OMAP5912 Multimedia Processor Device Overview and Architecture Reference Guide

Literature Number: SPRU748A
March 2004

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.

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
		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 document introduces the setup, components, and features of the OMAP5912 multimedia processor and provides a high-level view of the device architecture.

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.

Related Documentation From Texas Instruments

The following documents describe the OMAP5910 device and related peripherals. Copies of these documents are available on the Internet at www.ti.com. *Tip:* Enter the literature number in the search box provided at www.ti.com.

OMAP5912 Multimedia Processor Device Overview and Architecture Reference Guide (literature number SPRU748) introduces the setup, components, and features of the OMAP5912 multimedia processor and provides a high-level view of the device architecture.

OMAP5912 Multimedia Processor OMAP 3.2 Subsystem Reference Guide (literature number SPRU749) introduces and briefly defines the main features of the OMAP3.2 subsystem of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor DSP Sybsystem Reference Guide (literature number SPRU750) describes the OMAP5912 multimedia processor DSP subsystem. The digital signal processor (DSP) subsystem is built around a core processor and peripherals that interface with: 1) The ARM926EJS via the microprocessor unit interface (MPUI); 2) Various standard memories via the external memory interface (EMIF); 3) Various system peripherals via the TI peripheral bus (TIPB) bridge.

OMAP5912 Multimedia Processor Clocks Reference Guide (literature number SPRU751) describes the clocking mechanisms of the OMAP5912 multimedia processor. In OMAP5912, various clocks are created from special components such as the digital phase locked loop (DPLL) and the analog phase-locked loop (APLL).

OMAP5912 Multimedia Processor Initialization Reference Guide (literature number SPRU752) describes the reset architecture, the configuration, the initialization, and the boot ROM of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Power Management Reference Guide (literature number SPRU753) describes power management in the OMAP5912 multimedia processor. The ultralow-power device (ULPD) generates and manages clocks and reset signals to OMAP3.2 and to some peripherals. It controls chip-level power-down modes and handles chip-level wake-up events. In deep sleep mode, this module is still active to monitor wake-up events.This book describes the ULPD module and outline architecture.

OMAP5912 Multimedia Processor Security Features Reference Guide (literature number SPRU754) describes the security features of teh OMAP5912 multimedia processor. The OMAP5912 security scheme relies on the OMAP3.2 secure mode. The distributed security on the OMAP3.2 platform is a Texas Instruments solution to address m-commerce and security issues within a mobile phone environment. The OMAP3.2 secure mode was developed to bring hardware robustness to the overall OMAP5912 security scheme.

OMAP5912 Multimedia Processor Direct Memory Access (DMA) Support Reference Guide (literature number SPRU755) describes the direct memory access support of the OMAP5912 multimedia processor. The OMAP5912 processor has three DMAs:

- The system DMA is embedded in OMAP3.2. It handles DMA transfers associated with MPU and shared peripherals.
- The DSP DMA is embedded in OMAP3.2. It handles DMA transfers associated with DSP peripherals.
- The generic distributed DMA (GDD) is an OMAP5912 resource attached to the SSI peripheral. It handles only DMA transfers associated with the SSI peripheral.

OMAP5912 Multimedia Processor Memory Interfaces Reference Guide (literature number SPRU756) describes the memory interfaces of the OMAP5912 multimedia processor.

- SDRAM (external memory interface fast, or EMIFF)
- Asynchronous and synchronous burst memory (external memory interface slow, or EMIFS)
- NAND flash (hardware controller or software controller)
- CompactFlash on EMIFS interface
- Internal static RAM

OMAP5912 Multimedia Processor Interrupts Reference Guide (literature number SPRU757) describes the interrupts of the OMAP5912 multimedia processor. Three level 2 interrupt controllers are used in OMAP5912:

- One MPU level 2 interrupt handler (also referred to as MPU interrupt level 2) is implemented outside of OMAP3.2 and can handle 128 interrupts.
- One DSP level 2 interrupt handler (also referred to as DSP interrupt level 2.1) is instantiated outside of OMAP3.2 and can handle 64 interrupts.
- One OMAP3.2 DSP level 2 interrupt handler (referenced as DSP interrupt level 2.0) can handle 16 interrupts.

OMAP5912 Multimedia Processor Peripheral Interconnects Reference Guide (literature number SPRU758) describes various periperal interconnects of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Timers Reference Guide (literature number SPRU759) describes various timers of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Serial Interfaces Reference Guide (literature number SPRU760) describes the serial interfaces of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Universal Serial Bus (USB) Reference Guide (literature number SPRU761) describes the universal serial bus (USB) host on the OMAP5912 multimedia processor. The OMAP5912 processor provides several varieties of USB functionality. Flexible multiplexing of signals from the OMAP5912 USB host controller, the OMAP5912 USB function controller, and other OMAP5912 peripherals allow a wide variety of system-level USB capabilities. Many of the OMAP5912 pins can be used for USB-related signals or for signals from other OMAP5912 peripherals. The OMAP5912 top-level pin multiplexing

controls each pin individually to select one of several possible internal pin signal interconnections. When these shared pins are programmed for use as USB signals, the OMAP5912 USB signal multiplexing selects how the signals associated with the three OMAP5912 USB host ports and the OMAP5912 USB function controller can be brought out to OMAP5912 pins.

OMAP5912 Multimedia Processor Multi-channel Buffered Serial Ports (McBSPs) Reference Guide (literature number SPRU762) describes the three multi-channel buffered serial ports (McBSPs) available on the OMAP5912 device. The OMAP5912 device provides multiple high-speed multichannel buffered serial ports (McBSPs) that allow direct interface to codecs and other devices in a system.

OMAP5912 Multimedia Processor Camera Interface Reference Guide (literature number SPRU763) describes two camera inerfaces implemented in the OMAP5912 multimedia processor: compact serial camera port and camera parallel interface.

OMAP5912 Multimedia Processor Display Interface Reference Guide (literature number SPRU764) describes the display interface of the OMAP5912 multimedia processor.

- LCD module
- LCD data conversion module
- LED pulse generator
- Display interface

OMAP5912 Multimedia Processor Multimedia Card (MMC/SD/SDIO) (literature number SPRU765) describes the multimedia card (MMC) interface of the OMAP5912 multimedia processor. The multimedia card/secure data/secure digital IO (MMC/SD/SDIO) host controller provides an interface between a local host, such as a microprocessor unit (MPU) or digital signal processor (DSP), and either an MMC or SD memory card, plus up to four serial flash cards. The host controller handles MMC/SD/SDIO or serial port interface (SPI) transactions with minimal local host intervention.

OMAP5912 Multimedia Processor Keyboard Interface Reference Guide (literature number SPRU766) describes the keyboard interface of the OMAP5912 multimedia processor. The MPUIO module enables direct I/O communication between the MPU (through the public TIPB) and external devices. Two types of I/O can be used: specific I/Os dedicated for 8 x 8 keyboard connection, and general-purpose I/Os.

OMAP5912 Multimedia Processor General-Purpose Interface Reference Guide (literature number SPRU767) describes the general-purpose in-

terface of the OMAP5912 multimedia processor. There are four GPIO modules in the OMAP5912. Each GPIO peripheral controls 16 dedicated pins configurable either as input or output for general purposes. Each pin has an independent control direction set by a programmable register. The two−edge control registers configure events (rising edge, falling edge, or both edges) on an input pin to trigger interrupts or wake−up requests (depending on the system mode). In addition, an interrupt mask register masks out specified pins. Finally, the GPIO peripherals provide the set and clear capabilities on the data output registers and the interrupt mask registers. After detection, all event sources are merged and a single synchronous interrupt (per module) is generated in active mode, whereas a unique wake−up line is issued in idle mode. Eight data output lines of the GPIO3 are ORed together to generate a global output line at the OMAP5912 boundary. This global output line can be used in conjunction with the SSI to provide a CMT−APE interface to the OMAP5912.

OMAP5912 Multimedia Processor VLYNQ Serial Communications Interface Reference Guide (literature number SPRU768) describes the VLYNQ of the OMAP5912 multimedia processor.

VLYNQ is a serial communications interface that enables the extension of an internal bus segment to one or more external physical devices. The external devices are mapped into local, physical address space and appear as if they are on the internal bus of the OMAP 5912. The external devices must also have a VLYNQ interface. The VLYNQ module serializes bus transactions in one device, transfers the serialized data between devices via a VLYNQ port, and de-serializes the transaction in the external device.

OMAP5912 includes one VLYNQ module connected on OCPT2 target port and OCPI initiator port. These connections are configured via a static switch, which selects either SSI or VLYNQ module. This switch, forbids the simultaneous use of GDD/SSI and VLYNQ. The switch is controlled by the VLYNQ_EN bit in the OMAP5912 configuration control register (CONF_5912_CTRL).

OMAP5912 Multimedia Processor Pinout Reference Guide (literature number SPRU769) provides the pinout of the OMAP5912 multimedia processor. After power-up reset, the user can change the configuration of the default interfaces. If another interface is available on top of the default, it is possible to enable a new interface for each ball by setting the corresponding 3-bit field of the associated FUNC_MUX_CTRL register. It is also possible to configure on-chip pullup/pulldown. This document

also describes the various power domains so that the user can apply the different interfaces seamlessly with external components.

OMAP5912 Multimedia Processor Window Tracer (WT) Reference Guide (literature number SPRU770) describes the window tracer module used to capture the memory transactions from four interfaces: EMIFF, EMIFS, OCP-T1, and OCP-T2. This module is located in the OMAP3.2 traffic controller (TC).

OMAP5912 Multimedia Processor Real-Time Clock Reference Guide (literature number SPRUxxx) describes the real-time clock of the OMAP5912 multimedia processor. The real-time clock (RTC) block is an embedded real-time clock module directly accessible from the TIPB bus interface.

Trademarks

Contents

Figures

Tables

Introduction

This document introduces the setup, components, and features of the OMAP5912 multimedia processor and provides a high-level view of the device architecture.

1 Overview

OMAP5912 is the next generation of OMAP processors and succeeds the Texas Instruments OMAP5910 processor. It is built with TI 130-nm process technology and has dual input/output voltage (1.8 V−3.3 V) capability. Like its predecessor, it is fully programmable to perform one or more of the following personal communication system tasks:

- Call manager
- Mail/fax reader/composer
- Internet access
- Personal digital assistant (PDA)
- Personal information management (PIM)
- Multimedia engines

This system is optimized for various multimedia codecs, such as:

- MPEG4
- MP3
- JPEG
- H.263

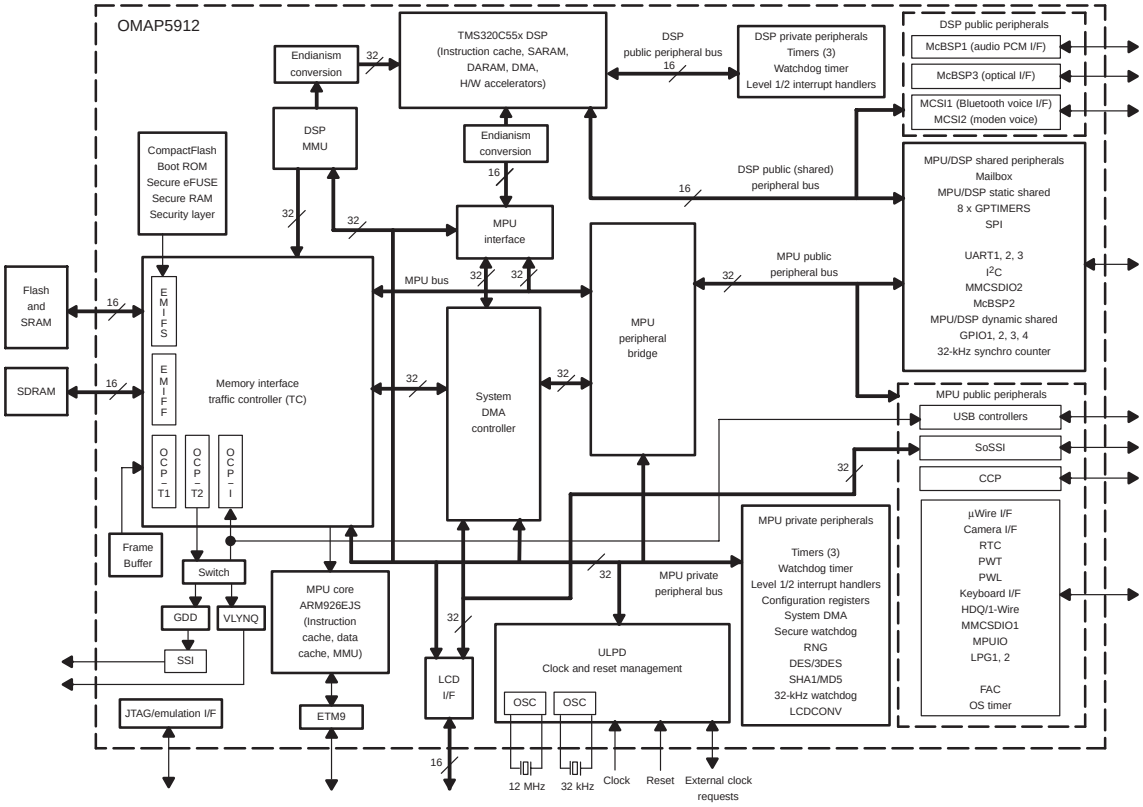
Furthermore, it is capable of running advanced speech applications:

- Text to speech
- Speech recognition

The OMAP5912 also supports secure operations.

Figure 1 shows OMAP5912 at the top level.

Figure 1. Top-Level OMAP5912



The OMAP5912 includes the microprocessor unit (MPU) subsystem, the digital signal processor (DSP) subsystem, the system direct memory access (SDMA), and part of the L3 and L4 interconnects. All other components of OMAP5912 are associated with the OMAP gigacell revision 3.2.

The OMAP5912 device provides a secure environment to run secure applications. The security hardware components include:

- OMAP3.2 gigacell security state machine
- Dedicated secure RAM
- Secure boot ROM
- Secure electrical fuse (eFuse) components
- Secure peripherals

1.1 Subsystems

The OMAP5912 chip subsystems have these characteristics:

- The MPU subsystem is based on the ARM926EJ core, which performs most operations on the chip.
- The DSP subsystem is based on a TMS320C55x™ DSP core, which performs intensive data computing tasks.
- The internal memory subsystem (frame buffer) is a single-port SRAM .
- The system DMA helps the MPU perform data memory transfer-specific tasks, leaving a million more instructions available per second (MIPS) for both processors.
- The system components (ULPD, oscillators, etc.) are responsible for managing system interactions, such as interrupt, clock control, and idle.
- The memory interfaces define the system memory access organization of OMAP5912.
- The peripheral subsystem defines all of the components used to interface OMAP5912 with specific external devices, such as compact camera port (CCP), keyboard, and smart display.
- The security modules include ROM, a single port SRAM, eFuse cells, and secure peripherals, which enable the system to support secure applications.
- The universal serial bus (USB) and synchronous serial interconnect (SSI) interface enable the platform to support a universal serial link and a dedicated modem interface, which in turn enable a high data transfer rate between the modem and the OMAP5912.
- The device features two types of interconnectivity:
  - The L3 interconnect is the memory interconnection among the MPU, DSP, DMA, and an external host (through USB or SSI interface).
  - The L4 interconnect provides all of the peripheral interconnections among the MPU, DSP, and DMA.

1.2 SDRAM/DDRAM Memory Interface

The OMAP5912 external memory interface fast (EMIFF) is an synchronous dynamic RAM (SDRAM) controller that manages all accesses by the various initiators of an OMAP system. It can support one 16-bit device or two 8-bit devices. The external interface data bus width is always 16 bits.

The following devices are supported:

- Standard single-data-rate SDRAM
- Low-power single-data-rate SDRAM (mobile SDRAM)
- Standard double-data-rate SDRAM (DDRAM)
- Low-power double-data-rate SDRAM (mobile DDRAM)

In terms of capacity and organization of the memory components that can be attached, the EMIFF can handle:

- 1G-bit, 512M-bit, 256M-bit, 128M-bit, 64M-bit, and 16M-bit devices
- 2-bank or 4-bank devices for 16M-bit or 64M-bit devices
- 4-bank devices only for any other capacity
- x8 (two devices) or x16 (single device) data bus configuration, except for the 1G-bit device. The EMIFF supports only two x8, 512M-bit devices and a single x16, 1G-bit device (128 megabytes, maximum memory configuration).

1.3 Synchronous/Asynchronous Burst Memory Interface

The synchronous/asynchronous external memory interface slow (EMIFS) supports most common memory interface protocols through flexible programming and timing signals control.

The EMIFS controls up to six devices without adding any external logic through six independent chip-selects (CSs) and through dedicated memory interface control signals. Two configurable options are supported:

1.4 CompactFlash

OMAP5912 supports CompactFlash with a dedicated compact flash controller.

1.5 Other Memory Types

OMAP5912 includes multiple serial interfaces that can be used to connect multiple types of memories, including:

- Up to two multimedia cards through the multimedia memory card interfaces (MMC/SDIO)

1.6 Serial Interfaces

OMAP5912 provides several serial interfaces:

- A master/slave serial port interface (SPI), enabling serial data transfer compatible with common SPI protocol
- Three multichannel buffered serial port (McBSP) peripherals, enabling emulation of the following serial protocols:
  - Serial port interface
  - An I2S-compatible interface
  - A DSP serial input/output
  - Multichannel time division multiplexer
- An I^2C with multimaster and slave support
- A multichannel and full duplex serial synchronous interface (SSI)
- A universal serial bus (USB) interface client, host, and USB On-The-Go configuration
- A serial µWire interface
- Two multichannel serial interfaces (MCSI)
- Three universal asynchronous transceivers (UART1, UART2, UART3)
- HDQ and 1-Wire interfaces

1.7 Camera and Display Interfaces

There are three video interfaces:

- The compact camera port (CCP) interface is a unidirectional serial interface that connects to a camera.
- The camera IF is an 8-bit parallel interface with horizontal/vertical camera input signal.
- Two types of LCD display IFs:
  - An 8-bit/16-bit configurable parallel bidirectional display interface that enables a smart display connection using the MeSSI standard
  - The OMAP internal LCD controller that enables simple LCD interfaces and control

1.8 Emulation and Test Interfaces

OMAP5912 has these emulation and test interface features:

- JTAG emulation capabilityfor non-realtime debug
- You can use a window tracer (WT) module to capture the memory transactions from four interfaces: EMIFF, EMIFS, OCP-T1, and OCP-T2. The captured information is sent outto the serial tracer interface (STI) for processing for debugging purposes. For more detail on this memory transaction trace, see Appendix E, *Window Tracer*.
- An embedded trace macrocell (ETM) to enable real-time trace of the MPU subsystem operations
- Concurrent DSP and MPU emulation

1.9 General-Purpose Interfaces

OMAP5912 includes dedicated modules to support specific modulation and control of general-purpose inputs and outputs:

- General-purpose input/output (GPIO) modules allow the monitoring and control of OMAP5912 device input/output pins with event detection for external interrupts support.
- The pulse width tone (PWT) module provides OMAP5912 output pins with waveform control for tone generation.
- The pulse width light (PWL) module allows OMAP5912 output pins waveform control for brightness light control.
- Two light pulse generator (LPG) modules allow OMAP5912 output pins waveform control for LED blinking control.
- Three general-purpose timers enable OMAP5912 output pins waveform control for pulse width modulation support and OMAP5912 input pins events detection with timing stamp.

1.10 Interfaces Multiplexing

Because of the device I/O limitations, the interfaces are not accessible at the same time, and some of them are multiplexed. For more detail on I/O multiplexing and control, see Chapter 22, *Pinout*.

1.11 OMAP5912 Interface Usage Examples

This section contains three examples of OMAP5912 usage:

- Audio interface
- Bluetooth™ interface
- Modem connection

1.11.1 Audio Interfaces

OMAP5912 supports a modem audio PCM codec interface, a modem voice interface, and a Bluetooth™ voice interface simultaneously. These interfaces can be provided through various OMAP5912 peripherals:

- McBSP, which is compliant with the I^2C standard and can be connected to an audio codec.
- MCSI, which can be used as voice interface (see Section 1.11.2 and Section 1.11.3)

1.11.2 Bluetooth Interface

The Bluetooth interface has the following data, voice, and power management interfaces:

- The data interface can be provided by UART1.
- The voice interface can be provided by MCSI1.
- The power interface is the clock request pins.

1.11.3 Modem Connection

Two examples propose ways to connect a modem to the OMAP5912 device:

- Using a CMT-APE interface: the SSI exchanges data between the modem and OMAP5912. The control uses GPIO pins.
- Using data, control, voice, and power management interfaces:
  - Data interface can be the McBSP2.
  - Control interface can be the UART2.
  - Power management interface is the clock request pins.
  - Voice interface can be the MCSI2.

2 OMAP5912 Description

Figure 2 shows the OMAP5912 in detail.

Figure 2. OMAP5912 Top-Level Detailed Diagram

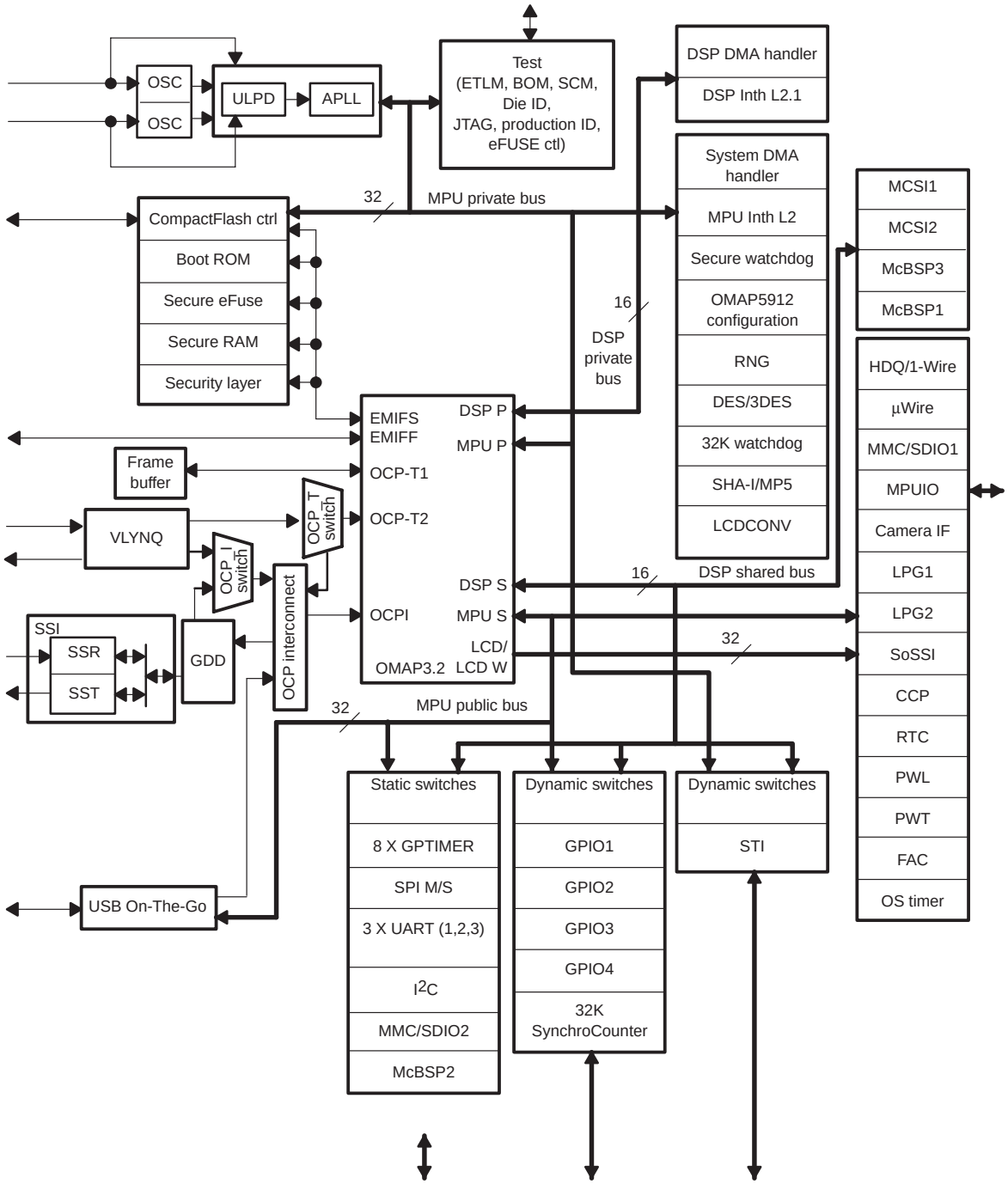


Table 1 through Table 8 describe the OMAP5912 top-level partitioning. For more detail on the OMAP5912 core, see Chapter 2, *OMAP3.2 Subsystem*.

Table 1. MPU Private Peripherals

Peripheral Name	General Description
MPU level 2 interrupt handler	There is one MPU interrupt handler outside the OMAP 3.2 gigacell. It supports up to 128 interrupt lines. Both IRQ and FIQ are connected to MPU interrupt handler level 1.
32-kHz watchdog	A general-purpose timer with watchdog functionality clocked at 32 kHz
Secure watchdog	Same watchdog module used for security and clocked differently
ULPD	System clock and reset module. Manages the idle mode, generates main reset to OMAP. Some peripherals clock through its APLL output to generate the OMAP input system clock.
CompactFlash controller	This module interfaces with the NOR flash memory to enable CompactFlash card support.
RNG	Random number generator module
DES/3DES	Data encryption standard and triple DES
SHA−1/MD5	Secure module that supports SHA−1 and MD5 secure hash algorithm
GDMA handler	64 DMA requests are multiplexed to generate 31 DMA request lines to the system DMA.
OMAP5912 configuration	Controls the various configurations of the OMAP5912 functions (I/O functions, debug functions, clock control, security control register)
LCDCONV	Enables conversion of the OMAP LCD 16-bit data bits into 18-bit data bits

Table 2. DSP Private Peripherals

Peripheral Name	General Description
DSP DMA handler	DMA requests are multiplexed to generate 19 DMA request lines to the DSP DMA.
DSP level 2.1 interrupt handler	There is one DSP interrupt handler outside the OMAP 3.2 gigacell. It supports up to 64 interrupt lines. Only IRQ is connected to the DSP level 1 interrupt handler. The DSP interrupt handler module is derived from the MPU interrupt handler module.

Table 3. MPU Shared Peripherals

Peripheral Name	General Description
USB On-The-Go	A combination of the USB client and USB host used in OMAP5910, including an additional module (OTG module)
RTC	An embedded real-time clock module
µWire	A serial interface that connects external devices such as EEPROM or LCD with µWire standard
HDQ/1-Wire	Implements the hardware protocol of the master function of the Benchmark HDQ and the Dallas Semiconductor 1-Wire protocol
MPUIO	Provides MPU GPIOs (general-purpose I/Os) and a keypad interface
Memory stick interface	Memory storage device
FAC	Frame adjustment counter
PWT	Pulse width time
PWL	Pulse width length
2 X LPG	Two instances of a light pulse generation (LPG) module
OS timer	32-kHz OS timer used in OMAP5910
SoSSI	A parallel display interface for smart displays. The output implements one 8-bit or 16-bit parallel bus between display driver and master. There are five control pins and one tearing effect pin. The bus is bidirectional.
MMC/SDIO1	Multimedia card interface, compatible with secure digital memory card interface version 1.0

Table 4. DSP Shared Peripherals

Peripheral Name	Description
McBSP1	Can be used to interface with I2S audio codec
McBSP3	Can be used as optical audio interface
2 X MCSI	Two instances of an MCSI module

All of the DSP shared peripherals are also accessible by the MPU through the OMAP 3.2 gigacell MPUI internal port.

Table 5. MPU/DSP Statically Shared Peripherals

Peripheral Name	Description
MMC/SDIO2	Multimedia card interface, compatible with secure digital memory card interface version 1.0
I^2C multimaster/slave	A generic serial link interface that controls several OMAP5912 external I^2C slave devices or slaves them to other OMAP5912 external I^2C master devices
SPI master/slave	A master/slave serial port interface, running up to 19.2M bits/s in master or slave mode
8x dual-mode timers	All of these general-purpose timers can be configured to count either from the 32-kHz clock or from the system clock. Three of them are also used to provide two PWM signals at the OMAP5912 boundary.
McBSP2	Used as communication processor data interface or I2S emulator
UART1	A UART modem including autobaud. The maximum baud rate is 3.6M bits/s.
UART2	A UART modem with autobaud, as in UART1
UART3	A UART IrDA enabling slow, medium, and fast configurations

Each host peripheral access can be configured by software as the DSP or the MPU for each peripheral.

Table 6. MPU/DSP Dynamically Shared Peripherals

Peripheral Name	Description
4x GPIOs	There are four instances of 16-bit module general-purpose I/Os. Each module can generate its own interrupt. Several general-purpose I/Os are multiplexed with primary I/Os, some of which are connected directly with dedicated I/Os.
32-kHz synchronization counter	This is a simple upward counter clock used by the 32-kHz input clock to enable synchronization between modem and application chips when OMAP5912 is used in conjunction with a modem having the same clock input. As soon as the power-up reset input is released, the counter starts counting.

Each host peripheral access can be either the DSP or the MPU; the arbitration is performed in hardware.

Table 7 lists the dedicated modules available in OMAP5912, a general description of each module, and their corresponding functional specifications reference.

Table 7. OMAP5912 Dedicated Modules

Module Name	Description
SSI	A synchronous serial interface that comprises the synchronous serial transmitter (SST) and the synchronous serial receiver (SSR)
SSR	Receive part of the SSI The OMAP5912 SSR can process up to 8 channels.
SST	Transmit part of the SSI The OMAP5912 SST can process up to 8 channels.
GDD	A generic distributed DMA attached to the SSI modules in OMAP5912. It performs DMA transfer between the SSI and the various subsystems (internal memory or external memory).
OCP interconnect	A synchronous interface enabling interconnection between the USB and the GDD with the OCP-I port
Boot ROM	A 64K-byte ROM accessible only by the MPU. Part of it can be accessed in secure mode.
Secure RAM	A 16K-byte SRAM accessible only in secure mode by the MPU
Secure eFuse	Includes two chains of 128-bit eFuse and 256-bit eFuse and the eFuse controller
Security layer	Prevents the secured access from appearing at the OMAP5912 boundary via the flash interface
CCP	Synchronous serial camera interface without frame memory
Camera interface	Parallel interface that connects an external camera sensor
Frame Buffer	This 256K−byte SRAM is used to store video frame before emission by the DMA LCD channel to the external or internal LCD controller

Table 8 includes the test modules used in OMAP5912, a general description of each module, and their corresponding functional specifications references.

Table 8. OMAP5912 Test Modules

Module Name	Description
ETLM	Emulation-compatible support for multiple TAP controllers.
SCM	Scan chain linking. Enables the combination of several scan chains in parallel or serially.
JTAG	The OMAP5912 1149.1 TAP controller IEEE compliant used to program the functional test modes
Die ID	A 64-bit register composed of eFuse cells programmed during fabrication process. Each die ID is unique, as it is created from the wafer, lot, and factory numbers, plus the die x/y coordinates in the wafer.
Production ID	An additional 64-bit register of eFuse cells used to include specific OMAP5912 needs. One R&D eFuse to distinguish the OMAP5912 production version from the prototype version.
Global eFuse controller	OMAP3.2 gigacell eFuse controller. This eFuse controller is closely coupled with the gigacell. It is mainly used to control its internal dedicated eFuses for memory repair and enables access to the die ID registers.
BCM	BIST combiner module. Manages OMAP5912 BIST controller from the MPU or TAP controller

2.1 OMAP3.2 Gigacell

The OMAP 3.2 gigacell used by OMAP5912 is called OMAP 3.2.

This gigacell features:

- ARM926EJ core including:
  - ARM926EJS, supporting multiple operating systems (Symbian, Linux, WinCE, and others)
  - MMU with translation lookaside buffer (TLB)
  - L1 16K-byte, four-way set-associative instruction cache
  - L1 8K-byte, four-way set-associative data cache with write buffer
- MPU level 1 interrupt handler
- Embedded trace macrocell module, ETM version 2.a in 13-bit mode configuration or in 17-bit demultiplexed mode configuration

- TMS320C55x™ (C55x™) DSP including:
  - Embedded ICE emulator interface through JTAG port
  - C55x™ DSP rev 2.11
  - L1 cache (24K bytes)
    - 16K-byte, two-way set-associative instruction cache
    - 2x 4K-byte RAM set for instruction
  - DARAM 64K-byte, zero wait state, 32-bit organization
  - SARAM 96K-byte, zero wait state, 32-bit organization
  - PDROM (32K bytes)
  - DMA controller: 6 physical channels, 5 ports
  - DSP trace module
  - Hardware accelerators motion estimation (ME), discrete/inverse discrete cosine transform (DCT/IDCT) and pixel interpolation (PI)
  - DSP level 1 interrupt handler in the C55x™ DSP core
- DSP MMU
- DSP level 2 interrupt handler, which enables connection to 16 additional interrupt lines outside OMAP. The priority of each interrupt line is controlled by software.
- DSP interrupt interface, which enables connection to the interrupt lines coming out of the level 2 interrupt handler and the interrupt lines requiring more priority. The outcome interrupt of this module is then connected to the C55x DSP to be processed. This module is mainly used to ensure that all interrupts going to the DSP are level-sensitive.
- DSP peripherals:
  - 3x 16-bit DSP private timers
  - 1x 16-bit DSP private watchdog
- Mailboxes:
  - Four mailboxes are implemented:
    - Two read/write accessible by MPU, read-only by the DSP
    - Two read/write accessible by the DSP, read-only by the MPU

    Each mailbox is implemented with 2x 16-bit registers. When a write is done into a register by one processor, it generates an interrupt, released by the read access of the other processor.

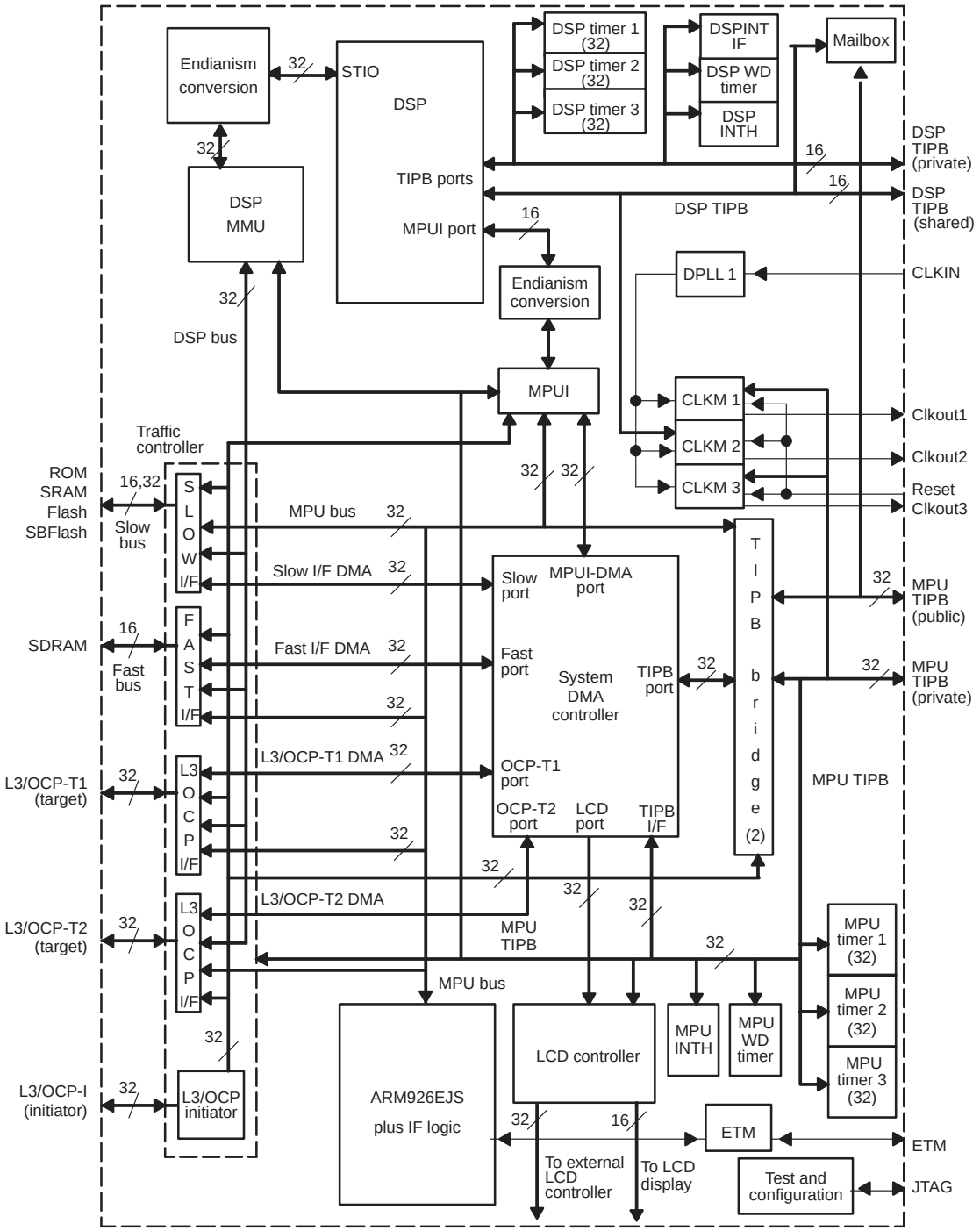
- MPU peripherals
  - 3x 32-bit private timers; their clock is either the OMAP3.2 reference input clock or the divided MPU clock.
  - 1x 16-bit private watchdog; can be configured as a 16-bit general-purpose timer by software. Its clock is the OMAP3.2 reference input clock divided by 14.
- External LCD controller support in addition to the OMAP LCD controller
  - LCD controller with its own tearing effect logic
- Memory traffic controller
  - External memory interface fast (EMIFF) is a memory interface that enables 16-bit data SDRAM memory access.
  - External memory interface slow (EMIFS); connects external device memories (such as common flash and SRAM memories). This interface is also used internally to connect the boot ROM, the secure RAM, and the secure eFuse components that are partially accessible in secure mode.

L3 OCP-T1 and L3 OCP-T2 ports are provided to enable memory access from OMAP 3.2 gigacell on a standard basis protocol. Only the L3 OCP T1 is used in OMAP5912 to access the frame buffer.

- Emulator interface through JTAG port
- System DMA which consists of:
  - Seventeen logical channels
  - Seven physical ports + one for configuration
  - Four physical channels

  The ports are connected to the L3 OCP targets, the external memory, the TIPB bridge, the MPUI, and one dedicated port connected to an LCD controller. The system DMA controller can be controlled via the MPU private TIPB or by an external host via the OCP-I port.
- One DPLL for the following clock domains:
  - MPU traffic controller clock domain
  - DSP clock domain
- Endianism conversion for DSP
  - The DSP uses big-endian format, whereas the MPU uses little-endian format. Also, as a rule, the OMAP5912 chip works in little endian. Thus, the endianism conversion is useful for all memory or peripheral accesses from on-chip peripherals or all shared memories to the C55x DSP.

Figure 3. OMAP3.2 Gigacell



2.2 OMAP5912 Module Descriptions

2.2.1 Clock and Reset Generation

32-kHz Oscillator

The 32-kHz oscillator uses a 32-kHz external quartz. It can be used to generate the 32-kHz clock on the chip. It can be disabled when the 32-kHz is an external source provided by an external clock.

12-MHz Oscillator

This oscillator is to be used with a 12-MHz or 13-MHz external quartz.

The 12- or 13-MHz oscillator allows the generation of the 48 MHz the USB requires. The APLL located in the ULPD provides the x4 factor. The 12- or 13-MHz clock is used as the input clock of the ULPD. The 12- or 13-MHz oscillator is on during awake and big sleep modes. It is off during deep sleep mode. The wake-upsequence is handled from the power management module.

Ultralow-Power Device (ULPD)

The ULPD generates and manages clocks and reset signals to OMAP and to some peripherals. It controls chip-level power-down modes and handles chip-level wake-up events.

The ULPD can handle the high-frequency oscillator on/off sequences, when used, and provides the resets to OMAP. It also allows configuration of the clock sources of the OMAP5912 device and management of the APLL interface.

OMAP5912 Clock and Reset Architecture

The global clock distribution and the reset distribution scheme are described in Chapter 5.

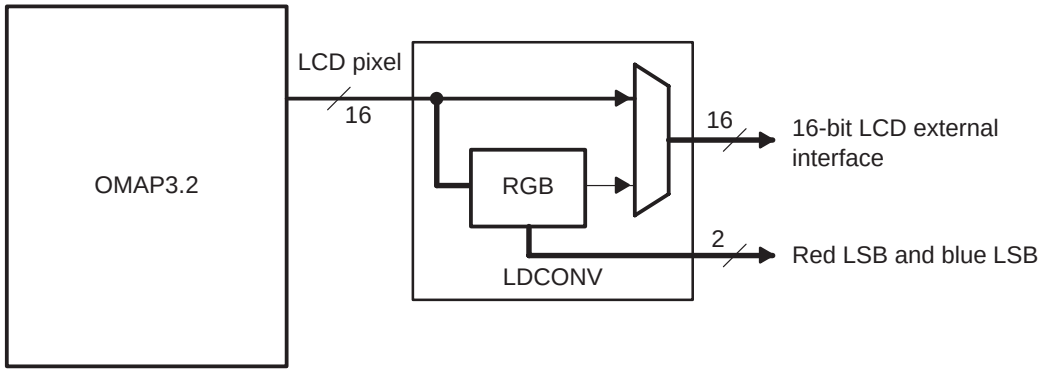
2.2.2 Peripherals Subsystem

LDCONV

This module provides 16-bit to 18-bit LCD data conversion to the OMAP3.2 LCD interface. It supports two operating modes:

- The 16-bit LCD
- The 18-bit LCD

Figure 4. LDCONV Integration

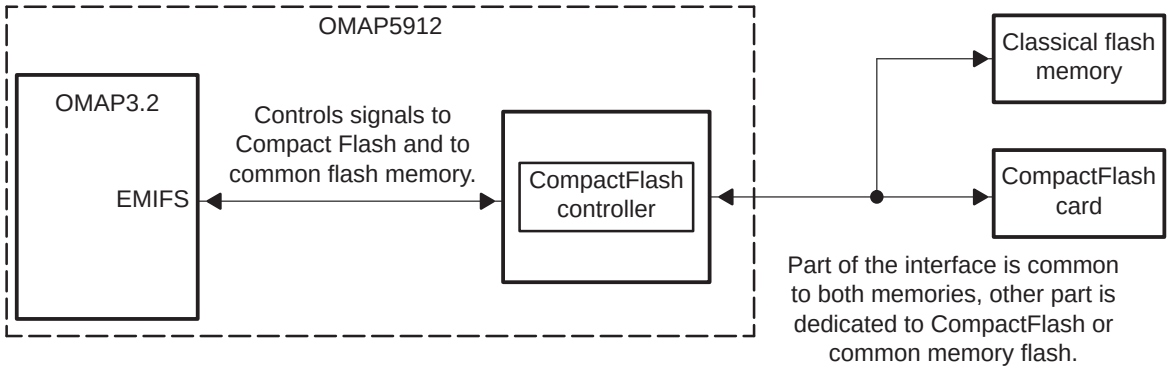


CompactFlash Controller

The CompactFlash controller interfaces a CompactFlash module and a classical memory interface. Control signals from the memory interface are processed through the CompactFlash controller to drive the CompactFlash module, and control signals from CompactFlash are processed to perform a data transfer to the memory interface. Some pins are shared with common flash memory, such as address bus, data bus, and control pins. The CompactFlash controller includes a bypass mode used when common flash memory, rather than a CompactFlash card, is connected.

The CompactFlash is connected on the EMIFS port of the OMAP3.2 gigacell.

Figure 5. CompactFlash Interface



Note: The CompactFlash controller can only be used when the system clock frequency is 12 MHz.

Frame Adjustment Counter (FAC)

The FAC consists of a frame synchronization capture pin (intended to count the number of rising edges of a frame synchronization signal from a synchronous serial port) and a frame start capture pin (intended to count the number of rising edges of a USB frame start signal). The respective count values can then be used by system software to adjust the duration of the two time domains with respect to each other in order to reduce the overflow and underflow.

The FAC module is used to control the number of McBSP2 and USB frames.

Light Pulse Generator (LPG)

The LPG module is used to control the blinking period of an external LED.

SHA-1/MD5 Accelerator

The SHA-1/MD5 security module provides hardware-accelerated hash functions. It can run either the SHA-1 algorithm in compliance with FIPS 180-1 standard or the MD5 message-digest algorithm developed by Rivest.

DES_3DES Accelerator

The DES_3DES module provides hardware-accelerated data encryption/decryption functions. It can run either the single DES algorithm or the triple DES algorithm in compliance with the FIPS 46−3 standard. It supports the electronic codebook (ECB) and cipher block chaining (CBC) modes of operation. It does not support the cipher feedback (CFB) or the output feedback (OFB) modes of operation in hardware.

Random Number Generator

The random number generator (RNG) module provides a true, nondeterministic noise source to generate keys, initialization vectors (IVs) and other random number requirements. It is designed for FIPS 140-1 compliance, facilitating system certification to this security standard. An ANSI X9.17, annex C postprocessor is available to meet the NIST requirements of FIPS 140-1.

Camera Interface

The camera interface supports 8-bit parallel image data port and horizontal/vertical signal ports separately (stand alone synchronization method).

Real-Time Counter

The real-time counter (RTC) block is an embedded real-time counter module, directly accessible from the MPU.

Basic functionalities of the RTC block are:

- Time information (seconds/minutes/hours) directly in BCD code
- Calendar information (day/month/year/day of the week) directly in BCD code, up to year 2099
- Interrupt generation, periodically (1s/1m/1h/1d period) or at a precise time of the day (alarm function)
- 30-s time correction: oscillator frequency calibration using reference clock input
- Can be used in standalone when rest of the chip is powered down

OS Timer

A programmable interval timer is required to generate a periodic interrupt, also called a system clock tick, to the OS. This is used to keep track of the current time and to control the operation of device drivers.

Pulse Width Time (PWT)

This module generates a modulated frequency signal for the external buzzer. Frequency is programmable between 349 Hz and 5276 Hz with 12 half-tone frequencies per octave. The volume is also programmable.

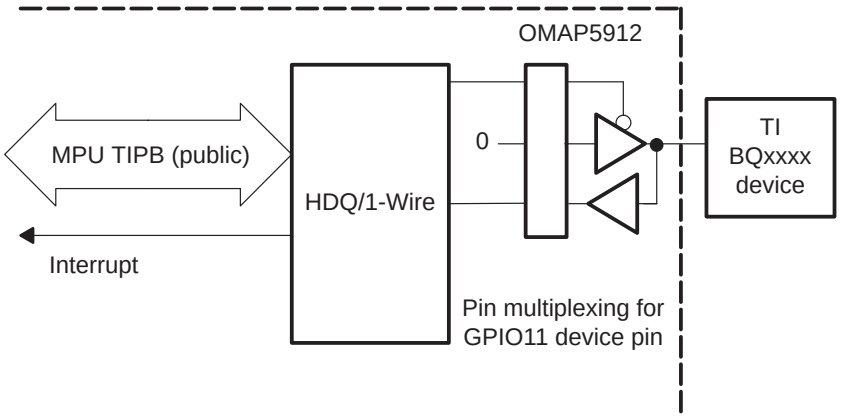
Pulse Width Length (PWL)

This module allows the control of the backlight of the LCD and the keypad by employing a 4096-bit random sequence. This voltage level control technique decreases the spectral power at the modulator harmonic frequencies. The block uses a switchable 32-kHz clock.

HDQ/1-Wire Battery Monitoring Serial Interface

The HDQ/1-Wire battery monitoring serial interface module implements the hardware protocol of the master function of Benchmark's HDQ and Dallas Semiconductor's 1-Wire protocol.

Figure 6. HDQ/1-Wire Overview

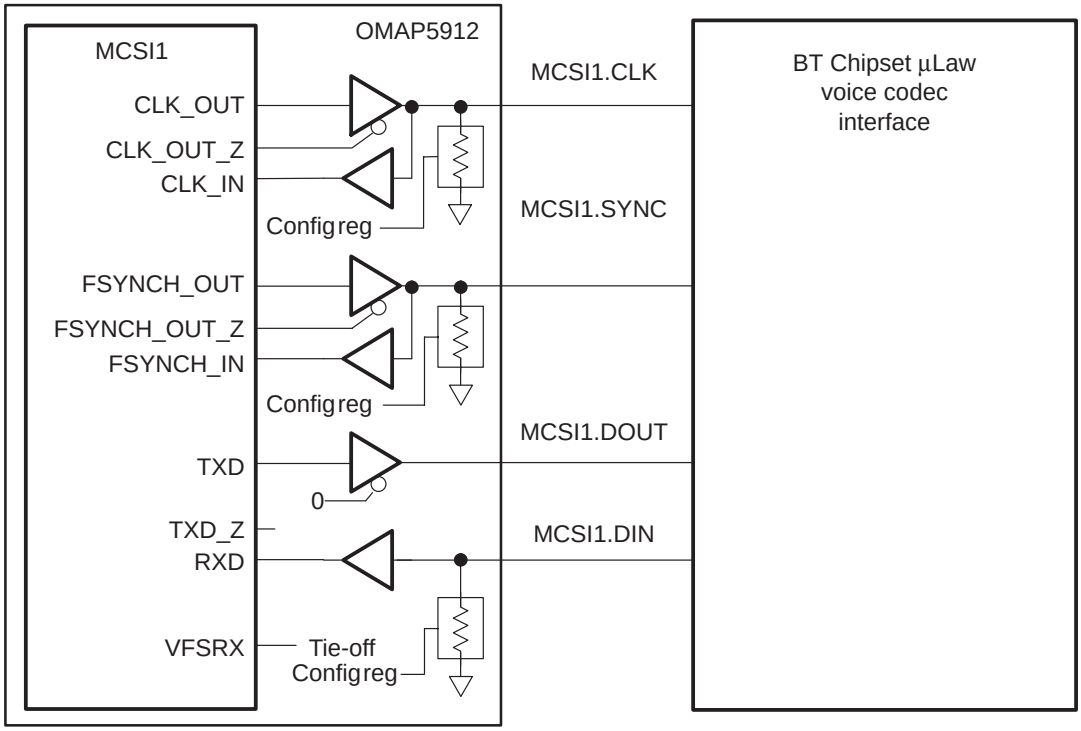


MCSI1

The MCSI1 is a multichannel serial interface, half-duplex, master/slave. MCSI1 can be used in OMAP5912 to interface the Bluetooth voice module.

In this case, the interface is an 8-kHz frame serial port, 8-bit data transfer. It is a 4-wire interface with a bidirectional serial clock and frame synchronization. If the Bluetooth baseband device is not synchronous with the modem network, the Bluetooth voice interface receives the clock and frame synchronization from OMAP5912 MCSI1.

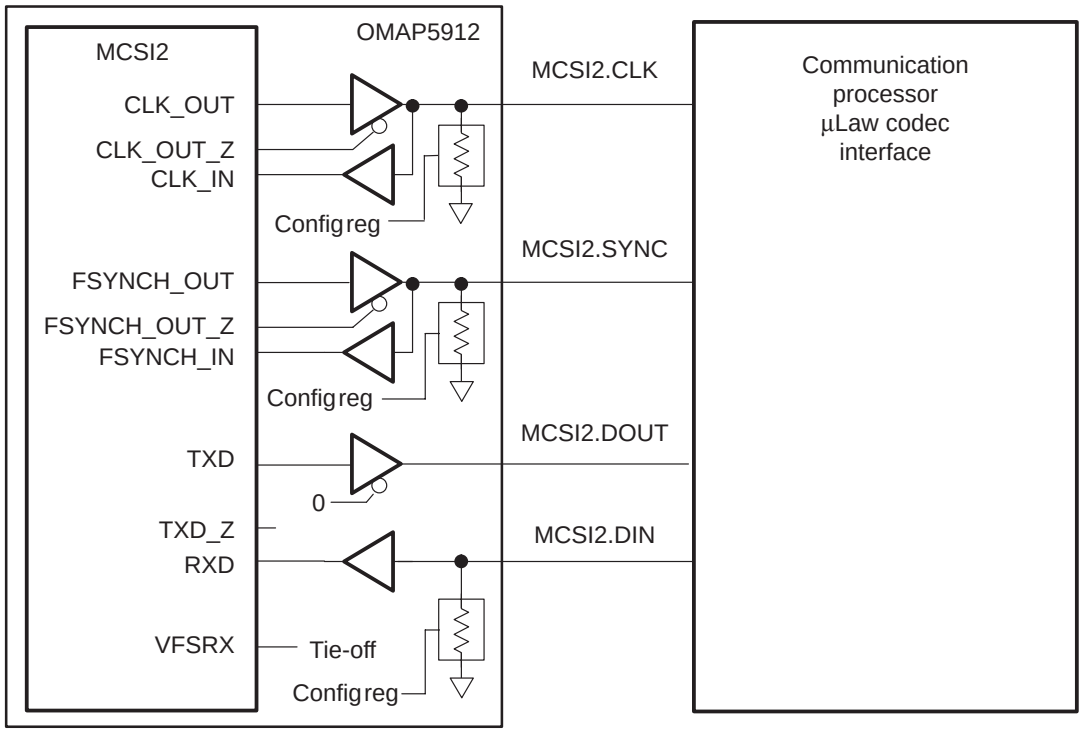
Figure 7. MCSI1 Interface



MCSI2

The MCSI2 is a multichannel serial interface, half-duplex, master/slave. MCSI2 can be used in OMAP5912 to interface the modem voice module. In this case, it is an 8-kHz frame serial port, 8-bit data transfer. Clock and frame synchronization are bidirectional.

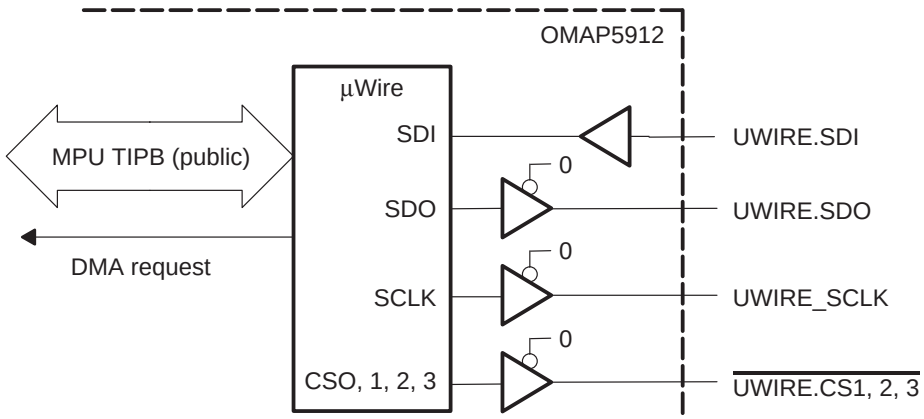
Figure 8. MCSI2 Interface



µ*Wire*

The µWire module is a serial interface to drive external devices like EEPROM or LCD with µWire standard.

Figure 9. µWire Interface



MPU GPIOs

The MPU has a 16-bit GPIO (MPUIO) with programmable debouncing circuit. It can handle maskable interrupt generation on high-to-low or low-to-high transition on pins configured as input.

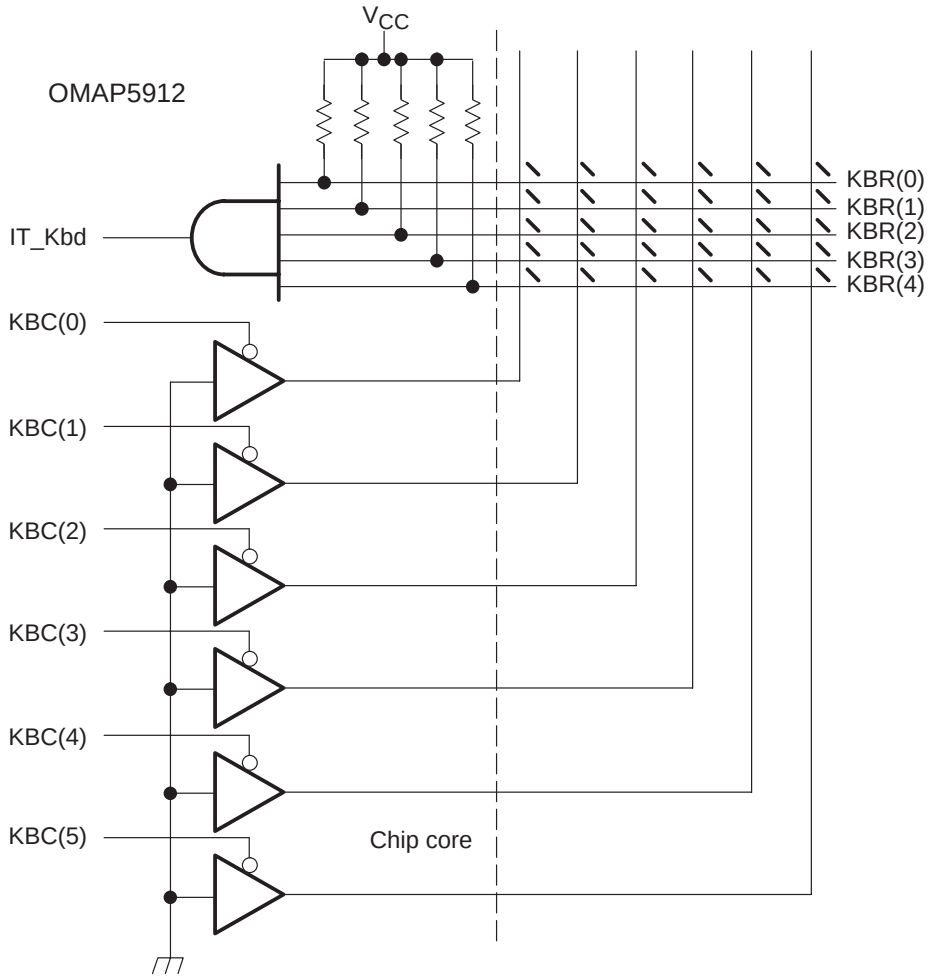
Keyboard

The keyboard interface is a 6 x 5 or 8 x 8 interface. Figure 10 shows the 6 x 5 interface.

The keyboard is connected to the chip using:

- ❏ KBR (7:0) input pins for row lines
- ❏ KBC (7:0) output pins for column lines

Figure 10. 6x5 Keyboard Connection



UARTs

There are three identical UART modules in OMAP5912:

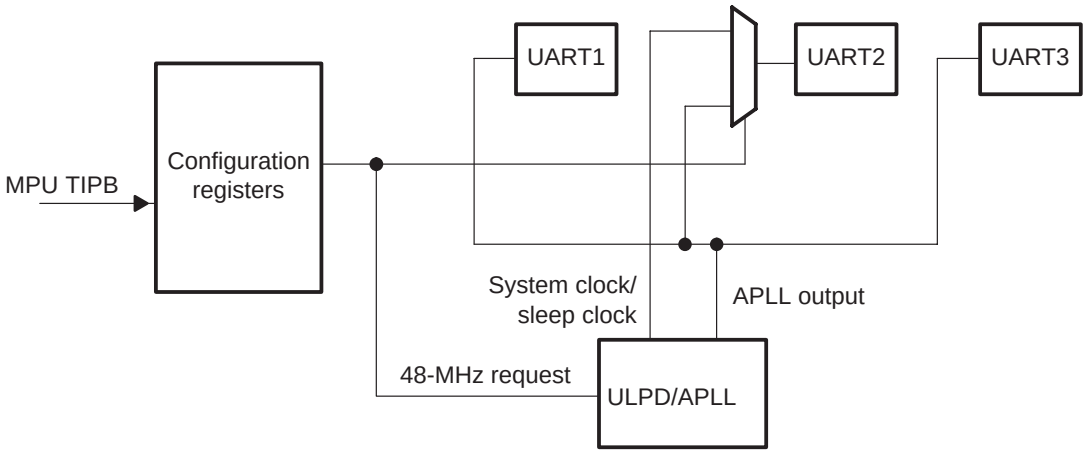
- UART1
- UART2
- UART3

UART1 and UART3 also support the IrDA mode if the proper set of module inputs/outputs is configured.

UART Clocking Scheme

Figure 11 describes the clocking scheme used to enable the UARTs to generate their requested baud rate.

Figure 11. UART Clock Scheme



The clock sources for UART1/UART3 can be the APLL output.

The clock source for UART2 can be:

- The system clock or the sleep clock
- The APLL output

Selection of the clock source is done statically from the OMAP5912 configuration register, which is accessible by software through a normal MPU peripheral access.

Available I/Os per UART

Table 9. Available I/Os per UART

Signal	UART1	UART2	UART3
RX	Yes	Yes	Yes
TX	Yes	Yes	Yes
RTS	Yes	Yes	By mux
CTS	Yes	Yes	By mux
DSR	By mux	No	By mux
DTR	By mux	No	By mux
SD_IRDA	No	No	By mux
BD_CLK	No	Yes	No

For more detail on I/Os, see Appendix B, *Inputs/Outputs*.

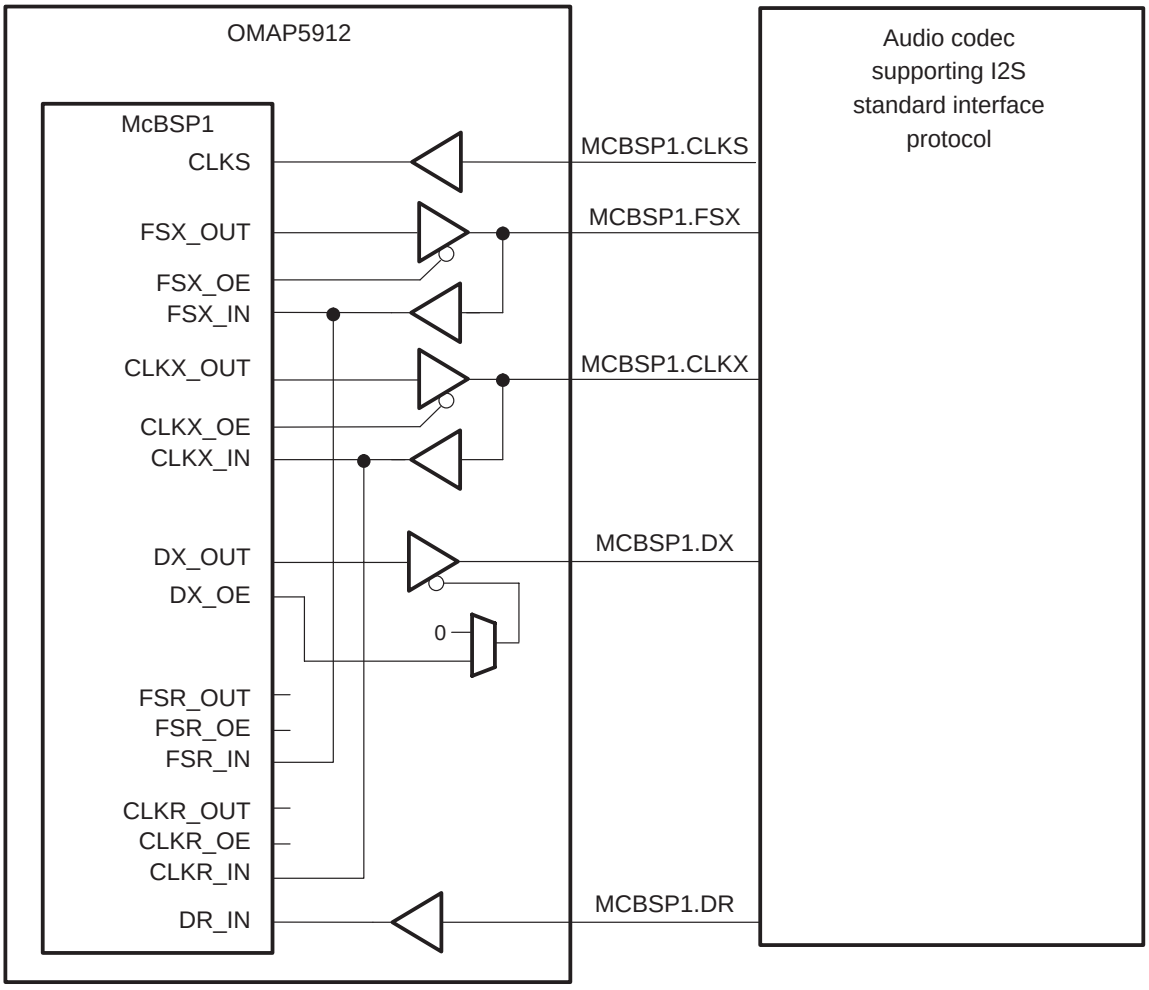
McBSP

The McBSP is a full-duplex serial interface. For example, it can be used to emulate the I2S protocol for interface to an external codec audio device.

McBSP1

McBSP1 is an instance of the McBSP module. The McBSP1 can be used in OMAP5912 to interface with an audio codec compliant with the I2S protocol (5-pin interface). The codec can provide the reference clock. The serial clock and the frame synchronization can be either inputs or outputs. When outputs, they are derived from the reference clock.

Figure 12. McBSP Interface With I2S-Compliant External Codec

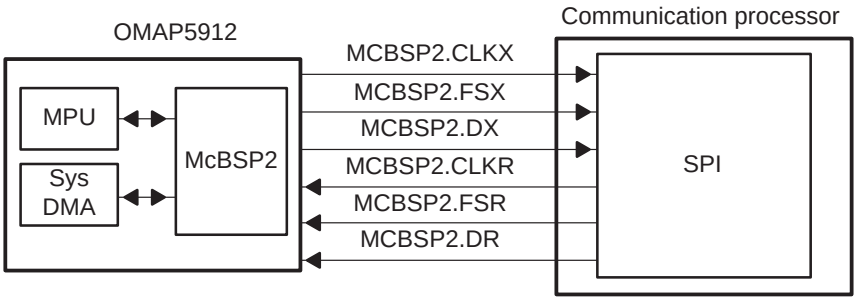


McBSP2

McBSP2 is also an instance of the McBSP module. It can be used either to support SPI mode or to emulate an I2S serial link.

The McBSP2 interface differs slightly from the McBSP1 interface in that there is no capability to connect an OMAP5912 external reference clock to it.

Figure 13. McBSP Interface With Communication Processor



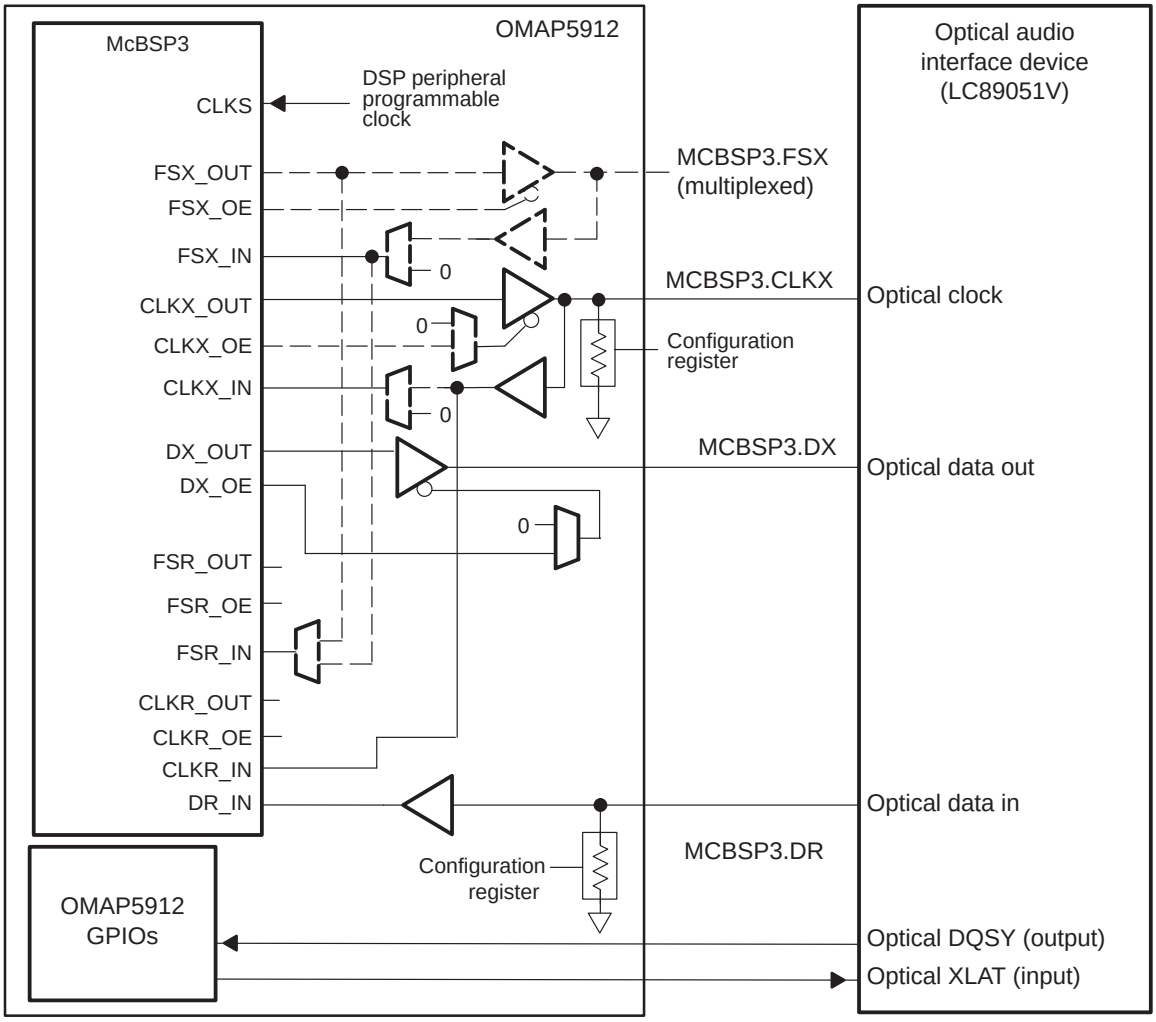
Note: To support I2S master and slave modes, McBSP2_CLKX, McBSP2_XSYNC, McBSP2_RSYNC, McBSP2_CLKR are inputs/outputs. They are used unidirectionally for the communication processor connection.

McBSP3

McBSP3 is a third instance of the McBSP module. There are two connection modes for this McBSP:

- The first connection mode is a 3-pin interface. The frame synchronization is internally looped back (FSXO to FSRI), as is the clock. This is the default reset configuration. In this case, the McBSP3 is half-duplex, master for transmission, slave for reception. With the assistance of two GPIOs, this McBSP mode (3 pins) can be configured to connect to an external optical audio interface device, such as the Sanyo−LC89051V.
- In addition, the frame synchronization signal is multiplexed to allow a 4-pin McBSP interface. The second connection mode is a 4-pin interface. The frame synchronization is bidirectional, as is the clock. In this case, the McBSP3 is half-duplex, master/slave for transmission, slave for reception, enabling an additional I2S emulator.

Figure 14. McBSP3 Interface Connected to Optical Device



Notes: 1) The solid lines show the reset connection configuration, whereas the dashed lines show the second connection configuration.

2) Selection of the interface (3-pin or 4-pin) is done by the software through the CONF_MOD_CTRL_1 register.

Serial Port Interface (SPI)

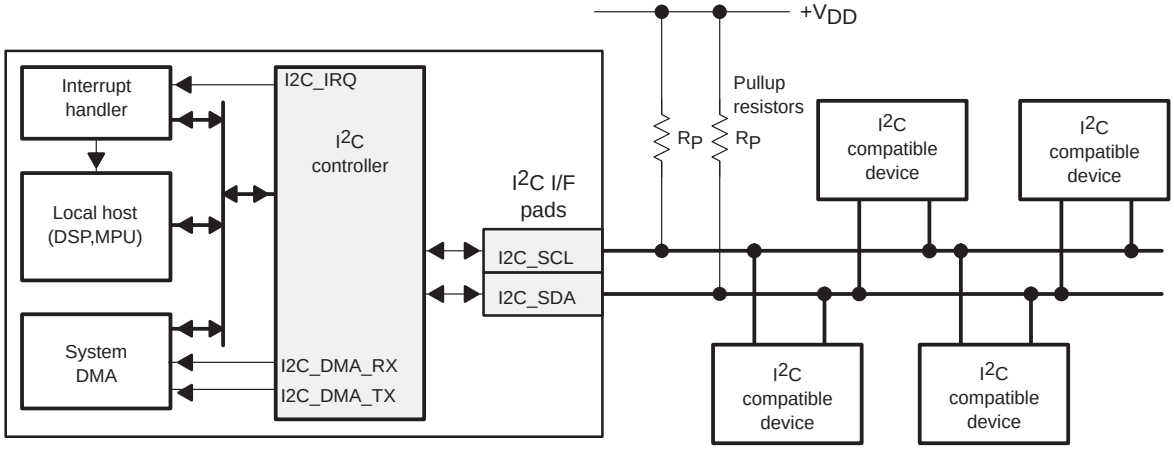
The serial port interface is a bidirectional, four-line interface dedicated to the transfer of data to and from external devices offering a four-line serial interface.

Inter-Integrated Circuit (I^2C)

The multimaster I^2C peripheral provides an interface between a local host (LH) such as an MPU or DSP processor and any I^2C-bus-compatible device that connects via 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 or a DMA through the two-wire I^2C interface.

This I^2C peripheral supports any slave or master I^2C-compatible device. Figure 15 shows an example of a system with multiple I^2C-compatible devices in which the I^2C serial ports are connected together for a two-way transfer from one device to other devices.

Figure 15. I^2C System Overview



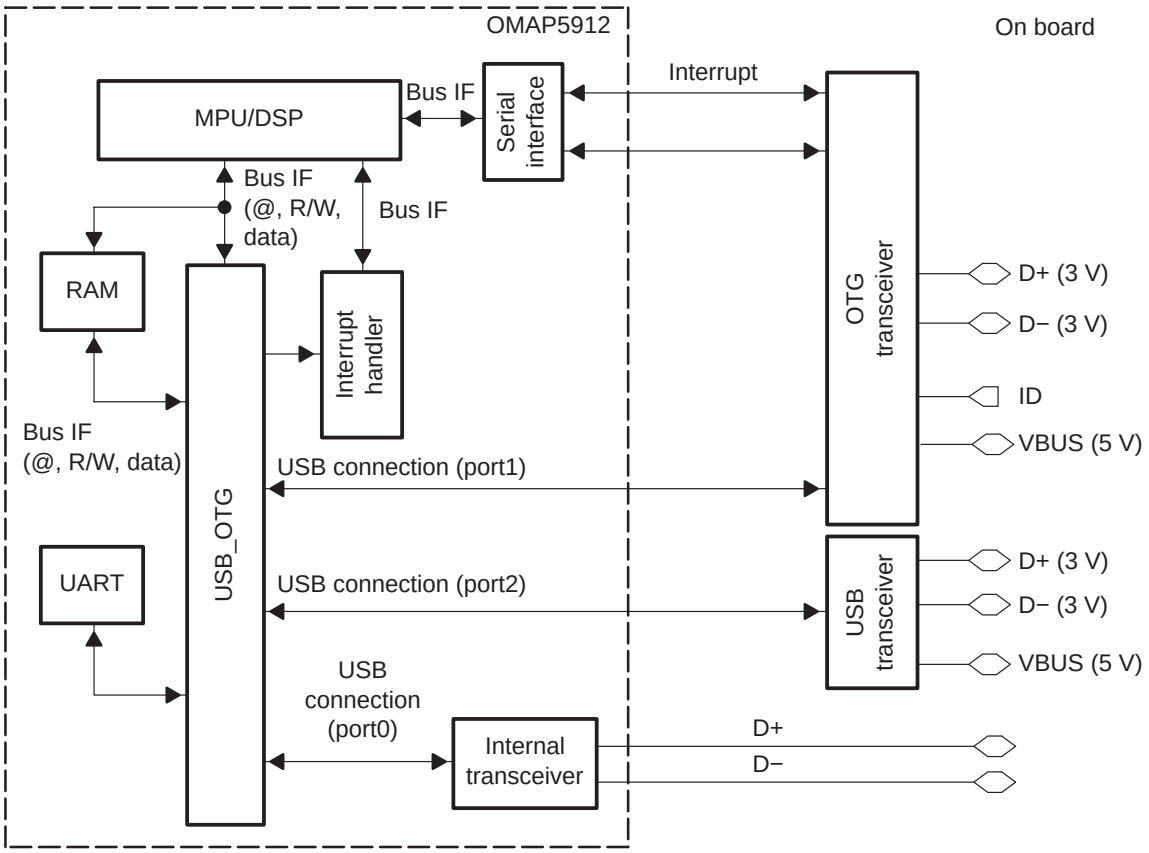
USB On-The-Go (OTG)

The OMAP 5912 OTG module provides a USB device controller, a three-port USB host controller, and an OTG controller that is capable of providing USB On-The-Go functionality using the USB device controller and one port o the USB host controller.

The main features of USB On-The-Go are:

- ❏ USB specification version 1.1 compatible with OTG enhanced features
- ❏ USB host is Open HCI Rev 1.0 compatible.

Figure 16. USB OTG Integration at System Level



Note: OTG functionality requires a special transceiver external to OMAP5912. The communication between the transceiver and the controller is done through an I^2C communication channel.

In this case, the USB OTG module port 2 is connected to an external USB OTG transceiver, port 1 is connected to a USB transceiver, and port 0 is connected to the integrated USB transceiver.

GPIO

There are four instances of the GPIO modules (GPIO1, GPIO2, GPIO3, GPIO4) included in OMAP5912.

Each GPIO module supports:

- Data input/output register
- Event detection capability:
  - To generate two synchronous interrupts in active mode
  - To generate a wake-up request while the system is in idle mode

This peripheral allows connection of 16 dedicated pins configurable either as input or output for general purposes.

Eight data output lines of the GPIO3 are ORed together to generate a global output line at the OMAP5912 boundary. This global output line can be used in conjunction with the SSI to provide a CMT-APE interface to the OMAP5912.

32-Bit Dual-Mode Timer

There are eight instances of this timer. It contains a free-running upward counter with autoreload capability on overflow. The timer counter can be read and written on the fly (while counting). The timer module includes compare logic to allow an interrupt event on a programmable counter matching value. A dedicated output signal can be pulsed or toggled on overflow and match event. This offers timing stamp trigger signal or PWM (pulse width modulation) signal sources. A dedicated input signal can be used to trigger automatic timer counter capture and interrupt event on programmable input signal transition type.

All of the general-purpose timers have the capability to run either from the system clock or from the sleep clock (32-kHz clock).

Note:

Three of the eight dual-mode timer PWM outputs are connected at OMAP5912 I/Os. Two of the eight dual-mode timer input capture are connected at OMAP5912 I/Os. The system clock can come either from OMAP or directly from the input clock. Actual implementation in OMAP5912 is as follows:

- PWM0 PWM output of GPTIMER1
- PWM1 PWM output of GPTIMER2
- PWM2 PWM output of GPTIMER3
- Event0 EVENT CAPTURE input of GPTIMER4
- Event1 EVENT CAPTURE input of GPTIMER5

32-kHz Synchronization Counter

This is a 32-bit counter, clocked by the falling edge of the 32-kHz clock. It is reset while the power-up reset ($\overline{\text{PWRON\_RESET}}$) primary I/O is active (main OMAP5912 reset). Then, on the rising edge of the power-up reset ($\overline{\text{PWRON\_RESET}}$ release), it starts to count forever. When the highest value is reached, it wraps back to zero and starts running again.The MPU can read it from a 32-bit peripheral access, whereas the DSP can access it only through two consecutive 16-bit accesses.

Watchdog

In addition to the two OMAP3.2 watchdogs, there are two instances of the watchdog module in OMAP5912. One is clocked at 32 kHz and the other by a 192-MHz clock (secure watchdog). The secure watchdog is used in conjunction with OMAP5912 security features.

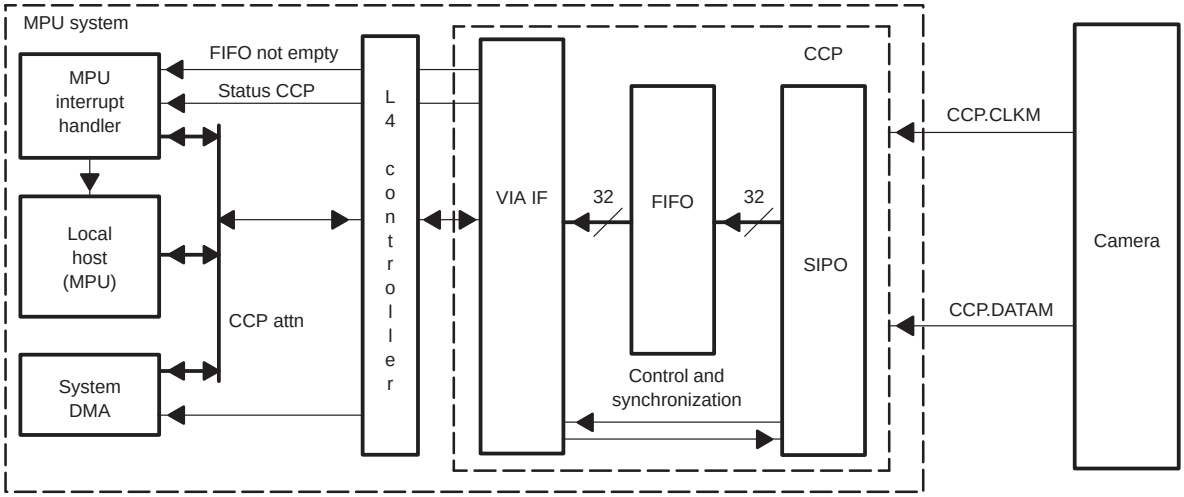
The watchdog module is a 32-bit general-purpose counter (same programming model as the 32-bit general-purpose timer) with watchdog capability (generates a reset).

When the watchdog module is used as secure watchdog, it is accessible only in secure mode.

Compact Camera Port (CCP)

The CCP is a serial interface from a camera. The CCP block combines the camera and the host through the VIA bus. The input data signal from the camera is expanded to 32 bits, which is the width of the VIA bus. The inputs are the serial input data (CCPDa) and a clock signal (CCPClk).

Figure 17. CCP Internal Block Diagram

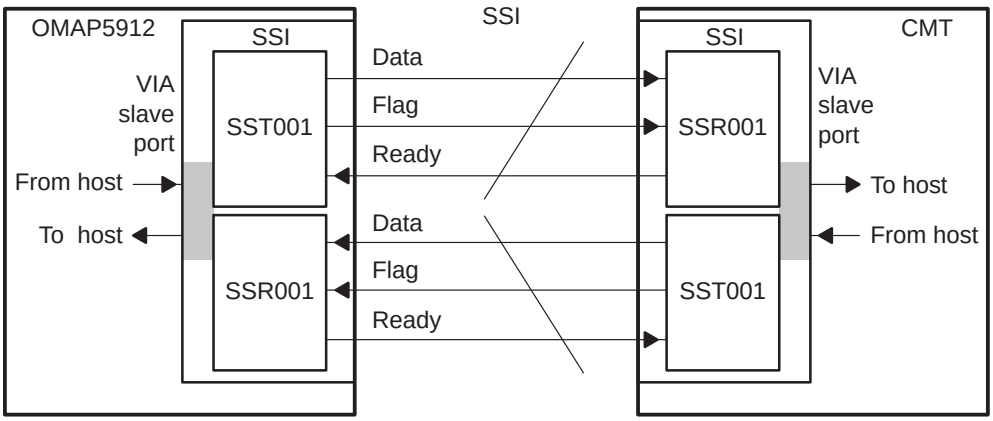


Synchronous Serial Interconnect

The synchronous serial interconnect (SSI) peripheral enables OMAP5912 to exchange information with an external modem. It enables a full duplex interface, using a synchronous serial interconnect protocol (SSI). This protocol consists of a transmitter and a receiver.

On the modem side, there is also a receiver and a transmitter.

Figure 18. CMT-APE Interchip Communication



The SSI module includes:

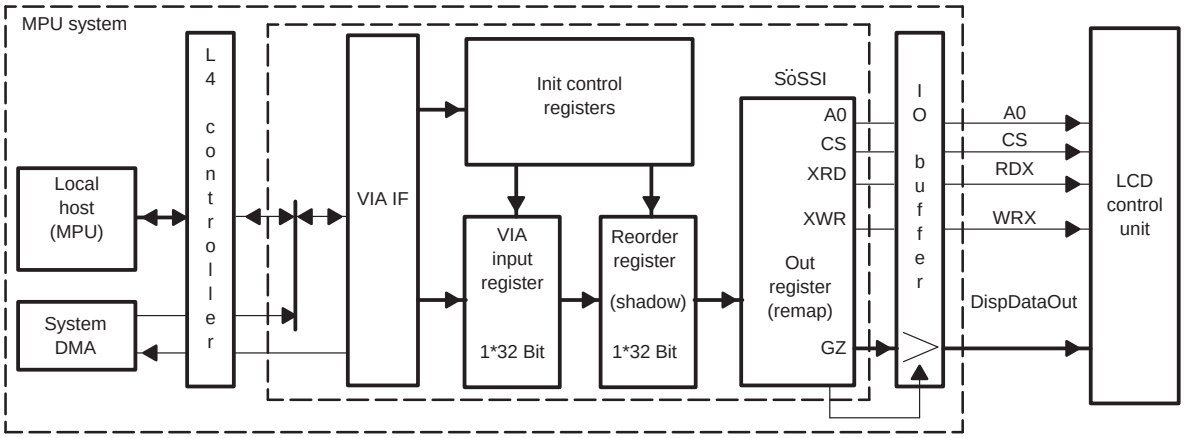
- One instance of each SST and SSR module
- To enable a CMT-APE interface, an additional output generates a wake-up signal to the CMT chip on the SSI interface. This output is driven by an OR gate between 8-bit GPIO3 (GPIO51 to GPIO58) output lines. The CMT chip can also wake up the APE chip with a dedicated GPIO (via its asynchronous interrupt detection).

SoSSI

The SoSSI peripheral supports the MeSSI display interface standard. This module includes a VIA-bus interface converted into a TIPB interface connected on the MPU shared peripheral bus and an interface with the system DMA, which enables connection of an extended LCD display to the OMAP5912 device.

The display interface is a parallel interface. It consists of a display data I/O bus (DispDataOut[15:0]), write enable (WRX), read enable (RDX), chip-select (CS), and command/pixel data (A0) signals.

Figure 19. SoSSI Internal Block Diagram



MMC/SDIO

There are two instances of the MMC/SDIO module in OMAP5912. This module is a superset of the MMC/SD module from OMAP5910, including I/O interfacing capability to support an SDIO card.

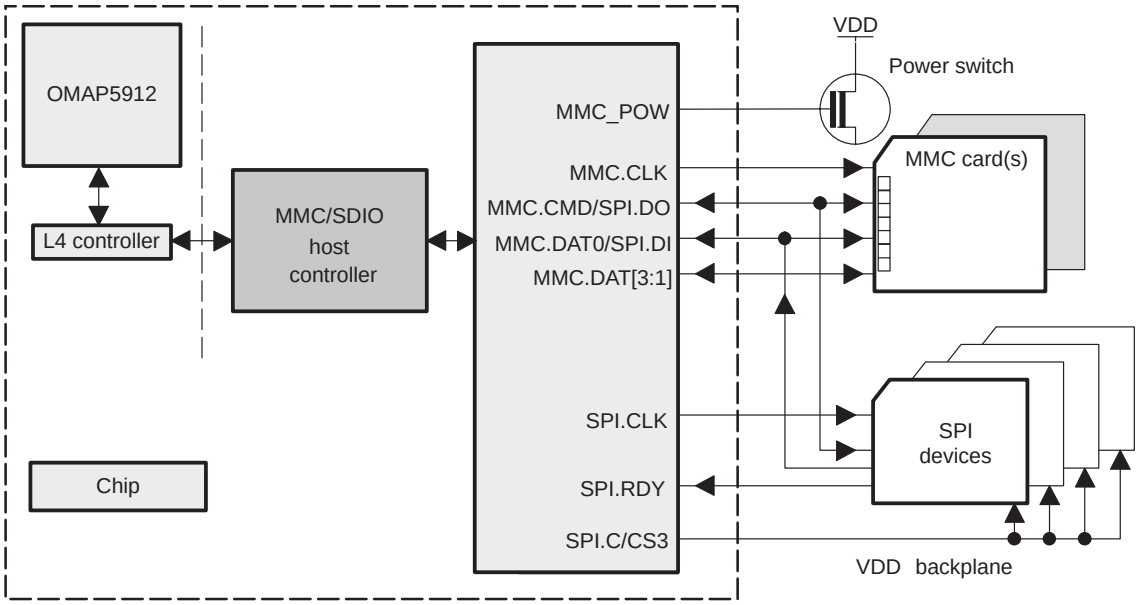
The MMC/SDIO1 replaces the MMC/SD from OMAP5910, whereas the MMC/SDIO2 is an additional peripheral.

The MMC/SDIO host controller provides an interface between a local host, such as the MPU/DSP and MMC/SD/SDIO memory cards plus up to four serial flash cards, and handles MMC/SDIO or SPI transactions with minimum local host intervention.

The following combination of external devices is supported:

- One or more MMC memory cards sharing the same bus and up to four devices with 8-bit SPI protocol interface (serial flash memories, and so on)
- One single SD memory card or SDIO card and up to four devices with 8-bit SPI protocol interface

Figure 20. MMC/SDIO Block Connection



The main features of the MMC/SDIO module are:

- Full compliance with MMC command/response sets as defined in the MMC standard specifications v.3.1
- Full compliance with SD command/response sets as defined in the SD physical layer specifications v.1.0
- Full compliance with SDIO command/response sets as defined in the SDIO card specification v1.0

OMAP5912 also supports control signals to external level shifters in case the voltage for OMAP5912 I/O is set to 1.8 V. These signals are derived from the direction control of the MMC_DAT0 and MMC_CMD I/O pads (one direction control per data bit line and one direction control for the command line).

Note:

- The MMC/SDIO2 clock is multiplexed between the 48-MHz clock (APLL output) and the system clock (19.2 MHz or 12 MHz).
- At reset, the MMC/SDIO2 clock selection is the system clock.
- Whereas the MMC/SDIO1 interface includes all of the MMC/SDIO pins except the direction controls (data and control), the full MMC/SDIO2 is routed at the OMAP5912 level. The OMAP5912 configuration selects only the part of the interface that is required.

VLYNQ

VLYNQ is a serial (that is, low pin count) communications interface that enables the extension of an internal bus segment to one or more external physical devices. The external devices are mapped into local, physical address space and appear as if they are on the internal bus of the OMAP 5912. The external devices must also have a VLYNQ interface. The VLYNQ module serializes bus transactions in one device, transfers the serialized data between devices via a VLYNQ port, and deserializes the transaction in the external device.

2.2.3 Other Modules

JTAG TAP Controller

OMAP5912 JTAG TAP controller handles standard IEEE JTAG interfaces. Boundary scan chain is implemented in OMAP5912.

eFuse Modules

The generic electrical fuse implementation refers to the combination of electrical fuse, control, and connectivity that enables the programming and use of electrical fuses before the packaging. Electrical fuses are generally used for:

- Memory repair
- Die identification
- Production identification (bits used in combination to qualify the OMAP5912 device, either in the emulation, high security, or general-purpose)
- Encryption key coding

The electrical fuse cell contains an electrically programmable element whose output can be permanently set to logic 1 instead of its natural unprogrammed state, logic 0.

The MPU can access the die ID and the production ID through an MPU peripheral access.

Boot Device Configuration

Depending on the device_type signal, the MPU core can boot either from the flash or from the boot ROM. The C55x DSP core can boot either from internal memory or from external memory. The boot is accomplished with an orderlycombination of hardware and software control sequences.

- MPU boot ROM

  Upon the deassertion of the reset input pin (cold reset exit) of OMAP 3.2, ARM926EJS traps to its reset vector, while the C55x DSP is held in reset.

- DSP boot modes

  System software controls the C55x DSP boot option by programming registers via the MPUI interface when the DSP is being held in reset.

Boot ROM

The OMAP5912 configuration is 16384 x 32 bits. For more detail, see Chapter 5, *Initialization*.

Security Layer

The OMAP 3.2 gigacell includes special-purpose security hardware that is used to activate a secure mode. The secure mode can be viewed as a third privilege level on the MPU. It is used to create an environment for protecting sensitive information from access by untrusted software. The secure mode is set with the assertion of a dedicated signal (secure bit) that propagates across OMAP5912 and creates a boundary between resources that trusted software might access and those available to any software.

Generic Distributed DMA (GDD)

The GDD is a module that interfaces one OCP target device (the peripheral target) to another OCP target device (the memory target) by providing a DMA service to the peripheral target. The DMA service is configured through a configuration port. GDD wraps a peripheral OCP host interface and presents the system with one initiator and one target port.

The GDD is attached to the SSI peripheral. It provides the necessary bandwidth between the SSI and the host without affecting the peripheral subsystem bandwidth.

3 OMAP5912 Wake-up Capabilities

Table 10 lists the peripherals able to wake up the system and their event captures. Wake-up can occur either directly or by GPIO.

Table 10. Event Captures for Peripheral Wake-up Capability

Peripheral	Event Capture	Receive Capability in Deep Sleep Mode
UART2	Low transition on RX process by the ULPD module or using GPIO configuration on the RTS interface	Provided that the external transmitter uses the UART2 baud clock, the data is received properly and captured.
GPIO1, GPIO2, GPIO3, GPIO4	Asynchronous interrupt detection; all wake-up events per module are merged into one asynchronous interrupt.	NA
GPIO2	Asynchronous interrupt detection	NA
GPIO3	Asynchronous interrupt detection	NA
GPIO4	Asynchronous interrupt detection	NA
SPI	Using GPIO configuration on the all SPI input pins	Not able to receive the first data in slave mode
I^2C	Using GPIO configuration on the all I^2C interfaces	Not able to receive the first data in slave mode
UART3	Using GPIO configuration on the all UART3 input pins	Not able to receive the data
McBSP1	Using GPIO configuration on the interface	Not able to receive the data
McBSP2	Using GPIO configuration on the all McBSP2 input pins	Not able to receive the data
STI (reserved)	Using GPIO configuration on the interface	Not able to receive the data
GPTIMER1…8	Can wake up the system when the clock is configured as 32-kHz through its own interrupt per general-purpose timer	NA
OS Timer	The timer can generate interrupt.	NA
32-kHz watchdog	The 32-kHz watchdog can generate reset.	NA

Notes: 1) When the wake-up is done through the GPIO, the programmer must ensure that the GPIO direction is set up as input.

2) Depending on the OMAP5912 configuration before entering into power-down mode, the GPIO can wake either the entire system or only a specific subsystem.

3) In addition to the wake-up capability mentioned in the above table, several peripherals are also able to generate their own wake-up request. These requests are merged and sent to OMAP3.2 and the MPU level 2 interrupt handler as a global wake-up request.

Table 10. Event Captures for Peripheral Wake-up Capability (Continued)

Peripheral	Event Capture	Receive Capability in Deep Sleep Mode
Emulation wake up	The emulation is also able to wake up the ARM926EJS. This wake up event is also merged with the application wake up event.	NA
USB_OTG	Asynchronous detection on USB device controller to request the USB clock	NA
MPUIO	Can wake up the system by sampling data at low frequency and sending interrupt to MCU	NA

Notes: 1) When the wake-up is done through the GPIO, the programmer must ensure that the GPIO direction is set up as input.

2) Depending on the OMAP5912 configuration before entering into power-down mode, the GPIO can wake either the entire system or only a specific subsystem.

3) In addition to the wake-up capability mentioned in the above table, several peripherals are also able to generate their own wake-up request. These requests are merged and sent to OMAP3.2 and the MPU level 2 interrupt handler as a global wake-up request.

3.1 OMAP5912 Pin Description

See Chapter 22, *Pinout*, for I/O pin description. It includes the pin description (location on the ballout, reset state, buffer used, signal description, electrical information, and buffer description).

The OMAP5912 boot configuration pins can select either:

- Boot from internal boot ROM or from external memory
- The configuration of the external memory (either address/data multiplexed or separate address and data)

Index

L

M

O

P

R

S

U

W

OMAP5912 Multimedia Processor
OMAP3.2 Subsystem
Reference Guide

Literature Number: SPRU749A
March 2004

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.

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
		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 document introduces and briefly defines the main features of the OMAP3.2 subsystem of the OMAP5912 multimedia processor.

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.

Related Documentation From Texas Instruments

The following documents describe the OMAP5910 device and related peripherals. Copies of these documents are available on the Internet at www.ti.com. *Tip:* Enter the literature number in the search box provided at www.ti.com.

OMAP5912 Multimedia Processor Device Overview and Architecture Reference Guide (literature number SPRU748) introduces the setup, components, and features of the OMAP5912 multimedia processor and provides a high-level view of the device architecture.

OMAP5912 Multimedia Processor OMAP 3.2 Subsystem Reference Guide (literature number SPRU749) introduces and briefly defines the main features of the OMAP3.2 subsystem of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor DSP Sybsystem Reference Guide (literature number SPRU750) describes the OMAP5912 multimedia processor DSP subsystem. The digital signal processor (DSP) subsystem is built around a core processor and peripherals that interface with: 1) The ARM926EJS via the microprocessor unit interface (MPUI); 2) Various standard memories via the external memory interface (EMIF); 3) Various system peripherals via the TI peripheral bus (TIPB) bridge.

OMAP5912 Multimedia Processor Clocks Reference Guide (literature number SPRU751) describes the clocking mechanisms of the OMAP5912 multimedia processor. In OMAP5912, various clocks are created from special components such as the digital phase locked loop (DPLL) and the analog phase-locked loop (APLL).

OMAP5912 Multimedia Processor Initialization Reference Guide (literature number SPRU752) describes the reset architecture, the configuration, the initialization, and the boot ROM of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Power Management Reference Guide (literature number SPRU753) describes power management in the OMAP5912 multimedia processor. The ultralow-power device (ULPD) generates and manages clocks and reset signals to OMAP3.2 and to some peripherals. It controls chip-level power-down modes and handles chip-level wake-up events. In deep sleep mode, this module is still active to monitor wake-up events.This book describes the ULPD module and outline architecture.

OMAP5912 Multimedia Processor Security Features Reference Guide (literature number SPRU754) describes the security features of teh OMAP5912 multimedia processor. The OMAP5912 security scheme relies on the OMAP3.2 secure mode. The distributed security on the OMAP3.2 platform is a Texas Instruments solution to address m-commerce and security issues within a mobile phone environment. The OMAP3.2 secure mode was developed to bring hardware robustness to the overall OMAP5912 security scheme.

OMAP5912 Multimedia Processor Direct Memory Access (DMA) Support Reference Guide (literature number SPRU755) describes the direct memory access support of the OMAP5912 multimedia processor. The OMAP5912 processor has three DMAs:

- The system DMA is embedded in OMAP3.2. It handles DMA transfers associated with MPU and shared peripherals.
- The DSP DMA is embedded in OMAP3.2. It handles DMA transfers associated with DSP peripherals.
- The generic distributed DMA (GDD) is an OMAP5912 resource attached to the SSI peripheral. It handles only DMA transfers associated with the SSI peripheral.

OMAP5912 Multimedia Processor Memory Interfaces Reference Guide (literature number SPRU756) describes the memory interfaces of the OMAP5912 multimedia processor.

- SDRAM (external memory interface fast, or EMIFF)
- Asynchronous and synchronous burst memory (external memory interface slow, or EMIFS)
- NAND flash (hardware controller or software controller)
- CompactFlash on EMIFS interface
- Internal static RAM

OMAP5912 Multimedia Processor Interrupts Reference Guide (literature number SPRU757) describes the interrupts of the OMAP5912 multimedia processor. Three level 2 interrupt controllers are used in OMAP5912:

- One MPU level 2 interrupt handler (also referred to as MPU interrupt level 2) is implemented outside of OMAP3.2 and can handle 128 interrupts.
- One DSP level 2 interrupt handler (also referred to as DSP interrupt level 2.1) is instantiated outside of OMAP3.2 and can handle 64 interrupts.
- One OMAP3.2 DSP level 2 interrupt handler (referenced as DSP interrupt level 2.0) can handle 16 interrupts.

OMAP5912 Multimedia Processor Peripheral Interconnects Reference Guide (literature number SPRU758) describes various periperal interconnects of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Timers Reference Guide (literature number SPRU759) describes various timers of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Serial Interfaces Reference Guide (literature number SPRU760) describes the serial interfaces of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Universal Serial Bus (USB) Reference Guide (literature number SPRU761) describes the universal serial bus (USB) host on the OMAP5912 multimedia processor. The OMAP5912 processor provides several varieties of USB functionality. Flexible multiplexing of signals from the OMAP5912 USB host controller, the OMAP5912 USB function controller, and other OMAP5912 peripherals allow a wide variety of system-level USB capabilities. Many of the OMAP5912 pins can be used for USB-related signals or for signals from other OMAP5912 peripherals. The OMAP5912 top-level pin multiplexing

controls each pin individually to select one of several possible internal pin signal interconnections. When these shared pins are programmed for use as USB signals, the OMAP5912 USB signal multiplexing selects how the signals associated with the three OMAP5912 USB host ports and the OMAP5912 USB function controller can be brought out to OMAP5912 pins.

OMAP5912 Multimedia Processor Multi-channel Buffered Serial Ports (McBSPs) Reference Guide (literature number SPRU762) describes the three multi-channel buffered serial ports (McBSPs) available on the OMAP5912 device. The OMAP5912 device provides multiple high-speed multichannel buffered serial ports (McBSPs) that allow direct interface to codecs and other devices in a system.

OMAP5912 Multimedia Processor Camera Interface Reference Guide (literature number SPRU763) describes two camera inerfaces implemented in the OMAP5912 multimedia processor: compact serial camera port and camera parallel interface.

OMAP5912 Multimedia Processor Display Interface Reference Guide (literature number SPRU764) describes the display interface of the OMAP5912 multimedia processor.

- LCD module
- LCD data conversion module
- LED pulse generator
- Display interface

OMAP5912 Multimedia Processor Multimedia Card (MMC/SD/SDIO) (literature number SPRU765) describes the multimedia card (MMC) interface of the OMAP5912 multimedia processor. The multimedia card/secure data/secure digital IO (MMC/SD/SDIO) host controller provides an interface between a local host, such as a microprocessor unit (MPU) or digital signal processor (DSP), and either an MMC or SD memory card, plus up to four serial flash cards. The host controller handles MMC/SD/SDIO or serial port interface (SPI) transactions with minimal local host intervention.

OMAP5912 Multimedia Processor Keyboard Interface Reference Guide (literature number SPRU766) describes the keyboard interface of the OMAP5912 multimedia processor. The MPUIO module enables direct I/O communication between the MPU (through the public TIPB) and external devices. Two types of I/O can be used: specific I/Os dedicated for 8 x 8 keyboard connection, and general-purpose I/Os.

OMAP5912 Multimedia Processor General-Purpose Interface Reference Guide (literature number SPRU767) describes the general-purpose in-

terface of the OMAP5912 multimedia processor. There are four GPIO modules in the OMAP5912. Each GPIO peripheral controls 16 dedicated pins configurable either as input or output for general purposes. Each pin has an independent control direction set by a programmable register. The two−edge control registers configure events (rising edge, falling edge, or both edges) on an input pin to trigger interrupts or wake−up requests (depending on the system mode). In addition, an interrupt mask register masks out specified pins. Finally, the GPIO peripherals provide the set and clear capabilities on the data output registers and the interrupt mask registers. After detection, all event sources are merged and a single synchronous interrupt (per module) is generated in active mode, whereas a unique wake−up line is issued in idle mode. Eight data output lines of the GPIO3 are ORed together to generate a global output line at the OMAP5912 boundary. This global output line can be used in conjunction with the SSI to provide a CMT−APE interface to the OMAP5912.

OMAP5912 Multimedia Processor VLYNQ Serial Communications Interface Reference Guide (literature number SPRU768) describes the VLYNQ of the OMAP5912 multimedia processor.

VLYNQ is a serial communications interface that enables the extension of an internal bus segment to one or more external physical devices. The external devices are mapped into local, physical address space and appear as if they are on the internal bus of the OMAP 5912. The external devices must also have a VLYNQ interface. The VLYNQ module serializes bus transactions in one device, transfers the serialized data between devices via a VLYNQ port, and de-serializes the transaction in the external device.

OMAP5912 includes one VLYNQ module connected on OCPT2 target port and OCPI initiator port. These connections are configured via a static switch, which selects either SSI or VLYNQ module. This switch, forbids the simultaneous use of GDD/SSI and VLYNQ. The switch is controlled by the VLYNQ_EN bit in the OMAP5912 configuration control register (CONF_5912_CTRL).

OMAP5912 Multimedia Processor Pinout Reference Guide (literature number SPRU769) provides the pinout of the OMAP5912 multimedia processor. After power-up reset, the user can change the configuration of the default interfaces. If another interface is available on top of the default, it is possible to enable a new interface for each ball by setting the corresponding 3-bit field of the associated FUNC_MUX_CTRL register. It is also possible to configure on-chip pullup/pulldown. This document

also describes the various power domains so that the user can apply the different interfaces seamlessly with external components.

OMAP5912 Multimedia Processor Window Tracer (WT) Reference Guide (literature number SPRU770) describes the window tracer module used to capture the memory transactions from four interfaces: EMIFF, EMIFS, OCP-T1, and OCP-T2. This module is located in the OMAP3.2 traffic controller (TC).

OMAP5912 Multimedia Processor Real-Time Clock Reference Guide (literature number SPRUxxx) describes the real-time clock of the OMAP5912 multimedia processor. The real-time clock (RTC) block is an embedded real-time clock module directly accessible from the TIPB bus interface.

Trademarks

Contents

Figures

Tables

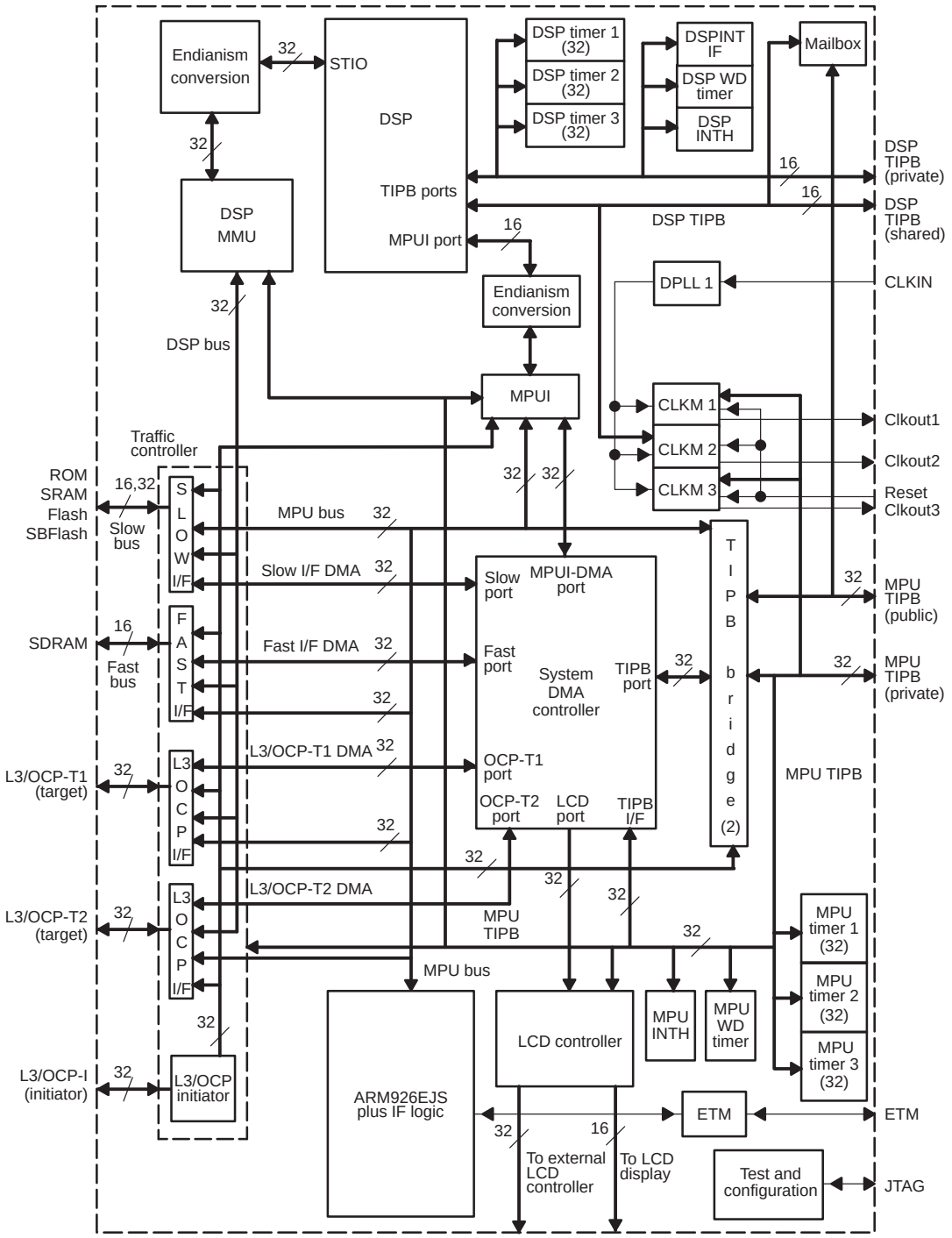
OMAP3.2 Subsystem

This document introduces and briefly defines the main features of the OMAP3.2 subsystem of the OMAP5912 multimedia processor.

1 Introduction

Figure 1 shows the OMAP5912 OMAP3.2 gigacell.

Figure 1. OMAP3.2 Gigacell



2 OMAP3.2 Features

The OMAP3.2 includes the following features:

- ARM926EJS megacell including:
  - ARM926EJS, running at a maximum frequency of 192 MHz
  - MMU with translation lookaside buffer (TLBx)
  - L1 16K-byte, four-way, set-associative instruction cache
  - L1 8K-byte, four-way, set-associative data cache with write buffer
- MPU interrupt handler level 1
- Embedded trace macrocell module, ETM version 2.a in a 13-bit mode configuration or in a 17-bit demultiplexed mode configuration
- DSP megacell rev 2.0a+ including:
  - Embedded ICE emulator interface through JTAG port
  - TMS320C55x (C55x) DSP rev 2.1, running at a maximum of 192 MHz
  - L1 cache (24K bytes)
    - 16K-byte, two-way, set-associative instruction cache *(on the OMAP5912 prototype, one wait state is introduced in case of discontinuity)*
    - 2 X 4K-byte RAM set for instruction
  - DARAM 64K-byte, zero wait state, 32-bit organization
  - SARAM 96K-byte, zero wait state, 32-bit organization
  - PDROM (32K bytes)
  - DMA controller: six physical channels, five ports
  - DSP trace module
  - Hardware accelerators motion estimation (ME), discrete/inverse discrete cosine transform (DCT/IDCT), and pixel interpolation (PI)
  - DSP interrupt handler level 1 in the C55x DSP core
- DSP MMU
- DSP level 2 interrupt handler, which enables connection to 16 additional interrupt lines outside OMAP. The priority of each interrupt line is controlled by software.
- DSP interrupt interface, which enables connection to the interrupt lines coming out of the level 2 interrupt handler and the interrupt lines requiring

more priority. The outcome interrupt of this module is then connected to the DSP megacell to be processed by the DSP. This module mainly ensures that all interrupts going to the DSP megacell are level-sensitive.

- DSP peripherals:
  - 3 x 16-bit DSP private timers
  - 1 x 16-bit DSP private watchdog
- Mailboxes:
  - Four mailboxes are implemented:
    - Two read/write accessible by MPU, read-only by the DSP
    - Two read/write accessible by the DSP, read-only by the MPU

    Each mailbox is implemented with 2 x 16-bit registers. When a write is done into a register by one processor, it generates an interrupt, released by the read access of the other processor.
- MPU peripherals
  - 3 x 32-bit private timers; their clock is either the OMAP3.2 reference input clock or the divided MPU clock.
  - 1 x 16-bit private watchdog; can be configured as a 16-bit general-purpose timer by software. Its clock is the OMAP3.2 reference input clock divided by 14.
- External LCD controller support in addition to the OMAP LCD controller
  - LCD controller with its own tearing-effect logic
- Memory traffic controller
  - External Memory Interface Fast(EMIFF) is a memory interface that enables 16-bit data SDRAM memory access at 96-MHz maximum frequency. It supports connection to a128M-byte maximum of SDRAM. The address width is 16 bits, and two bank selection bits are also provided. The OMAP5912 chip requires interfacing with a maximum of four banks of 64M x 16-bit SDRAM memory with DDR capability.
  - External Memory Interface Slow (EMIFS) connects external device memories (such as common flash and SRAM memories) at 80-MHz maximum frequency. This interface is also used internally to connect the boot ROM, the secure RAM, and their secure eFuse components accessible in secure mode. This interface enables 16-bit data accesses and provides four chip-selects. Each chip-select is able to

support up to 64M bytes of address space through a 25-bit address bus.

Note:

At the OMAP5912 level, two chip-selects can be split in half by configuration to provide four chip-selects. This enables OMAP5912 to provide up to six chip-selects supporting up to 32M bytes of address space on four chip-selects and up to 64M bytes on two chip-selects.

L3 OCP-T1 and L3 OCP-T2 ports are provided to enable memory access from the OMAP3.2 gigacell on a standard basis protocol. Only the L3 OCP-T1 is used in OMAP5912 to access the single SRAM memory.

- Emulator interface through JTAG port
- System DMA running at 96 MHz. It consists of:
  - Seventeen logical channels
  - Seven physical ports + one for configuration
  - Four physical channels

  The ports are connected to the L3 OCP targets, the external memory, the TIPB bridge, the MPUI, and one dedicated port connected to an LCD controller. The system DMA controller can be controlled via the MPU private TIPB or by an external host via the OCP-I port.

  The system DMA controller is designed for low-power operation. It is partitioned into several clock domains where each clock domain is enabled only when it is used. All clocks are disabled when no DMA transfers are active (synchronous to the MPU TIPB, this feature is totally under hardware control; no specific programming is needed).

  Five different logical channels types are supported, each one representing a specific feature set:

  - LCh−2D for memory to memory transfers, 1D and 2D
  - LCh−P for peripheral transfers
  - LCh−PD for peripheral transfers on a dedicated channel
  - LCh−G for graphical transfers/operations
  - LCh−D for display transfers

  The available features are:

  - Support for up to four address modes:

  - Constant
  - Post-increment
  - Single indexing
  - Double indexing
- Different indexing for source-respective destination
- Logical channel chaining
- Software enabling
- Hardware enabling
- Logical channel interleaving
- Logical channel preemption
- Two choices of logical channel arbitration of physical resources, round robin or fixed
- Two levels of logical channel priority
- Constant fill
- Transparent copy
- Rotation 0, 90, 180, and 270
- Seven ports enabling:
  - Memory-to-memory transfers
  - Peripheral-to-memory transfers
  - Memory-to-peripheral transfers
  - Peripheral-to-peripheral transfers
- Binary backward-compatible by default configuration
- Up to four logical channels active in parallel

The logical channel dedicated to the display, LCh−D, has several additional features:

- Channel can be shared by two LCD controllers
- Supports both single- and dual-block modes
- Supports separate indexing and numbering for dual-block mode for both elements and frames

- Two DPLLs:

  OMAP3.2 provides one DPPL per main clock domain:

  - MPU/traffic controller clock domain
  - DSP clock domain

  The OMAP3.2 gigacell enables the software to define either:

  - Two coupled domains in scalable mode: only one DPLL is active. The other clocks are a multiple of it.
  - Mixed mode: only one domain is working in asynchronous mode. The other domains are in scalable mode.

- Endianism conversion for DSP

  - The DSP uses big-endian format, whereas the MPU uses little-endian format. Also, as a rule, the OMAP5912 chip works in little endian. Thus, the endianism conversion is useful for all memory or peripheral accesses from on-chip peripherals or all shared memories to the DSP megacell.

OMAP 3.2 is considered a subchip of OMAP5912. To connect the OMAP peripherals, six buses are provided:

- MPU shared TIPB
- MPU private TIPB
- DSP shared TIPB
- DSP private TIPB
- OCP-T2
- OCP-I

3 Traffic Controller

The OMAP 3.2 traffic controller (TC) is the central interconnect that manages all accesses between the following:

- OMAP internal initiators and target resources
- OMAP external initiators and target resources

The TC can have its own clock domain or be synchronous to the MPU clock domain. See Chapter 4, *Clocks,* for more details on clock domains. Typically, the TC clock runs at half of the MPU and DSP clocks.

System initiators are:

- MPU. The MPU is connected to the TC using the MPU bus. The MPU can access memory devices or other type of targets connected to OCP-T1,

OCP-T2, EMIFF, and EMIFS. Selection of the destination target is based on address decoding within the TC.

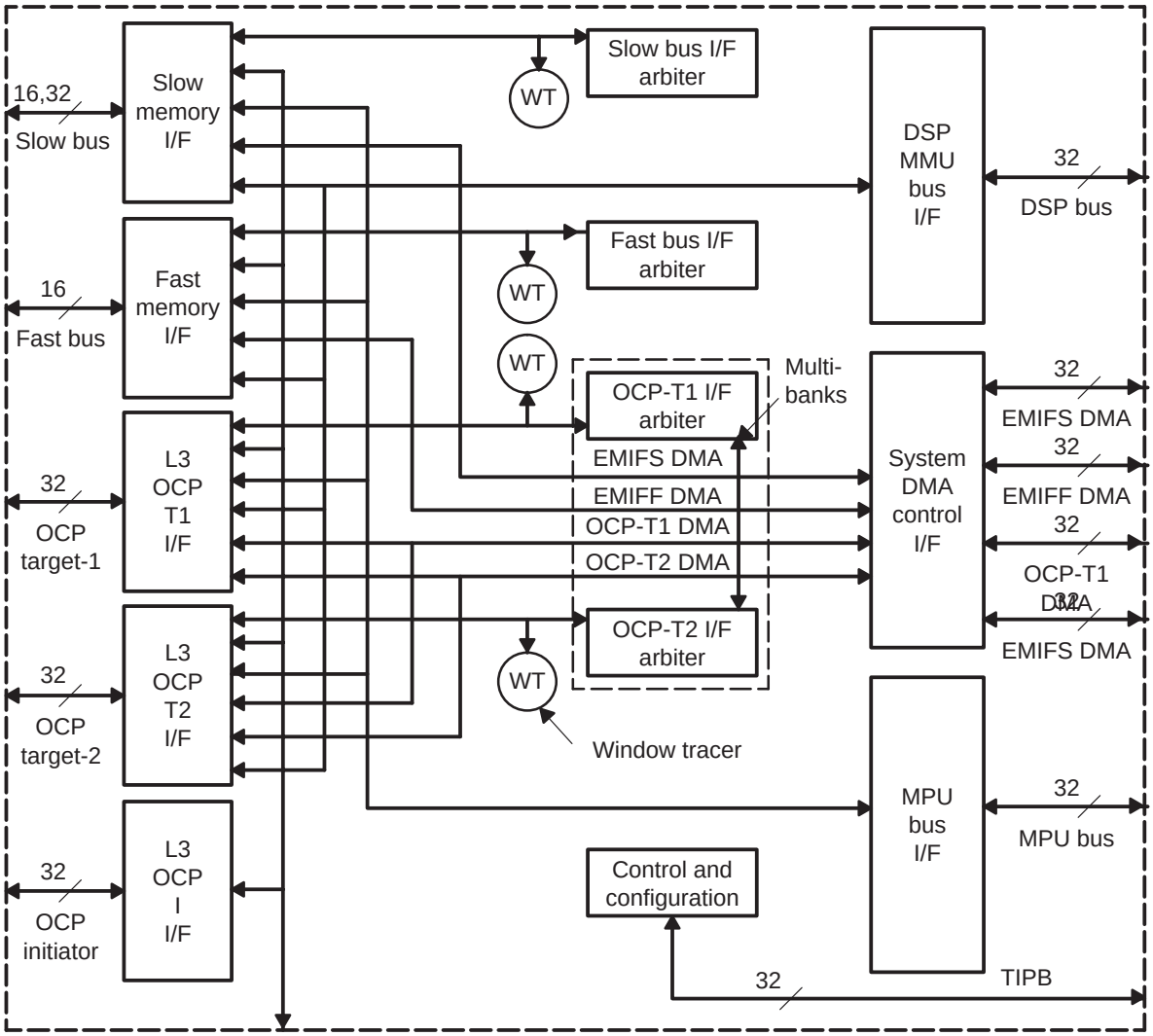
- DSP. The DSP is connected to the TC via the DSP bus. The DSP can access memory devices or other types of targets connected to OCP-T1, OCP-T2, EMIFF, and EMIFS. Selection of the destination target is based on address decoding within the TC.
- System DMA. The system DMA is connected to the TC with four dedicated ports: one for OCP-T1, one for OCP-T2, one for EMIFF, and one for EMIFS. It can access memory devices or other types of targets connected to any of these interfaces. Selection of the destination port is based on system DMA programming.
- External initiator. An external initiator can be connected to the TC via the OMAP OCP-I port. The external master can access memory devices or other types of targets connected to OCP-T1, OCP-T2, EMIFF, and EMIFS.

System targets are:

- EMIFS memory interface. This memory controller supports most common memory interface protocols through a flexible programming and timing control that also allows support of any type of internal or external IC target module.
  - External memory devices like asynchronous ROM, RAM, flash, or synchronous burst flash.
  - Internal TI ACE ROM, RAM memories.
  - Internal and external asynchronous target modules
- EMIFF memory interface. This memory controller supports most common SDRAM interface protocols through a flexible programming and timing control.
  - External synchronous standard single-data rate (SDR SDRAM) and low-power SDR SDRAM.
  - External synchronous standard double-data rate (DDR SDRAM) and mobile DDR SDRAM.
- OCP-T port interface. OMAP includes two ports on which OCP-compliant slaves can be connected. These two target ports are called OCP-T1 and OCP-T2. Attached target modules can be either internal or external memory subsystems or any type of target peripheral.

Figure 2 shows the traffic controller functional block diagram:

Figure 2. Traffic Controller Functional Block Diagram



3.1 OCP-T1/OCP-T2

The OMAP 3.2 hardware provides two identical general-purpose ports for the connection of custom peripherals or memory subsystems. These ports can be accessed by all the initiators mentioned in the previous sections.

Each port arbitrates between all initiator requests and allows atomic transactions for the MPU, for the DSP, and for the external initiator to implement semaphore-based synchronization schemes between software tasks (running either on the same CPU or on two different CPUs).

Once granted, the selected request is converted from an OMAP internal protocol into an OCP-compliant transaction. As seen at the OMAP boundary, the OCP-TX is an OCP master.

OCP-T1 and OCP-T2 have their own addressing spaces within the OMAP memory map (256M bytes accessible by OCP-T1 and 1.2 gigabytes accessible by OCP-T2) plus an additional common address range called the multibank address space (256M bytes). The multibank address space is used in some devices for connecting a dual-port OCP slave, such as a memory controller, that can provide concurrent data flow to/from memory banks.

A request that targets the multibank address space is analyzed and can be routed either to OCP-T1 or OCP-T2 by the traffic controller.This mechanism is transparent to the programmer.

OCP-T1 and OCP-T2 both implement the arbitration schemes described in Section 3.9, *Priority Algorithms*.

Bus error transactions that generate abort signals on the external OCP interconnect can be tracked within the OCP-T1/T2 controllers.

Two types of errors can be detected:

- Bus error returned by the OCP interface
- Time-out error. A request has been sent to the interface but has not been completed in the time-out delay. The timer is programmable from 0 to 255 TC clock cycles.

The abort interrupt handler can read the error reporting registers, which indicate the address that caused the error, the initiator at the origin of the request, and the type of error (see abort address register and abort type register, in Section 3.8).

3.2 EMIFS Programming

The synchronous/asynchronous external memory interface slow (EMIFS) supports most common memory interface protocols through a flexible programming and timing signals control.

- The EMIFS can control up to six devices without adding any external logic through the four independent chip-selects ($\overline{\text{FLASH.CS0−3}}$) (plus two optional chip selects within CS1 and CS2) and through dedicated memory interface control signals.
- The EMIFS supports common external memory control signals: OE, WE, ADV, BE[0−1], BAA, Ready, Device CLK, RST, and WP.

- Each chip-select controls an address range of 64M bytes with dedicated configuration registers to fulfill the protocol and the timing constraints compliant with the external device associated with it. Each chip-select configuration register supports dynamic configuration.
- CS0 and CS3 address mapping can be swapped with the BM bit in the EMIFS configuration (EMIFS_CONFIG) register.
- The EMIFS can support 16-bit interface width only. Based on the CS configuration, the interface adjusts the access size (splitting word32) according to external device attached to the CS. **User must refer to the IC device documentation for the external device width supported by the IC (16-bit width only or 16-bit and 32-bit width).**8-bit device width is not supported without adding external logic.
- The EMIFS can control multiplexed address and data memory devices without adding external logic based on CS configuration. The multiplexing scheme is supported for synchronous and asynchronous access mode. Both multiplexed and non-multiplexed devices can be supported with the same IC on different chip-selects (embedded IC non-multiplexed memories and external multiplexed devices).
- The EMIFS behavior conforms to the little endian protocol.
- The EMIFS supports 8-,16-, or 32-bit asynchronous and synchronous read, 4- x 32-bit synchronous burst read and 8-, 16-, or 32-bit asynchronous write.
- The EMIFS boot configuration is controlled by dedicated pins that are sampled at IC reset time. This provides flexible boot CS and configuration selection.
- The EMIFS is a multimaster memory interface. It supports flexible and programmable arbitration protocol (LRU priority ordering or dynamic time based priority ordering).
- The EMIFS includes a programmable time-out timer to prevent system hanging with nonresponding devices. Automatic access completion with interrupt and status logging are issued on time out events.
- The EMIFS supports dynamic local idle mode control. The EMIFS also supports IC deep power-down mode request synchronization.

3.2.1 General Description

EMIFS Synchronous and Asynchronous Modes

- The operation mode of the EMIFS for a given chip-select region is selected by the RDMODE bit field of the CS configuration registers. Operations supported are:

- Mode 0. Asynchronous read. Used for any asynchronous memory, including flash devices.
- Mode 1−2−3. Asynchronous page mode read with control of 4 (mode1), 8 (mode 2), 16 (mode 3) words (device width) per page. These modes are mainly used for page mode flash devices.
- Mode 4−5. Synchronous burst read (with burst advance control for mode 4). These modes are mainly used for synchronous burst flash devices.
- Mode 7. Synchronous pipelined burst read. This mode is mainly used for TI embedded IC ROM and RAM memories.

For all these modes write accesses are performed according to asynchronous write protocol.

- Single and burst access address alignment
  - OMAP master only issues Word16 and Word32 aligned access (word address must be aligned on word size address boundary).
  - OMAP master only issues linear, incrementing, and fixed size 4xWord32 access bursts. Burst access is aligned on burst size address boundary (starting burst LSB address A[0−3] is always equal to [0000]). External devices like synchronous flash memory may require a burst protocol programming to conform to the EMIFS burst protocol.

EMIFS Memory Timing Control

- In both asynchronous and synchronous modes, all EMIFS to memory control signals are controlled with an EMIFS internal reference clock (REF_CLK) which is the TC_CK divided down. Depending on the CS configuration, this internal clock can be available outside through the FLASH.CLK (ball N3) output pin.
  - The REF_CLK is divided from TC_CK (traffic controller clock, see section 4.2.5) by a programmable value contained in FCLKDIV bit field of the CS configuration register, Table 19. This accommodates the timing constraints of slow devices, even with high system clock rate.
- In asynchronous mode 0−1−2−3, the FLASH.CLK (ball N3) is low.
- In synchronous mode 4−5−7, REF_CLK is available through FLASH.CLK. This is also the case during asynchronous write access. FLASH.CLK clock is connected to the external synchronous device input clock

(synchronous flash or ASIC). The frequency must be set to comply with the attached device's timing constraints in combination with the others EMIFS timing controls.

- In synchronous mode 4−5, a retiming mode enables read data to be latched by a delayed REF_CLK. This retiming mode must not be used with asynchronous modes.
  - The REF_CLK delay is obtained through the IC I/O feedback of FLASH.CLK to offer optimum data and clock alignment. Pipelined access offers relaxed timing constraints.
  - The retiming mode is enabled through RT bit field in the CS configuration register (See Table 19).
- Memory control signal (CS, OE, WE, ADV, ADDRESS) setup and hold timing can be controlled by programmable internal delay generation.
  - OE valid/invalid timing from/to CS and address valid/invalid is programmable through OESETUP, OEHOLD bit field in the advanced CS configuration register.
  - WE valid timing from CS, address, and data valid is programmable through WRWST bit field in the CS configuration register. WE hold time is fixed to one REF_CLK.
  - ADV valid pulse time is programmable through ADVHOLD bit field in the advance CS configuration register.
    - In synchronous read mode, the ADV setup to REF_CLK rising edge is controlled by the ADVHOLD bit field. The ADV hold time from REF_CLK rising edge is fixed to one TC_CK.
  - Address setup and hold time with ADV control:
    - In synchronous mode, the address setup time from REF_CLK rising edge (ADV valid) is controlled by ADVHOLD bit field.
    - In multiplexed synchronous mode, the address hold time from REF_CLK rising edge is fixed to one REF_CLK from ADV invalid time.
    - In asynchronous mode, the address setup time to ADV invalid is controlled by ADVHOLD bit field.
    - In asynchronous mode, address hold time from ADV invalid is fixed to one REF_CLK.
- Read and write access time is controlled by programmable internal wait state generation (non-full-handshaking mode). An external Ready input

pin (FLASH.RDY on ball V2) can also be used in combination with internal wait state (full-handshaking mode).

- In non-full-handshaking mode, the RDWST and PGWST (mode 1−2−3 only) bit field in CS configuration register are used to control internal read wait state generation.
- In non-full-handshaking, the WELEN bit field in CS configuration register is used to control internal write wait state generation.
- In full-handshaking mode, FLASH.RDY is monitored by the EMIFS to control read and write access time. The access is completed when both the internal wait state has expired and FLASH.RDY is asserted by the external device. The FLASH.RDY assertion/deassertion timing constraint depends on synchronous or asynchronous access mode.
- Modes 0−4−5 are by default in full-handshaking mode. Full-handshaking support on particular CS can be disabled for these modes through the dynamic wait state register.

  Modes 1−2−3−7 always follow the non-full-handshaking protocol and FLASH.RDY is never monitored in these modes, even if the full-handshaking bit field in the dynamic wait state register is cleared.

- To prevent data bus contention when slow devices are attached to the IC, the BTWST bit field in the CS configuration register is used to control TC_CK cycle idle time between specific access sequences.
- The BTMODE field in the advance CS configuration register extends the previous mode. The BTWST bit field controls the CS negation time between successive accesses to the same CS.
- To prevent data bus floating when no access is requested at the interface (idle sequence), the EMIFS keep the data bus driven with the previous written data or read data (bus keeping feature). In case of read to idle sequence, the delay time from read to write-back is at least one TC_CK cycle or controlled by BTWST (TC_CK) cycles.
- If dynamic Wait state mode is used, then one REF_clk cycle must be added to all of the formulas describing CS and ADV width in this section.

3.2.2 EMIFS CS0 and CS3 Decoding Control

CS0 and CS3 address decoding (address in the TC memory mapping) can be swapped through the BM bit field in the EMIFS global control register. When the BM bit field is set, CS3 is activated in the 0000:0000−03FF:FFFF range and CS0 is activated in the 0C00:0000−0FFF:FFFF range. The BM bit is sampled at reset depending on two factors. If MPU_BOOT (ball J20) is 1 and

the device type is emulation then BM resets to 1. Otherwise, BM resets to 0. Thus, the boot is executed from CS0 or from CS3 attached memories. During normal execution, BM can be changed dynamically but obvious software precautions are required to prevent system crash.

3.2.3 EMIFS Miscellaneous Memory Signal Control

Common flash memory supports the write protection, WP, input pin ($\overline{\text{FLASH.WP}}$ on ball V4) to prevent erroneous write access sequence to corrupt their content. EMIFS interface includes WP output pin and provides full software control of it.

Common flash memory supports the RESET input pin to allow device state machine to be properly reset on power up and also to minimize power consumption in idle mode. The EMIFS interface includes a $\overline{\text{FLASH.RP}}$ (on ball W1) output pin and provides software and hardware control of it.

3.2.4 EMIFS Configuration

EMIFS control and configuration can be done dynamically. EMIFS ensures that a new CS configuration takes effect only when no access is requested (CS idle). To prevent inconsistency and critical behavior, the EMIFS configuration must be done by MPU software while other masters are inactive.

3.2.5 EMIFS Abort Control

In case of access restriction violation, and in case of access time-out errors, the EMIFS can handle an interrupt line and abort status register to allow abort events system monitoring.

3.2.6 EMIFS External Device Connections

Please refer to Chapter 9 for a detailed description of the EMIFS device connections.

3.2.7 EMIFS Address Mapping and Data Control in Multiplexed Mode

- In order to minimize the number of IC pins for external memory connection, the EMIFS can support multiplexed address and data memory devices without adding external logic.
- Selection of the multiplexed mode is done through the MAD bit in the CS configuration register. Multiplexed mode is only available in the following modes:
  - Mode 0 asynchronous read and write

- Mode 4 & mode 5 synchronous burst read

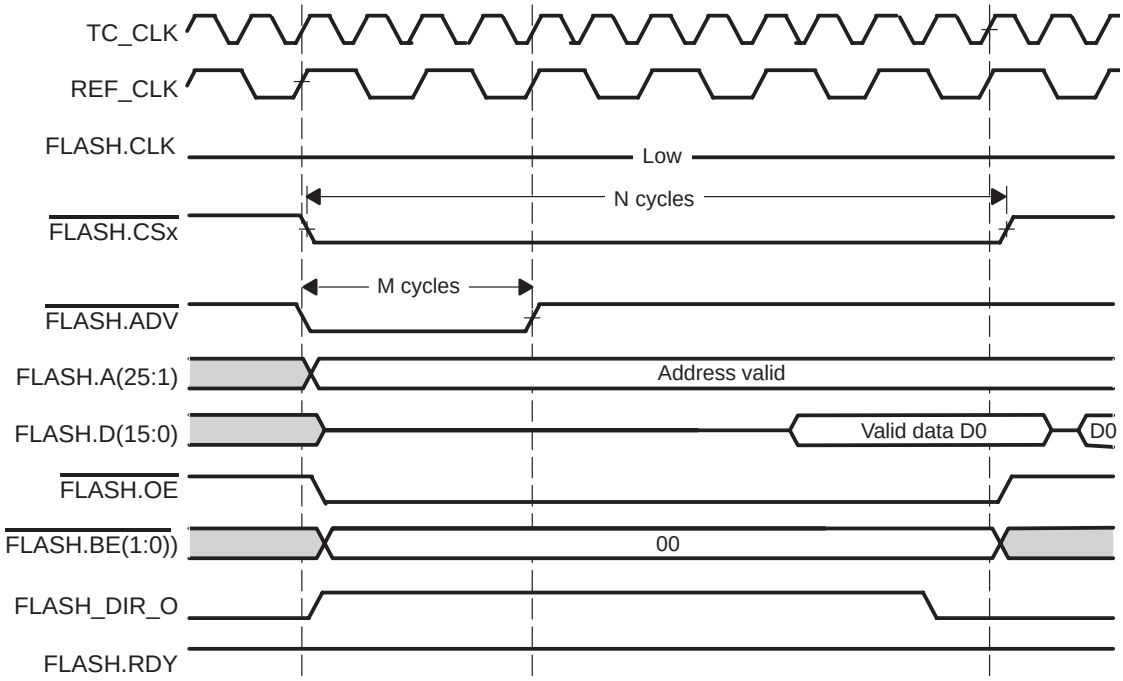
3.2.8 Mode 0—Asynchronous Read Operation

Basic Programming Model

The asynchronous read mode 0 is selected by setting RDMODE = 0 in the corresponding EMIFS chip-select configuration register.

Figure 3 shows a typical timing diagram.

Figure 3. Asynchronous 16-Bit Read Operation on a 16-Bit Width Device. RDWST=2 FCLKDIV=0 OESETUP = 0 OEHOLD = 0 ADVHOLD = 0. Data write-back on the bus after read completion.



- The REF_CLK is divided from TC_CK (traffic controller root clock) by a programmable value contained in FCLKDIV bit field of the CS configuration register.

FCLKDIV	REF_CLK
00	TC_CK:1
01	TC_CK:2
10	TC_CK:4
11	TC_CK:6

- The CS pulse width depends on RDWST bit field of the CS configuration register. CS pulse width equals:
  - (RDWST + 2) REF_CLK (N cycles in Figure 3)
  - CS minimum pulse width is 2 REF_CLK.
- The ADV pulse width depends on ADVHOLD bit field of the CS configuration register. ADV pulse width equals:
  - (ADVHOLD + 1) REF_CLK (M cycles in Figure 3)
  - ADV minimum pulse width is 1 REF_CLK.
- Address drive time follows CS activation (no setup time guaranty). Address setup time to ADV rising edge is controlled by ADVHOLD. Address hold time from ADV rising edge is controlled by CS pulse width.
- Read data is latched on same TC_CK rising edge that deactivate OE signal.
- After a read completion, if no other access (RD, WR) is pending, the data bus is driven with the previous read value. The bus turn-around time (OE going high to direction going out) is a minimum of 1 TC_CK cycle and can be extended through BTWST
- In asynchronous mode, REF_CLK is not provided outside the EMIFS and Flash_clk_o is kept low.

Figure 4. Asynchronous 16-Bit Read Operation on a 16-Bit Width Device. RDWST=4 FCLKDIV=1 OESETUP=0 OEHOLD=0 ADVHOLD=1. Data write-back on the bus after read completion.

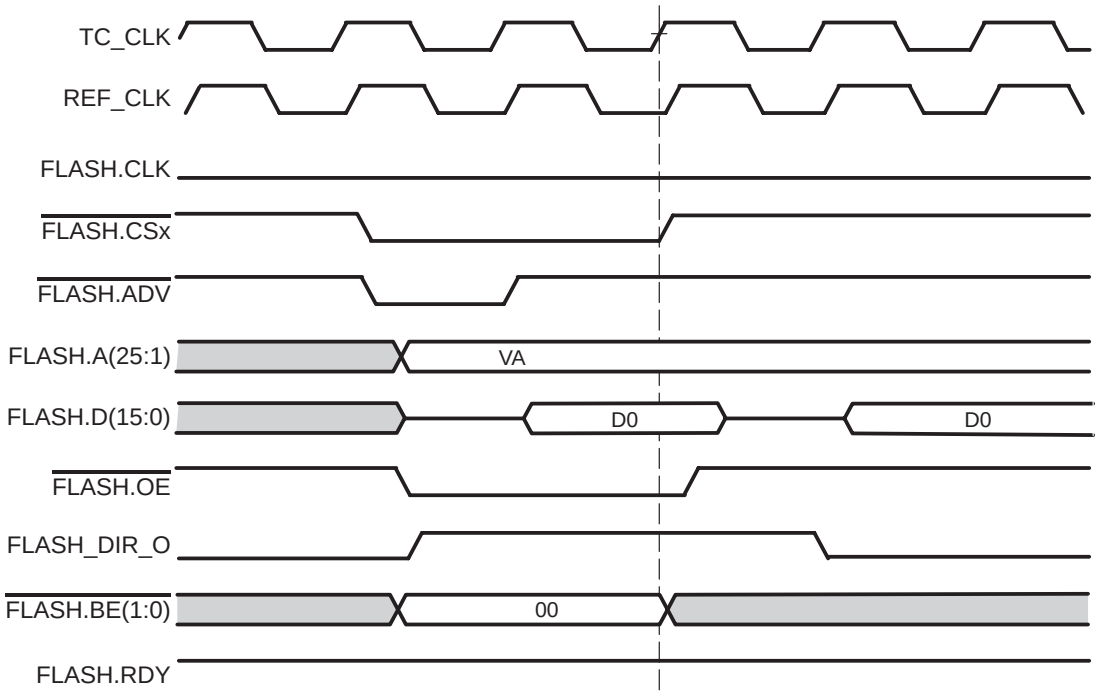
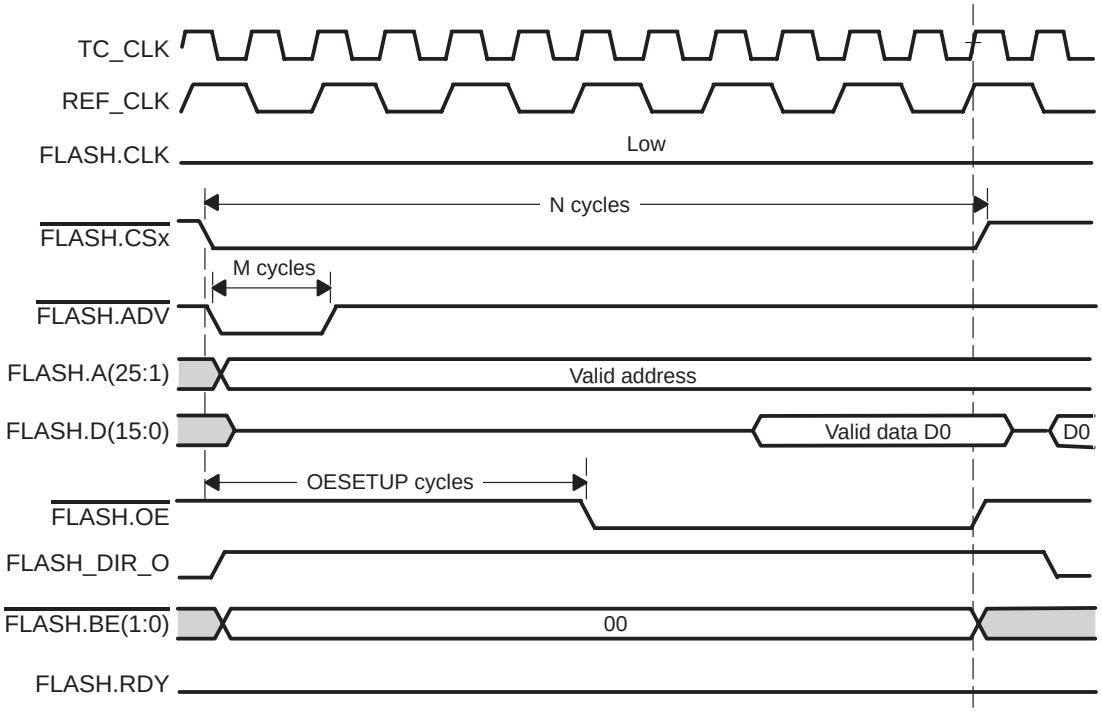


Figure 5. Asynchronous 16-Bit Read Operation on a 16-Bit Width Device. RDWST=0 FCLKDIV=0 OESETUP=0 OEHOLD=0 ADVHOLD=0. Data write-back on the bus after read completion.



Advanced OE Control

- OE activation delay time from CS and address valid is programmable through OESETUP bit field in the advanced CS configuration register. Activation delay timing is equal to:
  - (OESETUP) REF_CLK (OESETUP cycles in Figure 6).
- OE deactivation advance time to CS and address invalid is programmable through OEHOLD bit field in the advanced CS configuration register. Deactivation advance timing is equal to:
  - (OEHOLD) REF_CLK (OEHOLD cycles in Figure 6).
- Because CS minimum pulse width is 2 REF_CLK the OE delay and advance timing value must be set so:
  - (OESETUP + OEHOLD) is inferior or equal to RDWST. Noncompliant programming ends up with bad access completion.

Figure 6. Asynchronous 16-Bit Read Operation on a 16-Bit Width Device. RDWST=4 FCLKDIV=1 OESETUP=3 OEHOLD=0 ADVHOLD=0. Data write-back on the bus after read completion.

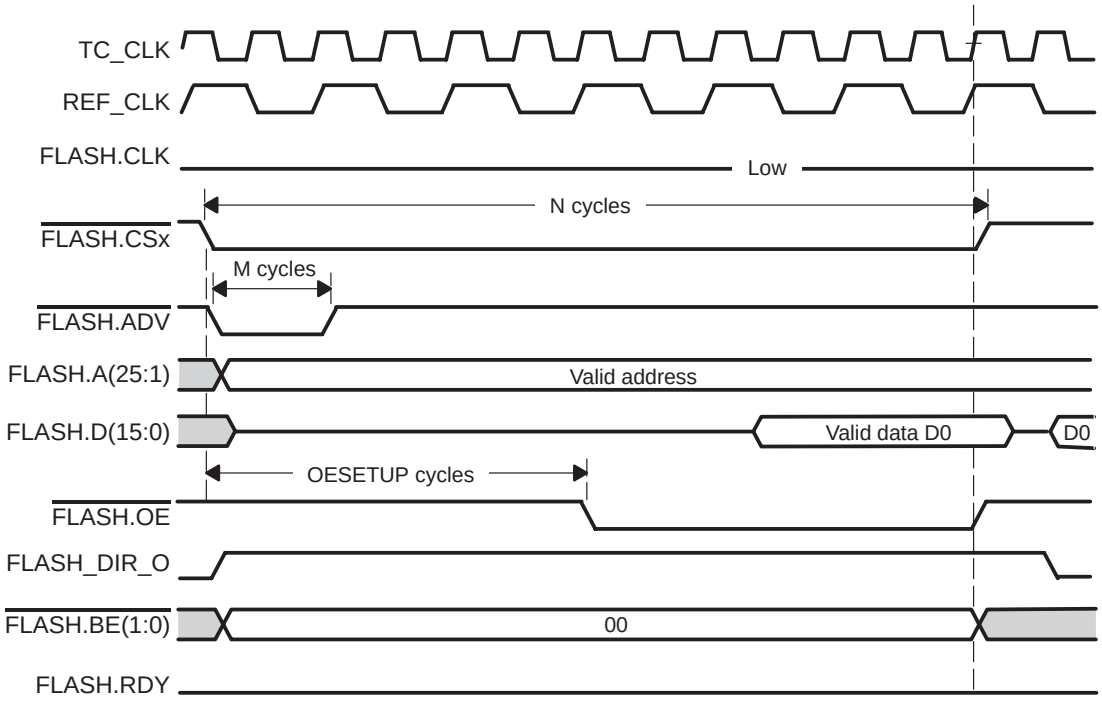
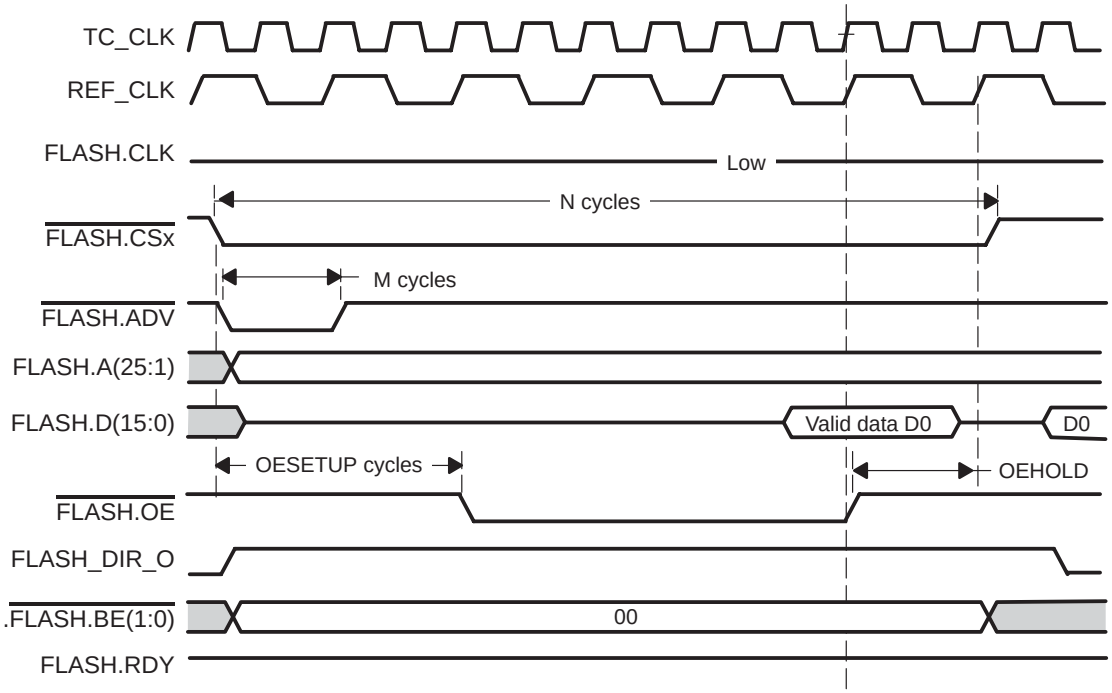


Figure 7. Asynchronous 16-Bit Read Operation on a 16-Bit Width Device. RDWST=4 FCLKDIV=1 OESETUP = 2 OEHOLD = 1 ADVHOLD = 0. Data write-back on the bus after read completion.



Read Access Size Adaptation and CS Pulse Width High Control

- In read mode 0, the EMIFS splits the Word32 access into two Word16 accesses in case of 16-bit device width. 4xWord32 burst read are split into eight successive Word16 accesses. The split process follows the little endian protocol (Word32 LSB part at lower Word16 address).
- During split read accesses and during burst read accesses, the CS signal is deactivated for at least one TC_CK between two successive accesses. CS pulse-width high time can be extended by the BTWST field in the CS configuration register (see also bus turn around and CS negation time control).
  - CS pulse width high = (BTWST +1) TC_CK

Figure 8. Asynchronous 32-Bit Read Operation on a 16-Bit Width Device. RDWST=4 FCLKDIV=0 OESETUP = 0 OEHOLD = 0 ADVHOLD = 0 BTWST=0 BTMODE=0. Data write-back on the bus after read completion.

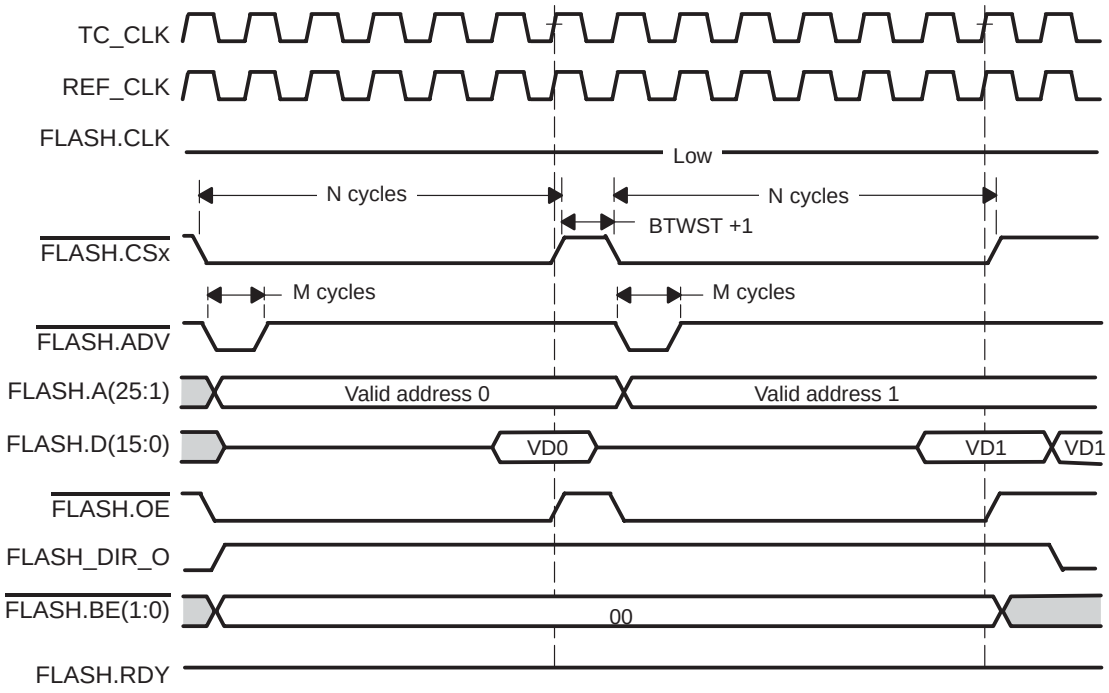
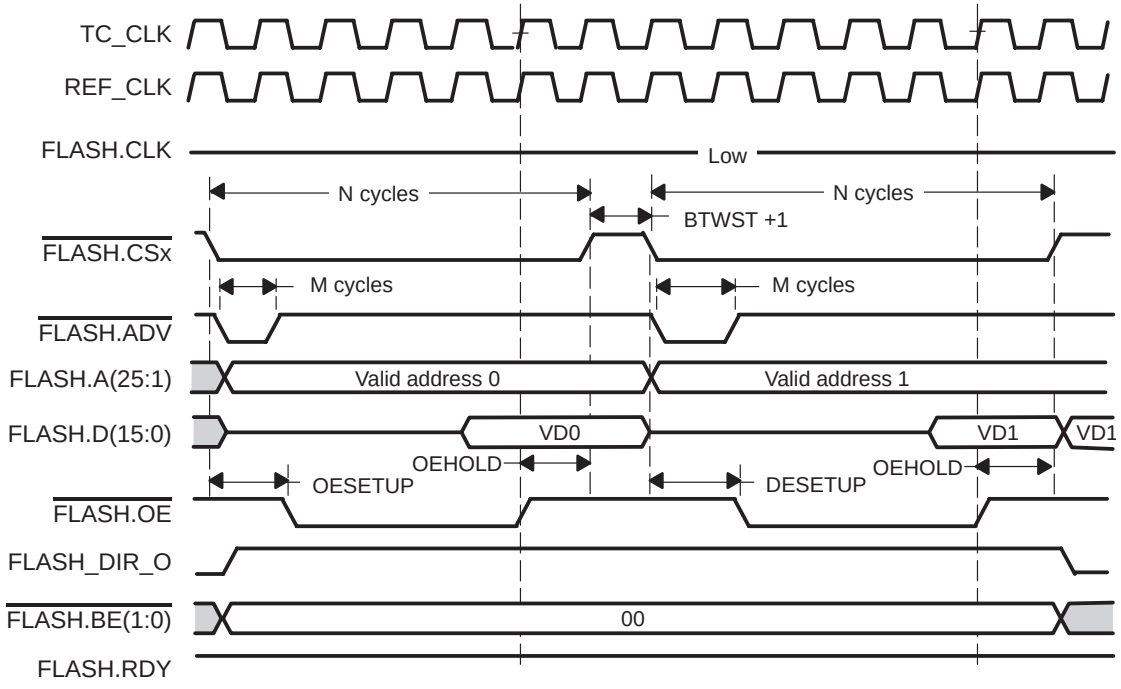


Figure 9. Asynchronous 32-Bit Read Operation on a 16-Bit Width Device. RDWST=4 FCLKDIV=0 OESETUP = 1 OEHOLD = 1 ADVHOLD = 0 BTWST=0 BTMODE=0. Data write-back on the bus after read completion.



