transfer (EOT), and the local host expects a given amount of data for the transfer. **Caution:** At end of transfer, the USB FIFO is disabled and all OUT transactions received to the assigned endpoint will be sent NAK by the core. The Local Host must set CTRL.Set_FIFO_En for the Endpoint to re-enable the endpoint OUT transactions. In double buffer mode, if RXn_Stop bit is enabled, EOT may be generated whereas RX FIFO is not empty. Because RX FIFO cannot be re-enabled while a DMA request is active (that is, SET_FIFO_EN is not taken into account while DMA request is being serviced), the software must repeat the re-enable action until FIFO_EN indicates the effectiveness of the command.	RW	0
14:8	Reserved	Reserved	RW	0x--
7:0	RXN_TC	Receive DMA channel n transactions count. The local host can ask for an interrupt each n OUT transactions where n is the encoded binary value + 1 programmed into RXDMAn.RXn_TC field. This register must be programmed to the desired transactions watermark limit prior to enabling the DMA transfer for the receive DMA channel n. **Caution:** A reached watermark does not disable an active DMA transfer if RXDMAn.RXn_Stop was not set. If RXDMAn.RXn_Stop was set for the transfer, both RXn_CNT and RXn_EOT interrupts are asserted. A read to that register returns the number of transactions remaining before the IRQ_SRC.RXn_CNT interrupt flag is asserted. This read mode is only provided for software debug purpose.	RW	0x00

Table 22−86. EP0

Address Offset	0x280		
Physical Address	0x4805 E280	**Instance**	USBOT1
Description	Endpoint 0 configuration register		
Type	RW		

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">RESERVED</td><td colspan="2">EP0_SIZE</td><td>RESERVED</td><td colspan="11">EP0_PTR</td></tr>
</table>

Table 22–86. EP0 (Continued)

Bits	Field Name	Description	Type	Reset
15:14	Reserved	Reserved	RW	0x-
13:12	EP0_SIZE	Endpoint 0 FIFO size. Contains the endpoint 0 FIFO size value and must match the value sent by the local host to the USB host during the GET_DEVICE_DESCRIPTOR request preceding configuration phase. Status flags (STAT_FLG.NON_ISO_FIFO_EMPTY, STAT_FLG.NON_ISO_FIFO_FULL), overrun and underrun conditions are based on this value for all IN and OUT transactions to endpoint 0. Note: The local host must fill this field before setting CFG_LOCK bit SYSCON1[8] . 0x0: 8 bytes 0x1: 16 bytes 0x2: 32 bytes 0x3: 64 bytes	RW	0x-
11	Reserved	Reserved	RW	-
10:0	EP0_PTR	Endpoint 0 pointer. Contains the address of the endpoint 0 pointer. Value 0x000 is forbidden (reserved for setup FIFO). **Note:** Pointer value must be set to a value < 0xFF because the memory size is 2K bytes and a pointer coded value = 0xFF corresponds to 2040 bytes. Addressing upper bytes issue in memory overlap.	RW	0x---

Table 22–87. EP_RX1...EP_RX15

Address Offset	0x284-0x2BC in 0x4 byte increments		
Physical Address	0x4805 E284-0x4805 E2BC	**Instance**	USBOT1
Description	Receive endpoint n configuration register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPN_RX_VALID	EPN_RX_SIZE_DB	EPN_RX_SIZE		EPN_RX_ISO	EPN_RX_PTR										

Table 22−87. EP_RX1...EP_RX15 (Continued)

Bits	Field Name	Description	Type	Reset
15	EPN_RX_VALID	Receive endpoint n valid bit. Must be set by the local host to allow receive endpoint n to be used for USB transfers as part of the device configuration. If not set, all transactions to this endpoint are ignored by the core. 0x0: Receive endpoint n does not exist for this configuration. 0x1: Receive endpoint n is part of the device configuration.	RW	-
14	EPN_RX_SIZE_ DB	Receive non-ISO (or ISO) endpoint n double-buffer (DB). Must be set by the local host to allow double buffering for receive Non-ISO endpoint n. This is used to reduce number of transactions resulting in NAK handshake. **Note:** This bit is only for non-ISO endpoints. For ISO endpoints, which are always double-buffered, this bit is Endpoint _s Size MSB. 0x0: No double buffer for non-ISO receive endpoint n 0x1: Double buffer used for non-ISO receive endpoint n	RW	-
13:12	EPN_RX_SIZE	Receive endpoint n size field. Contains the endpoint n FIFO size value. Status flags (STAT_FLG.NON_ISO_FIFO_EMPTY, STAT_FLG.NON_ISO_FIFO_FULL, STAT_FLG. ISO_FIFO_EMPTY, STAT_FLG.ISO_FIFO_FULL), overrun, and underrun conditions are based on this value for all OUT transactions to endpoint n. **Note:** This paragraph includes description of EPn_RX.[14] bit for ISO endpoints. **Warning:** For ISO endpoints, a size of 1023 bytes takes the whole memory and then cannot be programmed with a 2K bytes USB W2FC. 0x0: 8 bytes in non-ISO mode or ISO mode and EPn_RX_SIZE_DB = 0 else if ISO mode and EPn_RX_SIZE_DB=1 128 bytes 0x1: 16 bytes in non-ISO mode or ISO mode and EPn_RX_SIZE_DB = 0 else if ISO mode and EPn_RX_SIZE_DB=1 256 bytes 0x2: 32 bytes in non-ISO mode or ISO mode and EPn_RX_SIZE_DB = 0 else if ISO mode and EPn_RX_SIZE_DB=1 512 bytes 0x3: 64 bytes in non-ISO mode or ISO mode and EPn_RX_SIZE_DB = 0 else if ISO mode and EPn_RX_SIZE_DB=1 1023 bytes	RW	0x-
11	EPN_RX_ISO	Receive ISO endpoint n field. Must be set if the receive endpoint n type is isochronous in the desired device configuration. If not set, the endpoint type is bulk or interrupt (the hardware does not distinguish bulk type from interrupt). 0x0: Receive endpoint n type is bulk or interrupt. 0x1: Receive endpoint n type is isochronous.	RW	-

Table 22–87. EP_RX1...EP_RX15 (Continued)

Bits	Field Name	Description	Type	Reset
10:0	EPN_RX_PTR	Receive endpoint n pointer field. Contains the address of the receive endpoint n pointer. Value 0x000 is forbidden (reserved for setup FIFO). **Caution:** For ISO endpoints or for non-ISO endpoints that allow double-buffering, 2*RX buffer size must be reserved for ping-pong. **Note:** Pointer value must be set to a value < 0xFF because the memory size is 2K bytes and a pointer coded value = 0xFF correspond to 2040 bytes. Addressing upper bytes issue in memory overlap. 0x0: Address = BASE (forbidden) 0x1: Address = BASE + 8 bytes 0x2: Address = BASE + 16 bytes 0x3: Address = BASE + 24 bytes 0xFF: Address = BASE + 2040 bytes	RW	0x---

Table 22–88. EP_TX1...EP_TX15

Address Offset	0x2C4-0x2FC in 0x4 byte increments		
Physical Address	0x4805 E2C4-0x4805 E2FC	**Instance**	USBOT1
Description	Transmit endpoint n configuration register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPN_TX_VALID	EPN_TX_SIZE_DB	EPN_TX_SIZE		EPN_TX_ISO	EPN_TX_PTR										

Table 22−88. EP_TX1...EP_TX15 (Continued)

Bits	Field Name	Description	Type	Reset
15	EPN_TX_VALID	The transmit endpoint n valid bit. Must be set by the local host to allow transmit endpoint n to be used for USB transfers as part of the device configuration. If not set, all transactions to this endpoint are ignored by the core. 0x0: Transmit endpoint n does not exist for this configuration. 0x1: Transmit endpoint n is part of the device configuration.	RW	-
14	EPN_TX_SIZE_DB	Transmit non-ISO (or ISO) endpoint n double buffer. Must be set by the local host to allow double buffering for transmit non-ISO Endpoint n, when used in a DMA transfer. This is used to reduce number of transactions resulting in NAK handshake. **Note:** This bit is only for non-ISO endpoints used in DMA mode. For ISO endpoints, which are always double-buffered, this bit is Endpoint _s Size MSB. For non-ISO endpoints that are not used in a DMA transfer, double-buffering is not provided. 0x0: No double buffer for non-ISO transmit endpoint n 0x1: Double buffer used for non-ISO transmit endpoint n	RW	-
13:12	EPN_TX_SIZE	Transmit endpoint n size. Contains the endpoint n FIFO size value. Status flags (FIFO_Empty, FIFO_Full), and underrun condition are based on this value for all IN transactions to endpoint n. **Note:** EPn_TX.[14] bit description only applies to ISO endpoints. 0x0: 8 bytes in non-ISO mode or ISO mode and EPn_TX_SIZE_DB = 0 else if ISO mode and ETXn_RX_SIZE_DB=1 128 bytes 0x1: 16 bytes in non-ISO mode or ISO mode and EPn_TX_SIZE_DB = 0 else if ISO mode and ETXn_RX_SIZE_DB=1 256 bytes 0x2: 32 bytes in non-ISO mode or ISO mode and EPn_TX_SIZE_DB = 0 else if ISO mode and ETXn_RX_SIZE_DB=1 512 bytes 0x3: 64 bytes in non-ISO mode or ISO mode and EPn_TX_SIZE_DB = 0 else if ISO mode and ETXn_RX_SIZE_DB=1 1023 bytes	RW	0x-
11	EPN_TX_ISO	Transmit ISO endpoint n field. Must be set if the transmit endpoint n type is isochronous in the desired device configuration. If not set, the endpoint type is bulk or interrupt (the hardware does not distinguish bulk type from interrupt). 0x0: Transmit endpoint n type is bulk or interrupt. 0x1: Transmit endpoint n type is isochronous.	RW	-

Note: The USB device controller must fill this field before setting the CFG_LOCK bit SYSCON1[8] and must not change the values once the CFG_LOCK bit SYSCON1[8] is set.

Table 22−88. EP_TX1...EP_TX15 (Continued)

Bits	Field Name	Description	Type	Reset
10:0	EPN_TX_PTR	Transmit endpoint n pointer. Contains the address of the transmit endpoint n pointer. **Caution:** For ISO endpoints or for non-ISO endpoints that allow double-buffering, 2*TX buffer size must be reserved for ping-pong. **Note:** Pointer value must be set to a value < 0xFF because the memory size is 2K bytes and a pointer coded value = 0xFF corresponds to 2040 bytes. Addressing upper bytes issue in memory overlap. 0x0: Address = BASE (forbidden) 0x1: Address = BASE + 8 bytes 0x2: Address = BASE + 16 bytes 0x3: Address = BASE + 24 bytes 0xFF: Address = BASE + 2040 bytes	RW	0x---

Note: The USB device controller must fill this field before setting the CFG_LOCK bit SYSCON1[8] and must not change the values once the CFG_LOCK bit SYSCON1[8] is set.

Table 22−89. OTG_REV

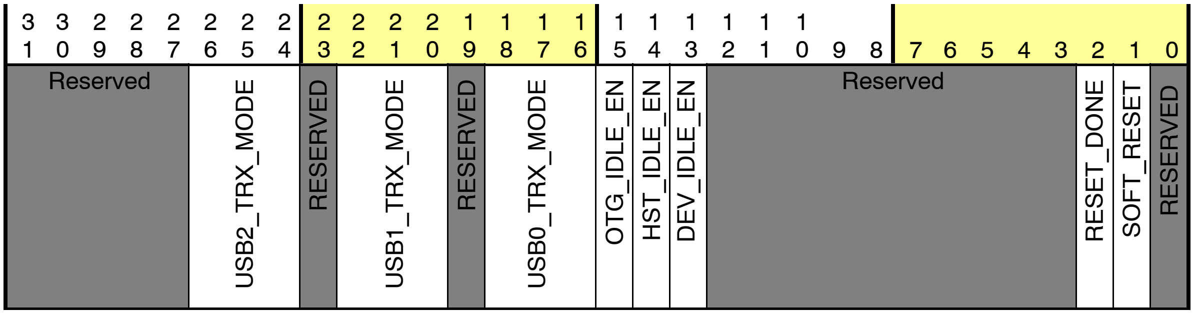
Address Offset	0x300		
Physical Address	0x4805 E300	**Instance**	USBOT1
Description	OTG revision number register		
Type	R		

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																								OTG_REV_NB							

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Reserved	R	0x------
7:0	OTG_REV_NB	OTG revision number Examples: 0x10 for 1.0 0x21 for 2.1	R	

Table 22–90. OTG_SYSCON_1

Address Offset	0x304		
Physical Address	0x4805 E304	**Instance**	USBOT1
Description	OTG system configuration register 1		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:27	Reserved	Reserved	RW	0x--
26:24	USB2_TRX_MODE	USB port 2 transceiver mode. Configure the transceiver signaling type used by the USB host, USB device, and USB OTG controllers when communicating with the transceiver connected to USB port 2. The transceiver signaling type depends on the signaling mode used by the transceiver. Must be set during the configuration of the OTG module. 0x0: Receiver only (6 pins) (not functional) 0x1: 4-pin bidirectional (VP_VM) mode transceiver signaling 0x2: 3-pin bidirectional (DAT_SE0) mode transceiver signaling 0x3: 6-pin unidirectional transceiver signaling 0x4: Can be used to force output values 0x5: Can be used to force output values 0x6: Can be used to force output values 0x7: Can be used to force output values	RW	0x0
23	Reserved	Reserved	RW	-
22:20	USB1_TRX_MODE	USB port 1 transceiver mode. Configure the transceiver signaling type used by the USB host, USB device, and USB OTG controllers when communicating with the transceiver connected to USB port 1. The transceiver signaling type depends on the signaling mode used by the transceiver. Must be set during the configuration of the OTG module. 0x0: Receiver only (6 pins) (not functional) 0x1: 4-pin bidirectional (VP_VM) mode transceiver signaling 0x2: 3-pin bidirectional (DAT_SE0) mode transceiver signaling	RW	0x0

Table 22−90. OTG_SYSCON_1 (Continued)

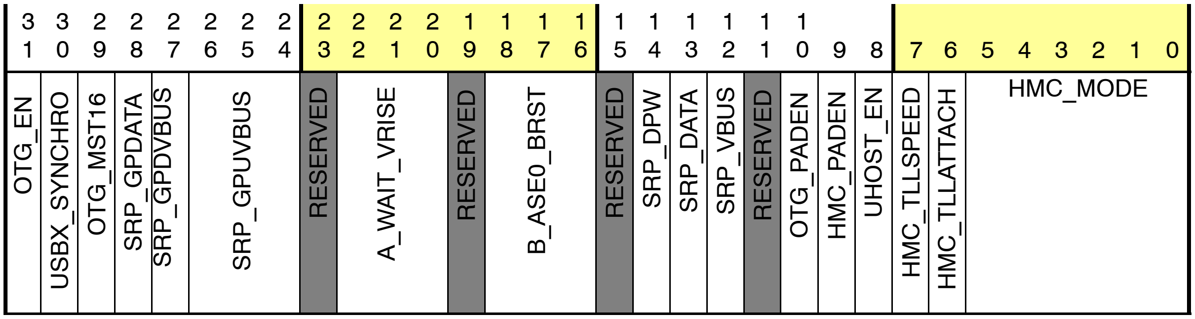
Bits	Field Name	Description	Type	Reset
22:20	USB1_TRX_ MODE	0x3: 6-pin unidirectional transceiver signaling 0x4: Can be used to force output values		
19	Reserved	Reserved	RW	-
18:16	USB0_TRX_ MODE	USB port 0 transceiver mode. Configure the transceiver signaling type used by the USB host, USB device, and USB OTG controllers when communicating with the transceiver connected to USB port 0. The transceiver signaling type depends on the signaling mode used by the transceiver. Must be set during the configuration of the OTG module. 0x0: Receiver only (6 pins) (not functional) 0x1: 4-pin bidirectional (VP_VM) mode transceiver signaling 0x2: 3-pin bidirectional (DAT_SE0) mode transceiver signaling 0x3: 6-pin unidirectional transceiver signaling 0x4: Can be used to force output values 0x5: Can be used to force output values 0x6: Can be used to force output values 0x7: Can be used to force output values	RW	0x0
15	OTG_IDLE_EN	OTG controller clock-gating control. Disables the clock to the OTG controller. Used to activate the OTG power saving circuitry. 0x0: The OTG controller clock is not gated (default value, free-running clock). 0x1: The OTG controller is idled by disabling the OTG controller.	RW	0
14	HST_IDLE_EN	Host controller clock-gating control	RW	0
13	DEV_IDLE_EN	Device controller clock-gating control. Used to activate the USB device power saving circuitry. 0x0: The USB device controller clock is not gated (default value, free-running clock). 0x1: The USB device controller clock is disabled.	RW	0
12:3	Reserved	Reserved	RW	0x---
2	RESET_DONE	Reset status information. Reflects the reset status of the module (hardware and software reset, internal reset monitoring of the USB device and the USB OTG controller sections). After a hardware or software reset, this bit must be checked to determine that the reset has completed. 0x0: Internal reset for OTG is ongoing (includes OTG controller and USB device). 0x1: Reset completed	R	0

Table 22−90. OTG_SYSCON_1 (Continued)

Bits	Field Name	Description	Type	Reset
1	SOFT_RESET	Software reset for the OTG, device, and host controllers. 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	Reserved	Reserved	RW	-

Table 22−91. OTG_SYSCON_2

Address Offset	0x308		
Physical Address	0x4805 E308	**Instance**	USBOT1
Description	OTG system configuration 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
OTG_EN	USBX_SYNCHRO	OTG_MST16	SRP_GPDATA	SRP_GPDVBUS	SRP_GPUVBUS			RESERVED	A_WAIT_VRISE			RESERVED	B_ASE0_BRST			RESERVED	SRP_DPW	SRP_DATA	SRP_VBUS	RESERVED	OTG_PADEN	HMC_PADEN	UHOST_EN	HMC_TLLSPEED	HMC_TLLATTACH	HMC_MODE					

Bits	Field Name	Description	Type	Reset
31	OTG_EN	OTG enable bit. Selects the OTG controller functionality. 0x0: The OTG controller circuitry is not activated. 0x1: The OTG controller circuitry is activated. The OTG state-machine (HNP) can be used on one of the USB ports.	RW	0
30	USBX_ SYNCHRO	USB output signals synchronized bit. Enables to synchronize (or not) the USB output signals. Must be set for USB use to guarantee USB signal balancing. 0x0: USB signals not synchronized 0x1: USB signals synchronized (on all USB ports)	RW	0
29	OTG_MST16	Enables compatibility with the 16-bit OTG master controller. Must be set to 0 to allow proper USB host controller access to system memory. 0x0: Host controller is connected to system memory via a 32-bit bus. 0x1: Host controller is connected to system memory via a 16-bit bus. OMAP does not support this setting.	RW	0

Table 22−91. OTG_SYSCON_2 (Continued)

Bits	Field Name	Description	Type	Reset
28	SRP_GPDATA	Session request protocol generation pulse width on D+. Enables to generate different D+ pullup periods during the DATA line pulsing process. Only used when the OTG controller acts as an OTG default-B device and asks for an SRP process (OTG_CTRL.B_BUSREQ). Only used when connected to the OTG cable as a B-device (OTG_CTRL.ID = 1). The SRP process first generates a DATA line pulsing and then a VBUS pulsing. 0x0: Data line pulse during SRP is 6 ms (nominal). 0x1: Data line pulse during SRP is 9 ms (nominal).	RW	0
27	SRP_GPDVBUS	Session request protocol generation discharge VBUS enable. Enables to activate or not the VBUS discharge functionality after any VBUS charge process. When activated, the OTG pulldown VBUS is activated at the end of the pullup VBUS process of the SRP (VBUS pulsing). Only used when connected to the OTG CABLE as an A-device (OTG_CTRL.ID = 0). 0x0: VBUS is not discharged after the SRP VBUS pulse. 0x1: VBUS is discharged after the SRP VBUS pulse.	RW	0
26:24	SRP_GPUVBUS	Session request protocol generation charge VBUS duration. Enables to generate different VBUS charge timings during the VBUS pulsing process. Only used when the OTG acts as an OTG default-B device and asks for an SRP process (OTG_CTRL.B_BUSREQ). Only used when connected to the OTG cable as a B-device (OTG_CTRL.ID = 1). Must be set in the function of the hardware implementation. The value must ensure that VBUS rises higher than 2 V to guarantee an A-device SRP detection. The value is also used to generate the discharge on VBUS (OTG_SYSCON_2.SRP_GPDVBUS) when enabled. The interrupt OTG_IRQ_SRC.B_SRP_DONE is generated after the additional time of: data pulsing + 2*VBUS (charge and discharge, even if the discharge of VBUS is not activated). 0x0: Does not issue a VBUS charge pulse as part of its SRP generation sequence. 0x1: VBUS is charged for 0.5 ms nominal. 0x2: During SRP generation, VBUS is charged for 1 ms nominal. 0x3: During SRP generation, VBUS is charged for 2 ms nominal. 0x4: During SRP generation, VBUS is charged for 4 ms nominal.	RW	0x0

Table 22−91. OTG_SYSCON_2 (Continued)

Bits	Field Name	Description	Type	Reset
26:24	SRP_GPUVBUS	0x5: During SRP generation, VBUS is charged for 6 ms nominal. 0x6: During SRP generation, VBUS is charged for 10 ms nominal. 0x7: During SRP generation, VBUS is charged for 40 ms nominal.		
23	Reserved	Reserved	RW	-
22:20	A_WAIT_VRISE	Offset for A_WAIT_VRISE timer. Check the delay between the decision of the A-device to drive the bus and th moment the A-device sees VBUS rising above the VBUS valid threshold. Only used when connected to the OTG cable as A-device (OTG_CTRL.ID = 0). Must not be changed when OTG_EN = 1 for proper operation. The A_WAIT_VRISE timeout is generated when the VBUS valid voltage is not active after the above timer rolls down to 0. This timer is activated when the A-device requests the bus or when an SRP is detected. In the case in which the counter rolls down to 0, the OTG_IRQ_SRC.A_VBUS_ERR interrupt is generated to alert the user that the power has not reached the USB valid voltage in time. 0x0: Maximum delay in A_WAIT_VRISE state is 100 ms nominal. 0x1: Maximum delay in A_WAIT_VRISE state is 140 ms nominal. 0x2: Maximum delay in A_WAIT_VRISE state is 200 ms nominal. 0x3: Maximum delay in A_WAIT_VRISE state is 548.6 ms nominal.	RW	0x0
19	Reserved	Reserved	RW	-

Table 22–91. OTG_SYSCON_2 (Continued)

Bits	Field Name	Description	Type	Reset
18:16	B_ASE0_BRST	Offset for B_ASE0_BRST timer. Controls how the OTG controller manages disabling the D+ pullup when acting as a default-B OTG device and transitioning from state b_peripheral to b_wait_acon, and when acting as a default-A OTG device and transitioning from state a_peripheral to a_wait_bcon. The setting of this register controls the mechanism by which the OTG controller knows that the D+ pullup is disabled. Because OMAP implementations control the D+ pullup using an I2C link to an OTG transceiver, the only allowed setting is 0x4. When in this mode of operation, system software must write 1 to OTG_CTRL.OTG_PU immediately after completion of the I2C operation that disables the D+ pullup. Must not be changed when OTG_EN = 1 for proper operation. The B_ASE0_BRST timeout is generated during the HNP process when the B-device waits for an A-device pullup connection. In the case in which the counter rolls down to 0, the OTG_IRQ_SRC.B_HNP_FAIL interrupt is generated to alert the user that the host has not processed the HNP. 0x0: Maximum delay in B_ASE0_BRST state is 3.125 ms nominal. 0x1: Maximum delay in B_ASE0_BRST state is 3.725 ms nominal. 0x2: Maximum delay in B_ASE0_BRST state is 5.195 ms nominal. 0x3: Maximum delay in B_ASE0_BRST state is 7.925 ms nominal. 0x4: Wait indefinitely until an acknowledge 1 is written to OTG_CTRL:OTG_PU.	RW	0x0
15	Reserved	Reserved	RW	-
14	SRP_DPW	Session request protocol detection _pulse width. Selects the width for the SRP detection. This value applies on both DATA and VBUS pulsing detections. Only used when connected to the OTG cable as A-device (OTG_CTRL.ID = 0). 0x0: SRP pulse must be greater than 1 ms (nominal) to be sensed as a valid D+, D–, or VBUS SRP pulse. 0x1: SRP pulse must be greater than 167 ns (nominal) to be sensed as a valid D+, D–, or VBUS SRP pulse.	RW	0

Table 22−91. OTG_SYSCON_2 (Continued)

Bits	Field Name	Description	Type	Reset
13	SRP_DATA	Session request protocol detection _data pulsing detection enable. Determines whether the OTG controller considers D+ or D− pulses as SRP requests. Only used when connected to the OTG cable as A-device (OTG_CTRL.ID = 0). SRP detection requires that either OTG_SYSCON_2.SRP_DATA or OTG_SYSCON_2.SRP_VBUS (or both) is set. 0x0: OTG controller does not consider D+ or D pulses as SRP requests. 0x1: OTG controller considers D+ or D pulses as SRP requests.	RW	0
12	SRP_VBUS	Session request protocol detection _VBUS pulsing detection enable. Determines whether the OTG controller considers VBUS pulses as SRP requests. Only used when connected to the OTG cable as A-device (OTG_CTRL.ID = 0). SRP detection requires that either OTG_SYSCON_2.SRP_DATA or OTG_SYSCON_2.SRP_VBUS (or both) is set. 0x0: OTG controller does not consider VBUS pulses as SRP requests. 0x1: OTG controller considers VBUS pulses as SRP requests.	RW	0
11	Reserved	Reserved	RW	-
10	OTG_PADEN	OTG transceiver-control and status-information selector. Determines how ID, VBUSVLD, BSESSVLD, BSESSEND, and ASESSEND register bits are controlled. When using OTG controller functionality to implement an OTG dual-role device, must be set to 0. In this case, those OTG_CTRL bits are read/write and controlled by system software.	RW	0
9	HMC_PADEN	USB pin multiplexing control selector 0x0: OTG_SYSCON_2 provides the controls. 0x1: Invalid in OMAP2420—Do not use this setting.	RW	0

Table 22–91. OTG_SYSCON_2 (Continued)

Bits	Field Name	Description	Type	Reset
8	UHOST_EN	Host USB controller enable bit. Controls the clock enable and hardware reset for the USB host controller when HMC_PADEN is 0. When HMC_PADEN is 1, writes to this bit have no effect on the clock enable. A read of this register gives the internal value used for UHOST_EN regardless of the setting of HMC_PADEN. 0x0: USB host controller is not clocked and is held in hardware reset. 0x1: USB host controller is clocked and is not held in hardware reset.	RW	1
7	HMC_TLLSPEED	HMC TLL SPEED configuration bit. Controls the way that the OMAP transceiverless link logic models the speed of the transceiverless link when HMC_PADEN is 0. When HMC_PADEN is 1, has no effect on the transceiverless link logic and writes to this bit have no effect. Proper operation of the transceiverless link logic requires that he USB host controller is clocked when the transceiverless link logic is used, even if the transceiverless link logic is not connected to a USB host controller port. 0x0: Transceiverless link logic models a low-speed (1.5M bits-per-second) USB link. 0x1: Transceiverless link logic models a full-speed (12M bits-per-second) USB link.	RW	0
6	HMC_ TLLATTACH	HMC TLL ATTACH configuration bit. Controls the OMAP transceiverless link logic attach/detach status when HMC_PADEN is 0. When HMC_PADEN is 1, has no effect on the transceiverless link logic and writes to this bit have no effect. Proper operation of the transceiverless link logic requires that the USB host controller is clocked when the transceiverless link logic is used, even if the transceiverless link logic is not connected to a USB host controller port. 0x0: Transceiverless link logic models a detached link. 0x1: Transceiverless link logic models an attached link.	RW	0
5:0	HMC_MODE	HMC mode configuration bits. Control the OMAP USB signal multiplexing selection when HMC_PADEN is 0. When HMC_PADEN is 1, is unused and writes to this register have no effect.	RW	0x00

Table 22−92. OTG_CTRL

Address Offset	0x30C		
Physical Address	0x4805 E30C	**Instance**	USBOT1
Description	OTG 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		USB2_EN	USB2_DP	USB2_DM	USB1_EN	USB1_DP	USB1_DM	USB0_EN	USB0_DP	USB0_DM	ASESSVLD	BSESSEND	BSESSVLD Reserved	VBUSVLD Reserved	ID Reserved	DRIVER_SEL	Reserved		A_SETB_HNPEN	A_BUSREQ	Reserved	B_HNPEN	B_BUSREQ	OTG_BUSDROP	Reserved	OTG_PD	OTG_PU	OTG_DRV_VBUS	OTG_PD_VBUS	OTG_PU_VBUS	OTG_PU_ID

Bits	Field Name	Description	Type	Reset
31:30	Reserved	Reserved	RW	0x---
29	USB2_EN	USB port 2 – enable information	RW	–
28	USB2_DP	USB port 2 – D+ information	RW	–
27	USB2_DM	USB port 2 – D– information	RW	–
26	USB1_EN	USB port 1 – enable information	RW	–
25	USB1_DP	USB port 1 – D+ information	RW	–
24	USB1_DM	USB port 1 – D– information	RW	–
23	USB0_EN	USB port 0 – enable information	RW	–
22	USB0_DP	USB port 0 – D+ information	RW	–
21	USB0_DM	USB port 0 – D– information	RW	–
20	ASESSVLD	Current A-device session valid value. When OTG is enabled (OTG_EN = 1), must be programmed to reflect the OTG transceiver VBUS voltage comparator that compares against VA_SESS_VLD. When VBUS is above VA_SESS_VLD, must be set to 1. When VBUS is below VA_SESS_VLD, must be set to 0. This information is only relevant when connected as A-device (OTG_CTRL.ID = 0) and OTG is enabled (OTG_EN = 1). 0x0: VBUS voltage is below VA_SESS_VLD. 0x1: VBUS voltage is above VA_SESS_VLD.	RW	0
19	BSESSEND	Current B-device session end value. When OTG is enabled (OTG_EN = 1), must be programmed to reflect the OTG transceiver VBUS voltage comparator that compares against VA_SESS_END. When VBUS is below VA_SESS_END must be set to 1. When VBUS is above VA_SESS_END, must be set to 0. This information is only relevant when connected as B-device (OTG_CTRL.ID = 1) and OTG is enabled (OTG_EN = 1). 0x0: VBUS voltage is above VB_SESS_END. 0x1: VBUS voltage is below VB_SESS_END.	RW	1

Table 22–92. OTG_CTRL (Continued)

Bits	Field Name	Description	Type	Reset
18	BSESSVLD	Current B-device session valid value. When OTG is enabled (OTG_EN = 1) and OMAP is connected as a dual-role B-device (OTG_CTRL.ID = 1), must be programmed to reflect the OTG transceiver VBUS voltage comparator that compares against VB_SESS_VLD. When VBUS is above VB_SESS_VLD must be set to 1. When VBUS is below VB_SESS_VLD, must be set to 0. When OTG is disabled, OTG_PADEN is 0, and the USB device controller is being used, this bit must be programmed to reflect whether VBUS is high enough to allow the OMAP USB device controller to function properly. 0x0: VBUS voltage is below VB_SESS_VLD. 0x1: VBUS voltage is above VB_SESS_VLD.	RW	0
17	VBUSVLD	Current VBUS valid value. When OTG is enabled (OTG_EN = 1), must be programmed to reflect the OTG transceiver VBUS voltage comparator that compares against VA_VBUS_VLD. When VBUS is above VA_VBUS_VLD, must be set to 1. When VBUS is below VA_VBUS_VLD, must be set to 0. Has no meaning when acting as OTG default-B device (ID = 1 and OTG_EN =1). 0x0: VBUS voltage is below VA_VBUS_VLD (if OTG is enabled), or VBUS is insufficient to allow OMAP USB device controller functionality (if OTG is disabled and OTG_PADEN is 0). 0x1: VBUS voltage is above VA_VBUS_VLD (if OTG is enabled), or VBUS is sufficient to allow OMAP USB device controller functionality (if OTG is disabled and OTG_PADEN is 0).	RW	0
16	ID	Current ID pin value. When OTG is enabled (OTG_EN = 1), system software must program this bit to reflect the OTG transceiver ID pin sensor status any time ID changes. When OTG is disabled (OTG_EN = 0), has no meaning. The OTG controller does not understand resistive ID connectivity as can be found in a car kit environment. When an OTG transceiver sees ID pin resistively tied, OTG is not possible and OTG_EN must not be set. 0x0: ID pin is grounded. 0x1: ID pin is high-impedance.	RW	1
15	DRIVER_SEL	Active controller/driver software. When OTG is enabled (OTG_EN = 1), determines which OMAP USB controller (and therefore software driver) has ownership of the OTG link. When HNP occurs, the OTG controller updates this bit and issues a driver change interrupt. When OTG is disabled (OTG_EN = 0), has no meaning. When a write changes OTG_EN from 0 to 1, this bit is updated to reflect the value of OTG_CTRL.ID. 0x0: Host driver has control of the OTG link. 0x1: Device driver has control of the OTG link.	RW	1
14:13	Reserved	Reserved	RW	0x-

Table 22−92. OTG_CTRL (Continued)

Bits	Field Name	Description	Type	Reset
12	A_SETB_HNPEN	B-device HNP indication (when acting as default A-device). When OTG is enabled (OTG_EN = 1) and OMAP acts as default-A device (OTG_CTRL.ID = 0), this bit must be programmed to reflect whether or not the B-device has been enabled to issue HNP. Has no effect when OMAP acts as default-B device (ID =1) or when OTG is disabled (OTG_EN = 0). 0: The B-device has not been enabled to issue HNP. The OMAP1710 OTG controller does not respond to HNP issued by the B-device. 1: The B-device has been enabled to issue HNP. The OMAP1710 OTG controller responds to HNP issued by the B-device. 0x0: The B-device has not been enabled to issue HNP. The OMAP OTG controller does not respond to HNP issued by the B-device. 0x1: The B-device is enabled to issue HNP. The OMAP OTG controller responds to HNP issued by the B-device.	RW	0
11	A_BUSREQ	Bus request (when acting as default-A device). When OTG is enabled (OTG_EN = 1) and OMAP is acting as default-A device (OTG_CTRL.ID = 0), system software must set this bit when it wishes to begin an OTG session. When acting as the OTG A-device, system software can suspend the appropriate USB host controller port and then clear this bit to allow HNP requests from the B-device. If no HNP occurs, system software can set this bit to resume ownership of the OTG link by the OMAP USB host controller. Has no effect if OTG_EN = 0 or if ID = 1. Cleared 0 when the A_REQ_TMROUT interrupt is set. 0x0: A-device does not request host role on the bus. Depending on OTG state machine state, either the session can end or the default-B device can issue an HNP. 0x1: A-device does request host role on the bus. OTG state machine begins a session (if the OTG session is ended) or attempts to return ownership of the OTG link to the OMAP USB host controller as soon as possible.	RW	0
10	Reserved	Reserved	RW	-

Table 22−92. OTG_CTRL (Continued)

Bits	Field Name	Description	Type	Reset
9	B_HNPEN	B-device HNP enable (when acting as default-B device). When OTG is enabled (OTG_EN = 1) and OMAP is acting as default-B device (OTG_CTRL.ID = 1), system software must set this bit when the USB device receives a set feature operation with feature selector B_HNP_ENABLE. System software clears this bit whenever it sends USB reset to the default-B device. Has no meaning when OTG_EN = 0 or when ID = 0. 0x0: B-device is not HNP enabled. The OTG controller does not issue HNP. 0x1: B-device is HNP enabled. The OTG controller can issue HNP when B_BUSREQ is set.	RW	0
8	B_BUSREQ	Bus request (when acting as default-B device). When OTG is enabled (OTG_EN = 1) and OMAP is acting as default-B device (OTG_CTRL.ID = 1), system software sets this bit when the application wishes to take ownership of the OTG link and begin acting as a host. When set, this allows the OTG controller to issue SRP if a session is not currently valid, and to begin HNP at the next available opportunity if HNP is enabled (OTG_CTRL.B_HNPEN = 1). This bit is ignored by the OTG controller when ID = 0. The OTG controller clears this bit if there is not a valid session and the SRP fails. The OTG controller clears this bit if there is a valid session but the HNP fails. If SRP or HNP fails, system software repeats the request several times. If the SRP or HNP request fails several times, system software must inform the user of the failure. 0x0: System software does not currently wish to begin acting as host. 0x1: System software wishes to begin acting as host. HNP and, if necessary, SRP are issued.	RW	0
7	OTG_BUSDROP	OTG bus drop request. When OTG is enabled (OTG_EN = 1) and OMAP is acting as default-A device (OTG_CTRL.ID = 0), system software requests the end of a session by setting this bit. This bit has no effect when ID = 1. 0x0: System software does not require the end of the OTG session. 0x1: System software requires the end of the OTG session.	RW	0
6	Reserved	Reserved	RW	-

Table 22−92. OTG_CTRL (Continued)

Bits	Field Name	Description	Type	Reset
5	OTG_PD	D+ pulldown enable. When OTG is enabled (OTG_EN = 1), indicates whether the OTG transceiver applies a pulldown to D+. When OTG is disabled, this bit has no meaning. This bit can be polled when the OTG_IRQ_EN.OPRT_CHG_EN = 0. When OPRT_CHG_EN = 1 and OTG_IRQ_SRC.OPRT_CHG = 1, retains its value until after OPRT_CHG is cleared. 0x0: OTG transceiver does not activate the D+ pulldown. 0x1: OTG transceiver activates the D+ pulldown.	R	0
4	OTG_PU	D+ pullup enable. When OTG is enabled (OTG_EN = 1), indicates whether the OTG transceiver applies a pullup to D+. When OTG is disabled, this bit has no meaning. This bit can be polled when the OTG_IRQ_EN.OPRT_CHG_EN = 0. When OPRT_CHG_EN = 1 and OTG_IRQ_SRC.OPRT_CHG = 1, retains its value until after OPRT_CHG is cleared. When OTG_SYSCON_2.B_ASE0_RST = 4 and acting as default-B device, and system software is performing HNP, the software must write 1 to this register when it knows that the OTG controller D+ pullup is active (typically upon completion of I2C activity). 0x0: OTG transceiver does not activate the D+ pullup. 0x1: OTG transceiver activates the D+ pullup.	RW	0
3	OTG_DRV_VBUS	VBUS drive enable. When OTG is enabled (OTG_EN = 1), indicates whether the OTG transceiver drives VBUS. When OTG is disabled (OTG_EN = 0), has no meaning. Driving VBUS is requested only when OMAP acts as a default-A device and an OTG session is needed. This bit can be polled when the OTG_IRQ_EN.OPRT_CHG_EN = 0. When OPRT_CHG_EN = 1 and OTG_IRQ_SRC.OPRT_CHG = 1, retains its value until after OPRT_CHG is cleared. 0x0: OTG transceiver does not drive VBUS. 0x1: OTG transceiver drives VBUS.	R	0

Table 22–92. OTG_CTRL (Continued)

Bits	Field Name	Description	Type	Reset
2	OTG_PD_VBUS	VBUS pulldown enable. When OTG is enabled (OTG_EN = 1), indicates whether the OTG transceiver is to discharge VBUS when generating SRP. Driving VBUS is requested only when OMAP acts as a default-A device and an OTG session is needed or is in progress. When OTG is disabled (OTG_EN = 0), has no meaning. Discharge of VBUS is requested only when OMAP is acting as default-B device, an OTG session needs to be requested, and OTG_SYSCON_2.SRT_GPDVBUS = 1. This bit can be polled when the OTG_IRQ_EN.OPRT_CHG_EN = 0. When OPRT_CHG_EN = 1 and OTG_IRQ_SRC.OPRT_CHG = 1, retains its value until after OPRT_CHG is cleared. 0x0: OTG transceiver does not discharge VBUS. 0x1: OTG transceiver discharges VBUS.	R	0
1	OTG_PU_VBUS	VBUS pullup enable. When OTG is enabled (OTG_EN = 1), indicates whether the OTG transceiver charges VBUS. Charging VBUS is only used when OMAP acts as default-B device and is performing SRP. When OTG is disabled (OTG_EN = 0), has no meaning. This bit can be polled when the OTG_IRQ_EN.OPRT_CHG_EN = 0. When OPRT_CHG_EN = 1 and OTG_IRQ_SRC.OPRT_CHG = 1, retains its value until after OPRT_CHG is cleared. 0x0: OTG transceiver does not charge VBUS. 0x1: OTG transceiver charges VBUS.	R	0
0	OTG_PU_ID	ID signal pullup enable. When OTG is enabled (OTG_EN = 1), indicates whether the OTG transceiver applies a pullup to the ID pin to assist in detection of the ID pin level. System software for implementations using typical OTG transceivers with I2C interfaces ignore this bit. This bit can be used for polling when OTG_IRQ_EN.OPRT_CHG_EN is cleared 0. Otherwise, this information is frozen when the interrupt status OTG_IRQ_SRC.OPRT_CHG is active 1. 0x0: OTG transceiver does not apply a pullup to ID. 0x1: OTG transceiver applies a pullup to ID.	R	0

Table 22−93. OTG_IRQ_EN

Address Offset	0x310		
Physical Address	0x4805 E310	**Instance**	USBOT1
Description	OTG interrupt enable register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DRIVER_SWITCH_EN	RESERVED	A_VBUS_ERR_EN	A_REQ_TMROUT_EN	A_SRP_DETECT_EN	B_HNP_FAIL_EN	B_SRP_TMROUT_EN	B_SRP_DONE_EN	B_SRP_STARTED_EN	Reserved						OPRT_CHG_EN

Bits	Field Name	Description	Type	Reset
15	DRIVER_SWITCH_EN	Driver switch interrupt enable. Enables to generate an interrupt when the OTG_IRQ_SRC:DRIVER_SWITCH bit is active. 0x0: No interrupt is generated. 0x1: An interrupt is generated.	RW	0
14	Reserved	Reserved	RW	-
13	A_VBUS_ERR_EN	A-device Vbus error interrupt enable. Enables to generate an interrupt when the OTG_IRQ_SRC:A_VBUS_ERR bit is active. 0x0: No interrupt is generated. 0x1: An interrupt is generated.	RW	0
12	A_REQ_TMROUT_EN	A-device request time-out interrupt enable. Enables to generate an Interrupt when the OTG_IRQ_SRC:A_REQ_TMROUT bit is active. 0x0: No interrupt is generated. 0x1: An interrupt is generated.	RW	0
11	A_SRP_DETECT_EN	A-device SRP detection interrupt enable. This bit enables to generate an interrupt when the OTG_IRQ_SRC:A_SRP_DETECT bit is active. 0x0: No interrupt is generated. 0x1: An interrupt is generated.	RW	0
10	B_HNP_FAIL_EN	B-device HNP failed interrupt enable. Enables to generate an interrupt when the OTG_IRQ_SRC:B_HNP_FAIL bit is active. 0x0: No interrupt is generated. 0x1: An interrupt is generated.	RW	0

Table 22−93. OTG_IRQ_EN (Continued)

Bits	Field Name	Description	Type	Reset
9	B_SRP_ TMROUT_EN	B-device SRP time-out interrupt enable. Enables to generate an interrupt when the OTG_IRQ_SRC:B_SRP_TMROUT bit is active. 0x0: No interrupt is generated. 0x1: An interrupt is generated.	RW	0
8	B_SRP_ DONE_EN	B-device SRP done interrupt enable. Enables to generate an interrupt when the OTG_IRQ_SRC:B_SRP_DONE bit is active. 0x0: No interrupt is generated. 0x1: An interrupt is generated.	RW	0
7	B_SRP_ STARTED_EN	B-device SRP started interrupt enable. Enables to generate an interrupt when the OTG_IRQ_SRC:B_SRP_STARTED bit is active. 0x0: No interrupt is generated. 0x1: An interrupt is generated.	RW	0
6:1	Reserved	Reserved	RW	0x--
0	OPRT_CHG_EN	OTG output port status change interrupt enable. Enables to generate an interrupt when the OTG_IRQ_SRC:OPRT_CHG bit is active. 0x0: No interrupt is generated. 0x1: An interrupt is generated.	RW	0

Table 22−94. OTG_IRQ_SRC

Address Offset	0x314		
Physical Address	0x4805 E314	**Instance**	USBOT1
Description	OTG interrupt status register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DRIVER_SWITCH	RESERVED	A_VBUS_ERR	A_REQ_TMROUT	A_SRP_DETECT	B_HNP_FAIL	B_SRP_TMROUT	B_SRP_DONE	B_SRP_STARTED	Reserved						OPRT_CHG

Bits	Field Name	Description	Type	Reset
15	DRIVER_ SWITCH	Driver switch interrupt status. Driver switch interrupts occur when HNP completes and control of the OTG link must switch between the OMAP USB host controller driver and the OMAP USB device controller driver (or vice versa). This interrupt also occurs upon enabling the OTG functionality by writing OTG_SYSCON_2.OTG_EN = 1. In this case, OTG_CTRL.DRIVER_SEL selects the appropriate controller driver, given the value of OTG_CTRL.ID. Cleared by writing a 1. Has no effect when OTG_EN = 0. 0x0: Driver switch interrupt is inactive. 0x1: Driver switch interrupt is active.	RW	0
14	Reserved	Reserved	RW	-
13	A_VBUS_ERR	A-device VBUS error interrupt status. Reflects the status of the A_VBUS_ERR interrupt. VBUS errors occur when the OMAP OTG controller state machine acts as default-A dual-role device and transitions into state A_VBUS_ERR. Usually, the state machine transitions into state A_VBUS_ERR if VBUS voltage drops below the A_VBUS_VALID threshold because of low battery conditions or heavy loading by the attached device. Transitions into this state can unintentionally occur if system software turns off VBUS power and writes 0 to OTG_CTRL.VBUSVLD before writing 1 to OTG_CTRL.OTG_BUS_DROP. Writing 1 to OTG_BUS_DROP clears the source of this interrupt. Cleared by writing a 1. Has no effect when OTG_EN is 0. 0x0: VBUS error interrupt is inactive. 0x1: VBUS error interrupt is active.	RW	0

Table 22–94. OTG_IRQ_SRC (Continued)

Bits	Field Name	Description	Type	Reset
12	A_REQ_ TMROUT	A-device request time-out interrupt status. Reflects the status of the A_REQ_TMROUT interrupt. This interrupt occurs when the OMAP OTG controller state machine acts as default-A dual-role device and transitions from state a_wait_bcon to a_wait_vfall because the state machine did not see an attach. This interrupt can occur if system software attempts to power up an OTG link when the cable is not attached at both ends, or if the device at the far end of the cable does not provide a pullup resistor. Cleared by writing a 1. Has no effect when OTG_EN is 0. 0x0: A-device request time-out interrupt is inactive. 0x1: A-device request time-out interrupt is active.	RW	0
11	A_SRP_DETECT	A-device SRP detection interrupt status. Reflects the status of SRP detection when the OMAP OTG controller acts as OTG default-A device. When the OTG controller sees a valid and enabled SRP method, this interrupt is generated. SRP is not detected if OTG_CTRL.OTG_BUS_DROP is 1 or if OTG_CTRL.A_BUSREQ is active. Cleared by writing 1. Has no effect when OTG_EN is 0. 0x0: SRP detection interrupt is inactive. 0x1: An SRP has been detected, using the VBUS pulsing and/or data pulsing detection mechanism.	RW	0
10	B_HNP_FAIL	B-device HNP failed interrupt status. Reflects interrupts that are generated when the OMAP OTG controller acts as OTG default-B device and an HNP attempt to issue fails. This interrupt is generated when the default-A device does not enable its D+ pullup in response to the HNP request within the allotted time. This interrupt is also generated if the OMAP OTG controller acting as default-B device issues HNP, but the default-A device issues a USB resume. A third instance where this interrupt can be generated is if OMAP acting as default-B device issues an HNP request, but VBUS fails before completion of the HNP. Cleared by writing 1. Has no effect when OTG_EN is 0. 0x0: B-device HNP failure interrupt is inactive. 0x1: B-device HNP failure interrupt is active.	RW	0

Table 22−94. OTG_IRQ_SRC (Continued)