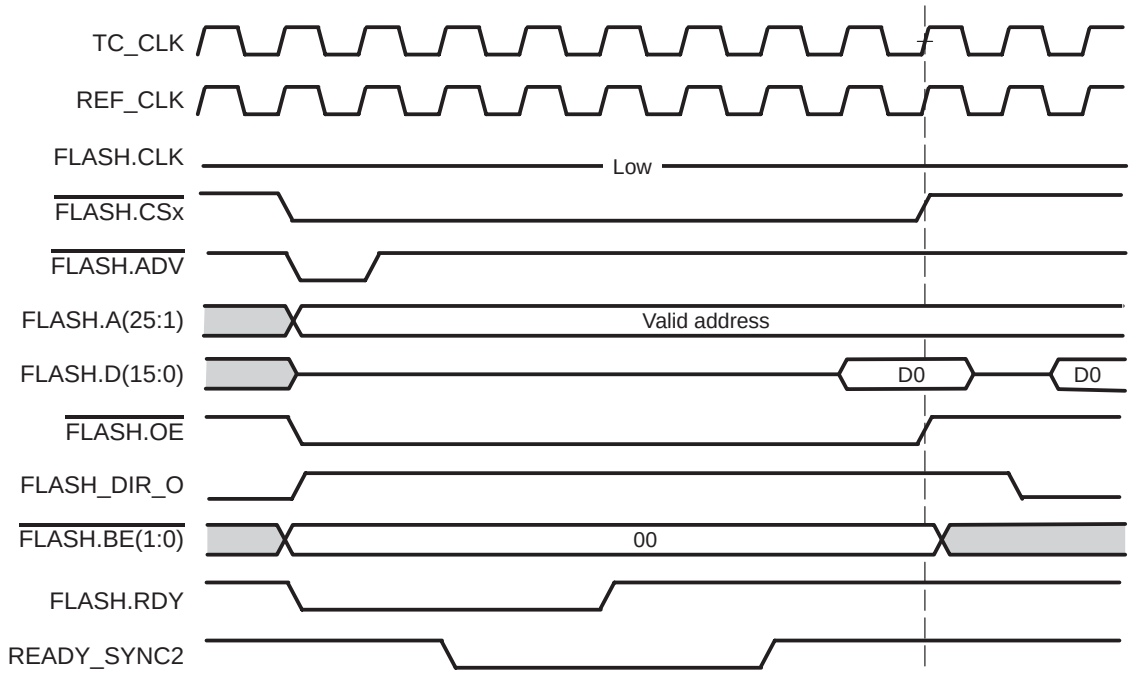
Full-Handshaking and Ready Pin Usage in Asynchronous Read Mode

- In full-handshaking mode, the READY input pin (FLASH.RDY on ball V2) is monitored by the EMIFS to control read access time. The access is completed when internal wait state RDWST expires and FLASH.RDY is asserted by the external device.
- ADV pulse width time and OE assertion time are not dependent on the FLASH.RDY pin state.
- When FLASH.RDY is used to extend the access time, the access completion is not controlled by internal delay generation. CS, OE, and address are deactivated when FLASH.RDY is detected high. No OEHOLD time can be control in this case and the bit field must be equal to zero.
- Since FLASH.RDY is an asynchronous signal, a nonready device must drive FLASH.RDY low enough time ahead of the minimum access time completion. Depending on FLASH.RDY assertion low delay from CS

active and depending on REF_CLK frequency, a minimum RDWST value may be needed for the FLASH.RDY state to be correctly monitored by the EMIFS.

- As an example, a minimum of RDWST = 2 is needed for a nonready device that drives FLASH.RDY low with 0 time delay from CS low and for a CS configuration FCLKDIV = 0.
- See the OMAP5912 Data Manual (SWPS012) for timing information regarding FLASH.RDY assertion timing constraint.

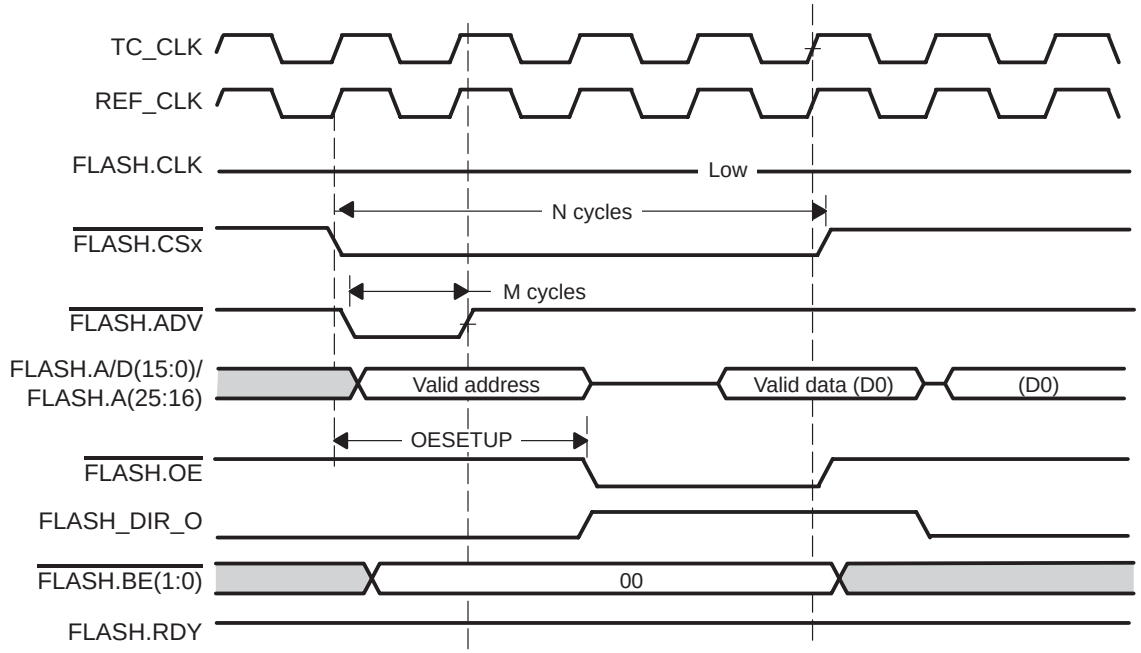
Figure 10. Asynchronous 16-Bit Read Operation with Ready. RDWST=2 FCLKDIV=0 OESETUP = 0 OEHOLD = 0 ADVHOLD = 0. Data write-back on the bus after read completion.



Asynchronous Read With Multiplexed Address and Data Memory

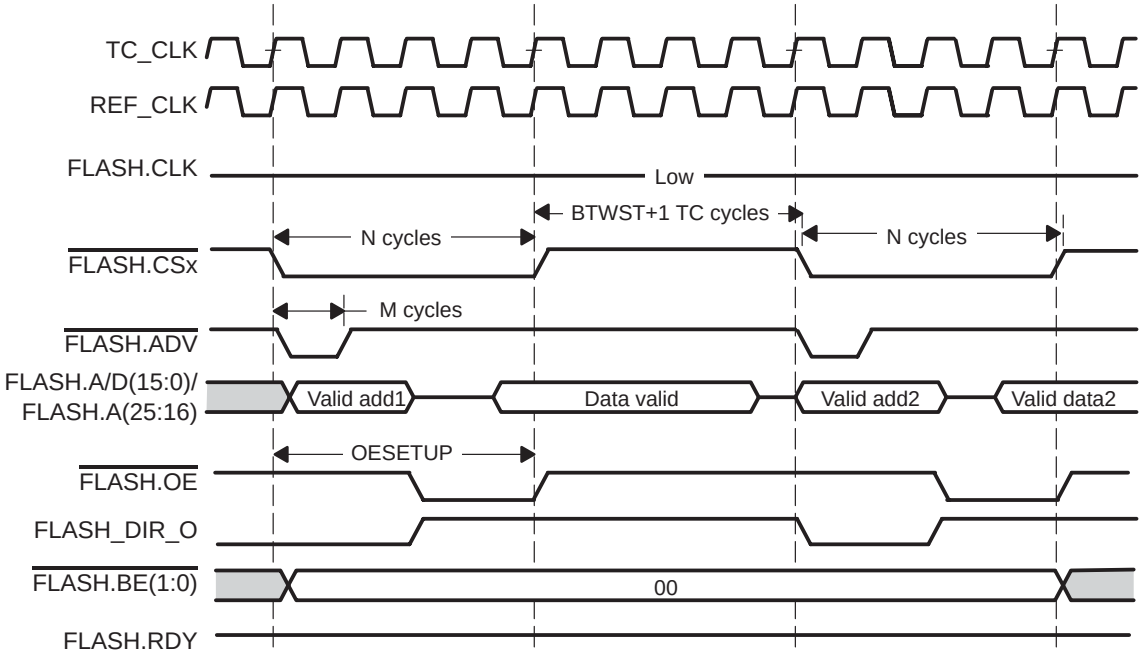
- The EMIFS can support multiplexed address and data memory devices without adding external logic. Multiplexed mode is enabled when the MAD bit field in the EMIFS CS configuration register is set to 1 (see Table 19).
- The following figure shows an asynchronous read operation with multiplexed address/data bus.

Figure 11. Asynchronous 16-Bit Read Operation With Multiplexed Address/Data Bus Memory. RDWST=2 FCLKDIV=0 OESETUP=2 OEHOLD=0 ADVHOLD=0. Data write-back on the bus after read completion.



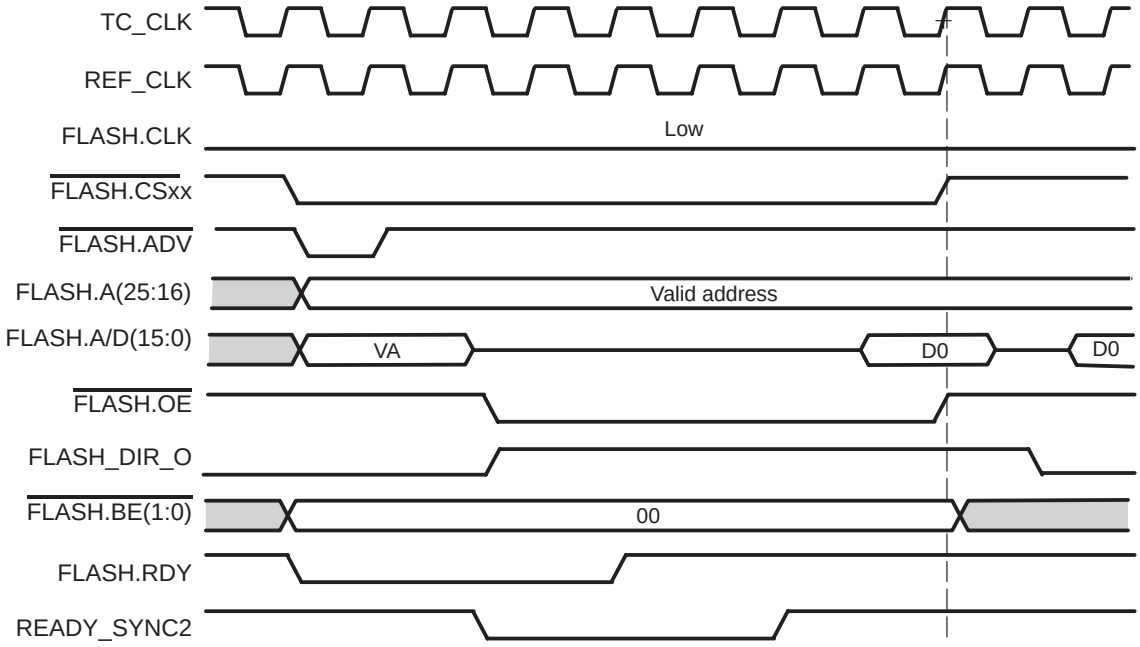
- Address drive time follows CS activation (no setup time guaranty). Address setup time to ADV rising edge is controlled by ADVHOLD. Address hold from ADV rising edge is guaranteed to be minimum one REF_CLK (delay for direction to change from out to in). FCLKDIV and OESETUP must be properly programmed to prevent bus contention and to ensure that address hold time device requirement is respected.
- During split read accesses and during burst read accesses, the CS signal is deactivated for at least one TC_CK between two successive accesses. CS pulse width high time can be extended by the BTWST field in the CS configuration register (see also bus turn around and CS negation time control).
  - CS pulse width high = (BTWST +1) TC_CK

Figure 12. Asynchronous 32-Bit Read Operation on a 16-Bit Multiplexed Address and Data Memory. RDWST=2 FCLKDIV=0 OESETUP=2 OEHOLD = 0 ADVHOLD = 0



- The full-handshaking scheme is also valid in multiplexing mode. The following diagram shows an asynchronous word16 read operation with a 16-bit multiplexed memory control by external ready pin.

Figure 13. Asynchronous 16-Bit Read Operation with Ready on 16-Bit Multiplexed Address and Data Memory. RDWST=2 FCLKDIV=0 OESETUP=2 OEHOLD = 0 ADVHOLD = 0 BTWST = 0, BTMODE = 0

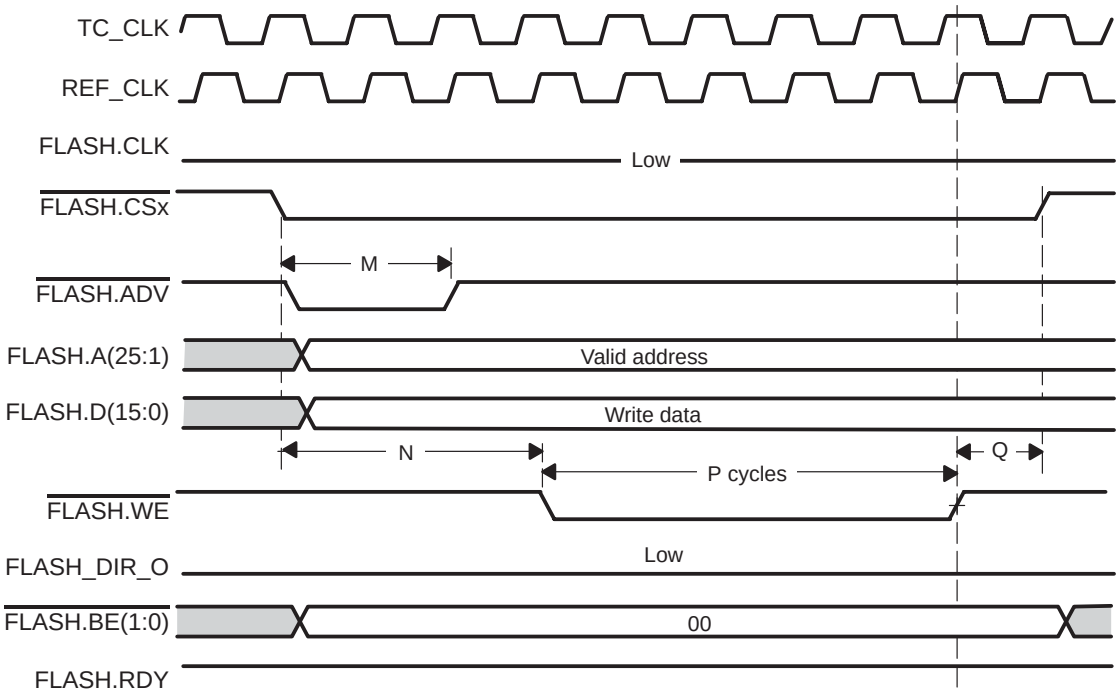


3.2.9 Asynchronous Write Operation

Non-Multiplexed Asynchronous Write Operation

- The asynchronous write is the only write protocol supported by the EMIFS. The asynchronous write access protocol is used in all EMIFS modes, whatever the CS configuration register RDMODE value is.

Figure 14. Asynchronous 16-Bit Write Operation on a 16-Bit Width Device (WRWST=2, WELEN=4 FCLKDIV=00 and ADVHOLD=1)



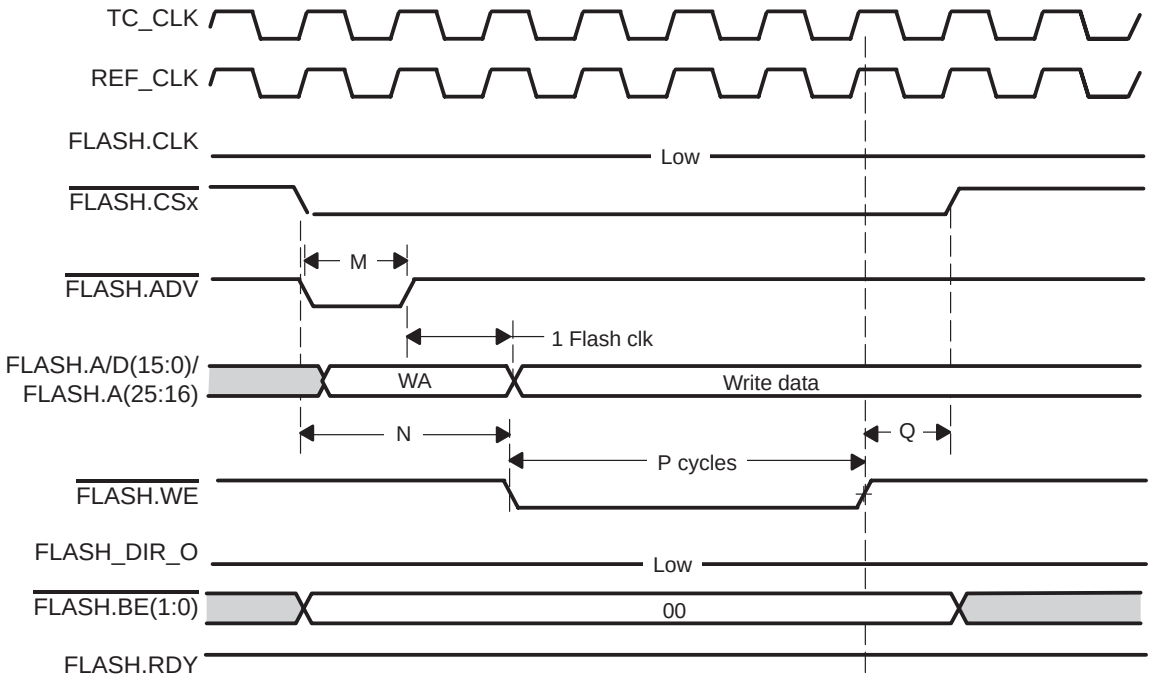
- The REF_CLK is divided from TC_CK by a programmable value contained in FCLKDIV bit field of the CS configuration register.
- The CS and address setup time from WE low is controlled by programmable WRWST bit field of the CS configuration register.
  - (WRWST + 1) REF_CLK (N cycles in Figure 14)
  - WRWST minimum pulse width is 1 REF_CLK.
- The ADV pulse width depends on ADVHOLD bit field of the Advanced CS configuration register (see Table 28). ADV pulse width equals:
  - (ADVHOLD + 1) REF_CLK (M cycles in Figure 14)
  - ADV minimum pulse width is 1 REF_CLK.
- The WE pulse width depends on WELEN bit field of the CS configuration register (see Table 19). WE pulse width equals:
  - (WELEN + 1) REF_CLK (P cycles in Figure 14)
  - WE minimum pulse width is 1 REF_CLK.

- The CS address and data hold time setup from WE high is fixed to one REF_CLK (Q cycle in Figure 14).
- In asynchronous mode 0−1−2−3, REF_CLK is not provided outside the EMIFS and FLASH.CLK is kept low. In synchronous mode 4−5, REF_CLK is provided outside the EMIFS through the FLASH.CLK (see mode 4,5). In synchronous mode 7, REF_CLK is inverted and provided outside the EMIFS through FLASH.CLK (see mode 7).

Multiplexed Asynchronous Write Operation

Figure 15 shows a timing diagram with multiplexed address/data bus.

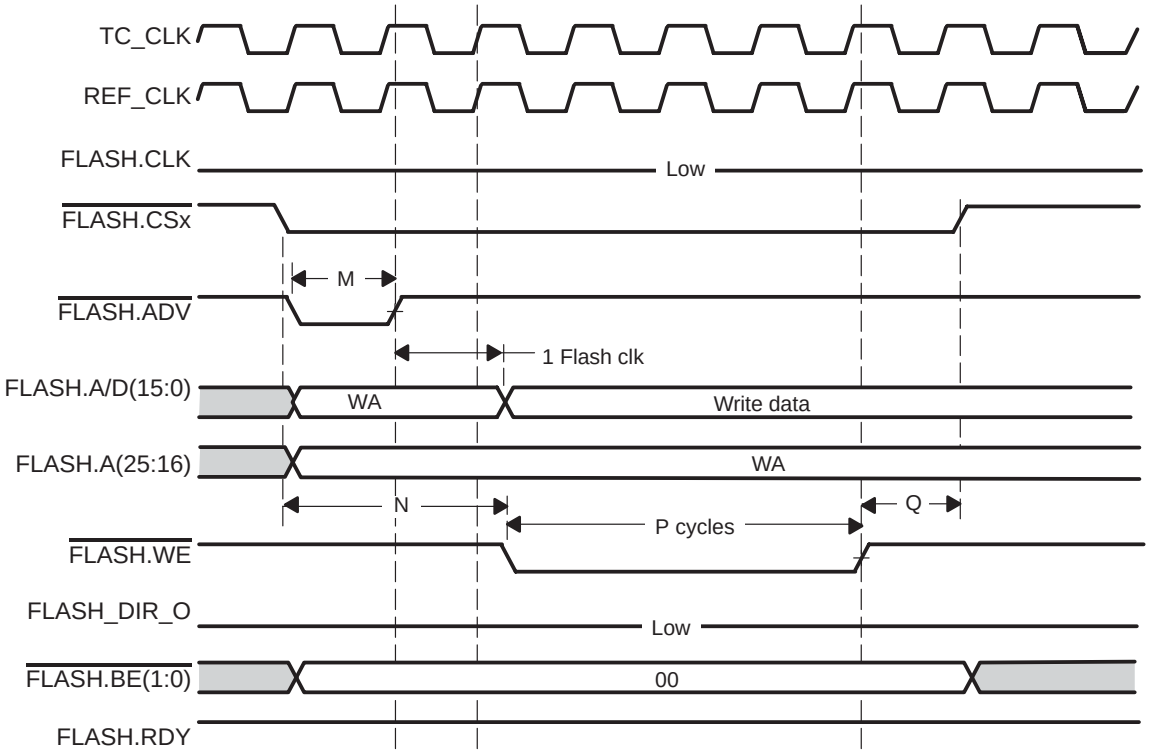
Figure 15. Asynchronous 16-Bit Write Operation on a Multiplexed Address/16-Bit Data Bus (WRWST=1, WELEN=3 , FCLKDIV=00 and ADVHOLD=0)



- Multiplexed mode is enabled when MAD bit field in CS configuration register is set (see Table 19).
- Address drive time follows CS activation (no setup time guaranty). Address setup time to ADV rising edge is controlled by ADVHOLD. Address invalid from ADV rising edge is guaranteed to be minimum one REF_CLK cycle.

- The CS and address setup time from WE low is controlled by programmable WRWST bit field of the CS configuration register (see Table 19).
  - (WRWST + 1) REF_CLK (N cycles in Figure 15)
  - WRWST minimum pulse width is 1 REF_CLK.
- The ADV pulse width depends on ADVHOLD bit field of the advanced CS configuration register (see Table 28). ADV pulse width equals:
  - (ADVHOLD + 1) REF_CLK (Mcycles in Figure 15)
  - ADV min pulse width is 1 REF_CLK.
- The WE pulse width depends on WELEN bit field of the CS configuration register (see Table 19). WE pulse width equals to:
  - (WELEN + 1) REF_CLK (P cycles in Figure 15)
  - WE minimum pulse width is 1 REF_CLK.
- The CS and data hold time setup from WE high is fixed to one REF_CLK (Q cycle in Figure 15).
- In asynchronous mode 0−1−2−3, REF_CLK is not provided outside the EMIFS and FLASH.CLK is kept low. In synchronous mode 4−5 REF_CLK is provided outside the EMIFS through FLASH.CLK. In synchronous mode 7 REF_CLK is inverted and provided outside the EMIFS through the FLASH.CLK (see mode 7).

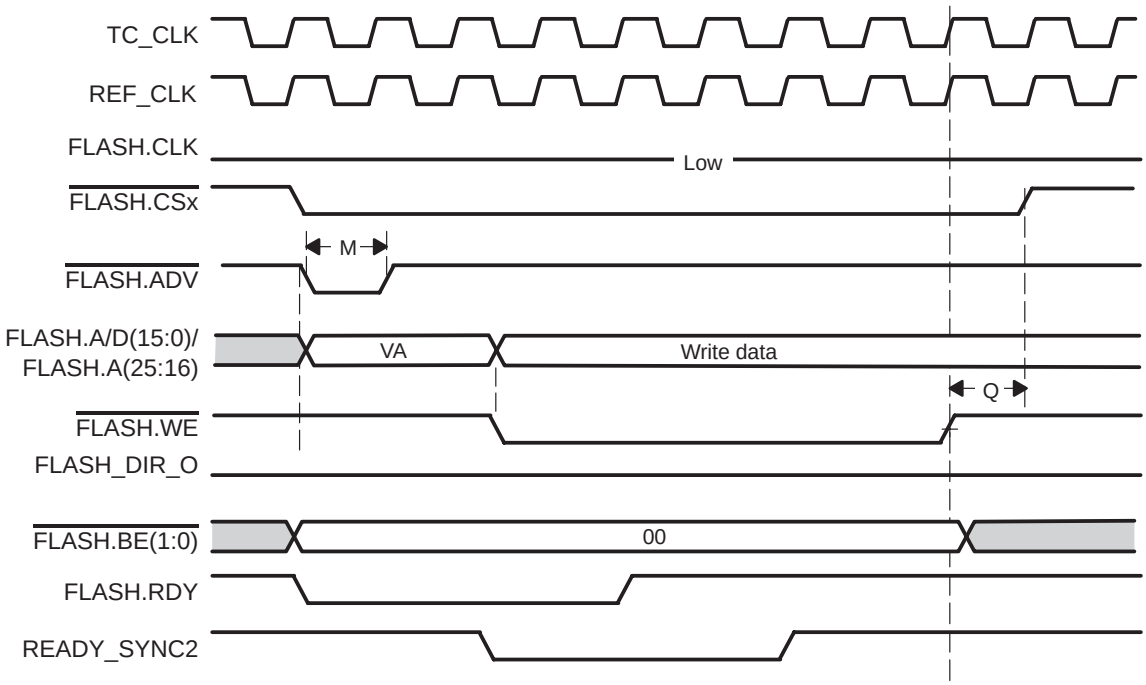
Figure 16. Asynchronous 16-Bit Write Operation on a Multiplexed Address/16-Bit Data Bus (WRWST = 1, WELEN = 3, FCLKDIV = 00 and ADVHOLD = 0)



Full-Handshaking and Ready Pin Usage in Asynchronous Write Mode

- In full-handshaking mode, FLASH.RDY is monitored by the EMIFS to control read access time. The access is completed when both internal wait state WELEN expires and FLASH.RDY (ball V2) is asserted by the external device. Full-handshaking is the default mode in both multiplexed and non-multiplexed modes.
- ADV pulse width time and WE assertion time are not dependent on the FLASH.RDY state.
- When FLASH.RDY is used to extend the access time, the access completion is not controlled by internal delay generation. The CS and data hold time setup from WE high is still fixed to one REF_CLK (Q cycle in previous figures).
- Because FLASH.RDY is an asynchronous signal, a nonready device must drive FLASH.RDY low enough time ahead of the minimum access time completion. Depending on FLASH.RDY assertion low delay from CS active and depending on REF_CLK frequency, a minimum WRWST value could be needed for the FLASH.RDY state to be correctly monitored by the EMIFS.
  - As an example, a minimum of WRWST=1 is needed for a nonready device that drives ready low with 0 time delay from CS low, for a CS configuration FCLKDIV=0.

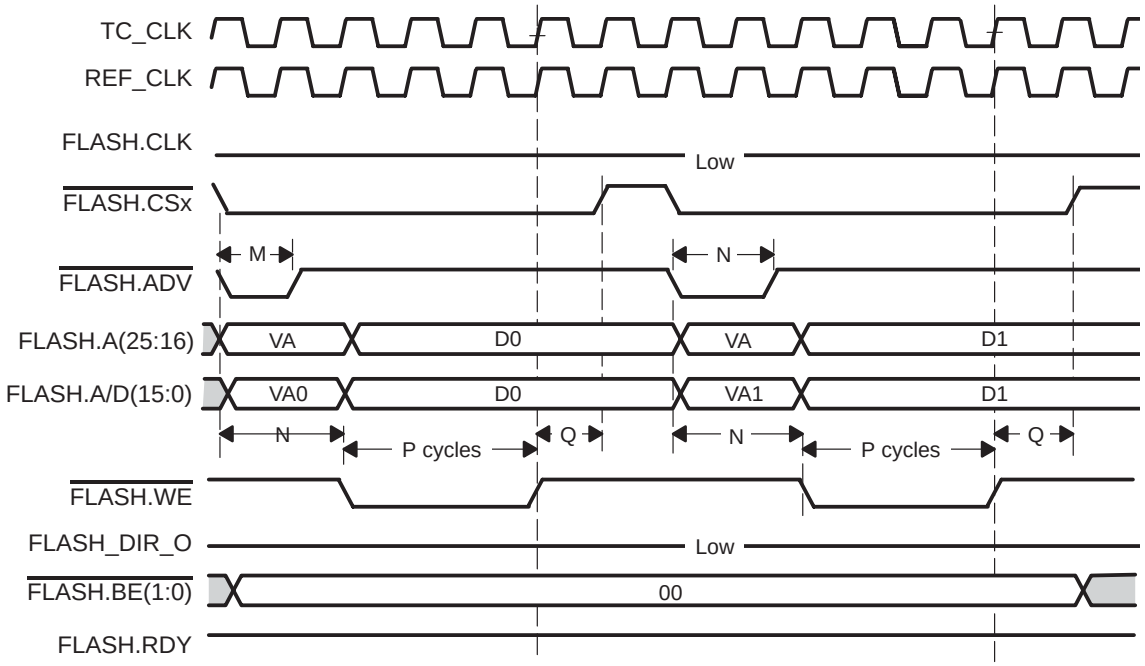
Figure 17. Asynchronous 16-Bit Write Operation on 16-Bit Multiplexed Address and Data Memory With Ready (WELEN = 2, WRWST = 0, FCLKDIV = 0)



Write Access Size Adaptation and CS Pulse Width High Control

- During write access, the EMIFS splits the Word32 access into two Word16 accesses in case of 16-bit device width. 4xWord32 burst write are split into 8 successive Word16 accesses. The split process follows the little endian protocol (Word32 LSB part at lower Word16 address).
- During split write accesses and during burst write accesses, the CS signal is not deactivated unless BTMODE in the Advanced CS configuration register is set (see Table 28). When BTMODE is set the CS pulse width high time can be controlled by the BTWST field in the CS configuration register, Table 19 (see also bus turn around and CS negation time control).
  - CS pulse width high= (BTWST +1) TC_CK
- This applicable to both multiplexed and non-multiplexed access modes.

Figure 18. Asynchronous 32-Bit Write Operation on 16-Bit Multiplexed Address and Data Memory (WELEN = 2, WRWST =1, FCLKDIV =0, BTWST = 0, and BTMODE = 1)



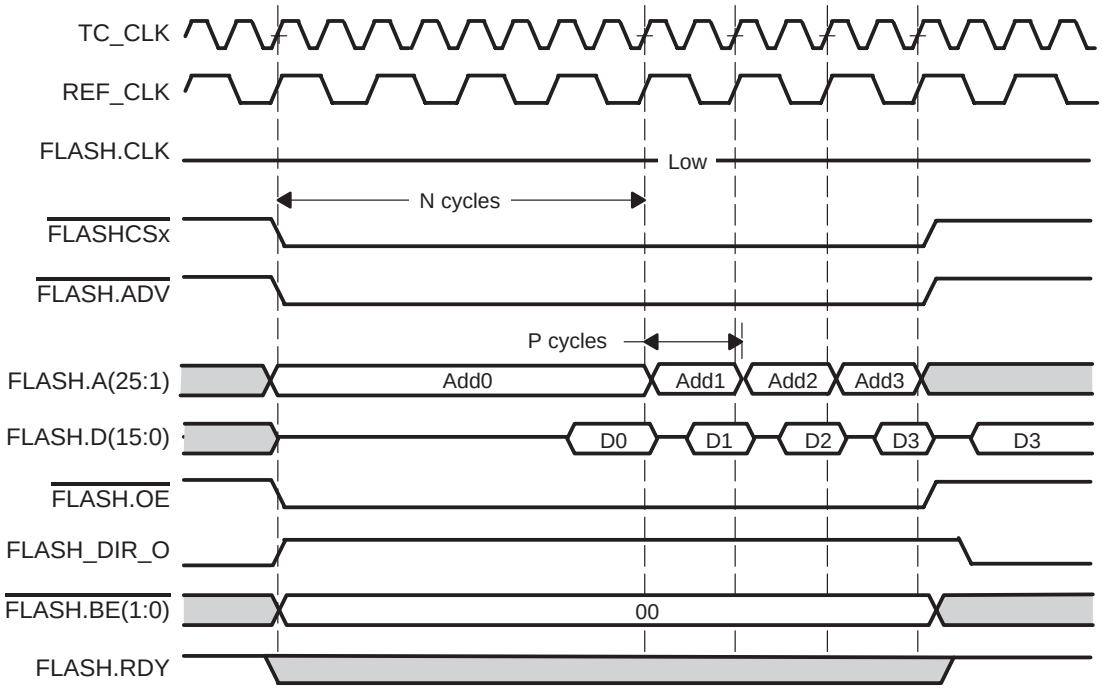
3.2.10 Mode 1−2−3− Asynchronous Page Mode Read Operation

- The asynchronous page mode read 1−2−3 is selected by setting the RDMODE bit field in the corresponding EMIFS chip-select configuration register.
  - RDMODE = 1 selects the 4 words per page mode
  - RDMODE = 2 selects the 8 words per page mode
  - RDMODE = 3 selects the 16 words per page mode
- This mode provides single access or fast consecutive accesses in a page. Optimized access time for 2×word16 read (word32 read in 16-bit width device) and for burst read (4×word32 or 8×Word16 in case of 16-bit width device). During consecutive accesses the EMIFS increments the address after each word read completion.
- The word length of the access is equal to the memory data bus width and is defined by the BW field of the CS configuration register (see Table 19).
- The delay for the first word in the page is controlled by RDWST bit field in the CS configuration register (initial wait state). Depending on the device

page length and word size (device width), the EMIFS can control device page crossing during a burst request (4×word32) and inserts initial wait state delay on purpose.

- The delay between successive words in the page is controlled by the PGWST bit field in the CS configuration register (in page wait state).

Figure 19. Asynchronous Page Mode 4x16-Bit Read Operation on 16-Bit Width Device (RDWST=2, PGWST=0 and FCLKDIV =1, RDMODE=2). Data write-back on the bus after read completion.

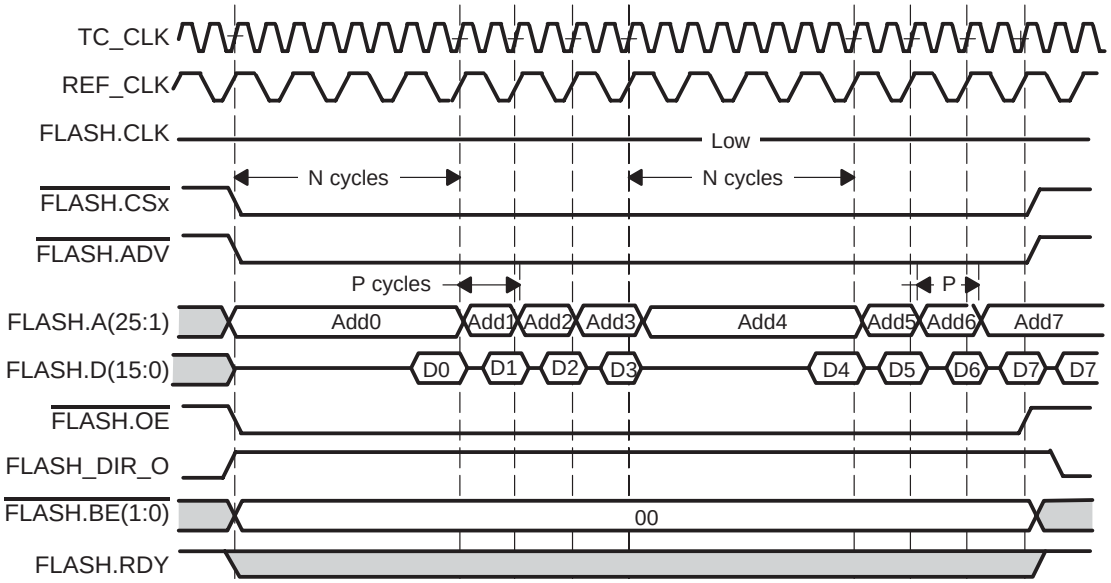


- The REF_CLK is divided from TC_CK by a programmable value contained in FCLKDIV bit field of the CS configuration register (Table 19).
- The initial wait state depends on RDWST bit field of the CS configuration register. Delay equals to:
  - (RDWST + 2) REF_CLK (N cycles in Figure 19)
- The in page wait state depends on:
  - PGWST/WELEN[15:12] bit field if PGWSTEN=0 in CS configuration register
  - PGWST[30:27] bit field if PGWSTEN=1 in CS configuration register

- Delay equals to (PGWST + 1) REF_CLK (P cycles in Figure 19).

- ADV (FLASH.ADV on ball L4) is kept low for the entire access.
- Address drive time follows CS activation (no setup time guaranty).
- Delay time (OESETUP) and advanced time (OEHOLD) are disabled (OESETUP and OEHOLD bit fields don't care).
- Address and data multiplexed scheme is not supported in mode 1−2−3.
- Read data are latched on the TC_CK rising edge corresponding to the wait state delay completion (initial and in page wait state).
- One TC_CK cycle after access completion (CS high), the data bus is driven with the previous read value (see Figure 19 for direction activation and data copy timing).
- In asynchronous mode, REF_CLK is not provided outside the EMIFS and FLASH.CLK is kept low.
- Page mode always follows the non-full-handshaking protocol and the FLASH.RDY pin is never monitored whatever the full-handshaking bit field value in the dynamic wait state register is.

Figure 20. Asynchronous Page Mode 8x16-Bit Read With Page Crossing Operation on 16-Bit Width Device (RDWST=2, PGWST=0 FCLKDIV=1, RDMODE=1). Data write-back on the bus after read completion.

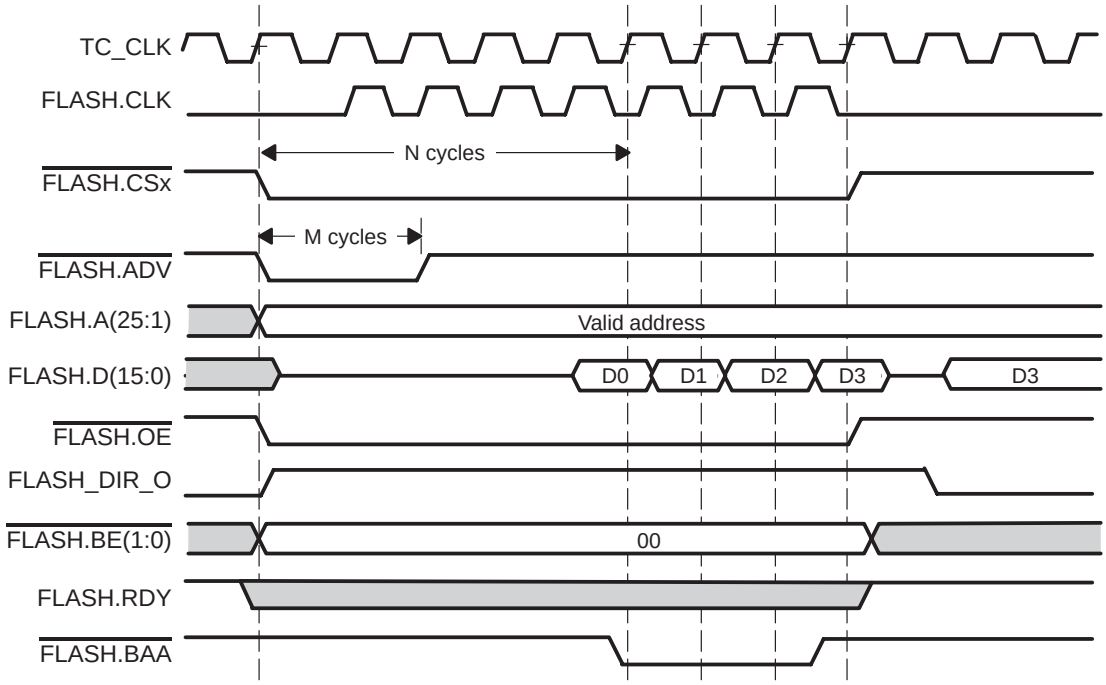


3.2.11 Mode 4 and Mode 5 Synchronous Burst Read Operation Mode

Synchronous Read in Non-Multiplexed Address and Data Memory

- The synchronous burst read mode 4 and 5 are selected by setting the RDMODE bit field in the corresponding EMIFS chip-select configuration register (see Table 19).
  - RDMODE = 4 or 5
- This mode only supports synchronous read accesses (single or consecutive). Flash devices usually require synchronous setup and enable mode to be done after power up. RDMODE must be changed to mode 4 or mode 5 only after flash device setup.

Figure 21. Mode 4 Synchronous Burst 4x16-Bit Read Operation on 16-Bit Width Device (RDWST=3, FCLKDIV =0, ADVHOLD=0, RDMODE=4). Data write-back on the bus after read completion.



- The REF_CLK is divided from TC_CK by a programmable value contained in FCLKDIV bit field of the CS configuration register (see Table 19).
- CS, ADV and address are driven one REF_CLK cycle before the first FLASH.CLK rising edge is provided externally. This ensures CS, ADV, and address valid setup time to device clock rising edge to be met.

- In case this one REF_CLK cycle advance is not enough to meet the setup time requirement, the ADV pulse width can be extended by ADVHOLD. The real access time start from CS & ADV & address setup time to device clock rising edge valid.
- The ADV pulse width depends on ADVHOLD bit field of the Advanced CS configuration register (see Table 28). ADV pulse width equals:
  - (ADVHOLD + 1) REF_CLK + 1 TC_CK (M cycles in Figure 21)
- Modes 4−5 are by default in full-handshaking mode. FLASH.RDY is monitored by the EMIFS to control read access time. FLASH.RDY must be asserted synchronously to FLASH.CLK.
- The first access is completed when both internal RDWST wait state expired and when FLASH.RDY is asserted by the external device.
- The internal initial wait state depends on RDWST bit field of the CS configuration register. RWDST value must include the extra “non active” output REF_CLK cycle used for CS & ADV & address setup time. Delay equals:
  - (RDWST + 2) REF_CLK (N cycles in Figure 21)
- Read data are latched on each TC_CK rising edge corresponding to a REF_CLK rising edge when FLASH.RDY has been sampled high on the previous REF_CLK rising edge.
- The following in-burst access wait state only depends on the FLASH.RDY state (RDWST expired).
- In mode 4, BAA control signal is asserted low on the first date sampling REF_CLK rising edge and is maintained low during the full burst access. BAA is kept high in mode 5 (no burst advance control is this mode).
- OE activation delay time from CS and address valid is programmable through OESETUP bit field in the advanced CS configuration register. Activation delay timing is equal to:
  - (OESETUP) REF_CLK
- Advanced time (OEHOLD) control is disabled (OEHOLD bit field doesn’t care).
- One TC_CK cycle after access completion (CS high) the data bus is driven with the previous read value (see Figure 21for direction activation and data copy timing).

Figure 22. Mode 5 Synchronous Burst 8x16-Bit Read Operation on 16-Bit Width Device (RDWST=3, FCLKDIV =0, ADVHOLD=0, RDMODE=5). Data write-back on the bus after read completion.

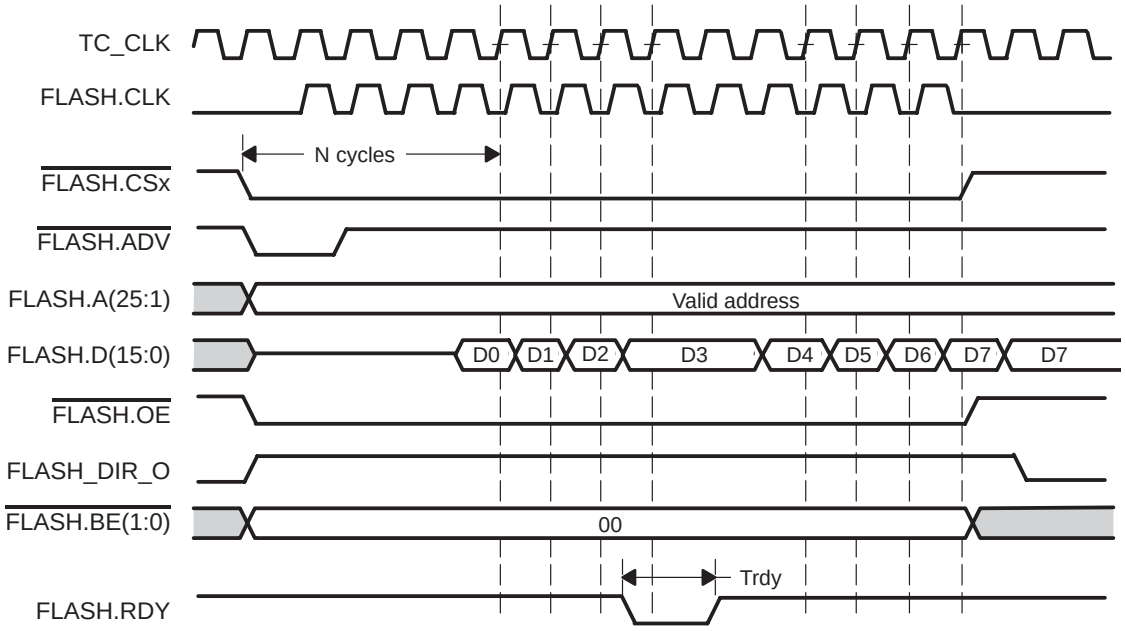
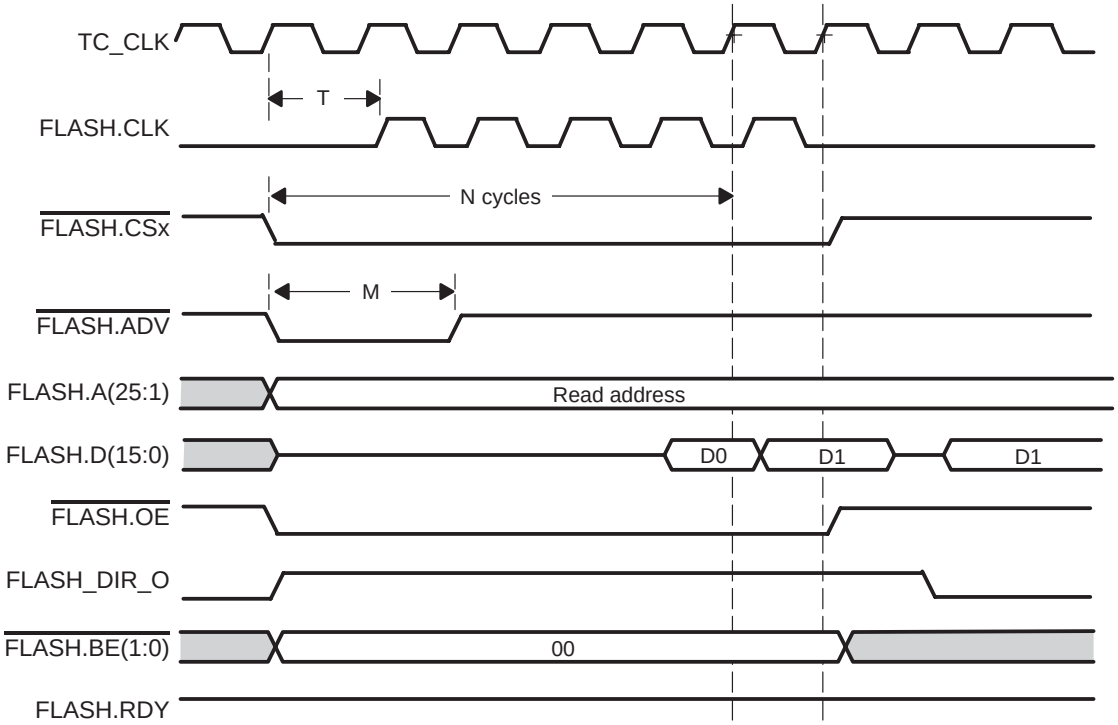


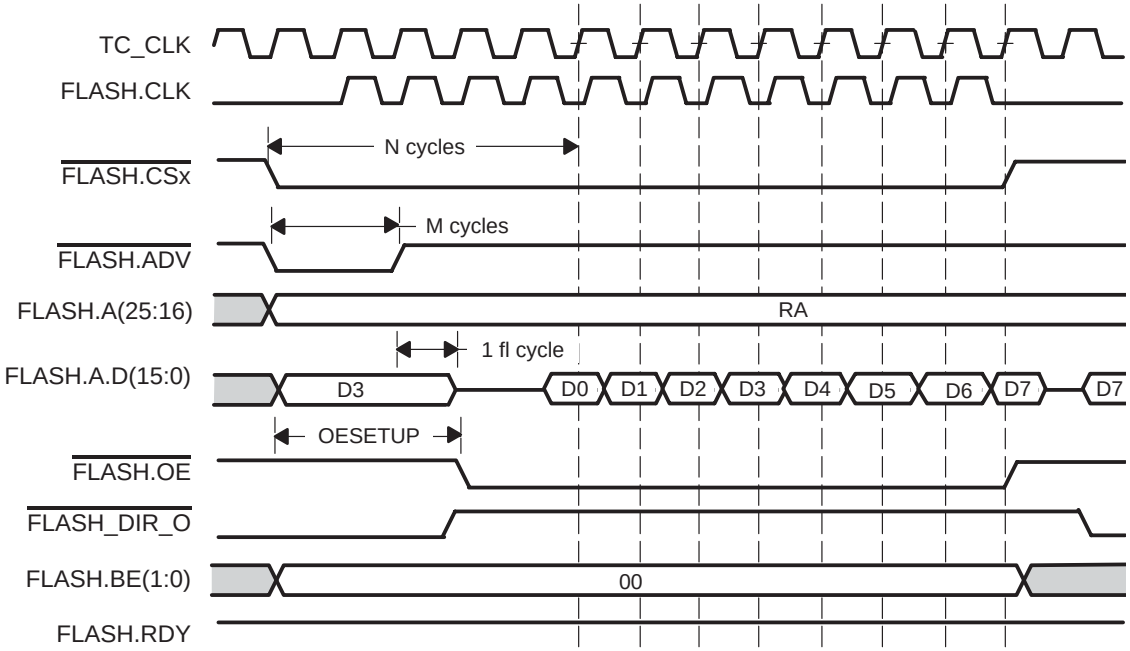
Figure 23. Mode 5 Synchronous Burst 2x16-Bit Read Operation on 16-Bit Width Device (RDWST=3, FCLKDIV =0, ADVHOLD=0, RDMODE=5). Data write-back on the bus after read completion



Synchronous Read in Multiplexed Address and Data Memory

- Multiplexed mode is enabled when the MAD bit field in CS configuration register is set to 1 (see Table 19).

Figure 24. Mode 5 Synchronous Burst 8x16-Bit Read Operation on Multiplexed Address/Data 16-Bit Width Device (RDWST=2, FCLKDIV =0, ADVHOLD=0, OESETUP = 3, RDMODE=5). Data write-back on the bus after read completion.



- CS, ADV, and address are driven one REF_CLK cycle before the first FLASH.CLK rising edge is provided externally. This ensures CS, ADV, and address valid setup time to device clock rising edge to be met.
- In case this one REF_CLK cycle advance is not enough to meet the setup time requirement, the ADV pulse width can be extended by ADVHOLD. The real access time start from CS & ADV & address setup time to device clock rising edge valid.
- Address hold time from ADV rising edge is guaranteed to be a minimum of one REF_CLK (delay for direction to change from out to in).
- FCLKDIV and OESETUP (REF_CLK) must be properly programmed to prevent bus contention and to ensure that address hold time device requirement is respected.
- Delay time OEHOLD is disabled.
- The ADV pulse width depends on ADVHOLD bit field of the Advanced CS configuration register (see Table 28). ADV pulse width equals to:
  - (ADVHOLD + 1) REF_CLK + 1 TC_CK (M cycles in Figure 24)

- Modes 4−5 are by default in full-handshaking mode. FLASH.RDY is monitored by the EMIFS to control read access time. FLASH.RDY must be asserted synchronously to REF_CLK.
- The first access is completed when both internal RDWST wait state expired and when ready pin is asserted by the external device.
- The internal initial wait state depends on RDWST bit field of the CS configuration register. RWDST value must include the extra nonactive output REF_CLK cycle used for CS and ADV and address setup time. Delay equals:
  - (RDWST + 2) REF_CLK (N cycles in Figure 24)
- Read data is latched on each TC_CK rising edge corresponding to a REF_CLK rising edge when FLASH.RDY has been sampled high on the previous REF_CLK rising edge.
- The following in-burst access wait state only depends on the FLASH.RDY pin state (RDWST expired).
- One TC_CK cycle after access completion (CS high), the data bus is driven with the previous read value (see Figure 24 direction activation and data copy timing).

Figure 25. Mode 5 Synchronous Burst 4x16-Bit Read Operation on Multiplexed Address/Data 16-Bit Width Device (RDWST=4, FCLKDIV =0, ADVHOLD=1, OESETUP = 4, RDMODE=5). Data write-back on the bus after read completion.

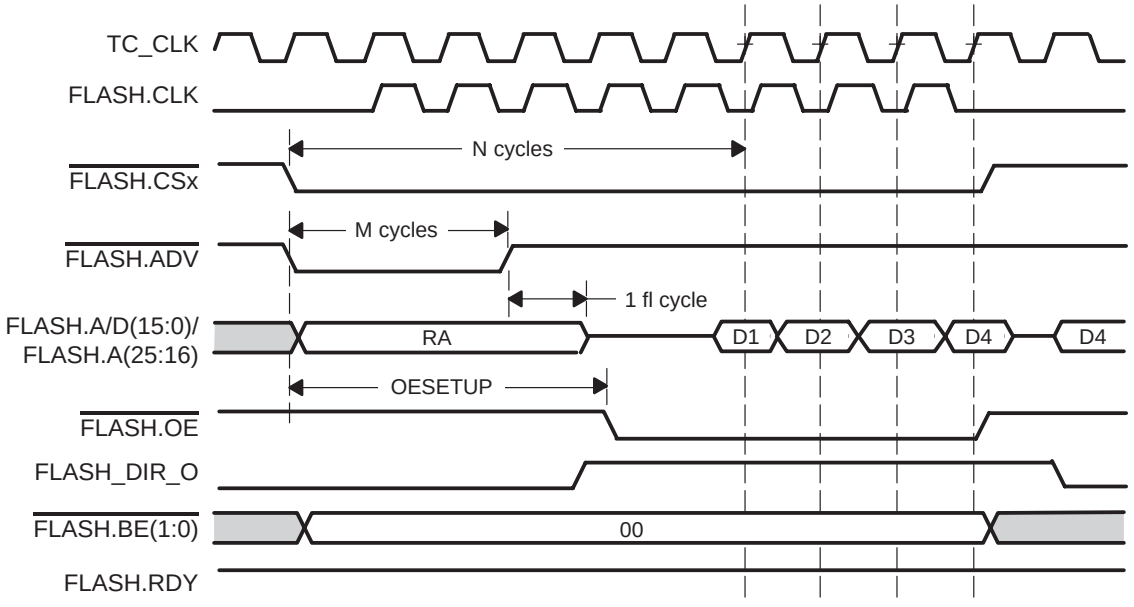
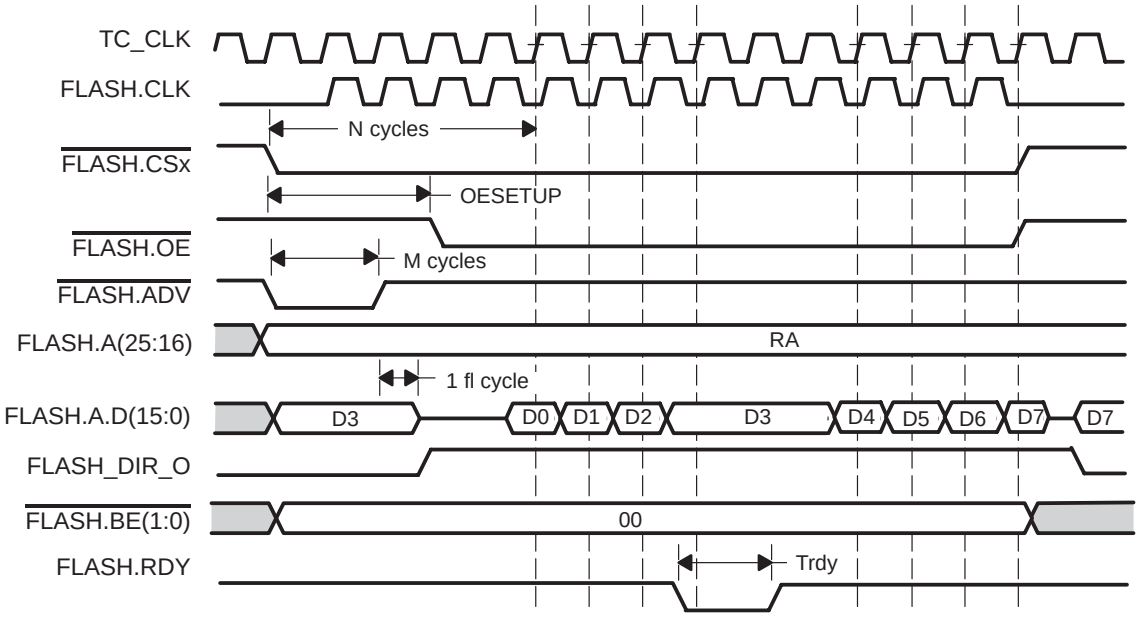


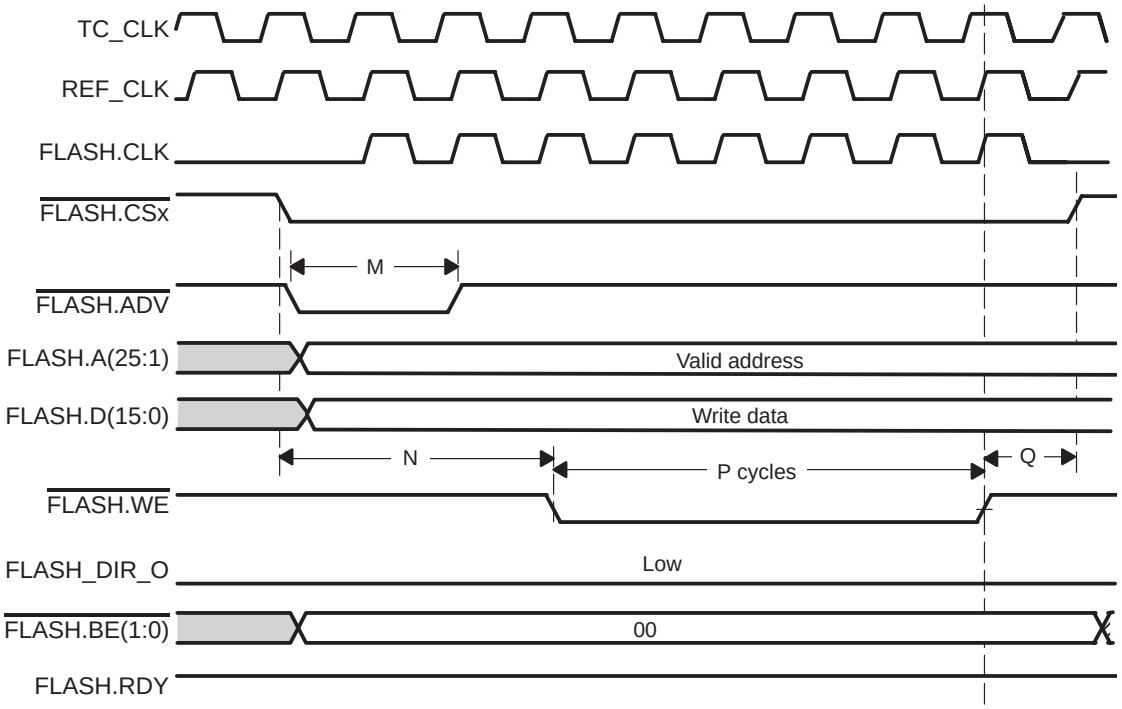
Figure 26. Mode 5 Synchronous Burst 8x16-Bit Read Operation on Multiplexed Address/Data 16-Bit Width Device (RDWST=3, FCLKDIV =0, ADVHOLD=0, OESETUP = 3, RDMODE=5). Data write-back on the bus after read completion.



Write Access in Mode 4 and 5

- Figure 27 shows FLASH.CLK activation details during a write access in mode 5 (non-multiplexed). Same behavior for the multiplexed address and data protocol.

Figure 27. Asynchronous 16-Bit Write Operation on a 16-Bit Width Device (RDMODE = 5, WRWST=2, WELEN=4 FCLKDIV=00 and ADVHOLD=1)



3.2.12 Read Retimed Protocol

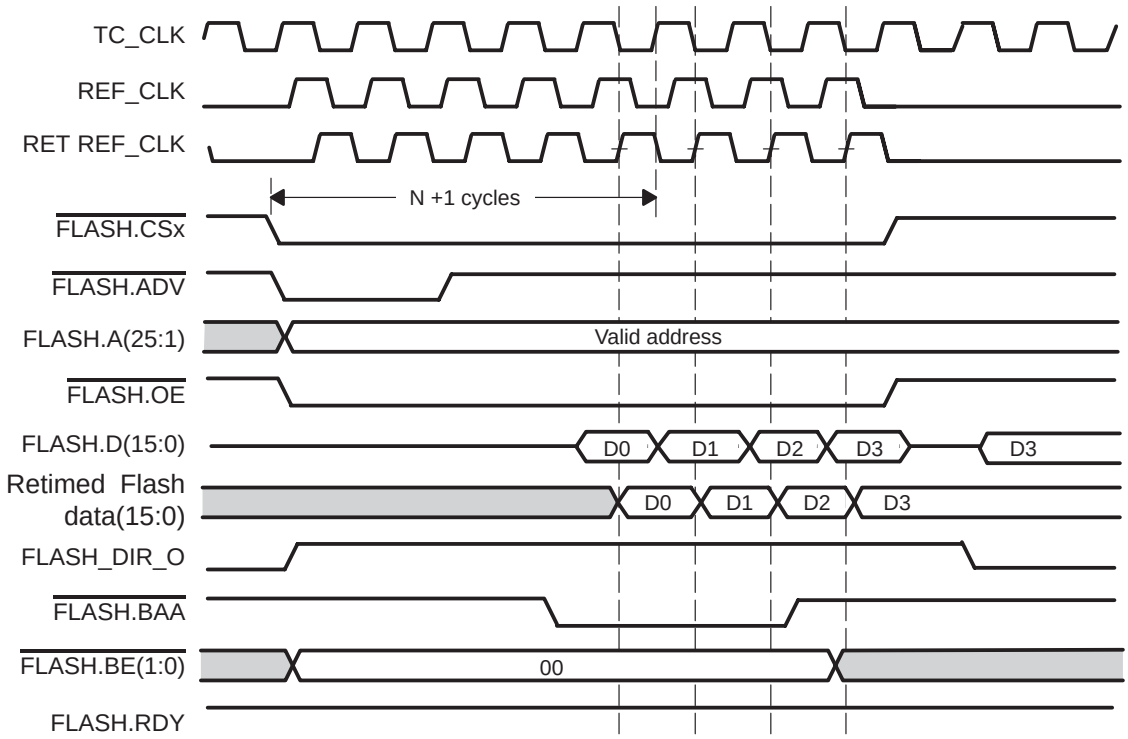
Warning

The RT bit in the EMIFS chip−select configuration register may be set only in RDMODE 4,5 and 7 only. The system hangs if the retiming bit is set in other modes (asynchronous modes because the retiming logic depends on the returned flash clock. In asynchronous mode, there is no flash clock and the system hangs.

- Due to IC I/O and board delays, the theoretical external memory maximum frequency may not be usable for REF_CLK value without retiming function. In synchronous mode 4−5, the retiming mode allows read data to be latched by a delayed Ret_REF_CLK obtained through the IC IO feedback of FLASH.CLK. This offers optimum data and sampling clock alignment.
- Retiming mode enables a pipelined read access. Compared to non-retimed access, the first access takes one extra REF_CLK cycle and the following accesses in a burst take one REF_CLK cycle each.

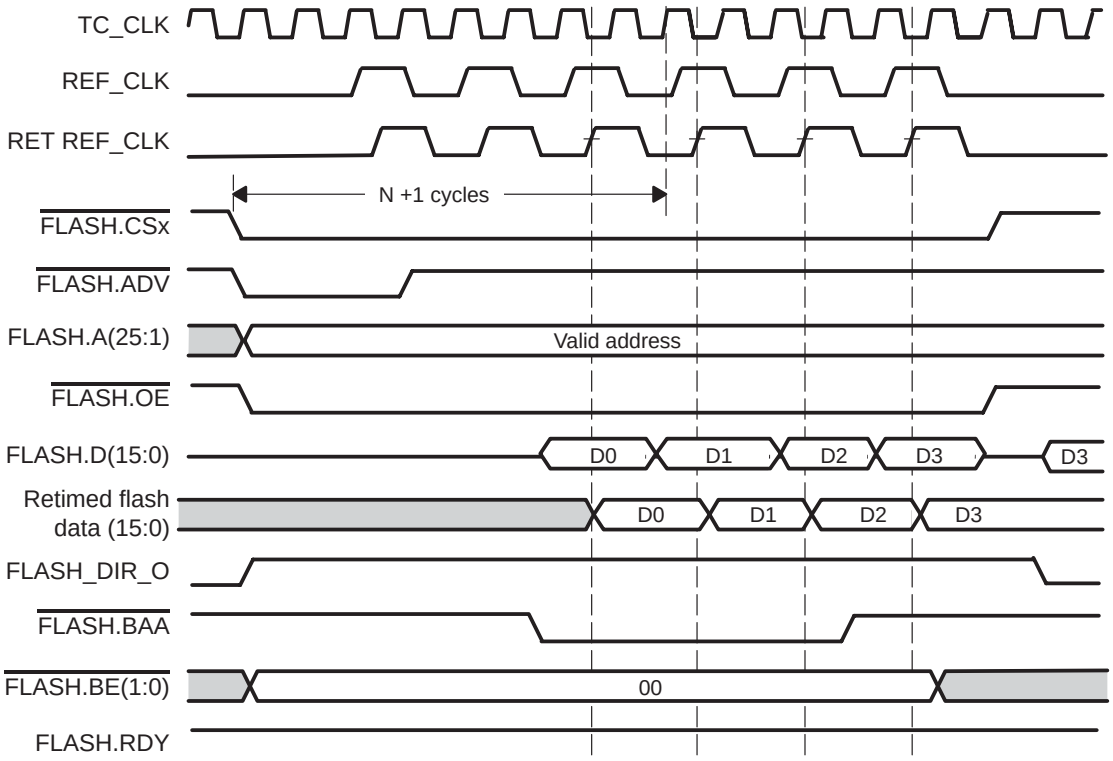
- The retiming mode is enabled through the RT bit field in the CS configuration register. Retiming mode is only allowed in synchronous modes 4−5−7 and has no effect on write accesses.

Figure 28. Mode 4 Synchronous Burst 4x16-Bit Read Operation on 16-Bit Width Device With Retiming on (RDWST=2, FCLKDIV =0, ADVHOLD=0, RDMODE=4). Data write-back on the bus after read completion.



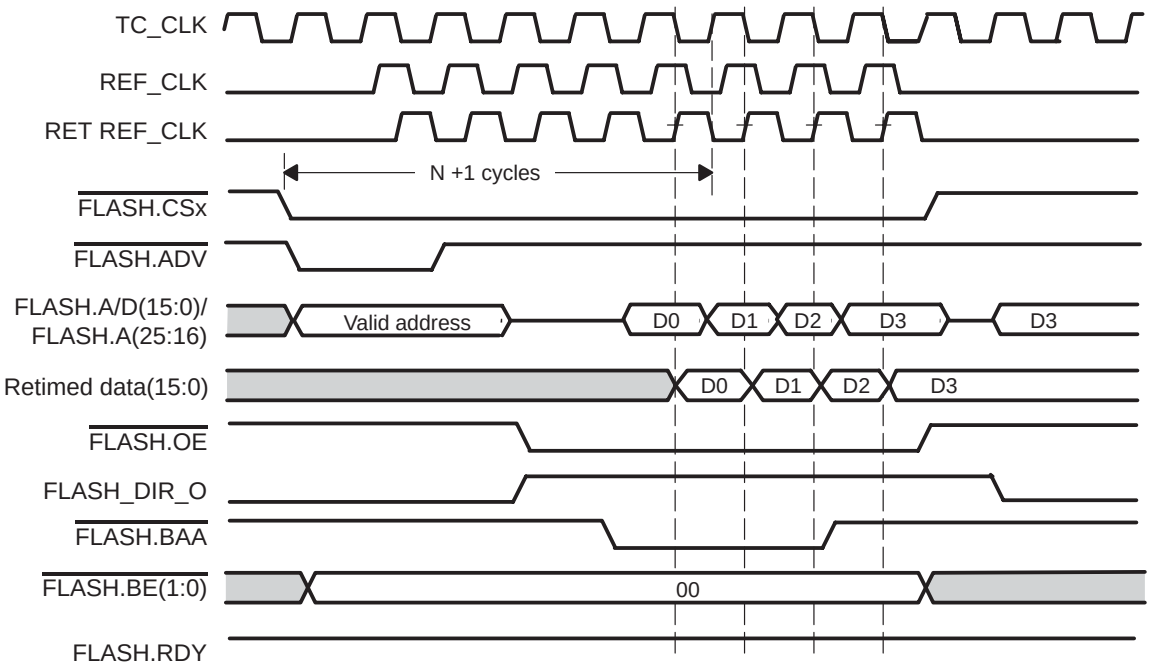
- As in non-retimed mode, CS, ADV, Address, BE, OE, and BAA are driven with respect to REF_CLK.
- Modes 4 and 5 programming model and protocol behavior remain the same as in non-retimed mode.
- In retiming mode, the RDWST is still referenced to REF_CLK. The retiming relaxed timing (extra delay for data valid) is included in the IC timing parameters.
- FLASH.RDY is also retimed with the Ret_REF_CLK. The retiming relaxed timing (extra delay for ready valid) is included in the IC timing parameters.

Figure 29. Mode 4 Synchronous Burst 4x16-Bit Read Operation on 16-Bit Width Device With Retiming on (RDWST=1, FCLKDIV =1, ADVHOLD=0, RDMODE=4)



- Retiming mode is also available in multiplexing address and data mode.

Figure 30. Mode 4 Synchronous Burst 4x16-Bit Read Operation on Multiplexed Address/Data 16-Bit Width Device With Retiming on (RDWST=3, FCLKDIV =0, ADVHOLD=0, OESETUP = 3, RMODE=4). Data write-back on the bus after read completion.

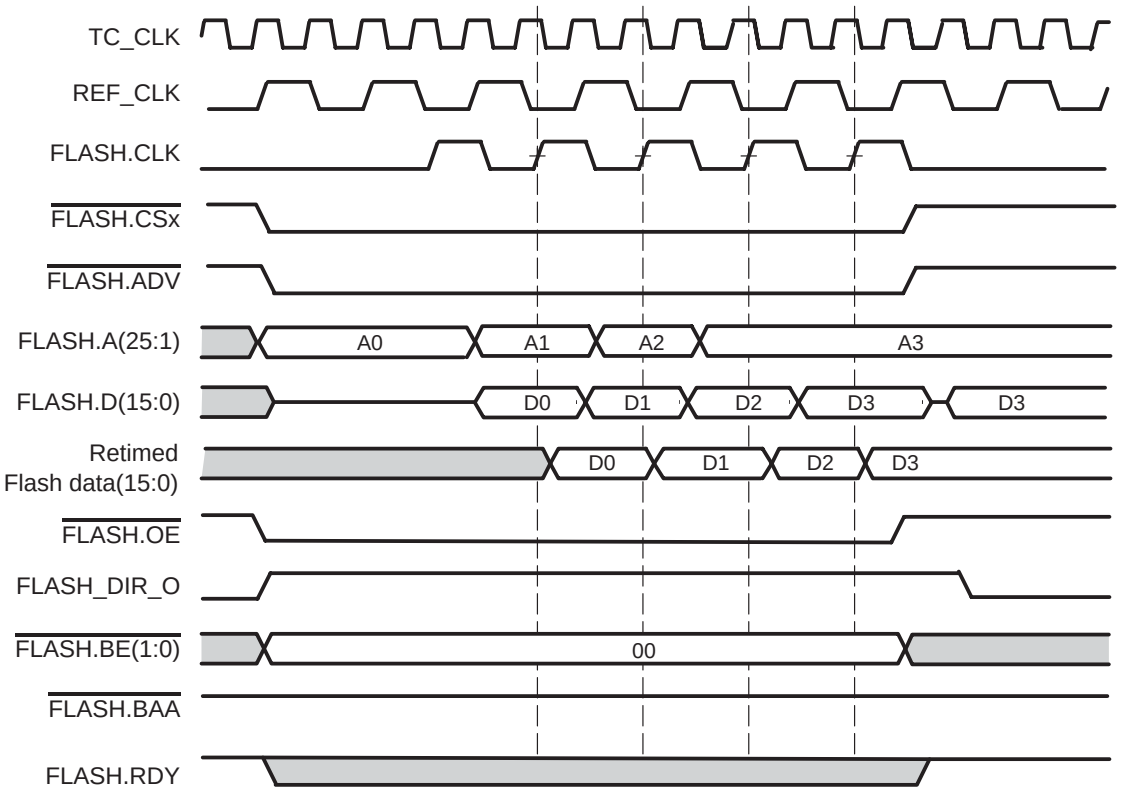


3.2.13 Mode 7—Synchronous Burst Read Operation Mode

- The synchronous burst read mode 7 is selected by setting the RDMODE bit field in the corresponding EMIFS chip-select configuration register
  - RDMODE = 7
- This mode only supports synchronous read accesses (single or consecutive). This protocol is used when accessing boot ROM and secure RAM.

The following diagrams show 16−bit external device widths. Similar accesses occur when using the 32−bit width of the secure RAM and boot ROM.

Figure 31. Mode 7 Synchronous Burst 4x16-Bit Read Operation on 16-Bit Width Device (RDMODE = 7, FCLKDIV =1). Data write-back on the bus after read completion.



- The REF_CLK is divided from TC_CK by a programmable value contained in FCLKDIV bit field of the CS configuration register. REF_CLK is inverted and provided externally as FLASH.CLK.
- The retimed mode must be enabled in mode 7. After reset, the RT bit is set for CS0 and CS3 if mode 7 is selected during the reset boot configuration process (see Table 18).
- Mode 7 enables a pipelined read access. Within the burst the addresses are provided one half FLASH.CLK cycle before the FLASH.CLK rising edge. Use the FLASH.CLK rising edge to latch addresses into the embedded ROM/RAM. The ROM/RAM must provide data with enough setup so data can be latched by the EMIFS on the next FLASH.CLK rising edge.
- REF_CLK frequency must be set so the memory access time can be met in less than one cycle. There is no internal or external wait state control

during read access in mode 7. RDWST field is not active and FLASH.RDY is not monitored in this mode (non-full-handshaking mode only).

- CS, ADV, OE are driven low for the entire access. ADVHOLD, OESETUP, OEHOLD time control are disabled in this mode.
- Address and data multiplexed protocol is not supported in mode 7 (MAD bit field not considered).
- One TC_CK cycle after access completion (CS high) the data bus is driven with the previous read value.

Write Access in Mode 7

- Figure 32 shows FLASH.CLK activation details during a write access in mode 7. Below example shows two successive write accesses with minimum access time.

Figure 32. Mode 7 Asynchronous Two Successive 16-Bit Write Operations on a 16-Bit Width Device (WRWST=0, WELEN=0 FCLKDIV=1 and ADVHOLD=0)

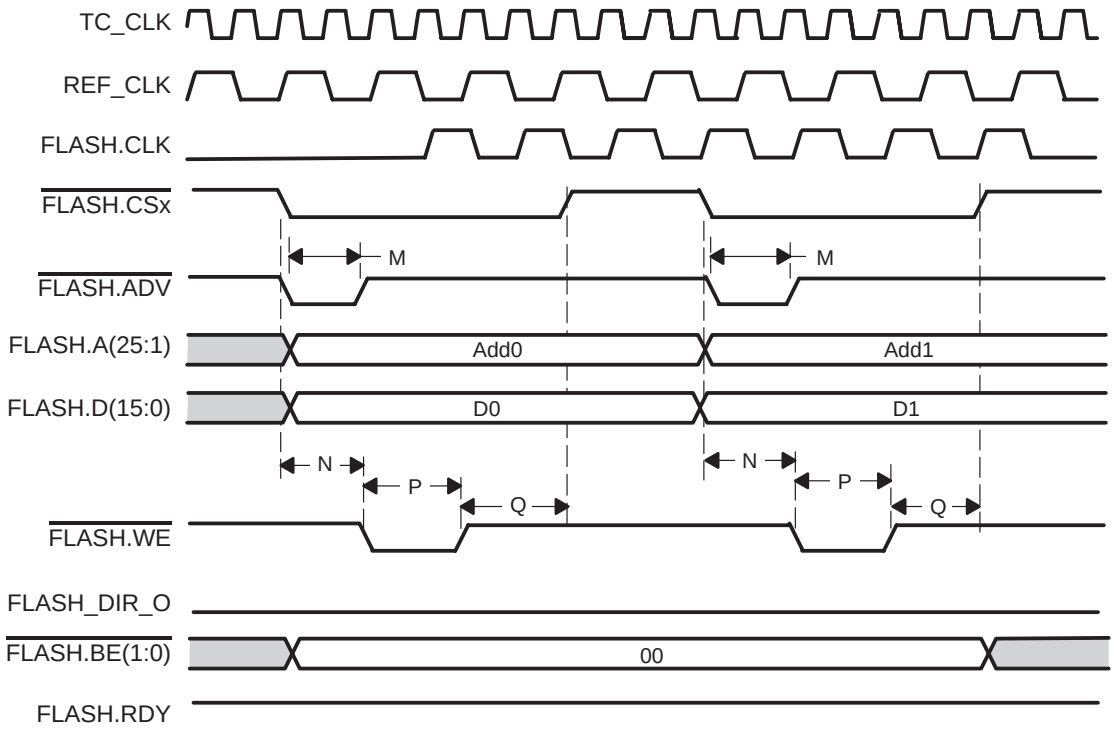


Figure 33. Mode 7 Asynchronous 16-Bit Burst Write Operations on a 16-Bit Width Device (WRWST=0, WELEN=0 FCLKDIV=1 and ADVHOLD=0, BTMODE = 0)

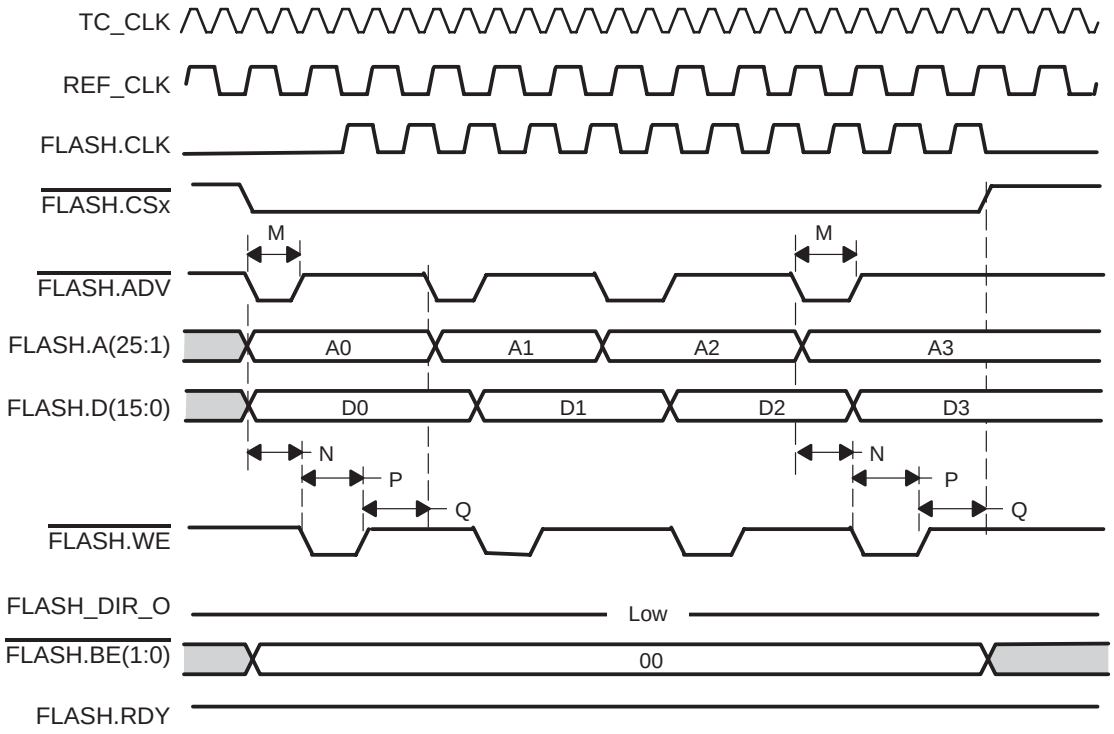
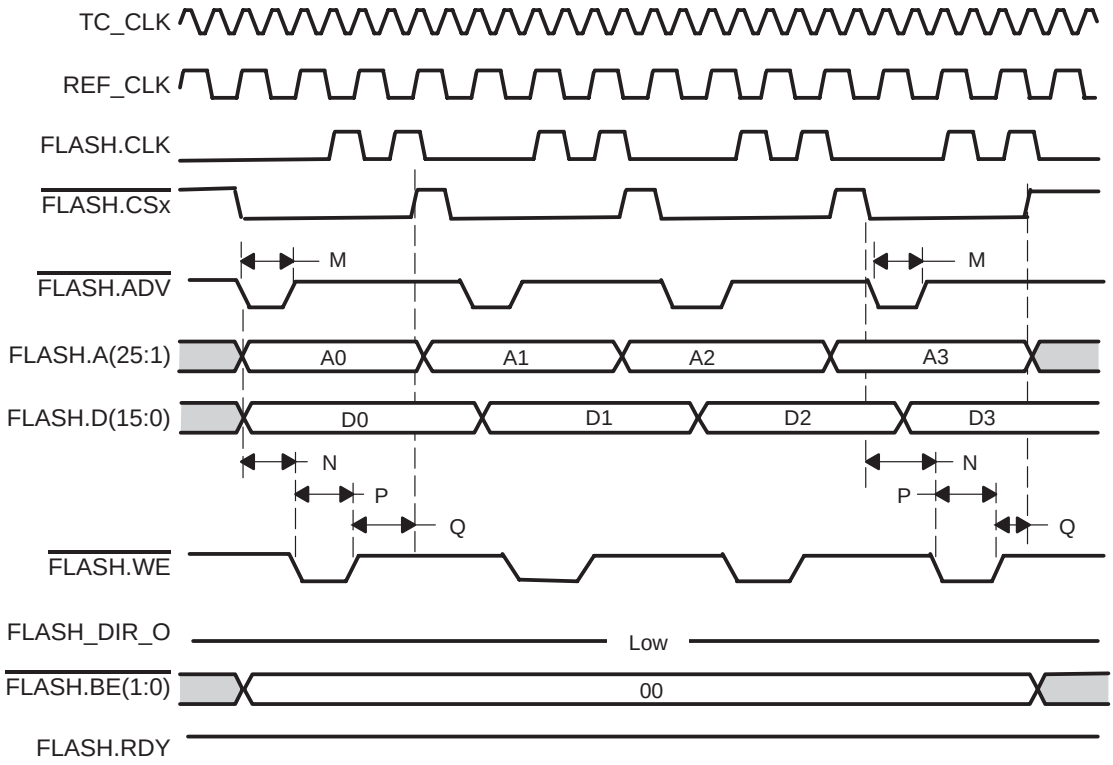


Figure 34. Mode 7 Asynchronous 16-Bit Burst Write Operations on a 16-Bit Width Device (WRWST=0, WELEN=0 FCLKDIV=1 and ADVHOLD=0, BTMODE = 1 and BTWST = 0)



3.2.14 Bus Turn-Around and CS Negation Time Control

- When slow devices are attached to the IC, it can be necessary to control the next data bus activation time after a read access to this slow device. This data bus activation time can be either an EMIFS write access or an EMIFS read access to a device attached to a different CS. The minimum idle time before next CS activation is two TC_CK cycles for independent access (no split or burst accesses). It can be extended to more than two TC_CK cycles by setting proper value in BTWST bit field of the CS configuration register attached to the slow device.
  - CS pulse width high = (BTWST +1) TC_CK.
  - BTWST must be at least 2 to have more than two-TC_CK-cycle idle time
- In case of successive read accesses due to EMIFS access size adaptation, or in case of burst read access in mode 0, the CS negation time between the successive read accesses is at least one TC_CK cycle. It can also be extended by the BTWST field.

- After a read completion, if no other access (RD, WR) is pending, the data bus is driven with the previous read value. The bus turn-around time (OE going high to direction going out) is a minimum of 1 TC_CK cycle and can be extended through BTWST.
- Table 1 shows the bus turn around cycles inserted for various transitions with EMIFS when BTMODE=0.

Table 1. Idle Time Between Different Bus Access Transitions (BTMODE = 0)

Access(n)	Access(n+1)	Chip-Select	Idle Time	Length(BTWST)
RD(csx)	RD(csx)	Same	Inserted	CSX
RD(csx)	WR(csx)	Same	Inserted	CSX
WR(csx)	RD(csx)	Same	Not inserted	–
WR(csx)	WR(csx)	Same	Not inserted	–
RD(csx)	RD(csy) x != y	Different	Inserted	CSX
RD(csx)	WR(csy) x != y	Different	Inserted	CSX
WR(csx)	RD(csy)x != y	Different	Not inserted	–
WR(csx)	WR(csy)x !=y	Different	Not inserted	–

Figure 35. Wait States During a Read to Read Operation (BTWST (CSX) = 2 and BTWST (CSY) = 1, BTMODE=0)

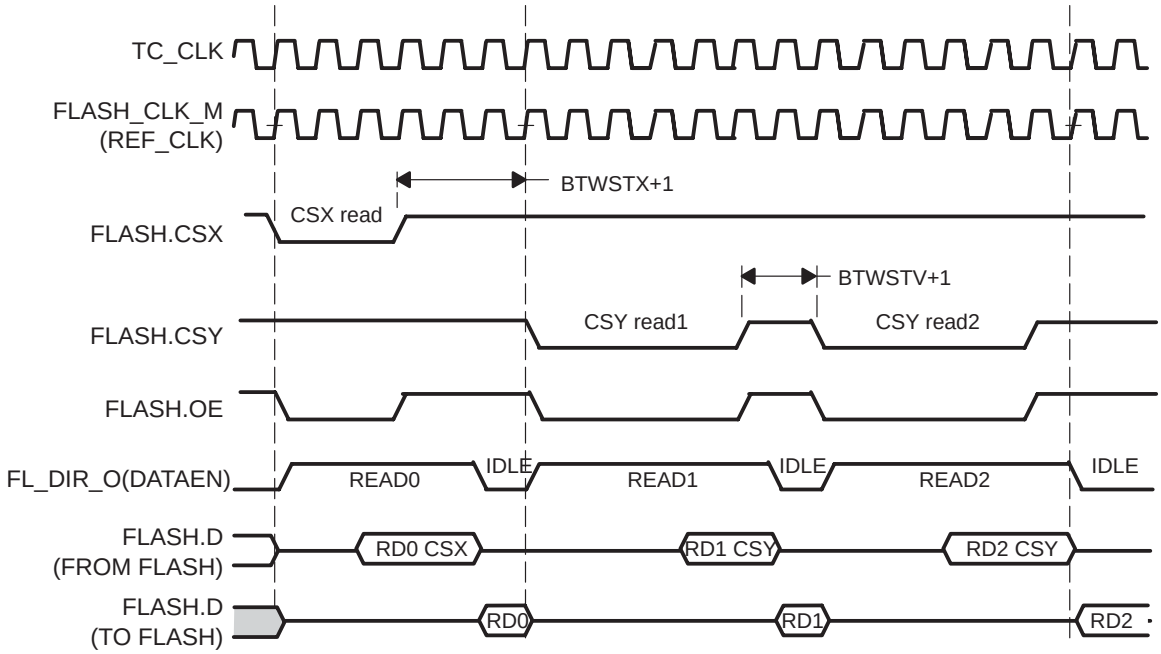
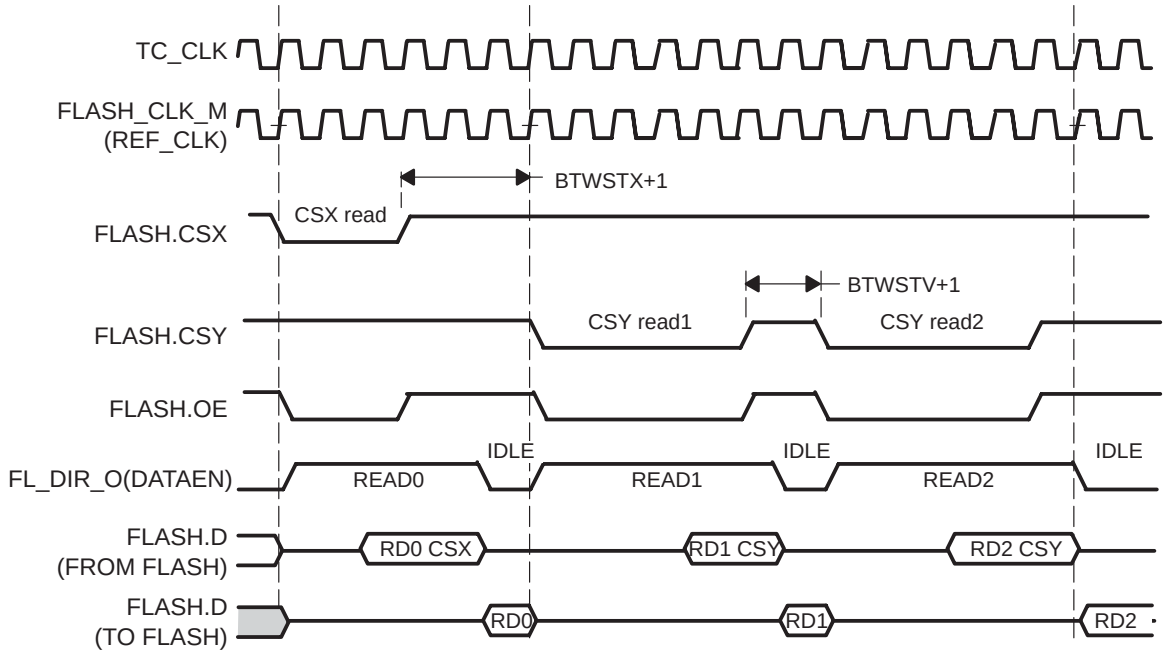


Figure 36. Wait States During a Read to Write Transition (BTWST(CSX)= 3 BTWST (CSY) = 2, BTMODE=0)

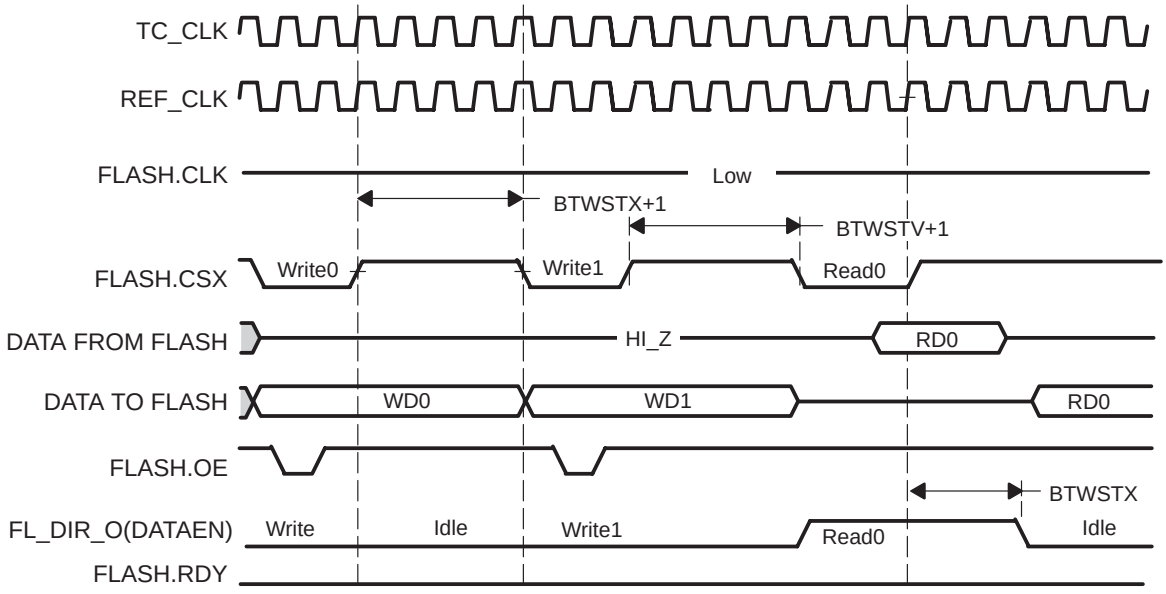


- The BTMODE field in the advance CS configuration register (see Table 28) extends the previous mode. When BTMODE=1, BTWST bit field controls CS negation time between any type of successive accesses to the same CS.
- In case of successive write accesses due to EMIFS access size adaptation or in case of write burst read access, there is no CS negation time between successive write accesses unless BTMODE is set. When BTMODE is set, the CS negation time between the successive write accesses is at least one TC_CK cycle and it can be extended by the BTWST field in the CS configuration (see Table 19) register (BTMODE set or clear)
  - CS pulse width high= (BTWST +1) TC_CK
- Table 2 shows the idle cycles inserted for various transitions with EMIFS when BTMODE=1.

Table 2. Idle Time Between Different Bus Access Transitions (BTMODE = 1)

Access(n)	Access(n+1)	Chip-Select	Idle Time	Length(BTWST)
RD(csx)	RD(csx)	Same	Inserted	CSX
RD(csx)	WR(csx)	Same	Inserted	CSX
WR(csx)	RD(csx)	Same	Inserted	CSX
WR(csx)	WR(csx)	Same	Inserted	CSX
RD(csx)	RD(csy) x != y	Different	Inserted	CSX
RD(csx)	WR(csy) x != y	Different	Inserted	CSX
WR(csx)	RD(csy)x != y	Different	Not inserted	–
WR(csx)	WR(csy)x !=y	Different	Not inserted	–

Figure 37. Wait States During a Write-to-Write and Write-to-Read Transition to Same Chip-Select (BTWST CSX = 3 BTMODE = 1)



3.2.15 External Device Reset Control

- EMIFS interface includes the $\overline{\text{FLASH.RP}}$ output signal. The $\overline{\text{FLASH.RP}}$ output pin is activated during OMAP warm and cold reset. $\overline{\text{FLASH.RP}}$ is also activated when the TC enters the idle state if the RESPWR_EN bit field of the clock and reset ARM_EWUPCT configuration register is set (See Table 66).

- The EXTPWR bit field in the ARM_EWUPCT register is used to specify the minimum time between FLASH.RP deasserted and TC going out of idle state (See section 4.3.4).

3.2.16 Dynamic Auto Idle and System Idle Synchronization

- EMIFS supports auto idle mode (clock gating) to dynamically reduce power consumption when no requests are pending and no accesses are on-going. The dynamic auto idle mode is enabled by setting to 1 the PWD_EN bit field of EMIFS configuration register.
- Upon clock and reset idle request, the EMIFS can send the idle request acknowledge when all on going transactions are completed. This allows the clock and reset module to cut the EMIFS source clock properly. The idle request acknowledge process is enabled by setting to 1 the PDE bit field of EMIFS configuration register (see Table 18).

3.2.17 Abort Management

The EMIFS can issue an interrupt in two cases of abort scenario:

- Restricted access mode violation on CS0
  - Access to CS0 address space is limitted to the MPU
  - When an access to CS0 space is initiated by the DMA, the DSP, or the OCP-I, the EMIFS completed the access giving back a 0 value as read value or without writing the given value to the final destination.
  - The EMIFS raises a TC abort interrupt (IRQ_ABORT) sets the abort flag bit, sets the restricted access error status bit, and updates the host ID bit field in the abort type register. The EMIFS updates the abort address register with the requested address causing the access violation.
  - The abort flag bit is cleared when the abort type register is read.
- Time-out issue during any access in case of full-handshaking mode
  - This feature can be enabled with the TIMEOUT_EN bit field and with an 8-bit TIMEOUT programmable value (expressed in REF_CLK cycles) in the EMIFS abort time-out register (see Table 12).
  - On beginning of an access, the time-out counter starts counting down. If the counter reaches 0 before the device response, the EMIFS raises an interrupt, sets the abort flag bit, sets the time out error status bit and updates the host ID bit field in the abort type register. The EMIFS updates the abort address register with the requested address causing the access time out.

- The abort flag bit is cleared when the abort type register is read.
- At reset, TIMEOUT_EN is set to 1 and time-out value is set to 255 REF_CLK cycles.

3.2.18 EMIFS Boot Mode

There are a number of external mechanisms that affect the initial state of the EMIFS during reset. The reset value of three bit fields in the EMIFS configuration (Table 18) and chip-select configuration (CS0 and CS3 only, Table 19) registers changes depending on the state of these external mechanisms. The three bits are BM (boot mode), BW (boot execution memory width, either 16 or 32 bits), and MAD (multiplexed address and data protocol). The reset value of these pins depends on the following:

BM reset value:

- BM is 0 when:
  - MPU_BOOT is 0

    OR
  - The device type is not emulation.
- BM is 1 when:
  - MPU_BOOT (ball J20) is 1

    AND
  - The device type is emulation.

BW reset value:

- BW is 0 (boot execution is from 16−bit wide memory, external boot):
  - BM is 1.
- BW is 1 (boot execution is from 32−bit wide memory, internal boot):
  - Device type is production type.

    OR
  - The device type is emulation type AND MPU_BOOT is 0.

MAD reset value:

- MAD is 0 (data and address non−multiplexed protocol):
  - Device type is production

    OR
  - Device type is emulator AND either MPU_BOOT OR GPIO_1 is 0.
- MAD is 1 (data and address multiplexed protocol):
  - Device type is emulation AND MPU_BOOT is 1 AND GPIO_1 is 1.

Note that the internal boot ROM may change the values of the MAD bit depending on the execution path.

When BW = 0, the following CS0 and CS3 configuration is selected.

- RDMODE=0
- FCLKDIV=3
- RDWST=15
- WELEN =15
- WRWST=15
- RT=0

When BW = 1, the following CS0 and CS3 configuration is selected.

- RDMODE=7
- FCLKDIV=0
- PGWST=0
- WRWST=0
- RDWST=0
- RT=1

The CS1 and CS2 reset configuration registers are independent of the boot input pins state.

Table 3. CS1 and CS2 Configuration Register Reset Value

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	PGWSTEN	PGWST				BTWST				MAD		BW		RDMODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	PGWST/WELEN				WRWST				RDWST					RT	FCLKDIV	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1

When BM is 0, CS0 is activated in the 0000:0000−03FF:FFFF range and CS3 is activated in the 0C00:0000−0FFF:FFFF range. When BM is 1, CS3 is activated in the 0000:0000−03FF:FFFF range and CS0 is activated in the 0C00:0000−0FFF:FFFF range.

3.3 EMIFF Programming

3.3.1 Main Features

The OMAP EMIFF is an SDRAM controller that manages all accesses by the various initiators of an OMAP-based system.

It can support one 16-bit device or two 8-bit devices. The external interface data bus width is always 16 bits.

The following devices are supported:

- Standard single-data-rate SDRAM
- Low-power single-data-rate SDRAM
- Mobile double-data-rate SDRAM

In terms of capacity and organization for the memory components that can be attached, the EMIFF can handle:

- 1G-bit, 512M-bit, 256M-bit, 128M-bit, 64M-bit, and 16M-bit devices
- 2-bank16M-bit devices, 2-bank or 4-bank 64M-bit devices, and 4-bank only for any other capacity
- x8 (two devices) or x16 (single device) data bus configuration, except for the 1G-bit device. The EMIFF only supports x16 1G-bit device (a single device). The maximum external SDRAM configuration is 128M bytes.

The SDRAM_type field of the EMIFF interface SDRAM configuration register must be used to specify the physical configuration of the devices.

The SDRAM type selection is the first action required from the software driver, using the SDRAM_Type field of the EMIFF SDRAM operation register. Types supported are regular SDR SDRAM, low-power SDRAM, and mobile DDR SDRAM.

The SDRAM controller supports:

- The self-refresh mode (idle), auto refresh, and other operating modes (HPHB, LPLB, and POM0 modes).
- MRS command and extended MRS command for DDR SDRAM and low-power SDRAM, sent via the SDRAM request manager.
- For SDR SDRAM, all burst sizes, between 1 and 32 consecutive accesses
- For DDR SDRAM, only bursts of 8.
- Two pipelined levels of request from the SDRAM request manager to enable page interleave timing and reduce overhead cycles by the burst interruption mechanism.

3.3.2 Initialization Sequence

Before it can be accessed for writing or reading after power on, or when exiting deep power-down mode, an SDRAM device must be initialized with various protocol parameters (burst size, CAS idle time, write burst, etc.). This is known in SDRAM literature as the *initialization sequence*.

Because SDR SDRAM and DDR SDRAM require different initialization sequences, the sequence is manually software driven, using the EMIFF SDRAM manual command register (writing a command opcode into that register, generates that command on the SDRAM interface).

The SDRAM operations register should only be programmed after the “a precharge all command” has been issued.

For SDR SDRAM devices, the initialization of the SDRAM bank is accomplished via a preprogrammed series of writes to the command register and the MRS register. There are two possibilities to generate the initialization sequence:

- Writing to the EMIFF SDRAM MRS register (legacy) in the SDRAM controller sets up the device to interface with OMAP. The following SDRAM command sequence is automatically generated:

  1) Precharge all
  2) Load mode register
  3) Autorefresh

  The EMIFF SDRAM MRS register (legacy) is provided for compatibility with existing code and must not be used by new software drivers.

- The new recommended procedure is:

  1) Wait for 200 µs after power up and memory clock is running and stable. The software is responsible for this initial wait. During this wait, NOP commands are the default commands automatically sent to the memory.
  2) Apply a PRECHARGE ALL command. This is accomplished by writing to the command register.
  3) Apply two AUTOREFRESH commands. This is accomplished by two consecutive writes to the command register.
  4) Apply an MRS LOAD command. This is accomplished by writing to the EMIFF SDRAM MRS register (new) (uses the new address, not the legacy address).

This procedure is also suitable for the mobile DDR device, as this kind of device does not include delay-locked loop technology (DLL).

3.3.3 Memory Mode Registers

The following restrictions apply to the MRS register programming:

- Burst length must be programmed to full page for SDR devices.
- For mobile DDR devices, burst length must be programmed to 8 and the CAS idle time must be set to 3.

All SDRAM internal mode registers are mirrored in the EMIFF controller. Standard MRS is provided. EMRS0 is provided for the DDR EMRS register. EMRS1 is provided for low-power SDRAM new features (such as partial array self-refresh, or temperature compensated self-refresh). EMRS2 is provided for future use.

For each register write access, a 12-bit data value is loaded in the memory device to support future new option bits in the existing registers. Always write 0 in all reserved bits.

When reading to these registers, the data in the mirrored register is returned.

3.3.4 EMIFF SDRAM Configuration

The EMIFF SDRAM configuration register must then be programmed to define the other parameters of the interface:

- Power-down strategy. When the PWD bit is set to 1, the power-down state is automatically entered between memory accesses (see Table 31). When there is no active transaction on the interface, CKE goes low. Any new access request awakes the device before making the access.
- SDRAM autorefresh control.

To optimize SDRAM bandwidth usage for data transfer, it is preferable to generate a sequence of autorefresh requests rather than a single autorefresh request. A sequence of four or eight successive autorefreshes can be programmed.

The autorefresh burst request is generated when a 16-bit refresh timer (see SDRAM configuration register, see Table 31) reaches the following user defined values, according to selected SDRAM frequency (via SDRAM configuration register):

- Memory size and configuration

- Entering self-refresh (SLRF bit). The self-refresh mode is automatically exited when the EMIFF receives an access request from OMAP initiators.

You can set the refresh counter value corresponding to system frequency. The following formula can be used for refresh counter value (the counter value is the number of TC_CK between refreshes).

Counter value = (64 ms (refresh rate) / Number of row / t_f) − 50 cycles

50 cycles margin due to a possible transfer currently starting when the refresh request occurs.

Where t_f = 1/system frequency [ns] (system freq = TC_CK freq)

Example: 100-MHz system clock, 4096 rows and 8-burst refresh. t_f = 10 ns.

Counter value = ((64.10^6 ns/4096)/10) − 50

Counter value = 1512 cycles (0x5E8)

If burst autorefresh is selected (4 or 8 consecutive autorefresh commands), this refresh period value is automatically scaled by the hardware accordingly.

3.3.5 EMIFF Configuration Power−Down Considerations

Self−refresh commands will not be issued and the SDRAM clock enable (CKE) will not deactivate under the following conditions:

- EMIFF_CONFIG.pwd = 1
- EMIFF_CONFIG.clk = 0
- EMIFF_CONFIG2.sd_auto_clk = 1

In addition, when in DPD (deep power down) mode, the SLRF bit must not be set because the CKE cannot operate after DPD exit.

3.3.6 Command Table

Table 4 lists the commands.

Table 4. Command List

Command	$\overline{CS}$	$\overline{RAS}$	$\overline{CAS}$	$\overline{WE}$	CKE	ADDR
Command inhibit (NOP)	H	X	X	X	H	X
No operation (NOP)	L	X	X	X	H	X
Active (select bank and row activate)	L	L	H	H	H	Bank/Row
Read (select bank and column and start read burst)	L	H	L	H	H	Bank/Col
Write (select bank and column and start write burst)	L	H	L	L	H	Bank/Col
Burst stop (for SDR only)	L	H	H	L	H	X
Precharge (deactivate row in a bank or all banks) Autoprecharge is not supported	L	L	H	L	H	A10 Low – Bank selected by BA1 and BA0 (for 4 bank) and BA0 (for 2 bank) High – All banks to be precharged
Autorefresh	L	L	L	H	H	X
Selfrefresh Entry	L	L	L	H	H → L	X
Selfrefresh Exit	X	X	X	X	L → H	X
MRS, EMRS0, EMRS1 (EMRS1 for M-SDR and M-DDR only)	L	L	L	L	H	Op Code [BA1:BA0] = 01: EMRS0 [BA1:BA0] = 10: EMRS1
Power down entry	H	X	X	X	H → L	X
Power down exit	H	X	X	X	L → H	X
Deep power down entry	L	H	H	L	H → L	X

Table 4. Command List (Continued)

Command	$\overline{CS}$	$\overline{RAS}$	$\overline{CAS}$	$\overline{WE}$	CKE	ADDR
Deep power down exit	H	X	X	X	L → H	X
Deep power down	The difference between power down and deep power down mode is that in case of power down mode whenever there is an access request to EMIFF, the memory is brought out of power down by pulling CKE high, but in case of deep power down mode even when there is an access request the memory is not brought out of deep power down mode (CKE is still low). The I/O buffers of the memory are deactivated and although further commands (after having entered deep power down mode) are accepted by the memory, it does not send or receive any data. After exit from deep power down mode NOP must be maintained for 200 µs. This must be followed by PRECHARGE ALLPrecharge all command, followed by eight autorefreshes, followed by MRS and EMRS commands.					

X = Don't Care

3.3.7 SDRAM Interface ac Parameters

The 2-bit SDRAM frequency (SDF) field in the EMIFF configuration register allows for selection of a set of ac timings parameters for the interface, depending on the type and speed grade of the memory component used. These parameters are commonly used in the SDRAM literature. As the naming can change from one memory manufacturer to another, the next table provides the functional description of each individual parameter, as named in this specification document.

Table 5. ac Parameters Description

tRC	Active to active command period
tRAS	Active to precharge command period
tRP	Precharge command period
tRCD	Active to write/read delay
tRRD	Active bank a to active bank b command

Four ac configurations are provided for standard SDR memory components, and another set of four configurations is provided for the mobile DDR memory. The selection between these two sets is done automatically, based on the memory type programmed in the EMIFF operation register.

All parameters are individually specified in number of clock cycles, whereas in the memory data sheets, absolute values are used for timing parameters. For a given working frequency, any ac parameter of the selected configuration, once converted into an absolute timing value, must be greater than the equivalent parameter specified by the memory manufacturer.

3.3.8 DLL Control for DDR SDRAM Support

The EMIF fast includes digitally controlled delay technology, for interfacing high-speed double-data-rate memory components to meet the strict timing requirements.

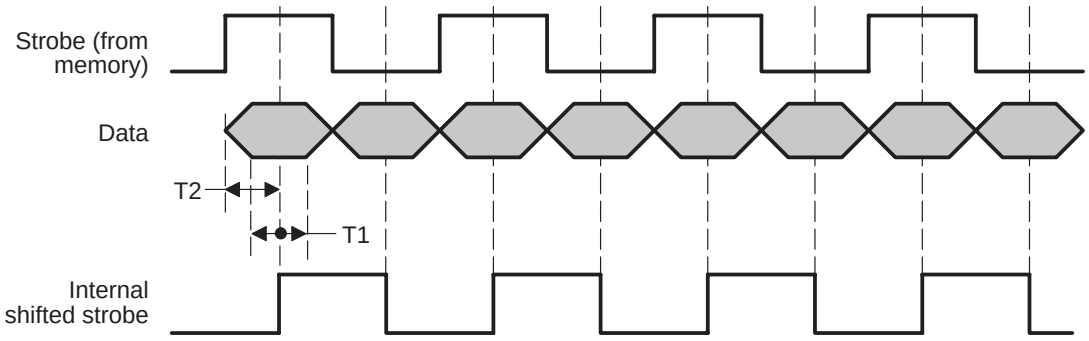
The DLL (delay locked loop) is a calibration module used to track voltage and temperature variations dynamically, as well as to compensate the silicon process dispersion (process voltage temperature (PVT) tracking). Two separate DLL elements are used in the EMIFF—one to control the read timing and one to control the write timing.

The DLLs controls several DCDL companion modules (digitally controlled delay line). by providing an 8-bit value, continuously updated so that it encodes a specific delay value (nominal 72 or 90 degrees) with respect to the memory interface clock frequency, in all PVT conditions. The amount of delay added by a DCDL element, controlled by this value under the same PVT conditions, matches the DLL 72 or 90-degree nominal delay.

EMIFF uses two DLLs: one for read operations, the other for write operations.

The read DLL controls two DCDLs. Each DCDL shifts the upper byte (respectively. lower byte) strobe coming from the DDR to ensure the data from the DDR can be properly sampled (see Figure 38).

Figure 38. EMIFF DDR Data Reads (with respect to DLL)

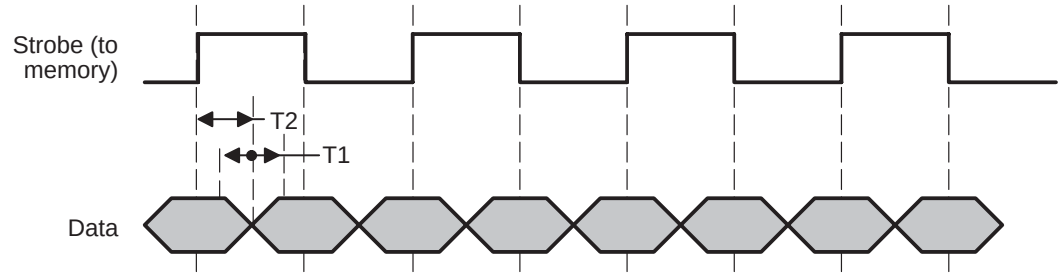


Notes: 1) T1: Fine tune delay control for shift of internal strobe (a manual adjustment). 64 steps of adjustment with a step value of 26.3ps (6 bit signed value in EMIFF_DLL_WRD_CTRL.WO). Note that 26.3 ps is a value valid in a nominal process at room temp and is subject to variation across process and temperature.

2) T2: The EMIFF_DLL_WRD_CTRL .DLLP control bit selects whether the internal strobe is 72 or 90 degrees (20% or 25% delay) from the external strobe

The write DLL controls one DCDL. This DCDL is used to shift the data lines from OMAP5912 to the DDR (see Figure 39).

Figure 39. EMIFF DDR Data Writes (with respect to DLL)

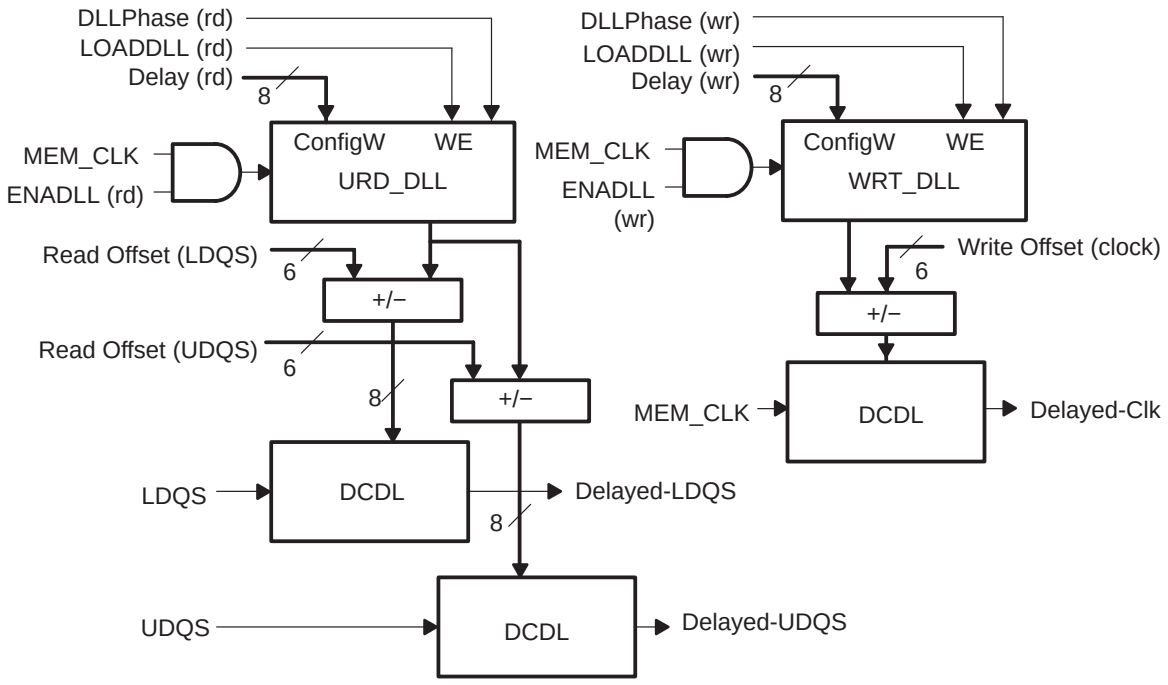


Notes: 1) T1: Fine tune delay control for shifting the data (a manual adjustment). 64 steps of adjustment with a step value of 26.3ps (6 bit signed value in EMIFF_DLL_WRD_CTRL.WO). Note that 26.3 ps is a value valid in a nominal process at room temp and is subject to variation across process and temperature.

2) T2: The DLLPhase control bit selects between whether or not the data is shifted 72 or 90 degrees (20% or 25% delay) from the strobe.

Figure 40 shows the controlled−delay block architecture. This block is configured by the EMIFF DLL control register (bit fields from this register are highlighted using bold characters).

Figure 40. Controlled Delay Subsystem Block Diagram



Thus, there are four important register controls:

- ENADLL in the EMIFF_DLL_WRD_CTRL register (See Table 37) enables or disables DLL.
- DLLPHASE in the EMIFF_DLL_WRD_CTRL register (See Table 37) allows a fine tuning of the delayed transaction to either 72 or 90 degrees out of phase with the internal strobe.
- Delay in the EMIFF_DLL_WRD_CTRL register (See Table 37) is the initial delay value of the strobe. The DLL then tries to lock in on the correct delay. A good initial value is 0x80.
- Write offset in the EMIFF_DLL_WRD_CTRL register (See Table 37) is a manually generated adjustment to the DLL delay. Generally, this field should not be used as the DLL will lock automatically with the correct delay.

The DLLPHASE control bit is used to setup the nominal delay tracked by the DLL. This delay can be either 72 degrees (20% of the clock period) or 90 degrees (25% of the clock period). The DLL locking range is 75/133 MHz if configured for 90 degrees, and 66/133 MHz if configured for 72 degrees.

The maximum working frequency for the DDR interface is device-dependant. It is limited by the speed of the traffic controller, as well as by the I/O buffer/receiver pair used for the interface (I/O voltage, LVCMOS technology versus SSTL2). The discussion that follows is based on a C035 device, with the traffic controller running at the maximum speed of 96 MHz.

For mobile DDR memories, there is no low-frequency limitation, as the DLL has been removed. The EMIF fast operational range for this type of memory is 0/100 MHz.

The DLL can be loaded with a programmable value. 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. In the latter case, the system is said to be working in unlock mode. The unlock mode is useful when interfacing mobile DDR at frequencies below the DLL minimum frequency, i.e. 66 MHz if 72 degrees, or 75 MHz if 90 degrees. Obviously, the accuracy of the delay when locked is much better, but the performance of the unlock mode is good enough below 66 MHz.

The unlock mode is simply setup by asserting the LOADDLL bit of the DLL control register continuously, with the delay field set to the expected value.

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 setup the controlled delay block for unlock mode:

- Write into the DLL control register, with ENADLL set, LOADDLL set, delay field set to the expected value.
- Write into the DLL control register, with ENADLL reset, to save power.

To setup the controlled delay block for lock mode (normal mode):

- Write into the DLL control register, with ENADLL set, LOADDLL set, Delay field set to 128 (half range to minimize locking time).
- Write into the DLL control register, with LOADDLL reset, to launch the tracking process.
- Wait for at least 336 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, assuming this period is short compared with the voltage and temperature variations. When disabled, the DLL counter is frozen to its current value. When enabled again, the tracking resumes from that DLL counter value.

The delay can be fine tuned using the programmable offsets. These offsets are 6-bit signed quantities, resulting in a +31/−32 adjustment range.

Step value: 26.3 ps +/− 10 ns. This value is valid in all PVT conditions (industrial temperature range −40 to 125 degrees, voltage range 1.35 V to 1.65 V, weak or strong process).

Table 6 provides the typical programming required for an OMAP3.2-based device, C035 process, with a 1.8 V−3 V I/O voltage range.

Table 6. EMIF Fast Controlled-Delay Block Programming

Memory Type	**DDR 1**	**Mobile DDR**	
Frequency range	83−100 MHz	0−66 MHz	66−100 MHz
DLLPhase (RD DLL/WR DLL)	90/72°	x	90/72°
Unlock mode	No	Yes	No
Forced delay value	x	104	X
Read offset	8	0	8
Write offset	−1	−18	−1

The DLL behavior can be monitored by the software using the EMIFF DLL status registers (URD status and WRD status registers). For each DLL, the current counter value is available (DLLCount field), as well as two status bits: overflow and underflow. Overflow is set when the DLL counter value is 255 during at least one TC clock cycle, and underflow is set when the DLL counter is 0 during at least one TC clock cycle.

These two status bits are both reset if the DLL is disabled (ENADLL is set to 0 in the DLL control register).

3.3.9 Page-Closing Strategy

The SDRAM operation register that selects between SDR and DDR also allows selection of a page-closing strategy. Three variants are available:

- High-power/high-bandwidth mode (HPHB)

  In this mode of operation, *the active page is always left open at the end of access*. The EMIFF controller keeps track of the open pages. It then determines if the current access is to an opened page (access proceeds without latency) or to a closed page (the currently active page within that internal bank is closed, the necessary page is activated, and the access proceeds after a necessary latency). This mode of operation is useful in situations where the power consumption is not critical, but the bandwidth achieved is.

- Low-power/low-bandwidth mode (LPLB)

  In this mode of operation, *the page being accessed is always closed at the end of the current access* (the controller adds a precharge command after any read or write to close the row). For every access, the page being accessed must be activated at the beginning of the access, which causes extra cycles of latency and lowers the memory bandwidth. Though this may not be an efficient or optimal operating mode, this scheme is very useful in situations where minimizing power consumption is of primary concern. The power consumption is minimized because only the currently accessed page is open and all pages are closed during idle cycles.

- Programmable operating mode (POM0)

  In this mode of operation, *a time-out register is associated with each open page* (see Table 42, SDRAM operation register). Because the maximum number of simultaneous open pages is four (one per bank), four time-out registers are provided. Upon an access to an open page, the time-out register is set to the user-programmed value and starts to count down. If the counter reaches zero, the controller then closes the inactive page. This mode of operation has better bandwidth performance than the low-power/low-bandwidth operating mode and does not consume power as heavily as the high-bandwidth/high-power mode.

  The effective time-out value (in SDRAM clock cycles) with respect to the 7-bit programmed value is programmed value x 4 + 3. The minimum value that can be programmed is 4; consequently, the minimum number of cycles before a row is automatically closed is 19.

As mentioned previously, the command register is used during the initialization sequence, but also controls the deep power mode (enter/exit commands) if supported by the memory device.

Arbitration in EMIFF is performed according to the priority algorithms described in Section 3.9, *Priority Algorithms.*

3.4 OCP-I Programming

The OMAP 3.2 OCP external initiator port is an interface intended to connect an external master device to the OMAP 3.2 platform. The OMAP 3.2 core appears as a slave, and its entire memory map, including TIPB peripherals, is accessible. The interconnect bus is compliant with the OCP specification.

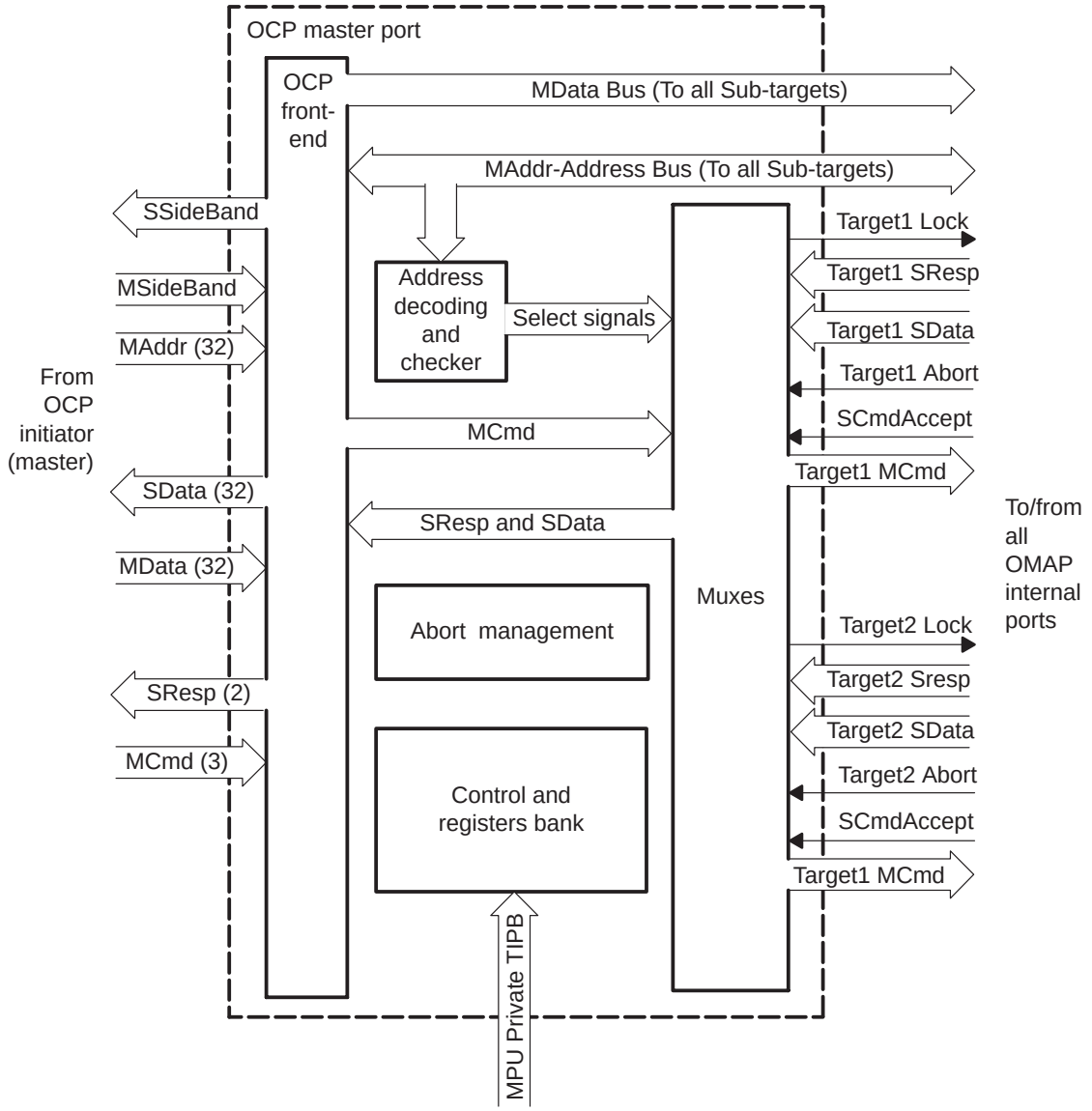
The OCP initiator is capable of single transfers in 8, 16, and 32 bits and burst mode transfers in 32 bits.

OCP-I provides access to the following OMAP targets:

- EMIFS (external slow memories)
- EMIFF (external fast memories)
- OCPT1
- OCPT2
- Multibank OCPT1/2
- MPUI (not part of the TC; see section 5, *MPUI*)
- MPU private TIPB bridge (not part of the TC; see section 7, *TIPB Bridge*)
- MPU public TIPB bridge (not part of the TC; see section 7, *TIPB Bridge*)

Figure 41 shows the OCP-I block diagram.

Figure 41. OCP-I Block Diagram



Though OCP-I has visibility to all OMAP resources, a protection register allows validation of the access permission on a per target basis. Any access to an unauthorized resource generates an abort.

The MPU controls the OCP-I port through its private TIPB; if permission is given to the external initiator to access the private TIPB, the external master also accesses the OCP-I configuration registers.

If OS protection is needed, it is recommended that any device integrating the OMAP 3.2 core provide an MMU or a translation table controlled by the OMAP MPU. Then it is the responsibility of the system software to ensure that undefined or unauthorized locations are not accessed.

An internal address checker continuously compares the OCP address bus and the OMAP 3.2 memory map. An abort signal is provided if the transaction is not correct.

OCPI contains other status and control registers listed in Table 7, *OCP-I Registers*.

3.4.1 Address and Command Fault Registers

When a non-OMAP-defined address is detected or a bus error occurs from the subtargets on a write access, an abort is sent to the initiator to indicate that the current access can not be completed. The access to the OMAP subtarget is terminated and the address and command buses are stored in the fault registers.

3.4.2 Abort Type Register

The type of the abort event is reported in a status register.

The interrupt handler can clear this register and the interrupt request by reading any value to this register.

3.4.3 Protection Register

This register provides access protection to the following targets:

- ❏ EMIF-slow and EMIF-fast
- ❏ OCPT1 and OCPT2
- ❏ Multibank OCPT1/2
- ❏ MPUI
- ❏ MPU private TIPB bridge and MPU public TIPB bridge

The reset value of the protection register bits is determined by the value of the static_reset_protect_mode pin.

3.5 Traffic Controller Registers

TC registers are distributed into the traffic controller submodules:

- ❏ OCP-T1/OCP-T2 registers
- ❏ EMIFS registers
- ❏ EMIFF registers
- ❏ OCPI registers

Table 7 lists the 32-bit OCP-T1 and OCP-T2 registers. Table 7 through Table 15 describe the register bits.

Table 7. OCP-T1/OCP-T2 Registers

Base Address = FFFE CC00			
Name	**Description**	**R/W**	**Offset**
OCPT1_PRIOR	OCP-T1 LRU priority register	R/W	0x00
OCPT1_PTOR1	OCP-T1 dynamic priority time-out 1	R/W	0xA0
OCPT1_PTOR2	OCP-T1 dynamic priority time-out 2	R/W	0xA4
OCPT1_PTOR3	OCP-T1 dynamic priority time-out 3	R/W	0xA8
OCPT1_ATOR	OCP-T1 abort time-out	R/W	0xAC
OCPT1_AADDR	OCP-T1 abort address	R	0xB0
OCPT1_ATYPER	OCP-T1 abort type	R	0xB4
OCPT_CONFIG_REG	OCP target configuration register	R/W	0xB8
OCPT2_PRIOR	OCP-T2 LRU priority register	R/W	0xD0
OCPT2_PTOR1	OCP-T2 dynamic priority time-out 1	R/W	0xD4
OCPT2_PTOR2	OCP-T2 dynamic priority time-out 2	R/W	0xD8
OCPT2_PTOR3	OCP-T2 dynamic priority time-out 3	R/W	0xDC
OCPT2_ATOR	OCP-T2 abort time-out	R/W	0xE0
OCPT2_AADDR	OCP-T2 abort address	R	0xE4
OCPT2_ATYPER	OCP-T2 abort type	R	0xE8

Table 8. OCP Priority Registers 1 and 2(OCPT1_PRIOR and OCPT2_PRIOR)

Base Address = 0xFFFE CC00, Offsets = 0x00 and 0xD0				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:16	RESERVED	Reserved. To ensure software compatibility, reserved bit should be written to 0 and read value should be considered undefined.	R/W	0x0000
15:12	OCP_PRIORITY	Number of consecutive accesses allowed for OCP-I	R/W	0000
11:8	DMA_PRIORITY	Number of consecutive accesses allowed for DMA	R/W	0000
7		Reserved		

Table 8. OCP Priority Registers 1 and 2(OCPT1_PRIOR and OCPT2_PRIOR) (Continued)

Base Address = 0xFFFE CC00, Offsets = 0x00 and 0xD0				
Bit	**Name**	**Function**	**R/W**	**Reset**
6:4	DSP_PRIORITY	Number of consecutive accesses allowed for DSP	R/W	000
3		Reserved		
2:0	ARM_PRIORITY	Number of consecutive accesses allowed for MPU	R/W	000

The OCP target priority registers (OCPTx_PRIOR) allow the target to give consecutive accesses to a host when the host is granted the OCP target.

The MPU and DSP can have from 1 to 8 consecutive accesses. The DMA and the OCP initiator can have 1 to 16 consecutive accesses, depending on the content of corresponding bits in their priority registers.

Table 9. OCP-T1 and OCP-T2 Priority Time-Out Registers 1 (OCPT1_PTOR1 and OCPT2_PTOR1)

Base Address = 0xFFFE CC00, Offsets = 0xA0 and 0xD4				
Bits	**Field**	**Description**	**R/W**	**Reset**
31:8	RESERVED	Reserved. To ensure software compatibility, reserved bit should be written to 0.	R/W	0x000000
7:0	DMA	Number of TC_CK cycles that DMA must wait in low-priority queue before going to high-priority queue	R/W	0x00

Table 10. OCP-T1 and OCP-T2 Priority Time-Out Registers 2 (OCPT1_PTOR2 and OCPT2_PTOR2)

Base Address = 0xFFFE CC00, Offsets = 0xA4 and 0xD8				
Bits	**Field**	**Description**	**R/W**	**Reset**
31:24	RESERVED	Reserved. To ensure software compatibility, reserved bit should be written to 0.	R/W	0x00
23:16	DSP	Number of TC_CK cycles that DSP must wait in low-priority queue before going to high-priority queue	R/W	0x00

Table 10. OCP-T1 and OCP-T2 Priority Time-Out Registers 2 (OCPT1_PTOR2 and OCPT2_PTOR2) (Continued)

Bits	Field	Description	R/W	Reset
15:8	RESERVED	Reserved. To ensure software compatibility, reserved bit should be written to 0 and read value should be considered undefined.	R/W	0x00
7:0	LCD	Number of TC_CK cycles that LCD must wait in low-priority queue before going to high-priority queue	R/W	0x00

Table 11. OCP-T1 and OCP-T2 Priority Time-Out Registers 3 (OCPT1_PTOR3 and OCPT2_PTOR3)

Base Address = 0xFFFE CC00, Offsets = 0xA8 and 0xDC				
Bits	**Field**	**Description**	**R/W**	**Reset**
31:8	RESERVED	Reserved. To ensure software compatibility, reserved bit should be written to 0 and read value should be considered undefined.	R/W	0x000000
7:0	OCP-I	Number of TC_CK cycles that OCP-I must wait in low-priority queue before going to high-priority queue.	R/W	0x00

The priority time-out registers (OCPTx_PTORy) are used to store the number of TC_CK clock cycles before DSP, DMA, LCD, or OCPI requests are made high priority in the dynamic priority scheme for the OCP target.

Table 12. OCP-T1 and OCP-T2 Abort Time-Out Registers (OCPT1_ATOR and OCPT2_ATOR)

Base Address = 0xFFFE CC00, Offsets = 0xAC and 0xE0				
Bits	**Field**	**Description**	**R/W**	**Reset**
31:9	RESERVED	Reserved. To ensure software compatibility, reserved bit should be written to 0 and read value should be considered undefined.	R/W	0x000000
8	TIMEOUT_EN	Enable time-out bit. 0: Disable 1: Enable	R/W	1
7:0	TIMEOUT	Number of counted-down clock cycles before sending out abort signal if there is no response from the slave.	R/W	0xFF

The abort time-out register (ATOR) is used to store the number of clock cycles the OCP target counts down before activating the abort signal because the peripheral did not return a SRESP signal.

Table 13. OCP-T1 and OCP-T2 Abort Address Registers—Access Address (OCPT1_AADDR and OCPT2_AADDR)

Base Address = 0xFFFE CC00, Offsets = 0xB0 and 0xE4				
Bits	**Field**	**Description**	**R/W**	**Reset**
31:0	Address	Address of access that caused abort.	R	0x0000 0000

Table 14. OCP-T1 and OCP-T2 Abort Type Registers—Access Address (OCPT1_ATYPER and OCPT2_ATYPER)

Base Address = 0xFFFE CC00, Offsets = 0xB4 and 0xE8				
Bits	**Field**	**Description**	**R/W**	**Reset**
31:5	RESERVED	Reserved. To ensure software compatibility, reserved bit should be written to 0 and read value should be considered undefined.	R/W	0
4	TIMEOUT_ERR	Abort generated by time-out register. 0: No abort 1: Abort	R	0
3	BUS_ERR	Abort generated by error coming from external peripherals 0: No abort 1: Abort	R	0

Table 14. OCP-T1 and OCP-T2 Abort Type Registers—Access Address (OCPT1_ATYPER and OCPT2_ATYPER) (Continued)

Base Address = 0xFFFE CC00, Offsets = 0xB4 and 0xE8				
Bits	**Field**	**Description**	**R/W**	**Reset**
2:1	HOST_ID	Host ID of request that caused memory fault 00: MPU 01: DSP 10: DMA 11: OCPI	R	00
0	ABORT_FLAG	0: No abort 1: Abort	R	0

Table 15. OCP Target Configuration Register (OCPT_CONFIG_REG)

Base Address = 0xFFFE CC00, Offset = 0xB8				
Bits	**Field**	**Description**	**R/W**	**Reset**
31:2	Reserved	Reserved, must be all 0s	R/W	0
1	PIPELN_RD_EN	0: Pipeline read operation disabled 1: Pipeline read operation enabled	R/W	1
0	AUTO_GATED_CLK	0: Autogating clock disabled 1: Autogating clock feature enabled to save power	R/W	0

This register contains the PIPELN_RD_EN bit to enable or disable the pipeline read operation in OCP Target. When enabled, the OCP target will start the next read command when the accept for the previous read command arrives.

3.6 EMIFS Registers

Table 16 lists the 32-bit EMIFS registers. Table 17 through Table 28 describe the register bits.

Table 16. EMIFS Registers

Base Address = FFFE CC00			
Name	**Description**	**R/W**	**Offset**
EMIFS_PRIOR	EMIFS LRU priority register	R/W	0x04
EMIFS_CONFIG	EMIFS configuration register	R/W	0x0C

Note: The EMIFS chip-select configuration register reset values depend on the input boot pin state at IC reset release time. For more details, see Section 3.2.18, *EMIFS Boot Mode*.

Table 16. EMIFS Registers (Continued)

Base Address = FFFE CC00			
Name	**Description**	**R/W**	**Offset**
EMIFS_CCS0	EMIFS chip-select configuration CS0	R/W	0x10
EMIFS_CCS1	EMIFS chip-select configuration CS1	R/W	0x14
EMIFS_CCS2	EMIFS chip-select configuration CS2	R/W	0x18
EMIFS_CCS3	EMIFS chip-select configuration CS3	R/W	0x1C
EMIFS_PTOR1	EMIFS dynamic priority time-out 1	R/W	0x28
EMIFS_PTOR2	EMIFS dynamic priority time-out 2	R/W	0x2C
EMIFS_PTOR3	EMIFS dynamic priority time-out 3	R/W	0x30
EMIFS_DWS	EMIFS dynamic wait states	R/W	0x40
EMIFS_AADDR	EMIFS abort address	R	0x44
EMIFS_ATYPER	EMIFS abort type	R	0x48
EMIFS_ATOR	EMIFS abort time-out	R/W	0x4C
EMIFS_ACS0	Advanced EMIFS chip-select configuration nCS0	R/W	0x50
EMIFS_ACS1	Advanced EMIFS chip-select configuration nCS1	R/W	0x54
EMIFS_ACS2	Advanced EMIFS chip-select configuration nCS2	R/W	0x58
EMIFS_ACS3	Advanced EMIFS chip-select configuration nCS3	R/W	0x5C

Note: The EMIFS chip-select configuration register reset values depend on the input boot pin state at IC reset release time. For more details, see Section 3.2.18, *EMIFS Boot Mode*.

Table 17. EMIFS Priority Register (EMIFS_PRIOR)

Base Address = 0xFFFE CC00, Offset = 0x04				
Bits	**Field**	**Description**	**R/W**	**Reset**
31:16	RESERVED	Reserved. To ensure software compatibility, reserved bit should be written to 0 and read value should be considered undefined.	R/W	0x0000
15:12	OCPI	OCPI consecutive access	R/W	0x0
11:8	DMA	DMA consecutive access	R/W	0x0
7	RESERVED	Reserved	R/W	0

Table 17. EMIFS Priority Register (EMIFS_PRIOR) (Continued)

Base Address = 0xFFFE CC00, Offset = 0x04				
Bits	**Field**	**Description**	**R/W**	**Reset**
6:4	DSP	DSP consecutive access	R/W	0
3	RESERVED	Reserved	R/W	0
2:0	MPU	MPU consecutive access	R/W	000

The EMIFS priority register allows the EMIFS to give consecutive accesses to a master when EMIFS is configured for least recently used (LRU) priority arbitration. The MPU and the DSP can have 0 to 7 consecutive accesses, whereas the DMA and the OCP-I can have 0 to 15 consecutive accesses, according to the content of the corresponding bits. For burst accesses, a four 32-bit burst access is considered as one access.

Table 18. EMIFS Configuration Register (EMIFS_CONFIG)

Base Address = 0xFFFE CC00, Offset = 0x0C				
Bit	**Field**	**Description**	**R/W**	**Reset**
31:5	RESERVED	Reserved. To ensure software compatibility, reserved bit should be written to 0 and read value should be considered undefined.	R/W	0x0000000
4	FR	Ready signal. This bit is a copy of the Ready input pin sample by TC_CK (2 TC_CK cycles delay from input pin to register update). 0: Ready pin is low 1: Ready pin is high	R	ND
3	PDE	System power-down acknowledge 0: Acknowledge disabled 1: Acknowledge enabled	R/W	0
2	PWD_EN	Dynamic auto idle 0: Disable 1: Enable	R/W	0

Table 18. EMIFS Configuration Register (EMIFS_CONFIG)

Base Address = 0xFFFE CC00, Offset = 0x0C				
Bit	**Field**	**Description**	**R/W**	**Reset**
1	BM	Boot mode. Enables CS0 and CS3 address decoding swapping (See section 3.2.18 for BM reset value.) 0: CS0 [0000:0000 − 03FF:FFFF] CS3 [0C00:0000 − 0FFF:FFFF] 1: CS0 [0C00:0000 − 0FFF:FFFF] CS3 [0000:0000 − 03FF:FFFF]	R/W	See note
0	WP	Write protect output pin control 0: WP output pin is set low. 1: WP output pin is set high.	R/W	0

Note: Reset value depends on external pins (see section 3.2.18).

Table 19. EMIFS Chip-Select Configuration Registers (EMIFS_CCS0, EMIFS_CCS1,...,EMIFS_CCS3)

Base Address = 0xFFFE CC00, Offsets = 0x10, 0x14, 0x18, 0x1C				
Bit	**Field**	**Description**	**R/W**	**Reset**
31	PGWSTEN	0: PGWST is specified by 15:12. 1: PGWST is specified by 30:27.	R/W	0
30:27	PGWST	When PGWSTEN is 1, this bit controls the wait states cycle number between accesses in a page for asynchronous page mode.	R/W	0000
26:23	BTWST	Controls the IDLE cycle number for bus turn-around and CS high-pulse-width timing.	R/W	0000
22	MAD	Enables EMIFS multiplexed address and data bus protocol (See section 3.2.18 for MAD reset value). 0: Non-multiplexed protocol 1: Multiplexed protocol	R/W	See note
21	RESERVED	Must be written to 0.	R/W	0
20	BW	Controls the data bus width used for this CS (See section 3.2.18 for BW reset value for CS0 and CS3). 0: Data bus is 16 bits wide. 1: Data bus is 32 bits wide.	R/W	See note or 1
19	RESERVED	Reserved for RDMODE expansion. Read value should be considered undefined.	R	ND

Note: The EMIFS chip-select configuration register reset values forCS0 and CS3 depend on a number of factors at reset release time. For more details, see Section 3.2.18, *EMIFS Boot Mode*.

Table 19. EMIFS Chip-Select Configuration Registers (EMIFS_CCS0, EMIFS_CCS1,...,EMIFS_CCS3) (Continued)(Continued)

Base Address = 0xFFFE CC00, Offsets = 0x10, 0x14, 0x18, 0x1C				
Bit	**Field**	**Description**	**R/W**	**Reset**
18:16	RDMODE	Read mode select (see table below and section 3.2.1). See section 3.2.18 for RDMODE reset value for CS0 and CS3.	R/W	See note or 000
15:12	PGWST/WELEN	Controls the wait states cycle number between accesses in a page for asynchronous page mode. Controls the WE pulse length during a write access. When PGWSTEN is 0, this bit specifies both PGWST/WELEN When PGWSTEN is 1, this bit specifies only WELEN	R/W	See note or 1111
11:8	WRWST	Controls the wait states cycle number for write operation.	R/W	See note or 1111
7:4	RDWST	Controls the wait states cycle number for asynchronous read operation and the initial idle time for asynchronous read page mode and synchronous read mode.	R/W	See note or 1111
3	RESERVED	Reserved. Writing to this bit has no effect. Reading it returns undefined value.	R/W	1
2	RT	Enable the read retimed protocol. This bit may be 1 only in RDMODE 4,5 and 7 only. The system will hang if the retiming bit is set in other modes. See section 3.2.18 for RT reset value for CS0 and CS3. 0: Non retimed protocol. 1: retimed protocol.	R/W	See note or 1
1:0	FCLKDIV	Controls the TC_CK divider. REF_CLK. 00: REF_CLK = TC_CK divide by 1. 01: REF_CLK = TC_CK divide by 2. 10: REF_CLK = TC_CK divide by 4. 11: REF_CLK = TC_CK divide by 6. See section 3.2.18 for RT reset value for CS0 and CS3.	R/W	See note or 11

Note: The EMIFS chip-select configuration register reset values forCS0 and CS3 depend on a number of factors at reset release time. For more details, see Section 3.2.18, *EMIFS Boot Mode*.

Table 20. EMIFS Chip-Select Configuration Register RDMODE Field Definition

RDMODE	Memory
000	Mode 0: Asynchronous read
001	Mode 1: Page mode ROM read—4 words per page
010	Mode 2: Page mode ROM read—8 words per page
011	Mode 3: Page mode ROM read—16 words per page
100	Mode 4: Synchronous burst read mode
101	Mode 5: Synchronous burst read mode
110	Reserved for future extension
111	Mode 7: Synchronous burst read mode

Table 21. EMIFS Time-Out Register 1 (EMIFS_PTOR1)

Base Address = 0xFFFE CC00, Offset = 0x28				
Bit	**Field**	**Description**	**R/W**	**Reset**
31:8	RESERVED	Reserved	R/W	0
7:0	DMA	Number of TC_CK cycles	R/W	0

Table 22. EMIFS Time-Out Register 2 (EMIFS_PTOR2)

Base Address = 0xFFFE CC00, Offset = 0x2C				
Bit	**Field**	**Description**	**R/W**	**Reset**
31:8	RESERVED	Reserved	R/W	0
7:0	DSP	Number of TC_CK cycles	R/W	0

Table 23. EMIFS Time-Out Register 3 (EMIFS_PTOR3)

Base Address = 0xFFFE CC00, Offset = 0x30				
Bit	**Field**	**Description**	**R/W**	**Reset**
31:8	RESERVED	Reserved	R/W	0
7:0	OCPI	Number of TC_CK cycles	R/W	0

Time-out registers 1−3 are used to control the number of TC clock cycles before DSP, DMA, or OCP requests are made high priority in the dynamic priority scheme used inside TC.

Table 24. EMIFS Dynamic Wait States Control Register (EMIFS_DWS)

Base Address = 0xFFFE CC00, Offset = 0x40				
Bit	**Field**	**Description**	**R/W**	**Reset**
31:8	Reserved	Reserved	R/W	0x000000
7	Full handshake enable for CS3	Enables Full-/non-full-handshaking mode for CS3 0: Full-handshaking 1: Non-full-handshaking	R/W	0
6	Full handshake enable for CS2	Enables Full-/non-full-handshaking mode for CS2 0: Full-handshaking 1: Non-full-handshaking	R/W	0
5	Full handshake enable for CS1	Enables Full-/non-full-handshaking mode for CS1 0: Full-handshaking 1: Non-full-handshaking	R/W	0
4	Full handshake enable for CS0	Enables Full-/non-full-handshaking mode for CS0 0: Full-handshaking 1: Non-full-handshaking	R/W	0
3	Dynamic wait states enable for CS3	Enables dynamic wait states mode for CS3 0: Dynamic wait states mode disabled 1: Dynamic wait states mode enabled	R/W	0
2	Dynamic wait states enable for CS2	Enables dynamic wait states mode for CS2 0: Dynamic wait states mode disabled 1: Dynamic wait states mode enabled	R/W	0
1	Dynamic wait states enable for CS1	Enables dynamic wait states mode for CS1 0: Dynamic wait states mode disabled 1: Dynamic wait states mode enabled	R/W	0
0	Dynamic wait states enable for CS0	Enables dynamic wait states mode for CS0 0: Dynamic wait states mode disabled 1: Dynamic wait states mode enabled	R/W	0

This register controls if EMIFS has dynamic wait states by using the ready signal.

Table 25. EMIFS Abort Address Register (EMIFS_AADDR)

Base Address = 0xFFFE CC00, Offset = 0x44				
Bit	**Field**	**Description**	**R/W**	**Reset**
31:0	AA	Abort address	R	0x00000000

This register holds the address involved in the aborted transaction.

Table 26. EMIFS Abort Type Register (EMIFS_ATYPER)

Base Address = 0xFFFE CC00, Offset = 0x48				
Bit	**Field**	**Description**	**R/W**	**Reset**
31:5	Reserved	Reserved. To ensure software compatibility, reserved bit should be write to 0 and read value should be considered undefined.	R/W	0x0000000
4	TOE	Time out error. Time out status bit set to 1 if abort was caused by a time-out error.	R	0
3	RAE	Restricted access error. Restricted access status bit is set to 1 if abort was caused by a restricted access error.	R	0
2:1	HID	Host ID. Specify the source of the aborted transaction: 00: MPU 01: DSP 10: DMA 11: OCP-I	R	00
0	ABORT_FLAG	Abort status bit. Reading the abort type register reset the abort status bit. 0: No abort 1: Abort	R	0

This register reports the type of the transaction that has been aborted.

Table 27. EMIFS Abort Time-Out Register (EMIFS_ATOR)

Base Address = 0xFFFE CC00, Offset = 0x4C				
Bit	**Field**	**Description**	**R/W**	**Reset**
31:9	Reserved	Reserved. To ensure software compatibility, reserved bit should be write to 0 and read value should be considered undefined.	R/W	0x0000000
8	TIMEOUT_EN	Enable the time-out timer. 0: Timer is disable 1: Timer is enabled	R/W	1
7:0	TIMEOUT	Time out counter value in REF_CLK clock cycles.	R/W	0xFF

Table 28. Advanced EMIFS Chip-Select Configuration Registers (EMIFS_ACS0, EMIFS_ACS1,...,EMIFS_ACS3)

Base Address = 0xFFFE CC00, Offset = 0x50, 0x54, 0x58, 0x5C				
Bit	**Field**	**Description**	**R/W**	**Reset**
31:10	Reserved	Reserved. To ensure software compatibility, reserved bit should be write to 0 and read value should be considered undefined.	R/W	0x000000
9	BTMODE	Enables extended BTWST usage 0: Bus turn around control and RD to RD/WR same CS pulse width high control 1: Bus turn around control and RD/WR to RD/WR same CS pulse width high control	R/W	0
8	ADVHOLD	Controls the ADV pulse width low	R/W	0
7:4	OEHOLD	Controls the number of cycles from OE high to CS high	R/W	0x0
3:0	OESETUP	Controls the number of cycles inserted from CS low to OE low. When the MAD reset value is 1, the reset value of OESETUP is 0x2 (See section 3.2.18 for MAD reset).	R/W	0x0 or 0x2

3.7 EMIFF Registers

Table 29 lists the 32-bit EMIFF registers. Table 30 through Table 53 describe the register bits.

Table 29. EMIFF Registers

Base Address = 0xFFFE CC00			
Name	**Description**	**R/W**	**Offset**
EMIFF_PRIOR	EMIFF priority register	R/W	0x08
EMIFF_CONFIG	EMIFF configuration register	R/W	0x20
EMIFF_MRS *	EMIFF SDRAM MRS register (legacy)	R/W	0x24
EMIFF_CONFIG2	EMIFF configuration register 2	R/W	0x3C
EMIFF_DLL_WRD_CTRL	DLL WRD control register (write byte)	R/W	0x64
EMIFF_DLL_WRD_STAT	DLL WRD status register	R	0x68
EMIFF_MRS_NEW *	EMIFF SDRAM MRS register (new)	R/W	0x70
EMIFF_EMRS0	EMIFF SDRAM EMRS0 register	R/W	0x74
EMIFF_EMRS1	EMIFF SDRAM EMRS1 register	R/W	0x78
EMIFF_OP	EMIFF SDRAM operation register	R/W	0x80
EMIFF_CMD	EMIFF SDRAM manual command register	R/W	0x84
EMIFF_PTOR1	EMIFF dynamic arbitration priority time-out1	R/W	0x8C
EMIFF_PTOR2	EMIFF dynamic arbitration priority time-out2	R/W	0x90
EMIFF_PTOR3	EMIFF dynamic arbitration priority time-out3	R/W	0x94
EMIFF_AADDR	EMIFF abort address register	R	0x98
EMIFF_ATYPER	EMIFF abort type register	R	0x9C
EMIFF_DLL_LRD_STAT	DLL LRD status register	R	0xBC
EMIFF_DLL_URD_CTRL	DLL URD control register (read lower byte)	R/W	0xC0
EMIFF_DLL_URD_STAT	DLL URD status register (read upper byte)	R	0xC4

Note: EMIFF_MRS is a legacy register. Old software can use this register at offset 0x24. However, new software should use the EMIFF_MRS_NEW register at 0x70.

Table 29. EMIFF Registers (Continued)

Base Address = 0xFFFE CC00			
Name	**Description**	**R/W**	**Offset**
EMIFF_EMRS2	EMIFF SDRAM EMRS2 register	R/W	0xC8
EMIFF_DLL_LRD_CTRL	DLL LRD control register	R/W	0xCC

Note: EMIFF_MRS is a legacy register. Old software can use this register at offset 0x24. However, new software should use the EMIFF_MRS_NEW register at 0x70.

Table 30. EMIFF Priority Register (EMIFF_PRIOR)

Base Address = 0xFFFE CC00, Offset = 0x08				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:16	RESERVED	Reserved	R	0x0000
15:12	L3_OCP	L3 OCP consecutive access	R/W	0000
11:8	DMA	DMA consecutive access	R/W	0000
7	RESERVED	Reserved	R	0
6:4	DSP	DSP consecutive access	R/W	000
3	RESERVED	Reserved	R	0
2:0	MPU	MPU consecutive access	R/W	000

The EMIFF priority register allows the EMIFF to give consecutive accesses to a master when the master has been granted the EMIFF interface. The MPU and DSP can have 0 to 7 consecutive accesses and the DMA and the Level3 OCP initiator can have 0 to 15 consecutive accesses, according to the content of the corresponding bits.