Bits	Field Name	Description	Type	Reset
9	B_SRP_TMROUT	B-device SRP time-out interrupt status. Reflects interrupts that are generated when the OMAP OTG controller acts as an OTG default-B device and issues an SRP request, but the default-A device does not power VBUS within 5.5 seconds of the beginning of the SRP request. When this interrupt occurs, system software must inform the user that something is wrong (such as bad or unconnected cabling). Cleared by writing 1. Has no effect when OTG_EN is 0. The SRP timer starts when the B-device bus request is activated. 0x0: SRP time-out interrupt is inactive. 0x1: SRP time-out interrupt is active.	RW	0
8	B_SRP_DONE	B-device SRP done interrupt status. Reflects interrupts that are generated upon completion of an SRP when the OMAP OTG controller acts as an OTG default-B device. The actual duration of SRP depends on the values programmed into OTG_SYSCON_2.SRP_GPDATA and OTG_SYSCON_2.SRP_GPUVBUS. Cleared by writing 1. Has no effect when OTG_EN is 0. 0x0: SRP done interrupt is inactive. 0x1: SRP done interrupt is active.	RW	0
7	B_SRP_ STARTED	B-device SRP started interrupt status. Reflects interrupts that are generated when the OMAP OTG controller acts as a default-B device and begins to pulse D+ at the beginning of an SRP request. This interrupt does not occur when OTG_CTRL.BSESSVLD is 1. Cleared by writing 1. Has no effect when OTG_EN is 0. 0x0: SRP begin interrupt is inactive. 0x1: SRP begin interrupt is active.	RW	0
6:1	Reserved	Reserved	RW	0x--

Table 22−94. OTG_IRQ_SRC (Continued)

0	OPRT_CHG	OTG output port status change interrupt status. Reflects interrupts that are generated when the OMAP OTG controller requires a change in the OTG transceiver control signals. Any change in OTG_CTRL.[5:0] causes this interrupt. These interrupts include changes to D+ pullup or pulldown; VBUS drive, pullup, and pulldown; or ID pin pullup. Generally, system software responds to this interrupt by issuing I2C operations to change the control register in the OTG transceiver. When 1, OTG_CTRL.[5:0] reflect their values at the time that the interrupt is generated. Other changes on OTG_CTRL.[5:0] can occur before the interrupt is processed, but these changes are not shown in OTG_CTRL.[5:0] until after the output port status change interrupt status is cleared. Cleared by writing 1. Has no effect when OTG_EN is 0. 0x0: Output port status change interrupt is inactive. 0x1: An interrupt is generated for a change on OTG output ports status.	RW	0

Table 22−95. OTG_OUTCTRL

Address Offset	0x318		
Physical Address	0x4805 E318	**Instance**	USBOT1
Description	Output pins control register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	OTGVPD	OTGVPU	OTGPUID	RESERVED	USB2VDR	USB2PDEN	USB2PUEN	RESERVED	USB1VDR	USB1PDEN	USB1PUEN	RESERVED	USB0VDR	USB0PDEN	USB0PUEN

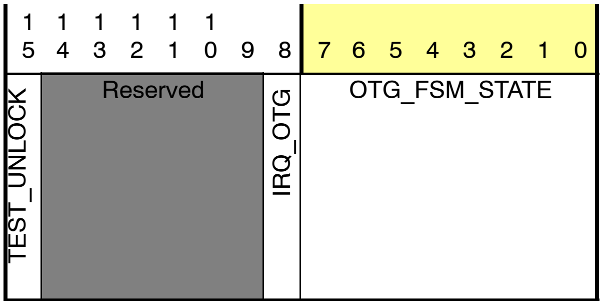
Bits	Field Name	Description	Type	Reset
15	Reserved	Reserved	RW	-
14	OTGVPD	OTG pulldown VBUS 0x0: OTGPDVBUSON value is function of the internal OTG decision. 0x1: OTGPDVBUSON is low.	RW	0
13	OTGVPU	OTG pullup VBUS 0x0: OTGPDVBUSON value is function of the internal OTG decision. 0x1: OTGPDVBUSON is low.	RW	0
12	OTGPUID	OTG pullup ID 0x0: OTGPUIDON value is active-low when OTG_EN is active; else, passive-high. 0x1: OTGPUIDON is active low.	RW	0

Table 22−95. OTG_OUTCTRL (Continued)

Bits	Field Name	Description	Type	Reset
11	Reserved	Reserved	RW	-
10	USB2VDR	USB2 port Vbus drive 0x0: USB2VBUSDRIVEO value is function of the internal OTG decision. 0x1: USB2VBUSDRIVEO is high (active).	RW	0
9	USB2PDEN	USB2 port pulldown enabled 0x0: USB2PDENON value is function of the internal OTG decision. 0x1: USB2PDENON is low.	RW	0
8	USB2PUEN	USB2 port pullup enabled 0x0: USB2PUENON value is function of the internal OTG decision. 0x1: USB2PUENON is low.	RW	0
7	Reserved	Reserved	RW	-
6	USB1VDR	USB1 port Vbus drive 0x0: USB1VBUSDRIVEO value is function of the internal OTG decision. 0x1: USB1VBUSDRIVEO is high (active).	RW	0
5	USB1PDEN	USB1 port pulldown enabled 0x0: USB1PDENON value is function of the internal OTG decision. 0x1: USB1PDENON is low.	RW	0
4	USB1PUEN	USB1 port pullup enabled 0x0: USB1PUENON value is function of the internal OTG decision. 0x1: USB1PUENON is low.	RW	0
3	Reserved	Reserved	RW	-
2	USB0VDR	USB0 port Vbus drive 0x0: USB0VBUSDRIVEO value is function of the internal OTG decision. 0x1: USB0VBUSDRIVEO is high (active).	RW	0
1	USB0PDEN	USB0 port pulldown enabled 0x0: USB0PDENON value is function of the internal OTG decision. 0x1: USB0PDENON is low.	RW	0
0	USB0PUEN	USB0 port pullup enabled 0x0: USB0PUENON value is function of the internal OTG decision. 0x1: USB0PUENON is low.	RW	0

Table 22–96. OTG_TEST

Address Offset	0x320		
Physical Address	0x4805 E320	**Instance**	USBOT1
Description	OTG test register		
Type	RW		



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEST_UNLOCK	Reserved						IRQ_OTG	OTG_FSM_STATE							

Bits	Field Name	Description		Type	Reset
15	TEST_UNLOCK	Test mode unlock. Always reads 0x0.		RW	0
		0x0:	Normal mode		
		0x1:	OTG test features enabled		
14:9	Reserved	Reserved		R	0x--
8	IRQ_OTG	IRQOTGON signal control. Always reads 0x0.		RW	0
		0x0:	Normal mode		
		0x1:	Forces IRQOTGON to 0 (active)		
7:0	OTG_FSM_STATE	OTG FSM state		R	0x00
		0x0:	OTG_BIDLE		
		0x1:	OTG_BPERIPH		
		0x3:	OTG_WAIT_BPERIPH_1		
		0x7:	OTG_WAIT_BPERIPH_0		
		0x11:	OTG_BSRP		
		0x12:	OTG_WAIT_BWAITACON_0		
		0x13:	OTG_WAIT_BWAITACON_1		
		0x16:	OTG_WAIT_BWAITACON_2		
		0x17:	OTG_BWAITACON		
		0x34:	OTG_BHOST		
		0x36:	OTG_AIDLE		
		0x3C:	OTG_AWAITVFALL		
		0x7D:	OTG_AVBUSERR		
		0x89:	OTG_APERIPHERAL		

Table 22−96. OTG_TEST (Continued)

Bits	Field Name	Description	Type	Reset
7:0	OTG_FSM_STATE	0x9E: OTG_APERIPHERALSUSP0		
		0x9F: OTG_APERIPHERALSUSP1		
		0xB7: OTG_AWAITVRISE		
		0xB8: OTG_AWAITBCON		
		0xB9: OTG_AHOST		
		0xBA: OTG_ASUSPEND		

Table 22−97. OTG_VC

Address Offset	0x3FC		
Physical Address	0x4805 E3FC	**Instance**	USBOT1
Description	OTG vendor code 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																VC															

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Reserved	R	0x----
15:0	VC	Vendor code identifier	R	

Chapter 23

General-Purpose Interface

This chapter describes the four general-purpose input/output (GPIO) modules that form the general-purpose interface in the OMAP2420 device.

Topic **Page**

23.1 GPIO Functional Overview

The OMAP2420 general-purpose interface is the combination of four general-purpose input/output (GPIO) modules.

Each GPIO module controls 32 dedicated general-purpose pins with input and output capabilities. Therefore, the general-purpose interface provides support for 128 (4 x 32) pins.

These pins can be configured to be used for the following applications:

- Data input (capture)/output (drive)
- Keyboard interface with a debounce (protection circuitry) cell
- Interrupt generation in active mode (module running synchronously with the interface clock) on 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 (power-saving mode; the interface clock is not present) on the detection of external events

The four GPIO modules, GPIO1 through GPIO4, present a total theoretical capability of 128 GPIO pins at the OMAP2420 device level. These four GPIO modules are regrouped into the quad-GPIO module, which connects to a single target agent port on the L4 interconnect.

The module does not include pad control (pull up/down control, open-drain feature), which is part of the chip top-level configuration.

23.1.1 Global Features

Four operating modes are defined for the GPIO module (see the *Power Management* subsection of Section 23.3.1.1, *Clocking, Reset, and Power-Management Scheme*):

- Active mode: The module runs synchronously with the provided clock; synchronous interrupts can be generated.
- Idle mode: The interface clock is not present; a wake-up request can be generated.
- Inactive mode: Same as idle mode, but wake-up request cannot be generated.
- Disabled mode: The module has no activity; internal clock paths are gated.

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 to be used independently by the DSP and the MPU subsystems. One of these interrupts is mapped on the DSP subsystem interrupt controller and the other on the MPU subsystem interrupt controller.

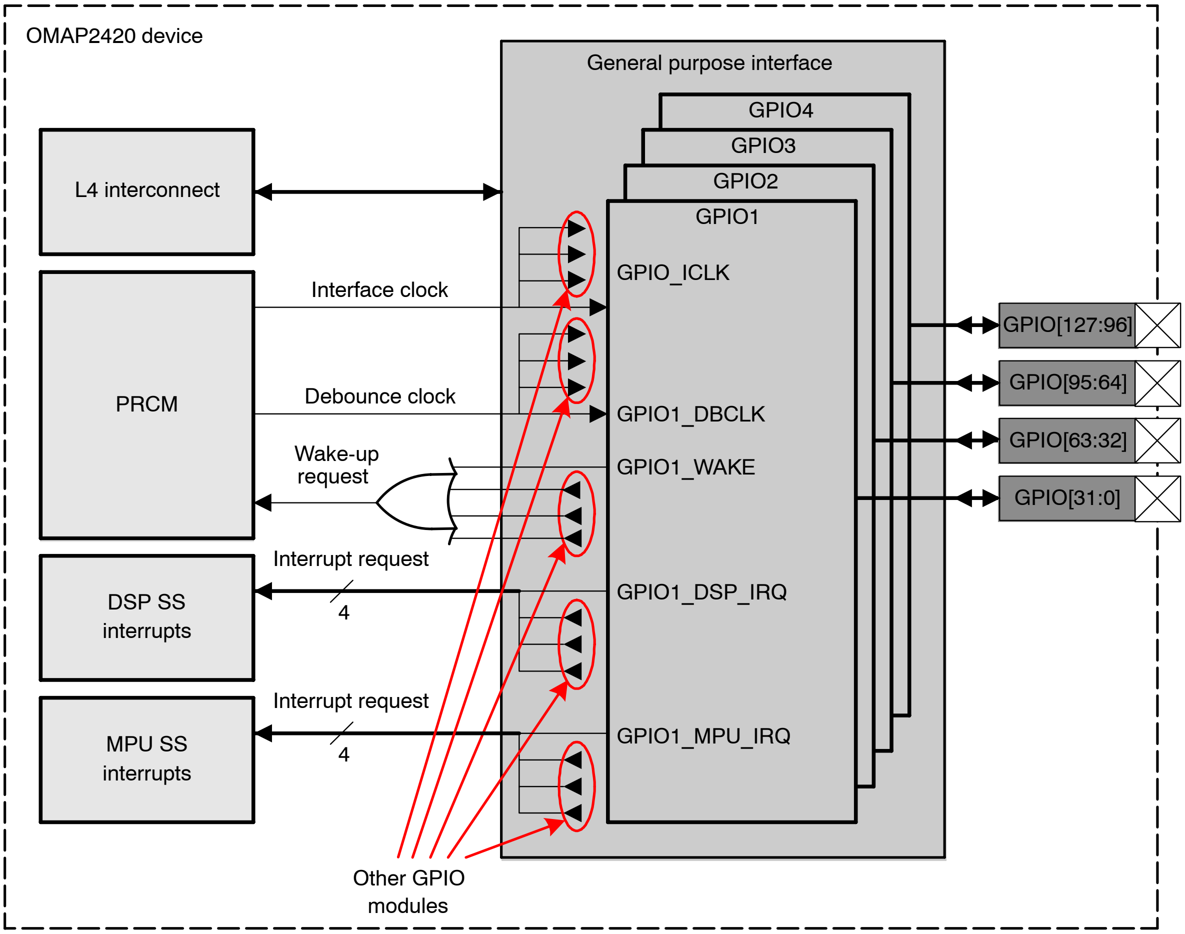
- Asynchronous wake-up requests in idle mode from input channels are merged to issue one wake-up signal.
- Data input (capture)/output (drive)

There are 8 interrupt lines (2 interrupt lines per GPIO module instance) in the general-purpose interface.

All wake-up signals (one signal per GPIO module) are internally ORed together in the OMAP2420 to produce a unique wake-up request to the power, reset, and clock management (PRCM) module.

Figure 23−1 shows the general-purpose interface.

Figure 23−1. General-Purpose Interface Highlight



Each channel in the GPIO modules has the following features:

- The output enable register (GPIO_OE) controls the output capability for each pin.
- The output line level reflects the value written in the data output register (GPIO_DATAOUT) through the L4 interconnect.
- The input line can be fed to the GPIO module through an optional and configurable debounce cell. (The value is global for all ports of one GPIO module, for up to four different debouncing values.)

- The input line value is sampled into the data input register (GPIO_DATAIN) 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) to be 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).

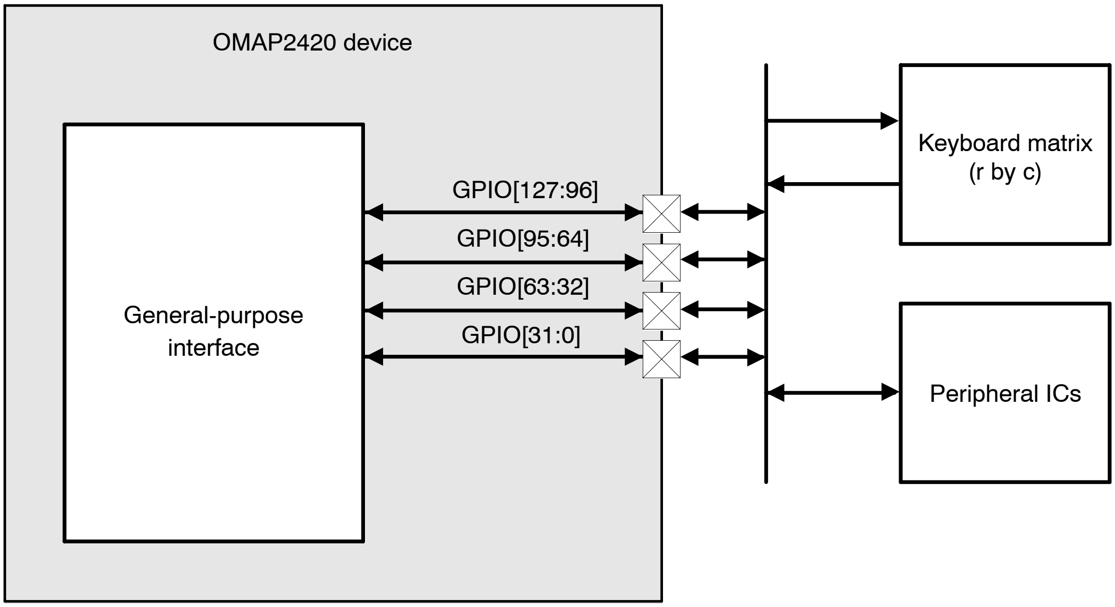
The module provides an alternative to the atomic test and set operations for the data output (GPIO_DATAOUT), interrupt enable (GPIO_IRQENABLE1 and GPIO_IRQENABLE2), and wake-up enable (GPIO_WAKEUPENABLE) registers. For these registers, the modules implement the set and clear protocol register update (see Section 23.5.2, *Set and Clear Instructions*).

23.2 GPIO Environment

The general-purpose interface, the combination of four GPIO modules, is a flexible, user-programmable, general-purpose I/O controller. It implements functions that are not implemented with the dedicated controllers in the OMAP2420 system and that 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 23−2 shows a typical application using the general-purpose interface.

Figure 23−2. General-Purpose Interface Typical Application System Overview



The general-purpose interface physically connects the OMAP2420 to a keyboard matrix and peripheral ICs.

- ❑ GPIO as a keyboard interface

  The general-purpose interface can be used as a keyboard interface. Channels can be dedicated according to the keyboard matrix size (r x c). Figure 23−3 shows row channels configured as input with the input debounce feature enabled. The row channels are driven high with an external pullup. Column channels are configured as output and drive a low level.

  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 according to the correct configuration (see Section 23.5.3, *Interrupt and Wakeup*).

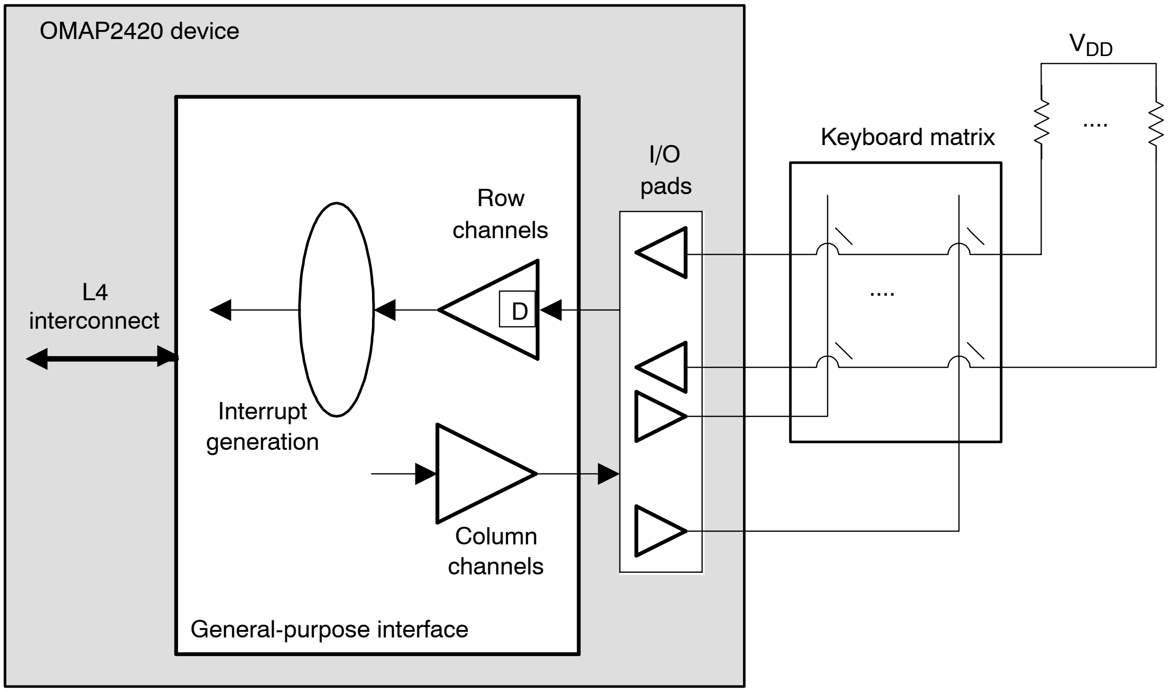
  When the keyboard interrupt is received, the processor (microprocessor unit [MPU] and/or digital signal processor [DSP] subsystem) can disable the keyboard interrupt and scan the column channels for the key coordinates.

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. Figure 23−3 shows the general-purpose interface used as a keyboard interface.

Figure 23−3. General-Purpose Interface Used as a Keyboard Interface

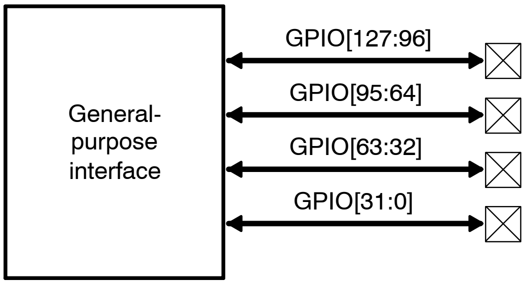


23.2.1 General-Purpose Interface Functional Interfaces

23.2.1.1 Basic General-Purpose Interface Pins

Figure 23−4 shows the general-purpose interface functional interface signals.

Figure 23−4. General-Purpose Interface Functional Interface Signals



23.2.1.2 General-Purpose Interface Functional Interface Description

Table 23−1 describes the I/O signal for the general-purpose interface.

Table 23−1. I/O Description

Signal Name	I/O[1]	Description	Module Reset Value
GPIO[127:0]	I/O	General-purpose input output signals	Input: Unknown Output: 0x00..0

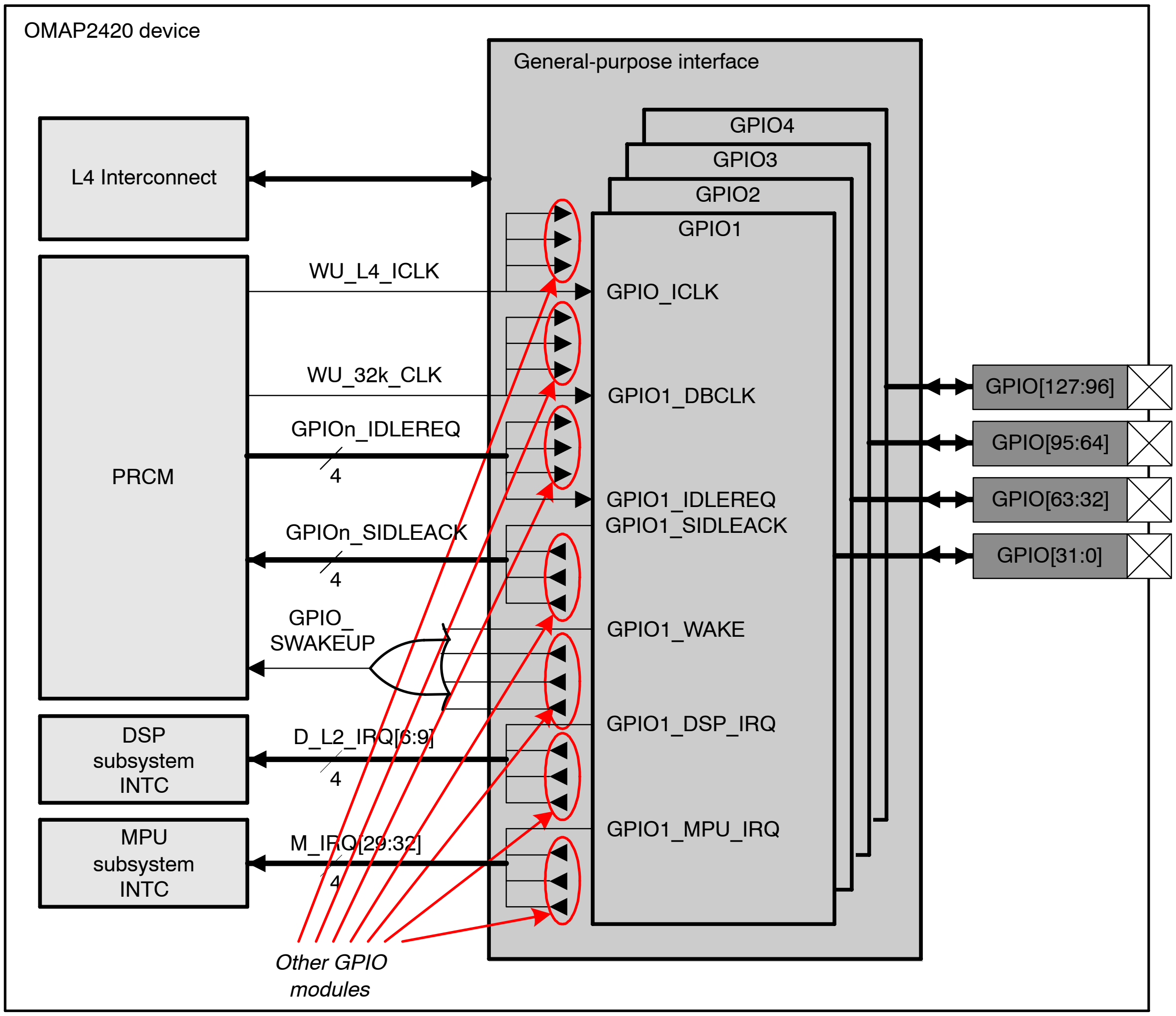
1) I = Input, O = Output

23.3 GPIO Integration

23.3.1 Description

Figure 23−5 shows general-purpose interface integration in the OMAP2420 device.

Figure 23−5. General-Purpose Interface Typical Application System Overview



23.3.1.1 Clocking, Reset, and Power-Management Scheme

Clocking

Each GPIO module uses two clocks:

❑ Debounce clock: The 32-kHz debounce clock, GPIOn_DBCLK, (where *n* is 1, 2, 3, and 4 with one debounce clock per module), comes from the PRCM module and is used for debounce cell logic (without the corresponding configuration registers). This cell can sample the input line and filter the input level using a programmed delay.

- ☐ Interface clock: The interface clock, GPIO_ICLK, comes from the PRCM module and is used throughout the GPIO module (except in the debounce cell logic).

  The interface clock clocks data exchanges between the L4 interconnect and the internal logic. Clock gating allows module power consumption to be adapted to the activity. Table 23−2 lists the GPIO clocks and their characteristics.

Table 23−2. Clocks

Attributes	Frequency	Name	Mapping	Comments
Debounce clock	32 kHz	GPIOn_DBCLK	WU_32k_CLK	Source is PRCM module.
Interface clock	Depending on PRCM register settings	GPIO_ICLK	WU_L4_ICLK	Source is PRCM module.

Reset

The general-purpose interface can be reset using the domain reset (hardware reset) or by setting a dedicated configuration bit (software reset) in each GPIO module.

- ☐ Hardware reset: The general-purpose interface is attached to the WKUP_RST reset domain. The WKUP_RST_N signal resets all of the modules (see Chapter 5, *Power, Reset, and Clock Management*).

  The hardware reset signal has a global reset effect on the general-purpose interface.

- ☐ Software reset: Each GPIO module has its own software reset through the SOFTRESET bit GPIO_SYSCONFIG[1] (0: normal mode, 1: module is reset).

  This bit controls the software reset. Writing 1 to this bit resets the module. Bit value 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 (see Section 23.4.1.1, *Clocking and Reset Strategy*).

Power Domain

The general-purpose interface is attached to the WKUP power domain (see Chapter 5, *Power, Reset, and Clock Management*). This domain is composed of the logic that must be permanently supplied to manage domain power state transitions and detect wake-up events. The WKUP power domain is continuously active.

Power Management

Idle Scheme

To save dynamic consumption, an efficient idle scheme is based on:

- 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 management.

The idle mode request (GPIOn_IDLEREQ, where n is 1, 2, 3, and 4), idle acknowledge (GPIOn_SIDLEACK, where *n* is 1, 2, 3, and 4), and wake-up request (GPIO_SWAKEUP) are sideband signals between the PRCM module and the general-purpose interface (see Section 23.3.1.2, *Hardware Requests*, and Section 23.4.1.1, *Clocking and Reset Strategy*).

Operating Modes

Four operating modes are defined for the modules:

- Active mode: The module runs synchronously on the interface clock; interrupts can be generated based on configuration and external signals.
- Idle mode: Power-saving mode with the module in a waiting state. The interface clock can be stopped, no interrupt can be generated, and a wake-up signal can be generated according to configuration and external signals.

  If the debounce clock 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, no interrupt can be generated, and the wake-up feature is inhibited.
- Disabled mode: The module is not used. The internal clock paths are gated, and no interrupt or wake-up request can be generated.

Idle and inactive modes are configured in the module and activated on request by the PRCM module (see Chapter 5, *Power, Reset, and Clock Management*) through sideband signals (see Section 23.4.1.1, Clock*ing and Reset Strategy*).

The disabled mode is set by software through a dedicated configuration bit, the DISABLEMODULE bit GPIO_CTRL[0] (0: module is enabled and clocks are not gated, 1: module is disabled and clocks are gated). It unconditionally gates the internal clock paths not used for the L4 interconnect.

System Power Management and Wakeup

The PRCM module can require the modules of the general-purpose interface to be idled for power saving.

The general-purpose interface has four identical idle mode request/acknowledge (handshake) mechanisms with the PRCM module (see Figure 23−5 and Section 23.3.1.2, *Hardware Requests*), one per GPIO module. The general-purpose interface allows the GPIO modules to go to idle mode according to the IDLEMODE bits GPIO_SYSCONFIG[4:3].

The PRCM module requests idle mode at the same time for all four GPIO modules. The idle acknowledge depends on the configuration and activity of each GPIO module.

☐ Smart-idle (recommended mode)

When the GPIO module is configured in smart-idle (GPIO_SYSCONFIG[4:3] (10)) mode and receives an idle request from the PRCM module (CM_FCLKEN_WKUP[2] set to 0, CM_ICLKEN_WKUP[2] set to 0, or CM_AUTOIDLE_WKUP[2] set to 1 and L4 interface clock idle transitions), the GPIO module checks for more activity (the data input register, GPIO_DATAIN, completed to capture the input GPIO pins, no pending interrupt, all interrupt status bits cleared).

The idle acknowledge is then asserted and the module 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).

When in 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 idle mode, if the ENAWAKEUP bit of the system configuration register (GPIO_SYSCONFIG[2] bit) is set to 1 (wake-up capability enabled) and the PRCM register PM_WKEN_WKUP[2] bit is also set to 1.

☐ Force-idle

When the GPIO module is configured in force-idle mode (GPIO_SYSCONFIG[4:3] (00)) and receives an idle request from the PRCM module (CM_FCLKEN_WKUP[2] set to 0 CM_ICLKEN_WKUP[2] set to 0, or CM_AUTOIDLE_WKUP[2] 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, no interrupt can be generated, and the wake-up feature is totally inhibited.

☐ No-idle

When the GPIO module is configured in no-idle mode (GPIO_SYSCONFIG[4:3] (01)) and receives an idle request from the PRCM module (CM_FCLKEN_WKUP[2] set to 0, CM_ICLKEN_WKUP[2] set to 0, or CM_AUTOIDLE_WKUP[2] set to 1 and L4 interface clock idle transitions), the GPIO module does not go to idle mode and the idle acknowledge is never sent.

Note:

The general-purpose interface idle state can be checked by the PRCM module register bit CM_IDLEST_WKUP[2] (0 idle, 1 active). It is idle only when all the GPIO modules (GPIO1 to 4) are configured in smart-idle and assert their idle acknowledge.

GPIO module wake-up status can be checked by the PRCM module register bit PM_WKST_WKUP[2] (read 0: no wake-up occurred, read 1: wake-up occurred, write 1: status bit reset).

Module Power Saving

The GPIO module manages local power through internal clock gating:

- Internal interface clock gating: The clock for the L4 interconnect logic can be gated when the module is not accessed, if the AUTOIDLE bit GPIO_SYSCONFIG[0] is set. Otherwise, this logic is free-running on the interface clock.
- Disabled mode: In disabled mode, all internal clock paths not used for the L4 interconnect are gated. The DISABLEMODULE bit GPIO_CTRL[0] controls a clock-gating feature at the module level. When set (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.

  Interface clock gating is controlled with the AUTOIDLE bit GPIO_SYSCONFIG[0], which is used to save power when the module is not in use, because of the multiplexing configuration selected at the chip level. This bit has precedence over all other internal configuration bits.

23.3.1.2 Hardware Requests

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 8 interrupt lines (2 interrupt lines per GPIO module instance).

The synchronous interrupt request lines 1 and 2 are active according to their respective interrupt enable 1 and 2 registers (GPIO_IRQENABLE1 and GPIO_IRQENABLE2).

Synchronous interrupt request line 1 is mapped on the MPU interrupt controller.

Synchronous interrupt request line 2 is mapped on the DSP interrupt controller.

Table 23−3 lists the interrupt lines that are driven out from the general-purpose interface to the MPU subsystem and DSP subsystem interrupt controllers.

Table 23−3. Interrupts

Name	Mapping	Comments
GPIO1 Module (GPIO[31:0])		
GPIO1_MPU_IRQ	M_IRQ_29	Destination is MPU subsystem interrupt controller.
GPIO1_DSP_IRQ	D_L2_IRQ_6	Destination is DSP subsystem interrupt controller.
GPIO2 Module (GPIO[63:32])		
GPIO2_MPU_IRQ	M_IRQ_30	Destination is MPU subsystem interrupt controller.
GPIO2_DSP_IRQ	D_L2_IRQ_7	Destination is DSP subsystem interrupt controller.
GPIO3 Module (GPIO[95:64])		
GPIO3_MPU_IRQ	M_IRQ_31	Destination is MPU subsystem interrupt controller.
GPIO3_DSP_IRQ	D_L2_IRQ_8	Destination is DSP subsystem interrupt controller.
GPIO4 Module (GPIO[127:96])		
GPIO4_MPU_IRQ	M_IRQ_32	Destination is MPU subsystem interrupt controller.
GPIO4_DSP_IRQ	D_L2_IRQ_9	Destination is DSP subsystem interrupt controller.

Note:

Before going to idle, if the WAKEUPENABLE bit is set to 1, the software must ensure that all events in the GPIO_IRQSTATUS1 and GPIO_IRQSTATUS2 registers are cleared.

Wake-Up Generation

The general-purpose interface is attached to the WKUP power domain (see Chapter 5, *Power, Reset, and Clock Management*) and can wake up the system.

All wake-up sources (the 32-input GPIO channels) are merged to issue a single asynchronous wake-up request in each GPIO module following the expected transition(s) (according to the register programming). Also, all wake-up signals (one signal per GPIO module) are internally ORed together in the OMAP2420 to produce a unique wake-up request (GPIO_SWAKEUP) to the PRCM module.

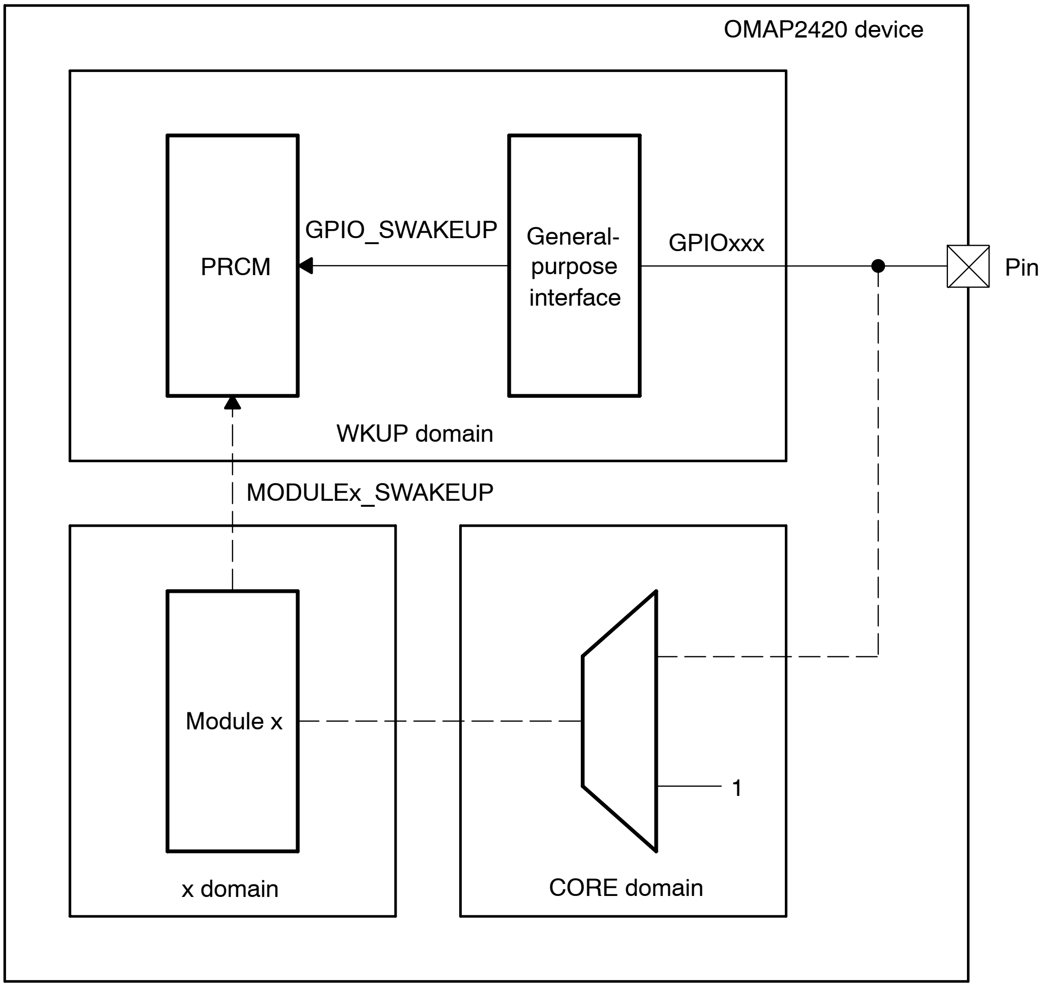
The asynchronous wake-up request line is active according to the wake-up enable register (GPIO_WAKEUPENABLE). Table 23−4 describes the wake-up signal.

Table 23−4. Wake-Up Signal

Name	Mapping	Comments
GPIOn_WAKE (where *n* is 1, 2 ,3 and 4)	GPIO_SWAKEUP	Destination is PRCM module

Figure 23−6 shows a wake-up description in the OMAP2420 device.

Figure 23−6. Wake-up Description



Module *x* can be any OMAP2420 module that does not belong to the WKUP power domain (for example, the UART module in the CORE power domain). This module can have a wake-up feature but cannot wake up the system on an external event when its CORE (*x)* power domain is off. The multiplexers used for pad multiplexing in the OMAP2420 belong to the CORE power domain (see Chapter 5, *Power, Reset, and Clock Management*).

Therefore, when all power domains are off (except for the WKUP power domain, which is continuously active; see Chapter 5, *Power, Reset, and Clock Management*), the pin is directly connected to the general-purpose interface. If the GPIO channel muxed on this pin is configured as a wake-up source (see Section 23.5.3, *Interrupt and Wakeup*), a wake-up request is asserted to the system.

Note:

The wake-up capability works whether or not the GPIO function of the pin is selected by the pinout configuration.

Note:

Some GPIO channels cannot be used as wake-up sources in the OMAP2420.

All GPIO channels are muxed with other device pins, but the wake-up feature cannot work if the muxes are inserted between the pin and the GPIO module. One GPIO channel corresponds to several pins.

Three GPIO channels (GPIO.29, GPIO.30, and GPIO.31) have only an output capability in the OMAP2420 and cannot be used as wake-up sources.

Table 23−5 lists the GPIO channels without the wake-up feature.

The GPIO_IRQENABLE1, GPIO_IRQENABLE2, and GPIO_WAKEUPENABLE register bits for the GPIO pins without wake-up capability must be disabled before the CORE power domain enters retention or off mode, then enabled after wakeup if required. Otherwise, spurious wake-up events are generated by an isolation cell between the CORE and WKUP power domains.

Table 23−5. GPIO Channels Without the Wake-Up Feature

GPIO Number	Alternate Signals			Comments
GPIO.6	GPMC.A7	GPIO6		
GPIO.7	GPMC.A6	MMC.DAT_DIR0		
GPIO.8	GPMC.A5	UART1.RTS	MMC.CMD_DIR	
GPIO.9	GPMC.A4	UART1.TX		
GPIO.10	GPMC.A3	UART1.RX		
GPIO.11	GPMC.A2	McBSP2.DR		
GPIO.12	GPMC.A1	McBSP2.CLKX		
GPIO.13	GPMC.D15			
GPIO.14	GPMC.D14			
GPIO.15	GPMC.D13			
GPIO.16	GPMC.D12			
GPIO.17	GPMC.D11			
GPIO.25	GPMC.nCS4			
GPIO.29	GPMC.nBE0			GPO only, no input
GPIO.30	GPMC.nBE1			GPO only, no input
GPIO.31	GPMC.nWP			GPO only, no input
GPIO.38	SDRC.CKE1	DSS.D8		
GPIO.52	CAM.D2	SYS.CLKREQ		
GPIO.53	CAM.D1	CAM.D9		

Table 23−5. GPIO Channels Without the Wake-up Feature (Continued)

GPIO Number	Alternate Signals			Comments
GPIO.54	CAM.D0	CAM.D8		
GPIO.59		MMC.CLKI		

Idle Mode Request/Acknowledge Feature

Two idle control signals allow each GPIO module to enter different idle modes to save power (see Section 23.4.1.1, *Clocking and Reset Activity*).

Table 23−6 lists the four GPIOn_IDLEREQ signals from the PRCM module and four GPIOn_SIDLEACK signals to the PRCM module (where *n* is 1, 2, 3, and 4).

Table 23−6. Idle Mode Request/Acknowledge Signals

Name	Mapping	Comments
GPIO1 Module (GPIO[31:0])		
GPIO1_IDLEREQ	GPIO1_IDLEREQ	Source is PRCM module.
GPIO1_SIDLEACK	GPIO1_SIDLEACK	Destination is PRCM module.
GPIO2 Module (GPIO[63:32])		
GPIO2_IDLEREQ	GPIO2_IDLEREQ	Source is PRCM module.
GPIO2_SIDLEACK	GPIO2_SIDLEACK	Destination is PRCM module.
GPIO3 Module (GPIO[95:64])		
GPIO3_IDLEREQ	GPIO3_IDLEREQ	Source is PRCM module.
GPIO3_SIDLEACK	GPIO3_SIDLEACK	Destination is PRCM module.
GPIO4 Module (GPIO[127:96])		
GPIO4_IDLEREQ	GPIO4_IDLEREQ	Source is PRCM module.
GPIO4_SIDLEACK	GPIO4_SIDLEACK	Destination is PRCM module.

23.3.1.3 Pin List and Pad Multiplexing With Other Functions

Table 23−7 describes the pin and pad multiplexing options for the general-purpose interface. The GPIO channels are available on Mode 3 (indicated in the table by gray shading) and few channels are duplicated on the same pin on Mode 0 (white cells show alternate pin functions).

Table 23−7. Pin Multiplexing for the General-Purpose Interface

Function n	Description	DIR	Ball	Alternate Functions					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
GPIO.0	GPIO 0	IO	B3	sdrc.a14					
GPIO.1	GPIO 1	IO	B4	sdrc.a13					
GPIO.2	GPIO 2	IO	D11	sdrc.a12					
GPIO.3	GPIO 3	IO	E3	gpmc.a10		sys. ndmareq5			
GPIO.4	GPIO 4	IO	D3	gpmc.a9		sys. ndmareq4			
GPIO.5	GPIO 5	IO	G4	gpmc.a8		sys. ndmareq3			
GPIO.6	GPIO 6	IO	E5		tv.detect		gpio.6		
			E4	gpmc.a7		sys. ndmareq2			
			E5	gpio.6	tv.detect				
GPIO.7	GPIO 7	IO	F3	gpmc.a6	dss.d23				
			F19	mmc.dat_ dir0	ms.dat0_ dir				
GPIO.8	GPIO 8	IO	F4	gpmc.a5	dss.d22				
			H21	uart1.rts		dss.d19			
			G18	mmc.cmd_ dir					
GPIO.9	GPIO 9	IO	G3	gpmc.a4	dss.d21				
			L20	uart1.tx		dss.d20			
GPIO.10	GPIO 10	IO	H3	gpmc.a3	dss.d20				
			T21	uart1.rx		dss.d21			
GPIO.11	GPIO 11	IO	H4	gpmc.a2	dss.d19				
			M21	mcbsp2.dr		dss.d22			
GPIO.12	GPIO 12	IO	J3	gpmc.a1	dss.d18				
			P21	mcbsp2. clkx		dss.d23			
GPIO.13	GPIO 13	IO	J7	gpmc.d15					
			AA10		usb2.se0	sys. ndmareq0			
GPIO.14	GPIO 14	IO	J8	gpmc.d14					
			AA6		usb2.rcv	sys. ndmareq1		cam_d8	

1) Input only. Nothing is multiplexed with SYS.CLKOUT output clock to avoid the risk of perturbation on clock signal.

Table 23−7. Pin Multiplexing for the General-Purpose Interface (Continued)

Function n	Description	DIR	Ball	Alternate Functions					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
GPIO.15	GPIO 15	IO	K7	gpmc.d13					
			AA4		usb2.tllse0			cam_d7	
GPIO.16	GPIO 16	IO	L8	gpmc.d12					
			Y11		usb2.dat	sys.clkout2			
GPIO.17	GPIO 17	IO	K8	gpmc.d11					
			AA12		usb2.txen				
GPIO.18	GPIO 18	IO	L7	gpmc.d10					
GPIO.19	GPIO 19	IO	M7	gpmc.d9					
GPIO.20	GPIO 20	IO	M8	gpmc.d8					
GPIO.21	GPIO 21	IO	J4	gpmc.clk					
GPIO.22	GPIO 22	IO	N8	gpmc.ncs1					
GPIO.23	GPIO 23	IO	E2	gpmc.ncs2					
GPIO.24	GPIO 24	IO	N2	gpmc.ncs3	gpmc.io_ dir				
GPIO.25	GPIO 25	IO	F1	gpmc.ncs4					
			V12		uart1.rts	usb1.rcv			
GPIO.26	GPIO 26	IO	H1	gpmc.ncs5					
GPIO.27	GPIO 27	IO	K1	gpmc.ncs6					
GPIO.28	GPIO 28	IO	L2	gpmc.ncs7	gpmc.io_ dir				
GPIO.29	GPIO 29	O	P7	gpmc.nbe0					
GPIO.30	GPIO 30	O	R1	gpmc.nbe1					
GPIO.31	GPIO 31	O	G2	gpmc.nwp					
GPIO.32	GPIO 32	IO	D21	uart1.cts		dss.d18			
GPIO.33	GPIO 33	IO	N7	gpmc. wait1					
GPIO.34	GPIO 34	IO	M1	gpmc. wait2					
GPIO.35	GPIO 35	IO	P1	gpmc. wait3					
GPIO.36	GPIO 36	IO	V17				gpio.36		sys.boot4
			V17	gpio.36					sys.boot4

1) Input only. Nothing is multiplexed with SYS.CLKOUT output clock to avoid the risk of perturbation on clock signal.

Table 23−7. Pin Multiplexing for the General-Purpose Interface (Continued)

Function n	Description	DIR	Ball	Alternate Functions					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
GPIO.37	GPIO 37	IO	C12	sdrc.ncs1					
GPIO.38	GPIO 38	IO	B13	sdrc.cke1					
			W9	dss.d8					
GPIO.39	GPIO 39	IO	P11	dss.d9					
GPIO.40	GPIO 40	IO	V10	dss.d10					
GPIO.41	GPIO 41	IO	Y10	dss.d11					
GPIO.42	GPIO 42	IO	W10	dss.d12					
GPIO.43	GPIO 43	IO	R11	dss.d13					
GPIO.44	GPIO 44	IO	V11	dss.d14					
GPIO.45	GPIO 45	IO	W11	dss.d15					
GPIO.46	GPIO 46	IO	P12	dss.d16					
GPIO.47	GPIO 47	IO	R12	dss.d17					
GPIO.48	GPIO 48	IO	W7	dss.acbias		mcbsp2.fsx			
GPIO.49	GPIO 49	IO	V4	cam.d5	hw.dbg7	mcbsp1.clkr			
GPIO.50	GPIO 50	IO	W2	cam.d4	hw.dbg6	mcbsp1.fsr			
GPIO.51	GPIO 51	IO	U4	cam.d3	hw.dbg5	mcbsp1.dr			
GPIO.52	GPIO 52	IO	V3	cam.d2	hw.dbg4	mcbsp1.clkx			
		I[1]	AA17	sys.clkreq					
GPIO.53	GPIO 53	IO	V6	cam.d9	hw.dbg11				
			V2	cam.d1	hw.dbg3	sti.din			
GPIO.54	GPIO 54	IO	Y4	cam.d8	hw.dbg10				
			T4	cam.d0	hw.dbg2	sti.dout			
GPIO.55	GPIO 55	IO	T3	cam.hs	hw.dbg1	mcbsp1.dx			
GPIO.56	GPIO 56	IO	U2	cam.vs	hw.dbg0	mcbsp1.fsx			
GPIO.57	GPIO 57	IO	V5	cam.lclk		mcbsp.clks			
GPIO.58	GPIO 58	IO	AA8			l4_ext_trig		cam_d6	

1) Input only. Nothing is multiplexed with SYS.CLKOUT output clock to avoid the risk of perturbation on clock signal.

Table 23−7. Pin Multiplexing for the General-Purpose Interface (Continued)

Function n	Description	DIR	Ball	Alternate Functions					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
GPIO.59	GPIO 59	IO	W12		uart1.tx	usb1.se0			
			H15	mmc.clki	ms.clki				
GPIO.60	GPIO 60	I	W19	sys_nirq					
GPIO.61	GPIO 61	IO	P13		uart1.cts	usb1.txen			
GPIO.62	GPIO 62	IO	R13		uart1.rx	usb1.dat	gpio.62		
			R13	gpio.62	uart1.rx	usb1.dat			
GPIO.63	GPIO 63	IO	Y12		eac.md_ sclk				
GPIO.64	GPIO 64	IO	W13		eac.md_ din				
GPIO.65	GPIO 65	IO	V13		eac.md_ dout				
GPIO.66	GPIO 66	IO	Y13		eac.md_fs				
GPIO.67	GPIO 67	IO	V19	uart2.cts	usb1.rcv	gpt9.pwm/ evt			
GPIO.68	GPIO 68	IO	W20	uart2.rts	usb1.txen	gpt10.pwm/ evt			
GPIO.69	GPIO 69	IO	N14	uart2.tx	usb1.se0	gpt11.pwm/ evt			
GPIO.70	GPIO 70	IO	P15	uart2.rx	usb1.dat	gpt12.pwm/ evt			
GPIO.71	GPIO 71	IO	R8	eac.bt_ sclk					
GPIO.72	GPIO 72	IO	P9	eac.bt_fs					
GPIO.73	GPIO 73	IO	Y3	eac.bt_din					
GPIO.74	GPIO 74	IO	W4	eac.bt_ dout		sti.clk			
GPIO.75	GPIO 75	IO	H14	mmc.dat1					
GPIO.76	GPIO 76	IO	E19	mmc.dat2		uart2.cts			
GPIO.77	GPIO 77	IO	D19	mmc.dat3		l4.ext_trig			
GPIO.78	GPIO 78	IO	E20	mmc.dat_ dir1		uart2.rts			
GPIO.79	GPIO 79	IO	F18	mmc.dat_ dir2		uart2.tx			
GPIO.80	GPIO 80	IO	E18	mmc.dat_ dir3		uart2.rx			
GPIO.81	GPIO 81	IO	U18	spi1.clk					
GPIO.82	GPIO 82	IO	V20	spi1.simo					

1) Input only. Nothing is multiplexed with SYS.CLKOUT output clock to avoid the risk of perturbation on clock signal.

Table 23−7. Pin Multiplexing for the General-Purpose Interface (Continued)

Function n	Description	DIR	Ball	Alternate Functions					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
GPIO.83	GPIO 83	IO	T18	spi1.somi					
GPIO.84	GPIO 84	IO	U19	spi1.ncs0					
GPIO.85	GPIO 85	IO	N15	spi1.ncs1					
GPIO.86	GPIO 86	IO	R18	spi1.ncs2					
GPIO.87	GPIO 87	IO	U21	spi1.ncs3					
GPIO.88	GPIO 88	IO	T19	spi2.clk					
GPIO.89	GPIO 89	IO	R19	spi2.simo	gpt10.pwm/ evt				
GPIO.90	GPIO 90	IO	R20	spi2.somi	gpt11.pwm/ evt				
GPIO.91	GPIO 91	IO	M14	spi2.ncs0	gpt12.pwm/ evt				
GPIO.92	GPIO 92	IO	M15	mcbsp1. clkr					
GPIO.93	GPIO 93	IO	P20	mcbsp1.fsr				spi2.ncs1	
GPIO.94	GPIO 94	IO	P19	mcbsp1.dx					
GPIO.95	GPIO 95	IO	P18	mcbsp1.dr					
GPIO.96	GPIO 96	IO	M18	mcbsp.clks					
GPIO.97	GPIO 97	IO	L14	mcbsp1. fsx					
GPIO.98	GPIO 98	IO	N19	mcbsp1. clkx					
GPIO.99	GPIO 99	IO	J15	i2c2.scl		gpt9.pwm/ evt			
GPIO.100	GPIO 100	IO	H19	i2c2.sda		spi2.ncs1			
GPIO.101	GPIO 101	IO	N18	hdq.sio	usb2.tllse0	sys.altclk			
GPIO.102	GPIO 102	IO	L18	uart3.cts/rctx	uart3.rx/irrx				
GPIO.103	GPIO 103	IO	L19	uart3.rts/sd	uart3.tx/irtx				
GPIO.104	GPIO 104	IO	K15	uart3.tx/irtx	uart3.rctx				
GPIO.105	GPIO 105	IO	K14	uart3.rx/irrx					
GPIO.106	GPIO 106	IO	J20	usb0.puen	mcbsp2.dx				
GPIO.107	GPIO 107	IO	J19	usb0.vp	mcbsp2.dr				
GPIO.108	GPIO 108	IO	K20	usb0.vm	mcbsp2. clkx			uart2.rx	
GPIO.109	GPIO 109	IO	J18	usb0.rcv	mcbsp2. fsx			uart2.cts	

1) Input only. Nothing is multiplexed with SYS.CLKOUT output clock to avoid the risk of perturbation on clock signal.

Table 23−7. Pin Multiplexing for the General-Purpose Interface (Continued)

Function n	Description	DIR	Ball	Alternate Functions					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
GPIO.110	GPIO 110	IO	K19	usb0.txen	uart3.cts/rctx	uart2.cts			
GPIO.111	GPIO 111	IO	J14	usb0.se0	uart3.tx/irtx	uart2.tx		uart2.rx	
GPIO.112	GPIO 112	IO	K18	usb0.dat	uart3.rx/irrx	uart2.rx		uart2.tx	
GPIO.113	GPIO 113	IO	Y15	eac.ac_ sclk	mcbsp2. clkx				
GPIO.114	GPIO 114	IO	R14	eac.ac_fs	mcbsp2. fsx				
GPIO.115	GPIO 115	IO	W15	eac.ac_din	mcbsp2.dr				
GPIO.116	GPIO 116	IO	V15	eac.ac_ dout	mcbsp2.dx				
GPIO.117	GPIO 117	IO	V14	eac.ac_ mclk					
GPIO.118	GPIO 118	IO	W16	eac.ac_rst	eac.bt_din	loop_eac. bt_din			
GPIO.119	GPIO 119	IO	W5				gpio.119		sys.boot0
			W5	gpio.119					sys.boot0
GPIO.120	GPIO 120	IO	Y5				gpio.120	cam.d9	sys.boot1
			Y5	gpio.120				cam.d9	sys.boot1
GPIO.121	GPIO 121	IO	R9				gpio.121	jtag.emu2	sys.boot2
			R9	gpio.121				jtag.emu2	sys.boot2
GPIO.122	GPIO 122	IO	P8				gpio.122	jtag.emu3	sys.boot3
			P8	gpio.122				jtag.emu3	sys.boot3
GPIO.123	GPIO 123	I[1]	W14	sys.clkout					
GPIO.124	GPIO 124	IO	V18				gpio.124		sys.boot5
			V18	gpio.124					sys.boot5
GPIO.125	GPIO 125	IO	P14		sys. jtagsel1	sys. jtagsel2	gpio.125		
			P14	gpio.125	sys. jtagsel1	sys. jtagsel2			
GPIO.126	GPIO 126	IO	AA21	jtag.emu1					
GPIO.127	GPIO 127	I/O	Y21	jtag.emu0					

1) Input only. Nothing is multiplexed with SYS.CLKOUT output clock to avoid the risk of perturbation on clock signal.

23.3.1.4 GPIO Register Summary

The following tables show the GPIO registers with their physical addresses (for a detailed description, see Section 23.6, *GPIO Registers*).

Table 23−8 shows the base address and address space for the GPIO.

Table 23−8. Instance Summary

Device Name	Description	Base Address	Size
GPIO:	GPIO1 module	0x4801 8000	1K byte
−GPIO1	GPIO top	0x4801 9000	256 bytes
−GPIO2	GPIO2 module	0x4801 A000	1K byte
−GIPO3	GPIO L4 interconnect	0x4801 B000	512 bytes
−GPIO4	GPIO3 module	0x4801 C000	1K byte
	GPIO4 module	0x4801 E000	1K byte

Table 23−9 lists GPIO1 module registers.
Table 23−10 lists GPIO2 module registers.
Table 23−11 lists GPIO3 module registers.
Table 23−12 lists GPIO4 module registers.
Table 23−13 lists the general-purpose interface top registers.
Table 23−14 lists the general-purpose interface L4 interconnect registers.

Table 23−9. GPIO1 Module Registers List

Register Name	Type	Register Width (Bits)	Physical Address
GPIO_REVISION	R	32	0x4801 8000
GPIO_SYSCONFIG	RW	32	0x4801 8010
GPIO_SYSSTATUS	R	32	0x4801 8014
GPIO_IRQSTATUS1	RW	32	0x4801 8018
GPIO_IRQENABLE1	RW	32	0x4801 801C
GPIO_WAKEUPENABLE	RW	32	0x4801 8020
GPIO_IRQSTATUS2	RW	32	0x4801 8028
GPIO_IRQENABLE2	RW	32	0x4801 802C
GPIO_CTRL	RW	32	0x4801 8030
GPIO_OE	RW	32	0x4801 8034
GPIO_DATAIN	R	32	0x4801 8038
GPIO_DATAOUT	RW	32	0x4801 803C
GPIO_LEVELDETECT0	RW	32	0x4801 8040
GPIO_LEVELDETECT1	RW	32	0x4801 8044
GPIO_RISINGDETECT	RW	32	0x4801 8048
GPIO_FALLINGDETECT	RW	32	0x4801 804C
GPIO_DEBOUNCENABLE	RW	32	0x4801 8050
GPIO_DEBOUNCINGTIME	RW	32	0x4801 8054

Table 23−9. GPIO1 Module Registers List (Continued)

Register Name	Type	Register Width (Bits)	Physical Address
GPIO_CLEARIRQENABLE1	RW	32	0x4801 8060
GPIO_SETIRQENABLE1	RW	32	0x4801 8064
GPIO_CLEARIRQENABLE2	RW	32	0x4801 8070
GPIO_SETIRQENABLE2	RW	32	0x4801 8074
GPIO_CLEARWKUENA	RW	32	0x4801 8080
GPIO_SETWKUENA	RW	32	0x4801 8084
GPIO_CLEARDATAOUT	RW	32	0x4801 8090
GPIO_SETDATAOUT	RW	32	0x4801 8094

Table 23−10. GPIO2 Module Registers List

Register Name	Type	Register Width (Bits)	Physical Address
GPIO_REVISION	R	32	0x4801 A000
GPIO_SYSCONFIG	RW	32	0x4801 A010
GPIO_SYSSTATUS	R	32	0x4801 A014
GPIO_IRQSTATUS1	RW	32	0x4801 A018
GPIO_IRQENABLE1	RW	32	0x4801 A01C
GPIO_WAKEUPENABLE	RW	32	0x4801 A020
GPIO_IRQSTATUS2	RW	32	0x4801 A028
GPIO_IRQENABLE2	RW	32	0x4801 A02C
GPIO_CTRL	RW	32	0x4801 A030
GPIO_OE	RW	32	0x4801 A034
GPIO_DATAIN	R	32	0x4801 A038
GPIO_DATAOUT	RW	32	0x4801 A03C
GPIO_LEVELDETECT0	RW	32	0x4801 A040
GPIO_LEVELDETECT1	RW	32	0x4801 A044
GPIO_RISINGDETECT	RW	32	0x4801 A048
GPIO_FALLINGDETECT	RW	32	0x4801 A04C
GPIO_DEBOUNCENABLE	RW	32	0x4801 A050
GPIO_DEBOUNCINGTIME	RW	32	0x4801 A054
GPIO_CLEARIRQENABLE1	RW	32	0x4801 A060
GPIO_SETIRQENABLE1	RW	32	0x4801 A064
GPIO_CLEARIRQENABLE2	RW	32	0x4801 A070
GPIO_SETIRQENABLE2	RW	32	0x4801 A074
GPIO_CLEARWKUENA	RW	32	0x4801 A080
GPIO_SETWKUENA	RW	32	0x4801 A084

Table 23−10. GPIO2 Module Registers List (Continued)

Register Name	Type	Register Width (Bits)	Physical Address
GPIO_CLEARDATAOUT	RW	32	0x4801 A090
GPIO_SETDATAOUT	RW	32	0x4801 A094

Table 23−11. GPIO3 Module Registers List

Register Name	Type	Register Width (Bits)	Physical Address
GPIO_REVISION	R	32	0x4801 C000
GPIO_SYSCONFIG	RW	32	0x4801 C010
GPIO_SYSSTATUS	R	32	0x4801 C014
GPIO_IRQSTATUS1	RW	32	0x4801 C018
GPIO_IRQENABLE1	RW	32	0x4801 C01C
GPIO_WAKEUPENABLE	RW	32	0x4801 C020
GPIO_IRQSTATUS2	RW	32	0x4801 C028
GPIO_IRQENABLE2	RW	32	0x4801 C02C
GPIO_CTRL	RW	32	0x4801 C030
GPIO_OE	RW	32	0x4801 C034
GPIO_DATAIN	R	32	0x4801 C038
GPIO_DATAOUT	RW	32	0x4801 C03C
GPIO_LEVELDETECT0	RW	32	0x4801 C040
GPIO_LEVELDETECT1	RW	32	0x4801 C044
GPIO_RISINGDETECT	RW	32	0x4801 C048
GPIO_FALLINGDETECT	RW	32	0x4801 C04C
GPIO_DEBOUNCENABLE	RW	32	0x4801 C050
GPIO_DEBOUNCINGTIME	RW	32	0x4801 C054
GPIO_CLEARIRQENABLE1	RW	32	0x4801 C060
GPIO_SETIRQENABLE1	RW	32	0x4801 C064
GPIO_CLEARIRQENABLE2	RW	32	0x4801 C070
GPIO_SETIRQENABLE2	RW	32	0x4801 C074
GPIO_CLEARWKUENA	RW	32	0x4801 C080
GPIO_SETWKUENA	RW	32	0x4801 C084
GPIO_CLEARDATAOUT	RW	32	0x4801 C090
GPIO_SETDATAOUT	RW	32	0x4801 C094

Table 23−12. GPIO4 Module Registers List

Register Name	Type	Register Width (Bits)	Physical Address
GPIO_REVISION	R	32	0x4801 E000
GPIO_SYSCONFIG	RW	32	0x4801 E010
GPIO_SYSSTATUS	R	32	0x4801 E014
GPIO_IRQSTATUS1	RW	32	0x4801 E018
GPIO_IRQENABLE1	RW	32	0x4801 E01C
GPIO_WAKEUPENABLE	RW	32	0x4801 E020
GPIO_IRQSTATUS2	RW	32	0x4801 E028
GPIO_IRQENABLE2	RW	32	0x4801 E02C
GPIO_CTRL	RW	32	0x4801 E030
GPIO_OE	RW	32	0x4801 E034
GPIO_DATAIN	R	32	0x4801 E038
GPIO_DATAOUT	RW	32	0x4801 E03C
GPIO_LEVELDETECT0	RW	32	0x4801 E040
GPIO_LEVELDETECT1	RW	32	0x4801 E044
GPIO_RISINGDETECT	RW	32	0x4801 E048
GPIO_FALLINGDETECT	RW	32	0x4801 E04C
GPIO_DEBOUNCENABLE	RW	32	0x4801 E050
GPIO_DEBOUNCINGTIME	RW	32	0x4801 E054
GPIO_CLEARIRQENABLE1	RW	32	0x4801 E060
GPIO_SETIRQENABLE1	RW	32	0x4801 E064
GPIO_CLEARIRQENABLE2	RW	32	0x4801 E070
GPIO_SETIRQENABLE2	RW	32	0x4801 E074
GPIO_CLEARWKUENA	RW	32	0x4801 E080
GPIO_SETWKUENA	RW	32	0x4801 E084
GPIO_CLEARDATAOUT	RW	32	0x4801 E090
GPIO_SETDATAOUT	RW	32	0x4801 E094

Table 23−13. General-Purpose Interface Top Registers List

Register Name	Type	Register Width (Bits)	Physical Address
IPGENERICOCPSPL_ REVISION	R	32	0x4801 9000
IPGENERICOCPSPL_ SYSCONFIG	RW	32	0x4801 9010
IPGENERICOCPSPL_ SYSSTATUS	R	32	0x4801 9014

Table 23−13. General-Purpose Interface Top Registers List (Continued)

Register Name	Type	Register Width (Bits)	Physical Address
IPGENERICOCPSPL_ IRQSTATUS	R	32	0x4801 9018
IPGENERICOCPSPL_GPO	RW	32	0x4801 9040
IPGENERICOCPSPL_GPI	R	32	0x4801 9050

Table 23−14. General-Purpose Interface L4 Interconnect Registers List

Register Name	Type	Register Width (Bits)	Physical Address
COMPONENT	R	32	0x4801 B000
AGENT_CONTROL	RW	32	0x4801 B020
AGENT_STATUS	R	32	0x4801 B028

23.4 GPIO Functional Description

Figure 23−7 shows the GPIO interface.

Figure 23−7. GPIO Interface

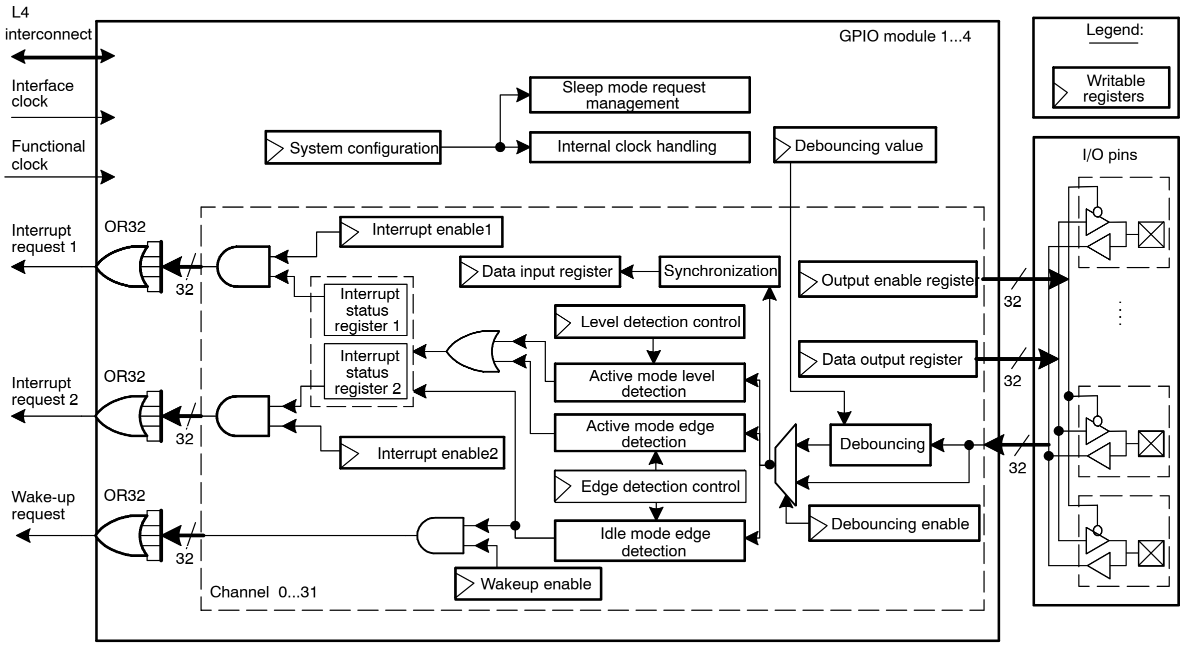
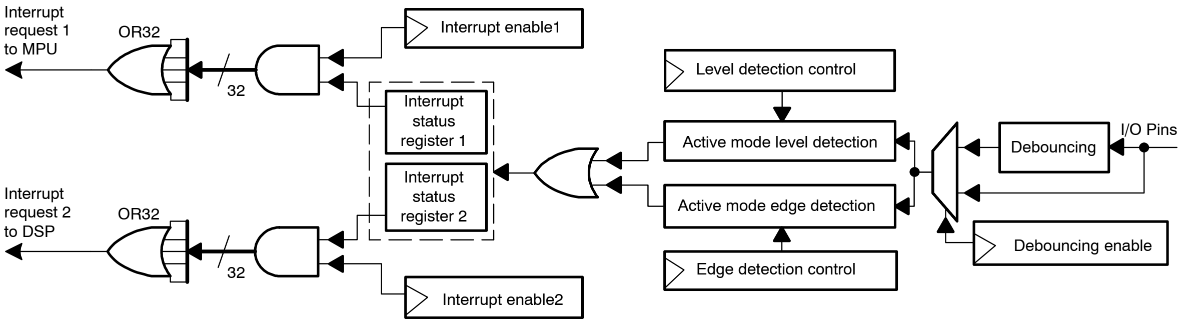


Figure 23−7 shows the GPIO modules in the general-purpose interface block diagram with their configuration registers and their main functional paths:

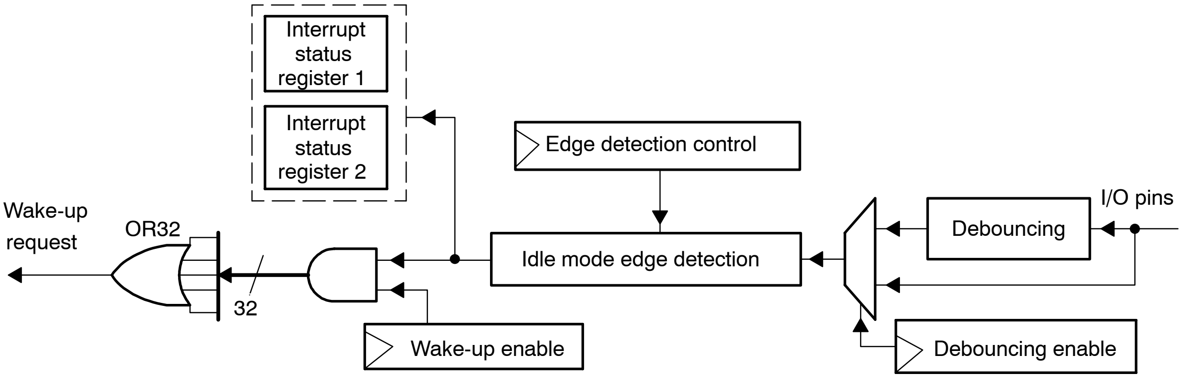
- The synchronous path (for active mode operations) is used to generate a synchronous interrupt request on expected event detection on any input GPIO; the synchronous interrupt request lines 1 and 2 are active according to their respective interrupt enable 1 and 2 registers. See Figure 23−8.

Figure 23−8. Synchronous Path


Interrupt request 1 to MPU
OR32
32
Interrupt enable1
Interrupt status register 1
Interrupt status register 2
Level detection control
Active mode level detection
Active mode edge detection
Edge detection control
Debouncing
I/O Pins
Debouncing enable
Interrupt request 2 to DSP
OR32
32
Interrupt enable2

- The asynchronous path (for idle mode operations) is used to generate an asynchronous wake-up request on expected edge detection on any input GPIO; the asynchronous wake-up request line is active according to the wake-up enable register. See Figure 23−9.

Figure 23−9. Asynchronous Path



- ❑ The blocks handle the internal clock (clock gating) and manage the sleep mode request/acknowledge protocol (enabling the synchronous path in active mode and the asynchronous path in idle mode).

23.4.1 Operational Description

23.4.1.1 Clocking and Reset Strategy

Clocks, Gating, and Active Edge Definitions

Each GPIO module uses two clocks: the debounce clock and the interface clock.

The debounce clock (GPIOn_DBCLK, where *n* is 1, 2, 3, and 4) comes from the PRCM module. It is used for the debounce cell. This cell can sample the input line, and it filters the input level using a programmed delay.

The interface clock (GPIO_ICLK) comes from the PRCM module. It is used through the entire GPIO module (except in the debounce cell). Two clock domains are defined: the L4 interconnect and the internal logic. Each clock domain can be controlled independently. In active mode, event detection (level or transition) is performed in the GPIO module using the interface clock. Detection precision is set by the frequency of this clock and the selected internal gating scheme. Sampling operations for data capture and event detection are performed using the rising edge. The data loaded in the data output register (GPIO_DATAOUT) is set at the output GPIO pins synchronously with the rising edge of the interface clock.

Clock-gating features allow adapting module power consumption to the activity by reducing module power consumption when the module is not used.

Five clock-gating features are available:

- ❑ Internal interface clock gating: The clock for L4 interconnect logic can be gated when the module is not accessed, if the AUTOIDLE bit GPIO_ SYSCONFIG[0] is set. Otherwise, this logic is free-running on the interface clock.
- ❑ Clock gating for input data sample logic: The clock for input data sample logic can be gated when the data in the register is not accessed.

- Clock gating for event-detection logic: Each GPIO module implements 4 clock groups used for logic in synchronous event 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 (cut off; see Section 23.5.1, *Power Saving by Grouping the Edge/Level Detection*). All channels are also gated using a one-out-of-*N* scheme. *N* is the GATINGRATIO bits GPIO_CTRL[2:1] and can take the values 1 (00), 2 (01), 4 (10), or 8 (11). 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 between 2 and 8, this logic runs at a frequency equal to the interface clock frequency divided by *N*.
- Inactive mode: In inactive mode, all internal clock paths are gated.
- Disabled mode: In disabled mode, all internal clock paths not used for the L4 interconnect are gated. The DISABLEMODULE bit GPIO_CTRL[0] controls a clock-gating feature at the module level. When set (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. Interface clock gating is controlled with the AUTOIDLE bit GPIO_SYSCONFIG[0]. This bit is used for power saving when the module is idle. This bit has precedence over all other internal configuration bits

All GPIO registers are accessible synchronously with the interface clock.

Idle Mode Request and Acknowledge

The general-purpose interface has four identical idle mode request/acknowledge (handshake) mechanisms with the PRCM module, one per GPIO module.

GPIOn_IDLEREQ and GPIOn_SIDLEACK (where *n* is 1, 2, 3, and 4) allow each GPIO module to enter into different idle modes and save power.

On an idle mode request (GPIOn_IDLEREQ, where *n* is 1, 2, 3, and 4, and is active) issued by the PRCM module (see Chapter 5), the GPIO modules go to idle mode according to the IDLEMODE bits GPIO_ SYSCONFIG[4:3].

- No-idle: If the IDLEMODE field sets no-idle mode (01), the module does not go to the idle mode, and the idle acknowledge (GPIOn_SIDLEACK, where n is 1, 2, 3, and 4) is never sent.
- Force-idle: If the IDLEMODE field sets force-idle mode (00), the module goes to inactive mode independently of the internal module state, and the idle acknowledge (GPIOn_SIDLEACK) is unconditionally sent. In force-idle mode, the module is in inactive mode and its wake-up feature is totally inhibited.
- Smart-idle: If the IDLEMODE field sets smart-idle mode (10), the module evaluates its internal capability to have the interface clock switched off.

When there is no internal activity (the data input register, GPIO_DATAIN, completes capture of the input GPIO pins, there is no pending interrupt, all interrupt status bits are cleared, and there is no write access to the GPIO_DEBOUNCETIME register waiting for synchronization), the idle acknowledge is asserted and the module enters idle mode, ready to issue a wake-up request when the expected transition occurs on an enabled GPIO input pin. This wake-up request is effectively sent only if the ENAWAKEUP bit GPIO_SYSCONFIG[2] enables the GPIO wake-up capability. When the system is awake, the idle request goes inactive, the wake-up request signal (GPIO_SWAKEUP) is deasserted (if the GPIO triggered the wakeup of the system), and the asynchronous wake-up request is reflected into the synchronous interrupt status registers (GPIO_IRQSTATUS1 and GPIO_IRQSTATUS2).

When the module acknowledges the idle mode request (GPIOn_SIDLEACK is sent), the interface clock can be stopped.

When smart-idle mode is acknowledged, the following situations are available (based on the module configuration):

- The interface clock (GPIO_ICLK) and debounce clock (GPIOn_DBCLK) can be cut off if the debounce feature is disabled.
- The interface clock can be cut off if the debounce feature is enabled.

When force-idle mode is acknowledged, the interface clock and the debounce clock can be cut off.

Before going to idle, software must make sure all the events in the GPIO_IRQSTATUS1 and GPIO_IRQSTATUS2 registers are cleared, if the WAKEUPENABLE bit is set to enable.

Reset

The general-purpose interface reset can be done through the L4 interconnect reset (hardware reset) or through the setting of a dedicated configuration bit (software reset) in each GPIO module.

The hardware reset signal (WKUP_RST_N; see Chapter 5, *Power, Reset, and Clock Management*) has a global reset action on the GPIO modules of the general-purpose interface. For a module, all configuration registers, all D flip-flops (DFFs) clocked with the interface clock, and all internal state-machines are reset when the hardware reset is active (low level). This hardware reset signal is synchronous with the interface clock and internally synchronized with the debounce clock domain to guarantee a correct reset of the DFFs.

In each GPIO module, the RESETDONE bit GPIO_SYSSTATUS[0] monitors the internal reset status; it is set when the reset is complete.

Each GPIO module of the general-purpose interface has its own software reset capability through the SOFTRESET bit GPIO_SYSCONFIG[1], which controls the software reset. Writing 1 to this bit resets the module. Bit value 1 remains until the reset is complete. When the software reset is complete, the SOFTRESET bit automatically returns to 0.

The software reset has the same effect as the hardware reset signal, and the RESETDONE bit GPIO_SYSSTATUS[0] is updated in the same condition.

23.4.1.2 Interrupt and Wake-Up Features

Synchronous Path: Interrupt Request Generation

The general-purpose interface has eight interrupt lines (two interrupt lines per a GPIO module instance). These eight interrupt signals are GPIOn_MPU_IRQ (used by the MPU subsystem) and GPIOn_DSP_IRQ (used by the DSP subsystem), where *n* is 1, 2, 3, and 4.

Synchronous interrupt requests from each channel are processed by two identical interrupt-generation submodules to be used independently by the DSP subsystem and the MPU subsystem. Each submodule controls its own synchronous interrupt request line and has its own interrupt enable (GPIO_IRQENABLE1 or GPIO_IRQENABLE2) and interrupt status (GPIO_IRQSTATUS1 or GPIO_IRQSTATUS2) registers. The interrupt enable register selects the channel(s) considered for interrupt request generation, and the interrupt status register determines which channel(s) activated the interrupt request. Event detection on GPIO channels is reflected into the interrupt status registers independently of interrupt-enable register content.

In active mode, when the GPIO configuration registers are set to enable interrupt generation (see Section 23.5.3, *Interrupt and Wakeup*), a synchronous path samples the transitions and levels on the input GPIO with the internally gated interface clock (see Section 23.4.1.1, *Clocking and Reset Strategy*). When an event matches the programmed settings (see Section 23.5.3), the corresponding bit in the interrupt status register is set to 1 and, on the following interface clock cycle, interrupt lines 1 and/or 2 are activated (depending on the interrupt enable registers).

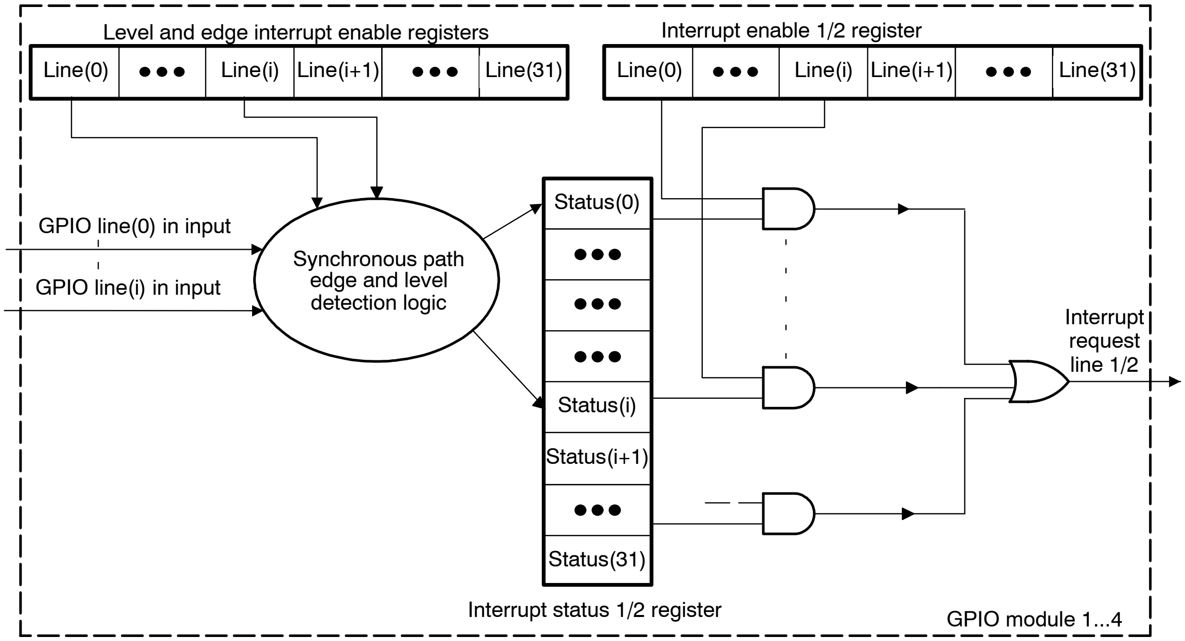
Because of the sampling operation, the minimum pulse width on the input GPIO to trigger a synchronous interrupt request is two times the internally gated interface clock period (the internally gated interface clock period is equal to *N* times the interface clock period; see Section 23.4.1.1, *Clocking and Reset Strategy*). 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, latency depends on the debouncing time register (GPIO_DEBOUNCINGTIME) value (see Section 23.5.5, *Debouncing Time*) and must be less than four internally gated interface clock cycles + one interface clock cycle + GPIO_DEBOUNCINGTIME 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 DSP interrupt controller. Figure 23−10 shows the generation of interrupt requests.

Figure 23−10. Interrupt Request Generation



Asynchronous Path: Wake-Up Request Generation

The general-purpose interface has four wake-up lines (one wake-up line per GPIO module instance). The four wake-up signals are internally ORed together in the OMAP2420 to produce a unique wake-up request (GPIO_SWAKEUP) to the PRCM module.

Asynchronous wake-up requests from input channels are merged to issue a single wake-up signal to the system for each GPIO module. The wake-up enable register (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 (GPIO_IRQSTATUS1 and GPIO_IRQSTATUS2).

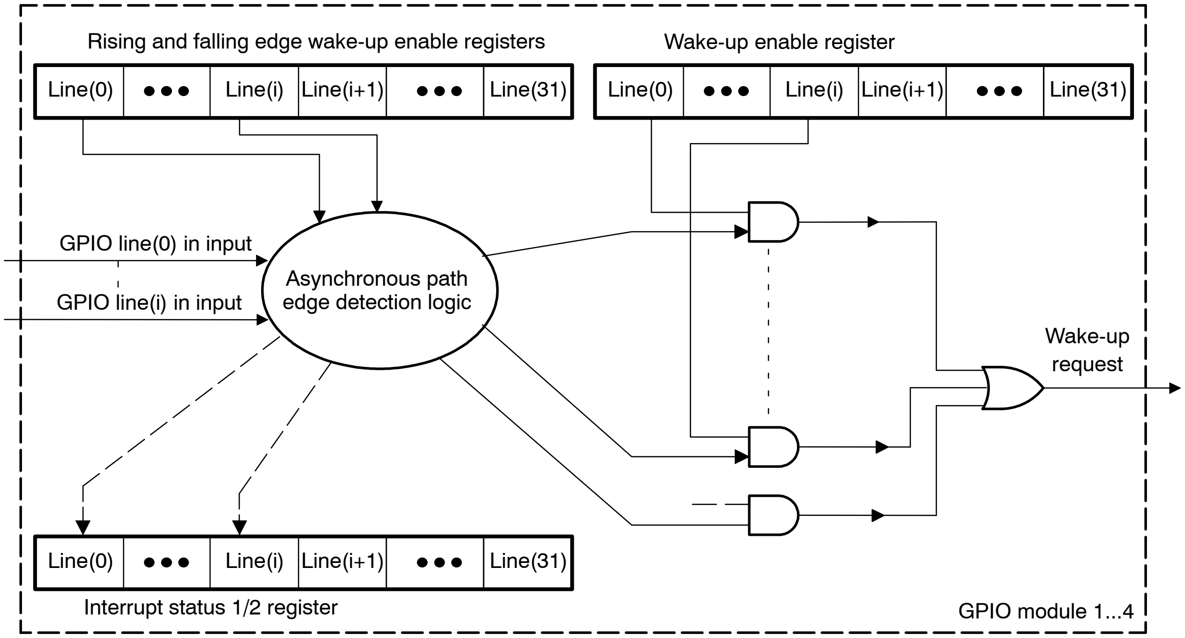
In idle mode (interface clock is shut down, the GPIO configuration registers have been programmed, see Section 23.5.3, *Interrupt and Wakeup*), an asynchronous path detects the expected transition(s) on GPIO input (according to the register programming) and activates an asynchronous wake-up request by the GPIO_SWAKEUP sideband signal if the wake-up enable register is set. As shown in Figure 23−11, there is only one external wake-up line, because all wake-up sources are merged. When the system wakes up, the interface clock is restarted and depending the input GPIO pin that triggered the wake-up request, the corresponding bit in the interrupt status registers is synchronously set to 1. On the following internal clock cycle, the interrupt lines 1 and/or 2 are active low when the corresponding bits in the interrupt enable registers (GPIO_IRQENABLE1 and GPIO_IRQENABLE2) are set.

Note:

- ☐ If the debouncing is not enabled, there is no minimum input pulse width to trigger the wake-up request, because there is no sampling operation.
- ☐ If debouncing is used, the minimum pulse width is set by the debouncing specified time.
- ☐ The ENAWAKEUP bit GPIO_SYSCONFIG[2] allows enabling or disabling of the GPIO wake-up feature globally; if this bit is 0, the wake-up enable register (GPIO_WAKEUPENABLE) has no effect.

Figure 23−11 shows the generation of wake-up requests.

Figure 23−11. Wake-Up Request Generation



Interrupt (or Wake-Up) Line Release

When the host processor (MPU and/or DSP subsystem in the OMAP2420) receives an interrupt request issued by the GPIO module, it can read the corresponding interrupt status register to find out 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 not masked by the interrupt enable register are not cleared), the interrupt line is reasserted.

23.5 GPIO Programming Model

23.5.1 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 GPIO input pins generates a separate enable signal depending on the edge/level detection register setting (because input is 32 bits, four groups of eight inputs are defined for each GPIO module). If a group requires no edge/level detection, the corresponding clock is gated (cut off).

If any of the following registers:

- GPIO_LEVELDETECT0
- GPIO_LEVELDETECT1
- GPIO_RISINGDETECT
- GPIO_FALLINGDETECT

is set to:

- 0x01 01 01 01, all clocks are active (power consumption is high).
- 0x00 00 00 FF, a single clock is active (power saving).

Note:

When the clocks are enabled by writing into the GPIO_LEVELDETECT0/GPIO_LEVELDETECT1/GPIO_RISINGDETECT/GPIO_FALLINGDETECT registers, detection is started after five clock cycles. This period is required to clean the synchronization edge/level detection pipeline. This mechanism is independent of each clock group. If the clock is already started and a new setting is performed, it is recommended to first set the new detection required, and then to disable the previous setting (if necessary). This way, the corresponding clock is not cut off, and detection starts immediately.

23.5.2 Set and Clear Instructions

23.5.2.1 Description

The GPIO module implements the set and clear protocol register update for the data output (GPIO_DATAOUT), interrupt enable (GPIO_IRQENABLE1 and GPIO_IRQENABLE2), and wake-up enable (GPIO_WAKEUPENABLE) registers. This protocol is an alternative to atomic test and set operations. It 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). Concerned registers can be accessed in two ways:

- The standard way: Full register read and write operations at the primary register address.
- The set and clear way (recommended): Separate addresses are provided to set (and to clear) bits in registers. Writing 1 at these addresses sets (or

clears) the corresponding bit into the equivalent register, writing 0 has no effect.

So, for these registers, three addresses are defined for one unique physical register. Reading these addresses has the same effect and returns the register value.

23.5.2.2 Clear Instruction

Clear Register Addresses

Clear interrupt enable registers (GPIO_CLEARIRQENABLE1 and GPIO_CLEARIRQENABLE2):

- A write operation in the clear interrupt enable1 (or 2) register clears the corresponding bit in the interrupt enable1 (or 2) register when the written bit is 1, whereas a written bit at 0 has no effect.
- A read of the clear interrupt enable1 (or 2) register returns the value of the interrupt enable1 (or 2) register.

Clear wake-up enable register (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, whereas 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 (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, whereas a written bit at 0 has no effect.
- A read of the clear data output register returns the value of the data output register.

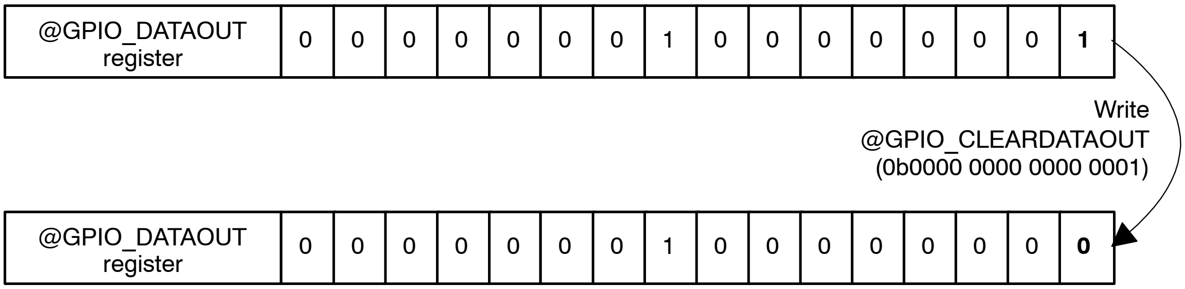
Clear Instruction Example

Assume that the data output register (or one of the interrupt/wake-up enable registers) 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.

Figure 23−12 shows this clear instruction example.

Figure 23−12. Write @GPIO_CLEARDATAOUT Register Example



23.5.2.3 Set Instruction

Set Register Addresses

Set interrupt enable registers (GPIO_SETIRQENABLE1 and GPIO_ SETIRQENABLE2):

- A write operation in the set interrupt enable1 (or 2) register sets the corresponding bit in the interrupt enable1 (or 2) register when the written bit is 1, whereas a written bit at 0 has no effect.
- A read of the set interrupt enable1 (or 2) register returns the value of the interrupt enable1 (or 2) register.

Set wake-up enable register (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, whereas 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 (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, whereas a written bit at 0 has no effect.
- A read of the set data output register returns the value of the data output register.

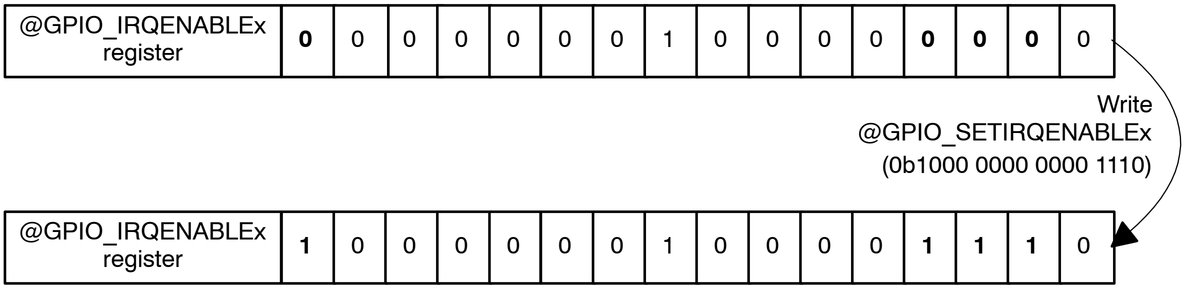
Set Instruction Example

Assume that the interrupt enable1 (or 2) 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 2) register (or at the address of the set data output register). After this write operation, a reading of the interrupt enable1 (or 2) register (or the data output register) returns 0b1000 0001 0000 1110; the bits 15, 3, 2, and 1 are set.

Figure 23−13 shows this set instruction example.

Figure 23−13. Write @GPIO_SETIRQENABLEx Register Example


@GPIO_IRQENABLEx register
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
Write @GPIO_SETIRQENABLEx (0b1000 0000 0000 1110)
@GPIO_IRQENABLEx register
1 0 0 0 0 0 0 1 0 0 0 0 1 1 1 0

The set wake-up enable register offers the same feature with the wake-up enable register.

23.5.3 Interrupt and Wakeup

23.5.3.1 Involved Configuration Registers

Interrupt enable registers (GPIO_IRQENABLE1 and GPIO_IRQENABLE2):

- The interrupt enable1 (or 2) register allows you to mask the expected transition on input GPIO from generating an interrupt request on line1 (or 2). 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 setting or clearing of specific bits of these registers with a single write access to the corresponding set interrupt enable1 (or 2) registers (or to the clear interrupt enable1 [or 2] registers) address (see Section 23.5.2, *Set and Clear Instructions*).

Wake-up enable register (GPIO_WAKEUPENABLE):

- The wake-up enable register allows you to mask the expected transition on input GPIO from generating a wakeup 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 using the alternate set and clear protocol register update feature. This feature enables setting or clearing of 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 23.5.2, *Set and Clear Instructions*).

Interrupt status registers (GPIO_IRQSTATUS1 and GPIO_IRQSTATUS2):

- The interrupt status1 (or 2) register is used to determine which input GPIO pin triggered the interrupt line1 (or 2) 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 wakeup). To reset a bit in this register, write 1 to the appropriate bit. However, writing 1 to the interrupt

status1 (or 2) register does not generate an interrupt. Writing 0 to a bit in this register does not change the value. The interrupt status1 (or 2) register is synchronous with the interface clock. In idle mode, the event is detected through an asynchronous path, and the corresponding bits in the interrupt status1 and 2 registers are set when the GPIO module is awake.

23.5.3.2 Description

To generate (by a GPIO module) an interrupt request to a host processor (the MPU and/or DSP subsystem in the OMAP2420) at a defined event (level or logic transition) occurring on a GPIO pin (interrupt source), the GPIO configuration registers must be programmed as follows:

- The GPIO pin must be configured as an input by the output enable register (GPIO_OE[x] bit = 1).
- Interrupts for the GPIO channel must be enabled in the interrupt 1 enable (GPIO_IRQENABLE1) register and/or the interrupt 2 enable (GPIO_IRQENABLE2) register.
- The expected event(s) on GPIO input to trigger the interrupt request must be selected in the low-level interrupt enable (GPIO_LEVELDETECT0), high-level interrupt enable (GPIO_LEVELDETECT1), rising edge interrupt/wake-up enable (GPIO_RISINGDETECT), and falling edge interrupt/wake-up enable (GPIO_FALLINGDETECT).

To generate (by a GPIO module) a wake-up request to a host processor (the MPU and/or DSP subsystem in the OMAP2420) at a defined event (logic transition) occurring on a GPIO pin (wake-up source), the GPIO configuration registers must be programmed as follows:

- The GPIO pin must be configured as an input by the output enable register (GPIO_OE[x] bit = 1).
- The GPIO channel must be enabled in the wake-up enable register (GPIO_WAKEUPENABLE).
- The expected event(s) on GPIO input to trigger the wake-up request must be selected in the rising edge interrupt/wake-up enable (GPIO_RISINGDETECT) and falling edge interrupt/wake-up enable (GPIO_FALLINGDETECT) registers. Wake-up request can be generated only on rising and/or on falling transitions.
- The ENAWAKEUP bit GPIO_SYSCONFIG[2] must be set to 1 to enable wake-up request generation on the expected transition occurring on the GPIO input pins.

For instance, interrupt generation on both edges on input *k* is configured by setting to 1 the k^{th} bit in the GPIO_RISINGDETECT and GPIO_FALLINGDETECT registers and interrupt enabling for one or both interrupt lines (GPIO_IRQENABLE1 and GPIO_IRQENABLE2).

Resetting Status Bit

After servicing the interrupt, the status bit in the interrupt status register (GPIO_IRQSTATUS1 or GPIO_IRQSTATUS2) should be reset and the interrupt line should be 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 (GPIO_IRQENABLE1 or GPIO_IRQENABLE2), to avoid unexpected interrupts when enabling an interrupt for the GPIO channel.

Note:

All interrupt (or wake-up) sources (the 32 input GPIO channels) are merged (see Figure 23−10 and Figure 23−11) to issue two synchronous interrupt requests 1 and 2, and a single asynchronous wake-up request in each GPIO module. All wake-up signals (one signal to a GPIO module) are internally ORed together in the OMAP2420 to produce a unique wake-up request (GPIO_SWAKEUP) to the PRCM module.

23.5.4 Data Input (Capture)/Output (Drive)

The output enable register (GPIO_OE) controls output capability for each pin. At reset, all GPIO-related pins configured as input and output capabilities are configured as disabled. This register is not used in the module. Its only function is to carry the pad configuration. When the application uses a pin as an output and does require interrupt/wake-up generation from this pin, the application must correctly configure the wake-up enable (GPIO_WAKEUP ENABLE) and the interrupt enable (GPIO_IRQENABLE1 and GPIO_IRQ ENABLE2) registers.