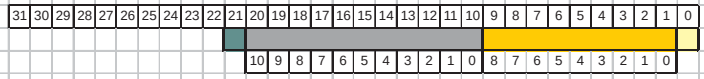
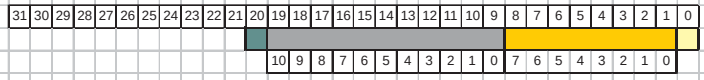
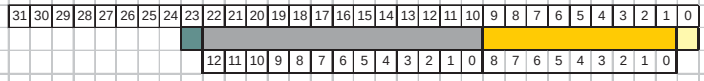
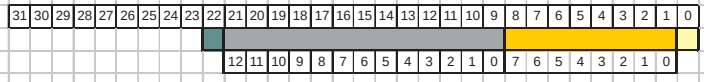
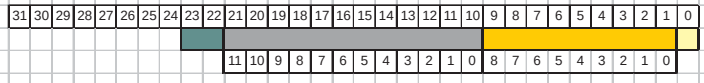
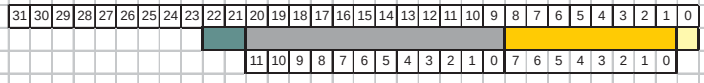
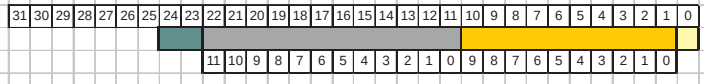
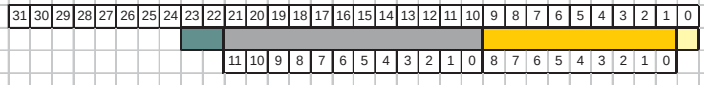
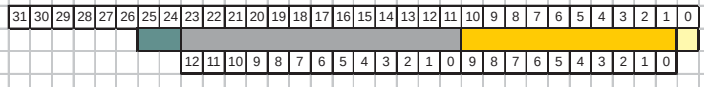
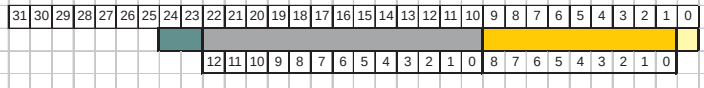
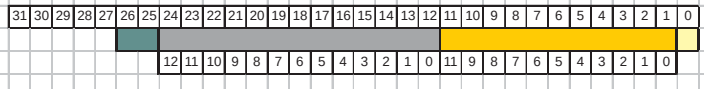
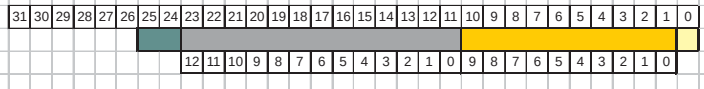
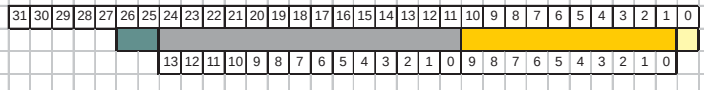
Table 31. EMIFF SDRAM Configuration Register (EMIFF_CONFIG)

Base Address = 0xFFFE CC00, Offset = 0x20				
Bit	**Field**	**Description**	**R/W**	**Reset**
31:30	Reserved	Must be 00	R	00
29:28	LG SDRAM Type	Used to define the larger SDRAM memories (256MB). See Table 32.	R/W	00
27	CLK	Disable SDRAM clock. 0: Enable the clock to the external SDRAM bank 1: Disable the clock to the external SDRAM bank	R/W	0

Table 31. EMIFF SDRAM Configuration Register (EMIFF_CONFIG)

Base Address = 0xFFFE CC00, Offset = 0x20				
Bit	**Field**	**Description**	**R/W**	**Reset**
26	PWD	Power down enable. Puts the SDRAM device into power down mode. The CKE signal to the device is held high only for an active transaction. This bit must be enabled if the autoclock gating is used (see SD_AUTO_CLK in the SDRAM configuration 2 register, Table 52).	R/W	0
25:24	SDRAM frequency	SDRAM frequency range. To control the idle time of SDRAM regarding to the clock organization: 00: SDF0 (reset value) 01: SDF1 10: SDF2 11: SDF3	R/W	00
23:8	ARCV	Autorefresh counter register value. This value is calculated using the formula: Value = (Refresh Interval/clock period/number of rows) − 50	R/W	0x6188
7:4	SDRAM type	Set the SDRAM internal organization.	R/W	0x0
3:2	ARE	Autorefresh enable. When autorefresh enable is set, the EMIFF generates a REFR request, depending on the autorefresh counter and the burst refresh counter. If refresh enable is not set, the refresh must be done as a RAS_only refresh under CPU control. 00: Autorefresh disable 01: Autorefresh enable (one command every 14.7 µs) 10: Autorefresh by burst of 4 commands 11: Autorefresh by burst of 8 commands	R/W	00
1	Reserved	Must be 1	R	1
0	Slrf	Self-refresh, when set, puts the SDRAM in self-refresh mode if an access is made to the SDRAM after the SDRAM automatically comes out of self-refresh.	R/W	0

Table 32. EMIFF SDRAM Register Memory/Data Bus Size

SDRAM address multiplexing with respect to system 32−bit address	SDRAM Type Size/Org./Bank	Number of devices	EMIFF config register [29:28][7:4]	Memory space size
	16M x 8 x 2b	2	000000	4 Mbytes
	16M x 16 x 2b	1	000010	2 Mbytes
	64M x 8 x 2b	2	000100	16 Mbytes
	64M x 16 x 2b	1	000110	8 Mbytes
	64M x 8 x 4b	2	000101	16 Mbytes
	64M x 16 x 4b	1	000111	8 Mbytes
	128M x 8 x 4b	2	001001	32 Mbytes
	128M x 16 x 4b	1	001011	16 Mbytes
	256M x 8 x 4b	2	001101	64 Mbytes
	256M x 16 x 4b	1	001111	32 Mbytes
	512M x 8 x 4b	2	010000	128 Mbytes
	512M x 16 x 4b	1	001101	64 Mbytes
	1G x 16 x 4b	1	100000	128 Mbytes

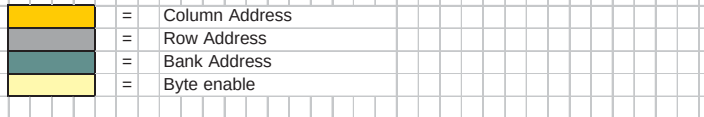


Table 33. Frequency Range (SDRAM)

ac Parameters	SDF0 (Cycles)	SDF1 (Cycles)	SDF2 (Cycles)	SDF3 (Cycles)
tRC	9	5	3	2
tRAS	6	3	3	2
tRP	3	2	2	2
tRCD	3	2	2	2
tRRD	2	2	2	2

Table 34. Frequency Range (Mobile DDR)

ac Parameters	SDF0 (Cycles)	SDF1 (Cycles)	SDF2 (Cycles)	SDF3 (Cycles)
tRC	12	7	3	2
tRAS	8	5	3	2
tRP	5	4	2	2
tRCD	4	2	2	2
tRRD	3	2	2	2

Depending on the value programmed into the SDRAM_TYPE field of the EMIFF SDRAM operation register, either the ac parameters for SDRAM or the ac parameters for mobile DDR are supported; that is, the SDRAM frequency field is decoded into the values in either Table 33 or Table 34, not both. See section 3.3.7 for more information.

Table 35. EMIFF SDRAM MRS Register (legacy for OMAP3.1)

Base Address = 0xFFFE CC00, Offset = 0x24				
Bit	**Field**	**Description**	**R/W**	**Reset**
31:10	Reserved	Must be all 0	R	0x000000
9	WBST	Write burst must be 0 (burst write same as burst read).	R/W	0
8:7	Reserved	Must be 00.	R	00
6:4	CASL	CAS idle time must be set to 3 for a mobile DDR device: 001: Reserved 011: CAS idle time = 3 (default)	R/W	011
3	S/I	Serial = 0 / Interleave = 1 (must be serial)	R/W	0
2:0	PGBL	Page burst length. Must be set to full page burst (111) for SDRAM and burst of 8 (011) for DDR SDRAM.	R/W	111

Note: After the memory exits self−refresh mode, the first thing the software should do is write to the EMIFF_MRS (legacy or new) register so that the SDRAM's mode register is set properly.

Table 36. EMIFF SDRAM Configuration Register 2 (EMIFF_SDRAM_ CONFIG_2_REG)

Base Address = 0xFFFE CC00, Offset = 0x3C				
Bit	**Field**	**Description**	**R/W**	**Reset**
31:3	Reserved	Must be all 0.	R	0x0000000
2	sd_auto_clk	Allow controller to suspend its internal clocks when idle. The clocks are automatically reenabled when there is an autorefresh or host request. 0: Disable (Reset) 1: Enable This bit must be set in conjunction with the CLK bit in the SDRAM configuration register in order to turn off the clock to the external SDRAM device. Also, the PWD bit in the configuration register must be set to 1 for autogating to be effective.	R/W	0
1	Rfrsh_reset	Place the SDRAM into self_refresh when in reset (active high).	R/W	1
0	Rfrsh_stdby	Place the SDRAM into self_refresh when in standby mode (active high).	R/W	1

Table 37. DLL WRD Control Register (EMIFF_DLL_WRD_CTRL)

Base Address = 0xFFFE CC00, Offset = 0x64				
Bit	**Field**	**Description**	**R/W**	**Reset**
31:26	Reserved	Must be all 0.	R	0x00
25:20	WO	Write offset. 6-bit delay fine adjustment, signed, range –32…+31. One step represents a 26.3 ps±10.5 ps delay adjustment. Effective in both DLL enabled and DLL disabled mode. Used for delaying the write clock signal	R/W	0x00
19:16	Reserved	Must be all 0.	R	0000
15:8	DLY	Delay. 8-bit delay to adjust the digitally controlled delay, to be used when the DLL is disabled. Range 0…225 One step represents a 26.3 ps±10.5 ps delay adjustment.	R/W	0x00
7:4	Reserved	Must be all 0s.	R	0000
3	LDLL	Load DLL. Allows loading the delay field value into the DLL module, as the initial value for the tracking counter, or to force a given delay. 0: No action 1: The DLL is loaded with the delay value, if ENADLL is 1. The DLL tracking engine is stalled.	R/W	0
2	DLLP	DLLPhase. Nominal digitally controlled delay selection. This bit has no effect if DLL is disabled 0: 72 degrees (20% of the clock period) 1: 90 degrees (25% of the clock period)	R/W	0
1	ENADLL	0: DLL is disabled 1: DLL is enabled.	R/W	0
0	RESERVED	Must be 0	R	0

Table 38. DLL WRD Status Register (EMIFF_DLL_WRD_STAT)

Base Address = 0xFFFE CC00, Offset = 0x68				
Bit	**Field**	**Description**	**R/W**	**Reset**
31:16	Reserved	Must be all 0.	R	0x0000
15:8	CNT	DLL Count. Current DLL counter value for monitoring/debug (assumes control bit ENADLL is 1 in DLL Control register.	R	0x00
7:3	Reserved	Must be all 0.	R	0x00
2	LOCK	DLL lock status (future, not in the current design) 0: DLL is not locked. 1: DLL is properly locked.	R	0
1	UDF	Underflow status 0: DLL is OK. 1: DLL counter underflow	R	0
0	OVF	Overflow status 0: DLL is OK. 1: DLL counter overflow	R	0

Table 39. EMIFF SDRAM MRS_NEW Register (EMIFF_MRS_NEW)

Base Address = 0xFFFE CC00, Offset = 0x70				
Bit	**Field**	**Description**	**R/W**	**Reset**
31:10	Reserved	Must be all 0	R	0x000000
9	WBST	Write burst musts be 0 (burst write same as burst read)	R/W	0
8	ResetDLL	This bit resets the memory device DLL when set to 1.	R/W	0
7	Reserved	Must be 0	R	0
6:4	CASL	CAS idle time: 001: Reserved 010: CAS idle time = 2 011: CAS idle time = 3 (default) Must be set to 2 for DDR	R/W	011
3	S/I	Serial = 0/Interleave=1 (must be serial)	R/W	0
2:0	PGBL	Page burst length. Must be set to full page burst (111) for SDRAM and burst of 8 (011) for DDR SDRAM.	R/W	111

Note: After the memory exits self−refresh mode, the first thing the software should do is write to the EMIFF_MRS (legacy or new) register so that the SDRAM's mode register is set properly.

Note that there is only one physical MRS register. Using that address, no automatic initialization sequence is generated; only a LOAD MODE register command is issued.

A CPU write to this register generates a LOAD MODE register command, with BA1,BA0 = 0,0.

Table 40. EMIFF DDR SDRAM Register (EMIFF_EMRS0)

Base Address = 0xFFFE CC00, Offset = 0x74				
Bit	**Field**	**Description**	**R/W**	**Reset**
31:10	Reserved	Must be all 0.	R	0x0000000
9:3	Unused	No special function, but these bits are passed to the memory.	R/W	0x00
2	QFC	QFC enable bit. 0: Disabled 1: Enabled	R/W	0
1	DS	DS driver strength bit. 0: Normal 1: Reduced	R/W	0
0	DLL	DLL enable bit. 00: Enabled 1: Disabled	R/W	0

This register is used for DDR SDRAM memory only. It provides access to extended configuration fields. The description is given here with reference to standard devices, but must be checked with the specification of the device actually used in a given application.

A CPU write to this register generates a LOAD MODE register command, with BA1=0 and BA0 = 1. Twelve bits can be loaded.

Table 41. EMIFF Low Power SDRAM Register (EMIFF_EMRS1)

Base Address = 0xFFFE CC00, Offset = 0x78				
Bit	**Field**	**Description**	**R/W**	**Reset**
31:10	Reserved	Must be all 0.	R	0x000000
9:5	Unused	No special function, but these bits are passed to the memory.	R/W	0000

Table 41. EMIFF Low Power SDRAM Register (EMIFF_EMRS1) (Continued)

Base Address = 0xFFFE CC00, Offset = 0x78				
Bit	**Field**	**Description**	**R/W**	**Reset**
4:3	TCSR	Temperature-compensated self-refresh 00: 70°C maximum temperature 01: 45°C maximum temperature 10: 15°C maximum temperature 11: 85°C maximum temperature	R/W	00
2:0	PASR	Partial-array self-refresh 000: All banks 001: 1/2 array (BA1=0) 010: 1/4 array (BA1=BA0=0) 011: Reserved 100: Reserved 101: 1/8 array (BA1=BA0=0, RA11=0) 110: 1/16 array (BA1=BA0=0, RA11=RA10=0) 111: Reserved	R/W	000

Note: Bit designation is given for a 128M-bit device.

This register is used for SDRAM memories dedicated for low power wireless applications. The description is given here with reference to standard devices, but must be checked with the specification of the device actually used in a given application.

A CPU write to this register generates a LOAD MODE register command, with BA1=1 and BA0 = 0. Twelve bits can be loaded.

Table 42. EMIFF SDRAM Operation Register (EMIFF_OP)

Base Address = 0xFFFE CC00, Offset = 0x80				
Bit	**Field**	**Description**	**R/W**	**Reset**
31:25	Time-out_B3	Time-out value for Bank3, in TC clock cycles.	R/W	0x00
24:18	Time-out_B2	Time-out value for Bank2.	R/W	0x00
17:11	Time-out_B1	Time-out value for Bank1.	R/W	0x00
10:4	Time-out_B0	Time-out value for Bank0.	R/W	0x00

Table 42. EMIFF SDRAM Operation Register (EMIFF_OP) (Continued)

Base Address = 0xFFFE CC00, Offset = 0x80				
Bit	**Field**	**Description**	**R/W**	**Reset**
3:2	Operation mode	00: Low-power/low-bandwidth mode (LPLB mode). 01: High-power/high-bandwidth mode (HPHB mode). 10: Programmable operating mode 0 (POM0 mode). 11: Reserved. Must not be used.	R/W	01
1:0	SDRAM type	The type of SDRAM: 00: Regular SDR SDRAM 01: Regular DDR SDRAM 10: Low-power SDR SDRAM 11: Mobile DDR SDRAM	R/W	00

Table 43. EMIFF SDRAM Manual Command Register (EMIFF_CMD)

Base Address = 0xFFFE CC00, Offset = 0x84				
Bit	**Field**	**Description**	**R/W**	**Reset**
31:4	Reserved	Must be 0000:000.	R	0x0000000
3:0	SDRAM	Manual command 0000: NOP command 0001: Precharge command 0010: Autorefresh command 0011: Enter deep sleep command 0100: Exit deep sleep command 0111: Set CKE signal high 1000: Set CKE signal low 1xxx: Reserved. Not for use.	R/W	0x0

Table 44. EMIFF Dynamic Arbitration Priority Time-Out Register 1 (EMIFF_PTOR1)

Base Address = 0xFFFE CC00, Offset = 0x8C				
Bit	**Field**	**Description**	**R/W**	**Reset**
31:7	Reserved	Must be all 0.	R	0x000000
7:0	DMA	Number of clock cycles before DMA requests are made high priority in the dynamic priority scheme for the EMIFF SDRAM interface.	R/W	0x00

Table 45. EMIFF Dynamic Arbitration Priority Time-Out Register 2 (EMIFF_PTOR2)

Base Address = 0xFFFE CC00, Offset = 0x90				
Bit	**Field**	**Description**	**R/W**	**Reset**
31:7	Reserved	Must be all 0.	R	0x000000
23:16	DSP	Number of clock cycles before DSP requests are made high priority in the dynamic priority scheme for the EMIFF SDRAM interface.	R/W	0x00
15:8	Reserved	Reserved	R	0x00
7:0	LCD	Number of clock cycles before LCD requests are made high priority in the dynamic priority scheme for the EMIFF SDRAM interface.	R/W	0x00

Table 46. EMIFF Dynamic Arbitration Priority Time-Out Register 3 (EMIFF_PTOR3)

Base Address = 0xFFFE CC00, Offset = 0x94				
Bit	**Field**	**Description**	**R/W**	**Reset**
31:7	Reserved	Must be all 0.	R	0x000000
7:0	L3	Number of clock cycles before L3 OCP initiator requests are made high priority in the dynamic priority scheme for the EMIFF SDRAM interface.	R/W	0x00

Time-out registers 1−3 are used to store the number of clock cycles before DSP, DMA, or Level 3 OCP initiator requests are made high priority in the dynamic priority scheme for the EMIFF SDRAM interface.5rrP@2a0mill

Table 47. EMIFF Abort Address Register (EMIFF_AADDR)

Base Address = 0xFFFE CC00, Offset = 0x98				
Bit	**Field**	**Description**	**R/W**	**Reset**
31:0	Abort Address	Address of the transaction aborted	R	0x00000000

Table 48. EMIFF Abort Type Register (EMIFF_ATYPER)

Base Address = 0xFFFE CC00, Offset = 0x9C				
Bit	**Field**	**Description**	**R/W**	**Reset**
31:3	Reserved	Must be all 0s	R	0x0000000
2:1	HOSTID	ID of the host whose transaction was aborted 00: MPU 01: DSP 10: DMA 11: OCP-I	R	00
0	ABORT_FLAG	Set when an abort occurs, reset when this register is read.	R	0

Table 49. DLL LRD Status Register (EMIFF_DLL_LRD_STAT)

Base Address = 0xFFFE CC00, Offset = 0xBC				
Bit	**Field**	**Description**	**R/W**	**Reset**
31:0	Reserved	Must be all 0s.	R	0x00000000

Table 50. DLL URD Control Register (EMIFF_DLL_URD_CTRL)

Base Address = 0xFFFE CC00, Offset = 0xC0				
Bit	**Field**	**Description**	**R/W**	**Reset**
31:26	Reserved	Must be all 0s	R	0x00
25:20	Read offset	6-bit QDS delay fine adjustment, signed, range –32…+31. Effective in both DLL enabled and DLL disabled mode. Used for delaying the read dqs for the upper byte. One step represents a 26.3 ps±10.5 ps delay adjustment.	R/W	0x00
19:16	Reserved	Must be all 0s	R	0x0

Table 50. DLL URD Control Register (EMIFF_DLL_URD_CTRL)(Continued)

Base Address = 0xFFFE CC00, Offset = 0xC0				
Bit	**Field**	**Description**	**R/W**	**Reset**
15:8	Delay	8-bit delay to adjust the digitally controlled delay, to be used when the DLL is disabled. Range 0…225 One step represents a 26.3 ps±10.5 ps delay adjustment	R/W	0x00
7:4	Reserved	Must be all 0s	R	0x0
3	LOADDLL	Allows loading the delay field value into the DLL module, as the initial value for the tracking counter, or to force a given delay. 0: No action. 1: The DLL is loaded with the delay value, if ENADLL is 1. The DLL tracking engine is stalled.	R/W	0
2	DLLPHASE	Nominal digitally controlled delay selection. This bit has no effect if DLL is disabled 0: 72 degrees (20% of the clock period) 1: 90 degrees (25% of the clock period)	R/W	0
1	ENADLL	0: DLL is disabled. 1: DLL is enabled.	R/W	0
0	Reserved	Must be 0.	R	0

This register controls the DLL for the upper read byte.

Table 51. DLL URD Status Register (EMIFF_DLL_URD_STAT)

Base Address = 0xFFFE CC00, Offset = 0xC4				
Bit	**Field**	**Description**	**R/W**	**Reset**
31:16	Reserved	Must be all 0.	R	0x0000
15:8	DLLCOUNT	Current DLL counter value for monitoring/debug (assumes control bit ENADLL is 1 in DLL control register.	R	0x00
7:3	Reserved	Must be all 0.	R	0x0
2	Lock	DLL lock status (future, not in the current design) 0: DLL is not locked. 1: DLL is properly locked.	R	0

Table 51. DLL URD Status Register (EMIFF_DLL_URD_STAT) (Continued)

Base Address = 0xFFFE CC00, Offset = 0xC4				
Bit	**Field**	**Description**	**R/W**	**Reset**
1	UDF	Underflow status 0: DLL is OK. 1: DLL counter underflow	R	0
0	OVF	Overflow status 0: DLL is OK. 1: DLL counter overflow	R	0

This register controls the DLL for the lower read byte.

Table 52. EMIFF SDRAM Register (EMIFF_EMRS2)

Base Address = 0xFFFE CC00, Offset = 0xC8				
Bit	**Field**	**Description**	**R/W**	**Reset**
31:0	Reserved	Must be all 0.		

This register is provided to anticipate its use in future designs by memory manufacturers and must not be used by current applications.

A CPU write to this register generates a LOAD MODE register command, with BA1, BA0 = 1,1. Twelve bits can be loaded.

Table 53. DLL LRD Control Register (EMIFF_DLL_LRD_CTRL)

Base Address = 0xFFFE CC00, Offset = 0xCC				
Bit	**Field**	**Description**	**R/W**	**Reset**
31:26	Reserved	Must be all 0s	R/W	0x00
25:20	Read offset	6-bit QDS delay fine adjustment, signed, range –32…+31. Effective in both DLL enabled and DLL disable modes. Used for delaying the read dqs for the lower byte. One step represents a 26.3 ps±10.5 ps delay adjustment.	R/W	0x00
19:0	Reserved	Must be all 0s	R/W	0x00000

3.8 OCPI Registers

Table 54 lists the OCP registers. Table 55 through Table 60 describe the register bits.

Table 54. OCP Registers

Base Address = 0xFFFE C320			
Name	**Register Description**	**R/W**	**Offset**
OCPI_AFR	OCP address fault	R	0x00
OCPI_MCFR	OCP master command fault	R	0x04
OCPI_ATYPER	OCP type of abort	R/W	0x0C
OCPI_PR	OCP protection	R/W	0x14
OCPI_SMR	OCP-I secure mode. See note.	R/W	0x18

Note: The reset value of the OCP protection register is all ones, such that all buses are protected on reset.

Table 55. OCPI Address Fault Register (OCPI_AFR)

Base Address = 0xFFFE C320, Offset = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:0	Address	Address accessed by the master that causes an abort or error.	R	0

Table 56. OCPI Master Command Fault Register (OCPI_MCFR)

Base Address = 0xFFFE C320, Offset = 0x04				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:3	Reserved	Reserved. Must be 0.	R	0
2:0	MCMD	Command that caused an abort or error.	R	0

Table 57 lists tThe supported commands.

Table 57. Master Command Register Supported Commands

MCmd	**Transaction Type**
000	Idle
001	Write
010	Read
011	ReadEX
OTHER	Not supported/not interpreted

For non-supported Mcmd encoding, OCPI translates it to IDLE.

The READEX command is for atomic transfer: Read-modify-write. This command is internally decoded to generate a lock signal for the OMAP3 traffic

controller and is replaced by a simple read with a lock signal to the subtargets. The READEX command *must* be followed by a write that matches the address of the READEX as specified in Sonic's OCP specification. A READEX command followed by a read results in unpredictable behavior.

Table 58. OCPI Type of Abort Register (OCPI_ATYPER)

Base Address = 0xFFFE C320, Offset = 0x0C				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:4	Reserved	Reserved. Must be 0.	R	0
3	Burst Error	Burst access to the MPUI or TIPB was requested.	R/C	0
2	PROTECT	Address hit a protected area.	R/C	0
1	TRGABORT	Abort coming from the accessed target	R/C	0
0	ADDDEC	Address decoding error (initiator sent unknown address)	R/C	0

Note: R/C is clear after read.

The abort type register is cleared after being read(R/C). In other words, reading the register once returns whatever bits correspond to the abort type. On the next read, the register returns all 0s. Also a write to this register has no effect.

Table 59. OCPI Protection Register (OCPI_PR)

Base Address = 0xFFFE C320, Offset = 0x14				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:8	Reserved	Reserved. Must be 0.	R/W	0
7	API	Access to MPUI is prohibited.	R/W	1
6	RHEA_PUB	Access to MPU public TIPB is prohibited.	R/W	1
5	RHEA_PRIV	Access to MPU private TIPB is prohibited.	R/W	1
4	OCPMULT	Access to OCPT multibank is prohibited.	R/W	1
3	OCPT2	Access to OCPT2 is prohibited.	R/W	1
2	OCPT1	Access to OCPT1 is prohibited.	R/W	1
1	EMIFF	Access to EMIFF is prohibited.	R/W	1
0	EMIFS	Access to EMIFS is prohibited.	R/W	1

When a bit in the OCPI_PR register is 1, access to the corresponding bit field from the OCPI bus is protected (prohibited).

Table 60. Secure Mode Register (OCPI_SMR)

Base Address = 0xFFFE C320, Offset = 0x18				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:7	Reserved	Reserved. Must be 0.	R/W	0
6	API	In secure mode 0: Access is allowed. 1: Access to API is prohibited from initiator.	R/W	1
5	RHEA_PRIV	In secure mode 0: Access is allowed. 1: Access to MPU TIPB public is prohibited from initiator. TIPB private bus is not accessible regardless of the setting of this bit.	R/W	1
4	OCPMult	In secure mode 0: Access is allowed. 1: Access to OCP multibank is prohibited from initiator.	R/W	1
3	OCPT2	In secure mode 0: Access is allowed. 1: Access to OCPT2 is prohibited from initiator.	R/W	1
2	OCPT1	In secure mode 0: Access is allowed. 1: Access to OCPT1 is prohibited from initiator.	R/W	1
1	EMIFF	In secure mode 0: Access is allowed. 1: Access to EMIFF is prohibited from initiator.	R/W	1
0	EMIFS	In secure mode 0: Access is allowed to CS1−CS3. 1: Access to EMIFS CS1−CS3 is prohibited from initiator; CS0 is not accessible regardless of the setting of this bit.	R/W	1

When not in secure mode, these secure mode register values are ignored. When in secure mode, EMIFS CS0 and TIPB private access is not allowed, regardless of the register values. Every other target secure mode is determined by the register value.

3.9 Priority Algorithms

The traffic controller provides a choice of two priority algorithms for simultaneous requests. Arbitration is performed in each TC target port (OCP-T1, OCP-T2, EMIFF, and EMIFS).

Selection of the arbitration scheme is common to all TC ports. Depending on the OMAP device, it can be either hardwired to one of the two algorithms or programmable (outside of OMAP).

- Enhanced round robin with LRU

  In accordance with the standard round-robin scheme, the requestor having the highest priority becomes the lowest-priority requestor after it has been granted access.

  The first enhancement is that when it gets the highest priority, a requestor can keep it for a programmable number of consecutive accesses.

  The number of consecutive accesses is individually programmable for the MPU, DSP, the system DMA, and OCP-I. See OCPT1_PRIORITY, OCPT2_PRIORITY, EMIF_SLOW_PRIORITY, and EMIF_FAST_PRIORITY registers.

  In parallel with the round-robin scheme, the second enhancement is that the least recently granted requestor always has the highest priority. That means if a requestor has missed its slot because it had no request pending at that time, it keeps the highest priority for the subsequent arbitration cycles.

- Dynamic priority order

  Most of the time, the priority order is fixed: highest priority is MPU, then DSP, OCP-I, and lowest priority is DMA. A programmable time-out is attached to any host except the MPU to ensure that it is not blocked indefinitely by higher priority hosts.

  When a low-priority requestor gains the arbitration (i.e. DSP, OCP-I, DMA), the associated time-out counter is loaded with the programmed value and starts decrementing.

  If a counter reaches 0, the associated requestor gets the highest priority for its next request (that may be already pending). If several requestors are in this situation, the priority order is: DMA LCD, DSP, OCP-I, DMA other than LCD.

4 Clock Generation and Reset Management

4.1 Overview

The clock generation and system reset module is part of the MPU subsystem in the OMAP 3.2 platform. This module manages the clock generation modes for the microprocessor unit (MPU), the digital signal processor (DSP), and various other subsystems (memory interface, system DMA controller, MPU port interface (MPUI), etc.). These clocks can be controlled by software from registers described in Section 4.5. It also monitors the system reset and initiates the reset sequences for each clock domain. Finally, it controls the power-saving modes and generates wake-up controls to the processors and peripherals.

Clock generation modes include:

- Programmable clocking mode (synchronous, synchronous scalable, and mix modes)
- Programmable clock for different clock domains (MPU, DSP, and traffic controller (TC) clock domains)
- Programmable clock for different peripherals (internal liquid crystal display (LCD) controller and external MPU and DSP TIPB peripherals)
- Programmable low-frequency clocks (derived from input reference clock) to supply the internal MPU and DSP timers
- Fixed low-frequency clocks to supply watchdog timers for the DSP and MPU
- Direct memory access (DMA) clock request mechanism (provides DMA clock during data transfer only)
- External visibility on internally generated clocks on POCLKOUT pins

System reset includes:

- Global software reset
- Reset control for the MPU, DSP, and external TIPB peripherals
- System and reset status monitoring

Power-saving modes and wake-up control includes:

- Programmable power-saving mode and idle mode controls for the MPU, the DSP, the traffic controller, and their respective subdomains

- Power control for external device reset/power on (flash memory)
- Wake-up functions initiated by interrupts (MPU and DSP) and DMA requests (traffic controller and TIPB) in the idle mode
- Initiation of the wake-up sequence by external devices during the idle mode

4.2 OMAP3.2 Clock Generation

The clock domains in the OMAP 3.2 hardware engine platform are synthesized by the digital phase-locked loops (DPLL). The DPLL input clock source (CK_REF) is supplied from the ULPD.

The DPLL1 output frequencies are programmable and can be further divided down to provide clocks to the MPU, the DSP, and the TC domains. The MPU domain, the DSP domain, and the TC domain are clocked from DPLL1.

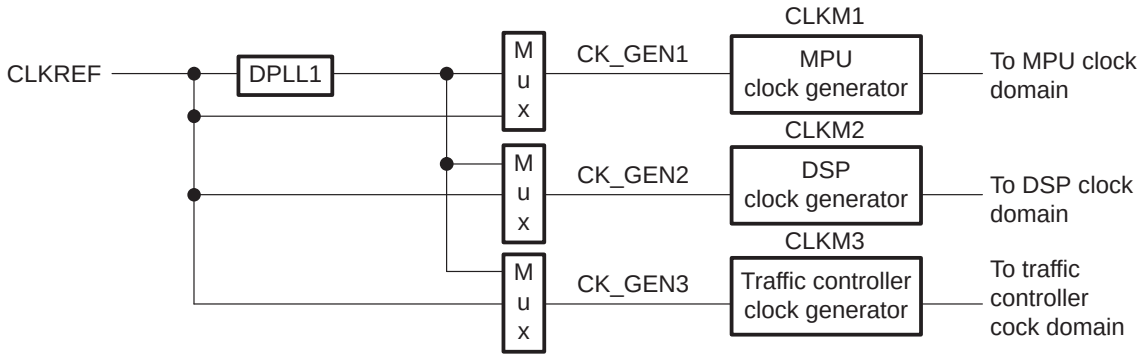
This implementation offers the clock rate selection flexibility to adjust the clock frequency of each clock domain and allows the OMAP 3.2 hardware engine to adjust each clock domain to its optimal frequency. In addition, each domain is further subdivided into subdomains so that each subdomain can be independently activated/deactivated while the remaining part of the clock network is in an idle state.

The OMAP clock system is organized around three main clock domains: MPU, DSP, and TC clock domains.

- The MPU clock domain contains: MPU, MPU external peripheral clocks, MPU watchdog timer, MPU internal timers, and MPU interrupt handler.
- The DSP clock domain contains: DSP, DSP MMU, DSP external peripheral clocks, DSP watchdog timer, DSP internal timers, and DSP interrupt handler.
- The TC clock domain contains: TC, L3 OCP-I, MPUI port interface, system DMA controller, MPU TIPB bridges, and LCD controller, and OCP-T1 and OCP–T2.

Figure 42 shows the clock generator module.

Figure 42. Clock Generator Module



4.2.1 Clock Generation Modes

The clock generation and system reset module of the OMAP 3.2 hardware engine supports four kinds of clocking modes:

- ❏ Fully synchronous
- ❏ Synchronous scalable
- ❏ Bypass mode
- ❏ Mix modes (#3 and #4)

These clocking modes provide the system with the maximum flexibility for performance and power-saving capabilities. They are programmable by the CLOCK_SELECT field of the (ARM_SYSST) register, and the power-up default mode is the full synchronous mode.

Table 61 details the hardware engine different clocking modes.

Table 61. OMAP 3.2 Hardware Engine Clocking Modes

Clock Select	Clocking Operating Mode	MPU Clock Source	DSP Clock Source	TC Clock Source	Remarks
000	Fully synchronous	DPLL1/N	DPLL1/N	DPLL1/N	See *Fully Synchronous Mode*
001	Reserved				
010	Synchronous scalable	DPLL1/M	DPLL1/N	DPLL1/O	See *Synchronous Scalable Mode*
011	Reserved				

Table 61. OMAP 3.2 Hardware Engine Clocking Modes (Continued)

Clock Select	Clocking Operating Mode	MPU Clock Source	DSP Clock Source	TC Clock Source	Remarks
100	Reserved				
101	Bypass mode	CK_REF	CK_REF	CK_REF	Input reference clock
110	Mix mode #3: MPU synchronous to TC, DSP synchronous scalable to TC and MPU	DPLL1/N	DPLL1/M	DPLL1/N	See *Mix Modes*
111	Mix mode #4: DSP synchronous to TC, MPU synchronous scalable to TC and DSP	DPLL1/M	DPLL1/N	DPLL1/N	See *Mix Modes*

Fully Synchronous Mode

In fully synchronous mode, the MPU, DSP, and TC domains run at the same clock frequency derived from DPLL1. This is the power-up default mode. The fully synchronous mode is a special case of synchronous scalable mode, where the clock divider bits for all domains are equal. However, there is separate clock select encoding for fully synchronous mode. It is the programmer's responsibility to ensure that all the clock divider select bits are set to the same value.

When the fully synchronous mode is selected, you must program the divide-down bits of the ARM_CKCTL register so that ARMDIV, DSPMMUDIV, DSPDIV, and TCDIV are equal.

Synchronous Scalable Mode

In synchronous scalable mode, the MPU, DSP, and TC domains are synchronous and run at different clock speeds. The clock feeding mechanism is similar to that of the fully synchronous mode, except that the clocks are multiples of one another.

In synchronous scalable mode, the divide-down bits ARMDIV, DSPDIV, and TCDIV of the ARM_CKCTL register define the prescaler value from the frequency of DPLL.

When the synchronous scalable mode is selected, you must program the divide-down bits of the ARM_CKCTL register so that DSPMMUDIV = DSPDIV or DSPDIV*2.

The TC clock frequency must be the same speed or slower than the MPU, DSP, and the DSP MMU clocks.

Mix Modes

Clock generation supports two mix modes.

- Mix mode #3

  The MPU and TC clock domains are synchronous (same clock frequency), and the DSP is scaled synchronous (synchronous but with a frequency that is a multiple of the MPU/TC clock frequency). In this mode, the TC, MPU, and DSP receive clocks from the DPLL1 output.

  When mix mode #3 is selected, you must program the divide-down bits of the ARM_CKCTL register so that ARMDIV = TCDIV.

  The TC clock frequency must be the same speed or slower than the DSP and the DSP MMU clock frequencies.

- Mix mode #4

  The DSP and TC clock domains are synchronous (same clock frequency), while MPU is scaled synchronous (synchronous but with a frequency that is a multiple of the DSP/TC clock frequency). In this mode, the TC, MPU, and DSP receive clocks from the DPLL1 output.

  When mix mode #4 is selected, you must program the divide-down bits of the ARM_CKCTL register so that DSPDIV = DSPMMUDIV = TCDIV. Because ARM_CK supplies the host processor and TC_CK supplies different memory interfaces, the restriction on the speed of TC_CK ensures that the rate of instruction/data fetch is never more than the rate at which data can be processed.

  The TC clock frequency must be the same speed or slower than the MPU clock frequency.

Bypass Mode

In bypass mode (CLOCK_SELECT = 101), the DPLL is bypassed and the input reference clock is directly fed to the MPU, DSP, and TC clock domains.

4.2.2 DPLL

The DPLL block synthesizes a frequency clock from the fixed reference input clock signal CK_REF using the digital phase locked loop mechanism. Only the MPU can access the DPLL control register.

DPLL Modes

The DPLL can operate either in bypass mode or in lock mode.

- Bypass mode

  In bypass mode (PLL_ENABLE bit of the DPLL1_CTL_REG register set to 0), the DPLL output clock can be CK_REF (input reference clock), CK_REF/2, or CK_REF/4, depending on the BYPASS_DIV bit-field value of the DPLL1_CTL_REG register.

- Lock mode

  In lock mode (PLL_ENABLE bit of DPLL1_CTL_REG register set to 1), the output frequency is an integer multiple or fractional multiple (m/n respectively, in the PLL_MULT and PLL_DIV bit fields of DPLL1_CTL_REG register) of the input reference clock CK_REF. With $1 \leq m \leq 31$ and $1 \leq n \leq 4$, the frequency output ranges from CK_REF/4 to 31×CK_REF.

Synthesizing a Clock

At reset, the DPLL is in bypass mode and the BYPASS_DIV bit field of the DPLL1_CTL_REG register is set to 0b00 (DPLL output clock = CK_REF).

The procedure to synthesize a clock at a desired frequency is as follows:

1) Set the PLL_MULT and PLL_DIV bit fields of the DPLL1_CTL_REG register to the correct value in order to get the desired multiplication factor.

2) Set the PLL_ENABLE bit to 1 to enter the lock mode.

3) When the DPLL has reached the desired synthesized clock frequency, the bit LOCK bit of DPLL1_CTL_REG register goes to 1 and the output clock gets the synthesized clock.

The bit fields PLL_MULT and PLL_DIV can be modified on-the-fly even when the DPLL is in lock mode.

Polling can be done on the LOCK bit to determine when the DPLL locks on the desired synthesized frequency. The DPLL output clock is switched smoothly between the bypass and the locked frequency because it is not mandatory to wait for the DPLL entering lock mode before proceeding the DPLL output clock.

When idle mode of the DPLL is exited, the DPLL is set in bypass mode and the output signal is valid (locked) after a maximum of 10 input reference clock cycles. The output is valid after a maximum of 12 input reference clock cycles

in bypass mode and switches to locked clock in an other 32 maximum reference clock cycles. If the DPLL was synthesizing a frequency prior to the idle state, the DPLL switches from bypass mode to synthesizer frequency when the lock state is reacquired.

4.2.3 MPU Clock Domain

The DPLL1 output frequency defines the speed of the MPU and the MPU external peripherals. The clock from DPLL1 output is supplied to the OMAP boundary. It has a software gating in the ARM_IDLECT1 (IDL_CLKOUT_ARM) and in ARM_IDLECT2 (EN_CKOUT_ARM).

At reset, DPLL1 is in bypass mode (CK_GEN1 = CK_REF).

The MPU clock domain is subdivided into five subdomains.

- MPU (ARM_CK)

  You can program the divide-down ARMDIV bits of the ARM_CKCTL register to have the DPLL1 output clock (CK_GEN1) further divided by 1, 2, 4, or 8 to supply the clock signal driving the MPU. At reset, the highest frequency (divided by 1) is selected: ARM_CK = CK_GEN1 = CK_REF.

- MPU external peripheral (ARMPER_CK or ARMXOR_CK)

  You can program the divide-down PERDIV bits of the ARM_CKCTL register to have CK_GEN1 further divided by 1, 2, 4, or 8 to supply the MPU external peripheral clock ARMPER_CK signal at the OMAP boundary. At reset, the highest frequency (divided by 1) is selected and ARMPER_CK is active. ARMXOR_CK, a gated version of CK_REF, can also be used to supply the external peripherals. At reset, this clock is inactive.

- OMAP3.2 MPU internal OS timers (ARMTIM_CK)

  The ARM_TIMXO bit of the ARM_CKCTL register selects either CK_GEN1 divided by 1 or the input reference clock (CK_REF) to supply the internal MPU timers. At reset, CK_GEN1 is selected but the timer clock is inactive.

- MPU Level 1 and 2 interrupt handlers (ARM_INTH_CK)

  The MPU interrupt handlers are supplied by a programmable clock, and the user can choose between the MPU clock or the divided-by-2 MPU clock using the ARM_INTHCK_SEL bit of the ARM_CKCTL register. The MPU clock is supplied as the default clock.

- 32-bit MPU watchdog timer (ARMWDI_CK)

  The 32-bit MPU watchdog timer is supplied with a low-frequency clock (CK_REF/14). This clock is active at reset.

Even when the MPU is not in idle mode, you have the option of individually disabling the clock to the MPU subdomains via the ARM_IDLECT2 register. This allows power saving when a module is not used.

4.2.4 DSP Clock Domain

Depending on the OMAP clocking mode, the DPLL1 output frequency defines the speed of the DSP domain. The clock output from DPLL1 (CK_GEN2) can be further divided in order to supply the clock of the DSP and its subsystems. At reset, DPLL1 is in bypass mode (CK_GEN1 = CK_GEN2 = CK_REF).

The DSP clock domain is divided into six subdomains:

- DSP (DSP_CK)

  The clock signal driving the DSP can be further divided by 2, 4, or 8 by programming the divide-down bits DSPDIV of ARM_CKCTL register. At reset, the highest frequency (divided by 1) is selected and DSP_CK = CK_GEN2 = CK_REF.

- DSP MMU (DSPMMU_CK)

  The clock signal driving the DSP MMU (DSPMMU_CK) can be further divided by 2, 4, or 8 by programming the divide-down bits DSPMMUDIV of the ARM_CKCTL register. At reset, the highest frequency (divided by 1) is selected but the DSP MMU clock is inactive.

Note:

You must program DSPDIV and DSPMMUDIV so that the DSPMMU_CK clock frequency is either one or one half times the DSP_CK clock frequency.

- DSP external peripheral clock (DSPPER_CK)

  The DSP external peripheral clock can be further divided by 2, 4, or 8 by programming the divide-down bits PERDIV of the DSP_CKCTL register.

  DSPXOR_CK, a gated version of the CK_REF, can also be used to supply the external peripherals. This clock is inactive at reset.

- DSP watchdog timer (DSPWDT_CK)

  DSP watchdog timer is supplied with a low-frequency clock (CK_REF/14). This clock is active at reset.

- DSP internal timers (DSPTIM_CK)

  The TIMXO bit of the DSP_CKCTL register selects either CK_GEN2 divided by 2, or the input reference clock (CK_REF) to supply the internal DSP

timers. At reset, the clock issued from the DPLL is selected but the timer clocks are inactive.

- DSP Level 1 and 2.0 interrupt handlers (DSP_INTH_CK)

  DSP interrupt handlers are supplied with CK_GEN2 divided by 2.

Even when the DSP is not in idle mode, you have the option of individually disabling these subdomains using the DSP_IDLECT2 register. This allows significant power saving when a module is not in use.

4.2.5 Traffic Controller Clock Domain

The DPLL output frequency, which drives the traffic controller, generates the traffic controller clock (TC_CK). This TC_CK feeds the traffic controller, the OCP initiator port (OCP-I), the OCP Target1 (OCP-T1) and OCP Target2 (OCP-T2) ports, the system DMA controller, the LCD controller, the MPUI port interface, and the MPU TIPB bridge. TC1_CK and TC2_CK are then broadcast outside the OMAP platform; they are identical to TC_CK and can be powered down independently in power saving options.

The TC clock domain is divided into two subdomains:

- Traffic controller, OCP-I port, OCP-T1 and OCP-T2 ports, MPUI port interface, system DMA controller, and MPU TIPB bridges.

  The clock signal driving these modules is basically the same as the TC_CK, except that it can be gated independently of TC_CK.

  You can program the divide-down TCDIV bits of the ARM_CKCTL register to have the CK_GEN3 further divided by 2, 4, or 8 to generate the TC_CK. At reset, the highest frequency (divided by 1) is selected and TC_CK = CK_REF.

  At reset, the MPUI port interface clock and the system DMA controller clock are inactive, while the OCP-I, OCP-T1, and OCP-T2 port clocks, TC clocks, and MPU TIPB bridge are active.

- LCD controller

  You can program the divide-down LCDDIV bits of the ARM_CKCTL register to have CK_GEN3 further divided by 2, 4, or 8 to generate the LCD controller clock. At reset, the highest frequency (divided by 1) is selected but the LCD controller clock is inactive.

These traffic controller subdomain clocks can be disabled even if the MPU, DSP, or TC are not in idle mode using the ARM_IDLECT2 register.

The DMA needs a free-running clock supplied to the external LCD controller (SoSSI) even when the DMA clock is turned off so that the LCD controller can

generate proper interrupts. This free-running clock for external LCD controller can only be cut off when the external LCD controller is in idle state.

4.3 Power-Saving Modes and Wake-Up Control

This section describes the following power-saving features:

- MPU idle control
- DSP idle control
- Traffic controller idle control
- System DMA idle control
- MPU TIPB bridge idle control
- External device power control
- DPLL idle control
- Chip idle mode/deep sleep mode
- Wake-up control

4.3.1 MPU Idle Control

The OMAP 3.2 hardware engine can operate in several power-saving modes that can reduce the operating current by stopping the clock signals of unused (inactive) domains without losing any data on operational context. When the idle state is entered, the MPU domain clocks are turned off according to the orderly sequenced events. The clock gating cell design ensures that clocks are properly stopped and restarted without parasitic pulses.

Activating the wait-for-interrupt MPU instruction initiates the MPU idle mode. It stops the MPU internal clocks, and then the STANDBYWFI signal from MPU megacell is asserted high, indicating that the MPU internal idle state is entered.

Before the idle mode is entered, you can stop the MPU internal timer clock, the LCD clock, the external peripheral clock, and the timer/watchdog clock by setting to 0 the corresponding bits of the ARM_IDLECT2 register, or you can set the corresponding bits of ARM_IDLECT1 register so that these peripherals automatically go to idle when the MPU goes to idle (except for the LCD).

When the timer/watchdog timer is configured as a watchdog, its clock (CK_REF/14) is never shut down, regardless of the value of the IDLWDT_ARM bit in the ARM_IDLECT1 register, or the EN_WDTCK bit in the ARM_IDLECT2 register.

The idle command is forwarded to the MPU interrupt handler and the MPU clock is stopped when the interrupt handler acknowledges this request (nopending interrupts).

When the MPU internal clocks are stopped, the MPU domain clocks are stopped if they were not already disabled using ARM_IDLECT2 before MPU went to idle.

- The ARM_CK is stopped in a low static state after some synchronization cycles.
- If the IDLCLKOUT_ARM bit field of ARM_IDLECT1 is set to 1, then the DPLL output clock also goes to idle after some synchronization cycles.
- If the IDLPER_ARM bit field of ARM_IDLECT1 is set to 1, then the ARMPER_CK also goes to idle if the ARMPER_IDLE_REQ request is acknowledged.
- If the IDLXOR_ARM bit field of ARM_IDLECT1 is set to 1, then the ARMXOR_CK also goes to idle after some synchronization cycles.
- If the IDLWDT_ARM bit field of ARM_IDLECT1 is set to 1, then the ARMWDT_CK also goes to idle after some synchronization cycles, if the MPU watchdog module is not set.
- If the IDLTIM_ARM bit field of ARM_IDLECT1 is set to 1, then the ARMTIM_CK also goes to idle after some synchronization cycles.
- If the IDLAPI_ARM bit field of ARM_IDLECT1 is set to 1, then the MPUI clock also goes to idle whenever the MPUI is not required.
- If the IDLDPLL_ARM bit field of ARM_IDLECT1 is set to 1, then the DPLL macro goes to idle when all the clock domains are in idle.

Even when the MPU is not in idle mode, the user has the option of individually disabling the MPU subdomain clocks by setting the corresponding bit in the ARM_IDLECT2.

A wake-up sequence is initiated in the MPU domain only upon:

- A system reset

or

- An unmasked MPU interrupt request, assuming that the WKUP_MODE bit of ARM_IDLECT1 is set to 1 or the chip is not in idle. In case of chip idle and WKUP_MODE set to 0, the external wake-up control feature is enabled and a CHIP_nWKUP low along with the interrupt is needed to wake-up the MPU.

On wake-up, all the subdomains put in idle mode using ARM_IDLECT1 are restarted if the corresponding enable bits of ARM_IDLECT2 are set.

4.3.2 DSP Idle Control

The DSP idle instruction must be executed in host-only mode (HOM) to initiate the DSP idle mode. Depending on the settings of the DSP idle control registers

(DSP_IDLECT1 and DSP_IDLECT2), different parts of the DSP subsystem go to idle mode when the IDLE instruction is executed.

The following procedure describes how the DSP enters idle mode:

1) Disable the watchdog timer.

   When the timer/watchdog timer is configured as a watchdog, its clock (CK_REF/14) is never shut down.

2) Disable the following DSP peripheral clocks by setting the corresponding bits in DSP_IDLECT2 register to 0s.

   - DSP external peripheral clock
   - External reference peripheral clock

   This disables the clocks immediately, regardless of whether the DSP clock is enabled or not. Or set the DSP_IDLECT1 corresponding bits to 1s, which disables the DSP peripheral clocks only when the DSP clock is disabled.

3) Switch the DSP TIPB and MPUI to shared access mode (SAM).

4) Program the DSP idle control register to put all the DSP subsystem domains in idle mode.

5) Switch the DSP TIPB and MPUI to host-only mode.

6) Execute the IDLE instruction.

7) When IDLE_DSP = 1 in ARM_SYSST register, the DSP_CK stops.

8) A signal is sent to the OMAP DSP interrupt handler to disable the interrupts to the DSP while the DSP clock is being disabled.

9) The DSP interrupt handler clock also stops after the synchronization cycles end.

10) The DSP clock subdomain goes to idle mode when both the DSP and the MPU clocks are disabled and the corresponding DSP_IDLECT1 bits are set to 1.

Even when the DSP or MPU is not in idle mode, you have the option of individually disabling the DSP domain clocks by setting the corresponding DSP_IDLECT2 enable bits to 0s.

A wake-up sequence is initiated in DSP domain only upon one of the following events:

- A system reset

or

- A DSP reset

or

- An unmasked DSP interrupt request. The interrupt request restarts the DSP clock if the WKUP_MODE bit of ARM_IDLECT1 is set to 1 or if the chip is not in idle. In case of chip idle and WKUP_MODE set to 0, the external wake-up control feature is enabled and a CHIP_nWKUP low in conjunction with the interrupt required to wake up the DSP clock.

On wake up, all the DSP subdomains put in idle mode using DSP_IDLECT1 are restarted following the sequence described below if corresponding enable bits of ARM_IDLECT2 are set.

- Service and clear the interrupt event.
- Switch the DSP TIPB and MPUI to SAM mode.
- Write 0 in the DSP ICR idle mode configuration register bits for each subdomain to be restarted.
- Execute an idle instruction to force the reread of the DSP ICR idle mode configuration register and restart the subdomain clocks affected.

4.3.3 Traffic Controller, System DMA Controller, and MPU TIPB Bridges Idle Control

Specific conditions must be met for the traffic controller, the system DMA controller, and the MPU TIPB bridges to enter the idle mode.

Traffic Controller Idle Control

To enter idle mode, the traffic controller must meet the following conditions:

- ❑ MPU and DSP must be set to global idle mode.
- ❑ L3 OCP-I is in idle mode. This can be done either by clearing the EN_OCPI_CK bit of the ARM_IDLECT3 register to 0, or by setting the IDLOCPI_ARM bit in the same register to 1 (which allows disabling of the OCPI clock in conjunction with the MPU clock). The OCP initiator bus enable signal (L3_OCPI_EN) must always be 0 before the OCPI clock stops.

- ❏ OCP-T1/T2 modules are in idle mode. This can be done by setting ARM_IDLECT3 (IDLTC1_CK) = 1 and ARM_IDLECT3 (IDLTC2_CK) = 1, which disables the TC1_CK and TC2_CK clocks when there is no activity and the idle request is acknowledged by the target. These modules can also be placed in idle mode by disabling TC1_CK and TC2_CK completely by setting EN_TC1_CK and EN_TC2_CK to 0.
- ❏ The idle interface (IDLIF_ARM) bit of the ARM_IDLECT1 register is set to logical 1.
- ❏ The MPUI clock is idle.
- ❏ There are no system DMA pending transfers.
- ❏ All the TC subdomains are stopped by setting the appropriate bits in the ARM_IDLECT2 and ARM_IDLECT3 registers.
- ❏ There are no MPU or DSP interrupt requests.
- ❏ Power-down enable bits PDE and PWD_EN of the EMIFS configuration register EMIFS_CONFIG are set to logical 1.
- ❏ Disable the SDRAM clock and set the power-down enable bit to 1 with the EMIFF configuration register. TC idle mode entry is affected by the RFRSH_STBY bit of the EMIFF_SDRAM_CONFIG_2REG TC register. When set to 1, the SDRAM must be put into self refresh state before going idle (SLRF bit of EMIFF_CONFIG register). Every time the SDRAM wakes up, the self-refresh state is cleared, so returning to idle requires that the SLRF bit of EMIFF_CONFIG_REG is set to 1 again.

It is important to note that ceratin bits in the ARM_IDLECT1,2,3 registers can prevent the chip from entering the idle state.

- ■ If the ARM_IDLECT2.EN_APICK bit is 1, you need either the ARM_EWUPCT.REPWR_EN bit to be 0 or you need the ARM_IDLECT3.IDLOCPI_ARM bit to be 1 to go to idle.
- ■ If ARM_IDLECT2.EN_APICK is 0, ARM_EWUPCT.REPWR_EN and ARM_IDLECT3.IDLOCPI_ARM are don’t care and will not prevent you from going to idle.
- ■ If the EN_API_CK bit field of ARM_IDLECT2 is 1, then IDLAPI_ARM bit field of ARM_IDLECT1 needs to be 1 to go to idle

Then the traffic controller completes its current operations and pulls the TCIDLE_ACK signal to a high level to indicate that TC_CK can now be safely stopped. In addition, the shut-down of the TC_CK clock indicates to the chip idle control logic to initiate the DPLL idle.

The TC_CK restarts upon:

- An MPU or DSP interrupt request
- A DMA request
- L3_OCPI_EN pin set to logic 1 (enables restarting of the clock to L3 OCP initiator bus)

System DMA Idle Control

The system DMA employs a built-in power-saving mechanism. The clock is only requested to the clock generator when DMA transfers are occurring.

The DMA clock can enter idle mode if one of the following conditions is true:

- DMACK_REQ = 1 and there are no DMA requests.
- DMACK_REQ = 0, the MPU clock is in idle, IDLIF_ARM = 1, there are no DMA requests, and the DMAIDLE_ACK signal is high.

MPU TIPB Bridges Idle Control

The TIPB bridges can enter idle mode only when all of the following conditions are true:

- MPU is set in idle mode.
- The idle interface bit IDLIF_ARM of ARM_IDLECT1 register is set to logical 1.
- There are no system DMA requests to the TIPB.
- There is no posted write (that is, posted write buffers are empty).

4.3.4 External Device Power Control

The FLASH.RP signal is an output pin that allows the reset/power-on control sequences of external devices such as flash memory.

Whenever the traffic controller enters the idle mode, the FLASH.RP pin switches from a high to a low level, allowing external components to be turned off. When a wake-up condition is detected, the pin is switched back to a high level and restores power to external devices.

Setting the bit REPWR_EN of ARM_EWUPCT to logical 0 enables this capability. At reset, this is disabled.

To allow the external device/component voltage to stabilize (ramp-up) when the power-down mode is released, the external power control is implemented

with a programmable counter that delays the restart of all clocks from FLASH.RP signal going high. The EXTPWR bit field of the ARM_EWUPCT register permits the delay to be defined as follows:

$WT_{(wake\text{-}up\ time)} = (EXTPWR_{(field\ value)} \pm 1) \times CK\_REF_{(period)}$

4.3.5 DPLL Idle Control

The DPLL can be set to idle mode if only the input reference clock (CK_REF) is needed.

The DPLL idle mode is entered when the IDLDPLL_ARM bit of the ARM_IDLECT1 register is set to logical 1 and all of the domains that use the DPLL clock are stopped. This means that the only domains running (domains that use CK_REF rather than the DPLL clock) are:

- MPU and DSP watchdog timers (CK_REF/14)
- Internal MPU timers when ARM_TIMXO bit of the ARM_CKCTL register is set to logical 0
- Internal DSP timers when TIMXO bit of the DSP_CKCTL register is set to logical 0

The DPLL idle mode entry/exit time can be significant. The input reference clock must be active for at least 24 input clock cycles from the idle request (idle rising edge) before the idle setup is complete. Once the idle mode is exited, the DPLL is set in bypass mode and the output signal is valid after a maximum of 10 input reference clock cycles. The total DPLL idle entry-to-exit sequence takes no less than 34 reference clock cycles. Therefore, it may be preferable not to shutdown the DPLL when MPU or DSP have to be stopped for a short period of time or when critical operations are likely to occur (that is, DMA transfer, interrupt handling).

4.3.6 Chip Idle Mode, Deep Sleep Mode, and Wake-up Control

The OMAP 3.2 hardware engine is considered to be in chip idle mode when the MPU, DSP, peripherals, DPLL, and peripherals using CK_REF as their source are stopped.

Once the procedures for MPU idle, DSP idle, traffic controller idle, and DPLL idle have been followed, the chip idle state is reached. The following ordering is recommended to ensure that all of the clock domains can be made idle:

- The first domain to idle is the DSP clock domain. Ensure that the MPUI interface has been correctly initialized, and that the API_SIZE value is zero. Then follow the sequence for DSP idle already described

(sending a command from the MPU to DSP via the mailbox is one way to start this sequence). If the DSP has not been enabled after reset, it is only necessary to clear the ARM_CKCTL(EN_DSPCK) register bit to 0 in order to reach the idle state for this domain.

- The MPU must wait until the DSP has reached the idle state.
- Disable the MPU watchdog timer.
- Set the ARM_IDLECT1, ARM_IDLECT2, and ARM_IDLECT3 register bits in preparation for going to idle (MPU, DSP, TC, and DPLL idle entries are all affected by these registers).
- Configure the EMIFS and EMIFF modules as described in the traffic controller idle control section. If SDRAM contents must be maintained during the idle state, then the SDRAM self-refresh mode must be enabled before going to chip idle (set the SLFR bit of the EMIFF_SDRAM_CONFIG register to 1, and the RFRSH_STDBY bit of the EMIFF_SDRAM_CONFIG2 register to 1). The self-refresh bit is cleared every time OMAP leaves chip idle.
- Prepare for wake up by enabling and unmasking MPU interrupts.
- Ensure that all interrupts and DMA status bits have been cleared. If all of the other idle conditions and controls have been met (as per the MPU, DSP, TC and DPLL descriptions), then activating the wait-for-interrupt instruction at this point leads to the full chip idle state, allowing the reference clock to be stopped.

In chip idle mode, the MPU, the DSP, the DPLL and peripherals that use CK_REF as their source are stopped, while the external clock source remains the only active clock signal.

In deep sleep mode, all internal system clocks (MPU, DSP, DPLL, peripherals, and timers) and the external reference clock source are stopped, leaving the OMAP3 in a static state in which it consumes the lowest possible power. In this mode, it is recommended that the WKUP_MODE bit of the ARM_IDLECT1 register be set to 0 before going into IDLE. A complete handshake between the OMAP and external module turns off the OMAP input clock. This handshake ensures that OMAP wakes up properly.

Any unmasked interrupt request, either to the MPU or the DSP, any DMA clock request, or setting the L3_OCPI_EN signal to high exits the idle mode.

When the WKUP_MODE bit of ARM_IDLECT1 is set to logical 0, the wake-up procedure can be controlled by the ULPD.

When the WKUP_MODE bit value is set to logic 1, a single wake-up condition initiates a chip wake-up procedure. The wake-up condition can be caused by:

- An interrupt request from MPU interrupt handler. The MPU interrupt handler sets the nIRQ_SET signal to logic low and initiates the restarting of the ARM_CK, ARM_INTH_CK, Rhea_CK, DMA_CK, and TC_CK clocks. Depending on the setting of the ARM_IDLECT1/2 registers, peripherals clocks can also restart. This is a valid wake-up condition for MPU, TC, and DPLL.

- An interrupt request from the DSP level2 interrupt handler. This initiates the restarting of the DSP_CK, DSP_INTH_CK, and TC_CK clocks. Depending on the setting of the ARM_IDLECT1/2 registers, peripheral clocks can also restart. This signal must remain active until the DSP asserts the DSP_IDLE signal low. This is a valid wake-up condition for DSP, TC, and DPLL.

- L3_OCPI_EN pin: When the L3_OCPI_EN pin is pulled high, the TC_CK, TC1_CK, and TC2_CK, and the L3_OCPI_CK restart and the TC_CK, L3_OCPI_CK, TC1_CK, and TC2_CK keep running as long as the pin remains asserted high. This is a valid wake-up condition for TC and DPLL only.

- TCLB_DMAREQ: When the system DMA controller receives an asynchronous request from the traffic controller, this signal is set high to enable the DMA_CK/TC_CK and DMA_CK/TC_CK to keep running as long as the DMA operates. This is a valid wake-up condition for TC and DPLL only.

- Rhea_DMAREQ: When the system DMA controller receives a request from the TIPB-bridge, this signal is asserted high to enable the TC_CK/Rhea_CK/DMA_CK and the TC_CK/Rhea_CK/DMA_CK to keep running as long as the DMA operates. This is a valid wake-up condition for TC and DPLL only.

4.4 Registers

All registers are 32-bit registers. MPU clock generation and system reset control registers are accessed by the MPU only. DSP control registers are accessed by the DSP and the MPU through the MPU interface.

4.4.1 MPU Registers

The MPU registers are listed in Table 62. Table 63 through Table 72 provide register bit descriptions.

Table 62. MPU Registers

Base Address = 0xFFFE CE00			
Name	**Description**	**R/W**	**Offset**
ARM_CKCTL	MPU clock control prescaler selection	R/W	0x00
ARM_IDLECT1	MPU idle enable control 1	R/W	0x04
ARM_IDLECT2	MPU idle enable control 2	R/W	0x08
ARM_EWUPCT	MPU restore power delay	R/W	0x0C
ARM_RSTCT1	Master software reset	R/W	0x10
ARM_RSTCT2	Peripherals reset	R/W	0x14
ARM_SYSST	MPU clock reset status	R/W	0x18
ARM_CKOUT1	MPU clock out definition	R/W	0x1C
ARM_CKOUT2	MPU reserved	R/W	0x20
ARM_IDLECT3	MPU idle enable control 3	R/W	0x24

Table 63. MPU Clock Control Prescaler Selection Register (ARM_CKCTL)

Base Address = 0xFFFE CE00, Offset = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:15	RESERVED	See note.	R/W	0x0000
14	ARM_INTHCK_SEL	This bit controls which clock is used for the ARM_INTH_CK 0: ARM_INTH_CK clock is same as ARM_CK (default). 1: ARM_INTH_CK is half frequency of ARM_CK.	R/W	0
13	EN_DSPCK	Turns on DSP_CK while the DSP is still in reset state. 0: Disables DSP_CK activation during the reset state. 1: Enables DSP_CK activation during the reset state.	R/W	1
12	ARM_TIMXO	Selects a subfrequency issued either from CK_GEN1 or from input reference clock (CK_REF) to supply internal MPU timers. 0: ARMTIM_CK clock frequency is the input reference clock (CK_REF). 1: ARMTIM_CK clock frequency is issued from CK_GEN1.	R/W	1

Note: For reserved bits, reading gives undefined values. Writing to has no effect.

Table 63. MPU Clock Control Prescaler Selection Register (ARM_CKCTL) (Continued)

Base Address = 0xFFFE CE00, Offset = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
11:10	DSPMMUDIV	Define prescaler value from the frequency of CK_GEN2 to DSPMMU clock domain. 00: CK_GEN2 01: CK_GEN2/2 10: CK_GEN2/4 11: CK_GEN2/8	R/W	00
9:8	TCDIV	Define prescaler value from the frequency of CK_GEN3 to TC clock domain. 00: CK_GEN3 01: CK_GEN3/2 10: CK_GEN3/4 11: CK_GEN3/8	R/W	00
7:6	DSPDIV	Define the prescaler value from the frequency of CK_GEN2 to DSP clock domain. 00: CK_GEN2 01: CK_GEN2/2 10: CK_GEN2/4 11: CK_GEN2/8	R/W	00
5:4	ARMDIV	Define the prescaler from the frequency of CK_GEN1 to MPU clock domain. 00: CK_GEN1 01: CK_GEN1/2 10: CK_GEN1/4 11: CK_GEN1/8	R/W	00

Note: For reserved bits, reading gives undefined values. Writing to has no effect.

Table 63. MPU Clock Control Prescaler Selection Register (ARM_CKCTL) (Continued)

Base Address = 0xFFFE CE00, Offset = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
3:2	LCDDIV	Define prescaler value from the frequency of CK_GEN3 to LCD controller clock signal 00: CK_GEN3 01: CK_GEN3/2 10: CK_GEN3/4 11: CK_GEN3/8	R/W	00
1:0	ARM_PERDIV	Define the prescaler value from the frequency of CK_GEN1 to MPU external peripheral clock domain. 00: CK_GEN1 01: CK_GEN1/2 10: CK_GEN1/4 11: CK_GEN1/8	R/W	00

Note: For reserved bits, reading gives undefined values. Writing to has no effect.

Table 64. MPU Idle Enable Control Register 1 (ARM_IDLECT1)

Base Address = 0xFFFE CE00, Offset = 0x04				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:13	RESERVED	See note.	R/W	0x0000
12	IDL_CLKOUT_ARM	This read-write bit selects the idle entry mode for the external DPLL output clock. 0: The clock supplied to the external DPLL output clock remains active when the MPU enters the idle mode (ARM_CK stopped). 1: The clock supplied to the external DPLL O/P clock is stopped in conjunction with the MPU clock when the MPU enters the idle mode (ARM_CK stopped).	R/W	0
11	RESERVED	See note.	R/W	0

Note: For reserved bits, reading gives undefined values. Writing to has no effect.

Table 64. MPU Idle Enable Control Register 1 (ARM_IDLECT1) (Continued)

Base Address = 0xFFFE CE00, Offset = 0x04				
Bit	**Name**	**Function**	**R/W**	**Reset**
10	WKUP_MODE	Controls how the MPU can exit the CHIP_IDLE state 0: After the interrupt has been asserted, the MPU idle mode is exited upon a low level at the external CHIP_nWKUP pin. Also, any of the wake-up conditions only wake up the OMAP out of CHIP_IDLE if CHIP_nWKUP is low. 1: Idle mode is exited upon an MPU interrupt (regardless the CHIP_nWKUP pin). Also, any wake-up condition wakes up the OMAP out of CHIP_IDLE regardless of the value on the CHIP_nWKUP pin.	R/W	1
9	IDLTIM_ARM	Selects the idle entry mode for internal MPU timer clock. 0: The clock supplied to the timers remains active when the MPU enters the idle mode. 1: The timer clock is stopped in conjunction with the MPU clock when the idle mode is entered.	R/W	0
8	IDLAPI_ARM	Selects the idle entry mode for MPUI clock. 0: The clock supplied to the MPUI is fully controlled by EN_APICK bit 1: The clock supplied to MPUI is on whenever it is required for any functionality; else it goes to IDLE This bit must be set to 1 active to go to chip idle mode. The EN_APICK bit must not be deactivated to go to chip idle as that may cause wake-up problems for certain sources.	R/W	0
7	IDLDPLL_ARM	Enables the DPLL macro to enter idle mode when DSP is set to global_idle mode, MPU is in idle mode, no active DMA transaction or TCLB_EN pin is asserted low, no TIPB posted write is queued, and the peripheral clocks are stopped. 0: DPLL remains active when the above conditions occur. 1: DPLL enters idle mode when above conditions are met.	R/W	0

Note: For reserved bits, reading gives undefined values. Writing to has no effect.

Table 64. MPU Idle Enable Control Register 1 (ARM_IDLECT1) (Continued)

Base Address = 0xFFFE CE00, Offset = 0x04				
Bit	**Name**	**Function**	**R/W**	**Reset**
6	IDLIF_ARM	Enables the TIPB bridge, the system DMA controller, and the TC to enter idle mode when the MPU processor executes the wait-for-interrupt instruction. 0: The clocks remain active when the MPU enters the idle mode. 1: The clocks are stopped in conjunction with the MPU clock when idle mode is entered and the DSP is also in idle.	R/W	0
5:3	RESERVED	See note.	R/W	000
2	IDLPER_ARM	Selects idle entry mode for external peripheral clock. (ARMPER_CK) 0: The peripheral clock remains active when the MPU and TC enter the idle mode. 1: The peripheral clock is stopped in conjunction with the MPU and TC clocks when the idle mode is entered. ARMPER_CK is no longer dependent only on ARM_IDLE conditions. As long as TC or MPU is active, ARMPER_CK clock is on. When both are inactive, ARMPER_CK is shutoff based on ARMPER IDLE/ACK.	R/W	0
1	IDLXORP_ARM	Selects idle entry mode for external reference peripheral clock ARMXOR_CK. 0: The external peripheral clock ARMXOR_CK remains active when the MPU enters the idle mode. 1: The external peripheral clock is stopped in conjunction with the MPU clock when the idle mode is entered.	R/W	0
0	IDLWDT_ARM	Selects the idle entry mode for internal timer/watchdog connected to MPU TIPB. When the timer/watchdog is configured as watchdog timer, the clock is never shutdown regardless of the IDLWDT_ARM bit. 0: The clock supplied to the timer/watchdog remains active when the MPU enters idle mode. 1: The timer/watchdog clock is stopped in conjunction with the MPU clock when the idle mode is entered.	R/W	0

Note: For reserved bits, reading gives undefined values. Writing to has no effect.

Table 65. MPU Idle Enable Control Register 2 (ARM_IDLECT2)

Base Address = 0xFFFE CE00, Offset = 0x08				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:12	RESERVED	See note.	R/W	0000
11	EN_CKOUT_ARM	This read-write bit enables the free running clock from DPLL1 output 0: The clock generated from DPLL1 output is stopped. This bit must be set to logic 1 to resume clock activity 1: The clock generated from DPLL1 output is active	R/W	0
10:9	RESERVED	See note.	R/W	00
8	DMACK_REQ	Disables the permanently-supplied-clock to the system DMA controller to function on a clock request basis. 0: The DMA clock is shutdown when the idle mode is entered if IDLIF_ARM bit of ARM_IDLECTL1 is set. 1: The DMA clock is stopped by default and is reactivated upon DMA request only.	R/W	1
7	EN_TIMCK	Enables the MPU internal timer clock connected to the MPU TIPB. 0: The MPU timer clock is stopped. 1: The MPU timer clock is active and can be stopped depending of the IDLTIM_ARM bit of ARM_IDLECTL1.	R/W	0
6	EN_APICK	Enables the clock of the MPUI. 0: The MPUI clock is stopped. This bit must be set to logic 1 to enable clock activity 1: The MPUI clock is active. The clock ON/OFF is now controlled as per IDLAPI_ARM bit.	R/W	0
5:4	RESERVED	Reading these bits gives undefined values. Writing to them has no effect.	R/W	00
3	EN_LCDCK	Enables the clock of the LCD controller connected to MPU TIPB. 0: The LCD clock is stopped. 1: The LCD clock is active.	R/W	0

Note: For reserved bits, reading gives undefined values. Writing to has no effect.

Table 65. MPU Idle Enable Control Register 2 (ARM_IDLECT2) (Continued)

Base Address = 0xFFFE CE00, Offset = 0x08				
Bit	**Name**	**Function**	**R/W**	**Reset**
2	EN_PERCK	Enables the external peripheral clock. 0: The external peripheral clock ARMPER_CK is stopped. 1: The external peripheral clock ARMPER_CK is active and can be stopped depending on the IDLLPER_ARM bit.	R/W	1
1	EN_XORPCK	Enables the clock of the OS timer connected to MPU TIPB and the external reference peripheral clock. 0: The OS timer clock and the external peripheral clock are stopped. 1: The OS timer clock and the external peripheral clock are active and can be stopped depending on the IDLXORP_ARM bit of ARM_IDLECTL1.	R/W	0
0	EN_WDTCK	Enables the clock of the timer/watchdog connected to MPU TIPB. (When the timer/watchdog is configured as watchdog timer, the clock is never shutdown regardless the value of IDLWDT_ARM and EN_WDTCK). 0: The timer/watchdog clock is stopped. 1: The clock supplied to timer/watchdog clock is active and can be stopped depending on the IDLWDT_ARM bit of ARM_IDLECTL1.	R/W	0

Note: For reserved bits, reading gives undefined values. Writing to has no effect.

Table 66. MPU Restore Power Delay Register (ARM_EWUPCT)

Base Address = 0xFFFE CE00, Offset = 0x0C				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:6	RESERVED	Reading these bits gives undefined values. Writing to them has no effect.	R/W	0x000
5	REPWR_EN	Enables the external power control feature. 0: The $\overline{\text{FLASH.RP}}$ pin is set to logic low when TC is in idle mode. 1: The $\overline{\text{FLASH.RP}}$ pin stays high when the TC idle mode is entered.	R/W	1
4:0	EXTPWR	Define the delay from $\overline{\text{FLASH.RP}}$ pin going high to the clocks restarting. Reference clock is the EMIFS CK_REF.	R/W	11111

Table 67. Master Software Reset Register (ARM_RSTCT1)

Base Address = 0xFFFE CE00, Offset = 0x10				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:4	RESERVED	Reading these bits gives undefined values. Writing to them has no effect.	R/W	0x000
3	SW_RST	Global system reset. Resets both the DSP and the MPU and peripherals. This bit is always read 0. 0: The DSP, the MPU, and the peripheral clock domains are enabled. 1: Resets the OMAP 3.2 hardware engine. Once set to logic 1 by the MPU processor, this bit returns to logic 0 on the next cycles.	R/W	0
2	DSP_RST	Resets the priority registers (TIPB module), the EMIF configuration registers, and the MPUI control logic in the DSP. This bit is set by the external reset pins and is released by writing a logic 1. 0: The priority registers, the EMIF configuration registers, and the MPUI are reset. 1: The priority registers and the EMIF configuration registers can be programmed.	R/W	0

Note: Writing the DSP_EN bit to 0 and ARM_RST bit to 1 together initiates a global software reset.

Table 67. Master Software Reset Register (ARM_RSTCT1) (Continued)

Base Address = 0xFFFE CE00, Offset = 0x10				
Bit	**Name**	**Function**	**R/W**	**Reset**
1	DSP_EN	Resets the DSP. 0: Resets the DSP, excluding the configuration setting. The reset state is maintained as long as this bit is asserted low. 1: The DSP is enabled. After a global reset, this bit must be set to 1 in order to enable the DSP megacell.	R/W	0
0	ARM_RST	Resets the MPU. This bit is always read 0 0: The MPU clock domain is enabled. 1: Reset the MPU. Once set to 1 by the MPU, this bit returns to 0 on the next cycles	R/W	0

Note: Writing the DSP_EN bit to 0 and ARM_RST bit to 1 together initiates a global software reset.

Table 68. Peripherals Reset Register (ARM_RSTCT2)

Base Address = 0xFFFE CE00, Offset = 0x14				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:1	RESERVED	Reading these bits gives undefined values. Writing to them has no effect.	R/W	0x0000
0	PER_EN	MPU Peripheral reset. Resets and/or enables the external peripherals connected to MPU TIPB (controls 3.2 ARMPER_nRST). 0: Resets MPU peripherals 1: Enables MPU peripherals	R/W	0

Table 69. MPU Clock Reset Status Register (ARM_SYSST)

Base Address = 0xFFFE CE00, Offset = 0x18				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:14	RESERVED	Reading these bits gives undefined values. Writing to them has no effect.	R/W	00
13:11	CLOCK_SELECT	Reading these bits indicates the clock_select pins and indicates the current clocking mode selection. Writing to these bits enables switching the OMAP3.2 clocking scheme. These bits are at logic 0 after reset: 000: Fully synchronous 001: Reserved 010: Synchronous scalable 011: Reserved 100: Reserved 101: Bypass 110: Mix mode #3, MPU synchronous to TC, DSP MMU synchronous scalar to MPU and TC 111: Mix mode #4, DSP MMU synchronous to TC, MPU synchronous scalar to DSP MMU and TC	R/W	000
10:7	RESERVED	These read only bits are undefined.	R	0
6	IDLE_DSP	Indicates the DSP state. 0: The DSP is active. 1: The DSP is in global-idle state.	R	0
5	POR	Indicates (in conjunction with EXT_RST bit) whether or not a power-on reset (cold start) has occurred. Writing it to logic 0 clears this bit. This bit cannot be written to logic 1 from the TIPB interface. 0: No power-on-reset has been detected. 1: A power-on-reset has occurred.	R/C	1
4	EXT_RST	Indicates that external reset has been asserted. Writing it to logic 0 clears this bit. This bit cannot be written to logic 1 from the TIPB interface. 0: No external reset has been detected. 1: An external reset has occurred.	R/C	1

Table 69. MPU Clock Reset Status Register (ARM_SYSST) (Continued)

Base Address = 0xFFFE CE00, Offset = 0x18				
Bit	**Name**	**Function**	**R/W**	**Reset**
3	ARM_MCRST	Indicates whether or not an MPU reset has occurred. This bit is cleared to 0 upon an external reset pulse asserting at the CHIP_nRESET pin, or by writing to it a logic 0. This bit cannot be written to logic 1 from the TIPB interface. 0: The MPU processor has not been reset. 1: The MPU processor has been reset.	R/C	1
2	ARM_WDRST	Indicates whether or not the reset has been asserted due to an MPU timer/watchdog underflow. This bit is cleared to 0 upon an external reset pulse asserting at the CHIP_nRESET pin, or by writing to it a logic 0. This bit cannot be written to logic 1 from the TIPB interface. 0: An MPU timer/watchdog underflow has not occurred. 1: An MPU timer/watchdog underflow has generated the reset.	R/C	0
1	GLOB_SWRST	Indicates whether or not the reset has been asserted due to global software reset (DSP_EN set to 0 and ARM_RST set to 1). This bit is cleared to 0 upon an external reset pulse asserting at the CHIP_nRESET pin, or by writing to it a logic 0. This bit cannot be written to logic 1 from the TIPB interface. 0: Global software reset has not been requested. 1: Global software reset has been requested.	R/C	0
0	DSP_WDRST	Indicates whether or not the reset has been asserted due to DSP timer/watchdog underflow. This bit cannot be written to logic 1 from the TIPB interface. 0: A DSP timer/watchdog underflow has not occurred. 1: A DSP timer/watchdog underflow has generated the reset.	R/C	0

Table 70. MPU Clock Out Definition Register (ARM_CKOUT1)

Base Address = 0xFFFE CE00, Offset = 0x1C				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:6	RESERVED	Reading these bits gives undefined values. Writing to them has no effect.	R/W	0x0000
5:4	TCLKOUT	The POCLKOUT3 pin functions are: 00: Reserved 01: POCLKOUT3 pin is an output and reflects CK_GEN3 clocking signal. 10: POCLKOUT3 pin is an output and reflects the TC_CK clock. 11: Reserved POCLKOUT3 is not a target clock; it is a free-running clock to select CK_GEN3 or TC_CK frequency.	R/W	01
3:2	DCLKOUT	The POCLKOUT2 pin functions are: 00: POCLKOUT2 pin is an output and reflects the DSPMMU_CK clock. 01: POCLKOUT2 pin is an output and reflects the CK_GEN2 clocking signal. 10: POCLKOUT2 pin is an output and reflects the DSP_CK clock. 11: POCLKOUT2 pin is an output and reflects low-frequency clock that supplies internal watchdog timers (CK_REF/14). POCLKOUT2 is not a target clock; it is a free-running clock to select DSPMMU_CK, CK_GEN2, or DSP_CK frequency.	R/W	01
1:0	ACLKOUT	The POCLKOUT1 pin functions are: 00: Reserved 01: POCLKOUT1 output pin reflects CK_GEN1 10: POCLKOUT1 output pin reflects ARM_CK 11: POCLKOUT1 output pin reflects the low-frequency clock that supplies the internal timers (CK_REF/14). POCLKOUT1 is not a target clock; it is a free-running clock to select CK_GEN1, ARM_CK, or CK_REF/14 clock frequency.	R/W	01

Table 71. MPU Reserved Register (ARM_CKOUT2)

Base Address = 0xFFFE CE00, Offset = 0x20				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:0	RESERVED	Reading these bits gives undefined values. Writing to them has no effect.	R/W	0x0000

Table 72. MPU Idle Enable Control Register 3 (ARM_IDLECT3)

Base Address = 0xFFFE CE00, Offset = 0x24				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:6	RESERVED	Reading these bits gives undefined values. Writing to them has no effect.	R/W	0x000
5	IDLTC2_ARM	Selects the idle entry mode for TC2 clock. 0: The TC2 clock remains active when the MPU enters the idle mode (ARM_CK stopped). 1: The TC2 clock is stopped in conjunction with the MPU clock when the idle mode is entered. Cutting off is based on IDLE/ACK protocol between CLKRST and peripherals outside OMAP.	R/W	0
4	EN_TC2_CK	Enables the TC2 clock. This is a generic clock supplied to peripherals outside OMAP boundary and is at the same frequency as the TC clock. 0: The TC2_CK clock is stopped. Ensure that all peripherals connected to TC2_CK are inactive before setting 0 on EN_TC2_CK. 1: The TC2_CK clock is active.	R/W	1
3	IDLTC1_ARM	Selects the idle entry mode for TC1 clock. 0: The TC1 clock remains active when the MPU enters the idle mode (ARM_CK stopped). 1: The TC1 clock is stopped in conjunction with the MPU clock when the idle mode is entered. Cutting off is based on IDLE/ACK protocol between CLKRST and peripherals outside OMAP.	R/W	0

Table 72. MPU Idle Enable Control Register 3 (ARM_IDLECT3) (Continued)

Base Address = 0xFFFE CE00, Offset = 0x24				
Bit	**Name**	**Function**	**R/W**	**Reset**
2	EN_TC1_CK	Enables the TC1 clock. This is a generic clock supplied to peripherals outside OMAP boundary and is at the same frequency as the TC clock. 0: The TC1_CK clock is stopped. Ensure that all peripherals connected to TC1_CK are inactive before setting 0 on EN_TC1_CK. 1: The TC1_CK clock is active.	R/W	1
1	IDLOCPI_ARM	Selects the idle entry mode for the L3 OCP initiator. 0: The L3 OCP-I clock remains active when the MPU enters the idle mode. 1: The L3 OCP-I clock is stopped in conjunction with the MPU clock when the idle mode is entered.	R/W	0
0	EN_OCPI_CK	Enables the L3 OCPI clock. 0: The L3 OCPI clock is stopped. 1: The L3 OCPI clock is active.	R/W	1

4.4.2 DSP Registers

These registers are accessible by the DSP or the MPU. The DSP control registers are 16-bit accessed. The offsets are given for byte addressing.

Table 73 lists the DSP registers. Table 74 through Table 82 provide register bit descriptions.

Table 73. DSP Registers

Base Address = 0xE100 8000 or 0x008000			
Name	**Description**	**R/W**	**Offset**
DSP_CKCTL	DSP clock control prescaler selection	R/W	0x00
DSP_IDLECT1	DSP idle enable control 1	R/W	0x04
DSP_IDLECT2	DSP idle enable control 2	R/W	0x08
DSP_EWUPCT	DSP reserved register 1	R/W	0x0C
DSP_RSTCT1	DSP reserved register 1	R/W	0x10
DSP_RSTCT2	DSP peripherals reset	R/W	0x14
DSP_SYSST	DSP clock reset status	R/W	0x18

Table 73. DSP Registers (Continued)

Base Address = 0xE100 8000 or 0x008000			
Name	**Description**	**R/W**	**Offset**
DSP_CKOUT1	DSP reserved register 3	R/W	0x1C
DSP_CKOUT2	DSP reserved register 4	R/W	0x20

Table 74. DSP Clock Control Prescaler Selection Register (DSP_CKCTL)

Base Address = 0xE100 8000 or 0x00 8000, Offset = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:9	RESERVED	Reading these bits gives undefined values. Writing to them has no effect.	R/W	0x00
8	TIMXO	Selects either a CK_GEN2 frequency clock or the input reference clock (CLK_REFIN) to supply timers. 0: The DSPTIM_CK clock frequency is the input reference clock. 1: The DSPTIM_CK clock frequency is issued from CK_GEN2 divided by 2.	R/W	1
7	RESERVED	This bit must be set to 1.	R/W	1
6:5	RESERVED	These bits must be set to 00.	R/W	00
4	RESERVED	This bit must be set to 1.	R/W	1
3:2	RESERVED	These bits must be set to 00.	R/W	00
1:0	DSP_PERDIV	Defines the prescaler value from CK_GEN2 to the DSP external peripheral clock. 00: CK_GEN2 01: CK_GEN2/2 10: CK_GEN2/4 11: CK_GEN2/8	R/W	00

Table 75. DSP Idle Enable Control Register 1 (DSP_IDLECT1)

Base Address = 0xE100 8000 or 0x008000, Offset = 0x04				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:9	RESERVED	Reading these bits gives undefined values. Writing to them has no effect.	R/W	0x00
8	IDLTIM_DSP	Selects the idle entry mode for the internal DSP timer clock. 0: The DSPTIM_CK clock remains active when DSP enters the idle mode. 1: The DSPTIM_CK clock is stopped in conjunction with DSP clock when the idle mode is set.	R/W	0
7	RESERVED	This bit must be set to 0.	R/W	0
6	WKUP_MODE	This bit has no effect in the OMAP 3.2 hardware engine.	R/W	1
5	IDLDPLL_DSP	This bit has no effect in the OMAP 3.2 hardware engine.	R/W	0
4	IDLIF_DSP	This bit has no effect in the OMAP 3.2 hardware engine	R/W	0
3	RESERVED	This bit must be set to 0.	R/W	0
2	IDLPER_DSP	Selects idle entry mode for external peripheral clock. 0: The DSPPER_CK clock remains active when DSP enters the idle mode. 1: The DSPPER_CK is stopped in conjunction with DSP clock when the idle mode is set.	R/W	0

Table 75. DSP Idle Enable Control Register 1 (DSP_IDLECT1) (Continued)

Base Address = 0xE100 8000 or 0x008000, Offset = 0x04				
Bit	**Name**	**Function**	**R/W**	**Reset**
1	IDLXORP_DSP	Selects idle entry mode for external reference peripheral clock. 0: The DSPXOR_CK clock remains active when DSP enter the idle mode. 1: The DSPXOR_CK clock is stopped in conjunction with DSP clock when the idle mode is set.	R/W	0
0	IDLWDT_DSP	Selects the idle entry mode for the internal timer/watchdog connected to DSP TIPB. 0: The clock supplied to timer/watchdog remains active when DSP enters the idle mode. 1: The timer/watchdog clock is stopped in conjunction with DSP clock when the idle mode is set. When the timer/watchdog is configured as watchdog timer, the clock is never shutdown regardless of the value of the IDLWDT_DSP bit.	R/W	0

Table 76. DSP Idle Enable Control Register 2 (DSP_IDLECT2)

Base Address = 0xE100 8000 or 0x008000, Offset = 0x08				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:6	RESERVED	Reading these bits gives undefined values. Writing to them has no effect.	R/W	0x00
5	EN_TIMCK	Enables the internal DSP timer clock (DSPTIM_CK). 0: DSPTIM_CK clock is stopped. 1: DSPTIM_CK clock is active and can be stopped depending on the IDLTIM_DSP bit of DSP_IDLECT1.	R/W	0
4	RESERVED	This bit must be set to 0.	R/W	0
3	RESERVED	This bit must be set to 0.	R/W	0
2	EN_PERCK	Enables external peripheral clock (DSPPER_CK). 0: DSPPER_CK clock is stopped. 1: DSPPER_CK clock is active and can be stopped depending on the IDLPER_DSP bit of DSP_IDLECT1.	R/W	0

Table 76. DSP Idle Enable Control Register 2 (DSP_IDLECT2) (Continued)

Base Address = 0xE100 8000 or 0x008000, Offset = 0x08				
Bit	**Name**	**Function**	**R/W**	**Reset**
1	EN_XORPCK	Enables the external reference clock (DSPXOR_CK). 0: DSPXOR_CK clock is stopped. 1: DSPXOR_CK clock is active and can be stopped depending on the IDLXORP_DSP bit of DSP_IDLECT1.	R/W	0
0	EN_WDTCK	Enables the internal timer/watchdog clock (DSPWDG_CK). 0: DSPWDG_CK clock is stopped. 1: DSPWDG_CK clock is active and can be stopped depending on the IDLWDT_DSP bit of DSP_IDLECT1.	R/W	0

Table 77. DSP Reserved Register 1 (DSP_EWUPCT)

Base Address = 0xE100 8000 or 0x008000, Offset = 0x0C				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	RESERVED	Reading these bits gives undefined values. Writing to them has no effect.	R/W	0x0

Table 78. DSP Reserved Register 2 (DSP_RSTCT1)

Base Address = 0xE100 8000 or 0x008000, Offset = 0x10				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	RESERVED	Reading these bits gives undefined values. Writing to them has no effect.	R/W	0x0

Table 79. DSP Peripherals Reset Register (DSP_RSTCT2)

Base Address = 0xE100 8000 or 0x008000, Offset = 0x14				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:2	RESERVED	Reading these bits gives undefined values. Writing to them has no effect.	R/W	0x0000
1	WD_PER_EN	Controls the WD_DSPPER_nRST output, which can be used to reset the external peripherals connected to DSP. This WD_DSPPER_nRST pin is also reset by an event on the DSP watchdog timer. 0: Sets WD_DSPPER_nRST pin to a low-level output voltage. 1: Sets WD_DSPPER_nRST pin to a high-level output voltage.	R/W	0
0	PER_EN	Controls the DSPPER_nRST output, which can be used to reset the external peripherals connected to DSP TIPB. 0: Sets DSPPER_nRST pin to a low-level output voltage. 1: Sets DSPPER_nRST pin to a high-level output voltage.	R/W	0

Table 80. DSP Clock Reset Status Register (DSP_SYSST)

Base Address = 0xE100 8000 or 0x008000, Offset = 0x18				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:14	RESERVED	Reading these bits gives undefined values. Writing to them has no effect.	R/W	00
13:11	CLOCK_SELECT	These read-only bits reflect the CLOCK_SELECT pins and indicate the current clocking mode selection. 000: Fully synchronous 001: Reserved 010: Synchronous scalable. 011: Reserved 100: Reserved 101: Bypass 110: Mix mode #3, MPU synchronous to TC, DSP MMU synchronous scalar to MPU and TC 111: Mix mode #4, DSP MMU synchronous to TC, MPU synchronous scalar to DSP MMU and TC	R	000
10:7	RESERVED	Reserved bits	R	0000
6	IDLE_ARM	Indicates the MPU state 0: The MPU is active. 1: The MPU is in idle state.	R	0
5	POR	Indicates (in conjunction with EXT_RST bit) whether or not a power-on reset (cold start) has occurred. Writing it to logic 0 clears this bit. This bit cannot be written to logic 1 from the TIPB interface. 0: No power-on reset has been detected. 1: A power-on reset has occurred.	R/C	1
4	EXT_RST	Indicates whether or not an external reset has been asserted. Writing it to logic 0 clears this bit. This bit cannot be written to logic 1 from the TIPB interface. 0: No external reset detected. 1: An external reset has been asserted.	R/C	1

Table 80. DSP Clock Reset Status Register (DSP_SYSST) (Continued)

Base Address = 0xE100 8000 or 0x008000, Offset = 0x18				
Bit	**Name**	**Function**	**R/W**	**Reset**
3	DSP_ARM_RST	Used by the DSP to hold the MPU in reset. This is for test and debug purposes only. Not for users. 0: The MPU is enabled. 1: Reset the MPU.	R/W	0
2	ARM_WDRST	Indicates whether or not the reset has been asserted due to a MPU timer/watchdog underflow. This bit cannot be written to logic 1 from the TIPB interface. 0: An MPU timer/watchdog underflow has not occurred. 1: An MPU timer/watchdog underflow has generated the reset.	R/C	0
1	GLOB_SWRST	Indicates whether or not the reset has been asserted due to global software reset. This bit cannot be written to logic 1 from the TIPB interface. 0: A global software reset has not been requested. 1: A global software reset has been requested.	R/C	0
0	DSP_WDRST	Indicates whether or not the reset has been asserted due to DSP timer/watchdog underflow. This bit cannot be written to logic 1 from the TIPB interface. 0: A DSP timer/watchdog underflow has not occurred. 1: A DSP timer/watchdog underflow has generated the reset.	R/C	0

Table 81. DSP Reserved Register 3 (DSP_CKOUT1)

Base Address = 0xE100 8000 or 0x008000, Offset = 0x1C				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	RESERVED	Reading these bits gives undefined values. Writing to them has no effect.	R/W	0x0000

Table 82. DSP Reserved Register 4 (DSP_CKOUT2)

Base Address = 0xE100 8000 or 0x008000, Offset = 0x20				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	RESERVED	Reading these bits gives undefined values. Writing to them has no effect.	R/W	0x0000

4.4.3 DPLL Registers

Table 83 lists the 16-bit DPLL registers Table 84 and Table 85 describe the register bits.

Table 83. DPLL Registers

Base Address = 0xFFFE CF00			
Name	**Description**	**R/W**	**Offset**
DPLL1_CTL_REG	DPLL1 control	R/W	0x00
DPLL2_CTL_REG	DPLL2 control (all reserved)	R	0x100

Table 84. DPLL1 Control Register (DPLL1_CTL_REG)

Base Address = 0xFFFE CF00, Offset = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
15	LS_DISABLE	Controls the level shifter power-down pin 0: Level shifter is in *transparent* mode; all signals between the wrapper and the DPLL core are connected 1: Level shifter is in *isolated* mode; the wrapper and the DPLL core are disconnected, so the DPLL core power supply (VDD_DPLL) can be turned off. There is no leakage current between VDD and VDD_DPLL.	R/W	0
14	IAI	Initialize after idle. Value of this bit must not be changed. Must be set to 0.	R/W	0
13	IOB	Initialize on break. When high, DPLL switches to bypass mode and starts a new locking sequence, even if the DPLL core indicates that it has lost the lock. When low, DPLL continues to output the synthesized clock, even if the core indicates it has lost the lock but the BREAKLN is active low.	R/W	1
12	TEST†	Controls the test output clock on the DPLL_TCLKOUT pin as given below: 0: DPLL_TCLKOUT = DPLL1 output clock when in test mode. 1: DPLL_TCLKOUT = DPLL1 output clock divided by 32 when in test mode. X: DPLL_TCLKOUT = 0 when not in test mode.	R/W	0

† POCLKOUT[1, 2, 3] is a nongated output from clock domain[1, 2, 3]. Using the [A/D/T] CLKOUT bits, select between different clocks used in the respective domains.

Table 84. DPLL1 Control Register (DPLL1_CTL_REG) (Continued)

Base Address = 0xFFFE CF00, Offset = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
11:7	PLL_MULT	DPLL multiply value. The maximum clock out frequency is 31 * CK_REF.	R/W	00000
6:5	PLL_DIV	DPLL divide value. The minimum DPLL1 clock out frequency is CK_REF/4. 00: CLKOUTDPLL1 output clock = CK_REF 01: DPLL1 output clock CLKOUT = CK_REF/2 10: DPLL1 output clock CLKOUT = CK_REF/3 11: DPLL1 output clock CLKOUT = CK_REF/4	R/W	00
4	PLL_ENABLE	Requests the DPLL to enter the lock mode. DPLL enters the lock mode only after it has synthesized the desired frequency. 0: DPLL enters the bypass mode. 1: DPLL enters the lock mode.	R/W	0
3:2	BYPASS_DIV	Determines the clock out frequency when in bypass mode. 00: DPLL1 output clock CLKOUT = CK_REF 01: DPLL1 output clock CLKOUT = CK_REF/2 1X: DPLL1 output clock CLKOUT = CK_REF/4	R/W	00
1	BREAKLN	Indicates whether DPLL has broken lock for some unknown reason. 0: DPLL has broken lock for some unknown reason. 1: Lock condition is restored or a write to a control register occurs.	R	0
0	LOCK	Indicates if DPLL is in lock mode and the clock out has the desired synthesized frequency. 0: DPLL is in bypass mode. 1: DPLL is in lock mode.	R	0

† POCLKOUT[1, 2, 3] is a nongated output from clock domain[1, 2, 3]. Using the [A/D/T] CLKOUT bits, select between different clocks used in the respective domains.

Table 85. DPLL2 Control Register (DPLL2_CTL_REG)

Base Address = 0xFFFE D000, Offset = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset Value**
15:0	RESERVED	Reserved. Do not write to these bits.	R	0x00002000

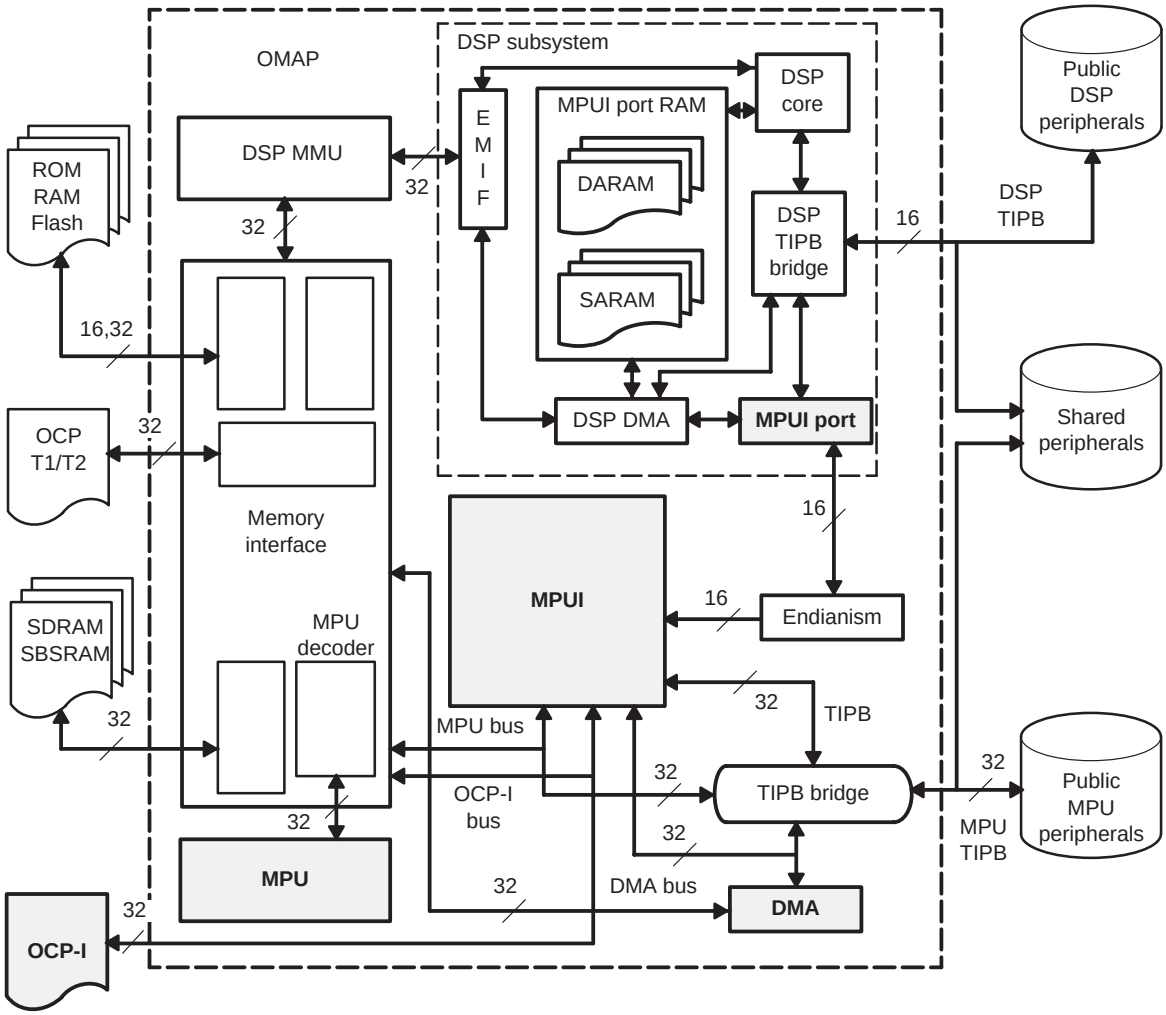
5 MPU and MPUI Port

The MPU, system DMA, and OCP initiator (OCP-I) can access the DSP memories and peripherals via two interfaces: the MPUI and the MPUI port. The MPUI and the MPUI port have distinct features and functions:

- The MPUI is a module in the MPU subsystem that connects to the MPUI port.
- The MPUI port is a module contained in the DSP subsystem.

Figure 8-1 shows the MPU- and DSP-relevant modules and connections. In this figurechapter, DMA represents the system DMA.

Figure 43. OMAP 3.2 MPUI and MPUI Port Environment



5.1 MPUI

The MPUI module connects the MPU, system DMA, and OCP-I to the MPUI port. The MPUI allows sharing of the DSP internal memories and peripherals with the MPU/system DMA/OCP-I.

The MPUI module supports two access modes: host-only mode (HOM) and shared access mode (SAM).

In HOM, the MPUI can access the SARAM and DSP peripherals. An MPUI port RAM configuration register can set the range of the SARAM (host-only RAM)

to which the MPUI has exclusive access. The DSP is denied access to the host-only RAM portion; however, both the MPUI and the DSP can access the other part of the SARAM (shared-access RAM). All access available to the MPUI in HOM remains available even if the DSP is in idle mode. The MPUI does not have access to the DARAM in HOM. The DSP TIPB peripherals are accessible only by MPUI in HOM.

In SAM, the entire MPUI port RAM and DSP peripherals are accessible by the MPUI and DSP. If both the DSP and the MPUI are accessing the same memory or peripheral at the same time, priority is given to the DSP. The access is synchronized to the internal DSP clock. Note that SARAM and the DSP TIPB peripherals can be accessed by the MPUI both in HOM and SAM. The DARAM and the EMIF, however, can be accessed by the MPUI only in SAM.

HOM is more efficient than the SAM because no synchronization is involved. However, the HOM depends on the host operating frequency, which is normally slower than the internal DSP clock. The system software can switch between HOM and SAM, or vice versa, if desired., and it is up to the software to manage the system resources.

Section 5.1.1 through Section 5.1.7 describe the MPUI functions.

5.1.1 Access Request

If the MPU, OCP-I, and system DMA request access to the DSP memory/peripherals at the same time, the MPUI gives priority to one of the three, based on the ACCESS_PRIORITY bits of (MPUI_CONTROL). The programmable priority scheme must be configured during the system boot process. When the MPUI initiates an MPUI port access, it must wait until the access is completed before starting a new one. Pipelining is not supported by the MPUI.

The MPUI supports 8-, 16-, or 32-bit access requests, even though the interface from the MPUI to the MPUI port is a 16-bit interface. The MPUI, on receiving the 32-bit access request from the MPU, OCP-I, or system DMA, initiates two 16-bit MPUI accesses, one after the other, one with address X and other with address X + 2. On a 32-bit read access, The MPUI packs the data from the two transactions and sends a signal to the MPU/OCP-I/system DMA once the second 16-bit read is completed.

5.1.2 Endian Conversion

Because the MPUI uses little endian ordering and the MPUI port uses big endian ordering, there is an endianism conversion block between the MPUI and the MPUI port. This endianism block converts data between the little and

big endian formats. The swapping logic is controlled by software to have maximum flexibility to handle different types of data, based on the BYTE_SWAP_CTL and WORD_SWAP_CTL bits of (MPUI_CONTROL) from the MPUI.

The ability to turn the endianism on or off makes it more convenient for peripheral (control) register read/writes. The DSP memory read/write transactions require conversion of the data, but it is more convenient for the MPU, OCP-I, and system DMA controller to read and write to registers without conversion.

The endianism conversion logic only performs byte (8-bit) or 2-byte (16-bit) swap from big endian to little endian and vice versa without address manipulation. The swapping mode can be programmed differently according to information type, information source, and information direction, such as 32-bit or 16-bit data, DSP access, or DSP DMA access and read or write.

Since the MPUI port does not allow accessing with 32-bit data, the MPUI port logic converts two 16-bit data accesses sequentially when the MPUI accesses it with 32-bit data. The 2-byte swap is performed during sequential access conversion.

Table 86 and Table 87 show examples of endian conversion.

Table 86. Data Swap for 32-Bit Access for MPUI Port

32-Bit Accesses	BYTE_SWAP_CTL	WORD_SWAP_CTL
INPUT DATA	AA BB CC DD	AA BB CC DD
OUTPUT DATA	DD CC BB AA	CC DD AA BB

Table 87. Data Swap for 16-Bit Access for MPUI Port

16-Bit Accesses	BYTE_SWAP_CTL	WORD_SWAP_CTL
INPUT DATA	00 11	00 11
OUTPUT DATA	11 00	00 11

5.1.3 MPUI Strobe and Access Factor

The MPUI output strobe is an internal signal used to enable bus transactions between the MPUI and the MPUI port. The strobe active (low) pulse defines the beginning and end of a transaction. Because the MPUI may be required to communicate (through the MPUI port) with peripherals of varying speeds,

a means to adjust the MPUI output strobe timing is available (in between the MPUI and the MPUI port. The strobe active (low) pulse defines the beginning and end of a transaction. Because the MPUI may be required to communicate (through the MPUI port) with peripherals of varying speeds, the beginning and end of a transaction. Because the MPUI may be required to communicate (through the MPUI port) with peripherals of varying speeds, a means to adjust the MPUI output strobe timing is available (in HOM).

To allow slow peripherals to answer, it is possible to stretch an access over 2 * *n* MPUI clock cycles using the ACCESS_FACTOR bits in (MPUI_CONTROL). The peripheral then has *n* clock cycles to answer (n cycles the strobe is high; n cycles the strobe is low). Note that the MPUI clock referenced here is the input clock reference to the MPUI module generated from the clock and reset management module. The clock rate for the MPUI is fixed to the same value as that of the OMAP 3.2 traffic controller module. For more detail, see Section 4, *Clock Generation and Reset Management.*

5.1.4 MPUI Port RAM Access

In HOM, only the MPU, system DMA or OCP-I can access the SARAM through the MPUI. Although the entire SARAM is accessible, the MPU must first set the accessible size of the SARAM. The API_SIZE bit field in the register (DSP_MPUI_CONFIG) sets the size using the formula, *(integer value of API_SIZE * 8K bytes*), starting from the first SARAM block. For detail on available SARAM memory space and SARAM start addresses, see the memory map in chapter 10. For detail on the translation of DSP internal (logical) addresses into OMAP (physical) addresses, see *OMAP5912 Multimedia Processor DSP Sybsystem Reference Guide* (literature number SPRU750).

The host can not access the SARAM before releasing the MPU reset. After releasing the MPU reset and before releasing the DSP reset, the DSP is in HOM and all the SARAM is accessible only by the host as the default API_SIZE value is 0xFFFF. Then the (DSP_MPUI_CONFIG) can be programmed to give the host exclusive access to a portion or to all the SARAM. After the DSP reset is released, the DSP is automatically changed to SAM; consequently, whatever the value of the (DSP_MPUI_CONFIG) , all the SARAM is shared between the DSP and the host. DSP_MPUI_CONFIG must be set when the ARM_RSTCT1.DSP_RST register bit is 0, and when MPU reset is deasserted. If DSP_MPUI_CONFIG changes while the ARM_RSTCT1.DSP_RST bit is 1, then ARM_RSTCT1.DSP_RST must be cleared to 0 again before the new DSP_MPUI_CONFIG value can take effect.”

In HOM, the SARAM memory requests are completely asynchronous relative to the DSP clock. Therefore, memory accesses can be performed without resynchronization, allowing faster communication between MPU/system DMA/OCP-I and SARAM. Any access to DARAM or EMIF causes a bus error.

In SAM, both DSP and MPU/system DMA/OCP-I can access the entire SARAM, DARAM, and EMIF (if API_SIZE in DSP_MPUI_CONFIG register equals 0xFFFF, the DSP can not access the SARAM). The DARAM is located at the byte address range from 0x000000 to 0x00FFFF in DSP memory space. In this mode, the asynchronous host accesses from the MPUI are resynchronized on the DSP clock internally in the MPUI port logic. In case of conflict between MPUI and DSP accesses (MPUI and DSP attempt to access same memory block), the DSP has the priority and the MPUI access is not acknowledged and is delayed by one or several cycles. Nevertheless, if an MPUI cycle begins before a DSP request, the cycle is finished before recognizing the DSP.

5.1.5 Peripheral Access

In HOM, only the MPU/system DMA/OCP-I can access the DSP shared peripherals. To determine the addresses of the peripherals. Peripherals requests are completely asynchronous relative to DSP clocks. Therefore, peripherals accesses can be performed without any wait states, allowing faster communication between the MPU and peripherals. In the HOM, the MPUI port is a simple bridge between MPU and DSP TIPB bridge for the address, data, and control signals. The ACCESS_FACTOR bits of (MPUI_CONTROL) control the access clock that synchronizes the transfer between the MPUI and the MPUI port.

In SAM, both the DSP and the MPU/system DMA/OCP-I can access the DSP peripherals. However, MPU/system DMA/OCP-I can access only the DSP shared peripherals, and any MPUI access to DSP private peripherals causes a time-out error.

In SAM, the asynchronous host accesses from the MPU/system DMA/OCP-I are resynchronized internally in the DSP logic. In case of conflict between MPUI and DSP accesses (MPUI and DSP attempt to access same peripheral), the DSP has the priority and the MPUI waits one or several cycles. Nevertheless, if a MPUI transfer begins before a DSP request, the access is completed before recognizing the DSP.

5.1.6 MPUI and DSP TIPB Bridge Time-Out

When operating in HOM, an MPUI access time-out limits the maximum time a peripheral can stall the processor. When starting an access on the DSP

TIPB, the time-out counter is loaded with the value programmed in the TIMEOUT bits of (MPUI_CONTROL). If the current cycle is not finished when the counter reaches 0, the MPUI terminates the access and an abort exception is generated to the MPU/system DMA/OCP-I. The maximum value for time-out is 256 MPUI clock cycles. Therefore, if the MPUI clock is 50 MHz (equal to the traffic controller clock frequency setting), maximum timeout is 256 × 20 ns, or 5.12 µs. The time-out can be disabled with the value of 0 for debug and test purposes.

In SAM, MPUI time-out is disabled automatically to avoid time-out conflict with DSP TIPB bridge time-out. In SAM, the DSP TIPB bridge can time-out a peripheral access if a predetermined period has elapsed and no response has been received from the peripheral. The DSP TIPB bridge generates a time-out error to MPUI through the MPUI port.

5.1.7 Debug

Three registers are provided for debug capability: (DEBUG_ADDRESS), (DEBUG_DATA), and (DEBUG_FLAG).

- (DEBUG_ADDRESS) indicates the location of a malfunction. If an access is aborted or has a size mismatch, the address of the access is saved in (DEBUG_ADDRESS).
- (DEBUG_DATA) stores the data of the malfunction. If a read access has a size mismatch, the read value is saved into (DEBUG_DATA). However, the value in this register is irrelevant when the read access is aborted. When a write access is aborted or has a size mismatch, the write value is saved in (DEBUG_DATA).
- The cause of a malfunction is always reported to (DEBUG_FLAG). This register also shows which host (MPU/system DMA/or OCP-I) is responsible for the abort.

5.2 MPUI Port

The MPUI port is an interface between DSP resources and the MPUI. The MPUI port accesses the MPUI port RAM via the DSP DMA, and it accesses the DSP shared peripherals via the DSP TIPB bridge.

Section 5.2.1 through Section 5.2.8 describe the MPUI port functions.

5.2.1 Access Modes

The MPUI port offers HOM and SAM based on the DSP resource. The modes are as follows:

HOM_M (host only for SARAM access)

HOM_R (host only for DSP TIPB access)

SAM_M (shared access for SARAM, DARAM, EMIF access)

SAM_R (shared access for DSP TIPB access)

Only the DSP can select HOM or SAM. This is controlled by the SMOD bits of (APIRS). Note that the MPU cannot write these bits, but it can read these bits.

Since SMOD has two bits, the following four mode combinations are available:

- SAM for DSP memory and TIPB. The host and the DSP can access the SARAM, DARAM, EMIF, and DSP TIPB peripherals.
- SAM for DSP memory and HOM for DSP TIPB. Only by the host can access the DSP TIPB peripherals via the DSP TIPB bus. The DSP is denied access to all the peripherals. SARAM, DARAM, and EMIF are accessible by the host and the DSP.
- HOM for DSP memory and SAM for DSP TIPB. The host can configure the SARAM for its exclusive access. The DSP can only access the nonexclusive part of SARAM. The DSP TIPB peripherals are accessible by the host and the DSP.
- HOM for DSP memory and TIPB. The host can configure the SARAM for its exclusive access. The DSP can only access the nonexclusive part of SARAM. Only the host can access the DSP TIPB peripherals.

When the DSP reset is activated, HOM_M and HOM_R are automatically selected. Thus, only the host can access the SARAM and DSP TIPB peripherals at DSP reset. This allows programs or data to be downloaded to SARAM to configure peripherals even during reset. When the DSP reset is released, SAM_M and SAM_R are automatically selected. Thus, the host and the DSP can access the SARAM, DARAM, EMIF, and DSP TIPB peripherals.

5.2.2 Memory Accesses in HOM

In HOM_M mode, only the MPU/system DMA/OCP-I can exclusively access the specified portion of SARAM with its range controlled by the API_SIZE bits of (DSP_MPUI_CONFIG). Memory requests are completely asynchronous relative to the DSP clock. Therefore, memory accesses can be performed without any wait states, allowing faster communication between MPUI and SARAM. In HOM, the MPUI port is a simple bridge between MPU/system DMA/OCP-I and SARAM memory bank for the address, data, and control signals.

5.2.3 Memory Accesses in SAM

When the MPUI port is in SAM_M mode, both the DSP and the MPU/system DMA/OCP-I can access the SARAM, DARAM, and EMIF. In this mode, the asynchronous host accesses from the MPU/system DMA/OCP-I are resynchronized on the DSP clock internally in the MPUI port logic. In case of conflict between the MPU/system DMA/OCP-I and the DSP accesses (i.e., the MPU/system DMA/OCP-I and the DSP attempt to access same memory block), the DSP has the priority and the MPU/system DMA/OCP-I access is not acknowledged and is delayed by one or more cycles.

However, if an MPU/system DMA/OCP-I transfer has started before the DSP request is received, the MPU/system DMA/OCP-I transfer is completed before the DSP access is initiated.

5.2.4 Peripheral Accesses in HOM

In HOM_R mode, only the MPU/system DMA/OCP-I can access the DSP shared peripherals and peripheral requests are completely asynchronous to DSP clock. Therefore, peripheral accesses can be performed without any resynchronization, allowing for faster communication between the MPU/system DMA/OCP-I and the peripherals. In this mode, the MPUI port is a simple bridge between the MPU/system DMA/OCP-I and the DSP TIPB bridge for the address, data, and control signals.

- Any DSP access to the shared peripherals generates a bus error from the DSP TIPB bridge.
- An MPUI access to a DSP private peripheral causes a time-out error.

5.2.5 Peripheral Accesses in SAM

When the MPUI port is in SAM_R mode, both the DSP and the MPU/system DMA/OCP-I can access the peripherals. In this mode, the asynchronous host accesses from the MPU/system DMA/OCP-I are resynchronized internally in the MPUI port logic. In case of conflict between the MPU/system DMA/OCP-I and the DSP accesses (i.e. the MPU/system DMA/OCP-I and the DSP attempt to access the same peripheral at the same time), the DSP has the priority and the MPU/system DMA/OCP-I waits one or more cycles.

However, if an MPU/system DMA/OCP-I transfer has begun before a DSP request is received, the MPU/system DMA/OCP-I transfer is completed before the DSP access is initiated. The goal is to ensure a smooth transition, with no spurious reads or writes and no lost reads or writes.

5.2.6 Posted Write Mode

System performances can be enhanced by enabling the posted write mode when the MPUI port RAM and DSP TIPB are in SAM. The MPU/system

DMA/OCP-I write accesses are released before the write completion on the peripheral side. The MPU/system DMA/OCP-I is then free to carry on with the next access and the posted writes run as slots become available. This functionality can be turned off to aid in debugging, because when write posting is enabled it becomes difficult to attribute a bus error to a particular access. Only the MPU has full control over the posted write enable bits (ENA_WPOST_APIRAM and ENA_WPOST_TIPB in (APIRS)). At reset, posted write is inactive (0), but, when active, the posted write mode is selected for memory and peripheral accesses. Posted writes do not slow down the host; this is useful because in SAM, the DSP has the priority. When the transition from SAM to HOM is issued, it occurs after any outstanding posted writes are completed.

5.2.7 Bus Error

There are two sources of bus error for the MPUI port: the MPUI port itself and the DSP TIPB bridge. (DEBUG_FLAG) stores these bus errors.

The DSP TIPB bridge limits the time allowed for any peripheral bus transaction. The DSP TIPB bridge can terminate a peripheral bus transaction if a predetermined period of time has elapsed and no response has been received from the peripheral. In this case the DSP TIPB bridge issues an abort to the MPU via the MPUI port.

The DSP TIPB bridge can also issue an abort to MPU when the MPU/system DMA/OCP-I addresses a peripheral in the wrong mode (performing an 8-bit access to a 16-bit peripheral or vice versa).

The MPUI port issues an abort to the MPU upon detecting an incorrect bus transaction, such as two chip-selects being active at the same time. A second source of bus error in the MPUI port occurs when a write-only register, such as the interrupt register, is read. The same is true if a read-only register is written.

5.2.8 Interrupt

The DSP can send one interrupt to the MPU. This is done by setting the corresponding bit in (ST3) (bit 12 of status register 3 of TMS320C55x DSP—see TMS320C55x functional specification) to active-low.

Similarly, the MPU can interrupt the DSP, and eight DSP interrupt request lines are mapped. An interrupt is generated when the MPU writes a 0 to any of the MCU_IRQ[7:0] bits of (APIRI). These bits are written only by the MPU or OCP-I.

5.3 MPUI Port and MPUI Registers

This section provides information about the MPUI port and MPUI registers. Table 88 lists the 16-bit MPUI port registers. Table 89 and Table 90 describe the register bits.

Table 88. MPUI Port Registers

Base Address = 0xE102 0000			
Name	**Description**	**R/W**	**Offset**
APIRI	MPUI port interrupt	R/W	0x00
APIRS	MPUI port control/status	R/W	0x02

Table 89. MPUI Port Interrupt Register (APIRI)

Base Address = 0xE102 0000, Offset = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:8	Reserved			
7:0	MPU_IRQ [7:0]	Interrupt flag register for interrupts from the MPU/OCP-I to the DSP. Active low. Only the host can set these bits. A DSP interrupt is generated when the MPU writes a zero to the MPU_IRQ bits. MPU_IRQ is automatically reset by MPUI port internal logic.	W by MPU/system DMA/OCP-I No access by DSP	0xFF

Table 90. MPUI Port Control/Status Register (APIRS)

Base Address = 0xE102 0000, Offset = 0x02				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:4	Reserved			
3	ENA_WPOST_ APIRAM	Enables posted write for writes to the MPUI port RAM. Available in SAMs only. 0: Posted write disabled. 1: Posted write enabled.	R/W by MPU/system DMA/OCP-I No access by DSP	0

Table 90. MPUI Port Control/Status Register (APIRS) (Continued)

Base Address = 0xE102 0000, Offset = 0x02				
Bit	**Name**	**Function**	**R/W**	**Reset**
2:1	SMOD [1:0]	HOM or SAM setting for MPUI port RAM and DSP TIPB peripherals. 00: SAM for MPUI port RAM and DSP TIPB. DSP and MPU/system DMA/OCP-I can access MPUI port RAM and DSP TIPB. 01: HOM for DSP TIPB: DSP TIPB peripherals are accessible from MPU/system DMA/OCP-I only, SAM for MPUI port RAM. 10: HOM for MPUI port RAM: MPUI port RAM is accessible from MPU/system DMA/OCP-I only, SAM for DSP TIPB peripherals. 11: HOM for MPUI port RAM and DSP TIPB: MPUI port RAM and DSP TIPB are accessible from MPU only. A DSP write to host only resources is not performed, and any DSP read or write results in an abort.	R by MPU/system DMA/OCP-I R/W by DSP	11
0	ENA_WPOST_ TIPB	Enables posted write for writes to the DSP TIPB peripherals. Available in SAMs only. 0: Posted write disabled. 1: Posted write enabled.	R/W by MPU/system DMA/OCP-I No access by DSP	0

5.3.1 MPUI Registers

All MPUI registers are 32-bit and are accessible only by MPU/system DMA/OCP-I. Read access can be performed in the user mode. Table 91 lists the 32-bit MPUI registers. Table 92 through Table 101 provide register bit descriptions.

Table 91. MPUI Registers

Base Address = 0xFFFE C900			
Name	**Description**	**R/W**	**Offset**
MPUI_CONTROL	MPUI Control register	R/W	0x00
DEBUG_ADDRESS	Debug address register	R/W	0x04
DEBUG_DATA	Debug data register	R/W	0x08
DEBUG_FLAG	Debug flag register	R/W	0x0C

Table 91. MPUI Registers (Continued)

Base Address = 0xFFFE C900			
Name	**Description**	**R/W**	**Offset**
MPUI_STATUS	MPUI status register	R/W	0x10
DSP_STATUS	DSP status register	R/W	0x14
DSP_BOOT_CONFIG	DSP boot configuration register	R/W	0x18
DSP_MPUI_CONFIG	MPUI port RAM configuration register	R/W	0x1C
DSP_MISC	DSP miscellaneous	R/W	0x20
MPUI_ENHANCED_ CTRL	MPUI enhanced control register	R/W	0x24

Table 92. MPUI Control Register (MPUI_CONTROL)

Base Address = 0xFFFE C900, Offset = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:23	Reserved			
22:21	WORD_SWAP_CTL	Bits to control word (16-bit swap) between MPUI and MPUI port for a 32-bit access: 00: Word swap for all accesses (OMAP 3.0) 01: Word swap only for DSP TIPB peripheral and MPUI port control/ status register accesses 10: Word swap only for MPUI port RAM accesses 11: Turn off word swap for all accesses.	R/W	00
20:18	ACCESS_PRIORITY	MPUI access priority between MPU, OCP-I, and system DMA requests: (Note: The lower the number, the higher the priority) 000: MPU → 1 DMA → 2 OCP-I → 3 001: MPU → 1 DMA → 3 OCP-I → 2 010: MPU → 2 DMA → 1 OCP-I → 3 011: MPU → 2 DMA → 3 OCP-I → 1 1X0: MPU → 3 DMA → 1 OCP-I → 2 1X1: MPU → 3 DMA → 2 OCP-I → 1	R/W	000

Note: For 32-bit words, enabling byte swaps with BYTE_SWAP_CTL does the following: (1) performs a 16-bit word swap, (2) within those 16-bit words, a byte swap occurs.

Table 92. MPUI Control Register (MPUI_CONTROL) (Continued)

Base Address = 0xFFFE C900, Offset = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
17:16	BYTE_SWAP_CTL	Bits to control byte swap between MPUI and MPUI port (see note below): 00: Turn off byte swap for all accesses 01: Byte swap only for DSP TIPB peripheral and MPUI port control/ status register accesses 10: Byte swap for all accesses 11: Byte swap only for MPUI port RAM accesses	R/W	11
15:8	TIMEOUT	MPUI port peripherals access time-out. When starting a HOM access to a DSP TIPB peripheral, the time-out counter is loaded with this value. If the current access is not finished when the counter reaches 0, the cycle is aborted and abort indications are given to the MPU/OCP-I/system DMA. Maximum value for time-out is 255. Note: TIMEOUT setting in MPUI is applicable only for HOM peripheral accesses.	R/W	0xFF
7:4	ACCESS_FACTOR	Division factor of MPUI output strobe signal from MPUI. Allows access of slow peripherals by reducing the access frequency (extending active low pulse) of the internal signal used to enable bus transactions between MPUI and MPUI port. Access factor = 0: Equivalent to access factor = 1. Access factor ≠ 0: Number of MPUI clock periods that the MPUI output strobe remains active (low) waiting for MPUI port to return a ready-to-send-or- receive-data signal. Note: Access factor setting in MPUI is applicable only for HOM memory/peripheral accesses.	R/W	0x1

Note: For 32-bit words, enabling byte swaps with BYTE_SWAP_CTL does the following: (1) performs a 16-bit word swap, (2) within those 16-bit words, a byte swap occurs.

Table 92. MPUI Control Register (MPUI_CONTROL) (Continued)

Base Address = 0xFFFE C900, Offset = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
3	API_ERR_EN	Send MPUI port abort. 0: Mask the abort 1: An abort signal is forwarded by MPUI to the MPU for an aborted MPUI port transaction. API_ERR_EN only affects MPUI port aborts. It does not enable/disable aborts to MPU/OCP-I/system DMA for aborts generated directly by MPUI. For instance, a timeout-generated abort in the MPUI is not affected by API_ERR_EN setting.	R/W	1
2	Reserved			1
1	TIMEOUT_EN	Enable or disable TIMEOUT functionality as characterized by register bits (15:8). 0: Do not enable TIMEOUT functionality. 1: Enable TIMEOUT functionality.	R/W	1
0	Reserved		R/W	1

Note: For 32-bit words, enabling byte swaps with BYTE_SWAP_CTL does the following: (1) performs a 16-bit word swap, (2) within those 16-bit words, a byte swap occurs.

Table 93. Debug Address Register (DEBUG_ADDRESS)

Base Address = 0xFFFE C900, Offset = 0x04				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:24	Reserved			
23:0	ADR_SAV	Bits 23 down to 0 of address bus from MPU, OCP-I or system DMA interface are saved when abort or access size mismatch occurs.	R	0xFFFFFF

Table 94. Debug Data Register (DEBUG_DATA)

Base Address = 0xFFFE C900, Offset = 0x08				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:0	DATA_SAV	The value of MPU/OCP-I/system DMA data input bus is saved when a read access has a size mismatch, and the MPU data output bus is saved when a write access is aborted or has a size mismatch. If a read access is aborted, the value of this register is irrelevant.	R	0xFFFFFFFF

Table 95. Debug Flag Register (DEBUG_FLAG)

Base Address = 0xFFFE C900, Offset = 0x0C				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:15	Reserved		R/W	0x0000
14:13	HOST_ID	Indicates the host responsible for the abort 00: MPU 01: System DMA 10: OCP-I 11: Reserved	R	00
12:11	SMOD_SAV	Encoded access mode for MPUI port. 00: Shared access MPUI port RAM, shared access peripheral 01: Shared access MPUI port RAM, host only access peripheral 10: Host only access MPUI port RAM, shared access peripheral 11: Host only access MPUI port RAM, host only access peripheral	R	11
10:9	CS_SAV	Indicates the transaction chip-select on abort. 01: MPUI port memory chip-select 10: DSP TIPB peripheral or MPUI port control register (depending upon address value for the aborted transaction)	R	00

Table 95. Debug Flag Register (DEBUG_FLAG) (Continued)

Base Address = 0xFFFE C900, Offset = 0x0C				
Bit	**Name**	**Function**	**R/W**	**Reset**
8:6	BURST_SIZE_SAV	System bus data burst size indicated from MPU core on abort. Can be used in conjunction with BURST_SIZE_ERR flag bit. Valid OMAP 3.2 burst size values: 000: One data burst (single-access) 011: Four-data burst Since MPUI port supports single-access only, a 000 value is expected; otherwise, an abort is generated.	R	000
5	RNW_SAV	Indicates read or write transaction on the MPUI upon abort. 0: Write transaction 1: Read transaction	R	0
4	BYTE_SAV	Data width of the MPUI port access upon abort. Note that the MPUI handles 32-bit accesses from MPU/OCP-I/system DMA as successive 16-bit transactions to fit the 16-bit interface of the MPUI port. 0: 8-bit access 1: 16-bit access	R	0
3	BURST_SIZE_ERR	Flag set to 1 when the system bus data burst size indicated from the MPU core is not 000. Because MPUI port supports single-access only, a 000 value is always expected. The value of the burst indicated by MPU core is saved in bits (8:6). When read, this bit is reset to 0.	R	0
2	TIMEOUT_ERR	Flag set to 1 when MPUI access is aborted by internal timeout. When read, this bit is reset to 0.	R	0
1	API_ERR	Flag set to 1 when MPUI port aborts access. When read, this bit is reset to 0.	R	0
0	ABORT_FLAG	Flag set to 1 when MPUI access is aborted. When read, this bit is reset to 0.	R	0

Table 96. MPUI Status Register (MPUI_STATUS)

Base Address = 0xFFFE C900, Offset = 0x10				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:13	Reserved		R/W	0x0000
12:11	ACCESS_STATUS	Current access in progress is: 00: MPU access 01: System DMA access 10: OCP-I access 11: No access	R	11
10:3	TIMEOUT_VAL	Current value of timeout counter	R	0xFF
2	CS_EN	MPUI busy status. 0: MPUI is busy executing a transaction or host/shared mode switch. All new MPU/OCP-I/system DMA accesses must wait until CS_EN is 1. 1: A new access may be started in the MPUI.	R	1
1	ACCESS_DONE	MPUI access status (similar to CS_EN, without mode switch indication) 0: MPUI is accessing MPUI port. 1: No access in progress; last access is completed.	R	1
0	HOMNSAM_FLAG	Current access mode when ACCESS_DONE = 0, or last access mode when ACCESS_DONE = 1. 0: SAM 1: HOM	R	1

Table 97 describes (DSP_STATUS) bits. For debug purposes, this register stores the state of several signals internal to the DSP subsystem. Values are captured at each MPUI clock cycle.

Table 97. DSP Status Register (DSP_STATUS)

Base Address = 0xFFFE C900, Offset = 0x14				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:12	Reserved		R	0x0000
11	HRHOMNSAM	Reflects DSP HOM or SAM setting for DSP TIPB	R	0
10	HAHOMNSAM	Reflects DSP HOM or SAM setting for MPUI port RAM	R	0

Table 97. DSP Status Register (DSP_STATUS) (Continued)

Base Address = 0xFFFE C900, Offset = 0x14				
Bit	**Name**	**Function**	**R/W**	**Reset**
9	PENRESETDPLL	Reflects level of asynchronous reset in the DSP (controlled by emulation)	R	0
8	PEIDLE7	Idle peripherals flag. Reflects bit 7 of (ISTR) from the DSP.	R	0
7	PEIDLE6	Idle peripherals flag. Reflects bit 6 of (ISTR) from the DSP.	R	0
6	PEIDLEDPLL	Idle DPLL flag. Reflects bit 4 of (ISTR) from the DSP.	R	0
5	PEIDLEPERIPH	Idle peripherals flag. Reflects bit 3 of (ISTR) from the DSP.	R	0
4	CPUIACK	Reflects level of Interrupt acknowledged signal from the DSP.	R	0
3	CPUAVIS	Reflects bit 4 from (DSP_STATUS).	R	0
2	CPUXF	Reflects level of XF output from (DSP_STATUS).	R	0
1	RESET_MCU	Reflects level of the secondary DSP subsystem reset originating from the MPU (active low). Signal used to reset the DSP TIPB interrupt priority encoder, the EMIF configuration registers, and the MPUI port control logic.	R	0
0	RESET	Reflects level of DSP subsystem master reset (active low). Signal used to reset the entire DSP subsystem except for the DSP TIPB interrupt priority encoder, the EMIF configuration registers, and the MPUI port control logic.	R	0

Table 98. DSP Boot Configuration Register (DSP_BOOT_CONFIG)

Base Address = 0xFFFE C900, Offset = 0x18				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:16	Reserved		R/W	0x0000
15:10	BOOT_RHEA_PTR2	User-defined pointer that can be used for application-specific boot code location	R/W	000000

Table 98. DSP Boot Configuration Register (DSP_BOOT_CONFIG) (Continued)

Base Address = 0xFFFE C900, Offset = 0x18				
Bit	**Name**	**Function**	**R/W**	**Reset**
9:4	BOOT_RHEA_PTR1	User-defined pointer that can be used for application-specific boot code location	R/W	000000
3:0	DSP_BOOT_MODE	DSP boot mode inputs.	R/W	0000

Table 99. MPUI Port RAM Configuration Register (DSP_MPUI_CONFIG)

Base Address = 0xFFFE C900, Offset = 0x1C				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:16	Reserved		R/W	0x0000
15:0	API_SIZE	Grants the MPUI exclusive access to the specified portion of DSP SARAM in HOM. For details on SARAM configuration, see Table 102	R/W	0xFFFF

Table 100 describes the DSP miscellaneous register bits. For debug purposes, this register stores the state of the BION signal (internal to the DSP subsystem). State is captured at each MPUI clock cycle.

Table 100. DSP Miscellaneous (DSP_MISC)

Base Address = 0xFFFE C900, Offset = 0x20				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:9	Reserved		R/W	0x0000
8	CPUBION	Reflects level of BION signal to DSP subsystem	R	0
7:0	Reserved		R/W	0x00

Table 101. MPUI Enhanced Control Register (MPUI_ENHANCED_CTRL)

Base Address = 0xFFFE C900, Offset = 0x24				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:1	Reserved		R/W	0x0000
0	DPS_EN	When 1, dynamic power saving (DPS) mode is enabled. Otherwise, DPS is disabled. When DPS is enabled, the MPUI clock is turned off when there is no active request from the MCU/DMA/OCPI.	R/W	0

Table 102. DSP SARAM Configuration Map

API SIZE	SARAM 28	SARAM 24	SARAM 20	SARAM 16	SARAM 12	SARAM 8	SARAM 4	SARAM 0
0x0000−0x0001	0000	0000	0000	0000	0000	0000	0000	0000
0x0002−0x0003	0000	0000	0000	0000	0000	0000	0000	0001
0x0004−0x0005	0000	0000	0000	0000	0000	0000	0000	0011
0x0006−0x0007	0000	0000	0000	0000	0000	0000	0000	0111
0x0008−0x0009	0000	0000	0000	0000	0000	0000	0000	1111
0x000A−0x000B	0000	0000	0000	0000	0000	0000	0001	1111
0x000C−0x000D	0000	0000	0000	0000	0000	0000	0011	1111
0x000E−0x000F	0000	0000	0000	0000	0000	0000	0111	1111
0x0010−0x0011	0000	0000	0000	0000	0000	0000	1111	1111
0x0012−0x0013	0000	0000	0000	0000	0000	0001	1111	1111
0x0014−0x0015	0000	0000	0000	0000	0000	0011	1111	1111
0x0016−0x0017	0000	0000	0000	0000	0000	0111	1111	1111
0x0018−0x0019	0000	0000	0000	0000	0000	1111	1111	1111
0x001A−0x001B	0000	0000	0000	0000	0001	1111	1111	1111
0x001C−0x001D	0000	0000	0000	0000	0011	1111	1111	1111
0x001E−0x001F	0000	0000	0000	0000	0111	1111	1111	1111
0x0020−0x0021	0000	0000	0000	0000	1111	1111	1111	1111
0x0022−0x0023	0000	0000	0000	0001	1111	1111	1111	1111
0x0024−0x0025	0000	0000	0000	0011	1111	1111	1111	1111
0x0026−0x0027	0000	0000	0000	0111	1111	1111	1111	1111
0x0028−0x0029	0000	0000	0000	1111	1111	1111	1111	1111
0x002A−0x002B	0000	0000	0001	1111	1111	1111	1111	1111
0x002C−0x002D	0000	0000	0011	1111	1111	1111	1111	1111
0x002E−0x002F	0000	0000	0111	1111	1111	1111	1111	1111
0x0030−0x0031	0000	0000	1111	1111	1111	1111	1111	1111
0x0032−0x0033	0000	0001	1111	1111	1111	1111	1111	1111

Table 102. DSP SARAM Configuration Map (Continued)

API SIZE	SARAM 28	SARAM 24	SARAM 20	SARAM 16	SARAM 12	SARAM 8	SARAM 4	SARAM 0
0x0034−0x0035	0000	0011	1111	1111	1111	1111	1111	1111
0x0036−0x0037	0000	0111	1111	1111	1111	1111	1111	1111
0x0038−0x0039	0000	1111	1111	1111	1111	1111	1111	1111
0x003A−0x003B	0001	1111	1111	1111	1111	1111	1111	1111
0x003C−0x003D	0011	1111	1111	1111	1111	1111	1111	1111
0x003E−0x003F	0111	1111	1111	1111	1111	1111	1111	1111
0x0040−OTHERS	1111	1111	1111	1111	1111	1111	1111	1111

This table decodes API_SIZE value into the exclusively accessible portion of SARAM. The SARAM has 32 blocks (SARAM0 through SARAM31) on 2 KB boundaries.

The exclusively accessible memory (host-only RAM) is marked with a 1, and nonexclusively accessible memory (shared access RAM) is marked with a 0.

5.4 DSP Endianism Register

Note that this reigster is part of the Traffic Controller block.

Table 103. DSP Endianism Register (DSP_ENDIAN_CONV)

Address = 0xFFFE CC34				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:2	Reserved	Reserved	R	ND
1	SWAP	0: Byte swap 1: Word swap	R/W	0
0	EN	0: Disables DSP Endianism conversion 1: Enables DSP Endianism conversion	R/W	0

This register controls endianism for all DSP accesses through the traffic controller.

6 Mailboxes

6.1 Mailbox Registers

- The MPU/DMA/OCP-I uses the following registers to communicate with the DSP:

- MPU2DSP1A and MPU2DSP1B for mailbox1
- MPU2DSP2A and MPU2DSP2B for mailbox2

The sequence is as follows:

- An interrupt for the DSP is generated when the MPU/DMA/OCP-I writes to the second register; that is, MPU2DSP1B or MPU2DSP2B. Writing to the first registers MPU2DSP1A or MPU2DSP2A does not generate an interrupt.
- Once an interrupt has been set up, registers associated with this mailbox interrupt are locked until DSP clears the interrupt flag by reading the second register. For example, when MPU2DSP1B is written by the MPU, the MPU can write to neither MPU2DSP1A or MPU2DSP1B until the DSP clears the interrupt flag by reading MPU2DSP1B.

❑ The DSP uses the following registers to communicate with the MPU:

- DSP2MPU1A and DSP2MPU1B for mailbox1
- DSP2MPU2A and DSP2MPU2B for mailbox2

6.1.1 Mailbox Interrupts

Each mailbox interrupt is also associated to an interrupt flag bit (INT) which is set to 1 when the interrupt is pending. The four flag registers are as follows:

❑ MPU flag registers:

- MPU2DSP1_FLAG
- MPU2DSP2_FLAG

❑ DSP flag registers:

- DSP2MPU1_FLAG
- DSP2MPU2_FLAG

These flag registers reflect only the state of the mailbox; a write into them has no effect.

The interrupt flag register is set to 1 whenever a write is detected in the second register of a mailbox. This interrupt flag is cleared whenever a read is detected on the second data register, by the interrupted processor, of a mailbox interrupt (see Section 6.2, *Registers,* for details).

When the interrupt flag has been set up by a processor, this processor can write to neither of the registers associated with that mailbox interrupt until the other processor cleans the interrupt flag.

For example, if the MPU sets MPU2DSP1_FLAG by writing to MPU2DSP1B, the MPU has no write access to MPU2DSP1A or MPU2DSP1B until the DSP clears the interrupt flag by reading the MPU2DSP1B.

Because a processor cannot write to registers associated with an interrupt after the interrupt flag has been set up, both processors must write the data or command that needs to be communicated before setting the interrupt flag register. For example, MPU2DSP1A must be written to before the MPU2DSP1B, and so on.

If the interrupt flag is not cleared and a processor tries to write to the associated mailbox register, then the write is ignored. Also, the TIPB bridge generates an abort (time-out abort), and the program returns to the processor to continue its normal mode of execution. Even though the interrupt flag is not cleared, the processor can read its own registers.

6.1.2 Mailbox Software

You should program the mailbox registers so the following sequence of steps occurs in the specified sequence. These steps are for using the mailbox to communicate from the MPU to the DSP; the programmed sequence to communicate from the DSP to the MPU is the same.

1) The MPU enables the MPU2DSP1 mailbox interrupt and configures the mailbox interrupt as level-sensitive in the interrupt handlers.

2) The MPU writes to MPU2DSP1A and MPU2DSP1B.

3) Writing to MPU2DSP1B sets MPU2DSP1_FLAG.INT = 1 and generates an interrupt toward the DSP.

4) The MPU can now poll MPU2DSP1_FLAG. As long as this register is set (it has a value of 1), the DSP has not cleared the interrupt and the MPU cannot generate the next mailbox interrupt.

5) The DSP services this interrupt in an interrupt service routine. MPU2DSP1_FLAG is cleared when the DSP reads from MPU2DSP1B. Note that reading by the MPU does not clear MPU2DSP1_FLAG.

6) The MPU can generate the next interrupt by writing MPU-to-DSP mailbox registers, after MPU2DSP1_FLAG has been cleared.

Masking a mailbox interrupt does not disable that interrupt. If a mailbox interrupt is generated while it is masked, it remains pending until the mask is removed, when the interrupt becomes an active interrupt. To clear a masked interrupt, appropriate mailbox registers must be read.

6.2 Registers

All these registers are 16-bit aligned on a 32-bit address boundary. The DSP-to-MPU mailbox registers are written by the DSP and read by the MPU/DMA/OCP-I, while the MPU-to-DSP mailbox registers are written by the MPU/DMA/OCP-I and read by the DSP. Table 104 lists the mailbox registers. Table 105 through Table 116 provide register bit descriptions.

Table 104. Mailbox Registers

Base Address = 0xFFFC F000			
Name	**Description**	**R/W**	**Offset**
MPU2DSP1A	MPU to DSP mailbox 1A	R/W	0x00
MPU2DSP1B	MPU to DSP mailbox 1B	R/W	0x04
DSP2MPU1A	DSP to MPU mailbox 1A	R/W	0x08
DSP2MPU1B	DSP to MPU mailbox 1B	R/W	0x0C
DSP2MPU2A	DSP to MPU mailbox 2A	R/W	0x10
DSP2MPU2B	DSP to MPU mailbox 2B	R/W	0x14
MPU2DSP1_FLAG	MPU to DSP mailbox 1 flag	R	0x18
DSP2MPU1_FLAG	DSP to MPU mailbox 1 flag	R	0x1C
DSP2MPU2_FLAG	DSP to MPU mailbox 2 flag	R	0x20
MPU2DSP2A	MPU to DSP mailbox 2A	R/W	0x24
MPU2DSP2B	MPU to DSP mailbox 2B	R/W	0x28
MPU2DSP2_FLAG	MPU to DSP mailbox 2 flag register	R	0x2C

Table 105. MPU to DSP Mailbox 1A Register (MPU2DSP1A)

Base Address = 0xFFFC F000, Offset = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	MPU2DSP1A	This register stores the data to be shared for the MPU-to-DSP interrupt in mailbox 1.	R/W by MPU/DMA/OCP-I R by DSP	0x0000

Table 106. MPU to DSP Mailbox 1B Register (MPU2DSP1B)

Base Address = 0xFFFC F000, Offset = 0x04				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	MPU2DSP1B	This register stores the data to be shared for the MPU-to-DSP interrupt in mailbox 1. The MPU2DSP1 interrupt is generated to DSP when this register is written. When this register is read by DSP, (MPU2DSP1_FLAG) is reset.	R/W by MPU/DMA/OCP-I R by DSP	0x0000

Table 107. DSP to MPU Mailbox 1A Register (DSP2MPU1A)

Base Address = 0xFFFC F000, Offset = 0x08				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	DSP2MPU1A	This register stores the data to be shared for the DSP-to-MPU interrupt in mailbox 1.	R/W by DSP R by MPU/DMA/OCP-I	0x0000

Table 108. DSP to MPU Mailbox 1B Register (DSP2MPU1B)

Base Address = 0xFFFC F000, Offset = 0x0C				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	DSP2MPU1B	This register stores the data to be shared for the DSP-to-MPU interrupt in mailbox 1. The DSP2MPU1 interrupt is generated to MPU/DMA/OCP-I when this register is written. When this register is read by MPU, (DSP2MPU1_FLAG) is reset.	R/W by DSP R by MPU/DMA/OCP-I	0x0000

Table 109. DSP to MPU Mailbox 2A Register (DSP2MPU2A)

Base Address = 0xFFFC F000, Offset = 0x10				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	DSP2MPU2A	This register stores the data to be shared for the DSP-to-MPU interrupt in mailbox 2.	R/W by DSP R by MPU/DMA/OCP-I	0x0000

Table 110. DSP to MPU Mailbox 2B Register (DSP2MPU2B)

Base Address = 0xFFFC F000, Offset = 0x14				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	DSP2MPU2B	This register stores the data to be shared for the DSP-to-MPU interrupt in mailbox 2. The DSP2MPU2 interrupt is generated to MPU/DMA/OCP-I when this register is written. When this register is read by MPU, (DSP2MPU2_FLAG) is reset.	R/W by DSP R by MPU/DMA/OCP-I	0x0000

Table 111. MPU to DSP Mailbox 1 Flag Register (MPU2DSP1_FLAG)

Base Address = 0xFFFC F000, Offset = 0x18				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:1	Reserved		R/W	0x0000
0	INT	0: No interrupt pending 1: Interrupt generated	R by MPU/DMA/OCP-I No access by DSP	0

Table 112. DSP to MPU Mailbox 1 Flag Register (DSP2MPU1_FLAG)

Base Address = 0xFFFC F000, Offset = 0x1C				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:1	Reserved		R/W	0x0000
0	INT	0: No interrupt pending 1: Interrupt generated	R by DSP No access by MPU/DMA/OCP-I	0

Table 113. DSP to MPU Mailbox 2 Flag Register (DSP2MPU2_FLAG)

Base Address = 0xFFFC F000, Offset = 0x20				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:1	Reserved		R/W	0x0000
0	INT	0: No interrupt pending 1: Interrupt generated	R by DSP No access by MPU/DMA/OCP-I	0

Table 114. MPU to DSP Mailbox 2A Register (MPU2DSP2A)

Base Address = 0xFFFC F000, Offset = 0x24				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	MPU2DSP2A	This register stores the data to be shared for the MPU-to-DSP interrupt in mailbox 2.	R/W by MPU/DMA/OCP-I R by DSP	0x0000

Table 115. MPU to DSP Mailbox 2B Register (MPU2DSP2B)

Base Address = 0xFFFC F000, Offset = 0x28				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	MPU2DSP2B	This register stores the data to be shared for the MPU-to-DSP interrupt in mailbox 2. The MPU2DSP2 interrupt is generated to DSP when this register is written. When this register is read by DSP, (MPU2DSP2_FLAG) is reset.	R/W by MPU/DMA/OCP-I R by DSP	0x0000

Table 116. MPU to DSP Mailbox 2 Flag Register (MPU2DSP2_FLAG)

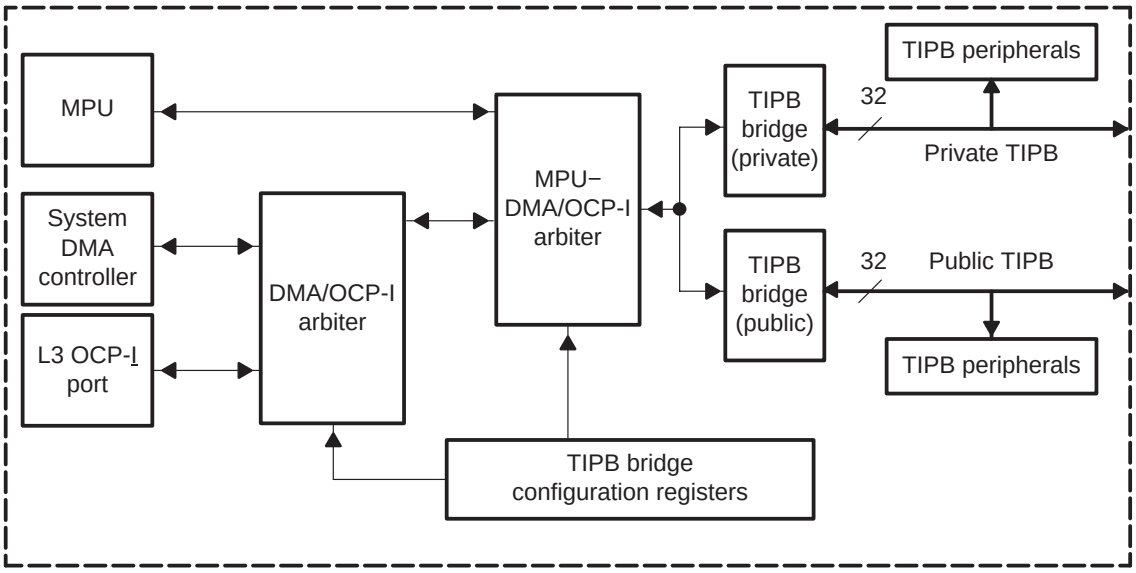
Base Address = 0xFFFC F000, Offset = 0x2C				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:1	Reserved	Reserved	R	0
0	INT	0: No Interrupt pending 1: MPU2DSP2 interrupt generated	R by MPU/DMA/OCP-I No access by DSP	0

7 TIPB Bridge

The TIPB bridge allows the TIPB and all peripherals connected to it to be shared with three hosts: the MPU, the OCP initiator (OCP-I), and the system DMA controller. The TIPB controls accesses to avoid conflicts between the hosts, and it allows the MPU to configure the protocol parameters of the TIPB.

The TIPB bridge module is shown in Figure 44.

Figure 44. OMAP 3.2 Platform TI Peripheral Bridge



7.1 Functionality

This section describes the functionality of the TI peripheral bus bridge.

7.1.1 Bus Allocation

The TIPB is shared between the MPU memory interface, the OCP initiator (OCP-I), and the DMA controller. Two levels of bus allocation are used to resolve conflicts and prioritize accesses among the three requestors.

The first level of the two-level arbitration process selects between the DMA and OCP-I for control of the TIPB. Fixed or round-robin priority schemes can be programmed in the FIXNROUND_PRIORITY bit field of the TIPB allocation control register, (RHEA_BUS_ALLOC). When fixed priority is selected, the EXTNINT_PRIORITY bit field determines whether DMA or OCP-I has fixed priority.

The second level of the two-level arbitration process selects between the MPU and the DMA/OCP-I. The value programmed in the RHEA_PRIORITY bits of (RHEA_BUS_ALLOC) defines the priority. If the value is 0, the MPU memory interface has priority over the DMA/OCP-I. If the value equals *n* (where *n* is from 1 to 7), the DMA/OCP-I has priority over the MPU and it can perform *n* accesses before giving the priority back to MPU memory interface.

7.1.2 Access Permissions

Access permissions to the MPU public and private TIPBs vary with initiator and target. The MPU has unconditional access permissions to the MPU public and private TIPBs.

The system DMA has unconditional access permission to the MPU public TIPB and conditional, software-controlled access permission to the MPU private TIPB. System DMA permission for the MPU private TIPB is programmed in (ACCESS_CNTL).

The OCP-I has conditional, software-controlled access to both the MPU public and private TIPBs. Refer to the *Traffic Controller* section for details on programming OCP-I access permissions.

7.1.3 TIPB Strobes and Access Factor

The TIPB strobes are an integral part of the TIPB bridge. The TIPB strobes are (active low) signals output from the TIPB bridge module that drive the peripheral interfaces of the MPU TIPB. The TIPB bridge uses two strobes (strobe 0 and strobe 1), and each strobe is fixed to control access to distinct ranges of the MPU TIPB peripherals address space. In addition, address spaces defined to strobe 0 and strobe 1 are further segmented into areas of MPU public TIPB and MPU private TIPB memory space. Refer to *MPU Peripherals Memory Mapping* (see Chapter 10), for details on strobe address space assignments.

The TIPB bridge may be required to communicate with peripherals of varying speeds. To allow slow peripherals to answer, it is possible to lengthen the strobe 0 and strobe 1 periods using ACCESS_FACTOR0 and ACCESS_FACTOR1 in (RHEA_CNTL). By programming ACCESS_FACTOR0 and ACCESS_FACTOR1 to *n*, the respective strobe stretches its access over 2**n* TIPB bridge clock cycles (*n* cycles the strobe is inactive high; *n* cycles the strobe is active low).

The TIPB bridge clock referenced here is the input clock reference to the TIPB bridge module generated from the clock and reset management module. The clock rate for the TIPB bridge is fixed to the same value as that of the OMAP 3.2 traffic controller module. For details, see Chapter 4, *Clock Generation and Reset Management*.

7.1.4 MPU Posted Write

The MPU can perform a posted write. When posted write is enabled inside (ARM_RHEA_CNTL), data sent by the MPU is buffered in the TIPB bridge module, and the MPU can continue accessing other locations. The bridge

handles the access to the TIPB peripheral so that the MPU is not stalled during the access.

If the MPU/system DMA/OCP-I performs another TIPB operation when there is a posted write, this operation must wait until the posted write is complete. If the system DMA or OCP-I performs a read operation to the same address as the posted write, the posted write data is not forwarded to the DMA or OCP-I. Posted write is not supported for DMA and OCP-I accesses.

Using (ARM_RHEA_CNTL), you can enable posted write independently for strobe 0 or strobe 1 address spaces. This provides some flexibility over which MPU TIPB peripherals are configured for posted write. Refer to the memory map in chapter 10 for details on strobe address space assignments.

7.1.5 Time-Out

A TIPB access time-out limits the maximum time a peripheral can stall the processor. When starting an access on the TIPB, the time-out counter is loaded with the value programmed in the TIMEOUT bits of (RHEA_CNTL). If the current access is not finished when the counter reaches 0, the cycle is aborted and an abort is generated to the MPU/system DMA/OCP-I.

The time-out value is calculated by

$t_{time\text{-}out} = (1/f_{bridge\_clk}) \times ((RHEA\_CNTL.TIMEOUT) + 1)$

where f_{bridge_clk} is the traffic controller clock frequency setting.

The time-out can be used in conjunction with the posted write. The counter begins counting down when the posted write transaction has been scheduled, and this count continues against the posted transaction even if another transaction to the TIPB bridge occurs.

The time-out can be disabled using the TIMEOUT_EN bit in (ENH_RHEA_CNTL).

7.1.6 Debug

Debug registers are saved on the occurrence of an MPU_TIPB abort. Abort can be caused by time-out or by size mismatch between the access word width and the word width of the addressed peripheral.

The access address, data and error flags are saved to (DEBUG_ADDRESS), (DEBUG_DATA_LSB), (DEBUG_DATA_MSB), and (DEBUG_CTRL_SIGNALS).

7.2 Registers

All TIPB bridge configuration and debug registers are 16-bit registers. Write accesses to all TIPB registers can be performed only in MPU supervisor mode. Read accesses can be performed in MPU supervisor or user modes.

Table 117 provides a list of the TIPB registers. Table 118 through Table 126 provide register bit descriptions.