When configured as an output (GPIO_OE[x] bit = 0), the value in the GPIO_DATAOUT[x] bit in the data output register (GPIO_DATAOUT) 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 using the alternate set and clear protocol register update feature. This feature enables setting or clearing of specific bits of this register with a single write access to the set output data register (GPIO_SETDATAOUT) or to the clear output data register (GPIO_CLEARDATAOUT) address (see Section 23.5.2, *Set and Clear Instructions*).

When configured as an input (GPIO_OE[x] bit = 1), the state of the input can be read from the corresponding GPIO_DATAIN[x] bit in the data input register (GPIO_DATAIN). Input data is sampled synchronously with the interface clock and then captured in the data input register synchronously with the interface clock (see Section 23.4.1.1, *Clocking and Reset Strategy*). Therefore, after changing, GPIO pin levels are captured into this register after two interface clock cycles (required to synchronize and write data).

23.5.5 Debouncing Time

The debouncing value register (GPIO_DEBOUNCINGTIME) is used to set the debouncing time for all input lines in the GPIO module. The value is global for all ports of one GPIO module, allowing up to four different debouncing values. The debounce cell runs with the debounce clock (32 kHz). This register represents the number of the clock cycle(s) (one cycle is 31μs long) to be used.

The following formula describes the required input stable time to be propagated to the debounced output:

Required input line stable = (DEBOUNCING VALUE + 1) * 31μs

where $0 \leq$ DEBOUNCING VALUE ≤ 255

23.6 GPIO Registers

This section summarizes the hardware interface for the GPIO product. Each module instance in the design is shown in this section, with the module register map and bit definitions for each bit field.

Table 23−15 shows the base address and address space for the GPIO module instances.

Table 23−15. Instance Summary

Module Name	Base Address	Size
GPIO1	0x4801 8000	1K byte
OSPL1	0x4801 9000	256 bytes
GPIO2	0x4801 A000	1K byte
GPIO3	0x4801 C000	1K byte
GPIO4	0x4801 E000	1K byte

23.6.1 GPIO 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 23−16 through Table 23−19 list the GPIO registers. Table 23−21 through Table 23−46 describe the register bits.

Table 23−16. GPIO1 Module Registers

Register Name	Type	Register Width (Bits)	Offset
GPIO_REVISION	R	32	0x000
GPIO_SYSCONFIG	RW	32	0x010
GPIO_SYSSTATUS	R	32	0x014
GPIO_IRQSTATUS1	RW	32	0x018
GPIO_IRQENABLE1	RW	32	0x01C
GPIO_WAKEUPENABLE	RW	32	0x020
GPIO_IRQSTATUS2	RW	32	0x028
GPIO_IRQENABLE2	RW	32	0x02C
GPIO_CTRL	RW	32	0x030
GPIO_OE	RW	32	0x034
GPIO_DATAIN	R	32	0x038
GPIO_DATAOUT	RW	32	0x03C
GPIO_LEVELDETECT0	RW	32	0x040
GPIO_LEVELDETECT1	RW	32	0x044
GPIO_RISINGDETECT	RW	32	0x048

Table 23−16. GPIO1 Module Registers (Continued)

Register Name	Type	Register Width (Bits)	Offset
GPIO_FALLINGDETECT	RW	32	0x04C
GPIO_DEBOUNCENABLE	RW	32	0x050
GPIO_DEBOUNCINGTIME	RW	32	0x054
GPIO_CLEARIRQENABLE1	RW	32	0x060
GPIO_SETIRQENABLE1	RW	32	0x064
GPIO_CLEARIRQENABLE2	RW	32	0x070
GPIO_SETIRQENABLE2	RW	32	0x074
GPIO_CLEARWKUENA	RW	32	0x080
GPIO_SETWKUENA	RW	32	0x084
GPIO_CLEARDATAOUT	RW	32	0x090
GPIO_SETDATAOUT	RW	32	0x094

Table 23−17. GPIO2 Module Registers

Register Name	Type	Register Width (Bits)	Offset
GPIO_REVISION	R	32	0x000
GPIO_SYSCONFIG	RW	32	0x010
GPIO_SYSSTATUS	R	32	0x014
GPIO_IRQSTATUS1	RW	32	0x018
GPIO_IRQENABLE1	RW	32	0x01C
GPIO_WAKEUPENABLE	RW	32	0x020
GPIO_IRQSTATUS2	RW	32	0x028
GPIO_IRQENABLE2	RW	32	0x02C
GPIO_CTRL	RW	32	0x030
GPIO_OE	RW	32	0x034
GPIO_DATAIN	R	32	0x038
GPIO_DATAOUT	RW	32	0x03C
GPIO_LEVELDETECT0	RW	32	0x040
GPIO_LEVELDETECT1	RW	32	0x044
GPIO_RISINGDETECT	RW	32	0x048
GPIO_FALLINGDETECT	RW	32	0x04C
GPIO_DEBOUNCENABLE	RW	32	0x050
GPIO_DEBOUNCINGTIME	RW	32	0x054
GPIO_CLEARIRQENABLE1	RW	32	0x060
GPIO_SETIRQENABLE1	RW	32	0x064
GPIO_CLEARIRQENABLE2	RW	32	0x070

Table 23−17. GPIO2 Module Registers (Continued)

Register Name	Type	Register Width (Bits)	Offset
GPIO_SETIRQENABLE2	RW	32	0x074
GPIO_CLEARWKUENA	RW	32	0x080
GPIO_SETWKUENA	RW	32	0x084
GPIO_CLEARDATAOUT	RW	32	0x090
GPIO_SETDATAOUT	RW	32	0x094

Table 23−18. GPIO3 Module Registers

Register Name	Type	Register Width (Bits)	Offset
GPIO_REVISION	R	32	0x000
GPIO_SYSCONFIG	RW	32	0x010
GPIO_SYSSTATUS	R	32	0x014
GPIO_IRQSTATUS1	RW	32	0x018
GPIO_IRQENABLE1	RW	32	0x01C
GPIO_WAKEUPENABLE	RW	32	0x020
GPIO_IRQSTATUS2	RW	32	0x028
GPIO_IRQENABLE2	RW	32	0x02C
GPIO_CTRL	RW	32	0x030
GPIO_OE	RW	32	0x034
GPIO_DATAIN	R	32	0x038
GPIO_DATAOUT	RW	32	0x03C
GPIO_LEVELDETECT0	RW	32	0x040
GPIO_LEVELDETECT1	RW	32	0x044
GPIO_RISINGDETECT	RW	32	0x048
GPIO_FALLINGDETECT	RW	32	0x04C
GPIO_DEBOUNCENABLE	RW	32	0x050
GPIO_DEBOUNCINGTIME	RW	32	0x054
GPIO_CLEARIRQENABLE1	RW	32	0x060
GPIO_SETIRQENABLE1	RW	32	0x064
GPIO_CLEARIRQENABLE2	RW	32	0x070
GPIO_SETIRQENABLE2	RW	32	0x074
GPIO_CLEARWKUENA	RW	32	0x080
GPIO_SETWKUENA	RW	32	0x084
GPIO_CLEARDATAOUT	RW	32	0x090
GPIO_SETDATAOUT	RW	32	0x094

Table 23−19. GPIO4 Module Registers

Register Name	Type	Register Width (Bits)	Offset
GPIO_REVISION	R	32	0x000
GPIO_SYSCONFIG	RW	32	0x010
GPIO_SYSSTATUS	R	32	0x014
GPIO_IRQSTATUS1	RW	32	0x018
GPIO_IRQENABLE1	RW	32	0x01C
GPIO_WAKEUPENABLE	RW	32	0x020
GPIO_IRQSTATUS2	RW	32	0x028
GPIO_IRQENABLE2	RW	32	0x02C
GPIO_CTRL	RW	32	0x030
GPIO_OE	RW	32	0x034
GPIO_DATAIN	R	32	0x038
GPIO_DATAOUT	RW	32	0x03C
GPIO_LEVELDETECT0	RW	32	0x040
GPIO_LEVELDETECT1	RW	32	0x044
GPIO_RISINGDETECT	RW	32	0x048
GPIO_FALLINGDETECT	RW	32	0x04C
GPIO_DEBOUNCENABLE	RW	32	0x050
GPIO_DEBOUNCINGTIME	RW	32	0x054
GPIO_CLEARIRQENABLE1	RW	32	0x060
GPIO_SETIRQENABLE1	RW	32	0x064
GPIO_CLEARIRQENABLE2	RW	32	0x070
GPIO_SETIRQENABLE2	RW	32	0x074
GPIO_CLEARWKUENA	RW	32	0x080
GPIO_SETWKUENA	RW	32	0x084
GPIO_CLEARDATAOUT	RW	32	0x090
GPIO_SETDATAOUT	RW	32	0x094

Table 23−20. General-Purpose Interface Top (OSPL1) Registers List

Register Name	Type	Register Width (Bits)	Offset
IPGENERICOCPSPL_REVISION	R	32	0x000
IPGENERICOCPSPL_SYSCONFIG	RW	32	0x010
IPGENERICOCPSPL_SYSSTATUS	R	32	0x014
IPGENERICOCPSPL_IRQSTATUS	R	32	0x018
IPGENERICOCPSPL_GPO	RW	32	0x040
IPGENERICOCPSPL_GPI	R	32	0x050

The write latency for all the R/W registers is immediate (with respect to the interface clock).

Note:

If two write accesses in the DEBOUNCE VALUE register are performed in less than 2 debounce clock cycles (32 kHz) + 4 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 Section 23.6.2, when a single register carries an individual configuration or setting that applies to all 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 forth, up to 31.

23.6.2 GPIO Register Descriptions

Table 23–21. GPIO_REVISION

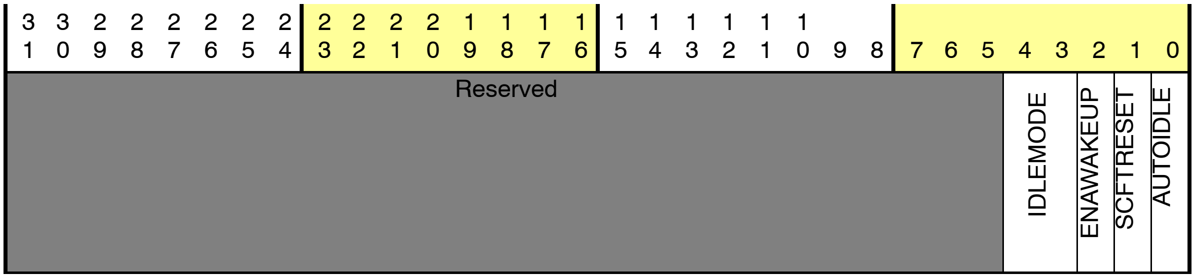
Address Offset	0x000		
Physical Address	0x4801 8000	**Instance**	GPIO1
	0x4801 A000		GPIO2
	0x4801 C000		GPIO3
	0x4801 E000		GPIO4
Description	This register contains the IP revision code.		
Type	R		

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	
7:0	GPIOREVISION	IP revision [7:4]: Major revision [3:0]: Minor revision Examples: 0x10 for 1.0, 0x21 for 2.1	R	

Table 23–22. GPIO_SYSCONFIG

Address Offset	0x010		
Physical Address	0x4801 8010	**Instance**	GPIO1
	0x4801 A010		GPIO2
	0x4801 C010		GPIO3
	0x4801 E010		GPIO4
Description	This register controls the L4 interconnect parameters.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:5	Reserved	Write 0s for future compatibility. Read returns 0.	R	0x0000000
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 capability enabled/disabled 0x0: Wake-up disable 0x1: Wake-up enable	RW	0
1	SOFTRESET	Software reset. This bit is automatically reset by the hardware. Read 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 interface clock-gating strategy is applied, based on L4 interconnect activity.	RW	0

Note: When the AUTOIDLE bit GPIO_SYSCONFIG[0] is set, the GPIO_DATAIN read command has 3 interface clock cycle latency because of the data in sample gating mechanism. When the AUTOIDLE bit GPIO_SYSCONFIG[0] is not set, the GPIO_DATAIN read command has 2 interface clock cycle latency.

Table 23−23. GPIO_SYSSTATUS

Address Offset	0x014		
Physical Address	0x4801 8014	**Instance**	GPIO1
	0x4801 A014		GPIO2
	0x4801 C014		GPIO3
	0x4801 E014		GPIO4
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	Read returns 0.	R	0x000000
7:1	Reserved	Read returns 0. Reserved for OCP socket status information	R	0x00
0	RESETDONE	Internal reset monitoring 0x0: Internal module reset is ongoing. 0x1: Reset completed[1]	R	–

[1] During reset, the value is 0, but the value read just after the reset is 1, because ICLK/FCLK is already operating.

Table 23−24. GPIO_IRQSTATUS1

Address Offset	0x018		
Physical Address	0x4801 8018	**Instance**	GPIO1
	0x4801 A018		GPIO2
	0x4801 C018		GPIO3
	0x4801 E018		GPIO4
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 0x0: IRQ pin N not triggered 0x1: IRQ pin N triggered	RW	0x00000000

Table 23–25. GPIO_IRQENABLE1

Address Offset	0x01C		
Physical Address	0x4801 801C	**Instance**	GPIO1
	0x4801 A01C		GPIO2
	0x4801 C01C		GPIO3
	0x4801 E01C		GPIO4
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 0x0: Disable IRQ pin N. 0x1: Enable IRQ pin N.	RW	0x00000000

Table 23–26. GPIO_WAKEUPENABLE

Address Offset	0x020		
Physical Address	0x4801 8020	**Instance**	GPIO1
	0x4801 A020		GPIO2
	0x4801 C020		GPIO3
	0x4801 E020		GPIO4
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 pin N. 0x1: Enable wakeup pin N.	RW	0x00000000

Note: In force-idle mode (see Section 23.2.1, *General-Purpose Interface Functional Interfaces*), the module wake-up feature is totally inhibited. Wake-up generation can also be gated at module level using the ENAWAKEUP bit in the GPIO_SYSCONFIG register.

Table 23−27. GPIO_IRQSTATUS2

Address Offset	0x028		
Physical Address	0x4801 8028	**Instance**	GPIO1
	0x4801 A028		GPIO2
	0x4801 C028		GPIO3
	0x4801 E028		GPIO4
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	RW	0x00000000
		0x0: IRQ pin N not triggered		
		0x1: IRQ pin N triggered		

Table 23−28. GPIO_IRQENABLE2

Address Offset	0x02C		
Physical Address	0x4801 802C	**Instance**	GPIO1
	0x4801 A02C		GPIO2
	0x4801 C02C		GPIO3
	0x4801 E02C		GPIO4
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	RW	0x00000000
		0x0: Disable IRQ pin N.		
		0x1: Enable IRQ pin N.		

Table 23−29. GPIO_CTRL

Address Offset	0x030		
Physical Address	0x4801 8030	**Instance**	GPIO1
	0x4801 A030		GPIO2
	0x4801 C030		GPIO3
	0x4801 E030		GPIO4
Description	This register controls the clock-gating 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																													GATINGRATIO		DISABLEMODE

Bits	Field Name	Description		Type	Reset
31:3	Reserved	Read returns 0.		R	0x00000000
2:1	GATINGRATIO	Gating ratio		RW	0x1
		0x0:	No gating and logic clock is interface clock.		
		0x1:	Gated clock is interface clock divided by 2.		
		0x2:	Gated clock is interface clock divided by 4.		
		0x3:	Gated clock is interface clock divided by 8.		
0	DISABLE MODULE	Module disable		RW	0
		0x0:	Module is enabled; clocks are not gated.		
		0x1:	Module is disabled; clocks are gated.		

Table 23–30. GPIO_OE

Address Offset	0x034		
Physical Address	0x4801 8034	**Instance**	GPIO1
	0x4801 A034		GPIO2
	0x4801 C034		GPIO3
	0x4801 E034		GPIO4
Description	This register is used to enable the pin’s output capabilities. Its only function is to carry the pad’s 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 23–31. GPIO_DATAIN

Address Offset	0x038		
Physical Address	0x4801 8038	**Instance**	GPIO1
	0x4801 A038		GPIO2
	0x4801 C038		GPIO3
	0x4801 E038		GPIO4
Description	This register is used to register data 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

Note: When the AUTOIDLE bit GPIO_SYSCONFIG[0] is set, the GPIO_DATAIN read command has three interface clock cycle latency because of the data in sample gating mechanism. When the AUTOIDLE bit is not set, the GPIO_DATAIN read command has 2 interface clock cycle latency.

Table 23–32. GPIO_DATAOUT

Address Offset	0x03C		
Physical Address	0x4801 803C	**Instance**	GPIO1
	0x4801 A03C		GPIO2
	0x4801 C03C		GPIO3
	0x4801 E03C		GPIO4
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 (drive) data	RW	0x00000000

Table 23–33. GPIO_LEVELDETECT0

Address Offset	0x040		
Physical Address	0x4801 8040	**Instance**	GPIO1
	0x4801 A040		GPIO2
	0x4801 C040		GPIO3
	0x4801 E040		GPIO4
Description	This register is used to enable/disable for each input line the low-level (0) detection used for 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 0x0: Disable IRQ assertion on low-level detect. 0x1: Enable IRQ assertion on low-level detect.	RW	0x00000000

Table 23–34. GPIO_LEVELDETECT1

Address Offset	0x044		
Physical Address	0x4801 8044	**Instance**	GPIO1
	0x4801 A044		GPIO2
	0x4801 C044		GPIO3
	0x4801 E044		GPIO4
Description	This register is used to enable/disable for each input line the high-level (1) detection used for 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

Note: Enabling high-level detection and low-level detection at the same time for one pin creates a constant interrupt generator.

Table 23–35. GPIO_RISINGDETECT

Address Offset	0x048		
Physical Address	0x4801 8048	**Instance**	GPIO1
	0x4801 A048		GPIO2
	0x4801 C048		GPIO3
	0x4801 E048		GPIO4
Description	This register is used to enable/disable rising-edge (transition 0 => 1) detection for interrupt request generation 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
RISINGEDGE																															

Bits	**Field Name**	**Description**	**Type**	**Reset**
31:0	RISINGEDGE	Rising edge interrupt/wake-up enable 0x0: Disable IRQ/wake-up on rising-edge detection. 0x1: Enable IRQ/wake-up on rising-edge detection.	RW	0x00000000

Table 23–36. GPIO_FALLINGDETECT

Address Offset	0x04C		
Physical Address	0x4801 804C	**Instance**	GPIO1
	0x4801 A04C		GPIO2
	0x4801 C04C		GPIO3
	0x4801 E04C		GPIO4
Description	This register is used to enable/disable falling-edge (transition 1 => 0) detection for interrupt request generation 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
FALLINGEDGE																															

Bits	**Field Name**	**Description**	**Type**	**Reset**
31:0	FALLINGEDGE	Falling edge interrupt/wake-up enable 0x0: Disable IRQ/wake-up on falling-edge detection. 0x1: Enable IRQ/wake-up on falling-edge detection.	RW	0x00000000

Table 23–37. GPIO_DEBOUNCENABLE

Address Offset	0x050		
Physical Address	0x4801 8050	**Instance**	GPIO1
	0x4801 A050		GPIO2
	0x4801 C050		GPIO3
	0x4801 E050		GPIO4
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 on the corresponding input port. 0x1: Enable debouncing on the corresponding input port.	RW	0x00000000

Table 23−38. GPIO_DEBOUNCINGTIME

Address Offset	0x054		
Physical Address	0x4801 8054	**Instance**	GPIO1
	0x4801 A054		GPIO2
	0x4801 C054		GPIO3
	0x4801 E054		GPIO4
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	Reads return 0.	R	0x000000
7:0	DEBOUNCEVAL	Input debouncing value	RW	0x00

Table 23−39. GPIO_CLEARIRQENABLE1

Address Offset	0x060		
Physical Address	0x4801 8060	**Instance**	GPIO1
	0x4801 A060		GPIO2
	0x4801 C060		GPIO3
	0x4801 E060		GPIO4
Description	Clear interrupt 1 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
CLEARIRQEN1																															

Bits	Field Name	Description	Type	Reset
31:0	CLEARIRQEN1	Clear interrupt enable 1. 0x0: No effect 0x1: Clear the corresponding bit in the relevant interrupt enable register.	RW	0x00000000

Table 23−40. GPIO_SETIRQENABLE1

Address Offset	0x064		
Physical Address	0x4801 8064	**Instance**	GPIO1
	0x4801 A064		GPIO2
	0x4801 C064		GPIO3
	0x4801 E064		GPIO4
Description	Set interrupt 1 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
SETIRQEN1																															

Bits	Field Name	Description	Type	Reset
31:0	SETIRQEN1	Set interrupt enable 1. 0x0: No effect 0x1: Set the corresponding bit in the relevant interrupt enable register.	RW	0x00000000

Table 23−41. GPIO_CLEARIRQENABLE2

Address Offset	0x070		
Physical Address	0x4801 8070	**Instance**	GPIO1
	0x4801 A070		GPIO2
	0x4801 C070		GPIO3
	0x4801 E070		GPIO4
Description	Clear interrupt 2 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
CLEARIRQEN2																															

Bits	Field Name	Description	Type	Reset
31:0	CLEARIRQEN2	Clear interrupt enable 2. 0x0: No effect 0x1: Clear the corresponding bit in the relevant interrupt enable register.	RW	0x00000000

Table 23−42. GPIO_SETIRQENABLE2

Address Offset	0x074		
Physical Address	0x4801 8074	**Instance**	GPIO1
	0x4801 A074		GPIO2
	0x4801 C074		GPIO3
	0x4801 E074		GPIO4
Description	Set interrupt 2 enable.		
Type	RW		

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="32">SETIRQEN2</td></tr>
</table>

Bits	Field Name	Description		Type	Reset
31:0	SETIRQEN2	Set interrupt enable 2.		RW	0x00000000
		0x0:	No effect		
		0x1:	Set the corresponding bit in the relevant interrupt enable register.		

Table 23−43. GPIO_CLEARWKUENA

Address Offset	0x080		
Physical Address	0x4801 8080	**Instance**	GPIO1
	0x4801 A080		GPIO2
	0x4801 C080		GPIO3
	0x4801 E080		GPIO4
Description	Clear wake-up enable.		
Type	RW		

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="32">CLEARWAKEUPEN</td></tr>
</table>

Bits	Field Name	Description		Type	Reset
31:0	CLEARWAKE UPEN	Clear wake-up enable.		RW	0x00000000
		0x0:	No effect		
		0x1:	Clear the corresponding bit in the relevant wake-up enable register.		

Table 23−44. GPIO_SETWKUENA

Address Offset	0x084		
Physical Address	0x4801 8084	**Instance**	GPIO1
	0x4801 A084		GPIO2
	0x4801 C084		GPIO3
	0x4801 E084		GPIO4
Description	Set wake-up enable.		
Type	RW		

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="32">SETWAKEUPEN</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
31:0	SETWAKEUPEN	Set wake-up enable. 0x0: No effect 0x1: Set the corresponding bit in the relevant wake-up enable register.	RW	0x00000000

Table 23−45. GPIO_CLEARDATAOUT

Address Offset	0x090		
Physical Address	0x4801 8090	**Instance**	GPIO1
	0x4801 A090		GPIO2
	0x4801 C090		GPIO3
	0x4801 E090		GPIO4
Description	Clear data output.		
Type	RW		

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="32">CLEARDATAOUT</td></tr>
</table>

Bits	Field Name	Description	Type	Reset
31:0	CLEARDATAOUT	Clear data output register. 0x0: No effect 0x1: Clear the corresponding bit in the relevant data output register.	RW	0x00000000

Table 23–46. GPIO_SETDATAOUT

Address Offset	0x094		
Physical Address	0x4801 8094	**Instance**	GPIO1
	0x4801 A094		GPIO2
	0x4801 C094		GPIO3
	0x4801 E094		GPIO4
Description	Set data output 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. 0x0: No effect 0x1: Set the corresponding bit in the relevant data output register.	RW	0x00000000

23.6.3 General-Purpose Interface Top (OSPL1) Registers

Table 23–47. IPGENERICOCPSPL_REVISION

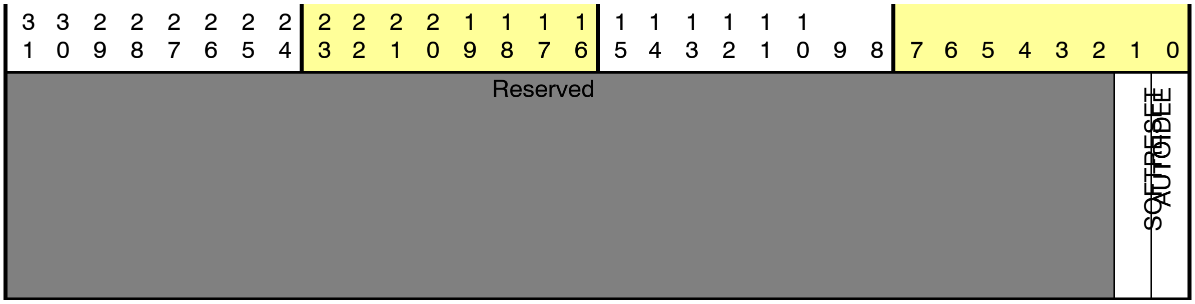
Address Offset	0x00		
Physical Address	0x4801 9000	**Instance**	OSPL1
Description	This register contains the IP revision code.		
Type	R		

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 Examples: 0x10 for 1.0, 0x21 for 2.1	R	

Table 23–48. IPGENERICOCPSPL_SYSCONFIG

Address Offset	0x10		
Physical Address	0x4801 9010	**Instance**	OSPL1
Description	This register controls the OCP interface parameters.		
Type	RW		

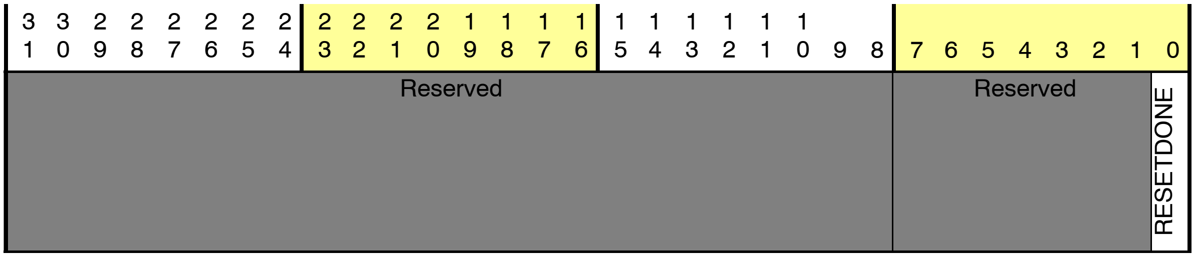


31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																														SOFTRESET	AUTOIDLE

Bits	Field Name	Description	Type	Reset
31:2	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000000
1	SOFTRESET	Software reset. Set this bit to trigger a module reset. The bit is automatically reset by the hardware. Read returns 0. 0x0: Normal mode 0x1: The module is reset.	RW	0
0	AUTOIDLE	Enable power-management capability.	RW	0

Table 23−49. IPGENERICOCPSPL_SYSSTATUS

Address Offset	0x14		
Physical Address	0x4801 9014	**Instance**	OSPL1
Description	This register provides status information about the module.		
Type	R		

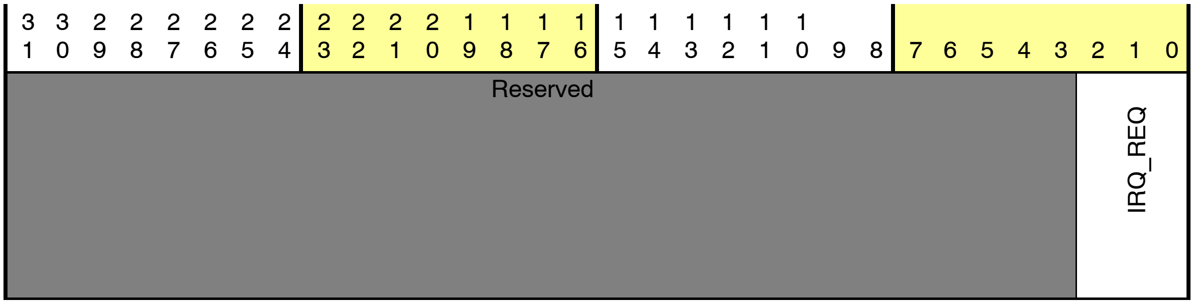


Bits	Field Name	Description	Type	Reset
31:8	Reserved	Reserved for module-specific status information. Read returns 0.	R	0x000000
7:1	Reserved	Reserved for module-specific status information. Read returns 0.	R	0x00
0	RESETDONE	Internal reset monitoring 0x0: Internal module reset is ongoing. 0x1: Reset complete.[1]	R	0

[1] During reset, the value is 0, but the value read just after the reset is 1, because ICLK/FCLK is already operating.

Table 23−50. IPGENERICOCPSPL_IRQSTATUS

Address Offset	0x18		
Physical Address	0x4801 9018	**Instance**	OSPL1
Description	This register provides the interrupt lines status.		
Type	R		



Bits	Field Name	Description	Type	Reset
31:3	Reserved	Reserved for module-specific number of IRQ lines. Read returns 0.	R	0x00000000
2:0	IRQ_REQ	Interrupt requests	R	0x0

Table 23−51. IPGENERICOCPSPL_GPO

Address Offset	0x40		
Physical Address	0x4801 9040	**Instance**	OSPL1
Description	This register provides general-purpose output 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																								Reserved							GPO

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Reserved for module-specific number of GPO lines. Read returns 0.	R	0x000000
0	GPO	Bit for supporting GPO capability	RW	0

Table 23−52. IPGENERICOCPSPL_GPI

Address Offset	0x50		
Physical Address	0x4801 9050	**Instance**	OSPL1
Description	This register provides general-purpose input control.		
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							GPI

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Reserved for module-specific number of GPI lines. Read returns 0.	R	0x000000
0	GPI	Bit for supporting GPI capability	R	0

Chapter 24

Pinout Overview

This chapter provides an overview of the pinout on the OMAP2420 high-performance multimedia device.

Topic **Page**

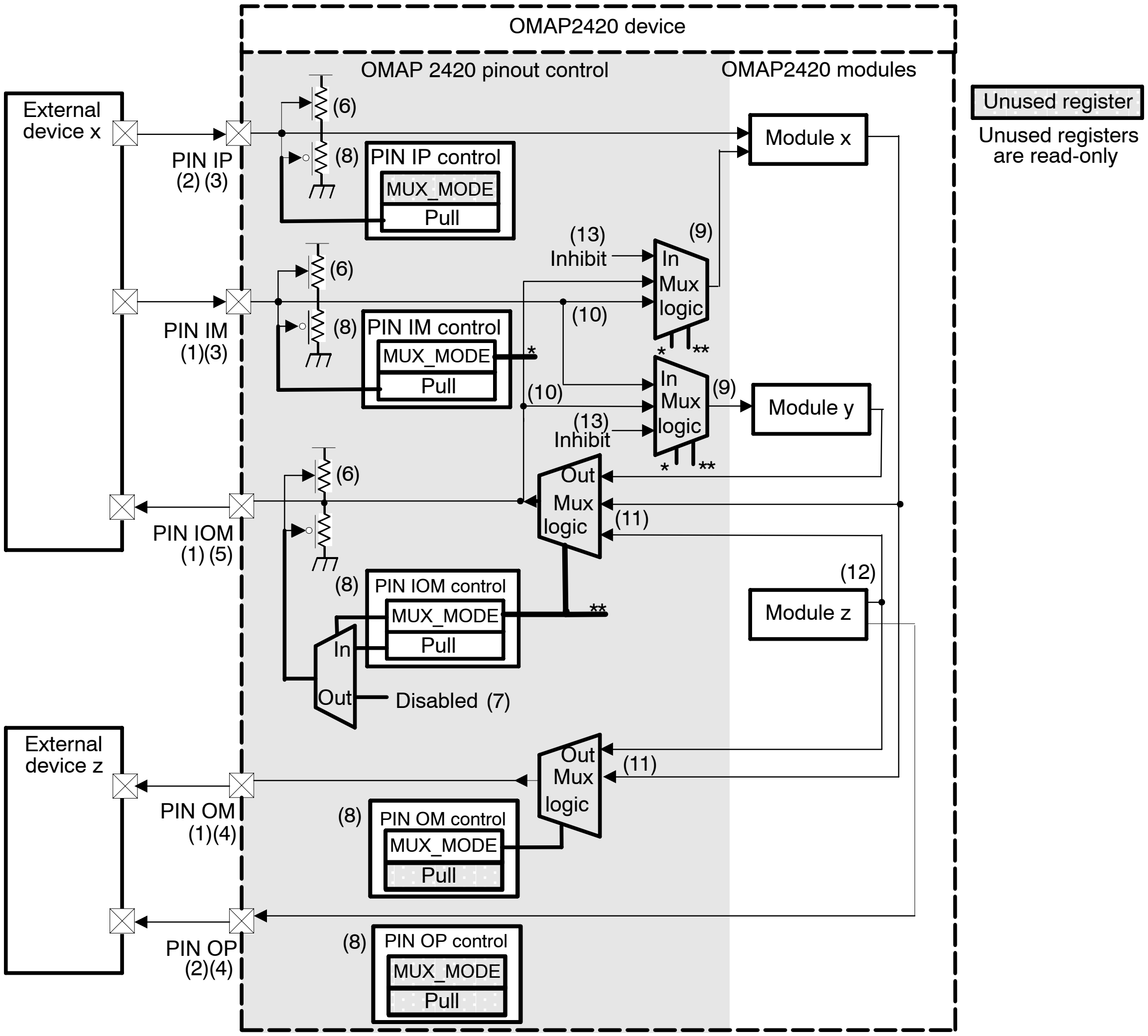
24.1 Pinout Overview

The OMAP2420 device can be interfaced easily with any wireless system through its programmable pinout. Pinout control enables each pin to be configured in a flexible way to match system requirements.

The default configuration limits electrical contention risks and can be changed by simple software access to the pinout control registers. Numerous combinations of internal modules and pins are available to cover a maximum number of applicative solutions.

Figure 24−1 shows the different pin categories with their control logic. It is not an exhaustive list of combintations of internal modules and pins.

Figure 24−1. Pinout Control Diagram



Note: Numbers in parentheses correspond to those referenced in Section 24.2, *Features*.

24.2 Features

The OMAP2420 programmable pinout includes the following main features (numbers in parentheses correspond to those referenced in Figure 24−1):

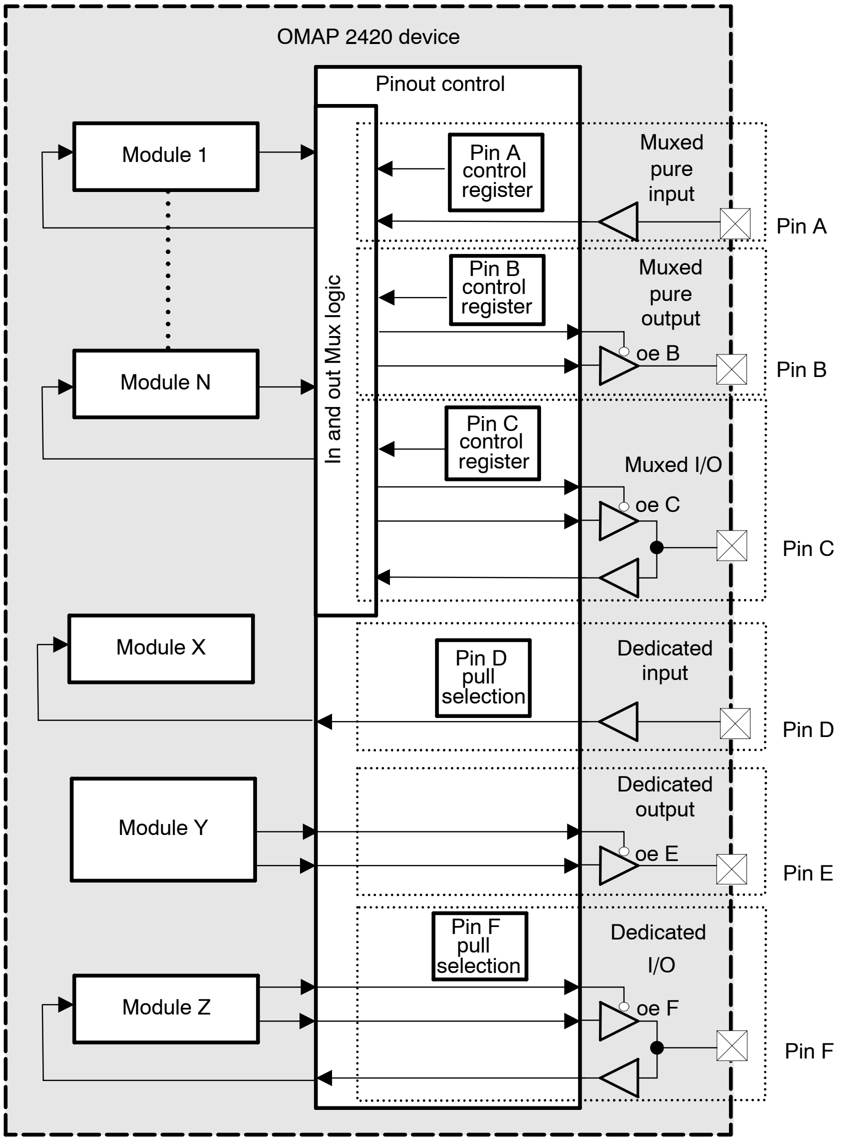
- 255 pins controlled for applications on the OMAP2420
- Support for muxed (1) and dedicated pins (2)
- Support for pure inputs (3), pure outputs (4), and I/O pins (5)
- Software-controllable combinational pullup/pulldown (6)
- Automatic disabling of pull when the pad is configured as an output (7)
- Single control register for each pin (8)
- Support access from different pins to the input of one module (9)
- Single pin mapping to the inputs of different modules (10)
- Single pin configuration to support outputs from different modules (11)
- Single-module output configuration on several pads (12)
- Hardware protection to avoid internal electrical contention from a bad configuration (13)
- Pins that are not fail-safe

24.3 Functional Description

24.3.1 Block Diagram

Figure 24−2 is a block diagram of the OMAP2420 pinout.

Figure 24−2. Pinout Block Diagram



24.3.2 Pin Control Register

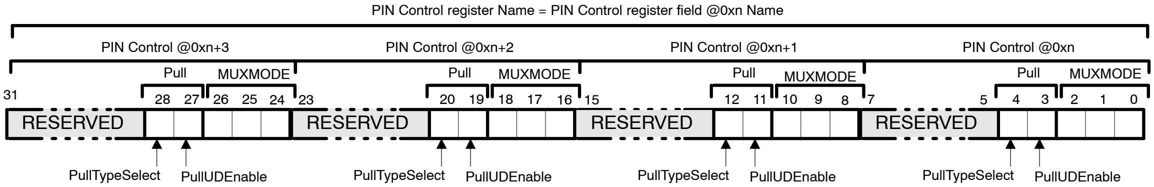
Each pin is configurable by software using its associated pin-control register field. A pin-control register field is 8 bits wide with only the lower 5 bits used.

Only one pin-control register field is available for a pin. Each 32-bit pinout control register is grouped into four 8-bit pin-control register fields. One pinout control register provides control for four different pins.

> **Note:**
>
> Each 32-bit pinout control register uses the name of the lower address pin-control register field.

figure shows the pinout structure.

Figure 24−3. Pinout Structure



The functional five bits of a pin-control register field are divided into two fields:

- MUXMODE: 3 bits for mode selection. 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 pullup/down configuration
  - PullTypeSelect: Selection between the pullup and the pulldown for the pin
  - PullUDEnable: Activation of the pull

24.3.2.1 Mode Selection

Table 24−1 lists the mode selection settings.

Table 24−1. Mode Selection Settings

MUXMODE	Mode Selected
000	Primary Mode = Mode 0
001	Mode 1
010	Mode 2
011	Mode 3
100	Mode 4
101	Mode 5
111	Protected mode

Mode 0 is the primary mode. When mode 0 is set, the function mapped to the pin corresponds to the name of the pin. A function is always mapped to the primary mode, which is not necessarily the default mode. The default mode is automatically configured at the release of the internal GLOBAL_PWRON reset.

Mode 1 to mode 5 are possible modes for alternate functions. On each pin, some modes are effectively used for alternate functions, but some modes are also unused and correspond to no functional configuration.

Mode 6: No functional interfaces are mapped to this mode.

The protected mode avoids any risk of electrical contention by configuring the pin as an input with no functional interface mapped to it. This mode is used mainly as the default mode for all pins containing no mandatory interface at the release of GLOBAL_PWRON reset.

24.3.2.2 Pull Selection

Regardless of which pull value is configured, when a pin is configured as an output, pulls are automatically disabled. Table 24−2 lists the pull selects.

Table 24−2. Pull Selects

Pull		
PullType Select	**PullUD Enable**	**PAD Behavior**
0	0	Pulldown selected but not activated
0	1	Pulldown selected and activated if pin configured in input
1	0	Pullup selected but not activated
1	1	Pullup selected and activated if pin configured in input

24.3.3 Dedicated Pins

Dedicated pins are used by one interface exclusively. A dedicated pin is hardwired to the module interface and no alternate modes are available for the pin. Pinout control registers with a MUXMODE bit field exist for these pins, but are not functional; access to these bits has no effect on functional behavior.

24.3.3.1 Dedicated Pure Inputs

If a pull exists on a dedicated pure input pin, pullup/pulldown disable can be configured using pull bits of the pin-control register field.

24.3.3.2 Dedicated Pure Outputs

Nothing about a dedicated pure output pin can be configured. The output and output enable (OE) signal are hardwired to the module. No pullup/down is available on the pin.

24.3.3.3 Dedicated I/O

A dedicated I/O pin combines a pure input and pure output. When the output enable signal is driven by a module (meaning the pin is an output), the pull is automatically disabled.

When the output enable signal is driven to 1 by a module (meaning the pin is configured as an input), the pull can be configured by the pull bits of the pin-control register field.

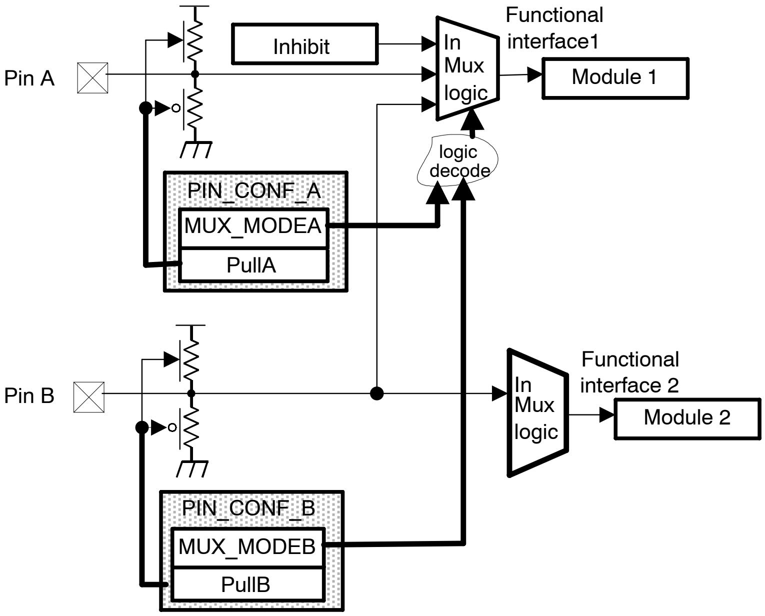
24.3.4 Muxed Pins

Muxed pins correspond to pins on which several different functions can be mapped, depending on the MUXMODE value of the pin-control register field.

24.3.4.1 Muxed Pure Inputs

Only input interfaces can be mapped to a muxed pure input pin. If a pull exists, it can be configured by writing to the pull bits of the pin-control register field. Figure 24−4 is a functional diagram of muxed pure inputs.

Figure 24−4. Muxed Pure Inputs Functional Diagram



Hardware protection is implemented to avoid damage from bad programming.

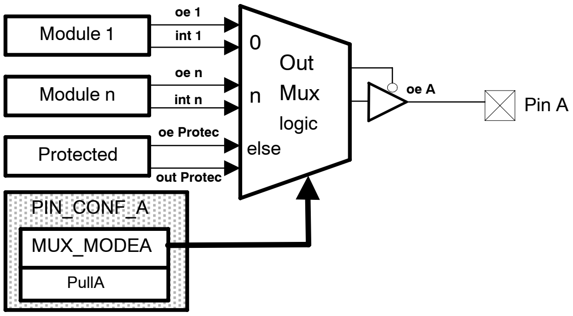
If no functional interface is selected for a module, the value driven is the inhibit value, which is a hardwired value chosen to keep the module inactive.

If several input pins are asserted to the same functional interface, conflict is automatically resolved by giving priority to the pin with the lower MUXMODE value.

24.3.4.2 Muxed Pure Outputs

No pull is available for a muxed pure output pin. Multiple alternate functions can be mapped on the pin chosen by configuring the MUXMODE bits of the pin configuration register field. Figure 24−5 is a functional diagram of muxed pure outputs.

Figure 24−5. Muxed Pure Outputs Functional Diagram



The protected value is automatically asserted if the mode selected does not correspond to an alternate function.

24.3.4.3 Muxed I/O

Muxed I/O is the combination of a muxed input and a muxed output. The MUXMODE value of the pin-control register field defines the type (input or output) depending on the interface mapped. Muxed I/O pin behavior is thus strictly equivalent to a pure input or output.

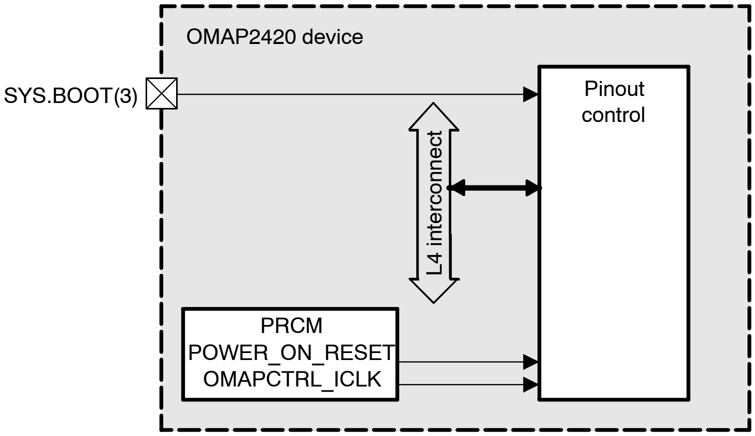
24.4 Pinout Control Integration

24.4.1 Pinout Description

The pinout control module contains a bank of registers used to control I/O pins. Considering no other functionality than register access for this module, only the interface clock is required.

Because pinout control is included in the WKUP power domain, the reset is the internal power-on reset. Consequently, the pinout is not altered by a warm-up reset. Figure 24−6 shows pinout integration in the OMAP2420.

Figure 24−6. Pinout Integration



SYS.BOOT(3) is sampled at the release of the reset and changes the reset value of the MUXMODE field for certain pins of the GPMC interface (the default mode, depending on the SYS.BOOT(3) value for these pins). (See Section 24.4.1.2, *Pin Characteristics*, for the list and behavior of these particular pins.)

24.4.1.1 Clocking, Reset, and Power-Management Scheme

❑ Clocking

Only the interface clock is required for the pinout control module (see Chapter 5, *Power, Reset, and Clock Management*).

Table 24−3 describes the pinout control module clock scheme.

Table 24−3. Pinout Control Module Clock Scheme

Clock	Frequency	Name	Comments
Interface	Core_L4_clk:	OMAPCTRL_ICLK	Source and control is power, reset, and clock management (PRCM) module.

- ☐ Hardware reset

Pinout control is sensitive only to the internal power-on reset, GLOBAL_PWRON.

The internal power-on reset is not a direct image of the power-on reset input pin. The internal power-on reset signal (GLOBAL_PWRON) is generated by the PRCM module. The internal power-on reset is activated by the PRCM module when the eFuse-related settings (such as DEVICE_TYPE) are initialized (for details, see Chapter 5, *Power, Reset, and Clock Management*).

- ☐ Software reset

No software reset is available for the pinout control module, even if software bit exists:

SOFTRESET bit: INTC_SYSCONFIG[1] controls the software reset; writing 1 to this bit has no effect.

- ☐ Power domain

The pinout control module is part of the WKUP power domain, which means it is always powered, regardless of the state of the OMAP2420.

24.4.1.2 Pin Characteristics

Table 24−4 describes the characteristics of each register pad configuration. Pads are classified alphabetically in ascending order.

- ☐ Ball: Ball index of the pin on the package
- ☐ I/O type: I for pure input, O for pure output, and IO for I/O pins
- ☐ Mux type: Muxed when the pin can be multiplexed on several different functions; dedicated when the pin is hardwired to one function
- ☐ Power: Define the power voltage source for the pin
- ☐ Name extension: Suffix of the pinout control register, which contains the pinout-control register field for the pin configuration. To obtain the full register name, concatenate CONTROL_PADCONF_ and suffix: CONTROL_PADCONF_<Name extension>.
- ☐ Offset: Offset address for the 32-bit pinout control register. To obtain the physical address, add 0x4800 0000 to this offset value.
- ☐ Byte: The position of the byte pinout-control register field inside the 32-bit pinout control register:
  - 0: Bits 0 to 7
  - 1: Bits 8 to 15
  - 2: Bits 16 to 23
  - 3: Bits 24 to 31

  The addition of the byte and offset give the address of the pinout control register field byte. Gray shading indicates that the byte is read-only (for dedicated pins).

- Pin name: The name of both the pin and the pinout control register field. The name refers to the primary function of the pin. Gray shading indicates that the pinout-control register field is unused (for dedicated pins). See Table 24−4.
- Control reset state: The state of the pin at reset:
  - Mode 0: The default mode is the primary mode; the function selected in mode 0 is available at the release of the power-on reset.
  - Protect: On release of the power-on reset, the pin is in protected mode to avoid potential electrical contention (high impedance). By default, no functionality is connected to this pin. The pinout-control register field must be programmed by software to enable functionality on the pin.
  - SBoot3: The pin state depends on the SYS_BOOT(3) input pin value sampled on release of the power-on reset. A high SYS_BOOT(3) value indicates an internal boot and protected mode reset state. A low value indicates an external boot and mode 0 reset state.

  Gray shading indicates a dedicated pin; this state is not significant (see Table 24−7).
- Control reset value: The default value for the MUXMODE field in the pinout-control register field (the MUXMODE value defines the mode on the pin).

  Gray shading indicates a dedicated pin; this value is not significant (the value is hardwired and cannot be modified by software).
  - 000: Mode 0 is the default mode.
  - 111: Protected mode is the default mode.
  - *** : are reset values determined by the SYS_BOOT(3) input.

    A high SYS_BOOT(3) indicates an internal boot; the bits reset to 111. A low SYS_BOOT(3) value indicates an external boot; the bits reset to 000.
- Pull type: The type of pull cell available internally on the pin:
  - PUPD: Combination of a pullup and a pulldown (software programmable)
  - PU: Pullup (enable or disable by software programming)
  - PD: Pulldown (enable or disable by software programming)
  - No: No internal pull available on the pin
- Pull reset state: The state of the pull on the pin:
  - U: Pullup active by default (at release of the power-on reset)
  - D: Pulldown active by default (at release of the power-on reset)
  - −: No pull active by default
  - *: Must be pulled either high or low externally according to SYS_BOOT configuration desired

  Gray shading indicates that no internal pull is available for this pin; this state is not significant.

- ❑ Pull reset value: The default value for the pull field of the pinout-control register field

Gray shading indicates that no internal pull is available on the pin; this value is hardwired and cannot be modified by software.

Note:

Gray shading in Table 24−4 indicates a static configuration that cannot be modified by software. Software access to gray-shaded values returns no error, but has no effect on the pin.

Table 24–4. OMAP2420 Pin Characteristics

Pin				Pin Control Register				Control		Pull		
Ball	**Type**	**MUX Type**	**Power**	**Name Extension**	**Offset**	**Byte**	**Pin Name**	**Reset State**	**Reset Value**	**Type**	**Reset State**	**Reset Value**
A1	O	Dedicated	VDDS1	jtag_emu0	0x013C	3	jtag_rtck	Mode0	000	No	–	00
A11	IO	Dedicated	VDDS1	sdrc_d12	0x0064	0	sdrc_d12	Mode0	000	PUPD	D	01
A20	I	Dedicated	VDDS1	jtag_tms	0x0140	1	jtag_tdi	Mode0	000	PUPD	D	01
A21	O	Dedicated	VDDS1	jtag_tms	0x0140	2	jtag_tdo	Mode0	000	No	–	00
AA10	IO	Muxed	VDDS		0x00E4	1		Protect	111	PUPD	U	11
AA12	IO	Muxed	VDDS		0x00E8	1		Protect	111	PUPD	D	01
AA14	I	Dedicated	VDDA	tv_rref	0x011C	0	tv_rref	Mode0	000	No	–	00
AA16	O	Dedicated	VDDA	uart3_tx_irtx	0x0118	2	tv_cvbs	Mode0	000	No	–	00
AA17	IO	Muxed	VDDS	sys_xtalout	0x0134	2	sys_clkreq	Protect	111	PUPD	U	11
AA18	I	Dedicated	VDDA	uart3_tx_irtx	0x0118	3	tv_vref	Mode0	000	No	–	00
AA20	I	Dedicated	VDDS2	jtag_emu0	0x013C	1	jtag_ntrst	Mode0	000	PUPD	D	01
AA21	IO	Muxed	VDDS2	gpio_6	0x0138	3	jtag_emu1	Protect	111	PUPD	U	11
AA4	IO	Muxed	VDDS		0x00E4	3		Protect	111	PUPD	D	01
AA6	IO	Muxed	VDDS		0x00E4	2		Protect	111	PUPD	D	01
AA8	IO	Muxed	VDDS		0x00E8	2		Protect	111	PUPD	U	11
B1	I	Dedicated	VDDS1	jtag_emu0	0x013C	2	jtag_tck	Mode0	000	PUPD	D	01
B10	O	Dedicated	VDDS1	sdrc_ba0	0x0034	3	sdrc_a9	Mode0	000	No	–	00
B12	O	Dedicated	VDDS1	sdrc_nras	0x00A4	0	sdrc_nras	Mode0	000	No	–	00
B13	IO	Muxed	VDDS1	sdrc_ncs0	0x00A0	3	sdrc_cke1	Protect	111	PUPD	U	11
B14	O	Dedicated	VDDS1	gpmc_wait2	0x009C	3	sdrc_nclk	Mode0	000	No	–	00
B15	IO	Dedicated	VDDS1	sdrc_d16	0x0060	0	sdrc_d16	Mode0	000	PUPD	D	01
B16	IO	Dedicated	VDDS1	sdrc_d24	0x0058	3	sdrc_d21	Mode0	000	PUPD	D	01
B17	IO	Dedicated	VDDS1	sdrc_d24	0x0058	2	sdrc_d22	Mode0	000	PUPD	D	01
B18	IO	Dedicated	VDDS1	sdrc_dqs1	0x00B0	2	sdrc_dqs3	Mode0	000	PUPD	D	01

Table 24−4. OMAP2420 Pin Characteristics (Continued)

Pin				Pin Control Register				Control		Pull		
Ball	**Type**	**MUX Type**	**Power**	**Name Extension**	**Offset**	**Byte**	**Pin Name**	**Reset State**	**Reset Value**	**Type**	**Reset State**	**Reset Value**
B19	IO	Dedicated	VDDS1	sdrc_d28	0x0054	2	sdrc_d26	Mode0	000	PUPD	D	01
B2	IO	Dedicated	VDDS1	sdrc_d4	0x006C	1	sdrc_d3	Mode0	000	PUPD	D	01
B20	IO	Dedicated	VDDS1	sdrc_d28	0x0054	1	sdrc_d27	Mode0	000	PUPD	D	01
B21	I	Dedicated	VDDS1	jtag_tms	0x0140	0	jtag_tms	Mode0	000	PUPD	D	01
B3	IO	Muxed	VDDS1	sdrc_a14	0x0030	0	sdrc_a14	Protect	111	PUPD	D	01
B4	IO	Muxed	VDDS1	sdrc_a14	0x0030	1	sdrc_a13	Protect	111	PUPD	D	01
B5	O	Dedicated	VDDS1	sdrc_nras	0x00A4	3	sdrc_dm0	Mode0	000	No	–	00
B6	O	Dedicated	VDDS1	sdrc_a4	0x003C	0	sdrc_a4	Mode0	000	No	–	00
B7	O	Dedicated	VDDS1	sdrc_a4	0x003C	2	sdrc_a2	Mode0	000	No	–	00
B9	O	Dedicated	VDDS1	sdrc_a0	0x0040	0	sdrc_a0	Mode0	000	No	–	00
C10	O	Dedicated	VDDS1	sdrc_ba0	0x0034	1	sdrc_a11	Mode0	000	No	–	00
C11	O	Dedicated	VDDS1	sdrc_ba0	0x0034	0	sdrc_ba0	Mode0	000	No	–	00
C12	IO	Muxed	VDDS1	sdrc_ncs0	0x00A0	1	sdrc_ncs1	Protect	111	PUPD	U	11
C13	O	Dedicated	VDDS1	sdrc_nras	0x00A4	2	sdrc_nwe	Mode0	000	No	–	00
C14	IO	Dedicated	VDDS1	gpmc_wait2	0x009C	2	sdrc_clk	Mode0	000	PUPD	–	00
C15	IO	Dedicated	VDDS1	sdrc_d20	0x005C	1	sdrc_d19	Mode0	000	PUPD	D	01
C16	IO	Dedicated	VDDS1	sdrc_d24	0x0058	1	sdrc_d23	Mode0	000	PUPD	D	01
C17	O	Dedicated	VDDS1	sdrc_dm1	0x00A8	1	sdrc_dm2	Mode0	000	No	–	00
C18	IO	Dedicated	VDDS1	sdrc_stk_d16	0x0050	2	sdrc_d30	Mode0	000	PUPD	D	01
C19	IO	Dedicated	VDDS1	sdrc_stk_d16	0x0050	3	sdrc_d29	Mode0	000	PUPD	D	01
C2	IO	Dedicated	VDDS1	sdrc_d0	0x0070	0	sdrc_d0	Mode0	000	PUPD	D	01
C20	IO	Dedicated	VDDS1	sdrc_stk_d16	0x0050	1	sdrc_d31	Mode0	000	PUPD	D	01
C3	IO	Dedicated	VDDS1	sdrc_d8	0x0068	3	sdrc_d5	Mode0	000	PUPD	D	01
C4	IO	Dedicated	VDDS1	sdrc_d4	0x006C	0	sdrc_d4	Mode0	000	PUPD	D	01

Table 24−4. OMAP2420 Pin Characteristics (Continued)

Pin				Pin Control Register				Control		Pull		
Ball	**Type**	**MUX Type**	**Power**	**Name Extension**	**Offset**	**Byte**	**Pin Name**	**Reset State**	**Reset Value**	**Type**	**Reset State**	**Reset Value**
C5	IO	Dedicated	VDDS1	sdrc_d8	0x0068	2	sdrc_d6	Mode0	000	PUPD	D	01
C6	IO	Dedicated	VDDS1	sdrc_d16	0x0060	1	sdrc_d15	Mode0	000	PUPD	D	01
C7	O	Dedicated	VDDS1	sdrc_a8	0x0038	3	sdrc_a5	Mode0	000	No	–	00
C8	O	Dedicated	VDDS1	sdrc_a8	0x0038	1	sdrc_a7	Mode0	000	No	–	00
C9	O	Dedicated	VDDS1	sdrc_ba0	0x0034	2	sdrc_a10	Mode0	000	No	–	00
D10	O	Dedicated	VDDS1	sdrc_a14	0x0030	3	sdrc_ba1	Mode0	000	No	–	00
D11	IO	Muxed	VDDS1	sdrc_a14	0x0030	2	sdrc_a12	Protect	111	PUPD	D	01
D12	O	Dedicated	VDDS1	sdrc_ncs0	0x00A0	0	sdrc_ncs0	Mode0	000	No	–	00
D13	O	Dedicated	VDDS1	sdrc_ncs0	0x00A0	2	sdrc_cke0	Mode0	000	No	–	00
D14	O	Dedicated	VDDS1	sdrc_nras	0x00A4	1	sdrc_ncas	Mode0	000	No	–	00
D15	IO	Dedicated	VDDS1	sdrc_d20	0x005C	2	sdrc_d18	Mode0	000	PUPD	D	01
D16	IO	Dedicated	VDDS1	sdrc_dqs1	0x00B0	1	sdrc_dqs2	Mode0	000	PUPD	D	01
D17	O	Dedicated	VDDS1	sdrc_dm1	0x00A8	2	sdrc_dm3	Mode0	000	No	–	00
D18	IO	Dedicated	VDDS1	sdrc_d28	0x0054	3	sdrc_d25	Mode0	000	PUPD	D	01
D19	IO	Muxed	VDDS2	mmc_dat3	0x00F8	0	mmc_dat3	Protect	111	PUPD	D	01
D21	IO	Muxed	VDDS2	dss_d17	0x00C4	1	uart1_cts	Protect	111	PUPD	U	11
D3	IO	Muxed	VDDS1	sdrc_d0	0x0070	2	gpmc_a9	SBoot3	***	PUPD	U	11
D4	IO	Dedicated	VDDS1	sdrc_d4	0x006C	2	sdrc_d2	Mode0	000	PUPD	D	01
D5	IO	Dedicated	VDDS1	sdrc_stk_dm1	0x00AC	3	sdrc_dqs0	Mode0	000	PUPD	D	01
D6	IO	Dedicated	VDDS1	sdrc_d16	0x0060	3	sdrc_d13	Mode0	000	PUPD	D	01
D7	O	Dedicated	VDDS1	sdrc_a4	0x003C	1	sdrc_a3	Mode0	000	No	–	00
D8	O	Dedicated	VDDS1	sdrc_a4	0x003C	3	sdrc_a1	Mode0	000	No	–	00
D9	O	Dedicated	VDDS1	sdrc_a8	0x0038	0	sdrc_a8	Mode0	000	No	–	00
E18	IO	Muxed	VDDS2	mmc_dat_dir3	0x00FC	0	mmc_dat_dir3	Protect	111	PUPD	D	01

Table 24–4. OMAP2420 Pin Characteristics (Continued)

Pin				Pin Control Register				Control		Pull		
Ball	**Type**	**MUX Type**	**Power**	**Name Extension**	**Offset**	**Byte**	**Pin Name**	**Reset State**	**Reset Value**	**Type**	**Reset State**	**Reset Value**
E19	IO	Muxed	VDDS2	mmc_cmd	0x00F4	3	mmc_dat2	Protect	111	PUPD	D	01
E2	IO	Muxed	VDDS1	gpmc_ncs0	0x008C	2	gpmc_ncs2	Protect	111	PUPD	U	11
E20	IO	Muxed	VDDS2	mmc_dat3	0x00F8	2	mmc_dat_dir1	Protect	111	PUPD	D	01
E3	IO	Muxed	VDDS1	sdrc_d0	0x0070	1	gpmc_a10	SBoot3	***	PUPD	U	11
E4	IO	Muxed	VDDS1	gpmc_a7	0x0074	0	gpmc_a7	SBoot3	***	PUPD	U	11
E5	IO	Muxed	VDDS1	gpio_6	0x0138	0	gpio_6	Protect	111	PUPD	D	01
F1	IO	Muxed	VDDS1	gpmc_ncs4	0x0090	0	gpmc_ncs4	Protect	111	PUPD	U	11
F18	IO	Muxed	VDDS2	mmc_dat3	0x00F8	3	mmc_dat_dir2	Protect	111	PUPD	D	01
F19	IO	Muxed	VDDS2	mmc_dat3	0x00F8	1	mmc_dat_dir0	Protect	111	PUPD	D	01
F20	IO	Muxed	VDDS2	mmc_cmd	0x00F4	1	mmc_dat0	Protect	111	PUPD	D	01
F3	IO	Muxed	VDDS1	gpmc_a7	0x0074	1	gpmc_a6	SBoot3	***	PUPD	U	11
F4	IO	Muxed	VDDS1	gpmc_a7	0x0074	2	gpmc_a5	SBoot3	***	PUPD	U	11
G10	IO	Dedicated	VDDS1	sdrc_d12	0x0064	2	sdrc_d10	Mode0	000	PUPD	D	01
G11	IO	Dedicated	VDDS1	sdrc_d8	0x0068	0	sdrc_d8	Mode0	000	PUPD	D	01
G12	O	Dedicated	VDDS1	sdrc_dm1	0x00A8	0	sdrc_dm1	Mode0	000	No	–	00
G13	IO	Dedicated	VDDS1	sdrc_d20	0x005C	0	sdrc_d20	Mode0	000	PUPD	D	01
G14	IO	Dedicated	VDDS1	sdrc_d24	0x0058	0	sdrc_d24	Mode0	000	PUPD	D	01
G18	IO	Muxed	VDDS2	mmc_dat_dir3	0x00FC	1	mmc_cmd_dir	Protect	111	PUPD	D	01
G19	IO	Muxed	VDDS2	eac_bt_fs	0x00F0	3	mmc_clko	Protect	111	PUPD	D	01
G2	O	Muxed	VDDS1	gpmc_nbe1	0x0098	1	gpmc_nwp	Mode0	000	No	–	00
G3	IO	Muxed	VDDS1	gpmc_a7	0x0074	3	gpmc_a4	SBoot3	***	PUPD	U	11
G4	IO	Muxed	VDDS1	sdrc_d0	0x0070	3	gpmc_a8	SBoot3	***	PUPD	U	11
G8	O	Dedicated	VDDS1	sdrc_a8	0x0038	2	sdrc_a6	Mode0	000	No	–	00
G9	IO	Dedicated	VDDS1	sdrc_dqs1	0x00B0	0	sdrc_dqs1	Mode0	000	PUPD	D	01

Table 24−4. OMAP2420 Pin Characteristics (Continued)

Pin				Pin Control Register				Control		Pull		
Ball	**Type**	**MUX Type**	**Power**	**Name Extension**	**Offset**	**Byte**	**Pin Name**	**Reset State**	**Reset Value**	**Type**	**Reset State**	**Reset Value**
H1	IO	Muxed	VDDS1	gpmc_ncs4	0x0090	1	gpmc_ncs5	Protect	111	PUPD	U	11
H10	IO	Dedicated	VDDS1	sdrc_d12	0x0064	1	sdrc_d11	Mode0	000	PUPD	D	01
H11	IO	Dedicated	VDDS1	sdrc_d12	0x0064	3	sdrc_d9	Mode0	000	PUPD	D	01
H12	IO	Dedicated	VDDS1	sdrc_d20	0x005C	3	sdrc_d17	Mode0	000	PUPD	D	01
H13	IO	Dedicated	VDDS1	sdrc_d28	0x0054	0	sdrc_d28	Mode0	000	PUPD	D	01
H14	IO	Muxed	VDDS2	mmc_cmd	0x00F4	2	mmc_dat1	Protect	111	PUPD	D	01
H15	IO	Muxed	VDDS2	mmc_dat_dir3	0x00FC	2	mmc_clki	Protect	111	PUPD	D	01
H18	IO	Muxed	VDDS2	mmc_cmd	0x00F4	0	mmc_cmd	Protect	111	PUPD	D	01
H19	IO	Muxed	VDDS2	i2c2_sda	0x0114	0	i2c2_sda	Protect	111	PU	U	11
H2	O	Dedicated	VDDS1	gpmc_nadv_ale	0x0094	1	gpmc_noe	Mode0	000	No	–	00
H21	IO	Muxed	VDDS2	dss_d17	0x00C4	2	uart1_rts	Protect	111	PUPD	U	11
H3	IO	Muxed	VDDS1	gpmc_a3	0x0078	0	gpmc_a3	SBoot3	***	PUPD	U	11
H4	IO	Muxed	VDDS1	gpmc_a3	0x0078	1	gpmc_a2	SBoot3	***	PUPD	U	11
H7	IO	Dedicated	VDDS1	sdrc_d4	0x006C	3	sdrc_d1	Mode0	000	PUPD	D	01
H8	IO	Dedicated	VDDS1	sdrc_d8	0x0068	1	sdrc_d7	Mode0	000	PUPD	D	01
H9	IO	Dedicated	VDDS1	sdrc_d16	0x0060	2	sdrc_d14	Mode0	000	PUPD	D	01
J14	IO	Muxed	VDDS2	usb0_rcv	0x0120	2	usb0_se0	Mode0	000	PUPD	–	00
J15	IO	Muxed	VDDS2	mcbsp1_clkx	0x0110	3	i2c2_scl	Protect	111	PU	U	11
J18	IO	Muxed	VDDS2	usb0_rcv	0x0120	0	usb0_rcv	Mode0	000	PUPD	–	00
J19	IO	Muxed	VDDS2	tv_rref	0x011C	2	usb0_vp	Mode0	000	PUPD	–	00
J20	IO	Muxed	VDDS2	tv_rref	0x011C	1	usb0_puen	Mode0	000	PUPD	–	00
J3	IO	Muxed	VDDS1	gpmc_a3	0x0078	2	gpmc_a1	SBoot3	***	PUPD	U	11
J4	IO	Muxed	VDDS1	gpmc_d2	0x0088	3	gpmc_clk	Protect	111	PUPD	D	01
J7	IO	Muxed	VDDS1	gpmc_a3	0x0078	3	gpmc_d15	SBoot3	***	PUPD	U	11

Table 24–4. OMAP2420 Pin Characteristics (Continued)

Pin				Pin Control Register				Control		Pull		
Ball	**Type**	**MUX Type**	**Power**	**Name Extension**	**Offset**	**Byte**	**Pin Name**	**Reset State**	**Reset Value**	**Type**	**Reset State**	**Reset Value**
J8	IO	Muxed	VDDS1	gpmc_d14	0x007C	0	gpmc_d14	SBoot3	***	PUPD	U	11
K1	IO	Muxed	VDDS1	gpmc_ncs4	0x0090	2	gpmc_ncs6	Protect	111	PUPD	U	11
K14	IO	Muxed	VDDS2	uart3_tx_irtx	0x0118	1	uart3_rx_irrx	Protect	111	PUPD	U	11
K15	IO	Muxed	VDDS2	uart3_tx_irtx	0x0118	0	uart3_tx_irtx	Protect	111	PUPD	U	11
K18	IO	Muxed	VDDS2	usb0_rcv	0x0120	3	usb0_dat	Mode0	000	PUPD	–	00
K19	IO	Muxed	VDDS2	usb0_rcv	0x0120	1	usb0_txen	Mode0	000	PUPD	–	00
K20	IO	Muxed	VDDS2	tv_rref	0x011C	3	usb0_vm	Mode0	000	PUPD	–	00
K3	O	Dedicated	VDDS1	gpmc_nadv_ale	0x0094	0	gpmc_nadv_ale	Mode0	000	No	–	00
K4	O	Dedicated	VDDS1	gpmc_nadv_ale	0x0094	2	gpmc_nwe	Mode0	000	No	–	00
K7	IO	Muxed	VDDS1	gpmc_d14	0x007C	1	gpmc_d13	SBoot3	***	PUPD	U	11
K8	IO	Muxed	VDDS1	gpmc_d14	0x007C	3	gpmc_d11	SBoot3	***	PUPD	U	11
L14	IO	Muxed	VDDS2	mcbsp1_dx	0x010C	3	mcbsp1_fsx	Protect	111	PUPD	D	01
L15	IO	Dedicated	VDDS2	mcbsp1_clkx	0x0110	2	i2c1_sda	Mode0	000	PU	U	11
L18	IO	Muxed	VDDS2	i2c2_sda	0x0114	2	uart3_cts_rctx	Protect	111	PUPD	U	11
L19	IO	Muxed	VDDS2	i2c2_sda	0x0114	3	uart3_rts_sd	Protect	111	PUPD	U	11
L2	IO	Muxed	VDDS1	gpmc_ncs4	0x0090	3	gpmc_ncs7	Protect	111	PUPD	U	11
L20	IO	Muxed	VDDS2	dss_d17	0x00C4	3	uart1_tx	Protect	111	PUPD	U	11
L3	I	Dedicated	VDDS1	gpmc_nbe1	0x0098	2	gpmc_wait0	Mode0	000	PUPD	U	11
L4	O	Dedicated	VDDS1	gpmc_ncs0	0x008C	0	gpmc_ncs0	Mode0	000	No	–	00
L7	IO	Muxed	VDDS1	gpmc_d10	0x0080	0	gpmc_d10	SBoot3	***	PUPD	U	11
L8	IO	Muxed	VDDS1	gpmc_d14	0x007C	2	gpmc_d12	SBoot3	***	PUPD	U	11
M1	IO	Muxed	VDDS1	gpmc_wait2	0x009C	0	gpmc_wait2	Protect	111	PUPD	U	11
M14	IO	Muxed	VDDS2	spi2_somi	0x0108	1	spi2_ncs0	Protect	111	PUPD	U	11
M15	IO	Muxed	VDDS2	spi2_somi	0x0108	2	mcbsp1_clkr	Protect	111	PUPD	D	01

Table 24−4. OMAP2420 Pin Characteristics (Continued)

Pin				Pin Control Register				Control		Pull		
Ball	**Type**	**MUX Type**	**Power**	**Name Extension**	**Offset**	**Byte**	**Pin Name**	**Reset State**	**Reset Value**	**Type**	**Reset State**	**Reset Value**
M18	IO	Muxed	VDDS2	mcbsp1_dx	0x010C	2	mcbsp_clks	Protect	111	PUPD	D	01
M19	IO	Dedicated	VDDS2	mcbsp1_clkx	0x0110	1	i2c1_scl	Mode0	000	PU	U	11
M21	IO	Muxed	VDDS2	uart1_rx	0x00C8	1	mcbsp2_dr	Protect	111	PUPD	D	01
M3	IO	Dedicated	VDDS1	gpmc_d2	0x0088	2	gpmc_d0	Mode0	000	PUPD	U	11
M4	IO	Dedicated	VDDS1	gpmc_d2	0x0088	1	gpmc_d1	Mode0	000	PUPD	U	11
M7	IO	Muxed	VDDS1	gpmc_d10	0x0080	1	gpmc_d9	SBoot3	***	PUPD	U	11
M8	IO	Muxed	VDDS1	gpmc_d10	0x0080	2	gpmc_d8	SBoot3	***	PUPD	U	11
N14	IO	Muxed	VDDS2	uart2_rts	0x00EC	1	uart2_tx	Protect	111	PUPD	U	11
N15	IO	Muxed	VDDS2	spi1_simo	0x0100	3	spi1_ncs1	Protect	111	PUPD	U	11
N18	IO	Muxed	VDDS2	i2c2_sda	0x0114	1	hdq_sio	Protect	111	PUPD	U	11
N19	IO	Muxed	VDDS2	mcbsp1_clkx	0x0110	0	mcbsp1_clkx	Protect	111	PUPD	D	01
N2	IO	Muxed	VDDS1	gpmc_ncs0	0x008C	3	gpmc_ncs3	Protect	111	PUPD	U	11
N3	IO	Dedicated	VDDS1	gpmc_d2	0x0088	0	gpmc_d2	Mode0	000	PUPD	U	11
N4	IO	Dedicated	VDDS1	gpmc_d6	0x0084	3	gpmc_d3	Mode0	000	PUPD	U	11
N7	IO	Muxed	VDDS1	gpmc_nbe1	0x0098	3	gpmc_wait1	Protect	111	PUPD	U	11
N8	IO	Muxed	VDDS1	gpmc_ncs0	0x008C	1	gpmc_ncs1	Protect	111	PUPD	U	11
P1	IO	Muxed	VDDS1	gpmc_wait2	0x009C	1	gpmc_wait3	Protect	111	PUPD	U	11
P10	IO	Dedicated	VDDS	dss_d1	0x00B4	0	dss_d1	Mode0	000	PUPD	D	01
P11	IO	Muxed	VDDS	dss_d9	0x00BC	0	dss_d9	Protect	111	PUPD	D	01
P12	IO	Muxed	VDDS	dss_d13	0x00C0	3	dss_d16	Protect	111	PUPD	D	01
P13	IO	Muxed	VDDS	cam_xclk	0x00DC	3		Mode0	000	PUPD	D	01
P14	IO	Muxed	VDDS	gpio_6	0x0138	2	gpio_125	Protect	111	PUPD	*	00
P15	IO	Muxed	VDDS2	uart2_rts	0x00EC	2	uart2_rx	Protect	111	PUPD	U	11
P18	IO	Muxed	VDDS2	mcbsp1_dx	0x010C	1	mcbsp1_dr	Protect	111	PUPD	D	01

Table 24−4. OMAP2420 Pin Characteristics (Continued)

Pin				Pin Control Register				Control		Pull		
Ball	**Type**	**MUX Type**	**Power**	**Name Extension**	**Offset**	**Byte**	**Pin Name**	**Reset State**	**Reset Value**	**Type**	**Reset State**	**Reset Value**
P19	IO	Muxed	VDDS2	mcbsp1_dx	0x010C	0	mcbsp1_dx	Protect	111	PUPD	D	01
P20	IO	Muxed	VDDS2	spi2_somi	0x0108	3	mcbsp1_fsr	Protect	111	PUPD	D	01
P21	IO	Muxed	VDDS2	uart1_rx	0x00C8	2	mcbsp2_clkx	Protect	111	PUPD	D	01
P3	IO	Dedicated	VDDS1	gpmc_d6	0x0084	2	gpmc_d4	Mode0	000	PUPD	U	11
P4	IO	Dedicated	VDDS1	gpmc_d6	0x0084	0	gpmc_d6	Mode0	000	PUPD	U	11
P7	O	Muxed	VDDS1	gpmc_nadv_ale	0x0094	3	gpmc_nbe0	Mode0	000	No	–	00
P8	IO	Muxed	VDDS	gpio_121	0x0130	1	gpio_122	Protect	111	No	–	00
P9	IO	Muxed	VDDS	eac_bt_fs	0x00F0	0	eac_bt_fs	Protect	111	PUPD	D	01
R1	O	Muxed	VDDS1	gpmc_nbe1	0x0098	0	gpmc_nbe1	Mode0	000	No	–	00
R10	IO	Dedicated	VDDS	dss_d5	0x00B8	0	dss_d5	Mode0	000	PUPD	D	01
R11	IO	Muxed	VDDS	dss_d13	0x00C0	0	dss_d13	Protect	111	PUPD	D	01
R12	IO	Muxed	VDDS	dss_d17	0x00C4	0	dss_d17	Protect	111	PUPD	D	01
R13	IO	Muxed	VDDS	gpio_62	0x00E0	0	gpio_62	Protect	111	PUPD	U	11
R14	IO	Muxed	VDDS	eac_ac_sclk	0x0124	1	eac_ac_fs	Protect	111	PUPD	D	01
R18	IO	Muxed	VDDS2	spi1_ncs2	0x0104	0	spi1_ncs2	Protect	111	PUPD	U	11
R19	IO	Muxed	VDDS2	spi1_ncs2	0x0104	3	spi2_simo	Protect	111	PUPD	D	01
R20	IO	Muxed	VDDS2	spi2_somi	0x0108	0	spi2_somi	Protect	111	PUPD	D	01
R3	IO	Dedicated	VDDS1	gpmc_d6	0x0084	1	gpmc_d5	Mode0	000	PUPD	U	11
R4	IO	Dedicated	VDDS1	gpmc_d10	0x0080	3	gpmc_d7	Mode0	000	PUPD	U	11
R8	IO	Muxed	VDDS	uart2_rts	0x00EC	3	eac_bt_sclk	Protect	111	PUPD	D	01
R9	IO	Muxed	VDDS	gpio_121	0x0130	0	gpio_121	Protect	111	No	–	00
T18	IO	Muxed	VDDS2	spi1_simo	0x0100	1	spi1_somi	Protect	111	PUPD	D	01
T19	IO	Muxed	VDDS2	spi1_ncs2	0x0104	2	spi2_clk	Protect	111	PUPD	D	01
T21	IO	Muxed	VDDS2	uart1_rx	0x00C8	0	uart1_rx	Protect	111	PUPD	U	11

Table 24−4. OMAP2420 Pin Characteristics (Continued)

Pin				Pin Control Register				Control		Pull		
Ball	Type	MUX Type	Power	Name Extension	Offset	Byte	Pin Name	Reset State	Reset Value	Type	Reset State	Reset Value
T3	IO	Muxed	VDDS	cam_d0	0x00D8	1	cam_hs	Protect	111	PUPD	D	01
T4	IO	Muxed	VDDS	cam_d0	0x00D8	0	cam_d0	Protect	111	PUPD	U	11
U18	IO	Muxed	VDDS2	mmc_dat_dir3	0x00FC	3	spi1_clk	Protect	111	PUPD	D	01
U19	IO	Muxed	VDDS2	spi1_simo	0x0100	2	spi1_ncs0	Protect	111	PUPD	U	11
U2	IO	Muxed	VDDS	cam_d0	0x00D8	2	cam_vs	Protect	111	PUPD	D	01
U21	IO	Muxed	VDDS2	spi1_ncs2	0x0104	1	spi1_ncs3	Protect	111	PUPD	U	11
U3	O	Muxed	VDDS	cam_xclk	0x00DC	0	cam_xclk	Mode0	000	No	–	00
U4	IO	Muxed	VDDS	cam_d4	0x00D4	1	cam_d3	Protect	111	PUPD	D	01
V10	IO	Muxed	VDDS	dss_d9	0x00BC	1	dss_d10	Protect	111	PUPD	D	01
V11	IO	Muxed	VDDS	dss_d13	0x00C0	1	dss_d14	Protect	111	PUPD	D	01
V12	IO	Muxed	VDDS	cam_xclk	0x00DC	2		Mode0	000	PUPD	–	00
V13	IO	Muxed	VDDS	gpio_62	0x00E0	3		Mode0	000	PUPD	–	00
V14	IO	Muxed	VDDS	eac_ac_mclk	0x0128	0	eac_ac_mclk	Protect	111	PUPD	D	01
V15	IO	Muxed	VDDS	eac_ac_sclk	0x0124	3	eac_ac_dout	Protect	111	PUPD	D	01
V16	O	Dedicated	VDDS	sys_xtalout	0x0134	0	sys_xtalout	Mode0	000	No	–	00
V17	IO	Muxed	VDDS	sys_xtalout	0x0134	1	gpio_36	Protect	111	No	–	00
V18	IO	Muxed	VDDS2	gpio_6	0x0138	1	gpio_124	Protect	111	PUPD	*	00
V19	IO	Muxed	VDDS2		0x00E8	3	uart2_cts	Protect	111	PUPD	U	11
V2	IO	Muxed	VDDS	cam_d4	0x00D4	3	cam_d1	Protect	111	PUPD	D	01
V20	IO	Muxed	VDDS2	spi1_simo	0x0100	0	spi1_simo	Protect	111	PUPD	D	01
V3	IO	Muxed	VDDS	cam_d4	0x00D4	2	cam_d2	Protect	111	PUPD	D	01
V4	IO	Muxed	VDDS	cam_d8	0x00D0	3	cam_d5	Protect	111	PUPD	D	01
V5	IO	Muxed	VDDS	cam_d0	0x00D8	3	cam_lclk	Protect	111	PUPD	D	01
V6	IO	Muxed	VDDS	dss_vsync	0x00CC	3	cam_d9	Protect	111	No	–	00

Table 24–4. OMAP2420 Pin Characteristics (Continued)

Pin				Pin Control Register				Control		Pull		
Ball	**Type**	**MUX Type**	**Power**	**Name Extension**	**Offset**	**Byte**	**Pin Name**	**Reset State**	**Reset Value**	**Type**	**Reset State**	**Reset Value**
V7	IO	Dedicated	VDDS	dss_vsync	0x00CC	0	dss_vsync	Mode0	000	PUPD	D	01
V8	IO	Dedicated	VDDS	dss_d1	0x00B4	1	dss_d2	Mode0	000	PUPD	D	01
V9	IO	Dedicated	VDDS	dss_d5	0x00B8	2	dss_d7	Mode0	000	PUPD	D	01
W10	IO	Muxed	VDDS	dss_d9	0x00BC	3	dss_d12	Protect	111	PUPD	D	01
W11	IO	Muxed	VDDS	dss_d13	0x00C0	2	dss_d15	Protect	111	PUPD	D	01
W12	IO	Muxed	VDDS	cam_xclk	0x00DC	1		Mode0	000	PUPD	–	00
W13	IO	Muxed	VDDS	gpio_62	0x00E0	2		Mode0	000	PUPD	–	00
W14	IO	Muxed	VDDS	sys_xtalout	0x0134	3	sys_clkout	Protect	111	PUPD	D	01
W15	IO	Muxed	VDDS	eac_ac_sclk	0x0124	2	eac_ac_din	Protect	111	PUPD	D	01
W16	IO	Muxed	VDDS	eac_ac_mclk	0x0128	1	eac_ac_rst	Protect	111	PUPD	D	01
W19	I	Muxed	VDDS2	sys_nirq	0x012C	0	sys_nirq	Protect	111	PUPD	U	11
W2	IO	Muxed	VDDS	cam_d4	0x00D4	0	cam_d4	Protect	111	PUPD	D	01
W20	IO	Muxed	VDDS2	uart2_rts	0x00EC	0	uart2_rts	Protect	111	PUPD	U	11
W3	IO	Muxed	VDDS	cam_d8	0x00D0	2	cam_d6	Protect	111	No	–	00
W4	IO	Muxed	VDDS	eac_bt_fs	0x00F0	2	eac_bt_dout	Protect	111	PUPD	D	01
W5	IO	Muxed	VDDS	sys_nirq	0x012C	2	gpio_119	Protect	111	No	–	00
W6	IO	Dedicated	VDDS	uart1_rx	0x00C8	3	dss_pclk	Mode0	000	PUPD	D	01
W7	IO	Muxed	VDDS	dss_vsync	0x00CC	2	dss_acbias	Protect	111	PUPD	D	01
W8	IO	Dedicated	VDDS	dss_d1	0x00B4	3	dss_d4	Mode0	000	PUPD	D	01
W9	IO	Muxed	VDDS	dss_d5	0x00B8	3	dss_d8	Protect	111	PUPD	D	01
Y10	IO	Muxed	VDDS	dss_d9	0x00BC	2	dss_d11	Protect	111	PUPD	D	01
Y11	IO	Muxed	VDDS		0x00E8	0		Protect	111	PUPD	D	01
Y12	IO	Muxed	VDDS	gpio_62	0x00E0	1		Mode0	000	PUPD	–	00
Y13	IO	Muxed	VDDS		0x00E4	0		Mode0	000	PUPD	–	00

Table 24−4. OMAP2420 Pin Characteristics (Continued)

Pin				Pin Control Register				Control		Pull		
Ball	Type	MUX Type	Power	Name Extension	Offset	Byte	Pin Name	Reset State	Reset Value	Type	Reset State	Reset Value
Y14	I	Dedicated	VDDS	eac_ac_mclk	0x0128	2	sys_nrespwron	Mode0	000	No	–	00
Y15	IO	Muxed	VDDS	eac_ac_sclk	0x0124	0	eac_ac_sclk	Protect	111	PUPD	D	01
Y16	IO	Dedicated	VDDS	eac_ac_mclk	0x0128	3	sys_nreswarm	Mode0	000	PUPD	U	11
Y17	I	Dedicated	VDDS	gpio_121	0x0130	2	sys_32k	Mode0	000	No	–	00
Y18	I	Dedicated	VDDS	gpio_121	0x0130	3	sys_xtalin	Mode0	000	No	–	00
Y2	IO	Muxed	VDDS	cam_d8	0x00D0	1	cam_d7	Protect	111	No	–	00
Y20	O	Dedicated	VDDS2	sys_nirq	0x012C	1	sys_nvmode	Mode0	000	No	–	00
Y21	IO	Muxed	VDDS2	jtag_emu0	0x013C	0	jtag_emu0	Protect	111	PUPD	U	11
Y3	IO	Muxed	VDDS	eac_bt_fs	0x00F0	1	eac_bt_din	Protect	111	PUPD	D	01
Y4	IO	Muxed	VDDS	cam_d8	0x00D0	0	cam_d8	Protect	111	No	–	00
Y5	IO	Muxed	VDDS	sys_nirq	0x012C	3	gpio_120	Protect	111	No	–	00
Y6	IO	Dedicated	VDDS	dss_vsync	0x00CC	1	dss_hsync	Mode0	000	PUPD	U	11
Y7	IO	Dedicated	VDDS	sdrc_dqs1	0x00B0	3	dss_d0	Mode0	000	PUPD	D	01
Y8	IO	Dedicated	VDDS	dss_d1	0x00B4	2	dss_d3	Mode0	000	PUPD	D	01
Y9	IO	Dedicated	VDDS	dss_d5	0x00B8	1	dss_d6	Mode0	000	PUPD	D	01

24.4.1.3 Power Voltage Pins

Pins not used for interfacing with external devices are used for OMAP2420 alimentation. Multiple pins are connected to the same voltage. Table 24−5 lists the voltage definitions.

Table 24−5. Power Pins Definition

Pin	Description
V_{DD}	OMAP processor cores and logic operating at full speed
	OMAP processor cores and logic operating in low-voltage functional mode
	Low voltage retention (low IDDQ mode)
V_{DDS}, V_{DDS2}	Supply voltage for I/O macros (except GPMC, SDRC)
V_{DDS1}	Supply voltage for I/O to memory interfaces (GPMC, SDRC)
V_{DDA}	Analog supply voltage for video DAC (OMAP2420 only)
VDD_PLL	Supply voltage for APLL and DPLLs
VDD_DLL[1]	Supply voltage for SDRC DLL core operating at full speed
	Supply voltage for SDRC DLL core operating in low-voltage functional mode
V_{SS}	Common ground
V_{SSA}	Analog ground for video DAC (OMAP2420 only)
V_{PP}	Supply voltage for internal eFuse. Can be connected to VDD or left open. The TI recommendation is to leave V_{PP} unconnected.

1) VDD_DLL must operate at the same voltage setting as VDD

24.4.1.4 OMAP2420 Package

Table 24−6. OMAP2420 Package

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
AA	vpp	vpp	vss		vdds		vss		vdds		vddpll		vss	tv_rref	vdds	tv_cvbs	sys_clkreq	tv_vref	vdds	jtag_ntrst	jtag_emu1
Y	vdd	cam_d7	eac_bt_din	cam_d8	gpio_120	dss_hsync	dss_d0	dss_d3	dss_d6	dss_d11				sys_nrespwron	eac_ac_sclk	sys_nreswarm	sys_32k	sys_xtalin	vdda-dac	sys_nvmode	jtag_emu0
W	vdds	cam_d4	cam_d6	eac_bt_dout	gpio_119	dss_pclk	dss_acbias	dss_d4	dss_d8	dss_d12	dss_d15			sys_clkout	eac_ac_din	eac_ac_rst	vss	vssa-dac	sys_nirq	uart2_rts	vdd
V	vdd	cam_d1	cam_d2	cam_d5	cam_lclk	cam_d9	dss_vsync	dss_d2	dss_d7	dss_d10	dss_d14			eac_ac_mclk	eac_ac_dout	sys_xtalout	gpio_36	gpio_124	uart2_cts	spi1_simo	vdds2
U	vss	cam_vs	cam_xclk	cam_d3														spi1_clk	spi1_cs0	vss	spi1_cs3
T	vdd	vdds	cam_hs	cam_d0														spi1_somi	spi2_clk	vdd	uart1_rx
R	gpmc_nbe1	vss	gpmc_d5	gpmc_d7				eac_bt_sclk	gpio_121	dss_d5	dss_d13	dss_d17	gpio_62	eac_ac_fs				spi1_cs2	spi2_simo	spi2_somi	vss
P	gpmc_wait3	vdd	gpmc_d4	gpmc_d6			gpmc_nbe0	gpio_122	eac_bt_fs	dss_d1	dss_d9	dss_d16		gpio_125	uart2_rx			mcbsp1_dr	mcbsp1_dx	mcbsp1_fsr	mcbsp2_clkx
N	vss	gpmc_ncs3	gpmc_d2	gpmc_d3			gpmc_wait1	gpmc_ncs1						uart2_tx	spi1_cs1			hdq_sio	mcbsp1_clkx	vdd	vdds2
M	gpmc_wait2	vdds1	gpmc_d0	gpmc_d1			gpmc_d9	gpmc_d8						spi2_cs0	mcbsp1_clkr			mcbsp_clks	i2c1_scl	vss	mcbsp2_dr
L	vdd	gpmc_ncs7	gpmc_wait0	gpmc_ncs0			gpmc_d10	gpmc_d12						mcbsp1_fsx	i2c1_sda			uart3_cts_rctx	uart3_rts_sd	uart1_tx	vdd
K	gpmc_ncs6	vss	gpmc_nadv_ale	gpmc_nwe			gpmc_d13	gpmc_d11						uart3_rx_irrx	uart3_tx_irtx			usb0_dat	usb0_txen	usb0_vm	vss
J	vdds1	vdd	gpmc_a1	gpmc_clk			gpmc_d15	gpmc_d14						usb0_se0	i2c2_scl			usb0_rcv	usb0_vp	usb0_puen	vss

Table 24−6. OMAP2420 Package (Continued)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
H	gpmc_ncs5	gpmc_noe	gpmc_a3	gpmc_a2			sdrc_d1	sdrc_d7	sdrc_d14	sdrc_d11	sdrc_d9	sdrc_d17	sdrc_d28	mmc_dat1	mmc_clki			mmc_cmd	i2c2_sda	vdd	uart1_rts
G	vss	gpmc_nwp	gpmc_a4	gpmc_a8				sdrc_a6	sdrc_dqs1	sdrc_d10	sdrc_d8	sdrc_dm1	sdrc_d20	sdrc_d24				mmc_cmd_dir	mmc_clko	vss	vdds2
F	gpmc_ncs4	vdd	gpmc_a6	gpmc_a5														mmc_dat_dir2	mmc_dat_dir0	mmc_dat0	vdd
E	vdds1	gpmc_ncs2	gpmc_a10	gpmc_a7	gpio_6													mmc_dat_dir3	mmc_dat2	mmc_dat_dir1	vss
D	vdds1	vss	gpmc_a9	sdrc_d2	sdrc_dqs0	sdrc_d13	sdrc_a3	sdrc_a1	sdrc_a8	sdrc_ba1	sdrc_a12	sdrc_ncs0	sdrc_cke0	sdrc_ncas	sdrc_d18	sdrc_dqs2	sdrc_dm3	sdrc_d25	mmc_dat3	vdd	uart1_cts
C	vdd	sdrc_d0	sdrc_d5	sdrc_d4	sdrc_d6	sdrc_d15	sdrc_a5	sdrc_a7	sdrc_a10	sdrc_a11	sdrc_ba0	sdrc_ncs1	sdrc_nwe	sdrc_clk	sdrc_d19	sdrc_d23	sdrc_dm2	sdrc_d30	sdrc_d29	sdrc_d31	vss
B	jtag_tck	sdrc_d3	sdrc_a14	sdrc_a13	sdrc_dm0	sdrc_a4	sdrc_a2	vddddll	sdrc_a0	sdrc_a9	vss	sdrc_nras	sdrc_cke1	sdrc_nclk	sdrc_d16	sdrc_d21	sdrc_d22	sdrc_dqs3	sdrc_d26	sdrc_d27	jtag_tms
A	jtag_rtck	vss	vdds1	vss	vss	vdds1	vdds1	vss	vss	vdds1	sdrc_d12	vss	vdds1	vdds1	vss	vss	vdds1	vss	vdds1	jtag_tdi	jtag_tdo

24.4.1.5 Multiplexing Summary

Table 24−7 represents the OMAP2420 functional multiplexing. The purpose of this table is to help you selecti the interface and can be checked line by line if conflicts occur. To solve conflicts, see Section 24.6.