Table 117. TIPB Registers

Base Address = 0xFFFE D300 (public), 0xFFFE CA00 (private)			
Name	**Description**	**R/W**	**Offset**
RHEA_CNTL	TIPB control		0x00
RHEA_BUS_ALLOC	TIPB allocation control		0x04
ARM_RHEA_CNTL	MPU TIPB control		0x08
ENH_RHEA_CNTL	EnhancedTIPB control		0x0C
DEBUG_ADDRESS	Debug address		0x10
DEBUG_DATA_LSB	Debug data LSB		0x14
DEBUG_DATA_MSB	Debug data MSB		0x18
DEBUG_CTRL_SIGNALS	Debug control signals		0x1C
ACCESS_CNTL	Access control		0x20

Table 118. TIPB Control Register (RHEA_CNTL)

Base Address = 0xFFFE D300 (Public), 0xFFFE CA00 (Private), Offset = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:8	TIMEOUT	TIPB bus access time-out. When starting an access on TIPB bus, the time-out counter is loaded with this value. If the current access is not finished when the counter reaches 0, the cycle is aborted and abort indications are given to the peripheral and the MPU, DMA, or OCP-I. Maximum value for TIMEOUT is 255.	R/W	0xFF

Table 118. TIPB Control Register (RHEA_CNTL) (Continued)

Base Address = 0xFFFE D300 (Public), 0xFFFE CA00 (Private), Offset = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
7:4	ACCESS_FACTOR1	Clock period multiplication factor for TIPB strobe 1. Allows access to slow peripherals by lengthening TIPB strobe 1 period by a multiple of the internal TIPB bridge clock period. Note that the TIPB bridge clock period is the same as the OMAP traffic controller clock period. 0000: Same as 0001 0001: Strobe period = TIPB bridge clock period x 2 0010: Strobe period = TIPB bridge clock period x 4 0011: Strobe period = TIPB bridge clock period x 6 0100: Strobe period = TIPB bridge clock period x 8 0101: Strobe period = TIPB bridge clock period x 10 0110: Strobe period = TIPB bridge clock period x 12 0111: Strobe period = TIPB bridge clock period x 14 1000: Strobe period = TIPB bridge clock period x 1001: Strobe period = TIPB bridge clock period x 18 1010: Strobe period = TIPB bridge clock period x 20 1011: Strobe period = TIPB bridge clock period x 22 1100: Strobe period = TIPB bridge clock period x 24 1101: Strobe period = TIPB bridge clock period x 26 1110: Strobe period = TIPB bridge clock period x 28 1111: Strobe period = TIPB bridge clock period x 30	R/W	0x1

Table 118. TIPB Control Register (RHEA_CNTL) (Continued)

Base Address = 0xFFFE D300 (Public), 0xFFFE CA00 (Private), Offset = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
3:0	ACCESS_FACTOR0	Clock period multiplication factor for TIPB strobe 0. Allows access to slow peripherals by lengthening TIPB strobe 0 period by a multiple of the internal TIPB bridge clock period. Note that the TIPB bridge clock period is the same as the OMAP traffic controller clock period. 0000: Same as 0001. 0001: Strobe period = TIPB bridge clock period x 2 0010: Strobe period = TIPB bridge clock period x 4 0011: Strobe period = TIPB bridge clock period x 6 0100: Strobe period = TIPB bridge clock period x 8 0101: Strobe period = TIPB bridge clock period x 10 0110: Strobe period = TIPB bridge clock period x 12 0111: Strobe period = TIPB bridge clock period x 14 1000: Strobe period = TIPB bridge clock period x 1001: Strobe period = TIPB bridge clock period x 18 1010: Strobe period = TIPB bridge clock period x 20 1011: Strobe period = TIPB bridge clock period x 22 1100: Strobe period = TIPB bridge clock period x 24 1101: Strobe period = TIPB bridge clock period x 26 1110: Strobe period = TIPB bridge clock period x 28 1111: Strobe period = TIPB bridge clock period x 30	R/W	0x1

Table 119. TIPB Allocation Control Register (RHEA_BUS_ALLOC)

Base Address = 0xFFFE D300 (Public), 0xFFFE CA00 (Private), Offset = 0x04				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:6	Reserved		R/W	0x000
5	EXTNINT_PRIORITY	Priority between DMA and OCP-I when fixed priority scheme is selected. 0: DMA has priority. 1: OCP-I has priority.	R/W	0

Table 119. TIPB Allocation Control Register (RHEA_BUS_ALLOC) (Continued)

Base Address = 0xFFFE D300 (Public), 0xFFFE CA00 (Private), Offset = 0x04				
Bit	**Name**	**Function**	**R/W**	**Reset**
4	FIXNROUND_PRIORITY	Type of priority scheme used in DMA and OCP-I arbitration: 0: Round-robin scheme used 1: Fixed priority scheme used	R/W	0
3	PRIORITY_ENABLE	0: TIPB bus allocation is done using the RHEA_PRIORITY bits. 1: MPU has the same priority as the DMA/OCP-I transfers regarding TIPB bus allocation when it is in exception mode (IRQ and FIQ).	R/W	1
2:0	RHEA_PRIORITY	Defines TIPB priority between MPU and DMA/OCP-I. 000: MPU has priority over the DMA/OCP-I. 001 through 111: DMA/OCP-I has priority over the MPU, and can perform the programmed number of accesses before the MPU can access the bus.	R/W	001

Table 120. MPU TIPB Control Register (ARM_RHEA_CNTL)

Base Address = 0xFFFE D300 (Public), 0xFFFE CA00 (Private), Offset = 0x08				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:2	Reserved		R/W	0x0000
1	W_BUF_EN_1	0: Posted write buffer is bypassed. 1: Posted write buffer is enabled for MPU public and private TIPB peripherals having address space assigned to TIPB strobe 1.	R/W	0
0	W_BUF_EN_0	0: Posted write buffer is bypassed. 1: Posted write buffer is enabled for MPU public and private TIPB peripherals that have address space assigned to TIPB strobe 0.	R/W	0

Table 121. Enhanced TIPB Control Register (ENH_RHEA_CNTL)

Base Address = 0xFFFE D300 (Public), 0xFFFE CA00 (Private), Offset = 0x0C				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:4	Reserved		R/W	0x000
3	MASK_ABORT	0: An abort signal is sent to the MPU whenever an MPU to TIPB access is aborted. 1: The abort signal is not sent. MASK_ABORT does not mask/unmask DMA or OCP-I aborts.	R/W	1
2	Not Used	Not used in OMAP 3.2 (always one). Legacy HIGH_FREQ mode from OMAP 3.0/3.1		1
1	MASK_IT	0: An interrupt is sent to the MPU whenever a_TIPB write access (from MPU/DMA/OCP-I) is aborted or any TIPB access has a size mismatch. 1: The interrupt is masked.	R/W	1
0	TIMEOUT_EN	0: Do not enable the TIMEOUT feature. 1: Enable the TIMEOUT feature.	R/W	1

Table 122. Debug Address Register (DEBUG_ADDRESS)

Base Address = 0xFFFE D300 (Public), 0xFFFE CA00 (Private), Offset = 0x10				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	ADDRESS_DBG	Address from MPU memory interface; saved when an abort or access size mismatch occurs.	R	0xFFFF

Table 123. Debug Data LSB Register (DEBUG_DATA_LSB)

Base Address = 0xFFFE D300 (Public), 0xFFFE CA00 (Private), Offset = 0x14				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	DATA_DBG_LOW	Bits 15 to 0 of data bus from MPU. The value of the MPU data input is saved when a read access has a size mismatch, and the MPU data output bus is saved when a write access is aborted or has a size mismatch. If a read access is aborted, the value of this register is irrelevant.	R	0xFFFF

Table 124. Debug Data MSB Register (DEBUG_DATA_MSB)

Base Address = 0xFFFE D300 (Public), 0xFFFE CA00 (Private), Offset = 0x18				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	DATA_DBG_HIGH	Bits 31 to 16 of data bus from MPU. The value of the MPU data input bus is saved when a read access has a size mismatch, and the MPU data output bus is saved when a write access is aborted or has a size mismatch. If a read access is aborted, the value of this register is irrelevant.	R	0xFFFF

Table 125. Debug Control Signals Register (DEBUG_CTRL_SIGNALS)

Base Address = 0xFFFE D300 (Public), 0xFFFE CA00 (Private), Offset = 0x1C				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:11	Reserved		R/W	0x00
10:9	HOST_ID	Host-ID that caused the abort 00: MPU 01: DMA 10: OCP-I 11: Invalid	R	00
8	BURST_ACC	Indicates single or burst access on the TIPB; saved when abort or access size mismatch occurs. 0: Single access 1: Burst access	R	0
7:6	DBG_PERHMAS(1:0)	Peripheral memory access size on TIPB; saved when abort or access size mismatch occurs. 00: 8 bits 01: 16 bits 1x: 32 bits	R	11
5:4	DBG_MAS(1:0)	Memory access size on TIPB; saved when abort or access size mismatch occurs. 00: 8 bits 01: 16 bits 1x: 32 bits	R	11

Table 125. Debug Control Signals Register (DEBUG_CTRL_SIGNALS) (Continued)

Base Address = 0xFFFE D300 (Public), 0xFFFE CA00 (Private), Offset = 0x1C				
Bit	**Name**	**Function**	**R/W**	**Reset**
3	DBG_NSUPV	Indicates supervisor mode status of MPU; saved when abort or access size mismatch occurs. 0: Processor in supervisor mode 1: Processor not in supervisor mode	R	1
2	DBG_RNW	Indicates read or write transaction on the TIPB; saved when abort or access size mismatch occurs. 0: Write transaction 1: Read transaction	R	1
1	WR_SIZE_FLAG	Flag set to 1 when there is a mismatch between memory access size and peripheral memory access size. When read, the bit is reset to 0.	R	0
0	ABORT_FLAG	Flag set to 1 when TIPB access is aborted. When read, the bit is reset to 0.	R	0

Table 126. Access Control Register (ACCESS_CNTL)

Base Address = 0xFFFE D300 (Public), 0xFFFE CA00 (Private), Offset = 0x20				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:4	Reserved		R/W	0x000
3	DPS_EN	0: Dynamic power-saving mode is disabled. 1: Dynamic power-saving mode is enabled. When DPS is enabled, the bridge clock is turned.	R/W	0
2	MASK_OCPI NABORT	1: The abort for DMA access is masked before sending back to the DMA. 0: The abort for DMA access is sent back to the DMA. 1: The abort for OCPI access is masked before sending back to the OCPI.	R/W	0

Table 126. Access Control Register (ACCESS_CNTL) (Continued)

Base Address = 0xFFFE D300 (Public), 0xFFFE CA00 (Private), Offset = 0x20				
Bit	**Name**	**Function**	**R/W**	**Reset**
1	MASK_DMA NABORT	1: The abort for DMA access is masked before sending back to the DMA. 0: The abort for DMA access is sent back to the DMA.	R/W	0
0	DMA_ENABLE	1: DMA can access peripherals on the TIPB bridge 0: DMA cannot access peripherals on the TIPB bridge	R/W	1

Index

OMAP5912 Multimedia Processor
DSP Subsystem
Reference Guide

Literature Number: SPRU750A
March 2004

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.

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
		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 chapter describes the OMAP5912 multimedia processor DSP subsystem.

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.

Related Documentation From Texas Instruments

The following documents describe the OMAP5910 device and related peripherals. Copies of these documents are available on the Internet at www.ti.com. *Tip:* Enter the literature number in the search box provided at www.ti.com.

OMAP5912 Multimedia Processor Device Overview and Architecture Reference Guide (literature number SPRU748) introduces the setup, components, and features of the OMAP5912 multimedia processor and provides a high-level view of the device architecture.

OMAP5912 Multimedia Processor OMAP 3.2 Subsystem Reference Guide (literature number SPRU749) introduces and briefly defines the main features of the OMAP3.2 subsystem of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor DSP Sybsystem Reference Guide (literature number SPRU750) describes the OMAP5912 multimedia processor DSP subsystem. The digital signal processor (DSP) subsystem is built around a core processor and peripherals that interface with: 1) The ARM926EJS via the microprocessor unit interface (MPUI); 2) Various standard memories via the external memory interface (EMIF); 3) Various system peripherals via the TI peripheral bus (TIPB) bridge.

OMAP5912 Multimedia Processor Clocks Reference Guide (literature number SPRU751) describes the clocking mechanisms of the OMAP5912 multimedia processor. In OMAP5912, various clocks are created from special components such as the digital phase locked loop (DPLL) and the analog phase-locked loop (APLL).

OMAP5912 Multimedia Processor Initialization Reference Guide (literature number SPRU752) describes the reset architecture, the configuration, the initialization, and the boot ROM of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Power Management Reference Guide (literature number SPRU753) describes power management in the OMAP5912 multimedia processor. The ultralow-power device (ULPD) generates and manages clocks and reset signals to OMAP3.2 and to some peripherals. It controls chip-level power-down modes and handles chip-level wake-up events. In deep sleep mode, this module is still active to monitor wake-up events.This book describes the ULPD module and outline architecture.

OMAP5912 Multimedia Processor Security Features Reference Guide (literature number SPRU754) describes the security features of teh OMAP5912 multimedia processor. The OMAP5912 security scheme relies on the OMAP3.2 secure mode. The distributed security on the OMAP3.2 platform is a Texas Instruments solution to address m-commerce and security issues within a mobile phone environment. The OMAP3.2 secure mode was developed to bring hardware robustness to the overall OMAP5912 security scheme.

OMAP5912 Multimedia Processor Direct Memory Access (DMA) Support Reference Guide (literature number SPRU755) describes the direct memory access support of the OMAP5912 multimedia processor. The OMAP5912 processor has three DMAs:

- The system DMA is embedded in OMAP3.2. It handles DMA transfers associated with MPU and shared peripherals.
- The DSP DMA is embedded in OMAP3.2. It handles DMA transfers associated with DSP peripherals.
- The generic distributed DMA (GDD) is an OMAP5912 resource attached to the SSI peripheral. It handles only DMA transfers associated with the SSI peripheral.

OMAP5912 Multimedia Processor Memory Interfaces Reference Guide (literature number SPRU756) describes the memory interfaces of the OMAP5912 multimedia processor.

- SDRAM (external memory interface fast, or EMIFF)
- Asynchronous and synchronous burst memory (external memory interface slow, or EMIFS)
- NAND flash (hardware controller or software controller)
- CompactFlash on EMIFS interface
- Internal static RAM

OMAP5912 Multimedia Processor Interrupts Reference Guide (literature number SPRU757) describes the interrupts of the OMAP5912 multimedia processor. Three level 2 interrupt controllers are used in OMAP5912:

- One MPU level 2 interrupt handler (also referred to as MPU interrupt level 2) is implemented outside of OMAP3.2 and can handle 128 interrupts.
- One DSP level 2 interrupt handler (also referred to as DSP interrupt level 2.1) is instantiated outside of OMAP3.2 and can handle 64 interrupts.
- One OMAP3.2 DSP level 2 interrupt handler (referenced as DSP interrupt level 2.0) can handle 16 interrupts.

OMAP5912 Multimedia Processor Peripheral Interconnects Reference Guide (literature number SPRU758) describes various periperal interconnects of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Timers Reference Guide (literature number SPRU759) describes various timers of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Serial Interfaces Reference Guide (literature number SPRU760) describes the serial interfaces of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Universal Serial Bus (USB) Reference Guide (literature number SPRU761) describes the universal serial bus (USB) host on the OMAP5912 multimedia processor. The OMAP5912 processor provides several varieties of USB functionality. Flexible multiplexing of signals from the OMAP5912 USB host controller, the OMAP5912 USB function controller, and other OMAP5912 peripherals allow a wide variety of system-level USB capabilities. Many of the OMAP5912 pins can be used for USB-related signals or for signals from other OMAP5912 peripherals. The OMAP5912 top-level pin multiplexing

controls each pin individually to select one of several possible internal pin signal interconnections. When these shared pins are programmed for use as USB signals, the OMAP5912 USB signal multiplexing selects how the signals associated with the three OMAP5912 USB host ports and the OMAP5912 USB function controller can be brought out to OMAP5912 pins.

OMAP5912 Multimedia Processor Multi-channel Buffered Serial Ports (McBSPs) Reference Guide (literature number SPRU762) describes the three multi-channel buffered serial ports (McBSPs) available on the OMAP5912 device. The OMAP5912 device provides multiple high-speed multichannel buffered serial ports (McBSPs) that allow direct interface to codecs and other devices in a system.

OMAP5912 Multimedia Processor Camera Interface Reference Guide (literature number SPRU763) describes two camera inerfaces implemented in the OMAP5912 multimedia processor: compact serial camera port and camera parallel interface.

OMAP5912 Multimedia Processor Display Interface Reference Guide (literature number SPRU764) describes the display interface of the OMAP5912 multimedia processor.

- LCD module
- LCD data conversion module
- LED pulse generator
- Display interface

OMAP5912 Multimedia Processor Multimedia Card (MMC/SD/SDIO) (literature number SPRU765) describes the multimedia card (MMC) interface of the OMAP5912 multimedia processor. The multimedia card/secure data/secure digital IO (MMC/SD/SDIO) host controller provides an interface between a local host, such as a microprocessor unit (MPU) or digital signal processor (DSP), and either an MMC or SD memory card, plus up to four serial flash cards. The host controller handles MMC/SD/SDIO or serial port interface (SPI) transactions with minimal local host intervention.

OMAP5912 Multimedia Processor Keyboard Interface Reference Guide (literature number SPRU766) describes the keyboard interface of the OMAP5912 multimedia processor. The MPUIO module enables direct I/O communication between the MPU (through the public TIPB) and external devices. Two types of I/O can be used: specific I/Os dedicated for 8 x 8 keyboard connection, and general-purpose I/Os.

OMAP5912 Multimedia Processor General-Purpose Interface Reference Guide (literature number SPRU767) describes the general-purpose in-

terface of the OMAP5912 multimedia processor. There are four GPIO modules in the OMAP5912. Each GPIO peripheral controls 16 dedicated pins configurable either as input or output for general purposes. Each pin has an independent control direction set by a programmable register. The two−edge control registers configure events (rising edge, falling edge, or both edges) on an input pin to trigger interrupts or wake−up requests (depending on the system mode). In addition, an interrupt mask register masks out specified pins. Finally, the GPIO peripherals provide the set and clear capabilities on the data output registers and the interrupt mask registers. After detection, all event sources are merged and a single synchronous interrupt (per module) is generated in active mode, whereas a unique wake−up line is issued in idle mode. Eight data output lines of the GPIO3 are ORed together to generate a global output line at the OMAP5912 boundary. This global output line can be used in conjunction with the SSI to provide a CMT−APE interface to the OMAP5912.

OMAP5912 Multimedia Processor VLYNQ Serial Communications Interface Reference Guide (literature number SPRU768) describes the VLYNQ of the OMAP5912 multimedia processor.

VLYNQ is a serial communications interface that enables the extension of an internal bus segment to one or more external physical devices. The external devices are mapped into local, physical address space and appear as if they are on the internal bus of the OMAP 5912. The external devices must also have a VLYNQ interface. The VLYNQ module serializes bus transactions in one device, transfers the serialized data between devices via a VLYNQ port, and de-serializes the transaction in the external device.

OMAP5912 includes one VLYNQ module connected on OCPT2 target port and OCPI initiator port. These connections are configured via a static switch, which selects either SSI or VLYNQ module. This switch, forbids the simultaneous use of GDD/SSI and VLYNQ. The switch is controlled by the VLYNQ_EN bit in the OMAP5912 configuration control register (CONF_5912_CTRL).

OMAP5912 Multimedia Processor Pinout Reference Guide (literature number SPRU769) provides the pinout of the OMAP5912 multimedia processor. After power-up reset, the user can change the configuration of the default interfaces. If another interface is available on top of the default, it is possible to enable a new interface for each ball by setting the corresponding 3-bit field of the associated FUNC_MUX_CTRL register. It is also possible to configure on-chip pullup/pulldown. This document

also describes the various power domains so that the user can apply the different interfaces seamlessly with external components.

OMAP5912 Multimedia Processor Window Tracer (WT) Reference Guide (literature number SPRU770) describes the window tracer module used to capture the memory transactions from four interfaces: EMIFF, EMIFS, OCP-T1, and OCP-T2. This module is located in the OMAP3.2 traffic controller (TC).

OMAP5912 Multimedia Processor Real-Time Clock Reference Guide (literature number SPRUxxx) describes the real-time clock of the OMAP5912 multimedia processor. The real-time clock (RTC) block is an embedded real-time clock module directly accessible from the TIPB bus interface.

Trademarks

Contents

Figures

Tables

DSP Subsystem

This chapter describes the OMAP5912 multimedia processor DSP subsystem.

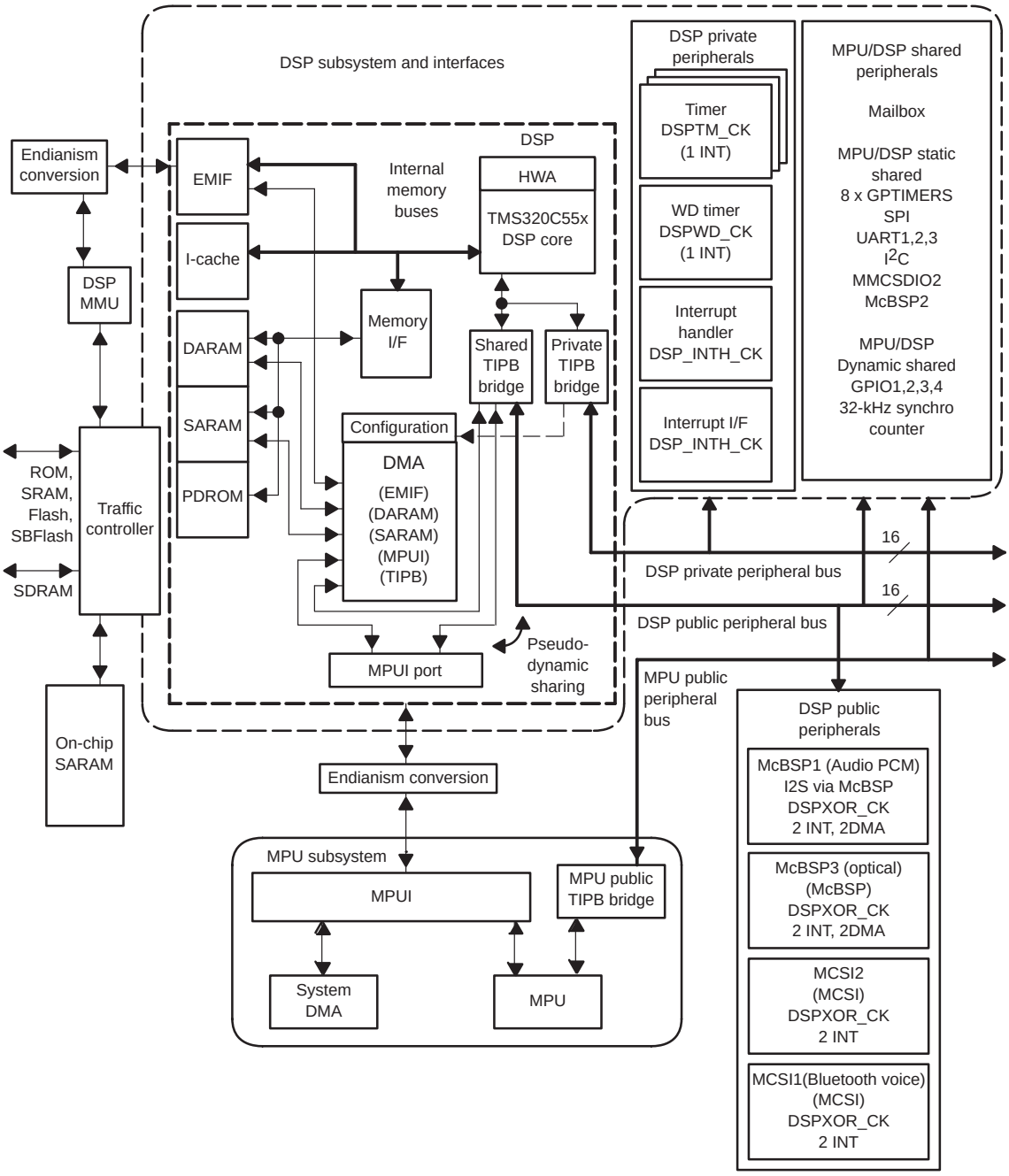
1 Architecture Overview

The digital signal processor (DSP) subsystem is built around a core processor and peripherals that interface with:

- The ARM926EJS via the microprocessor unit interface (MPUI)
- Various standard memories via the external memory interface (EMIF)
- Various system peripherals via the TI peripheral bus (TIPB) bridge

Figure 1 shows the DSP subsystem and the modules with which it interfaces.

Figure 1. DSP Subsystem and Modules



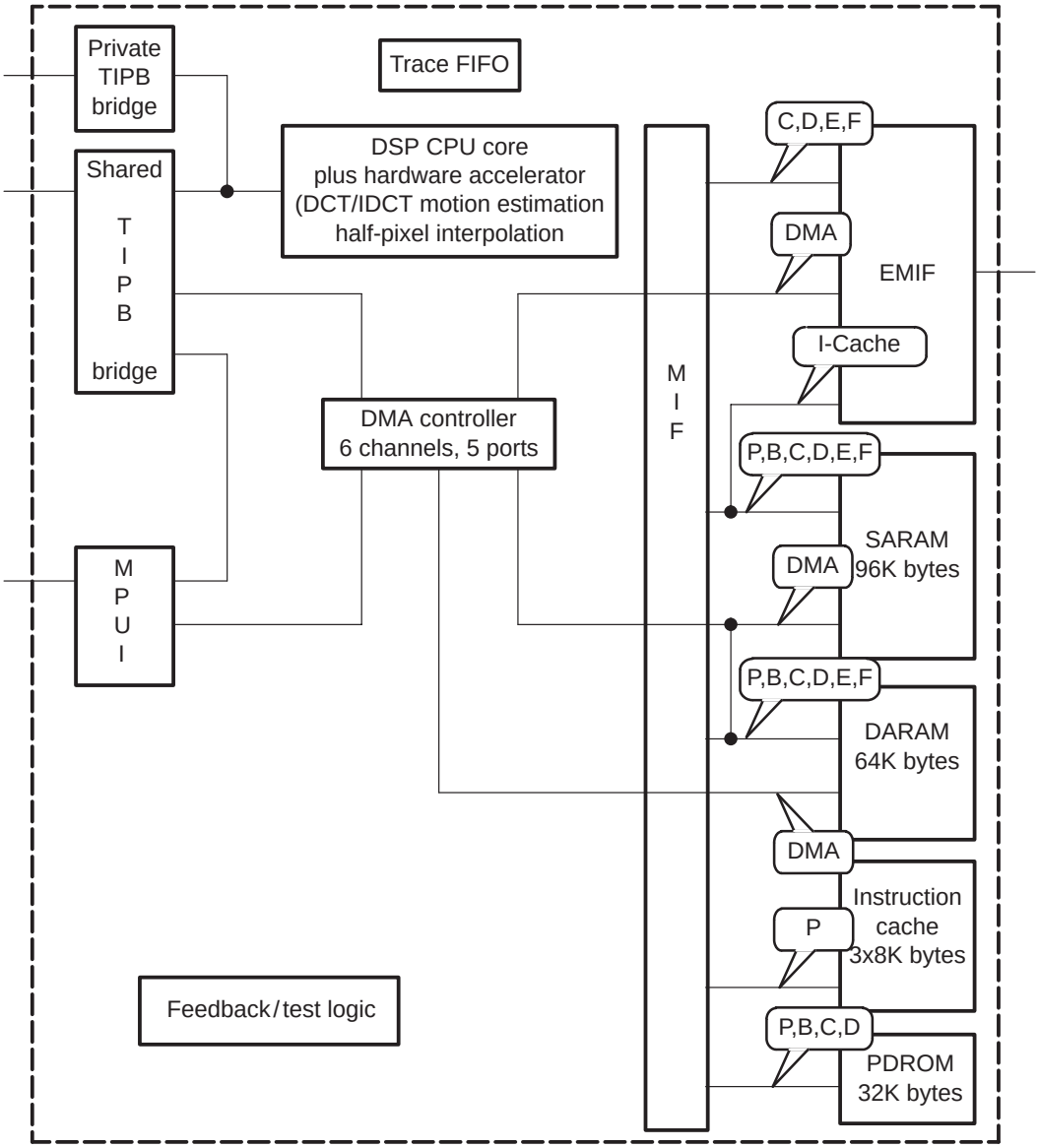
The DSP subsystem has the following components:

- DSP module:
  - TMS320C55x (C55x) DSP CPU core revision 2.11
  - Tightly coupled hardware accelerator: discrete cosine transform/ inverse discrete cosine transform (DCT/IDCT), motion estimation, and half-pixel interpolation
  - Tightly coupled memories and their interfaces: dual-access RAM (DARAM), single-access RAM (SARAM), programmable dynamic ROM (PDROM), and instruction cache (I-cache)
  - External memory interface (EMIF) that connects the CPU to external and loosely coupled memories
  - A six-channel DMA controller that can copy memory contents from one address to another without CPU intervention
  - MPUI that permits high-bandwidth parallel access to DSP resources by the MPU and system DMA
  - TIPB bridge that provides two external bus interfaces for private and public peripherals
- DSP subsystem peripherals:
  - Three general-purpose 32-bit timers
  - One general-purpose UART
  - A 16-signal general-purpose input/output (GPIO) module for bit input or output
  - A mailbox module to permit interrupt-based signaling between the DSP and MPU
  - Watchdog timer
  - Level 2 interrupt handler

1.1 DSP Core

Figure 2 shows the DSP core.

Figure 2. DSP Core and Internal Bus Designations



2 TMS320C55x DSP CPU Overview

Features for the high-performance, low-power C55x DSP CPU include:

- Advanced multiple-bus architecture with one internal program memory bus and five internal data buses (three dedicated to reads and two dedicated to writes)

- Unified program/data memory architecture
- Dual 17-bit x17-bit multipliers coupled to 40-bit dedicated adders for non-pipelined single-cycle multiply accumulate (MAC) operations
- Add/compare/select (CSSU) unit for the add/compare section of the Viterbi operator
- Exponent encoder to compute an exponent value of a 40-bit accumulator value in a single cycle
- Two address generators with eight auxiliary registers and two auxiliary register arithmetic units
- 8M x 16-bit (16M bytes) total addressable memory space
- Single-instruction repeat or block repeat operations for program code
- Conditional execution
- Seven-stage pipeline for high instruction throughput
- Instruction buffer unit that loads, parses, queues, and decodes instructions to decouple the program fetch function from the pipeline
- Program flow unit that coordinates program actions among multiple parallel CPU functional units
- Address data flow unit that provides data address generation and includes a 16-bit arithmetic unit capable of performing arithmetic, logical, shift, and saturation operations
- Data computation unit containing the primary computation units of the CPU, including a 40-bit arithmetic logic unit, two MAC units, and a shifter

2.1 On-Chip Memory

Features include:

- DARAM that supports two memory accesses per cycle per block
- SARAM that supports one memory access per cycle per block
- PDROM that provides nonvolatile storage for program or data

2.1.1 Power Conservation

Features include:

- Software-programmable idle domains that provide configurable low-power modes
- Automatic power management
- Advanced low-power complimentary metal-oxide semiconductor (CMOS) process

2.2 Hardware Acceleration Modules

The OMAP5912 device contains several hardware acceleration modules to improve performance and reduce power consumption for certain computations relating to image and video processing. These coprocessors include:

- DCT/IDCT accelerator
- Motion estimation calculation accelerator
- Half-pixel interpolation accelerator

2.3 CPU Overview

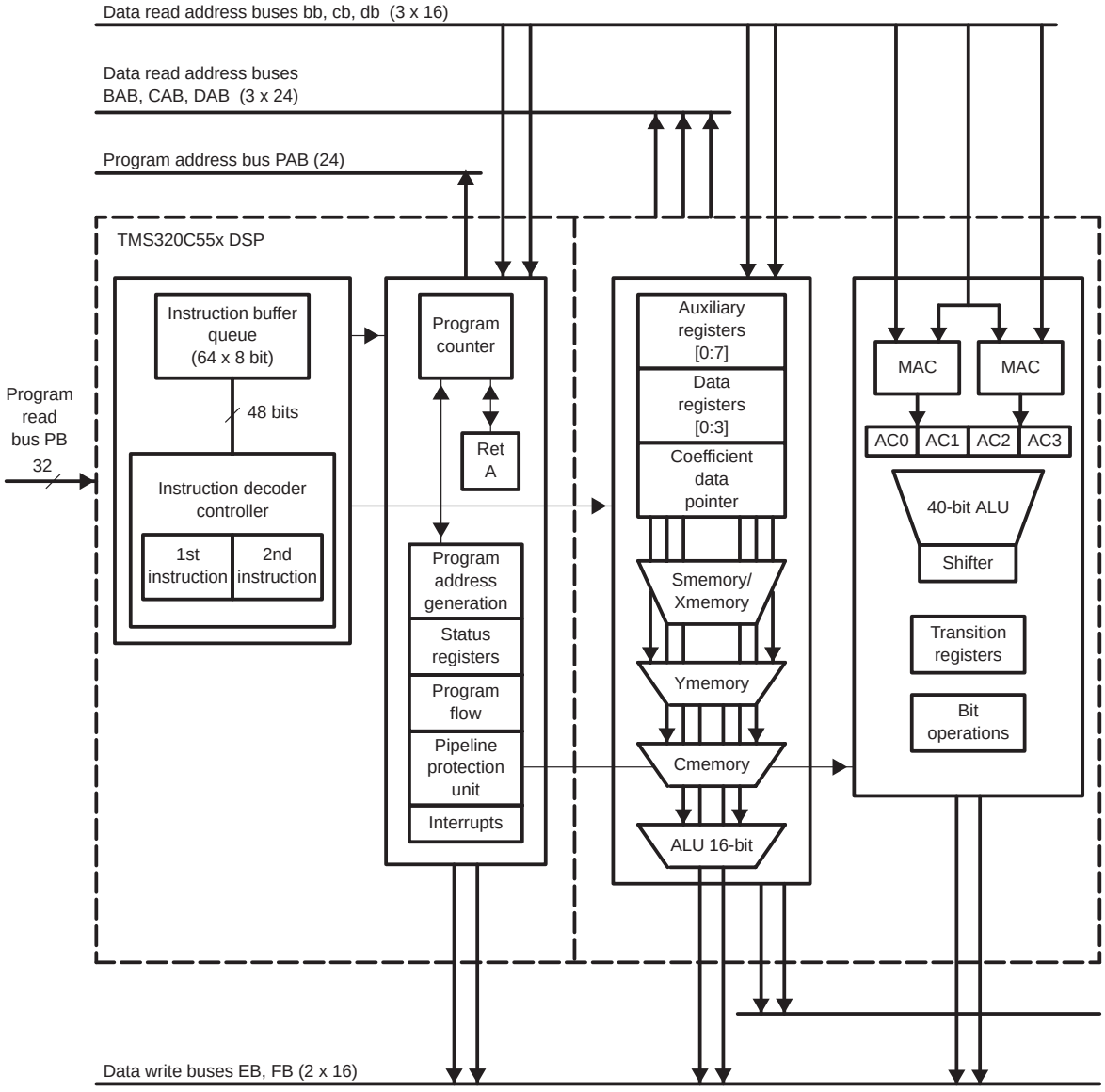
The DSP core has four functional units:

- The instruction unit (IU) loads, parses, queues, and decodes instructions and includes an instruction buffer unit (IBQ) to decouple the program fetch function from the pipeline.
- The program flow unit (PU) coordinates program actions among multiple parallel CPU functional units.
- The address data flow unit (AU) provides data address generation and includes a 16-bit arithmetic unit capable of performing arithmetic, logical, shift, and saturation operations.
- The data computation unit (DU) contains the primary computation units of the CPU including a 40-bit arithmetic logic unit, two multiply-accumulate units (MACs), and a shifter.

To permit high computational throughput and a fast instruction cycle rate, the CPU employs several sets of parallel buses to access code and data structures. The program address and data buses (P-bus) perform 32-bit instruction fetches to feed the instruction unit. The B, C, and D addresses and data buses enable the CPU to access up to three 16-bit data operands per cycle. E and F addresses and data buses allow the CPU to write up to two 16-bit quantities per cycle.

Figure 3 shows the C55x DSP architecture.

Figure 3. C55x DSP Architecture



For details on CPU architecture and instruction set, see the following documents:

- *TMS320C55x Technical Overview* (SPRU393)
- *TMS320C55x DSP CPU Reference Guide* (SPRU371)

- *TMS320C5510 DSP Functional Overview* (SPRU312) (only CPU sections apply to the OMAP5912 device)

3 DSP Memory

The DSP subsystem contains four types of tightly coupled memory to provide maximum efficiency of the DSP CPU.

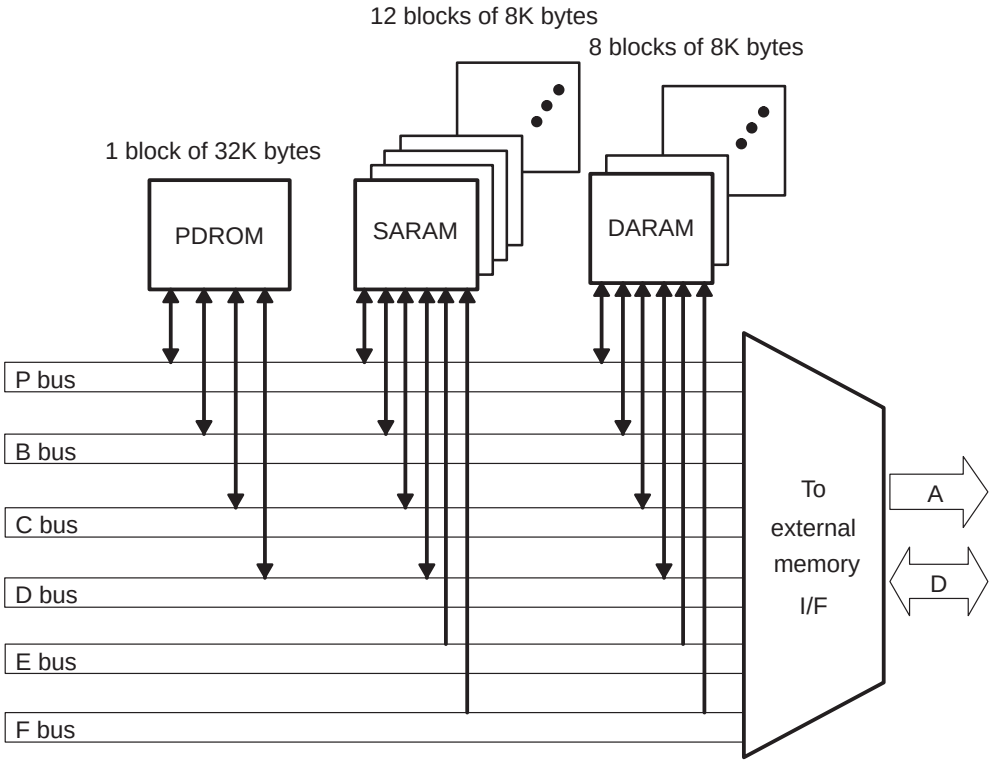
- Dual-access RAM (DARAM)
- Single-access RAM (SARAM)
- Programmable dynamic ROM (PDROM)
- Configurable I-cache structure

The CPU uses six sets of buses to simultaneously fetch up to 32 bits of program and to read up to 48 bits of data operands from memory (or write up to 32 bits to memory). To achieve maximum performance from the architecture, the programmer must pay close attention to placement of code and data structures within the on-chip memory resources. For more details, see *TMS320C55x DSP Programmers Guide* (SPRU376).

Loosely coupled memory devices can be accessed via the traffic controller module. This flexible memory interface permits DSP access to another block of SRAM (shared with the MPU) as well as external memory devices such as flash memory and SDRAM.

Figure 4 shows DSP memory connections.

Figure 4. DSP Memory Connections



3.1 Internal Memory

- The DARAM (64K bytes) can support up to two memory accesses in one CPU clock cycle into each RAM block. Accesses can be made from any internal data, program, or DMA bus. The DARAM memory consists of eight blocks of 8K bytes each.
- The SARAM (96K bytes) can support one memory access in one 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. The SARAM memory consists of 32 blocks of 8K bytes each.
- The PDROM (32K bytes) can support one memory read in one CPU clock cycle. 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 32K bytes.

3.2 Instruction Cache

The DSP instruction cache (I-cache) module is a special-purpose, tightly coupled, RAM-based program memory. The module is designed to

significantly improve the CPU performance by buffering the instructions most recently fetched from external memory. The entire external program memory space is cacheable.

The I-cache total size is 24K bytes divided into three memory banks. Two 8K-byte banks are traditional cache modules. The third module is a special type of cache called ramset, which is divided into two 4K-byte blocks. Each block is called ½ ramset. Details of the organization and usage are explained below. The I-cache can be configured in any the following organizations:

1) Two-way set-associative (two cache modules)
2) Direct-mapped (one cache module)
3) Two-way set-associative and one or two ½ ramset
4) Direct-mapped and one or two ½ ramset

The I-cache can be enabled, disabled, or modified at any time by the programmer using software control. The TIPB bridge allows access to the cache configuration registers in the DSP I/O space. At reset the I-cache is disabled. The user must configure the I-cache to be able to use the ramset.

The initial normal cache hit is a one-wait-state operation. Thereafter, the I-cache performs a simple branch prediction for cache access (that is, a branch not taken is always assumed). With this feature, no-branch continuous fetches are no-wait-state operations. The I-cache returns one 32-bit word for each fetch. Fetches are always aligned on a 32-bit boundary.

When a cache miss occurs, wait states are inserted that are dependent upon the external memory access time. The I-cache retrieves instructions from external memory in a burst of four 32-bit words (to fill cache line). To reduce the penalty of misses, a streaming feature is implemented where the instruction word requested by the DSP is sent back as soon as it is retrieved from external memory, so all four 32-bit words do not have to be loaded into the cache line first. Additionally, program fetch requests that fall in a cache line already being retrieved because of a previous miss are serviced as soon as the word becomes available. This streaming feature can increase the performance of even non-looping code executing from external memory.

The I-cache supports emulation debug read and breakpoint/watchpoint insertion by invalidating cache lines with the corresponding data changed in the external memory space.

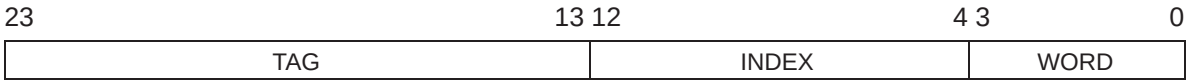
3.2.1 Cache Memory Organization

The two-way set-associative cache configurations are organized as one or 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 word, line index, and TAG sections, as shown in Figure 5. The line index determines which cache entry to test. The TAG is used to check the cache contents against the requested instruction address. The word field determines which bytes on the cache line to fetch when a cache hit occurs. A two-way set-associative cache structure has two 8K-byte RAM blocks in parallel, so there are two cache lines and two TAGs associated with each line number defined by the index. When an instruction is written in the cache, its TAG is also written.

Figure 5. Virtual Address Mapped to Cache Line



WORD field size = 4 bits
INDEX field size = 9 bits
TAG field size = 11 bits

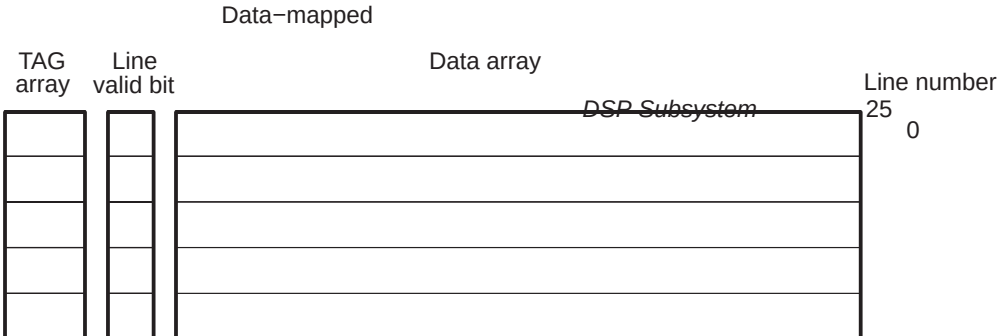
3.2.2 Ramset Memory Organization

Ramset memory is a special type of cache memory. The 8K-byte memory block is divided into two 4K-byte sections (called ½ ramset) that can be controlled independently. Because the ½ ramset memory block is 4K bytes long and the line size is still 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 the user to dynamically remap 4K-byte memory blocks from external to internal memory (once the ½ ramset memory has been filled).

3.2.3 Instruction Cache Structure

The I-cache block has status bits associated with each cache line. A line-valid bit (LVB) defines whether or not the data in the cache line has valid instructions resulting from a fetch from external program memory. When the cache is first enabled (or after flushing), all the cache lines in the data array and TAGs in the TAG array are random. The line valid bit is set to 0 for all cache entries. When an external program fetch occurs and the corresponding line valid bit is zero, a cache miss occurs automatically. Figure 6 shows the cache structure for a direct-mapped cache block.

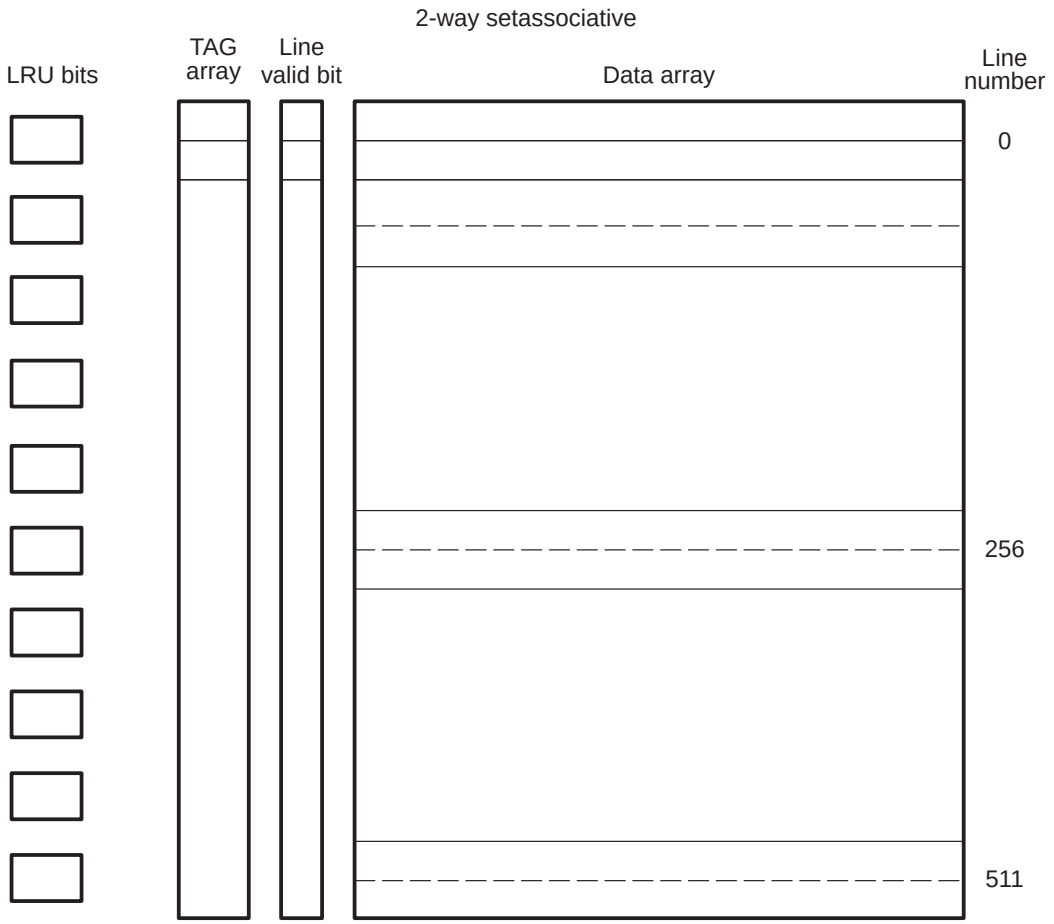
Figure 6. Cache Structure—Direct-Mapped Cache Block



The two-way set associative cache has an extra status bit for each cache line index. The least recently used (LRU) bits are associated with the pair of cache lines assigned to the same line index. The LRU bit indicates which of the two cache lines did not have the last cache hit. When a cache miss occurs, the line pointed to by the LRU bit is replaced by the new cache line fetched from external program memory. Then the value in the LRU bit field is toggled to point to the other cache line.

Figure 7 shows the two-way set-associative cache structure.

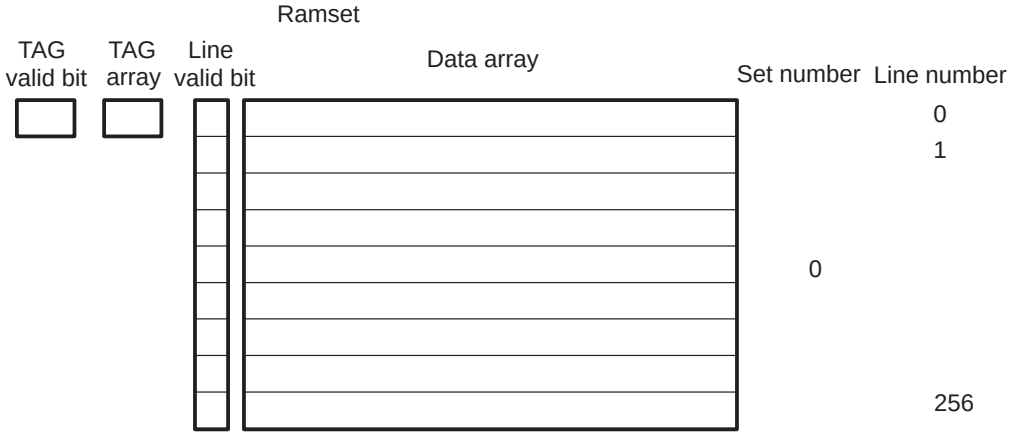
Figure 7. Two-Way Set-Associative Cache Structure



3.2.4 Ramset Structure

Ramset memory uses the live valid bit (LVB) to indicate whether or not an individual line in the memory block is valid like regular cache memory. In addition, there is a TAG valid bit (TVB) associated with each ½ ramset memory block that indicates whether the TAG entry is valid. The TVB is set to 0 when the ½ ramset is enabled or flushed. Once the ramset data array is filled (all LVB=1), the TAG valid bit is set. This field can be monitored for ramset load completion. While the ramset data array is being loaded from external memory, DSP requests have lower priority than ramset fill requests. Figure 8 shows the ½ ramset cache structure.

Figure 8. ½ Ramset Cache Structure



3.2.5 Instruction Cache Operation

The operation of the cache is initiated if an external program fetch is requested by the DSP. The overall process follows these steps:

1) The CPU generates a program instruction fetch request.

2) The MIF decodes the program address to route the request to either the I-cache or the internal memories.

3) The I-cache returns the requested instruction (hit), or, if not present (miss), initiates an external memory access via the EMIF.

To determine if a cache hit or miss results, a control circuit parses the program address into TAG, index number, and word number. The index number selects the cache line(s) to test. If there is a match between the TAG number and the TAG(s) stored in the TAG array, then the line valid bit is also verified to confirm that the line matching the tag is valid. This is called a presence check. On a

cache hit, the word number is used to select which word from the cache line is returned to the DSP to complete the fetch request.

When ½ ramset memory is enabled, the presence check is performed on both the cache and ramset blocks simultaneously. Table 1 describes the instruction presence-check behavior. Ramset presence check is for TVB-LVB.

Table 1. Presence-Check Truth Table

Cache	Ramset	Presence	Behavior
Miss	Miss	False	The cache is loaded with the corresponding line.
Miss	Hit-Hit	True	The instruction is read from the ramset.
Hit	Miss	True	The instruction is read from the cache.
Hit	Hit-Hit	True	The instruction is read from the ramset.
Miss	Hit-Miss	False	The ramset is loaded with the corresponding line.
Hit	Hit-Miss	True	The ramset is loaded with the corresponding line.

If there is no match between the TAG number and the stored TAG(s), then a cache miss has occurred. The I-cache then initiates a burst fetch request to the EMIF. Four words are fetched in a burst read to fill the least recently used cache line of the pair located at the index number offset into the data array. At that point, the requested instruction is returned to the DSP, while the cache line continues to fill. The LRU, LVB, and TVB are updated when line fill is completed.

3.3 Instruction Cache Configuration

Control of the I-cache is maintained through a set of memory mapped registers in the DSP I/O space and through three bits located in the CPU register ST3. Cache-related bits located in ST3 are described in Table 2.

3.3.1 ST3 Control Register (ST3)

Table 2 describes the cache-related bits located in ST3. Through the ST3 register, the cache can also be frozen to lock a time-critical routine so it cannot be overwritten.

Table 2. ST3 CPU Register (ST3)

Bit	Name	Description
15	CAFRZ	Instruction cache freeze CAFRZ = 1: The cache contents are locked. In this mode, the cache contents are not updated on a cache miss, but its contents still are available for cache hits. CAFRZ = 0: The cache contents can be unlocked: this is the default operating mode of the cache. CAFRZ is cleared at reset.
14	CAEN	Instruction cache enable CAEN = 1: Program fetches either occur from the cache, from the internal memory, or from the external memory through the external memory interface of the device. Program fetches depend on the fetched program code addresses. CAEN = 0: The cache controller never receives a program request, so all program fetch requests are handled either by the internal memory or the external memory. Default value after reset.
13	CACLR	Instruction cache clear CACLR = 1: The cache clear process is ongoing. During the cache clear process, all the cache blocks are invalid. CACLR = 0: The cache clear process is completed. CACLR bit is cleared by the cache hardware upon completion of cache clear process. CACLR is cleared at reset.
12:0		Not applicable (non-cache bits)

3.3.2 I/O Mapped Cache Control Registers

Table 3 through Table 9 describe the register bits.

Table 3. Global Control Register (GCR)

Bit	Name	Description	Access
15	Cut Clock	Stops I-cache control clock clock domain when I-cache is disabled	R/W
14	Auto Gating	Enables automatic clock gating in EMIF interface clock domain, active high	R/W
13	Reserved		
12	Flush Line	Flush the line specified by the flush line address register: 0: No flush 1: Flush the specified line; once the line flush occurs, the flush line bit is automatically reset to 0.	R/W

Table 3. Global Control Register (GCR) (Continued)

Bit	Name	Description	Access
11	Global Flush	Global flush configuration: 0: The flush configuration must take into account the specific flush bits (n-way and ½ ramset). 1: All the I-cache is flushed when CACLR = 1. (Line valid bits are invalidated and ramset tag valid bit is invalidated.)	R/W
10	½ ramset Presence	½ ramset presence: 0: No ½ ramset 1: N x ½ ramset	R/W
9	Way Presence	Way presence: 0: No way 1: N-way (set-associative or direct-mapped)	R/W
8:5	½ ramset Number	I-cache ½ ramset number: 1..2 ½ ramset number(1 is coded by 0000/b and 2 otherwise)	R/W
4:3	Way Number	I-cache way number: x0: 1-way (direct-mapped) x1: 2-way (set-associative)	R/W
2	Streaming	Streaming enable 1: Streaming enabled This is the only supported configuration.	R/W
1	Ram Fill Mode	Type of load for a ramset flush line 1: RAM fill mode (fill all lines of set) (Invalidation of the TAG valid bit during fill/ after fill is complete; TAG valid bit is set). This is the only supported configuration.	R/W
0	Global Enable	Global enable configuration: 0: Takes into account the specific enable bits (n-way and ½ ramset) 1: All I-cache enabled when CAEN=1	R/W

Table 4. Flush Line Register 0 (FLR0)

Bit	Name	Description	Access
15:0	Line address	Low 16 bits of line address: byte address This address is used to select the line when a flush line occurs.	R/W

Table 5. Flush Line Register 1 (FLR1)

Bit	Name	Description	Access
15:8	Reserved		R
7:0	Line address	High 8 bits of line address: byte address This address is used to select the line when a flush line occurs.	R/W

Table 6. I-Cache N-Way Register (NWCR)

Bit	Name	Description	Access
15:5	Reserved		R
4:2	Way size	Way size = 011: 8K bytes: Way size configuration must not be set to any other value (reserved value).	R/W
1	Flush	N-way flush mask: 0: N-way not flushed by the GL_CACHECLR_TR input 1: N-way flushed by the GL_CACHECLR_TR input (all the line valid bits are invalidated)	R/W
0	Enable	N-way enable mask: 0: N-way not enabled when CAEN = 1 1: N-way enabled when CAEN = 1	R/W

Table 7. I-Cache Status Register (ISR)

Bit	Name	Description	Access
15:3	Reserved		R
2	I-cache_Enable	Disable/enable I-cache: 0: I-cache disabled 1: I-cache enabled This bit must be queried to ensure that the i-cache is enabled before writing to the ramset TAG register.	R

1	Debug mode	When either the ½ ramset1..2 or TAG registers are written through the TIPB bus in debug mode, this bit is set. This bit is reset to 0 if this register is read in normal operation; it is not reset to 0 if this register is read during emulation.	R
0	Bus error	When the two ½ ramset TAG registers are configured to be the same address, I-cache sets this bit and returns a bus error to the CPU. This bit is reset to 0 if this register is read in normal operation; it is not reset to 0 if this register is read during emulation.	R

Table 8. ½ Ramset Control Registers (RCR1, RCR2)

Bit	Name	Description	Access
15	Tag valid	½ ramset enable mask: 0: ½ ramset n is not enabled when CAEN = 1. 1: ½ ramset n is enabled when CAEN = 1.	R
14:2	Reserved		R
1	Flush	½ ramset flush mask: 0: ½ ramset n is not flushed when CACLR = 1. 1: ½ ramset n is flushed when CACLR = 1. (All the line valid bits are invalidated; tag valid bit is invalidated).	R/W
0	Enable	Indicates that the ½ ramset fill has been completed and all of its line valid bits are set: 1: ½ ramset fill has been completed. 0: ½ ramset fill has not started or not completed.	R/W

Table 9. ½ Ramset TAG Registers (RTR1, RTR2)

Bit	Name	Description	Access
14:0	Tag	Byte address: Because a ramset size is 4K bytes, the TAG field represents PADDR (23:12).	R/W

3.3.3 LVB and LRU Status Registers

The LRU bits of the two-way cache and the line valid bits of cache and ramsets are readable through the TIPB bus. They are mapped to the I/O space, as shown in Table 10.

Table 10. I-Cache LVB and LRU I/O Space Map

Start Word Address	End Word Address	Description
0x1600	0x161F	512 LVBs of bank0 = 32 words

0x1620	0x163F	512 LVBs of bank1 = 32 words
0x1640	0x165F	512 LVBs of bank2 = 32 words
0x1660	0x167F	512 LRUs = 32 words

3.4 I-Cache Operations

3.4.1 Enable and Disable I-Cache

The cache and ramset modules can be enabled together by setting the global enable bit in GCR, in conjunction with the way/ramset presence bits. The I-cache is then enabled when CAEN is set in the DSP CPU ST3 register.

The cache and ramset modules can be enabled separately by resetting the global enable bit in the GCR and by setting the individual enable bits in NWCR and RCR1/2, respectively. CAEN must be set in ST3 register for effects to take place. The GCR presence bits still must be set when enabling the way/ramset using the individual enable bits. The I-cache is disabled when CAEN is reset in ST3.

The above fields must be set before the I-cache is enabled using CAEN = 1. Setting CAEN = 1 without enabling a specific configuration is not recommended.

3.4.2 Cache Functional Configuration

The configuration of the cache, when enabled, is determined by the way presence field in GCR and the way number field in GCR.

These fields must be set before the I-cache is enabled by CAEN, and must not be changed when the I-cache is enabled.

3.4.3 Ramset Functional Configuration

The configuration of ramsets, when enabled, is determined by the following fields:

- Ramset presence field in GCR
- ½ ramset number field in GCR
- Enable field in RCR1/2
- Tag field in RTR1...

Set the first three fields before the I-cache is enabled by setting CAEN=1; do not change the first three fields when I-cache is enabled.

Set the last field after setting CAEN in ST3 CPU register to start the filling of ramset. Then query the I-CACHE_ENABLE bit in the ISR to verify that the

I-cache is enabled. The CAEN = 1 request to enable the I-cache is not instantaneous, and latency can occur before the I-cache is truly enabled. RTR1/2 can be changed to map ramsets to different locations anytime when the I-cache is enabled. Again, note that the ramset must be used in conjunction with the cache memory (two-way or direct).

3.4.4 Freeze Mode

On a write to CAFRZ = 1 in the DSP CPU ST3 register, the content in the I-cache is locked. During freeze mode, any I-cache miss does not update the I-cache. Data that has been cached before freeze is still accessible if there is a cache hit.

Any I-cache miss during freeze mode is forwarded to the EMIF as a burst request. Therefore, each single request to a frozen I-cache is serviced by one complete EMIF burst cycle. For this reason, code profiling during freeze states is recommended to achieve optimal performance and to limit the number of misses.

3.5 Configuration Examples

3.5.1 Example of Configuring and Enabling Two-Way Set-Associative Cache

1) Set the GCR to have the following settings:
   a) Global enable = 0
   b) Way presence = 1
   c) Way number = 01
   d) Streaming = 1
   e) Others = Don’t care
2) Set the NWCR to have the following settings:
   a) Enable = 1
   b) Way size = 011
   c) Others = Don’t care
3) Write 1 to CAEN bit in CPU ST3 register.

3.5.2 Example of Configuring and Enabling Direct Map With One ½ Ramset

This sequence of operations configures a direct map ramset with one ½ ramset. A ramset configuration must be used in conjunction with a cache two-way configuration.

1) Set the GCR to have the following settings:
   a) Global enable = 1
   b) Way presence = 1
   c) Way number = 00
   d) ½ ramset presence = 1
   e) ½ ramset number = 0000
   f) Streaming = 1
   g) Line_Fill_Mode = 1
2) Set the NWCR to have the following settings:
   a) Way size = 011
   b) Others = Don't care
3) Write 1 to CAEN bit in CPU ST3 register.
4) Query the I-CACHE_ENABLE bit in the I-cache status register to verify that the cache is enabled.
5) Set the RTR1 to map to desired address space. For example, to load the first 4K bytes starting from address 800000h, 0x0800h is loaded into RTR1

The following sequence then occurs during ramset mode = 1.

1) The tag valid bit in the RCR is reset and the ramset is automatically reinitialized/reloaded.
2) During ramset loading, CPU requests have lower priority than ramset fill requests and are not serviced until the ramset is loaded. It is recommended then for optimal performance to avoid external memory accesses during ramset load operations. As each line of the ramset is filled, the lVB of the line is set so that the progress of the ramset fill operation can be monitored.
3) Once the ramset is filled, the tag valid bit in the RCR is set. This field can be monitored for ramset load completion.

3.5.3 Flush I-Cache

To flush the whole I-cache (invalidate all the line valid bits), you must:

1) Set the global flush bit in GCR.

2) Write CACLR = 1 in DSP CPU ST3 register.

Once the I-cache has been flushed, the CACLR bit is automatically reset to 0.

3.5.4 Flush Cache Line

This operation flushes a single line in the cache (for example, when an instruction is directly modified in the external memory by the software):

1) Write the address, which identifies the flush line (see the flush line address register: FLAR).

2) Set the flush line bit in GCR.

Once the line is flushed, the flush line bit in GCR is reset by the I-cache.

When the flush line is activated, the I-cache searches the flush line in its data array. Three cases are possible:

- This line is found in its two-way (or direct-mapped) data array, and then the corresponding line valid bit is invalidated. Consequently, for the next access at the same address, a miss and a new line is loaded from the external memory.
- This line is found in one of the ramset data arrays, and then the corresponding line valid bit is invalidated. Consequently, for the next access at the same address, a hit/miss and a new line is loaded from the external memory.
- This line is not found. In this case, no action is performed and the flush line is terminated.

3.5.5 Flush the n-Way I-Cache

To flush the n-way I-cache:

1) Reset the global flush bit (see the global control register). See Table 17.
2) Set the n-way-flush bit.
3) Write CACLR = 1 in DSP CPU ST3 register.

All of the line valid bits of the n-way I-cache are then invalidated.

3.5.6 Flush a ½ Ramset

To flush a ½ ramset, do the following:

1) Reset the global flush bit (see the global control register). See Table 17.
2) Set the ½ ramset flush bit of the corresponding RCR register.
3) Write CACLR = 1 in DSP CPU ST3 register.

All the line valid bits of the ½ ramset are then invalidated, as well as the TAG_VALID bit.

3.6 I-Cache Performance

The I-cache performances can be characterized by:

- Average memory-access time = (Hit rate* Hit time) + (Miss rate * Miss penalty.)

Hit time is described in Section 3.6.1 and miss penalty is described in Section 3.6.2.

3.6.1 Hit Time

The hit time is the time required for the I-cache to send back one instruction when this instruction is present in the I-cache (cache hit).

The hit time falls under one of the following scenarios:

1) An initial request takes the first wait state.
2) A second request that is issued immediately after the first returns in 0 wait states.
3) Subsequent requests that are consecutive and that are to sequential addresses return in 0 wait states.
4) Subsequent requests that are not consecutive or do not fall in a sequential address pattern are regarded as initial requests.

3.6.2 Miss Penalty

External memory accesses use burst reads to minimize latency for cache misses. All the instructions can be sent back in one cycle after a latency of several cycles for the first access.

The first access latency consists of:

- First cycle: Request is received.
- Fourth cycle: Request is forwarded to the EMIF after a cache miss.
- X + 4 cycles: External memory access through the EMIF. Dependent on memory speed and traffic controller loading.
- X+5 cycles: EMIF returns ready.
- X+6 cycles: I-cache returns ready.

- X+7 cycles: I-cache returns data.

All in all, X + 7 cycles are required for the first access.

If a second request is issued that is also a miss but lies in the present occurring line fill, those requests either fill with streaming as the data is returned by the EMIF or wait for the present line fill to complete before returning the data.

This latter occurs if the second request comes after the EMIF has already returned that data within the burst request.

3.7 Emulation Mode

3.7.1 Emulator Visibility

The emulator can read code from the I-cache during debug mode. The I-cache does not update its tag, data, LVB, and LRU for an emulator read. Only the corresponding code is returned to the emulator.

3.7.2 Software Breakpoint Detection

When a software breakpoint is set under emulation, the corresponding software performs the following operation:

- Multiple double reads to read original instruction
- A byte write (E_STOP0 code) to insert the software breakpoint
- Multiple double reads to check the change

The I-cache checks the byte writes performed by the emulator in the external memory space. If a byte write modifies an instruction in the external memory that is present in the I-cache, then the corresponding line valid bit is invalidated.

3.8 System Memory

The DSP has access to all system memory managed by the traffic controller. External memory space ranges from 0x28000 to 0xFF8000 if the internal PDROM is enabled, or to 0xFFFFFF if the PDROM is not enabled.

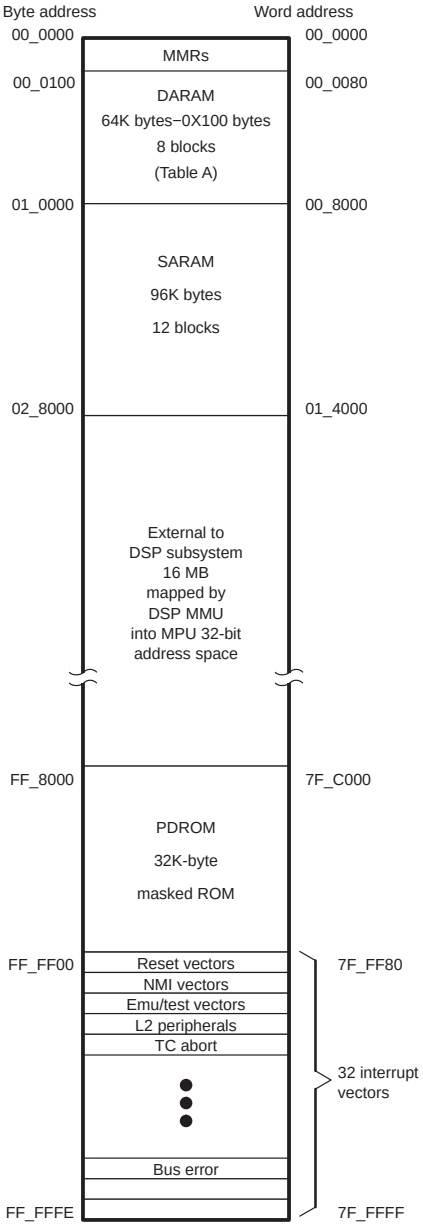
To access memory external to the DSP subsystem, the EMIF issues a memory access request. The access request is passed through the DSP memory management unit (MMU), which (if enabled and configured by the MPU) translates the DSP virtual address into a physical address that is passed to the traffic controller. The traffic controller completes the access through one of the

three system memory interfaces: internal memory (IMIF), slow external memory (EMIFS), or fast external memory (EMIFF). If the MMU is not enabled, then the access request is passed directly to the system traffic controller. In this case, the DSP virtual addresses are mapped to the first 16M bytes of CS0 of the system memory.

3.9 Memory Map

Figure 9 shows the DSP memory space.

Figure 9. DSP Memory Space



The TMS320C55x DSP has 24-bit unified address space for both data and program references.

* Program accesses are specified and displayed as byte addresses.
* Data objects are word addressable and are specified by 16-bit word addresses.
* The DMA controller references byte addresses.

To access control and data registers associated with various OMAP5912 peripherals, the DSP uses 16-bit I/O space. This space is referenced by using appropriate I/O access qualifiers with load or store instructions.

Restrictions apply for data objects spanning 64K-word (128K-byte) boundaries. See TMS320C55x DSP CPU Reference Guide (SPRU371).

Table A DARAM block boundaries

	Byte address	Word address
daram0	00 _0000	00 _0000
daram1	00 _2000	00_1000
daram2	00 _4000	00_2000
daram3	00 _6000	00_3000
daram4	00 _8000	00_4000
daram5	00 _A000	00_5000
daram6	00 _C000	00_6000
daram7	00 _E000	00_7000

Table B SARAM block boundaries

	Byte address	Word address
saram0	01 _0000	00_8000
saram1	01 _2000	00_9000
saram2	01 _4000	00_A000
saram3	01 _6000	00_B000
saram4	01 _8000	00_C000
saram5	01 _A000	00_D000
saram6	01 _C000	00_E000
saram7	01 _E000	00_F000
saram8	02 _0000	01_0000
saram9	02 _2000	01_1000
saram10	02 _4000	01_2000
saram11	02 _6000	01_3000

Note: Byte addresses 0xFF8000-0xFFFFFF map to PDROM for mpnmc = 0; otherwise, this range is mapped externally.

Table 11. I/O Spaces

Byte	Name	Word
00000	TIPB bridge	00000
00800	EMULATOR/TEST	00400
01000	STIO(EMIF)	00800
01800	DMA	00C00
02000		01000
02800	ICACHE	01400
03000		01800
03800		01C00
04000	TRACE	02000
04800	Configurable (private/shared)	02400
05000	TIMER 1	02800
05800	TIMER 2	02C00
06000	TIMER 3	03000
06800	WD_TIMER	03400
07000	Interrupt priority	03800
07800	Private (reserved)	03C00
08000	CLK_M	04000
08800	Configurable (private/shared)	4400
09000	L2 interrupt handler 2.0	04800
09800	L2 interrupt handler 2.1	04C00
0A000		05000
0A800		05400
0B000		05800
0B800 to 0F800		
10000	UART1	08000
10800	UART2	08400

Table 11. I/O Spaces (Continued)

Byte	Name	Word
10C00	SPI	08600
11000	McBSP2	08800
11400	GPTIMER1	08A00
11800	McBSP1	08C00
11C00	GPTIMER2	08E00
12000	MCSI2	09000
12400	GPTIMER3	09200
12800	MCSI1	09400
12C00	GPTIMER4	09600
13000		09800
13400	GPTIMER5	09A00
13800	I^2C multimaster	09C00
13C00	GPTIMER6	09E00
14000		0A000
14800		0A400
15000		0A800
15800		0AC00
16000		0B000
16800		0B400
17000	McBSP3	0B800
17400	GPTIMER7	0BA00
17C00	MMC/SDIO2	0BE00
18000		0C000
18400		0C200
18800		0C400
18C00		0C600

Table 11. I/O Spaces (Continued)

Byte	Name	Word
19000		0C800
19400	GPIO3	0CA00
19800	UART3	0CC00
19C00	GPIO4	0CE00
1A000		0D000
1A400	(reserved)	0D200
1A800		0D400
1B000		0D800
1B800		0DC00
1C000		0E000
1C400	32-kHz synchronization timer	0E200
1C800	OMAP5912 TIPB switch	0E400
1CC00		0E600
1D000		0E800
1D400	GPTIMER8	0EA00
1D800		0EC00
1E400	GPIO1	0F200
1EC00	GPIO2	0F600
1F000	Mailbox	0F800
1F800	DSP MPUI register	0FC00

3.10 Peripheral Register Addresses

The DSP CPU and the DMA controller can access several classes of peripheral devices:

- DSP private peripherals
  - Three general-purpose timers
  - A watchdog timer

- An interrupt handler

❑ MPU/DSP shared peripherals

- SPI
- I^2C
- MMCSDIO2
- GPIO (x4)

❑ DSP public peripherals

- Two multichannel buffered serial ports (McBSPs) for synchronous serial communications
- Two multichannel serial interfaces (MCSIs)

Configuration and data registers for all peripherals reside in the DSP subsystem I/O space, which consists of 64K-word addresses, with each peripheral mapping into a 1K-word section of I/O memory. To read or write these registers, you must access the DSP I/O space either through C language constructs or by using the assembly language peripheral port register access qualifier. See *TMS320C55x DSP Mnemonic Instruction Set Reference Guide* (SPRU374D) for more details.

Table 12 shows the DSP peripheral mapping.

Table 12. DSP Peripheral Mapping

Start Byte Address (hex)	Name	CS	Strobe
x 000000	TIPB bridge	0	Strobe0
x 000800	EMULATOR/TEST	1	Stroben
x 001000	STIO(EMIF)	2	Stroben
x 001800	DMA	3	Stroben
x 002000	Reserved	4	
x 002800	ICACHE	5	Stroben
x 003000	Reserved	6	
x 003800	Reserved	7	
x 004000	TRACE	8	Strobe0

† All other I/O memory addresses are reserved.
‡ Internal wait states for accessing peripherals are set by strobe fields in TIPB CM register (see Section 4.1, *Control Mode Register*).

Table 12. DSP Peripheral Mapping (Continued)

Start Byte Address (hex)	Name	CS	Strobe
x 004800	Configurable (private/shared)	9	Strobe0/n
x 005000	TIMER 1	10	Stroben
x 005800	TIMER 2	11	Stroben
x 006000	TIMER 3	12	Stroben
x 006800	WD_TIMER	13	Stroben
x 007000	DSPINT IF	14	Stroben
x 007800	Private	15	
x 008000	CLK_M2	16	Strobe0
x 008800	Configurable (private/shared)	17	Strobe0/n
x 009000	Level 2 interrupt handler 2.0	18	Strobe0
x 009800	Level 2 interrupt handler 2.1	19	Strobe0
x 00A000	Reserved	20	
x 00A800	Reserved	21	
x 00B000	Reserved	22	
x 00B800 to x 00F800	Reserved	23 to 31	
X01 0000	UART1	0	Strobe1
X01 0800	UART2	1	Strobe1
X01 0C00	SPI	1	Strobe1
X01 1000	McBSP2	2	Strobe1
X01 1400	GPTIMER1	2	Strobe1
X01 1800	McBSP1	3	Strobe1
X01 1C00	GPTIMER2	3	Strobe1
X01 2000	MCSI2	4	Strobe1
X01 2400	GPTIMER3	4	Strobe1

† All other I/O memory addresses are reserved.
‡ Internal wait states for accessing peripherals are set by strobe fields in TIPB CM register (see Section 4.1, *Control Mode Register*).

Table 12. DSP Peripheral Mapping (Continued)

Start Byte Address (hex)	Name	CS	Strobe
X01 2800	MCSI1	5	Strobe1
X01 2C00	GPTIMER4	5	Strobe1
X01 3000	Reserved	6	Strobe1
X01 3400	GPTIMER5	6	Strobe1
X01 3800	I^2C multimaster	7	Strobe1
X01 3C00	GPTIMER6	7	Strobe1
X01 4000	Reserved	8	Strobe1
X01 4800	Reserved	9	Strobe1
X01 5000	Reserved	10	Strobe1
X01 5800	Reserved	11	Strobe1
X01 6000	Reserved	12	Strobe1
X01 6800	Reserved	13	Strobe1
X01 7000	McBSP3	14	Strobe1
X01 7400	GPTIMER7	14	Strobe1
X01 7C00	MMC/SDIO2	15	Strobe1
X01 8000	Reserved	16	Strobe1
X01 8400	Reserved	16	Strobe1
X01 8800	Reserved	17	Strobe1
X01 8C00	Reserved	17	Strobe1
X01 9000	Reserved	18	Strobe1
X01 9400	GPIO3	18	Strobe1
X01 9800	UART3	19	Strobe1
X01 9C00	GPIO4	19	Strobe1
X01 A000	Reserved	20	Strobe1

† All other I/O memory addresses are reserved.
‡ Internal wait states for accessing peripherals are set by strobe fields in TIPB CM register (see Section 4.1, *Control Mode Register*).

Table 12. DSP Peripheral Mapping (Continued)

Start Byte Address (hex)	Name	CS	Strobe
X01 A400	(reserved)	20	Strobe1
X01 A800	Reserved	21	Strobe1
X01 B000	Reserved	22	Strobe1
X01 B800	Reserved	23	Strobe1
X01 C000	Reserved	24	Strobe1
X01 C400	32-kHz synchronization timer	24	Strobe1
X01 C800	OMAP5912 TIPB switch	25	Strobe1
X01 CC00	Reserved	25	Strobe1
X01 D000	Reserved	26	Strobe1
X01 D400	GPTIMER8	26	Strobe1
X01 D800	Reserved	27	Strobe1
X01 E400	GPIO1	28	Strobe1
X01 EC00	GPIO2	28	Strobe1
X01 E800	Reserved	29	Strobe1
X01 F000	Mailbox	30	Strobe1
X01 F800	MGS3 MPUI control register	31	Strobe1

† All other I/O memory addresses are reserved.
‡ Internal wait states for accessing peripherals are set by strobe fields in TIPB CM register (see Section 4.1, *Control Mode Register*).

4 TIPB Bridge

The TIPB bridge module manages access to peripheral control and data registers by the DSP CPU, DSP DMA controller, and MPUI via two peripheral buses (see Figure 10):

- Private TIPB: peripherals connected here (timers and interrupt handler) cannot be accessed by the MPU via the MPUI.
- Public TIPB: peripherals connected here (McBSP1, McBSP2, MCSI1, MCSI2, mailbox, and GPIO UART1-3) can be accessed by the MPU via the MPUI port.

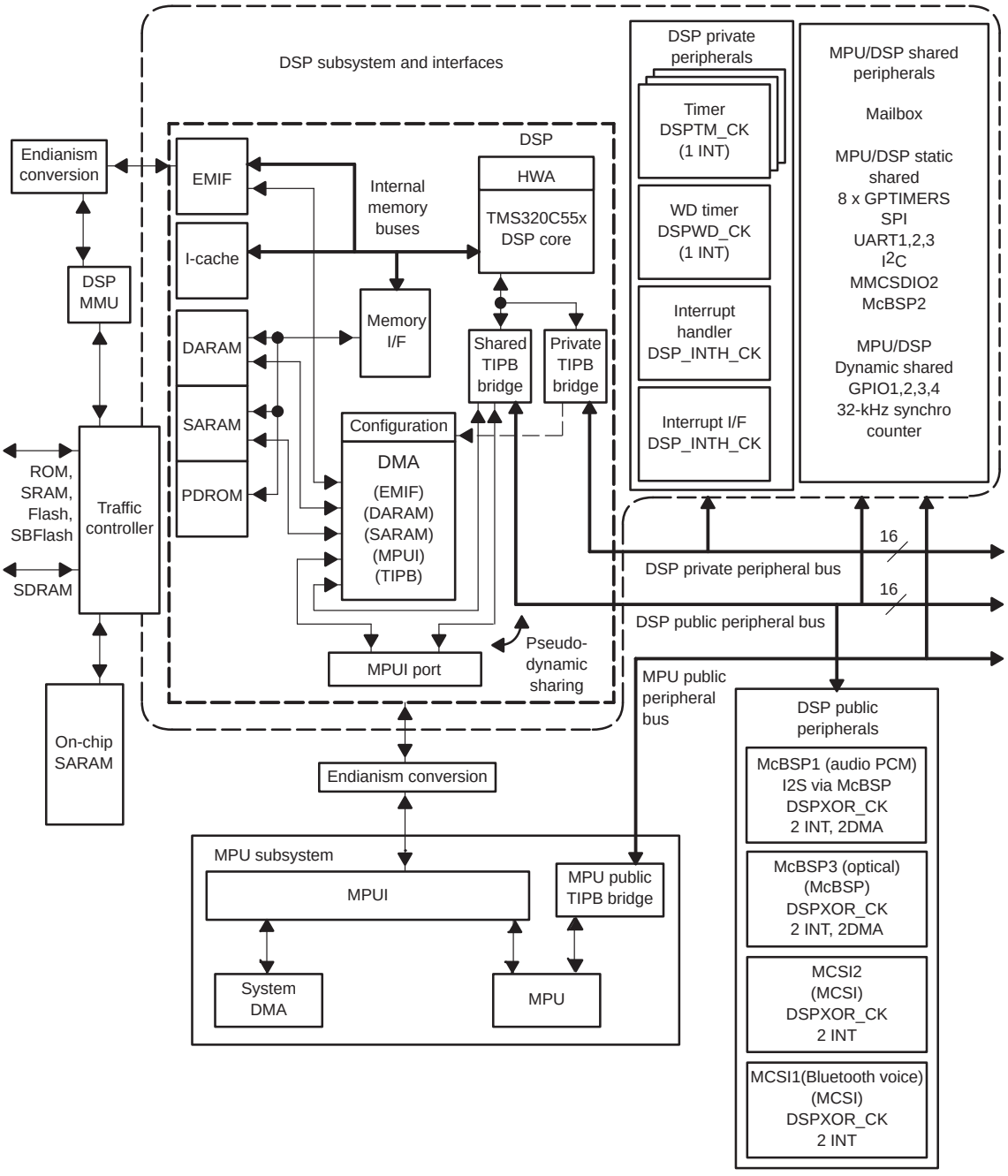
The TIPB bridge consists of two components:

- The private TIPB bridge provides a preconfigured bus interface to peripherals residing on the the DSP private TIPB.
- The public TIPB bridge provides a user-configurable interface to peripherals on the DSP public TIPB. It includes functions to tailor the interface timing to the complement of peripherals operating at a given time.

The TIPB bridge also contains registers to control and monitor the DSP subsystem idle state. The DSP TIPB bridge can be configured using the following registers in DSP I/O space:

- Control mode register (CMR): DSP I/O word address is 0x0000.
- Idle control register (ICR): DSP I/O word address is 0x0001.
- Idle status register (ISTR): DSP I/O word address is 0x0002.

Figure 10. DSP Subsystem Modules



4.1 Control Mode Register

The control mode register (CMR) indicates the shared-access mode/host-only mode (SAM/HOM) status of the MPUI and bus error condition status for accesses to the TIPB bridge. It also controls CPU priority versus the MPUI and DMA for accesses to peripherals on the TIPB bridge.

Table 13. Control Mode Register (CMR)—Value at Reset is 0xFE4D

CMR [15−0]	Designation	Description	Reset Value	CPU Access	MPU Access
15−9	Time-out (6:0)	Strobe cycles (0-127)	0x7F	Read/Write	Read
8−6	Wait state (strobe 1)	Strobe1 length (low, medium, high bits)	0	Read/Write	Read
5−3	Wait state (strobe 0)	Strobe 0 length (low, medium, high bits)	1	Read/Write	Read
2	CPU priority	Priority modes	1	Read/Write	Read
1	Bus error	Application flag error	0	Read/Clear	Read (0 in HOM)
0	Mode	SAM or HOM	1 (HOM)	Read	Read

- Mode bit

  This bit is a read-only indication of whether the MPUI is in host-only mode (HOM) or in single-access mode (SAM). HOM and SAM are described in Section 3.6, *External Memory Interface*.

- Bus error

  This bit is set to 1 if the TIPB bridge generates a bus error (because of a time-out condition or SAM/HOM change error), indicating that an error signal that can read this bit to identify the source of the error condition has been sent to the DSP CPU. The bit is cleared upon read by the DSP CPU. This bit cannot be read during HOM (always registers as zero during HOM).

- CPU priority bit

  When CPU_PRIORITY = 1, the DSP subsystem CPU, MPUI, and DMA have the following priority in arbitration of TIPB bridge accesses:

  1) CPU

  2) MPUI

  3) DMA

  If CPU_PRIORITY = 0, the CPU, MPUI, and DMA accesses to the TIPB bridge are arbitrated in rotating priority fashion.

- Wait state bits for strobe 0 and strobe 1

  The strobe 0 field sets the access rate for the following peripherals:

  - TIPB registers
  - CLKM2 registers
  - DSP interrupt handler 2.0
  - DSP interrupt handler 2.1

  The strobe 1 field sets the access rate for the following peripherals:

  - UART3 (test)
  - McBSP1 (audio PCM)
  - McBSP3 (optical)
  - MCSI-1
  - MCSI-2
  - GPIO
  - Mailbox
  - DSP MPUI register
  - OMAP5912 TIPB switch
  - GP timer (x8)
  - 32-kHz synchronization timer
  - SPI
  - I^2C
  - MMCSDIO2
  - GPIO (x4)

The control mode register bits [5−3] and [8−6] contain the number of wait states required to generate the appropriate strobe frequency (see Table 14).

Table 14. Wait States

Number of Wait States	Strobe Period
0	DSP clk/2
1	DSP clk/3
2	DSP clk/4
3	DSP clk/5
4	DSP clk/6
5	DSP clk/7
6	DSP clk/8
7	DSP clk/9

- Time-out[6:0]

This field specifies the number of cycles that can elapse before the TIPB returns a bus error condition. The seven-bit field specifies the number of wait states. The time-out period is determined as

Time-out = value of time out[6:0] + 2 measured in DSP subsystem master clock cycles

The default value is 0x7f (127).

4.2 Idle Control and Idle Status Registers

To conserve power, the DSP subsystem is capable of idling certain circuits. The DSP CPU and peripherals contain several clock domains that can be turned off individually to conserve power. The active/idle status of the various domains is controlled by the idle control register. When the DSP software executes the IDLE instruction, the clock domains are configured according to the settings of the idle control register (ICR) (see Table 15). The current idle domain status is reflected by the state of the idle status register (ISTR) (see Table 16).

The idle domains are:

0 CPU
1 DMA
2 Cache
3 Peripherals
4 DPLL

5 EMIF

The DSP DPLL is controlled by the MPU subsystem. When entering low-power mode requiring DSP DPLL off, the DSP sets DPLL idle domain on, followed by the MPU idling the DPLL source by writing the appropriate control registers (see Chapters 4 and 5).

Table 15. Idle Configuration Register (ICR)

ICR [15−0]	Description	DSP Access	MPU Access	Reset Value
15−8	Reserved (not connected)	Read	Read	0x0
7	Reserved idle domain	Read/Write	Read	0
6	Reserved idle domain	Read/Write	Read	0
5	EMIF idle domain	Read/Write	Read	0
4	DPLL idle domain	Read/Write	Read	0
3	Peripherals idle domain	Read/Write	Read	0
2	Cache idle domain	Read/Write	Read	0
1	DMA idle domain	Read/Write	Read	0
0	CPU idle domain	Read/Write	Read	0

Note: When the DSP subsystem comes out of IDLE, the ICR configuration is retained until modified by the CPU. The next time an IDLE instruction is executed, the same domains enter the idle state.

Table 16. Idle Status Register (ISTR)

ISTR[15−0]	Description	DSP Access	MPU Access	Reset Value
15−8	Not connected	Read	Read	0x0
7	Reserved idle status	Read	Read	0
6	Reserved idle status	Read	Read	0
5	EMIF idle status	Read	Read	0
4	DPLL idle status	Read	Read	0
3	Peripherals idle status	Read	Read	0
2	Cache idle status	Read	Read	0
1	DMA idle status	Read	Read	0
0	CPU idle status	Read	Read	0

The DSP must not attempt to read the ISTR while the DPLL domain is idled because this causes a time-out error.

5 MPU Interface

The MPU interface (MPUI) is a 16-bit parallel port that allows the MPU and the system DMA controller to communicate with the DSP and its peripherals, facilitating software downloads and data transfers. For additional information, please see *OMAP5912 Multimedia Processor OMAP3.2 Subsystem Reference Guide* (literature number SPRU749).

The MPUI provides the MPU with access to the full memory space of the DSP (16M bytes). In addition, the MPUI allows the MPU to access devices on the DSP public peripheral bus through duplicate memory-mapped peripheral registers in the MPU address space. The MPU domain can also access the control registers of the TIPB bridge module and the CLKM2 configuration registers. The DSP private peripherals are not accessible via the MPUI.

MPUI transfers are facilitated by an auxiliary channel of the DSP subsystem DMA controller; however, this dedicated DMA channel is preconfigured and need not be user-configured for MPUI support.

The MPU domain (including ARM926EJS and system DMA) always masters the transfer operation. It initiates the read or write of DSP memory or peripherals. The MPU also controls the parameters of the MPUI by configuring the MPUI_CTRL_REG and the MPUI_DSP_MPUI_CONFIG register. There are five additional registers the MPU can read to observe the state of the MPUI:

- MPUI_DEBUG_ADDR
- MPUI_DEBUG_DATA
- MPUI_DEBUG_FLAG
- MPUI_STATUS_REG
- MPUI_DSP_STATUS_REG

The MPUI port supports four access modes:

- Single-access mode, memory (SAM_M): SARAM, DARAM and, external memory interface are shared between the DSP domain and the MPU domain.
- Single-access mode, peripheral (SAM_P): DSP public peripheral bus is shared between the DSP domain and the MPU domain.
- Host-only mode, memory (HOM_M): MPU has exclusive access to DSP SARAM, but it cannot access other DSP memory resources.
- Host-only mode, peripheral (HOM_P): MPU has exclusive access to the DSP public peripheral bus.

SAM is the normal operating mode in which all the DSP internal memory and the public peripherals are accessible by the MPUI interface as well as the DSP. If both the DSP and the MPU controllers (ARM926EJS and/or system DMA) access the same memory at the same time, priority is given to the DSP controllers. The MPU domain access in SAM is synchronized to the internal DSP CPU clock, which can add access latency for the MPU transfers.

HOM provides the MPU with exclusive access to the DSP SARAM or public peripherals, primarily to support high-speed transfers from/to DSP during DSP reset or IDLE conditions. During DSP reset condition, HOM_M and HOM_P are invoked. In HOM_M the MPUI interface does not have access to the DARAM (0x000000−0x00FFFF), but it has access to all the SARAM (0x010000−0x050000). The MPU must configure the MPUI_DSP_MPUI CONFIG register to specify which blocks of SARAM are accessible in HOM before access, because the reset default is for no SARAM access during HOM_M.

An additional condition is that in HOM_P only the MPU can access the DSP peripheral bus.

5.1 HOM/SAM Change Outside of Reset

Only the DSP can invoke a HOM/SAM change outside of reset. The mode change is initiated by a DSP write to HOM_P bit (bit 8) and HOM_R bit (bit 9) of the ST3 register. The appropriate bit is written to request the SAM_M/HOM_M or SAM_P/HOM_P change. The mode change is not reflected on bits 8 and 9 in ST3 until the internal controller completes the mode switch. Therefore, the DSP polls bits 8 and 9 after requesting a mode change to ensure that the mode change is complete.

The HOM_M/SAM_M and HOM_R/SAM_R status can be observed by the MPU by reading the MPU_DSP_Status_Register.

5.2 ST3—HOM_P Bit (Bit 8)

The host-only mode for peripherals (HOM_P bit) determines whether the peripherals are owned only by the MPU or shared by the MPU and the DSP:

- 0: Off

  Peripherals are shared by the MPU and the DSP. If you clear the HOM_P bit, a request for sharing is sent to the peripheral domain controller. If the peripheral domain controller clears the HOM_P bit, the clearing indicates that the MPU no longer has exclusive ownership of the peripherals.

- 1: On

  Peripherals are owned only by the MPU. If you set the HOM_P bit, a request for HOM is sent to the peripheral domain controller. If the peripheral domain controller sets the HOM_P bit, the setting indicates that the MPU has exclusive ownership of the peripherals.

5.3 ST3—HOM_R Bit (Bit 9)

The MPUI RAM is the portion of the DSP RAM that is accessible by the MPUI. The HOM_R bit determines/shows whether the MPUI RAM is owned only by the MPUI or shared by the host processor and the C55x DSP:

- 0: Off

  The MPUI RAM is shared by the host processor and the DSP. If you clear the HOM_R bit, a request for sharing is sent to the MPUI. If the MPUI clears the HOM_R bit, the clearing indicates that the MPU no longer has exclusive ownership of the MPU RAM.

- 1: On

  The MPUI RAM is owned only by the host processor. If you set the HOM_R bit, a request for host-only mode is sent to the MPUI. If the MPUI sets the HOM_R bit, the setting indicates that the host processor has exclusive ownership of the MPU RAM.

6 External Memory Interface

The external memory interface (EMIF) is a DSP subsystem module that gives the DSP access to the shared system memory managed by the traffic controller. The EMIF interfaces directly to a 32-bit-wide system bus. This bus can operate at the CPU clock rate with sustained throughput during burst accesses. The EMIF has two control registers for user configuration:

- EMIF global control register (GCR)
- EMIF global reset register (GRR)

EMIF does not have support for 8 bit DMA read line.

6.1 EMIF Global Control Register

The EMIF global control register (EMIF_GCR) configures the general operation of the EMIF module. The EMIF GCR appears at word address 0x0800 in the DSP I/O space.

Table 17. EMIF Global Control Register (EMIF GCR)

Bit	Name	Function	Type	Reset Value
15−12	Reserved		R	0
11−8	Reserved		RW	0
7	WPE	Write posting enable WPE=0, write posting is disabled (for debug). WPE=1, write posting is enabled.	RW	0
6	Reserved		RW	0
5	Reserved		RW	1
4	Reserved		R	0
3	Reserved		R	x
2	Reserved		R	x

Table 17. EMIF Global Control Register (EMIF GCR) (Continued)

Bit	Name	Function	Type	Reset Value
1	Reserved		R	0
0	Reserved		RW	0

6.2 EMIF Global Reset Register

Any write in the EMIF global reset register (EMIF_GRR) register causes a software reset of the EMIF state machines. This register cannot be read. A software reset does not change the current configuration register values (EMIF_GCR, and so on); only the EMIF state machines are reset. The EMIF GRR appears at word address 0x0801 in the DSP I/O space.

7 DSP Memory Management Unit

The DSP MMU maps the 16M bytes of the DSP virtual external addresses to anyplace in the 4G-byte address space of the OMAP5912 device. At reset the MMU is disabled and the DSP external memory space is mapped to the first 16M bytes of CS0 system memory.

The DSP MMU translates the 24-bit DSP external addresses (028000 to FF8000 or FFFF00) 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 of the DSP byte address onto another set of most significant bits of a 32-bit MCU byte address. The least significant bits of the DSP-generated byte address are not altered when forming the new address. The TTB translations are expedited by a cache-like translation look-aside buffer mechanism (TLB). The 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 1M-byte, 64K-byte, 4K-byte, or 1K-byte segments of memory.

The DSP MMU is programmed by the ARM926EJS. In general, the MMU is initialized at boot time, but it also can be reprogrammed dynamically. The MMU is programmed through the TIPB registers. DSP MMU registers have an MPU base address of 0xFFFE:D200.

7.1 Description

The DSP MMU is in the TC clock frequency domain (i.e. runs as fast as the TC_CK). The DSP memory management unit (MMU) supports memory mapping for the DSP. The platform system software can relocate regions of the DSP logical address space via a page table and the DSP MMU.

The DSP MMU contains a 32-entry translation lookaside buffer (TLB) that holds translations and permissions for current pages. This TLB is often managed statically by the MPU OS; but the MMU also includes hardware table walking logic, as in the MPU, to autonomously traverse the page table on a TLB miss. In this case, the TLB can be seen as a cache of recently used page table entries.

The format of the page table and TLB entries of the DSP match those of the MPU.

The DSP subsystem internal memory is analogous to the DSP cache; it is directly addressed by the processor (logically addressed), and therefore no translation is carried out. The one difference between the DSP and MPU subsystems is that within the DSP subsystem no permission checks are carried out when the DSP accesses its internal memory.

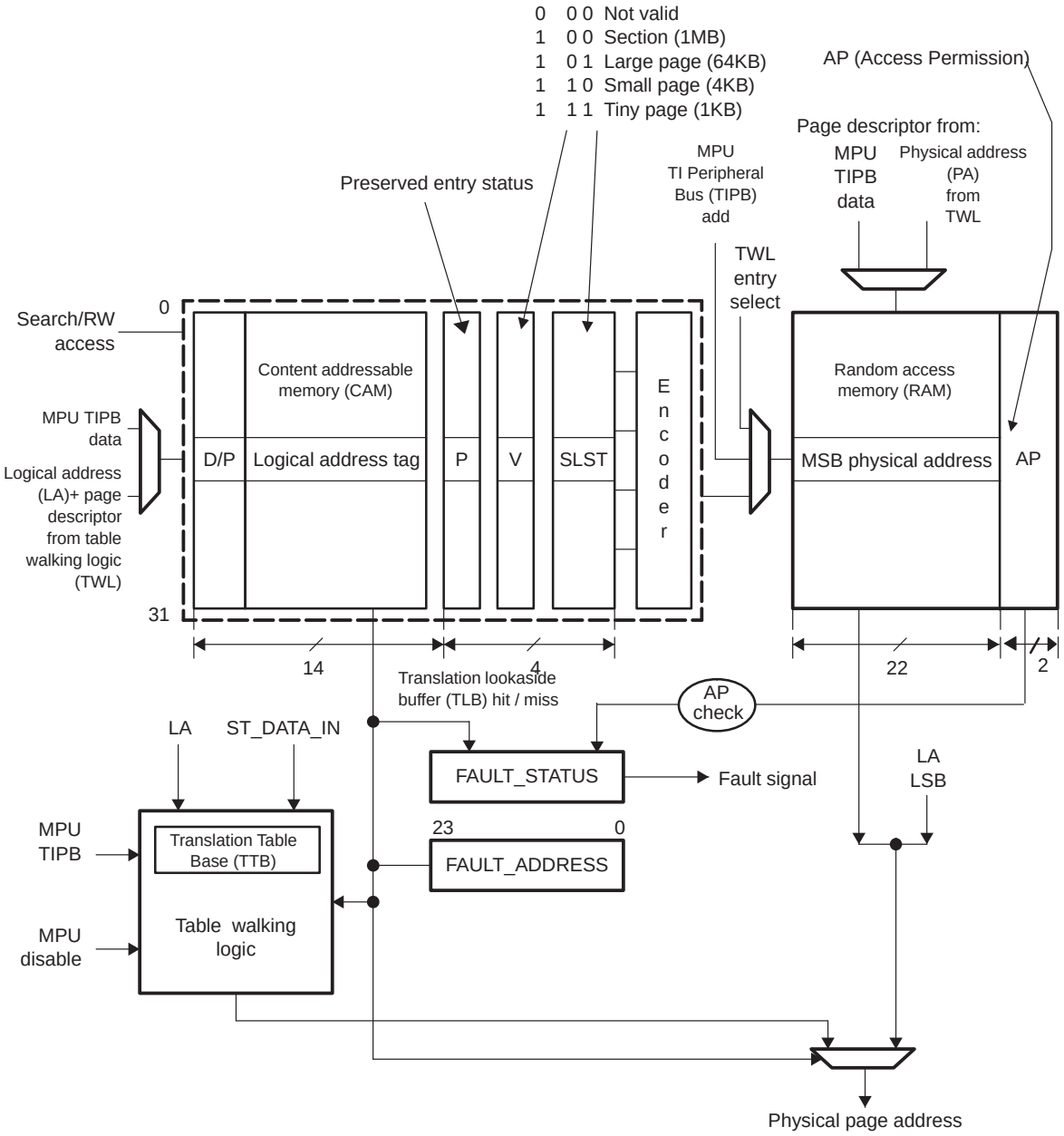
If the DSP MMU requires software intervention, the MPU is responsible for servicing the event; DSP MMU errors are signaled to the MPU with a dedicated interrupt. The DSP MMU is programmed by the MPU.

The main goals of the MMU are:

- To translate the DSP internal (logical) addresses into OMAP (physical) addresses
- To prevent the DSP software from making invalid accesses to system memory

Figure 11 shows the relation between the major blocks of the MMU. Subsequent sections describe the functions of each of the blocks.

Figure 11. DSP MMU Architecture



7.2 Address Translation

The address translation from logical address to physical address is made using the translation lookaside buffer (TLB) contained in the MMU.

The MPU software typically loads the TLB entries of the MMU before enabling the MMU (Mmu_en bit in CNTL_REG).

When the MMU is disabled, no translation is done, the host addresses pass through untranslated, and no permission checks or table walking are performed.

The TLB contains the two embedded memories.

- CAM

  Each entry contains the logical address tag, the preserved bits, valid bits, and page size.

- RAM

  Each entry contains the upper part of the associated physical address and the access protection field.

The MPU can access the CAM and the RAM when hardware table walking logic is disabled.

7.2.1 Translation Process

This section includes a brief introduction to MMU behavior. For more details, see the MPU subsystem documentation.

The following page sizes are supported:

- Section: 1M byte
- Large page: 64K bytes
- Small page: 4K bytes
- Tiny page: 1K byte

The page size and the upper bits of the input (logical) address are used to index the TLB CAM. Assuming there is a match (hit) in the CAM, the upper bits of the output (physical) address are read from the associated TLB RAM entry. These bits are then concatenated with the lower bits from the logical address. The boundary of translated and untranslated bits is a function of the page size.

Figure 12 shows an example of how the physical address is built with the logical address.

Figure 12. Address Translation Process for a Section

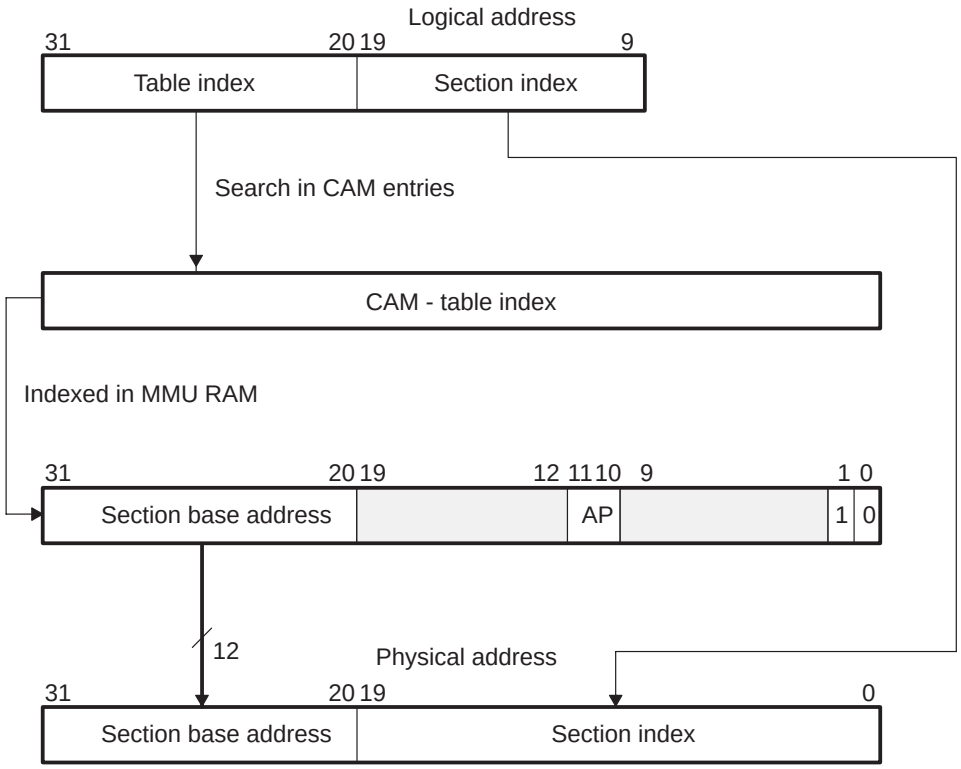
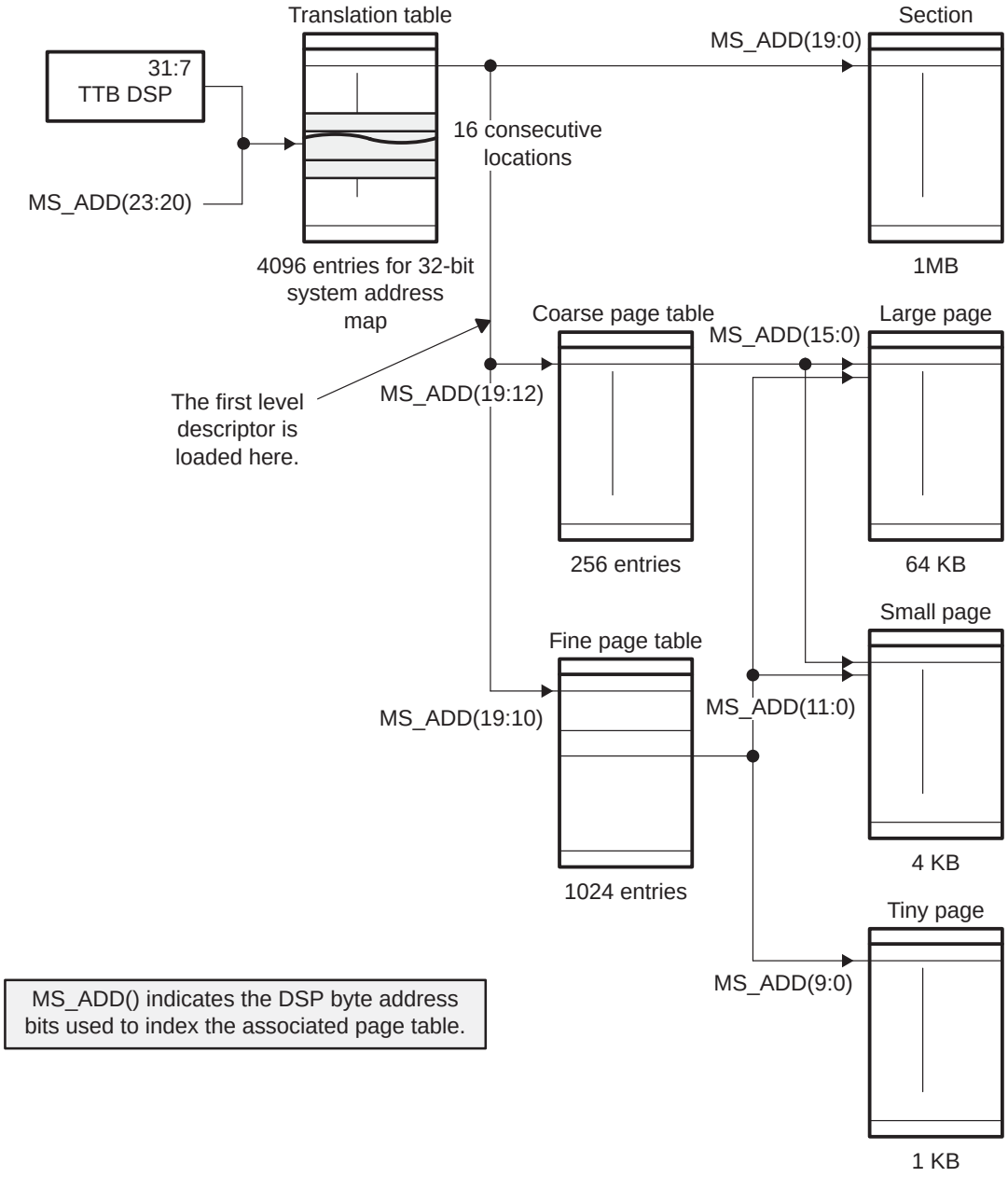


Figure 13 shows the translation table hierarchy. The physical address is built with the logical address.

As with the MPU, the page table may be hierarchical, as shown in Figure 13.

Figure 13. Translation Table Hierarchy



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

7.2.2 Page Table Format

Figure 14. Level One Descriptor

31:20	19:12	11:10	9:2	1	0	
				0	0	Fault
Coarse page table base address			R	0	1	Coarse page
Section base address		AP		1	0	Section
Fine page table base address		R		1	1	Fine page

Note: AP = Protection bits for the page.

Note: R indicates reserved bits to be written as 0 and read value to be ignored.

Figure 15. Level Two Descriptor

31:16	15:12	11:10	9:6	5:4	3:2	1	0	
						0	0	Fault
Large page table base address	R			AP	R	0	1	Large page
Small page table base address		R		AP	R	1	0	Small page
Tiny page table base address			R	AP	R	1	1	Tiny page

Note: AP = Protection bits for the page.

Note: R indicates reserved bits to be written as 0 and read value to be ignored.

Table 18. Page Protection Field

AP	Access	Description
0x	No access	Any access causes a permission fault.
10	Read only	Write causes a permission fault.
11	Full access	Read and write access allowed.

7.2.3 Coarse Page Tables

Coarse page tables have 256 entries and each entry describes 4K bytes. These entries provide a base address for either small or large pages. Large page descriptors must be repeated in 16 consecutive entries.

Figure 16 shows the translation for a section.

Figure 16. Translation for a Section

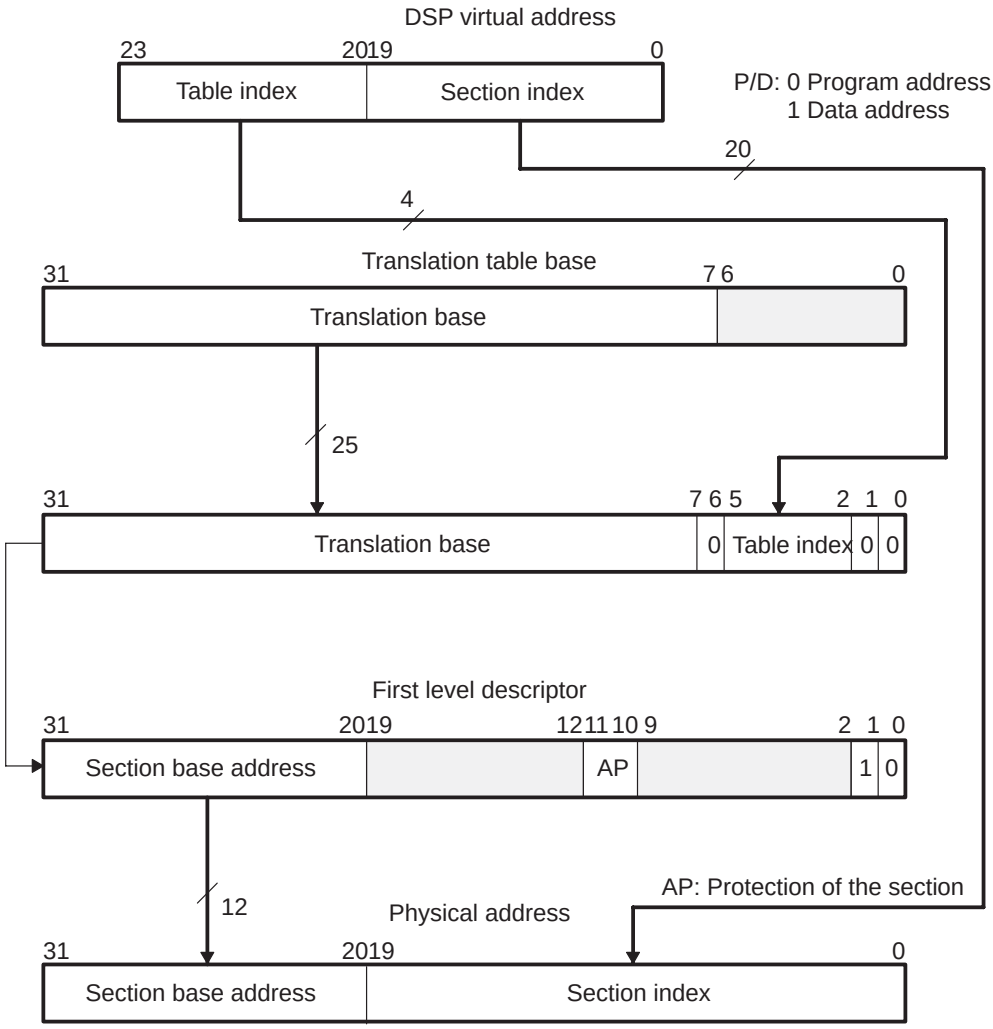


Figure 17 and Figure 18 show how the physical address is built as a function of the page sizes and hierarchy.

Figure 17. Translation for a Large Page Included in a Coarse Page

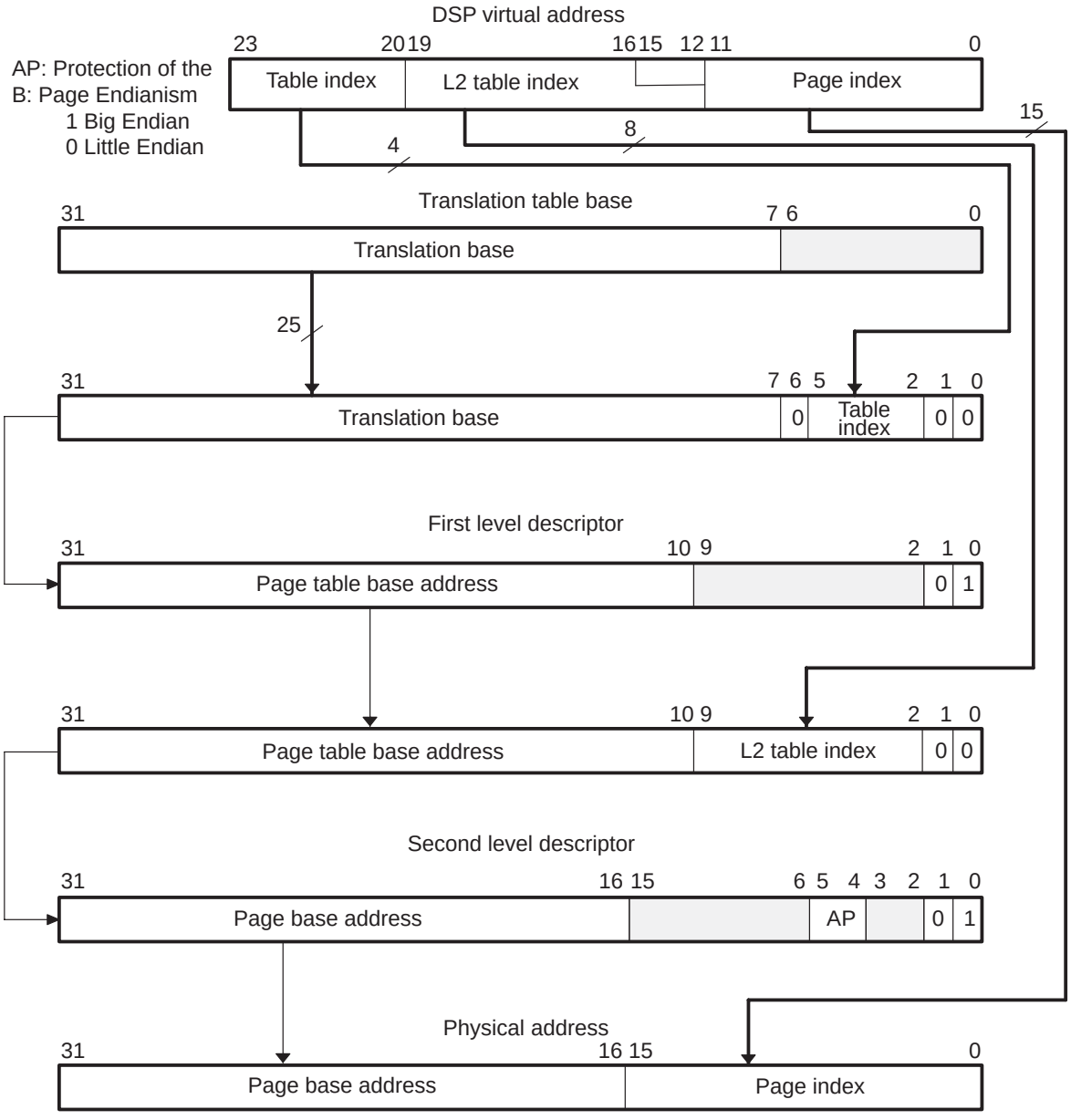
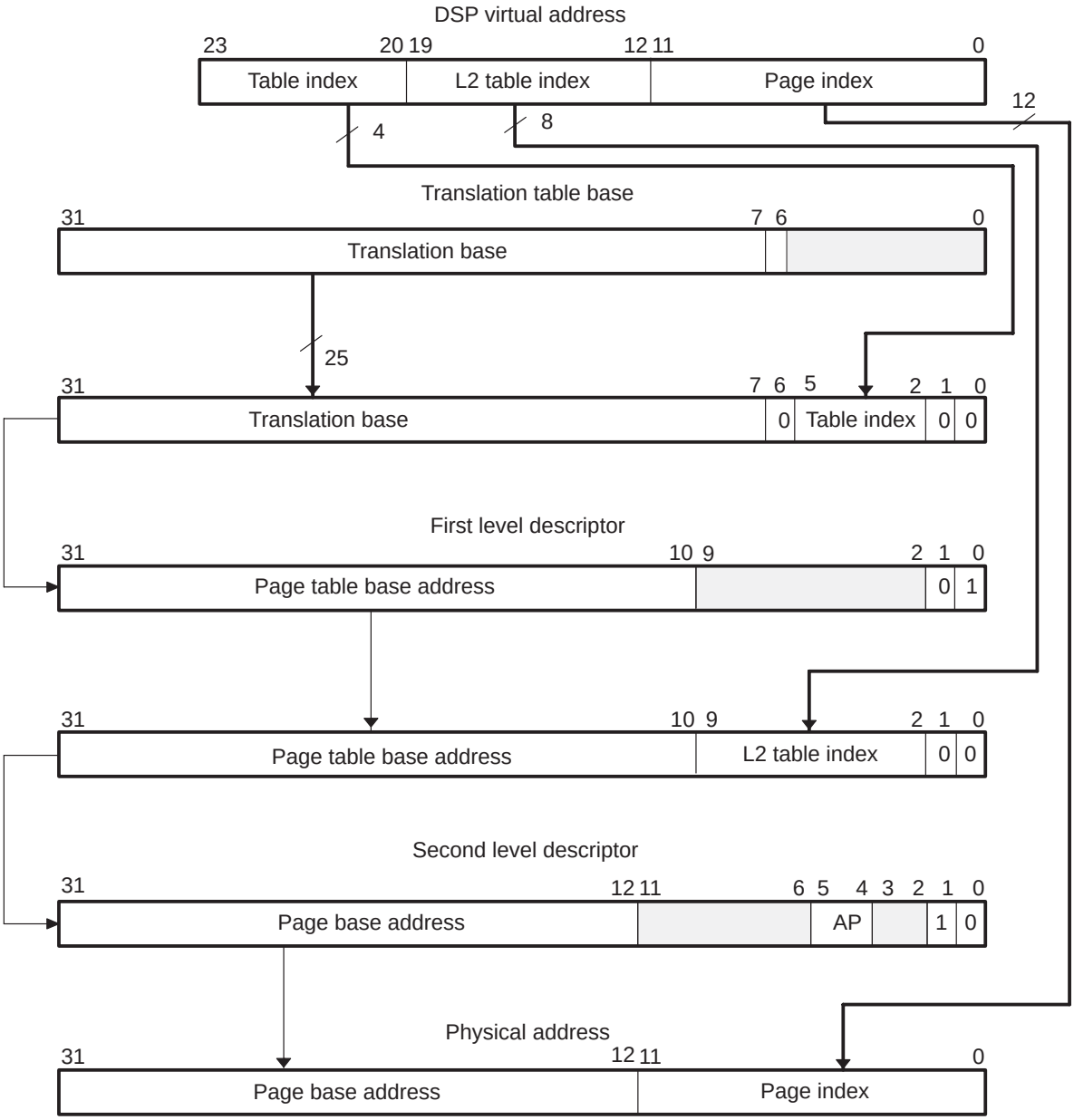


Figure 18. Translation for a Small Page Included in a Coarse Page



7.2.4 Fine Page Tables

Fine page tables have 1024 entries, and each entry describes 1K byte. These entries provide a base address for tiny, small, or large pages. Small page descriptors must be repeated in four consecutive entries. Large page descriptors must be repeated in 64 consecutive entries.

Figure 19, Figure 20, and Figure 21 show how the physical address is built as a function of the page sizes and hierarchy.

Figure 19. Translation for a Large Page Included in a Fine Page

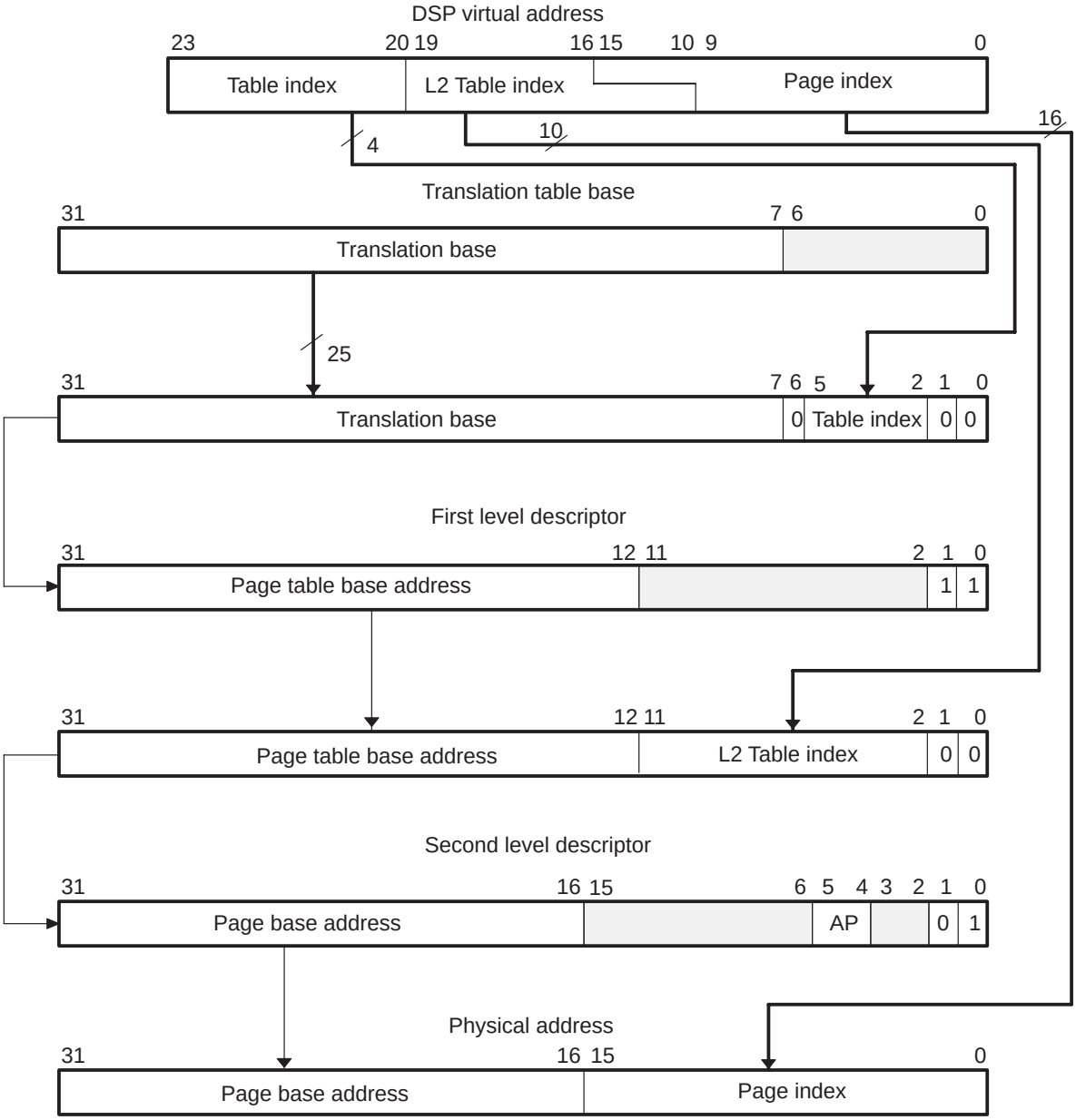


Figure 20. Translation for a Small Page Included in a Fine Page

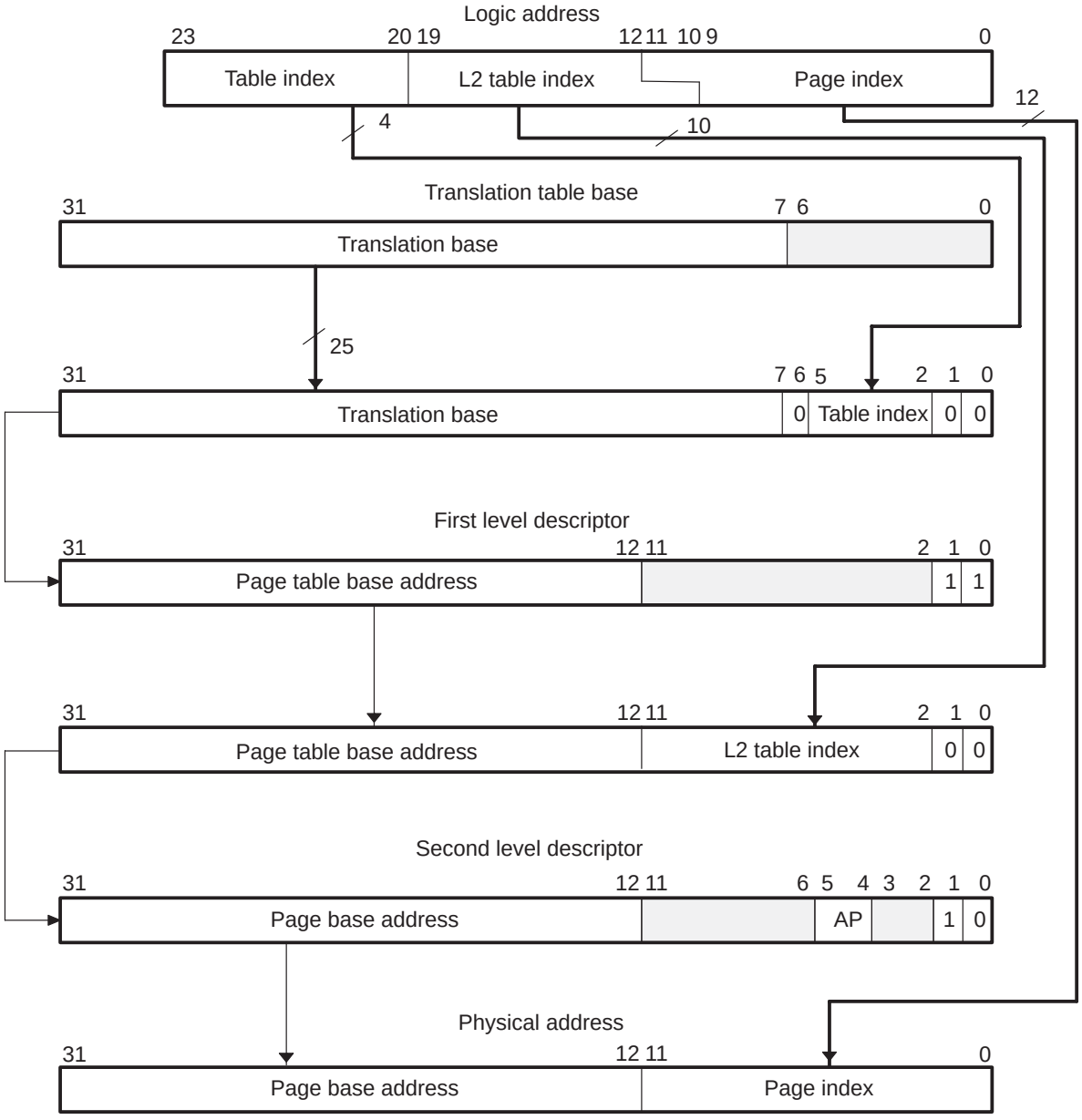
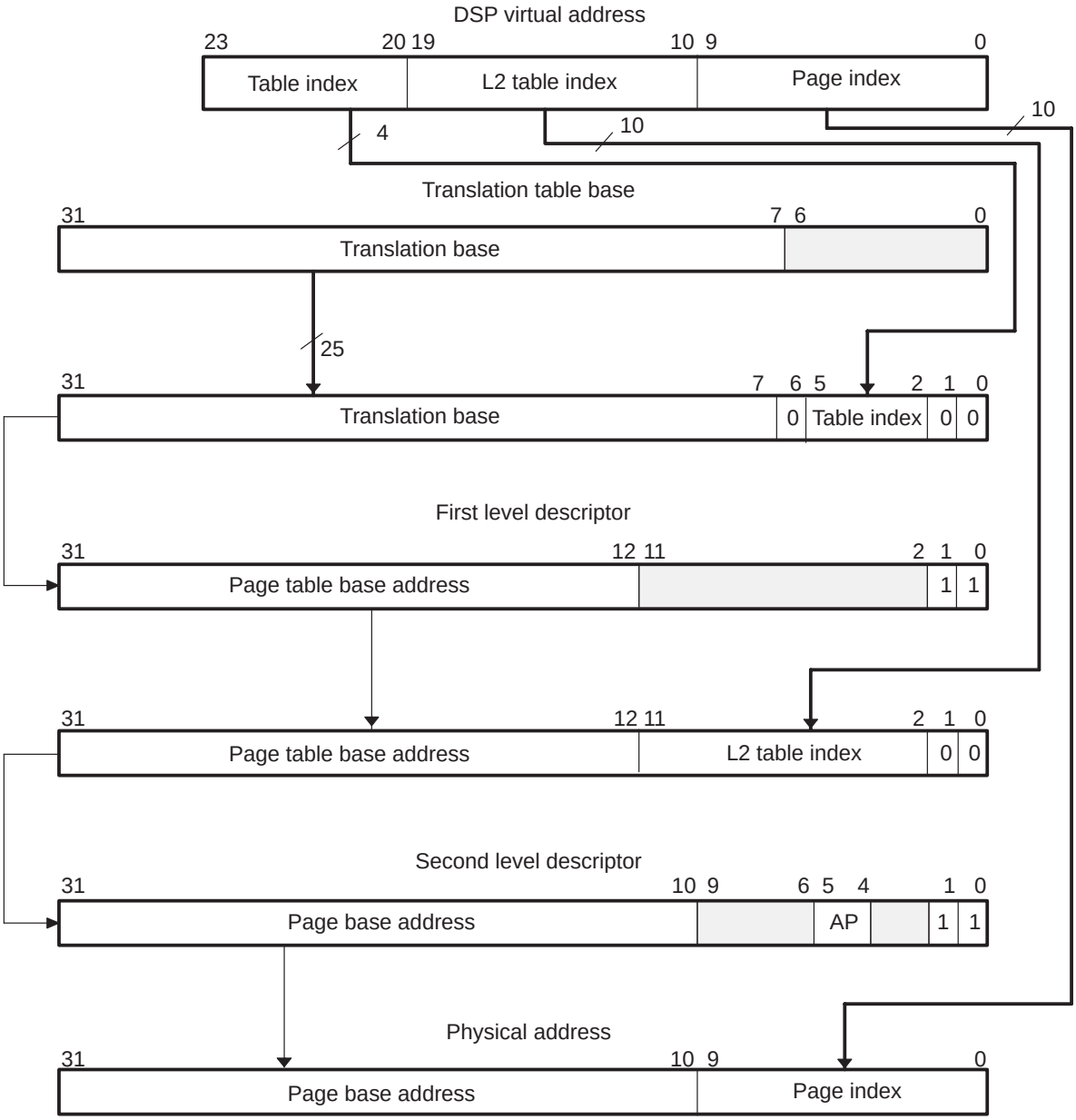


Figure 21. Translation for a Tiny Page Included in a Fine Page

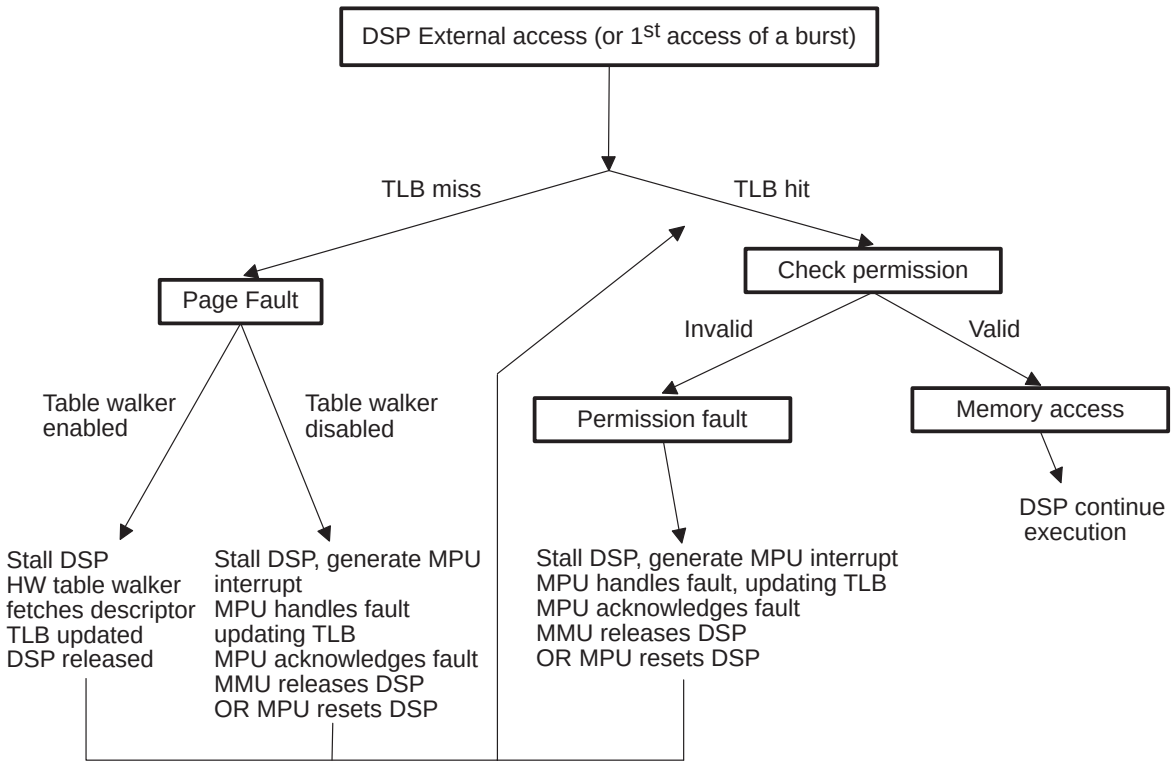


7.3 Functionality

7.3.1 Translation Summary

Figure 22 shows the possible results of a DSP memory request.

Figure 22. DSP Memory Request Results Example



7.3.2 Lock Mechanism and the Current_Victim Counter

Any TLB entry can be locked, but 31 entries is the maximum number that 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 Base_Value field of LOCK_REG is > 0, TLB entries from Base_value − 1 down to 0 are locked.

Frequently used memory areas or memory areas used by applications with critical real-time constraints must be loaded in these locations and the base_value counter must be set accordingly.

The current_victim counter of the LOCK_REG 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. All TLB entries can not be locked (a maximum of 31 entries can be locked).

7.3.3 Fault Handling

The following types of faults may occur:

- TLB miss with table walker disabled

  TLB entries can be locked from [Base_value−1] down to 0.

  No translation is found for the logical address required. If the hardware table walker is disabled, a fault is generated.

- Translation fault

  No translation is found for the logical address required (TLB miss). The hardware table walker is enabled but no page table entry exists for the requested address.

- Access permission fault

  Read or write access and access permission (AP) set to no access, or write access and AP set to read only.

When a fault occurs, an interrupt is generated to the MPU. The interrupt service routine (ISR) is then responsible for fault recovery. For example, for a TLB miss, the ISR routine might load the missing entry from the page table. The DSP is stalled by the MMU while the fault is handled.

The ISR can determine the cause of the abort interrupt by reading the F_ST_REG register. The logical address that caused the fault can be determined by reading the FAULT_AD_H_REG and FAULT_AD_L_REG registers.

Following the fault and completion of the fault handling code, the ISR releases the MMU by writing to the interrupt acknowledge register (IT_ACK_REG). The MMU then continues servicing the DSP request. The ISR may also terminate DSP operation by resetting the DSP and the MMU.

7.3.4 Initializing Locked TLB Entries

Follow this procedure to load a translation lookaside buffer manually.

1) Load the logical address, the preserved bit, and the section/page format (SLST field) in the CAM_H_REG and CAM_L_REG registers.
2) Load the physical address and the access permission bits in the RAM_H_REG and RAM_L_REG registers.
3) Update the current_victim field (in the LOCK_REG register), which specifies the location of the entry to be loaded.
4) Set the ld_tlb_item bit of the LD_TLB_REG register to load these values.

7.3.5 Table Walking Logic

The entire TLB can be flushed at once by setting the global_flush bit in the GFLUSH_REG register. TLB entries with a preserved bit set to 1 (bit P of the CAM_L_REG register) are not flushed.

Regardless of the preserved bit setting, a specific TLB entry can be flushed by setting the flush_entry bit in the FLUSH_ENTRY_REG register. The specific entry to be flushed is specified by the logical address in the CAM_H_REG and CAM_L_REG registers.

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 logical 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. A more efficient option is to enable the table walker, as described in this section.

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 logical address, the table walker loads the missing entry from the page table stored in system (OMAP) memory (the MMU autonomously performs the memory access and no intervention of the processor is required).

The OMAP memory area must be initialized with the level 1 and level 2 section and page descriptors. The translation table base address, in the TTB_H_REG and TTB_L_REG registers, must also be initialized with the (physical) address of this OMAP memory area before enabling the MMU.

When a TLB miss occurs, a level 1 descriptor is read based on the logical address and the value of TTB register, the read value gives information to the MMU about the page (page size, protection, upper bits of the physical address). If the page size read is not a section, but a coarse page or fine page a second level descriptor is loaded. The level 1 descriptor address field and the middle part of the logical address give the address where this level 2 descriptor is read. Once the value of this second level descriptor is read, the MMU finally builds the TLB entry.

The TLB entry is written into one of the 32 TLB lines (CAM/RAM), replacing a randomly chosen existing entry.

7.3.6 Boot

After reset, the TLB is empty, the MMU is disabled, and the DSP is held in reset.

The fields of LOCK_REG must be initialized from the MPU before enabling the MMU. TLB entries can then be initialized as needed, starting from entry zero and incrementing current_victim in LOCK_REG as each entry is written. The base_value field in LOCK_REG 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 then can be released from reset and enabled (a single write to CNTL_REG), and the DSP can be released from reset.

If some TLB entries are not initialized (and the current_victim counter has not reached 31), the table walker fills the TLB based on misses until current_victim = 31. The random replacement algorithm is then activated for subsequent misses. When disabled, the MMU performs no translation and builds the 32-bit OMAP address, setting bits 31-24 to 0 and bits 23-0 to the DSP byte address.

7.4 Registers

The DSP MMU registers have the following characteristics:

- All registers are 32 bits wide
- Bits marked as unused must be written as zero and return an unpredictable value when read.
- The value returned from a read of a bit marked as not readable is unpredictable.
- Writes to bits marked as not writable are ignored.

Table 19 lists the DSP MMU registers. Table 21 through Table 41 describe the register bits.

Table 19. DSP MMU Registers

Base Address = FFFE D200			
Name	**Description**	**R/W**	**Offset**
PREFETCH_REG	Prefetch register	R/W	0x00
WALKING_ST_REG	Status	R	004
CNTL_REG	Control	R/W	
FAULT_AD_H_REG	MSB fault address	R	
FAULT_AD_L_REG	LSB fault address	R	010
FAULT_ST_REG	Fault status	R	014
IT_ACK_REG	It acknowledge	W	018
TTB_H_REG	MSB TTB	R/W	01C
TTB_L_REG	LSB TTB	R/W	020
LOCK_REG	Lock counter	R/W	024
LD_TLB_REG	Load entry in TLB	R/W	028
CAM_H_REG	MSB of CAM entry	R/W	02C
CAM_L_REG	LSB of CAM entry	R/W	030
RAM_H_REG	MSB of RAM entry	R/W	034
RAM_L_REG	LSB of RAM entry	R/W	038
GFLUSH_REG	Global flush	R/W	03C
FLUSH_ENTRY_REG	Flush one	R/W	040
READ_CAM_H_REG	MSB read CAM	R	044
READ_CAM_L_REG	LSB read CAM	R	048
READ_RAM_H_REG	MSB read RAM	R	04C
READ_RAM_L_REG	LSB read RAM	R	050
DSPMMU_IDLE_CTRL	Idle control register	R/W	054

Table 20. Prefetch Register (PREFETCH_REG)

Base Address = FFFE D200, Offset = 000				
Bit	**Name**	**Function**	**Reset**	**R/W**
31:14	Unused		0x00000	R
13:0	PrefAddr	MSB of virtual address of the TLB entry to be prefetched.	0	W

Table 21. Status Register (WALKING_ST_REG)

Base Address = FFFE D200, Offset = 004				
Bit	**Name**	**Function**	**Reset**	**R/W**
31:2	Unused			
1	Walk_ working	When 1, the table walking logic is handling a page miss	0	R
0	Prefetch on	Writing in the prefetch register sets this bit, the acknowledge of the prefetch resets the bit.	0	R

Table 22. Control Register (CNTL_REG)

Base Address = FFFE D200, Offset = 008				
Bit	**Name**	**Function**	**Reset**	**R/W**
31:3	Unused			
2	Wtl_en	Enables the table walking logic When active, writes to the LD_TLB_REG, TTB_H_REG, TTB_L_REG, and LOCK_REG have undefined effect. Active high	0	R/W
1	Mmu_en	Enables the MMU When the MMU is disabled, the logical addresses are remapped into CS0 from 0000:0000 to 00FF:FFFF. Active high	0	R/W
0	Reset_sw	Resets the module. This bit must be set manually before activating the MMU. Active low	0	R/W

Table 23. MSB Fault Address Register (FAULT_AD_H_REG)

Base Address = FFFE D200, Offset = 00C				
Bit	**Name**	**Function**	**Reset**	**R/W**
31:8	Unused			
7:0	Fault_address_msb	MSB of logical address of the access that generated a permission fault	0	R

Table 24. LSB Fault Address Register (FAULT_AD_L_REG)

Base Address = FFFE D200, Offset = 010				
Bit	**Name**	**Function**	**Reset**	**R/W**
31:16	Unused			
15:0	Fault_address_lsb	LSB of logical address of the access that generated a permission fault	0	R

Table 25. Fault Status Register (FAULT_ST_REG)

Base Address = FFFE D200, Offset = 014				
Bit	**Name**	**Function**	**Reset**	**R/W**
31:4	Unused			
3	Prefetch_err	This error occurs during a prefetch Active high	0	R
2	Perm_fault	Permission fault Active high	0	R
1	Tlb_miss	TLB miss (when WTB is disabled) Active high	0	R
0	Trans_fault	Translation fault (invalid descriptor) Active high	0	R

These fault registers contain the logical address from the DSP that generated the interruption.

Table 26. It Acknowledge Register (IT_ACK_REG)

Base Address = FFFE D200, Offset = 018				
Bit	**Name**	**Function**	**Reset**	**R/W**
31:1	Unused			
0	It_ack	Writing 1 in this register acknowledges the interrupt.	0	W

Table 27. MSB TTB Register (TTB_H_REG)

Base Address = FFFE D200, Offset = 01C				
Bit	**Name**	**Function**	**Reset**	**R/W**
31: 16	Unused			
15:0	TTB_H_REG	MSB of TTB	0	R/W

Table 28. LSB TTB Register (TTB_L_REG)

Base Address = FFFE D200, Offset = 020				
Bit	**Name**	**Function**	**Reset**	**R/W**
31: 16	Unused			
15:7	TTB_L_REG	LSB of TTB	0	R/W
6:0	Unused			

Table 29. Lock Counter Register (LOCK_REG)

Base Address = FFFE D200, Offset = 024				
Bit	**Name**	**Function**	**Reset**	**R/W**
31:16	Unused		0	R/W
15:10	Base_value	Locked entries base value. 0 ≤ Base_value	0	R/W
9:4	Current_victim	Current entry pointed by the hardware table walk logic Base value ≤ 31	0	R/W
3:0	Unused		0	R/W

Table 30. Load Entry in TLB Register (LD_TLB_REG)

Base Address = FFFE D200, Offset = 028				
Bit	**Name**	**Function**	**Reset**	**R/W**
31:1	Unused			
0	Ld_tlb_item	Load data in TLB. Toggle bit. Loaded when 1 is written. Always 0 when read. Automatically reset.	0	R/W

Table 31. MSB of CAM Entry Register (CAM_H_REG)

Base Address = FFFE D200, Offset = 02C				
Bit	**Name**	**Function**	**Reset**	**R/W**
31:16	Unused			
15:0	VA_tag_l1_H	Table index level 1 MSB Should be used only when WTL_EN is disabled	0	R/W

Table 32. LSB of CAM Entry Register (CAM_L_REG)

Base Address = FFFE D200, Offset = 030				
Bit	**Name**	**Function**	**Reset**	**R/W**
31:16	Unused			
15:14	VA_tag_l1_L	Table index level 1 LSB which means bits (21:20) of logical address	0	R/W
13:4	VA_tag_l2	Table index level 2: Tiny page: bits 13:4 Small page: bits 13:6, rest of bits discarded Large page: bits 13:10, rest of bits discarded Unused bits must be written as zero; read value is unpredictable.	0	R/W
3	P	Preserved bit : 0: CAM entry not preserved 1: CAM entry preserved	0	R/W
2	V	Valid bit: 0: CAM entry not valid 1: CAM entry valid	0	R
1:0	SLST	SLST: When 00 section (1MB) 01 Large pages (64KB) 10 Small page (4KB) 11 Tiny page (1KB)	0	R/W

Table 33. MSB of RAM Entry Register (RAM_H_REG)

Base Address = FFFE D200, Offset = 034				
Bit	**Name**	**Function**	**Reset**	**R/W**
31:16	Unused			
15:0	Ram_msb	MSB physical address	0	R/W

Table 34. LSB of RAM Entry Register (RAM_L_REG)

Base Address = FFFE D200, Offset = 038				
Bit	**Name**	**Function**	**Reset**	**R/W**
15:10	Ram_lsb	LSB physical address	0	R/W
9:8	AP	Access permission bits	0	R/W
7:0	Unused			

Table 35. Global Flush Register (GFLUSH_REG)

Base Address = FFFE D200, Offset = 03C				
Bit	**Name**	**Function**	**Reset**	**R/W**
31:1	Unused			
0	Global_flush	Flush all the nonprotected TLB entries when 1 is written. Always 0 when read.	0	R/W

Warning: Flushing the whole TLB does not change the base value and the victim counter of the lock counter register.

Table 36. Flush One Register (FLUSH_ENTRY_REG)

Base Address = FFFE D200, Offset = 040				
Bit	**Name**	**Function**	**Reset**	**R/W**
31:1	Unused			
0	Flush_entry	Flush the TLB entry pointed by the logical address in CAM_H_REG and CAM_L_REG registers (even if this entry is set protected). Active high.	0	R/W

Table 37. MSB Read CAM Register (READ_CAM_H_REG)

Base Address = FFFE D200, Offset = 044				
Bit	**Name**	**Function**	**Reset**	**R/W**
31:3	Unused			
2:0	VA_tag_l1_H	Table index level 1 MSB	0	R

Table 38. LSB Read CAM Register (READ_CAM_L_REG)

Base Address = FFFE D200, Offset = 048				
Bit	**Name**	**Function**	**Reset**	**R/W**
31:16	Unused			
15:14	VA_tag_l1_L	Table index level 1 LSB which means bits (21:20) of logical address	0	R
13:4	VA_tag_l2	Table index level 2: Tiny page: bits 13:4 Small page: bits 13:6, rest of bits discarded Large page: bits 13:10, rest of bits discarded Unused bits must be written as zero; read value is unpredictable.	0	R
3	P	Preserved bit : 0: CAM entry not preserved 1: CAM entry preserved	0	R
2	V	Valid bit : 0: CAM entry not valid 1: CAM entry valid	0	R
1:0	SLST	SLST: When 00 section (1MB) 01 Large pages (64KB) 10 Small page (4KB) 11 Tiny page (1KB)	0	R

Table 39. MSB Read RAM Register (READ_RAM_H_REG)

Base Address = FFFE D200, Offset = 04C				
Bit	**Name**	**Function**	**Reset**	**R/W**
31:16	Unused			
15:0	Ram_msb	MSB physical address	0	R

Table 40. LSB Read RAM Register (READ_RAM_L_REG)

Base Address = FFFE D200, Offset = 050				
Bit	**Name**	**Function**	**Reset**	**R/W**
31:16	Unused			
15:8	Ram_lsb	LSB physical address	0	R
7:0	Unused			

Table 41. DSP MMU Idle Control Register (DSPMMU_IDLE_CTRL)

Base Address = FFFE D200, Offset = 054				
Bit	**Name**	**Function**	**Reset**	**R/W**
31:2	Unused			
1	GL_PDE	Global power-down enable 0 − DSPMMU clock is always running and autogating works when it is enabled 1 − DSPMMU clock is totally shut off and autogatong does not work	0	R/W
0	AUTOGATING_EN	Enables/Disables autogating	0	R/W

8 DSP Subsystem Clocking and Reset Control

The clock generator and system reset module (CLK and RST) manages operations such as the reset sequences, the clock generation function, the power-saving modes, idle controls, and setup for the OMAP5912. The clock domains in the OMAP5912 platform are synthesized by the DPLL1. The DPLL input clock source is externally supplied from the CLKIN pin.

The MPU manages the master clock configuration for the OMAP5912 device. The DSP subsystem master clock DSP_CK is enabled at reset until the DSP is enabled. The EN_DSPCK bit in the clock control register ARM_CKCTL allows turning off the DSP_CK while the DSP is still in a reset state.

The CLKM2 module generates the individual clock domains for the DSP subsystem. These clock signals have programmable frequencies based on divisors of several possible input clock sources. CLKM2 is considered an MPU private peripheral, except for configuration of subdomain clocks for the DSP subsystem. See chapter 4 for more information on clocking.

9 System Operating Details

9.1 DSP Private Peripherals

The DSP private peripherals are connected to the DSP CPU by a private TIPB bridge. This provides reduced latency for DSP access to these particular peripherals. The private peripherals consist of the following modules:

- Three general-purpose timers
- A watchdog timer
- An interrupt handler

These modules are clocked by dedicated signals controlled by the CLKM2 DSP_CKCTL register.

The access rate to these peripherals is fixed by the TIPB bridge module and does not have to be user-configured.

9.2 DSP Public Peripherals

The public TIPB connects the DSP public peripherals to the DSP CPU to provide a flexible communications scheme where the DSP or MPU domains can access these devices. Because the peripheral registers are also mapped in the MPU memory space, the MPU domain can access these peripherals indirectly via the MPUI and public TIPB bridge. This results in a *pseudodynamic* sharing scheme. The DSP public peripherals consist of the following modules:

- Two McBSPs: McBSP1 and McBSP3
- Two MCSIs: MCSI1 and MCSI2

These peripherals are clocked by the DSPXOR_CK signal, which is a buffered version of the OMAP5912 CLKIN signal.

The access rate to these peripherals is configured by strobe 2 control bits in the DSP TIPB CMR register. See Section 4.1, *Control Mode Register*.

9.3 DSP/MPU Shared Peripherals

The DSP/MPU shared peripherals are designed with two TIPB connections, one for the DSP public TIPB and another for the MPU public TIPB. This dual connection provides a flexible communications scheme where either the DSP domain or the MPU domain can access a peripheral without monopolizing the alternate processor public peripheral bus. The DSP/MPU shared peripherals consist of the following modules:

- Mailbox registers for interprocessor communication
- General-purpose I/O (GPIO)
- Three UARTs: UART1, UART2, UART3

These peripherals can be clocked by signals from the DSP subsystem CLKM2 module or the MPU subsystem CLKM1 module.

The access rate to these peripherals via the DSP is configured by the strobe2 control bits in the TIPB CMR register. See Section 4.1, *Control Mode Register*.

9.4 Boot Mode for DSP Subsystem

The OMAP5912 device contains a bootloader that is a ROM-based utility residing in the DSP subsystem ROM. It consists of a program (code) that

facilitates downloading (bootloading) of DSP code into the DSP subsystem internal memory from either the DSP EMIF interface to the traffic controller or the MPUI interface when it is held in reset by the MPU. The boot mode used by the DSP subsystem bootloader is specified by the MPU using the DSP_BOOT_CONFIG register when it is released from reset by the MPU. This register is read-only for the DSP and is mapped to address 0x000F in the DSP I/O space (within the DSP TIPB address space). The register is read/write for the MPU and appears at address 0xFFFE:C900 in the MPUI address space. The MPU controls the boot process by programming bits BOOT_MOD[3:0] while the DSP subsystem is held in reset state. Table 42 shows the DSP boot configuration.

Table 42. DSP Boot Configuration

Relative Word Address	Register Name	Bits	Reset Value
0x00000F	DSP_BOOT_CONFIG	BOOT_MODE [3:0]	Depends on external implementation

9.4.1 Boot Modes

The DSP is reset by two signals:

- nRESET is a global reset (active low) that resets the DSP subsystem except for the TIPB interrupt priority encoder, the DSP EMIF configuration registers, and the MPUI port control logic.
- nMCURESET is a reset signal driven by the MPU (active low). It sets the TIPB interrupt priority encoder registers, configures the DSP-EMIF registers, and resets the MPUI control logic.

The MPU controls these reset signals by writing into the RSTCT1 clock and reset module register.

There are five boot modes for the DSP subsystem. The MPU can select any of these boot modes by writing to the DSP_BOOT_CONFIG register. When the DSP subsystem is released from reset (boot), the CPU always fetches instruction at address 0xFFFF00. The physical location of this address depends on the value of the BOOT_MOD[3:0] bits.

If BOOT_MOD[3:0] is equal to 0000, the on-chip ROM is not available and the boot address is located off-chip. If BOOT_MOD[3:0] is not equal to 0000, the on-chip ROM is enabled and the boot address is located at the on-chip ROM. The boot modes supported are listed in Table 43.