Table 24−7. Multiplexing Summary

Ball	Pin List Specification						
	Mode0	**Mode1**	**Mode2**	**Mode3**	**Mode4**	**Mode5**	**Mode6**
D4	sdrc_d2						
H7	sdrc_d1						
C2	sdrc_d0						
E3	gpmc_a10		sys_ndmareq5	gpio_3			
D3	gpmc_a9		sys_ndmareq4	gpio_4			
G4	gpmc_a8		sys_ndmareq3	gpio_5			
E4	gpmc_a7		sys_ndmareq2	gpio_6			
F3	gpmc_a6	dss_d23		gpio_7			
F4	gpmc_a5	dss_d22		gpio_8			
G3	gpmc_a4	dss_d21		gpio_9			
H3	gpmc_a3	dss_d20		gpio_10			
H4	gpmc_a2	dss_d19		gpio_11			
J3	gpmc_a1	dss_d18		gpio_12			
J7	gpmc_d15			gpio_13			
J8	gpmc_d14			gpio_14			
K7	gpmc_d13			gpio_15			
L8	gpmc_d12			gpio_16			
K8	gpmc_d11			gpio_17			
L7	gpmc_d10			gpio_18			
M7	gpmc_d9			gpio_19			
M8	gpmc_d8			gpio_20			
R4	gpmc_d7						
P4	gpmc_d6						
R3	gpmc_d5						
P3	gpmc_d4						
N4	gpmc_d3						
N3	gpmc_d2						
M4	gpmc_d1						

Table 24−7. Multiplexing Summary (Continued)

Ball	**Pin List Specification**						
	Mode0	**Mode1**	**Mode2**	**Mode3**	**Mode4**	**Mode5**	**Mode6**
M3	gpmc_d0						
J4	gpmc_clk			gpio_21			
L4	gpmc_ncs0						
N8	gpmc_ncs1			gpio_22			
E2	gpmc_ncs2			gpio_23			
N2	gpmc_ncs3	gpmc_io_dir		gpio_24			
F1	gpmc_ncs4			gpio_25			
H1	gpmc_ncs5			gpio_26			
K1	gpmc_ncs6			gpio_27			
L2	gpmc_ncs7	gpmc_io_dir		gpio_28			
K3	gpmc_nadv_ale						
H2	gpmc_noe						
K4	gpmc_nwe						
P7	gpmc_nbe0			gpio_29			
R1	gpmc_nbe1			gpio_30			
G2	gpmc_nwp			gpio_31			
L3	gpmc_wait0						
N7	gpmc_wait1			gpio_33			
M1	gpmc_wait2			gpio_34			
P1	gpmc_wait3			gpio_35			
C14	sdrc_clk						
B14	sdrc_nclk						
D12	sdrc_ncs0						
C12	sdrc_ncs1			gpio_37			
D13	sdrc_cke0						
B13	sdrc_cke1			gpio_38			
B12	sdrc_nras						
D14	sdrc_ncas						
C13	sdrc_nwe						
B5	sdrc_dm0						
G12	sdrc_dm1						

Table 24−7. Multiplexing Summary (Continued)

Ball	**Pin List Specification**						
	Mode0	**Mode1**	**Mode2**	**Mode3**	**Mode4**	**Mode5**	**Mode6**
C17	sdrc_dm2						
D17	sdrc_dm3						
D5	sdrc_dqs0						
G9	sdrc_dqs1						
D16	sdrc_dqs2						
B18	sdrc_dqs3						
Y7	dss_d0						
P10	dss_d1						
V8	dss_d2						
Y8	dss_d3						
W8	dss_d4						
R10	dss_d5						
Y9	dss_d6						
V9	dss_d7						
W9	dss_d8			gpio_38			
P11	dss_d9			gpio_39			
V10	dss_d10			gpio_40			
Y10	dss_d11			gpio_41			
W10	dss_d12			gpio_42			
R11	dss_d13			gpio_43			
V11	dss_d14			gpio_44			
W11	dss_d15			gpio_45			
P12	dss_d16			gpio_46			
R12	dss_d17			gpio_47			
D21	uart1_cts		dss_d18	gpio_32			
H21	uart1_rts		dss_d19	gpio_8			
L20	uart1_tx		dss_d20	gpio_9			
T21	uart1_rx		dss_d21	gpio_10			
M21	mcbsp2_dr		dss_d22	gpio_11			
P21	mcbsp_clkx		dss_d23	gpio_12			
W6	dss_pclk						

Table 24−7. Multiplexing Summary (Continued)

Ball	**Pin List Specification**						
	Mode0	**Mode1**	**Mode2**	**Mode3**	**Mode4**	**Mode5**	**Mode6**
V7	dss_vsync						
Y6	dss_hsync						
W7	dss_acbias		mcbsp2_fsx	gpio_48			
V6	cam_d9	hw_dbg11		gpio_53			etk_d11
Y4	cam_d8	hw_dbg10		gpio_54			etk_d10
Y2	cam_d7	hw_dbg9					etk_d9
W3	cam_d6	hw_dbg8					etk_d8
V4	cam_d5	hw_dbg7	mcbsp1_clkr	gpio_49			etk_d7
W2	cam_d4	hw_dbg6	mcbsp1_fsr	gpio_50			etk_d6
U4	cam_d3	hw_dbg5	mcbsp1_dr	gpio_51			etk_d5
V3	cam_d2	hw_dbg4†	mcbsp1_clkx	gpio_52			etk_d4
V2	cam_d1	hw_dbg3†		gpio_53			etk_d3
T4	cam_d0	hw_dbg2†		gpio_54			etk_d2
T3	cam_hs	hw_dbg1†	mcbsp1_dx	gpio_55			etk_d1
U2	cam_vs	hw_dbg0†	mcbsp1_fsx	gpio_56			etk_d0
V5	cam_lclk		mcbsp_clks	gpio_57			etk_c1
U3	cam_xclk	l4_ext_trig					etk_c2
W12		uart1_tx	usb1_se0	gpio_59			
V12		uart1_rts	usb1_rcv	gpio_25			
P13		uart1_cts	usb1_txen	gpio_61			
R13	gpio_62	uart1_rx	usb1_dat	gpio_62			
Y12		eac_md_sclk		gpio_63			
W13		eac_md_din		gpio_64			
V13		eac_md_dout		gpio_65			
Y13		eac_md_fs		gpio_66			
AA10		usb2_se0	sys_ndmareq0	gpio_13			
AA6		usb2_rcv	sys_ndmareq1	gpio_14	cam_d8		
AA4		usb2_tllse0		gpio_15	cam_d7		
Y11		usb2_dat	sys_clkout2	gpio_16			
AA12		usb2_txen		gpio_17			
AA8			l4_ext_trig	gpio_58	cam_d6		

Table 24−7. Multiplexing Summary (Continued)

Ball	Pin List Specification						
	Mode0	**Mode1**	**Mode2**	**Mode3**	**Mode4**	**Mode5**	**Mode6**
V19	uart2_cts	usb1_rcv	gpt9_pwm_evt	gpio_67			
W20	uart2_rts	usb1_txen	gpt10_pwm_evt	gpio_68			
N14	uart2_tx	usb1_se0	gpt11_pwm_evt	gpio_69			
P15	uart2_rx	usb1_dat	gpt12_pwm_evt	gpio_70			
R8	eac_bt_sclk			gpio_71			etk_d11
P9	eac_bt_fs			gpio_72			etk_d10
Y3	eac_bt_din			gpio_73			etk_d9
W4	eac_bt_dout			gpio_74			etk_d8
G19	mmc_clko						
H18	mmc_cmd						
F20	mmc_dat0						
H14	mmc_dat1			gpio_75			
E19	mmc_dat2		uart2_cts	gpio_76			
D19	mmc_dat3		l4_ext_trig	gpio_77			
F19	mmc_dat_dir0			gpio_7			
E20	mmc_dat_dir1		uart2_rts	gpio_78			
F18	mmc_dat_dir2		uart2_tx	gpio_79			
E18	mmc_dat_dir3		uart2_rx	gpio_80			
G18	mmc_cmd_dir			gpio_8			
H15	mmc_clki			gpio_59			
U18	spi1_clk			gpio_81			
V20	spi1_simo			gpio_82			
T18	spi1_somi			gpio_83			
U19	spi_ncs0			gpio_84			
N15	spi_ncs1			gpio_85			
R18	spi_ncs2			gpio_86			
U21	spi_ncs3			gpio_87			
T19	spi2_clk			gpio_88			
R19	spi2_simo	gpt10_pwm_evt		gpio_89			
R20	spi2_somi	gpt11_pwm_evt		gpio_90			
M14	spi2_cs0	gpt12_pwm_evt		gpio_91			

Table 24−7. Multiplexing Summary (Continued)

Ball	**Pin List Specification**						
	Mode0	**Mode1**	**Mode2**	**Mode3**	**Mode4**	**Mode5**	**Mode6**
M15	mcbsp1_clkr			gpio_92			
P20	mcbsp1_fsr			gpio_93	spi2_ncs1		
P19	mcbsp1_dx			gpio_94			
P18	mcbsp1_dr			gpio_95			
M18	mcbsp_clks			gpio_96			
L14	mcbsp1_fsx			gpio_97			
N19	mcbsp1_clkx			gpio_98			
M19	i2c1_scl						
L15	i2c1_sda						
J15	i2c2_scl		gpt9_pwm_evt	gpio_99			
H19	i2c2_sda		spi2_cs1	gpio_100			
N18	hdq_sio	usb2_tllse0	sys_altclk	gpio_101			
L18	uart3_cts_rctx	uart3_rx_irrx		gpio_102			
L19	uart3_rts_sd	uart3_tx_irtx		gpio_103			
K15	uart3_tx_irtx	uart3_rctx		gpio_104			
K14	uart3_rx_irrx			gpio_105			
AA16	tv_cvbs						
AA18	tv_vref						
AA14	tv_rref						
J20	usb0_puen	mcbsp2_dx		gpio_106			
J19	usb0_vp	mcbsp2_dr		gpio_107			
K20	usb0_vm	mcbsp2_clkx		gpio_108	uart2_rx		
J18	usb0_rcv	mcbsp2_fsx		gpio_109	uart2_cts		
K19	usb0_txen	uart3_cts_rctx	uart2_cts	gpio_110			
J14	usb0_se0	uart3_tx_irtx	uart2_tx	gpio_111	uart2_rx		
K18	usb0_dat	uart3_rx_irrx	uart2_rx	gpio_112	uart2_tx		
Y15	eac_ac_sclk	mcbsp2_clkx		gpio_113			
R14	eac_ac_fs	mcbsp2_fsx		gpio_114			
W15	eac_ac_din	mcbsp2_dr		gpio_115			
V15	eac_ac_dout	mcbsp2_dx		gpio_116			
V14	eac_ac_mclk			gpio_117			

Table 24−7. Multiplexing Summary (Continued)

Ball	Pin List Specification						
	Mode0	**Mode1**	**Mode2**	**Mode3**	**Mode4**	**Mode5**	**Mode6**
W16	eac_ac_rst	eac_bt_din	loop_eac_bt_din	gpio_118			
Y14	sys_nrespwron						
Y16	sys_nreswarm						
W19	sys_nirq			gpio_60			
Y20	sys_nvmode						
W5	gpio_119			gpio_119		sys_boot0	etk_d12
Y5	gpio_120			gpio_120		sys_boot1	etk_d13
R9	gpio_121			gpio_121	jtag_emu2	sys_boot2	etk_d14
P8	gpio_122			gpio_122	jtag_emu3	sys_boot3	etk_d15
Y17	sys_32k						
Y18	sys_xtalin						
V16	sys_xtalout						
V17	gpio_36			gpio_36		sys_boot4	
AA17	sys_clkreq			gpio_52			
W14	sys_clkout			gpio_123			
E5	gpio_6	tv_detect		gpio_6			
V18	gpio_124			gpio_124		sys_boot5	
P14	gpio_125	sys_jtagsel1	sys_jtagsel1	gpio_125			
AA21				gpio_126			
Y21				gpio_127			

24.4.1.6 L4 Interconnect Interface

The pinout control module has an interface with the L4 interconnect via a dedicated target agent (TA). This TA, which is part of the L4 interconnect, provides status and configuration registers as listed in Table 24−8. By default, TA register values are functional, but it is possible to overwrite them if necessary. For a complete description, see Chapter 6.

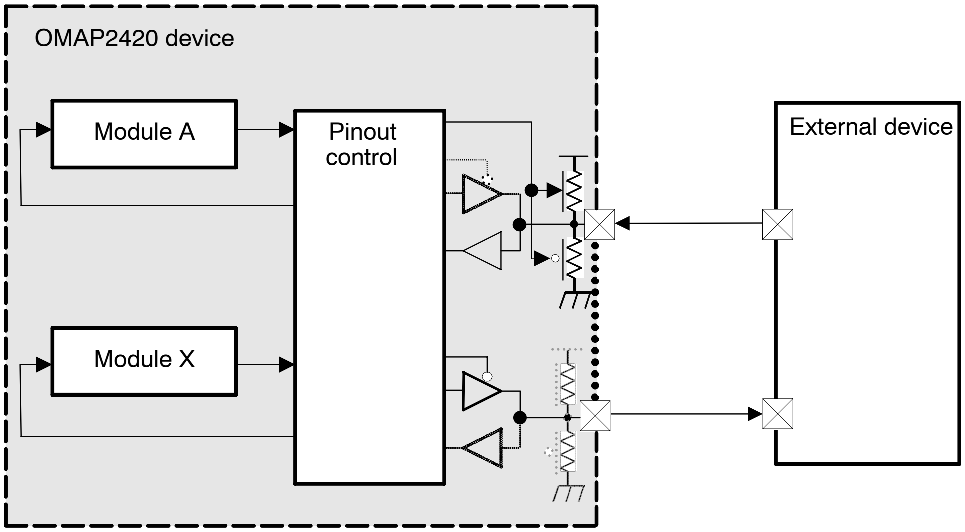
Table 24−8. L4 Interconnect Registers

Register Name	Type	Register Width (Bits)	Physical Address
COMPONENT	R	32	0x480B 1000
AGENT_CONTROL	RW	32	0x480B 1020
AGENT_STATUS	R	32	0x480B 1028

24.5 Pinout Environment

Pinout control enables the routing of internal module input/output to the chip boundary for interconnection with an external device.

Figure 24−7. Pinout Overview



For each external device interfacing with the OMAP2420 device, it is software's responsibility to ensure proper programming of the pinout control module. For maximum flexibility, each pin can be programmed independently.

24.5.1 Pinout Functional Interface

This section identifies all possible pins that can be configured for each module signal classified by function.

All possible interfaces for each function are summarized in the appropriate table.

Note:

The purpose of this example is to show table organization; all signals and functions do not correspond to real OMAP 2420 signals or function.

Table 24−9. Pinout Interface

Function n	Description	DIR	Ball	Alternate Functions					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
FUNCn.SIG1	Signal 1 of function n Description	I	T3	▨	funcx.sig2	funcy.sig1	funck.sig5	■	■
FUNCn.SIG2	Signal 2 of function n Description	I	U2	funcx.sig2	funcy.sig4	▨	funck.sig6	■	■
FUNCn.SIG3	Signal 3 of function n Description	IO	U3	funcz.sig1	▨	funcy.sig2	■	■	■
FUNCn.SIG4	Signal 4 of function n Description	O	W3	▨	funck.sig2	■	■	■	■
			V2	funcy.sig3	■	■	▨	■	■

- This table is related to function n: All interface signals related to function n are listed in the Interface column.

- A maximum of four different signals, named SIG1, SIG2, SIG3, and SIG4, are available for function n.

- A short description of each signal can be found in description column. Some outputs are described as open drain. This means the output is not able to drive a high level on the pin but only low level or high impedance. These open drain outputs must use a pullup (internal or external) to ensure correct behavior. Open drain outputs are: HDQ.SIO, I2C1.SCL, I2C1.SDA, I2C2.SCL, I2C2.SDA, SYS.nRESWARM.

- The direction of the signal can be found in column DIR.

  - Notice that this direction is related to the signal not to the pin: SIG1 is Input only but pin T3 can be an I/O pin. Information related to the pin itself can be found in Section 24.4.1.2.

  - Information related to the direction of alternates function is not in this table but in the dedicated table for the alternate function. Funcx.sig2 direction would be available in function x table.

- The Ball column gives pin index to identify the concerned ball on the chip package and where the signal is multiplexed on the OMAP2420 device.

  - One signal can be multiplexed on several pins. In the example, SIG4 is both available on pin W3 and V2. It is the user responsibility to configure SIG4 on one pin only. In case of multiple affectation of a signal, the device is protected from damage by hardware, but only one affectation is functional.

- A signal is effectively associated with a pin only if the corresponding mode is selected using pinout control register programming. The mode associated to the described function is highlighted in gray in the corresponding cell (pinout control register configuration). In this example:

- FUNCn.SIG1 is selected if mode 0 is programmed for pin T3. Mode 0 is programmed by writing 000 into the MUXMODE field within the pin control register field for pin T3.
- Funcn.SIG2 is selected if mode2 is programmed for pin U2.
- Funcn.SIG3 is selected if mode1 is programmed for pin U3.
- Funcn.SIG4 is selected if mode1 is programmed for pin W3 or if mode3 is programmed for pin V2

❑ Alternate functions inform on the signals which can be multiplexed on the same pin. In this example, if mode1 is programmed for pin T3 then funcx.sig2 is available in spite of FUNCn.SIG1. Alternate function visibility enable programmer to identify potential resource conflicts on pin. If several signals which must be used simultaneously in user application are on the same line of the table, it must be checked that this alternate signals can be multiplexed on other pins.

❑ When shaded black cells appears in the table, that means that this mode is not used. In this example, Mode4 and mode 5 are never used; Mode 3 is not used on pin U3.

24.5.1.1 Memory Interface

A second level of multiplexing is at the module level for memory interfaces; see Chapter 12 for internal multiplexing management.

Table 24−10. GPMC Interface

GPMC	Description	DIR	Ball	ALTERNATE FUNCTIONS					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
GPMC.A1	General Purpose Memory Address Bit1	O	J3		dss.d18	■	gpio.12	■	■
GPMC.A2	General Purpose Memory Address Bit2	O	H4		dss.d19	■	gpio.11	■	■
GPMC.A3	General Purpose Memory Address Bit3	O	H3		dss.d20	■	gpio.10	■	■
GPMC.A4	General Purpose Memory Address Bit4	O	G3		dss.d21	■	gpio.9	■	■
GPMC.A5	General Purpose Memory Address Bit5	O	F4		dss.d22	■	gpio.8	■	■
GPMC.A6	General Purpose Memory Address Bit6	O	F3		dss.d23	■	gpio.7	■	■
GPMC.A7	General Purpose Memory Address Bit7	O	E4		■	sys.ndma-req2	gpio.6	■	■

Table 24−10. GPMC Interface (Continued)

GPMC	Description	DIR	Ball	ALTERNATE FUNCTIONS					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
GPMC.A8	General Purpose Memory Address Bit8	O	G4			sys.ndma-req3	gpio.5		
GPMC.A9	General Purpose Memory Address Bit9	O	D3			sys.ndma-req4	gpio.4		
GPMC.A10	General Purpose Memory Address Bit10	O	E3			sys.ndma-req5	gpio.3		
GPMC.A11	General Purpose Memory Address Bit11	IO	C2	Muxed with SDRC.D0					
GPMC.A12	General Purpose Memory Address Bit12	IO	H7	Muxed with SDRC.D1					
GPMC.A13	General Purpose Memory Address Bit13	IO	D4	Muxed with SDRC.D2					
GPMC.A14	General Purpose Memory Address Bit14	IO	B2	Muxed with SDRC.D3					
GPMC.A15	General Purpose Memory Address Bit15	IO	C4	Muxed with SDRC.D4					
GPMC.A16	General Purpose Memory Address Bit16	IO	C3	Muxed with SDRC.D5					
GPMC.A17	General Purpose Memory Address Bit17	IO	C5	Muxed with SDRC.D6					
GPMC.A18	General Purpose Memory Address Bit18	IO	H8	Muxed with SDRC.D7					
GPMC.A19	General Purpose Memory Address Bit19	IO	G11	Muxed with SDRC.D8					
GPMC.A20	General Purpose Memory Address Bit20	IO	H11	Muxed with SDRC.D9					
GPMC.A21	General Purpose Memory Address Bit21	IO	G10	Muxed with SDRC.D10					
GPMC.A22	General Purpose Memory Address Bit22	IO	H10	Muxed with SDRC.D11					
GPMC.A23	General Purpose Memory Address Bit23	IO	A11	Muxed with SDRC.D12					

Table 24−10. GPMC Interface (Continued)

GPMC	Description Interface	DIR	Ball	ALTERNATE FUNCTIONS					
				Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
GPMC.A24	General Purpose Memory Address Bit24	IO	D6	Muxed with SDRC.D13					
GPMC.A25	General Purpose Memory Address Bit25	IO	H9	Muxed with SDRC.D14					
GPMC.A26	General Purpose Memory Address Bit26	IO	C6	Muxed with SDRC.D15					
GPMC.D0	GPMC Data Bit 0	IO	M3						
GPMC.D1	GPMC Data Bit 1	IO	M4						
GPMC.D2	GPMC Data Bit 2	IO	N3						
GPMC.D3	GPMC Data Bit 3	IO	N4						
GPMC.D4	GPMC Data Bit 4	IO	P3						
GPMC.D5	GPMC Data Bit 5	IO	R3						
GPMC.D6	GPMC Data Bit 6	IO	P4						
GPMC.D7	GPMC Data Bit 7	IO	R4						
GPMC.D8	GPMC Data Bit 8	IO	M8				gpio.20		
GPMC.D9	GPMC Data Bit 9	IO	M7				gpio.19		
GPMC.D10	GPMC Data Bit 10	IO	L7				gpio.18		
GPMC.D11	GPMC Data Bit 11	IO	K8				gpio.17		
GPMC.D12	GPMC Data Bit 12	IO	L8				gpio.16		
GPMC.D13	GPMC Data Bit 13	IO	K7				gpio.15		
GPMC.D14	GPMC Data Bit 14	IO	J8				gpio.14		
GPMC.D15	GPMC Data Bit 15	IO	J7				gpio.13		
GPMC. nCS0	GPMC Chip Select Bit 0	O	L4						

Table 24−10. GPMC Interface (Continued)

GPMC	Description	DIR	Ball	ALTERNATE FUNCTIONS					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
GPMC. nCS1	GPMC Chip Select Bit 1	O	N8				gpio.22		
GPMC. nCS2	GPMC Chip Select Bit 2	O	E2				gpio.23		
GPMC. nCS3	GPMC Chip Select Bit 3	O	N2		gpmc. io.dir		gpio.24		
GPMC. nCS4	GPMC Chip Select Bit 4	O	F1				gpio.25		
GPMC. nCS5	GPMC Chip Select Bit 5	O	H1				gpio.26		
GPMC. nCS6	GPMC Chip Select Bit 6	O	K1				gpio.27		
GPMC.nCS7	GPMC Chip Select Bit 7	O	L2		gpmc. io.dir		gpio.28		
GPMC.IO_ DIR	GPMC IO Direction Control	O	N2	gpmc.ncs3			gpio.24		
			L2	gpmc.ncs7			gpio.28		
GPMC.CLK	GPMC Clock	IO	J4				gpio.21		
GPMC. nADV/ALE	Address Valid or Address Latch Enable	O	K3						
GPMC.nOE	Output Enable	O	H2						
GPMC.nWE	Write Enable	O	K4						
GPMC. nBE0	Lower Byte Enable. Also used for Command Latch Enable	O	P7				gpio.29		
GPMC. nBE1	Upper Byte Enable	O	R1				gpio.30		
GPMC.nWP	Flash Write Protect	O	G2				gpio.31		
GPMC. WAIT0	External indication of Wait	I	L3						
GPMC. WAIT1	External indication of Wait	I	N7				gpio.33		
GPMC. WAIT2	External Indication of Wait	I	M1				gpio.34		
GPMC. WAIT3	External Indication of Wait	I	P1				gpio.35		

Table 24−11. SDRC Interface

SDRAM	Description	DIR	Ball	ALTERNATE FUNCTIONS					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
SDRC.D0	SDRAM Data Bit 0	IO	C2	Muxed with GPMC.A11					
SDRC.D1	SDRAM Data Bit 1	IO	H7	Muxed with GPMC.A12					
SDRC.D2	SDRAM Data Bit 2	IO	D4	Muxed with GPMC.A13					
SDRC.D3	SDRAM Data Bit 3	IO	B2	Muxed with GPMC.A14					
SDRC.D4	SDRAM Data Bit 4	IO	C4	Muxed with GPMC.A15					
SDRC.D5	SDRAM Data Bit 5	IO	C3	Muxed with GPMC.A16					
SDRC.D6	SDRAM Data Bit 6	IO	C5	Muxed with GPMC.A17					
SDRC.D7	SDRAM Data Bit 7	IO	H8	Muxed with GPMC.A18					
SDRC.D8	SDRAM Data Bit 8	IO	G11	Muxed with GPMC.A19					
SDRC.D9	SDRAM Data Bit 9	IO	H11	Muxed with GPMC.A20					
SDRC.D10	SDRAM Data Bit 10	IO	G10	Muxed with GPMC.A21					
SDRC.D11	SDRAM Data Bit 11	IO	H10	Muxed with GPMC.A22					
SDRC.D12	SDRAM Data Bit 12	IO	A11	Muxed with GPMC.A23					
SDRC.D13	SDRAM Data Bit 13	IO	D6	Muxed with GPMC.A24					
SDRC.D14	SDRAM Data Bit 14	IO	H9	Muxed with GPMC.A25					
SDRC.D15	SDRAM Data Bit 15	IO	C6	Muxed with GPMC.A26					
SDRC.D16	SDRAM Data Bit 16	IO	B15						
SDRC.D17	SDRAM Data Bit 17	IO	H12						
SDRC.D18	SDRAM Data Bit 18	IO	D15						
SDRC.D19	SDRAM Data Bit 19	IO	C15						
SDRC.D20	SDRAM Data Bit 20	IO	G13						

Table 24−11. SDRC Interface (Continued)

SDRAM	Description	DIR	Ball	ALTERNATE FUNCTIONS					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
SDRC.D21	SDRAM Data Bit 21	IO	B16						
SDRC.D22	SDRAM Data Bit 22	IO	B17						
SDRC.D23	SDRAM Data Bit 23	IO	C16						
SDRC.D24	SDRAM Data Bit 24	IO	G14						
SDRC.D25	SDRAM Data Bit 25	IO	D18						
SDRC.D26	SDRAM Data Bit 26	IO	B19						
SDRC.D27	SDRAM Data Bit 27	IO	B20						
SDRC.D28	SDRAM Data Bit 28	IO	H13						
SDRC.D29	SDRAM Data Bit 29	IO	C19						
SDRC.D30	SDRAM Data Bit 30	IO	C18						
SDRC.D31	SDRAM Data Bit 31	IO	C20						
SDRC. BA0	SDRAM Bank Select 0	O	C11						
SDRC. BA1	SDRAM Bank Select 1	O	D10						
SDRC.A0	SDRAM Address Bit 0	O	B9						
SDRC.A1	SDRAM Address Bit 1	O	D8						
SDRC.A2	SDRAM Address Bit 2	O	B7						
SDRC.A3	SDRAM Address Bit 3	O	D7						
SDRC.A4	SDRAM Address Bit 4	O	B6						
SDRC.A5	SDRAM Address Bit 5	O	C7						
SDRC.A6	SDRAM Address Bit 6	O	G8						
SDRC.A7	SDRAM Address Bit 7	O	C8						

Table 24−11. SDRC Interface (Continued)

SDRAM	Description	DIR	Ball	ALTERNATE FUNCTIONS					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
SDRC.A8	SDRAM Address Bit 8	O	D9						
SDRC.A9	SDRAM Address Bit 9	O	B10						
SDRC.A10	SDRAM Address Bit 10	O	C9						
SDRC.A11	SDRAM Address Bit 11	O	C10						
SDRC.A12	SDRAM addr Bit 12 or V-Sync Segment Addr 0	O	D11				gpio.2		
SDRC.A13	SDRAM Addr Bit 13 or V-Sync Segment Addr 1	O	B4				gpio.1		
SDRC.A14	SDRAM Addr Bit 14 or V-Sync Segment Addr 2	O	B3				gpio.0		
SDRC. nCS0	Chip Select 0	O	D12						
SDRC. nCS1	Chip Select 1	O	C12				gpio.37		
SDRC. CLK	Clock	IO	C14						
SDRC. nCLK	Clock Invert	O	B14						
SDRC. CKE0	Clock Enable0	O	D13						
SDRC. CKE1	Clock Enable1	O	B13				gpio.38		
SDRC. NRAS	SDRAM Row Access	O	B12						
SDRC. NCAS	SDRAM Column Address Strobe	O	D14						
SDRC. NWE	SDRAM Write Enable	O	C13						
SDRC. DM0	Data Mask 0	O	B5						
SDRC. DM1	Data Mask 1	O	G12						
SDRC. DM2	Data Mask 2	O	C17						

Table 24−11. SDRC Interface (Continued)

SDRAM	Description	DIR	Ball	ALTERNATE FUNCTIONS					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
SDRC. DM3	Data Mask 3	O	D17						
SDRC. DQS0	Data Strobe 0	IO	D5						
SDRC. DQS1	Data Strobe 1	IO	G9						
SDRC. DQS2	Data Strobe 2	IO	D16						
SDRC. DQS3	Data Strobe 3	IO	B18						

Table 24−12. GP Timer Interface

Function n	Description	DIR	Ball	Alternate Functions					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
GPT9.PWM/EVT	PWM or Event for GP Timer 9	IO	V19	uart2.cts	usb1.rcv		gpio.67		
			J15	i2c2.scl			gpio.99		
GPT10.PWM/EVT	PWM or Event for GP Timer 10	IO	R19	spi2. simo			gpio.89		
			W20	uart2.rts	usb1. txen		gpio.68		
GPT11.PWM/EVT	PWM or Event for GP Timer 11	IO	R20	spi2. somi			gpio.90		
			N14	uart2.tx	usb1.se0		gpio.69		
GPT12.PWM/EVT	PWM or Event for GP Timer 12	IO	M14	spi2. ncs0			gpio.91		
			P15	uart2.rx	usb1.dat		gpio.70		

Table 24−13. UAR/IrDA/CIR Interface

Function n	Description	DIR	Ball	Alternate Functions					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
UART1.CTS	UART1 Clear To Send	I	D21			dss.d18	gpio.32		
			P13			usb1.txen	gpio.61		
UART1.RTS	UART1 Request To Send	O	H21			dss.d19	gpio.8		
			V12			usb1.rcv	gpio.25		
UART1.RX	UART1 Receive Data	I	T21			dss.d21	gpio.10		
			R13	gpio.62		usb1.dat	gpio.62		
UART1.TX	UART1 Transmit Data	O	L20			dss.d20	gpio.9		
			W12			usb1.se0	gpio.59		
UART2.CTS	UART2 Clear To Send	I	V19		usb1.rcv	gpt9.pwm/evt	gpio.67		
			E19	mmc.dat2			gpio.76		
			K19	usb0.txen	uart3.cts/rctx		gpio.110		
			J18	usb0.rcv	mcbsp2.fsx		gpio.109	uart2.cts	
UART2.RTS	UART2 Request To Send	O	W20		usb1.txen	gpt10.pwm/evt	gpio.68		
			E20	mmc.dat_dir1			gpio.78		
UART2.RX	UART2 Receive Data	I	P15		usb1.dat	gpt12.pwm/evt	gpio.70		
			E18	mmc.dat_dir3			gpio.80		
			K18	usb0.dat	uart3.rx/irrx		gpio.112	uart2.tx	
			K20	usb0.vm	mcbsp2.clkx		gpio.108		
			J14	usb0.se0	uart3.tx/irtx	uart2.tx	gpio.111		
UARdT2.TX	UART2 Transmit Data	O	N14		usb1.se0	gpt11.pwm/evt	gpio.69		
			F18	mmc.dat_dir2			gpio.79		
			J14	usb0.se0	uart3.tx/irtx		gpio.111	uart2.rx	
			K18	usb0.dat	uart3.rx/irrx	uart2.rx	gpio.112		

Table 24−13. UAR/IrDA/CIR Interface (Continued)

Function n	Description	DIR	Ball	Alternate Functions					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
UART3.CTS/RCTX	UART3 Clear To Send (input) or Irda Transciever Remote control (output)	IO	L18		uart3.rx/irrx		gpio.102		
			K19	usb0.txen		uart2.cts	gpio.110		
UART3.RCTX	UART3 Clear To Send (input) or IrDA Transciever Remote Control (output)	O	K15	uart3.tx/irtx			gpio.104		
UART3.RTS/SD	UART3 Request To Send or IrDA Tranceiver Shutdown/ Mode Select	O	L19		uart3.tx/irtx		gpio.103		
UART3.RX/IRRX	UART3 Receive Data or IrDA Receive Data	I	K14				gpio.105		
			L18	uart3.cts/ rctx			gpio.102		
			K18	usb0.dat		uart2.rx	gpio.112	uart2.tx	
UART3.TX/IRTX	UART3 Transmit Data or IrDA Transmit data	O	K15		uart3.rctx		gpio.104		
			L19	uart3.rts/sd			gpio.103		
			J14	usb0.se0		uart2.tx	gpio.111	uart2.rx	

Table 24−14. I2C Interface

Function n	Description	DIR	Ball	Alternate Functions					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
I2C1.SCL	I2C Master Serial clock. Output is open drain	IO	M19						
I2C1.SDA	I2C Serial Bidir Data. Output is open drain	IO	L15						
I2C2.SCL	I2C Master Serial clock. Output is open drain	IO	J15			gpt9.pwm/ evt	gpio.99		
I2C2.SDA	I2C Serial Bidir Data. Output is open drain	IO	H19			spi2.ncs1	gpio.100		

Table 24−15. Multi-SPI Interface

Function n	Description	DIR	Ball	Alternate Functions					
Interface				Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
SPI1.CLK	SPI Clock	IO	U18	▨	■	■	gpio.81	■	■
SPI1.SIMO	slave Data In, Master Data Out	IO	V20	▨	■	■	gpio.82	■	■
SPI1.SOMI	slave Data Out, Master Data In	IO	T18	▨	■	■	gpio.83	■	■
SPI1.nCS0	SPI Enable 0	IO	U19	▨	■	■	gpio.84	■	■
SPI1.nCS1	SPI Enable 1	O	N15	▨	■	■	gpio.85	■	■
SPI1.nCS2	SPI Enable 2	O	R18	▨	■	■	gpio.86	■	■
SPI1.nCS3	SPI Enable 3	O	U21	▨	■	■	gpio.87	■	■
SPI2.CLK	SPI Clock	IO	T19	▨	■	■	gpio.88	■	■
SPI2.SIMO	slave Data In, Master Data Out	IO	R19	▨	gpt10.pwm/evt	■	gpio.89	■	■
SPI2.SOMI	slave Data Out, Master Data In	IO	R20	▨	gpt11.pwm/evt	■	gpio.90	■	■
SPI2.nCS0	SPI Enable 0	IO	M14	▨	gpt12.pwm/evt	■	gpio.91	■	■
SPI2.nCS1	SPI Enable 1	O	H19	i2c2.sda	■	▨	gpio.100	■	■
			P20	mcbsp1.fsr			gpio.93	▨	■

Table 24−16. HDQ/1-Wire Interface

Function n	Description	DIR	Ball	Alternate Functions					
Interface				Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
HDQ.SIO	Bidir HDQ 1-Wire Control and Data interface. Output is open drain	IO	N18	▨	usb2.tllse0	sys.altclk	gpio.101	■	■

Table 24−17. USB Interface

Function n	Description	DIR	Ball	Alternate Functions					
Interface				Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
USB0.SE0	Single Ended Zero. Used as VM in 4-pin VP_VM mode.	IO	J14	▨	uart3.tx/irtx	uart2.tx	gpio.111	uart2.rx	■

Table 24−17. USB Interface (Continued)

Function n	Description	DIR	Ball	Alternate Functions					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
USB0.DAT	USB Data. Used as VP in 4−pin VP_VM mode.	IO	K18		uart3.rx/irrx	uart2.rx	gpio.112	uart2.tx	
USB0.TXEN	Transmit Enable	O	K19		uart3.cts/rctx	uart2.cts	gpio.110		
USB0.RCV	Differential Receiver Signal Input (not used in 3 pin mode)	I	J18		mcbsp2. fsx		gpio.109	uart2.cts	
USB0.VP	Vplus Receive Data (not used in 3 or 4 pin configurations)	I	J19		mcbsp2.dr		gpio.107		
USB0.VM	Vminus receive data (not used in 3 or 4 pin configurations)	I	K20		mcbsp2. clkx		gpio.108	uart2.rx	
USB0.PUEN	USB PullUp Enable	O	J20		mcbsp2.dx		gpio.106		
USB1.SE0	Single Ended Zero. Used as VM in 4−pin VP_VM mode, Also used as DP_TLL in 2 or 4 −pin TLL mode.	IO	N14	uart2.tx		gpt11.pwm/ evt	gpio.69		
			W12		uart1.tx		gpio.59		
USB1.DAT	USB Data. Used as VP in 4-Pin VP_VM Mode and as DN_TLL in 2 or 4 pin TLL.	IO	P15	uart2.rx		gpt12.pwm/ evt	gpio.70		
			R13	gpio.62	uart1.rx		gpio.62		
USB1.TXEN	Transmit Enable (not used in 2 or 3 pin TLL configurations)	O	W20	uart2.rts		gpt10.pwm/ evt	gpio.68		
			P13		uart1.cts		gpio.61		
USB1.RCV	Differential Receiver Signal Input (not used in 2 pin configuration)	I	V19	uart2.cts		gpt9.pwm/ evt	gpio.67		
			V12		uart1.rts		gpio.25		
USB2.SE0	Single Ended Zero. Used as VM in 4-Pin VP_VM Mode, Used as DP_TLL in 2 or 4 pin TLL mode.	IO	AA10			sys. ndmareq0	gpio.13		

Table 24−17. USB Interface (Continued)

Function n	Description	DIR	Ball	Alternate Functions					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
USB2.DAT	USB Data. Used as VP in 4-Pin VP_VM Mode and as DN_TLL in 2 pin TLL.	IO	Y11			sys.clkout2	gpio.16		
USB2.TXEN	Transmit Enable (not used in 2 pin TLL configuration),	IO	AA12				gpio.17		
USB2.RCV	Differential Receiver Signal Input. Also used as output for RCV_TLL/VP_TLL in 5-pin TLL mode	I	AA6			sys. ndmareq1	gpio.14	cam_d8	
USB2.TLLSE0	VM_TLL Data in 5-Pin TLL Mode (not used in 2, 3, or 4 pin configurations)	I	AA4				gpio.15	cam_d7	

Table 24−18. McBSP Interface

Function n	Description	DIR	Ball	Alternate Functions					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
McBSP.CLKS	External Clock Input (shared by both McBSP's)	I	M18				gpio.96		
			V5	cam.lclk			gpio.57		
McBSP1.DR	Received Serial Data	I	P18				gpio.95		
			U4	cam.d3	hw.dbg5		gpio.51		
McBSP1.CLKR	Receive Clock	IO	M15				gpio.92		
			V4	cam.d5	hw.dbg7		gpio.49		
McBSP1.FSR	Receive Frame Synchronization	IO	P20				gpio.93	spi2.ncs1	
			W2	cam.d4	hw.dbg6		gpio.50		

Table 24−18. McBSP Interface (Continued)

Function n	Description	DIR	Ball	Alternate Functions					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
McBSP1.DX	Transmitted Serial Data	O	P19	▨			gpio.94	■	■
			T3	cam.hs	hw.dbg1	▨	gpio.55	■	■
McBSP1.CLKX	Transmit Clock	IO	N19	▨			gpio.98	■	■
			V3	cam.d2	hw.dbg4	▨	gpio.52	■	■
McBSP1.FSX	Transmit Frame Synchronization	IO	L14	▨	ssi2.rdy_rx	■	gpio.97	■	■
			U2	cam.vs	hw.dbg0	▨	gpio.56	■	■
McBSP2.DR	Received Serial Data	I	M21	▨	■	dss.d22	gpio.11	■	■
			J19	usb0.vp	▨	■	gpio.107	■	■
			W15	eac.ac_din	▨	■	gpio.115	■	■
McBSP2.DX	TranSmitted Serial Data	O	J20	usb0.puen	▨	■	gpio.106	■	■
			V15	eac.ac_dout	▨	■	gpio.116	■	■
McBSP2.CLKX	Combined Serial Clock	IO	P21	▨	■	dss.d23	gpio.12	■	■
			K20	usb0.vm	▨	■	gpio.108	uart2.rx	■
			Y15	eac.ac_sclk	▨	■	gpio.113	■	■
McBSP2.FSX	Combined Frame Synchronization	IO	J18	usb0.rcv	▨	■	gpio.109	uart2.cts	■
			R14	eac.ac_fs	▨	■	gpio.114	■	■
			W7	dss.acbias	■	mcbsp2.fsx	gpio.48	■	■

Table 24−19. EAC Interface

Function n	Description	DIR	Ball	Alternate Functions					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
EAC.AC_MCLK	Audio Codec Master Clock Input or Output	IO	V14				gpio.117		
EAC.AC_FS	Audio Codec Interface Frame Sync	IO	R14		mcbsp2.fsx		gpio.114		
EAC.AC_SCLK	Audio Codec Interface Serial Clock	IO	Y15		mcbsp2.clkx		gpio.113		
EAC.AC_DIN	Audio Codec Interface Serial Data In	I	W15		mcbsp2.dr		gpio.115		
EAC.AC_DOUT	Audio Codec Interface Serial Data Out	O	V15		mcbsp2.dx		gpio.116		
EAC.AC_RST	Audio Codec Interface Reset Output (AC97 mode)	O	W16		eac.bt_din	loop_eac.bt_din	gpio.118		
EAC.MD_SCLK	Modem voice Interface Serial Clock	IO	Y12				gpio.63		
EAC.MD_FS	Modem Voice Interface Frame Synchro	IO	Y13				gpio.66		
EAC.MD_DIN	Modem Voice Interface Serial Data In	I	W13				gpio.64		
EAC.MD_DOUT	Modem Voice Interface Serial Data Out	O	V13				gpio.65		
EAC.BT_SCLK	Bluetooth Voice Interface Serial Clock	IO	R8				gpio.71		
EAC.BT_FS	Bluetooth Voice Interface Frame Sync	IO	P9				gpio.72		
EAC.BT_DIN	BT Voice Interface Serial Data In	I	Y3				gpio.73		
			W16	eac.ac_rst		loop_eac.bt_din	gpio.118		
EAC.BT_DOUT	BT Voice Interface Serial Data Out	O	W4				gpio.74		
LOOP_EAC.BT_DIN	Loop signal between EAC.BT_DIN pin and EAC.AC_RST pin	O	W4				gpio.74		

Special pins (sub-LVDS IO pins) are available on cam.d7, cam.d8 and cam.d9 for differential lines support.

Table 24−20. Camera Interface

Interface				Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
Function n	**Description**	**DIR**	**Ball**	**Alternate Functions**					
CAM.D2	Camera Digital Image Data Bit 2	I	V2	░	hw.dbg4	mcbsp1. clkx	gpio.52	■	■
CAM.D3	Camera Digital Image Data Bit 3	I	U4	░	hw.dbg5	mcbsp1.dr	gpio.51	■	■
CAM.D4	Camera Digital Image Data Bit 4	I	W2	░	hw.dbg6	mcbsp1. fsr	gpio.50	■	■
CAM.D5	Camera Digital Image Data Bit 5	I	V4	░	hw.dbg7	mcbsp1. clkr	gpio.49	■	■
CAM.D6	Camera Digital Image Data Bit 6	I	W3	░	hw.dbg8	■	■	■	■
			AA8		■	14.ext_trig	gpio.58	░	■
CAM.D7	Camera Digital Image Data Bit 7	I	Y2	░	hw.dbg9	■	■	■	■
			AA4		usb2.tllse0	■	gpio.15	░	■
CAM.D8	Camera Digital Image Data Bit 8	I	Y4	░	hw.dbg10	■	gpio.54	■	■
			AA6		usb2.rcv	sys. ndmareq1	gpio.14	░	■
CAM.D9	Camera Digital Image Data Bit 9	I	V6	░	hw.dbg11	■	gpio.53	■	■
			Y5	gpio.120	■	■	gpio.120	░	■
CAM.LCLK	Camera Image Data Latch clock	I	V5	░	■	mcbsp.clks	gpio.57	■	■

Table 24−21. Display Interface

Function n	Description	DIR	Ball	Alternate Functions					
Interface				Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
DSS.PCLK	LCD Pixel Clock	IO	W6						
DSS.HSYNC	LCD Horizental Synchronization	IO	Y6						
DSS.VSYNC	LCD Vertical Synchronization	IO	V7						
DSS.ACBIAS	AC Bias Control (STN) or Pixel Data Enable (TFT) Output; also A0 output (Command/Data selection) in config 1	O	W7			mcbsp2.fsx	gpio.48		
DSS.D0	LCD Pixel Data Bit 0	IO	Y7						
DSS.D1	LCD Pixel Data Bit 1	IO	P10						
DSS.D2	LCD Pixel Data Bit 2	IO	V8						
DSS.D3	LCD Pixel Data Bit 3	IO	Y8						
DSS.D4	LCD Pixel Data Bit 4	IO	W8						
DSS.D5	LCD Pixel Data Bit 5	IO	R10						
DSS.D6	LCD Pixel Data Bit 6	IO	Y9						
DSS.D7	LCD Pixel Data Bit 7	IO	V9						
DSS.D8	LCD Pixel Data Bit 8	IO		dss.d8			gpio.38		
DSS.D9	LCD Pixel Data Bit 9	IO		dss.d9			gpio.39		
DSS.D10	LCD Pixel Data Bit 10	IO		dss.d10			gpio.40		
DSS.D11	LCD Pixel Data Bit 11	IO		dss.d11			gpio.41		
DSS.D12	LCD Pixel Data Bit 12	IO		dss.d12			gpio.42		
DSS.D13	LCD Pixel Data Bit 13	IO		dss.d13			gpio.43		
DSS.D14	LCD Pixel Data Bit 14	IO		dss.d14			gpio.44		
DSS.D15	LCD Pixel Data Bit 15	IO		dss.d15			gpio.45		
DSS.D16	LCD Pixel Data Bit 16	IO		dss.d16			gpio.46		

Table 24−21. Display Interface (Continued)

Function n	Description	DIR	Ball	Alternate Functions					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
DSS.D17	LCD Pixel Data Bit 17	IO		dss.d17			gpio.47		
DSS.D18	LCD Pixel Data Bit 18	IO	J3	gpmc.a1			gpio.12		
			D21	uart1.cts			gpio.32		
DSS.D19	LCD Pixel Data Bit 19	IO	H4	gpmc.a2			gpio.11		
			H21	uart1.rts			gpio.8		
DSS.D20	LCD Pixel Data Bit 20	O	H3	gpmc.a3			gpio.10		
			L20	uart1.tx			gpio.9		
DSS.D21	LCD Pixel Data Bit 21	O	G3	gpmc.a4			gpio.9		
			T21	uart1.rx			gpio.10		
DSS.D22	LCD Pixel Data Bit 22	O	F4	gpmc.a5			gpio.8		
			M21	mcbsp2.dr			gpio.11		
DSS.D23	LCD Pixel Data Bit 23	O	F3	gpmc.a6			gpio.7		
			P21	mcbsp2. clkx			gpio.12		

Table 24−22. TV Out Interface

TV	Description	DIR	Ball	Alternate Functions					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
TV.CVBS	Composite video output (analog)	**O**	AA16	tv_cvbs					
TV.VREF	Input voltage reference (analog)	I	AA18	tv_vref					
TV.RREF	External resistor reference (analog − 4 kΩ)	I	AA14	tv_rref					
TV.DETECT	Pulse Detection	O	E5	gpio.6	tv.detect		gpio.6		

Table 24−23. MMC/SDIO Interface

Function n	Description	DIR	Ball	Alternate Functions					
Interface				Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
MMC.CLKO	MMC/SD Output Clock	IO	G19						
MMC.CLKI	MMC/SD Input Clock	I	H15				gpio.59		
MMC.CMD	MMC/SD Command Signal	IO	H18						
MMC.DAT0	MMC/SD Card Data bit 0/SPI Serial Input	IO	F20						
MMC.DAT1	MMC/SD Card Data Bit 1	IO	H14				gpio.75		
MMC.DAT2	MMC/SD Card Data Bit 2	IO	E19			uart2.cts	gpio.76		
MMC.DAT3	MMC/SD Card Data Bit 3	IO	D19			l4.ext_trig	gpio.77		
MMC.CMD_DIR	MMC/SD Command Direction	O	G18				gpio.8		
MMC.DAT_DIR0	MMC/SD Data Direction Bit 0	O	F19				gpio.7		
MMC.DAT_DIR1	MMC/SD Data Direction Bit 1	O	E20			uart2.rts	gpio.78		
MMC.DAT_DIR2	MMC/SD Data Direction Bit 2	O	F18			uart2.tx	gpio.79		
MMC.DAT_DIR3	MMC/SD data direction bit 3	O	E18			uart2.rx	gpio.80		

Table 24−24. GPIO Interface

Function n	Description	DIR	Ball	Alternate Functions					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
GPIO.0	General Purpose IO 0	IO	B3	sdrc.a14					
GPIO.1	General Purpose IO 1	IO	B4	sdrc.a13					
GPIO.2	General Purpose IO 2	IO	D11	sdrc.a12					
GPIO.3	General Purpose IO 3	IO	E3	gpmc.a10		sys. ndmareq5			
GPIO.4	General Purpose IO 4	IO	D3	gpmc.a9		sys. ndmareq4			
GPIO.5	General Purpose IO 5	IO	G4	gpmc.a8		sys. ndmareq3			
GPIO.6	General Purpose IO 6	IO	E5		tv.detect		gpio.6		
			E4	gpmc.a7		sys. ndmareq2			
			E5	gpio.6	tv.detect				
GPIO.7	General Purpose IO 7	IO	F3	gpmc.a6	dss.d23				
			F19	mmc.dat_dir0					
GPIO.8	General Purpose IO 8	IO	F4	gpmc.a5	dss.d22				
			H21	uart1.rts		dss.d19			
			G18	mmc.cmd_dir					
GPIO.9	General Purpose IO 9	IO	G3	gpmc.a4	dss.d21				
			L20	uart1.tx		dss.d20			
GPIO.10	General Purpose IO 10	IO	H3	gpmc.a3	dss.d20				
			T21	uart1.rx		dss.d21			
GPIO.11	General Purpose IO 11	IO	H4	gpmc.a2	dss.d19				
			M21	mcbsp2.dr		dss.d22			
GPIO.12	General Purpose IO 12	IO	J3	gpmc.a1	dss.d18				
			P21	mcbsp2.clkx		dss.d23			

† Input only. Nothing is multiplexed with SYS.CLKOUT output clock to avoid the risk of perturbation on clock signal.