Table 43. Boot Modes

BOOT_MOD[3−0]	Boot Process	Starting Address of DSP MPU
0000	No boot download	0xFFFF00 (direct boot from a 32-bit asynchronous interface)
0001	No boot download	Pseudodirect boot from a 32-bit asynchronous interface (bootloader configures the EMIF, and then branches to address 0x080000)
0010	Put DSP into IDLE state	DSP is put into idle mode.
0011	16-bit boot download	Download code from 0x80000 to user-specified address (residing at address 0x080004 as 16-bit data)
0100	32-bit boot download	Download code from 0x80000 to user-specified address (residing at address 0x080004 as 32-bit data).
0101	MPUI boot download	Branch to address 0x10000
Other	Internal boot	Branch to address 0x24000

Note: The DSP bootloader checks the BOOT_MOD[3:0] bits only once on DPS subsystem release from reset.

9.4.2 Boot Table Formats

For 16-bit boot download, the boot table in external memory must be in the format shown in Table 44.

Table 44. External Memory Boot Table for 16-Bit Boot Download

Word Address (16-Bit Word)	Contents
Start of external memory Address Emifaddr	Number of elements of the first section to transfer = N1. After the number of words to transfer from this section, the next word must be zero to indicate the end of the source program. Otherwise, another section is assumed to follow.
Emifaddr+1h	Most significant word of destination address for the 1st section. Can be 0, 1, or 2.
Emifaddr+2h	Least significant word of destination address for the 1st section. Can be from 0000h to FFFFh.
Emifaddr+3h	1st word of 1st section to transfer
Emifaddr+4h	2nd word of 1st section to transfer
...	.....
	N1th word of 1st section to transfer

Table 44. External Memory Boot Table for 16-Bit Boot Download (Continued)

Word Address (16-Bit Word)	Contents
	Number of elements of the 2nd section to transfer = N2
	Most significant word of destination address for the 2nd section. Can be 0, 1, or 2.
	Least significant word of destination address for the 2nd section. Can be from 0000h to FFFFh.
	1st word of 2nd section to transfer
	2nd word of 2nd section to transfer
	.....
	N2th word of 2nd section to transfer
	.....
	Number of elements of the last section to transfer = NL
	Most significant word of destination address for the last section. Can be 0, 1, or 2.
	Least significant word of destination address for the last section. Can be from 0000h to FFFFh.
	1st word of last section to transfer
	2nd word of last section to transfer
	.....
	NLth word of last section to transfer
	0000h to indicate the end of source program

For 32-bit boot download, the boot table in external memory must be in the format shown in Table 45.

Table 45. External Memory Boot Table for 32-Bit Boot Download

Word Address (16-bit word)	Contents
Start of external memory Address Emifaddr	LSB of number of elements of the first section to transfer = N1 After the number of words to transfer from this section, the next word must be zero to indicate the end of the source program. Otherwise, another section is assumed to follow.
Emifaddr+1h	MSB of number of elements of the first section to transfer = N1

Table 45. External Memory Boot Table for 32-Bit Boot Download (Continued)

Word Address (16-bit word)	Contents
Emifaddr+ 2h	Least significant word of destination address for the 1st section. Can be from 0000h to FFFFh.
Emifaddr+3h	Most significant word of destination address for the 1st section. Can be 0, 1, or 2.
Emifaddr+4h	1st word of 1st section to transfer
Emifaddr+5h	2nd word of 1st section to transfer
...	.....
	N1th word of 1st section to transfer
	Number of elements of the 2nd section to transfer = N2
	Most significant word of destination address for the 2nd section. Can be 0, 1, or 2.
*	Least significant word of destination address for the 2nd section. Can be from 0000h to FFFFh.
	1st word of 2nd section to transfer
	2nd word of 2nd section to transfer
	.....
	N2th word of 2nd section to transfer
	.....
	Number of elements of the last section to transfer = NL
	Most significant word of destination address for the last section. Can be 0, 1, or 2.
	Least significant word of destination address for the last section. Can be from 0000h to FFFFh.WA
	1st word of last section to transfer
	2nd word of last section to transfer
	.....
	NLth word of last section to transfer
	0000h to indicate the end of source program

9.4.3 Bootloader Description

When the MPU releases the DSP subsystem from reset, if pins BOOT_MOD[3:0] = 0000, then the address 0xFFFF00 is mapped into external memory space. If pins BOOT_MOD[3:0] ≠ 0000, then the address 0xFFFF00 maps to internal ROM that has vector to the bootloader at 0xFF800. At this point, the bootloader starts to execute. It determines whether the DSP subsystem in is SAM. If not, the bootloader keeps checking indefinitely. As soon as the DSP subsystem is in SAM, it performs the following initialization:

- Set up stack pointer (stack size is set to 0x20; default stack mode is fast dual stack)
- Disable interrupts globally
- Mask off interrupts
- Set up EMIF

After this is done, the bootloader starts to check the BOOT_MOD[3:0] bits of the DSP_BOOT_CONFIG register. Depending on the bit combination as described in Table 45, it then boots the DSP subsystem.

Index

E

H

I

O

P

R

S

OMAP5912 Multimedia Processor Clocks Reference Guide

Literature Number: SPRU751A
March 2004

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.

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
		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 document describes the clocking mechanisms of the OMAP5912 multimedia processor.

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.

Related Documentation From Texas Instruments

The following documents describe the OMAP5910 device and related peripherals. Copies of these documents are available on the Internet at www.ti.com. *Tip:* Enter the literature number in the search box provided at www.ti.com.

OMAP5912 Multimedia Processor Device Overview and Architecture Reference Guide (literature number SPRU748) introduces the setup, components, and features of the OMAP5912 multimedia processor and provides a high-level view of the device architecture.

OMAP5912 Multimedia Processor OMAP 3.2 Subsystem Reference Guide (literature number SPRU749) introduces and briefly defines the main features of the OMAP3.2 subsystem of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor DSP Sybsystem Reference Guide (literature number SPRU750) describes the OMAP5912 multimedia processor DSP subsystem. The digital signal processor (DSP) subsystem is built around a core processor and peripherals that interface with: 1) The ARM926EJS via the microprocessor unit interface (MPUI); 2) Various standard memories via the external memory interface (EMIF); 3) Various system peripherals via the TI peripheral bus (TIPB) bridge.

OMAP5912 Multimedia Processor Clocks Reference Guide (literature number SPRU751) describes the clocking mechanisms of the OMAP5912 multimedia processor. In OMAP5912, various clocks are created from special components such as the digital phase locked loop (DPLL) and the analog phase-locked loop (APLL).

OMAP5912 Multimedia Processor Initialization Reference Guide (literature number SPRU752) describes the reset architecture, the configuration, the initialization, and the boot ROM of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Power Management Reference Guide (literature number SPRU753) describes power management in the OMAP5912 multimedia processor. The ultralow-power device (ULPD) generates and manages clocks and reset signals to OMAP3.2 and to some peripherals. It controls chip-level power-down modes and handles chip-level wake-up events. In deep sleep mode, this module is still active to monitor wake-up events.This book describes the ULPD module and outline architecture.

OMAP5912 Multimedia Processor Security Features Reference Guide (literature number SPRU754) describes the security features of teh OMAP5912 multimedia processor. The OMAP5912 security scheme relies on the OMAP3.2 secure mode. The distributed security on the OMAP3.2 platform is a Texas Instruments solution to address m-commerce and security issues within a mobile phone environment. The OMAP3.2 secure mode was developed to bring hardware robustness to the overall OMAP5912 security scheme.

OMAP5912 Multimedia Processor Direct Memory Access (DMA) Support Reference Guide (literature number SPRU755) describes the direct memory access support of the OMAP5912 multimedia processor. The OMAP5912 processor has three DMAs:

- The system DMA is embedded in OMAP3.2. It handles DMA transfers associated with MPU and shared peripherals.
- The DSP DMA is embedded in OMAP3.2. It handles DMA transfers associated with DSP peripherals.
- The generic distributed DMA (GDD) is an OMAP5912 resource attached to the SSI peripheral. It handles only DMA transfers associated with the SSI peripheral.

OMAP5912 Multimedia Processor Memory Interfaces Reference Guide (literature number SPRU756) describes the memory interfaces of the OMAP5912 multimedia processor.

- SDRAM (external memory interface fast, or EMIFF)
- Asynchronous and synchronous burst memory (external memory interface slow, or EMIFS)
- NAND flash (hardware controller or software controller)
- CompactFlash on EMIFS interface
- Internal static RAM

OMAP5912 Multimedia Processor Interrupts Reference Guide (literature number SPRU757) describes the interrupts of the OMAP5912 multimedia processor. Three level 2 interrupt controllers are used in OMAP5912:

- One MPU level 2 interrupt handler (also referred to as MPU interrupt level 2) is implemented outside of OMAP3.2 and can handle 128 interrupts.
- One DSP level 2 interrupt handler (also referred to as DSP interrupt level 2.1) is instantiated outside of OMAP3.2 and can handle 64 interrupts.
- One OMAP3.2 DSP level 2 interrupt handler (referenced as DSP interrupt level 2.0) can handle 16 interrupts.

OMAP5912 Multimedia Processor Peripheral Interconnects Reference Guide (literature number SPRU758) describes various periperal interconnects of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Timers Reference Guide (literature number SPRU759) describes various timers of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Serial Interfaces Reference Guide (literature number SPRU760) describes the serial interfaces of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Universal Serial Bus (USB) Reference Guide (literature number SPRU761) describes the universal serial bus (USB) host on the OMAP5912 multimedia processor. The OMAP5912 processor provides several varieties of USB functionality. Flexible multiplexing of signals from the OMAP5912 USB host controller, the OMAP5912 USB function controller, and other OMAP5912 peripherals allow a wide variety of system-level USB capabilities. Many of the OMAP5912 pins can be used for USB-related signals or for signals from other OMAP5912 peripherals. The OMAP5912 top-level pin multiplexing

controls each pin individually to select one of several possible internal pin signal interconnections. When these shared pins are programmed for use as USB signals, the OMAP5912 USB signal multiplexing selects how the signals associated with the three OMAP5912 USB host ports and the OMAP5912 USB function controller can be brought out to OMAP5912 pins.

OMAP5912 Multimedia Processor Multi-channel Buffered Serial Ports (McBSPs) Reference Guide (literature number SPRU762) describes the three multi-channel buffered serial ports (McBSPs) available on the OMAP5912 device. The OMAP5912 device provides multiple high-speed multichannel buffered serial ports (McBSPs) that allow direct interface to codecs and other devices in a system.

OMAP5912 Multimedia Processor Camera Interface Reference Guide (literature number SPRU763) describes two camera inerfaces implemented in the OMAP5912 multimedia processor: compact serial camera port and camera parallel interface.

OMAP5912 Multimedia Processor Display Interface Reference Guide (literature number SPRU764) describes the display interface of the OMAP5912 multimedia processor.

- LCD module
- LCD data conversion module
- LED pulse generator
- Display interface

OMAP5912 Multimedia Processor Multimedia Card (MMC/SD/SDIO) (literature number SPRU765) describes the multimedia card (MMC) interface of the OMAP5912 multimedia processor. The multimedia card/secure data/secure digital IO (MMC/SD/SDIO) host controller provides an interface between a local host, such as a microprocessor unit (MPU) or digital signal processor (DSP), and either an MMC or SD memory card, plus up to four serial flash cards. The host controller handles MMC/SD/SDIO or serial port interface (SPI) transactions with minimal local host intervention.

OMAP5912 Multimedia Processor Keyboard Interface Reference Guide (literature number SPRU766) describes the keyboard interface of the OMAP5912 multimedia processor. The MPUIO module enables direct I/O communication between the MPU (through the public TIPB) and external devices. Two types of I/O can be used: specific I/Os dedicated for 8 x 8 keyboard connection, and general-purpose I/Os.

OMAP5912 Multimedia Processor General-Purpose Interface Reference Guide (literature number SPRU767) describes the general-purpose in-

terface of the OMAP5912 multimedia processor. There are four GPIO modules in the OMAP5912. Each GPIO peripheral controls 16 dedicated pins configurable either as input or output for general purposes. Each pin has an independent control direction set by a programmable register. The two−edge control registers configure events (rising edge, falling edge, or both edges) on an input pin to trigger interrupts or wake−up requests (depending on the system mode). In addition, an interrupt mask register masks out specified pins. Finally, the GPIO peripherals provide the set and clear capabilities on the data output registers and the interrupt mask registers. After detection, all event sources are merged and a single synchronous interrupt (per module) is generated in active mode, whereas a unique wake−up line is issued in idle mode. Eight data output lines of the GPIO3 are ORed together to generate a global output line at the OMAP5912 boundary. This global output line can be used in conjunction with the SSI to provide a CMT−APE interface to the OMAP5912.

OMAP5912 Multimedia Processor VLYNQ Serial Communications Interface Reference Guide (literature number SPRU768) describes the VLYNQ of the OMAP5912 multimedia processor.

VLYNQ is a serial communications interface that enables the extension of an internal bus segment to one or more external physical devices. The external devices are mapped into local, physical address space and appear as if they are on the internal bus of the OMAP 5912. The external devices must also have a VLYNQ interface. The VLYNQ module serializes bus transactions in one device, transfers the serialized data between devices via a VLYNQ port, and de-serializes the transaction in the external device.

OMAP5912 includes one VLYNQ module connected on OCPT2 target port and OCPI initiator port. These connections are configured via a static switch, which selects either SSI or VLYNQ module. This switch, forbids the simultaneous use of GDD/SSI and VLYNQ. TThe switch is controlled by the VLYNQ_EN bit in the OMAP5912 configuration control register (CONF_5912_CTRL).

OMAP5912 Multimedia Processor Pinout Reference Guide (literature number SPRU769) provides the pinout of the OMAP5912 multimedia processor. After power-up reset, the user can change the configuration of the default interfaces. If another interface is available on top of the default, it is possible to enable a new interface for each ball by setting the corresponding 3-bit field of the associated FUNC_MUX_CTRL register. It is also possible to configure on-chip pullup/pulldown. This document

also describes the various power domains so that the user can apply the different interfaces seamlessly with external components.

OMAP5912 Multimedia Processor Window Tracer (WT) Reference Guide (literature number SPRU770) describes the window tracer module used to capture the memory transactions from four interfaces: EMIFF, EMIFS, OCP-T1, and OCP-T2. This module is located in the OMAP3.2 traffic controller (TC).

OMAP5912 Multimedia Processor Real-Time Clock Reference Guide (literature number SPRUxxx) describes the real-time clock of the OMAP5912 multimedia processor. The real-time clock (RTC) block is an embedded real-time clock module directly accessible from the TIPB bus interface.

Trademarks

Contents

Figures

Tables

Clocks

This document describes the clocking mechanisms of the OMAP5912 multimedia processor.

1 Overview

In OMAP5912, various clocks are created from special components such as the digital phase locked loop (DPLL) and the analog phase-locked loop (APLL).

- The DPLL converts the input clock (12 MHz–20 MHz) to a high-frequency clock (200 MHz), which is then distributed within OMAP 3.2 gigacell and to the various on-chip peripherals.

  At power-up reset, the DPLL is in bypass mode and acts as a clock divider. The desired frequency for the DPLL output clock is achieved by enabling the DPLL and setting the multiplication and divider ratios.

Note:

At power-up reset, the DPLL, by default, is in divide-by-one mode.

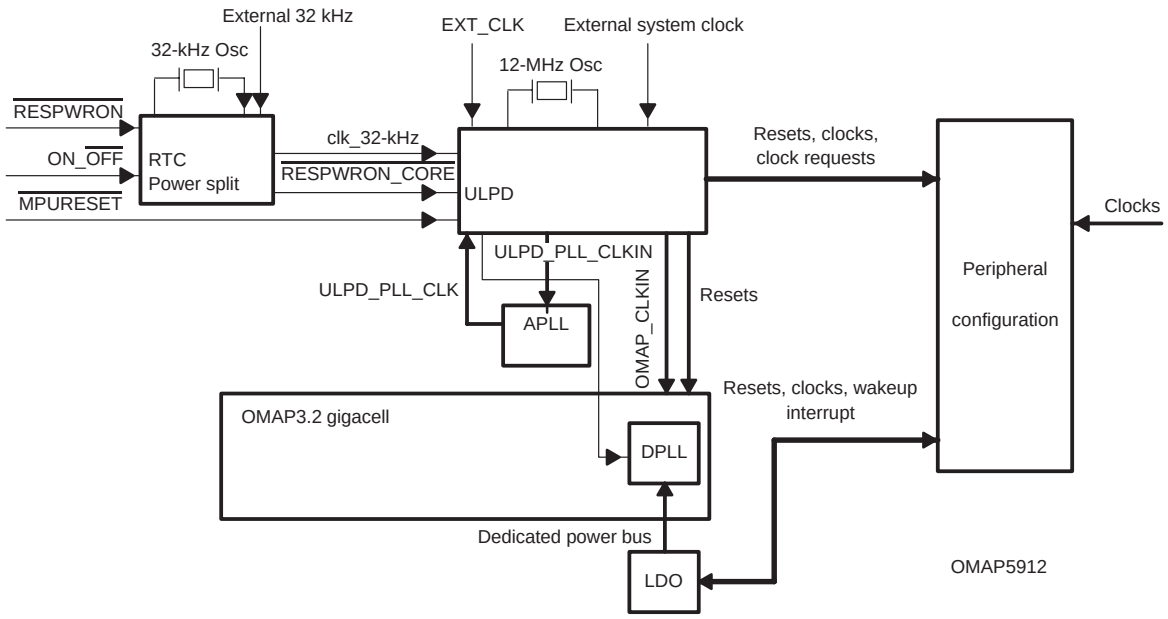
  The input clock is provided by an external device or is derived from the on-chip 12-MHz oscillator. For more details, see Chapter 5, *Initialization*.

  An embedded LDO provides a dedicated power supply to the analog portion of the DPLL to ensure low jitter on the DPLL output clock. The regular core voltage supplies the wrapper of the DPLL. The OMAP5912 configuration modules control the LDO and observe its state.

- The APLL is a clock multiplier that provides a fixed 96-MHz clock from the same input clock used to feed the DPLL. The input clock frequency is 12 MHz, 13 MHz, or 19.2 MHz. The APLL is enabled whenever the ULPD requests a 48-MHz clock. The ULPD_PLL_CTRL_STATUS register is set to ensure the multiplication factor. The 48-MHz frequency is then achieved by a divide-by-2 cell in the ULPD.

Figure 1 shows the OMAP5912 global-clocking environment.

Figure 1. OMAP5912 Clocks



1.1 OMAP5912 Integration

1.1.1 Low-Dropout (LDO) Voltage Regulator

After the power supplies have ramped up and the power-up reset has been released, the LDO ramps up. When its output voltage is no more than 200 mV below its final value, the LDO delivers a steady signal set to 1 to the ULPD. The DPLL is configured in DPLL enable mode, as the LDO output voltage is stable.

Before entering low-power modes such as deep sleep and big sleep, the LDO is put in sleep mode to reduce power dissipation. In that mode, the LDO delivers a voltage derived from the VDD core through a pass-gate transistor. Although the DPLL is in low-power mode, the DPLL settings stay the same. In this configuration, the latency is reduced with returning to active mode.

The LDO can be powered down, where it behaves as a feed-through cell. Power to the DPLL must be provided through an external power supply source. This mode can also be a back-up mode if an external power supply is preferred (see Table 1).

Table 1. LDO Control and Observability

LDO Mode	Signal	Register Bit Description	Notes
LDO in power-down mode	PWRDWN	CONF_LDO_PWRDN_CNTRL_R	OMAP5912 configuration
LDO in sleep mode	SLEEP	SOFT_LDO_SLEEP	ULPD register file
LDO stable	STEADY	LDO_STEADY	ULPD register file

1.1.2 96-MHz APLL

The APLL is enabled whenever a clock request for a 48-MHz clock is active. The clock request can be either hardware or software. The APLL generates a 96-MHz clock, which is then divided by 2 in the ULPD module. An external 48-MHz clock can be selected by software if the APLL is not used.

Table 2 lists the different requests to activate the APLL. When no request is active, the APLL is in power-down mode.

Table 2. Request for 48-MHz Clock

Request	Source	Type of Request	Destination Module
CONF_MOD_UART1_CLK_MODE_R	OMAP5912 configuration	Software	UART1
CONF_MOD_UART2_CLK_MODE_R	OMAP5912 configuration	Software	UART2
CONF_MOD_UART3_CLK_MODE_R	OMAP5912 configuration	Software	UART3
CONF_MOD_USB_HOST_HHC_UHOST_EN_R	OMAP5912 configuration	Software	USB OTG
USB_DPLL_MCLK_REQ	USB OTG	Hardware	USB OTG
CONF_MOD_MMC_SD_CLK_REQ_R	OMAP5912 configuration	Software	MMC/SDIO1
CONF_MOD_MMC_SD2_CLK_REQ_R	OMAP5912 configuration	Software	MMC/SDIO2
CONF_CAM_CLKMUX_R	OMAP 5912 configuration	Software	Camera I/F

The selection of the APLL modes is done in the ULPD module, as shown in Table 3.

Table 3. APLL Mode Selection

APLL SEL2	APLL SEL1	APLL SEL0	CLKIN (MHz)	CLKOUT (MHz)	APLL MODE	ULPD_PLL_CTRL_STATUS[2:0]
L	L	L	19.2	96	Application mode 0	000
L	H	L	13	96	Application mode 2	010
L	H	H	12	96	Application mode 3	011

The reset value is 011, which selects APLL application mode 3. The LOCK_STATUS flag in the ULPD_PLL_CTRL_STATUS register indicates that the APLL has locked.

The internally generated 48-MHz clock or an external 48-MHz clock are selected via the TEST_DBG_CTRL_0 bit in the OMAP5912 configuration. If the TEST_DBG_CTRL_0 bit is set to 1, GPIO_14 becomes the source of the 48 MHz for the device.

1.1.3 OMAP3.2 DPLL

The DPLL is controlled through the DPLL1_CTL_REG, which is mapped in the OMAP3.2 register file. See Section 3, *OMAP3.2 DPLL,* for more detail.

2 Analog Phase-Locked Loop

The APLL is a clock multiplier for creating a 96-MHz clock from 12-MHz, 13-MHz, and 19.2-MHz input frequencies.

It includes an APLL circuit to generate an output clock, which is related to the frequency of the input reference clock. The frequency multiplication factor is an integer (12 MHz/19.2 MHz to 96 MHz) or a fractional (13 MHz−96 MHz):

- In the integer multiplication scheme, the rising edge of the output clock is synchronized to the rising edge of the input clock.
- In the fractional multiplication scheme, the rising edge of the output clock is not synchronized to the rising edge of the input clock for all input clock periods. Nevertheless, the cell produces pulses to notify when the rising edges of the input and output clocks coincide.

This output can be used to decimate the cycles where the phase relationship between input and output clocks is unknown.

Dedicated power supply pins are required to isolate the core analog circuit from the switching noise generated by the core logic that can cause jitter on the clock output signal.

The cell has level shifters on all input ports to avoid through-current when V_{DD} and V_{DDA} are not at the same voltage. Natural level shifting is done at the cell outputs to provide V_{DDA}-compatible CMOS levels.

Three modes of operation and one power-down mode can be selected from the core inputs:

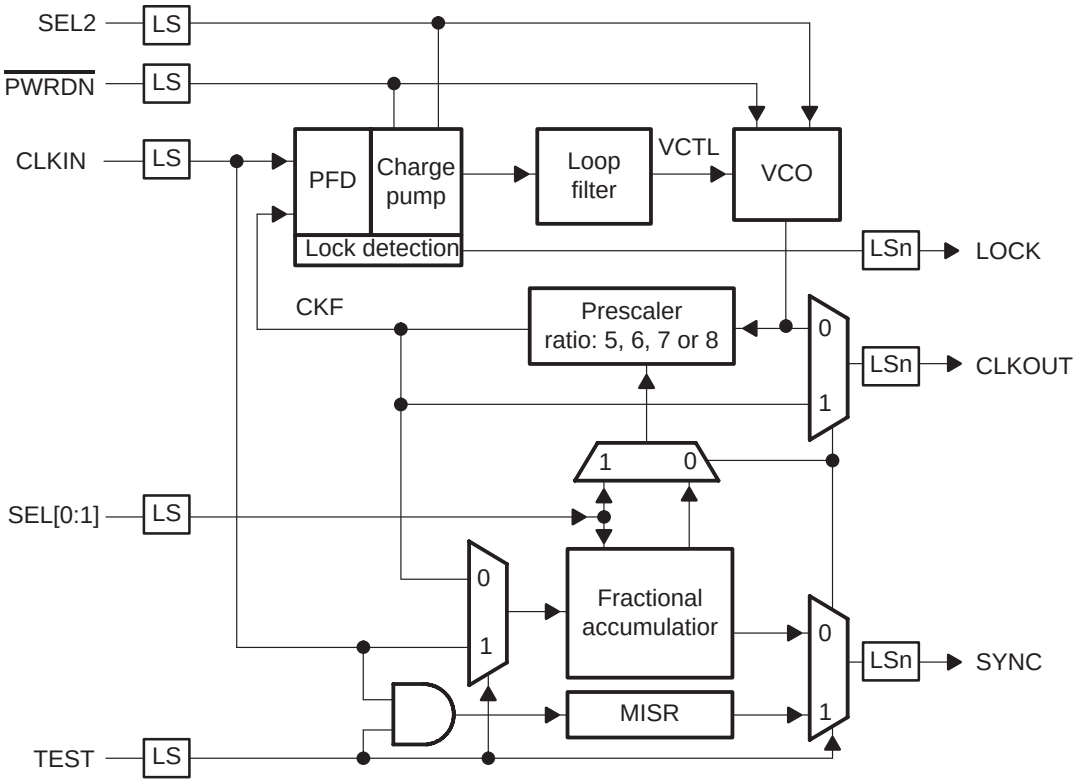
- Application mode 0: The input frequency is 19.2 MHz, and the multiplication factor is 5.
- Application mode 2: The input frequency is 13 MHz, and the multiplication factor is 96/13.
- Application mode 3: The input frequency is 12 MHz, and the multiplication factor is 8.
- Power-down mode: Tests I_{DDQ} on the analog power supply and saves power when the multiplied output clock is not required.

Because the core voltage can vary while V_{DDA} is stable at 1.8 V, the cell is in power-down mode if the voltage on V_{DD} is lower than 200 mV.

Note:

OMAP5912 supports only the above application modes. Although other application modes are possible because of the generic nature of APLL, they are not included in the ULPD architecture.

Figure 2. APLL Block Diagram



- LS: Level shifter
- LSn: Natural level shifter or two successive buffers tied to different supply domains. The usual level-shifting scheme is not suited to the clock output for speed and duty-cycle reasons.
- MISR: Multiple input-shift register for digital test purposes

Table 4. Mode Selection Table

PWRDN	TEST	SEL2	SEL1	SEL0	CLKIN (MHz)	CLKOUT (MHz)	SYNC	MODE
L	-	-	-	-	0	-	-	Power down (1)
H	L	L	L	L	19.2	96	H	Application mode 0
H	L	L	H	L	13	96	Pulsed (1MHz)	Application mode 2
H	L	L	H	H	12	96	H	Application mode 3

Note: Full IDDQ mode is reached when the input clock is not toggling; otherwise, a residual current may be found.

2.1 Application Guidelines

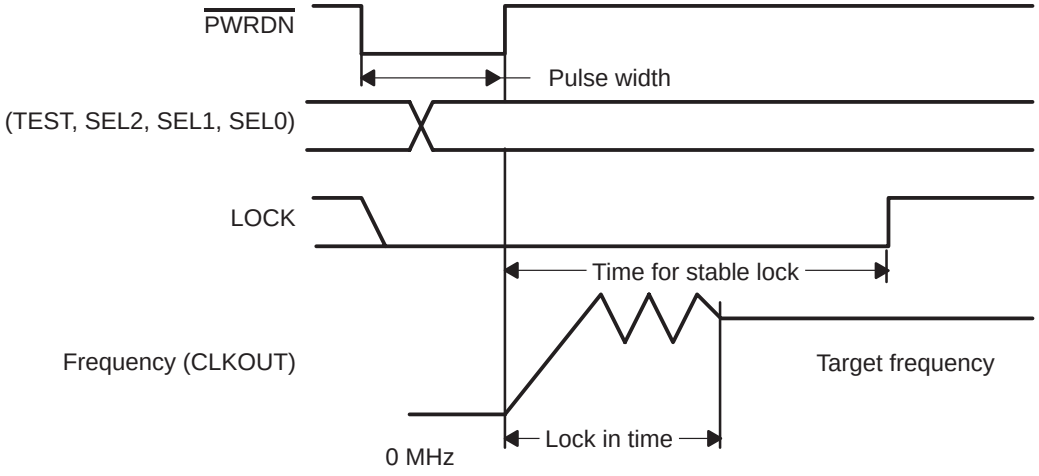
Adhere to the following guidelines to get the best performance from the cell.

Fast Lockup

A fast lockup follows this control sequence (see Figure 3):

1) $\overline{\text{PWRDN}}$ must be set to 0 any time the APLL must be used.
2) $\overline{\text{PWRDN}}$ must stay at low for a minimum time.
3) The mode selection bits (TEST, SEL2, SEL1, SEL0) must be set during the time $\overline{\text{PWRDN}}$ is low.
4) LOCK always falls to 0 when $\overline{\text{PWRDN}}$ is set to low. This may take between several 100 ns and 1 µs.
5) When $\overline{\text{PWRDN}}$ is raised to high, and if the input clock toggles within the specified frequency range, the APLL locks on the target frequency and the lock signal goes to high.
6) Jitter is in specification after LOCK has gone to high.

Figure 3. Fast Lockup



BMode Switching

The fast lockup scheme changes the APLL operation mode. The selection bits must not be changed during normal operation.

If the above condition is not met, be aware that:

- Digital state machines have been designed in such a way to avoid any deadlock condition, but any switching different from the above can lead to unpredictable long transients that differ considerably from case to case.
- The lock signal behavior is not deterministic during mode switching. It goes back to a lower state or it stays high, depending on conditions. Because of the important time-consistent feature of this signal, it is reasonable to expect the APLL to react faster. The lock signal goes low only in major failure conditions, that is, absence of an input clock for a long period of time or a bad frequency range.

Missing Input Clock

A missing input clock is a major concern. It is associated with a lock going low after several 100 µs, depending on mode and condition.

It also causes the output frequency on CLKOUT to decrease quickly because the APLL tries to lock on a 0 frequency signal. However, this is limited to the APLL lock-in range. An unstable frequency can be expected in association with potential beats.

When the input clock is reactivated for a long period of time, the APLL returns to lock condition afterwards. However, this may take longer than usual, up to five times more than the specified lock-in time. In that event, the fast lockup sequence can be used.

One clock source can be switched to another in the same mode without using $\overline{\text{PWRDN}}$. Clock-gating cells must handle this switching to prevent glitches and clock uncertainty. A jitter outside the specification affects this clock switching, especially if a large guard is inserted between the clock source activations with many missing pulses and a large phase jump.

3 OMAP3.2 DPLL

The two most important uses of phase-locked loops are as synthesizers and synchronizers. Synthesizers generate a variable-frequency clock from a fixed-frequency reference clock. The output frequency of the PLL is an integer multiple or a fractional multiple (M/N) of the input reference. Synchronizers

match the on-chip clock to a system clock so that the integrity of the data transfer from one clock to the other is maintained irrespective of the clock skew and timing variations over different process, voltage, and temperature conditions.

Conventional PLLs are analog in nature in the sense that they all use a charge pump/RC filter combination to stabilize the loop. The DPLL, on the other hand, uses a digital control circuitry to stabilize the loop. The feedback control circuit generates digital signals that precisely control the oscillator settings so that the output clock locks-in phase and frequency to the input reference clock.

3.1 Features

The OMAP3.2 DPLL has the following features:

- Frequency-multiplier mode and direct-clock divider mode
- Programmable reset settings
- Idle mode, to save power
- Lock signal, to indicate that the DPLL has locked properly
- Fast lock reacquisition when changed from the idle to active mode, provided the temperature/voltage shifts are not significant
- Automatic switch to the bypass mode during programming/loss of lock
- Support of the TI peripheral bus (TIPB) interface

3.2 Functional Description

The control register is mapped within the MPU space. Writing to the control register causes the DPLL to immediately switch to the bypass mode if not in idle state. If the PLL_ENABLE bit of the control register is set, it begins its sequence to enter the locked mode. This prevents changing the multiple or divide values without reentering the DPLL lock sequence.

Table 5 describes the control register bits. The bit positions are read directly from the TIPB bus. The actual address of the register is programmable to any value. The register has multiple default values depending on the CLKMD pin values.

Table 5. Control Register

Base Address = 0xFFFE CF00, Offset = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
15	LS_DISABLE	Level shifter disable: 0: Level shifter in transparent mode; all signals between wrapper and DPLL core connected. 1: Level shifter in isolated mode; wrapper and DPLL core disconnected. DPLL core power supply turned off: no leakage current between VDD and VDD_DPLL. Power-on value is 0.	R/W	0x0
14	IAI	Initialize after idle: Set high: DPLL starts entire locking sequence after idle is deactivated. Set low: DPLL attempts to lock using internal delay chain setting before entering lock mode. Power-on value is 0. Bit no longer useful and must be set to 0.	R/W	0x0
13	IOB	Initialize on breako When high: DPLL switches to bypass mode and starts new locking sequence if DPLL core indicates lock lost. When low: DPLL continues ro putput synthesized clock even if core indicates lock lost but BREAKLN remains low. Power-on value is 1.	R/W	0x1
12	TEST	Controls test output clock on TCLKOUT pin: Test = 0: TCLKOUT = CLKOUT when in test mode Test = 1: TCLKOUT = CLKOUT/32 when in test mode Test = X: TCLKOUT = 0 when not in test mode	R/W	0x0

Table 5. Control Register (Continued)

Base Address = 0xFFFE CF00, Offset = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
11:7	PLL_MULT (4:0)	DPLL multiple value: 00000: CLKOUT = CLKREF * 1 00001: CLKOUT = CLKREF * 1 00010: CLKOUT = CLKREF * 2 00011: CLKOUT = CLKREF * 3 11111: CLKOUT = CLKREF * 31	R/W	0x0
6:5	PLL_DEV (1:0)	DPLL divide value: 00: CLKOUT = CLKREF 01: CLKOUT = CLKREF/2 10: CLKOUT = CLKREF/3 11: CLKOUT = CLKREF/4 Minimum CLKOUT frequency is 0.25 * CLKREF. When PLL_MULT(4:0) = 0 or 1, CLKOUT is not synthesized by the DPLL, but is simply a divided-down version of CLKREF. Affects lock mode only.	R/R	0x0
4	PLL_ENABLE	DPLL enable: 0: Switched DPLL to bypass mode 1: Requests DPLL to enter lock mode: DPLL starts locking sequence and changes to DPLL synthesized value after locking CLKOUT.	R/W	0x0
3:2	BYPASS_DIV (1:0)	Determines CLKOUT frequency when in bypass mode: 00: CLKOUT = CLKREF 01: CLKOUT = CLKREF/2 1X: CLKOUT = CLKREF/4	R/W	0x0

Table 5. Control Register (Continued)

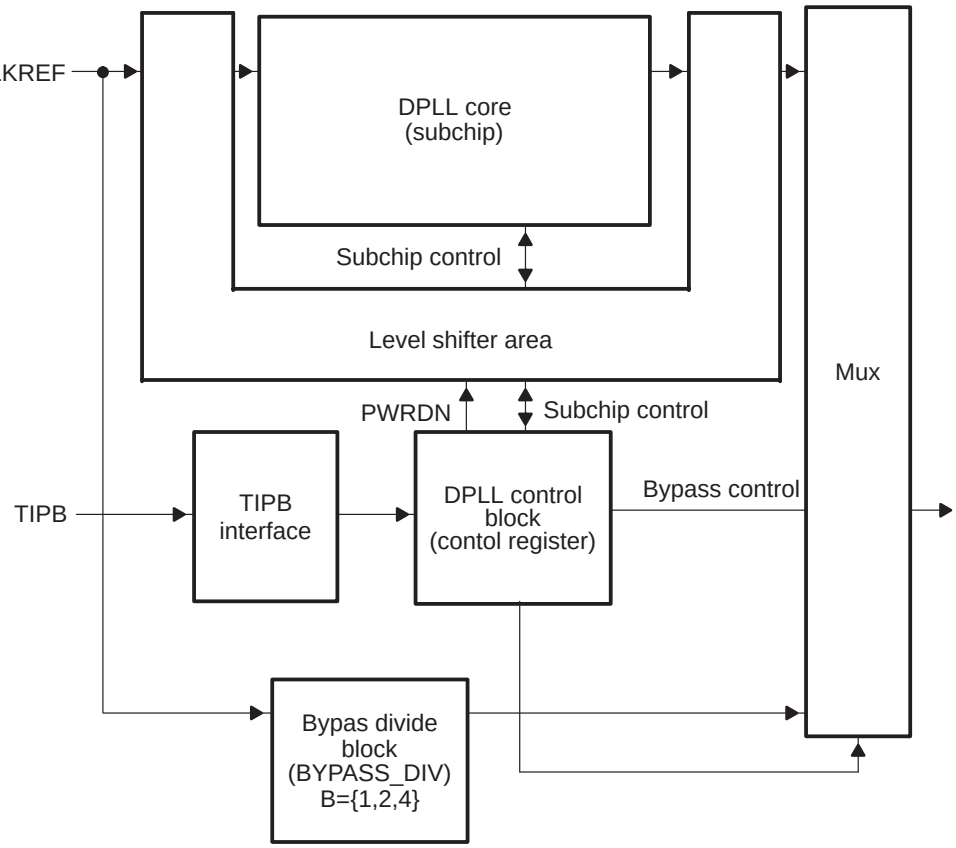
Base Address = 0xFFFE CF00, Offset = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
1	BREAKLN	Indicates break: 0: DPLL has broken lock for some reason. 1: Lock condition has been restored or write to control register has occurred.	R	0x0
0	LOCK	Indicates lock status: 0: DPLL in bypass mode and CLKOUT contains divided-down output clock. 1: DPLL in Lock mode and CLKOUT is desired synthesized clock frequency.	R	0x0

Because the dedicated VDD for the DPLL (VDD_DPLL) is either greater or less than the VDD core, a collar is implemented to isolate the DPLL. This collar includes level shifters that adapt the incoming control signals and the DPLL outputs to the appropriate level (see Figure 4).

- When enabled (LS_DISABLE = 0), the level shifter is seen as a buffer.
- When disabled (LS_DISABLE=1), the level shifter is in isolated mode and its output is grounded.

  LS_DISABLE is a read/write control bit in the DPLL control register. Its reset value is 0.

Figure 4. DPLL



The DPLL has two modes of operation, bypass mode and lock mode.

- In the bypass mode, clkout=clkref divided by 1, 2, or 4. Bypass mode saves power because the DPLL is disabled. This mode also provides an output clock while the DPLL circuitry is locking.
  - clkout = clkref for BYPASS_DIV[1:0] = 00
  - clkout = clkref/2 for BYPASS_DIV[1:0] = 01
  - clkout = clkref/4 for BYPASS_DIV[1:0] = 1X
- In the lock mode, the DPLL provides a synthesized output frequency that is locked to the input reference. Lock mode is entered if the PLL_ENABLE bit of the control register is set and the locking sequence is completed. In this mode, the clkout contains a synthesized clock frequency as defined below:
  - clkout = PLL_MULT / (PLL_DIV+1) * clkref for 1 < PLL_MULT <= 31.
  - clkout = 1 / (PLL_DIV + 1) * clkref for PLL_MULT = 0 or 1

For PLL_MULT = 0 or 1, the clkout is not synthesized. Hence, the output -clock duty cycle (clkout) is directly dependent on the input-clock duty cycle (clkref).

The lock times depend on the values of PLL_MULT and PLL_DIV and the clkout frequency as given below :

Lock time in number of clkref cycles:

clkref clocks=4N(11D+28)

where D=1+$\log_2$(N/(fin*M*xmin)) rounded up to the nearest positive integer (7 max), and fin is the input clock frequency.

The value of xmin in the equation for D is dependent on technology. It must be set to the delay of A with 10 stages in the delay chain measured in min delay conditions.

- 1.5 V xmin = 5.6 ns

Example:

- 1.5 V
- CKref = 10 MHz
- N = 1
- M = 2

D = 1+$\log_2$(1/(2*10E6*5.6E-9)) = 1+$\log_2$(8.93) = 4.158 = 5 (rounded up)

Number of clkref clocks = 4(11(5)+28) = 332

Time = (332)(100 nsec) = 33.2 µsec

Each time the DPLL control register is written to, the mode in which the DPLL operates changes automatically. If the DPLL is operating in the synthesizer mode, it switches automatically to the bypass mode. Depending on the new control register content, the DPLL either initiates a new lock sequence or remains in the bypass mode (see Figure 5).

Figure 5. Operational Flow

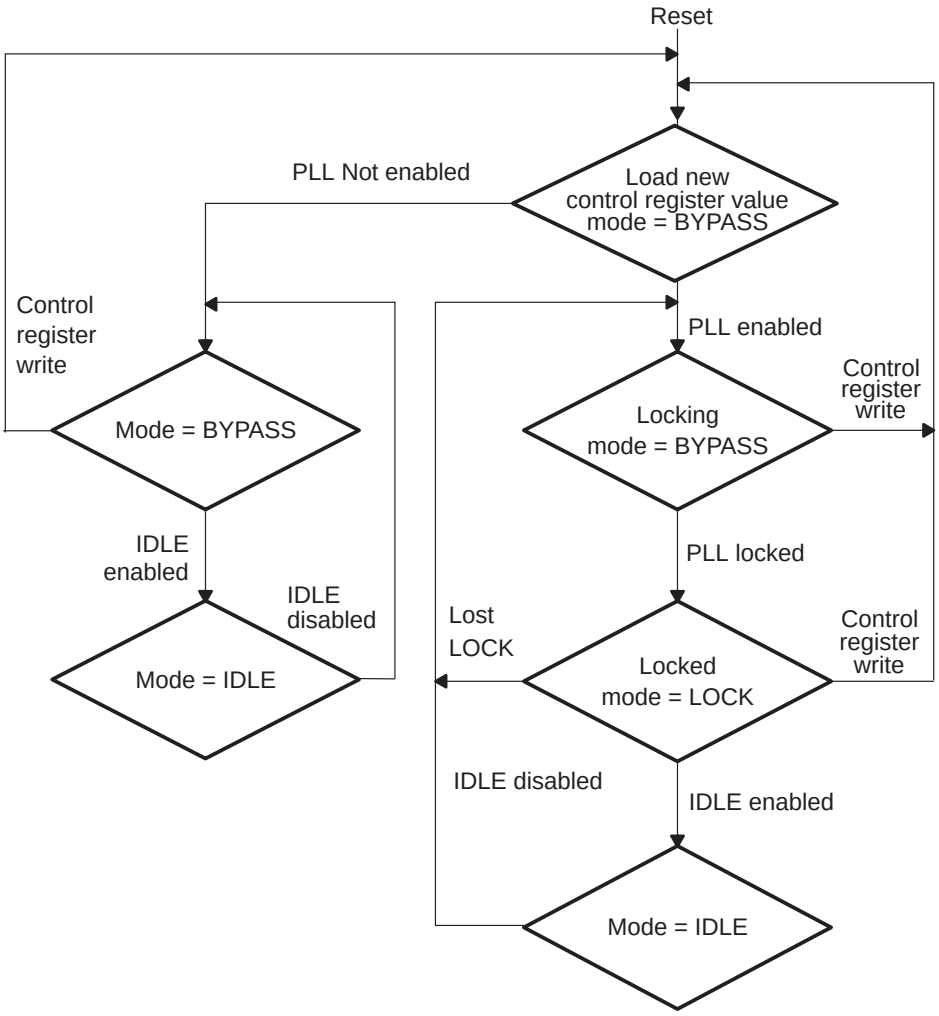


Table 6 lists the clock timings.

Table 6. Clock Timings

Description	Value	Unit
CLKREF duty cycle	From 40 to 60	%
Min CLKREF frequency	2.5	MHz
Max CLKREF frequency	50	MHz
Max CLKOUT frequency	250	MHz

4 Low-Dropout Voltage Regulator

LDO005 is a linear voltage regulator that supplies the OMAP3.2 DPLL macro. This LDO uses peripheral supply input voltage to make the DPLL a quiet power supply.

LDO005 contains:

- Embedded voltage and current reference circuit
- Adaptive biasing error amplifier as voltage regulator
- Offset steady comparator
- Initialization circuit to ensure local powerdown during VDD fast rampup
- Level shifter
- Bypass switch

The regulated supply is delivered to DPLL macros and is available on a unique bond pad. (LDO005 is a complex I/O cell). A decoupling capacitor of 1 µF must be connected externally between the pin of the cell and ground.

The LDO is bypassed when it is in power-down mode because the node VOUT is in high-impedance mode. In this configuration, the DPLL circuit is supplied from the pad (external power supply). To support this mode of operation, the LDO is also in power-down mode after ramp-up of VDD, to avoid potential contention between VDDS and VOUT.

The cell operates in sleep mode when sleep input is high. In this mode, VDD is shorted to VOUT with a resistive switch, and the voltage regulator is kept in power down. This mode retains the status in the registers of the DPLL macros when they are in idle mode (they do not pull current from the power line).

Figure 6 shows the LDO005. Table 7 describes the pins.

Figure 6. LDO005

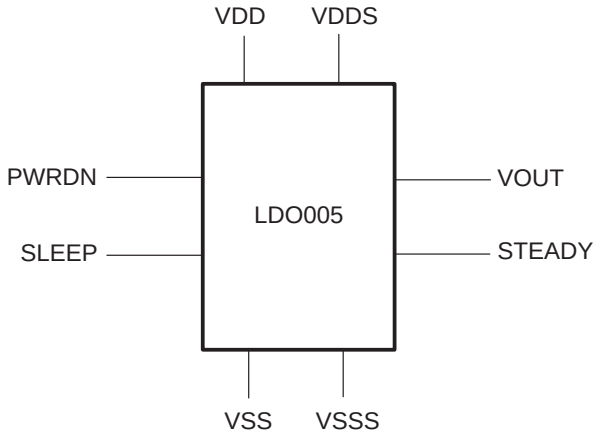
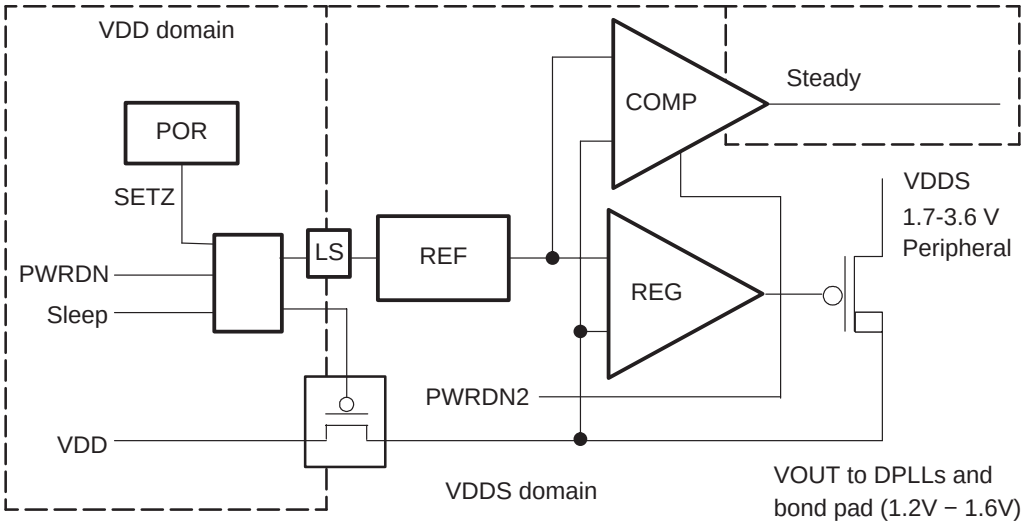


Table 7. LDO005 Pins

Name	Type		Description
VDD	Power signal ring	I/O	Positive core power supply
VDDS	Power signal ring	I/O	Positive periphery power supply input voltage
VOUT	Power signal/pad side	Core	Positive output voltage connected to DPLL power supply and bond pad
VSS	Power signal ring	I/O	Core ground
VSSS	Power signal ring	I/O	Periphery ground connected to DPLLs ground
PWRDN	Digital signal input	Core	Powerdown mode when PWRDN is high
SLEEP	Digital signal input	Core	Control input bypass VDD core voltage to VOUT: active high
STEADY	Digital signal output	Core	Output flag: high when the regulator is active

Figure 7 shows the LDO005 block.

Figure 7. LDO005 Block



The reference is a circuit that delivers voltage and current to the regulator and steady comparator. A local comparator within the reference powers up the regulator and steadies the comparator once the reference is settled using the PWRDN2 signal.

The voltage regulator uses an adaptive biasing technique that has a quiescent current linearly dependent upon the load current.

The voltage comparator has an inherent dc offset. Therefore, the steady signal is logic high when $V_{OUT} > V_{OUTfinal-offset}$ to indicate that the output voltage has almost reached its steady state.

A power-on reset circuit sets the status of the LDO in power down during ramp-up of VDD, before the status of the control inputs PWRDN and SLEEP are in a stable state.

The sleep input signal turns the voltage regulator to power-down mode to cut its quiescient current and turns on the PMOS switch between VDD and VOUT. During the transition from sleep to active modes, the external capacitor holds the voltage on VOUT to the proper level to ensure that the DPLL macros retain the status on the internal registers for fast locking. When sleep input is low, the PMOS switch is off.

Table 8. Mode Selection

PWRDN	SLEEP	Mode
L	L	Application, active
H	Don’t care	Powerdown
L	H	Sleep

4.1 Timing Diagrams

Figure 8. VDD_CORE Ramps First

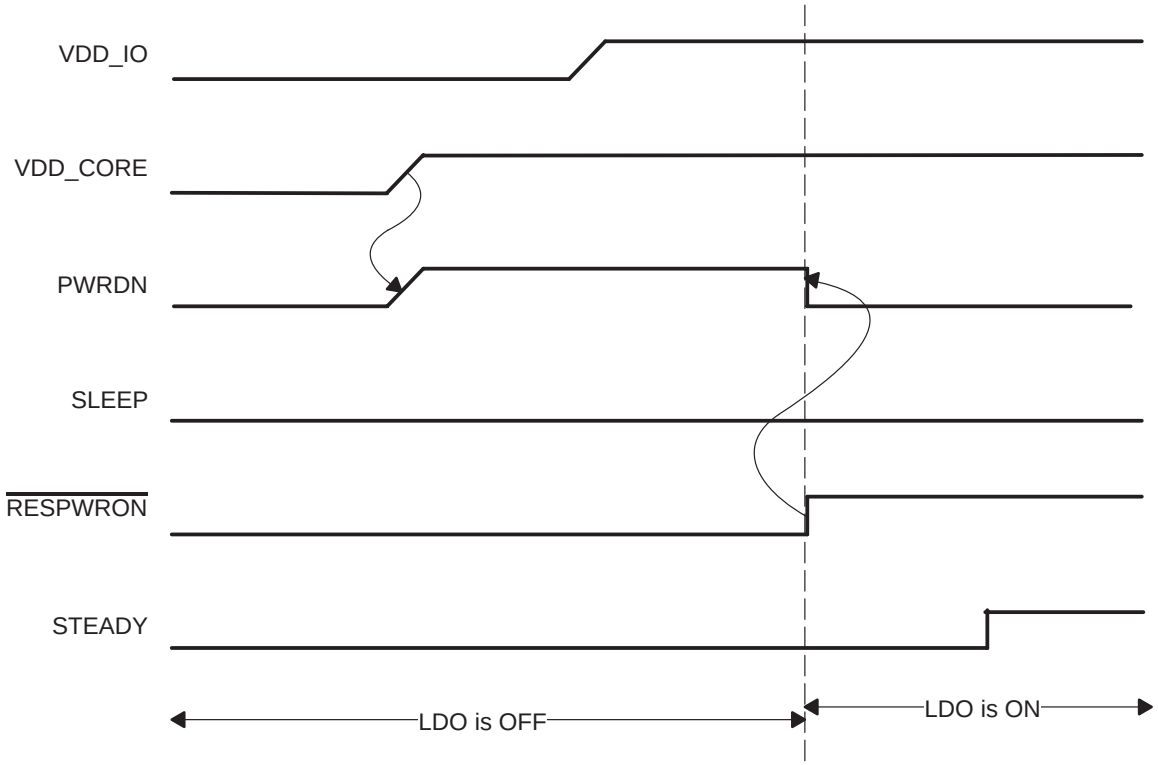
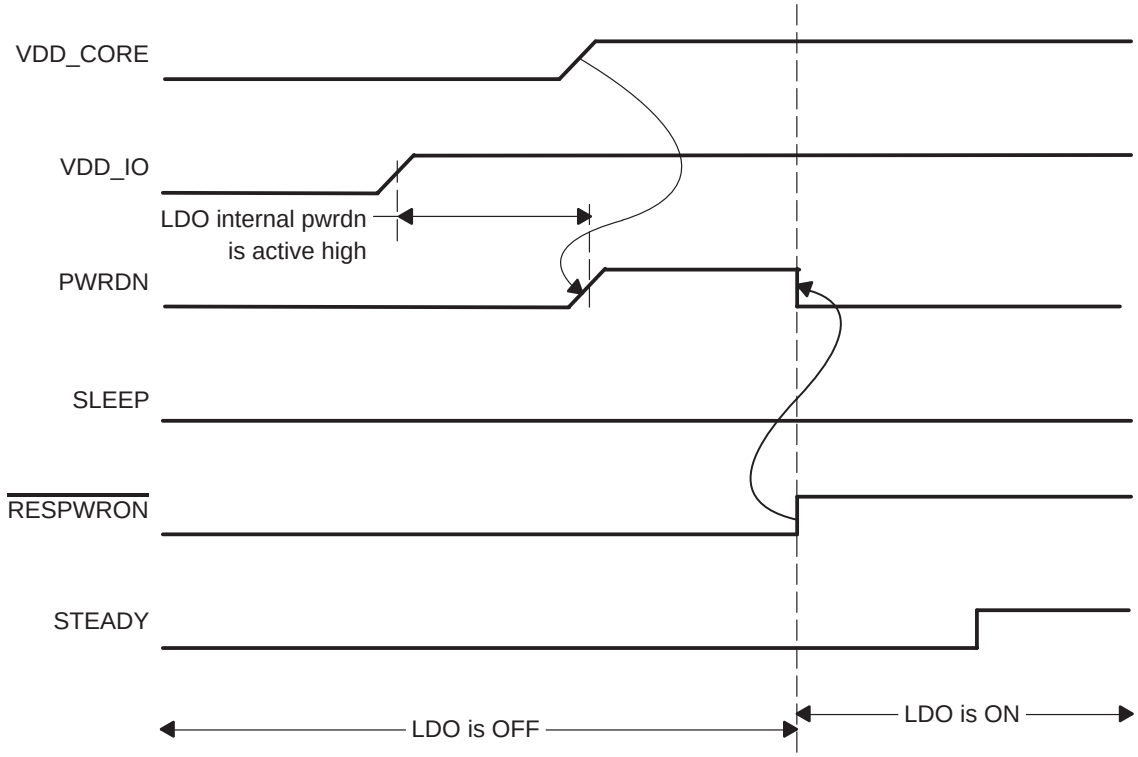


Figure 9. VDD_IO Ramps First



5 OMAP5912 Clock Architecture

The OMAP5912 clock architecture includes the functional and interface clock distribution for the OMAP5912 peripherals.

OMAP5912 receives its system clock from an oscillator or an external squarewave input clock. The system clock first goes through the ultralow-power device module (ULPD), which is responsible for power-mode transitions and clock management. System clock frequencies are 12, 13, or 19.2 MHz. OMAP5912 is also clocked by a 32-kHz clock used by the ULPD finite state machine (FSM) and for specific clocking needs such as the real−time counter (RTC) or the general-purpose timers.

The ULPD output clocks are connected directly to a few peripherals with specific clock requirements. The main system clock output of the ULPD is connected to the MPU subsystem (often referred to as the OMAP 3.2 gigacell). The 32−kHz clock is also one of the ULPD outputs. OMAP 3.2 gigacell is responsible for controlling, multiplying, and distributing clocks to the remaining peripherals.

The ULPD (see chapter 6) manages transitions among deep sleep mode, big sleep mode, and awake mode. Transitions are triggered by external events (resets and clock requests) or by internal events (software reset, watchdog time-out, and software clock requests). Some ULPD outputs can be sent to external power management devices to adjust the OMAP5912 internal corevoltage for optimum power dissipation, versus on-chip activity.

5.1 Reset Modes and Clocking Options

Reset mode 0 has several clocking options while reset mode 1 is much more restrictive.

Table 9. Clocking Options with respect to Reset Modes

Reset Mode = 0	12 MHz	13 MHz	19.2 MHz
OSC1_IN support via crystal?	Yes	Yes	Yes
OSC1_IN support w/external clock source?	Yes	Yes	Yes
SYS_CLK_IN support w/external clock source?	No	No	No
Reset Mode = 1	**12 MHz**	**13 MHz**	**19.2 MHz**
OSC1_IN support via crystal?	No	No	No
OSC1_IN support w/external clock source?	No	No	No
SYS_CLK_IN support w/external clock source?	No	No	Yes

5.2 External System Clock with Reset Mode 0

The OMAP5912 system clock can be driven at 12MHz, 13MHz, or 19.2MHz. For an external system clock, the following hardware connections are used (see Figure 10). Figure 10 applies to reset mode 0 only.

- Hardware considerations:
  - The external clock is applied to the OSC1_IN pin (ball Y2).
  - The OSC1_OUT pin (ball W3) must be left unconnected.
- Software considerations:
  - FUNC_MUX_CTRL_D(2:0) register bits should be 000 to ensure that ball Y4 is *not* pin multiplexed as the SYS_CLK_IN.
  - Set CONF_OSC1_PWRDN_R to 1 in the MOD_CONF_CTRL_1 register so that the oscillator circuit is disabled, reducing power consumption.
  - Set CONF_OSC1_GZ_R to 0 in MOD_CONF_CTRL_1 (when 1, this bit disables square input clocks on the OSC1_IN pin).

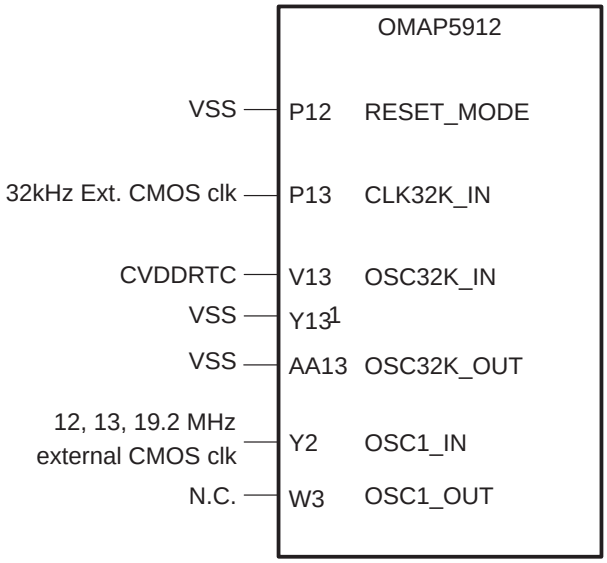
- During power-on reset, the ULPD OSC1 delay timer is reset to 32 ms. Thus, the system clock is not distributed before the timer reaches 0. During this period, the device does not start operation (or transition out of deep sleep state in the FSM of the ULPD). The ULPD OSC1 delay timer can be reprogrammed to 0 in the SETUP_ANALOG_CELL3_REG after the power−on reset has occurred.

5.3 External 32-kHz Clock with Reset Mode 0

The OMAP5912 32-kHz clock can be driven by an external squarewave clock. See Figure 10 for the hardware connections of the external 32-kHz clock. This figure applies to reset mode 0 only.

- Hardware considerations:
  - The external clock is applied to the CLK32K_IN pin (ball P13).
  - OSC32K_IN must be tied to CVDDRTC (see Chapter 22 for a listing of power supply voltages on OMAP5912).
  - OSC32K_OUT must be tied to VSS.
  - Ball Y13 can be connected to board ground (VSS).
- Software considerations:
  - Set OSC32K_PWRDN_R to 1 in RTC_OSC_REG for power-saving purposes.

Figure 10. External CMOS Clock Connections



Notes: 1) Ball Y13 may be connected to board ground.

5.4 Using the Internal Oscillator for the System Clock with Reset Mode 0

The system clock can be 12MHz, 13MHz, or 19.2MHz. To drive the system clock with the internal oscillator circuitry, the following hardware connections are used (see Figure 11). This figure applies to reset mode 0 only.

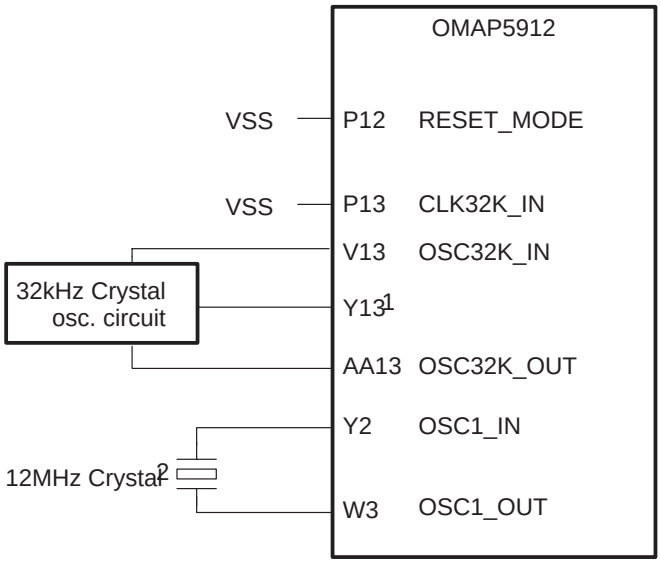
- Hardware considerations:
  - The crystal circuit is applied to the OSC1_IN pin (ball Y2) and the OSC1_OUT pin (ball W3).
- Software considerations:
  - FUNC_MUX_CTRL_D(2:0) register bits should be 000 to ensure that ball Y4 is NOT pin multiplexed as the SYS_CLK_IN.
  - Ensure CONF_OSC1_PWRDN_R is 0 in the MOD_CONF_CTRL_1 register so that the oscillator circuit is enabled.
  - Set CONF_OSC1_GZ_R to 0 in MOD_CONF_CTRL_1 (setting this bit to 1 will also disable the oscillator circuit).

5.5 Using the Internal Oscillator for a 32-kHz Clock with Reset Mode 0

The OMAP5912 32-kHz clock can be driven by an internal oscillator and crystal. See Figure 11 for the hardware connections of the external 32-kHz clock. This figure applies to reset mode 0 only.

- Hardware considerations:
  - The crystal circuit is applied to the OSC32K_IN pin (ball V13) and the OSC32K_OUT pin (ball AA13).
  - Ball Y13 must not be connected to board ground.
  - CLK32K_IN (ball P13) must be tied low directly or with a pulldown. Any activity on this pin will corrupt the 32-kHz clock.
- Software considerations:
  - Ensure OSC32K_PWRDN_R is 0 in the RTC_OSC_REG register.

Figure 11. Internal Oscillator Clock Connections



Notes: 1) Ball Y13 *must not* be connected to board ground (VSS).

2) The crystal may be 12, 13, or 19.2MHz and the full crystal circuit was excluded for the sake of simplicity. Please see the OMAP5912 Data Manual (SPRS231) for more details on clock connections.

The pin Y13 is the oscillator circuit ground pin. This pin must not be connected to board ground when using a 32-kHz oscillator circuit.

5.6 External System Clock with Reset Mode 1

The OMAP5912 system clock is driven at 19.2 MHz only in reset mode 1. For an external system clock, the following hardware connections are used (see Figure 12). Figure 12 applies to reset mode 1 only.

- Hardware considerations:
  - The external clock is applied to the SYS_CLK_IN pin (ball Y4).

- The OSC1_IN pin (ball Y2) must be driven to VSS either by a pulldown or tied directly.
- The OSC1_OUT pin (ball W3) must be left unconnected.

❏ Software considerations:

- Before setting the COMP_MODE_CTRL_0 register to 0x0000EAEF, the FUNC_MUX_CTRL_D(2:0) register bits must be set to 110 to ensure that ball Y4 is pin multiplexed as the SYS_CLK_IN.
- The ULPD OSC1 delay timer is bypassed in reset mode 1. Consequently, there is no delay in the ULPD due to the delay timer.
- If CONF_OSC1_GZ_R = 1, the OSC1_IN pin requires a pulldown, because it is configured as output.

5.7 External 32-kHz Clock with Reset Mode 1

The OMAP5912 32-kHz clock can be driven by an external clock. See Figure 12 for the hardware connections of the external 32-kHz clock. This figure applies to reset mode 1 only.

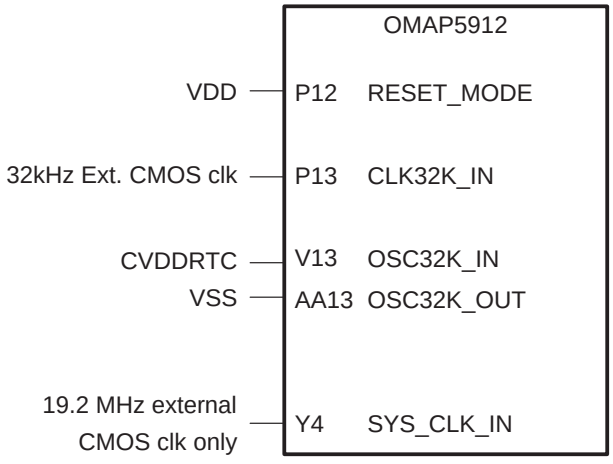
❏ Hardware considerations:

- The external clock is applied to the CLK32K_IN pin (ball P13).
- OSC32K_IN must be tied to CVDDRTC.
- OSC32K_OUT must be tied to VSS.

❏ Software considerations:

- Set OSC32K_PWRDN_R to 1 in RTC_OSC_REG for power-saving purposes.

Figure 12. External CMOS Clock Connections (Reset Mode 1)



5.8 Using the Internal Oscillator for the System Clock with Reset Mode 1

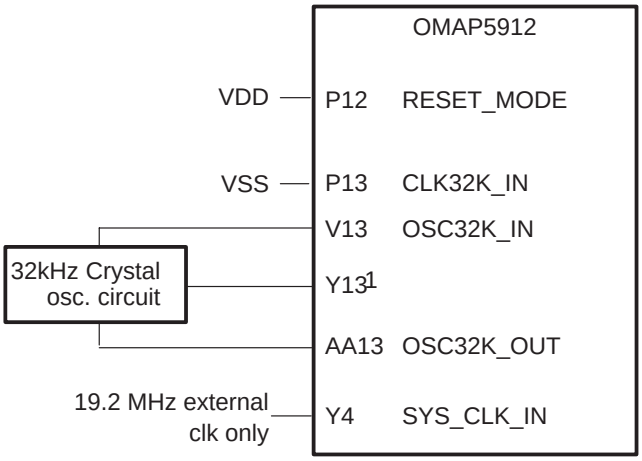
The OMAP5912 system clock cannot be driven by the internal oscillator when reset mode is 1. The internal oscillator is automatically placed in low-power mode when reset mode is 1.

5.9 Using the Internal Oscillator for a 32-kHz Clock with Reset Mode 1

The OMAP5912 32-kHz clock can be driven by an internal oscillator and crystal. See Figure 13 for the hardware connections of the external 32-kHz clock. This figure applies to reset mode 1 only.

- Hardware considerations:
  - The crystal circuit is applied to the OSC32K_IN pin (ball V13) and the OSC32K_OUT pin (ball AA13).
  - CLK32K_IN (ball P13) must be tied low directly or with a pulldown resistor. Any activity on this pin will corrupt the 32-kHz clock.

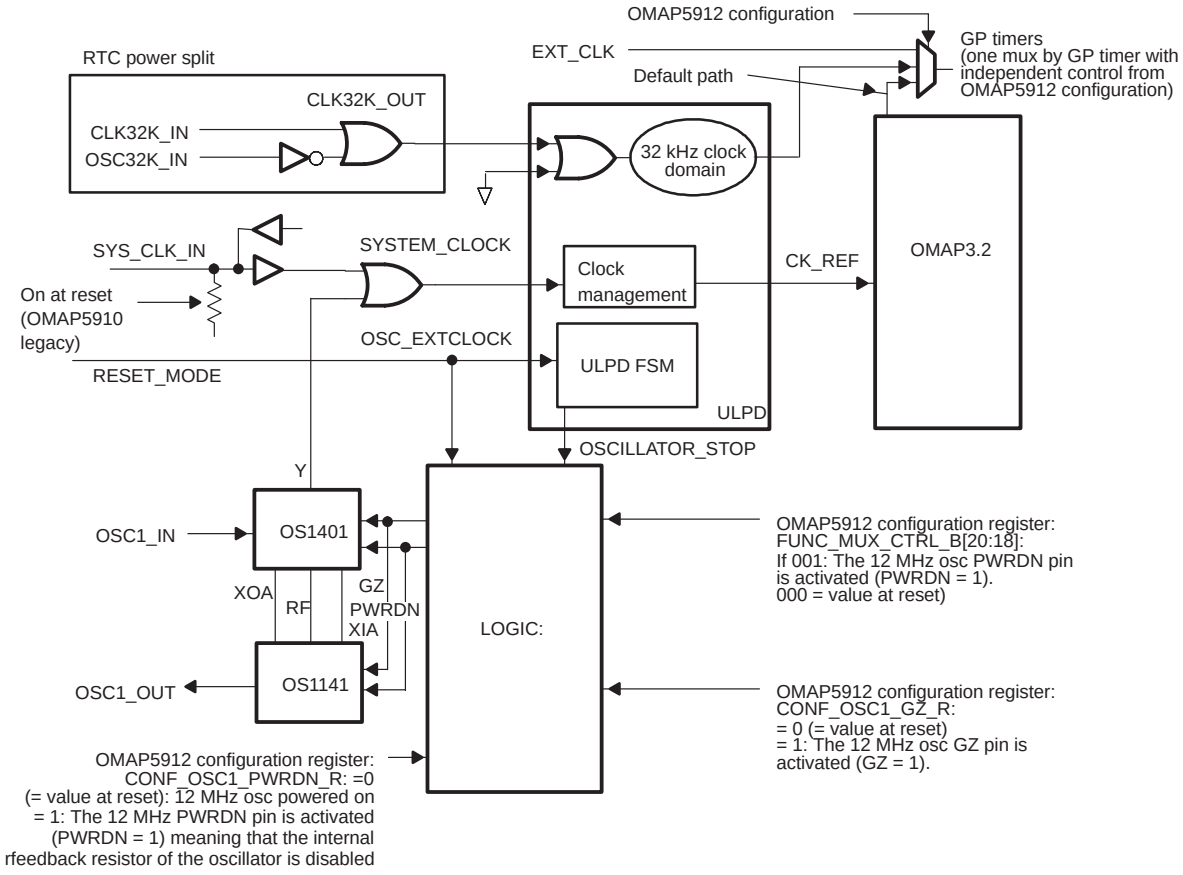
Figure 13. 32-kHz Oscillator Clock Connections (Reset Mode 1)



Notes: 1) Ball Y13 MUST NOT be connected to board ground (VSS). Please see the OMAP5912 Data Manual (SWPS012) for more details on clock connections.

See Figure 14 for a summary of the 32-kHz and system clock internal connections.

Figure 14. 32-kHz and System Clock Scheme



5.10 Clock Distribution in OMAP5912

5.10.1 Clock Inputs to ULPD

The ultralow-power device (ULPD) manages transitions between deep sleep mode, big sleep mode, and awake mode. In each mode, the clocks from the ULPD are gated or ungated either by hardware or by software. The ULPD module is also in charge of managing communication with an external power management device for dynamic voltage scaling. Communication is ensured through $\overline{\text{LOW\_POWER}}$ outputs from OMAP5912.

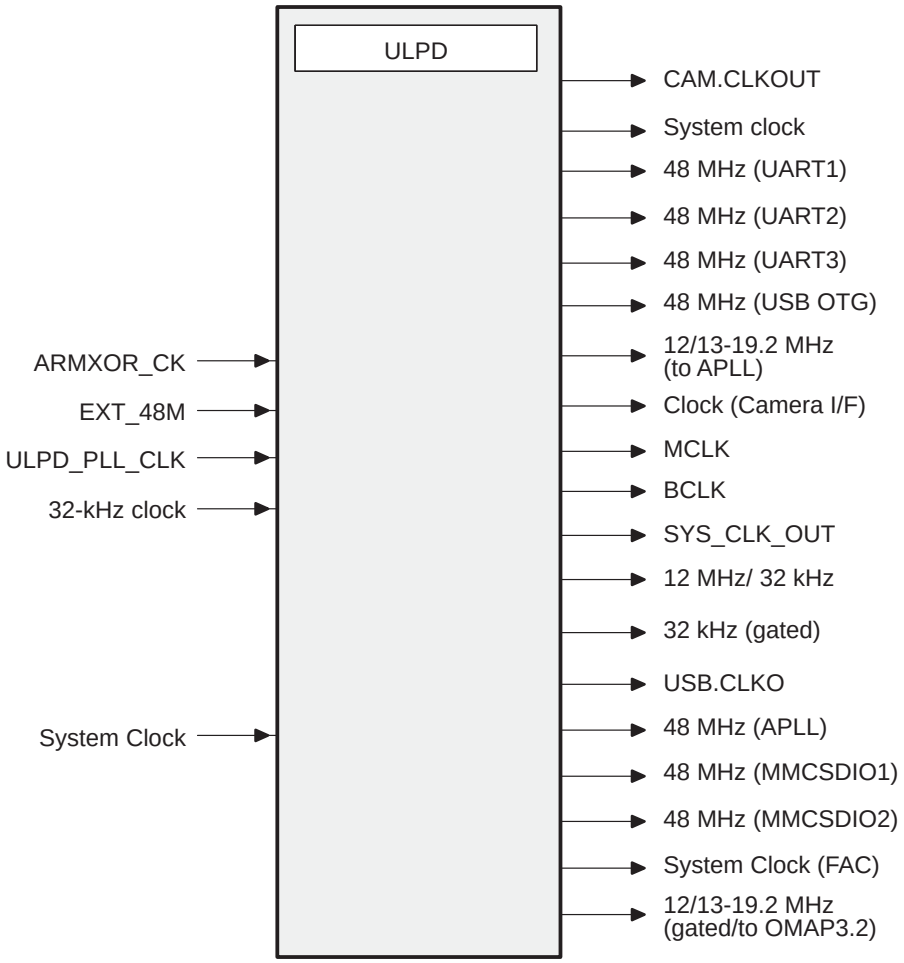
Figure 15 provides an overview of ULPD clocking. See *OMAP5912 Multimedia Processor Power Management Reference Guide* (literature number SPRU753) for further details on the ULPD.

Table 10. ULPD Input Clocks

Input Name	Description	Origin
ARMXOR_CK	MPU peripheral clock	OMAP3.2
EXT_48M	External 48-MHz clock	GPIO_14
ULPD_PLL_CLK	Clock from 96-MHz PLL	APLL
32-kHz Clock	32-kHz clock (either from 32-kHz oscillator or from external 32-kHz clock input)	RTC
System Clock	12- to19.2-MHz system clock (12−19.2-MHz oscillator or from external clock input)	I/O

Figure 15 provides descriptions of the ULPD input clocks.

Figure 15. ULPD Clocking Overview



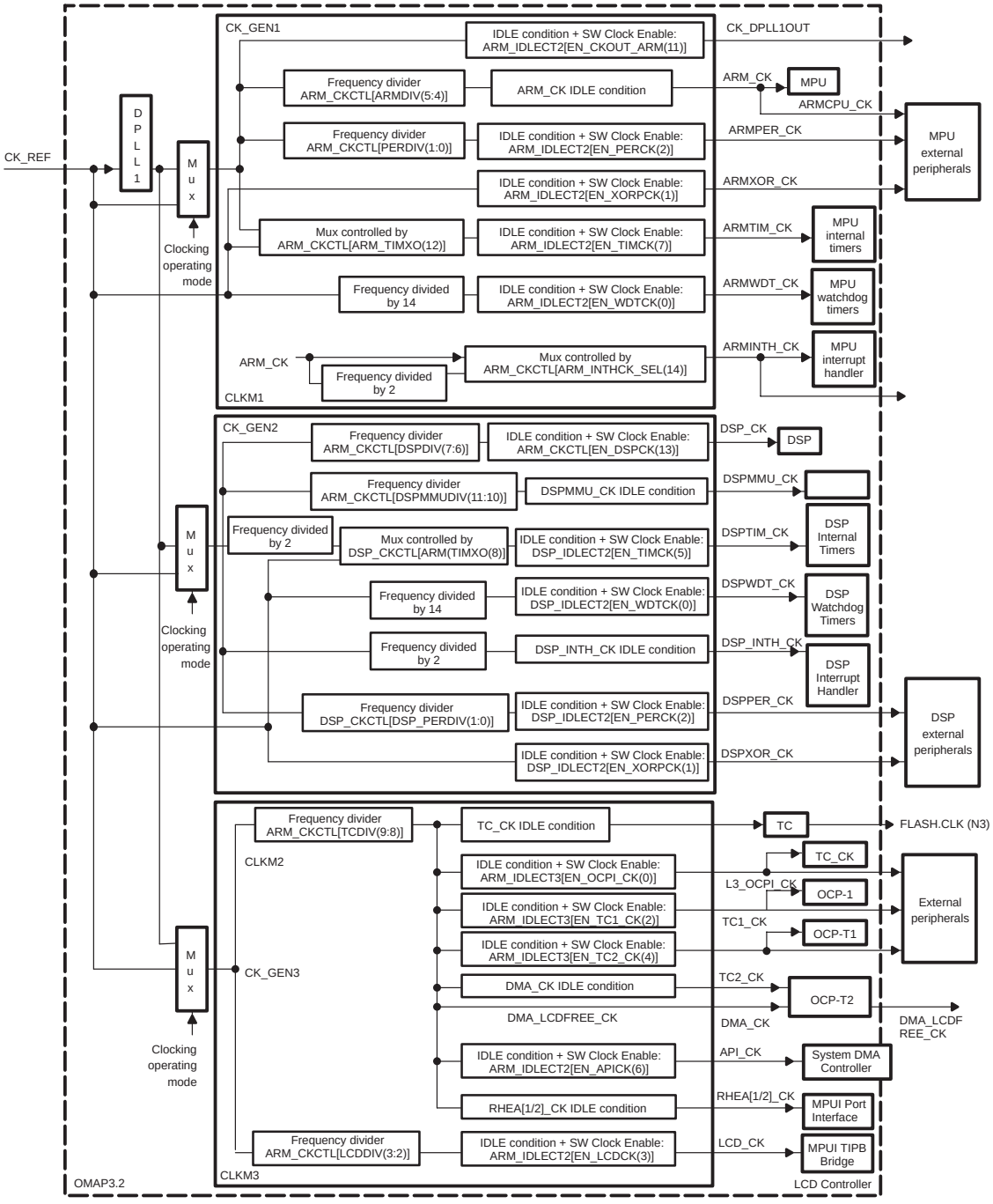
5.11 OMAP 3.2 Clocks

Table 11 describes the OMAP3.2 gigacell clocking distributed to external peripherals.

Table 11. OMAP 3.2 Clocks

Clock Name	Description
CK_REF	System clock, input to DPPL1
CK_DPLL1OUT	Clock from DPPL1, same as MPU clock
ARMCPU_CK	Clock with same frequency as MPU clock, same as ARM_CK
ARMXOR_CK	MPU peripheral clock, fixed, generated from CK_REF, can be gated
ARMPER_CK	MPU peripheral clock, divided from CK_GEN1, can be gated
ARM_INTH_CK	MPU clock interrupt handler
DSPXOR_CK	DSP peripheral clock, fixed, generated from CK_GEN2, can be gated
DSPPER_CK	DSP peripheral clock, divided from CK_GEN2, can be gated
FLASH.CLK (ball N3)	Divided TC_CK clock from EMIFS
TC1_CK	Clocks from OCP-T1
TC2_CK	Clocks from OCP-T2
L3_OCPI_CK	Same frequency as TC clock
DMA_LCDFREE_CK	Interface clock for LCD

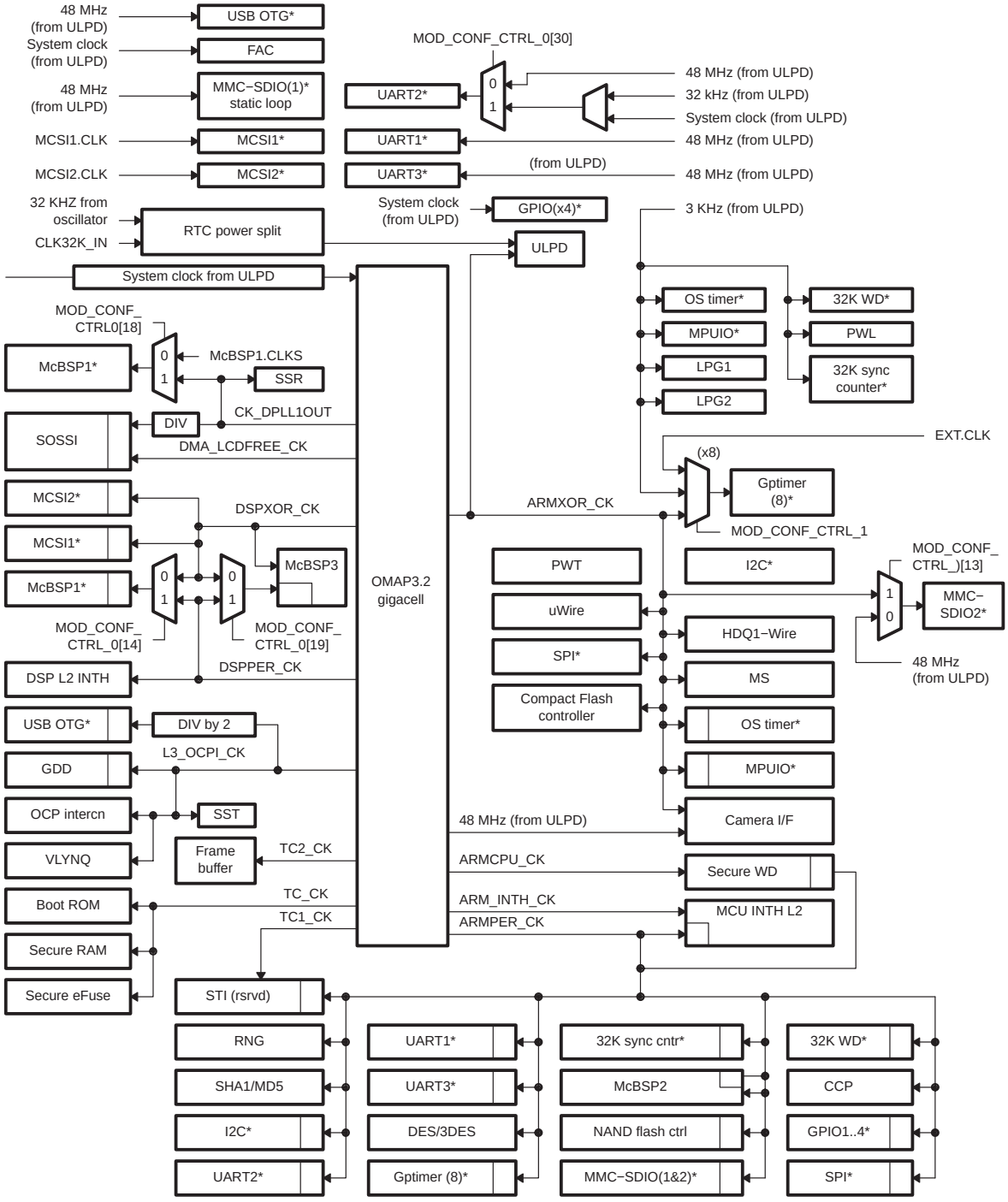
Figure 16. OMAP 3.2 Clock Generation



5.12 Clock Distribution to Peripherals

Figure 17 shows OMAP5912 peripherals (outside of the MPU subsystem) and associated clock distribution.

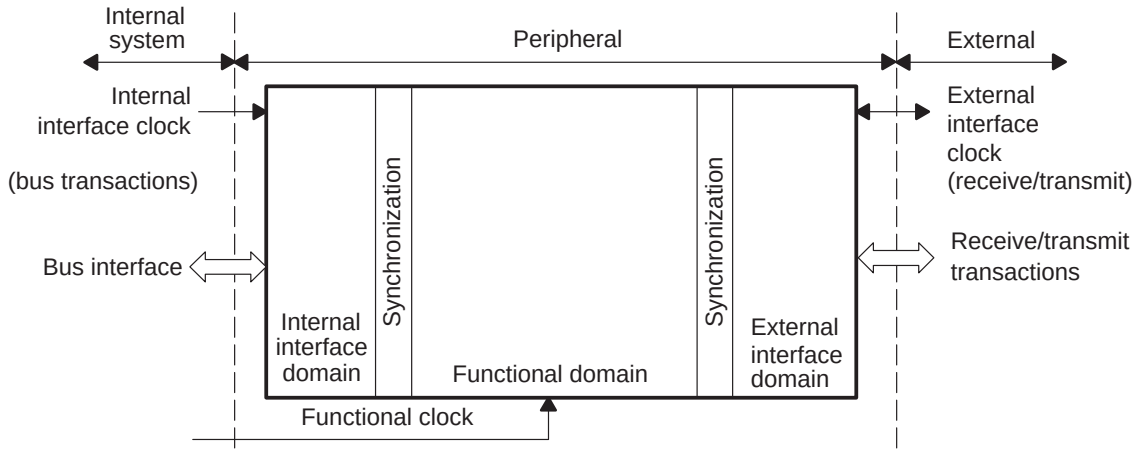
Figure 17. Peripheral Clock Distribution



5.13 Peripheral Module Clocking

Figure 18 shows the peripheral clocking.

Figure 18. Peripheral Clocking



Peripherals can have three types of clocks:

- A functional clock, which is used for peripheral internal logic clocking
- An internal interface clock, which is used for specific register clocking, that is, the OCP clock
- External interface clocks, which are used for transmit/receive protocols, for example (master/slave operation)

See each peripheral specification to better understand how the peripheral uses these clocks.

5.13.1 Peripheral Clocks

Table 12. Peripheral Clocks and Associated Controls

Peripheral Name	**Functional Clock Domain**	**Interface Clock**	**MOD_CONF_ CTRL_0 (OMAP5912 Configuration)**	**MOD_CONF_ CTRL_1 (OMAP5912 Configuration**)
ULPD	32 kHz from RTC			
	ARMXOR_CK			

Notes: 1) The GPIO functional clock is controlled by CAM_CLK_CTRL[2]. If CAM_CLK_CTRL[2] is set , the GPIO functional clock is idle.

2) The SoSSI functional clock can be divided further from 1 to 8 by setting CONF_MODE_CLK_SEL[19:17].

Table 12. Peripheral Clocks and Associated Controls (Continued)

Peripheral Name	Functional Clock Domain	Interface Clock	MOD_CONF_ CTRL_0 (OMAP5912 Configuration)	MOD_CONF_ CTRL_1 (OMAP5912 Configuration)
Boot ROM	TC_CK	Not applicable		
Secure RAM	TC_CK	Not applicable		
STI	TC1_CK	ARMPER_CK		
Secure watchdog	ARMXOR_CK	ARMPER_CK		
Real-time clock (RTC)	Ext 32-kHz clock	Not applicable		
	32-kHz oscillator			
OMAP5912 configuration	Not applicable (TIPB strobe)			
GDD	L3_OCPI_CK	L3_OCPI_CK		
SSR	CK_DPLL1OUT			
SST	L3_OCPI_CK	Not applicable		
OCP interconnect	L3_OCPI_CK	Not applicable		
VLYNQ	L3_OCPI_CK	L3_OCPI_CK		
Frame Buffer	TC2_CK	Not applicable		
General-purpose timer 1	32 kHz from ULPD EXTCLK ARMXOR_CLK	ARMPER_CK		CONF_MODE_ CLK_SEL[1:0]
General-purpose timer 2	32 kHz from ULPD EXTCLK ARMXOR_CLK	ARMPER_CK		CONF_MODE_ CLK_SEL[3:2]
General-purpose timer 3	32 kHz from ULPD EXTCLK ARMXOR_CLK	ARMPER_CK		CONF_MODE_ CLK_SEL[5:4]
General-purpose timer 4	32 kHz from ULPD EXTCLK ARMXOR_CLK	ARMPER_CK		CONF_MODE_ CLK_SEL[7:6]

Notes: 1) The GPIO functional clock is controlled by CAM_CLK_CTRL[2]. If CAM_CLK_CTRL[2] is set , the GPIO functional clock is idle.

2) The SoSSI functional clock can be divided further from 1 to 8 by setting CONF_MODE_CLK_SEL[19:17].

Table 12. Peripheral Clocks and Associated Controls (Continued)

Peripheral Name	Functional Clock Domain	Interface Clock	MOD_CONF_ CTRL_0 (OMAP5912 Configuration)	MOD_CONF_ CTRL_1 (OMAP5912 Configuration)
General-purpose timer 5	32 kHz from ULPD EXTCLK ARMXOR_CLK	ARMPER_CK		CONF_MODE_ CLK_SEL[9:8]
General-purpose timer 6	32 kHz from ULPD EXTCLK ARMXOR_CLK	ARMPER_CK		CONF_MODE_ CLK_SEL[11:10]
I^2C multimaster/slave	ARMXOR_CK	ARMPER_CK		
SPI master/slave	ARMXOR_CK	ARMPER_CK		
32-kHz synchro counter	32 kHz from ULPD	ARMPER_CK		
MMC/SDIO2	ARMXOR_CK 48 MHz from ULPD	ARMPER_CK	CONF_MOD_ MMC2_CLK_ SEL_R	
UART2	32 kHz from ULPD System clock from ULPD 48 MHz from ULPD	ARMPER_CK	CONF_MOD_ UART2_CLK_ MODE_R	
GPIO4	CK_REF	ARMPER_CK		
GPIO1	CK_REF	ARMPER_CK		
GPIO2	CK_REF	ARMPER_CK		
GPIO3	CK_REF	ARMPER_CK		
UART1	48 MHz from ULPD	ARMPER_CK		
UART3	48 MHz from ULPD	ARMPER_CK		
General-purpose timer 7	32 kHz from ULPD EXTCLK ARMXOR_CLK			CONF_MODE_ CLK_SEL[13:12]

Notes: 1) The GPIO functional clock is controlled by CAM_CLK_CTRL[2]. If CAM_CLK_CTRL[2] is set , the GPIO functional clock is idle.

2) The SoSSI functional clock can be divided further from 1 to 8 by setting CONF_MODE_CLK_SEL[19:17].

Table 12. Peripheral Clocks and Associated Controls (Continued)

Peripheral Name	Functional Clock Domain	Interface Clock	MOD_CONF_ CTRL_0 (OMAP5912 Configuration)	MOD_CONF_ CTRL_1 (OMAP5912 Configuration)
General-purpose timer 8	32 kHz from ULPD EXTCLK ARMXOR_CLK			CONF_MODE_ CLK_SEL[15:14]
PWL	32 kHz from ULPD	Not applicable		
LPG1	32 kHz from ULPD	Not applicable		
LPG2	32 kHz from ULPD	Not applicable		
RNG	ARMPER_CK	Not applicable		
SHA1/MD5	ARMPER_CK	Not applicable		
DES/3DES	ARMPER_CK	Not applicable		
CCP	ARMPER_CK	Not applicable		
PWT	ARMXOR_CK	Not applicable		
CompactFlash controller	ARMXOR_CK			
32-kHz watchdog	32 kHz from ULPD	ARMPER_CK		
OS timer	32 kHz from ULPD	ARMXOR_CK		
USB On-The-Go	48 MHz from ULPD	L3_OCPI_CK divided by 2		
µWire	ARMXOR_CK			
HDQ/1-Wire	ARMXOR_CK			
Camera IF	ARMXOR_CK 48 MHz from ULPD			CONF_CAM_ CLKMUX_R
McBSP2	ARMPER_CK	ARMPER_CK		
MPU interrupt handler level 2	ARM_INTH_CK	ARMPER_CK		

Notes: 1) The GPIO functional clock is controlled by CAM_CLK_CTRL[2]. If CAM_CLK_CTRL[2] is set , the GPIO functional clock is idle.

2) The SoSSI functional clock can be divided further from 1 to 8 by setting CONF_MODE_CLK_SEL[19:17].

Table 12. Peripheral Clocks and Associated Controls (Continued)

Peripheral Name	Functional Clock Domain	Interface Clock	MOD_CONF_ CTRL_0 (OMAP5912 Configuration)	MOD_CONF_ CTRL_1 (OMAP5912 Configuration)
MMC/SDIO1	48 MHz from ULPD	ARMPER_CK		
FAC	ARMXOR_CK (gated in ULPD)	Not applicable		
MPUIO	32 kHz from ULPD			
DSP interrupt handler 2.1	DSPPER_CK	DSPPER_CK		
McBSP1	MCBS1_CLKS	DSPPER_CK	CONF_MOD_ MCBSP1_CLK_ SEL_R	
	CK_DPLL1OUT	DSPXOR_CK	CONF_MOD_ MCBSP1_CLKS_ SEL_R	
McBSP3	DSPXOR_CK	DSPPER_CK	CONF_MOD_ MCBSP3_CLK_ SEL_R	
		DSPXOR_CK		
MCSI1	MCSI1_BCLK	DSPXOR_CK		
MCSI2	MCSI2_CLK	DSPXOR_CK		
SoSSI	CK_DPLL1OUT	DMA_ LCDFREE_CK		CONF_MODE_ CLK_SEL[16]
				CONF_MODE_ CLK_SEL[19:17][2]

Notes: 1) The GPIO functional clock is controlled by CAM_CLK_CTRL[2]. If CAM_CLK_CTRL[2] is set , the GPIO functional clock is idle.

2) The SoSSI functional clock can be divided further from 1 to 8 by setting CONF_MODE_CLK_SEL[19:17].

5.13.2 Clock Gating in ULPD

Table 13 shows clocks from the ULPD that can be gated or ungated. Clocks can be made active if either one of the respective clock selections is high or if there is a wake-up request. Wake-up requests can be hardware or software and can be disabled by software. For software requests, disables, and associated clocks, see Table 13. In addition, some software requests can be set up in the OMAP5912 configuration module.

Table 13. Hardware Requests

Hardware Request	Clock Requested	Event	Notes
PERIPH_NREQ (internal)	Input clock to OMAP3.2 DPLL	Falling edge detected on the UART2 RX	ULPD state machine transitions to awake.
	UART_MCKO (internal)		UART_MCKO transitions from 32-kHz (deep sleep mode) to system clock.
	CAM.EXCLK		
WAKEUP_NREQ (internal)	Input clock to OMAP3.2 DPLL	Any unmasked interrupt from GPIO	ULPD state machine transitions to awake.
	UART_MCKO		
	CAM.EXCLK		
BCLKREQ	System clock on ball Y15 (BCLK)	External input set high	
MCLKREQ	System clock on ball V5 (MCLK)	External input set high	
USB_MCLK_REQ	System clock	Permanent request. Can be disabled by software.	Interface clock to access USB register file
USB_DPLL_MCLK_REQ	48 MHz for USB	USB cable detection/USB resume	48-MHz USB clock. Refer to USB OTG integration.

By default, USB_MCLK_REQ requests the internal interface clock. When low, it enables the ULPD to enter deep sleep mode, as the USB is in idle mode and USB functionality is no longer required (OTG functionality is no longer used, the idle modes are activated, the USB host is not used, and the USB device is not used, is unconnected, or is put in suspend after the software acknowledgment).

By default, USB_DPLL_MCLK_REQ requests the clock for 48 MHz for the USB core clocks. When low, it enables the ULPD to enter deep sleep mode,

as the USB bus is no longer used and USB functionality is no longer required. (OTG functionality is no longer used, the idle modes are activated, the USB host is not used, and the USB device is not used, is unconnected, or is put in suspend).

Table 14 summarizes software clock requests that are mapped in the OMAP5912 configuration module and in the ULPD module and that differ from the generic ULPD software requests in SOFT_REQ_REG.

Table 14. Software Requests

Software Requests	Clock Requested	Event	Origin
UART1_DPLL_REQ	48 MHz for UART1	MOD_CONF_CTRL0[29] set to 1	OMAP5912 configuration
UART2_DPLL_REQ	48 MHz for UART2	MOD_CONF_CTRL0[30] set to 1	OMAP5912 configuration
UART3_DPLL_REQ	48 MHz for UART3	MOD_CONF_CTRL0[31set to 1]	OMAP5912 configuration
MMC_DPLL_CLK	48 MHz for MMCSDIO1	MOD_CONF_CTRL_0[23] set to 1	OMAP5912 configuration
MMC2_DPLL_CLK	48 MHz for MMCSDIO2	MOD_CONF_CTRL_0[20] set to 1	OMAP5912 configuration
CONF_MOD_USB_ HOST_HHC_ UHOST_EN	48 MHz for USB	CONF_MOD_USB_HOST_HHC_ UHOST_EN_R set to 1	OMAP5912 configuration. Kept in OMAP5912 for software compatibility with OMAP5910
CONF_CAM_ CLKMUX_EN_R	Clock for camera I/F	CONF_ CAM_CLKMUX_R set to 1	OMAP5912 ULPD

As long as at least one of the clock requests is active, the ULPD is prevented from entering deep sleep mode.

In deep sleep mode, all clocks are idle except UART2 UART_MCKO, which is derived from the 32-kHz oscillator.

In big sleep mode, the clocks listed in Table 15 are active as long as their respective requests are enabled.

Table 15. Active Clocks in Big Sleep Mode

Module/ OMAP5912 I/O Destination	Clock Description	Active Request	Notes
BCLK	System clock or 48-MHz clock from APLL	BCLKREQ (see note 5)	When 48MHz clock is selected SDW_CLK_DIV_CTRL_ SEL[7:2] further divides BCLK. See Note 1.
MCLK	System clock or 48-MHz clock from APLL	MCLKREQ (see note 5) SOFT_REQ_REG[6] COM_CLK_DIV_CTRL_SEL[1] SOFT_REQ_REG[1]	When 48MHz clock is selected, COM_RATIO_SEL[7:2] further divides MCLK. See Note 2.
UART2	32-kHz or system clock	WAKEUP_NREQ PERIPH_NREQ SOFT_REQ_REG[5]	
USB.CLK0	EXT_48M divided by 8 48 MHz from ULPD divided by 8	Clock request to ULPD PLL SOFT_REQ_REG[0]	See Note 3.
SYS_CLK_OUT	System clock	MCLKREQ (see note 5) SOFT_REQ_REG[1]	System clock
CAM.D[7]	EXT_48M 48 MHz from ULPD	SOFT_REQ_REG[0] Clock request to ULPD PLL	External source for 48 MHz selected if CONF_DPLL_EXT_SEL=1. See Note 4.
UART1	EXT_48M 48 MHz from ULPD	MOD_CONF_CTRL0[29] SOFT_REQ_REG[9]	
UART2	EXT_48M 48 MHz from ULPD	MOD_CONF_CTRL0[30] SOFT_REQ_REG[10]	
UART3	EXT_48M 48 MHz from ULPD	MOD_CONF_CTRL0[31] SOFT_REQ_REG[11	

Table 15. Active Clocks in Big Sleep Mode (Continued)

Module/ OMAP5912 I/O Destination	Clock Description	Active Request	Notes
USB OTG	EXT_48M 48 MHz from ULPD	USB_DPLL_MCLK_REQ CONF_MOD_USB_HOST_ HHC_UHOST_EN_R SOFT_REQ_REG[8]	
MMC/SDIO1	EXT_48M 48 MHz from ULPD	MOD_CONF_CTRL_0[23] SOFT_REQ_REG[12]	
MMC/SDIO2	EXT_48M 48 MHz from ULPD	MOD_CONF_CTRL_0[20] SOFT_REQ_REG[13]	

Notes: 1) The frequency on the BCLK can be set accordingly: SDW_CLK_DIV_CTRL_SEL[7:2]. The resulting frequency is given in the following table:

SDW_CLK_DIV_CTRL_SEL[7:2] = 0X00000, BCLK = 48 MHz
SDW_CLK_DIV_CTRL_SEL[7:2] = 0X00001, BCLK = 32 MHz
SDW_CLK_DIV_CTRL_SEL[7:2] = 0X00002, BCLK = 24 MHz
SDW_CLK_DIV_CTRL_SEL[7:2] = 0X00003, BCLK = 19.2 MHz
SDW_CLK_DIV_CTRL_SEL[7:2] = 0X00004, BCLK = 16 MHz
SDW_CLK_DIV_CTRL_SEL[7:2] = 0X00005, BCLK = 13.7 MHz
SDW_CLK_DIV_CTRL_SEL[7:2] = 0X00006, BCLK = 12 MHz
SDW_CLK_DIV_CTRL_SEL[7:2] = 0X00007, BCLK = 9.6 MHz
SDW_CLK_DIV_CTRL_SEL[7:2] = 0X00008, BCLK = 8 MHz
SDW_CLK_DIV_CTRL_SEL[7:2] = 0X00009, BCLK = 6.9 MHz
SDW_CLK_DIV_CTRL_SEL[7:2] = 0X000012, BCLK = 3 MHz
SDW_CLK_DIV_CTRL_SEL[7:2] = 0X000032, BCLK = 1 MHz

2) The frequency on the MCLK can be set accordingly: COM_RATIO_SEL[7:2]. The resulting frequency is given in the following table:

COM_RATIO_SEL[7:2] = 0X00000, MCLK = 48 MHz
COM_RATIO_SEL[7:2] = 0X00001, MCLK = 32 MHz
COM_RATIO_SEL[7:2] = 0X00002, MCLK = 24 MHz
COM_RATIO_SEL[7:2] = 0X00003, MCLK = 19.2 MHz
COM_RATIO_SEL[7:2] = 0X00004, MCLK = 16 MHz
COM_RATIO_SEL[7:2] = 0X00005, MCLK = 13.7 MHz
COM_RATIO_SEL[7:2] = 0X00006, MCLK = 12 MHz
COM_RATIO_SEL[7:2] = 0X00007, MCLK = 9.6 MHz
COM_RATIO_SEL[7:2] = 0X00008, MCLK = 8 MHz
COM_RATIO_SEL[7:2] = 0X00009, MCLK = 6.9 MHz
COM_RATIO_SEL[7:2] = 0X000012, MCLK = 3 MHz
COM_RATIO_SEL[7:2] = 0X000032, MCLK = 1 MHz

3) EXT_48M is an external 48−MHz input clock multiplexed with GPIO4.
4) The 48 MHz from the APLL, which can be observed on CAM.D[7], if the observability mode is configured.
5) Please refer to section 5.14 for more information on the relationship between activating requests.

5.14 OMAP5912 Output Clocks

There are four OMAP5912 output clocks that can be used to clock external ICs. All of these output clocks are on pins that are multiplexed with other functions. Therefore, the appropriate pin multiplexing must be programmed by software (e.g. MCLK is selected on ball V5, MCLKREQ is selected on ball R10, etc.).

5.14.1 MCLK and MCLKREQ

The hardware request for MCLK, MCLKREQ, is available on ball R10, in multiplexing mode 0. It is active high. Using this request, the user can get the system clock on MCLK.

To get the system clock on MCLK using a hardware request,

- Set the CONF_MOD_COM_MCK_12_48_SEL_R bit to 0 in the MOD_CONF_CTRL_0 register.
- Set the COM_SYSCLK_PLLCLK_SEL bit to 1 in the ULPD COM_CLK_CTRL_DIV_SEL register.
- Set the DIS_COM_MCLK_REQ bit to 0 in the ULPD SOFT_DISABLE_REQ_REG register.
- Set the SOFT_COM_MCKO_REQ bit to 0 in the ULPD SOFT_REQ_REG.

Consequently, setting MCLKREQ (R10) high puts the system clock on MCLK (V5). The software can mask out MCLKREQ, and deactivate MCLK by changing SOFT_DIS_COM_MCLK_REQ. When the system clock is output on

MCLK, the software can also use the ULPD CLOCK_CTRL_REG.COM_MCLK_INV to select the inactive level of MCLK.

A software request for MCLK is also available. Depending on the software request used, MCLK can be:

- 48MHz coming from the ULPD APLL (legacy support)
- 48MHz coming from the ULPD APLL divided by a programmable ratio
- The system clock

For the system clock on MCLK with a software request, the following software procedure must be used:

- Set the CONF_MOD_COM_MCK_12_48_SEL_R bit to 0 in the MOD_CONF_CTRL_0 register.
- Set the COM_SYSCLK_PLLCLK_SEL bit to 1 in the ULPD COM_CLK_CTRL_DIV_SEL register.
- Set the DIS_COM_MCLK_REQ bit to 1 in the ULPD SOFT_DISABLE_REQ_REG register.
- Set the SOFT_COM_MCKO_REQ to 1 in the ULPD SOFT_REQ_REG register.

Consequently, the system clock is available on MCLK. Use SOFT_COM_MCKO_REQ to disable or reenable MCLK. When the system clock is output on MCLK, the software can also use the ULPD CLOCK_CTRL_REG.COM_MCLK_INV to select the inactive level of MCLK.

For a 48-MHz clock on MCLK with a software request (legacy support), the following software procedure must be used:

- Set the CONF_MOD_COM_MCK_12_48_SEL_R bit to 1 in the MOD_CONF_CTRL_0 register.
- Set the COM_SYSCLK_PLLCLK_SEL bit to 1 in the ULPD COM_CLK_CTRL_DIV_SEL register.
- Set the DIS_COM_MCLK_REQ bit to 0 in the ULPD SOFT_DISABLE_REQ_REG register.

The CONF_MOD_COM_MCK_12_48_SEL_R bit acts both as selection for 48MHz on MCLK and as a request for ULPD APLL. This sends the 48-MHz clock to MCLK. Setting DIS_COM_MCLK_REQ to 1 does not gate the output of the APLL or the MCLK clock source selection, but it does disable the APLL clock request associated to CONF_MOD_COM_MCK_12_48_SEL_R.

For a divided 48-MHz clock on MCLK with a software request, the following software procedure must be used:

- Set ULPD COM_CLK_CTRL_DIV_SEL.COM_RATIO_SEL to the desired ratio.
- Set the COM_SYSCLK_PLLCLK_SEL bit to 0 in the ULPD COM_CLK_CTRL_DIV_SEL register.
- Set the COM_ULPD_PLL_CLK_REQ to 1 in the ULPD COM_CLK_CTRL_DIV_SEL register.

Consequently, the 48MHz clock divided by the programmed ratio will be available on MCLK. Use COM_ULPD_PLL_CLK_REQ to disable or re-enable MCLK clock. Note that by setting COM_RATIO_SEL to 00000, 48MHz will be available on MCLK.

5.14.2 BCLK and BCLKREQ

The hardware request for BCLK, BCLKREQ (ball W15), is available in multiplexing mode 0. It is active high. Using this request, the user can get the system clock on BCLK (ball Y15).

To get the system clock on BCLK using a hardware request, the following software procedure is used:

- Set the COM_SYSCLK_PLLCLK_SEL bit to 1 in the ULPD SDW_CLK_CTRL_DIV_SEL register.
- Set the DIS_COM_MCLK_REQ bit to 0 in the ULPD SOFT_DISABLE_REQ_REG register.
- Set the SOFT_SDW_REQ bit to 0 in the ULPD SOFT_REQ_REG.

Consequently, setting BCLKREQ (W15) high puts the system clock on BCLK (Y15). The software can mask out BCLKREQ, and deactivate BCLK by changing SOFT_DIS_SDW_MCLK_REQ. When the system clock is output on MCLK, the software can also use the ULPD CLOCK_CTRL_REG.SDW_MCLK_INV to select the inactive level of BCLK.

A software request for BCLK is also available. Depending on the software request used, BCLK can be:

- 48MHz coming from the ULPD APLL divided by a programmable ratio.
- The system clock.

For the system clock on BCLK with a software request, the following software procedure must be used:

- Set the SDW_SYSCLK_PLLCLK_SEL bit to 1 in the ULPD SDW_CLK_CTRL_DIV_SEL register.
- Set the DIS_SDW_MCLK_REQ bit to 1 in the ULPD SOFT_DISABLE_REQ_REG register.

- Set the SOFT_SDW_REQ bit to 1 in the ULPD SOFT_REQ_REG register.

Consequently, the system clock is available on BCLK. Use SOFT_SDW_REQ to disable or reenable BCLK (regardless of the BCLKREQ pin). When the system clock is output on BCLK, the software can also use the ULPD CLOCK_CTRL_REG.SDW_MCLK_INV to select the inactive level of BCLK.

For a divided 48-MHz clock on BCLK with a software request, the following software procedure must be used. In the ULPD SDW_CLK_CTRL_DIV_SEL register:

- Set SDW_RATIO_SEL to the desired ratio.
- Set the SDW_SYSCLK_PLLCLK_SEL bit to 0.
- Set the SDW_ULPD_PLL_CLK_REQ to 1.

Consequently, the 48MHz clock divided by the programmed ratio will be available on BCLK. Use SDW_ULPD_PLL_CLK_REQ to disable or re-enable BCLK clock. By setting SDW_RATIO_SEL to 00000, 48MHz is made available on BCLK.

5.14.3 CAM_CLK_OUT

Ball Y15 can also be configured to provide an external clock to a camera sensor (CAM_CLK_OUT function, in multiplexing mode 6). The frequency of this external clock is the system clock frequency divided by a programmable ratio. It can only be enabled by software with the following procedure:

- Set the CAM_CLK_DIV field to the appropriate dividing ratio in the ULPD CAM_CLK_CTRL register.
- Set the CAM_CLOCK_ENABLE bit to 1 in the ULPD CAM_CLK_CTRL reigster.

Note:

CAM_CLK_OUT must not be confused with the CAM.EXCLK function: CAM.EXCLK is the output pixel clock of the camera module; CAM_CLK_OUT is merely the system clock divided. These two clocks have different sources, and they are enabled differently

5.14.4 SYS_CLK_OUT

Ball B15 can be configured to function as SYS_CLK_OUT (in multiplexing mode 1). The system clock can be enabled on this ball either by a hardware or a software request.

SYS_CLK_OUT uses the same hardware request as MCLK (MCLKREQ). The pin multiplexing should be setup so that ball R10 acts as MCLKREQ and ball

B15 acts as SYS_CLK_OUT. The procedure to enable SYS_CLK_OUT using a hardware request is:

- Set the TI_RESERVED_EN bit to 1 in the ULPD CLOCK_CTRL_REG register.
- Set the DIS_COM_MCLK_REQ bit to 0 in the ULPD SOFT_DISABLE_REQ_REG register.

Consequently, the system clock is available on ball B15. Modifying the SOFT_DIS_COM_MCLK_REQ or TI_RESERVED_EN bits masks out MCLKREQ and deactivates MCLK clock.

SYS_CLK_OUT uses the same software request as the 48-MHz MCLK. The procedure to enable SYS_CLK_OUT using a software request is:

- Set ULPD CLOCK_CTRL_REG.TI_RESERVED_EN bit to 1
- Set ULPD SOFT_DISABLE_REQ_REG.DIS_COM_MCLK_REQ to 1
- Set ULPD SOFT_REQ_REG.SOFT_COM_REQ to 1

Consequently, the system clock appears on ball B15. By modifying SOFT_COM_REQ or TI_RESERVED_EN bits, the software can deactivate MCLK clock.

5.15 Low-Power Modes

The ULPD module controls transitions to the two low-power modes referred to as big sleep and deep sleep modes. Awake is the only active mode for which OMAP3.2 can fetch instructions.

In addition, if some components are not used, they can be put in low-power mode to reduce power dissipation.

Table 16. Control for Special Components

Components	Software Control	Notes
32-kHz oscillator	OSC32K_PWRDN_R	Control in RTC register file
12-MHz oscillator	CONF_OSC1_PWRDN_R	Control in OMAP5912 configuration register file
LDO	CONF_LDO_PWRDN_CNTRL_R SOFT_LDO_SLEEP	Dedicated embedded LDO for DPLL control in OMAP5912 configuration register file Control in ULPD register file
DPLL	PLL_ENABLE	Converts 12-MHz to19.2-MHz clock input to high-frequency clock used in OMAP3.2 clock tree Control in DPLL register file
APLL	None	Converts 12-MHz to 19.2-MHz clock to 48-MHz clock APLL active whenever a request for 48-MHz clock is set to 1.

Table 17 details the OMAP5912 sleep modes as they relate to power dissipation and clocks.

Table 17. Mode Description

Modes	Power Dissipation	Clocks
Deep sleep	Lowest	32 kHz for wake-up detection
Big sleep	Caused by gates clocked by 32-kHz clocks and clocks derived from 96-MHz PLL	Each active clock (32-kHz clocks and clocks from 96-MHz PLL) can be gated.
Awake	Nominal	32-kHz clocks and system clocks

Index

A

C

D

L

O

OMAP5912 Multimedia Processor Initialization Reference Guide

Literature Number: SPRU752B
October 2004

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.

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
		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 document describes the reset architecture, the configuration, and the initialization of the OMAP5912 multimedia processor.

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.

Related Documentation From Texas Instruments

Documentation that describes the OMAP5912 device, related peripherals, and other technical collateral, is available in the OMAP5912 Product Folder on TI's website: www.ti.com/omap5912.

Trademarks

OMAP and the OMAP symbol are trademarks of Texas Instruments.

Contents

Tables

Initialization

This document describes the reset architecture, the configuration, and the initialization of the OMAP5912 multimedia processor. All references to device package ball number in this document refer to the ZZG package. Please see the data manual (SPRS231) for complete pinout information for both the ZZG and ZDY packages.

1 Reset Architecture

The reset architecture describes the reset signal distribution to the peripherals.

1.1 Reset Modes and Clocking Options

Reset mode 0 has several clocking options while reset mode 1 is much more restrictive. It is important to examine Table 1 when determining which reset mode to use.

Table 1. Clocking Options with respect to Reset Modes

Reset Mode = 0	12 MHz	13 MHz	19.2 MHz
OSC1_IN support via crystal?	Yes	Yes	Yes
OSC1_IN support w/external clock source?	Yes	Yes	Yes
SYS_CLK_IN support w/external clock source?	No	No	No
Reset Mode = 1	**12 MHz**	**13 MHz**	**19.2 MHz**
OSC1_IN support via crystal?	No	No	No
OSC1_IN support w/external clock source?	No	No	No
SYS_CLK_IN support w/external clock source?	No	No	Yes

It is critical to understand that certain systems and functionalities may not be available in both reset modes. The selection between reset modes should be made at an early stage in the development cycle. Generally, reset mode 0 is flexible while reset mode 1 is more restrictive. Reset mode is based on the value of the RESET_MODE pin (sampled on the rising edge of $\overline{\text{PWRON\_RESET}}$).

In addition to affecting the clocking options, reset mode affects many other features such as pin multiplexing at reset time, I/O direction and impedance at reset time, and more. Throughout this document, the reset mode will be specified when it is relevant to the discussion.

1.2 Resets

This processor has up to three external reset pins depending on the reset mode. PWRON_RESET is the cold reset for the entire chip. MPU_RST is the MPU subsystem reset (this pin has special setup requirements in reset mode 1). RTC_ON_NOFF is a software controlled, power-on reset (unavailable in reset mode 1).

Table 2. External Reset

External Reset	Description	Notes
	Reset Mode = 0	
PWRON_RESET	Power-up reset	If RTC is used, must be considered as battery power-up reset. Pulse duration must be two periods of 32 kHz.
	Reset Mode = 0 (Continued)	
RTC_ON_NOFF	Power-up reset	Gated by bit 7 of RTC_CTRL_REG. RTC_ON_NOFF is gated (i.e. disabled) at PWRON_RESET.
MPU_RST	MPU reset	Cold reset of MPU subsystem. Does not reset on chip DPLL.
	Reset Mode = 1	
PWRON_RESET	Power-up reset	Cold reset. Pulse duration must be 2 periods of 32 kHz.
RTC_ON_NOFF	Not used	Gated by reset mode 1.
MPU_RST	MPU reset	Proper pin functionality must be configured (this pin defaults to MPUIO4).

In reset mode 1, the RTC split-power functionality cannot be used, since the RTC_ON_NOFF pin is inactive. The real-time clock functionality of the RTC works correctly in both modes.

In the following sections:

- Table 3 summarizes sources and status bits for global resets.
- Table 4 covers the resets specific to OMAP3.2 (MPU subsystem).
- Table 5 summarizes the peripheral resets for both the MPU and the DSP.

1.2.1 RTC Split Power

RTC split power is available with reset mode 0 only. The RTC power domain is split from the core power domain so that the real-time clock and its associated oscillator can continue to run off of the battery while the MPU subsystem is completely powered off. With RTC split power, $\overline{\text{PWRON\_RESET}}$ acts as a battery power-on reset while RTC_ON_OFF is used to subsequently reset the MPU subsystem without disrupting the time stored in the RTC.

1.2.2 Global Reset

Table 3 summarizes sources and status bits for global resets. The table groups some peripherals into general classes. Peripheral classes group peripherals based on the source of the reset input. For example, a Class-1 module resets directly from the cold, power-on reset while a Class-2 peripheral (module) resets from the MPU subsystem. See Figure 1 and Table 5 for more information on peripheral classes.

Table 3. Global Resets

Reset	Source	Event	Reset Description	Status Bit
Cold reset	External PWRON_RESET	Low on PWRON_RESET	Reset LCD controller System DMA MPU port interface L3 OCP initiator L4 controller Traffic controller DSP MMU MPU TIPB bridge and peripherals Shared peripherals	POR (bit 5) MPU clock reset status register (ARM_SYSST)
				EXT_RST (bit 4) MPU clock reset status register (ARM_SYSST)
			In RESET_MODE 1 sole reset source for Class1 modules: boot ROM, ULPD, OMAP5912 conf, sync counter. Sole reset source for RTC.	GLOB_SWRST (bit1) MPU clock reset status register (ARM_SYSST)
	External RTC_ON_NOFF	Low on RTC_ON_NOFF if RTC split power set to 1 (bit 6 of RTC_CTRL_REG)	Reset LCD controller System DMA MPU port interface L3 OCP initiator L4 controller Traffic controller DSP MMU MPU TIPB bridge and peripherals Shared peripherals	POR (bit5 5) MPU clock reset status register (ARM_SYSST)
				EXT_RST (bit 4) MPU clock reset status register (ARM_SYSST)
			Reset source for Class 1 modules boot ROM,ULPD, OMAP5912 conf, sync counter in RESET_MODE 1. In RESET_MODE 0, does not reset RTC.	GLOB_SWRST (bit 1) MPU clock reset status register (ARM_SYSST)
Warm reset	External MPU_RST	Low on MPU_RST	Reset LCD controller System DMA MPU port interface L3 OCP initiator L4 controller Traffic controller DSP MMU MPU TIPB bridge and peripherals Shared peripherals	EXT_RST (bit 4) MPU clock reset status register (ARM_SYSST)
			Class 2 and class 3 modules. SDRAM refresh mode switched to self-refresh if previously set to autorefresh state. Not applicable in RESET_MODE 1.	GLOB_SWRST (bit 1) MPU clock reset status register (ARM_SYSST)

Table 3. Global Resets (Continued)

Reset	Source	Event	Reset Description	Status Bit
	Production eFuse	eFuse programmed value to BAD	Warm reset permanently asserted by ULPD	EXT_RST (bit 4) MPU clock reset status register (ARM_SYSST)
				GLOB_SWRST (bit 1) MPU clock reset status register (ARM_SYSST)
Warm reset (continued)	32-kHz WD reset	32-kHz WD time-out default configuration leads to 32s time-out with 32-kHz input frequency.	Reset LCD controller System DMA MPU port interface L3 OCP initiator L4 controller Traffic controller SDRAM refresh mode switched to self-refresh if previously set to autorefresh state, DSP MMU, MPU TIPB bridge and peripherals, shared peripherals Class 2 and class 3 modules.	EXT_RST (bit 4) MPU clock reset status register (ARM_SYSST)
				Reset Done (bit 0) in 32K watchdog system status register (WD_SYSSTATUS)
				GOB_SWRST (bit 1) MPU clock reset status register (ARM_SYSST)
				EXTERNAL_RESET_ SOURCE_3 (bit 3) of ULPD status register

Table 3. Global Resets (Continued)

Reset	Source	Event	Reset Description	Status Bit
Warm reset (continued)	Global system reset (software)	SW_RST (bit 3) in ARM_ RSTCT1 is set to 1 or set ARM_RST (bit 0) in ARM_RSTC1 and clear DSP_EN (bit 1) in ARM_RSTC1	Reset LCD controller System DMA MPU port interface L3 OCP initiator L4 controller Traffic controller DSP MMU MPU TIPB bridge and peripherals Shared peripherals SDRAM refresh mode switched to self-refresh if previously set to autorefresh state. Class 2 and class 3 modules.	GLOB_SWRST (bit 1) MPU clock reset status register (ARM_SYSST)
	MPU WD reset	MPU WD underflow	Reset LCD controller System DMA MPU port interface L3 OCP initiator L4 controller Traffic controller DSP MMU MPU TIPB bridge and peripherals Shared peripherals SDRAM refresh mode switched to self-refresh if previously set to autorefresh state. Class 2 and class 3 modules.	ARM_WDRST (bit 2) in MPU clock reset status register (ARM_SYSST)
				GLOB_SWRST (bit 1) MPU clock reset status register (ARM_SYSST)
DSP WD reset	DSP WD timeout	DSP WD Underflow	Reset DSP system and peripheral modules wrapper switches. Reset WD_PER_EN (bit 1) in DSP_RSTCT2.	DSP_WDRST (bit 0) in MPU clock reset status register (ARM_SYSST)
MPU software reset	Software	ARM_RST (bit 0) in ARM_RSTC1 is set to 1	Reset the MPU.	ARM_MCRST (bit 3) in MPU clock reset status register (ARM_SYSST)
MPU peripheral reset	Software	PER_EN (bit 0) in ARM_RSTC2 is cleared to 0.	Reset peripheral class 2 modules and peripheral modules wrapper switches.	

Table 3. Global Resets (Continued)

Reset	Source	Event	Reset Description	Status Bit
DSP core software reset	Software	DSP_EN (bit 1) in ARM_RSTC1 is cleared to 0.	Reset the DSP, excluding the configuration settings (config registers of EMIF internal to DSP and the MPUI control logic internal to DSP).	
DSP peripheral reset	Software	DSP_PEREN (bit 0) in DSP_RSTC2 is cleared to 0.	Reset peripheral Class 3 modules.	
DSP Software Reset	Software	DSP_RST (bit 2) in ARM_RSTC1 is set to 1.	Reset the priority registers (TIPB) module, EMIF configuration register, and the MPUI control logic in the DSP.	

In conjunction with the reset status register in OMAP3.2, the user can poll the ULPD reset status register to determine whether the reset was caused by a 32K watchdog time-out. See Table 3.

1.2.3 OMAP 3.2 Resets

Table 4 shows how OMAP3.2 components are affected by various reset sources. The MPU and DSP peripheral resets are not included in this table as they are not part of OMAP3.2 (see Table 5 for peripheral resets).

Table 4. OMAP 3.2 Resets

Components	Type	Cold Resets	Warm Resets	DSP WD Reset	MPU Software Reset	DSP Core Software Reset	DSP Software Reset
CLKM_1	MPU	Yes	Yes	No	No	No	No
CLKM_2	DSP	Yes	Yes	No	No	No	No
CLKM_3	MPU	Yes	Yes	No	No	No	No
DPLL_1		Yes	See Note	No	No	No	No
ARM926EJS	MPU	Yes	Yes	No	Yes	No	No

† DSP-MMU SW reset bit in CNTL_REG of DSP-MMU module must also be set correctly for DSP-MMU to be reset.
‡ SDRAM is in self-refresh; some of the registers controlled by SDRAM FSM do not reset.
§ See Table 3 for a listing of all cold/warm resets.

Note: DPLL is reset by $\overline{\text{MPU\_RST}}$ or 32-kHz WD. Other warm resets do not reset the DPLL.

Table 4. OMAP 3.2 Resets (Continued)

Components	Type	Cold Resets	Warm Resets	DSP WD Reset	MPU Software Reset	DSP Core Software Reset	DSP Software Reset
ETM	MPU	Yes	Yes	No	No	No	No
MPU timer 1	MPU	Yes	Yes	No	No	No	No
MPU timer 2	MPU	Yes	Yes	No	No	No	No
MPU timer 3	MPU	Yes	Yes	No	No	No	No
MPU WD timer	MPU	Yes	Yes	No	No	No	No
MPU INTH	MPU	Yes	Yes	No	No	No	No
LCD controller	MPU	Yes	Yes	No	No	No	No
DSP	DSP	Yes	Yes	Yes	No	Yes DSP core only, not EMIF config regs and MPUI control logic	Yes Only EMIF config regs and MPUI control logic in DSP, not DSP core
DSP MMU	DSP	Yes †	Yes †	No	No	No	No
DSP timer 1	DSP	Yes	Yes	Yes	No	No	No
DSP timer 2	DSP	Yes	Yes	Yes	No	No	No
DSP timer 3	DSP	Yes	Yes	Yes	No	No	No
DSP WD timer	DSP	Yes	Yes	Yes	No	No	No
DSP INTIF	DSP	Yes	Yes	Yes	No	No	No
DSP INTH	DSP	Yes	Yes	Yes	No	No	No
Mailbox		Yes	Yes	No	No	No	No
MPUI		Yes	Yes	No	No	No	No
System DMA controller	MPU	Yes	Yes	No	No	No	No

† DSP-MMU SW reset bit in CNTL_REG of DSP-MMU module must also be set correctly for DSP-MMU to be reset.
‡ SDRAM is in self-refresh; some of the registers controlled by SDRAM FSM do not reset.
§ See Table 3 for a listing of all cold/warm resets.

Note: DPLL is reset by $\overline{\text{MPU\_RST}}$ or 32-kHz WD. Other warm resets do not reset the DPLL.

Table 4. OMAP 3.2 Resets (Continued)

Components	Type	Cold Resets	Warm Resets	DSP WD Reset	MPU Software Reset	DSP Core Software Reset	DSP Software Reset
TIPB bridge	MPU	Yes	Yes	No	No	No	No
EMIFF	MPU	Yes	Self-Ref ‡	No	No	No	No
EMIFS	MPU	Yes	Yes	No	No	No	No
L3/OCP/T1 (target)	MPU	Yes	Yes	No	No	No	No
L3/OCP/T2 (target)	MPU	Yes	Yes	No	No	No	No
L3/OCP(initiator)	MPU	Yes	Yes	No	No	No	No
WT	MPU	Yes	No	No	No	No	No

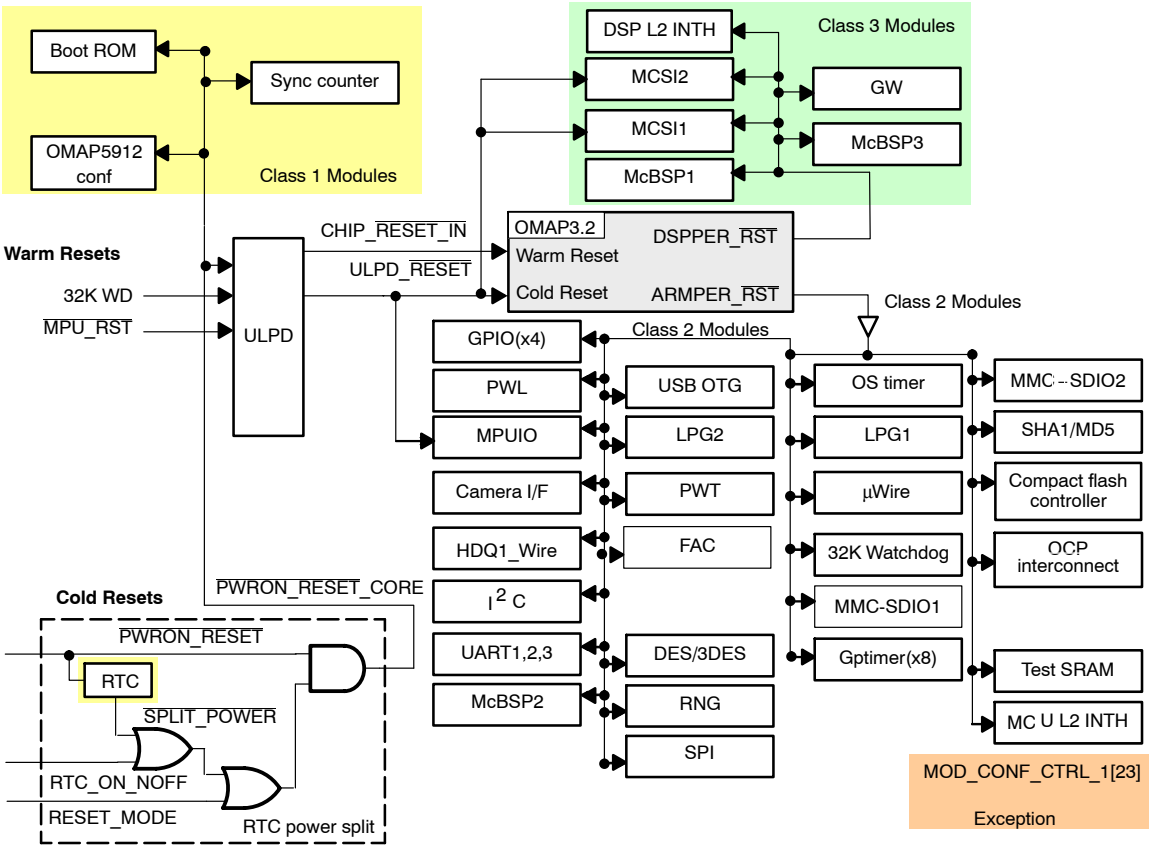
† DSP-MMU SW reset bit in CNTL_REG of DSP-MMU module must also be set correctly for DSP-MMU to be reset.
‡ SDRAM is in self-refresh; some of the registers controlled by SDRAM FSM do not reset.
§ See Table 3 for a listing of all cold/warm resets.

Note: DPLL is reset by $\overline{\text{MPU\_RST}}$ or 32-kHz WD. Other warm resets do not reset the DPLL.

1.2.4 Peripheral Resets

Figure 1 shows how the resets are distributed to the peripherals. See Table 5 for a description of the different peripheral classes.

Figure 1. Reset Distribution to Peripherals



Note that in Figure 1, peripheral classes group peripherals based on the source of the reset input.

The ULPD is in charge of generating two functional reset signals to the MPU subsystem (OMAP3.2 core): cold reset (PIPORN) and warm reset (PICHIPNRST). It is then the task of the MPU subsystem to reset processors and peripherals. In addition, there are three input resets to the ULPD. Two of these resets are external pins: PWRON_RESET (cold reset on ball R12) and MPU_RST (warm reset on ball U20). The 32-kHz watchdog timer reset is a warm reset.

1.2.5 Peripheral Reset Table

Table 5 shows the various reset sources for each peripheral. Many of the OMAP peripherals (external to the MPU subsystem) have module wrappers or switches that facilitate conversion between different bus protocols (wrapper) or control DSP and MPU access to that peripheral (switch).

For example, the UART1 peripheral is on the TIPB, but the internal bus protocol for UART1 is OCP. UART1 has a wrapper that converts between the TIBP and OCP bus protocols. The *wrapper* must be released from reset before access to the UART1 peripheral is possible. This wrapper also acts as a *switch* that determines whether the DSP or MPU has control over the UART1.

For peripherals with wrappers/switches, functionality is gated by the reset for the wrapper/switch. Consequently, the wrapper/switch must be released from reset before the peripheral becomes available. Table 5 details which peripherals have wrappers or switches. In addition, the Multimedia Processor Peripheral Interconnects Reference Guide (SPRU758) discusses the specific types of wrappers and switches for those peripherals. It is also important to understand that there are two types of switches, each with distinct default behaviors when released from reset. SPRU758 details such behaviors.

Table 5. Reset Sources for Peripherals

Peripheral Name	HW Reset	SW Reset	Wrapper/ Switch	Wrapper/Switch Reset
Class 1 Modules				
Real-time clock (RTC)	Cold reset	No	No	No
OMAP5912configuration	Cold reset	No	No	No
Boot ROM	Cold reset	No	No	No
32-kHz synchro counter	Cold reset	No	Dynamic switch	Cold reset/Warm reset/ARM_WD/DSP_ WD/SWRST3#/ SWRST2/SWRST1

† OCP SWRST: Reset software done in corresponding module.
‡ SWRST1: PER_EN (bit 0) in ARM_RSTC2 is cleared to 0.
§ SWRST2: Set ARM_RST (bit 0) in ARM_RSTCT1 and clear DSP_EN (bit 1) in ARM_RSTCT1.
¶ SWRST3: DSP_PEREN (bit 0) in DSP_RSTCT2 is cleared to 0.
# Warm reset: Source can be $\overline{\text{MPU\_RST}}$, global software reset, or 32-kHz watchdog time-out.
|| SW control via RESET_CONTROL register (see Table 51).
☆ SW control via MOD_CONF_CTRL_1[23].

Table 5. Reset Sources for Peripherals (Continued)

Peripheral Name	HW Reset	SW Reset	Wrapper/ Switch	Wrapper/Switch Reset
	Class 2 Modules			
µWire ‖	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	No	No	
HDQ/1-Wire	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	No	No	
Camera I/F ‖	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	No	No	
PWT	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	No	No	
PWL	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	No	No	
LPG1	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	No	No	
LPG2	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	No	No	
OCP interconnect ‖	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	No	No	
Frame Buffer	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	No	No	

† OCP SWRST: Reset software done in corresponding module.
‡ SWRST1: PER_EN (bit 0) in ARM_RSTC2 is cleared to 0.
§ SWRST2: Set ARM_RST (bit 0) in ARM_RSTCT1 and clear DSP_EN (bit 1) in ARM_RSTCT1.
¶ SWRST3: DSP_PEREN (bit 0) in DSP_RSTCT2 is cleared to 0.
# Warm reset: Source can be $\overline{\text{MPU\_RST}}$, global software reset, or 32-kHz watchdog time-out.
‖ SW control via RESET_CONTROL register (see Table 51).
☆ SW control via MOD_CONF_CTRL_1[23].

Table 5. Reset Sources for Peripherals (Continued)

Peripheral Name	HW Reset	SW Reset	Wrapper/ Switch	Wrapper/Switch Reset
Class 2 Modules (Continued)				
CompactFlash controller	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	No	No	
RNG	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	OCP SWRST	OCP wrapper	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1
GPIO1	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	OCP SWRST	Dynamic switch	Cold reset/Warm reset/ARM_WD/DSP_WD/SWRST3/SWRST2/SWRST1
GPIO2	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	OCP SWRST	Dynamic switch	Cold reset/Warm reset/ARM_WD/DSP_WD/SWRST3/SWRST2/SWRST1
GPIO3	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	OCP SWRST	Dynamic switch	Cold reset/Warm reset/ARM_WD/DSP_WD/SWRST3/SWRST2/SWRST1
GPIO4	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	OCP SWRST	Dynamic switch	Cold reset/Warm reset/ARM_WD/DSP_WD/SWRST3/SWRST2/SWRST1
USB On-the-Go	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	OCP SWRST	OCP MPU	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1
UART1	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	OCP SWRST	Static switch	Cold reset/Warm reset/ARM_WD/DSP_WD/SWRST3/SWRST2/SWRST1

† OCP SWRST: Reset software done in corresponding module.
‡ SWRST1: PER_EN (bit 0) in ARM_RSTC2 is cleared to 0.
§ SWRST2: Set ARM_RST (bit 0) in ARM_RSTCT1 and clear DSP_EN (bit 1) in ARM_RSTCT1.
¶ SWRST3: DSP_PEREN (bit 0) in DSP_RSTCT2 is cleared to 0.
# Warm reset: Source can be $\overline{\text{MPU\_RST}}$, global software reset, or 32-kHz watchdog time-out.
|| SW control via RESET_CONTROL register (see Table 51).
☆ SW control via MOD_CONF_CTRL_1[23].

Table 5. Reset Sources for Peripherals (Continued)

Peripheral Name	HW Reset	SW Reset	Wrapper/ Switch	Wrapper/Switch Reset
Class 2 Modules (Continued)				
UART2	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	OCP SWRST	Static switch	Cold reset/Warm reset/ARM_WD/DSP_WD/SWRST3/SWRST2/SWRST1
UART3	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	OCP SWRST	Static switch	Cold reset/Warm reset/ARM_WD/DSP_WD/SWRST3/SWRST2/SWRST1
I^2C multi-master/slave	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	OCP SWRST	Static switch	Cold reset/Warm reset/ARM_WD/DSP_WD/SWRST3/SWRST2/SWRST1
SPI master/ slave	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	OCP SWRST	Static switch	Cold reset/Warm reset/ARM_WD/DSP_WD/SWRST3/SWRST2/SWRST1
MMC/SDIO2	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	OCP SWRST	Static switch	Cold reset/Warm reset/ARM_WD/DSP_WD/SWRST3/SWRST2/SWRST1
General-purpose timer 1	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	OCP SWRST	Static switch	Cold reset/Warm reset/ARM_WD/DSP_WD/SWRST3/SWRST2/SWRST1
General-purpose timer 2	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	OCP SWRST	Static switch	Cold reset/Warm reset/ARM_WD/DSP_WD/SWRST3/SWRST2/SWRST1

† OCP SWRST: Reset software done in corresponding module.
‡ SWRST1: PER_EN (bit 0) in ARM_RSTC2 is cleared to 0.
§ SWRST2: Set ARM_RST (bit 0) in ARM_RSTCT1 and clear DSP_EN (bit 1) in ARM_RSTCT1.
¶ SWRST3: DSP_PEREN (bit 0) in DSP_RSTCT2 is cleared to 0.
# Warm reset: Source can be $\overline{\text{MPU\_RST}}$, global software reset, or 32-kHz watchdog time-out.
‖ SW control via RESET_CONTROL register (see Table 51).
☆ SW control via MOD_CONF_CTRL_1[23].

Table 5. Reset Sources for Peripherals (Continued)

Peripheral Name	HW Reset	SW Reset	Wrapper/ Switch	Wrapper/Switch Reset
Class 2 Modules (Continued)				
General-purpose timer 3	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	OCP SWRST	Static switch	Cold reset/Warm reset/ARM_WD/DSP_WD/SWRST3/SWRST2/SWRST1
General-purpose timer 4	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	OCP SWRST	Static switch	Cold reset/Warm reset/ARM_WD/DSP_WD/SWRST3/SWRST2/SWRST1
General-purpose timer 5	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	OCP SWRST	Static switch	Cold reset/Warm reset/ARM_WD/DSP_WD/SWRST3/SWRST2/SWRST1
General-purpose timer 6	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	OCP SWRST	Static switch	Cold reset/Warm reset/ARM_WD/DSP_WD/SWRST3/SWRST2/SWRST1
General-purpose timer 7 (sleep timer)	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	OCP SWRST	Static switch	Cold reset/Warm reset/ARM_WD/DSP_WD/SWRST3/SWRST2/SWRST1
General-purpose timer 8 (32K DSP)	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	OCP SWRST	Static switch	Cold reset/Warm reset/ARM_WD/DSP_WD/SWRST3/SWRST2/SWRST1
McBSP2	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	Partial	Static switch	Cold reset/Warm reset/ARM_WD/DSP_WD/SWRST3/SWRST2/SWRST1

† OCP SWRST: Reset software done in corresponding module.
‡ SWRST1: PER_EN (bit 0) in ARM_RSTC2 is cleared to 0.
§ SWRST2: Set ARM_RST (bit 0) in ARM_RSTCT1 and clear DSP_EN (bit 1) in ARM_RSTCT1.
¶ SWRST3: DSP_PEREN (bit 0) in DSP_RSTCT2 is cleared to 0.
# Warm reset: Source can be $\overline{\text{MPU\_RST}}$, global software reset, or 32-kHz watchdog time-out.
|| SW control via RESET_CONTROL register (see Table 51).
☆ SW control via MOD_CONF_CTRL_1[23].

Table 5. Reset Sources for Peripherals (Continued)

Peripheral Name	HW Reset	SW Reset	Wrapper/ Switch	Wrapper/Switch Reset
Class 2 Modules (Continued)				
MPU level 2 interrupt handler	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	OCP SWRST	OCP wrapper	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1
MMC/SDIO1	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	OCP SWRST	OCP wrapper	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1
32-kHz watchdog	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	OCP SWRST	OCP wrapper	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1
SHA-1/MD5	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	OCP SWRST	OCP wrapper	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1
DES/3DES	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	OCP SWRST	OCP wrapper	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1
FAC	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	No	pvci2rhea	
OS Timer ‖	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1	No	TIPB	
MPUIO‖	Cold reset/Warm reset/ARM_WD/SWRST2/SWRST1/ 32-kHz WD	No	No	

† OCP SWRST: Reset software done in corresponding module.
‡ SWRST1: PER_EN (bit 0) in ARM_RSTC2 is cleared to 0.
§ SWRST2: Set ARM_RST (bit 0) in ARM_RSTCT1 and clear DSP_EN (bit 1) in ARM_RSTCT1.
¶ SWRST3: DSP_PEREN (bit 0) in DSP_RSTCT2 is cleared to 0.
# Warm reset: Source can be $\overline{\text{MPU\_RST}}$, global software reset, or 32-kHz watchdog time-out.
‖ SW control via RESET_CONTROL register (see Table 51).
☆ SW control via MOD_CONF_CTRL_1[23].

Table 5. Reset Sources for Peripherals (Continued)

Peripheral Name	HW Reset	SW Reset	Wrapper/ Switch	Wrapper/Switch Reset
Class 3 Modules				
DSP Interrupt handler 2.1	Cold reset/Warm reset/ARM_WD/SWRST3/SWRST2	OCP SWRST	OCP wrapper	Cold reset/Warm reset/ARM_WD/ DSP_WD/SWRST3/SWRST2
McBSP1	Cold reset/Warm reset/ARM_WD/SWRST3/SWRST2	Partial	OCP wrapper	Cold reset/Warm reset/ARM_WD/DSP_WD/SWRST3/SWRST2
McBSP3	Cold reset/Warm reset/ARM_WD/SWRST3/SWRST2	Partial	OCP wrapper	Cold reset/Warm reset/ARM_WD/DSP_WD/SWRST3/SWRST2
MCSI1	Cold reset/Warm reset/ARM_WD/SWRST3/ SWRST2/ 32-kHz WD/ SEC WD /SEC FSM	Partial	TIPB	
MCSI2	Cold reset/Warm reset/ARM_WD/SWRST3/SWRST2/ 32-kHz WD/SEC WD/ SEC FSM	Partial	TIPB	

† OCP SWRST: Reset software done in corresponding module.
‡ SWRST1: PER_EN (bit 0) in ARM_RSTC2 is cleared to 0.
§ SWRST2: Set ARM_RST (bit 0) in ARM_RSTCT1 and clear DSP_EN (bit 1) in ARM_RSTCT1.
¶ SWRST3: DSP_PEREN (bit 0) in DSP_RSTCT2 is cleared to 0.
# Warm reset: Source can be $\overline{\text{MPU\_RST}}$, global software reset, or 32-kHz watchdog time-out.
|| SW control via RESET_CONTROL register (see Table 51).
☆ SW control via MOD_CONF_CTRL_1[23].

Note: DSP_WD is controlled by the DSP_RSTCT2.WD_PER_EN.

1.3 Input/Output

Table 6 lists all I/Os that are related to the clock and reset module.

Table 6. Input/Output for Clock and Reset

Pin Name	Dir.	Ball	Description	Default Mode in RESET_MODE 0	Default Mode in RESET_MODE 1	Notes
RESET_MODE	In	P12	Reset_mode	Yes	Yes	Sampled at power-up reset.
$\overline{\text{PWRON\_RESET}}$	IN	R12	Power-up reset	Yes	Yes	Is battery asserted only once when RTC split power backup is used. See the Multimedia Processor Power Management Reference Guide (SPRU753).
RTC_ON_NOFF	IN	Y12	Power-up reset	Yes	No	Bit 7 of RTC_CTRL_REG must be set to 1.
$\overline{\text{MPU\_RST}}$	IN	U20	Warm reset	Yes	No	
$\overline{\text{RST\_OUT}}$	OUT	AA20		Yes		
OSC32K_IN	IN	V13	32-kHz oscillator	Yes	Yes	If external 32-kHz clock is used, must be tied high.
OSC32K_OUT	OUT	AA13	32-kHz oscillator	Yes	Yes	If external 32-kHz clock is used, must be tied low.
CLK32K_IN	IN	P13	32-kHz clock In	Yes	Yes	Must be tied low if on-chip 32-kHz oscillator is used.
OSC1_IN	IN	Y2	12-MHz oscillator	Yes	Yes	If external SYS_CLK_IN is used, must be tied low.
OSC1_OUT	OUT	W3	12-MHz oscillator	Yes	Yes	

Table 6. Input/Output for Clock and Reset (Continued)

Pin Name	Dir.	Ball	Descrip-tion	Default Mode in RESET_MODE 0	Default Mode in RESET_MODE 1	Notes
SYS_CLK_IN	IN	Y4	12-MHz to 19.2-MHz clock in	No	Yes	Must be tied low if on-chip oscillator is used. Must select mode 110 for Ball Y4 with software.
EXT_CLK	IN	N18	External clock for GP timers	No	No	
EXT_48M	IN	N21	Backup 48-MHz clock in	No	No	Through GPIO14 input pin. Used if on-chip APLL is disabled.
BCLKREQ	IN	W15	Request for BCLK	Yes	Yes	
MCLKREQ	IN	R10	Request for MCLK	Yes	Yes	
CLK32K_OUT	OUT	R13	32-kHz clock out	Yes	No	
EXT_MASTER_REQ	OUT	R10	Request for external clock	No	No	
RST_HOST_OUT	OUT	W13	Modem shut down if battery fails	No	No	Can be controlled by software (POWER_CTRL_REG[3] if POWER_CTRL_REG[2]=0)
BCLK	OUT	Y15	System clock or derived from APLL clock	Yes	Yes	Can be divided further by setting SDW_CLK_DIV_CTRL_SEL[7:2]
MCLK	OUT	V5	System clock or derived from APLL clock	Yes		When 48MHz, can be divided further by setting COM_RATIO_SEL[7:2]

Table 6. Input/Output for Clock and Reset (Continued)

Pin Name	Dir.	Ball	Description	Default Mode in RESET_MODE 0	Default Mode in RESET_MODE 1	Notes
USB.CLK0	OUT	W4	Derived from APLL clock	No	No	96-MHz APLL clock divided by 16
SYS_CLK_OUT	OUT	B15	System clock output	No		Same frequency as DPLL1 input (CK_REF)
CAM.EXCLK	OUT	H19	Derived from CK_REF	Yes		Camera clock output
ULPD_DPLL48M	OUT	J18	Output from APLL	No	No	In test/ observability mode only
LOW_PWR	OUT	T20	Low-voltage mode	No	No	High in deep sleep or can be set high by software (POWER_CTRL_REG[1] if POWER_CTRL_REG[0]=1)
$\overline{\text{LOW\_PWR}}$	OUT	W4	Low-voltage mode	No	Yes	Low in deep sleep mode

2 Configuration

The configuration module allows software to control the various static modes supported by the device. This module is the primary point of control for the following areas of the device:

- ❏ Functional I/O multiplexing
- ❏ Debug and observation I/O multiplexing
- ❏ I/O gating and inhibiting for power-down modes
- ❏ Pullup and pulldown enable and selection
- ❏ Interface voltage selection
- ❏ Static module configuration
- ❏ Control of clock multiplexing
- ❏ Multiplexing mode status
- ❏ Control of USB integrated transceivers
- ❏ LDO bypassing control
- ❏ Reset control

2.1 Configuration Register Capabilities

The configuration module is a bank of 32-bit registers that can be read and written by software. The module is reset only with a cold reset. This bank of registers can be broken down into the following sections:

Table 7. Configuration Registers

Legacy configuration registers	FUNC_MUX_CTRL(0-2)
I/O multiplex enable register	COMP_MODE_CTRL_0
Generic multiplexing registers	FUNC_MUX_CTRL(3-12)
Pullup/pulldown enable registers	PULL_DWN_CTRL(0-4)
Pullup/pulldown select registers	PU_PD_SEL(0-4)
Gating and inhibiting registers	GATE_INH_CTRL_0
Voltage control registers	VOLTAGE_CTRL_0
Test and debug registers	TEST_DBG_CTRL_0
Configuration module version number	CONF_REV
Module configuration registers	MOD_CONF_CTRL(0-1)
DSP DMA request multiplex registers	FUNC_MUX_DSP_DMA(A-D)

Table 7. Configuration Registers (Continued)

MPU DMA request multiplex registers	FUNC_MUX_ARM_DMA(A-G)
Status register	CONF_STATUS
LDO bypassing control register	LDO_PWRDN_CTRL
USB Transceivers control register	USB_TRANSCEIVER_CTRL
Reset control register	RESET_CONTROL
OMAP5912 control register	CONF_5912_CTRL

2.2 Pin Multiplexing and Pullups/Pulldowns

Each pin that has a multiplexing function is assigned a 3-bit field in the register set FUNC_MUX_CTRL(3-12), thus creating up to eight possible multiplexing options per pin. At reset ($\overline{\text{PWRON\_RESET}}$ is low), the multiplexing of each pin is asynchronously forced by the RESET_MODE pin, regardless of the value written to the FUNC_MUX_CTRL(3-12) registers.

Most of the pins can be configured either as a pullup or a pulldown. The PULL_DWN_CTRL (0–4) registers control whether the pullup or the pulldown is enabled. The pulls (up and down) are disabled by setting a 1 in the corresponding field.

The PU_PD_SEL(0-4) registers configure pin-by-pin whether a pullup or a pulldown is selected. At reset, the pin-for-pin, pullup/pulldown state is forced to be consistent with reset state. The pullups and pulldowns remain in this forced state until 0x0000EAEF is written to the COMP_MODE_CTRL_0 register.

2.2.1 Pin Multiplexing Considerations with Respect to RESET_MODE and GPIO1

The RESET_MODE pin is sampled on the rising edge of $\overline{\text{PWRON\_RESET}}$. When $\overline{\text{PWRON\_RESET}}$ is low, RESET_MODE affects the impedance state or direction of some of the pins because the intended use of these pins differs between reset modes. The RESET_MODE affects the initial mulitplexing state of some pins irrespective of the FUNC_MUX_CTRL registers and it affects the initial state of the EMIFS interface.

Note: Choosing between reset modes

It is critical to understand that by changing the RESET_MODE, certain systems and functionalities may not be available in both modes. The decision to choose between reset mode 0 or 1 should be made at an early stage of the development cycle. Generally, reset mode 0 is flexible while reset mode 1 is more restrictive.

It is important to distinguish between pin functionalities before and after writing 0x0000EAEF to COMP_MODE_CTRL_0:

- Prior to writing 0x0000EAEF, pin multiplexing for all pins depends solely on:
  - RESET_MODE latched by the rising edge of $\overline{\text{PWRON\_RESET}}$.
  - GPIO1 latched by the rising edge of $\overline{\text{PWRON\_RESET}}$ if RESET_MODE = 0.
- After writing 0x0000EAEF, pin multiplexing depends on the FUNC_MUX_CTRL registers for all pins.

> **CAUTION**
>
> Even though all FUNC_MUX_CTRL registers reset to 0x0000, reset mode 1 causes some pins to default to a pin mux mode other than 000. To maintain the same mux mode on such pins, the appropriate FUNC_MUX_CTRL register must be programmed prior to writing 0x0000EAEF to COMP_MODE_CTRL_0.

For example, in reset mode 1, pin Y4 defaults to SYS_CLK_IN (or mux mode 6) rather than UART2.BCLK (mux mode 0). However, FUNC_MUX_CTRL_D(2:0) bits are at their reset value of 000. The FUNC_MUX_CTRL_D(2:0) bits must be set to 110 (mux mode 6) prior to writing 0x0000EAEF to COMP_MODE_CTRL_0 in order for SYS_CLK_IN to maintain its functionality. Otherwise, the system clock is killed when 0x0000EAEF is written to COMP_MODE_CTRL_0 (because the default mux mode 0, UART2.BCLK, takes effect).

In a similar manner, the reset mode determines how the pullup/pulldown configuration is set.

Finally, it is important to note that GPIO1 affects the default pin multiplexing for some of the EMIFS pins if reset mode is 0. The pins are K7, H3, H4, K8, G2, G3, G4, F3, J7, E3, F4, D2, E4, C1, D3 and J8.

2.2.2 Multiplexing Exceptions with USB Host Client

The pin muxing for USB can be overridden by a JTAG sequence. USB host client multiplexing is an exception to the previously described pin multiplexing and pullup-/pulldown-enable generation. For more information contact your TI representative.

2.2.3 Configuration of USB Ports 0, 1, and 2

USB port 0 has an integrated USB transceiver cell able to operate UART or I^2C transactions (3-V mode). It also can be programmed to enable an external pullup.

The default mode is USB mode.

Programming of the UART, I^2C modes, and control of the external pullup are done in the USB_TRANSCEIVER_CTRL register.

The USB ports 1 or 2 can be configured to operate with USB IC transceivers, whose interfaces are either bidirectional or unidirectional. Configuration is done with the USB_TRANSCEIVER_CTRL[8:7] register bits.

See the Multimedia Processor Universal Serial Bus (USB) Reference Guide (SPRU761) for additional information.

2.2.4 Procedure for Setting the Pin Multiplexing

The value of the RESET_MODE pin can be read in the CONF_RESET_MODE_STAT_R bit field of the CONF_STATUS register:

- All mux mode registers (FUNC_MUX_CTRL(3-12)) reset to 000, regardless of RESET_MODE input pin status.
- All pullup/pulldown enable registers (PULL_DWN_CTRL(0-4)) reset to 0, regardless of the RESET_MODE input pin status. However, the actual states of the pulldowns does not depend on the values in the PULL_DWN_CTRL(0–4) registers until programming 0x0000EAEF in the COMP_MODE_CTRL_0 register.
- All pullup/pulldown select registers (PU_PD_SEL(0-4)) reset to 0, regardless of the RESET_MODE input pin status.

See the Application Processor Data Manual (SPRS231) for default states for pin multiplexing and pulls.

Note:

Register values are ignored until 0x0000EAEF is written to the COMP_MODE_CTRL_0 register.

When configuring the pinout of the device:

1) Determine the desired values for each FUNC_MUX_CTRL (3–12), PU_PD_SEL(0-4), and PULL_DWN_CTRL (0–4) configuration register.
2) Program the desired values by writing to the appropriate register.
3) Program the COMP_MODE_CTRL_0 register to 0x0000EAEF.

This procedure allows the user to make all of the multiplex configuration settings and enable all of the modes at once. The bit values as they correspond to pin multiplexing modes are shown in Table 8.

The desired pin multiplexing and the pullup/pulldown modes are then activated. Once 0x0000EAEF is written to COMP_MODE_CTRL_0, it must not be changed because the device will revert to the original multiplexing and pullup/down states.

Table 8. Functional Multiplexing Modes

FUNC_MUX_CTRL 3-Bit Value	Corresponding Multiplexing Mode for Pin Configuration
000	Default configuration/functional multiplexing 0
001	Functional pin multiplexing mode 1
010	Functional pin multiplexing mode 2
011	Functional pin multiplexing mode 3
100	Functional pin multiplexing mode 4
101	Functional pin multiplexing mode 5
110	Functional pin multiplexing mode 6
111	Functional pin multiplexing mode 7

For a given interface, the values of the FUNC_MUX_CTRL bits may vary from pin to pin. For example, the USB1_HOST port is split between functional multiplexing 1 and functional multiplexing 2 columns. In this case, four of the pins have a mode value of 001, and the other four have a mode value of 010.

Note:

The programming of a pin in a mux mode that is not defined in the pinout section of the Application Processor Data Manual (SPRS231) leads to undefined operation of the pin, and thus must be strictly avoided.

2.3 Parallel Observability During Functional Mode

For debug purposes, various internal signals can be brought to the boundary during functional mode. The software procedure for enabling observability of internal signals is as follows:

- Set FUNC_MUX_CTRL_2.CONF_OBS_MUX_SEL_R, depending on the signal that must be observed (see Table 9).
- Set FUNC_MUX_CTRL_0.OBS_288_1 bit to 1.

There is no need to write 0xEAEF to COMP_MODE_CTRL_0 to set observability mode. Table 9 gives the observable signals, and their locations on the boundary, depending on CONF_OBS_MUX_SEL_R.

Table 9. Parallel Observability Multiplexing Signals

	CONF_OBS_MUX_SEL_R Value			
Ball	**0**	**1**	**2**	**3**
J18	Reserved	ARMXOR_CK	LDO_SLEEP	ULPD_DPLL48M
J19	Reserved	DSPXOR_CK	LDO_STEADY	ULPD_ USBW2FCCLK12M
Ball	**0**	**1**	**2**	**3**
J14	USB RXD	ARMPER_CK	LDO_PWRDN	OMAP3.2 functional clock
K18	USB RXDP	DSPPER_CK	UART3 functional clock	Reserved
K19	USB RXDM	Reserved	UART2 functional clock	Reserved
K15	Reserved	Reserved	UART1 functional clock	Reserved
Ball	**0**	**1**	**2**	**3**
K14	Reserved	Reserved	MMC2 functional clock	Reserved
L19	Reserved	Reserved	MMC1 functional clock	Reserved
	4	**5**	**6**	**7**
J18	MCLK	DPLL2 clock	LDO sleep	APLL PWRDN
J19	BCLK	Reserved	MPU DMA REQ observable	APLL SYNC
J14	BCLK	Reserved	DSP DMA REQ observable	OMAP3.2 MPU warmreset
K18	USB OTG functional clock	Reserved	MPU interrupt observable	CLK12M STABLE
K19	USB host functional clock	DPLL1 clock	DSP interrupt observable	APLL output clock
K15	Camera functional clock	Lv2 INTH functional clock	BIST FAIL	WKUP_NREQ
K14	CAM.OUTCLK	OMAP3.2 Idle	BIST DONE	APLL LOCK

Table 9. Parallel Observability Multiplexing Signals (Continued)

	CONF_OBS_MUX_SEL_R Value			
L19	SYS_CLK_OUT	OMAP3.2 DPLL enable	BIST GO	CHIP_NWAKEUP
	8	**9**	**10**	
J18	DLL_PMT_DCB[7]	ULPD clock switch status	RNG functional clock	
J19	DLL_PMT_DCB[6]	UART2 functional clock (from system clock)	Reserved	
	8	**9**	**10**	
J14	DLL_PMT_DCB[5]	Internal 32k clock (distributed to peripherals)	Reserved	
K18	DLL_PMT_DCB[4]	Internal system clock	Reserved	
K19	DLL_PMT_DCB[3]	Internal 32k clock (feeding ULPD)	Reserved	
K15	DLL_PMT_DCB[2]	32k oscillator PWRDN	Reserved	
K14	DLL_PMT_DCB[1]	12-MHz oscillator GZ	Reserved	
L19	DLL_PMT_DCB[0]	12-MHz oscillator PWRDN	Reserved	

2.4 OMAP5912/5910 Software and Hardware Compatibility

The device resets based on the RESET_MODE pin are forced either to the functional multiplexing mode 000, with the appropriate pullups or pulldowns configured and enabled to preserve the OMAP5910 pinout reset condition, or to the new reset mode 1. The multiplexing per pin remains software compatible with OMAP5910 and is controlled by the same 3-bit field (eight mux modes possible).

Pullup and pulldown control has changed from OMAP5910 to OMAP5912. As applicable, the OMAP5910 device has either a pullup or a pulldown per pin, and a 1-bit register per pin enables or disables the pullup or pulldown. The OMAP5912 device, however, has both a pullup and a pulldown available on each pin. Thus, a new OMAP5912 register set is implemented (PU_PD_SEL (0-4)) to allow the software to select between the use of a pullup or a pulldown on each pin. The reset condition of PU_PD_SEL(0-4) is determined, to keep compatibility with the OMAP5910 reset condition. The OMAP5910 pullup/pulldown enable register set remains (PULL_DWN_CTRL (0-3). The reset state, however, is changed to match what the pinout requires at reset.

In the OMAP5910, programming of the following registers does not take effect until the COMP_MODE_CTRL_0 register is written with 0x0000EAEF. For OMAP5912, however, these registers do not have this restriction, and programming the register takes effect immediately, regardless of the state of the COMP_MODE_CTRL_0 register.

- GATE_INH_CTRL_0
- VOLTAGE_CTRL_0
- TEST_DBG_CTRL_0
- MOD_CONF_CTRL_0

For OMAP5912, all I/O power supplies have a low- and a high-voltage mode VS. In OMAP5910, only the EMIFS, EMIFF, and communication processor interface have dual voltage interfaces. For OMAP5912 at reset, the voltage mode has changed to default to low-voltage mode, whereas in OMAP5910 it is high-voltage mode.

The OMAP5912 software/hardware compatibility summary for I/O multiplexing and configuration is as follows:

- The device resets to functional multiplexing mode 000 with appropriate pullups or pulldowns configured and enabled to preserve the OMAP5910 pinout reset condition if RESET_MODE pin is set to 0.
- Multiplexing-per-pin stays the same as in the OMAP5910 mode. More pins are multiplexed in OMAP5912, so that there are additional FUNC_MUX_CTRL registers.
- The PULL_DWN_CTRL register set remains unchanged. It is responsible for pullup or pulldown enable, as determined by the state of the newly created PU_PD_SEL register set. Because there are additional pins requiring pullup/pulldown capability, extra PULL_DWN_CTRL registers have been added. Because not all pins need a pullup or pulldown function, see the OMAP5912 Application Processor Data Manual (SPRS231) to determine whether a pullup/pulldown is implemented.

Table 10 lists the configuration registers. Table 11 through Table 52 describe the register bits.

2.5 Configuration Registers

Table 10. Configuration Registers

Base Address = 0xFFFE 1000			
Name	**Description**	**R/W**	**Offset**
FUNC_MUX_CTRL_0	Functional multiplexing control 0	R/W	0x00
FUNC_MUX_CTRL_1	Functional multiplexing control 1	R/W	0x04
FUNC_MUX_CTRL_2	Functional multiplexing control 2	R/W	0x08
COMP_MODE_CTRL_0	Compatibility mode control 0	R/W	0x0C
FUNC_MUX_CTRL_3	Functional multiplexing control 3	R/W	0x10
FUNC_MUX_CTRL_4	Functional multiplexing control 4	R/W	0x14
FUNC_MUX_CTRL_5	Functional multiplexing control 5	R/W	0x18
FUNC_MUX_CTRL_6	Functional multiplexing control 6	R/W	0x1C
FUNC_MUX_CTRL_7	Functional multiplexing control 7	R/W	0x20
FUNC_MUX_CTRL_8	Functional multiplexing control 8	R/W	0x24
FUNC_MUX_CTRL_9	Functional multiplexing control 9	R/W	0x28
FUNC_MUX_CTRL_A	Functional multiplexing control	R/W	0x2C
FUNC_MUX_CTRL_B	Functional multiplexing control	R/W	0x30
FUNC_MUX_CTRL_C	Functional multiplexing control	R/W	0x34
FUNC_MUX_CTRL_D	Functional multiplexing control	R/W	0x38
PULL_DWN_CTRL_0	Pulldown control 0	R/W	0x40
PULL_DWN_CTRL_1	Pulldown control 1	R/W	0x44
PULL_DWN_CTRL_2	Pulldown control 2	R/W	0x48
PULL_DWN_CTRL_3	Pulldown control 3	R/W	0x4C
GATE_INH_CTRL_0	Gate and inhibit control 0	R/W	0x50
CONF_REV	Configuration revision	R	0x58
VOLTAGE_CTRL_0	Voltage control	R/W	0x60

Table 10. Configuration Registers (Continued)

Base Address = 0xFFFE 1000			
Name	**Description**	**R/W**	**Offset**
USB_TRANSCEIVER_CTRL	USB transceiver control	R/W	0x64
LDO_PWRDN_CBTRK	LDO powerdown control	R/W	0x68
TEST_DBG_CTRL_0	Test debug control 0	R/W	0x70
MOD_CONF_CTRL_0	Module configuration control 0	R/W	0x80
FUNC_MUX_CTRL_E	Functional multiplexing control	R/W	0x90
FUNC_MUX_CTRL_F	Functional multiplexing control	R/W	0x94
FUNC_MUX_CTRL_10	Functional multiplexing control	R/W	0x98
FUNC_MUX_CTRL11	Functional multiplexing control	R/W	0x9C
FUNC_MUX_CTRL_12	Functional multiplexing control	R/W	0xA0
PULL_DWN_CTRL_4	Pulldown control 4	R/W	0xAC
PU_PD_SEL_0	Pullup/pulldown selection 0	R/W	0xB4
PU_PD_SEL_1	Pullup/pulldown selection 1	R/W	0xB8
PU_PD_SEL_2	Pullup/pulldown selection 2	R/W	0xBC
PU_PD_SEL_3	Pullup/pulldown selection 3	R/W	0xC0
PU_PD_SEL_4	Pullup/pulldown selection 4	R/W	0xC4
FUNC_MUX_DSP_DMA_A	Functional multiplexing DSP DMA A	R/W	0xD0*
FUNC_MUX_DSP_DMA_B	Functional multiplexing DSP DMA B	R/W	0xD4*
FUNC_MUX_DSP_DMA_C	Functional multiplexing DSP DMA C	R/W	0xD8*
FUNC_MUX_DSP_DMA_D	Functional multiplexing DSP DMA D	R/W	0xDC*
FUNC_MUX_ARM_DMA_A	Functional multiplexing MPU DMA A	R/W	0xEC*
FUNC_MUX_ARM_DMA_B	Functional multiplexing MPU DMA B	R/W	0xF0*
FUNC_MUX_ARM_DMA_C	Functional multiplexing MPU DMA C	R/W	0xF4*
FUNC_MUX_ARM_DMA_D	Functional multiplexing MPU DMA D	R/W	0xF8*
FUNC_MUX_ARM_DMA_E	Functional multiplexing MPU DMA E	R/W	0xFC*
FUNC_MUX_ARM_DMA_F	Functional multiplexing MPU DMA F	R/W	0x100*

Table 10. Configuration Registers (Continued)

Base Address = 0xFFFE 1000			
Name	**Description**	**R/W**	**Offset**
FUNC_MUX_ARM_DMA_G	Functional multiplexing MPU DMA G	R/W	0x104*
MOD_CONF_CTRL_1	Module configuration control 1	R/W	0x110
CONF_STATUS	Configuration status		0x130
RESET_CONTROL	Reset control register	R/W	0x140
CONF_5912_CTRL	OMAP5912 configuration control	R/W	0x150

Note: Register descriptions for DMA functional mux registers are in the Multimedia Processor Direct Memory Access (DMA) Support Reference Guide (SPRU755).

Table 11. Functional Multiplexing Control O Register (FUNC_MUX_CTRL_0)

Base Address = 0xFFFE 1000, Offset Address = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
31	CTRL_288_1	This bit configures the control mode 288_1, which enables the control of the OMAP CHIP_NWAKEUP signal from the RTCK pad. 0: Functional mode. ULPD controls the OMAP CHIP_NWAKEUP signal. 1: Debug. The RTCK pad controls the OMAP CHIP_NWAKEUP signal.	R/W	0x0
30	OBS_288_2	See Note 1	R/W	0x0
29	OBS_600_1	See Note 1	R/W	0x0

Notes: 1) This function has been removed. Writing a 1 or a 0 to this register is acceptable. However, it is recommended to write a 0, in case functions are added to this register space in the future.

Table 11. Functional Multiplexing Control O Register (FUNC_MUX_CTRL_0) (Continued)

Base Address = 0xFFFE 1000, Offset Address = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
28	OBS_288_1	Configures observation mode multiplexing on the camera interface (See Section 2.3). 0: Observation mode is disabled and camera interface is in functional mode. 1: Enables the observation mode. Programming sequence: 1) Configure CONF_OBS_MUX_CTRL_R (see Section 2.3 for details). 2) Enable the observability multiplexing by programming OBS_288_1 to 1.	R/W	0x0
27	UWIRE_HIZ_DISABLE	See Note 1	R/W	0x0
26	GIGA_UART_GATING	See Note 1	R/W	0x0
25	BT_UART_GATING	See Note 1	R/W	0x0
24	COM_UART_GATING	See Note 1	R/W	0x0
23	USB_TRANS_MUX	See Note 1	R/W	0x0
22	LB_RESET_DISABLE	See Note 1	R/W	0x0
21	HSAB_RESET_DISABLE	See Note 1	R/W	0x0
20	LRU_SEL	This field configures the OMAP traffic controller arbitration algorithm. 0: LRU priority scheme is used for arbitration. 1: Fixed priority scheme is used for arbitration. This bit can be changed only when the DSP is in reset.	R/W	0x0
19	VBUS_CTRL	See Note 1	R/W	0x0
18	VBUS_MODE	See Note 1	R/W	0x0
17:16	UWIRE_DCD	See Note 1	R/W	0x0
15	OS_TYPE	Reserved	R/W	0x0

Notes: 1) This function has been removed. Writing a 1 or a 0 to this register is acceptable. However, it is recommended to write a 0, in case functions are added to this register space in the future.

Table 11. Functional Multiplexing Control O Register (FUNC_MUX_CTRL_0) (Continued)

Base Address = 0xFFFE 1000, Offset Address = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
14	NRESET_ENABLE	Allows AND gating of outputs with the OMAP CHIP_NRESET_OUT 0: Disabled 1: Enabled	R/W	0x0
13	PWR_MASK_IN	This register enables the Inhibit2 function. Depending on the status of GPIO(9) and ARMIO(3), subject inputs are gated to low internally when this function is enabled. 1: Enable the gating of inputs by the combination of COM_PWR_REQ (GPIO(9)) and COM_STS (MPUIO(3)) input pins. 0: Disable the gating of inputs by the combination of COM_PWR_REQ (GPIO(9)) and COM_STS (MPUIO(3)) input pins. This is the control for the function called Inhibit2 in SPRS231.	R/W	0x0
12	PWR_MASK_OUT	This register enables the Gated2 function. Depending on the status of GPIO(9) and ARMIO(3), the subject outputs are gated to low when this function is enabled. 1: Enable AND gating of outputs with COM_PWR_REQ (GPIO(9)) and COM_STS (MPUIO(3)) input pins. 0: Disables AND gating of outputs with COM_PWR_REQ (GPIO(9)) and COM_STS (MPUIO(3)) input pins. This function is called Gated2 in the SPRS231 document.	R/W	0x0

Notes: 1) This function has been removed. Writing a 1 or a 0 to this register is acceptable. However, it is recommended to write a 0, in case functions are added to this register space in the future.

Table 11. Functional Multiplexing Control O Register (FUNC_MUX_CTRL_0) (Continued)

Base Address = 0xFFFE 1000, Offset Address = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
11	BVLZ_MASK_IN	This register enables the Inhibit1 function. Depending on the status of BVLZ input pin, subject inputs are gated to low internally when this function is enabled. 1: Enables AND gating of inputs with BVLZ input pin. 0: Disables AND gating of inputs with BVLZ input pin. This function is called Inhibit1 in SPRS231.	R/W	0x0
10	BVLZ_MASK_OUT	This register enables the Gated1 function. Depending on the status of BVLZ, subject outputs are gated to low when this function is enabled. 1: Enables AND gating of outputs with BVLZ input pin. 0: Disables AND gating of outputs with BVLZ input pin. This function is called Gated1 in SPRS231.	R/W	0x0
9	BLUETOOTH	See Note 1	R/W	0x0
8	CAMERA	See Note 1	R/W	0x0
7	USB.CLK0	See Note 1	R/W	0x0
6	COM_SHUT	See Note 1	R/W	0x0
5	GIGA_UART	See Note 1	R/W	0x0
4	MEDIA_HIZ_DISABLE	See Note 1	R/W	0x0
3	LCD	See Note 1	R/W	0x0
2	NFCS2	See Note 1	R/W	0x0
1:0	LB_HSAB_RHEA	See Note 1	R/W	0x0

Notes: 1) This function has been removed. Writing a 1 or a 0 to this register is acceptable. However, it is recommended to write a 0, in case functions are added to this register space in the future.

This register primarily controls legacy functional multiplexing. Many mux modes have been removed and made more generic. When a function has been removed, it is noted in the register description.

Table 12. Functional Multiplexing Control 1 Register (FUNC_MUX_CTRL_1)

Base Address = 0xFFFE 1000, Offset Address = 0x04				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:0	RESERVED	Reserved †	R/W	0x0000

† The register functions have been removed. Writing a 1 or a 0 to this register is acceptable. However, it is recommended to write a 0, in case functions are added to this register space in the future.

Table 13. Functional Multiplexing Control 2 Register (FUNC_MUX_CTRL_2)

Base Address = 0xFFFE 1000, Offset Address = 0x08				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:28	RESERVED	Reserved	R/W	0x0
27:24	CONF_OBS_MUX_SEL_R	Sets the parallel observation multiplexing mode. These bits determine which of the 10 observation multiplexing modes is selected (See Section 2.3 for details).	R/W	0x0
23:19	CONF_DSP_DMA_REQ_19	Maps DSP DMA_REQUEST for observability. The pin configuration for each observation mode is specified (See Section 2.3). Observability of the DSP DMA request(i) is done before the crossbar.	R/W	0x00
18:13	CONF_ARM_DMA_SEL_REQ31	Maps MPU DMA_REQUEST for observability. The pin configuration for each observation mode is specified (See Section 2.3). Observability of the MPU DMA request(i) is done before the crossbar.	R/W	0x00

Table 13. Functional Multiplexing Control 2 Register (FUNC_MUX_CTRL_2)

Base Address = 0xFFFE 1000, Offset Address = 0x08				
Bit	**Name**	**Function**	**R/W**	**Reset**
12:7	CONF_DSP_INT_SEL_R	Select the DSP level 2 interrupts to be observed. The pin configuration for each observation mode is specified (See Section 2.3). Observing interrupts is done after multiplexers that select between level and edge activity.	R/W	0x00
6:0	CONF_ARM_INT_SEL_R	Select the MPU level 2 interrupts to be observed. The pin configuration for each observation mode is specified (See Section 2.3). Observing interrupts is done after multiplexers that select between level and edge activity.	R/W	0x00

Table 14. Compatibility Mode Control 0 Register (COMP_MODE_CTRL_0)

Base Address = 0xFFFE 1000, Offset Address = 0x0C				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:16	CONF_COMP_ RESERVED_R	Reserved for future expansion. These bits must be written to 0x0000h	R/W	0x0000
15:0	CONF_MUX_EN_R	A value of 0xEAEF must be written to this register to enable the pin multiplexing controlled by the registers FUNC_MUX_CTRL (3-12), PU_PD_SEL(0-4), and PULL_DWN_CTRL (0-4). The boot code must configure all the pin multiplexing registers FUNC_MUX_CTRL (3-12), PU_PD_SEL(0-4), and PULL_DWN_CTRL (0-4) first. Writing 0xEAEF enables all the pin multiplexing to take effect simultaneously.	R/W	0x0000

This is the I/O multiplex enable register. At reset, any value written to the registers FUNC_MUX_CTRL (3-12), PU_PD_SEL(0-4), and PULL_DWN_CTRL (0-4) does not change the configuration of the pins. However, setting the COMP_MODE_CTRL_0 register to 0x0000EAEF activates:

- Generic multiplexing registers: FUNC_MUX_CTRL_x, $x \in \{3..12\}$
- Pullup/pulldown enable registers: PULL_DWN_CTRL_y, $y \in \{0..4\}$
- Pullup/pulldown select registers: PU_PD_SEL_z, $z \in \{0..4\}$

Table 15. Functional Multiplexing Control 3 Register (FUNC_MUX_CTRL_3)

Base Address = 0xFFFE 1000, Offset Address = 0x10				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:30	RESERVED	Reserved for future expansion.	R/W	0x0
29:27	CONF_F19	Controls the multiplexing on F19. Formerly, CONF_KBR_1_R.	R/W	000
26:24	CONF_H14	Controls the multiplexing on H14. Formerly, CONF_KBR_2_R.	R/W	000
23:21	CONF_E20	Controls the multiplexing on E20. Formerly, CONF_KBR_3_R.	R/W	000
20:18	CONF_E19	Controls the multiplexing on E19. Formerly, CONF_KBR_4_R.	R/W	000
17:15	CONF_F18	Controls the multiplexing on F18. Formerly, CONF_KBC_0_R.	R/W	000
14:12	CONF_D20	Controls the multiplexing on D20. Formerly, CONF_KBC_1_R.	R/W	000
11:9	CONF_D19	Controls the multiplexing on D19. Formerly, CONF_KBC_2_R.	R/W	000
8:6	CONF_E18	Controls the multiplexing on E18. Formerly, CONF_KBC_3_R.	R/W	000
5:3	CONF_C21	Controls the multiplexing on C21. Formerly, CONF_KBC_4_R.	R/W	000
2:0	CONF_G19	Controls the multiplexing on G19. Formerly, CONF_KBC_5_R.	R/W	000

This register controls functional multiplexing. COMP_MODE_CTRL_0 must be programmed to 0xEAEFh for this register to control functional multiplexing. See Table 8 for bit−field values.

Table 16. Functional Multiplexing Control 4 Register (FUNC_MUX_CTRL_4)

Base Address = 0xFFFE 1000, Offset Address = 0x14				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:30	RESERVED	Reserved for future expansion.	R/W	0x0
29:27	CONF_J18	Controls multiplexing on J18. Formerly CON_CAM_D_7_R.	R/W	000
26:24	CONF_J15	Controls multiplexing on J15. Formerly CONF_CAM_LCLK_R.	R/W	000
23:21	CONF_H19	Controls multiplexing on H19. Formerly CONF_CAMEXCLK_R.	R/W	000
20:18	CONF_H20	Controls multiplexing on H20. Formerly CONF_MCBSP1_DIN_R.	R/W	000
17:15	CONF_H18	Controls multiplexing on H18. Formerly CONF_MCBSP1_DOUT_R.	R/W	000

Table 16. Functional Multiplexing Control 4 Register (FUNC_MUX_CTRL_4) (Continued)

Base Address = 0xFFFE 1000, Offset Address = 0x14				
Bit	**Name**	**Function**	**R/W**	**Reset**
14:12	CONF_H15	Controls multiplexing on H15. Formerly CONF_MCBSP1_SYNC_R.	R/W	000
11:9	CONF_G21	Controls multiplexing on G21. Formerly CONF_MCBSP1_BCLK_R.	R/W	000
8:6	CONF_G20	Controls multiplexing on G20. Formerly CONF_MCBSP1_CLKS_R.	R/W	000
5:3	RESERVED	Reserved for future expansion	R/W	000
2:0	CONF_G18	Controls multiplexing on G18. Formerly CONF_KBR_0_R.	R/W	000

This register controls functional multiplexing. COMP_MODE_CTRL_0 must be programmed to 0xEAEFh for this register to control functional multiplexing. See Table 8 for bit field values.

Table 17. Functional Multiplexing Control 5 Register (FUNC_MUX_CTRL_5)

Base Address = 0xFFFE 1000, Offset Address = 0x18				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:30	RESERVED	Reserved for future expansion.	R/W	0x0
29:27	CONF_M19	Controls multiplexing on M19. Formerly, CONF_CAM_RSTZ_R.	R/W	0x0
26:24	CONF_L15	Controls multiplexing on L15. Formerly, CONF_CAM_HS_R.	R/W	0x0
23:21	CONF_L18	Controls multiplexing on L18. Formerly, CONF_CAM_VS_R.	R/W	0x0
20:18	CONF_L19	Controls multiplexing on L19. Formerly, CONF_CAM_D_0_R.	R/W	0x0
17:15	CONF_K14	Controls multiplexing on K14. Formerly, CONF_CAM_D_1_R.	R/W	0x0
14:12	CONF_K15	Controls multiplexing on K15. Formerly, CONF_CAM_D_2_R.	R/W	0x0
11:9	CONF_K19	Controls multiplexing on K19. Formerly, CONF_CAM_D_3_R.	R/W	0x0
8:6	CONF_K18	Controls multiplexing on K18. Formerly, CONF_CAM_D_4_R.	R/W	0x0
5:3	CONF_J14	Controls multiplexing on J14. Formerly, CONF_CAM_D_5_R.	R/W	0x0
2:0	CONF_J19	Controls multiplexing on J19. Formerly, CONF_CAM_D_6_R.	R/W	0x0

This register controls functional multiplexing. COMP_MODE_CTRL_0 must be programmed to 0xEAEFh for this register to control functional multiplexing. See Table 8 for bit field values.

Table 18. Functional Multiplexing Control 6 Register (FUNC_MUX_CTRL_6)

Base Address = 0xFFFE 1000, Offset Address = 0x1C				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:30	RESERVED	Reserved for future expansion.	R/W	0x0
29:27	CONF_P20	Controls multiplexing on P20. Formerly, CONF_GPIO_4_R.	R/W	0x0
26:24	CONF_P19	Controls multiplexing on P19. Formerly, CONF_GPIO_6_R.	R/W	0x0
23:21	CONF_M15	Controls multiplexing on M15. Formerly, CONF_GPIO_7_R.	R/W	0x0
20:18	CONF_N20	Controls multiplexing on N20. Formerly, CONF_GPIO_11_R.	R/W	0x0
17:15	CONF_N18	Controls multiplexing on N18. Formerly, CONF_GPIO_11_R.	R/W	0x0
14:12	CONF_N19	Controls multiplexing on N19. Formerly, CONF_GPIO_11_R.	R/W	0x0
11:9	CONF_N21	Controls multiplexing on N21. Formerly, CONF_GPIO_11_R.	R/W	0x0
8:6	CONF_M20	Controls multiplexing on M20. Formerly, CONF_GPIO_11_R.	R/W	0x0
5:3	CONF_L14	Controls multiplexing on L14. Formerly, CONF_RX3_R.	R/W	0x0
2:0	CONF_M18	Controls multiplexing on M18. Formerly, CONF_TX3_R.	R/W	0x0

This register controls functional multiplexing. COMP_MODE_CTRL_0 must be programmed to 0xEAEFh for this register to control functional multiplexing. See Table 8 for bit field values.

Table 19. Functional Multiplexing Control 7 Register (FUNC_MUX_CTRL_7)

Base Address = 0xFFFE 1000, Offset Address = 0x20				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:30	RESERVED	Reserved for future expansion.	R/W	0x0
29:27	CONF_V20	Controls multiplexing on V20. Formerly, CONF_SDA_R.	R/W	0x0
26:24	CONF_T18	Controls multiplexing on T18. Formerly, CONF_SCL_R.	R/W	0x0
23:21	CONF_U19	Controls multiplexing on U19. Formerly, CONF_ARMIO_1_R.	R/W	0x0
20:18	CONF_N15	Controls multiplexing on N15. Formerly, CONF_ARMIO_2_R.	R/W	0x0

Table 19. Functional Multiplexing Control 7 Register (FUNC_MUX_CTRL_7) (Continued)

Base Address = 0xFFFE 1000, Offset Address = 0x20				
Bit	**Name**	**Function**	**R/W**	**Reset**
17:15	CONF_T19	Controls multiplexing on T19. Formerly, CONF_ARMIO_4_R.	R/W	0x0
14:12	CONF_T20	Controls multiplexing on T20. Formerly, CONF_ARMIO_5_R.	R/W	0x0
11:9	CONF_R18	Controls multiplexing on R18. Formerly, CONF_GPIO_0_R.	R/W	0x0
8:6	CONF_R19	Controls multiplexing on R19. Formerly, CONF_GPIO_1_R.	R/W	0x0
5:3	CONF_M14	Controls multiplexing on M14. Formerly, CONF_GPIO_2_R.	R/W	0x0
2:0	CONF_P18	Controls multiplexing on P18. Formerly, CONF_GPIO_3_R.	R/W	0x0

This register controls functional multiplexing. COMP_MODE_CTRL_0 must be programmed to 0xEAEFh for this register to control functional multiplexing. See Table 8 for bit field values.

Table 20. Functional Multiplexing Control 8 Register (FUNC_MUX_CTRL_8)

Base Address = 0xFFFE 1000, Offset Address = 0x24				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:30	RESERVED	Reserved for future expansion.	R/W	0x0
29:27	CONF_J20	Controls multiplexing on J20. Formerly, CONF_ARM_BOOT_R.	R/W	0x0
26:24	CONF_W17	Controls multiplexing on W17. Formerly, CONF_NEMU1_R.	R/W	0x0
23:21	CONF_V16	Controls multiplexing on V16. Formerly, CONF_NEMU0_R.	R/W	0x0
20:18	CONF_Y12	Controls multiplexing on Y12. Formerly, CONF_ON_OFF_R.	R/W	0x0
17:15	CONF_W19	Controls multiplexing on W19. Formerly, CONF_BVLZ_R.	R/W	0x0
14:12	CONF_P15	Controls multiplexing on P15. Formerly, CONF_WIRE_NSCS3_R.	R/W	0x0
11:9	CONF_N14	Controls multiplexing on N14. Formerly, CONF_WIRE_NSCS0_R.	R/W	0x0
8:6	CONF_V19	Controls multiplexing on V19. Formerly, CONF_WIRE_SCLK_R.	R/W	0x0
5:3	CONF_W21	Controls multiplexing on W21. Formerly, CONF_WIRE_SDO_R.	R/W	0x0
2:0	CONF_U18	Controls multiplexing on U18. Formerly, CONF_WIRE_SDI_R.	R/W	0x0

This register controls functional multiplexing. COMP_MODE_CTRL_0 must be programmed to 0xEAEFh for this register to control functional multiplexing. See Table 8 for bit field values.

Table 21. Functional Multiplexing Control 9 Register (FUNC_MUX_CTRL_9)

Base Address = 0xFFFE 1000, Offset Address = 0x28				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:30	RESERVED	Reserved for future expansion.	R/W	0x0
29:27	CONF_W15	Controls multiplexing on W15. Formerly CONF_UARTS_CLKREQ_R.	R/W	0x0
26:24	CONF_W14	Controls multiplexing on W14. Formerly CONF_MCSI1_DOUT_R.	R/W	0x0
23:21	CONF_Y14	Controls multiplexing on Y14. Formerly CONF_TX1_R.	R/W	0x0
20:18	CONF_V14	Controls multiplexing on V14. Formerly CONF_RX1_R.	R/W	0x0
17:15	CONF_R14	Controls multiplexing on R14. Formerly CONF_CTS1_R.	R/W	0x0
14:12	CONF_AA15	Controls multiplexing on AA15. Formerly CONF_RTS1_R.	R/W	0x0
11:9	CONF_AA20	Controls multiplexing on AA20. Formerly CONF_NRESET_OUT_R.	R/W	0x0
8:6	CONF_U20	Controls multiplexing on U20. Formerly CONF_MPU_NRESET_R.	R/W	0x0
5:3	CONF_W16	Controls multiplexing on W16. Formerly CONF_MCBSP3_CLK_R.	R/W	0x0
2:0	CONF_W13	Controls multiplexing on W13. Formerly CONF_WAKEUP_INT_R.	R/W	0x0

This register controls functional multiplexing. COMP_MODE_CTRL_0 must be programmed to 0xEAEFh for this register to control functional multiplexing. See Table 8 for bit field values.

Table 22. Functional Multiplexing Control A Register (FUNC_MUX_CTRL_A)

Base Address = 0xFFFE 1000, Offset Address = 0x2C				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:30	RESERVED	Reserved for future expansion.	R/W	0x0
29:27	CONF_P11	Controls multiplexing on P11. Formerly CONF_MMC_CMD_R.	R/W	0x0

Table 22. Functional Multiplexing Control A Register (FUNC_MUX_CTRL_A) (Continued)

Base Address = 0xFFFE 1000, Offset Address = 0x2C				
Bit	**Name**	**Function**	**R/W**	**Reset**
26:24	CONF_V10	Controls multiplexing on V10. Formerly CONF_MMC_DAT1_R.	R/W	0x0
23:21	CONF_V11	Controls multiplexing on V11. Formerly CONF_MMC_CLK_R.	R/W	0x0
20:18	CONF_W10	Controls multiplexing on W10. Formerly CONF_MMC_DAT2_R.	R/W	0x0
17:15	CONF_P13	Controls multiplexing on P13. Formerly CONF_CLK32K_IN_R.	R/W	0x0
14:12	CONF_R13	Controls multiplexing on R13. Formerly CONF_CLK32K_OUT_R.	R/W	0x0
11:9	CONF_V15	Controls multiplexing on V15. Formerly CONF_MCSI1_DIN_R.	R/W	0x0
8:6	CONF_P14	Controls multiplexing on P14. Formerly CONF_MCSI1_BCLK_R.	R/W	0x0
5:3	CONF_AA17	Controls multiplexing on AA17. Formerly CONF_MCSI1_SYNC_R.	R/W	0x0
2:0	CONF_Y15	Controls multiplexing on Y15. Formerly CONF_BCLK_R.	R/W	0x0

This register controls functional multiplexing. COMP_MODE_CTRL_0 must be programmed to 0xEAEFh for this register to control functional multiplexing. See Table 8 for bit field values.

Table 23. Functional Multiplexing Control B Register (FUNC_MUX_CTRL_B)

Base Address = 0xFFFE 1000, Offset Address = 0x30				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:30	RESERVED	Reserved for future expansion.	R/W	0x0
29:27	CONF_V8	Controls multiplexing on V8. Formerly CONF_ARMIO_3_R.	R/W	0x0
26:24	CONF_Y8	Controls multiplexing on Y8. Formerly CONF_GPIO_8_R.	R/W	0x0
23:21	CONF_W8	Controls multiplexing on W8. Formerly CONF_GPIO_9_R.	R/W	0x0
20:18	CONF_R10	Controls multiplexing on R10. Formerly CONF_COM_MCLK_REQ_R.	R/W	0x0
17:15	CONF_V5	Controls multiplexing on V5. Formerly CONF_COM_MCLK_OUT_R.	R/W	0x0
14:12	CONF_V9	Controls multiplexing on V9. Formerly CONF_MCSI2_SYNC_R.	R/W	0x0
11:9	CONF_W9	Controls multiplexing on W9 Formerly CONF_MCSI2_DOUT_R.	R/W	0x0
8:6	CONF_AA9	Controls multiplexing on AA9. Formerly CONF_MCSI2_DIN_R.	R/W	0x0

Table 23. Functional Multiplexing Control B Register (FUNC_MUX_CTRL_B)

Base Address = 0xFFFE 1000, Offset Address = 0x30				
Bit	**Name**	**Function**	**R/W**	**Reset**
5:3	CONF_Y10	Controls multiplexing on Y10. Formerly CONF_MCSI2_CLK_R.	R/W	0x0
2:0	CONF_R11	Controls multiplexing on R11. Formerly CONF_MMC_DAT0_R.	R/W	0x0

This register controls functional multiplexing. COMP_MODE_CTRL_0 must be programmed to 0xEAEFh for this register to control functional multiplexing. See Table 8 for bit field values.

Table 24. Functional Multiplexing Control C Register (FUNC_MUX_CTRL_C)

Base Address = 0xFFFE 1000, Offset Address = 0x34				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:30	RESERVED	Reserved for future expansion.	R/W	0x0
29:27	CONF_V6	Controls multiplexing on V6. Formerly CONF_TX2_R.	R/W	0x0
26:24	CONF_W5	Controls multiplexing on W5. Formerly CONF_RTS2_R.	R/W	0x0
23:21	CONF_Y5	Controls multiplexing on Y5. Formerly CONF_CTS2_R.	R/W	0x0
20:18	CONF_R9	Controls multiplexing on R9. Formerly CONF_RX2_R.	R/W	0x0
17:15	CONF_AA5	Controls multiplexing on AA5. Formerly CONF_MCBSP2_DOUT_R.	R/W	0x0
14:12	CONF_W6	Controls multiplexing on W6. Formerly CONF_MCBSP2_RSYNC_R.	R/W	0x0
11:9	CONF_Y6	Controls multiplexing on Y6. Formerly CONF_MCBSP2_CLKX_R.	R/W	0x0
8:6	CONF_V7	Controls multiplexing on V7. Formerly CONF_MCBSP2_CLKR_R.	R/W	0x0
5:3	CONF_W7	Controls multiplexing on W7. Formerly CONF_MCBSP2_XSYNC_R.	R/W	0x0
2:0	CONF_P10	Controls multiplexing on P10. Formerly CONF_MCBSP2_DIN_R.	R/W	0x0

This register controls functional multiplexing. COMP_MODE_CTRL_0 must be programmed to 0xEAEFh for this register to control functional multiplexing. See Table 8 for bit field values.

Table 25. Functional Multiplexing Control D Register (FUNC_MUX_CTRL_D)

Base Address = 0xFFFE 1000, Offset Address = 0x38				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:30	RESERVED	Reserved for future expansion.	R/W	0x0
29:27	CONF_G13	Controls multiplexing on G13. Formerly CONF_LCD_PIXEL_12_R.	R/W	0x0
26:24	CONF_A17	Controls multiplexing on A17. Formerly CONF_LCD_PIXEL_13_R.	R/W	0x0
23:21	CONF_C16	Controls multiplexing on C16. Formerly CONF_LCD_PIXEL_14_R.	R/W	0x0
20:18	CONF_D15	Controls multiplexing on D15. Formerly CONF_LCD_PIXEL_15_R.	R/W	0x0
17:15	CONF_C15	Controls multiplexing on C15. Formerly CONF_LCD_PCLK_R.	R/W	0x0
14:12	CONF_C20	Controls multiplexing on C20. Formerly CONF_LDC_HSYNC_R.	R/W	0x0
11:9	CONF_B15	Controls multiplexing on B15. Formerly CONF_LCD_AC_R.	R/W	0x0
8:6	CONF_M4	Controls multiplexing on M4. Formerly CONF_NFCS2_R.	R/W	0x0
5:3	CONF_W4	Controls multiplexing on W4. Formerly CONF_USB_PUEN_R.	R/W	0x0
2:0	CONF_Y4	Controls multiplexing on Y4. Formerly CONF_BDCLK2_R.	R/W	0x0

This register controls functional multiplexing. COMP_MODE_CTRL_0 must be programmed to 0xEAEFh for this register to control functional multiplexing. See Table 8 for bit field values.

Table 26. Pulldown Control 0 Register (PULL_DWN_CTRL_0)

Base Address = 0xFFFE 1000, Offset Address = 0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
31	CONF_PDEN_L14	Enables (0) pullup or pulldown on L14.	R/W	0x0
30	CONF_PDEN_M18	Enables (0) pullup or pulldown on M18.	R/W	0x0
29	CONF_PDEN_M19	Enables (0) pullup or pulldown on M19.	R/W	0x0
28	CONF_PDEN_L15	Enables (0) pullup or pulldown on L15.	R/W	0x0
27	CONF_PDEN_L18	Enables (0) pullup or pulldown on L18.	R/W	0x0
26	CONF_PDEN_L19	Enables (0) pullup or pulldown on L19.	R/W	0x0
25	CONF_PDEN_K14	Enables (0) pullup or pulldown on K14.	R/W	0x0
24	CONF_PDEN_K15	Enables (0) pullup or pulldown on K15.	R/W	0x0
23	CONF_PDEN_K19	Enables (0) pullup or pulldown on K19.	R/W	0x0

Table 26. Pulldown Control 0 Register (PULL_DWN_CTRL_0) (Continued)

Base Address = 0xFFFE 1000, Offset Address = 0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
22	CONF_PDEN_K18	Enables (0) pullup or pulldown on K18.	R/W	0x0
21	CONF_PDEN_J14	Enables (0) pullup or pulldown on J14.	R/W	0x0
20	CONF_PDEN_J19	Enables (0) pullup or pulldown on J19.	R/W	0x0
19	CONF_PDEN_J18	Enables (0) pullup or pulldown on J18.	R/W	0x0
18	CONF_PDEN_J15	Enables (0) pullup or pulldown on J15.	R/W	0x0
17	CONF_PDEN_H19	Enables (0) pullup or pulldown on H19.	R/W	0x0
16	CONF_PDEN_H20	Enables (0) pullup or pulldown on H20.	R/W	0x0
15	CONF_PDEN_H18	Enables (0) pullup or pulldown on H18.	R/W	0x0
14	CONF_PDEN_H15	Enables (0) pullup or pulldown on H15.	R/W	0x0
13	CONF_PDEN_G21	Enables (0) pullup or pulldown on G21.	R/W	0x0
12	CONF_PDEN_G20	Enables (0) pullup or pulldown on G20.	R/W	0x0
11	RESERVED	Reserved for future expansion.	R/W	0x0
10	CONF_PDEN_G18	Enables (0) pullup or pulldown on G18.	R/W	0x0
9	CONF_PDEN_F19	Enables (0) pullup or pulldown on F19.	R/W	0x0
8	CONF_PDEN_H14	Enables (0) pullup or pulldown on H14.	R/W	0x0
7	CONF_PDEN_E20	Enables (0) pullup or pulldown on E20.	R/W	0x0
6	CONF_PDEN_E19	Enables (0) pullup or pulldown on E19.	R/W	0x0
5	CONF_PDEN_F18	Enables (0) pullup or pulldown on F18.	R/W	0x0
4	CONF_PDEN_D20	Enables (0) pullup or pulldown on D20.	R/W	0x0
3	CONF_PDEN_D19	Enables (0) pullup or pulldown on D19.	R/W	0x0
2	CONF_PDEN_E18	Enables (0) pullup or pulldown on E18.	R/W	0x0
1	CONF_PDEN_C21	Enables (0) pullup or pulldown on C21.	R/W	0x0
0	CONF_PDEN_G19	Enables (0) pullup or pulldown on G19.	R/W	0x0

This register controls the enable or disable of the combined pullup/pulldown cell (0 = enabled, 1 = disabled). When enabling the cell, the user must set the corresponding bit in PU_PD_SEL_0 to select either a pullup or a pulldown. The

pinout documentation in SPRS231 must be consulted to determine whether a pullup or pulldown exists on the specified I/O. COMP_MODE_CTRL_0 must be programmed to 0xEAEF, so that the programming is taken into account in the corresponding tactical cell.

Table 27. Pulldown Control 1 Register (PULL_DWN_CTRL_1)

Base Address = 0xFFFE 1000, Offset Address = 0x44				
Bit	**Name**	**Function**	**R/W**	**Reset**
31	CONF_PDEN_AA20	Enables (0) pullup or pulldown on AA20.	R/W	0x0
30	CONF_PDEN_U20	Enables (0) pullup or pulldown on U20.	R/W	0x0
29	CONF_PDEN_W16	Enables (0) pullup or pulldown on W16.	R/W	0x0
28	CONF_PDEN_W13	Enables (0) pullup or pulldown on W13.	R/W	0x0
27	CONF_PDEN_J20	Enables (0) pullup or pulldown on J20.	R/W	0x0
26	CONF_PDEN_W17	Enables (0) pullup or pulldown on W17.	R/W	0x0
25	CONF_PDEN_V16	Enables (0) pullup or pulldown on V16.	R/W	0x0
24	CONF_PDEN_Y12	Enables (0) pullup or pulldown on Y12.	R/W	0x0
23	CONF_PDEN_W19	Enables (0) pullup or pulldown on W19.	R/W	0x0
22	CONF_PDEN_P15	Enables (0) pullup or pulldown on P15.	R/W	0x0
21	CONF_PDEN_N14	Enables (0) pullup or pulldown on N14.	R/W	0x0
20	CONF_PDEN_V19	Enables (0) pullup or pulldown on V19.	R/W	0x0
19	CONF_PDEN_W21	Enables (0) pullup or pulldown on W21.	R/W	0x0
18	CONF_PDEN_U18	Enables (0) pullup or pulldown on U18.	R/W	0x0
17	Reserved	Reserved.	R/W	0x0
16	Reserved	Reserved.	R/W	0x0
15	CONF_PDEN_U19	Enables (0) pullup or pulldown on U19.	R/W	0x0
14	CONF_PDEN_N15	Enables (0) pullup or pulldown on N15.	R/W	0x0
13	CONF_PDEN_T19	Enables (0) pullup or pulldown on T19.	R/W	0x0
12	CONF_PDEN_T20	Enables (0) pullup or pulldown on T20.	R/W	0x0
11	CONF_PDEN_R18	Enables (0) pullup or pulldown on R18.	R/W	0x0
10	CONF_PDEN_R19	Enables (0) pullup or pulldown on R19.	R/W	0x0

Table 27. Pulldown Control 1 Register (PULL_DWN_CTRL_1) (Continued)

Base Address = 0xFFFE 1000, Offset Address = 0x44				
Bit	**Name**	**Function**	**R/W**	**Reset**
9	CONF_PDEN_M14	Enables (0) pullup or pulldown on M14.	R/W	0x0
8	CONF_PDEN_P18	Enables (0) pullup or pulldown on P18.	R/W	0x0
7	CONF_PDEN_P20	Enables (0) pullup or pulldown on P20.	R/W	0x0
6	CONF_PDEN_P19	Enables (0) pullup or pulldown on P19.	R/W	0x0
5	CONF_PDEN_M15	Enables (0) pullup or pulldown on M15.	R/W	0x0
4	CONF_PDEN_N20	Enables (0) pullup or pulldown on N20.	R/W	0x0
3	CONF_PDEN_N18	Enables (0) pullup or pulldown on N18.	R/W	0x0
2	CONF_PDEN_N19	Enables (0) pullup or pulldown on N19.	R/W	0x0
1	CONF_PDEN_N21	Enables (0) pullup or pulldown on N21.	R/W	0x0
0	CONF_PDEN_M20	Enables (0) pullup or pulldown on M20.	R/W	0x0

This register controls the enable or disable of the combined pullup/pulldown cell (0 = enabled, 1 = disabled). When enabling the cell, the user must set the corresponding bit in PU_PD_SEL_1 to select either a pullup or a pulldown. The pinout documentation in SPRS231 must be consulted to determine whether a pullup or pulldown exists on the specified I/O. COMP_MODE_CTRL_0 must be programmed to 0xEAEF, so that the programming is taken into account in the corresponding tactical cell.

Table 28. Pulldown Control 2 Register (PULL_DWN_CTRL_2)

Base Address = 0xFFFE 1000, Offset Address = 0x48				
Bit	**Name**	**Function**	**R/W**	**Reset**
31	CONF_PDEN_AA5	Enables (0) pullup or pulldown on AA5.	R/W	0x0
30	CONF_PDEN_W6	Enables (0) pullup or pulldown on W6.	R/W	0x0
29	CONF_PDEN_Y6	Enables (0) pullup or pulldown on Y6.	R/W	0x0
28	CONF_PDEN_V7	Enables (0) pullup or pulldown on V7.	R/W	0x0
27	CONF_PDEN_W7	Enables (0) pullup or pulldown on W7.	R/W	0x0
26	CONF_PDEN_P10	Enables (0) pullup or pulldown on P10.	R/W	0x0
25	CONF_PDEN_V8	Enables (0) pullup or pulldown on V8.	R/W	0x0
24	CONF_PDEN_Y8	Enables (0) pullup or pulldown on Y8.	R/W	0x0
23	CONF_PDEN_W8	Enables (0) pullup or pulldown on W8.	R/W	0x0
22	CONF_PDEN_R10	Enables (0) pullup or pulldown on R10.	R/W	0x0
21	CONF_PDEN_V5	Enables (0) pullup or pulldown on V5.	R/W	0x0
20	CONF_PDEN_V9	Enables (0) pullup or pulldown on V9.	R/W	0x0
19	CONF_PDEN_W9	Enables (0) pullup or pulldown on W9.	R/W	0x0
18	CONF_PDEN_AA9	Enables (0) pullup or pulldown on AA9.	R/W	0x0
17	CONF_PDEN_Y10	Enables (0) pullup or pulldown on Y10.	R/W	0x0
16	CONF_PDEN_R11	Enables (0) pullup or pulldown on R11.	R/W	0x0
15	CONF_PDEN_P11	Enables (0) pullup or pulldown on P11.	R/W	0x0
14	CONF_PDEN_V10	Enables (0) pullup or pulldown on V10.	R/W	0x0
13	CONF_PDEN_V11	Enables (0) pullup or pulldown on V11.	R/W	0x0
12	CONF_PDEN_W10	Enables (0) pullup or pulldown on W10.	R/W	0x0

Table 28. Pulldown Control 2 Register (PULL_DWN_CTRL_2) (Continued)

Base Address = 0xFFFE 1000, Offset Address = 0x48				
Bit	**Name**	**Function**	**R/W**	**Reset**
11	CONF_PDEN_P13	Enables (0) pullup or pulldown on P13.	R/W	0x0
10	CONF_PDEN_R13	Enables (0) pullup or pulldown on R13.	R/W	0x0
9	CONF_PDEN_V15	Enables (0) pullup or pulldown on V15.	R/W	0x0
8	CONF_PDEN_P14	Enables (0) pullup or pulldown on P14.	R/W	0x0
7	CONF_PDEN_AA17	Enables (0) pullup or pulldown on AA17.	R/W	0x0
6	CONF_PDEN_Y15	Enables (0) pullup or pulldown on Y15.	R/W	0x0
5	CONF_PDEN_W15	Enables (0) pullup or pulldown on W15.	R/W	0x0
4	CONF_PDEN_W14	Enables (0) pullup or pulldown on W14.	R/W	0x0
3	CONF_PDEN_Y14	Enables (0) pullup or pulldown on Y14.	R/W	0x0
2	CONF_PDEN_V14	Enables (0) pullup or pulldown on V14.	R/W	0x0
1	CONF_PDEN_R14	Enables (0) pullup or pulldown on R14.	R/W	0x0
0	CONF_PDEN_AA15	Enables (0) pullup or pulldown on AA15.	R/W	0x0

This register controls the enable or disable of the combined pullup/pulldown cell (0 = enabled, 1 = disabled). When enabling the cell, the user must set the corresponding bit in PU_PD_SEL_2 to select either a pullup or a pulldown. The pinout documentation in SPRS231 must be consulted to determine whether a pullup or pulldown exists on the specified I/O. COMP_MODE_CTRL_0 must be programmed to 0xEAEF, so that the programming is taken into account in the corresponding tactical cell.

Table 29. Pulldown Control 3 Register (PULL_DWN_CTRL_3)

Base Address = 0xFFFE 1000, Offset Address = 0x4C				
Bit	**Name**	**Function**	**R/W**	**Reset**
31	CONF_PDEN_W2	Enables (0) pullup or pulldown on W2.	R/W	0X0
30	CONF_PDEN_V4	Enables (0) pullup or pulldown on V4.	R/W	0X0
29	CONF_PDEN_Y1	Enables (0) pullup or pulldown on Y1.	R/W	0X0
28	CONF_PDEN_Y17	Enables (0) pullup or pulldown on Y17.	R/W	0X0
27	CONF_PDEN_D18	Enables (0) pullup or pulldown on D18.	R/W	0X0
26	CONF_PDEN_B21	Enables (0) pullup or pulldown on B21.	R/W	0X0
25	CONF_PDEN_C19	Enables (0) pullup or pulldown on C19.	R/W	0X0
24	CONF_PDEN_G14	Enables (0) pullup or pulldown on G14.	R/W	0X0
23	CONF_PDEN_H13	Enables (0) pullup or pulldown on H13.	R/W	0X0
22	CONF_PDEN_A20	Enables (0) pullup or pulldown on A20.	R/W	0X0
21	CONF_PDEN_B19	Enables (0) pullup or pulldown on B19.	R/W	0X0
20	CONF_PDEN_C18	Enables (0) pullup or pulldown on C18.	R/W	0X0
19	Reserved	Reserved.	R/W	0X0
18	CONF_PDEN_D17	Enables (0) pullup or pulldown on D17.	R/W	0X0
17	CONF_PDEN_D16	Enables (0) pullup or pulldown on D16.	R/W	0X0
16	CONF_PDEN_C17	Enables (0) pullup or pulldown on C17.	R/W	0X0
15	CONF_PDEN_B17	Enables (0) pullup or pulldown on B17.	R/W	0X0
14	CONF_PDEN_G13	Enables (0) pullup or pulldown on G13.	R/W	0X0
13	CONF_PDEN_A17	Enables (0) pullup or pulldown on A17.	R/W	0X0
12	CONF_PDEN_C16	Enables (0) pullup or pulldown on C16.	R/W	0X0
11	CONF_PDEN_D15	Enables (0) pullup or pulldown on D15.	R/W	0X0
10	Reserved	Reserved.	R/W	0X0
9	Reserved	Reserved.	R/W	0X0
8	CONF_PDEN_W11	Enables (0) pullup or pulldown on W11.	R/W	0x0
7	Reserved	Reserved.	R/W	0x0

Table 29. Pulldown Control 3 Register (PULL_DWN_CTRL_3) (Continued)

Base Address = 0xFFFE 1000, Offset Address = 0x4C				
Bit	**Name**	**Function**	**R/W**	**Reset**
6	CONF_PDEN_M4	Enables (0) pullup or pulldown on M4.	R/W	0x0
5	CONF_PDEN_W4	Enables (0) pullup or pulldown on W4.	R/W	0x0
4	CONF_PDEN_Y4	Enables (0) pullup or pulldown on Y4.	R/W	0x0
3	CONF_PDEN_V6	Enables (0) pullup or pulldown on V6.	R/W	0x0
2	CONF_PDEN_W5	Enables (0) pullup or pulldown on W5.	R/W	0x0
1	CONF_PDEN_Y5	Enables (0) pullup or pulldown on Y5.	R/W	0x0
0	CONF_PDEN_R9	Enables (0) pullup or pulldown on R9.	R/W	0x0

This register controls the enable or disable of the combined pullup/pulldown cell (0 = enabled, 1 = disabled). When enabling the cell, it is assumed that the user has cleared or set the corresponding bit in PU_PD_SEL_3 to select a pullup or a pulldown. The pinout section in the Application Processor Data Manual (SPRS231) should be consulted to determine whether a pullup or pulldown exists on the specified I/O. COMP_MODE_CTRL_0 must be programmed to 0xEAEF, so that the programming is taken into account in the corresponding tactical cell.

Table 30. Gate and Inhibit Control Register (GATE_INH_CTRL_0)

Base Address = 0xFFFE 1000, Offset Address = 0x50				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:6	CONF_GATE_INH_ RESERVED	Reserved for future expansion.	R/W	0x0000000
5	RESERVED1	Reserved for future expansion.	R/W	0x0
4	RESERVED	Reserved for future expansion.	R/W	0x0
3	CONF_HIGH_IMP3	Controls high impedance on MCSI1.DOUT. 0: Normal function 1: Hi-Z	R/W	0x0
2	CONF_SOFTWARE_PWR _R	Controls software gating and inhibiting of the I/O, which is gated or inhibited by COM_PWR status. If the gating and inhibiting logic is enabled by FUNC_MUX_CTRL_0 (10 – 13 bits) and CONF_SOFTWARE_GATE_ENA_R is set to 1, this bit controls the com_pwr gating and inhibiting, instead of device pins.	R/W	0x0
1	CONF_SOFTWARE_BVLZ _R	This bit controls software gating and inhibiting of the I/O, which is gated or inhibited by the BFAIL/$\overline{\text{EXT\_FIQ}}$ signal. If the gating and inhibiting logic is enabled by FUNC_MUX_CTRL_0(10 – 13)bits and CONF_SOFTWARE_GATE_ENA_R is set to 1, this bit controls the BFAIL/$\overline{\text{EXT\_FIQ}}$ gating and inhibiting, instead of device pins.	R/W	0x0
0	CONF_SOFTWARE_GATE _ENA_R	This bit controls software gating of the I/O, which is gated or inhibited. If the gating and inhibiting logic is enabled by FUNC_MUX_CTRL_0(10 – 13) bits, the software is enabled to control the gating and inhibiting, instead of device pins.	R/W	0x0

This register controls the software gating and inhibiting functionality for the device. The pinout section of the Application Processor Data Manual (SPRS231) should be consulted to understand which pins are affected by the Gated1, Gated2, Inhibit1, and Inhibit2 functions of the device.

Table 31. Configuration Revision Register (CONF_REV)

Base Address = 0xFFFE 1000, Offset Address = 0x58				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:8	CONF_REV_RESERVED	Reserved for future expansion.	R	0x000000
7:0	CONF_REV_R	This 8-bit field indicates the revision number of the current module. This value is fixed by hardware. The 4-bit LSBs indicate a minor revision. The 4-bit MSBs indicate a major revision. Example: 0x10 => Version 1.0 Reset has no effect on the value returned.	R	0x10

This is a read-only register that contains the revision number of the module. A write to this register has no effect.

Table 32. Voltage Control 0 Register (VOLTAGE_CTRL_0)

Base Address = 0xFFFE 1000, Offset Address = 0x60				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:14	CONF_VOLTAGE_ RESERVED	Reserved for future expansion.	R/W	0x00000
13	RESERVED	This bit should be set as 0.	R/W	0X0
12	CONF_VOLTAGE_RTC_R	This bit controls the drive strength of the DVDD10 RTC voltage domain. The bit controls the low/high voltage mode. The voltage range supported is defined in the SPRS231 document. 0: Low-voltage mode drive strength of nominal (nominal 1.80-V range). 1: Interface in high-voltage mode drive strength (nominal 2.75-V range).	R/W	0x0

† For EMIFS and EMIFF 3.0-V and 3.3-V operation, use the 2.75V setting.
Note: For a description of 1.8-V and 2.75-V nominal voltage ranges, see the Data Manual (SPRS231).

Table 32. Voltage Control 0 Register (VOLTAGE_CTRL_0) (Continued)

Base Address = 0xFFFE 1000, Offset Address = 0x60				
Bit	**Name**	**Function**	**R/W**	**Reset**
11	CONF_VOLTAGE_ VDDSHV9_R	This bit controls the drive strength of the DVDD9 voltage domain. The bit controls the low/high voltage mode. The voltage range supported is defined in the SPRS231 document. 0: Low-voltage mode drive strength of nominal (nominal 1.80-V range). 1: Interface in high-voltage mode drive strength (nominal 2.75-V range).	R/W	0x0
10	CONF_VOLTAGE_ VDDSHV8_R	This bit controls the drive strength of the DVDD8 voltage domain. The bit controls the low/high voltage mode. The voltage range supported is defined in the SPRS231 document. 0: Low-voltage mode drive strength of nominal (nominal 1.80-V range). 1: Interface in high-voltage mode drive strength (nominal 2.75-V range).	R/W	0x0
9	CONF_VOLTAGE_ VDDHSV7_R	This bit controls the drive strength of the DVDD7 voltage domain. The bit controls the low/high voltage mode. The voltage range supported is defined in the SPRS231 document. 0: Low-voltage mode drive strength of nominal (nominal 1.80-V range). 1: Interface in high-voltage mode drive strength (nominal 2.75-V range).	R/W	0x0
8	CONF_VOLTAGE_ VDDHSV6_R	This bit controls the drive strength of the DVDD6 voltage domain. The bit controls the low/high voltage mode. The voltage range supported is defined in the SPRS231 document. 0: Low-voltage mode drive strength of nominal (nominal 1.80-V range). 1: Interface in high-voltage mode drive strength (nominal 2.75-V range).	R/W	0x0

† For EMIFS and EMIFF 3.0-V and 3.3-V operation, use the 2.75V setting.
Note: For a description of 1.8-V and 2.75-V nominal voltage ranges, see the Data Manual (SPRS231).

Table 32. Voltage Control 0 Register (VOLTAGE_CTRL_0) (Continued)

Base Address = 0xFFFE 1000, Offset Address = 0x60				
Bit	**Name**	**Function**	**R/W**	**Reset**
7	CONF_VOLTAGE_ VDDSHV2_R	This bit controls the drive strength of the DVDD2 voltage domain. The bit controls the low/high voltage mode. The voltage range supported is defined in the SPRS231 document. 0: Low-voltage mode drive strength of nominal (nominal 1.80-V range). 1: Interface in high-voltage mode drive strength (nominal 2.75-V range).	R/W	0x0
6	CONF_VOLTAGE_ VDDSHV1_R	This bit controls the drive strength of the DVDD1 voltage domain. The bit controls the low/high voltage mode. The voltage range supported is defined in the SPRS231 document. 0: Low-voltage mode drive strength of nominal (nominal 1.80-V range). 1: Interface in high-voltage mode drive strength (nominal 2.75-V range).	R/W	0x0
5	RESERVED1	Reserved for potential low-power operation control	R/W	0x0
4	RESERVED2	Reserved for potential low-power operation control	R/W	0x0
3	RESERVED	Reserved for potential low-power operation control	R/W	0x0

† For EMIFS and EMIFF 3.0-V and 3.3-V operation, use the 2.75V setting.
Note: For a description of 1.8-V and 2.75-V nominal voltage ranges, see the Data Manual (SPRS231).

Table 32. Voltage Control 0 Register (VOLTAGE_CTRL_0) (Continued)

Base Address = 0xFFFE 1000, Offset Address = 0x60				
Bit	**Name**	**Function**	**R/W**	**Reset**
2	CONF_VOLTAGE_ COMIF_R	This bit controls the drive strength of the DVDD3 communication processor interface voltage domain. The bit controls the low/high voltage mode. The voltage range is defined in the SPRS231 document. 0: Low-voltage mode drive strength of nominal (nominal 1.80-V range). 1: Interface in high-voltage mode drive strength (nominal 2.75-V range).	R/W	0x0
1	CONF_VOLTAGE_ SDRAM_R	This bit controls the drive strength of the DVDD4 SDRAM interface voltage domain. The bit controls the low/high voltage mode. The voltage range supported is defined in the SPRS231 document. 0: Low-voltage mode drive strength of nominal (nominal 1.80-V range). 1: Interface in high-voltage mode drive strength (nominal 2.75-V range)†.	R/W	0x0
0	CONF_VOLTAGE_ FLASH_R	This bit controls the drive strength of the DVDD5 flash interface voltage domain. The bit controls the low/high voltage mode. The voltage range supported is defined in the SPRS231 document. 0: Low-voltage mode drive strength of nominal (nominal 1.80-V range). 1: Interface in high-voltage mode drive strength (nominal 2.75-V range)†.	R/W	0x0

† For EMIFS and EMIFF 3.0-V and 3.3-V operation, use the 2.75V setting.
Note: For a description of 1.8-V and 2.75-V nominal voltage ranges, see the Data Manual (SPRS231).

This register controls the low- or high-voltage mode for each I/O power supply domain. Low-voltage mode is defined as 1.8-V range, and high-voltage mode is defined as 2.75-V range. The exact I/O voltage ranges supported by the interfaces of the device are specified in the Data Manual (SPRS231). After reset, the device interfaces are configured in low-voltage mode, and the software must adjust the programming of this register to match the actual voltages applied to each I/O power supply domain. The effect of applying 2.75 V while the interface is in low-voltage mode, for example, is simply that it is faster and more powerful than is usually required.

Table 33. USB Transceiver Control Register (USB_TRANSCEIVER_CTRL)

Base Address = 0xFFFE 1000, Offset Address = 0x64				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:9	UNUSED	These bits are not implemented.	R	0x000000
8	CONF_USB2_UNI_R	This bit configures the way USB port 2 interfaces with external USB transceiver. 0: Bidirectional mode USB2.SE0 is a bidirectional I/O, rather than an output only. Input of the USB2.SE0 bidirectional I/O is connected to usb2_vm port of the USB OTG module. USB2.TXD is a bidirectional I/O, rather than an output only. Input of the USB2.TXD bidirectional I/O is connected to USB2.VP port of the USB OTG module. 1: Unidirectional mode	R/W	0x0
7	CONF_USB1_UNI_R	This bit configures the way USB port 1 interfaces with external USB transceiver. 0: Bidirectional mode USB1.SE0 is a bidirectional I/O, rather than an output only. Input of the USB1.SE0 bidirectional I/O is connected to USB1.VM port of the USB OTG module. USB1.TXD is a bidirectional I/O, rather than an output only. Input of the USB1.TXD bidirectional I/O is connected to USB1.VP port of the USB OTG module. 1: Unidirectional mode	R/W	0x0

Table 33. USB Transceiver Control Register (USB_TRANSCEIVER_CTRL) (Continued)

Base Address = 0xFFFE 1000, Offset Address = 0x64				
Bit	**Name**	**Function**	**R/W**	**Reset**
6:4	CONF_USB_PORT0_R	These bits control the multiplexing on the I/O, which defaults to USB.DP and USB.DM at reset. These 3 bits configure USB port 0 for alternate operation. COMP_MODE_CTRL_0 must be programmed to 0xEAEF for this register to control functional multiplexing. 000: USB.DP/USB.DM 100: I2C.SDA/I2C.SCl 101: UART1.RX/UART1.TX 111: USB2.PUEN/HI-Z Other values are not allowed. Programming this field on some other value produces unpredictable results.	R/W	0x0
3	CONF_USB0_ISOLATE_R	Isolates the USB port 0 controller from the integrated USB transceiver. 0: Normal mode 1: Isolation mode	R/W	0x0
2	CONF_USB_PWRDN_DM_R	Enable/disable of pulldown on USB DM pin. 0: Enable of pulldown on DM 1: Disable of pulldown on DM	R/W	0x1
1	CONF_USB_PWRDN_DP_R	Enable/disable of pulldown on USB DP pin. 0: Enable of pulldown on DP 1: Disable of pulldown on DP	R/W	0x1
0	RESERVED	Reserved for future expansion.	R/W	0x0

Table 34. LDO Powerdown Control Register (LDO_PWRDN_CNTRL)

Base Address = 0xFFFE 1000, Offset Address = 0x68				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:1	UNUSED	These bits are not implemented.	R	0x00000000
0	CONF_LDO_PWRDN_CNTRL_R	If the user sets LDO_PWRDN to 1, the LDO can be powered-down and bypassed.	R/W	0x0

This 1-bit register bypasses the LDO.

Table 35. Test Debug Control 0 Register (TEST_DBG_CTRL_0)

Base Address = 0xFFFE 1000, Offset Address = 0x70				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:22	CONF_TEST_DBG_ RESERVED	Reserved for future expansion.	R/W	0x0000
21	CONF_RNG_TEST_OSC	Ring oscillator divider enable.	R/W	0x0
20	CONF_RNG_SELECT_ OSC	Ring oscillator selection for characterization Active high.	R/W	0x0
19	CONF_DSP_BRTE_ WRITE_R	DFT WRITE signal to control DSP BRTE memories.	R/W	0x1
18	CONF_DSP_BRTE_ READ_R	DFT READ signal to control DSP BRTE memories.	R/W	0x1
17:8	CONF_TDBG_RESERVED _R	These are reserved test and debug bits for the device. They must be set to 0 at all times to avoid errant behavior.	R/W	0x000
7:5	CONF_TEST_DBG_ RESERVED1	Reserved for future expansion.	R/W	0x0
4	CONF_TEST_VBUS_CELL	This bit tests the UIS480 VBUS detect cell. When set to 1, the output of the VBUS detect logic outputs a 0.	R/W	0x0
3	CONF_DPLL_EXT_SEL	This bit selects between the internal 48-MHz clock generated by the APLL and an external 48-MHz clock source. When this bit is set to 1, GPIO14 becomes the source of 48 MHz for the device.	R/W	0x0
2	CONF_VBOX_EN	OMAP VBOX SW test enable. This register must be set to 1 to enable writing to the bit fields CONF_VBOX1 CONF_VBOX2, CONF_DSP_BRTE_READ_R, and CONF_DSP_BRTE_WRITE_R.	R/W	0x0
1	CONF_VBOX2	OMAP VBOX – DFT WRITE signal to control MPU BRTE memories.	R/W	0x1
0	CONF_VBOX1	OMAP VBOX 1 – DFT READ signal to control MPU BRTE memories.	R/W	0x1

Table 36. Module Configuration Control 0 Register (MOD_CONF_CTRL_0)

Base Address = 0xFFFE 1000, Offset Address = 0x80				
Bit	**Name**	**Function**	**R/W**	**Reset**
31	CONF_MOD_UART3_CLK_MODE_R	This bit determines the 48-MHz clock request for UART3. 0: 48-MHz clock request is inactive. 1: 48-MHz clock request is active.	R/W	0x0
30	CONF_MOD_UART2_CLK_MODE_R	This bit determines the clock source of UART2. 0: UART_MCLKO from ULPD that can be either the system clock or 32 kHz, depending on system state and ULPD register programming. 1: 48 MHz.	R/W	0x0
29	CONF_MOD_UART1_CLK_MODE_R	This bit determines whether 48-MHz clock request for UART1 is active. 0: 48-MHz clock request is inactive. 1: 48-MHz clock request is active.	R/W	0x0
28	RESERVED	See Note 1.	R/W	0x0
27:24	MOD_32KOSC_SW_R	See Note 1.	R/W	0x0
23	CONF_MOD_MMC_SD_CLK_REQ_R	This is the functional 48-MHz clock request for the MMCSDIO1 interface. This bit resets to 0. This corresponds to the MMCSDIO1 clock *not* being requested. The user must set the bit to 1 to request the clock for the MMCSDIO1 interface.	R/W	0x0
22	CONF_MOD_DPRAM_ENABLE_R	See Note 1.	R/W	0x0
21	CONF_MOD_MSMMC_VSS_HIZ_OVERRIDE	See Note 1.	R/W	0x0
20	CONF_MOD_MMC_SD2_CLK_REQ_R	This is the functional 48-MHz clock request for the device MMC/SD 2 interface. This bit resets to 0. This corresponds to the MMC/SD clock *not* being requested. The user must set the bit to 1 to request the clock for the MMC/SD interface.	R/W	0x0

Notes: 1) This function has been removed. Writing a 1 or a 0 to this register is acceptable. However, it is recommended to write a 0, in case functions are added to this register space in the future.

Table 36. Module Configuration Control 0 Register (MOD_CONF_CTRL_0) (Continued)

Base Address = 0xFFFE 1000, Offset Address = 0x80				
Bit	**Name**	**Function**	**R/W**	**Reset**
19	CONF_MOD_MCBSP3_ CLK_SEL_R	This bit determines the method of frame sync wrap-around used on MCBSP3. 0: Wrap-around done in hardware external to the McBSP (3 pins mode). 1: Wrap-around disabled. Wrap-around can be performed within the McBSP module (4 pins mode). This bit also selects the Interface clock used for the McBSP3 module: 0: DSPXOR_CLK is selected. 1: DSPPER_CLK is selected.	R/W	0x0
18	CONF_MOD_MCBSP1_ CLKS_SEL_R	This register selects the clock used for the McBSP1 clocks input: 0: McBSP1.CLKS pin of the device. 1: OMAP DPLL1 clockout.	R/W	0x0
17	CONF_MOD_USB_W2FC_ VBUS_MODE_R	This bit determines the method used for USB VBUS detection. 0: The VBUS detection is controlled by GPIO_0 state (if in muxed mode 2). 1: The VBUS detection is controlled by DVDD2 detection on the line.	R/W	0x0
16	CONF_MOD_I2C_SELECT _R	See Note 1.	R/W	0x0
15	CONF_MOD_UART3_ IRDA_MODE_R	See Note 1.	R/W	0x0
14	CONF_MOD_MCBSP1_ CLK_SEL_R	This register selects the clock used for the McBSP1 module: 0: DSPXOR_CLK is selected. 1: DSPPER_CLK is selected.	R/W	0x0

Notes: 1) This function has been removed. Writing a 1 or a 0 to this register is acceptable. However, it is recommended to write a 0, in case functions are added to this register space in the future.

Table 36. Module Configuration Control 0 Register (MOD_CONF_CTRL_0) (Continued)

Base Address = 0xFFFE 1000, Offset Address = 0x80				
Bit	**Name**	**Function**	**R/W**	**Reset**
13	CONF_MOD_MMC2_CLK_SEL_R	This register selects the clock used for the MMC/SD2 module: 0: 48-MHz MMC2_DPLL_CLK from the ULPD. 1: ARM_XOR_CLK from OMAP clock module	R/W	0x0
12	CONF_MOD_COM_MCLK_12_48_SEL_R	This bit determines whether MCLK of the device outputs the system clock or the 48-MHz clock (selection must be in accordance in ULPD and IO_CTRL). 0: MCLK is system clock. 1: MCLK is 48-MHz clock (or divided down).	R/W	0x0
11	CONF_MOD_USB_HOST_UART_SELECT_R	0: Disables UART multiplexing on USB port 1. 1: Enables UART multiplexing on USB port 1.	R/W	0x0
10	CONF_MOD_USB_HOST_LB_ARB_EN_R	Not currently used.	R/W	0x0
9	CONF_MOD_USB_HOST_HHC_UHOST_EN_R	Enable input for functional-mode clocking of USB_HHC. 0: Internal functional mode 48-MHz and 12-MHz clocks are disabled. USB_HHC cannot function as a USB host. 1: Internal functional mode 48-MHz and 12-MHz clocks are enabled.	R/W	0x0

Notes: 1) This function has been removed. Writing a 1 or a 0 to this register is acceptable. However, it is recommended to write a 0, in case functions are added to this register space in the future.

Table 36. Module Configuration Control 0 Register (MOD_CONF_CTRL_0) (Continued)

Base Address = 0xFFFE 1000, Offset Address = 0x80				
Bit	**Name**	**Function**	**R/W**	**Reset**
8	CONF_MOD_USB_HOST_ HMC_TLL_SPEED_R	Transceiverless link logic (TLL) USB speed control. For HMC modes (as defined by HMC_MODE_I and HMC_JTAG_EN_I) where the TLL is used, this bit determines whether the modeling of the device pullup resistor is on the internal D+ or the internal D− signal. The pullup is modeled only when HMC_TLL_ATTACH_I is active. This signal is ignored when either device is driving USB data, and whenever HMC_MODE or HMC_JTAG_EN_I specify that the TLL is not being used. 0: When HMC_TLL_ATTACH_I is high and the TLL is enabled and neither the USB host nor the external USB device attempts to drive, the pullup is modeled on the D− signal to indicate a low-speed device. 1: When HMC_TLL_ATTACH_I is high and the TLL is enabled and neither the USB host nor the external USB device attempts to drive, the pullup is modeled on the D− signal to indicate a full-speed device.	R/W	0x0

Notes: 1) This function has been removed. Writing a 1 or a 0 to this register is acceptable. However, it is recommended to write a 0, in case functions are added to this register space in the future.

Table 36. Module Configuration Control 0 Register (MOD_CONF_CTRL_0) (Continued)

Base Address = 0xFFFE 1000, Offset Address = 0x80				
Bit	**Name**	**Function**	**R/W**	**Reset**
7	CONF_MOD_USB_HOST_ HMC_TLL_ATTACH_R	Transceiverless link logic (TLL) USB attach control. For HMC modes (as defined by HMC_MODE_I and HMC_JTAG_EN_I) where the TLL is used, this bit determines whether or not the TLL models its internal representation of USB differential data signals with or without a pullup when neither the internal USB host nor the external USB device is attempting to drive the signals. This signal is ignored when either device is driving USB data. 0: When neither the USB host nor the external USB device attempts to drive, no pullup is modeled. The associated USB host port interprets this as no device attached. 1: When neither the USB host nor the external USB device attempts to drive, a pullup is modeled on either the internal representation of D+ or D−. The associated USB host port interprets this as an attached device with the bus in an idle condition.	R/W	0x0
6:1	CONF_MOD_USB_HOST_ HMC_MODE_R	USB_HHC port multiplexing control. This resets to the following configuration. 000000b: USB port 0 is controlled by the USB_W2FC, and USB ports 1 and 2 are held in benign states.	R/W	0x0
0	CONF_MOD_USB_HOST_ EXTCLKENI_R	See Note 1.	R/W	0x0

Notes: 1) This function has been removed. Writing a 1 or a 0 to this register is acceptable. However, it is recommended to write a 0, in case functions are added to this register space in the future.

Table 37. Functional Multiplexing Control E Register (FUNC_MUX_CTRL_E)

Base Address = 0xFFFE 1000, Offset Address = 0x90				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:30	RESERVED	Reserved for future expansion.	R/W	0x0
29:27	CONF_G14	Controls multiplexing on G14. Formerly CONF_LCD_PIXEL_3_R.	R/W	0x0
26:24	CONF_H13	Controls multiplexing on H13. Formerly CONF_LCD_PIXEL_4_R.	R/W	0x0
23:21	CONF_A20	Controls multiplexing on A20. Formerly CONF_LCD_PIXEL_5_R.	R/W	0x0
20:18	CONF_B19	Controls multiplexing on B19. Formerly CONF_LCD_PIXEL_6_R.	R/W	0x0
17:15	CONF_C18	Controls multiplexing on C18. Formerly CONF_LCD_PIXEL_7_R.	R/W	0x0
14:12	CONF_D17	Controls multiplexing on D17. Formerly CONF_LCD_PIXEL_8_R.	R/W	0x0
11:9	CONF_B18	Controls multiplexing on B18. Formerly CONF_LCD_VSYNC_R.	R/W	0X0
8:6	CONF_D16	Controls multiplexing on D16. Formerly CONF_LCD_PIXEL_9_R.	R/W	0x0
5:3	CONF_C17	Controls multiplexing on C17. Formerly CONF_LCD_ PIXEL_10_R.	R/W	0X0
2:0	CONF_B17	Controls multiplexing on B17. Formerly CONF_LCD_ PIXEL_11_R.	R/W	0X0

This register controls functional multiplexing. COMP_MODE_CTRL_0 must be programmed to 0xEAEF for this register to control functional multiplexing. See Table 8 for bit-field values.

Table 38. Functional Multiplexing Control F Register (FUNC_MUX_CTRL_F)

Base Address = 0xFFFE 1000, Offset Address = 0x94				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:30	RESERVED	Reserved for future expansion.	R/W	0X0
29:27	CONF_V2	Controls multiplexing on V2. Formerly CONF_FRDY_R.	R/W	0x0
26:24	CONF_U4	Controls multiplexing on U4. Formerly CONF_NFOE_R.	R/W	0X0
23:21	CONF_W1	Controls multiplexing on W1. Formerly CONF_NFRP_R.	R/W	0X0
20:18	CONF_W2	Controls multiplexing on W2. Formerly CONF_NFWE_R.	R/W	0X0
17:15	CONF_V4	Controls multiplexing on V4. Formerly CONF_NFWP_R.	R/W	0X0
14:12	CONF_Y1	Controls multiplexing on Y1. Formerly CONF_NFCS_1B_R.	R/W	0X0
11:9	CONF_Y17	Controls multiplexing on Y17. Formerly CONF_RTCK_R.	R/W	0x0
8:6	CONF_D18	Controls multiplexing on D18. Formerly CONF_LCD_PIXEL_0_R.	R/W	0x0
5:3	CONF_B21	Controls multiplexing on B21. Formerly CONF_LCD_PIXEL_1_R.	R/W	0x0
2:0	CONF_C19	Controls multiplexing on C19. Formerly CONF_LCD_PIXEL_2_R.	R/W	0x0

This register controls functional multiplexing. COMP_MODE_CTRL_0 must be programmed to 0xEAEF for this register to control functional multiplexing. See Table 8 for bit-field values.

Table 39. Functional Multiplexing Control 10 Register (FUNC_MUX_CTRL_10)

Base Address = 0xFFFE 1000, Offset Address = 0x98				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:30	RESERVED	Reserved for future expansion.	R/W	0X0
29:27	CONF_M3	Controls multiplexing on M3. Formerly CONF_NFCS_1_R.	R/W	0X0
26:24	CONF_N8	Controls multiplexing on N8. Formerly CONF_NFCS_3_R.	R/W	0X0
23:21	CONF_N3	Controls multiplexing on N3. Formerly CONF_FCLK_R.	R/W	0X0
20:18	CONF_P3	Controls multiplexing on P3. Formerly CONF_NFCS_2B_R.	R/W	0X0
17:15	CONF_W11	Controls multiplexing on W11. Formerly CONF_MMC_DAT3_R.	R/W	0X0
14:12	RESERVED 1	Reserved for future expansion.	R/W	0X0

Table 39. Functional Multiplexing Control 10 Register (FUNC_MUX_CTRL_10)

Base Address = 0xFFFE 1000, Offset Address = 0x98				
Bit	**Name**	**Function**	**R/W**	**Reset**
11:9	CONF_L4	Controls multiplexing on L4. Formerly CONF_NFADV_R.	R/W	0X0
8:6	CONF_L3	Controls multiplexing on L3. Formerly CONF_NFBE_0_R.	R/W	0x0
5:3	CONF_M8	Controls multiplexing on M8. Formerly CONF_NFBE_1_R.	R/W	0x0
2:0	CONF_M7	Controls multiplexing on M7. Formerly CONF_GPIO_1_R.	R/W	0x0

This register controls functional multiplexing. COMP_MODE_CTRL_0 must be programmed to 0xEAEF for this register to control functional multiplexing. See Table 8 for bit-field values.

Table 40. Functional Multiplexing Control 11 Register (FUNC_MUX_CTRL_11)

Base Address = 0xFFFE 1000, Offset Address = 0x9C				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:30	RESERVED	Reserved for future expansion.	R/W	0X0
29:27	RESERVED	Reserved for future expansion.	R/W	0X0
26:24	CONF_F3	Controls multiplexing on F3. Formerly CONF_FADD_9_R.	R/W	0X0
23:21	CONF_G4	Controls multiplexing on G4. Formerly CONF_FADD_10_R.	R/W	0X0
20:18	CONF_G3	Controls multiplexing on G3. Formerly CONF_FADD_11_R.	R/W	0X0
17:15	CONF_G2	Controls multiplexing on G2. Formerly CONF_FADD_12_R.	R/W	0X0
14:12	CONF_K8	Controls multiplexing on K8. Formerly CONF_FADD_13_R.	R/W	0X0
11:9	CONF_H4	Controls multiplexing on H4. Formerly CONF_FADD_14_R.	R/W	0X0
8:6	CONF_H3	Controls multiplexing on H3. Formerly CONF_FADD_15_R.	R/W	0X0
5:3	CONF_K7	Controls multiplexing on K7. Formerly CONF_FADD_16_R.	R/W	0X0
2:0	RESERVED	Reserved for future expansion.	R/W	0X0

This register controls functional multiplexing. COMP_MODE_CTRL_0 must be programmed to 0xEAEF for this register to control functional multiplexing. See Table 8 for bit−field values.

Table 41. Functional Multiplexing Control 12 Register (FUNC_MUX_CTRL_12)

Base Address = 0xFFFE 1000, Offset Address = 0xA0				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:30	RESERVED	Reserved for future expansion.	R/W	0X0
29:27	RESERVED	Reserved for future expansion.	R/W	0x0
26:24	CONF_J8	Controls multiplexing on J8. Formerly CONF_FADD_1_R.	R/W	0X0
23:21	CONF_D3	Controls multiplexing on D3. Formerly CONF_FADD_2_R.	R/W	0X0
20:18	CONF_C1	Controls multiplexing on C1. Formerly CONF_FADD_3_R.	R/W	0X0
17:15	CONF_E4	Controls multiplexing on E4. Formerly CONF_FADD_4_R.	R/W	0X0
14:12	CONF_D2	Controls multiplexing on D2. Formerly CONF_FADD_5_R.	R/W	0X0
11:9	CONF_F4	Controls multiplexing on F4. Formerly CONF_FADD_6_R.	R/W	0X0
8:6	CONF_E3	Controls multiplexing on E3. Formerly CONF_FADD_7_R.	R/W	0X0
5:3	CONF_J7	Controls multiplexing on J7. Formerly CONF_FADD_8_R.	R/W	0X0
2:0	RESERVED	Reserved for future expansion.	R/W	0X0

This register controls functional multiplexing. COMP_MODE_CTRL_0 must be programmed to 0xEAEF for this register to control functional multiplexing. See Table 8 for bit-field values.

Table 42. Pulldown Control 4 Register (PULL_DWN_CTRL_4)

Base Address = 0xFFFE 1000, Offset Address = 0xAC				
Bit	**Name**	**Function**	**R/W**	**Reset**
31	CONF_PDEN_Y18	Enables (0) pullup or pulldown on Y18.	R/W	0X0
30	CONF_PDEN_W18	Enables (0) pullup or pulldown on W18.	R/W	0X0
29	CONF_PDEN_V17	Enables (0) pullup or pulldown on V17.	R/W	0X0
28	CONF_PDEN_Y19	Enables (0) pullup or pulldown on Y19.	R/W	0X0
27	CONF_PDEN_V18	Enables (0) pullup or pulldown on V18.	R/W	0X0
26	CONF_PDEN_L3	Enables (0) pullup or pulldown on L3.	R/W	0x0
25	CONF_PDEN_M8	Enables (0) pullup or pulldown on M8.	R/W	0x0
24	CONF_PDEN_M7	Enables (0) pullup or pulldown on M7.	R/W	0x0

Table 42. Pulldown Control 4 Register (PULL_DWN_CTRL_4) (Continued)

Base Address = 0xFFFE 1000, Offset Address = 0xAC				
Bit	**Name**	**Function**	**R/W**	**Reset**
23	CONF_PDEN_M3	Enables (0) pullup or pulldown on M3.	R/W	0x0
22	CONF_PDEN_N8	Enables (0) pullup or pulldown on N8.	R/W	0x0
21	CONF_PDEN_N3	Enables (0) pullup or pulldown on N3.	R/W	0X0
20	CONF_PDEN_J8	Enables (0) pullup or pulldown on J8.	R/W	0x0
19	CONF_PDEN_D3	Enables (0) pullup or pulldown on D3.	R/W	0x0
18	CONF_PDEN_C1	Enables (0) pullup or pulldown on C1.	R/W	0x0
17	CONF_PDEN_E4	Enables (0) pullup or pulldown on E4.	R/W	0x0
16	CONF_PDEN_D2	Enables (0) pullup or pulldown on D2.	R/W	0x0
15	CONF_PDEN_F4	Enables (0) pullup or pulldown on F4.	R/W	0x0
14	CONF_PDEN_E3	Enables (0) pullup or pulldown on E3.	R/W	0x0
13	CONF_PDEN_J7	Enables (0) pullup or pulldown on J7.	R/W	0x0
12	CONF_PDEN_F3	Enables (0) pullup or pulldown on F3.	R/W	0x0
11	CONF_PDEN_G4	Enables (0) pullup or pulldown on G4.	R/W	0x0
10	CONF_PDEN_G3	Enables (0) pullup or pulldown on G3.	R/W	0X0
9	CONF_PDEN_G2	Enables (0) pullup or pulldown on G2.	R/W	0X0
8	CONF_PDEN_K8	Enables (0) pullup or pulldown on K8.	R/W	0X0
7	CONF_PDEN_H4	Enables (0) pullup or pulldown on H4.	R/W	0X0
6	CONF_PDEN_H3	Enables (0) pullup or pulldown on H3.	R/W	0X0
5	CONF_PDEN_K7	Enables (0) pullup or pulldown on K7.	R/W	0X0
4	CONF_PDEN_L4	Enables (0) pullup or pulldown on L4.	R/W	0X0
3	CONF_PDEN_V2	Enables (0) pullup or pulldown on V2.	R/W	0X0
2	CONF_PDEN_P3	Enables (0) pullup or pulldown on P3.	R/W	0X0
1	CONF_PDEN_U4	Enables (0) pullup or pulldown on U4.	R/W	0X0
0	CONF_PDEN_W1	Enables (0) pullup or pulldown on W1.	R/W	0X0

This register controls the enable or disable of the combined pullup/pulldown cell (0 = enabled, 1 = disabled). When enabling the cell, the user must set the corresponding bit in PU_PD_SEL_4 to select either a pullup or a pulldown. The pinout documentation (Application Processor Data Manual (SPRS231)) must be consulted to determine whether a pullup or pulldown exists on the specified I/O. COMP_MODE_CTRL_0 must be programmed to 0xEAEF, so that the programming is taken into account in the corresponding tactical cell.

Table 43. Pullup/Pulldown Selection 0 Register (PU_PD_SEL_0)

Base Address = 0xFFFE 1000, Offset Address = 0xB4				
Bit	**Name**	**Function**	**R/W**	**Reset**
31	CONF_PU_PD_L14	Configure pullup (=1) or pulldown (=0) on L14.	R/W	0x0
30	CONF_PU_PD_M18	Configure pullup (=1) or pulldown (=0) on M18.	R/W	0x0
29	CONF_PU_PD_M19	Configure pullup (=1) or pulldown (=0) on M19.	R/W	0x0
28	CONF_PU_PD_L15	Configure pullup (=1) or pulldown (=0) on L15.	R/W	0x0
27	CONF_PU_PD_L18	Configure pullup (=1) or pulldown (=0) on L18.	R/W	0x0
26	CONF_PU_PD_L19	Configure pullup (=1) or pulldown (=0) on L19.	R/W	0x0
25	CONF_PU_PD_K14	Configure pullup (=1) or pulldown (=0) on K14.	R/W	0x0
24	CONF_PU_PD_K15	Configure pullup (=1) or pulldown (=0) on K15.	R/W	0x0
23	CONF_PU_PD_K19	Configure pullup (=1) or pulldown (=0) on K19.	R/W	0x0
22	CONF_PU_PD_K18	Configure pullup (=1) or pulldown (=0) on K18.	R/W	0x0
21	CONF_PU_PD_J14	Configure pullup (=1) or pulldown (=0) on J14.	R/W	0x0
20	CONF_PU_PD_J19	Configure pullup (=1) or pulldown (=0) on J19.	R/W	0x0
19	CONF_PU_PD_J18	Configure pullup (=1) or pulldown (=0) on J18.	R/W	0x0
18	CONF_PU_PD_J15	Configure pullup (=1) or pulldown (=0) on J15.	R/W	0x0
17	CONF_PU_PD_H19	Configure pullup (=1) or pulldown (=0) on H19.	R/W	0x0
16	CONF_PU_PD_H20	Configure pullup (=1) or pulldown (=0) on H20.	R/W	0x0
15	CONF_PU_PD_H18	Configure pullup (=1) or pulldown (=0) on H18.	R/W	0x0
14	CONF_PU_PD_H15	Configure pullup (=1) or pulldown (=0) on H15.	R/W	0x0
13	CONF_PU_PD_G21	Configure pullup (=1) or pulldown (=0) on G21.	R/W	0x0
12	CONF_PU_PD_G20	Configure pullup (=1) or pulldown (=0) on G20.	R/W	0x0

Table 43. Pullup/Pulldown Selection 0 Register (PU_PD_SEL_0) (Continued)

Base Address = 0xFFFE 1000, Offset Address = 0xB4				
Bit	**Name**	**Function**	**R/W**	**Reset**
11	RESERVED	Reserved for future expansion.	R/W	0x0
10	CONF_PU_PD_G18	Configure pullup (=1) or pulldown (=0) on G18.	R/W	0x0
9	CONF_PU_PD_F19	Configure pullup (=1) or pulldown (=0) on F19.	R/W	0x0
8	CONF_PU_PD_H14	Configure pullup (=1) or pulldown (=0) on H14.	R/W	0x0
7	CONF_PU_PD_E20	Configure pullup (=1) or pulldown (=0) on E20.	R/W	0x0
6	CONF_PU_PD_E19	Configure pullup (=1) or pulldown (=0) on E19.	R/W	0x0
5	CONF_PU_PD_F18	Configure pullup (=1) or pulldown (=0) on F18.	R/W	0x0
4	CONF_PU_PD_D20	Configure pullup (=1) or pulldown (=0) on D20.	R/W	0x0
3	CONF_PU_PD_D19	Configure pullup (=1) or pulldown (=0) on D19.	R/W	0x0
2	CONF_PU_PD_E18	Configure pullup (=1) or pulldown (=0) on E18.	R/W	0x0
1	CONF_PU_PD_C21	Configure pullup (=1) or pulldown (=0) on C21.	R/W	0x0
0	CONF_PU_PD_G19	Configure pullup (=1) or pulldown (=0) on G19.	R/W	0x0

This register controls the selection of the pullup or pulldown (0 = pulldown, 1 = pullup). The pinout documentation (SPRS231) must be consulted to determine whether a pullup or pulldown exists on the specified I/O. COMP_MODE_CTRL_0 must be programmed to 0xEAEF for this register to control the selection.

Table 44. Pullup/Pulldown Selection 1 Register (PU_PD_SEL_1)

Base Address = 0xFFFE 1000, Offset Address = 0xB8				
Bit	**Name**	**Function**	**R/W**	**Reset**
31	CONF_PU_PD_AA20	Configure pullup (=1) or pulldown (=0) on AA20.	R/W	0x0
30	CONF_PU_PD_U20	Configure pullup (=1) or pulldown (=0) on U20.	R/W	0x0
29	CONF_PU_PD_W16	Configure pullup (=1) or pulldown (=0) on W16.	R/W	0x0
28	CONF_PU_PD_W13	Configure pullup (=1) or pulldown (=0) on W13.	R/W	0x0
27	CONF_PU_PD_J20	Configure pullup (=1) or pulldown (=0) on J20.	R/W	0x0
26	CONF_PU_PD_W17	Configure pullup (=1) or pulldown (=0) on W17.	R/W	0x0
25	CONF_PU_PD_V16	Configure pullup (=1) or pulldown (=0) on V16.	R/W	0x0

Table 44. Pullup/Pulldown Selection 1 Register (PU_PD_SEL_1) (Continued)

Base Address = 0xFFFE 1000, Offset Address = 0xB8				
Bit	**Name**	**Function**	**R/W**	**Reset**
24	CONF_PU_PD_Y12	Configure pullup (=1) or pulldown (=0) on Y12.	R/W	0x0
23	CONF_PU_PD_W19	Configure pullup (=1) or pulldown (=0) on W19.	R/W	0x0
22	CONF_PU_PD_P15	Configure pullup (=1) or pulldown (=0) on P15.	R/W	0x0
21	CONF_PU_PD_N14	Configure pullup (=1) or pulldown (=0) on N14.	R/W	0x0
20	CONF_PU_PD_V19	Configure pullup (=1) or pulldown (=0) on V19.	R/W	0x0
19	CONF_PU_PD_W21	Configure pullup (=1) or pulldown (=0) on W21.	R/W	0x0
18	CONF_PU_PD_U18	Configure pullup (=1) or pulldown (=0) on U18.	R/W	0x0
17	Reserved	Reserved.	R/W	0x0
16	Reserved	Reserved.	R/W	0x0
15	CONF_PU_PD_U19	Configure pullup (=1) or pulldown (=0) on U19.	R/W	0x0
14	CONF_PU_PD_N15	Configure pullup (=1) or pulldown (=0) on N15.	R/W	0x0
13	CONF_PU_PD_T19	Configure pullup (=1) or pulldown (=0) on T19.	R/W	0x0
12	CONF_PU_PD_T20	Configure pullup (=1) or pulldown (=0) on T20.	R/W	0x0
11	CONF_PU_PD_R18	Configure pullup (=1) or pulldown (=0) on R18.	R/W	0x0
10	CONF_PU_PD_R19	Configure pullup (=1) or pulldown (=0) on R19.	R/W	0x0
9	CONF_PU_PD_M14	Configure pullup (=1) or pulldown (=0) on M14.	R/W	0x0
8	CONF_PU_PD_P18	Configure pullup (=1) or pulldown (=0) on P18.	R/W	0x0
7	CONF_PU_PD_P20	Configure pullup (=1) or pulldown (=0) on P20.	R/W	0x0
6	CONF_PU_PD_P19	Configure pullup (=1) or pulldown (=0) on P19.	R/W	0x0
5	CONF_PU_PD_M15	Configure pullup (=1) or pulldown (=0) on M15.	R/W	0x0
4	CONF_PU_PD_N20	Configure pullup (=1) or pulldown (=0) on N20.	R/W	0x0
3	CONF_PU_PD_N18	Configure pullup (=1) or pulldown (=0) on N18.	R/W	0x0
2	CONF_PU_PD_N19	Configure pullup (=1) or pulldown (=0) on N19.	R/W	0x0
1	CONF_PU_PD_N21	Configure pullup (=1) or pulldown (=0) on N21.	R/W	0x0
0	CONF_PU_PD_M20	Configure pullup (=1) or pulldown (=0) on M20.	R/W	0x0

This register controls the selection of the pullup or pulldown (0 = pulldown, 1 = pullup). The pinout documentation must be consulted to determine whether a pullup or pulldown exists on the specified I/O. COMP_MODE_CTRL_0 must be programmed to 0xEAEF for this register to control the selection.

Table 45. Pullup/Pulldown Selection 2 Register (PU_PD_SEL_2)

Base Address = 0xFFFE 1000, Offset Address = 0xBC				
Bit	**Name**	**Function**	**R/W**	**Reset**
31	CONF_PU_PD_AA5	Configure pullup (=1) or pulldown (=0) on AA5.	R/W	0x0
30	CONF_PU_PD_W6	Configure pullup (=1) or pulldown (=0) on W6.	R/W	0x0
29	CONF_PU_PD_Y6	Configure pullup (=1) or pulldown (=0) on Y6.	R/W	0x0
28	CONF_PU_PD_V7	Configure pullup (=1) or pulldown (=0) on V7.	R/W	0x0
27	CONF_PU_PD_W7	Configure pullup (=1) or pulldown (=0) on W7.	R/W	0x0
26	CONF_PU_PD_P10	Configure pullup (=1) or pulldown (=0) on P10.	R/W	0x0
25	CONF_PU_PD_V8	Configure pullup (=1) or pulldown (=0) on V8.	R/W	0x0
24	CONF_PU_PD_Y8	Configure pullup (=1) or pulldown (=0) on Y8.	R/W	0x0
23	CONF_PU_PD_W8	Configure pullup (=1) or pulldown (=0) on W8.	R/W	0x0
22	CONF_PU_PD_R10	Configure pullup (=1) or pulldown (=0) on R10.	R/W	0x0
21	CONF_PU_PD_V5	Configure pullup (=1) or pulldown (=0) on V5.	R/W	0x0
20	CONF_PU_PD_V9	Configure pullup (=1) or pulldown (=0) on V9.	R/W	0x0
19	CONF_PU_PD_W9	Configure pullup (=1) or pulldown (=0) on W9.	R/W	0x0
18	CONF_PU_PD_AA9	Configure pullup (=1) or pulldown (=0) on AA9.	R/W	0x0
17	CONF_PU_PD_Y10	Configure pullup (=1) or pulldown (=0) on Y10.	R/W	0x0
16	CONF_PU_PD_R11	Configure pullup (=1) or pulldown (=0) on R11.	R/W	0x0
15	CONF_PU_PD_P11	Configure pullup (=1) or pulldown (=0) on P11.	R/W	0x0
14	CONF_PU_PD_V10	Configure pullup (=1) or pulldown (=0) on V10.	R/W	0x0
13	CONF_PU_PD_V11	Configure pullup (=1) or pulldown (=0) on V11.	R/W	0x0
12	CONF_PU_PD_W10	Configure pullup (=1) or pulldown (=0) on W10.	R/W	0x0
11	CONF_PU_PD_P13	Configure pullup (=1) or pulldown (=0) on P13.	R/W	0x0
10	CONF_PU_PD_R13	Configure pullup (=1) or pulldown (=0) on R13.	R/W	0x0

Table 45. Pullup/Pulldown Selection 2 Register (PU_PD_SEL_2) (Continued)

Base Address = 0xFFFE 1000, Offset Address = 0xBC				
Bit	**Name**	**Function**	**R/W**	**Reset**
9	CONF_PU_PD_V15	Configure pullup (=1) or pulldown (=0) on V15.	R/W	0x0
8	CONF_PU_PD_P14	Configure pullup (=1) or pulldown (=0) on P14.	R/W	0x0
7	CONF_PU_PD_AA17	Configure pullup (=1) or pulldown (=0) on AA17.	R/W	0x0
6	CONF_PU_PD_Y15	Configure pullup (=1) or pulldown (=0) on Y15.	R/W	0x0
5	CONF_PU_PD_W15	Configure pullup (=1) or pulldown (=0) on W15.	R/W	0x0
4	CONF_PU_PD_W14	Configure pullup (=1) or pulldown (=0) on W14.	R/W	0x0
3	CONF_PU_PD_Y14	Configure pullup (=1) or pulldown (=0) on Y14.	R/W	0x0
2	CONF_PU_PD_V14	Configure pullup (=1) or pulldown (=0) on V14.	R/W	0x0
1	CONF_PU_PD_R14	Configure pullup (=1) or pulldown (=0) on R14.	R/W	0x0
0	CONF_PU_PD_AA15	Configure pullup (=1) or pulldown (=0) on AA15.	R/W	0x0

This register controls the selection of the pullup or pulldown (0 = pulldown, 1 = pullup). The pinout documentation (Application Processor Data Manual (SPRS231)) must be consulted to determine whether a pullup or pulldown exists on the specified I/O. COMP_MODE_CTRL_0 must be programmed to 0xEAEF for this register to control the selection.

Table 46. Pullup/Pulldown Selection 3 Register (PU_PD_SEL_3)

Base Address = 0xFFFE 1000, Offset Address = 0xC0				
Bit	**Name**	**Function**	**R/W**	**Reset**
31	CONF_PU_PD_W2	Configure pullup (=1) or pulldown (=0) on W2.	R/W	0X0
30	CONF_PU_PD_V4	Configure pullup (=1) or pulldown (=0) on V4.	R/W	0X0
29	CONF_PU_PD_Y1	Configure pullup (=1) or pulldown (=0) on Y1.	R/W	0X0
28	CONF_PU_PD_Y17	Configure pullup (=1) or pulldown (=0) on Y17.	R/W	0X0
27	CONF_PU_PD_D18	Configure pullup (=1) or pulldown (=0) on D18.	R/W	0X0
26	CONF_PU_PD_B21	Configure pullup (=1) or pulldown (=0) on B21.	R/W	0X0
25	CONF_PU_PD_C19	Configure pullup (=1) or pulldown (=0) on C19.	R/W	0X0
24	CONF_PU_PD_G14	Configure pullup (=1) or pulldown (=0) on G14.	R/W	0X0
23	CONF_PU_PD_H13	Configure pullup (=1) or pulldown (=0) on H13.	R/W	0X0

Table 46. Pullup/Pulldown Selection 3 Register (PU_PD_SEL_3) (Continued)

Base Address = 0xFFFE 1000, Offset Address = 0xC0				
Bit	**Name**	**Function**	**R/W**	**Reset**
22	CONF_PU_PD_A20	Configure pullup (=1) or pulldown (=0) on A20.	R/W	0X0
21	CONF_PU_PD_B19	Configure pullup (=1) or pulldown (=0) on B19.	R/W	0X0
20	CONF_PU_PD_C18	Configure pullup (=1) or pulldown (=0) on C18.	R/W	0X0
19	Reserved	Reserved.	R/W	0X0
18	CONF_PU_PD_D17	Configure pullup (=1) or pulldown (=0) on D17.	R/W	0X0
17	CONF_PU_PD_D16	Configure pullup (=1) or pulldown (=0) on D16.	R/W	0X0
16	CONF_PU_PD_C17	Configure pullup (=1) or pulldown (=0) on C17.	R/W	0X0
15	CONF_PU_PD_B17	Configure pullup (=1) or pulldown (=0) on B17.	R/W	0X0
14	CONF_PU_PD_G13	Configure pullup (=1) or pulldown (=0) on G13.	R/W	0X0
13	CONF_PU_PD_A17	Configure pullup (=1) or pulldown (=0) on A17.	R/W	0X0
12	CONF_PU_PD_C16	Configure pullup (=1) or pulldown (=0) on C16.	R/W	0X0
11	CONF_PU_PD_D15	Configure pullup (=1) or pulldown (=0) on D15.	R/W	0X0
10	Reserved	Reserved.	R/W	0X0
9	Reserved	Reserved.	R/W	0X0
8	CONF_PU_PD_W11	Configure pullup (=1) or pulldown (=0) on W11.	R/W	0x0
7	Reserved	Reserved.	R/W	0x0
6	CONF_PU_PD_M4	Configure pullup (=1) or pulldown (=0) on M4.	R/W	0x0
5	CONF_PU_PD_W4	Configure pullup (=1) or pulldown (=0) on W4.	R/W	0x0
4	CONF_PU_PD_Y4	Configure pullup (=1) or pulldown (=0) on Y4.	R/W	0x0
3	CONF_PU_PD_V6	Configure pullup (=1) or pulldown (=0) on V6.	R/W	0x0
2	CONF_PU_PD_W5	Configure pullup (=1) or pulldown (=0) on W5.	R/W	0x0
1	CONF_PU_PD_Y5	Configure pullup (=1) or pulldown (=0) on Y5.	R/W	0x0
0	CONF_PU_PD_R9	Configure pullup (=1) or pulldown (=0) on R9.	R/W	0x0

This register controls the selection of the pullup or pulldown (0 = pulldown, 1 = pullup). The pinout documentation in SPRS231 should be consulted to determine whether a pullup or pulldown exists on the specified I/O. COMP_MODE_CTRL_0 must be programmed to 0xEAEF for this register to control the selection.

Table 47. Pullup/Pulldown Selection 4 Register (PU_PD_SEL_4)

Base Address = 0xFFFE 1000, Offset Address = 0xC4				
Bit	**Name**	**Function**	**R/W**	**Reset**
31	CONF_PU_PD_Y18	Configure pullup (=1) or pulldown (=0) on Y18.	R/W	0X0
30	CONF_PU_PD_W18	Configure pullup (=1) or pulldown (=0) on W18.	R/W	0X0
29	CONF_PU_PD_V17	Configure pullup (=1) or pulldown (=0) on V17.	R/W	0X0
28	CONF_PU_PD_Y19	Configure pullup (=1) or pulldown (=0) on Y19.	R/W	0X0
27	CONF_PU_PD_V18	Configure pullup (=1) or pulldown (=0) on V18.	R/W	0X0
26	CONF_PU_PD_L3	Configure pullup (=1) or pulldown (=0) on L3.	R/W	0x0
25	CONF_PU_PD_M8	Configure pullup (=1) or pulldown (=0) on M8.	R/W	0x0
24	CONF_PU_PD_M7	Configure pullup (=1) or pulldown (=0) on M7.	R/W	0x0
23	CONF_PU_PD_M3	Configure pullup (=1) or pulldown (=0) on M3.	R/W	0x0
22	CONF_PU_PD_N8	Configure pullup (=1) or pulldown (=0) on N8.	R/W	0x0
21	CONF_PU_PD_N3	Configure pullup (=1) or pulldown (=0) on N3.	R/W	0X0
20	CONF_PU_PD_J8	Configure pullup (=1) or pulldown (=0) on J8.	R/W	0x0
19	CONF_PU_PD_D3	Configure pullup (=1) or pulldown (=0) on D3.	R/W	0x0
18	CONF_PU_PD_C1	Configure pullup (=1) or pulldown (=0) on C1.	R/W	0x0
17	CONF_PU_PD_E4	Configure pullup (=1) or pulldown (=0) on E4.	R/W	0x0
16	CONF_PU_PD_D2	Configure pullup (=1) or pulldown (=0) on D2.	R/W	0x0
15	CONF_PU_PD_F4	Configure pullup (=1) or pulldown (=0) on F4.	R/W	0x0
14	CONF_PU_PD_E3	Configure pullup (=1) or pulldown (=0) on E3.	R/W	0x0
13	CONF_PU_PD_J7	Configure pullup (=1) or pulldown (=0) on J7.	R/W	0x0
12	CONF_PU_PD_F3	Configure pullup (=1) or pulldown (=0) on F3.	R/W	0x0
11	CONF_PU_PD_G4	Configure pullup (=1) or pulldown (=0) on G4.	R/W	0x0
10	CONF_PU_PD_G3	Configure pullup (=1) or pulldown (=0) on G3.	R/W	0X0

Table 47. Pullup/Pulldown Selection 4 Register (PU_PD_SEL_4) (Continued)

Base Address = 0xFFFE 1000, Offset Address = 0xC4				
Bit	**Name**	**Function**	**R/W**	**Reset**
9	CONF_PU_PD_G2	Configure pullup (=1) or pulldown (=0) on G2.	R/W	0X0
8	CONF_PU_PD_K8	Configure pullup (=1) or pulldown (=0) on K8.	R/W	0X0
7	CONF_PU_PD_H4	Configure pullup (=1) or pulldown (=0) on H4.	R/W	0X0
6	CONF_PU_PD_H3	Configure pullup (=1) or pulldown (=0) on H3.	R/W	0X0
5	CONF_PU_PD_K7	Configure pullup (=1) or pulldown (=0) on K7.	R/W	0X0
4	CONF_PU_PD_L4	Configure pullup (=1) or pulldown (=0) on L4.	R/W	0X0
3	CONF_PU_PD_V2	Configure pullup (=1) or pulldown (=0) on V2.	R/W	0X0
2	CONF_PU_PD_P3	Configure pullup (=1) or pulldown (=0) on P3.	R/W	0X0
1	CONF_PU_PD_U4	Configure pullup (=1) or pulldown (=0) on U4.	R/W	0X0
0	CONF_PU_PD_W1	Configure pullup (=1) or pulldown (=0) on W1.	R/W	0X0

This register controls the selection of the pullup or pulldown (0 = pulldown, 1 = pullup). Consult the pinout section of the Application Processor Data Manual (SPRS231) to determine whether a pullup or pulldown exists on the specified I/O. COMP_MODE_CTRL_0 must be programmed to 0xEAEF for this register to control the selection.

Table 48. Module Configuration Control 1 Register (MOD_CONF_CTRL_1)

Base Address = 0xFFFE 1000, Offset Address = 0x110				
Bit	**Name**	**Function**	**R/W**	**Reset**
31	CONF_CAM_CLKMUX_R	Selection of camera_interface clock (MCLK) between ARMXOR and CAM_MCKO. 0: Corresponds to ARMXOR clock from OMAP3.2. 1: Corresponds to CAM_MCKO from ULPD module.	R/W	0x0
30:29	CONF_PMT_DCB_ SELECT_R	Generates the selection signal for the multiplexer in OMAP3.2 for observability of DLL output bus (DCB[7:0]). 00 and 10: observe DCB bus from URD_DLL. 01 and 11: observe DCB bus from WRQ_DLL.	R/W	0x0

Table 48. Module Configuration Control 1 Register (MOD_CONF_CTRL_1) (Continued)

Base Address = 0xFFFE 1000, Offset Address = 0x110				
Bit	**Name**	**Function**	**R/W**	**Reset**
28	CONF_OSC1_GZ_R	Disables oscillator. See Table 49. 0: 12-MHz oscillator GZ pin is not activated. 1: 12-MHz oscillator GZ pin is activated (oscillator is disabled).	R/W	0x0
27	CONF_OSC1_PWRDN_R	See Table 49. 0: 12-MHz oscillator is not powered-down (uses crystal). 1: 12-MHZ PWRDN pin is activated.	R/W	0x0
26	RESERVED	Reserved.	R/W	0x1
25	OCP_INTERCON_ GATE_EN_R	This bit enables the OCP interconnect to use its autoidle feature on OCP interface.	R/W	0x1
24	CONF_MMC2_CLKFB_ SEL_R	This bit selects clock feedback into MMC/SDIO. 0: Output clock from MMC/SDIO is feedback. 1: Feedback comes from pad.	R/W	0x0
23	RESERVED	Reserved.	R/W	0x0
22	CONF_MCBSP3_CLK_ DIS_R	This bit enables the McBSP3 interface clock. 0: McBSP3 clock is enabled. 1: McBSP3 clock is disabled.	R/W	0x0
21	CONF_MCBSP2_CLK_ DIS_R	This bit enables the McBSP2 interface clock. 0: McBSP2 clock is enabled. 1: McBSP2 clock is disabled.	R/W	0x0
20	CONF_MCBSP1_CLK_ DIS_R	This bit enables the McBSP1 interface clock. 0: McBSP1 clock is enabled. 1: McBSP1 clock is disabled.	R/W	0x0
19:17	RESERVED	Reserved.	R/W	0x2
16	RESERVED	Reserved.	R/W	0x0 or 0x1

Table 48. Module Configuration Control 1 Register (MOD_CONF_CTRL_1) (Continued)

Base Address = 0xFFFE 1000, Offset Address = 0x110				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:14	CONF_MOD_GPTIMER8_CLK_SEL_R	This register selects the clock source for general-purpose timer 8. 00 = ARMXOR_CLK 01 = 32-kHz clock 10 = EXT_CLK of the device external timer clock 11 = Reserved; do not use	R/W	0x0
13:12	CONF_MOD_GPTIMER7_CLK_SEL_R	This register selects the clock source for general-purpose timer 7. 00 = ARMXOR_CLK 01 = 32-kHz clock 10 = EXT_CLK of the device external timer clock 11 = Reserved; do not use	R/W	0x0
11:10	CONF_MOD_GPTIMER6_CLK_SEL_R	This register selects the clock source for general-purpose timer 6. 00 = ARMXOR_CLK 01 = 32-kHz clock 10 = EXT_CLK of the device external timer clock 11 = Reserved; do not use	R/W	0x0
9:8	CONF_MOD_GPTIMER5_CLK_SEL_R	This register selects the clock source for general-purpose timer 5. 00 = ARMXOR_CLK 01 = 32-kHz clock 10 = EXT_CLK of the device external timer clock 11 = Reserved; do not use	R/W	0x0
7:6	CONF_MOD_GPTIMER4_CLK_SEL_R	This register selects the clock source for general-purpose timer 4. 00 = ARMXOR_CLK 01 = 32-kHz clock 10 = EXT_CLK of the device external timer clock 11 = Reserved; do not use.	R/W	0x0

Table 48. Module Configuration Control 1 Register (MOD_CONF_CTRL_1) (Continued)

Base Address = 0xFFFE 1000, Offset Address = 0x110				
Bit	**Name**	**Function**	**R/W**	**Reset**
5:4	CONF_MOD_GPTIMER3_ CLK_SEL_R	This register selects the clock source for general-purpose timer 3. 00 = ARMXOR_CLK 01 = 32-kHz clock 10 = EXT_CLK of the device external timer clock 11 = Reserved; do not use	R/W	0x0
3:2	CONF_MOD_GPTIMER2_ CLK_SEL_R	This register selects the clock source for general-purpose timer 2. 00 = ARMXOR_CLK 01 = 32-kHz clock 10 = EXT_CLK of the device external timer clock 11 = Reserved; do not use	R/W	0x0
1:0	CONF_MOD_GPTIMER1_ CLK_SEL_R	This register selects the clock source for general-purpose timer 1. 00 = ARMXOR_CLK 01 = 32-kHz clock 10 = EXT_CLK of the device external timer clock 11 = Reserved; do not use	R/W	0x0

This register controls the module configuration of the device.

Table 49. GZ and PWRDN Bits in MOD_CONF_CTRL_1

CONF_OSC1_GZ_R	CONF_OSC1_PWRDN_R†	Comments
0	0	Normal operating mode w/internal bias resistor.
0	1	Normal operating mode w/disconnected internal bias resistor. Enables the use of an external square clock.
1	X	Oscillator disabled.

† In reset mode 1, the PWRDN bit is forced to 1.

Table 50. Configuration Status Register (CONF_STATUS)

Base Address = 0xFFFE 1000, Offset Address = 0x130				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:6	UNUSED	These bits are not implemented.	R	N/A
5:4	CONF_DEVICE_TYPE_R	Contains the status of the eFuses that determine the type of device. 00: Production (normal) device 01: Bad device 10: Emulator device 11: Test device	R	N/A
3	CONF_STATUS[3]	Contains the status of GPIO1 input pin. Value is latched on the rising edge of PWRON_RESET. For internal boot, software configures the external chip-select after reading this bit. For external boot, the external chip-select configuration is forced after reading this bit. 0: Nonaddress/data multiplexed 1: Address/data multiplexed	R	N/A
2	CONF_EMIFS_MUX_STAT_R	This bit contains information about EMIFS protocol at boot. 0: Nonaddress/data multiplexed 1: Address/data multiplexed Reset value for this signal depends on GPIO1, MPU_BOOT, and EFUSE_DEVICE_TYPE signals.	R	N/A

Table 50. Configuration Status Register (CONF_STATUS) (Continued)

Base Address = 0xFFFE 1000, Offset Address = 0x130				
Bit	**Name**	**Function**	**R/W**	**Reset**
1	CONF_ARM_BOOT_STAT_R	This register contains boot mode active chip-select (CONF_ARM BOOT MODE), latched at the rising edge of PWRON_RESET. Emulator type devices only, 0: MPU boots from internal ROM 1: MPU boots from external memory Reset value for this signal depends on MPU_BOOT and EFUSE_DEVICE_TYPE signals.	R	N/A
0	CONF_RESET_MODE_STAT_R	This register contains the status of the RESET_MODE pin latched at the rising edge of the reset pin PWRON_RESET. 0: Reset mode 0 1: Reset mode 1	R	N/A

The actual reset value for all bits in this register depends on the device configuration at power-up reset. The values from the Reset column are not applicable.

The boot ROM code uses CONF_STATUS register bits to determine the execution path during boot time.

Table 51. Reset Control Register (RESET_CONTROL)

Base Address = 0xFFFE 1000, Offset Address = 0x140				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:7	UNUSED	Not implemented.	R	0x0000000
6	CONF_RNG_IDLE_MODE	RNGidle control. 0: RNGidle disabled 1: RNGidle enabled	R/W	0x1
5	CONF_CAMERAIF_RESET_R	This register controls reset of the camera IF. 0: Module is in reset. 1: Module is in functional mode.	R/W	0x1
4	CONF_UWIRE_RESET_R	This register controls reset of the µWire. 0: µWire is in reset. 1: µWire is in functional mode.	R/W	0x1

Table 51. Reset Control Register (RESET_CONTROL) (Continued)

Base Address = 0xFFFE 1000, Offset Address = 0x140				
Bit	**Name**	**Function**	**R/W**	**Reset**
3	CONF_OSTIMER_ RESET_R	This register controls reset of the OS timer. 0: OS timer is in reset. 1: OS timer is in functional mode.	R/W	0x1
2	CONF_ARMIO_RESET_R	This register controls reset of the MPUIO. 0: MPUIO is in reset. 1: MPUIO is in functional mode.	R/W	0x1
1	RESERVED	This bit should be set to 0.	R/W	0x1
0	CONF_OCP_RESET_R	This register controls reset of the OCP interconnect. 0: OCP interconnect is in reset. 1: OCP interconnect is in functional mode.	R/W	0x1

This register can be used for power management for the various modules listed in the bit fields above. These peripherals can be held under reset by this register.

Table 52. OMAP5912 Configuration Control Register (CONF_5912_CTRL)

Base Address = 0xFFFE 1000, Offset Address = 0x150				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:16	RESERVED	Reserved.	R	0x0000
15:4	RESERVED	Reserved.	R/W	0x000
3	RESERVED	Reserved.	R/W	0
2	RESERVED	This bit should be set to 0.	R/W	0
1	RESERVED	Reserved.	R/W	1
0	RESERVED	Reserved.	R/W	0

3 External Interfaces

This section describes the various external interfaces supported by this device. It also gives an overview of duplicated interfaces.

3.1 External Interface Descriptions

This device supports different classes of interfaces:

- Reset, clocks, and power management
- General-purpose input/output
- Keyboard interface
- General-purpose counter
- Memory interfaces
  - SDRAM interface
  - Mobile DDR interface
  - NOR flash interface (nonmultiplexed address/data and multiplexed address/data)
  - CompactFlash interface
  - MMC/SD (1,2)
- Serial interfaces
  - SPI (master/slave)
  - μWire
  - McBSP (1,2,3)
  - MCSI (1,2)
  - I^2C (master/slave)
  - HDQ/1-Wire
  - UART (1,2,3)
  - USB On The Go (OTG)
    - 1 port with on-chip transceiver
    - 3 ports that can interface with external transceivers
- Camera interface
  - Parallel camera interface

- Display interface
  - LCD interface
- Emulation
  - JTAG
  - Trace interface

The user selects the appropriate interfaces for an application during the boot sequence. In effect, only a subset of all possible interfaces is available at power-on reset. For flexibility, some interfaces are duplicated. See the pinout section of the Application Processor Data Manual (SPRS231) to select the desired configuration.

3.2 Duplicated Interfaces

Table 53 provides an example of duplicated interfaces with the MMC/SDIO2. The software programs (via the appropriate FUNC_MUX_CTRL register) the desired configuration and ensures that no input/output is enabled twice.

Table 53. Example of Duplicated Interfaces

Ball	I/O Description	Mode
M19	MMC2.CLK	011
Y10	MMC2.CLK	110
R18	MMC2.CLKIN	110
J19	MMC2.CMD/SPI.DO	011
Y8	MMC2.CMD/SPI.DO	110
V9	MMC2.CMDDIR	110
L15	MMC2.DAT0/SPI.DI	011
W8	MMC2.DAT0/SPI.DI	110
L18	MMC2.DAT1	011
V8	MMC2.DAT1	110
J18	MMC2.DAT2	011
W15	MMC2.DAT2	110
L19	MMC2.DAT3	011
R10	MMC2.DAT3	110
V5	MMC2.DATDIR0	110
W19	MMC2.DATDIR1	110

Table 54 indicates the interfaces after programming the configuration registers.

Table 54. Configuration After Programming

Ball	I/O Description	Mode
Y10	MMC2.CLK	110
Y8	MMC2.CMD/SPI.DO	110
V9	MMC2.CMDDIR	110
W8	MMC2.DAT0/SPI.DI	110
V8	MMC2.DAT1	110
R10	MMC2.DAT3	110
V5	MMC2.DATDIR0	110
W19	MMC2.DATDIR1	110
L15	ETM_PSTAT[1]	001
L18	ETM_PSTAT[2]	001
M19	ETM_PSTAT[0]	001
J19	ETM_D[6]	001
J18	ETM_D[7]	001
L19	ETM_D[0]	001

4 Reset/Boot Overview

The memory interfaces and clock sources are controlled and configured during power-on reset by external pins and on-chip electronic fuses (eFuses).

4.1 Boot Mode Control and EMIFS Multiplexing Control Generation

The device type changes depending on the programming at the probe of the eFuse bits. The device type can be observed with the CONF_STATUS[5:4] bits. See Section 5 for more information on the device types.

The EMIFS chip-selects and the flash protocol can be either address/data non-multiplexed or address/data multiplexed (note that multiplexing address/data lines on the EMIFS bus should be considered a completely different topic from the pin multiplexing controlled by the FUNC_MUX_CTRL(3–12) registers).

There are two ways to affect the EMIFS bus protocol for the data and address lines.

First, the value of GPIO1, sampled at power-on reset, determines whether or not the address/data lines are multiplexed when RESET_MODE is 0. GPIO1 affects the EMIFS protocol only at boot time, after which it can be changed by software (e.g. by the boot ROM code). When RESET_MODE is 1, the address/data lines are multiplexed at boot time (GPIO1 should be 0 when reset mode is 1, otherwise the results are unpredictable).

Second, the value of the MAD bit in the EMIFS chip select configuration registers controls the EMIFS bus protocol. This is a software selectable switch on a per chip-select basis.

Table 55 summarizes EMIFS configuration at and after boot time.

Table 55. EMIFS Multiplexing Control and MPU_BOOT Mode Signal Generation for Reset Mode 0

DEVICE_TYPE Production Emulation	MPU_BOOT (sampled at reset)	GPIO1 (sampled at reset)	Boot	Default EMIFS Protocol (at Boot)	EMIFS Protocol (after boot)
Production	Don't care	0	Internal on CS0	Nonaddress/data multiplexed	Nonaddress/data multiplexed Set by boot code software (CONF_STATUS[3] = 0)
Production	Don't care	1	Internal on CS0	Nonaddress/data multiplexed	Address/data multiplexed Set by boot code software (CONF_STATUS[3] = 1)

Table 55. EMIFS Multiplexing Control and MPU_BOOT Mode Signal Generation for Reset Mode 0 (Continued)

DEVICE_TYPE Production Emulation	MPU_BOOT (sampled at reset)	GPIO1 (sampled at reset)	Boot	Default EMIFS Protocol (at Boot)	EMIFS Protocol (after boot)
Emulation	0	0	Internal on CS0	Nonaddress/data multiplexed	Nonaddress/data multiplexed Set by boot code software (CONF_STATUS[3] = 0)
Emulation	0	1	Internal on CS0	Nonaddress/data multiplexed	Address/data multiplexed Set by boot code software (CONF_STATUS[3] = 1)
Emulation	1	0	External on CS3	Nonaddress/data multiplexed	Nonaddress/data multiplexed Set by hardware at reset release from CONF_STATUS[3] = 0
Emulation	1	1	External on CS3	Address/data multiplexed	Address/data multiplexed Set by hardware at reset release from CONF_STATUS[3] = 1

4.2 Configuration of Interfaces in Internal Boot ROM

Based on the value of the RESET_MODE pin at power-on reset, several different configurations are supported by the internal boot ROM. Below, each of the possible configurations is discussed.

The boot ROM modifies a number of different interfaces with the configuration module (FUNC_MUX_CTRL_(3−12), PULL_DWN_CTRL_(0−4), and PU_PD_SEL_(0−4) registers).

Depending on the boot ROM execution path, some pin multiplexing registers are modified from their reset values (all zeros) to accommodate the relevant interface(s). One or more of the following configurations may be implemented by the boot ROM code.

- Configuration needed for NAND EMIFS (enable GPIO_10).
  - FUNC_MUX_CTRL_F: 0x0800 0000
- Configuration needed for USB Client Initialization
  - FUNC_MUX_CTRL_D: 0x0000 0018
- Configuration needed for USB port1 detection
  - FUNC_MUX_CTRL_8: 0x1000 0000
  - FUNC_MUX_CTRL_9: 0x0100 0014
  - FUNC_MUX_CTRL_A: 0x0000 0290
- Configuration needed to enable UART pins
  - FUNC_MUX_CTRL_6: 0x0000 0001
  - FUNC_MUX_CTRL_9: 0x0020 1000
  - FUNC_MUX_CTRL_10: 0x0000 0001
  - PULL_DWN_CTRL_2: 0xD028 2E59
  - PULL_DWN_CTRL_3: 0xFFFF FEFC
  - PULL_DWN_CTRL_4: 0x87FF FFF7
  - PU_PD_SEL_4: 0x0000 0008

5 OMAP Device Identification Registers

The following registers are chip revision registers for the device. Reset values depend on the device are are listed in the tables as unknown.

Table 56. OMAP Die ID Register (OMAP_DIE_ID_0)

Address = 0xFFFE 1800				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:0	RESERVED	Reserved for special identification.	R	U

Table 57. OMAP Die ID Register (OMAP_DIE_ID_1)

Address = 0xFFFE 1804				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:21	RESERVED	Reserved for special identification.	R	U
20:17	DEV_REV	Device revision.	R	See Table 61
16:0	RESERVED	Reserved for special identification.	R	U

Table 58. OMAP Die ID Register (OMAP_PRODUCTION_ID_0)

Address = 0xFFFE 2000				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:30	SECURITY	Security device type. 00: General-purpose device type (GP)	R	U
29:25	RESERVED	Reserved.	R	U
24:9	ID_KEY	Reserved for special identification.	R	0x5555
8	SP	Secure protect.	R	U
7	WA	DFT write MPU. MPU DFT write value.	R	U
6	RA	DFT read MPU. MPU DFT read value.	R	U
5	WD	DFT write MGS3. MGS3 DFT write value.	R	U
4	RD	DFT read MGS3. MGS3 DFT read value.	R	U
3:2	NORMAL	00: Not a normal (production) device. 01,10,11: Normal device.	R	U
1:0	EMULATION	00: Not an emulation device. 01,10,11: Emulation device.	R	U

Table 59. OMAP Die ID Register (OMAP_PRODUCTION_ID_1)

Address = 0xFFFE 2004				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:17	RESERVED	Reserved.	R	U
16:1	PROD_ID	Prod ID.	R	See Table 61
0	RESERVED	Reserved.	R	U

Table 60. OMAP32_ID Register (OMAP32_ID)

Address = 0xFFFE D400				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:0	OMAP32_ID	OMAP 3.2 device revision.	R	U

Some of the bit fields described above will change depending on the chip revision. Table 61 lists device revisions. Note that new revisions may exist. Please contact your TI representative for more information.

Table 61. Revision Table

Revision	Part #	PROD_ID	DEV_REV	OMAP32_ID
1.1	XOMAP5912B	0xB576	0x2	0x0332 0200
2.2	POMAP5912	0xB65F	0x2	0x0332 0500
2.2	OMAP5912B	0xB65F	0x2	0x0332 0500

This page is intentionally left blank.

Index

OMAP5912 Multimedia Processor Power Management Reference Guide

Literature Number: SPRU753A
March 2004

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.

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
		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 document describes power management in the OMAP5912 multimedia processor.

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.

Related Documentation From Texas Instruments

The following documents describe the OMAP5910 device and related peripherals. Copies of these documents are available on the Internet at www.ti.com. *Tip:* Enter the literature number in the search box provided at www.ti.com.

OMAP5912 Multimedia Processor Device Overview and Architecture Reference Guide (literature number SPRU748) introduces the setup, components, and features of the OMAP5912 multimedia processor and provides a high-level view of the device architecture.

OMAP5912 Multimedia Processor OMAP 3.2 Subsystem Reference Guide (literature number SPRU749) introduces and briefly defines the main features of the OMAP3.2 subsystem of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor DSP Sybsystem Reference Guide (literature number SPRU750) describes the OMAP5912 multimedia processor DSP subsystem. The digital signal processor (DSP) subsystem is built around a core processor and peripherals that interface with: 1) The ARM926EJS via the microprocessor unit interface (MPUI); 2) Various standard memories via the external memory interface (EMIF); 3) Various system peripherals via the TI peripheral bus (TIPB) bridge.

OMAP5912 Multimedia Processor Clocks Reference Guide (literature number SPRU751) describes the clocking mechanisms of the OMAP5912 multimedia processor. In OMAP5912, various clocks are created from special components such as the digital phase locked loop (DPLL) and the analog phase-locked loop (APLL).

OMAP5912 Multimedia Processor Initialization Reference Guide (literature number SPRU752) describes the reset architecture, the configuration, the initialization, and the boot ROM of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Power Management Reference Guide (literature number SPRU753) describes power management in the OMAP5912 multimedia processor. The ultralow-power device (ULPD) generates and manages clocks and reset signals to OMAP3.2 and to some peripherals. It controls chip-level power-down modes and handles chip-level wake-up events. In deep sleep mode, this module is still active to monitor wake-up events.This book describes the ULPD module and outline architecture.

OMAP5912 Multimedia Processor Security Features Reference Guide (literature number SPRU754) describes the security features of teh OMAP5912 multimedia processor. The OMAP5912 security scheme relies on the OMAP3.2 secure mode. The distributed security on the OMAP3.2 platform is a Texas Instruments solution to address m-commerce and security issues within a mobile phone environment. The OMAP3.2 secure mode was developed to bring hardware robustness to the overall OMAP5912 security scheme.

OMAP5912 Multimedia Processor Direct Memory Access (DMA) Support Reference Guide (literature number SPRU755) describes the direct memory access support of the OMAP5912 multimedia processor. The OMAP5912 processor has three DMAs:

- The system DMA is embedded in OMAP3.2. It handles DMA transfers associated with MPU and shared peripherals.
- The DSP DMA is embedded in OMAP3.2. It handles DMA transfers associated with DSP peripherals.
- The generic distributed DMA (GDD) is an OMAP5912 resource attached to the SSI peripheral. It handles only DMA transfers associated with the SSI peripheral.

OMAP5912 Multimedia Processor Memory Interfaces Reference Guide (literature number SPRU756) describes the memory interfaces of the OMAP5912 multimedia processor.

- SDRAM (external memory interface fast, or EMIFF)
- Asynchronous and synchronous burst memory (external memory interface slow, or EMIFS)
- NAND flash (hardware controller or software controller)
- CompactFlash on EMIFS interface
- Internal static RAM

OMAP5912 Multimedia Processor Interrupts Reference Guide (literature number SPRU757) describes the interrupts of the OMAP5912 multimedia processor. Three level 2 interrupt controllers are used in OMAP5912:

- One MPU level 2 interrupt handler (also referred to as MPU interrupt level 2) is implemented outside of OMAP3.2 and can handle 128 interrupts.
- One DSP level 2 interrupt handler (also referred to as DSP interrupt level 2.1) is instantiated outside of OMAP3.2 and can handle 64 interrupts.
- One OMAP3.2 DSP level 2 interrupt handler (referenced as DSP interrupt level 2.0) can handle 16 interrupts.

OMAP5912 Multimedia Processor Peripheral Interconnects Reference Guide (literature number SPRU758) describes various periperal interconnects of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Timers Reference Guide (literature number SPRU759) describes various timers of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Serial Interfaces Reference Guide (literature number SPRU760) describes the serial interfaces of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Universal Serial Bus (USB) Reference Guide (literature number SPRU761) describes the universal serial bus (USB) host on the OMAP5912 multimedia processor. The OMAP5912 processor provides several varieties of USB functionality. Flexible multiplexing of signals from the OMAP5912 USB host controller, the OMAP5912 USB function controller, and other OMAP5912 peripherals allow a wide variety of system-level USB capabilities. Many of the OMAP5912 pins can be used for USB-related signals or for signals from other OMAP5912 peripherals. The OMAP5912 top-level pin multiplexing

controls each pin individually to select one of several possible internal pin signal interconnections. When these shared pins are programmed for use as USB signals, the OMAP5912 USB signal multiplexing selects how the signals associated with the three OMAP5912 USB host ports and the OMAP5912 USB function controller can be brought out to OMAP5912 pins.

OMAP5912 Multimedia Processor Multi-channel Buffered Serial Ports (McBSPs) Reference Guide (literature number SPRU762) describes the three multi-channel buffered serial ports (McBSPs) available on the OMAP5912 device. The OMAP5912 device provides multiple high-speed multichannel buffered serial ports (McBSPs) that allow direct interface to codecs and other devices in a system.

OMAP5912 Multimedia Processor Camera Interface Reference Guide (literature number SPRU763) describes two camera inerfaces implemented in the OMAP5912 multimedia processor: compact serial camera port and camera parallel interface.

OMAP5912 Multimedia Processor Display Interface Reference Guide (literature number SPRU764) describes the display interface of the OMAP5912 multimedia processor.

- LCD module
- LCD data conversion module
- LED pulse generator
- Display interface

OMAP5912 Multimedia Processor Multimedia Card (MMC/SD/SDIO) (literature number SPRU765) describes the multimedia card (MMC) interface of the OMAP5912 multimedia processor. The multimedia card/secure data/secure digital IO (MMC/SD/SDIO) host controller provides an interface between a local host, such as a microprocessor unit (MPU) or digital signal processor (DSP), and either an MMC or SD memory card, plus up to four serial flash cards. The host controller handles MMC/SD/SDIO or serial port interface (SPI) transactions with minimal local host intervention.

OMAP5912 Multimedia Processor Keyboard Interface Reference Guide (literature number SPRU766) describes the keyboard interface of the OMAP5912 multimedia processor. The MPUIO module enables direct I/O communication between the MPU (through the public TIPB) and external devices. Two types of I/O can be used: specific I/Os dedicated for 8 x 8 keyboard connection, and general-purpose I/Os.

OMAP5912 Multimedia Processor General-Purpose Interface Reference Guide (literature number SPRU767) describes the general-purpose in-

terface of the OMAP5912 multimedia processor. There are four GPIO modules in the OMAP5912. Each GPIO peripheral controls 16 dedicated pins configurable either as input or output for general purposes. Each pin has an independent control direction set by a programmable register. The two−edge control registers configure events (rising edge, falling edge, or both edges) on an input pin to trigger interrupts or wake−up requests (depending on the system mode). In addition, an interrupt mask register masks out specified pins. Finally, the GPIO peripherals provide the set and clear capabilities on the data output registers and the interrupt mask registers. After detection, all event sources are merged and a single synchronous interrupt (per module) is generated in active mode, whereas a unique wake−up line is issued in idle mode. Eight data output lines of the GPIO3 are ORed together to generate a global output line at the OMAP5912 boundary. This global output line can be used in conjunction with the SSI to provide a CMT−APE interface to the OMAP5912.

OMAP5912 Multimedia Processor VLYNQ Serial Communications Interface Reference Guide (literature number SPRU768) describes the VLYNQ of the OMAP5912 multimedia processor.

VLYNQ is a serial communications interface that enables the extension of an internal bus segment to one or more external physical devices. The external devices are mapped into local, physical address space and appear as if they are on the internal bus of the OMAP 5912. The external devices must also have a VLYNQ interface. The VLYNQ module serializes bus transactions in one device, transfers the serialized data between devices via a VLYNQ port, and de-serializes the transaction in the external device.

OMAP5912 includes one VLYNQ module connected on OCPT2 target port and OCPI initiator port. These connections are configured via a static switch, which selects either SSI or VLYNQ module. This switch, forbids the simultaneous use of GDD/SSI and VLYNQ. The switch is controlled by the VLYNQ_EN bit in the OMAP5912 configuration control register (CONF_5912_CTRL).

OMAP5912 Multimedia Processor Pinout Reference Guide (literature number SPRU769) provides the pinout of the OMAP5912 multimedia processor. After power-up reset, the user can change the configuration of the default interfaces. If another interface is available on top of the default, it is possible to enable a new interface for each ball by setting the corresponding 3-bit field of the associated FUNC_MUX_CTRL register. It is also possible to configure on-chip pullup/pulldown. This document

also describes the various power domains so that the user can apply the different interfaces seamlessly with external components.

OMAP5912 Multimedia Processor Window Tracer (WT) Reference Guide (literature number SPRU770) describes the window tracer module used to capture the memory transactions from four interfaces: EMIFF, EMIFS, OCP-T1, and OCP-T2. This module is located in the OMAP3.2 traffic controller (TC).

OMAP5912 Multimedia Processor Real-Time Clock Reference Guide (literature number SPRUxxx) describes the real-time clock of the OMAP5912 multimedia processor. The real-time clock (RTC) block is an embedded real-time clock module directly accessible from the TIPB bus interface.

Trademarks

Contents

Figures

Tables

Power Management

This document describes power management in the OMAP5912 multimedia processor.

1 Ultralow-Power Device

The ultralow-power device (ULPD) generates and manages clocks and reset signals to OMAP3.2 and to some peripherals. It controls chip-level power-down modes and handles chip-level wake-up events. In deep sleep mode, this module is still active to monitor wake-up events.

This chapter describes the ULPD module and outline architecture**.** For further information on clock sources and clock and reset architecture, see Chapter 4 and Chapter 5.

1.1 ULPD Features

The ULPD has the following features:

- ❏ Performs transitions among power modes (awake, big sleep, and deep sleep)
- ❏ Handles idle/wake-up handshake with OMAP3.2
- ❏ Monitors wake-up events
- ❏ Controls system clock input sources for several possible configurations (oscillator/external clocks)
- ❏ Performs calibration of the 32-kHz oscillator
- ❏ Manages the clocks and resets distributed to OMAP3.2 and to some peripherals
- ❏ Handles the power-up sequence
- ❏ Controls an on-chip PLL that generates a 96-MHz clock (for 48-MHz peripheral clocks)
- ❏ Is controlled by the MPU for set up and configuration
- ❏ Manages the sleep signal of an embedded LDO used to regulate the supply voltage for OMAP3.2 digital phase locked loop (DPLL) and the system clock oscillator

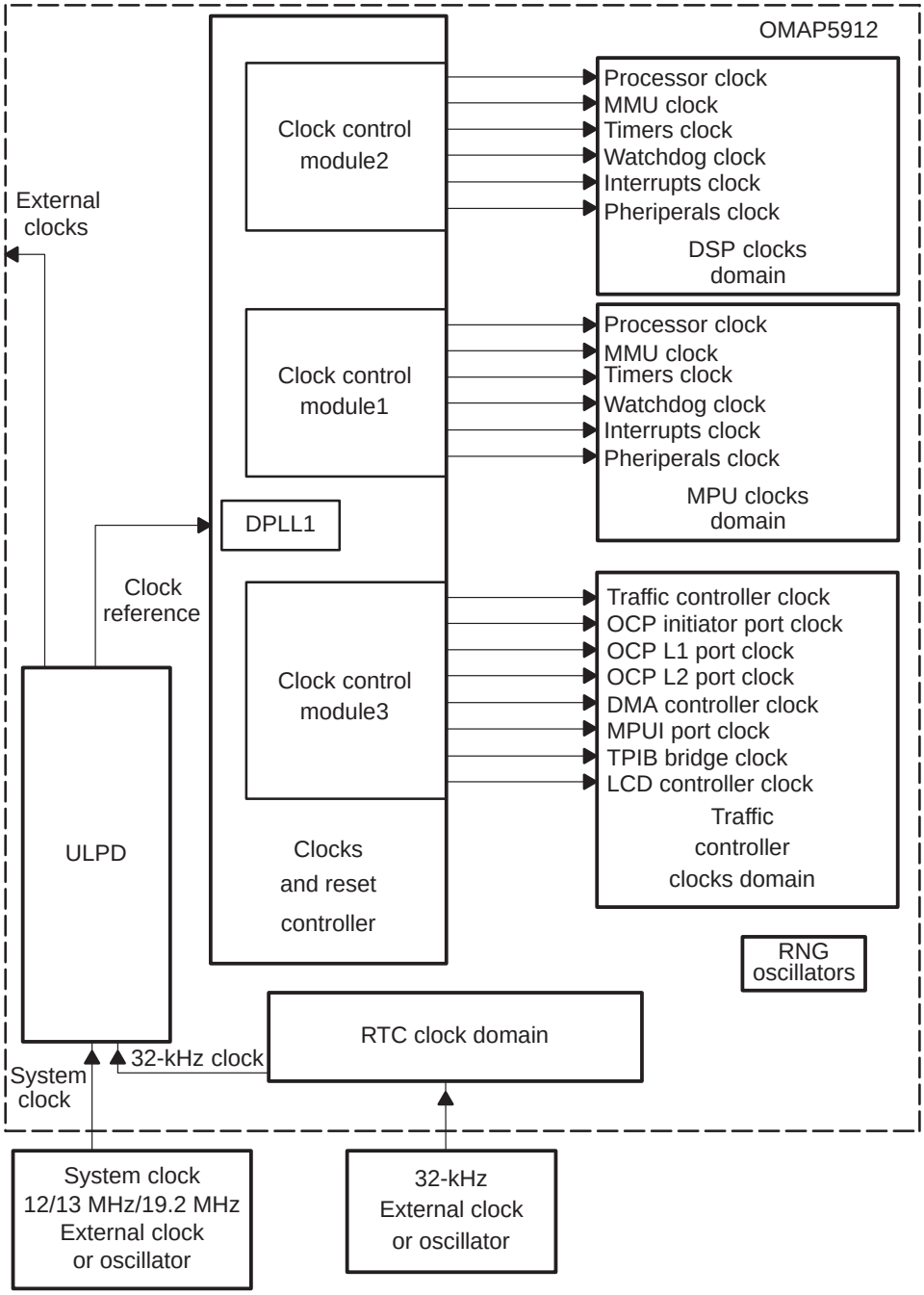
1.2 Overview

The ULPD is a power management module running at 32 kHz (CLK32K).

The ULPD employs three global-system power modes: awake mode, big sleep mode, and deep sleep mode. See Section 1.5, *Power Modes*, for a description of these modes and the transitions between them.

The ULPD controls the various module clocks with several input clock sources provided by the oscillator and by the OMAP3.2 DPLL and the 96-MHz analog phase-locked loop (APLL). It also can sequence the wake-up of the system properly by enabling various analog cells successively. Analog cells are on-chip or external modules that are involved in the generation of the device input clock or supply voltage. Typical analog cells are on-chip oscillators or external supply voltage regulators. ULPD can manage up to six analog cells Figure 1.

Figure 1. ULPD and Clock Domains in OMAP5912



The ULPD is composed of one state machine, a clock management module, and a control register file.

- The state machine (FSM1) manages the global power modes transitions. It handles the idle/wake-up handshake with OMAP3.2 and monitors the wake-up events.

  The state machine controls the input system clock (internal oscillator or external clock source) and generates resets to OMAP3.2 and to some peripherals. It also manages the power-up sequence. The FSM1 uses a set-up timer to manage the sequencing of the wake-up procedure. Each setup timer is associated with a specific analog cell.

- The clock management module is composed of clock-gating logic, multiplexers, and clock dividers. It generates and manages clocks to OMAP3.2 and to some peripherals. It also manages the 48-MHz clocks. Those clocks are generated from an on-chip analog phase locked loop (APLL) operating at 96 MHz. The output of the APLL is divided by two to create a 48-MHz reference clock. From the reference clock are derived the various 48-MHz clocks.

- The control register file is an MPU peripheral connected to the MPU private peripheral bus; it is used to set/configure the features of ULPD.

See Section 1.24 for a detailed description of the registers.

1.3 ULPD Input Clock Sources

The ULPD has two main clock sources: a 32-kHz clock and a system clock of medium frequency (12 MHz, 13 MHz, and 19.2 MHz). These frequencies are also the ULPD_PLL input clock frequency. See Chapter 5 for additional information on the clock source for ULPD.

1.4 ULPD Setup Counters

The ULPD can sequence the wake-up of the system from deep sleep properly by enabling up to six analog cells successively (for example, oscillator and regulator).

The ULPD FSM1 is instantiated in OMAP5912 with two setup counters, each associated with an analog cell. Setup counters are cascaded and must be programmed with the stabilization time of the associated analog cell.

Whenever a counter underflows, it enables the next analog cell and triggers the associated counter.

The cascaded flow of the SETUP_ANALOG_CELL starts with SETUP_ANALOG_CELL2 and ends with SETUP_ANALOG_CELL3.

When SETUP_ANALOG_CELL3 underflows, all the analog cells must be stable and the input system clock is released internally in ULPD.

The ULPD FSM1 can then move in big sleep or awake mode.

1.5 Power Modes

The ULPD handles three global power modes: deep sleep, big sleep, and awake. The ULPD state machine, which is in charge of the wake-up/idle handshake with OMAP3.2, manages the states of the system in each mode and performs transitions between the modes.

1.5.1 Deep Sleep Mode

In deep sleep mode, all internal clocks are inactive except the 32-kHz clock, which is the ULPD state machine clock. In this mode ULPD_PLL is always inactive.

In oscillator mode, the oscillator is disabled; therefore, the system input clock is off, except when POWER_CTRL_REG [9] =0.

In external mode, the input system clock can be on or off; it is not controlled by ULPD.

Note:

OSC12M_STOP output of ULPD is active high every time the state machine is in a deep-sleep state. EXT_CLK_REQ is the same signal but with inverted polarity.

OMAP5912 cannot go into deep−sleep while an emulator (JTAG) is connected

CAUTION

1.5.2 Big Sleep Mode

In big sleep mode, the OMAP input clock is inactive, the 32-kHz clock is active, and the system input clock is active in both oscillator and external modes.

This state is characterized by an external clock request to go active or POWER_CTRL_REG [4] =0.

This mode has a shorter wake-up latency. It also provides clocks (system frequency clocks and/or ULPD_PLL clock) to peripherals whenever requested and while the OMAP3.2 input clock is stopped.

1.5.3 Awake Mode

In awake mode, the OMAP input clock and any requested peripheral clocks are active. In awake mode, the 32-kHz, system clock, OMAP input clock, and any requested peripheral clocks are active.

1.6 External Clock and Voltage Supply Control

The ULPD provides two signals to control the activation or the shut down of the external clock and core voltage supplies.

These two signals, $\overline{\text{LOW\_PWR}}$ and LOW_PWR, behave similarly except that they do not have the same polarity.

LOW_PWR can be controlled by software, whereas $\overline{\text{LOW\_PWR}}$ cannot.

1.6.1 Behavior of $\overline{\text{LOW\_PWR}}$

The $\overline{\text{LOW\_PWR}}$ signal is used in external clock mode.

When low, $\overline{\text{LOW\_PWR}}$ indicates to external devices that the input system clock (SYS_CLK_IN) can be shut down. It can also indicate to an external power management device that the core voltage supply can be lowered to 1.1 V.

The $\overline{\text{LOW\_PWR}}$ signal is asserted low when the ULPD enters the deep sleep state (except at power-up reset) and released upon deep sleep exit (except at power-up reset).

At power-up reset, $\overline{\text{LOW\_PWR}}$ is reset to its inactive value (high).

Figure 2. Assertion of $\overline{\text{LOW\_PWR}}$

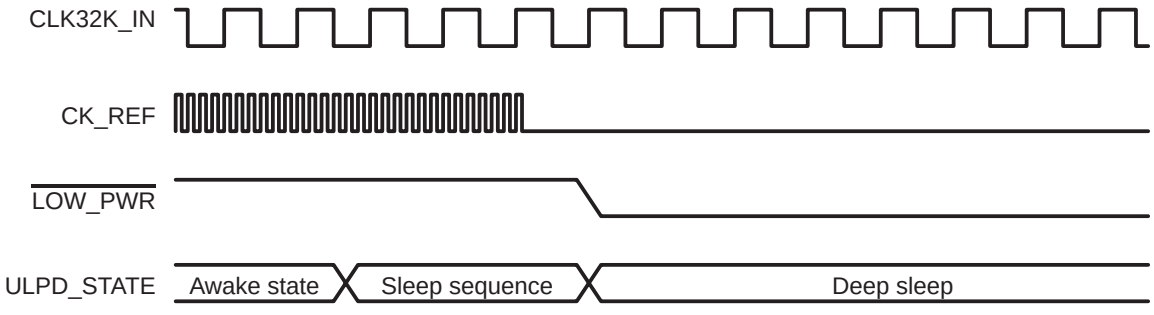
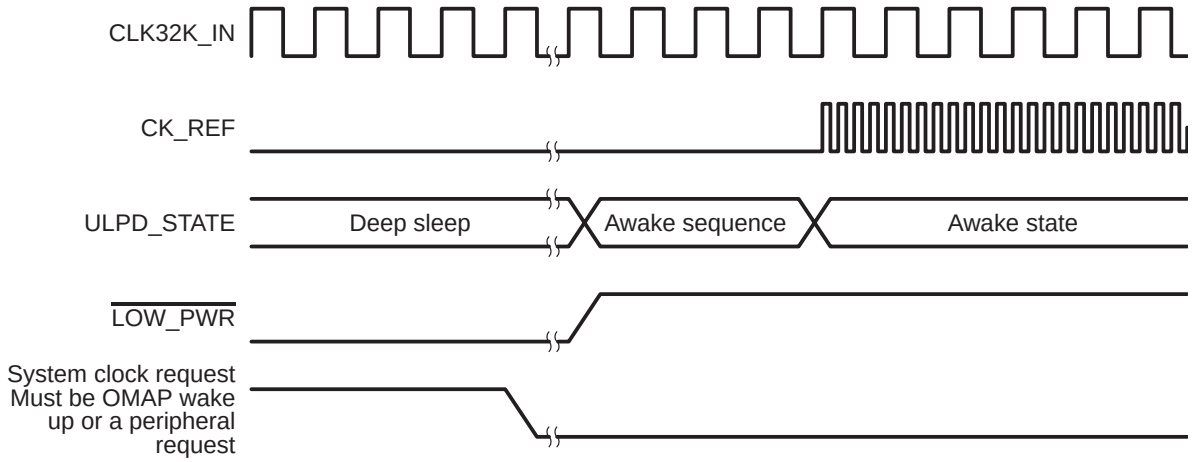


Figure 3. Release of $\overline{LOW\_PWR}$



1.6.2 Behavior of LOW_PWR

The LOW_PWR signal is used in oscillator clock mode to control an external power management device.

When high, the signal LOW_PWR drives the external core voltage supply in low voltage (1.1 V) operations.

LOW_PWR can be set by software so that two types of operations are allowed:

- Reduction of leakage current
- Low-voltage operation at reduced clock frequency, also known as dynamic voltage scaling (DVS)

1.7 Leakage Current Management

The conditions below cause the listed events:

- POWER_CTRL_REG[0] set to 1: Enable LOW_PWR feature
- POWER_CTRL_REG[4] set to 1: Enable transition to deep sleep mode

  or

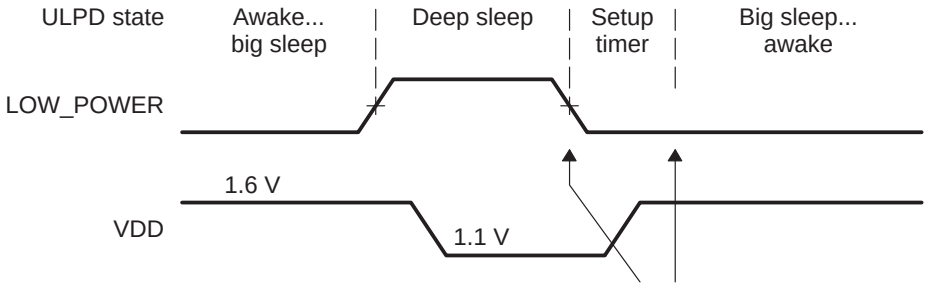
  POWER_CTRL_REG[10] set to 0: DVS disabled

LOW_PWR switches to active high whenever the ULPD enters deep sleep state. In this way, the external core voltage supply can be driven in low-voltage operation by the external power management device.

When the ULPD exits deep sleep mode, LOW_PWR switches back to inactive low and the external core voltage supply ramps up to a nominal 1.5 V.

At reset, the low-power feature is disabled (POWER_CTRL_REG[0] is set to 0). The LOW_PWR signal is inactive low, which indicates a nominal voltage requirement.

Figure 4. Behavior of LOW_PWR in RESET_MODE 0



ULPD setup analog cell timer2 delays deep sleep to big sleep/awake transition while the regulator ramps from 1.1 V to 1.6 V.

1.8 Low-Voltage Operation at Reduced Clock Frequency

In this mode of operation, also known as dynamic voltage scaling (DVS), the following conditions must be met:

- ❏ POWER_CTRL_REG[4] set to 0: Disables transition to deep sleep mode
- ❏ POWER_CTRL_REG[10] set to 1: Enables DVS

Whenever OMAP3.2 indicates to ULPD that it is prepared to go into idle, the transition to deep sleep is prevented, and the ULPD moves into big sleep mode. In this mode, the OMAP3.2 input clock is shut down, but the oscillator is still active.

In this case, LOW_PWR follows the value programmed in POWER_CTRL_REG [11].

If POWER_CTRL_REG [11] =1 (min), LOW_PWR switches to active high, driving the external regulator in low-voltage operation.

If POWER_CTRL_REG [11] =0 (max), LOW_PWR switches to inactive low, driving the external regulator in nominal voltage operation.

Whenever OMAP3.2 initiates a wake-up procedure, the ULPD moves back to awake mode but LOW_PWR keeps the value programmed in POWER_CTRL_REG [11].

In this way, when POWER_CTRL_REG [11] =1, OMAP3.2 restarts operations at low voltage. To ramp up the operating voltage back to nominal value, a new

OMAP3.2 idle/wake-up procedure must be performed with POWER_CTRL_REG[11]=0.