Table 24−24. GPIO Interface (Continued)

Function n	Description	DIR	Ball	Alternate Functions					
Interface				Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
GPIO.13	General Purpose IO 13	IO	J7	gpmc.d15					
			AA10		usb2.se0	sys. ndmareq0			
GPIO.14	General Purpose IO 14	IO	J8	gpmc.d14					
			AA6		usb2.rcv	sys. ndmareq1		cam_d8	
GPIO.15	General Purpose IO 15	IO	K7	gpmc.d13					
			AA4		usb2.tllse0			cam_d7	
GPIO.16	General Purpose IO 16	IO	L8	gpmc.d12					
			Y11		usb2.dat	sys.clkout2			
GPIO.17	General Purpose IO 17	IO	K8	gpmc.d11					
			AA12		usb2.txen				
GPIO.18	General Purpose IO 18	IO	L7	gpmc.d10					
GPIO.19	General Purpose IO 19	IO	M7	gpmc.d9					
GPIO.20	General Purpose IO 20	IO	M8	gpmc.d8					
GPIO.21	General Purpose IO 21	IO	J4	gpmc.clk					
GPIO.22	General Purpose IO 22	IO	N8	gpmc.ncs1					
GPIO.23	General Purpose IO 23	IO	E2	gpmc.ncs2					
GPIO.24	General Purpose IO 24	IO	N2	gpmc.ncs3	gpmc.io_dir				
GPIO.25	General Purpose IO 25	IO	F1	gpmc.ncs4					
			V12		uart1.rts	usb1.rcv			

† Input only. Nothing is multiplexed with SYS.CLKOUT output clock to avoid the risk of perturbation on clock signal.

Table 24−24. GPIO Interface (Continued)

Function n	Description	DIR	Ball	Alternate Functions					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
GPIO.26	General Purpose IO 26	IO	H1	gpmc.ncs5					
GPIO.27	General Purpose IO 27	IO	K1	gpmc.ncs6					
GPIO.28	General Purpose IO 28	IO	L2	gpmc.ncs7	gpmc.io_dir				
GPIO.29	General Purpose IO 29	O	P7	gpmc.nbe0					
GPIO.30	General Purpose IO 30	O	R1	gpmc.nbe1					
GPIO.31	General Purpose IO 31	O	G2	gpmc.nwp					
GPIO.32	General Purpose IO 32	IO	D21	uart1.cts		dss.d18			
GPIO.33	General Purpose IO 33	IO	N7	gpmc.wait1					
GPIO.34	General Purpose IO 34	IO	M1	gpmc.wait2					
GPIO.35	General Purpose IO 35	IO	P1	gpmc.wait3					

† Input only. Nothing is multiplexed with SYS.CLKOUT output clock to avoid the risk of perturbation on clock signal.

Table 24−24. GPIO Interface (Continued)

Function n	**Description**	**DIR**	**Ball**	**Alternate Functions**					
Interface				**Mode0**	**Mode1**	**Mode2**	**Mode3**	**Mode4**	**Mode5**
GPIO.36	General Purpose IO 36	IO	V17				gpio.36		sys.boot4
			V17	gpio.36					sys.boot4
GPIO.37	General Purpose IO 37	IO	C12	sdrc.ncs1					
GPIO.38	General Purpose IO 38	IO	B13	sdrc.cke1					
			W9	dss.d8					
GPIO.39	General Purpose IO 39	IO	P11	dss.d9					
GPIO.40	General Purpose IO 40	IO	V10	dss.d10					
GPIO.41	General Purpose IO 41	IO	Y10	dss.d11					
GPIO.42	General Purpose IO 42	IO	W10	dss.d12					
GPIO.43	General Purpose IO 43	IO	R11	dss.d13					
GPIO.44	General Purpose IO 44	IO	V11	dss.d14					
GPIO.45	General Purpose IO 45	IO	W11	dss.d15					
GPIO.46	General Purpose IO 46	IO	P12	dss.d16					
GPIO.47	General Purpose IO 47	IO	R12	dss.d17					
GPIO.48	General Purpose IO 48	IO	W7	dss.acbias		mcbsp2.fsx			
GPIO.49	General Purpose IO 49	IO	V4	cam.d5	hw.dbg7	mcbsp1.clkr			
GPIO.50	General Purpose IO 50	IO	W2	cam.d4	hw.dbg6	mcbsp1.fsr			
GPIO.51	General Purpose IO 51	IO	U4	cam.d3	hw.dbg5	mcbsp1.dr			

† Input only. Nothing is multiplexed with SYS.CLKOUT output clock to avoid the risk of perturbation on clock signal.

Table 24−24. GPIO Interface (Continued)

Function n	Description	DIR	Ball	Alternate Functions					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
GPIO.52	General Purpose IO 52	IO	V3	cam.d2	hw.dbg4	mcbsp1.clkx			
		I†	AA17	sys.clkreq					
GPIO.53	General Purpose IO 53	IO	V6	cam.d9	hw.dbg11				
			V2	cam.d1	hw.dbg3				
GPIO.54	General Purpose IO 54	IO	Y4	cam.d8	hw.dbg10				
			T4	cam.d0	hw.dbg2				
GPIO.55	General Purpose IO 55	IO	T3	cam.hs	hw.dbg1	mcbsp1.dx			
GPIO.56	General Purpose IO 56	IO	U2	cam.vs	hw.dbg0	mcbsp1.fsx			
GPIO.57	General Purpose IO 57	IO	V5	cam.lclk		mcbsp.clks			
GPIO.58	General Purpose IO 58	IO	AA8			l4_ext_trig		cam_d6	
GPIO.59	General Purpose IO 59	IO	W12		uart1.tx	usb1.se0			
			H15	mmc.clki					
GPIO.60	General Purpose IO 60	I	W19	sys_nirq					
GPIO.61	General Purpose IO 61	IO	P13		uart1.cts	usb1.txen			

† Input only. Nothing is multiplexed with SYS.CLKOUT output clock to avoid the risk of perturbation on clock signal.

Table 24−24. GPIO Interface (Continued)

Function n	Description	DIR	Ball	Alternate Functions					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
GPIO.62	General Purpose IO 62	IO	R13		uart1.rx	usb1.dat	gpio.62		
			R13	gpio.62	uart1.rx	usb1.dat			
GPIO.63	General Purpose IO 63	IO	Y12		eac.md_ sclk				
GPIO.64	General Purpose IO 64	IO	W13		eac.md_ din				
GPIO.65	General Purpose IO 65	IO	V13		eac.md_ dout				
GPIO.66	General Purpose IO 66	IO	Y13		eac.md_fs				
GPIO.67	General Purpose IO 67	IO	V19	uart2.cts	usb1.rcv	gpt9.pwm/evt			
GPIO.68	General Purpose IO 68	IO	W20	uart2.rts	usb1.txen	gpt10.pwm/ evt			
GPIO.69	General Purpose IO 69	IO	N14	uart2.tx	usb1.se0	gpt11.pwm/ evt			
GPIO.70	General Purpose IO 70	IO	P15	uart2.rx	usb1.dat	gpt12.pwm/ evt			
GPIO.71	General Purpose IO 71	IO	R8	eac.bt_ sclk					
GPIO.72	General Purpose IO 72	IO	P9	eac.bt_fs					
GPIO.73	General Purpose IO 73	IO	Y3	eac.bt_din					
GPIO.74	General Purpose IO 74	IO	W4	eac.bt_ dout					
GPIO.75	General Purpose IO 75	IO	H14	mmc.dat1					
GPIO.76	General Purpose IO 76	IO	E19	mmc.dat2		uart2.cts			
GPIO.77	General Purpose IO 77	IO	D19	mmc.dat3		l4.ext_trig			

† Input only. Nothing is multiplexed with SYS.CLKOUT output clock to avoid the risk of perturbation on clock signal.

Table 24−24. GPIO Interface (Continued)

Function n	Description	DIR	Ball	Alternate Functions					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
GPIO.78	General Purpose IO 78	IO	E20	mmc.dat_dir1		uart2.rts			
GPIO.79	General Purpose IO 79	IO	F18	mmc.dat_dir2		uart2.tx			
GPIO.80	General Purpose IO 80	IO	E18	mmc.dat_dir3		uart2.rx			
GPIO.81	General Purpose IO 81	IO	U18	spi1.clk					
GPIO.82	General Purpose IO 82	IO	V20	spi1.simo					
GPIO.83	General Purpose IO 83	IO	T18	spi1.somi					
GPIO.84	General Purpose IO 84	IO	U19	spi1.ncs0					
GPIO.85	General Purpose IO 85	IO	N15	spi1.ncs1					
GPIO.86	General Purpose IO 86	IO	R18	spi1.ncs2					
GPIO.87	General Purpose IO 87	IO	U21	spi1.ncs3					
GPIO.88	General Purpose IO 88	IO	T19	spi2.clk					
GPIO.89	General Purpose IO 89	IO	R19	spi2.simo	gpt10.pwm/evt				
GPIO.90	General Purpose IO 90	IO	R20	spi2.somi	gpt11.pwm/evt				
GPIO.91	General Purpose IO 91	IO	M14	spi2.ncs0	gpt12.pwm/evt				
GPIO.92	General Purpose IO 92	IO	M15	mcbsp1.clkr					
GPIO.93	General Purpose IO 93	IO	P20	mcbsp1.fsr				spi2.ncs1	
GPIO.94	General Purpose IO 94	IO	P19	mcbsp1.dx					

† Input only. Nothing is multiplexed with SYS.CLKOUT output clock to avoid the risk of perturbation on clock signal.

Table 24−24. GPIO Interface (Continued)

Function n	Description	DIR	Ball	Alternate Functions					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
GPIO.95	General Purpose IO 95	IO	P18	mcbsp1.dr					
GPIO.96	General Purpose IO 96	IO	M18	mcbsp.clks					
GPIO.97	General Purpose IO 97	IO	L14	mcbsp1.fsx					
GPIO.98	General Purpose IO 98	IO	N19	mcbsp1.clkx					
GPIO.99	General Purpose IO 99	IO	J15	i2c2.scl		gpt9.pwm/evt			
GPIO.100	General Purpose IO 100	IO	H19	i2c2.sda		spi2.ncs1			
GPIO.101	General Purpose IO 101	IO	N18	hdq.sio	usb2.tllse0	sys.altclk			
GPIO.102	General Purpose IO 102	IO	L18	uart3.cts/rctx	uart3.rx/irrx				
GPIO.103	General Purpose IO 103	IO	L19	uart3.rts/sd	uart3.tx/irtx				
GPIO.104	General Purpose IO 104	IO	K15	uart3.tx/irtx	uart3.rctx				
GPIO.105	General Purpose IO 105	IO	K14	uart3.rx/irrx					
GPIO.106	General Purpose IO 106	IO	J20	usb0.puen	mcbsp2.dx				
GPIO.107	General Purpose IO 107	IO	J19	usb0.vp	mcbsp2.dr				
GPIO.108	General Purpose IO 108	IO	K20	usb0.vm	mcbsp2.clkx			uart2.rx	
GPIO.109	General Purpose IO 109	IO	J18	usb0.rcv	mcbsp2.fsx			uart2.cts	
GPIO.110	General Purpose IO 110	IO	K19	usb0.txen	uart3.cts/rctx	uart2.cts			
GPIO.111	General Purpose IO 111	IO	J14	usb0.se0	uart3.tx/irtx	uart2.tx		uart2.rx	

---

† Input only. Nothing is multiplexed with SYS.CLKOUT output clock to avoid the risk of perturbation on clock signal.

Table 24−24. GPIO Interface (Continued)

Function n	Description	DIR	Ball	Alternate Functions					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
GPIO.112	General Purpose IO 112	IO	K18	usb0.dat	uart3.rx/irrx	uart2.rx		uart2.tx	
GPIO.113	General Purpose IO 113	IO	Y15	eac.ac_ sclk	mcbsp2. clkx				
GPIO.114	General Purpose IO 114	IO	R14	eac.ac_fs	mcbsp2. fsx				
GPIO.115	General Purpose IO 115	IO	W15	eac.ac_din	mcbsp2.dr				
GPIO.116	General Purpose IO 116	IO	V15	eac.ac_ dout	mcbsp2.dx				
GPIO.117	General Purpose IO 117	IO	V14	eac.ac_ mclk					
GPIO.118	General Purpose IO 118	IO	W16	eac.ac_rst	eac.bt_din	loop_eac. bt_din			
GPIO.119	General Purpose IO 119	IO	W5				gpio.119		sys.boot0
			W5	gpio.119					sys.boot0
GPIO.120	General Purpose IO 120	IO	Y5				gpio.120	cam.d9	sys.boot1
			Y5	gpio.120				cam.d9	sys.boot1
GPIO.121	General Purpose IO 121	IO	R9				gpio.121	jtag.emu2	sys.boot2
			R9	gpio.121				jtag.emu2	sys.boot2
GPIO.122	General Purpose IO 122	IO	P8				gpio.122	jtag.emu3	sys.boot3
			P8	gpio.122				jtag.emu3	sys.boot3
GPIO.123	General Purpose IO 123	I[†]	W14	sys.clkout					
GPIO.124	General Purpose IO 124	IO	V18				gpio.124		sys.boot5
			V18	gpio.124					sys.boot5
GPIO.125	General Purpose IO 125	IO	P14		sys. jtagsel1	sys. jtagsel2	gpio.125		
			P14	gpio.125	sys. jtagsel1	sys. jtagsel2			
GPIO.126	General Purpose IO 126	IO	AA21	jtag.emu1					
GPIO.127	General Purpose IO 127	IO	Y21	jtag.emu					

[†] Input only. Nothing is multiplexed with SYS.CLKOUT output clock to avoid the risk of perturbation on clock signal.

Table 24−25. System Interface

Function n	Description	DIR	Ball	Alternate Functions					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
SYS.32K	32 kHz Clock Input	I	Y17						
SYS.XTALIN	Main Input Clock. Oscillator input or LVCMOS at 19.2, 13, or 12 Mhz.	IO	Y18						
SYS.XTALOUT	Output of Oscillator	O	V16						
SYS.ALTCLK	Alternate Clock Source Selectable for GPTIMERs (max 54 MHz), USB (48 MHz) or NTSC/PAL (54 MHz).	I	N18	hdq.sio	usb2.tllse0		gpio.101		
SYS.CLKREQ	Request From OMAP24xx Device for system Clock	O	AA17				gpio.52		
SYS.CLKOUT	Configurable Output Clock	O	W14				gpio.123		
SYS.CLKOUT2	Configurable Output Clock 2	O	Y11		usb2.dat		gpio.16		
SYS.BOOT0	Boot Configuration Mode Bit 0	I	W5	gpio.119			gpio.119		
SYS.BOOT1	Boot Configuration Mode Bit 1	I	Y5	gpio.120			gpio.120	cam.d9	
SYS.BOOT2	Boot Configuration Mode Bit 2	I	R9	gpio.121			gpio.121	jtag.emu2	
SYS.BOOT3	Boot Configuration Mode Bit 3	I	P8	gpio.122			gpio.122	jtag.emu3	
SYS.BOOT4	Boot Configuration Mode Bit 4	I	V17	gpio.36			gpio.36		
SYS.BOOT5	Boot Configuration Mode Bit 5	I	V18	gpio.124			gpio.124		
SYS.nRESPWRON	Power On Reset	I	Y14						
SYS.nRESWARM	Warm Boot Reset (open drain output)	IO	Y16						

Table 24−25. System Interface (Continued)

Function n	Description	DIR	Ball	Alternate Functions					
	Interface			Mode0	Mode1	Mode2	Mode3	Mode4	Mode5
SYS.nĪRQ	Battery Voltage Failure Detection/External FIQ Input	I	W19				gpio.60		
SYS.nVMODE	Indicates the Voltage Mode	O	Y20						
SYS.JTAGSEL1	Enables Alternate Pin Mapping for JTAG Port	I	P14	gpio.125	sys. jtagsel1	sys. jtagsel2	gpio.125		
SYS.JTAGSEL2	Enables Alternate Pin Mapping for JTAG port	I	P14	gpio.125	sys. jtagsel1		gpio.125		
SYS.nDMAREQ0	Ext DMA Request	I	AA10		usb2.se0		gpio.13		
SYS.nDMAREQ1	Ext DMA Request	I	AA6		usb2.rcv		gpio.14	cam.d8	
SYS.nDMAREQ2	Ext DMA Request	I	E4	gpmc.a7			gpio.6		
SYS.nDMAREQ3	Ext DMA Request	I	G4	gpmc.a8			gpio.5		
SYS.nDMAREQ4	Ext DMA Request	I	D3	gpmc.a9			gpio.4		
SYS.nDMAREQ5	Ext DMA Request	I	E3	gpmc.a10			gpio.3		

24.6 Programming Model

24.6.1 Pinout Initialization Sequence

At release of the SYS.nRESPWRON, the internall GLOBAL_PWRON reset connected to pinout control module is released and pins take the default configuration. Pins takes default configuration as described in pin characteristics section with respect to value driven on SYS.BOOT(3) pin.

1) Default mode is set on each PAD.
2) Pinout configuration slightly modified by boot code.
3) Pinout ready to be configured for application purposes.
4) Select required interfaces with pinout functional interfaces table
5) Select pin mapping using the summary table to solve conflicts. For solving conflicts, see Section 24.6.2.
6) Program each pinout configuration register using the pin characteristics table.

24.6.2 Solving Conflicts on Muxed PADs

Some functional interfaces can be mapped on different pins; this section provides a methodology for choosing the optimized configuration.

1) Identify all required interfaces that are mapped on only one pin. You can identify them easily: only one ball is proposed in pinout functional interface table for this interface.
2) For each of these pins confirm in the alternate function list that there is no required function.
    a) There is no required function in the alternate function list, you can select this pin.
    b) There is an alternate function which is also required on this pin:
        i) This alternate function can be found on another pin; select the pin and the alternate pin for the alternate function
        ii) There is no available alternative pin for the alternate function: No solution, this configuration is not valid; choose between the function and the alternate function. There is no way to use both functions at the same time on the OMAP2420 device.

3) Apply now the same rule on the interfaces which are multiplexed at several places. Three different cases could occur:

   a) There is only one solution considering all multiplexed pins; one is already used by other selected interface. No conflict, use this solution.

   b) There are several solutions (function can be mapped on several pins): Choose the solution for which MUXMODE value of pin configuration register is the lower value.

   c) No solution left for this function (all the pins for which the function is available are already used): OMAP2420 doesn't support the chosen configuration.

24.7 Pinout Control Module Registers

Table 24−26. Instance Summary

Module Name	Base Address	Size
Pinout_control	0x4800 0000	1K bytes

24.7.1 Pinout Control Register Mapping Summary

Table 24−27. Pinout Register Summary

Register Name	Type	Register Width (Bits)	Physical Address
CONTROL_PADCONF_SDRC_A14	RW	32	0x4800 0030
CONTROL_PADCONF_SDRC_BA0	RW	32	0x4800 0034
CONTROL_PADCONF_SDRC_A8	RW	32	0x4800 0038
CONTROL_PADCONF_SDRC_A4	RW	32	0x4800 003C
CONTROL_PADCONF_SDRC_A0	RW	32	0x4800 0040
CONTROL_PADCONF_SDRC_STK_D28†	RW	32	0x4800 0044
CONTROL_PADCONF_SDRC_STK_D24†	RW	32	0x4800 0048
CONTROL_PADCONF_SDRC_STK_D24†	RW	32	0x4800 004C
CONTROL_PADCONF_SDRC_STK_D16	RW	32	0x4800 0050
CONTROL_PADCONF_SDRC_D28	RW	32	0x4800 0054
CONTROL_PADCONF_SDRC_D24	RW	32	0x4800 0058
CONTROL_PADCONF_SDRC_D20	RW	32	0x4800 005C
CONTROL_PADCONF_SDRC_D16	RW	32	0x4800 0060
CONTROL_PADCONF_SDRC_D12	RW	32	0x4800 0064
CONTROL_PADCONF_SDRC_D8	RW	32	0x4800 0068
CONTROL_PADCONF_SDRC_D4	RW	32	0x4800 006C
CONTROL_PADCONF_SDRC_D0	RW	32	0x4800 0070
CONTROL_PADCONF_GPMC_A7	RW	32	0x4800 0074
CONTROL_PADCONF_GPMC_A3	RW	32	0x4800 0078
CONTROL_PADCONF_GPMC_D14	RW	32	0x4800 007C
CONTROL_PADCONF_GPMC_D10	RW	32	0x4800 0080
CONTROL_PADCONF_GPMC_D6	RW	32	0x4800 0084
CONTROL_PADCONF_GPMC_D2	RW	32	0x4800 0088
CONTROL_PADCONF_GPMC_NCS0	RW	32	0x4800 008C
CONTROL_PADCONF_GPMC_NCS4	RW	32	0x4800 0090

Table 24−27. Pinout Register Summary (Continued)

Register Name	Type	Register Width (Bits)	Physical Address
CONTROL_PADCONF_GPMC_NADV_ALE	RW	32	0x4800 0094
CONTROL_PADCONF_GPMC_NBE1	RW	32	0x4800 0098
CONTROL_PADCONF_GPMC_WAIT2	RW	32	0x4800 009C
CONTROL_PADCONF_SDRC_NCS0	RW	32	0x4800 00A0
CONTROL_PADCONF_SDRC_NRAS	RW	32	0x4800 00A4
CONTROL_PADCONF_SDRC_DM1	RW	32	0x4800 00A8
CONTROL_PADCONF_SDRC_STK_DM1	RW	32	0x4800 00AC
CONTROL_PADCONF_SDRC_DQS1	RW	32	0x4800 00B0
CONTROL_PADCONF_DSS_D1	RW	32	0x4800 00B4
CONTROL_PADCONF_DSS_D5	RW	32	0x4800 00B8
CONTROL_PADCONF_DSS_D9	RW	32	0x4800 00BC
CONTROL_PADCONF_DSS_D13	RW	32	0x4800 00C0
CONTROL_PADCONF_DSS_D17	RW	32	0x4800 00C4
CONTROL_PADCONF_UART1_RX	RW	32	0x4800 00C8
CONTROL_PADCONF_DSS_VSYNC	RW	32	0x4800 00CC
CONTROL_PADCONF_CAM_D8	RW	32	0x4800 00D0
CONTROL_PADCONF_CAM_D4	RW	32	0x4800 00D4
CONTROL_PADCONF_CAM_D0	RW	32	0x4800 00D8
CONTROL_PADCONF_CAM_XCLK	RW	32	0x4800 00DC
CONTROL_PADCONF_GPIO_62	RW	32	0x4800 00E0
CONTROL_PADCONF_UART2_RTS	RW	32	0x4800 00EC
CONTROL_PADCONF_EAC_BT_FS	RW	32	0x4800 00F0
CONTROL_PADCONF_MMC_CMD	RW	32	0x4800 00F4
CONTROL_PADCONF_MMC_DAT3	RW	32	0x4800 00F8
CONTROL_PADCONF_MMC_DAT_DIR3	RW	32	0x4800 00FC
CONTROL_PADCONF_SPI1_SIMO	RW	32	0x4800 0100
CONTROL_PADCONF_SPI1_NCS2	RW	32	0x4800 0104
CONTROL_PADCONF_SPI2_SOMI	RW	32	0x4800 0108
CONTROL_PADCONF_MCBSP1_DX	RW	32	0x4800 010C
CONTROL_PADCONF_MCBSP1_CLKX	RW	32	0x4800 0110
CONTROL_PADCONF_I2C2_SDA	RW	32	0x4800 0114

Table 24−27. Pinout Register Summary (Continued)

Register Name	Type	Register Width (Bits)	Physical Address
CONTROL_PADCONF_UART3_TX_IRTX	RW	32	0x4800 0118
CONTROL_PADCONF_TV_RREF	RW	32	0x4800 011C
CONTROL_PADCONF_USB0_RCV	RW	32	0x4800 0120
CONTROL_PADCONF_EAC_AC_SCLK	RW	32	0x4800 0124
CONTROL_PADCONF_EAC_AC_MCLK	RW	32	0x4800 0128
CONTROL_PADCONF_SYS_NIRQ	RW	32	0x4800 012C
CONTROL_PADCONF_GPIO_121	RW	32	0x4800 0130
CONTROL_PADCONF_SYS_XTALOUT	RW	32	0x4800 0134
CONTROL_PADCONF_GPIO_6	RW	32	0x4800 0138
CONTROL_PADCONF_JTAG_EMU0	RW	32	0x4800 013C
CONTROL_PADCONF_JTAG_TMS	RW	32	0x4800 0140

† These registers are only used in the stacked package version of the OMAP2420 device.

24.7.2 Register Description

All registers are strictly identical; only the offset address changes from one register to one another, so the register structure is described only once.

Table 24−28. CONTROL_PADCONF_X Register

Address Offset	0x030 to 0x140 (see Section 24.7.1)
Physical Address	0x4800 0030 to 0x4800 0140 (see Section 24.7.1)
Description	Configuration register for pad x − 8 bit registers
Type	RW
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_3			PULLTYPESELECT_3	PULLUDENABLE_3	MUXMODE_3			RESERVED_2			PULLTYPESELECT_2	PULLUDENABLE_2	MUXMODE_2			RESERVED_1			PULLTYPESELECT_1	PULLUDENABLE_1	MUXMODE_1			RESERVED_0			PULLTYPESELECT_0	PULLUDENABLE_0	MUXMODE_0		

Bits	Field Name	Description	Type	Reset
31:29	RESERVED_3	Reserved for future use	R	*
28	PULLTYPE SELECT_3	Pullup/down selection for pad_x 0x0: Pulldown selected 0x1: Pullup selected	RW	*
27	PULLUDENABLE_3	Pullup/down enable for pad_x 0x0: Pullup/down disabled 0x1: Pullup/down enabled	RW	*
26:24	MUXMODE_3	Functional multiplexing selection for pad_x	RW	*
23:21	RESERVED_2	Reserved for future use	R	*
20	PULLTYPE SELECT_2	Pullup/down selection for pad_x 0x0: Pulldown selected 0x1: Pullup selected	RW	*
19	PULLUDENABLE_2	Pullup/down enable for pad_x 0x0: Pullup/down disabled 0x1: Pullup/down enabled	RW	*

Table 24−28. CONTROL_PADCONF_X Register (Continued)

Bits	Field Name	Description	Type	Reset
18:16	MUXMODE_2	Functional multiplexing selection for pad_x	RW	*
15:13	RESERVED_1	Reserved for future use	R	*
12	PULLTYPE SELECT_1	Pullup/down selection for pad_x 0x0: Pulldown selected 0x1: Pullup selected	RW	*
11	PULLUDENABLE_1	Pullup/down enable for pad_x 0x0: Pullup/down disabled 0x1: Pullup/down enabled	RW	*
10:8	MUXMODE_1	Functional multiplexing selection for pad_x	RW	*
7:5	RESERVED_0	Reserved for future use	R	*
4	PULLTYPE SELECT_0	Pullup/down selection for pad_x 0x0: Pulldown selected 0x1: Pullup selected	RW	*
3	PULLUDENABLE_0	Pullup/down enable for pad_x 0x0: Pullup/down disabled 0x1: Pullup/down enabled	RW	*
2:0	MUXMODE_0	Functional multiplexing selection for pad_x	RW	*

☆ Please see table 29−4 for the detailed reset value of each bit field for each register.

24.8 Acronyms Abbreviations and Definitions

Pinout Control Register	32-bit register which contains four different PIN control register fields. Name of the register is the name of PIN control register field stored in the least significant byte.
PIN Control Register Field	This field control mode selection and pull selection for one PIN.
Pure Input	A pure input PIN can be muxed on module inputs only.
Pure Output	A pure output PIN can be muxed between module outputs only.
Muxed PIN	A PIN is muxed when its PIN control register field can be reconfigured by software to change the function associated with the PIN.
Dedicated PIN	A dedicated PIN is hardwired to one function. Whatever the value of PIN control register field is, the PIN is always associated to the same function.
Primary Mode	Primary mode is mode0. The function muxed in mode 0 gives its name to the PIN
Default Mode	The default mode is the mode selected at the release of the power-on reset.
In Mux Logic	Combinational logic implied in input signals functional multiplexing
Out Mux Logic	Combinational logic used in output signal functional multiplexing.
MuxMode	3-bit field of the PIN control register field that enables to change the mode. Mode programming is assumed by software and selects a function on the OMAP2420 external interface.

Chapter 25

Device Booting

This chapter describes the high-level booting concepts on the OMAP2420 multimedia device and provides basic knowledge of booting on the device.

Topic **Page**

25.1 Booting Types

There are four types of booting on OMAP2420 devices: peripheral booting, memory booting, context restore, and overlay booting. Each is controlled by SYS.BOOT pins and RM_RSTST_MPU registers (see Table 25−1).

In peripheral booting, the ROM code polls selected interfaces and it downloads and executes software in RAM. It is used for but not limited to external memory programming (preflashing). When preflashing is used, the downloaded software is a flash loader that burns a new client application image into external flash memory. After the image is burned, a software reset starts.

In memory booting, ROM code finds the bootstrap in external flash memory, authenticates it, and executes it.

In context restore, after a wake-up reset from off mode, the ROM code restores the SDRC registers if necessary, executes the client application from SDRAM, and restores the context in secure RAM.

In overlay booting, the device executes client applications from external flash memory (see Table 25−1).

Table 25−1. Booting Types

Booting Modes	**SYS.BOOT**			**RM_RSTST_MPU**			
	5	**4**	**3**	**3**	**2**	**1**	**0**
Peripheral booting	X	1	1	X	X	0	1
Memory booting	X	0	1	X	X	0	1
	X	X	1	X	X	1	X
Context restore	X	X	1	0	1	0	0
	X	X	1	1	1	0	0
Overlay booting	X	X	0	X	X	X	X

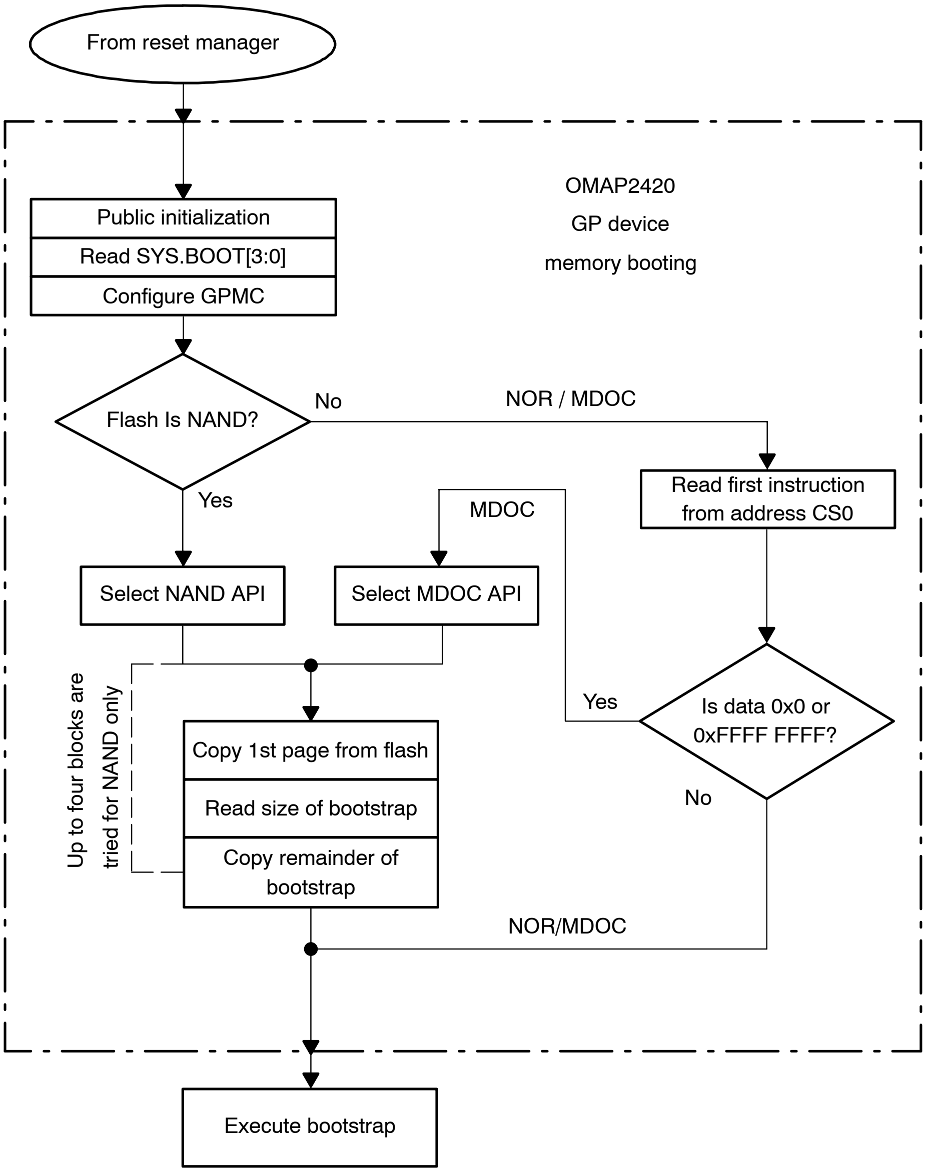
Peripheral booting, memory booting, and context restore mean booting starts from internal ROM; overlay booting starts from external flash memory.

25.2 Memory Booting

25.2.1 Memory Booting on General-Purpose Devices

Figure 25−1 shows memory booting on general-purpose devices.

Figure 25−1. Memory Booting on General-Purpose Devices



25.2.1.1 From NOR

The bootstrap starts with the entry point at address CS0 + 0x0000. The ROM code tests whether the content of address CS0 + 0x0000 is 0x0 or 0xFFFF FFFF. If it is neither of these two values, ROM code directly jumps to CS0 + 0x0000. If it is one of these, ROM code boots using the Memory Disk On Chip (MDOC), by M-Systems, booting scenario.

25.2.1.2 From NAND

The bootstrap starts with a 4-byte length indicator, followed by a 4-byte destination load address. After it is copied to the RAM, a jump is made to that address, and the bootstrap is executed.

25.2.1.3 From MDOC

If the SYS.BOOT[3:0] pins are configured for NOR and wait monitoring is active, the external flash memory can also be an MDOC.

If the bootstrap image fits in the NOR part of the MDOC, the NOR booting scenario is performed. Otherwise, the NAND scenario is attempted using the MDOC driver.

If any MDOC API functions return an error, the ROM code initiates a software reset.

Chapter 26

Initialization

This chapter provides an overview of the requirements for initializing an OMAP2420 device from power-on to OS load and applications running. An overview of the overall initialization process is given, with both hardware- and software-related steps, a general overview of the boot ROM operational requirements, and behavior expectations.

This chapter does not give a detailed review of the OMAP boot ROM functionality. For this information, see Chapter 25, *Device Booting*.

Topic **Page**

26.1 Introduction

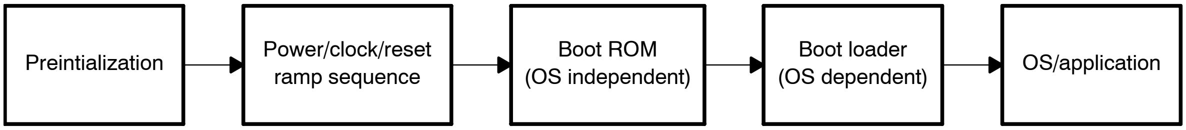
This chapter provides an overview of the requirements for initializing an OMAP2420 device from the power-on to OS load and applications running.

This chapter does not give a detailed review of OMAP boot ROM functionality; for this information, see Chapter 25, *Device Booting*.

26.1.1 Overview of OMAP2420 Initialization Process

Figure 26−1 is an overview of the initialization process. Initialization of an OMAP2420 device consists of several steps: pre-initialization; ramp sequence for power, clocks, and resets; boot ROM; boot loader (bootstrap); and OS start. Each step, up to OS/applications running, is explained in detail in the following sections.

Figure 26−1. Initialization Process for OMAP2420 Devices


Preintialization
Power/clock/reset ramp sequence
Boot ROM (OS independent)
Boot loader (OS dependent)
OS/application

While the first two steps in the initialization process are hardware oriented, they require a good understanding of the process of configuring system interface pads (pins or balls on the device) that have software-configurable functionality. Pad configuration is an essential part of chip configuration and is application-dependent; this chapter details those pads and the associated control registers that are vital for correct device initialization. For detailed information about the use of the pad configuration registers, see Chapter 8, *System Control Module*.

26.2 Preinitialization Requirements

Certain hardware configuration settings must be made to accomplish a successful boot-up operation with an OMAP2420 device. Clock, reset, and power connections, as well as pads involved in setting the boot memory space for the microprocessor unit (MPU), must be connected and driven correctly for successful device initialization. This section details the specific requirements for the system interface ports.

26.2.1 Power Connections

Table 26−1 lists the voltage requirements for the power resources.

Table 26−1. OMAP2420 Voltage Levels

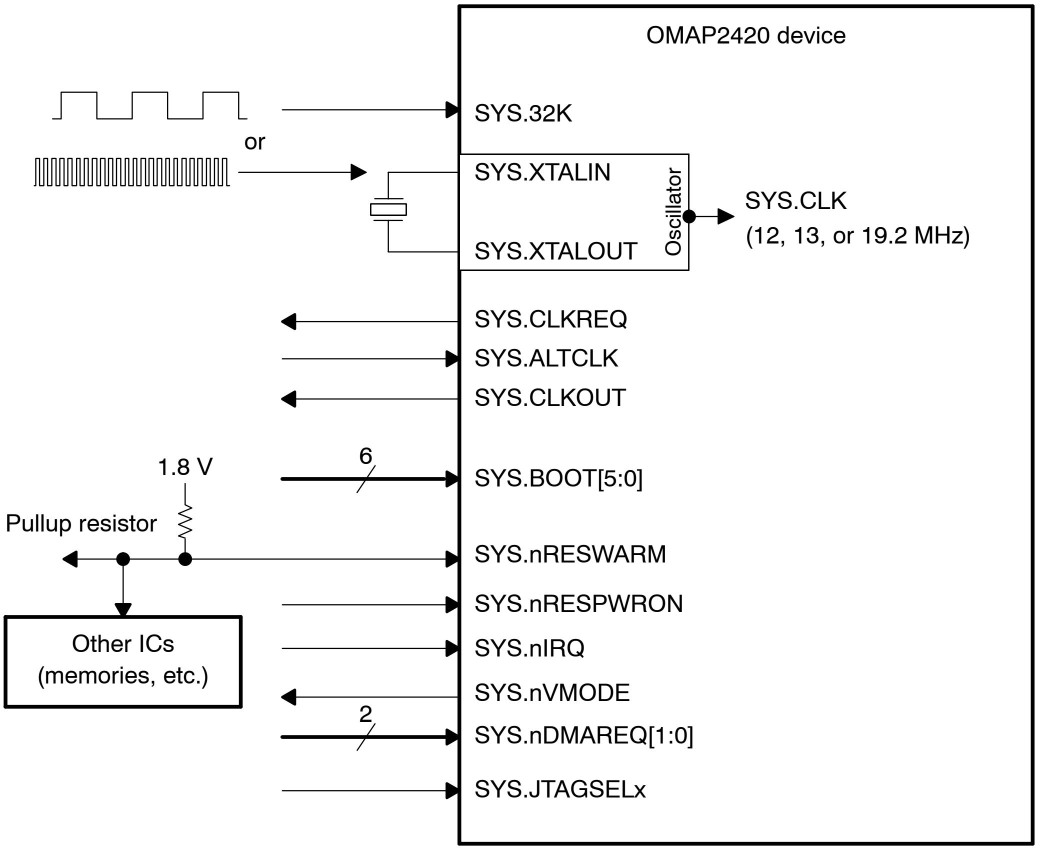
Voltage	Description	Operating Levels (Volts)			Absolute Minimum-Maximum (Volts)	
		Min	Nom	Max	Min	Max
VDD	OMAP processor cores and logic operating at full speed	1.235	High voltage	1.372	−0.5	1.7
	OMAP processor cores and logic operating in low-voltage functional mode	0.99	Low voltage	1.11		
VDDS VDDS2	Supply voltage for input/output (I/O) macros (except GPMC, SDRC)	1.71	1.8	1.89	−0.5	2.43
VDDS1	Supply voltage for I/O-to-memory interfaces (GPMC, SDRC)	1.71	1.8	1.89	−0.5	2.43
VDDA	Analog supply voltage for video DAC	1.71	1.8	1.89	−0.5	2.43
VDD_PLL	Supply voltage for APLL and DPLLs	1.235	High voltage	1.372	−0.5	1.7
VDD_DLL[1]	Supply voltage for SDRC DLL core operating at full speed	1.235	High voltage	1.372	−0.5	1.7
	Supply voltage for SDRC DLL core operating in low-voltage functional mode	0.99	Low voltage	1.11		
VSS	Common ground	0				
VSSA	Analog ground for video DAC	0				

1) VDD_DLL voltage must match VDD voltage.

26.2.2 System Interface

Figure 26−2 shows the OMAP2420 device system interface, which gathers all signals that interact with system control: clocks, resets, and power, as well as system expansion signals such as extra IRQ and DMAREQ signals. Figure 26−2 shows the system interface with the different options for the system clock: oscillator or digital clock input.

Figure 26−2. OMAP2420 System Interface



The system interface has the following main features:

- 12-, 13-, or 19.2-MHz reference clock from either the crystal oscillator or digital clock input
- Input 32-kHz CMOS clock
- Configurable output clock
- Six input signals to define the boot mode
- Two reset sources
  - Power-on reset (cold reset)
  - Bidirectional warm reset
- External interrupt input
- Output for voltage control (in support of dynamic voltage scaling for lower operational power)
- Two external direct memory access (DMA) requests for system expansion

26.2.2.1 System Interface Signal Descriptions

Table 26−2 lists the I/O pins and describes the system interface signals.

Table 26−2. System Interface Signals

Pin	Type[1,2]	Description	Alternate Function
SYS.32K	I	32-kHz clock input. Must be valid for two clocks before the rising edge of SYS.nRESPWRON.	None
SYS.XTALIN	I	Input of the oscillator or digital clock (I/O voltage level)	None
SYS.XTALOUT	O	Output of the oscillator	None
SYS.ALTCLK	I	Alternate clock source for GP timers (max 54 MHz), USB (48 MHz), or NTSC/PAL (54 MHz) modules	See pinout.
SYS.CLKREQ	O	Active high request for system clock	See pinout.
SYS.CLKOUT	O	Configurable output clock	See pinout.
SYS.BOOT[5:0]	I	Boot mode inputs. Sampled and latched at power-on reset and can be reused for other functions. Remark: SYSBOOT[3] pin value is a don’t care for high-security (HS) and general-purpose (GP) devices.	See pinout.
SYS.nRESPWRON	I	Active-low signal indicating that voltages are correctly set. It resets all the logic, analog, and I/Os. If the crystal oscillator is used as the system clock, this signal must be held low for a sufficient time to allow the crystal oscillator to start.	None
SYS.nRESWARM	I/OD	Active-low warm reset generated internally by, for example, watchdog time-out events or by external reset events that require saving important device and SDRAM memory contents.	None
SYS.nIRQ	I	Interrupt line input (active low)	GPIO
SYS.nVMODE	O	Active-low signal that indicates the internal state of the device and that the external power IC can lower the core voltage	See pinout.
SYS.nDMAREQ0	I	External DMA request 0 (system expansion) Level (active low) or edge (falling) sensitive **Note:** The selection level/edge is handled by the control module.	See pinout.
SYS.nDMAREQ1	I	External DMA request 1 (system expansion) Level (active low) or edge (falling) sensitive. **Note:** The selection level/edge is handled by the control module.	See pinout.
SYS.JTAGSEL1	I	If set, this signal selects an alternate set of pins for the JTAG debug port.	See pinout.
SYS.JTAGSEL2	I	If set, this signal selects alternate set of pins for the JTAG debug port.	See pinout.

1) I = Input, O = Output, OD = Open-drain output

2) When configured for that function.

26.2.3 Clock Configuration

26.2.3.1 System Clocks (Required)

Operation of the OMAP2420 requires two external input clocks:

- ☐ SYS.32K: The 32-kHz frequency is used for low-frequency operation. It supplies the wake-up domain for operation in lowest power mode.
- ☐ SYS.XTAL: The system clock (12, 13, or 19.2 MHz) is the main source clock of the chip. It supplies the internal phase-locked loops (PLLs) and several OMAP modules. The system clock input can be connected in either of two ways, as shown in Table 26−3.

Table 26−3. System Clock Input Configurations

Input Source	Mapping	Comment
Quartz crystal	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.

SYS.32K, SYS.XTALIN, and SYS.XTALOUT have permanent pin locations. No preconfiguration of a system control module (SCM) CONTROL_PADCONF register is required.

The hardware configuration after power-on reset has the internal oscillator enabled (that is, it assumes the input clock from a crystal). The selection to bypass the internal oscillator is controlled by the power, reset, and clock management (PRCM) module register PRCM_CLKSRC_CTRL[1:0] at physical address 0x4800 8060).

SYS.ALTCLK Input (Optional)

An additional clock can be provided (through the SYS.ALTCLK pin) to supply internal peripherals or PLLs or to provide a precise clock for NTSC (54 MHz) and/or USB (48 MHz). If not used as a clock input, this pin can be configured as a general-purpose input/output (GPIO) using SCM register CONTROL_PADCONF_HDQ_SIO (physical address 0x4800 0115). For SYS.ALTCLK to be input on the HDQ_SIO pin, the MUXMODE field (SCM register bits CONTROL_PADCONF_HDQ_SIO[2:0]) must be set to 010.

SYS.CLKOUT Output (Optional)

An output clock (SYS.CLKOUT pad) is available and can provide SYS.CLK (12, 13, or 19.2 MHz), 96 MHz, CORE_CLOCK (DPLL output), or 54 MHz as an output to the external system.

SYS.CLKOUT can be divided by 2, 4, 8, or 16, and its OFF state polarity is programmable.

SYS_CLKOUT can be managed by software using the PRCM register PRCM_CLKOUT_CTRL (physical address 0x4800 8070). The clock frequency value can be:

- Equal to the clock source (12, 13, or 19.2 MHz depending on the clock source)
- 96 MHz (from APLL_96)
- 48 MHz (from APLL_96)
- 12 MHz (from APLL_96)
- 54 MHz (from APLL_54)

The system clock frequency is evaluated by counting the system clock cycle in a 32-kHz clock cycle. This operation is realized with a timer. (For details, see Chapter 16, *Timers*).

The SYS.CLKREQ pin is an output of OMAP to switch on or off the system clock. If not used as clock request, the pin is a GPIO.

26.2.3.2 Reset Configuration

Reset pin SYS.nRESPWRON is used to reset the entire chip at power-on reset when the chip core voltage and I/O voltage are correctly set.

Reset pin SYS.nRESWARM is used to reset part of the chip when it has already booted (for example, to recover from a software crash).

Table 26−4 shows part of the chip reset by SYS.nRESWARM.

Table 26−4. Part of the Chip Reset by SYS.nRESWARM

Function/Block	SYS.nRESWARM Reset Action
System control (I/O multiplexing)	No
32-kHz synchronization timer	No
DPLL	No
SDRC	Conditionally put the SDRAM in self-refresh.
All other	Yes

SYS.nRESWARM is a bidirectional reset. When an internal OMAP reset occurs, SYS.nRESWARM goes low and resets all the peripherals. The SYS.nRESWARM output is open-drain, and, consequently, an external pullup resistor is required.

Both SYS.nRESPWRON and SYS.nRESWARM have permanently assigned pin locations with no preconfiguration of a PRCM CONTROL_PADCONF register required.

The cause of the reset is stored in the PRCM register RM_RSTST_MPU and this information is used by the boot ROM to decide whether to do a context restore operation, a normal boot, or a peripheral boot (also with pins SYS.BOOT[5:4], see Chapter 25, *Device Booting*).

26.2.3.3 SYS.BOOT Configuration

Six external pins, SYS.BOOT[5:0], are used to select the boot memory interface configuration and the boot mode. These pins are sampled and latched onto a status register following a power-on reset.

The lower four pins, SYS.BOOT[3:0], allow choosing the memory type where the ROM code is going to find the user application for booting. The remaining upper two pins, SYS.BOOT[5:4], are used by the boot ROM code for peripheral booting and flashing. After booting, these pins can be used as GPIO for other functions.

Table 26−5 lists the options for SYS.BOOT[3:0].