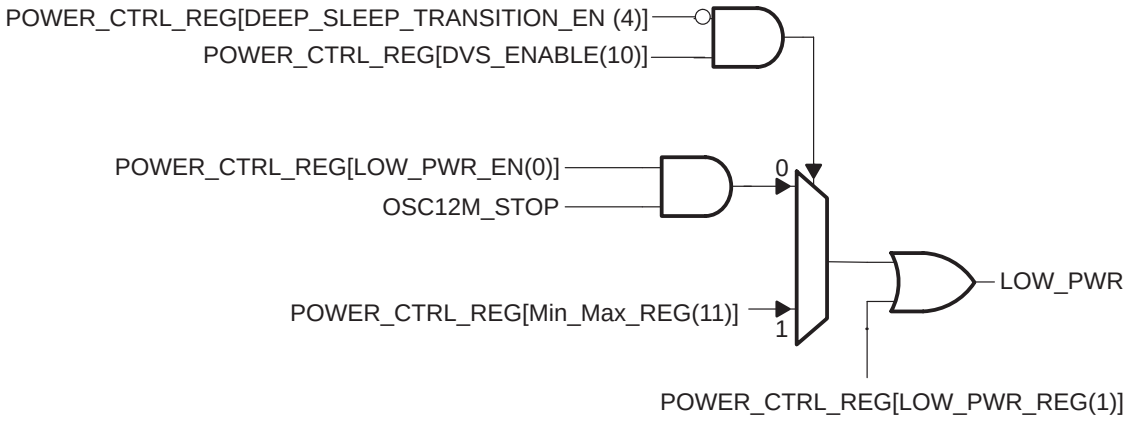
This feature allows dynamic control of the operating voltage of OMAP5912. It provides two operating points (voltage, frequency) to adapt the operating voltage to the performance requirement.

The OMAP3.2 DPLL frequency must be set accordingly before initiating the procedure.

The DVS procedure described here goes through the OMAP3.2 idle and wake-up sequences. To overcome limitations of this procedure, a more direct way to control the OMAP5912 operating voltage is provided.

The user can directly force the LOW_PWR signal to 1 and thereby force the operating voltage to 1.1 V by setting the bit POWER_CTRL_REG[1]. POWER_CTRL_REG [1] cannot force the LOW_PWR signal to 0 (see Figure 5).

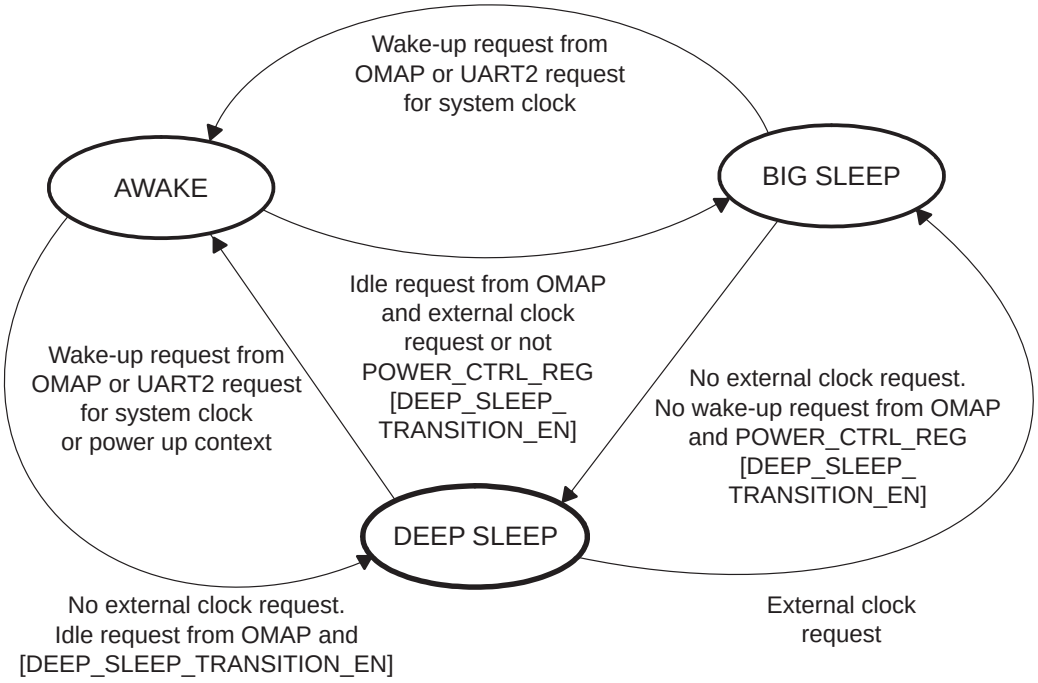
Figure 5. Control of OMAP5912 Low-Power Output by ULPD POWER_CTRL_REG



1.9 Transitions Between Power Modes

Figure 6 shows the basic functional scheme of the FSM1:

Figure 6. Simplified State Diagram of the ULPD FSM1



The ULPD handles the state transitions among deep sleep, big sleep, and awake. The transitions are triggered by the following:

- ❏ External events: Hardware resets or clock requests
- ❏ Internal events: Software resets, watchdog time-out, software clock request, or MPU idle request

Table 1. Sleep Modes versus Active Mode (Summary)

Mode	Power Dissipation	Active Clocks	Comments
Deep sleep	Lowest	32 kHz for wake-up detection	Only the UART2 functional clock using 32 kHz is active.
Big sleep	From modules clocked by active 32-kHz or 48-MHz clocks	System and 32-kHz clock. ULDP output clocks to peripherals if requested	For BCLK and MCLK, the 48-MHz frequency can be divided further by setting respectively SDW_CLK_DIV_CTRL_SEL[7:2] and COM_RATIO_SEL[7:2] in ULPD control registers.
Awake	Nominal	All 32-kHz, 48-MHz, and clocks derived from system clock can be active.	OMAP3.2 input clock is active.

1.10 Power-on Transition to Deep Sleep Mode

At power-up, namely when the power-up input signal $\overline{\text{PWRON\_RESET}}$ is asserted low, the ULPD FSM1 enters deep sleep mode.

In this case, the $\overline{\text{LOW\_PWR}}$ signal is reset to inactive state 1.

When $\overline{\text{PWRON\_RESET}}$ is released, the FSM automatically switches from deep sleep to awake.

The transition follows this procedure:

1) $\overline{\text{PWRON\_RESET}}$ is asserted low, and $\overline{\text{LOW\_PWR}}$ is reset to 1 (inactive).
2) FSM enters the deep sleep mode and stays in this mode until the $\overline{\text{PWRON\_RESET}}$ signal is released.

1.11 Transitions From Deep Sleep Mode

In deep sleep mode, the FSM monitors the OMAP3.2 wake-up request and external clock requests. OMAP3.2 propagates asynchronously unmasked peripheral interrupts to generate a wake-up request.

External clock requests and the OMAP3.2 wake-up request are respectively initiators of deep sleep to big sleep and deep sleep to awake transitions. The power-up reset initiates a transition from deep sleep to awake.

1.11.1 Transition from Deep Sleep to Big Sleep Mode

Transition to big sleep state occurs when there is at least one specific external clock request (see Table 2).

Table 2. Initiators to Deep Sleep −> Big Sleep Transition

Transition from Deep Sleep to Big Sleep	
External Clock Request	
CONF_CAM_CLKMUX_R (register)	CAMERA I/F
External Clock Request (Continued)	
CONF_MOD_UART1_CLK_MODE_R (register)	UART1
CONF_MOD_UART2_CLK_MODE_R (register)	UART2
CONF_MOD_UART3_CLK_MODE_R (register)	UART3
CONF_MOD_USB_HOST_HHC_UHOST_EN_R (register)	USB OTG
CONF_MOD_MMC_SD_CLK_REQ_R (register)	MMC/SDIO1

Table 2. Initiators to Deep Sleep −> Big Sleep Transition (Continued)

Transition from Deep Sleep to Big Sleep	
CONF_MOD_MMC_SD2_CLK_REQ_R (register)	MMC/SDIO2
Clock request by USB	USB OTG. See Note 4.
MCLKREQ	OMAP5912 input pin
BCLKREQ	OMAP5912 I/O input pin
SOFT_REQ_REG (active)	Software requests. See Note 5.
Other	
DEEP_TRANSITION_ENABLE	ULPD

Notes: 1) Software requests prevent the transition to deep sleep when leaving awake state but are not initiators of deep sleep to big sleep transition.

2) DEEP_SLEEP_TRANSITION_EN is not an initiator of deep sleep to big sleep transition. It keeps FSM1 in big sleep mode and prevents transition to deep sleep mode when leaving the awake state.

3) When in external 48-MHz clock mode (CONF_DPLL_EXT_SEL = 0), clock requests related to 48-MHz clock have no effect on the FSM.

4) The USB can request clock when an external host or device is detected or when the USB enters the resume state.

5) Software requests for clock can be programmed in the ULPD SOFT_REQ_REG (see Table 20).

The transition sequence is as follows:

1) External clock request occurs.

2) $\overline{\text{LOW\_PWR}}$ is asserted high.

3) Depending on the system input clock source mode:

   a) External mode: In this case, the setup counter, SETUP_ANALOG_CELL3, is loaded with the related setup value from the ULPD register that corresponds to the ramp-up time of the external voltage supply. When the counter underflow is generated, it globally enables the system input clock to peripherals.

      At this point it is possible that the external system input clock is not present yet. The clocks for the peripherals are effectively restarted whenever the system input clock arrives, if the corresponding clock request is active.

      This setup stage allows the supply voltage to be stable before enabling the input clocks to peripherals.

b) Oscillator mode: In this case, the setup counter, SETUP_ANALOG_CELL2, is loaded with the related setup value from the ULPD register that corresponds to the maximum time between ramp-up time of the external voltage supply and LDO stabilization time.

When the counter underflow is generated, it enables the oscillator. Then the setup counter, SETUP_ANALOG_CELL3, is loaded with the related setup value from the ULPD register that corresponds to the stabilization delay of the oscillator.

When the counter underflow is generated, it globally enables the system input clock to peripherals.

The clocks for the peripherals are restarted if the corresponding clock request is active. These two setup stages allow the supply voltage and the input system clock to be stable before enabling the input clock to peripherals.

4) FSM1 enters big sleep mode.

1.11.2 Transition from Deep Sleep to Awake Mode

Transition to awake mode occurs whenever a wake-up event occurs. Wake-up events are:

- $\overline{\text{MPU\_RST}}$
- OMAP3.2 asserts the wake-up request. A wake-up request is initiated by peripheral unmasked interrupts.
- UART2 requests system clock.
- $\overline{\text{PWRON\_RESET}}$
- RTC_ON_NOFF
- 32-kHz watchdog reset

Table 3. Initiators of Deep Sleep −> Awake Transition

Transition from Deep Sleep to Awake	
Wake-up Event	
$\overline{\text{PWRON\_RESET}}$	Power-on reset pin
RTC_ON_NOFF	Power-on reset pin
$\overline{\text{MPU\_RST}}$	System reset pin
32-kHz watchdog reset	32-kHz watchdog time-out
Wake-up request	Peripheral unmasked interrupts
PERIPH_$\overline{\text{REQ}}$	
PERIPH_$\overline{\text{REQ}}$	System clock request from UART2

The transition follows one of two sequences:

❑ Sequence 1

1) The following wake-up event occurs:
   a) OMAP3.2 asserts a wake-up request. The wake-up request is initiated by a peripheral unmasked interrupt.
   b) UART2 requests system clock.
   c) Power-up reset, system reset (low on MPU_RST), secure watchdog reset, or 32-kHz watchdog reset
2) $\overline{\text{LOW\_PWR}}$ is asserted high.
3) Depending on the system input clock source mode:
   a) External mode: In this case, the setup down counter, SETUP_ANALOG_CELL3, is loaded with the related setup value from the ULPD register that corresponds to the ramp-up time of the external voltage supply.

      When the counter underflow is generated, it globally enables the system input clock to peripherals.

      At this point, it is possible that the external system input clock is not present yet.

      The peripheral clocks are effectively restarted whenever the system input clock arrives, if the corresponding clock request is active.

      This setup stage is intended to allow the supply voltage to be stable before enabling the input clocks to peripherals.

      At power up, the supply voltage and the input system clock must be stable when the $\overline{\text{PWRON\_RESET}}$ signal is released. In this case, this setup stage is skipped.

b) Oscillator mode: In this case, the setup counter, SETUP_ANALOG_CELL2, is loaded with the related setup value from the ULPD register that corresponds to the maximum time between ramp-up time of the external voltage supply and LDO stabilization time.

When the counter underflow is generated, it enables the oscillator. Then the setup counter, SETUP_ANALOG_CELL3, is loaded with the related setup value from the ULPD register that corresponds to the stabilization delay of the oscillator.

When the counter underflow is generated, it globally enables the system input clock to peripherals.

The clocks for the peripherals are restarted if the corresponding clock request is active. These two setup stages are intended to allow the supply voltage and the input system clock to be stable before enabling the input clock to peripherals.

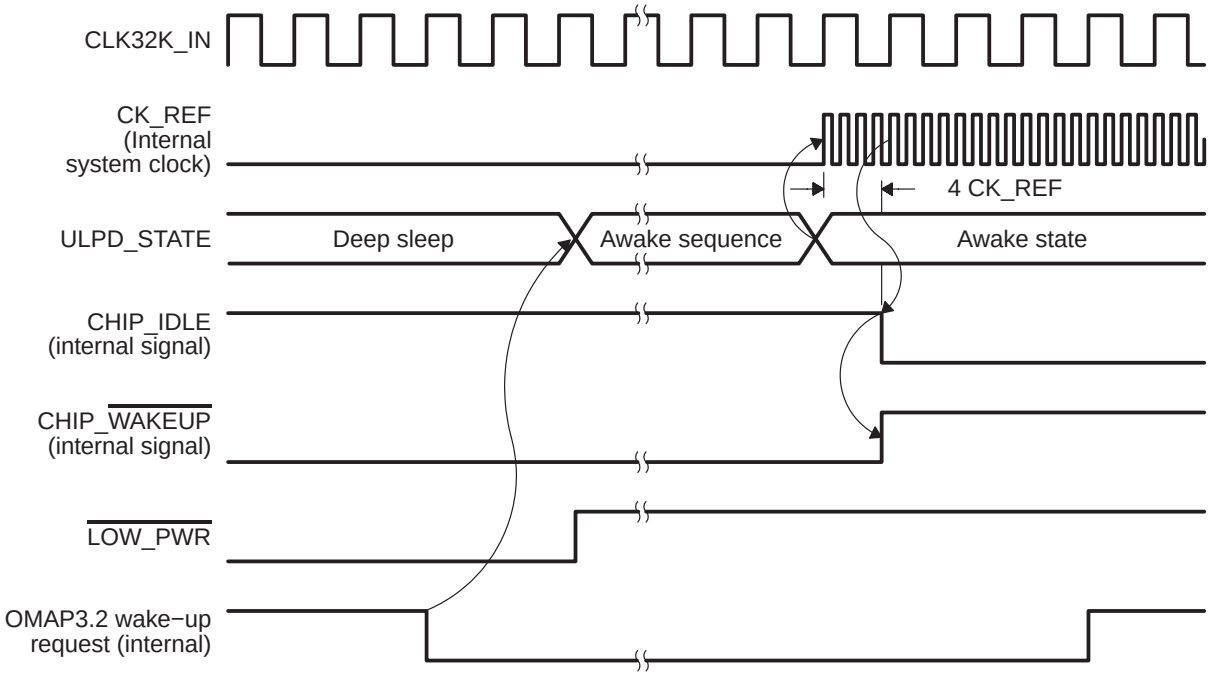
4) FSM1 enters the awake mode.

5) OMAP3.2 input clock is enabled. In external mode, the clock is effectively restarted whenever the system input clock arrives.

6) OMAP3.2 deasserts the CHIP_IDLE signal.

7) ULPD releases the CHIP_$\overline{\text{WAKEUP}}$ signal high.

Figure 7. 2OMAP3.2-Initiated Wake-up Sequence

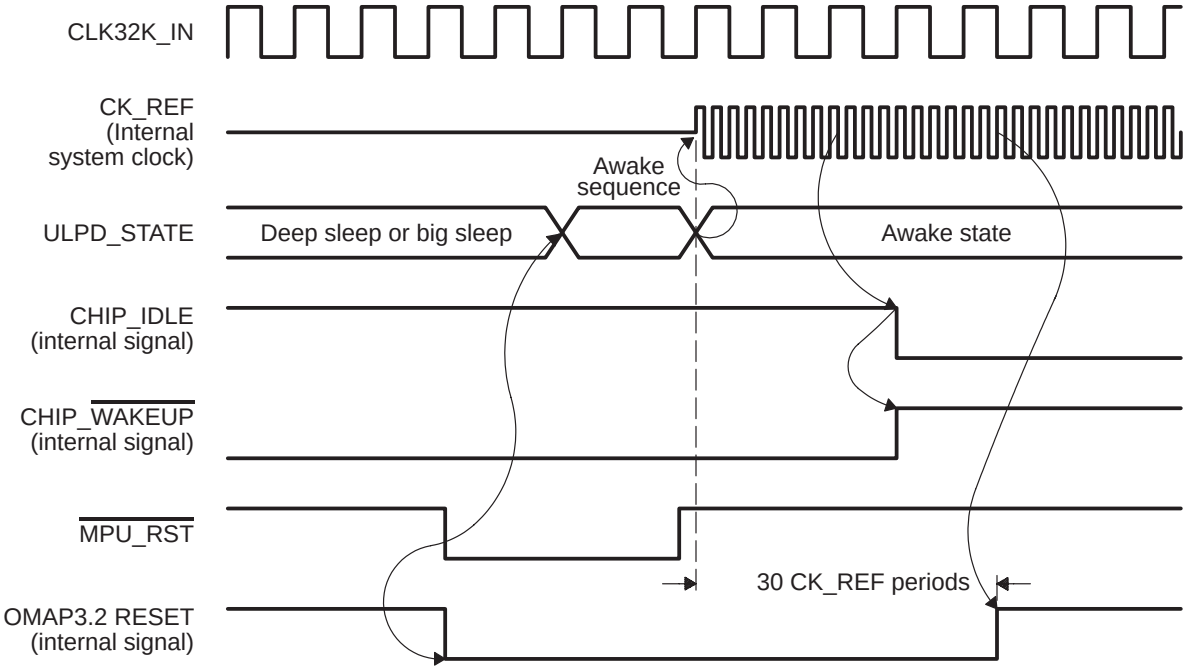


The wake-up sequence requires five CLK32K clock cycles from assertion of OMAP3.2 wake-up request to release of CK_REF.

❑ Sequence 2

1) The following wake-up events occur:
   a) $\overline{\text{MPU\_RST}}$ low event
   b) 32-kHz watchdog time-out
2) ULPD performs the transition to awake as described previously and releases the system clock to OMAP3.2.
3) As soon as the system clock is back on, the ULPD asserts low the OMAP3.2 reset for at least 30 system clock cycles.
4) On detection of OMAP3.2 reset low, OMAP3.2 deasserts the CHIP_IDLE signal.
5) ULPD asserts the CHIP_$\overline{\text{WAKEUP}}$ signal high.

Figure 8. Wake-up Sequence in Case of Warm Reset



1.12 Transitions From Big Sleep Mode

1.12.1 Transition From Big Sleep Mode to Deep Sleep Mode

Three necessary conditions lead to deep sleep mode:

- ❑ Wake-up request from OMAP is not active.
- ❑ Peripherals clock request and software clock request are inactive. Namely, every clock request that is in is not active.
- ❑ POWER_CTRL_REG[4]=1

The transition follows this sequence:

1) All clock requests are inactive; there is no wake-up request from OMAP and POWER_CTRL_REG[4]=1.
2) Input clocks to peripherals are globally disabled.
3) $\overline{\text{LOW\_PWR}}$ is activate low and in oscillator mode the oscillator is also disabled, unless POWER_CTRL_REG[9]=0.
4) FSM1 enters deep sleep mode.

1.12.2 Transition From Big Sleep Mode to Awake Mode

The transition to awake mode occurs when OMAP3.2 requests wake-up or UART2 requests system clock. An OMAP3.2 wake-up request is initiated by the unmasked interrupts of the peripherals.

The transition follows this sequence:

1) Wake-up event occurs (OMAP3.2 wake-up request or UART2 requests system clock)

2) FSM1 enters the awake mode.

3) OMAP3.2 input clock in enabled. In external mode, the clock is effectively restarted whenever the external system input clock arrives.

1.13 Transitions From Awake Mode

1.13.1 Transition From Awake Mode to Deep Sleep Mode

Three necessary conditions lead to deep sleep mode:

- ❑ OMAP3.2 asserts the CHIP_IDLE signal when no clocks are needed in OMAP3.2 or by any peripheral using OMAP3.2 output clocks. For example, DMA_LCD_CTRL.lcd_destination_port = 1 is a condition that prevents the OMAP3.2 chip_idle to be asserted because the clock request corresponding to the external LCD controller clock is kept active.
- ❑ No clock request or software request by peripherals
- ❑ POWER_CTRL_REG [4] =1

The transition follows this sequence:

1) All clock requests are inactive, CHIP_IDLE is asserted high and POWER_CTRL_REG[4]=1. Any incoming interrupt is ignored.

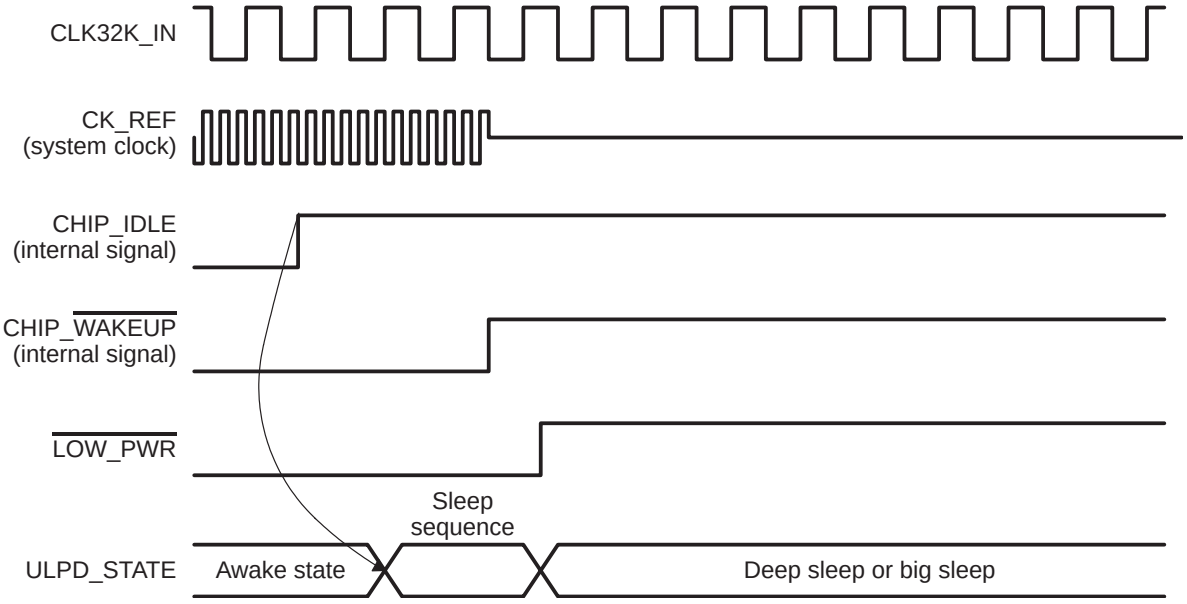
2) The ULPD asserts CHIP_$\overline{\text{WAKEUP}}$ to 0. This is a sleep acknowledge, and OMAP3.2 expects its input clock to be cut off. Interrupts can then be taken again.

3) Input clock to peripherals is globally disabled.

4) $\overline{\text{LOW\_PWR}}$ is asserted low and in oscillator mode the oscillator is also disabled, unless POWER_CTRL_REG[9]=0.

5) FSM1 enters the deep sleep mode.

Figure 9. Sleep Sequence



The sleep sequence requires three cycles of the CLK32K clock from assertion of the CHIP_IDLE signal to assertion of the LOW_PWR signal.

Note:

The name of the signal CHIP_WAKEUP may confuse users. This signal is not a wake-up signal but is an idle acknowledge.

1.13.2 Transition From Awake Mode to Big Sleep Mode

Two necessary conditions lead to big sleep mode:

- ❑ OMAP asserts the CHIP_IDLE (active high) when no clocks are needed in OMAP3.2 or by any peripherals using OMAP3.2 output clocks.
- ❑ At least one specific peripheral clock request is active or POWER_CTRL_REG[4]=0

Sequence:

1) CHIP_IDLE is asserted high. There are active external clock requests or POWER_CTRL_REG[4]=0.
2) OMAP3.2 input clock is disabled.
3) FSM1 enters the big sleep mode.

1.14 ULPD Output Clocks

The ULPD controls the system clock (12 MHz or 19.2 MHz) and the various 48-MHz clocks used by OMAP5912 peripherals.

These clocks supply the OMAP3.2 subsystem and some of the OMAP5912 peripherals.

Figure 10 shows a simplified diagram of the ULPD clock generation scheme. Most of the clocks generated by ULPD can be controlled independently by hardware and by software. In fact, a hardware clock request signal and a software clock request (programmable) bit in the SOFT_REQ_REG register are associated with most of the ULPD-generated clocks.

These clock requests are used in the ULPD to enable/disable the associated clocks.

In addition, the hardware request serves as a wake-up event for the ULPD state machine and the clock generation module it controls (the oscillator and ULPD_PLL).

The software requests are involved in the conditions of the awake to big sleep transition.

Figure 10. Basic Diagram of the ULPD Output and Input Clocks

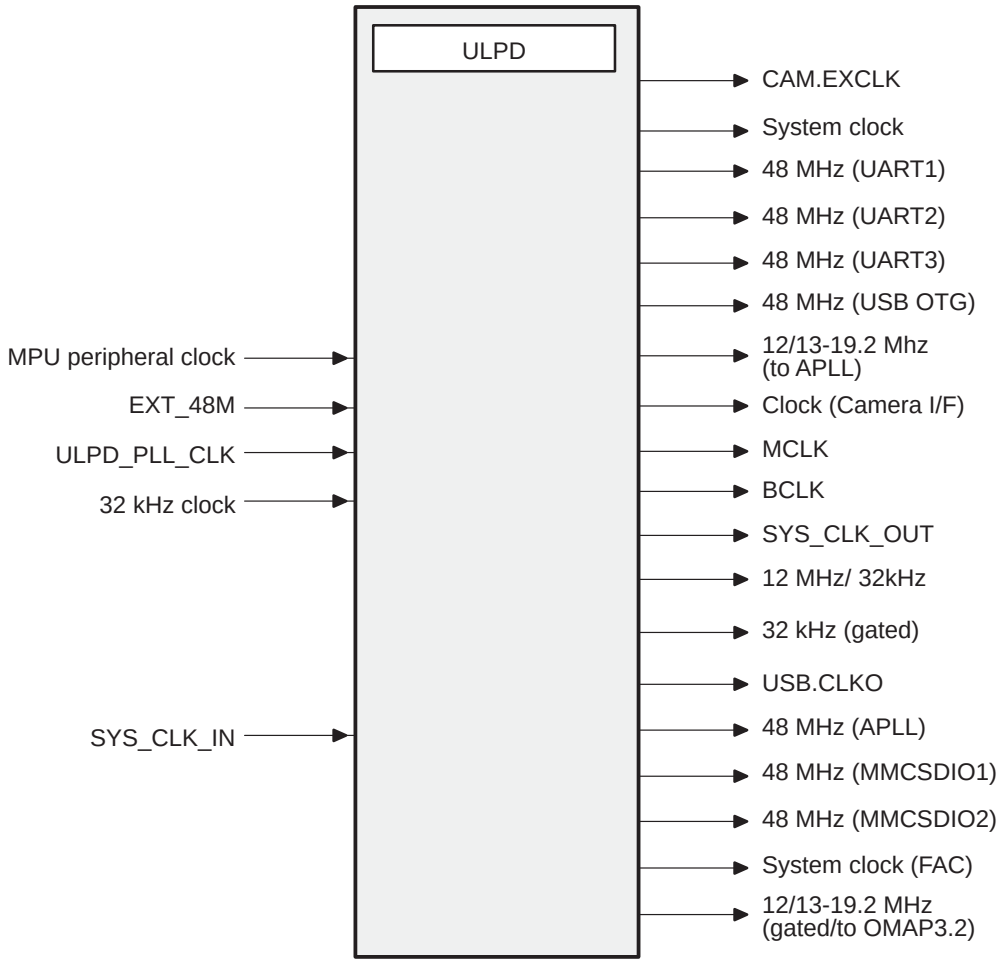


Table 4 describes how clocks are gated by the ULPD. It shows:

- The source of each clock output with its associated selection signal
- The specific enable signal and clock request whenever the clock can be gated

See Section 1.24 for more details on registers.

Table 4. ULPD Output Clocks Description

OMAP5912 I/O Destination	Clock Selected	Clock Source Selection	Wake-up Request	Soft Request Disable
BCLK	Derived from APLL CLK/2	SDW_SYSCLK_ PLLCLK_SEL CLOCK_CTRL_ REG[2] SDW_CLK_DIV_ CTRL_SEL[7:2] (see Note 1)	BCLKREQ	SOFT_DISABLE_ REQ_REG[1]
MCLK	Derived from APLL CLK/2	COM_SYSCLK_ PLLCLK_SEL CONF_MOD_COM_ MCLK_12_48_ SEL_R CONF_DPLL_EXT_ SEL (see Note 2) CLOCK_CTRL_ REQ[1] COM_RATIO_ SEL[7:2] (see Note 3)	MCLKREQ SOFT_REQ_ REG[6] COM_CLK_DIV_ CTRL_SEL[1] SOFT_REQ_ REG[1]	SOFT_DISABLE_ REQ_REG[0]
UART2	System clock	CLOCK_CTRL_ REG[0]	OMAP3.2 wake-up request UART2 request for system clock SOFT_REQ_ REG[5]	SOFT_DISABLE_ REQ_REG[3]
USB.CLKO	EXT.48M divided by 8 APLL CLK/2	CONF_DPLL_EXT_ SEL CLOCK_CTRL_ REG[4]	Clock request to APLL SOFT_REQ_ REG[0]	
SYS_CLK_OUT	System clock	CLOCK_CTRL_ REG[3]	MCLKREQ SOFT_REQ_ REG[1]	SOFT_DISABLE_ REQ_REG[0]

Table 4. ULPD Output Clocks Description (Continued)

OMAP5912 I/O Destination	Clock Selected	Clock Source Selection	Wake-up Request	Soft Request Disable
CAM.D[7] (See Note 4.)	EXT.48M APLL CLK/2	CONF_DPLL_EXT_SEL	SOFT_REQ_REG[0] Clock request to APLL	
UART1	EXT.48M APLL CLK/2	CONF_DPLL_EXT_SEL	MOD_CONF_CTRL0[29] SOFT_REQ_REG[9]	SOFT_DISABLE_REQ_REG[7]
UART2	EXT.48M APLL CLK/2	CONF_DPLL_EXT_SEL	MOD_CONF_CTRL0[30] SOFT_REQ_REG[10]	SOFT_DISABLE_REQ_REG[8]
UART3	EXT.48M APLL CLK/2	CONF_DPLL_EXT_SEL	MOD_CONF_CTRL0[31] SOFT_REQ_REG[11]	SOFT_DISABLE_REQ_REG[9]
CAMERA I/F	EXT.48M APLL CLK/2	CONF_DPLL_EXT_SEL	CONF_ CAM_CLKMUX_R SOFT_REQ_REG[11]	SOFT_DISABLE_REQ_REG[5]
FAC	MPU peripheral clock	CLOCK_CTRL_REG[5]	Clock request by the USB SOFT_REQ_REG[3]	

Table 4. ULPD Output Clocks Description (Continued)

OMAP5912 I/O Destination	Clock Selected	Clock Source Selection	Wake-up Request	Soft Request Disable
USB OTG	EXT.48M APLL CLK/2	CONF_DPLL_ EXT_SEL	USB request for 48 MHz (see Note 5) CONF_MOD_ USB_HOST_ HHC_UHOST_ EN_R SOFT_REQ_ REG[8]	SOFT_REQ_ REG[4] SOFT_DISABLE_ REQ_REG[6]
MMC/SDIO1	EXT.48M APLL CLK/2	CONF_DPLL_ EXT_SEL	MOD_CONF_ CTRL_0[23] SOFT_REQ_ REG[12]	SOFT_DISABLE_ REQ_REG[10]
MMC/SDIO2	EXT.48M APLL CLK/2	CONF_DPLL_ EXT_SEL	MOD_CONF_ CTRL_0[20] SOFT_REQ_ REG[13]	SOFT_DISABLE_ REQ_REG[11]
CAM.EXCLK	System clock System clock divided by 2	CAM_CLK_CTRL[0] CAM_CLK_CTRL[1]	OMAP3.2 wake-up request	
GPIO	System clock	CAM_CLK_CTRL[2]	OMAP3.2 wake-up request	

Notes: 1) The frequency on the BCLK can be set accordingly: SDW_CLK_DIV_CTRL_SEL[7:2]. The resulting frequency is given in the following table:

SDW_CLK_DIV_CTRL_SEL[7:2] = 0X00000, BCLK = 48 MHz
SDW_CLK_DIV_CTRL_SEL[7:2] = 0X00001, BCLK = 32 MHz
SDW_CLK_DIV_CTRL_SEL[7:2] = 0X00002, BCLK = 24 MHz
SDW_CLK_DIV_CTRL_SEL[7:2] = 0X00003, BCLK = 19.2 MHz
SDW_CLK_DIV_CTRL_SEL[7:2] = 0X00004, BCLK = 16 MHz
SDW_CLK_DIV_CTRL_SEL[7:2] = 0X00005, BCLK = 13.7 MHz
SDW_CLK_DIV_CTRL_SEL[7:2] = 0X00006, BCLK = 12 MHz
SDW_CLK_DIV_CTRL_SEL[7:2] = 0X00007, BCLK = 9.6 MHz
SDW_CLK_DIV_CTRL_SEL[7:2] = 0X00008, BCLK = 8 MHz
SDW_CLK_DIV_CTRL_SEL[7:2] = 0X00009, BCLK = 6.9 MHz
SDW_CLK_DIV_CTRL_SEL[7:2] = 0X000012, BCLK = 3 MHz
SDW_CLK_DIV_CTRL_SEL[7:2] = 0X000032, BCLK = 1 MHz

Other programmed values in SDW_CLK_DIV_CTRL_SEL[7:2] result in BCLK = 48 MHz.

2) If CONF_DPLL_EXT_SEL is set to 1, external 48 MHz clock source is selected instead of the APLL clock source. The external 48 MHz clock is provided through GPIO[14].

3) The frequency on the MCLK can be set accordingly: COM_RATIO_SEL[7:2]. The resulting frequency is given in the following table:

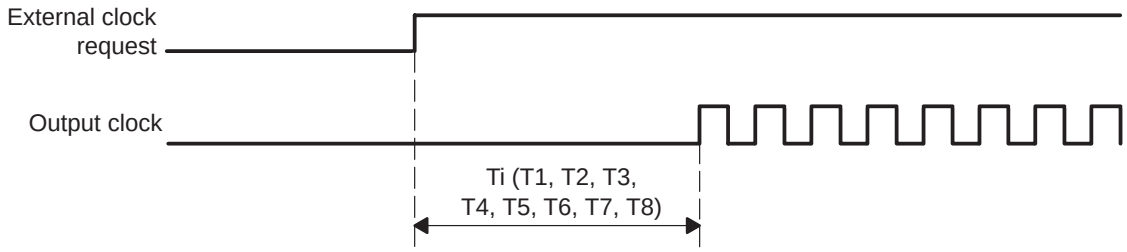
   COM_RATIO_SEL[7:2] = 0X00000, MCLK = 48 MHz
   COM_RATIO_SEL[7:2] = 0X00001, MCLK = 32 MHz
   COM_RATIO_SEL[7:2] = 0X00002, MCLK = 24 MHz
   COM_RATIO_SEL[7:2] = 0X00003, MCLK = 19.2 MHz
   COM_RATIO_SEL[7:2] = 0X00004, MCLK = 16 MHz
   COM_RATIO_SEL[7:2] = 0X00005, MCLK = 13.7 MHz
   COM_RATIO_SEL[7:2] = 0X00006, MCLK = 12 MHz
   COM_RATIO_SEL[7:2] = 0X00007, MCLK = 9.6 MHz
   COM_RATIO_SEL[7:2] = 0X00008, MCLK = 8 MHz
   COM_RATIO_SEL[7:2] = 0X00009, MCLK = 6.9 MHz
   COM_RATIO_SEL[7:2] = 0X000012, MCLK = 3 MHz
   COM_RATIO_SEL[7:2] = 0X000032, MCLK = 1 MHz

   Other programmed values in COM_RATIO_SEL[7:2] result in MCLK = 48 MHz.

4) The 48 MHz from the APLL, which can be observed on CAM.D[7], is the observability mode as configured.
5) The USB will request 48 MHz clock in case of the following event:
   − The USB has detected that either an external host or an external device is attached to one of the configured OMAP5912 USB ports, or
   − The USB exits the suspend mode and enters the resume mode.

Figure 11 shows the latency between the clock request and the clock activation.

Figure 11. Timing Diagram for Clock Request to Clock Available Latency



The parameters (T1 to T8) are the latencies between peripheral clock requests and clock available (see Table 5 and Table 6). These latencies depend on ULPD setup counters and on the FSM1 state when the request is received.

Table 5. Clock Request to Clock Available Latencies

Latency Name	Description
T1	4 x 32-kHz clock cycles + setup analog cell
T2	3 x 32-kHz clock cycles
T3	2 x 32-kHz clock cycles + setup analog cell
T4	1 x 32-kHz clock cycles
T5	1 x 32-kHz clock cycles + APLL lock time
T6	2 x 32-kHz cycles + setup analog cell + APLL lock time
T7	2 x 32-kHz clock cycles
T8	3 x 32-kHz clock cycles + setup analog cell

Table 6. Latencies for Each Peripheral Clock

Name	Wake-Up Request	Time to Get the Clock Active Depending on Initial FSM State		
		Deep Sleep	Big Sleep	Awake
CK_REF	• $\overline{\text{PWRON\_RESET}}$	T1	T2	
	• $\overline{\text{MPU\_RST}}$			
	• RTC_ON_NO$\overline{\text{FF}}$			
	• 32-kHz watchdog time-out			
	• Wake-up request			
	• UART2 requests system clock			
BCLK	• BCLKREQ	System clock		
	• SOFT_REQ_REG[2]	T3	T4	T5
	• SDW_CLK_DIV_CTRL_SEL[1]	PLL clock		
		T6	T5	T5
MCLK	• MCLKREQ	System clock		
	• SOFT_REQ_REG[6]	T3	T4	T4
	• COM_CLK_DIV_CTRL_SEL[1]	PLL clock		
	• CONF_MOD_COM_MCLK_12_48_SEL_R	T6	T5	T5

Table 6. Latencies for Each Peripheral (Continued)Clock (Continued)

Name	Wake-Up Request	Time to Get the Clock Active Depending on Initial FSM State		
		Deep Sleep	Big Sleep	Awake
Functional Clock UART2	• $\overline{\text{PWRON\_RESET}}$ • $\overline{\text{MPU\_RST}}$ • RTC_ON_NOFF • 32-kHz watchdog time-out • Wake-up request • UART2 requests system clock • SOFT_REQ_REG[5]	T8	T7	0
48-MHz USB OTG	• Any hard or soft request related to APLL • SOFT_REQ_REG[0]	T6	T5	T5
SYS_CLK_OUT	• MCLKREQ • SOFT_REQ_REG[1]	T3	T4	T4
48 MHz from APLL	• SOFT_REQ_REG[0] • Any hard or soft request related to APLL	T6	T5	T5
48-MHz UART1	• CONF_MOD_UART1_CLK_MODE_R • SOFT_REQ_REG[9]	T6	T5	T5
48-MHz UART2	• CONF_MOD_UART2_CLK_MODE_R • SOFT_REQ_REG[10]			
48-MHz UART3	• CONF_MOD_UART3_CLK_MODE_R • SOFT_REQ_REG[11]			
Clock for CAMERA IF	• CONF_CAM_CLKMUX_R • SOFT_REQ_REG[7]	T6	T5	T5
System clock for FAC	• Request for 48 MHz from the USB • SOFT_REQ_REG[3]	T6	T2	T2
48 MHz for USB OTG	• Request for 48 MHz from the USB • CONF_MOD_USB_HOST_HHC_UHOST_EN_R • SOFT_REQ_REG[8]	T6	T5	T5

Table 6. Latencies for Each Peripheral (Continued)Clock (Continued)

Name	Wake-Up Request	Time to Get the Clock Active Depending on Initial FSM State		
		Deep Sleep	Big Sleep	Awake
48 MHz for MMCSDIO1	• CONF_MOD_MMC_SD_CLK_REQ_R • SOFT_REQ_REG[12]	T6	T5	T5
48 MHz for MMCSDIO2	• CONF_MOD_MMC_SD2_CLK_REQ_R • SOFT_REQ_REG[13]	T6	T5	T5
CAM.EXCLK	• $\overline{\text{PWRON\_RESET}}$ • $\overline{\text{MPU\_RST}}$ • RTC_ON_NOFF • 32-kHz watchdog time-out • Wake-up request • UART2 requests system clock	T1	T2	
System clock for GPIO	• $\overline{\text{PWRON\_RESET}}$ • $\overline{\text{MPU\_RST}}$ • RTC_ON_NOFF • 32-kHz watchdog time-out • Wake-up request • UART2 requests system clock	T1	T2	

1.15 Power-up and Reset Management

1.15.1 Device Power up

The $\overline{\text{PWRON\_RESET}}$ signal is the power-on reset and is used to reset the ULPD 32-kHz logic and drive the input resets of OMAP3.2.

The $\overline{\text{PWRON\_RESET}}$ is resynchronized on 32 kHz to achieve a clean reset of the ULPD.

It is therefore required to maintain the power-on reset active low for a minimum of two 32-kHz clock cycles.

At power-up reset, RESET_MODE selects between the external mode and the oscillator mode.

❑ If RESET_MODE is at 0, the ULPD starts in oscillator mode. In this case, an on-chip oscillator generates the system input clock.

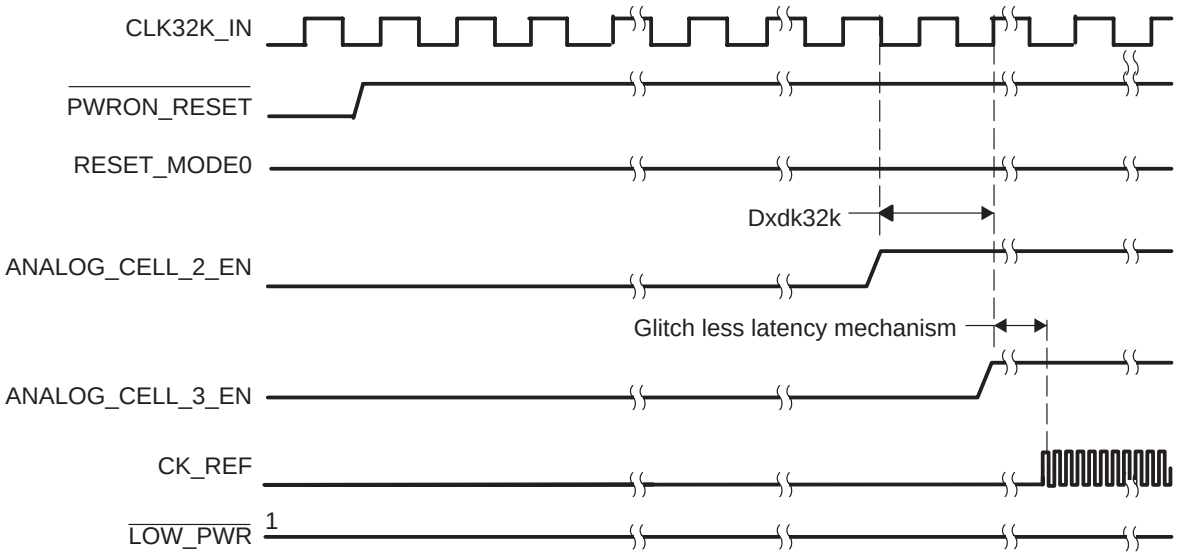
- The ULPD controls the oscillator and inserts proper setup times at power-up to ensure that a stable clock is released to the system.
- In this mode, $\overline{\text{PWRON\_RESET}}$ must be released only when the 32-kHz clock and the supply voltage are stable.
- If RESET_MODE is at 1, the ULPD starts in external clock mode. In this case, no setup time delay is inserted at power up.
- An external chip provides the input clock and ensures that it is stable.
- In external clock mode the $\overline{\text{PWRON\_RESET}}$ must be released only when the system clock, the 32-kHz clock, and the supply voltage are stable.

1.15.2 Generic Power-up Sequence in Oscillator Mode

Power-up in oscillator mode follows this sequence:

1) $\overline{\text{PWRON\_RESET}}$ is released, and the first analog cell is enabled
2) The SETUP_ANALOG_CELL2 counter starts counting in order to account for the power-supply setup time. The reset value of SETUP_ANALOG_CELL2 is 0, because the power supplies are stable before the deassertion of the power-up reset. The SETUP_ANALOG_CELL2 counter is to be used during exit of deep sleep mode while the core power supply is ramping from 1 V.
3) When SETUP_ANALOG_CELL2 underflows, the SETUP_ANALOG_CELL3 counter starts counting. SETUP_ANALOG_CELL3 is used to account for the stabilization of the 12-MHz oscillator.
4) When SETUP_ANALOG_CELL3 underflows, all the analog cells must be stable. The input system clock is released internally in ULPD.
5) The ULPD FSM1 transitions to awake mode.
6) The ULPD enables the input clock to OMAP3.2.

Figure 12. Power-up Sequence in Oscillator Mode



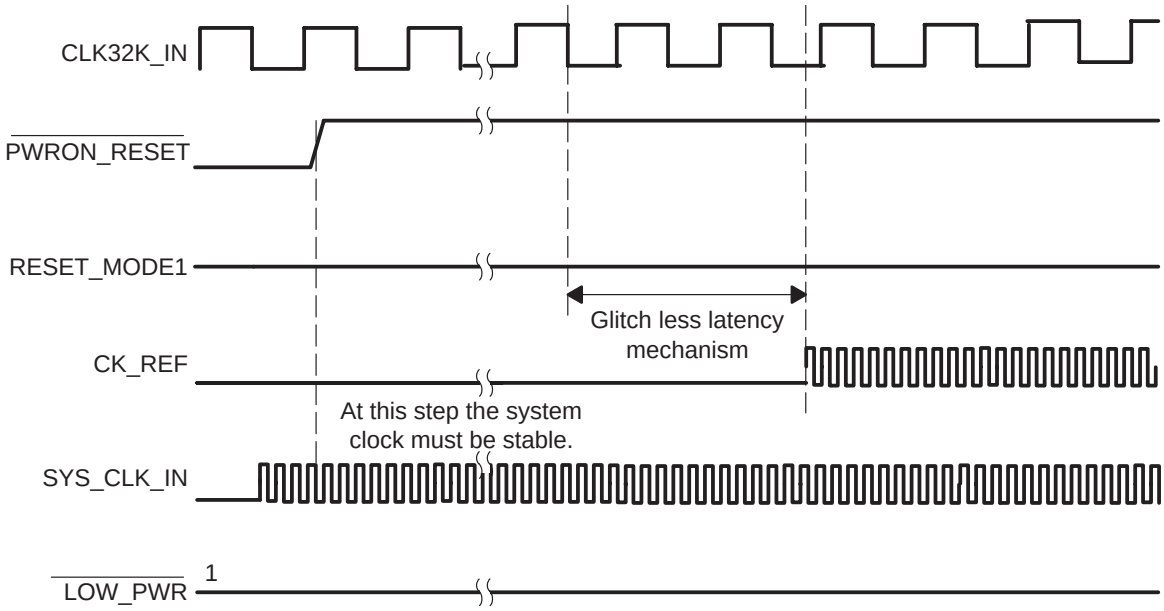
1.15.3 Power-up Sequence in External Clock Mode

In external mode, all the analog cell setup counters are bypassed at power-up reset.

It follows this sequence:

1) $\overline{\text{PWRON\_RESET}}$ is released. $\overline{\text{LOW\_PWR}}$ is reset to inactive state.
2) The input system clock is released internally in ULPD.
3) The ULPD FSM1 moves to awake mode.
4) The ULPD enables the input clock to OMAP3.2.

Figure 13. Power-up Sequence in External Clock Mode



1.16 ULPD Reset Inputs

The ULPD has five distinct reset inputs that act differently on the ULPD generated output reset.

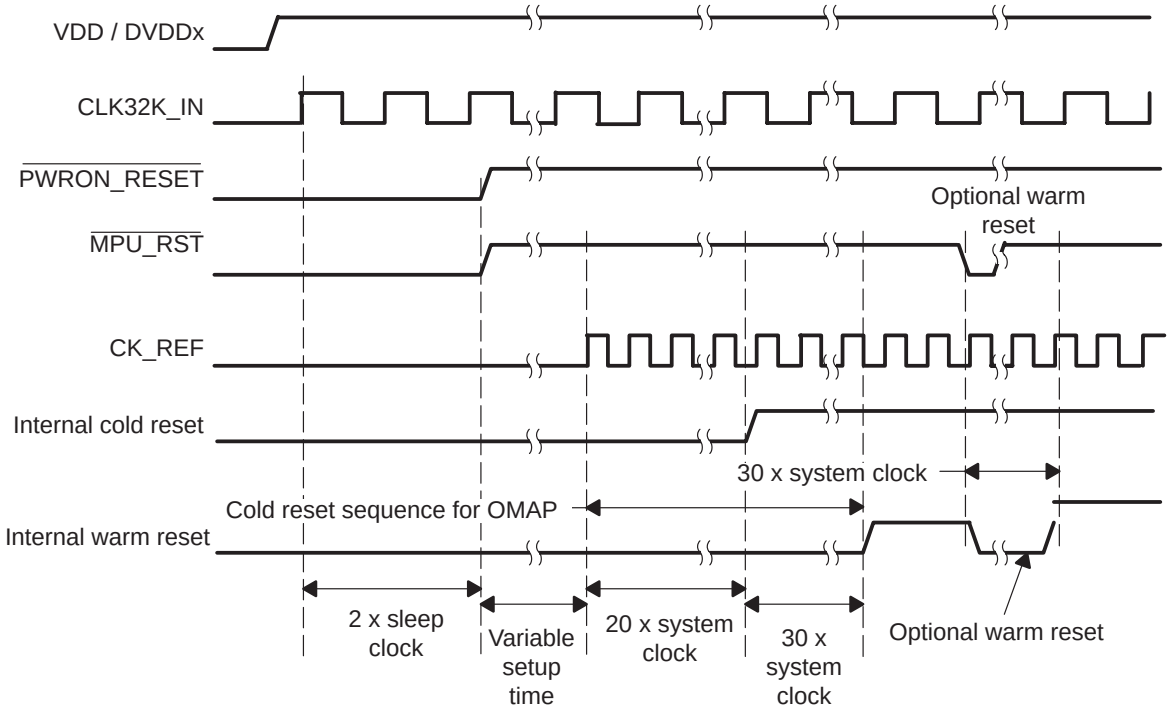
- $\overline{\text{PWRON\_RESET}}$ corresponds to the device power reset and as such must correspond to a device input pin. $\overline{\text{PWRON\_RESET}}$ is a cold reset.
- The $\overline{\text{MPU\_RST}}$ is a device global system reset and is intended to correspond to a device input pin. $\overline{\text{MPU\_RST}}$ is a warm reset.
- Security violation
- 32-kHz watchdog reset
- Secure watchdog reset

1.17 OMAP3.2 Reset Generation

The ULPD propogates the cold and warm resets to OMAP3.2.

OAMP3.2 resets the processors and peripherals. Figure 14 shows the reset procedures.

Figure 14. OMAP3.2 Input Reset Generation



1.18 OMAP3.2 Embedded LDO for DPLL[3] Control

An embedded LDO provides the DPLL of OMAP3.2 and the system clock oscillator with a quiet voltage supply.

The ULPD manages the sleep transition of this e-LDO through the LDO_SLEEP signal.

At reset, the e-LDO is not asleep.

At boot, the software can check that the e-LDO output voltage, POWER_CTRL_REG [6], is stable before switching the DPLL to lock mode.

The e-LDO can be put to sleep permanently by software with POWER_CTRL_REG [8] = 1. Before putting the e-LDO to sleep, the user must first program the DPLL into bypass mode. The device can still run out of the DPLL in bypass mode.

When POWER_CTRL_REG[7] = 1, the ULPD state machine automatically powers down the e-LDO in deep sleep mode and powers it up on exit from deep sleep mode. The ramp-up time of the e-LDO is hidden by the transition time from deep sleep mode to awake mode.

If needed, the e-LDO can be tuned by adjusting SETUP_ANALOG_CELL2 in oscillator mode or SETUP_ANALOG_CELL3 in external mode.

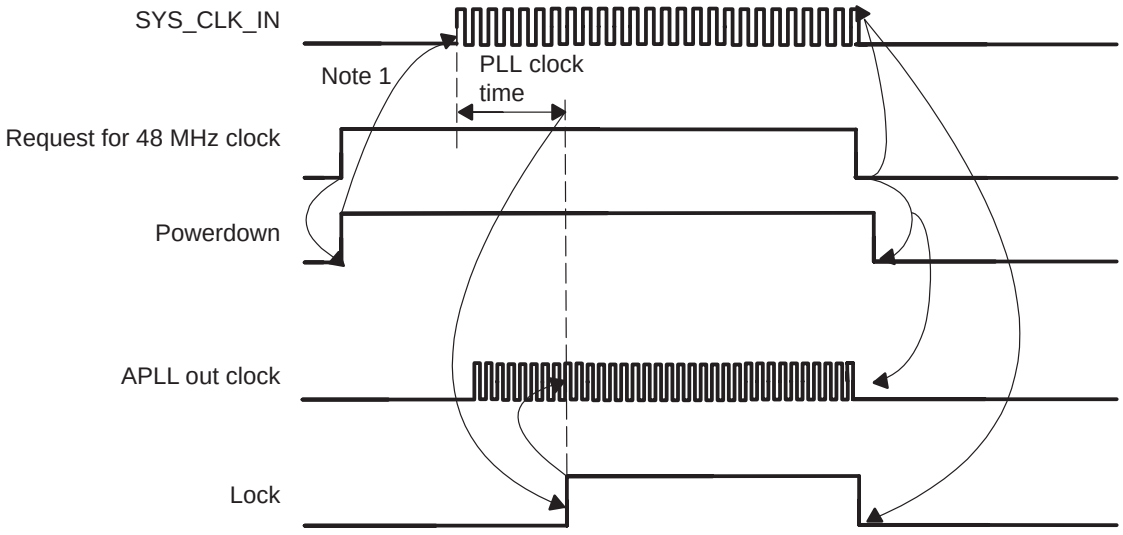
1.19 Analog Phase-Locked Loop Control

To provide a 48-MHz clock to peripherals that request it, the ULPD controls the activation and deactivation of an on-chip analog phase-locked loop (APLL). This APLL delivers a 96-MHz clock that is further divided inside the ULPD.

The ULPD enables APLL and its input clock whenever the system clock is present and a 48-MHz clock request is active.

The APLL signals to the ULPD when it reaches lock state. By reading the LOCK_STATUS bit in the ULPD_PLL_CTRL_STATUS register, the software can determine whether the 48-MHz clocks are stable.

Figure 15. ULPD_PLL Clock Management

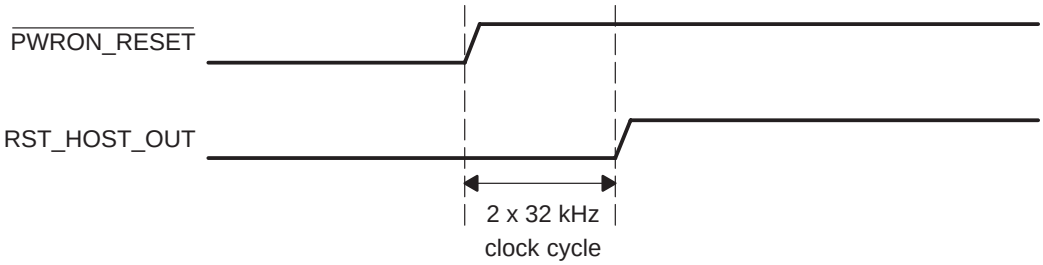


See Section 4−2 for additional information on the APLL.

1.20 Battery Failed Interrupt

After releasing $\overline{\text{PWRON\_RESET}}$, $\overline{\text{RST\_HOST\_OUT}}$ is kept low for two 32-kHz cycles, and then goes inactive high.

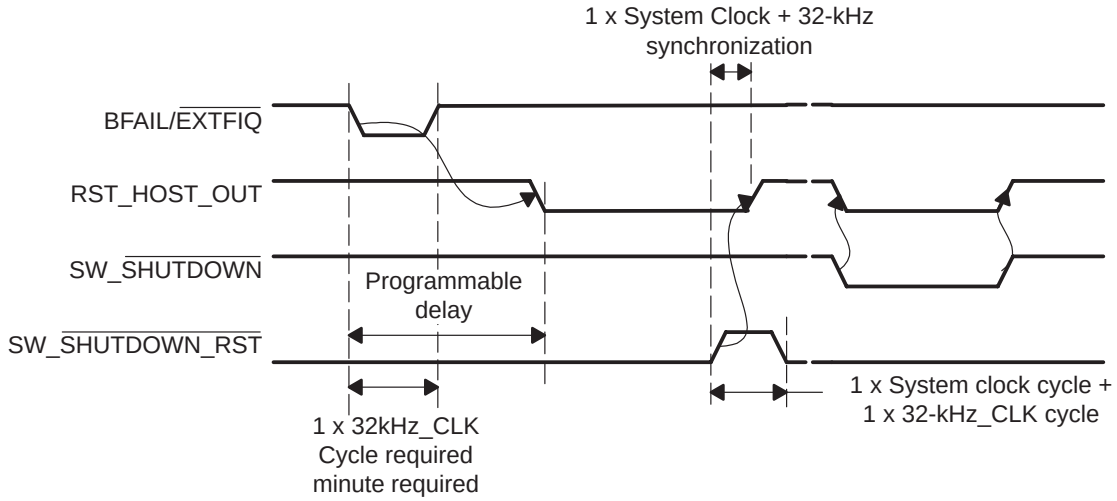
Figure 16. RST_HOST_OUT Activation on PWRON_RESET



Upon a low level on BFAIL/EXTFIQ input, which signals a battery fail event, the ULPD starts a programmable counter. When the counter underflows, ULPD asserts low RST_HOST_OUT and places OMAP3.2 in power-down mode.

The delay from the falling edge of BFAIL/EXTFIQ to the falling edge of RST_HOST_OUT is software programmable (see Table 21). The default value is one CLK32K cycle. The release of the shutdown signal is also controlled by software (see Table 28).

Figure 17. RST_HOST_OUT Activation on BFAIL/EXTFIQ and SW_SHUTDOWN



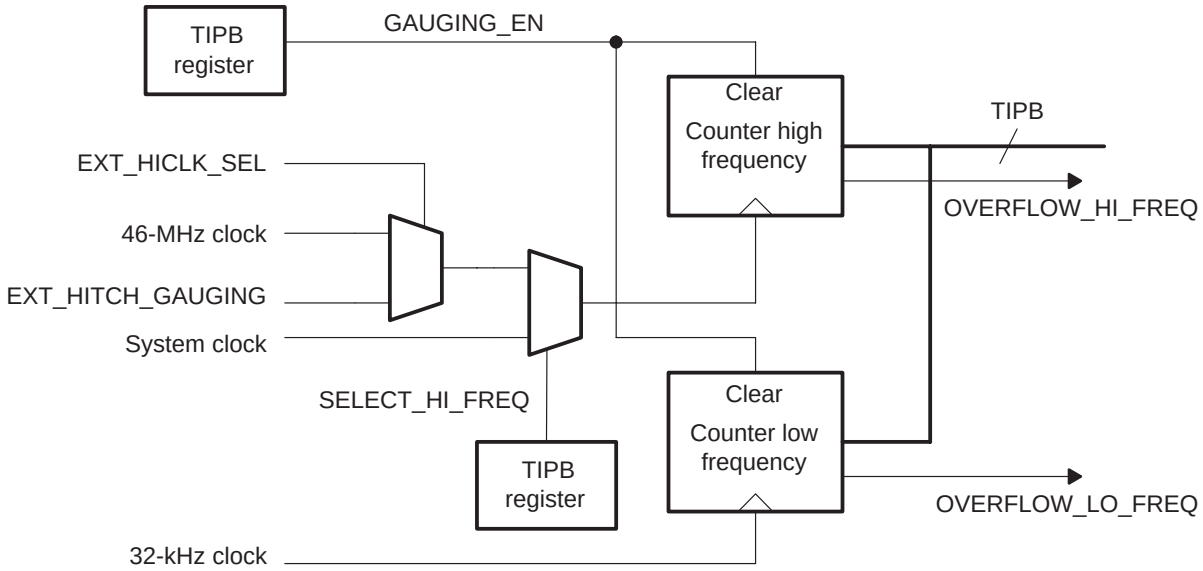
It is also possible to use SW_SHUTDOWN (bit [3] of POWER_CTRL_REG) to force low RST_HOST_OUT.

Note:

What is shown in Figure 17 as programmable delay corresponds to the counter start time (three 32-kHz cycles) plus the counter decrement to 0. The counter initial value corresponds to COUNTER_32_FIQ_REG.

1.21 32-kHz Oscillator Calibration

Figure 18. Functional Block Diagram of Gauging



Because the exact 32-kHz clock frequency is unknown, it is necessary to gauge it by comparing the 32-kHz clock with a higher frequency clock (system clock, APLL clock out, or EXT_HICLK_GAUGING) during any active period.

Note: Software Limitation

The counter is not resynchronized on the TIPB strobe. Therefore, the value is not readable while the counter is running (when gauging is enabled). The procedure is first to disable the gauging (GAUGIG_CTRLREG[0] to 0), and then to read the counter high frequency and the 32-kHz counter value.

1.22 Bad Devices

When the DEVICE_TYPE[1:0] indicates that the device is bad (DEVICE_TYPE = 01), the ULPD keeps both internal cold and warm resets asserted low. In this configuration the CPU cannot boot.

Note:

DEVICE_TYPE corresponds to EFUSE_DEVICE_TYPE.

1.23 ULPD Interrupt Generation

The ULPD generates an interrupt when one of the following events has been detected:

- An overflow occurred on the 32-kHz counter during gauging. This event is the same as the one that triggers the IT_STATUS_REG [2] bit.
- An overflow occurred on the HI_FREQ counter during gauging versus high frequency. This event is the same as the one that triggers the IT_STATUS_REG [1] bit.
- Gauging has stopped. This event is the same as the one that triggers the IT_STATUS_REG [0] bit.
- The following two conditions have been met:
  - USB_MCLK_REQ clock request has been set or SOFT_REQ_REG [3] has been active high for more than two 32-kHz clock cycles.
  - CLOCK_CTRL_REG [5] is cleared to 0.

Table 7 lists the 16-bit ULPD registers. Table 8 through Table 39 describe the register bits.

1.24 ULPD Registers

Unless otherwise specified, all of the registers in this table are reset by any warm reset (for a listing of warm resets, please see Chapter 5).

Table 7. ULPD Registers

Base Address = 0xFFFE 0800			
Name	**Description**	**R/W**	**Offset**
COUNTER_32_LSB_REG	Counter 32 LSB		0x00
COUNTER_32_MSB_REG	Counter 32 MSB		0x04
COUNTER_HIGH_FREQ_LSB_REG	Counter high-frequency LSB		0x08
COUNTER_HIGH_FREQ_MSB_REG	Counter high-frequency MSB		0x0C
GAUGING_CTRL_REG	Gauging control		0x10
IT_STATUS_REG	Interrupt status		0x14
RESERVED	Reserved		0x18
RESERVED	Reserved		0x1C
RESERVED	Reserved		0x20
SETUP_ANALOG_CELL3_ULPD1_REG	Setup analog cell3 ULPD1		0x24

Table 7. ULPD Registers (Continued)

Base Address = 0xFFFE 0800			
Name	**Description**	**R/W**	**Offset**
SETUP_ANALOG_CELL2_ULPD1_REG	Setup analog cell2 ULPD1		0x28
SETUP_ANALOG_CELL1_ULPD1_REG	Setup analog cell1 ULPD1		0x2C
CLOCK_CTRL_REG	Clock control		0x30
SOFT_REQ_REG	Software request		0x34
COUNTER_32_FIQ_REG	Counter 32 FIQ		0x38
RESERVED	Reserved		0x3C
STATUS_REQ_REG	Status request		0x40
PLL_DIV_REG	PLL division		0x44
RESERVED_48	Reserved 48		0x48
ULPD_PLL_CTRL_STATUS	ULPD PLL control status		0x4C
POWER_CTRL_REG	Power control		0x50
STATUS_REQ_REG2	Status request 2		0x54
SLEEP_STATUS	Sleep status		0x58
SETUP_ANALOG_CELL4_ULPD1_REG	Setup analog cell4 ULPD1		0x5C
SETUP_ANALOG_CELL5_ULPD1_REG	Setup analog cell5 ULPD1		0x60
SETUP_ANLOG_CELL6_ULPD1_REG	Setup analog cell6 ULPD1		0x64
SOFT_DISABLE_REQ_REG	Software disable request		0x68
RESET_STATUS	Reset status		0x6C
REVISION_NUMBER	Revision number		0x70
SDW_CLK_DIV_CTRL_SEL	SDW clock divider control select		0x74
COM_CLK_DIV_CTRL_SEL	COM clock divider control select		0x78
CAM_CLK_CTRL	CAM clock control		0x7C
SOFT_REQ_REG2	Software request 2		0x80

Table 8. Counter 32 LSB Register (COUNTER_32_LSB_REG)

Base Address = FFFE 0800, Offset = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	COUNTER_SLEEP_ CLK_LSB	Lower value of the number of sleep clock cycles during gauging time	R	0x1

Note: The CPU must check that the IT_STATUS_REG[0] bit has been asserted high before reading this register.

Table 9. Counter 32 MSB Register (COUNTER_32_MSB_REG)

Base Address = 0xFFFE 0800, Offset = 0x04				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:4	RESERVED	Reserved	R	0x0
3:0	COUNTER_32_MSB	Upper value of the number of 32-kHz clock cycles during gauging time	R	0x0

Note: The CPU must check that the IT_STATUS_REG[0] bit has been asserted high before reading this register.

Table 10. Counter High-Frequency LSB Register (COUNTER_HIGH_FREQ_LSB_REG)

Base Address = 0xFFFE 0800, Offset = 0x08				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	COUNTER_HIGH_ FREQ_LSB	Lower value of the number of high-frequency clock during gauging time	R	0x1

Note: The CPU must check that the IT_STATUS_REG[0] bit has been asserted high before reading this register.

Table 11. Counter High-Frequency MSB Register (COUNTER_HIGH_FREQ_MSB_REG)

Base Address = 0xFFFE 0800, Offset = 0x0C				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:6	UNUSED	Unused	R	0x0
5:0	COUNTER_HIGH_ FREQ_MSB	Upper value of the number of high-frequency clock during gauging time	R	0x0

Note: The CPU must check that the IT_STATUS_REG[0] bit has been asserted high before reading this register.

Table 12. Gauging Control Register (GAUGING_CTRL_REG)

Base Address = 0xFFFE 0800, Offset = 0x10				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:2	UNUSED	Unused	R/W	0x0
1	SELECT_HI_FREQ_CLOCK	1: High-frequency clock = auxiliary gauging clock 0: High-frequency clock = 12-Mhz clock	R/W	0x0
0	GAUGING_EN	1: Gauging is running. 0: Gauging is stopped.	R/W	0x0

Table 13. Interrupt Status Register (IT_STATUS_REG)

Base Address = 0xFFFE 0800, Offset = 0x14				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:4	UNUSED	Unused	R	0x0
3	IT_WAKEUP_USB	Wake-up interrupt from USB W2FC	R	0x0
2	OVERFLOW_32	An overflow occurred on the 32-kHz counter during gauging.	R	0x0
1	OVERFLOW_HI_FREQ	An overflow occurred on the hi_freq counter during gauging versus high-frequency clock.	R	0x0
0	IT_GAUGING	To inform CPU that gauging is stopped and high-and low-frequency registers can be read.	R	0x0

Table 14. Reserved Register (RESERVED)

Base Address = 0xFFFE 0800, Offset = 0x18				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	RESERVED	Reserved	R/W	0x3FF

Table 15. Reserved Register (RESERVED)

Base Address = 0xFFFE 0800, ?Offset = 0x1C				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	RESERVED	Reserved	R/W	0x3FF

Table 16. Setup Analog Cell3 ULPD1 Register (SETUP_ANALOG_CELL3_REG)

Base Address = 0xFFFE 0800, Offset = 0x24				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	SETUP_ANALOG_CELL3	Setup time of analog cell3 in number of sleep clock cycles	R/W	0x3FF

Note: This setup stage is for the FSM1 of the ULPD.

Table 17. Setup Analog Cell2 ULPD1 Register (SETUP_ANALOG_CELL2_REG)

Base Address = 0xFFFE 0800, Offset = 0x28				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	SETUP_ANALOG_CELL2	Setup time of analog cell2 in number of sleep clock cycles	R/W	0x0

Note: This setup stage is for the FSM1 of the ULPD.

Table 18. Setup Analog Cell1 ULPD1 Register (SETUP_ANALOG_CELL1_REG)

Base Address = 0xFFFE 0800, Offset = 0x2C				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	SETUP_ANALOG_CELL1	Setup time of analog cell1 in number of sleep clock cycles	R/W	0x0

Note: This setup stage is for the FSM1 of the ULPD.

Table 19. Clock Control Register (CLOCK_CTRL_REG)

Base Address = 0xFFFE 0800, Offset = 0x30				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:7	UNUSED	Unused	R/W	0x0
6	SLICER_BYPASS	Reserved	R/W	0x0
5	DIS_USB_PVCI_CLK	1: Disable USB W2FC PVCI clock. 0: Enable.	R/W	0x0
4	USB_MCLK_EN	0: Disable USB hub clock output. 1: Enable.	R/W	0x0

Table 19. Clock Control Register (CLOCK_CTRL_REG) (Continued)

Base Address = 0xFFFE 0800, Offset = 0x30				
Bit	**Name**	**Function**	**R/W**	**Reset**
3	TI_RESERVED_EN	0: Disable clock on SYS_CLK_OUT output. 1: Enable.	R/W	0x0
2	SDW_MCLK_INV	0: BLUETOOTH_CLK is low when inactive. 1: CLK is high when inactive	R/W	0x0
1	COM_MCLK_INV	0: Modem clock is low when inactive. 1: CLK is high when inactive.	R/W	0x0
0	MODEM_32K_EN	0: Disable sleep clock on UART (force to 1). 1: Enable sleep clock on UART.	R/W	0x0

Table 20. Software Request Register (SOFT_REQ_REG)

Base Address = 0xFFFE 0800, Offset = 0x34				
Bit	**Name**	**Function**	**R/W**	**Reset**
15	SOFT_CLOCK2_DPLL_REQ	PLL software request reserved for future use 1: Request active 0: Request inactive	R/W	0X0
14	SOFT_CLOCK1_DPLL_REQ	PLL software request reserved for future use. 1: Request active 0: Request inactive	R/W	0x0
13	SOFT_MMC2_DPLL_REQ	ULPD_PLL clock request for MMC2. 1: Active 0: Inactive	R/W	0x0
12	SOFT_MMC_DPLL_REQ	Software ULPD_PLL req for MMC 1: Request active 0: Request inactive	R/W	0x0
11	SOFT_UART3_DPLL_REQ	Software UART3 ULPD_PLL request. 1: Request active 0: Request inactive	R/W	0x0
10	SOFT_UART2_DPLL_REQ	Software UART2 ULPD_PLL request. 1: Request active 0: Request inactive	R/W	0x0
9	SOFT_UART1_DPLL_REQ	Software UART1 ULPD_PLL request. 1: Request active 0: Request inactive	R/W	0x0

Table 20. Software Request Register (SOFT_REQ_REG) (Continued)

Base Address = 0xFFFE 0800, Offset = 0x34				
Bit	**Name**	**Function**	**R/W**	**Reset**
8	SOFT_USB_OTG_DPLL_REQ	Software request for USB OTG for ULPD_PLL clock. 1: Request active 0: Request inactive	R/W	0x0
7	SOFT_CAM_DPLL_MCKO_REQ	Software camera ULPD_PLL request. 1: Request active 0: Request inactive	R/W	0x0
6	SOFT_COM_MCKO_REQ	Software request for COM_MCKO clock. 1: Request active 0: Request inactive	R/W	0x0
5	SOFT_PERIPH_$\overline{\text{REQ}}$	Software system clock request for UART2. 1: Request active 0: Request inactive	R/W	0x0
4	USB_REQ_EN	0: Disable USB client hardware DPLL request 1: Enable	R/W	0x1
3	SOFT_USB_REQ	Software system clock request for USB host. 1: Request active 0: Request inactive	R/W	0x0
2	SOFT_SDW_REQ	Software systm clock request for Bluetooth. 1: Request active 0: Request inactive	R/W	0x0
1	SOFT_COM_REQ	Software system clock request for com processor 1: Request active 0: Request inactive	R/W	0x0
0	SOFT_DPLL_REQ	Software ULPD_PLL clock request 1: Request active 0: Request inactive	R/W	0x0

Table 21. Counter 32 FIQ Register (COUNTER_32_FIQ_REG)

Base Address = 0xFFFE 0800, Offset = 0x38				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:8	UNUSED	Unused	R/W	0x0
7:0	COUNTER_32_FIQ	Number of 32-kHz clock cycles to delay active modem shutdown signal after receiving FIQ	R/W	0x01

Table 22. Reserved Register (RESERVED)

Base Address = 0xFFFE 0800, Offset = 0x3C				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	RESERVED	Reserved	R	0x0

Table 23. Status Request Register (STATUS_REQ_REG)

Base Address = 0xFFFE 0800, Offset = 0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15	CLOCK3_DPLL_REQ	ULPD_PLL clock request from request RESERVED3 1: Active 0: Inactive	R	Unknown
14	CLOCK2_DPLL_REQ	ULPD_PLL clock request from request RESERVED2 1: Active 0: Inactive	R	Unknown
13	CLOCK1_DPLL_REQ	ULPD_PLL clock request from request RESERVED1 1: Active 0: Inactive	R	Unknown
12	MMC_DPLL_REQ	ULPD_PLL clock request from MMC 1: Request active 0: Request inactive	R	Unknown
11	UART3_DPLL_REQ	ULPD_PLL clock request from UART3 1: Request active 0: Request inactive	R	Unknown
10	UART2_DPLL_REQ	ULPD_PLL clock request from UART2 1: Request active 0: Request inactive	R	Unknown
9	UART1_DPLL_REQ	ULPD_PLL clock request from UART1 1: Request active 0: Request inactive	R	Unknown
8	USB_HOST_DPLL_REQ	ULPD_PLL clock request from USB host 1: Request active 0: Request inactive	R	Unknown
7	CAM_DPLL_MCLK_REQ	ULPD_PLL clock request from camera interface 1: Request active 0: Request inactive	R	Unknown

Table 23. Status Request Register (STATUS_REQ_REG) (Continued)

Base Address = 0xFFFE 0800, Offset = 0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
6	USB_DPLL_MCLK_REQ	ULPD_PLL clock request from USB client 1: Request active 0: Request inactive	R	Unknown
5	USB_MCLK_REQ	Hardware system clock request by USB client 1: Request active 0: Request inactive	R	Unknown
4	SDW_MCLK_REQ	Hardware system clock request from Bluetooth 1: Request active 0: Request inactive	R	Unknown
3	COM_MCLK_REQ	System clock request from com processor 1: Request active 0: Request inactive	R	Unknown
2	PERIPH_$\overline{\text{REQ}}$	System clock request from UART2 0: Request active 1: Request inactive	R	Unknown
1	WAKEUP_$\overline{\text{REQ}}$	System clock request from OMAP 0: Request active 1: Request inactive	R	Unknown
0	CHIP_IDLE	Sleep request received from OMAP 1: Request active 0: Request inactive	R	Unknown

Note: Bits 15:3 in this register reflect the state of the clock request regardless of whether they are masked or not (by the corresponding SOFT_DISABLE_REQ_REG bit).

Table 24. PLL Division Register (PLL_DIV_REG)

Base Address = 0xFFFE 0800, Offset = 0x44				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	PLL_DIV_FACTOR	PLL clk out division factor. Bypassed if programmed value is 0, else: 0001h: divided by 2 0002h: divided by 4 0004h: divided by 8 .... 8000h: divided by 65536	R/W	0x0

Table 25. Reserved Register (RESERVED_48)

Base Address = 0xFFFE 0800, Offset = 0x48				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	RESERVED	Kept for software compatibility reason. Has no effect on ULPD behavior.	R/W	0x960

Table 26. ULPD PLL Control Status Register (ULPD_PLL_CTRL_STATUS)

Base Address = 0xFFFE 0800, Offset = 0x4C				
Bit	**Name**	**Function**	**R/W**	**Reset**
15	LOCK_STATUS	Gives the lock status of the module that provides the high frequency clock 1: PLL locked 0: PLL unlocked	R	Unknown
14:3	PLL_CTRL_RES	Reserved	R/W	0x0
2:0	PLL_CONTROL	Selects the APLL mode. 000: 19.2 MHz 010: 13 MHz 011: 12 MHz	R/W	0x3

Table 27. Power Control Register (POWER_CTRL_REG)

Base Address = 0xFFFE 0800, Offset = 0x50				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:13	UNUSED	Unused	R	0x0
12	ISOLATION_CONTROL	0: Electrical isolation inactive 1: Electrical isolation active Reset of this bit is done on powerup reset only.@@@@	R/W	0x0
11	MIN_MAX_REG	1: Operation at minimum voltage 0: Operation at nominal voltage	R/W	0x1
10	DVS_ENABLE	0: DVS feature disabled 1: DVS feature enabled	R/W	0x0
9	OSC_STOP_EN	0: The oscillator is not stopped in deep sleep mode. 1: The oscillator is stopped in deep sleep mode.	R/W	0x1

Table 27. Power Control Register (POWER_CTRL_REG) (Continued)

Base Address = 0xFFFE 0800, Offset = 0x50				
Bit	**Name**	**Function**	**R/W**	**Reset**
8	SOFT_LDO_SLEEP	Control the sleep of the e-LDO that supplies the DPLL 1=> LDO sleep is forced to active state. 0=> LDO not in sleep (except in deep sleep mode if LDO_CTRL_EN=1)	R/W	0x0
7	LDO_CTRL_EN	0: The sleep of the LDO is fully controlled by SOFT_LDO_SLEEP bit. 1: The LDO is powered down in deep sleep mode OR if SOFT_LDO_SLEEP=1.	R/W	0x0
6	LDO_STEADY	Stability of LDO output voltage status 0: LDO output voltage not stable 1: LDO output voltage stable	R	Unknown
5	UNUSED	Unused	R	0x0
4	DEEP_SLEEP_TRANSITION_EN	1: Transition to deep sleep mode allowed 0: Transition to deep sleep mode forbidden	R/W	0x1
3	SW_$\overline{\text{SHUTDOWN}}$	Software generation of RST_HOST_OUT 0: RST_HOST_OUT forced to low 1: RST_HOST_OUT not forced to low	R/W	0x1
2	SW_$\overline{\text{SHUTDOWN}}$_RST	Allow to deactivate the output RST_HOST_OUT when at 1	R/W	0x0
1	LOW_PWR_REQ	Low-power software request. 0: Does not force the LOW_PWR signal to active state 1: Forces LOW_PWR signal to active state	R/W	0x0
0	LOW_PWR_EN	0: Low-power feature is disabled. 1: Low-power feature is available.	R/W	0x0

Table 28. Status Request Register 2 (STATUS_REQ_REG2)

Base Address =0xFFFE 0800, Offset = 0x54				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:1	UNUSED	Unused	R	0x0
0	MMC2_DPLL_REQ	Status of the MMC2_PLL_REQ 0: Inactive 1: Active	R	Unknown

Note: Bit 0 in this register reflects the state of the clock request regardless of whether it is masked or not (by the corresponding SOFT_DISABLE_REQ_REG bit).

Table 29. Sleep Status Register (SLEEP_STATUS)

Base Address = 0xFFFE 0800, Offset = 0x58				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:2	UNUSED	Unused	R	0x0
1	BIG_SLEEP	This bit is asserted (1) when waking up from big sleep. This bit is cleared (0) when the FSM goes out of awake.	R	Unknown
0	DEEP_SLEEP	This bit is asserted (1) when waking up from deep sleep. This bit is cleared (0) when the FSM goes out of awake.	R	Unknown

Note: Both bits can be read at 1 in case ULPD goes from deep sleep to big sleep before the transition back to awake.

Table 30. Setup Analog Cell4 ULPD1 Register (SETUP_ANALOG_CELL4_REG)

Base Address = 0xFFFE 0800, Offset = 0x5C				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	SETUP_ANALOG_CELL4	Setup time of analog cell4 in number of sleep clock cycles	R/W	0x0

Note: This setup stage is for the FSM1 of the ULPD.

Table 31. Setup Analog Cell5 ULPD1 Register (SETUP_ANALOG_CELL5_REG)

Base Address = 0xFFFE 0800, Offset = 0x60				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	SETUP_ANALOG_CELL5	Setup time of analog cell5 in number of sleep clock cycles	R/W	0x0

Note: This setup stage is for the FSM1 of the ULPD.

Table 32. Setup Analog Cell6 ULPD1 Register (SETUP_ANALOG_CELL6_REG)

Base Address = 0xFFFE 0800, Offset = 0x64				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	SETUP_ANALOG_CELL6	Setup time of analog cell6 in number of sleep clock cycles	R/W	0x0

Note: This setup stage is for the FSM1 of the ULPD.

Table 33. Software Disable Request Register (SOFT_DISABLE_REQ_REG)

Base Address =0xFFFE 0800, Offset = 0x68				
Bit	**Name**	**Function**	**R/W**	**Reset**
15	UNUSED	Unused	R	0x0
14	DIS_CLOCK3_DPLL_REQ	Disable for the PLL hardware request reserved (CLOCK3_DPLL_REQ). 0: Not disabled 1: Disabled	R/W	0x0
13	DIS_CLOCK2_DPLL_REQ	Disable for the PLL hardware request reserved (CLOCK2_DPLL_REQ). 0: Not disabled 1: Disabled	R/W	0x0
12	DIS_CLOCK1_DPLL_REQ	Disable for the PLL hardware request reserved (CLOCK1_DPLL_REQ). 0: Not disabled 1: Disabled	R/W	0x0
11	DIS_MMC2_DPLL_REQ	Disable for hardware request MMC2_DPLL_REQ. 0: Not disabled 1: Disabled	R/W	0x0
10	DIS_MMC_DPLL_REQ	Disable the current hardware request if active 0: Not disabled 1: Disabled	R/W	0x0
9	DIS_UART3_DPLL_REQ	Disable hardware request for UART3. 0: Not disabled 1: Disabled	R/W	0x0
8	DIS_UART2_DPLL_REQ	Disable UART2 PLL hardware request. 0: Not disabled 1: Disabled	R/W	0x0

Table 33. Software Disable Request Register (SOFT_DISABLE_REQ_REG) (Continued)

Base Address =0xFFFE 0800, Offset = 0x68				
Bit	**Name**	**Function**	**R/W**	**Reset**
7	DIS_UART1_DPLL_REQ	Disable UART1 PLL hardware request 0: Not disabled 1: Disabled	R/W	0x0
6	DIS_USB_HOST_DPLL_REQ	Disable the USB host system clock hardware 0: Not disabled 1: Disabled	R/W	0x0
5	DIS_CAM_DPLL_MCLK_REQ	Disable hardware CAM_PLL_MCLK_REQ. 0: Not disabled 1: Disabled	R/W	0x0
4	UNUSED	Unused	R/W	0x0
3	DIS_PERIPH_REQ	Disable hardware PERIPH_REQ request. 0: Not disabled 1: Disabled	R/W	0x0
2	UNUSED	Unused	R/W	0x0
1	DIS_SDW_MCLK_REQ	Disable SDW_MCLK_REQ if active 0: Not disabled 1: Disabled	R/W	0x0
0	DIS_COM_MCLK_REQ	Disable COM_MCLK_REQ and COM_MCKO_SEL if active 0: Not disabled 1: Disabled	R/W	0x0

Table 34. Reset Status Register (RESET_STATUS)

Base Address = 0xFFFE 0800, Offset = 0x6C				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:4	UNUSED	Unused	R	0x0
3	32K watchdog time-out	Whenever a 32-kHz watchdog time-out event occurs, this bit is asserted. The user clears this bit by writing a 0. TIPB reset has no effect on this bit. 0: 32-kHz watchdog time-out has not occurred since last PWRON_RESET (or user has cleared this bit). 1: 32-kHz watchdog time-out has occurred since last PWRON_RESET (and since last time user has cleared this bit).	R/W	0x0

Table 34. Reset Status Register (RESET_STATUS) (Continued)

Base Address = 0xFFFE 0800, Offset = 0x6C				
Bit	**Name**	**Function**	**R/W**	**Reset**
2	Security Violation	Whenever a security violation event occurs, this bit is asserted.The user clears this bit by writing a 0. TIPB reset has no effect on this bit. 0: Security violation has not occurred since last PWRON_RESET (or user has cleared this bit). 1: Security violation has occurred since last PWRON_RESET (and since last time user has cleared this bit).	R/W	0x0
1	Secure watchdog time-out	Whenever a secure watchdog time-out event occurs, this bit is asserted.The user clears this bit by writing a 0. TIPB reset has no effect on this bit. 0: Secure watchdog time-out has not occurred since last PWRON_RESET (or user has cleared this bit). 1: Secure watchdog time-out has occurred since last PWRON_RESET (and since last time user has cleared this bit).	R/W	0x0
0	POWER_ON_RESET	Whenever a $\overline{\text{PWRON\_RESET}}$ reset occurs, this bit is asserted.The user clears this bit by writing a 0 TIPB reset has no effect on this bit. 0: $\overline{\text{PWRON\_RESET}}$ not reset (or user has cleared this bit) 1: $\overline{\text{PWRON\_RESET}}$ reset	R/W	0x1

Note: This register is reset by $\overline{\text{PWRON\_RESET}}$ and not by the TIPB reset.

Table 35. Revision Number Register (REVISION_NUMBER)

Base Address = 0xFFFE 0800, Offset = 0x70				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:8	UNUSED	Unused	R	0x0
7:0	REVISION_NUMBER	Indicates the revision number of the ULPD. The revision number must be read as follows. [7..4].[3..0]. For example, 0x14 must be read like 1.4 revision.	R	Unknown

Note: The 4 LSBs indicate a minor revision, and the 4 MSBs indicate a major revision.

Table 36. SDW Clock Divider Control Select Register (SDW_CLK_DIV_CTRL_SEL)

Base Address = 0xFFFE 0800, Offset = 0x74				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:8	UNUSED	Unused	R	0x0
7:2	SDW_RATIO_SEL	Select the divider ratio to apply to the APLL output clock to generate BCLK. 000000=>1; 000001=>1.5; 000010=>2; 000011=>2.5 000100=>3; 000101=>3.5; 000110=>4; 000111=>5; 001000=>6; 001001=>7; .... 010010=>16 ... 110010=>48.	R/W	0x0
1	SDW_ULPD_PLL_CLK_REQ	BCLK clock software request. 0: Request inactive 1: Request active	R/W	0x0
0	SDW_SYSCLK_PLLCLK_SEL	0: Select the divided version of APLL output clock for BCLK 1: Select SYSTEM_CLOCK for BCLK	R/W	0x1

Table 37. COM Clock Divider Control Select Register (COM_CLK_DIV_CTRL_SEL)

Base Address = 0xFFFE 0800, Offset = 0x78				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:8	UNUSED	Unused	R	0x0
7:2	COM_RATIO_SEL	Select the divider ratio to apply to the APLL output clock to generate MCLK. 000000=> 1 000001=>1.5; 000010=>2; 000011=>2.5 000100=> 3; 000101=>3.5 000110=>4; 000111=>5 001000=>6 001001=>7 010010=>16 110010=>48	R/W	0x0

Table 37. COM Clock Divider Control Select Register (COM_CLK_DIV_CTRL_SEL) (Continued)

Base Address = 0xFFFE 0800, Offset = 0x78				
Bit	**Name**	**Function**	**R/W**	**Reset**
1	COM_ULPD_PLL_CLK_REQ	MCLK clock software request 0: Request inactive 1: Request active	R/W	0x0
0	COM_SYSCLK_PLLCLK_SEL	0: Select the divided version of APLL output clock for MCLK 1: Select SYSTEM_CLOCK for MCLK or 48 MHz	R/W	0x1

Table 38. CAM Clock Control Register (CAM_CLK_CTRL)

Base Address = 0xFFFE 0800, Offset = 0x7C				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:3	UNUSED	Unused	R	0x0
2	SYSTEM_CLK_EN	Clock enable of the system clock for GPIO modules 0: Clock disabled 1: Clock enabled	R/W	0x0
1	CAM_CLK_DIV	When 0, the CAM.CLKOUT is the system clock. When 1, the CAM.CLKOUT is the system clock.	R/W	0x0
0	CAM_CLOCK_EN	Enable of the CAM.CLKOUT. When 0, the CAM.CLKOUT is off. When 1, the CAM.CLKOUT is on.	R/W	0x0

Table 39. Software Request Register2 (SOFT_REQ_REG2)

Base Address = 0xFFFE 0800, Offset = 0x80				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:1	UNUSED	Unused	R	0x0
0	SOFT_CLOCK3_DPLL_REQ	PLL software request reserved for future use. 1: Request active 0: Request inactive	R/W	0X0

2 Power System Overview

This section provides guidelines for optimizing and reducing the overall device power dissipation.

For a global perspective, see Chapter 4, *Clocks*, and Chapter 5, *Initialization*.

Power consumption has two components:

- Dynamic power consumption
- Static power consumption because of leakage currents

To minimize dynamic power consumption, one of the following must be done:

- Minimize the activity in the circuit by enabling the automatic clock gating or by switching off some unused resources (MGS3/DSP or other module)
- Adjust the clock frequency on each clock domain according to the required MIPS or required bandwidth
- Minimize the external voltage supply (dynamic voltage scaling) if the application does not require the maximum clock frequencies

To minimize the static power consumption or leakage currents when the circuit is in stand-by mode, minimize the external voltage supply once the circuit is in deep sleep mode.

The OMAP5912 architecture implements hardware features so that various power management strategies can be enacted.

2.1 Power Domains

Table 40 lists all OMAP5912 power supplies with their respective nominal value and power domains.

Figure 19 shows how those power domains are connected or can be isolated from one another.

Table 40. Power Domains With Associated Power Supply and Planes

Power Domains	Power Supply	Power Planes
MPU DOMAIN	CVDD (1.05 V−1.65 V)	Core: ASIC gates
	CVDD1 (1.05 V−1.65 V)	Core: Arm926EJS
	CVDD2 (1.05 V−1.65 V)	Core: DPLL
	LDO.FILTER	Core: Analog
	CVDDa (1.6 V)	Core: SDRAM interface
	CVDDDLL (1.6 V)	Display interface
	DVDD1 (1.8 V−2.75 V)	USB interface
	DVDD2 (1.8 V, 3.3 V)[1]	Miscellaneous interfaces
	DVDD3 (1.8 V−3 V)	Peripheral: SDRAM interface
	DVDD4 (1.8 V−2.75 V)[2]	Peripheral: flash interface
	DVDD5 (1.8 V−2.75 V)	MMC interface
	DVDD6 (1.8 V−2.75 V)	Miscellaneous interfaces
	DVDD7 (1.8 V− 2.75 V)	Camera interface
	DVDD (1.8 V−3 V)	Miscellaneous interfaces
	DVDD9 (1.8 V−2.75 V)	Peripheral: RTC I/O
	DVDDRTC (1.8 V−2.75 V)	
DSP DOMAIN	CVDD3 (1.05 V−1.65 V)	Core: DSP
RTC DOMAIN	CVDDRTC (1.6 V)	Core: RTC

Notes: 1) If the integrated USB transceivers are used, the spec for DVDD2 should be nominal 3.3V. If the integrated USB transceivers are not intented to be used, the spec for DVV2 should be either nominal 3.3V or nominal 1.8V

Notes: 2) The SDRAM interface voltage includes 3.3V. You can power DVDD4 in the 3.0~3.6 power range. When using 3.3V power, CONF_VOLTAGE_SDRAM_R bit should be set to 1 (2.75V) for better performance.

Notes: 3) Refer to the *OMAP5912 Data Manual (SPRS231)* for more information on nominal voltage min/max ranges.

OMAP 5912 can be split into power planes and power domains:

- A power plane is composed of components that are supplied by a dedicated power rail. Components from two different power plans can be interconnected (bus interface, peripheral interfaces, and so on.). This interconnection creates leakage current between power planes.
- A power domain is composed of one or several power planes. Components from two different power domains can be interconnected (bus interface, peripheral interfaces, and so on). This interconnection

through isolation layer or level-shifter controls can be electrically cut. In this case, we reduce sensibly leakage current between power domains.

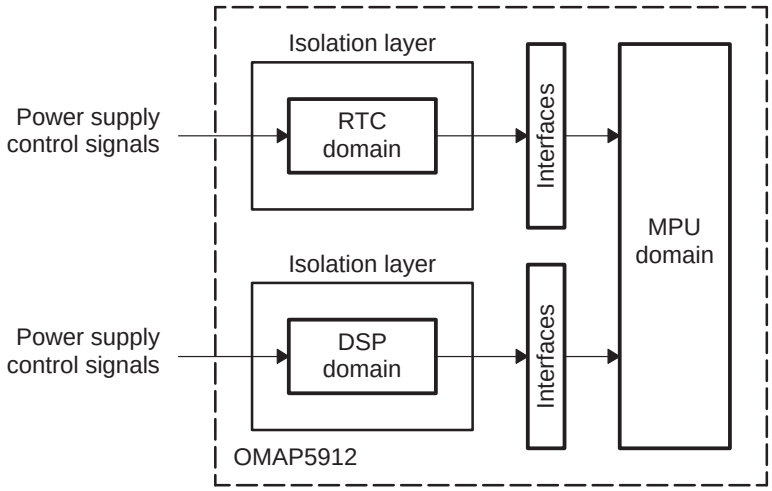
OMAP5912 is divided into three power domains:

- MPU
- DSP
- RTC

Power domains allow users to reduce power during specific system phases (idle phases). This reduction is achieved by switching off one or several domains and allowing others to be powered.

Figure 19 describes the OMAP5912 power domains.

Figure 19. OMAP5912 Power Domains



The different OMAP5912 power domains (RTC, MPU, and DSP) can be powered as shown in Table 41.

Table 41. Powering OMAP5912 Domains

	RTC	MPU	DSP
State 1	V†	V	V
State 2	V	V	0 V
State 3	V	0 V	0 V

† V = 1.5 V or 1.1 V.

In any given state, V is the same voltage for all the domains.

2.2 Clock Domain

OMAP5912 can be split into clock domains and subdomains.

As shown in Figure 20, OMAP5912 is divided into four clock domains.

The RTC clock domain is totally independent and can be isolated by hardware from other clock domains.

The clock and reset management (CLKRST) module manages clocks for the MPU, DSP and traffic controller. For each clock domain, it is possible to adjust the frequency for each clock.

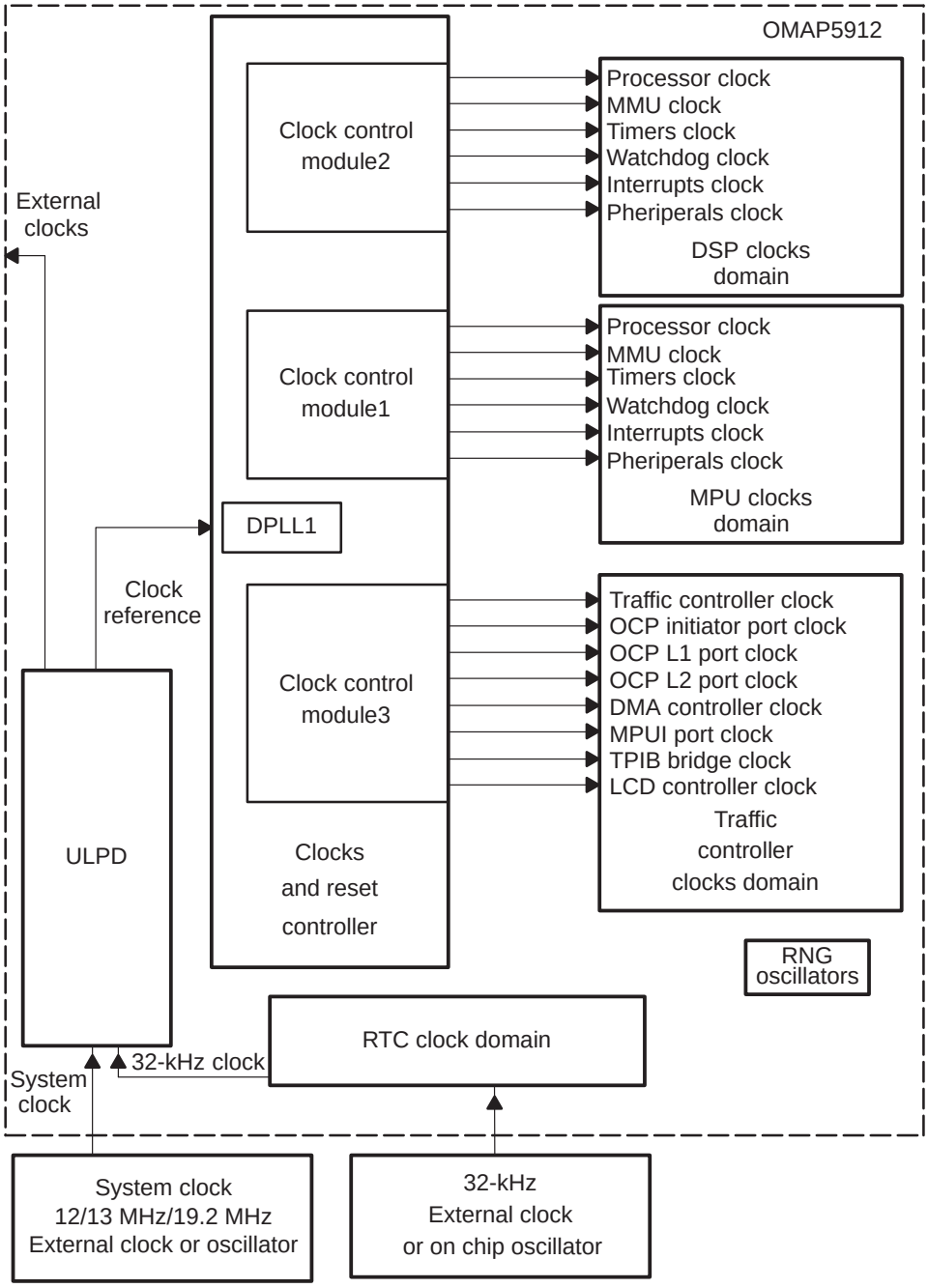
The DPLL1 output frequency is programmable and can be further divided down to provide specific clock values to each domain.

Each domain is further subdivided into subdomains that can be independently activated or deactivated.

This flexible architecture allows different power-saving mechanisms for different users.

For a complete overview of the clock and reset module, see chapter 2 *OMAP3.2 Subsystem.*

Figure 20. OMAP5912 Clock Domains



3 Power Management User Services

3.1 Power Services

Power services include software and hardware mechanisms that allow the user to reduce OMAP5912 chip power-consumption during all system phases.

3.2 Static Clock Management

During major system phases, the clocks of each domain can be immediately validated or reconfigured by software register modification. Each new static configuration is held until the next control register modifications. This management is defined as static because no external hardware events are allowed to modify the last clock tree configuration .

3.2.1 DPLL1 Clock

DPLL1 synthesizes a frequency clock from an input clock reference. Each clock domain can use the DPLL clock or directly clock reference to build each clock subdomain.

Static management of the DPLL clock allows users to reduce global chip consumption through one or several clock domain controls.

Action to reduce consumption:

- ❏ Reduce DPLL frequency value (divider register configuration)

Clock domain control through DPLL configuration is done by software writes to CLKRST register *DPPL1_CTL*.

See Section 6 for software configuration details.

3.2.2 DSP/MPU/TRAFFIC Clocks

Static management allows users to reduce global chip consumption through one or several specific clock subdomain controls.

Actions to reduce consumption:

- ❏ Reduce clock subdomain frequency value (divider register configuration).
- ❏ For each domain that is idled, deactivate the respective clocks (bit register configuration).

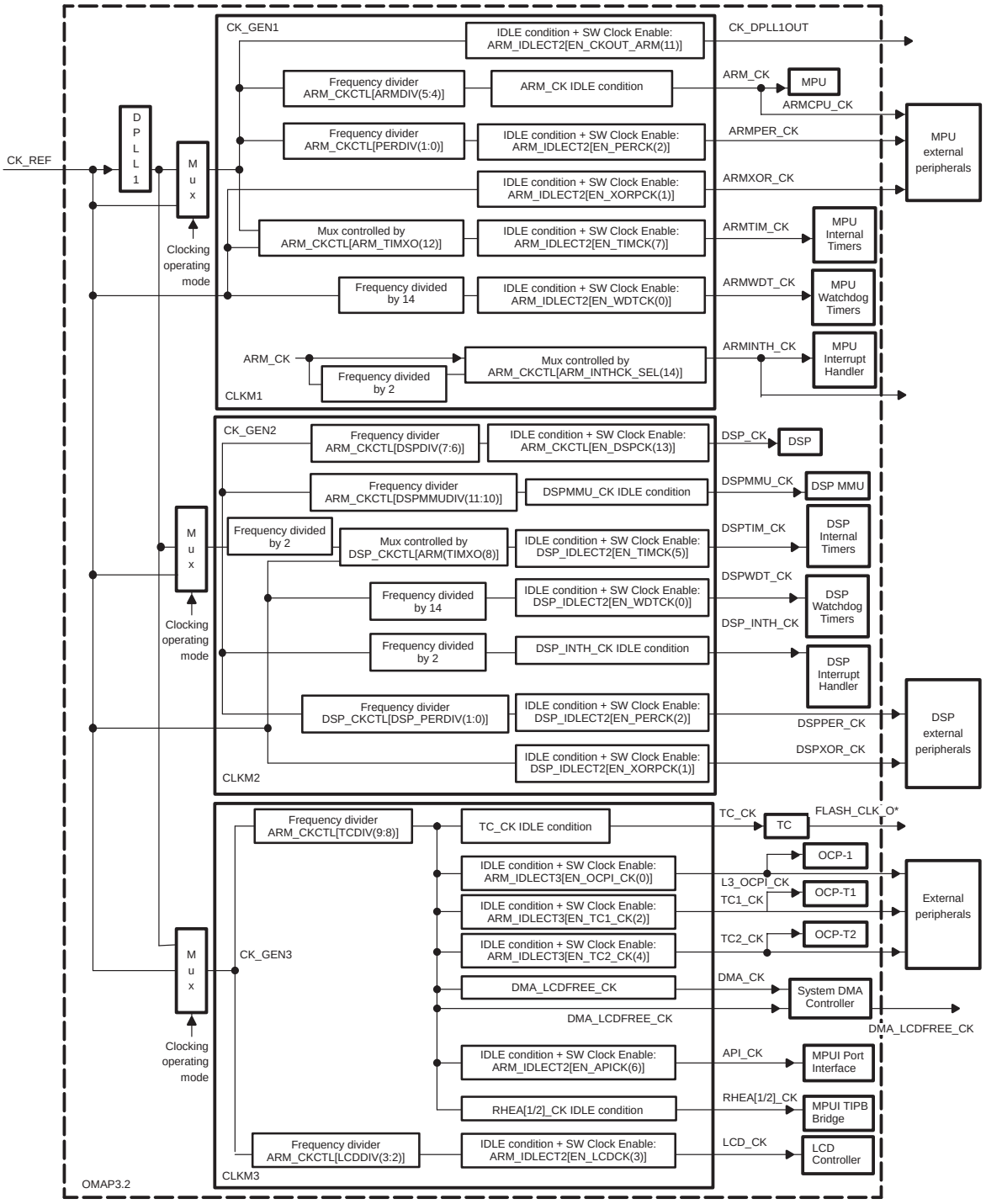
In all cases, software is responsible for determining and managing the effects of clock modifications on the system (error transitions, performance decrease, transition delays, and so on).

Table 42 describes all OMAP3.2 subsystem clocks, and Figure 21 provides an overview of their generation.

Table 42. OMAP3.2 Subsystem Clocks

Clock Name	Description
CK_DPLL1OUT	Clock from DPPL1, same as MPU clock
ARMCPU_CK	Clock with same frequency as MPU clock, same as ARM_CK
ARMXOR_CK	MPU peripheral clock, fixed, generated from CK_GEN1, can be gated
ARMPER_CK	MPU peripheral clock, divided from CK_GEN1, can be gated
ARM_INTH_CK	MPU clock interrupt handler
DSPXOR_CK	DSP peripheral clock, fixed, generated from CK_GEN2, can be gated
DSPPER_CK	DSP peripheral clock, divided from CK_GEN2, can be gated
TC_CK	Clock from EMIFS
TC1_CK	Clocks from OCP-T1
TC2_CK	Clocks from OCP-T2
L3_OCPI_CK	Same frequency as TC clock
DMA_LCDFREE_CK	Interface clock for LCD

Figure 21. OMAP3.2 Clock Generation



Clock subdomain control is done by software write to the CLKRST registers:

- ARM_CKCTL
- ARM_IDLECT1
- ARM_IDLECT2
- ARM_SYSST
- DSP_CKCTL
- DSP_IDLECT1
- DSP_IDLECT2
- DSP_SYSST

As shown in Figure 21, the clock reference to build all other clocks is provided and controlled by the ULPD (ultralow-power device) module.

The ULPD module performs several functions, which can be divided into three groups:

- Power-mode control
- Clock reference management and calibration
- External clocks management

The ULPD can manage three principal OMAP 5912 power states: awake, big sleep, and deep sleep.

ULPD management directly affects activation or deactivation of the CLKREF clock.

Note:

CLKRST static configuration affects ULPD dynamic-power mode mechanisms.

3.2.3 DSP (MGS3) Clocks Management

Global Power Management

Power can be globally managed by the idle configuration register (ICR) located in the TIPB bridge.

Table 43. Idle Configuration Register (ICR)

MPU Base Address (byte) = 0xE100 0000, DSP Base Address (word) = 0x00 0000, Offset = 0x01 (word)				
Bit	**Name**	**Function**	**R/W ***	**Reset**
15:8	Reserved	Reserved	R	0
7	RSV	Reserved idle domain.	R/W	0

Table 43. Idle Configuration Register (ICR) (Continued)

MPU Base Address (byte) = 0xE100 0000, DSP Base Address (word) = 0x00 0000, Offset = 0x01 (word)				
Bit	**Name**	**Function**	**R/W ***	**Reset**
6	RSV	Reserved idle domain.	R/W	0
5	EMIF	EMIF idle domain.	R/W	0
4	DPLL	DPLL idle domain.	R/W	0
3	PER	PER idle domain.	R/W	0
2	CACHE	CACHE idle domain.	R/W	0
1	DMA	DMA idle domain.	R/W	0
0	CPU	CPU idle domain.	R/W	0

Note: The R/W values indicate DSP access only. MPU access is read only.

This register defines six domains:

- CPU
- DMA
- ICache
- Peripherals
- Clock generator (DPLL)
- EMIF

To put one or more domains in idle mode, the user must set the corresponding bits in this register to 1 and then execute the DSP idle instruction.

A second register, the idle status register (ISTR), also located in the TIP bridge, reflects the state of the DSP when the idle instruction is executed. Table 44 defines ISTR.

Table 44. Idle Status Register (ISTR)

MPU Base Address (byte) = 0xE100 0000, DSP Base Address (word) = 0x00 0000, Offset = 0x02 (word)				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:8	Reserved	Reserved	R	0
7	RHEA_IDLE7_TR	Reserved idle status	R	0
6	RHEA_IDLE6_TR	Reserved idle status	R	0
5	RHEA_IDLEEMIF_TR	EMIF idle status	R	0
4	RHEA_IDLEDPLL_TR	DPLL idle status.	R	0
3	RHEA_IDLEPERH_TR	PER idle status.	R	0

Table 44. Idle Status Register (ISTR) (Continued)

MPU Base Address (byte) = 0xE100 0000, DSP Base Address (word) = 0x00 0000, Offset = 0x02 (word)				
Bit	**Name**	**Function**	**R/W**	**Reset**
2	RHEA_IDLECACHE_TR	CACHE idle status.	R	0
1	RHEA_IDLEDMA_TR	DMA idle status.	R	0
0	RHEA_IDLECPU_TR	CPU idle status.	R	0

The TIPB bridge performs some checks when the DSP idle instruction is executed:

- If the DPLL bit is set to 1 in the ICR and one or more of the CPU, DMA, EMIF, or CACHE bits are set to 0, a bus error is generated and the idle request is canceled.
- Before setting the DPLL into idle, the TIPB bridge checks whether DMA, EMIF, and CACHE are inactive. If they are not, it waits for these blocks to become inactive before entering idle mode.

All peripherals are part of the same domain. Therefore, when the peripherals domain is put in idle mode, the clocks of all peripherals are stopped.

TIPB and MPU blocks are not part of any domain. The DPLL must be disabledto put them in idle mode.

If some blocks are in idle mode but both the CPU and the DPLL are not, the program flow is maintained. It is thus possible to exit from idle mode by writing in the ICR register and by executing an idle instruction. This is the only wake procedure under software control; the other wake procedures are under hardware control.

Local Power Management

The EMIF global control register (see Table 45) allows control of the SDRAM clock (it can be enabled or stopped) and the SBSRAM clock (it can be enabled or stopped and the frequency can also be divided by 2). The EMIF external bus can also be placed in low-power state with this register.

Table 45. EMIF Global Control Register (GCR)

MPU Base Address (byte) = 0xE100 0800, DSP Base Address (word) = 0x00 0800, Offset = 0x00 (word)				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:11	RESERVED	Reserved	R	0
10:9	MEMFREQ	MEMory clock FREQuency	R	0
		MEMFREQ = 00: SBSRAM and/or the SDRAM interface is configured for 1x mode and the CLKMEM clock frequency is equal to the DSP clock frequency		
		MEMFREQ = 01: SBSRAM and/or the SDRAM interface is configured for 1/2x mode and the CLKMEM clock frequency is equal to the DSP clock frequency divided by 2.		
		MEMFREQ = 10 or 11: Reserved for future clock rates.		
8	RESERVED	Reserved	R/W	0
7	WPE	Write posting enable	R/W	0
		WPE = 0, write posting is disabled (for debug). WPE = 1, write posting is enabled.		
6	RESERVED	Reserved	R/W	0
5	MEMCEN	MEM Clock ENable	R/W	1
		MEMCEN = 0, CLKMEM held high MEMCEN = 1, CLKMEM enabled to clock		
4	ARDYOFF	Async. ready off.	R/W	0
3	ARDY	Value of the ARDY input.	R/W	0
2	HOLD	Value of HOLD input	R/W	0
1	HOLDA	Value of HOLDA output	R/W	0
0	NOHOLD	External HOLD disable	R/W	0
		NOHOLD = 0, hold enabled NOHOLD = 1, hold disabled		

The TIPB control mode register (CMR) includes wait state bits that can be used to set the strobe frequency of the peripherals. Table 46 defines CMR.

Table 46. TIPB Control Mode Register (CMR)

MPU Base Address (byte) = 0xE100 0000, DSP Base Address (word) = 0x00 0000, Offset = 0x00 (word)					
Bit	**Name**	**Description**	**Reset**	**CPU Access**	**MPU Access**
[15−9]	Timeout(6:0)	Strobe cycles	0x7F	Read/Write	Read
[8]	Wait State 3 (strobe 1)	Strobe 1 length (high bit)	0	Read/Write	Read
[7]	Wait State 2 (strobe 1)	Strobe 1 length (medium bit)	0	Read/Write	Read
[6]	Wait state1 (strobe 1)	Strobe 1 length (low bit)	1	Read/Write	Read
[5]	Wait state 3 (strobe 0)	Strobe 0 length (high bit)	0	Read/Write	Read
[4]	Wait state 2 (strobe 0)	Strobe 0 length (medium bit)	0	Read/Write	Read
[3]	Wait state 1 (strobe 0)	Strobe 0 length (low bit)	1	Read/Write	Read
[2]	First priority	Priority modes	1	Read/Write	Read
[1]	Bus error	Application flag error	0	Read/Clear	Read(0 in HOM)
[0]	Mode	SAM or HOM	1 (HOM)	Read	Read

The strobe frequency can go from 1/2x to 1/9x of the DSP clock frequency.

Table 47. Wait State Strobe Frequency NIL

Number of Wait States	Strobe Period	DSP Max Frequency (MHz)	Strobe Frequency (MHz)
0	DSP clk/2	200	100
1	DSP clk/3	200	66
2	DSP clk/4	200	50
3	DSP clk/5	200	40
4	DSP clk/6	200	33
5	DSP clk/7	200	28
6	DSP clk/8	200	25
7	DSP clk/9	200	22

3.2.4 RNG CLOCKS

The RNG module is divided into four principal blocks:

- RING oscillators
- RNG generator
- State machine
- Output and input registers

The random numbers are accessible to the application in a 32-bit read-only register (RNG_OUT). Once the register is read, the RNG module immediately generates a new value. If no RNG read is performed, the RNG module goes idle (IDLERNG) after a maximum of 2^{24} cycles of the input system clock.

In this state the ring oscillators and RNG generator are completely stopped, while only the internal state machine stays active, clocked by the input system clock.

There are three ways to reduce RNG module consumption:

- Total RNG shutdown (automatic)
- Partial RNG shutdown
- Total RNG shutdown (reset RNG)

Total RNG Shutdown

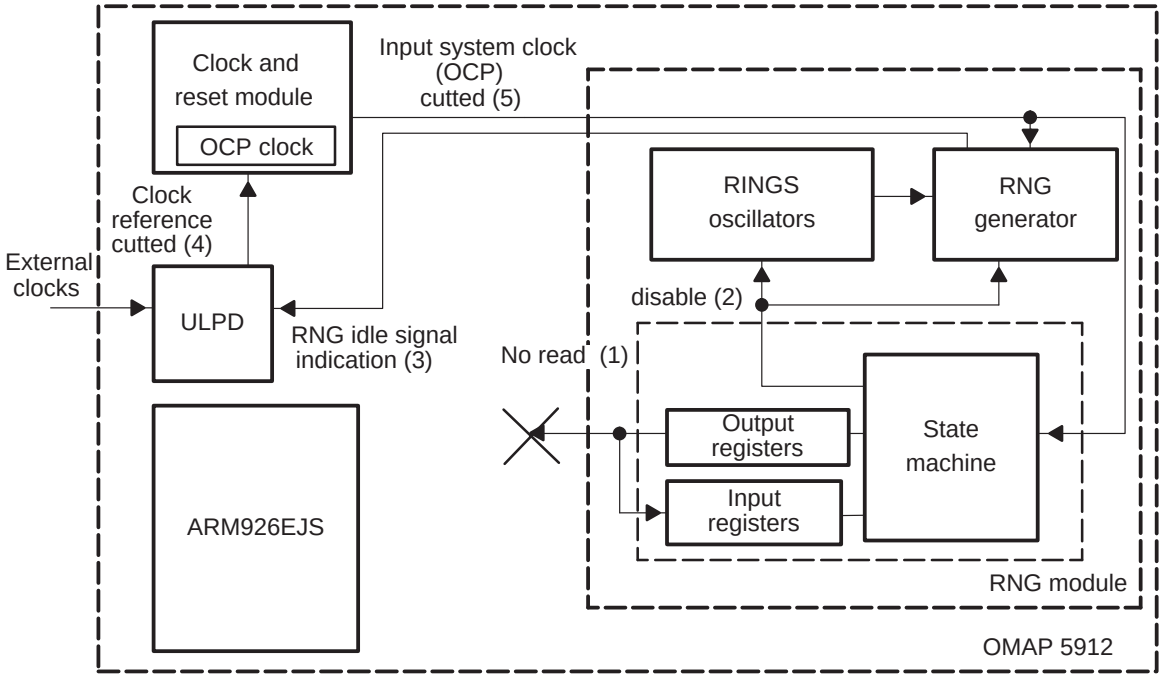
The best and safest way to go into sleep mode is to stop reading the last generated random number. After a maximum of 2^{24} cycles of the input clock, the state machine is in its IDLERNG state. This requires that the input clock be present so that the state machine can jump to its different state. At this state, the rings and the 24-bit counter are off (if bit 0 of RNG_MASK is 1). The RNG can be kept in this state indefinitely and as soon as a random number is read, the state machine wakes up. When the state machine is in its IDLERNG state, a signal at the boundary goes low, indicating that the input clock can be cut off. There is no need to perform a soft reset.

The counter, being synchronous with a clock derived from the input system clock, is still incrementing if the autoidle bit of the module is not set by writing 1 in the bit 0 of the RNG_MASK register (offset address 0x40).

Because all RNG activities are shut off, this is the best method for RNG power consumption reduction. It takes a maximum of 2^{24} cycles to reach the IDLERNG state where the RNG oscillator is shut off.

The input clock can be restarted anytime and no reset is required to restart RNG module.

Figure 22. Total Automatic RNG Shutdown

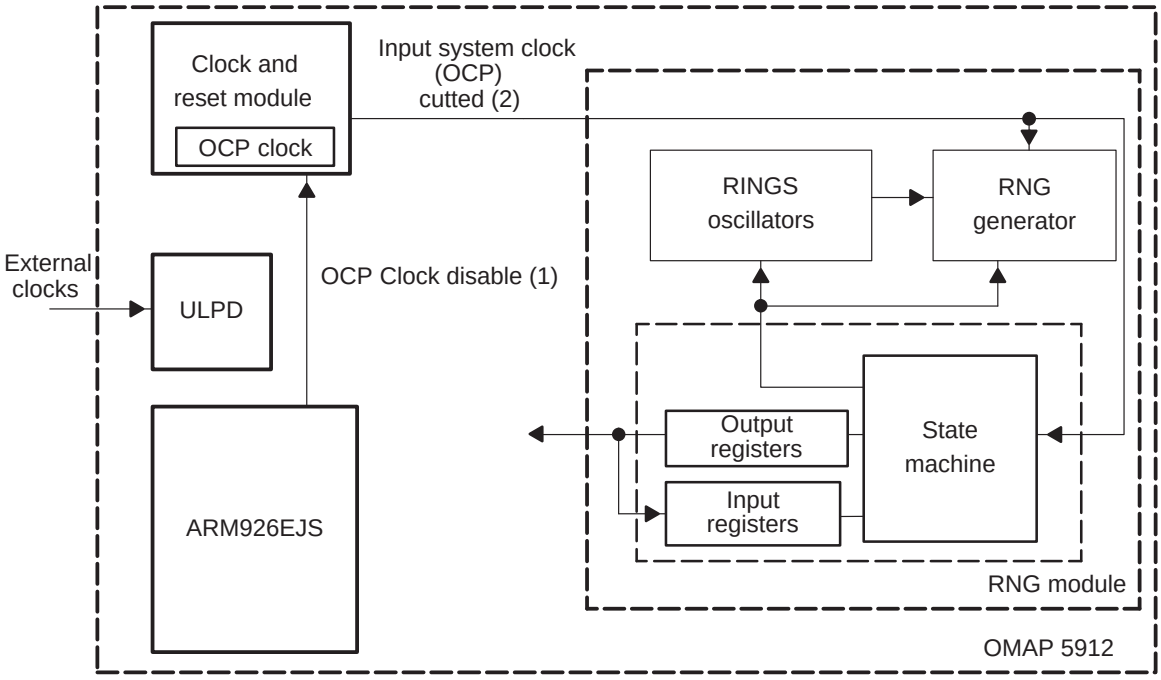


Partial RNG Shutdown: Input Clock Cut Off

The second safest wayto reduce RNG module consumption is to cut the input clock (OCP). The state machine does not jump any more to the different states, the entropy is preserved in the system, and cutting the clock does not affect the randomness. The input clock can be restarted anytime and no reset is needed to correctly restart the RNG module.

Because the state machine can be in any state when the input clock is cut off, one of the rings can still be on. The state machine does not reach the IDLERNG anymore, so the ring is still on. Hence, there is a penalty on power consumption reduction.

Figure 23. Partial RNG Shut Down



Total RNG Shutdown: Reset RNG Module

If an application no longer needs the RNG functions and needs to go into deep sleep mode without waiting for a transition to the IDLERNG state, the application can write 0 in the field RESET_COUNT of the RNG_CONFIG and the input system clock can be switched off directly.

If this scenario is completed, and if a random number is needed afterwards, it is then best to perform a soft reset, because randomness cannot be ensured. The penalty is that again 2^{25} cycles are needed before the first random number is ready. This method must be done as a last resort when RNG is not used anymore.

Enter secure mode through normal entry sequence, and then activate the RNG test mode by writing 1 into the RNG_CTRL (0x08) bit number 1.

This unlocks access to the RESET_COUNT field (bits 11−6) of RNG_CONFIG register (0x28).

At this point you have broken the RNG functionality and the RNG cannot be used again for security functions unless a proper soft reset is performed with bit 11 set to 1.

Fill the field RESET_COUNT (bits 11−6) with zeros. The ring oscillators are OFF.

Stop the OCP clock in OMAP and the RNG is OFF.

3.2.5 Externals Clocks

Clocks to external interfaces can be requested on demand. See Chapter 5, *Initialization*, for additional information.

3.3 Dynamic Management

3.3.1 Autogating Mechanisms

Some modules provide internal mechanisms to allow automatic cutting of their internal clocks when they are not required; in this way, the power is decreased. With this automatic clock gating, the internal clock can be halted when no external request is active or when the last task is ended.

These mechanisms can be enabled or disabled by software configuration; when enabled, the clock gating is completely transparent to the user.

OMAP3.2 Autogating

Table 48 lists the OMAP3.2 modules that provide the autogating clock feature along with the relevant bit register to enable or disable the feature.

Table 48. OMAP3.2 Modules With Clock Autogating Enable Feature

Module	Register Name	Autogating Enable Bit (Register Field)
System DMA	DMA_GCR	Bit 3, CLK_AUTOGATING_ON
Traffic controller (TC) OCP−T1, OCP−T2	CONFIG_REG	Bit 0, AUTO_GATED_CLK
EMIFS	CONFIG_REG	Bit 2, PWD_EN
EMIFF	SDRAM_CONFIG_2	Bit 2, SD_AUTO_CLK

OMAP5912 Peripherals Autogating

Table 49 lists the OMAP5912 peripherals that provide the autogating clock feature along with the relevant bit register to enable or disable the feature.

Table 49. OMAP5912 Peripherals With Clock Autogating Enable Feature

Module	Register Name	Autogating Enable Bit (Register Field)
32-kHz watchdog	WD_SYSCONFIG	Bit 0, AutoIdle
Secure watchdog	WD_SYSCONFIG	Bit 0, AutoIdle
RNG	RNG_MASK	Bit 0, AutoIdle
DES3DES	DES_MASK	Bit 0, AutoIdle
SHA-1/MD5	SHA_MASK	Bit 0, AutoIdle
OMAP5912 configuration	MOD_CONF_CTRL_1	Bit 26, Enable for the SSI interconnect Bit 25, Enable for the OCP interconnect
MPU level 2 interrupt handler	OCP_CFG	Bit 0, AutoIdle
DSP level 2.1 interrupt handler	OCP_CFG	Bit 0, AutoIdle
NAND flash controller	NND_SYSCFG	Bit 0, AutoIdle
SPI	SPI_SCR	Bit 0, AutoIdle
Dual-mode timer [7:0]	TIOCP_CFG	Bit 0, AutoIdle
UART [3:1]	SYSC	Bit 0, AutoIdle
GPIO [4:1]	GPIO_SYSCONFIG	Bit 0, AutoIdle
GDD	GDD_GCR	Bit 3, CLOCK_AUTOGATING_ON

Notes: 1) At reset, the autogating enable bit is disabled to minimize the power consumption. It is the software responsibility to enable the clock autogating feature for the above modules.

2) When not used, the STI module can be totally shut down by software resetting the bit 15 (STIEN field) of the STI enable register (STI_ER).

3) When not used, the GDD module can be totally shut down by software setting the bit 0 (SWITCH_OFF field) of the GDD global control register (GDD_GCR).

MGS3/DSP Autogating

The EMIF is able to disable some internal and external clocks depending on its activity. This autogating feature can be managed by software via the power management register.

The power management register controls whether or not the EMIF tries to disable internal and external clocks when not needed, to reduce power consumption. Some of these options may not work at the maximum operating frequency or with all memory devices. Table 50 describes the fields in the power management control register.

Table 50. Power Management Control Register

Field	Description
MEMIPM	Memory contoller internal clock power management. MEMIPM = 1 Each internal memory controller clock is disabled when no access is pending or active on that memory interface. MEMIPM = 0. All memory controller clocks are disabled only when the EMIF is in the IDLE or HOLD state.
CPUIPM	CPU interface internal clock power management. CPUIPM = 1 The internal clock driving the CPU/DMA pipeline registers and stall logic is disabled when no request input is active, no request is pending, and no access is active in the EMIF. CPUIPM = 0 The internal clock driving the CPU/DMA pipeline registers and stall logic is disabled only when the EMIF is in the IDLE or HOLD state.
MAINIPM	Main internal clock power management. MAINIPM = 1: The internal clock driving the arbitration, scheduler, external address, data logic, and so on, is disabled when no request is pending and no access is active in the EMIF. MAINIPM = 0: The internal clock driving the arbitration, scheduler, external address, data logic, and so on, is only disabled when the EMIF is in the IDLE or HOLD state.
SBSXPM	SBSRAM external clock power management. SBSXPM = 1: The SBSCLK_R clock is disabled by the SBSCEN bit whenever no SBSRAM accesses are pending. The EMIF ensures that the memory device is clocked at least twice before the access starts. SBSRPM = 0: The SBSCLK_R clock is disabled only by the SBSCEN TIPB bit.

3.4 ULPD Power Modes Management

3.4.1 Introduction

The ULPD is a power management module running at 32 kHz.

The ULPD has two main clock sources: a 32-kHz clock and a system clock of medium frequency (12 MHz, 13 MHz, and 19.2 MHz).

The ULPD provides system reference used by all clock domains.

The ULPD internal state machine provides three system power modes: awake, big sleep, and deep sleep. These three modes determine global clock activity and overall consumption in OMAP5912.

Idle states are defined for each clock domain or subdomain.

3.4.2 ULPD Mode Descriptions

The following describes the three system power modes:

- In deep sleep (lowest-consumption mode):
  - Only Clk32k clock, used by the ULPD and RTC modules, is on.
  - The system clock is off.
- In big sleep (low-consumption mode):
  - Clk32k and the system clocks, used by ULPD, RTC modules, and external peripherals, are on.
  - The OMAP5912 clock reference is off.
- In awake mode:
  - Clk32k and system clocks are on.
  - OMAP5912 clock reference provide by ULPD is on.

From a consumption point of view, it is best to go into idle mode (standby MPU signal activated) whenever possible.

3.4.3 ULPD Mechanisms Description

To handle power mode transition, the ULPD module controls principally:

- Standby wait for interruption signal received from the ARM926EJS processor.This signal is generated from specific MPU instruction decoding (software running).
- Idle request signals sent to all OMAP5912 clock domains
- Idle acknowledge signals from all OMAP5912 clock domains
- Wake-up signal (from external or internal requests)

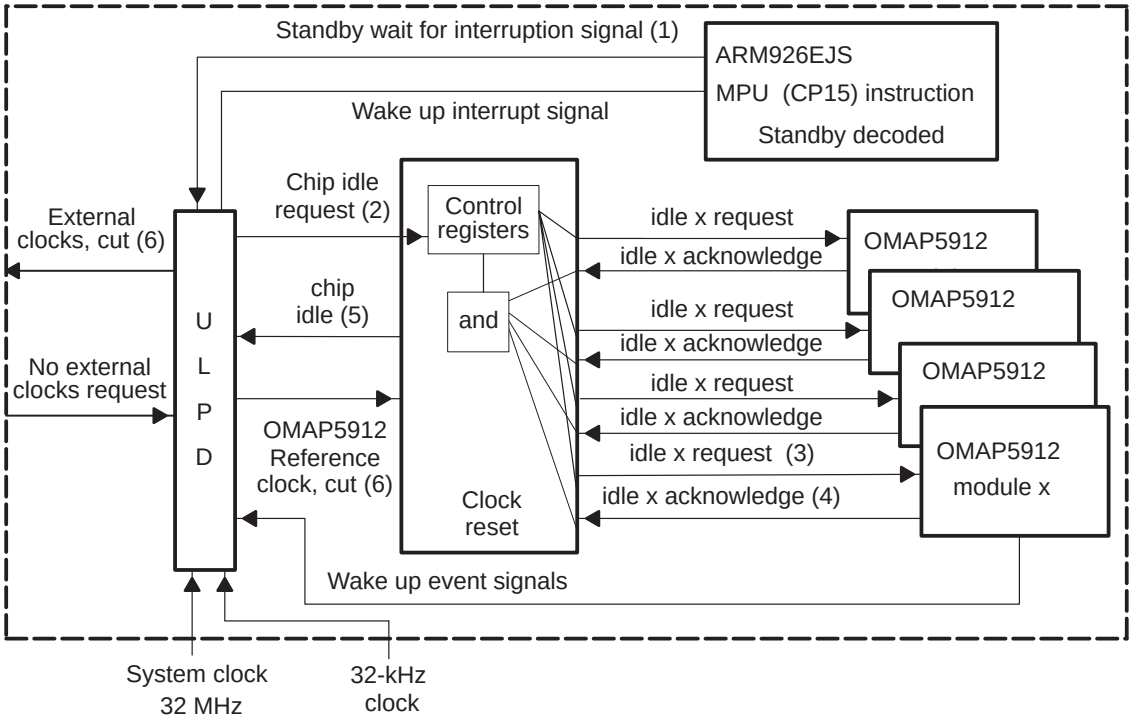
The different clock requests to the ULPD are described in Table 4.

Software is responsible for requesting the transition from the awake mode to the deep sleep or big sleep modes.

All OMAP clock domains determine whether to validate transitions (with acknowledge signals) to the next state according to their activity.

The main steps in the transition from the awake mode to other modes are shown in Figure 24.

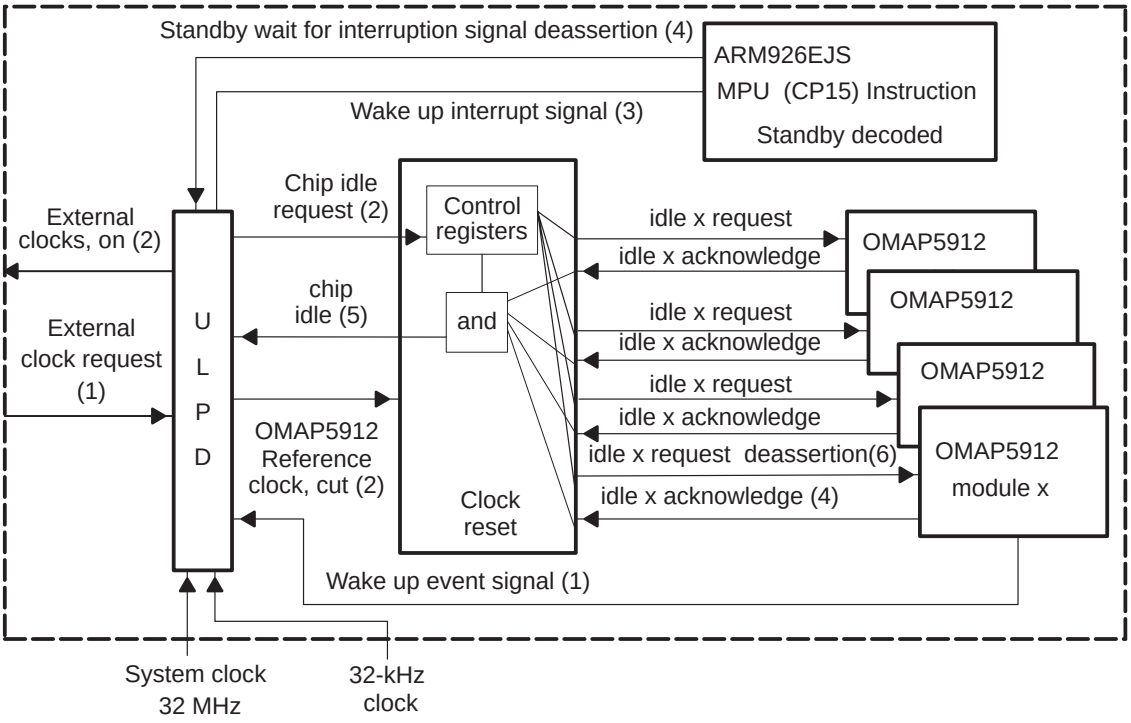
Figure 24. OMAP5912 Shutdown Request Management



During the low-consumption states of deep sleep and big sleep, the ULPD scans hardware events, interrupts, and external clock requests to restore the clock reference system (or/and external clocks) and wake up the ARM926EJS processor.

The main steps in the transition from the deep sleep or big sleep modes to the awake mode are shown in Figure 25.

Figure 25. OMAP5912 Wake-Up Management



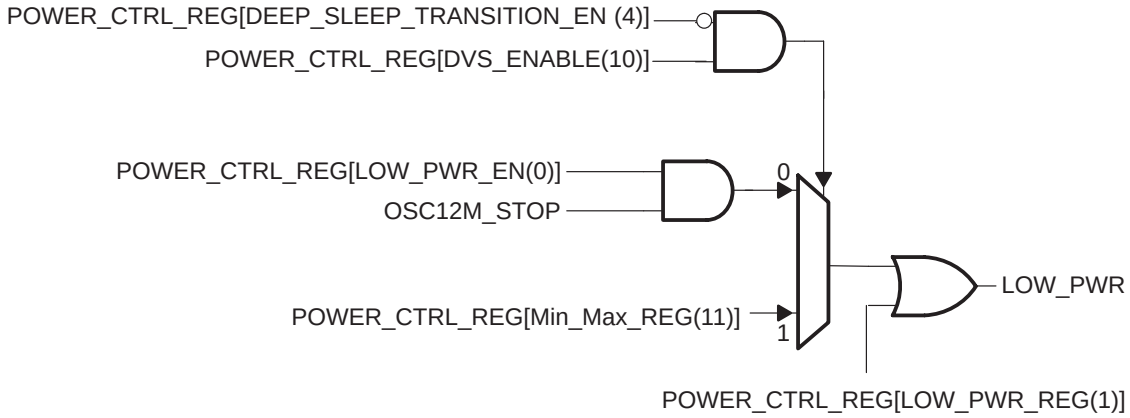
3.4.4 Control of External Clock and Voltage Supplies

The ULPD provides the $\overline{\text{LOW\_PWR}}$ and LOW_PWR signals to control the activation or the shutdown of the external clock and the core voltage supplies.

- $\overline{\text{LOW\_PWR}}$ indicates to external devices that the system clock can be shut down (in external clock mode).
- LOW_PWR drives the external core voltage supply in low voltage (1.1 V) operations. The behavior of the LOW_PWR signal is set by software and can be configured to allow two types of operation:
  - Reduction of leakage current in deep sleep mode
  - Low-voltage operation at reduced clock frequency (dynamic voltage scaling)

To enable the operation at reduced clock frequency under external low-voltage supplies, the LOW_PWR signal is set by software through the ULPD power control register (POWER_CTRL_REG bits 1, 4, 10, and 11). Figure 26 summarizes the LOW_PWR signal behavior.

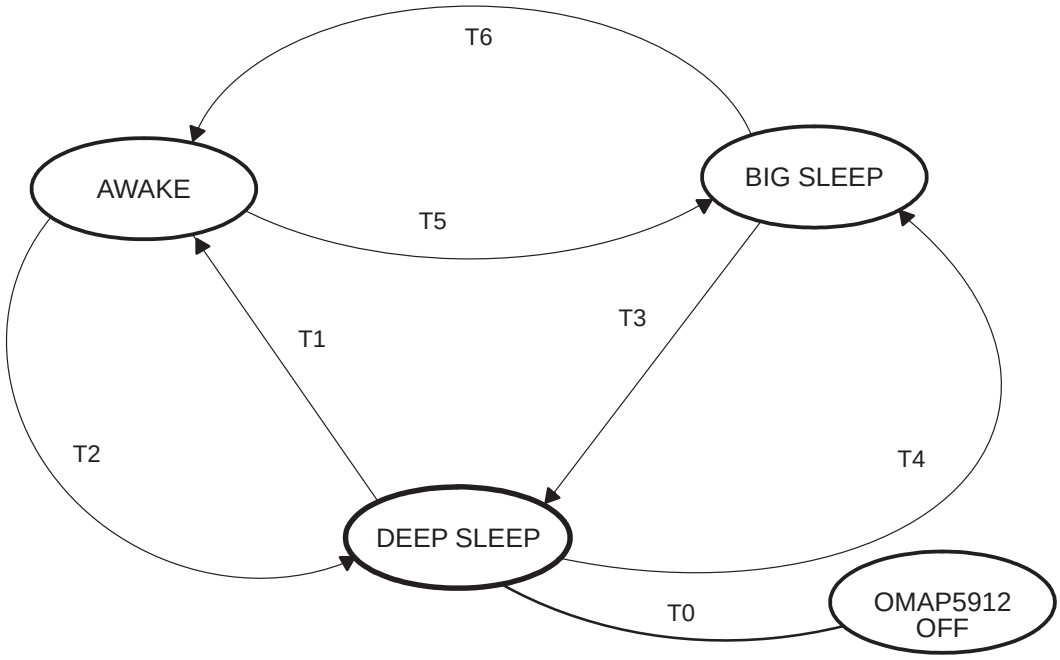
Figure 26. Software Control of the LOW_PWR Signal



3.4.5 Transitions Between ULPD Modes

Figure 27 describes the transition flow between the ULPD modes.

Figure 27. Transition Flow



T0: Power up

T1: Wake-up request from OMAP3.2 or from peripherals or after power up context

T2: Idle request from OMAP3.2, externals clocks not requested, and deep sleep authorized

T3: No wake-up request from OMAP3.2 nor from external peripherals, and deep sleep authorized

T4: External clock request

T5: Idle request from OMAP3.2 but either at least one active external clock request or deep sleep transition not authorized

T6: Either wake-up request from OMAP3.2 or from external peripherals

3.5 Power Domain Management

3.5.1 RTC Domain Management

The RTC power domain allows the user to achieve minimum consumption during the OFF state of OMAP5912.

In the OFF state only the RTC power domain OMAP 5912 is supplied. The DSP and MPU domains are switched off (all power supplies associated equal 0 volts).

The RTC power domain supplies only real-time clock functions and associated reset controls.

The user must control the ON_OFF signal during MPU and DSP domain switch-off and switch-on.

Figure 28 describes the OFF state of OMAP5912, and Figure 29 describes the ON state.

Figure 28. OMAP 5912 State OFF

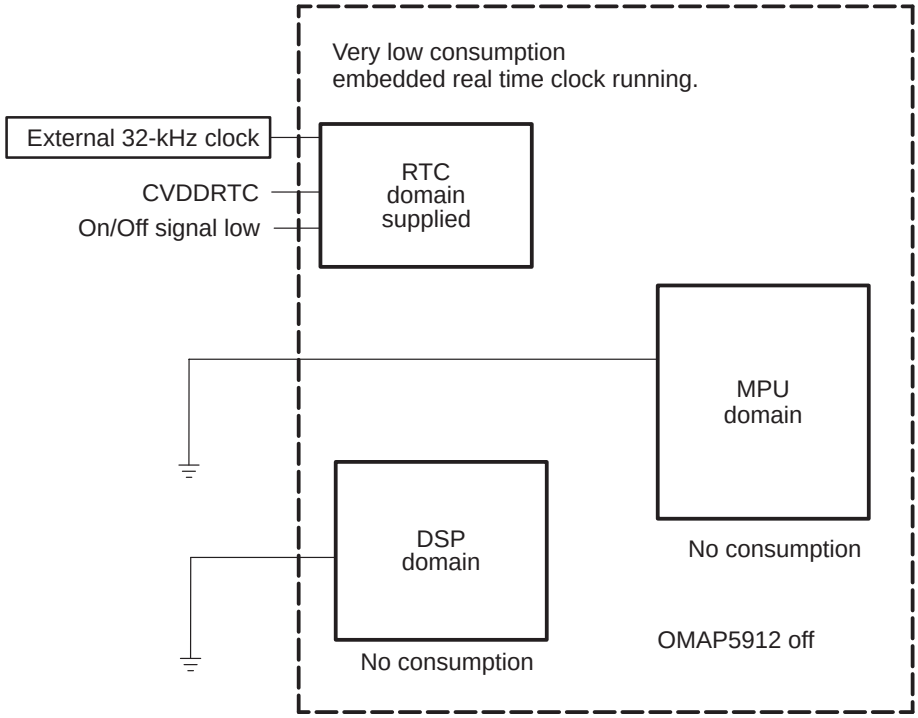
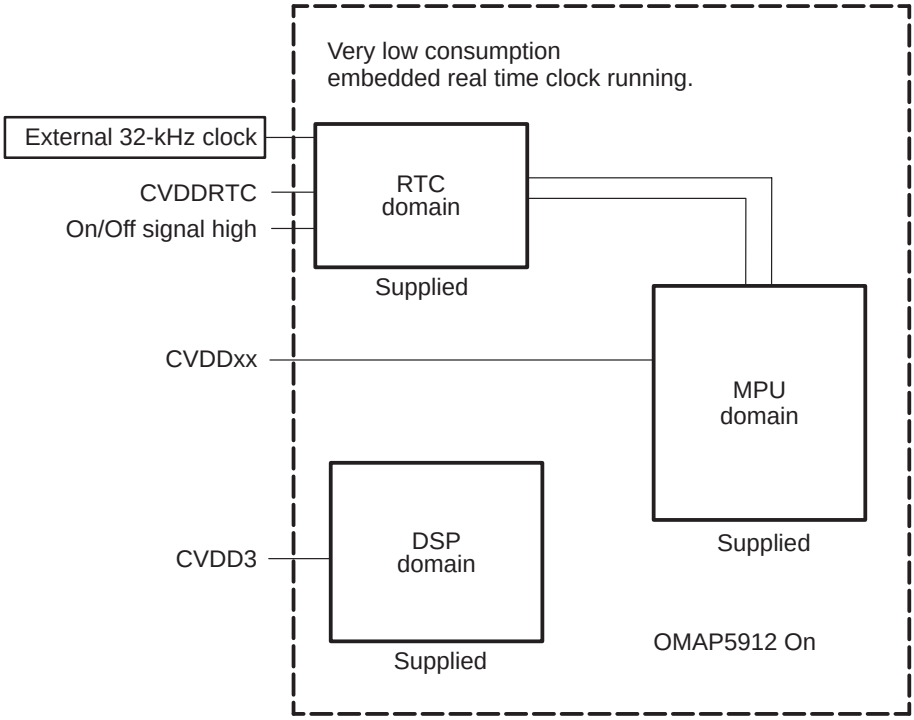


Figure 29. OMAP 5912 State ON



3.5.2 DSP Domain Management

The DSP power domain is surrounded by an isolation wrapper to provide proper electrical isolation and the appropriate interface state at the boundary of the MPU and DSP power domains.

The assumption for OMAP5912 is that the power domains are connected exclusively to external power supplies (not necessarily dedicated). No e-LDO and no power switch are implemented in OMAP5912 to supply the MPU and DSP domains.

When the DSP is in isolation mode, it is possible to cut its dedicated power supplies. The isolation wrapper is controlled by bit[12] of POWER_CTRL_REG in the ULPD register file. Table 51 describes the isolation control bit.

Table 51. DSP Isolation Control

12	ISOLATION_CONTROL	0: Electrical isolation inactive 1: Electrical isolation active Reset of this bit is done upon power up reset only (PWRON_RESET).	R/W	0x0

4 Dynamic Voltage Scaling

To minimize the leakage current when OMAP5912 is in deep sleep mode, decrease the external supply voltages once the deep sleep state is validated. The dynamic voltage scaling (DVS) feature enables the operation at reduced clock frequency when the external supply voltages are low.

The OMAP5912 provides two signals that control the core voltage supplies and activation or shutdown of the external clock, depending on the clock mode (oscillator or external). These two signals, LOW_PWR and LOW_PWR, behave similarly except that they do not have the same polarity, and LOW_PWR can be controlled by software, whereas LOW_PWR cannot.

4.1 Low Voltage With Chip Totally Shut Down

4.1.1 Oscillator Clock Mode

In deep sleep mode, to minimize the power consumption caused by leakage currents, decrease the external supply voltages. To enable this feature, the following conditions must be met:

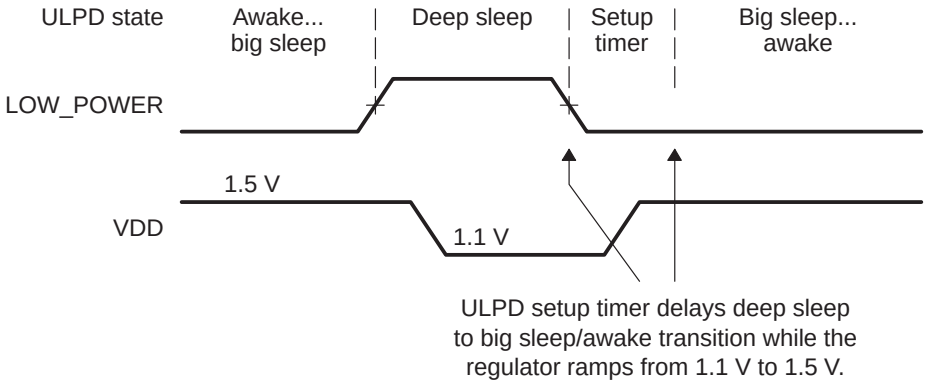
- Set the ULPD POWER_CTRL_REG[0] bit (LOW_PWR_EN field) to 1 to enable the low-power feature.
- Set the ULPD POWER_CTRL_REG[4] bit (DEEP_SLEEP_TRANSITION_EN field) to 1 to enable transition to the deep sleep state or to reset the POWER_CTRL_REG[10] bit to disable the DVS feature.

If the above register settings are made, the external signal LOW_PWR switches to active high whenever the ULPD enters the deep sleep state. In this way, the external core voltage supply can be driven at low voltage, reducing the leakage current.

When the ULPD exits the deep sleep state, the signal LOW_PWR switches back to inactive low and the external core voltage supply ramps up to nominal 1.6 V.

At reset, the LOW_PWR feature is disabled and POWER_CTRL_REG[0] is set to 0. The LOW_PWR signal is inactive low, which indicates nominal voltage requirement. Figure 30 describes the behavior of the signal.

Figure 30. Behavior of LOW_PWR Signal



4.1.2 External Clock Mode

The $\overline{\text{LOW\_PWR}}$ signal is used in external clock mode. When active low, $\overline{\text{LOW\_PWR}}$ indicates to external devices that the input system clock (system clock) can be shut down. It also indicates that the external core voltage supply can be lowered to 1.1 V to reduce the chip leakage-current consumption.

The $\overline{\text{LOW\_PWR}}$ signal is asserted low when the ULPD enters the deep sleep state (except at power-up reset) and is released upon deep sleep exit (except at power-up reset). At power-up reset, $\overline{\text{LOW\_PWR}}$ is reset to its inactive value (high). Figure 31 describes the assertion of the signal, and Figure 32 describes the release of the signal.

Figure 31. Assertion of the $\overline{\text{LOW\_PWR}}$ Signal

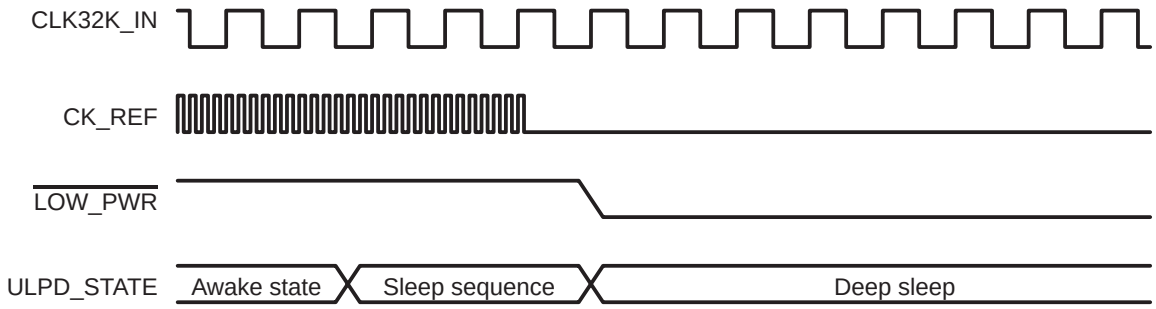
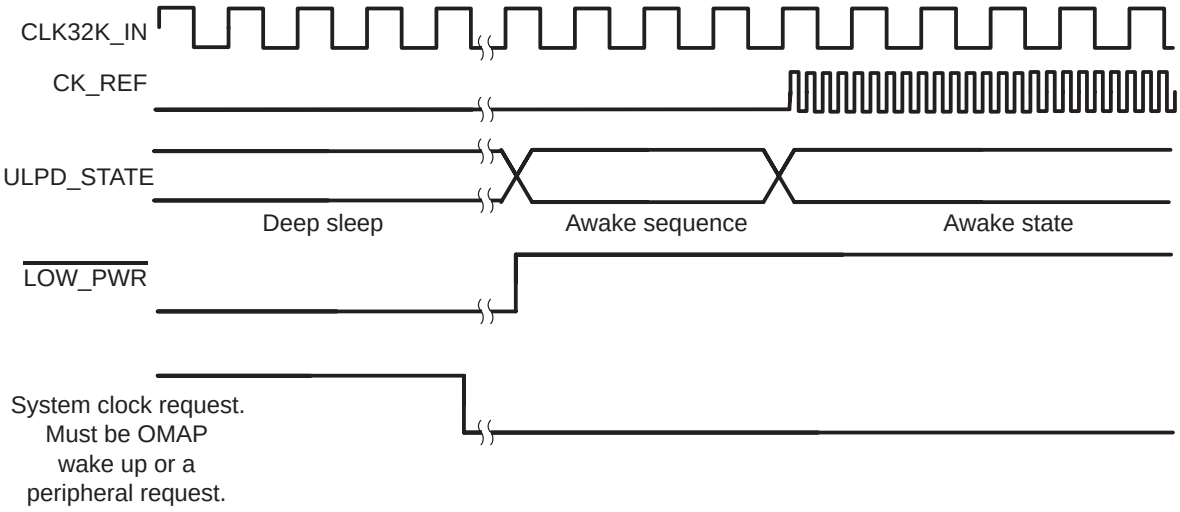


Figure 32. Release of the $\overline{\text{LOW\_PWR}}$ Signal



4.2 Low Voltage With Chip Running at Reduced Clock Frequencies

If the application does not require many MIPS, operate at low voltage and reduced frequency. For instance, at a reduced core voltage of 1.05 V, the MPU frequency can be scaled down to 80 MHz and the traffic controller frequency to 40 MHz.

5 OMAP5912 Power Modes

The global OMAP5912 system power mode depends on the authorized combinations of MPU and DSP domain states.

On OMAP5912, the power management policy (software) is run exclusively on the MPU. The DSP domain behaves as a slave, so the DSP domain power state transitions are performed under the control of the MPU software. However, the transition from active to inactive for the DSP domain state is an exception to this rule. This transition is controlled by DSP software (idle instruction).

The power state transition of the MPU domain is controlled by software and hardware. The hardware part is composed mainly of the ULPD module FSM, which performs automatic transition between some of the MPU domain states.

Table 52 lists the eight possible system power modes for OMAP5912.

Table 52. Global OMAP5912 System Power Modes

Global OMAP5912 Power Mode	Chip State	MPU Domain State	DSP Domain State	Input System Clock	32-kHz Clock
ACTIVE MODE	1.1	Active state	Active state	On	On
	1.2	Active state	Inactive state	On	On
	1.3	Active state	Sleep state	On	On
BIG SLEEP MODE	2.1	Inactive state	Inactive state	On	On
	2.2	Inactive state	Sleep state	On	On
DEEP SLEEP MODE	3.1	Pending state	Pending state	On or Off	On
	3.2	Pending state	Sleep state	On or Off	On
OFF MODE	0	Sleep state	Sleep state	Off	On or Off

The possible MPU system (ARM926EJS, TC, DMA, and memory interfaces—all OMAP5912 peripherals) states are described in Table 53.

Table 53. MPU Domain States

MPU Domain State	CLOCKS						ULPD State	Power Supply Voltage
	OMAP3.2 Input Clock	OMAP3.2 PLL	MPU and TC Clocks	32 kHZ Clock	ULPD PLL	Peripheral Clocks		
Active	On	On or Off	On or Off	On	On or Off	On or Off	Awake	1.6 or 1.1 V
Inactive	Off	Off	Off	On	Off	Off	Big Sleep	1.6 or 1.1 V
Pending	Off	Off	Off	On	Off	Off	Deep Sleep	1.1 V
Sleep	Off	Off	Off	Off	Off	Off	Off	0 V

State retention is ensured when the MPU subsystem is in active, inactive, and pending states.

In sleep state, there is no state retention.

The possible DSP system (MGS3) states are described in Table 54.

Table 54. DSP Domain States

DSP Domain State	CLOCKS		MGS3 Reset	DSP Isolation Wrapper	Power Supply Voltage
	MGS3 Input Clock	MGS3 Sub-domain Clocks			
Active	On	On or Off	Released	Deactivated	1.6 or 1.1 V
Inactive	Off	Off	Released	Deactivated	1.6 or 1.1 V
Pending (see Note)	Off	Off	Released	Deactivated	1.1 V
Sleep	Off	Off	Asserted	Activated	0 V

Note: The DSP subsystem is in pending state when it was set to inactive state before the MPU subsystem goes into pending state (with the ULPD in deep sleep state).