Table 26−5. SYS.BOOT[3:0] Options

SYS.BOOT[3:0]	Internal/External	Wait Monitoring[1]	Memory Interface Type[1]
0000	External[2]	Active low	Nonmultiplexed 16-bit
0001	External[2]	Don't care	Nonmultiplexed 16-bit
0010	External[2]	Active low	Address/data-multiplexed 16-bit
0011	External[2]	Don't care	Address/data-multiplexed 16-bit
0100	External[2]	Reserved	Reserved
0101	External[2]	Reserved	Reserved
0110	External[2]	Reserved	Reserved
0111	External[2]	Reserved	Reserved
1000	Internal	Active low	Nonmultiplexed 16-bit
1001	Internal	Don't care	Nonmultiplexed 16-bit
1010	Internal	Active low	Address/data-multiplexed 16-bit
1011	Internal	Don't care	Address/data-multiplexed 16-bit
1100	Internal	−	NAND 8-bit
1101	Internal	−	NAND 16-bit
1110	Internal	Reserved	Reserved
1111	Reserved	Reserved	Reserved

1) External direct boot mode only supported on emulator devices

2) External boot memory must connect to GPMC.nCS0 and GPMC.WAIT0 pins.

SYS.BOOT Pin Mapping

Table 26−6 shows the SYS.BOOT[5:0] map to OMAP2420 pins.

Table 26−6. SYS.BOOT Pad Configuration Registers

	2420 Ball No.	**Pin Name**	**Corresponding CONTROL_PADCONF Register (System Control Module Registers)**	**Physical Address**
SYS.BOOT[0]	W5	GPIO_119	CONTROL_PADCONF_GPIO_119	0x4800 012E
SYS.BOOT[1]	Y5	GPIO_120	CONTROL_PADCONF_GPIO_120	0x4800 012F
SYS.BOOT[2]	R9	GPIO_121	CONTROL_PADCONF_GPIO_121	0x4800 0130
SYS.BOOT[3]	P8	GPIO_122	CONTROL_PADCONF_GPIO_122	0x4800 0131
SYS.BOOT[4]	V17	GPIO_36	CONTROL_PADCONF_GPIO_36	0x4800 0135
SYS.BOOT[5]	V18	GPIO_124	CONTROL_PADCONF_GPIO_124	0x4800 0139

Note:

Although the SYS.BOOT[5:0] signals are not the default (CONTROL_PADCONF_xxx[2:0] = 000) setting for their respective pads, those pads are always interpreted as SYS.BOOT[5:0] during the power-on sequence and sampled and stored accordingly. After the power-on sequence, those pads revert to the signals assigned by their CONTROL_PADCONF_xxx[2:0] register bits in the SCM.

SYS.BOOT[3] Setting

SYS.BOOT[3] controls the reset state of certain pins used for the general-purpose memory controller interface on the OMAP2420. According to the SYS.BOOT[3] setting shown in Table 26−7, SYS.BOOT[3] indicates whether the boot ROM code is in internal or external memory, and thus, how the GPMC pins are assigned at power-on reset.

Table 26−7. GPMC Interface Pin Setting per SYS.BOOT[3]

SYS.BOOT[3] Setting	**Boot ROM Memory**	**GPMC.A[10:1] Pins Setting**	**GPMC.D[15:8] Pins Setting**
0	External	GPMC.A[10:1]	GPMC.D[15:8]
1	Internal	Safe mode (MuxMode = 111)	Safe mode (MuxMode = 111)

After power-on reset, the signal assignment to these pins reverts to the signal selected by their corresponding SCM CONTROL_PADCONF_X register MUXMODE field. (For details, see Chapter 8, *System Control Module,* and Chapter 24, *Pinout Overview*.)

26.2.3.4 JTAG Interface Configuration

Table 26−8 lists alternate JTAG ports.

Table 26−8. Alternate JTAG Ports

JTAG Function	Pin Mapping when SYS_JTAGSEL1 = 1	2420 Ball No.	Pin Mapping when SYS_JTAGSEL2 = 1	2420 Ball No.
TCK	SPI2_SIMO	R19	EAC_BT_SCLK	R8
TMS	SPI2_CLK	T19	EAC_BT_DIN	Y3
TDI	USB0_VM	K20	USB0_VM	K20
TDO	USB0_DAT	K18	USB0_DAT	K18
RTCK	SPI1_CNS3	U21	GPIO124[1]	U21
NTRST	SPI2_NCS0	M14	EAC_BT_DOUT	W4
EMU0	SPI2_SOMI	R20	EAC_BT_FS	P9
EMU1	USB0_RCV	J18	USB0_RCV	J18

1) The GPIO124 signal must be programmed for output on the SPI1_NCS3 pin as indicated in the CONTROL_PADCONF_GPMC_NCS3 register.

When an alternate JTAG interface is enabled, the JTAG input signals come from the selected interface and the output signals are duplicated on both the primary and alternate interfaces.

Either the SYS_JTAGSEL1 or the SYS_JTAGSEL2 input signal can be programmed to be available on the GPI0_125 pin through the system control module register, CONTROL_PADCONF_GPIO_125 (physical address 0x4800 013A). MUXMODE = 001 selects SYS_JTAGSEL1; MUXMODE = 010 selects SYS_JTAGSEL2.

26.2.3.5 SYS.nVMODE

The pin SYS.nVMODE function can be used as an output to indicate to an external power-management chip which of two core voltages is to be supplied to the OMAP2420. The polarity of this signal is controlled through PRCM register bit PRCM_POLCTRL[0] (EXTVOL_POL), and the selection of VMODE level 0 voltage or level 1 voltage is set by PRCM register bits PRCM_VOLTCTRL[1:0] (for details, see Chapter 5, Power, Reset, and Clock Management). Configure the high and low core-voltage levels between the authorized values (low voltage to high voltage). This feature allows support for dynamic voltage scaling.

26.2.3.6 External Interrupt, SYS.nIRQ

The input signal SYS.nIRQ is an optional external active-low interrupt that can be managed directly by the MPU interrupt handler. Otherwise, this pin can be configured as a GPIO.

26.2.3.7 External DMA Request, SYS.nDMAREQ[1:0]

The two SYS.nDMAREQ[1:0] pins are optional external DMA requests that can be managed directly by the system DMA (sDMA) controller. DMA requests are programmable to be either level-sensitive (active low) or edge-sensitive (falling edge). Table 26−9 shows the SCM registers and physical pins where the external DMAREQ signals can be found and configured.

Table 26−9. External DMAREQ[1:0] Settings

External DMAREQ	Pin Location	Default Signal (@ MUX-MODE = 000)	Corresponding CONTROL_PADCONF Register	MUX-MODE[2:0]	Physical Address
SYS.nDMAREQ0	J15	I2C2_SCL	CONTROL_PADCONF_ I2C2_SCL	001	0x4800 0113
SYS.nDMAREQ1	H19	I2C2_SDA	CONTROL_PADCONF_ I2C2_SDA	001	0x4800 0114

To select either edge or level sensitivity for the external DMAREQ signals, the SCM register (CONTROL_DEVCONF[1:0]) must be used. Table 26−10 shows the external DMAREQ sensitivity settings.

Table 26−10. External DMAREQ Sensitivity Setting

SCM Register	Bit Setting	Sensitivity Setting
CONTROL_DEVCONF[1]	0	DMAREQ1 is level-sensitive.
	1	DMAREQ1 is edge-sensitive.
CONTROL_DEVCONF[0]	0	DMAREQ0 is level-sensitive.
	1	DMAREQ0 is edge-sensitive.

26.3 Device Initialization by ROM Code

For a production device, the ROM code is the first software to execute. This section discusses the flow of ROM code, including memory booting, peripheral booting, context restore, and initial software.

26.3.1 System Clock Detection/Initialization

The OMAP 2420 uses the following input clocks:

- ❑ A 32-kHz clock at SYS.32KIN
- ❑ A 12-, 13-, or 19.2-MHz clock at SYS.XTLIN/OUT inputs

The ROM code uses GP timer 1 (GPT1) to detect the input frequency supplied (second item above). This information is necessary for correctly programming the clock frequencies required for the USB and UART.

The 32k_sync timer and the GPT1 in the WKUP power domain are used to calculate the input frequency. The GPT1 is controlled by the CM_FCLKEN_WKUP and CM_CLKSEL_WKUP PRCM registers. The 32k_sync timer is always active. The registers of GPT1, TCAR1, and TCAR2 are used to measure on a 32-kHz cycle.

The frequency is programmed into the APLL input frequency to match the correct value (12, 13, or 19.2 MHz), using the CM_CLKSEL1_PLL[25:23] APLLS_CLKIN field (therefore, the 96-MHz PLL relocks). After this, the USB and UART are correctly supplied by the 48-MHz clock. At this point, the UART can be programmed to the correct baud rate.

Immediately after clock detection, this process also sets the DPLL output clock to 24 MHz, which is the lowest frequency that can be set for the input frequencies above. This ensures the same ROM code performance, regardless of the clock input.

The DPLL output clock can also be changed during client configuration at speedup or later by the customer software.

26.3.1.1 GPMC Initialization

The ROM code initializes the general-purpose memory controller (GPMC) as follows:

- ❑ Sets the GPMC registers, if NOR connects to CS0, to set the CS0 at 0x0800 0000, and for a size of 128MB
- ❑ Sets the GPMC multiplexing mode according to the setting in the SYS.BOOT pins
- ❑ Sets the GPMC waiting monitoring configuration according to the setting in the SYS.BOOT pins

In initial software, you must configure the GPMC to be in line with the timing of flash memory (for details, see Chapter 12, *Memory Subsystem*.

26.3.2 Registers Read/Changed by ROM Code

This section summarizes some of the registers that are read/written during booting:

- RM_RSTSTST_MPU saves the reset reason for the device. The customer software resets this register after it reads the reset reason.
- The CONTROL_STATUS register stores the device type and contains SYS_BOOT[5:0]. These bits also describe the NAND type used.
- PAD_CON for GPMC registers is set to configuration 0 for setting active A1 to A10 and D0 to D15 and all the GPMC signals.
- SDRC is set according to GPR1 to GPR13. The PAD_CONF registers for SDRC are also set.
- PRCM registers are associated with clock detection and clock settings.

26.3.3 Memory Use by ROM Code

The RAM used by the ROM code is contained in a section called ROM_CODE_VARs; it has a size of 2K bytes and is at address RAM + 62K bytes (+ 0xF800). The customer software can use this section of RAM after it has booted, but it must be aware that data in the section can be changed by the ROM code after any reset.

Table 26−11 lists the memory addresses used by the ROM code.

Table 26−11. Memory Used by ROM Code

ROM ADDRESS	Description
0x00000000	ROM exception vectors. The customer software must remap this address to a more suitable location or use the in-directed vectors at ROM_CODE_VARs listed in this table.
0x00001100	32-bit CRC of the public ROM, patched after the calculation is performed
0x00001104	4-byte-long ROM version information (integer)
Internal RAM ADDRESS	**Description**
ROM_CODE_VARs+0x0000	Interrupt vector: Undefined instruction
ROM_CODE_VARs+0x0004	Interrupt vector: Software interrupt
ROM_CODE_VARs+0x0008	Interrupt vector: Prefetch abort
ROM_CODE_VARs+0x000C	Interrupt vector: Data abort
ROM_CODE_VARs+0x0010	Interrupt vector: Reserved instruction
ROM_CODE_VARs+0x0014	Interrupt vector: IRQ
ROM_CODE_VARs+0x0018	Interrupt vector: FIQ
ROM_CODE_VARs+0x0030	Trace buffer 8 bytes. 64-bit encoded status information about the path the ROM code followed, and where it failed.

26.3.4 Initial Software

The bootstrap is the initial software after the ROM code executes during memory booting. It can be small, by including only code to branch to the start boot code of the operating systems, or it can be large, with all the functions of a start boot code of the operating systems. Users are fully responsible for the contents of the bootstrap.

Note:

All the information in this section is recommended but is not mandatory for OMAP2420 users.

26.3.4.1 Bootstrap Overview

Bootstrap puts the device in a state from which customer software can begin execution. This typically involves managing the watchdog timer, initializing CPU registers, searching for and getting memory size, initializing peripherals and platform hardware modules, and mapping virtual address space by enabling the CPU memory management unit (MMU) feature.

Bootstrap also saves the parameter passed by the ROM code (R0), which is the least-significant byte in the boot message, so that the customer software can retrieve it to fulfill user requirements (for example, for implementing different boot methods). Details of boot parameters are in Chapter 25, *Device Booting*.

26.3.4.2 Hardware Initialization

The primary role of the bootstrap is to initialize hardware devices to set the device in a stable state. The list of initializations is provided below (some steps are not always required):

- ☐ Define microprocessor mode (that is, start from thumb mode).
- ☐ Disable watchdog timers to prevent the processor from unnecessary resetting.
- ☐ Disable all interrupts to prevent unexpected interrupts from going off when hardware is being initialized.
- ☐ Power off the DSP domain.
- ☐ Power off the GFX domain.
- ☐ Configure the DPLL (for example, setting clock dividers), the memories (for example, SDRC, GPMC), and the CPU clock frequencies. (SDRAM cannot be reconfigured if code is executed from SDRAM.)
- ☐ Define CPU capabilities (for example, separating instruction and data cache support from write buffer support [writing through or writing back caching support]).
- ☐ Set up processor pin multiplexing with the SCM.
- ☐ Initialize a debug serial port if needed, which can be any onboard UART.
- ☐ Load application image to external flash memory or SDRAM.

- Load interrupt vectors (that is, IRQ, FIQ, software, RESET, and ABORT [prefetch/data]).
- Peripheral port mapping
- Module clock initialization
- Memory stack (that is, SVC, data abort, FIQ, IRQ)
- Enable interrupts.
- Initialize microprocessor registers.
- Set up the MMU and all required translation table entries.

It is possible that at some point the bootstrap will switch software handlers for executing C code.

26.3.4.3 Examples of Start Boot Code in OS

Any operating system implements a start-up booting code to manage the earliest hardware configuration.

Symbian has a bootstrap that performs some of the functions mentioned in Section 26.3.4.2, *Hardware Initialization*.

The WinCE operating system uses eBOOT, which is a boot loader responsible for downloading, flashing, and booting the WinCE image. It can be XIP in NOR flash. If it is on NAND flash, once eBOOT is loaded into RAM, it loads the XIP portion of the CE image from NAND to SDRAM. This XIP image runs and loads additional portions of the OS into RAM as needed.

uBoot is a standard boot loader for Linux.

26.3.4.4 Flash Loader

The flash loader is an initial software that is downloaded from the host to RAM and executed in RAM. The functionality of flash loader can include, but is not limited to, the following:

- Disable watchdog.
- Initialize.
  - GPMC based on the flash memory connected to CS0, based on the SYS.BOOT pin
  - The interface to the host (for example, UART or USB)
  - The flash memory used (for example, NOR, NAND, or MDOC)
  - SDRAM (for example, DDR or SDR)
- Download the customer software image (for example, OS) from the host to SDRAM.
- Write the customer software image to external flash memory from SDRAM.

Appendix A

Application Notes References

Title	Reference
Understanding Power Management on the OMAP2420 Device	SWPA034
Configuring JTAG Emulators for the OMAP24xx Software Development Platform	SWPA037
Using the OMAP2420 Device to Rotate and Display Images	SWPA040
Understanding Image Rotation Setup on the OMAP2420 Device	SWPA041
OMAP24xx Display Low Power Refresh	SWPA042
Performing a Basic DSP Initialization on the OMAP2420 Device	SWPA046
Configuring XDS510 USB JTAG Emulator and Code Composer Studio for the OMAP2420 Device SDP	SWPA057
OMAP2420 EAC Audio Application for the OMAP2420 Software Development Platform	SWPA071
Using OpenGL ES SDK for a Simple 3D Graphics Application under Symbian	SWPA083
OMAP2420 GPMC Configuration For Synchronous Mode	WMN_119_1
OMAP2420 DLL/DCDL Usage	WMN_120_1
OMAP2420 DSS Interface and Functional Clocks Enable Time	WMN_122_1
Using OMAP2420 With a Square System Clock Input Requiring Stabilization	WMN_123_1

Contact your TI representatives for more information about these Application Notes.

Appendix B

Glossary

A

AAC: *Advanced Audio Coding*

ABB: *Analog Baseband*

ABU: *Autobuffering Unit*

AC97: *Audio Codec 1997.* A standard audio interface that defines a high-quality 16-bit audio architecture.

ACB: *ac-bias frequency*

ACBI: *ac-bias Line Transitions per Interrupt*

ACE: *ASIC Compiler Environment.* The graphic user interface delivery mechanism for submicron gate array memory compiler elements.

ADC: *Analog-to-Digital Converter/Conversion*

AHB: *Advanced High Performance Bus*

AIC: *Analog Interface Chip*

ALE: *Address Latch Enable*

AMBA:*Advanced Microcrontroller Bus Architecture*

AMR: *Adaptive Multirate*

AMR: *Audio Modem Riser.* An Intel specification that defines a new architecture for the design of motherboards.

ANSI: *American National Standards Institute*

APE: *Application Engine*

API: *Application Programming Interface*

APLL: *Analog Phase-Locked Loop*

AR: *Automatic Reload*

ARGB: *Alpha, Red, Green, Blue*

ARM: *Advanced RISC Machine*

ARMIO: *Advanced RISC Machine Input/Output.* See MPUIO.

ASIC: *Application-Specific Integrated Circuit.* A chip built for a particular application. In the context of this document, this refers to the FPGA that resides on the EVM board.

ASP: *Application-Specific Peripheral*

ASSP: *Application-Specific Silicon Product or Application-Specific Standard Product*

AuSPI: *Audio Serial Port Interface*

B

B: *Byte, 8 bits*

B_MMU: *Bufferable Memory Management Unit.* See MMU.

BB: *Busy Bus*

BCD: *Binary-Coded Decimal.* A representation of decimal digits (0–9) using a nibble that uses a certain number of bits. For example, by using 4 bits, two BCDs can be packed into one byte.

BCM: *BIST Combiner Module*

BCPM: *BIST Controller Programmable Module*

BE: *Big Endian.* An addressing protocol in which bytes are numbered from left to right within a word. More significant bytes in a word have lower numbered addresses. Endian ordering is specific to hardware and is determined at reset. See also little endian (LE).

BIOS: *Built-In Operating System*

BGA: *Ball Grid Array*

BIST: *Built-In Self-Test*

BLOCK: A group of interconnected cells. May contain instances of other blocks.

Bluetooth: A short-range radio technology aimed at simplifying communications among network devices and between devices and the Internet. It also aims to simplify data synchronization between network devices and other computers.

BNC: *British Naval Connector, Bayonet Nut Connector, or Bayonet Neill Concelman.* A type of connector used with coaxial cables.

BOF: *Beginning of Frame*

BPP: *Bits Per Pixel*

BSP: *Buffered Serial Port.* An on-chip module that consists of a full-duplex, double-buffered serial port interface and an autobuffering unit (ABU). The double-buffered serial port of the BSP is an enhanced version of the standard serial port interface. The double-buffered serial port allows transfer of a continuous communication stream (8-,10-,12- or 16-bit data packets).

BTA: *Bus Turn Around*

C

C_MMU: *Cacheable Memory Management Unit.* See MMU.

C-Port: *Codec Port*

CamDMA: *Camera Subsystem Direct Memory Access Module*

CASP: *Customer Application−Specific Peripheral*

CB: *Copy Back*

CBC: *Cipher Block Chaining*

CCP: *Compact Camera Port*

CDMA: *Code Division Multiple Access*

CDP: *Coprocessor Data Operation*

CE: *Chip Enable*

Certificate: Data block that is digitally signed with private key. Used enabling or disabling certain functionalities.

CFC: *CompactFlash Controller*

CIF: *Common Intermediate Format.* A video format used in video conferencing systems that easily supports both NTSC and PAL signals. CIF is part of the ITU H.261 video conferencing standard. It specifies a data rate of 30 frames per second (fps), with each frame containing 288 lines and 352 pixels per line.

CIO: *Channel Input/Output*

CLE: *Command Latch Enable*

CLK: *Clock*

CLKM: *Clock and Reset Management*

CLKSTP: *Clock Stop*

Clock Gating: Removing the ac frequency of the clock, usually through a logic (AND, OR) gate.

Clock Throttling: Reducing the frequency of the clock

CLUT: *Color Look-Up Table*

CMOS: *Complimentary Metal Oxide Semiconductor*

CMT: *Cellular Mobile Telephone*

CO: *Controlled Oscillator*

CODEC: *Coder/Decoder or Compression/Decompression.* A device that codes in one direction of transmission and decodes in another direction of transmission.

COFF: *Common Object File Format*

COM: *Communication*

Companding: *Compressing and expanding.* A quantization scheme for audio signals in which the input signal is compressed and then, after processing, is reconstructed at the output by expansion. There are two distinct companding schemes: A-law, used in Europe, and μ-law, used in the United States.

ConnID: *Connection Identifier.* An initiator module identifier. A ConnID is transmitted in-band with the request and is used for security and error logging mechanism.

CP15: *Coprocessor 15.* This coprocessor controls the operation and configuration of the TI925T.

CPLD: *Complex Programmable Logic Device.* An integrated circuit that can be programmed to perform complex functions.

CPU: *Central Processing Unit.* The CPU is the portion of the processor involved in arithmetic, shifting, and Boolean logic operations, as well as the generation of data and program memory addresses. The CPU includes the central arithmetic logic unit (CALU), the multiplier, and the auxiliary register arithmetic unit (ARAU).

CPR: *Clock, Power, Reset*

CRC: *Cyclic Redundancy Check*

CS: *Chip-Select*

CSMI: *Coprocessor-Shared Memory Interface*

CSMM: *Coprocessor-Shared Memory Model*

CTRL: *Control*

CTS: *Clear to Send*

D

DABORT: *Data Abort*

DAC: *Digital to Analog Converter*

DAI: *Digital Audio Interface.* A GSM test interface that is used to determine the routing of speech data for the devices being tested.

DARAM: *Dual Access Random Access Memory.* RAM that can be accessed twice in a single CPU clock cycle. For example, your code can read from and write to DARAM in the same clock cycle.

DBB: *Digital Baseband*

D-CACHE: *Data Cache*

DCD: *Data Carrier Detect*

DCDL: *Digitally-Controlled Delay Line*

DCT: *Discrete Cosine Transform.* A fast Fourier transform used in manipulating compressed still and moving picture data.

DDMA: *DSP Subsystem Direct Memory Access Module*

DDR: *Dual Data Rate*

Default Mode: The default mode is the mode selected at the release of the power on reset.

DFA: *Differential Frequency Analysis*

DFF: *Digital Flip-Flop*

DI: *Data In*

Die ID: A 128-bit register composed of eFuse cells programmed during the fabrication process, which identifies uniquely each OMAP2420 silicon die.

DIF: *Display Interface*

DIM: *Dynamic Identification Mechanism*

DIMM: *Dual Inline Memory Module*

DIN: *Data In*

DIP: *Dual Inline Package*

DISPC: *Display Controller*

DLB: *Data Loopback.* A synchronous serial port test mode in which the receive pins are connected internally to the transmit pins on the same device. This mode, enabled or disabled by the DLB bit, allows you to test whether the port is operating correctly.

DLL: *Delay-Locked Loop*

DMA: *Direct Memory Access.* A mechanism whereby a device other than the host processor contends for and receives mastery of the memory bus so that data transfers can take place independent of the host.

DMA Controller: *Direct Memory Access Controller.* Controls data block transfers between memories, peripherals, and processors.

DO: *Data Out*

DPA: *Differential Power Analysis*

DPLL: *Digital Phase-Locked Loop.* Digital implementation of PLL.

DRD: *Dual Role Device*

DRM: *Digital Rights Management*

DSP: *Digital Signal Processor.* A semiconductor that manipulates discrete or discontinuous electrical impulses in a manner that implements adesired algorithm.

DSP SS: *Digital Signal Processor Subsystem*

DSP/BIOS: *Digital Signal Processor/Basic Input/Output System*

DSR: *Data Set Ready*

D-TLB: *Data Transition Lookaside Buffer.* See TLB.

DTR: *Data Transmit Ready*

DRAM: *Dynamic Random Access Memory.* Single-access data RAM generally used in RAM-based modules to emulate data ROM in future ROM megamodules.

DPRAM: *Dual-Port RAM*

DSA: *Digital Signature Algorithm*

DVS: *Dynamic Voltage Scaling*

E

E^{2}ICE: *Enhanced Embedded ICE Module*

EAC: *Enhanced Audio Controller*

ECB: *Electronic Code Book*

ECC: *Electronic Code Book*

EDGE: *Electronic Code Book*

EEPROM: *Electrical Fuse.* A one-time programmable memory location usually set at the factory

EGA: *Enhanced Graphics Adapter*

EMIF: *Extended Memory Interface.* Consists of the EMIFS and EMIFF.

EMIFF: *Extended Memory Interface Fast.* The 16−bit bus can interface with synchronous DRAM (SDRAM).

EMIFS: *Extended Memory Interface Slow.* This 16−bit wide bus can interface with and handle all transactions to flash memory, ROM, asynchronous memories, and synchronous burst flash.

EOF: *End of Frame*

EP: *Entry Point*

EPROM: *Erasable Programmable Read-Only Memory*

ES: *Erase Status*

ESC: *Escape*

ESK: *Emulation Software Kit*

ESS: *Erase Suspend Status*

ETK: *Embedded Toolkit*

ETLM: *Emulation TAP Linking Module*

ETM: *Embedded Trace Macrocell*

EVM: *Evaluation Module*

EXS: *Exit Sequence*

F

FAC: *Frame Adjustment Counter*

FAR: *Fault Address Register.* The FAR holds the virtual address of the access, which was attempted when a fault occurred.

FARC: *Frame Adjustment Reference Count*

FC: *Flow Control*

FCELL: *eFuse Cell*

FDD: *FIFO DMA Request Delay*

FE: *Framing Error.* An error that occurs when the asynchronous serial port receives a data character that does not have a valid stop bit.

FEC: *Frame End Code*

FIFO: *First In First Out.* A queue; a data structure or hardware buffer from which items are taken out in the same order they were put in. A FIFO is useful for buffering a stream of data between a sender and receiver which are not synchronized; that is, the sender and receiver are not sending and receiving at exactly the same rate. If the rates differ by too much in one direction for too long, the FIFO becomes either full (blocking the sender) or empty (blocking the receiver).

FIPS: *Federal Information Processing Standards*

FIQ: *Fast Interrupt Request.* See ISR.

FIR: *Fast Infrared*

Flash Loader: A client application that is downloaded during the flashing phase into internal SRAM by the ROM code SW, then executed after successful authentication and integrity checking. This application may download a new image, burn it into flash, then cause a warm reboot.

FPGA: *Field Programmable Gate Array.* A type of logic chip that can be programmed. An FPGA is similar to a PLD; whereas PLDs are generally limited to hundreds of gates, FPGAs support thousands of gates.

Flashing: How a programming host preloads the internal RAM with a flash loader to program flash memories connected to the device.

FEMA: *Failure Mode and Effects Analysis*

FS: *Frame Synchronization*

FSC: *Frame Start Code.* Also Frame Start Count.

FSM: *Finite State Machine.* A model of computation consisting of a set of states, a start state, an input alphabet, and a transition function that maps input symbols and current states to a next state.

FSR: *Fault Status Register*

G

GCRM: *Global Clock and Reset Module*

GDD: *Generic Distributed DMA*

GDI: *Graphics Driver Interface*

GP Device: *General-Purpose Device.* A device in which the security is disabled.

GF: *Galois Field*

GND: *Ground*

GPMC: *General-Purpose Memory Controller*

GPE: *General-Purpose Event*

GPP: *General-Purpose Processor*

GPIO: *General-Purpose Input/Output.* Pins that can be used to accept input signals and/or send output signals but are not linked to specific uses.

GPS: *Global Positioning System*

GSM: *Global System for Mobile Communications*

GSM-S: *GSM Subsystem*

H

HC: *Host Controller*

HCI: *Host Controller Interface*

H/W: *Hardware*

HBP: *Horizontal Back Porch*

HFP: *Horizontal Front Porch*

HIO: *Host I/O*

HIVECT: *High Interrupts Vector*

HOM: *Host-Only Mode.* A mode that allows only the host to access host port interface (HPI) memory. The CPU has no access to the HPI memory block during HOM.

HPI: *Host Port Interface*

HSW: *Horizontal Synchronization Pulse Width*

HSYNC: *Horizontal Synchronization.* A bidirectional horizontal timing signal occurring once per line with a pulse width defined as an integral number of frame clock (FCLK) periods. Synchronization signals can be used to enable retrace of the electron beam of a display screen. Also HS.

HWA: *Hardware Accelerators.* In the context of this document, this refers to the DCT/IDCT, motion estimation and half-pixel interpolation accelerators in the OMAP1610 device.

I

IA: *Identifier Address*

IA: *Initiator Agent*

I2S: *Inter-IC Sound.* A digital audio interface standard.

I/O: *Input/Output*

IABORT: *Instruction Abort*

I-Cache: *Instruction Cache*

IC: *Integrated Circuit*

ICE: *In-Circuit Emulation*

ICR: *Intersystem Communication Registers*

ID: *Identification*

IDCT: *Inverse Discrete Cosine Transform.* See DCT.

IDE: *Integrated Development Environment.* A programming environment integrated into an application.

IF: *Interface*

IHV: *Independent Hardware Vendor*

ILR: *Interrupt Level Register*

IM: *Initiator Module.* A module is an initiator whenever it is able to initiate read and write requests to the chip interconnect (typically: processors, DMA, etc.).

IMIF: *Internal Memory Interface*

IMX: *Image Extension Coprocessor*

In Mux Logic: Combinational logic implied in input signals functional multiplexing

INT: *Interrupt.* A signal sent by hardware or software to a processor requesting attention. An interrupt tells the processor to suspend its current operation, save the current task status, and perform a particular set of instructions. Interrupts communicate with the operating system and prioritize tasks to be performed.

INTC: *Interrupt Controller*

I/O: *Input/Output*

IOM-2: *ISDN Oriented Modular Interface Revision 2*

IPC: *Interprocessor Communication.* (also referred to as “mailbox” on occasion)

IrDA: *Infrared Data Association.* Represents the group of device manufacturers that developed a standard for transmitting data via infrared light waves.

IR: *Infrared*

IRQ: *Interrupt Request.* IRQs are hardware lines over which devices can send interrupt signals to the microprocessor.

ISO: *Isochronous.* This refers to processes where data must be delivered within certain time constraints. For example, multimedia streams require an isochronous transport mechanism to ensure that data is delivered as fast as it is displayed and to ensure that the audio is synchronized with the video. Also, International Standards Organization.

ISR: *Interrupt Service Routine.* A function or set of functions that are called when an interrupt is encountered.

IST: *Interrupt Service Thread.* A thread that provides additional processing necessary for an interrupt, but which may be on the order of several microseconds.

ISV: *Independent Software Vendor*

IT: *Interrupt*

I-TLB: *Instruction Translation Lookaside Buffer.* See TLB.

ITR: *Interrupt Input Register*

J

JPEG: *Joint Photographics Experts Group*

JTAG: *Joint Test Action Group.* The Joint Test Action Group was formed in 1985 to develop economical test methodologies for systems designed around complex integrated circuits and assembled with surface-mount technologies. The group drafted a standard that was subsequently adopted by IEEE as IEEE Standard 1149.1−1990, IEEE Standard Test Access Port and Boundary-Scan Architecture.

K

KB: *Kilobyte, 1024 B*

KBC: *Keyboard Column*

KBD: *Keyboard*

Kbps: *Kilobits per second*

KBR: *Keyboard Row*

L

L1: *Level 1 Cache/Memory*

L2: *Level 2 Cache/Memory*

L3: *Level 3 Interconnect*

L3 initiator: *L3 port that initiates transfers on the L3 interconnect*

L3 target: *L3 port that receives transfers on the L3 interconnect*

L4: *Level 4 Interconnect*

LAN: *Local Area Network*

LCD: *Liquid Crystal Display.* A display that uses two sheets of polarizing material with a liquid crystal solution between them.

LCh: *Logical DMA Channel*

LDC: *Load (from memory) to Coprocessor*

LDM: *Load Multiple*

LDO: *Low Dropout*

LE: *Little Endian.* An addressing protocol in which bytes are numbered from right to left within a word. More significant bytes in a word have higher numbered addresses. Endian ordering is specific to hardware and is determined at reset. See also big endian (BE).

LEC: *Line End Code*

LED: *Light Emitting Diode.* An electronic device that lights up when electricity is passed through it.

LFSR: *Linear-Feedback Shift Register*

LH: *Local Host*

LLRC: *Low Latency Response Crossbar.* Subpart of the L3 interconnect optimized for low latency.

LPG: *Light Pulse Generator*

LPP: *Lines Per Panel*

LRU: *Least Recently Used*

LS: *Level Shifter*

LSB: *Least Significant Bit*

LSC: *Line Start Code*

LVDS: *Low-Voltage Differential Signaling*

M

MAC: Multiply Accumulate

MB: Megabyte, 1024 KB

McBSP: Multichannel Buffered Serial Port. An enhanced buffered serial port that includes the following standard features: buffered data registers, full duplex communication, and independent clocking and framing for receive and transmit. In addition, the McBSP includes the following enhanced features: internal programmable clock and frame generation, multichannel mode, and general-purpose I/O.

MCSI: *Multichannel Serial Interface.* A serial interface with multichannel transmission capability.

MCSI1: *Multichannel Serial Interface 1*

MCSI2: *Multichannel Serial Interface 2*

MCSPI: *Multichannel Serial Port Interface*

MCU: *Microcontroller Unit.* Refers to the MPU.

MD5: *Message Digest Algorithm Revision 5*

MDDR: *Mobile Double-Data-Rate* SDRAM, dedicated to mobile applications

MDOC: *Memory Disk On Chip by M-Systems.* A type of flash.

ME: *Motion Estimation*

MEMIF: *Memory Interface*

MeSSI: *Medium-Speed Screen Interface*

MicroWire: This module provides a serial synchronous interface that can drive four serial external components.

MIPS: *Million Instructions Per Second*

MIR: *Medium Infrared*

MMA: *Multimedia Applications*

MMC: *Multimedia Card*

MMC/SD: *Multimedia Card/Secure Data*

MMIO: *Multimedia Card/Secure Data*

MMU: *Memory Management Unit.* The MMU performs virtual-to-physical address translations, performs access permission checks for access to the system memory, and provides the flexibility and security required for the OS to manage a shared physical memory space between the two processors.

MP3: *MPEG Layer 3.* An audio compression format.

MPEG: *Motion Pictures Expert Group.* A compression scheme for full motion video.

MPEG1: The first MPEG compression scheme specification.

MPEG4: The most current MPEG compression scheme specification, intended for very narrow bandwidths.

MPU: *Microprocessor Unit*

MPU SS: *MPU-Subsystem*

MSB: *Most Significant Bit.* The highest order bit in a word. The plural form (MSBs) refers to a specified number of high-order bits, beginning with the highest order bit and counting to the right. For example, the eight MSBs of a 16-bit value are bits 15 through 8.

MSDR/LPSDR: Single-Data-Rate SDRAM, Mobile (or low-power) devices, dedicated to mobile applications

MUX: *Multiplex/Multiplexer*

Muxed pin: A pin is muxed when its pin control register field can be reconfigured by software to change the function associated with the PIN.

MuxMode: 3-bit field of the pin control register field which enables to change the mode. Mode programming is assumed by software and selects a function on the device external interface.

MVIP: *Multivendor Integration Protocol*

N

NAND: NAND flash memory is a high-capacity, low-cost embedded permanent data storage solution.

NAND CE: *NAND Chip Enable*

NAND CE Don't Care: *NAND Chip Enable Don't Care*

NC: *Not Connected*

NCB: *Non-Cacheable and Buffered*

NCNB: *Non-Cacheable and Non-Buffered*

NFC: *NAND Flash Controller*

NFMC: *NAND Flash Memory Core*

NIRQ: *Negative (Logic) Interrupt Request.* See IRQ.

NIST: *National Institute of Standards and Technology*

NMI: *Nonmaskable Interrupt.* An interrupt that can be neither masked nor disabled.

NOR: A type of flash memory

NRT: *Non-Real-Time*

NSC: *National Semiconductor Corporation*

NTSC: *National Television System Committee.* Television broadcast system.

O

OAL: *OEM Adaptation Layer*

OCM: *On-Chip Memory*

OCP: *Open-Core Protocol*

OCPI: *Open-Core Protocol Interface*

OE: *Output Enable*

OEM: *Original Equipment Manufacturers*

OHCI: *Open Host Controller Interface.* This is an industry standard USB Host Controller Interface.

OMAP: An open software and hardware platform targeted at second/third generation cellular phones with multimedia capabilities.

Out Mux logic: Combinational logic used in output signal functional multiplexing.

OS: *Operating System*

OTG: *On-The-Go*

P

PA: *Program Address*

PAL: *Programmable Array Logic*

PB: *Peripheral Bus.* Refers to the TIPB.

PCB: *Printed Circuit Board*

PCI: *Peripheral Component Interconnect.* A local bus standard that is 64 bits wide, though it is usually implemented as a 32-bit bus. It can run at clock speeds of 33 or 66 MHz. At 32 bits and 33 MHz, it yields a throughput rate of 133M bps.

PCM: *Pulse Code Modulation.* A technique for digitizing speech by sampling the sound waves and converting each sample into a binary number.

PCS: *Personal Communication System.* The U.S. Federal Communications Commission (FCC) term used to describe a set of digital cellular technologies being deployed in the U.S. PCS works over CDMA (also called IS–95), GSM, and North American TDMA (also called IS–136) air interfaces.

PD: *Program Data*

PDA: *Personal Digital Assistant*

PDRAM: *Program Data Random Access Memory*

PDROM: *Program Data Read-Only Memory*

PE: *Parity Error*

PGA: *Pin Grid Array*

PHY: *Physical Layer Controller*

PI: *Pixel Interpolation*

PID: *Protocol Identifier.* The PID register is used in Windows CE mode only.

PIM: *Personal Information Management*

PISO: *Parallel In/Serial Out*

PK: *Public Key*

PLD: *Programmable Logic Devices*

PLL: *Phase-Locked Loop.* A closed loop frequency control system whose function is based on the phase-sensitive detection of the phase difference between the input signal and the output signal of the controlled oscillator (CO).

PMT: *Parallel Module Test.* One of the test configuration modes of the OMAP1509 processor.

POR: *Power-On Reset*

PPA: *Primary Protected Application*

PPC: *Palm-Size PC*

PPCELL: *Program Protection Cell*

PPL: *Pixels per Line*

PRAM: *Single-Access Program RAM. Generally used in RAM-based modules to emulate program ROM in future ROM Megamodules.*

PRBS: *Pseudorandom Bit Sequence*

PRCM: *Power, Reset, Clock Management module*

Primary Mode: *Primary mode is mode0.* The function muxed in mode 0 gives its name to the PIN

PRNS: *Pseudo-Random Noise Source*

Production ID: An additional 64-bit register of eFuse cells used to include specific OMAP2420 needs.

PROM: *Programmable Read-Only Memory.* A memory chip on which data can be written only once.

Protected Application: Application that is signed by authenticated author and meant to be executed in secure execution environment

PSA: *Parallel Signature Analyzer*

PSC: *Prescaler Counter*

PSS: *Program Suspend Status*

PTV: *Prescale Clock Timer Value.* Sets the value of the divisor used in scaling the clock.

Public Debug Mode: A mode of operation of an emulator/test/secure device. In this mode public code debug is enabled and secure code execution is allowed.

Public Debug Mode: Used to verify the digital signature generated by corresponding private key

Pure Input: A pure input pin can be muxed on module inputs only.

Pure Output: A pure output pin can be muxed between module outputs only.

PWL: *Pulse Width Light (modulator).* A 4096-bit random sequence generator that provides control of the LCD backlighting and keypad.

PWM: *Pulse Width Modulation*

PWRON: *Power-On Reset*

PWT: *Pulse Width Tone.* Creates the output tone signal for a buzzer, programmable both in frequency as well as volume.

Q

QCIF: *Quarter Common Intermediate Format.* A video conferencing format that specifies data rates of 30 frames per second (fps), with each frame containing 144 lines and 176 pixels per line. This is one-fourth the resolution of CIF. QCIF support is required by the ITU H.261 video conferencing standard.

QOS: *Quality of Service*

QVGA: *Quarter Video Graphics Array.* One-fourth the resolution of VGA.

R

RAM: *Random Access Memory.* A memory element that can be written to, as well as read.

R/B: *Read/Busy*

RDR: *Receive Data Ready*

RDRY: *Receive Data Ready*

RE: *Read Enable*

RF: *Radio Frequency*

RFBI: *Remote Frame Buffer Interface*

RGB: *Red, Green, Blue*

RI: *Ring Indicator*

RISC: *Reduced Instruction Set Computer.* A computer whose instruction set and related decode mechanism are much simpler than those of microprogrammed complex instruction set computers. The result is a higher instruction throughput and a faster real-time interrupt service response from a smaller, cost-effective chip.

ROM: *Read Only Memory.* A semiconductor storage element containing permanent data that cannot be changed.

RSA: Stands for (Ron) *Rivest*, (Adi) *Shamir* and (Leonard) *Adleman*, its creators. A public key algorithm.

RST: *Reset*

RT: *Real-Time*

RTC: *Real-Time Clock.* A clock that keeps track of the time even when the device is turned off.

RTOS: *Real-Time Operating System*

RTS: *Request to Send*

R/W: *Read/Write*

RX: *Receive/Receiver*

RXC: *Bidirectional Serial Receive Clock*

RXD: *Receive Data*

S

S/W: *Software*

SAM: *Shared Access Mode.* The mode that allows both the DSP and the host to access host port interface (HPI) memory. In this mode, asynchronous host accesses are synchronized internally, and, in case of conflict, the host has access priority and the DSP waits one cycle.

SARAM: *Single Access Random Access Memory.* Memory space that can only be read from or written to in a single clock cycle; RAM that can be accessed (read from or written to) once in a single CPU cycle.

SB: *Silicon Backplane ™*

SB Flash: *Synchronous Burst Flash Memory*

SBIM: *Silicon Backplane Initiator Module Agent registers.* All initiator modules interfaced with SB used the same set of registers for their Initiator Agent: Registers of the same names control the same functions and are mapped to the same offset addresses but at different base addresses.

SBTM: *Silicon Backplane Target Module Agent registers.* All Target modules interfaced with SB used the same set of registers for their Target Agent: Registers of the same name control the same functions and are mapped to the same offset addresses but at different base addresses.

SBZ: *Should Be Zero*

SCL: *Serial Clock.* Programmable serial clock used in the I^{2}C interface. Also SCLK.

SCM: *Scan Combiner Module*

SCSA: *Signal Computing System Architecture.* An open, modular architecture for computer telephony that leverages applicable standards and evolves solutions to fill the gaps.

SD: *Starting Delimiter or Secure Data*

SDA: *Serial Data.* Serial data bus in the I^2C interface.

SDB: *Standard Development Board*

SDF: *Standard Delay Format*

SDIO: *Secure Digital Input/Output*

SDK: *Software Development Kit*

SDMA: *System Direct Memory Access* module

SDR: *Single Data Rate*

SDRC: *SDRAM Controller*

SDRAM: *Synchronous Dynamic Random Access Memory*

SDW: *Short Distance Wireless*

SERROR: *Slave Error*

SHA-1: *Secure Hash Algorithm Revision 1.* One of the most popular cryptographic one-way hash algorithms, which generates 160 bits digest from any length message.

SIM: *Subscriber Identity Module*

SIPO: *Serial In Parallel Out*

SIR: *Slow Infrared*

SMS: *SDRAM Memory Scheduler*

SoSSI: *Specially Optimized Screen Interface*

SP: *Serial Port or Small Page*

SPC: *Serial Port Control*

SPI: *Serial Port Interface.* A signaling protocol for exchanging serial data.

SRAM: *Static Random Access Memory.* Fast memory that does not require refreshing, as DRAM does. It is more expensive than DRAM, though, and is not available in as high a density as DRAM.

SRC: *Sample Rate Conversion*

SRG: *Sample Rate Generator*

SRP: *Session Request Protocol*

SS: *Subsystem*

SSR: *Serial Synchronous Receiver*

SST: *Serial Synchronous Transmitter*

ST: *Start Timer*

ST-BUS: *Serial Telecom Bus*

STC: Store from coprocessor (to memory) or *System Time Clock*, which is the master clock in an MPEG2 encoder or decoder system.

STM: *Synchronous Transfer Mode or Store Multiple*

STN: *Super-Twist Nematic.* A technique for improving LCD display screens by twisting light rays.

T

TA: *Target Agent*

TAP: *Test Access Port*

TC: *Traffic Controller.* Allows asynchronous operation among the external memory interface, the MPU, and the DSP.

TCIF: *Traffic Controller Interface*

TCK: *Test Clock*

TDDR: *Timer Divide-Down Register*

TDI: *Test Data Input*

TDMA: *Time Division Multiple Access*

TDM: *Time Division Multiplex/Multiplexing.* The process by which a single serial bus is shared by multiple devices with each device taking turns to communicate on the bus. The total number of time slots (channels) depends on the number of devices connected. During a time slot, a given device may talk to any combination of devices on the bus.

TDO: *Test Data Output*

TDRY: *Transmit Data Ready*

TFT: *Thin Film Transistor.* A type of LCD flat panel display screen in which each pixel is controlled by one to four transistors.

TI: *Texas Instruments*

TIM: *Timer.* Main count register.

TINT: *Timer Interrupt*

TIPB: *Texas Instruments Peripheral Bus.* Consists of two buses (private and public) that connect the TI925T to the external and internal peripherals.

TLB: *Translation Lookaside Buffer.* A cache that contains entries for virtual-to-physical address translation and access permission checking.

TLL: *Transceiverless Link.* This is logic which allows the user to connect two USB transceiver interfaces together directly without the use of differential transceivers.

TM: *Target Module.* A target module cannot generate read/write requests to the chip interconnects, but respond to these requests. However it may generate interrupts or DMA request to the system (typically: peripherals, memory controllers).

TMS: *Test Mode Select*

TOC: *Table of Contents*

TP: *Tiny Page*

TRST: *Test Reset*

TRX: *USB Transceiver.* The USB analog driver/receiver.

TSS: *Timer Stop Status*

TTB: *Translation Table Base.* It points to the base of a table in physical memory that contains section and page table descriptors.

TTL: *Transistor Transistor Logic*

TX: *Transmit//Transmitter*

TXC: *Bidirectional Serial Transmit Clock*

TXD: *Transmit Data*

U

UART: *Universal Asynchronous Receiver/Transmitter.* Another name for the asynchronous serial port.

UE: *Unrecoverable Error*

ULPD: *Ultralow-Power Device.* A state machine that can stop the oscillator and restart it on a wake-up signal.

UND: *Undefined*

UNP: *Unpredictable*

USAR: *Universal Synchronous/Asynchronous Receiver*

USART: *Universal Synchronous/Asynchronous Receiver/Transmitter*

USB: *Universal Serial Bus.* An external bus standard that supports data transfer rates of 12M bps (12 million bits per second). A single USB port can be used to connect up to 127 peripheral devices.

V

VA: *Volt-Amps.* A form of power management. A VA rating is the volts rating multiplied by the amps (current) rating, used to indicate the output capacity of an uninterruptible power supply (UPS) or other power source.

VBP: *Vertical Back Porch*

VCO: *Voltage Controlled Oscillator*

VENC: *Video Encoder*

VFIR: *Very Fast Infrared*

VFP: *Vertical Front Porch*

VGA: *Video Graphics Array.* An industry standard for video cards.

VGP: *Vertex Geometry Processor*

VIA: *Versatile Interconnection Architecture*

VIVT: *Virtual Index Virtual Tag*

VLCD: *Variable Length Coding and Decoding* coprocessor

VRFB: *Virtual Rotated Frame Buffer*

VSW: *Vertical Synchronization Pulse Width*

VSYNC: *Vertical Synchronization.* A bidirectional vertical timing signal occurring once per frame with a pulse-width defined as an integral number of lines (half-lines for interlaced mode). Also VS.

W

WB: *Write Buffer*

WCDMA: *Wideband Code Division Multiple Access.* A third-generation digital cellular technology that uses spread-spectrum techniques.

WCS: *Wireless Computing System*

WD: *Watchdog.* A timer that requires the user program or OS periodically write to the count register before the counter underflows.

WE: *Write Enable*

WMA: *Windows Media Audio*

Word16: *16-bit word*

Word32: *32-bit word*

Word8: *8-bit word*

WP: *Write Protect*

WSMS: *Write State Machine Status*

WS: *Wait State.* A period of time that the CPU must wait for external program, data, or I/O memory to respond when reading from or writing to that external memory. The CPU waits one extra cycle for every wait state.

WT: *Write Through*

X

Xbar: *Crossbar*

XIO: External Memory Interface (Input/Output) of the lead processor.

XIP: *eXecution In Place*

X-Loader: A user-defined pre-operating system bootstrap code that resides at the beginning of the external flash.

Appendix C

Index