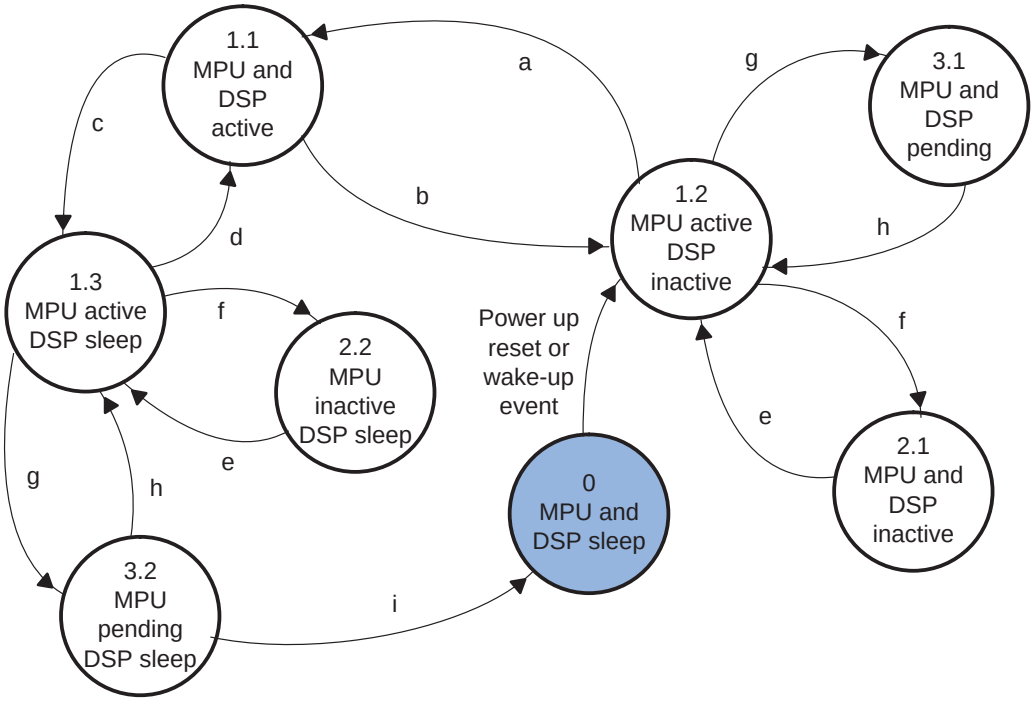
State retention is ensured when the DSP subsystem is in active, inactive, and pending states.

In sleep state, there is no state retention.

5.1 OMAP5912 Power Mode Transitions

Figure 33 describes the OMAP5912 power mode transitions. Note that the initial state is state 0 (the only shaded state).

Figure 33. Power Mode Transitions



1) DSP wake-up from inactive state

   Wake-up is automatically handled by the OMAP3.2 CLKRST module upon any unmasked interrupts or reset.

   a) Any unmasked interrupt that fires up is asynchronously propagated to the OMAP3.2 CLKRST module as a wake-up event.

   b) The CLKRST module detects the wake-up event.

   The OMAP3.2 CLKRST module automatically restarts the DSP input clock DSP_CK and the DSP subdomain clock.

2) DSP active to inactive state transition

   a) DSP power domain is in active state.

   b) Either the MPU requests the DSP to enter idle mode for inactive state via mailbox or shared memory, or the DSP OS enters the IDLE process.

   c) The DSP programs the ICR register of the DSP to shut down all megacell subdomain clocks.

   d) The DSP programs registers DSP_IDLECT1 and/or DSP_IDLECT2 of the OMAP3.2 CLKRST module to set the state of the OMAP3.2

DSP clock subdomains (CLKM2) (these belong to the MPU power domain and must be requested by the MPU).

e) The DSP masks/unmasks interrupts in the DSP interrupt handler to ensure a wake-up path.

f) The DSP executes the IDLE instruction.

g) The DSP input clock (DSP_CK) is shut down automatically by the OMAP CLKRST module.

h) The DSP power domain is in inactive state. The MPU can detect when the transition is completed by monitoring the IDLE_DSP bit in the ARM_SYSST register of the OMAP3.2 CLKRST module.

3) DSP active to sleep state transition

a) The MPU requests the DSP to enter into DSP idle mode for sleep state via mailbox or shared memory.

b) DSP programs the ICR register of DSP to shut down *all* megacell subdomain clocks.

c) DSP programs registers DSP_IDLECT1 and/or DSP_IDLECT2 of the OMAP3.2 CLKRST module to set the state of the OMAP3.2 DSP clock subdomains (CLKM2) (these belong to the MPU domain and are requested by MPU).

d) The DSP masks all interrupts in the DSP interrupt handler. There is no direct wake-up path through DSP interrupts.

e) DSP executes an IDLE instruction.

f) The DSP input clock (DSP_CK) is shut down automatically by the OMAP3.2 CLKRST module.

g) The DSP power domain is in inactive state. The MPU can detect when the transition is completed by monitoring the IDLE_DSP bit in the ARM_SYSST register of the OMAP3.2 CLKRST module.

h) The MPU puts the DSP into reset state by asserting the DSP_EN and DSP_RST bits into the ARM_RSTCT1 register of the CLKRST module.

i) The MPU disables the DSP clock by deasserting the EN_DSPCK bit in the ARM_CKCTL register of CLKRST.

j) The MPU programs the ULPD POWER_CTRL_REG[12] to assert the ISOLATION_CONTROL bit to activate the DSP domain isolation wrapper.

k) The MPU sets the GPIO that controls the external analog switch to power down the DSP domain. (It may also be done through an SPI/I^2C interface). The DSP domain is in sleep state.

4) DSP wake-up from sleep state

   In this case, the wake-up is handled entirely by the MPU software. There is no wake-up path through DSP interruptions.

   a) The MPU sets the GPIOs that control the external analog switch to power on the DSP domain. (It can also be done through an SPI/I^2C interface.)

   b) The MPU waits during the ramp-up time of the DSP domain VDD.

   c) The MPU programs the ULPD POWER_CTRL_REG[12] to release the ISOLATION_CONTROL bit to deactivate the DSP domain isolation wrapper.

   d) The MPU enables the DSP clock by asserting the EN_DSPCK bit in the ARM_CKCTL register of CLKRST.

   e) The MPU releases the DSP reset state by deasserting the DSP_EN and DSP_RST bits into the ARM_RSTCT1 register of the CLKRST module.

   f) The DSP boots.

5) MPU wake-up from inactive state

   a) Any unmasked interrupt in the MPU interrupt handler is asynchronously forwarded to ULPD: theWAKEUP_$\overline{REQ}$ signal is asserted.

   b) If DSP is also in inactive state, any unmasked interrupt in the DSP interrupt handler is asynchronously forwarded to the ULPD: the WAKEUP_$\overline{REQ}$ signal is asserted.

   c) ULPD detects wake-up request: WAKEUP_$\overline{REQ}$ or clock request at ULPD

   d) ULPD FSM automatically moves into its awake mode and enables the OMAP3.2 input clock.

   e) The DPLL clock restarts automatically.

   f) The MPU and TC clocks restart automatically.

6) MPU active to inactive state transition

   a) The MPU places the external SDRAM in self-refresh mode (program EMIFF register).

   b) The MPU programs the ARM_IDLECT1, ARM_IDLECT2, and ARM_IDLECT3 registers to ensure that all OMAP MPU and TC subdomain clocks are shut down.

c) The MPU programs the IDLDPLL_ARM and IDLIF _ARM bits of ARM_IDLECT1 to allow TC and DPLL to go into idle mode.

d) The MPU programs the MPU interrupt handlers to mask/unmask interrupts and ensure the wake-up path.

e) The MPU programs the SOFT_DISABLE_REQ_REQ register of ULPD to ensure a wake-up path.

f) The MPU clears POWER_CTRL_REG[4] of ULPD (ULPD big sleep mode)

g) MPU executes the STANDBYWFI instruction.

h) All OMAP MPU and TC clock domains are automatically shutdown by the CLKRST module.

i) DPLL is automatically put in idle by the CLKRST module.

j) CLKRST asserts the CHIP_IDLE signal to the ULPD.

k) The ULPD disables the OMAP3.2 input clock and asserts the CHIP_$\overline{\text{WAKEUP}}$ signal.

l) The ULPD FSM automatically moves into sleep mode. The MPU domain is in inactive state.

7) MPU active to pending state transition

a) The MPU places the external SDRAM in self-refresh mode (program EMIFF register).

b) The MPU programs the ARM_IDLECT1, ARM_IDLECT2, and ARM_IDLECT3 registers to ensure that all OMAP MPU and TC subdomain clocks are shut down.

c) The MPU programs the IDLDPLL_ARM and IDLIF_ARM bits of ARM_IDLECT1 to allow TC and DPLL to go into idle mode.

d) The MPU programs the MPU interrupt handlers to mask/unmask interrupts and ensure the wake-up path.

e) The MPU programs the SOFT_DISABLE_REQ_REQ register of the ULPD to ensure the wake-up path.

f) The MPU sets bit POWER_CTRL_REG[4] of the ULPD (ULPD deep sleep mode).

g) The MPU sets bit POWER_CTRL_REG[0] of the ULPD (LOW_PWR feature enabled).

h) The MPU sets/clears bit POWER_CTRL_REG[9] of the ULPD (oscillator control).

i) The MPU programs the ULPD registers SETUP_ANALOG_CELL2 and SETUP_ANALOG_CELL3 with appropriate stabilization time for the external regulator and the oscillator.

j) The MPU executes the STANDBYWFI instruction.

k) All OMAP MPU and TC clock domains are automatically shutdown by the CLKRST module.

l) The DPLL is automatically put in idle by the CLKRST module.

m) CLKRST asserts the CHIP_IDLE signal to the ULPD.

n) The ULPD disables the OMAP3.2 input clock and asserts the CHIP_$\overline{\text{WAKEUP}}$ signal.

o) The ULPD FSM automatically moves into deep sleep mode.

p) If POWER_CTRL_REG[9] of the ULPD was set, the oscillator is stopped.

q) The ULPD asserts the signal LOW_PWR. This drives the external MPU domain regulator into low-voltage operations at 1.1 V. The MPU domain is in pending state.

r) If the DSP domain was set to inactive state before initiating the MPU domain transition, the final state of the DSP domain is also pending.

8) MPU wake-up from pending state

a) Any unmasked interrupt in the MPU interrupt handler is asynchronously forwarded to ULPD: WAKEUP_$\overline{\text{REQ}}$ signal is asserted.

b) If DSP is also in pending state, any unmasked interrupt in DSP interrupt handler is asynchronously forwarded to ULPD: WAKEUP_$\overline{\text{REQ}}$
signal is asserted.

c) The ULPD detects wake-up requests: WAKEUP_$\overline{\text{REQ}}$ or clock request at ULPD.

d) The ULPD exits its deep sleep mode and releases the LOW_PWR signal.

e) The MPU domain external regulator ramps up.

f) The ULPD FSM automatically moves into its awake mode. Delays programmed in SETUP_ANALOG_CELL2 and SETUP_ANALOG_CELL3 are inserted during the deep sleep to awake transition to allow the oscillator and voltage supplies to stabilize.

g) The ULPD enables the OMAP3.2 input clock.

h) The DPLL clock restarts automatically.

i) The MPU and TC clocks restart automatically.

9) The OMAP5912 chip to sleep state transition

For OMAP5912 the MPU domain sleep state is not a true domain state in the sense that it cannot be entered or exited without the intervention of another device. An external device must shut down the MPU domain regulator.

The wake-up can be performed only through a power-up sequence by asserting the OMAP5912 power up reset.

6 OMAP5912 Power Management Software User Guide

10) Set RNG_CONF_IDLE_MODE, bit 6 of the RESET_CONTROL register, to 0 so that there is no waiting for RNG to enter the idle state to enter CHIP_IDLE:

- Command WR32 0xFFFE 1140 0x0000 003F

Note: By clearing bit 6 in RESET_CONTROL, the RNG idle acknowledge is ignored. This implies that the RNG will not prevent the chip from entering idle but the internal ring oscillators may still consume power. If bit 6 in RESET_CONTROL is not cleared, then the RNG must be configured to shut down its internal ring oscillators before allowing the system to idle. This implies that the latency between awake and idle states is longer.

11) Disable the MPU watchdog timer by writing the sequence F5, A0 in the TIMER_MODE register:

- Command WR32 0xFFFE C808 0x0000 00F5
- Command WR32 0xFFFE C808 0x0000 00A0

12) Disable the DSP watchdog timer by writing the sequence F5, A0 in the TIMER_MODE register:

- Command WR16 0xE100 6808 0x00F5
- Command WR16 0xE100 6808 0x00A0

13) Configure the ARM_IDLECT1 register to turn off DPLL1, LCD, DMA, ARMPER, internal timer, MPUI, ARMXOR, and MPU watchdog clocks when the MPU enters idle state:

- Command WR32 0xFFFE CE04 0x0000 17FF

14) Configure the ARM_IDLECT2 register to turn off the external GPIO clock:

- Command WR32 0xFFFE CE08 0x0000 FDFF

15) Configure the ARM_IDLECT3 register to turn off the TC2, TC1, L3OCP clocks when the MPU enters idle state:

    - Command WR32 0xFFFE CE24 0x0000 FFFF

16) Configure the DSP_IDLECT1 register to turn off GPIO, internal timer, MPUI, DSPPER , DSPXOR, and DSP watchdog clocks when the MPU enters idle state:

    - Command WR32 0xFFFE CE84 0x0000 01CF

17) Configure the DSP_IDLECT2 register to turn off the external UART clock:

    - Command WR32 0xFFFE CE88 0x0000 FFF7

18) Configure the TC_EMIF_SLOW_IF_CONFIG_REG to set global power-down enable and IMIF power-down enable bits:

    - Command WR32 0xFFFE CC0C 0x0000 000C

19) Configure the TC_EMIF_FAST_SDRAM_CONFIG_REG to disable the SDRAM clock and put SDRAM FSM into power-down mode:

    - Command WR32 0xFFFE CC20 0x0C00 B07F

Index

T

U

OMAP5912 Multimedia Processor Direct Memory Access (DMA) Support Reference Guide

Literature Number: SPRU755B
October 2004

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.

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
		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 document describes the direct memory access (DMA) support of the OMAP5912 multimedia processor.

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.

Related Documentation From Texas Instruments

Documentation that describes the OMAP5912 device, related peripherals, and other technical collateral, is available in the OMAP5912 Product Folder on TI's website: www.ti.com/omap5912.

Trademarks

OMAP and the OMAP symbol are trademarks of Texas Instruments.

Contents

Figures

Tables

Examples

Direct Memory Access (DMA) Support

This document describes the direct memory access (DMA) support of the OMAP5912 multimedia processor.

1 DMA Overview

The processor has two DMAs:

- The system DMA is embedded in OMAP3.2. It handles DMA transfers associated with MPU and shared peripherals.
- The DSP DMA is embedded in OMAP3.2. It handles DMA transfers associated with DSP peripherals.

The system DMA can support up to 31 hardware DMA requests. The DSP DMA supports 19 hardware DMA requests.

In OMAP5912, up to 56 different hardware DMA requests can be generated by MPU and shared peripherals. An embedded crossbar, called MPU GDMA handler, allows mapping any of these 56 requests to any of 31 the system DMA request.

Similarly, DSP and shared peripherals can generate up to 28 hardware DMA requests. An embedded crossbar, called DSP GDMA handler allows mapping any of these 28 requests to any of the 19 DSP DMA requests.

Both MPU and DSP GDMA handlers are configured through registers implemented in the configuration module.

2 GDMA Handlers

MPU and DSP peripherals DMA request mapping is configurable through two blocks called GDMA handlers. MPU GDMA handler is described in section 2.1. DSP GDMA handler is described in section 2.2.

2.1 MPU GDMA Handler

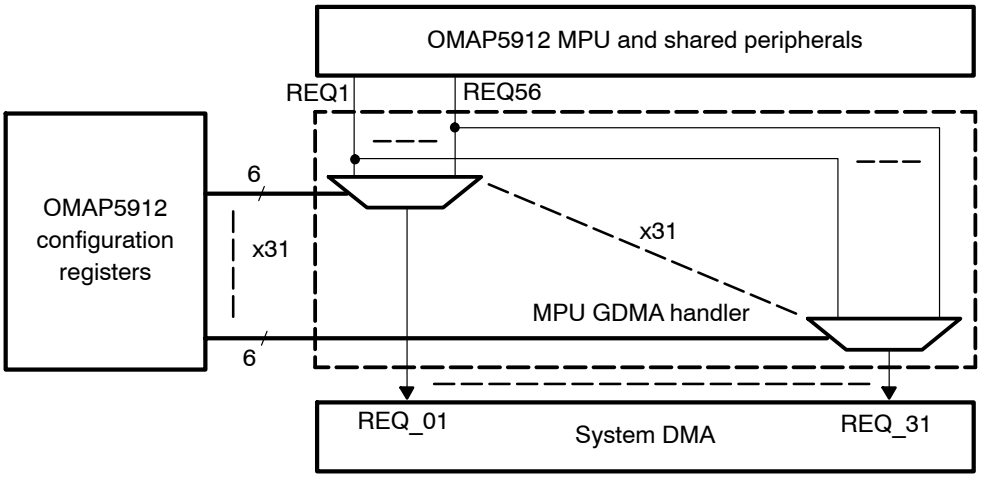
The MPU and shared peripherals control up to 56 DMA requests, whereas the system DMA can handle only 31 DMA requests.

The GDMA handler acts as a crossbar so that each of the incoming DMA requests can be remapped to any of the system DMA requests.

The mapping of each DMA request is done through the FUNC_MPU_DMA_x registers located in the configuration module.

A 6-bit field is associated with each system channel so that any incoming DMA request is remapped to the proper system DMA request. See Figure 1-12 for a description of the MPU GDMA handler.

Figure 1. MPU GDMA Handler



The default configuration after reset ensures compatibility with the previous OMAP5912 generation. Programmers have flexibility to remap up to 31 requests according to the application task requirements. Table 1 describes the mapping between MPU and shared peripherals DMA requests and the MPU GDMA handler inputs.

Table 1. MPU GDMA Handler Mapping

MPU GDMA Handler Input Line	Peripheral Request
REQ1	MCSI1 TX
REQ2	MCSI1 RX
REQ3	I^2C RX
REQ4	I^2C TX
REQ5	External DMA request 0
REQ6	External DMA request 1
REQ7	μWire TX
REQ8	McBSP1 DMA TX
REQ9	McBSP1 DMA RX
REQ10	McBSP3 DMA TX
REQ11	McBSP3 DMA RX
REQ12	UART1 TX
REQ13	UART1 RX
REQ14	UART2 TX
REQ15	UART2 RX
REQ16	McBSP2 TX
REQ17	McBSP2 RX
REQ18	UART3 TX
REQ19	UART3 RX
REQ20	Camera IF RX
REQ21	MMC/SDIO1 TX
REQ22	MMC/SDIO1 RX
REQ24	IRQ_LCD_LINE
REQ26	USB W2FC RX0
REQ27	USB W2FC RX1
REQ28	USB W2FC RX2
REQ29	USB W2FC TX0

Table 1. MPU GDMA Handler Mapping (Continued)

MPU GDMA Handler Input Line	Peripheral Request
REQ30	USB W2FC TX1
REQ31	USB W2FC TX2
REQ33	SPI TX
REQ34	SPI RX
REQ38	CMT-APE TX (channel 0)
REQ39	CMT-APE RV (channel 0)
REQ40	CMT-APE TX (channel 1)
REQ41	CMT-APE RV (channel 1)
REQ42	CMT-APE TX (channel 2)
REQ43	CMT-APE RV (channel 2)
REQ44	CMT-APE TX (channel 3)
REQ45	CMT-APE RV (channel 3)
REQ46	CMT-APE TX (channel 4)
REQ47	CMT-APE RV (channel 4)
REQ48	CMT-APE TX (channel 5)
REQ49	CMT-APE RV (channel 5)
REQ50	CMT-APE TX (channel 6)
REQ51	CMT-APE RV (channel 6)
REQ52	CMT-APE TX (channel 7)
REQ53	CMT-APE RV (channel 7)
REQ54	MMC/SDIO2 TX
REQ55	MMC/SDIO2 RX
REQ58	Unconnected
REQ59	Unconnected
REQ60	Unconnected
REQ61	Unconnected

Table 1. MPU GDMA Handler Mapping (Continued)

MPU GDMA Handler Input Line	Peripheral Request
REQ62	Unconnected
REQ63	Unconnected
REQ64	Unconnected

2.1.1 MPU GDMA Handler Configuration

The mapping of the system DMA requests is done through the GDMA registers (shown in Table 2), which are implemented in the configuration module. Table 3 through Table 9 describe the register bits.

The values programmed into these registers represent a zero-based numbering of the DMA request. Thus, peripheral request number *n* is programmed as *n-1*, not *n*.

Table 2. MPU GDMA Handler Control Registers

Base Address = 0xFFFE 1000			
Name	**Description**	**R/W**	**Offset**
FUNC_MUX_MPU_DMA_A	Controls mapping for system DMA requests 1 to 5.	R/W	0xEC
FUNC_MUX_MPU_DMA_B	Controls mapping for system DMA requests 6 to 10.	R/W	0xF0
FUNC_MUX_MPU_DMA_C	Controls mapping for system DMA requests 11 to 15.	R/W	0xF4
FUNC_MUX_MPU_DMA_D	Controls mapping for system DMA requests 16 to 20.	R/W	0xF8
FUNC_MUX_ARM_DMA_E	Controls mapping for system DMA requests 21 to 25.	R/W	0xFC
FUNC_MUX_ARM_DMA_F	Controls mapping for system DMA requests 26 to 30.	R/W	0x100
FUNC_MUX_ARM_DMA_G	Controls mapping for system DMA request 31.	R/W	0x104

Table 3. Functional Multiplexing MPU DMA A Register (FUNC_MUX_MPU_DMA_A)

Base Address = 0xFFFE 1000, Offset Address = 0xEC				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:30	RESERVED	Reserved.	R/W	0x0
29:24	CONF_ARM_DMA_REQ_05	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(5). *n* is between 0 and 55.	R/W	0x04
23:18	CONF_ARM_DMA_REQ_04	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(4). *n* is between 0 and 55.	R/W	0x03
17:12	CONF_ARM_DMA_REQ_03	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(3). *n* is between 0 and 55.	R/W	0x02
11:6	CONF_ARM_DMA_REQ_02	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(2). *n* is between 0 and 55.	R/W	0x01
5:0	CONF_ARM_DMA_REQ_01	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(1). *n* is between 0 and 55.	R/W	0x00

This register controls the system DMA crossbar and defines the mapping of MPU peripheral DMA requests 1 through 5 to system DMA requests. 56 MPU peripheral DMA requests can be mapped to the 31 system DMA controller requests. The values programmed in the register represent a zero-based numbering of peripheral DMA_REQn (starting with 1). Peripheral DMA_REQ1 is written as zero. For example, if bits 5:0 are equal to 3, then n+1 = 4, and DMA MPU peripheral request source 4 (I^2C TX) is connected to system DMA_REQ(1).

Table 4. Functional Multiplexing MPU DMA B Register (FUNC_MUX_MPU_DMA_B)

Base Address = 0xFFFE 1000, Offset Address = 0xF0				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:30	RESERVED	Reserved.	R/W	0x0
29:24	CONF_ARM_DMA_REQ_10	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(10). *n* is between 0 and 55.	R/W	0x09
23:18	CONF_ARM_DMA_REQ_09	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(9). *n* is between 0 and 55.	R/W	0x08
17:12	CONF_ARM_DMA_REQ_08	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(8). *n* is between 0 and 55.	R/W	0x07
11:6	CONF_ARM_DMA_REQ_07	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(7). *n* is between 0 and 55.	R/W	0x06
5:0	CONF_ARM_DMA_REQ_06	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(6). *n* is between 0 and 55.	R/W	0x05

Table 5. Functional Multiplexing MPU DMA C Register (FUNC_MUX_MPU_DMA_C)

Base Address = 0xFFFE 1000, Offset Address = 0xF4				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:30	RESERVED	Reserved.	R/W	0x0
29:24	CONF_ARM_DMA_REQ_15	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(15). *n* is between 0 and 55.	R/W	0x0E
23:18	CONF_ARM_DMA_REQ_14	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(14). *n* is between 0 and 55.	R/W	0x0D

Table 5. Functional Multiplexing MPU DMA C Register (FUNC_MUX_MPU_DMA_C) (Continued)

Bit	Name	Function	R/W	Reset
17:12	CONF_ARM_DMA_REQ_13	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(13). *n* is between 0 and 55.	R/W	0x0C
11:6	CONF_ARM_DMA_REQ_12	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(12). *n* is between 0 and 55.	R/W	0x0B
5:0	CONF_ARM_DMA_REQ_11	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(11). *n* is between 0 and 55.	R/W	0x0A

Table 6. Functional Multiplexing MPU DMA D Register(FUNC_MUX_MPU_DMA_D)

Base Address = 0xFFFE 1000, Offset Address = 0xF8				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:30	RESERVED	Reserved.	R/W	0x0
29:24	CONF_ARM_DMA_REQ_20	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(20). *n* is between 0 and 55.	R/W	0x13
23:18	CONF_ARM_DMA_REQ_19	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(19). *n* is between 0 and 55.	R/W	0x12
17:12	CONF_ARM_DMA_REQ_18	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(18). *n* is between 0 and 55.	R/W	0x11
11:6	CONF_ARM_DMA_REQ_17	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(17). *n* is between 0 and 55.	R/W	0x10
5:0	CONF_ARM_DMA_REQ_16	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(16). *n* is between 0 and 55.	R/W	0x0F

Table 7. Functional Multiplexing MPU DMA E Register (FUNC_MUX_MPU_DMA_E)

Base Address = 0xFFFE 1000, Offset Address = 0xFC				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:30	RESERVED	Reserved.	R/W	0x0
29:24	CONF_ARM_DMA_REQ_25	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(25). *n* is between 0 and 55.	R/W	0x18
23:18	CONF_ARM_DMA_REQ_24	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(24). *n* is between 0 and 55.	R/W	0x17
17:12	CONF_ARM_DMA_REQ_23	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(23). *n* is between 0 and 55.	R/W	0x16
11:6	CONF_ARM_DMA_REQ_22	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(22). *n* is between 0 and 55.	R/W	0x15
5:0	CONF_ARM_DMA_REQ_21	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(21). *n* is between 0 and 55.	R/W	0x14

Table 8. Functional Multiplexing MPU DMA F Register (FUNC_MUX_ARM_DMA_F)

Base Address = 0xFFFE 1000, Offset Address = 0x100				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:30	RESERVED	Reserved.	R/W	0x0
29:24	CONF_ARM_DMA_REQ_30	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(30). *n* is between 0 and 55.	R/W	0x1D
23:18	CONF_ARM_DMA_REQ_29	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(29). *n* is between 0 and 55.	R/W	0x1C
17:12	CONF_ARM_DMA_REQ_28	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(28). *n* is between 0 and 55.	R/W	0x1B

Table 8. Functional Multiplexing MPU DMA F Register (FUNC_MUX_ARM_DMA_F) (Continued)

Bit	Name	Function	R/W	Reset
11:6	CONF_ARM_DMA_REQ_27	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(27). *n* is between 0 and 55.	R/W	0x1A
5:0	CONF_ARM_DMA_REQ_26	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(26). *n* is between 0 and 55.	R/W	0x19

Table 9. Functional Multiplexing MPU DMA G Register (FUNC_MUX_ARM_DMA_G)

Base Address = 0xFFFE 1000, Offset Address = 0x104				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:6	RESERVED	Reserved.	R/W	0x0000000
5:0	CONF_ARM_DMA_REQ_31	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(31). *n* is between 0 and 55.	R/W	0x1E

2.2 DSP GDMA Handler

The various DSP and shared peripherals control up to 28 DMA requests, whereas the DSP DMA in the OMAP3.2 can handle only 19 DMA requests.

The DSP GDMA handler acts as a crossbar so that each of the incoming DMA requests can be remapped to any of the DSP DMA requests.

The mapping of each DMA request is done through the FUNC_DSP_DMA_x registers located in the configuration module.

A 5-bit field is associated with each DSP channel so that any incoming DMA request is remapped to the proper DSP DMA request.

The default configuration from reset ensures compatibility with the previous OMAP5912 generation. Programmers have the flexibility to remap up to 19 requests according to the application task requirements. See Figure 2 for a description of the DSP GDMA handler and Table 10 for DSP GDMA mapping.

Figure 2. DSP GDMA Handler

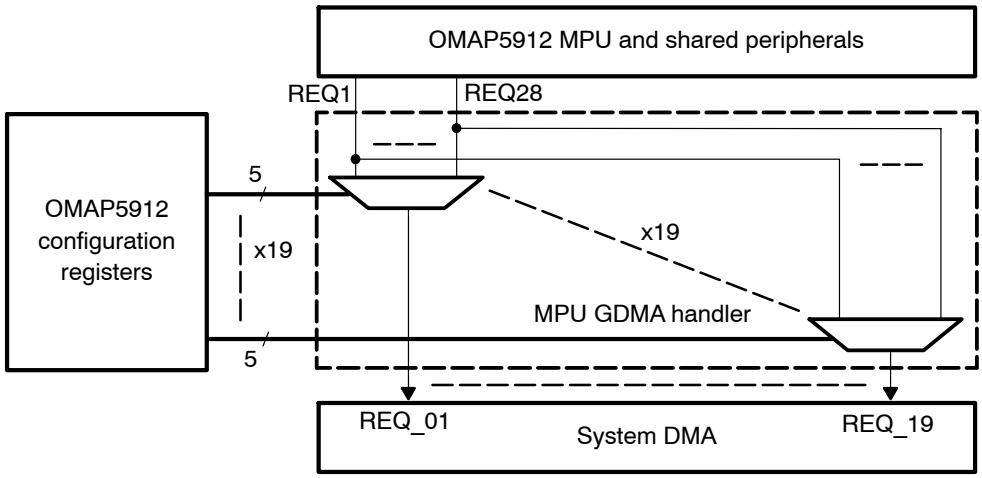


Table 10. DSP GDMA Mapping

DSP GDMA Handler Input Line	Peripheral Request
REQ1	MCSI1 TX
REQ2	MCSI1 RX
REQ3	MCSI2 TX
REQ4	MCSI2 RX
REQ5	MMC/SDIO2 TX
REQ6	MMC/SDIO2 RX
REQ7	Reserved
REQ8	McBSP1 TX
REQ9	McBSP1 RX
REQ10	McBSP3 TX
REQ11	McBSP3 RX
REQ12	UART1 TX
REQ13	UART1 RX
REQ14	UART2 TX
REQ15	UART2 RX
REQ16	I^2C RX
REQ17	I^2C TX

Table 10. DSP GDMA Mapping (Continued)

DSP GDMA Handler Input Line	Peripheral Request
REQ18	UART3 TX
REQ19	UART3 RX
REQ20	CMT-APE TX (channel 1)
REQ21	CMT-APE RV (channel 1)
REQ22	CMT-APE TX (channel 2)
REQ23	CMT-APE RV (channel 2)
REQ25	SPI TX
REQ26	SPI RX
REQ27	McBSP2 TX
REQ28	McBSP2 RX
REQ29	Unconnected
REQ30	Unconnected
REQ31	Unconnected
REQ32	Unconnected

2.3 DSP GDMA Handler configuration

The mapping of the DSP DMA requests is done through the FUNC_MUX_DSP_DMA_X registers (shown in Table 11), which are implemented in the configuration module. Table 12 through Table 15 describe the register bits.

The values programmed into these registers represent a zero-based numbering of the DMA request. Thus, peripheral request number *n* is programmed as *n-1*, not *n*.

Table 11. DSP DMA Handler Control Registers

Base Address = 0xFFFE 1000			
Name	**Description**	**R/W**	**Offset**
FUNC_MUX_DSP_DMA_A	Controls mapping for DSP DMA requests 1 to 6	R/W	0xD0
FUNC_MUX_DSP_DMA_B	Controls mapping for DSP DMA requests 7 to 12	R/W	0xD4
FUNC_MUX_DSP_DMA_C	Controls mapping for DSP DMA requests 13 to 18	R/W	0xD8
FUNC_MUX_DSP_DMA_D	Controls mapping for DSP DMA request 19	R/W	0xDC

Table 12. Functional Multiplexing DSP DMA A Register (FUNC_MUX_DSP_DMA_A)

Base Address = 0xFFFE 1000, Offset Address = 0xD0				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:30	RESERVED	Reserved	R/W	0x0
29:25	CONF_DSP_DMA_REQ_06	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(6). *n* is between 0 and 27.	R/W	0x05
24:20	CONF_DSP_DMA_REQ_05	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(5). *n* is between 0 and 27.	R/W	0x04
19:15	CONF_DSP_DMA_REQ_04	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(4). *n* is between 0 and 27.	R/W	0x03
14:10	CONF_DSP_DMA_REQ_03	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(3). *n* is between 0 and 27.	R/W	0x02
9:5	CONF_DSP_DMA_REQ_02	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(2). *n* is between 0 and 27.	R/W	0x01
4:0	CONF_DSP_DMA_REQ_01	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(1). *n* is between 0 and 27.	R/W	0x00

This register controls the DSP DMA crossbar and defines the mapping of DSP peripheral DMA requests 1 through 6 to OMAP3.2 DSP DMA requests. 28 DSP peripheral DMA requests can be mapped to the 19 DSP DMA controller requests. The values programmed in the register represent a zero-based numbering of DMA_REQn (starting with 1). Thus, peripheral DMA_REQ1 is written as zero. For example, if bits 4:0 are equal to 3, then n+1=4, and DSP DMA_REQ(1) maps to DMA DSP peripheral request source 4 (MCSI2 RX).

Table 13. Functional Multiplexing DSP DMA B Register (FUNC_MUX_DSP_DMA_B)

Base Address = 0xFFFE 1000, Offset Address = 0xD4				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:30	RESERVED	Reserved	R/W	0x0
29:25	CONF_DSP_DMA_REQ_12	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(12). *n* is between 0 and 27.	R/W	0x0B
24:20	CONF_DSP_DMA_REQ_11	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(11). *n* is between 0 and 27.	R/W	0x0A
19:15	CONF_DSP_DMA_REQ_10	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(10). *n* is between 0 and 27.	R/W	0x09
14:10	CONF_DSP_DMA_REQ_09	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(9). *n* is between 0 and 27.	R/W	0x08
9:5	CONF_DSP_DMA_REQ_08	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(8). *n* is between 0 and 27.	R/W	0x07
4:0	CONF_DSP_DMA_REQ_07	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(7). *n* is between 0 and 27.	R/W	0x06

Table 14. Functional Multiplexing DSP DMA C Register (FUNC_MUX_DSP_DMA_C

Base Address = 0xFFFE 1000, Offset Address = 0xD8				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:30	RESERVED	Reserved	R/W	0x0
29:25	CONF_DSP_DMA_REQ_18	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(18). *n* is between 0 and 27.	R/W	0x11
24:20	CONF_DSP_DMA_REQ_17	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(17). *n* is between 0 and 27.	R/W	0x10
19:15	CONF_DSP_DMA_REQ_16	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(16). *n* is between 0 and 27.	R/W	0x0F
14:10	CONF_DSP_DMA_REQ_15	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(15). *n* is between 0 and 27.	R/W	0x0E
9:5	CONF_DSP_DMA_REQ_14	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(14). *n* is between 0 and 27.	R/W	0x0D
4:0	CONF_DSP_DMA_REQ_13	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(13). *n* is between 0 and 27.	R/W	0x0C

Table 15. Functional Multiplexing DSP DMA D Register (FUNC_MUX_DSP_DMA_D)

Base Address = 0xFFFE 1000, Offset Address = 0xDC				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:5	RESERVED	Reserved for future expansion	R/W	0x0000000
4:0	CONF_DSP_DMA_REQ_19	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(19). *n* is between 0 and 27.	R/W	0x12

3 System DMA

The system DMA is designed to off-load the block data transfer function from the MPU.

The OMAP 3.2 system DMA controller consists of:

- Sixteen logical channels plus one LCD logical channel
- Seven physical ports plus one for configuration
- Three physical channels plus one LCD dedicated physical channel

The ports are connected to the OCP-T1 and OCP-T2 targets, the external memory, the TIPB bridge, the MPUI, and one dedicated port connected to an LCD controller. The system DMA controller can be controlled via the MPU private TIPB or by an external host via the OCP-I port.

The system DMA controller is designed for low-power operation. It is partitioned into several clock domains where each clock domain is enabled only when it is used. All clocks are disabled when no DMA transfers are active.

Five different logical channels types are supported; each one represents a specific feature set.

- LCh-2D for memory to memory transfers, 1D and 2D
- LCh-P for peripheral transfers
- LCh-PD for peripheral transfers on a dedicated channel
- LCh-G for graphical transfers/operations
- LCh-D for display transfers

The features available are:

- Support for up to four address modes:
  - Constant
  - Post-incremented mode
  - Single-indexed
  - Double-indexed
- Different indexing for source-respective destination
- Logical channel chaining
- Software enabling
- Hardware enabling– 31 DMA request lines available
- Logical channel interleaving
- Logical channel preemption

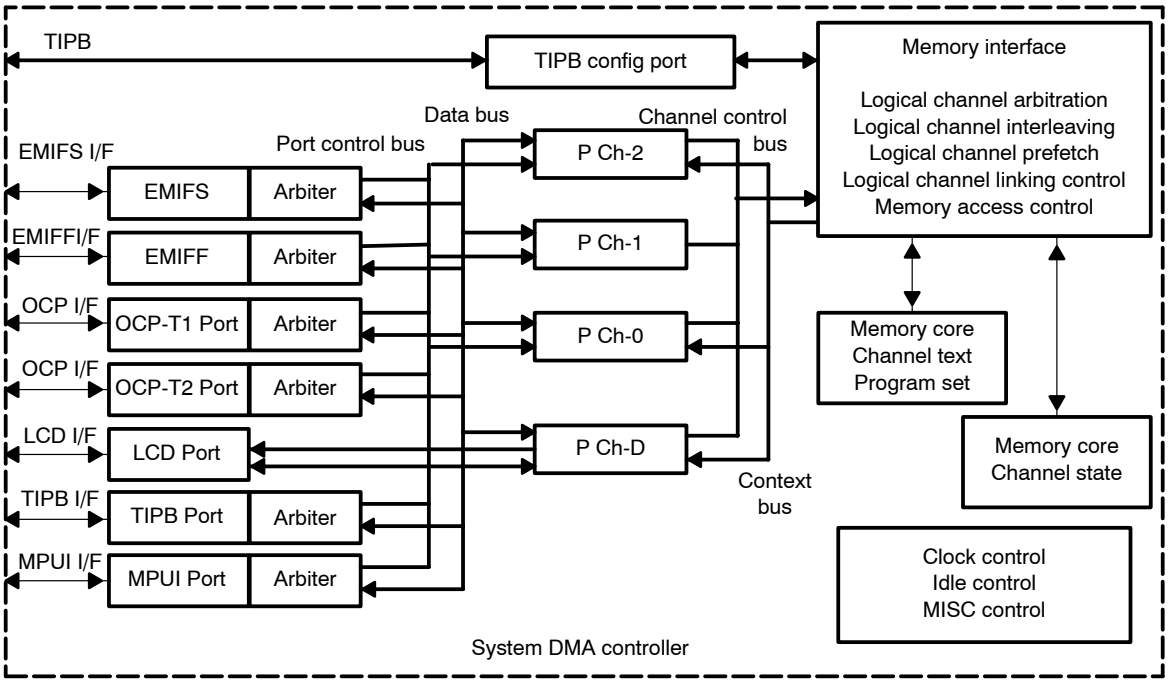
- Two choices of logical channel arbitration of physical resources: round robin or fixed
- Two levels of logical channel priority
- Constant fill
- Transparent copy
- Rotation 0°, 90°, 180°, and 270°
- Seven ports enabling:
  - Memory-to-memory transfers
  - Peripheral-to-memory transfers
  - Memory-to-peripheral transfers
  - Peripheral-to-peripheral transfers
- Binary backward-compatible by default configuration
- Up to four logical channels active in parallel

The logical channel dedicated to the display, LCh-D, has several additional features.

- Channel can be shared by two LCD controllers
- Supports both single and dual block modes
- Supports separate indexing and numbering for dual block mode for both elements and frames

A graphical overview of the system DMA and its external connections is shown in Figure 3.

Figure 3. System DMA Controller Simplified Block Diagram



3.1 Functional Description

This section describes the system DMA capabilities and programming.

The configuration of the system DMA logical channel registers can be done in any order with the exception of the enable bit in the channel control register, DMA_CCR. This bit enables all logical channel types, so this must be the last thing done when configuring a logical channel (LCh). During the time an LCh is enabled it is not allowed to change its configuration registers, which causes undefined effects. All global system DMA configuration registers must be changed before any LChs are enabled; if not, it undefined effects result.

The dedicated LCD channel has some additional features and different channel behavior compared to the generic channels. For more details, see section 3.2.1, *Display Logical Channel*. Some channel features and behavior are common to both generic and LCD channels; this section describes them and indicates which are supported by the LCD channel.

3.1.1 Logical Channel Types

The OMAP system DMA is based on logical channels (LChs).

Each generic LCh can be configured to one of four different logical channel types; the dedicated LCD LCh can only be configured to LCh type D.

Logical channel types (LCh types) supported are:

- LCh-2D for nonsynchronized transfers (memory transfers, 1D and 2D)
- LCh-P for synchronized transfers (mostly peripheral transfers)
- LCh-PD similar to LCh-P but runs on a dedicated physical channel
- LCh-G for graphical transfers/operations
- LCh-D for display transfers

Note:

The logical channel dedicated for the display, LCh-D, must be configured to LCh type LCh-D.

Each logical channel type represents a specific subset of the system DMA features list. Even if the requirements of a specific transfer can fit into several LCh-types, it is important to select the correct LCh-type. Table 16 summarizes features per logical channel type.

Table 16. Summary of Features Per Logical Channel Type (Without LCh-D)

Supported Transfer Features		LCh-2D (2D)	LCh-P (P)	LCh-PD (PD)	LCh-G (G)
LCh Control	*Transfer type*				
	❑ Synchronized	√	√	√	√
	Two levels of priority	√	√	√	√
	Preemption at element boundary	√	√		√
	LCh interleaving on DMA request		√	√	
	Linking logical channel capability	√	√	√	√
Addressing	*Types of addressing modes for SRC and DST*				
	❑ Constant mode	√	√	√	√
	❑ Post-increment mode	√	√	√	√
	❑ Single-indexed mode with separate SRC/DST index	√	See Note	See Note	√
	❑ Double-indexed mode with separate SRC/DST index	√	See Note	See Note	√
Other	*Other LCh-type features*				
	Constant fill	√			√
	Transparent color				√

Note: Supported only on one end of the transfer (memory side—can be a source or a destination).

A separate table applies for the LCh-D (D), because several elements differ between that channel and these generic LChs. See Table 29, *Features Summary for LCh-D*, in section 3.2.1.

The abbreviations in parentheses in the Table 16 headings are used throughout this document as a guide to features supported for different LCh types.

3.1.2 OMAP 3.2 System DMA Instances

The OMAP 3.2 system DMA has 16 generic logical channels plus one logical channel dedicated to the LCD controller as well as three physical channels plus one dedicated to the LCD controller (PCh-0, PCh-1, PCh-2 plus PCh-D).

PCh-0 and PCh-1 are always dynamically allocated to any active LCh of type LCh-2D, LCh-G, and LCh-P. Dynamic allocation means that the physical channel that becomes free first services the active logical channel. PCh-2 is different from PCh-0 and PCh-1, because it is possible to configure one or several synchronized LChs to only use this physical channel. For example, the physical channel can be dedicated to one LCh or to several interleaved synchronized channels. This is done by specifying an LCh as LCh-PD type.

The fourth channel, PCh-D, is specifically designed for transfers to the display and can only be used by the dedicated LCh-D. LCh-D must always be configured as type LCh-D.

Table 17. Associated Physical Channels Per Logical Channel Type

LCh-Type	PCh Assigned
LCh-2D	PCh-0 or PCh-1
LCh-P	(dynamic allocation)
LCh-G	
LCh-PD	PCh-2
LCh-D	PCh-D

There are six global registers: PCh status register, PCh ID register, and one PCh status register per PCh channel. See respective register descriptions for more details.

3.1.3 Synchronized Channel

LCh Types Supporting Synchronized Channels	2D	P	PD	G	D
	√	√	√	√	√

A synchronized channel (hardware activated) is a channel that only becomes active when it is enabled by software and subsequently receives a DMA request signal either from an peripheral or from an dedicated ball. There are 31 possible hardware DMA request sources, 1 to 31. A hardware DMA request cannot be shared between several concurrent channels (enabled and active). However, a hardware DMA request can be shared between different channels if they are part of a chain. The peripheral hardware request associated with each of the 31 system DMA request is controllable through MPU GDMA handler configuration.

Software must program the five SYNC bits located in the channel control register, DMA_CCR, to configure the external DMA request that activates the channel. The logical channel becomes a synchronized channel when this field is set to reflect the number of the request line. Remember that no DMA request can be mapped as 0, which is reserved to specify a nonsynchronized transfer.

Each time a DMA request is received for a synchronized channel, the logical channel is activated and a block of data is transferred when a physical channel is assigned to it. This block of data can be:

- An element

  A complete element, which is defined by Data_type. For example, 8/16/32 bits are transferred in response to a DMA request.

- An entire frame

  A complete frame of several elements is transferred in response to a DMA request.

- An entire block

  A complete block of several frames is transferred in response to a DMA request.

One DMA request can trigger several logical channels at the same time.

To configure an LCh to synchronize by element, frame, or block, program the frame synchronization (FS) bit in the DMA_CCR register and the block synchronization (BS) bit in the DMA_CCR2 register. If both bits are set to 0, the channel is synchronized by element. Setting both bits to 1 causes undefined effects.

- To configure an LCh to transfer *one element* per DMA request:

  1) Configure the data type, also referenced as element size (ES), in the field Data_type in the channel source destination parameter register (DMA_CSDP).
  2) Configure the number of transfers (elements) to take place before the LCh gets disabled again in the channel element number register (DMA_CEN).
  3) Configure both FS and BS to 0.

- To configure an LCh to transfer *one frame* per DMA request:

  1) Configure the element size as described above.
  2) Configure the element number as described above. This represents the number of elements sent per frame, hence per DMA request.
  3) Configure the number of transfers (frames) to take place before the LCh gets disabled again in the channel frame number registers (DMA_CFN).
  4) Configure FS to 1 and BS to 0.

- To configure an LCh to transfer *one block* per DMA request:
  1) Configure the element size as described above.
  2) Configure the element number as described above. This represents the number of elements sent per frame.
  3) Configure the frame number as described above. This represents the number of frames sent per block.
  4) Configure FS to 0 and BS to 1.

It is also possible to stop a transfer by disabling the channel. This is done by resetting the ENABLE bit in the DMA_CCR register.

3.1.4 Physical Ports

The system DMA can access different data types, depending on which DMA port is used and what the memory or peripheral is supporting. The system DMA ports can support different types of protocols and bus widths. This is important to understand, because it governs which type of transfer an LCh can perform. Table 18 summarizes the type of transfers that each port supports.

Table 18. OMAP3.2 System DMA Supported Interface Port Type Table

Port _Name	Port Functionality
OMAP EMIFF port	Supports: ❑ 8/16/32-bit, 4x32-bit burst serialized access ❑ 4x32-bit burst serialized access, if it is the source port of the LCD channel.
OMAP EMIFS port	
OMAP OCP-T1 port	
OMAP OCP-T2 port	
OMAP TIPB port	Supports: ❑ 8/16/32-bit serialized access
OMAP MPUI port	
OMAP LCD port	Supports: ❑ 16-bit serialized access for OMAP LCD controller ❑ 32-bit serialized access for OMAP external LCD controller
OMAP TIPB configuration port	TIPB configuration interface for system DMA. It supports 16-bit serialized access.

Six of the physical ports can be selected to be the source and/or destination of a DMA transfer. The DMA LCD port can only function as a destination port. The TIPB configuration port cannot be used as a source or destination port.

- TIPB configuration interface:
  - Used by the MPU to control/configure the system DMA. Cannot be used as source or destination in a DMA transfer.
- External slow memory interface (Flash/ROM):
  - DMA accesses can be either single (8/16/32-bit) or burst (4x32-bit). If burst access is used and data packets are not 16-byte aligned, then single accesses (8/16/32-bit) are performed.
- External fast memory interface (SDRAM, DDR):
  - DMA accesses can be either single (8/16/32-bit) or burst (4x32-bit). If burst access is used and data packets are not 16-byte aligned, then single accesses (8/16/32-bit) are performed.
- OCP-T1 and OCP-T2 interface (Test RAM, Camera peripheral):
  - DMA accesses can be either single (8/16/32-bit) or burst (4x32-bit). If burst access is used and data packets are not 16-byte aligned, then single accesses (8/16/32-bit) are performed.
- TIPB interface (to peripherals via TIPB bridge):
  - All DMA accesses are done in single-access mode (8/16/32 bits).
- MPU interface (to SARAM and DARAM inside DSP subsystem):
  - All DMA accesses are done in single access mode (8/16/32 bits). For 32-bit transfer, the MPU interface does the packing and unpacking.
- LCD interface
  - Port to the OMAP embedded LCD controller. Supports 16-bit access for the OMAP LCD controller.

For all ports, only the number of programmed bytes are transferred; that is, there are no trailing or dirty bytes at the end of transfer.

Table 19 provides possible source ports (SRC), destination ports (DST), and data transfers.

Table 19. Possible Data Transfer

	DST						
SRC	**EMIFS**	**EMIFF**	**OCP-T1**	**OCP-T2**	**TIPB Bridge**	**MPUI**	**LCD[1]**
EMIFS	Yes	Yes	Yes	Yes	Yes	Yes	No
EMIFF	Yes	Yes	Yes	Yes	Yes	Yes	Yes
OCP-T1	Yes	Yes	Yes	Yes	Yes	Yes	Yes
OCP-T2	Yes	Yes	Yes	Yes	Yes	Yes	Yes
TIPB bridge	Yes	Yes	Yes	Yes	Yes	Yes	No
MPUI	Yes	Yes	Yes	Yes	Yes	Yes	No
LCD[1]	No	No	No	No	No	No	No

† Used for the OMAP internal LCD controller.

The port to use for source respective destination is configured in the channel source destination parameters register, DMA_CSDP. The data type is also configured in this register.

Note:

No address space check is performed by the system DMA. All addressing outside the source and destination memory space causes undefined effects.

3.1.5 Port Channel Scheduling

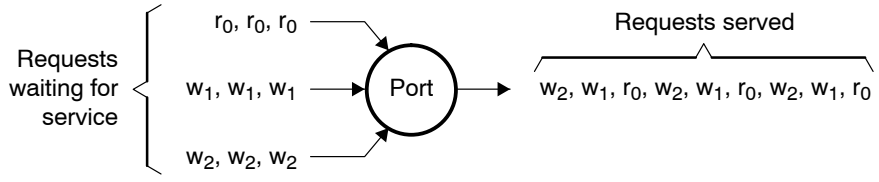
Each DMA logical channel can be configured independently from other logical channels. Each DMA physical channel can have its own port or share it with other physical channels. If physical channels share a DMA port, a port arbiter prioritizes physical channel access.

The system DMA port arbiters follow a round robin scheme.

In Figure 4, a DMA port services three DMA physical channel requests:

- Physical channel 0 as a source port (read requests r_0)
- Physical channel 1 as a destination port (write requests w_1)
- Physical channel 2 as a destination port (write requests w_2)

Figure 4. Time Sharing Access on a System DMA Port



The system DMA port arbiters use a round robin scheme, but they are also dependent on the LCh priority. For more information about LCh priority, see the section on *Logical Channel Priorities.*

The arbiters use the following scheme each time the previous DMA port access has finished:

1) Service all high-priority PChs using a round robin scheme. A low-priority transfer currently being served by the same port is not aborted or suspended by a high-priority transfer. The high-priority transfer waits.

   Port arbitration is each-access based. If the current access is burst access, then the arbitration is on the boundary of burst. The higher priority PCh transfer takes the port as soon as the next available slot is open at the burst boundary.

2) If no highly prioritized PCh is present, then all PCh are arbitrated/served according to the round robin scheme.

3.1.6 Logical Channel Scheduling

A logical channel is marked as an active channel only when either of the following conditions has been meet:

- Case 1**:**
  - The logical channel is a synchronized channel.
  - The logical channel enable field is set.
  - The associated DMA request is triggered.
- Case 2**:**
  - The logical channel is a nonsynchronized channel.
  - The logical channel enable field is set.

A logical channel can be assigned to a physical channel only when the logical channel is active.

When a physical channel is granted, the DMA controller loads the physical channel with the logical channel configuration register set, which controls the data transfer through the system DMA. At the end of a channel transfer, the current channel status is updated into the logical channel status register, and the current logical channel enable is cleared if it is a nonsynchronized channel.

Logical Channel Scheduling Scheme

Each physical channel can only serve one logical channel at a time. If several logical channels are active and waiting to be served, they are interleaved based on an arbitration scheme in a TDMA manner. The supported arbitration schemes are:

- ❏ Round robin scheduling
- ❏ Fixed scheduling from low LCH ID to high LCH ID

The scheme to be used can be controlled by software on a global basis.

In the global control register, (DMA_GCR), the ROUND_ROBIN_DISABLE bit controls which scheme to follow. This bit can only be changed when the DMA is quiescent (that is, when no LChs are enabled). Any change of this bit when a LCh is enabled causes undefined behaviors.

Logical Channel Priorities

LCh Types Supporting this Feature	2D	P	PD	G	D
	√	√	√	√	N/A

Each logical channel can be given a low or high priority level. When a DMA physical channel receives requests from several logical channels, it looks at their priorities. The physical channel assignment to logical channels follow the scheme described below:

1) Requests from high-priority logical channels are served first. A higher priority logical channel can preempt the current on-going low-priority logical transfer and start the higher priority logical channel transferring. The preempted logical channel continues the transfer as soon as all high-priority channels are served. A transfer can be preempted on element boundary. See section 3.1.9, *Logical Channel Preempting*, for more information on channel preempting.

2) Requests from low-priority logical channels are served only if there are no requests from high-priority logical channels. This can occur if no high-priority logical channels are activated, or if the high-priority logical channels are waiting for a synchronization event.

3) Requests of the same priority level are served in a round robin or fixed scheduling scheme, as mentioned earlier in this document.

Use the PRIO bit in the logical channel register DMA_CCR to configure the LCh priority.

3.1.7 Logical Channel Interleaving For Synchronized Transfers

LCh Types Supporting this Feature	2D	P	PD	G	D
		√	√		

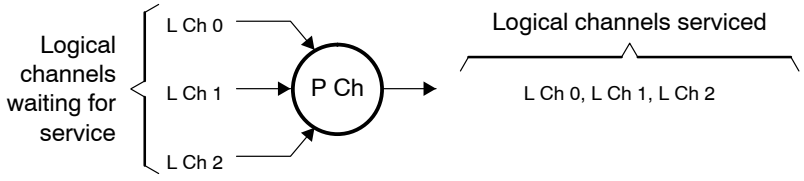
Logical channel interleaving is a term used when more than one synchronized channel shares the same physical channel. A synchronized channel is only active and requesting access to a physical channel when a DMA request is received.

When a DMA request is served but more data remains to be transmitted, the logical channel stays enabled waiting for next DMA request. During this time, the physical channel is released if the LCh-type currently running supports logical channel interleaving.

A nonsynchronized LCh can use the physical channel as well (i.e., interleave), but it runs to finish before it releases the channel again. This means it is the responsibility of the software to configure nonsynchronized transfers in small enough blocks so that synchronized LChs are served in time. This is true only if all PCh are occupied.

Figure 5 shows an example of a logical channel interleaving scheme for three synchronized LChs where the DMA requests are received in order: LCh 0 request, LCh 1 request, and LCh 2 request.

Figure 5. Logical Channel Interleaving on Channel Boundary With the Same Priority



It is possible to disable the interleaving on a logical channel basis. If this is done for a specific logical channel, that logical channel does not release the physical channel between the DMA requests until all the transfers are done or until the LCh is disabled by software. This is a way of dedicating a physical channel to a logical channel of type LCh-P or LCh-PD.

To disable LCh interleave for synchronized transfers, set the LCH_INTERLEAVE_DISABLE bit in the logical channel control register, DMA_LCH_CTRL.

A nonsynchronized transfer is not affected by the LCH_INTERLEAVE_DISABLE bit in any circumstances. A nonsynchronized transfer always releases the physical channel between transfers even if it is linked to another LCh using the logical channel linking feature.

3.1.8 Linking Logical Channels

LCh Types Supporting This Feature	2D	P	PD	G	D
	√	√	√	√	

Software can configure DMA logical channels to an LCh chain.

To configure and start a linked LCh chain:

1) To link the LChs, configure the NEXTLCH_ID bit in the logical channel link control register DMA_CLNK_CTRL to indicate the next logical channel to be enabled as soon as the current logical channel has finished the transfer.

2) To define the LCh as being part of a linked queue, set the ENABLE_LNK bit in the same register. In this way, the logical channel, defined by NEXTLCH_ID, is enabled after the current channel finishes transferring.

3) Start the chain by enabling the first LCh in the chain (set the ENABLE bit of the DMA_CCR register to 1).

In order to stop a linked chain, software can write to the STOP_LNK bit in logical channel link control register to disable chain and queue. This must be done to each LCh that is part of the chain. Writing to the STOP_LNK bit resets both the ENABLE_LNK bit and the LCh enable bit in the DMA_CCR register. If a linked chain is run to finish without being stopped, the whole chain remains linked; that is, all LChs still have the ENABLE_LNK bit set to 1. To start the same linked chain again, just enable the first LCh in the chain.

An LCh can be linked to all LCH 0 to 15 but not to/from the LCH-D. An LCh can link to itself, and two LChs can link to each other. Linking several LCh in a circular fashion is also possible.

Note:

When a linked LCh is stopped with STOP_LNK, the bit ENABLE_LNK is disabled and hence the LCh is not part of the chain any more.

The restrictions on linking logical channels are:

- Constraint 1**:**

  It is the software's responsibility to make sure that all NEXTLCH_ID fields and ENABLE_LNK bits are configured before the logical channel chain is started (first LCh is enabled).

  Undefined effects occur if the software modifies these fields when the chain already is enabled.

- Constraint 2**:**

  It is the software's responsibility to make sure that only the head of chained logical channel is enabled. Undefined effects occur if the software enables a logical channel that is inside a chain.

- Constraint 3:

  It is the software's responsibility to ensure the chained logical channels have the same priority level; otherwise, undefined effects result.

- Constraint 4**:**

  It is the software's responsibility to make sure that the channel context does not change on the fly for a chained logic channel. The channel context must be set up by the software before the head of the chained logic channel is enabled. It can only be updated when the linked chain is disabled. Otherwise*,* undefined effects occur. An exception to this constraint is the STOP_LNK bit.

- Constraint 5:

  If the chain is configured as a looping chain (for example, the LCh is linked to it self or two LChs are linked to each other), it is the software's responsibility to disable/stop it.

- Constraint 6:

  If the bit STOP_LNK is set while the logical channel is running, then the logical channel is deactivated, the transfer is stopped, and no further linked logical channel is activated.

- Constraint 7:

  It is the software's responsibility to set the bit ENABLE_LNK to 0 (or use STOP_LNK) for all the LCh of a chain, if the LCh is to be reused without the chaining capability. Otherwise, activating any of these LCh activates the chain starting from this logical channel.

  In other words, the ENABLE_LNK bit is not reset by hardware at the end of a linked logical channel. A linked chain remains linked after the chain has been executed (if STOP_LNK is not used).

- Constraint 8:

  If a synchronized LCh is part of the chain, it is the responsibility of the software to synchronize the LCh chain with the peripheral so DMA requests are not issued before the LCh is enabled.

  A DMA request is ignored if it is received before the synchronized LCh is enabled by the chain.

3.1.9 Logical Channel Preempting

LCh Types Supporting this Feature	2D	P	PD	G	D
	√	√		√	N/A

A logical channel can be preempted on the element boundary, so that the current element and any ongoing bursts are fully transferred before the channel gets preempted.

Preemption occurs when a high-priority LCh suspends a low-priority LCh. This happens if no other PChs are free when the high-priority LCh gets active. For more details, see section 3.1.6, *Logical Channel Scheduling.*

A synchronized channel goes into an interleaved state once a DMA request is served, and it waits for a new DMA request. It goes into a preempted state once a higher priority logical channel is activated on the same physical channel.

A nonsynchronized channel goes into a preempted state once a higher logical channel is activated on the same physical channel.

A suspended (preempted) low-priority LCh always has a higher priority than a lower priority LCh to be scheduled later. In other words, the preempted LCh does not have to be rearbitrated. However, a suspended LCh takes over the next free PCh. This means that the same PCh is not necessarily to be used if the preempted LCh is of an LCh type supported by several PChs.

An active, preempted, synchronized LCh (the preempted LCh that received a DMA request and was active before the preemption) does not need a new DMA request. The LCh continues the transfer as soon as a PCh becomes free.

If new DMA requests are received when a synchronized LCh is preempted, an event drop is issued on the second new DMA request and the LCH is disabled. If the DROP_IE bit is set to 1 in the DMA_CICR register, then an interrupt is generated and the DROP bit in the DMA_CSR register is set.

The content in the FIFOs in the physical channels is always transferred before the logical channel releases the physical channel. When a physical channel is preempted, it is always on the source element boundary. Therefore, the physical channel drains the FIFO data to the destination before the LCh releases the PCh.

3.1.10 Addressing Modes

An addressing mode is an address computation algorithm that a DMA channel uses to find where to access data. The system DMA supports four types of addressing modes:

- Constant index mode
- Post-incremented mode
- Single-indexed (element index) mode
- Double-indexed (element and frame index) mode

Based on LCh types, the summary of addressing modes is as follows:

Table 20. Logical Channel Type Address Mode Summary

	LCh-P and LCh-PD			
Addressing Mode	**Peripheral Port**	**Memory Port**	**LCh-2D**	**LCh-G**
Constant	√	√	√	√
Post-incremented	√	√	√	√
Single-indexed		√	√	√
Double-indexed		√	√	√

The amount of data (block size) to be transferred is programmed in bytes.

The data block to transfer is split into frames and elements. The data block size in bytes can be expressed as:

$BSi = FN \times EN \times ES$

where:

BSi: The block size in bytes.

FN: The number of frames in the block, $1 \leq FN \leq 65535$ (unsigned) which is defined in the DMA channel frame number register (DMA_CFN).

EN: The number of elements per frame, $1 \leq EN \leq 65535$ (unsigned) which is defined in each DMA channel element number register (DMA_CEN).

ES: The number of bytes per element, $ES \in \{1, 2, 4\}$, which is defined with the field DATA_TYPE in register DMA channel source destination parameters (DMA_CSDP).

A frame size in bytes: $FSi = ES \times EN$.

Setting FN or EN equal to 0 is not allowed, because it causes undefined effects.

An element (data type) can be:

- 8-bit scalar data, s8 (which means that ES = 1)
- 16-bit scalar data, s16 (which means that ES = 2)
- 32-bit scalar data, s32 (which means that ES = 4)

To set up a channel for a transfer, the software must program two addressing modes:

- The source addressing mode, source index size
- The destination addressing mode, destination index size

Both address modes for source and destination are independent. For example, to transfer data from TIPB port to internal memory, the source-addressing mode can be constant and the destination mode can be post-incremented. However, the number of frames, the number of elements, and the element size are the same for both source and destination.

Note:

The channel source and destination start addresses are configured in separate registers: DMA_CSSA_L/U and DMA_CDSA_L/U respectively. Frame index and element index are configured in separate registers for both source and destination: DMA_CSFI and DMA_CSEI, respectively, for source, and DMA_CDFI and DMA_CDEI, respectively, for destination.

Data Alignment

During a transfer, the start address and all the addresses computed by the DMA must be aligned on the type of data transferred (data type):

- If the data type is s8 (8-bit scalar data), then addresses can have any value.
- If the data type is s16 (16-bit scalar data), then addresses must be aligned on 16-bit word boundary (the lowest bit of the address always 0).
- If the data type is s32 (32-bit scalar data), then addresses must be aligned on 32-bit word boundary (the two lowest bit of the address always 00).
- If it is 4x32 burst access with s32 (32-bit scalar data), then addresses must be aligned on burst boundary (the four lowest bit of the address always 0000).
- If frame index is used, it must always produce addresses aligned on data type boundary.
- If element index is used, it must always produce addresses aligned on data type boundary.
- Transfer block size must be aligned on the data type boundary.

Failure to follow these rules generates undefined operation due to wrong endianism switching.

In summary, the general constraints are:

Start address (SA), Element_Index (EI), and Frame_Index (FI) must be aligned on data type (ES):

SA mod ES= 0
(FI-1) mod ES= 0
(EI-1) mod ES= 0

Constant Addressing Mode

Address remains constant for each element to be transferred.

$A(n+1) = A(n)$

where:

A(n): Byte address of the element n within the transfer.

Note:

A(0) is always equal to the start address of the transfer. The same goes for all addressing modes.

Post-Incremented Addressing Mode

Address is post-incremented by element size (ES).

$A(n+1) = A(n) + ES$

where:

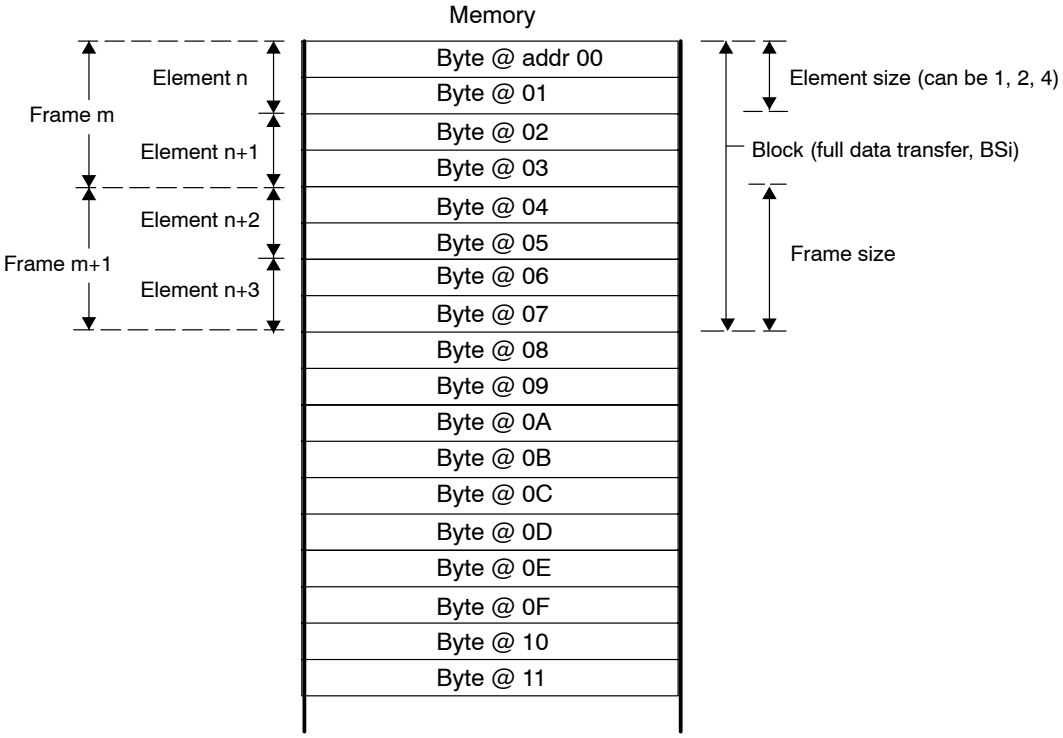
An: Byte address of the element n within the transfer.

ES: Element size in bytes, ES $\in$ {1, 2, 4}.

Figure 6 illustrates how the memory accesses are performed if an LCh is configured with post-increment addressing mode and:

- Starting address: 00
- Element size: 2 (16 bits)
- Element number: 2
- Element index: Ignored
- Frame number: 2
- Frame index: Ignored

Figure 6. Post-Incremented Addressing Mode Memory Accesses



Single-Indexed Addressing Mode

Address is post-incremented by element size and an element index (expressed in bytes).

$A(n+1) = A(n) + ES + (EI - 1)$

$EI = ((Stride\ EI - 1) * ES) + 1$

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 between the beginning of current element, n, to the beginning of next element, n+1.

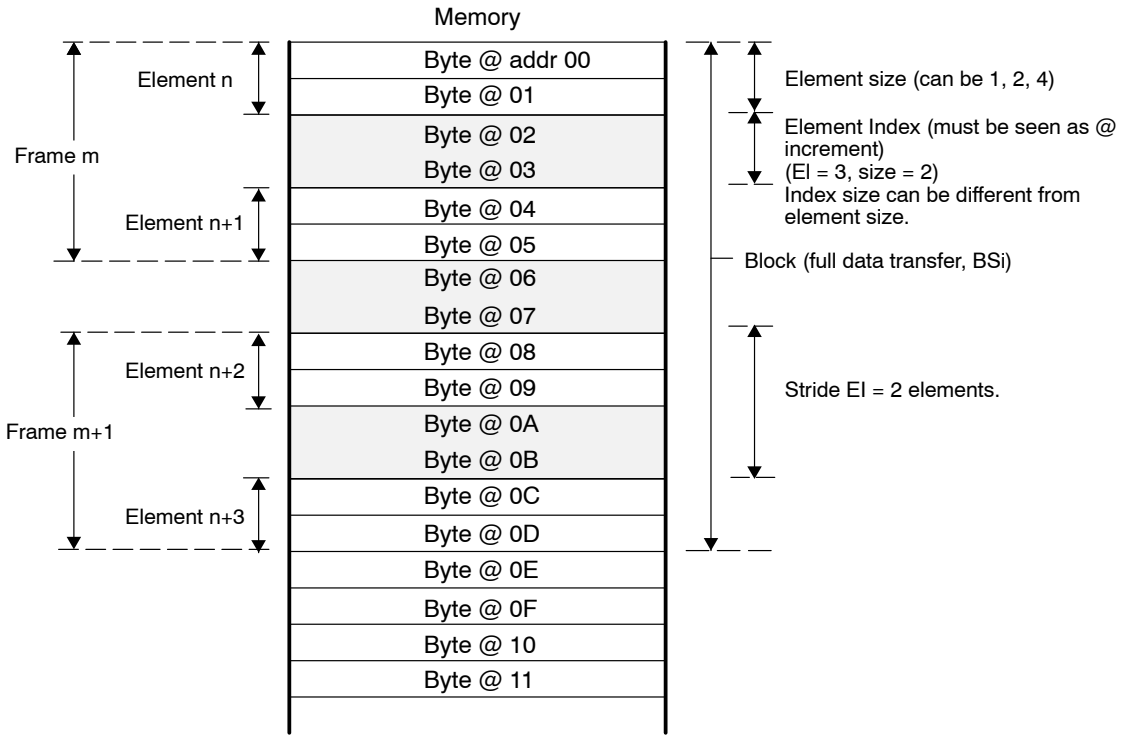
Note:

EI = 1 results in consecutive element accesses (the same behavior as with the post-incremented addressing mode).

Figure 7 illustrates how the memory accesses are performed if a LCh is configured with single indexed addressing mode and:

- Starting address: 00
- Element size: 2 (16 bits)
- Element number: 2
- Element index: 3
- Frame number: 2
- Frame index: Ignored

Figure 7. Single-Indexed Addressing Mode Memory Accesses



Note: EI used between frames for single-indexed addressing mode. This is why this figure has three equal, big strides between the elements.

Double-Indexed Addressing Mode

Address is incremented by element size and a frame index if the end of the current frame is reached. Address is incremented by element size and an element index if the end of the current element is reached but the end of frame is not reached.

When not at end of a frame or transfer, that is, as long as element counter ≠ 0:

$$A(n+1) = A(n) + ES + (EI - 1)$$

When at end of a frame but not at the end of the transfer, that is, as long as element counter = 0 and frame counter ≠ 0:

$$A(n+1) = A(n) + ES + (FI - 1)$$

Calculate element and frame index as follows:

$$EI = ((\text{Stride } EI - 1) * ES) + 1$$
$$FI = ((\text{Stride } FI - 1) * ES) + 1$$

where:

A(n): Byte address of the element n within the transfer.

ES: Element size in bytes, ES ∈ {1, 2, 4}.

EI: Element index in bytes, specified in a configuration register, -32768 ≤ EI ≤ 32767.

Stride EI: Number of elements between the beginning of current element, n, to the beginning of next element, n+1.

Element: Counter that is (re)initiated with the number of element per frame or per transfer. Decreased by 1 for each element transferred. Initial value is configured in register DMA channel element number, DMA_CEN.

FI: Frame index in bytes, specified in a configuration register, - 32768 ≤ FI ≤ 32767.

Stride FI: Number of elements between the beginning of last element of the current element and the beginning of first element of the next frame.

Frame counter: Counter that is (re)initiated with the number of frames per transfer. Decreased by 1 for each frame transferred. Initial value is configured in register DMA channel frame number, DMA_CFN.

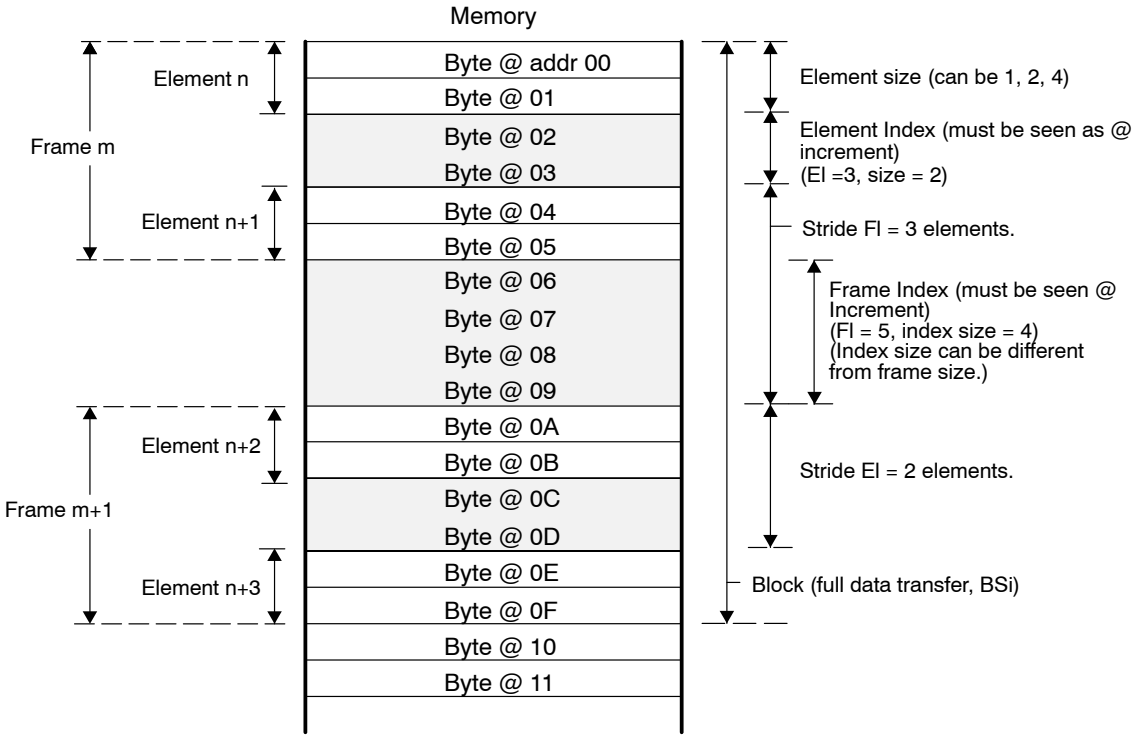
Note:

FI = 1 provides consecutive element accesses at end of frames. If EI = 1, the behavior is the same as with the post incremented addressing mode.

Figure 8 illustrates how the memory accesses are performed if an LCh is configured with double-indexed addressing mode and:

- Starting address: 00
- Element size: 2 (16 bits)
- Element number: 2
- Element index: 3
- Frame number: 2
- Frame index: 5

Figure 8. Double-Indexed Addressing Mode Memory Accesses



Note:

For all addressing modes, independent element and frame indexing were not supported for OMAP 3.0 and 3.1. The EI and FI configuration is dependent on the OMAP3_1_Compatible_Disable bit in the DMA_CCR register.

If DMA_CCR[OMAP3_1_Compatible_Disable]= 0:

- EI(source) = EI(destination) = DMA_CSEI register
- FI(source) = FI(destination) = DMA_CSFI register

If DMA_CCR[OMAP3_1_Compatible_Disable] = 1:

- EI(source) = DMA_CSEI register
- EI(destination) = DMA_CDEI register
- FI(source) = DMA_CSFI register
- FI(destination) = DMA_CDFI register

3.1.11 Data Packing and Bursting

LCh Types Supporting this Feature	2D	P	PD	G	D
	√	√	√	√	√

A DMA channel has the capacity to:

- Pack several consecutive byte accesses in a single word16 or word32 access. This increases the transfer rate. For a channel, the decision to pack to its source port is dependent on the source port access capability and whether source packing is enabled by software. The same goes for the destination port. Software can control packing for both destination and source with the bits SRC_PACK and DST_PACK in the DMA channel source destination parameters register (DMA_CSDP).
- Split a single access, defined by its DMA Port capability, into subset access size if the address of single access is not aligned on access size. For example:
  - Split a word16 transfer into several byte accesses, if access address is not aligned on word16 address.
  - Split a word32 access into single word accesses, if access address is not aligned on word32 boundary.
  - Split 4x32-bit burst access into several byte/word/double-word accesses, if access address is not aligned on 4x32-bit burst access boundary.

- Burst 4x32 bits if bursting is enabled and the accessed source or destination port supports it. Software can enable or disable bursting with the bit-fields SRC_BURST_EN and DST_BURST_EN in the DMA_CSDP register. These bit-fields are ignored if the accessed port does not support bursting and hence, result in single accesses.

Table 21 summarizes the possible transfer configurations and shows the cases where packing and splitting are performed.

Table 21. Packing and Splitting Summary

Data Type	Port Access Capability	Packing / Splitting
s8	s8	-
	s16	pack 2 x s8 => 16
	s32	pack 4 x s8 => 32
s16	s8	split s16 => 2 x 8
	s16	-
	s32	pack 2x s16 => 32
s32	s8	split s32 => 4 x 8
	s16	split s32 => 2 x 16
	s32	-

To compute the type of an access (8/16/32-bit or 4x32-bit burst) and decide to pack consecutive accesses, an address calculation unit checks:

- Its related DMA port capabilities:
  - Can the port perform bytes, 16-bit, or 32-bit access?
  - Can the port perform 4x32-bit burst access?
- What is allowed by the software in the configuration registers:
  - Is packing allowed?
  - Is bursting allowed?
- The last bits of the address:
  - Is the address even or odd?
  - Is the address word16, word32, 4x32-bit burst aligned?
- The number of elements remaining in the frame
- The synchronized transfer type for the LCh

The restrictions for packing and bursting are as follows:

- Constant address mode automatically disables packing and bursting on the DMA source or destination port where it is used. The other end of the DMA channel can still utilize packing and bursting if that DMA port is not configured to use constant address mode.

- All elements must be transferred continuously within a frame. That means that it cannot be synchronized on element boundary, and if single/dual-indexed addressing mode is used, element index (EI) must be set to 1.

- No packing or bursting can be done over frame boundary. It does not help to set frame index FI_ to 1.

- No packing or bursting can be done over block boundary.

Once the type of access is decided, the current byte address can be incremented by 1, 2, 4, or 16 to reach the next memory space location. Then, the DMA port checks its physical channel FIFO to see if there is enough data (write access) or enough space (read access) in the FIFO, before issuing the access.

It is assumed that:

- A 16-bit target memory or peripheral also has a byte-write/read capability.

- A 32-bit target memory or peripheral also has a byte and word16 write/read capability.

If this is not the case, then the software must carefully set up the transfer as follows:

- Follow the address alignment rules in Section 4.12, *Data Alignment.*

- Set the data_type to the same size as the peripheral register width that is the source or the destination of the DMA transfer.

- These rules apply only when the constant address mode is used (which is likely to be the case when a single register in source/destination), and no packing can be enabled.

Packing is controlled by the source and destination packing bits and packs several elements into a larger access element (word16 or word32). For instance, if the data_type is set to s16, the source port supports 32-bit accesses, the source packing is enabled, and the address is aligned on word32, then the source port makes two 16-bit accesses and packs them to a word32 before sending word32 to the destination port. If the source port uses the constant addressing mode and the packing bit is enabled, then the packing bit is ignored.

Another sort of packing groups bytes or word16 accesses within an element. This is always done if the LCh is configured as such. Packing within an element is done if the access types are smaller than the element sizes (data_type). For instance, if the data_type is set to s32 but the source target port only supports 16 bits, then two consecutive word16 source accesses are packed to build the word32 element. This element is then sent to the destination port. This is done independently of the source and destination packing bits.

Packing within an element is supported even if the constant addressing mode is used or if EI is different from 1.

Example 1 shows an example of packing enabled.

Example 1. Packing Enabled

A detailed example (see Table 22 and Table 23) of packing 2 x s16 => 32 is:

- A logical channel is set up for a transfer with the following parameters for its source:
  - Number of frames in the block: FN = 2 elements
  - Number of elements per frame: EN = 5 elements
  - Type of data: s16
  - Frame index in bytes: FI = 13 bytes
  - Element index in bytes: EI = 1 byte
  - Source start address: SA = 2 bytes
  - Source addressing mode: Double-indexed addressing mode
  - The source port is a 32-bit port with byte/word16/word32 access capability.
  - Packing is enabled but burst is disabled.
  - Memory block to transfer is: element i, j (where j is the element number of frame i).
  - i and j start with the highest number and decreases, so the first element written is element 2,5.

Table 22. Channel Data Block to Transfer

Address	Byte 0	Byte 1	Byte 2	Byte 3
0			element 2,5	
4	element 2,4		element 2,3	
8	element 2,2		element 2,1	
12				
16				
20				
24	element 1,5		element 1,4	
28	element 1,3		element 1,2	
32	element 1,1			
36				
40				

The computed addresses and access type are:

Table 23. Channel Addresses and Access Types

Access	Frame Number j	Element Number i	Address	Access Type
0	2	5	2	16 bits
1	2	4	4	32 bits
2	2	2	8	32 bits
3	1	5	24	32 bits
4	1	3	28	32 bits
5	1	1	32	16 bits
6		End of transfer		

In this example, the first word16 is not packed to word32 because the address is not aligned on word32. The next accesses are packed because packing is enabled, constant addressing is not used, EI = 1, and the address is aligned on word32.

Example 2. Packing and Burst Enabled

Example 2 shows an example (see Table 24 and Table 25) with source packing 2 x s16=> 32 plus burst 4x32-bit enabled,. This example results in exactly the same behavior as Example 1, because the burst capability is ignored. It is ignored because when the address is on a 4x32-bit aligned boundary, less than 4x32 bits are left in the frame.

If the frame size is increased by doubling the element size, EN = 10, the example is as follows (assumed that the targeted source support 4x32 bits burst):

A logical channel is set up for a transfer with the following parameters for its source:

- Number of frames in the block: FN = 2 elements.
- Number of elements per frame: EN = 10 elements.
- Type of data: s16.
- Frame index in bytes: FI = 9 bytes.
- Element index in bytes: EI = 1 byte.
- Source start address: SA = 14 bytes.
- Source addressing mode: Double indexed addressing mode.
- The source port is a 32-bit port with byte/word16/word32/4x32bit burst access capability.
- Source packing is enabled.
- Source burst is enabled.
- Memory block to transfer is: element i, j (where j is the element number of frame i).
- Note that i and j starts with the highest number and decreases, so the first element written is element 2,10.

Table 24 shows the channel blocks to transfer.

Table 24. Channel Data Block to Transfer

Address	Byte 0	Byte 1	Byte 2	Byte 3
12				Element 2,10
16		Element 2,9		Element 2,8
20		Element 2,7		Element 2,6
24		Element 2,5		Element 2,4
28		Element 2,3		Element 2,2
32		Element 2,1		
36				
40				Element 1,10
44		Element 1,9		Element 1,8
48		Element 1,7		Element 1,6
52		Element 1,5		Element 1,4
64		Element 1,3		Element 1,2
68		Element 1,1		
72				

Table 25 lists the computed addresses and access types.

Table 25. Channel Addresses and Access Types

Access	Frame Number j	Element Number i	Address [byte]	Access Type
0	2	10	10	16 bits
1	2	9-2	12	4x32 bits
2	2	1	28	16 bits
3	1	10	42	16 bits
4	1	9,8	44	32 bits
5	1	7,6	48	32 bits
6	1	5,4	52	32 bits
7	1	3,2	64	32 bits
8	1	1	68	16 bits
9	End of transfer			

In this example the first word16 is not packed to word32 because the address is not aligned on word32. The next accesses bursts 4x32 bits, because burst is enabled, constant addressing is not used, EI = 1, address is 4x32 bits burst aligned, and enough data is left in the frame (16 bytes). All other transfers are word16 single accesses or 32 bit packed accesses. There are no more bursts because the next time the address is burst aligned, there is not enough data left in the frame (less than 16 bytes).

3.1.12 Interrupt Generation

LCh Types Supporting this Feature	2D	P	PD	G	D
	√	√	√	√	√

Each generic LCh can generate six different interrupts. The following interrupt sources can be programmed:

- End of block: The last byte of the transfer has been written into destination, enabled with bit BLOCK_IE in register DMA_CICR.
- End of frame: The last byte of the current frame has been written into destination, enabled with bit FRAME_IE in register DMA_CICR.
- Half of frame: The middle byte of current frame has been written into destination, enabled with bit HALF_IE in register DMA_CICR.
- Start of last frame: The first word of the last frame has been written into destination, enabled with bit LAST_IE in register DMA_CICR.
- Request collision: Two new DMA requests occurred before the end of service of previous request, enabled with DROP_IE bit in register DMA_CICR.

- Time-out error: An access error occurred in the transfer to the source or the destination, enabled with bit TOUT_IE in register DMA_CICR. The countdown values are configured at the source and destination ports. No countdown value can be specified in the system DMA.

Note:

Configure the interrupt(s) to be generated for each generic LCh in the channel interrupt control register (DMA_CICR). The LCh-D supports only two kinds of interrupts. See Table 77, *DMA LCD Control (DMA_LCD_CTRL)* for more information.

Each DMA logical channel can generate an interrupt to the MPU to reflect the transfer status. Each system DMA logical channel has a dedicated interrupt line to the MPU. All DMA interrupts are level sensitive interrupts; that is, an interrupt line is held active-low until the MPU reads the associated logical channel status register.

No new interrupts can be generated until the status register is read and thereby cleared. For each logical channel, all the interrupt sources are connected together to generate one interrupt. When an interrupt is issued by a logical channel, its status register, DMA_CSR, is set to record the interrupt cause if its associated DMA_CICR enable bit is set. The MPU interrupt service routine (ISR) can read this channel status register to find the sources of the interrupt. The status bits are automatically cleared after they are read by MPU.

The interrupt enable bits are used to choose the events that trigger the DMA channel to send an interrupt to the processor. There are two classes of events:

- Error event: errors during the transfer. These are time-out and request collision.
- Status event: DMA transfer status during DMA channel transfers. These are start of last frame, half of frame, end of frame, and end of block.

When an error event occurs and the corresponding interrupt enable bit is enabled, the following happens:

- The status register bit is activated.
- An interrupt is generated.
- The logical channel is disabled.
- The physical channel is released.

If there is an error but error interrupt is not enabled, no event is generated, the status register bit is not set, and no interrupt is generated. However, the LCh is disabled.

When a status event occurs and the corresponding interrupt enable bit is enabled, the following happens:

- The status register bit is activated.
- An interrupt is generated.
- No new interrupts can be generated until the status register is read and thereby cleared.

Note:

- One read in the status register clears all the status bits.
- No read in the status register keeps all the status bits and DMA interrupt output stays active low.

Therefore, software must always read the associated status register for each DMA interrupt received; otherwise, interrupts can be missed.

System DMA Interrupt Mapping

System DMA interrupt mapping depends on the compatibility mode used.

If the OMAP3_1_COMPATIBLE_DISABLE bit is set to 1 in the DMA_CCR register, then the system DMA has one interrupt line per logical channel.

If this bit is set to zero, then several channels can share the same interrupt line. See 3.1.16 for more details.

Each logical channel has an associated status register, DMA_CSR, where the source of the interrupt is shown.

3.1.13 DMA IDLE Modes

The system DMA can automatically enter an idle mode dynamically as soon as it is not active, or it can be put into idle/suspend mode on request from MPU via the clock generator module.

Dynamic Idle Mode

To save power, the system DMA has a built-in dynamic idle mode that is enabled by setting the CLOCK_AUTOGATING_ON bit in the global control register, DMA_GCR.

The system DMA clock domain is split into several subdomains in this mode; each one of them is disabled if not used. This mode does not add any extra latency in the system DMA.

All internal clocks are in idle mode, disabled, when the following is fulfilled:

1) Clock_Autogating_on = 1 in DMA_GCR.
2) No nonsynchronized LChs are enabled.
3) Either no synchronized LChs are enabled, or synchronized LChs are enabled but no DMA request is received or pending.

The system DMA wakes up if software enables a new LCh or if a DMA request is received. The system DMA also wakes up temporarily if the MPU or OCP-I wants to read or write to any of the registers.

System IDLE Request

The system DMA can be put into IDLE mode, when system needs to go into power saving mode. See *Multimedia Processor OMAP3.2 Subsystem Reference Guide (SPRU749)* and *Multimedia Processor Power Management Reference Guide (SPRU753)* for more information on prerequisite steps to put the system and/or the system DMA in idle. An idle request signal is sent to the DMA when the system wants it to go idle.

As soon as the DMA detects an idle request, it enters idle mode when all PChs are free (including PCh-D). All nonsynchronized LChs, synchronized LChs without DMA request detected, and suspended LChs are unscheduled PChs. The DMA enters idle mode even if one of the LCD controllers is assigned PCh-D but is in sleep mode. The DMA temporarily wakes up if:

- ❏ A synchronized LChs DMA request is detected
- ❏ There are any writes to generic LCh registers
- ❏ Any of the LCD controllers gets active

The DMA goes back to idle mode when all these tasks are completed. This also means as long as the system is requesting the system DMA to be in idle, nonsynchronized LChs can be enabled, but they are never activated. Synchronized LChs can be enabled and hence, activated at DMA request during this time.

The DMA finishes pending accesses of physical channels and resumes the logical channel transfer when released from idle. Therefore, it is the responsibility of software to ensure that when software issues IDLE instruction, all logical channels have been serviced.

The power-saving difference between the dynamic idle mode and the system idle request is provided by the clock tree between the clock generation module and the system DMA clock input.

3.1.14 DMA Debug State

During debug mode the MPU can send a request to suspend the DMA. This is useful if, for instance, the MPU is halted by a breakpoint. How the system DMA responds to this request is controlled by software by configuring the FREE bit in the DMA global control register, DMA_GCR. When the FREE bit is set to 0, all current transfers are suspended when a request is received from the MPU. The transfers resume when the MPU releases the debug request signal. If the FREE bit is set to 1, the DMA continues to run as usual, even if the MPU is sending a debug request signal. The channel status of the DMA interrupts is read on the DMA_CSR register bits. In contrast with the functional mode, the DMA_CSR bits are not cleared after an emulation read.

3.1.15 Other Logical Channel Features

The LCh-G type channel not only transfers data, but also provides hardware with 2-D graphic data processing features to improve 2-D graphic processing speed and graphic quality. It supports the following graphic features:

- ❏ Transparent copy
- ❏ Constant fill (or constant solid color fill)
- ❏ Rotation

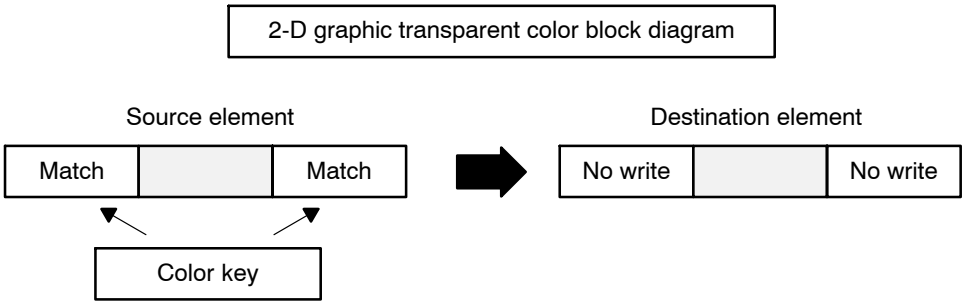
LCh-2D also supports constant fill and rotation. Rotation is supported by all LCh types supporting separate element and frame indexed addressing (2 dimension addressing).

Transparent Copy

LCh Types Supporting this Feature	2D	P	PD	G	D
				√	

It is often desirable to transfer irregular shapes, especially in software for games. The system DMA supports the COLOR KEY feature for 8 BPP, 16 BPP, and 32 BPP from source to destination; that is, each element of channel source is compared to a color key and those data bits (pixels) that match the color key are not written to the destination.

Figure 9. 2-D Transparent Color Block Diagram



This feature is enabled by setting the TRANSPARENT_COPY_ENABLE bit in the channel control 2 register, DMA_CCR2. If this bit is enabled, the DMA color parameter registers, DMA_COLOR_L and DMA_COLOR_U, are used to specify the color key.

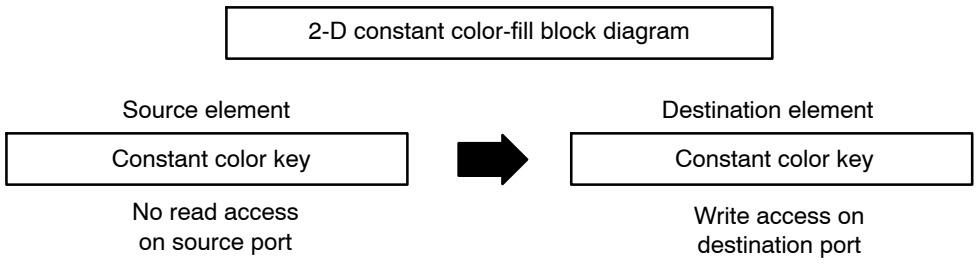
If the data type, ES, is 8 or 16 bits, only the DMA_COLOR_L register is used; if the data type is 32 bits, the DMA_COLOR_U register is also used. The system DMA always compares all bits (8, 16, or 32 bits). If other color depths (other than 8, 16, or 32) are used, it is the responsibility of the software to ensure that unused upper bits are constant. See the register description for more detailed information.

Constant Fill

LCh Types Supporting this Feature	2D	P	PD	G	D
	√			√	

This feature allows filling a region with a constant (solid color) or a pattern by repeating the data horizontally and/or vertically in the region. The system DMA writes to the destination a constant color/value defined in a register. There is no read access on any system DMA source port.

Figure 10. 2-D Constant Color Fill Block Diagram



This feature is enabled by setting the bit Constant_Fill_Enable in the channel control register 2, DMA_CCR2. If this bit is enabled, the register DMA color parameter, DMA_COLOR_L and DMA_COLOR_U, is used to specify the color key.

If the data type, ES, is 8 or 16 bits, only the DMA_COLOR_L register is used; if the data type is 32 bits, register DMA_COLOR_U is also used. See the register description for more detailed information.

Rotation

LCh Types Supporting this Feature	2D	P	PD	G	D
	√	√	√	√	√

All LCh types with independent source and destination indexing support four types of rotation: 0°, 90°, 180°, and 270°. This can be established with the help of double indexed addressing; that is, separate element/frame indexing. See section 3.1.10, *Addressing Modes*, for details of this addressing capability.

3.1.16 Compatibility with OMAP 3.0 and 3.1

After reset, the DMA and all LChs default to the OMAP 3.0/3.1 programming model. This is characterized by a reduced feature set, no LCH chaining, fewer LChs, program and active register sets per LCh, and a restricted number of interrupts (see Table 26).

New code for the DMA must use the programming model presented in previous sections of this document. Future versions of the DMA will be optimized for this programming model and new features will be added to it rather than the 3.0/1 compatible mode.

The DMA described here does not exhibit the same cycle-by-cycle behavior as previous implementations in OMAP 3.0/3.1 and hence time critical applications must be reverified when upgrading.

Two orthogonal mechanisms control the compatibility modes of the system DMA:

- ❏ A global register bit controls the DMA register and interrupt mapping.
- ❏ A register bit per logical channel controls the disabling/enabling of the OMAP 3.0/1 programmers model.

Note:

OMAP 3.2 is configured from reset to be OMAP 3.1 binary compatible.

When in OMAP 3.1, compatible mode PCh-0 and 1 are dynamically shared to serve the generic LChs.

To enable the new features introduced beyond OMAP 3.1, the following configuration register bits must be configured:

- ❑ OMAP 3_1_MAPPING_DISABLE in register DMA_GSCR
- ❑ OMAP3_1_COMPATIBLE_DISABLE in register DMA_CCR/DMA_LCD_CCR
- ❑ LCh register bit that enables/disables the OMAP 3.2 programming model.

Table 26. Summary Table of Different Compatibility Modes

Global Register Bit OMAP3_1_Mapping_Disable	Global Effects	LCh Register Bit OMAP3_1compatible_disable	Per-LCH Effects
0	❑ Reset values ❑ 9 LCh + 1 LCh-D available ❑ OMAP 3.1 configuration register mapping ❑ OMAP 3.1 interrupt line mapping ❑ 7 interrupt lines	0	Only OMAP 3.1 programming model/feature set is supported. PCh-2 cannot be used, so only the LCD channel and two generic logical physical channels are available to these LChs.
		1	All features are available as described in the remainder of this document. All four physical channels are available to these LChs.
1	❑ 16LCh + 1 LCh-D available ❑ OMAP3.2 configuration register mapping ❑ OMAP3.2 interrupt line mapping ❑ 17 interrupt lines	0	Only OMAP 3.1 .programming model/feature set is supported. PCh-2 cannot be used, so only the LCD channel and two generic physical channels are available to these LChs.
		1	All features are available as earlier described in this document. All four physical channels are available to these LChs.

Note: The LCh compatibility bit, OMAP3_1_compatible_disable, is configured per logical channel, so it is possible to mix the programmer model for different logical channels.

The OMAP 3_1_MAPPING_DISABLE bit in register DMA_GSCR is a global register bit that enables/disables OMAP 3.1 register mapping and interrupt line mapping. DMA_GSCR is a global register and must therefore be configured before any LCh is enabled. The DMA_CCR/DMA_LCD_CCR registers are configured per LCh, which enables having different programming models for the LChs. The register and interrupt line mapping must be configured first and affect all the LChs, but the OMAP 3.0/3.1 and OMAP 3.2 programming models, respectively, can be mixed among the LChs.

Autoinitialization of Logical Channels

In OMAP 3.0/3.1 compatible mode, autoinitialization is supported.

If a logical channel is autoinitialized, then the logical channel is automatically enabled itself again when a transfer is ended. A new or old logical channel configuration register set is loaded, and a new block of data is transferred.

Autoinitialization means that the logical channel gets reenabled. It does not mean that the physical channel is kept for the logical channel. The logical channel must return to the same scheduling schemes as normal before it gets access to a physical channel.

The autoinitialization bit is used to set the DMA into autoinitialization mode while END_PROG and REPEAT bits are used to control DMA behavior while in autoinitialization mode. The bits AUTO_INIT, END_PROG, and REPEAT are all located in the channel control register, DMA_CCR. Table 27 summarizes the autoinitialization mode.

Table 27. Autoinitialization Configuration Bits Summary

Auto_init	Repeat	End_prog	Autoinitialization Behavior
0	Don't care	Don't care	No autoinitialization. Waits until software sets enable = 1 to enable the LCh and loads the PCh with its programming register set.
1	0	0	As both repeat and end_prog bits equal 0, the LCh is still enabled, but pending. At the end of the current transfer, the LCh channel waits until repeat or end_prog = 1 to re-activate itself again. When end_prog = 1, the programming register set is copied to the active register set: a new context is programmed.
1	0	1	At the end of the current transfer, the LCh immediately loads the PCh with its programming register set when PCh is granted (end_prog = 1 allows loading the new LCh context, disregarding the repeat bit). The channel reinitializes itself and starts a new transfer with the new context.
1	1	Don't care	The LCh reinitializes itself at the end of the current transfer and starts a new transfer with the previous channel context (active register set).

This table differs slightly from what is implemented in OMAP 3.0/3.1 hardware, where the programming register set is always loaded when END_PROG or REPEAT is set to 1.

OMAP3.0/3.1 System DMA Interrupt Mapping Rule

The OMAP 3.1 system DMA has nine channels plus one LCD channel, which share seven interrupt lines; the OMAP 3.2 system DMA has one interrupt line per channel. In order to be backward compatible, the below interrupt mapping modes have been implemented.

To control the mappings:

OMAP 3.2 Mapping: DMA_CCR.OMAP3_1_Mapping_Disable = 1

OMAP 3.1 Mapping: DMA_CCR.OMAP3_1_Mapping_Disable = 0

See the *Multimedia Processor Interrupts Reference Guide (SPRU757)* for more details about interrupts.

Table 28. Interrupt Mapping per LCh for Both Compatible Modes

Interrupt Line	OMAP 3.2 Mapping	OMAP 3.1 Mapping
MPU level 1 IRQ 19	LCH_0	LCH_0 and LCH_6
MPU level 1 IRQ 20	LCH_1	LCH_1 and LCH_7
MPU level 1 IRQ 21	LCH_2	LCH_2 and LCH_8
MPU level 1 IRQ 22	LCH_3	LCH_3
MPU level 1 IRQ 23	LCH_4	LCH_4
MPU level 1 IRQ 24	LCH_5	LCH_5
MPU level 2 IRQ 53	LCH_6	N/A
MPU level 2 IRQ 54	LCH_7	N/A
MPU level 2 IRQ 55	LCH_8	N/A
MPU level 2 IRQ 56	LCH_9	N/A
MPU level 2 IRQ 57	LCH_10	N/A
MPU level 2 IRQ 58	LCH_11	N/A
MPU level 2 IRQ 59	LCH_12	N/A
MPU level 2 IRQ 60	LCH_13	N/A
MPU level 2 IRQ 61	LCH_14	N/A
MPU level 2 IRQ 62	LCH_15	N/A
MPU level 1 IRQ 25	LCH_D	LCH_12

In case of simultaneous events in two physical channels that share the same interrupt line, only one interrupt is generated and all the relevant status bits are set.

Each physical channel has a seven-bit status register. When an interrupt is shared by two logical channels, the MPU can read the status from the two channels in one TIPB access. The data read has the format shown in Figure 11.

Figure 11. DMA Packed Channel Status Register for Compatible Mode

	[13:7]	[6:0]
00	DMA_CSR [LCH_6] [6:0]	DMA_CSR [LCH_0] [6:0]

3.2 LCD Channel

3.2.1 Display Logical Channel

The display logical channel, LCh-D, transfers data to the LCD controller from a video block buffer stored in memory. In the OMAP 3.2 system, the memory source for the transfer can be Test RAM (OCP-T1), or EMIFF. These transfers can be arranged to have the source in one or two blocks.

Table 29 gives a summary of features for the LCD LCh type.

Table 29. Features Summary for LCh-D

Supported Transfer Features		LCh-D (D)
LCh Control	Transfer type	
	❑ Memory to external LCD controller	√
	❑ Memory to OMAP LCD controller	√
	❑ Nonsynchronized	√
	❑ Synchronized	See Note 1
	Two levels of priority	
	Preemption at element boundary	
	LCh interleaving on DMA request	
	Linking logical channel capability	
	Automatic initialization	√
Addressing	Types of addressing modes for src and dst	
	❑ Constant mode	
	❑ Post increment mode	See Note 2
	❑ Single-indexed mode, with separate SRC/DST index	See Note 2
	❑ Double-indexed mode, with separate SRC/DST index	See Note 2
	❑ Supports single and dual block modes	See Note 2
	❑ Supports separate element/frame index and element/frame numbers for dual block mode	See Note 2

Notes: 1) LCh-D can be synchronized with the external LCD controller but not with the OMAP internal LCD controller.

2) The LCh-D destination port is the LCD port. The access address to the display is controlled by the LCD controller. Hence, these features only apply on the source.

The dual-block mode allows concurrent transfer and image processing (reloading of one block while a second is being processed). Switching from one block to another is achieved by loading the configuration register of the second block buffer and then starting the second block transfer after the first block buffer has been fully transferred. This is automatic, enabling the dual-block feature by setting the BLOCK_MODE bit in the DMA LCD control register (DMA_LCD CTRL).

Separate element/frame index and number of elements between the two buffers are also supported in the dual-block mode.

The LCD channel sends the read request to the relevant port, defined by the lcd_source_port bit in the DMA LCD control register (DMA_LCD CTRL) just as the destination port is selected by the lcd_destination_port in the same register. The destination port can be either the OMAP internal LCD controller or the external LCD controller port.

The dedicated DMA channel contains a FIFO used as an elastic buffer to prevent underflow or overrun to/of the LCD.

Hardware ignores the bottom address register of blocks for dual- and single-block access mode. Each block size is defined by

Block_Size = ES x EN x FN in bytes

where:

ES = Element Size: number of bytes within an element. Bit-field data_type is configured in register DMA LCD channel source destination parameters (DMA_LCD_CSDP).

EN = Element number (within one frame).

FN = Frame number.

In order to support graphic functionality, this DMA LCD channel also supports rotation capability at 0°, 90°, 180°, and 270°, by utilizing its source double index mode.

3.2.2 LCD Channel Addressing Modes

Post-incremented, single-indexed (element index), and double indexed (element and frame indexes) addressing modes are supported from the source port side. However, note that there is no write address to compute in the destination for the display channel. The read FIFO address is controlled by the LCD controller, so there is no write address to a port.

Source Address and Block Size Alignment

On the destination side, the accesses are fixed to 32 bits for external LCD controller and 16 bits for OMAP LCD controller. There is no address alignment because the LCD controller does not use addresses to access data from the LCD channel.

Source memory accesses can be 32/16/8-bit or 4x32-bit burst accesses. The LCD channel is compliant with the generic logical channel constraint; that is, address must be aligned on data_type.

Data block size must be a multiple of the 32-bit for external LCD controller, and a multiple of the 16-bit for OMAP LCD controller.

However, the data_type can still be 32/16/8-bit for both the OMAP and external LCD controllers.

Source Address Modes

The four different LCD channel source address algorithms use the following notations. The address mode descriptions are as follows:

A(n): Byte address of the element n within the transfer.

ES_B1: Element size in block 1 (in bytes): ES $\in$ {1, 2, 4}.

ES_B2: Element size in block 2 (in bytes): ES $\in$ {1, 2, 4}.

BS_B1: Block size of block 1 (in bytes).

BS_B1 = BB1 – TB1 = ES_B1 x EN_B1 x FN_B1

BS_B2: Block size of block 2 (in bytes).

BS_B2 = BB2 – TB2= ES_B2 x EN_B2 x FN_B2

BB1: Bottom address of block 1.

BB2: Bottom address of block 2.

TB1: Top address of block 1.

TB2: Top address of block 2.

DBM: Dual-block mode.

EN_B1: Number of elements within a frame of block 1.

EN_B2: Number of elements within a frame of block 2.

Element_counter_B1: Counter that is (re)initiated with the number of elements per frame or per transfer inside block 1. Decreased by one at each element transferred. Initial value EN_B1 is configured in register DMA channel element number, DMA_CEN.

Element_counter_B2: Counter that is (re)initiated with the number of elements per frame inside block 2. Decreased by one at each element transferred. Initial value EN_B2 is configured in register DMA channel element number, DMA_CEN.

Frame_counter_B1: Counter that is (re)initiated with the number of frames inside block 1. Decreased by one at each frame transferred. Initial value FN_B1 is configured in register DMA channel frame number, DMA_CFN.

Frame_counter_B2: Counter that is (re)initiated with the number of frames inside block 2. Decreased by one at each frame transferred. Initial value FN_B2 is configured in register DMA channel frame number, DMA_CFN.

EI_B1: Block 1 element index in bytes $-32768 \leq EI\_B1 \leq 32767$.

EI_B2: Block 2 element index in bytes $-32768 \leq EI\_B1 \leq 32767$.

Stride_EI_B1: Number of elements between the beginning of the current element (n) and the beginning of the next one (n+1) in block 1.

Stride_EI_B2: Number of elements between the beginning of the current element (n) and the beginning of the next one (n+1) in block 2.

Stride_FI_B1: Number of elements between the beginning of the last element of the current frame and the beginning of the first element of the next frame in block 1.

Stride_FI_B2: Number of elements between the beginning of the last element of the current frame and the beginning of the first element of the next frame in block 2.

FI_B1: block 1 frame index in bytes:

$-2147483648 < FI\_B1 < 2147483647$.

FI_B2: block 2 frame index in bytes:

$-2147483648 < FI\_B2 < 2147483647$.

Post-Incremented Addressing Mode

Address is post-incremented by element size (ES_B1 or ES_B2 depending on which block is active if in dual-block mode).

DBM = 0 (one block mode)

Only block 1 is active:

A(0) = TB1.
A(n+1) = A(n) + ES_B1 until the end of block 1.
Then, when A(n) reaches BB1, A(n+1) = TB1 again

DBM = 1 (dual block mode)

When block 1 is active:

A(0) = TB1.
A(n+1) = A(n) + ES_B1 until the end of block 1.
Then, when A(n) reaches BB1, A(n+1) = TB2: block 2 becomes active.

When block 2 is active:

A(n) = TB2.
A(n+1) = A(n) + ES_B2 until the end of block 2.
Then, when A(n) reaches BB2, A(n+1) = TB1: block 1 becomes active again.

Single-Indexed Addressing Mode

Address is incremented by element size and element index. All is expressed in bytes.

DBM = 0 (one block mode)

Only block 1 is active:

A(0) = TB1.
A(n+1) = A(n) + ES_B1 + (EI_B1 – 1)
where EI_B1 = ((Stride_EI_B1 – 1) * ES_B1) + 1

A(n) is incremented in this way until the end of block 1. Then, when A(n) reaches BB1, A(n+1) = TB1 again.

Note:

Stride_EI_B1 = 1 (equivalent to EI_B1 = 1) gives consecutive element accesses, hence the same behavior as with the post-incremented addressing mode.

DBM = 1 (dual block mode)

When block 1 is active:

A(0) = TB1.
A(n+1) = A(n) + ES_B1 + (EI_B1 – 1)
where EI_B1 = ((Stride_EI_B1 – 1) * ES_B1) + 1

A(n) is incremented in this way until the end of block 1. Then, when A(n) reaches BB1, A(n+1) = TB2: block 2 becomes active.

When block 2 is active:

A(n) = TB2.
A(n+1) = A(n) + ES_B2 + (EI_B2 – 1)
where EI_B2 = ((Stride_EI_B2 – 1) * ES_B2) + 1

A(n) is incremented in this way until the end of block 2. Then, when A(n) reaches BB2, A(n+1) = TB1: block 1 becomes active again.

Note:

Stride_EI_B1/2 = 1 (equivalent to EI_B1/2 = 1) give consecutive element accesses, hence the same behavior as with the Post_Incremented addressing mode.

In dual-block mode, block 2 can be active first.

Double-Indexed Addressing Mode

Address is incremented by element size and element index if the end of the current frame is not reached.

Address is incremented by element size and frame index if the end of the current frame is reached.

All is expressed in bytes.

DBM = 0 (one block mode)

Only block 1 is active:

A(0) = TB1.
A(n+1) = A(n) + ES_B1 + (EI_B1 − 1)
where EI_B1 = ((Stride_EI_B1 − 1) * ES_B1) + 1

A(n) is incremented in this way until the end of the current frame, that is: as long as Element_counter_B1 ≠ 0.

When end of frame (but not end of block 1) is reached, that is:
Element_counter_B1 = 0 and Frame_counter_B1 ≠ 0:

A(n+1) = A(n) + ES_B1 + (FI_B1 − 1)
where FI_B1 = ((Stride_FI_B1 − 1) * ES_B1) + 1

When A(n) reaches BB1, A(n+1) = TB1 again

DBM = 1 (dual block mode)

When block 1 is active:

A(0) = TB1.
A(n+1) = A(n) + ES_B1 + (EI_B1 − 1)
where EI_B1 = ((Stride_EI_B1 − 1) * ES_B1) + 1

A(n) is incremented in this way until the end of the current frame, that is: as long as Element_counter_B1 ≠ 0.

When end of frame (but not end of block 1) is reached, that is:
Element_counter_B1 = 0 and Frame_counter_B1 ≠ 0:

A(n+1) = A(n) + ES_B1 + (FI_B1 − 1)
where FI_B1 = ((Stride_FI_B1 − 1) * ES_B1) + 1

When A(n) reaches BB1, A(n+1) = TB2: block 2 becomes active.

When block 2 is active:

A(0) = TB2A.
A(n+1) = A(n) + ES_B2 + (EI_B2 − 1)
where EI_B2 = ((Stride_EI_B2 − 1) * ES_B2) + 1

A(n) is incremented in this way until the end of the current frame, that is: as long as Element_counter_B2 ≠ 0.

When end of frame (but not end of block 2) is reached, that is: Element_counter_B2 = 0 and Frame_counter_B2 ≠ 0:

A(n+1) = A(n) + ES_B2 + (FI_B2 − 1)
where FI_B2 = ((Stride_FI_B2 − 1) * ES_B2) + 1

When A(n) reaches BB2, A(n+1) = TB1: block 1 becomes active again.

Note:

Both Stride_EI_B1/2= 1 (equivalent to EI_B1/2 = 1) and Stride_FI_B1/2 = 1 (equivalent to EI_B1/2 = 1) give consecutive element accesses, hence the same behavior as with the post-incremented addressing mode.

In dual-block mode, block 2 can be active first.

3.2.3 DMA LCD Channel Sharing Feature

The LCD channel in the OMAP 3.2 system DMA supports the LCD controller. The lcd_destination_port bit in the DMA LCD control register (DMA_LCD_CTRL) sets the DMA to use the LCD controller. The OMAP LCD controller supports 16-bit single accesses. See the *Multimedia Processor Display Interface Reference Guide (SPRU764)* for detailed information about the LCD controller.

LCh-D is enabled differently depending on which LCD controller used:

- In OMAP LCD controller mode, the LCD channel is enabled when the OMAP LCD controller is enabled; that is, when bit LCDEN = 1 in the LCD control register (LCDCONTROL). For more information, see the LCD control register description in the *Multimedia Processor Display Interface Reference Guide (SPRU764).*

3.2.4 DMA LCD Channel Rotation

This DMA LCD channel supports four types of rotation: 0°, 90°, 180°, and 270°, by using the double indexed addressing mode in LCD channel source port. This feature allows the use of display screens that are not oriented as they were designed. For example, a 320x240 screen can be used in a 240x320 application.

3.2.5 DMA LCD Channel Autoinitialization Feature

In order to support special requirements for LCD display, the detailed autoinitialization feature for the LCD channel is described below.

- If the DMA LCD channel is connected to the OMAP external LCD controller:

  When software sets the DMA_LCD_CCR.Enable bit to 1, then the logical LCD channel is enabled. It loads LCD channel programming register set to its channel active register set. Then the logical LCD channel (LCh_D) is active, and data starts transferring.

- If the DMA LCD channel is connected to the OMAP LCD controller:

  When software enables the OMAP LCD controller, bit LCDCONTROL. LCDEN is set to 1 and the DMA LCD channel is automatically enabled at the same time. It loads LCD channel programming register set to channel active register set. The logical LCD channel is active, and data starts transferring.

At the end of the transfer, the LCD logical channel is enabled again, when auto-init is set to 1, and it loads LCD physical channel with:

- Channel programming set, if end_prog = 1 and repeat = 1
- Channel active set, if end_prog = 0 and repeat = 1

Table 30 provides a bit summary.

Table 30. Autoinitialization Bits Summary for LCD Channel in Noncompatible Mode

Auto_init	Repeat	end_prog	Autoinitialization Behavior
0	Don't care	Don't care	No autoinitialization. It waits until enable = 1 to enable the LCD logical channel, and loads the physical LCD channel with its programming register set. However, the LCD logical channel is active only when the LCD controller enables it.
1	0	0	As both repeat and end_prog bits equal 0, the channel is still enabled but pending. At the end of the current transfer, *the logical LCD channel waits until repeat or end_prog = 1 to reactivate itself again*. When end_prog = 1 the programming register set is copied to the active register set: *a new context is programmed.*
1	0	1	At the end of the current transfer, the logical LCD channel immediately loads the physical LCD channel with its programming register set, when physical LCD channel is granted (end_prog = 1 allows loading the new context, disregarding the repeat bit). *The channel reinitializes itself and starts a new transfer with the new context*.
1	1	Don't care	The channel reinitializes itself at the end of the current transfer *and starts a new transfer with the previous context* (active register set).

3.2.6 DMA_LCD_Disable/Bus Error Feature

Software can disable the LCD channel on the fly. If the LCD channel is disabled, then transfer stops immediately.

During LCD channel transfer, if an underflow occurs, the DMA LCD channel resets its enable bit and the LCD channel stops immediately.

LCD PCh underflow is possible when the destination is the OMAP LCD controller (reads from external controller are stalled until data is ready).

If underflow occurs, a signal is sent to the OMAP LCD controller, and the FUF bit is set in the OMAP LCD controller status register (LCSR). An interrupt is generated when a LCSR bit is set. See the *Multimedia Processor Display Interface Reference Guide (SPRU764).*

If one hardware request is currently being serviced, and other hardware requests are triggered, then the DMA_LCD controller will stop and signal an event drop interrupt on the second new hardware request (bus error).

3.2.7 LCD Channel Usage Restrictions

Exclusive Blocks

The hardware design does not detect any overlap between two block buffers; that is, the start and stop addresses of each buffer must represent two different physical parts into the memory. In dual-block mode, the top address of the second block must be greater (and not equal to) than the bottom address of the first block.

Both Blocks Must Belong to a Single Source

In case of dual-block mode operation, it is not possible to have one block read from one source and one block read from a second source. *To change from a source to another, the LCD_LCD_EN (enable transfer) signal must not be asserted and all pending LCD interrupts must be processed.*

LCD Registers Can Be Configured During a Transfer

There is a shadow register set for the LCD channel, which allows the software to change the LCD channel context on the fly after the LCD channel enabled. This is not supported in the OMAP 3.0/3.1 system DMA.

Example 3 shows a transfer from a video block, located in SDRAM, to the OMAP LCD controller.

Example 3. LCD Transfer: EMIFF (SDRAM) → LCD, One Block

The size for the LCD display is 6 x 16 pixels with 16 bits per pixels. So the length of the video frame is 6x16x2 (in bytes) + 32 bytes for the palette = 224 bytes. If the video block starts at address 0x0B0000, the bottom address of the video block is 0x0B00DE.

Step 1: Registers are set as:

DMA_LCD_CTRL

BLOCK_MODE = 0 (one block)
BLOCK_IT_IE = 1
BUS_ERROR_IT_IE = 1
LCD_SOURCE_PORT = 0 (SDRAM)

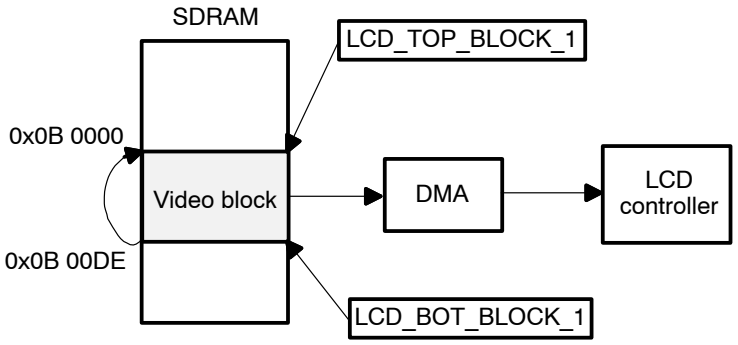
DMA_LCD_TOP_B1_U = 0x000B
DMA_LCD_TOP_B1_L = 0x0000
DMA_LCD_BOT_B1_U = 0x000B
DMA_LCD_BOT_B1_L = 0x00DE

DMA_LCD_TOP_B2_U = irrelevant
DMA_LCD_TOP_B2_L = irrelevant
DMA_LCD_BOT_B2_U = irrelevant
DMA_LCD_BOT_B2_L = irrelevant

Step 2: The transfer starts when the enable (hardware) signal from the OMAP LCD controller is asserted high.

The transfer runs, and an interrupt is generated at the end of the block.

Figure 12. LCD One-Block Mode Transfer Scheme



Step 3: When an interrupt occurs, read the DMA_LCD_CTRL register, to know the source of the interrupt.

If DMA_LCD_CTRL[3] = 1 (that is to say, block_1_it_cond = 1), end of block 1 interrupt is detected.

If end of block is reached, the DMA restarts at the top of the block.

Step 4: Reset DMA_LCD_CTRL [3] and wait for another interrupt.

Example 4. LCD Transfer: OCP_T1 (Test RAM) → LCD, Two Blocks

Example 4 shows a transfer from two video blocks located in memory connected to the OCP_T1 port to the LCD controller.

The size for the LCD display is 6 x 16 pixels with 16 bits per pixels. So the length of one video block is 6 x 16 x 2 (in bytes) + 32 bytes for the palette = 224 bytes. If the video block 1 starts at address 0x0B0000, the bottom address of the video block is 0x0B00DE. If the video block 2 starts at address 0x0C0000, the bottom address of the video block is 0x0C00DE.

Step 1: Registers are set as:

DMA_LCD_CTRL

BLOCK_MODE = 1 (two blocks)
BLOCK_IT_IE = 1
BUS_ERROR_IT_IE = 1
LCD_SOURCE_PORT = 1 (IMIF)

DMA_LCD_TOP_B1_U = 0x000B
DMA_LCD_TOP_B1_L = 0x0000
DMA_LCD_BOT_B1_U = 0x000B
DMA_LCD_BOT_B1_L = 0x00DE

DMA_LCD_TOP_B2_U = 0x000C
DMA_LCD_TOP_B2_L = 0x00000
DMA_LCD_BOT_B2_U = 0x000C
DMA_LCD_BOT_B2_L = 0x00DE

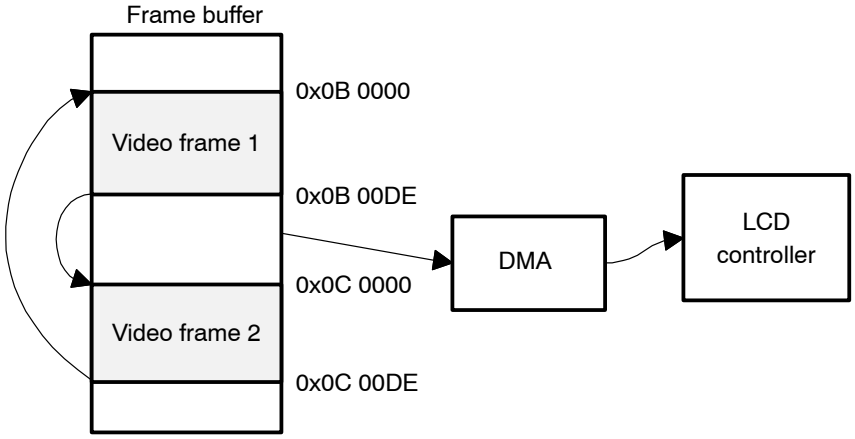
Step 2: The transfer starts when the enable (hardware) signal from the OMAP LCD controller is asserted high.

The transfer runs, and the interrupts are generated at the end of respectively block 1 and 2.

DMA_LCD_CTRL [3] = block_1_it_cond = 1. This bit is a status bit, which detects the interrupt.

When end of block 1 is reached, the DMA restarts at the top address of block 2 and DMA_LCD_CTRL [3] is reset to be able to detect a next interrupt.

Figure 13. LCD Dual-Block Mode Transfer Scheme



DMA_LCD_CTRL [4] = BLOCK_2_IT_COND = 1: end of block 2 is reached. The DMA restarts at the top address of block 1, and DMA_LCD_CTRL [4] is reset to be able to detect a next interrupt.

Switching between the two frames is automatic; it does not need any reconfiguration of the channel.

3.2.8 LCD Channel OMAP 3.0/3.1 Compatible Mode Programming

Users have the option of having new LCD channel features and maintaining compatibility with the previous LCD channel. To keep compatible with the OMAP3.1 programming mode, care must be taken on the following points. Everywhere else, there is no compatibility obstacle.

Configuration Registers

The DMA_LCD_CTRL register is one of the LCD configuration registers. This register manages the operation of dual or single block mode, the interrupt enable bits, and the source port for the next transfer. Then it returns information by setting the status bits in its register. An interrupt can be sent at the end of the transfer of each block; this interrupt line is connected to the LCD interrupt line of the DMA.

Addressing Mode

In OMAP 3.1 compatible mode, the LCD channel only supports the *post-increment* addressing mode on the source side.

Address is post-incremented by element size (ES_B1 or ES_B2 depending on which block is active if in dual-block mode)

$A(n+1) = A(n) + ES\_B1/2 \quad \text{if } TB1/2 \leq A(n+1) \leq BB1/2$

where:

A(n) is the byte address of element n within the transfer.

ES_B1/2 is block1/2 element size.

TB1/2 is top address for block1/2 and BB1/2 is bottom address for block1/2.

Refer to post-increment details in section 3.2.2.

Hardware rotation is not supported in compatible mode due to restrictions on addressing modes.

DMA LCD Channel Sharing Feature

A second external LCD controller is not supported in compatible mode. Only the OMAP3.2 embedded LCD controller is available.

Disabling Feature

The LCD channel cannot be disabled by software. However, it stops transferring as soon as the OMAP LCD controller disables the LCD channel. Then it restarts from the beginning again, after the LCD controller enables the LCD channel.

LCD Channel Restriction

In OMAP 3.1 compatible mode, it is not possible to change any bit of any LCD channel register until a transfer has been fully completed, because no shadow registers exist in LCD channel as in regular channels. To update registers, the LCD_Controller must not be enabled and all pending LCD interrupts must be processed.

The LCD channel source can only have 32-bit access from L3_OCP_T1 (Test RAM) or EMIFF (SDRAM).

Selecting the lcd_source_port bit-field (in the DMA_LCD_CTRL register) to binary 10 or 11 causes undefined effects.

DMA LCD Channel Autoinitialization Feature

At the end of transfer, the LCD logical channel is enabled again if LCD_LCD_EN is set to 1 in the OMAP LCD controller. Table 31 summarizes the autoinitialization behavior in compatible mode.

Table 31. Autoinitialization Configuration Bits Summary for LCD Channel in Compatible Mode

Auto_init	Repeat	End_prog	Autoinitialization Behavior
Hardwired 1	Hardwired 1	Hardwired 1	At the end of the current transfer, the logical LCD channel immediately loads the physical LCD channel with its programming register set, when physical LCD channel is granted (end_prog = 1 allows loading the new context, disregarding the repeat bit). *The channel reinitializes itself and starts a new transfer with the new context.* However, the LCD logical channel is active only when the LCD controller enables it.
0 (Software)	X (Software)	X (Software)	Not supported
1 (Software)	X (Software)	X (Software)	Not supported

DMA Configuration Registers I/O Space

Each channel has its own address space. The configuration address is split into several fields. Each field is decoded to generate a channel select and a register select within the channel.

More configuration registers were added to the LCD channel OMAP 3.2. This results in two different LCD channel register mappings which are dependent on the mode selected, which is controlled by bit OMAP3_1_MAPPING_DISABLE in register DMA_GSCR.

Table 32 shows the register mapping in the two different modes (OMAP 3.0/3.1 compatible or not).

Table 32. LCD Channel Register Mapping for OMAP 3.2 Respectively OMAP 3.0/3.1 Compatible Modes

Offset	Register Mapping in OMAP 3.2 Mapping Mode	Register Mapping in OMAP 3.0/3.1 Mapping Mode
E3C0	DMA_LCD_CSDP	DMA_LCD_CTRL
E3C2	DMA_LCD_CCR	DMA_LCD_TOP_B1_L
E3C4	DMA_LCD_CTRL	DMA_LCD_TOP_B1_U
E3C6	Reserved	DMA_LCD_BOT_B1_L
E3C8	DMA_LCD_TOP_B1_L	DMA_LCD_BOT_B1_U
E3CA	DMA_LCD_TOP_B1_U	DMA_LCD_TOP_B2_L
E3CC	DMA_LCD_BOT_B1_L	DMA_LCD_TOP_B2_U
E3CE	DMA_LCD_BOT_B1_U	DMA_LCD_BOT_B2_L

Table 32. LCD Channel Register Mapping for OMAP 3.2 Respectively OMAP 3.0/3.1 Compatible Modes (Continued)

Offset	Register Mapping in OMAP 3.2 Mapping Mode	Register Mapping in OMAP 3.0/3.1 Mapping Mode
E3D0	DMA_LCD_TOP_B2_L	DMA_LCD_BOT_B2_U
E3D2	DMA_LCD_TOP_B2_U	N/A
E3D4	DMA_LCD_BOT_B2_L	N/A
E3D6	DMA_LCD_BOT_B2_U	N/A
E3D8	DMA_LCD_SRC_EI_B1	N/A
E3DA	DMA_LCD_SRC_FI_B1_L	N/A
E3DC	DMA_LCD_SRC_EI_B2	N/A
E3DE	DMA_LCD_SRC_FI_B2_L	N/A
E3E0	DMA_LCD_SRC_EN_B1	N/A
E3E2	DMA_LCD_SRC_EN_B2	N/A
E3E4	DMA_LCD_SRC_FN_B1	N/A
E3E6	DMA_LCD_SRC_FN_B2	N/A
E3EA	DMA_LCH_CTRL	N/A
E3F4	DMA_LCD_SRC_FI_B1_U	N/A
E3F6	DMA_LCD_SRC_FI_B2_U	N/A

3.3 System DMA Registers

- All reserved bits and all reserved registers must be written to as 0x0000.
- All reserved bits are read as 0.
- The configuration and status registers are part of a superset that is designed for a 16-bit port and a 16-bit channels DMA. This superset enables optimal design reuse in hardware and software, and this design reuse capability requires generic register mapping. These requirements cause some registers to seem at times almost empty.

3.3.1 Summary DMA Global Registers

Table 33 lists the DMA global control registers. Table 34 through Table 50 describe the register bits. You must configure global registers before any LCh is enabled, to avoid undefined effects.

Table 33. DMA Global Control Registers

Base Address = 0xFFFE DC00			
Name	**Description**	**R/W**	**Offset**
DMA_GCR	DMA global control	R/W	0x00
DMA_GSCR	DMA software compatible	R/W	0x04
DMA_GRST	Software reset control	R/W	0x08
DMA_HW_ID	DMA version ID	R/W	0x42
DMA_PCh2_ID	Physical channel 2 version ID	R/W	0x44
DMA_PCh0_ID	Physical channel 0 version ID	R/W	0x46
DMA_PCh1_ID	Physical channel 1 version ID	R/W	0x48
DMA_PChG_ID	Physical channel G version ID	R/W	0x4A
DMA_PChD_ID	Physical channel D version ID	R/W	0x4C
DMA_CAPS_0_U	Global DMA capability 0 upper	R/W	0x4E
DMA_CAPS_0_L	Global DMA capability 0 lower	R/W	0x50
DMA_CAPS_1_U	Global DMA capability 1 upper	R/W	0x52
DMA_CAPS_1_L	Global DMA capability 1 lower	R/W	0x54
DMA_CAPS_2	Global DMA capability 2	R/W	0x56
DMA_CAPS_3	Global DMA capability 3	R/W	0x58
DMA_CAPS_4	Global DMA capability 4	R/W	0x5A
DMA_PCh2_SR	Physical channel 2 status	R/W	0x60
DMA_PCh0_SR	Physical channel 0 status	R/W	0x80
DMA_PCh1_SR	Physical channel 1 status	R/W	0x82
DMA_PChD_SR_0	Physical channel D status	R/W	0xC0

Table 34. DMA Global Control Register (DMA_GCR)

Base Address = 0xFFFE DC00, Offset = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:5	RESERVED	Reserved	R/W	ND
4	ROUND_ROBIN_DISABLE	DMA physical channel scheduler round robin scheduling disable: 0: DMA physical channel scheduler uses round robin scheduling scheme to schedule next available logical channel. 1: DMA physical channel scheduler uses fixed weighted scheduling scheme (from LCH_0 to LCH_i) to schedule next available logical channel.	R/W	0
3	CLK_AUTOGATING_ON	DMA Clock autogating on: 0 = DMA clocks are always on. 1 = DMA cuts off its clocks, according to its activity. Clock_Autogating_on must always be enabled but can be disabled for silicon debug reasons.	R/W	1
2	FREE	DMA reaction to the suspend signal: 0 = DMA suspends all the current transfers when it receives the suspend signal from the processor. Transfers resume when the MPU releases the suspend signal. 1 = DMA continues running when it receives the suspend signal from the processor (For example: when the processor is halted for debug by a breakpoint).	R/W	0
1:0	RESERVED	Reserved	R/W	0

Table 35. DMA Software Compatible Register (DMA_GSCR)

Base Address = 0xFFFE DC00, Offset = 0x04				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:4	RESERVED	Reserved	R/W	ND
3	OMAP3_1_MAPPING DISABLE	OMAP3.1 mapping disable 0 = DMA compatible with OMAP 3.0/3.1 system DMA interrupt line mapping and logical channel configuration register address mapping (system DMA I/O space). 1 = DMA compatible with OMAP_3.2 DMA. Maps interrupt lines and logical channel configuration register address mapping as defined for OMAP 3.2. Compatibility is also dependent on the OMAP3_1_Compatible_Disable bit in register DMA_LCH_CCR.	R/W	0
2:0	RESERVED	Reserved	R/W	ND

Table 36. DMA Software Reset Control Register (DMA_GRST)

Base Address = 0xFFFE DC00, Offset = 0x08				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:1	RESERVED	Reserved	R/W	ND
0	SW_RESET	DMA software reset control: 0: SW will always read this bit as 0. 1: Resets the whole DMA when software writes 1 to this bit (automatically clears to 0).	R/W	0

Table 37. DMA Hardware Version ID Register (DMA_HW_ID)

Base Address = 0xFFFE DC00, Offset = 0x42				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0		DMA version ID number	R	0x0001

Table 38. PCh-2 Version ID Register (DMA_PCh2_ID)

Base Address = 0xFFFE DC00, Offset = 0x44				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	PCH2_ID	DMA PCh-2 version ID number	R	0x0001

Table 39. PCh-0 Version ID Register (DMA_PCh0_ID)

Base Address = 0xFFFE DC00, Offset = 0x46				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	PCH0_ID	DMA PCh-0 version ID number	R	0x0001

Table 40. PCh-1 Version ID Register (DMA_PCh1_ID)

Base Address = 0xFFFE DC00, Offset = 0x48				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	PCH1_ID	DMA PCh-1 version ID number	R	0x0001

Table 41. PCh-G Version ID Register (DMA_PChG_ID)

Base Address = 0xFFFE DC00, Offset = 0x4A				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	PCHG_ID	DMA PCh-G version ID number	R	0x0001

Table 42. PCh-D Version ID Register (DMA_PChD_ID)

Base Address = 0xFFFE DC00, Offset = 0x4C				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	PCHD_ID	DMA PCh-D version ID number	R	0x0001

Table 43. Global DMA Capability U Register 0 (DMA_ CAPS_0_U)

Base Address = 0xFFFE DC00, Offset = 0x4E				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:4	RESERVED	Reserved	R	ND
3	CFC	Constant fill capacity: 0: PCh-G/PCh-0, or -1 cannot do constant fill copy. 1: PCh-G/PCh-0, or -1 can do constant fill copy.	R	1
2	TBLTC	Transparent BLT capability: 0: PCh-G/PCh-0, or -1 cannot do transparent BLT copy. 1: PCh-G/PCh-0, or -1 can do transparent BLT copy.	R	1

Table 43. Global DMA Capability U Register 0 (DMA_ CAPS_0_U) (Continued)

Base Address = 0xFFFE DC00, Offset = 0x4E				
Bit	**Name**	**Function**	**R/W**	**Reset**
1	OD	Overlap detection capability (not available in OMAP3.2): 0: PCh-G cannot do overlap detection. 1: PCh-G can do overlap detection.	R	0
0	DBLTC	Directional BLT capability (not available in OMAP3.2): 0: PCh-G cannot do directional BLT copy. 1: PCh-G can do directional BLT copy.	R	0

Table 44. Global DMA Capability L Register 0 (DMA_ CAPS_0_L)

Base Address = 0xFFFE DC00, Offset = 0x50				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:3	RESERVED	Reserved	R	ND
2		Sub-byte destination capability (not available in OMAP 3.2); 0: PCh-G cannot do bub-byte adjust for expansion. 1: PCh-G can do bub-byte adjust for expansion.	R	0
1	RESERVED	Reserved	R	ND
0		Origin coordinate capability (not available in OMAP 3.2): 0: PCh-G cannot do origin coordinate calculation. 1: PCh-G can do origin coordinate calculation.	R	0

Table 45. Global DMA Capability U Register 1 (DMA_CAPS_1_U)

Base Address = 0xFFFE DC00, Offset = 0x52				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	RESERVED	Reserved	R	ND

Table 46. Global DMA Capability L Register 1 (DMA_CAPS_1_L)

Base Address = 0xFFFE DC00, Offset = 0x54				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:2	RESERVED	Reserved	R	ND
1	CE	1-bit palletized capability (not available in OMAP 3.2): 0: PCh-G cannot do 1-bit color expansion. 1: PCh-G can do 1-bit color expansion.		
0	RESERVED	Reserved	R	ND

Table 47. Global DMA Capability Register 2 (DMA_CAPS_2)

Base Address = 0xFFFE DC00, Offset = 0x56				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:9	RESERVED	Reserved	R	ND
8	SSDIC	Separate source/double-index capability: 0: Does not support separate src/dst index for 2-D addressing. 1: Supports separate src/dst index for 2-D addressing.	R	1
7	DDIAC	Destination double-index address capability: 0: Does not support double-index address mode in destination port. 1: Supports double-index address mode in destination port.	R	1
6	DSIAC	Destination single-index address capability: 0: Does not support single-index address mode in destination port. 1: Supports single-index address mode in destination port.	R	1
5	DPIAC	Destination post-increment address capability: 0: Does not support post-increment address mode in destination port. 1: Supports post-increment address mode in destination port.	R	1
4	DCAC	Destination constant address capability: 0: Does not support constant address mode in destination port. 1: Supports constant address mode in destination port.	R	1
3	SDIAC	Source double-index address capability: 0: Does not support double-index address mode in source port. 1: Supports double-index address mode in source port.	R	1

Table 47. Global DMA Capability Register 2 (DMA_CAPS_2) (Continued)

Base Address = 0xFFFE DC00, Offset = 0x56				
Bit	**Name**	**Function**	**R/W**	**Reset**
2	SSIAC	Source single-index address capability: 0: Does not support single-index address mode in source port. 1: Supports single-index address mode in source port.	R	1
1	SPIAC	Source post-increment address capability: 0: Does not support post-increment address mode in source port. 1: Supports post-increment address mode in source port.	R	1
0	SCAC	Source constant address capability: 0: Does not support constant address mode in source port. 1: Supports constant address mode in source port.	R	1

Table 48. Global DMA Capability Register 3 (DMA_CAPS_3)

Base Address = 0xFFFE DC00, Offset = 0x58				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:6	RESERVED	Reserved	R	ND
5	CCC	Channel chaining capability: 0: Does not support logical channel chaining capability. 1: Supports logical channel chaining capability.	R	1
4	IC	LCh interleave capability: 0: Does not support logical channel interleave capability. 1: Supports logical channel interleave capability.	R	1
3	ARC	Autoinit and repeat capability: 0: Does not support repeat feature in autoinitialization mode. 1: Supports repeat feature in autoinitialization mode (supported only in OMAP 3.0/1 compatible mode).	R	1
2	AEC	Autoinit and End_prog capability: 0: Does not support End_Prog feature in autoinit mode. 1: Supports End_Prog feature in autoinit mode (supported only in OMAP 3.0/1 compatible mode).	R	1

Table 48. Global DMA Capability Register 3 (DMA_CAPS_3) (Continued)

Base Address = 0xFFFE DC00, Offset = 0x58				
Bit	**Name**	**Function**	**R/W**	**Reset**
1	FSC	Frame synchronization capability: 0: Does not support synchronization transfer on frame boundary. 1: Supports synchronization transfer on frame boundary.	R	1
0	ESC	Element synchronization capability: 0: Does not support synchronization transfer on element boundary. 1: Supports synchronization transfer on element boundary.	R	1

Table 49. Global DMA Capability Register 4 (DMA_CAPS_4)

Base Address = 0xFFFE DC00, Offset = 0x5A				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:7	RESERVED	Reserved	R	ND
6	SSC	Sync status capability: 0: Does not support synchronization transfer status bit generation. 1: Supports synchronization transfer status bit generation.	R	1
5	BIC	Block interrupt capability (end of block): 0: Does not support block interrupt generation capability. 1: Supports block interrupt generation capability.	R	1
4	LFIC	Last frame interrupt capability (start of last frame): 0: Does not support last frame interrupt generation capability. 1: Supports last frame interrupt generation capability.	R	1
3	FIC	Frame interrupt capability (end of frame): 0: Does not support frame interrupt generation capability. 1: Supports frame interrupt generation capability.	R	1
2	HFIC	Half frame interrupt capability (half of frame): 0: Does not support half frame interrupt generation capability. 1: Supports half frame interrupt generation capability.	R	1

Table 49. Global DMA Capability Register 4 (DMA_CAPS_4) (Continued)

Base Address = 0xFFFE DC00, Offset = 0x5A				
Bit	**Name**	**Function**	**R/W**	**Reset**
1	EDIC	Event drop interrupt capability (request collision): 0: Does not support event drop interrupt generation capability. 1: Supports event drop interrupt generation capability.	R	1
0	TOIC	Time-out interrupt capability (time-out error): 0: Does not support time-out interrupt generation capability. 1: Supports time-out interrupt generation capability.	R	1

Table 50. Physical Channel-x Status Registers (DMA_PCh2_SR,..., DMA_PHhD_SR_0)

Base Address = 0xFFFE DC00, Offset = 0x60, 0x80. 0x82, 0xC0				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:8	RESERVED	Reserved	R	ND
7:0	ALCN	Active logical channel number for associated physical channel.	R	0xFF

Table 33 specifies which physical channels match each offset. These return the logical channel ID, which is active in the associated physical channel. These four registers can be used to monitor the progress of a DMA physical channel transfer.

Each register is a snapshot of the current logical channel number, which is active on the physical channel:

FF: DMA Physical_Channel_x is IDLE.

Others: DMA Logical_Channel_Number is active on the physical channel.

If PchD is active, DMA_PChD_SR_0 will read as 0x1F if DMA is in OMAP3.2 compatible mode and 0xC in OMAP3.0/3.1 compatible mode.

3.3.2 Logical Channel Registers

This set of registers is instantiated within each logical channel to the DMA. The registers for the display channel, LCh-D are specific for that channel. Hence, these registers are collected in a dedicated section for the LCh-D, see Section 3.3.3, *LCD Channel Dedicated Registers*.

All registers for a specific LCh must be configured before the LCh is enabled; if not, it will cause undefined effects. There are some exceptions to this rule when in OMAP3.0/3.1 compatible mode.

Table 51 lists the logical channel configuration registers. Table 52 through Table 96 describe the register bits.

Table 51. DMA Logical Channel Configuration Registers

Base Address = FFFE D800			
Name	**Description**	**R/W**	**Offset** †
DMA_CSDP	Channel source destination parameters	R/W	0x00 + (n*0x40)
DMA_CCR	Channel control register	R/W	0x02 + (n*0x40)
DMA_CICR	Channel interrupt control register	R/W	0x04 + (n*0x40)
DMA_CSR	Channel status register	R	0x06 + (n*0x40)
DMA_CSSA_L	Channel source start addr, lower bits	R/W	0x08 + (n*0x40)
DMA_CSSA_U	Channel source start addr, upper bits	R/W	0x0A + (n*0x40)
DMA_CDSA_L	Channel destination start addr, lower bits	R/W	0x0C + (n*0x40)
DMA_CDSA_U	Channel destination start addr, upper bits	R/W	0x0E + (n*0x40)
DMA_CEN	Channel element number	R/W	0x10 + (n*0x40)
DMA_CFN	Channel frame number	R/W	0x12 + (n*0x40)
DMA_CSFI	Channel source frame index	R/W	0x14 + (n*0x40)
DMA_CSEI	Channel source element index	R/W	0x16 + (n*0x40)
DMA_CSAC	Channel source addr counter	R	0x18 + (n*0x40)
DMA_CDAC	Channel destination addr counter	R	0x1A + (n*0x40)
DMA_CDEI	Channel destination element index	R/W	0x1C + (n*0x40)
DMA_CDFI	Channel destination frame index	R/W	0x1E + (n*0x40)
DMA_COLOR_L	Color parameter register, lower bits	R/W	0x20 + (n*0x40)
DMA_COLOR_U	Color parameter register, upper bits	R/W	0x22 + (n*0x40)
DMA_CCR2	Channel control register 2	R/W	0x24 + (n*0x40)
DMA_CLNK_CTRL	Channel link control register	R/W	0x28 + (n*0x40)
DMA_LCH_CTRL	Logical channel control register	R/W	0x2A + (n*0x40)

† n is the logical channel numbered 0x0 through 0xF.

Table 52. Channel Source Destination Parameters Register (DMA_CSDP)

Base Address = 0xFFFE D800, Offset Address = 0x00 + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:14	DST_BURST_EN	Destination burst enable	R/W	00
13	DST_PACK	Destination packing	R/W	0
12:9	DST	Transfer destination	R/W	0000
8:7	SRC_BURST_EN	Source burst enable	R/W	00
6	SRC_PACK	Source packing	R/W	0
5:2	SRC	Transfer source	R/W	0000
1:0	DATA_TYPE	Defines the type of the data moved in the channel (see Table 53)	R/W	00

Table 53. Data Types

Data_type	Type
00	s8: 8 bits scalar
01	s16: 16 bits scalar
10	s32: 32 bits scalar
11	Illegal (cause undefined effects)

Start_Address must be aligned on the boundary of the type of data moved. For example, if data_type is s32, the source and destination start address must be aligned on a word32. If data_type is s8, source and destination start_address can have any value. Otherwise, the DMA is forced to transfer a different type of data to the PORT because of the unaligned start address value, which can cause undefined effects.

It is the software's responsibility to make sure that start address is aligned with channel data type.

- src: Transfer source

  A unique identifier is given to each port. This field identifies the port originator of the transfer.

src	OMAP Source Port
0000	EMIFF
0001	EMIFS
0010	OCP_T1
0011	TIPB
0100	OCP_T2
0101	MPUI
Others	Illegal (cause undefined effects)

- SRC_PACK: source packing

  The DMA ports can have a data bus width different from the type of data moved by the DMA channel. For example, we can read s8 data_type on a 32-bit DMA port. The DMA channel has the capacity to pack four consecutive s8 data reads in a single 32-bit read access, to increase transfer bandwidth. The packing is made in respect of endianism.

  SRC PACK = 1: the source port makes packed accesses;
  SRC PACK = 0: the source port never makes packed accesses.

- SRC_BURST_EN: source burst enable

  Used to enable bursting on the source port. When bursting is enabled, the source port performs bursts 4 x src_width access. When bursting is disabled, the source port performs single accesses of SRC_WIDTH access.

SRC_BURST_EN	Burst Type
00	Single access
01	Single access
10	Burst 4x Src_width (Only support 4x32bit)
11	Burst 8x Src_width (Not supported)

If the source port of the channel has no burst access capability, this field is ignored.

- destination fields: (dst, dst_pack, dst_burst_en) contains the same parameters as the source fields, but for the destination port

Table 54. DMA Channel Control Register (DMA_CCR)

Base Address = 0xFFFE D800, Offset Address = 0x02 + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:14	DST_AMODE	Destination addressing mode	R/W	00
13:12	SRC_AMODE	Source addressing mode	R/W	00
11	END_PROG	End of programming status bit	R/W	0
10	OMAP_3_1_COMPATIBLE_DISABLE	OMAP 3.0/3.1 channel compatibility control	R/W	0
9	REPEAT	Repetitive operation	R/W	0
8	AUTO_INIT	Autoinitialization at the end of the transfer	R/W	0
7	ENABLE	Logical channel enable	R/W	0
6	PRIO	Channel priority	R/W	0
5	FS	Frame synchronization	R/W	0
4:0	SYNC	Synchronization control	R/W	00000

- SYNC: Synchronization control

  This field is used to specify the external DMA request, which can trigger the transfer for the LCh. There are 31 possible choices for the system DMA. Each LCh can be triggered by one of the DMA request inputs. A hardware DMA request cannot be shared between several concurrent channels (enabled and active). However, a hardware DMA request can be shared between different channels if they are part of a chain. Therefore, the user must carefully generate DMA requests if a sharing strategy is chosen. Otherwise, event drop of channel transfer may occur when the next DMA request is issued to trigger another channel transfer while the current channel transfer is still going on. For nonsynchronized LCh transfers, this field must be set to binary 00000.

- FS: Frame synchronization

  This bit and the bs bit in the DMA_CCR2 register are used to program the way that a DMA_request is serviced in a synchronized transfer.

  fs = 1 and bs = 0: An entire frame is transferred each time a DMA_request is made. This frame can be interleaved on the DMA ports with other channel requests.

fs = 0 and bs = 1: An entire block is transferred each time a DMA_request is made. This block can be interleaved on the DMA port with other channel requests.

fs = 0 and bs = 0: An element is transferred each time a DMA_request is made. This element can be interleaved on the DMA port with other channel requests.

fs = 1 and bs = 1: This is not allowed and causes undefined results.

- PRIO: Channel priority

  Prio = 1: The logical channel has the highest priority level.

  Prio = 0: The logical channel has the lowest priority level.

- ENABLE: Logical channel enable

  This bit is used to enable/disable the transfer in the DMA channel.

  Enable = 1: The transfer starts.
  Enable = 0: The transfer stops, and it is reset.

  This bit is automatically cleared by the DMA hardware once the transfer is accomplished. Clearing of this bit by the DMA has the priority over write by the processor. If both occur simultaneously, the configuration write is discarded.

- AUTO_INIT: Autoinitialization at the end of the transfer

  Only supported in OMAP 3.0/1 compatible mode. In OMAP 3.2 mode, when bit OMAP3_1_COMPATIBLE_DISABLE = 1, this bit must always be set to 0.

  Auto_init = 1: DMA is in autoinitialization mode. Once the current transfer is complete, the channel automatically reinitializes itself and starts a new transfer if repeat or end_prog bits is set.

  Auto_init = 0: DMA is in non-autoinitialization mode. The DMA channel stops at the end of the current transfer.

  There are two ways to stop a channel while it is in auto-init mode:

  - Write 0 in DMA_CCR.ENABLE bit: the channel immediately stops.
  - Write 0 in DMA_CCR.AUTO-INIT: the channel completes the current transfer and stops.

- REPEAT: Repetitive operation

  Only supported in OMAP 3.0/3.1 compatible mode. In OMAP 3.2 mode, when bit OMAP3_1_Compatible_Disable = 1, this bit must always be set to 0. The functionality of repeat and end_prog are dependent on each other. See Table 27, *Autoinitialization Configuration Bits Summary,* for more details (following descriptions do not cover all possibilities).

Repeat = 1: When DMA channel is in autoinitialization mode and end_prog = 0, once the current transfer is complete, the channel automatically reinitializes itself and starts a new transfer.

Repeat = 0: When DMA channel is in autoinitialization mode and end_prog = 0, once the current transfer is complete, the channel automatically reinitializes itself and starts a new transfer.

Note that even if REPEAT bit is only supported in OMAP3.0/3.1 compatible mode, its behavior with respect to END_PROG bit is slightly different than in OMAP3.1.

- OMAP3_1_COMPATIBLE_DISABLE: OMAP 3.0/3.1 channel compatibility control

  This bit is used to set the DMA logical channel programming model to be used. By default, all DMA logical channels are in omap_3.1 compatible mode.

  omap3_1_compatible_disable = 0: The logical channel is in OMAP_3.0/3.1 compatible mode.

  omap3_1_compatible_disable = 1: The logical channel is in OMAP_3.2 compatible mode.

- END_PROG: End of programming status bit

  Only supported in OMAP 3.0/1 compatible mode. In OMAP 3.2 mode, when bit OMAP3_1_COMPATIBLE_DISABLE = 1, this bit must always be set to 0.

  End_prog = 1: When DMA channel is in autoinitialization mode, it allows the channel to reinitialize itself after the current DMA channel transfer has been finished.

  End_prog = 0: When DMA channel is in autoinitialization mode and REPEAT bit is 0, DMA delays the channel autoinitialization until end_prog bit is set to 1 after the current DMA channel transfer has been finished.

  Note that even if END_PROG bit is only supported in OMAP3.0/3.1 compatible mode, its behavior with respect to REPEAT bit is slightly different than in OMAP3.1.

- SRC_AMODE: Source addressing mode

  This field is used to choose the addressing mode on the source port of a channel.

  src_amode = 00: Constant address
  src_amode = 01: Post-incremented address
  src_amode = 10: Single index (element index)
  src_amode = 11: Double index (element index and frame index)

❑ DST_AMODE: Destination addressing mode

This field is used to choose the addressing mode on the destination port of a channel.

dst_amode = 00: Constant address
dst_amode = 01: Post-incremented address
dst_amode = 10: Single index (element index)
dst_amode = 11: Double index (element index and frame index)

Table 55. DMA Channel Interrupt Control Register(DMA_CICR)

Base Address = 0xFFFE D800, Offset Address = 0x04 + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:6	RESERVED	Reserved	R/W	N/A
5	BLOCK_IE	End block interrupt enable (end of block)	R/W	0
4	LAST_IE	Last frame interrupt enable (start of last frame)	R/W	0
3	FRAME_IE	Frame interrupt enable (end of frame)	R/W	0
2	HALF_IE	Half-frame interrupt enable (half of frame)	R/W	0
1	DROP_IE	Synchronization event drop interrupt enable (request collision)	R/W	1
0	TOUT_IE	Time-out interrupt enable (time-out error)	R/W	1

The interrupt enable bits are used to choose the events that cause the DMA channel send an interrupt to the processor. There are two classes of events:

❑ Error event: Errors during the transfer; for example time-out and event drop.

❑ Status event : DMA transfer status, during DMA channel transfers; for example new frame starts, end of data block to transfer is reached.

Each time an event occurs, if the corresponding interrupt enable bit is set, the channel sends an interrupt to the processor. At the same time, the corresponding status bit is set in DMA_CSR (DMA channel status register). A status bit is not set, if the corresponding interrupt enable bit in the DMA_CICR equals 0.

❑ TOUT_IE: Time-out interrupt enable (time-out error)

tout_ie = 1: The DMA sends an interrupt to the processor if a time-out error occurs either in the source or in the destination port of the channel.

tout_ie = 0: The DMA does not send an interrupt to the processor if a time-out error occurs.

- DROP_IE: Synchronization event drop interrupt enable (request collision)

  drop_ie = 1: The channel sends an interrupt to the processor if the channel transfer is synchronized on DMA requests and two successive DMA requests drop. This occurs when a new DMA request is made while the service of the previous one has not been finished yet.

  drop_ie = 0: The channel does not interrupt the processor when a synchronization event drop occurs.

- HALF_IE: Half-frame interrupt enable (half of frame)

  half_ie = 1: The channel sends an interrupt to the processor when the transfer of the first half of the current frame completes.

  half_ie = 0: The channel does not interrupt the processor when the transfer of the first half of the current frame completes.

- FRAME_IE: Frame interrupt enable (end of frame)

  frame_ie = 1: The channel sends an interrupt to the processor when the transfer of the current frame completes.

  frame_ie = 0: The channel does not interrupt the processor when the transfer of the current frame completes.

- LAST_IE: Last frame interrupt enable (start of last frame)

  last_ie = 1: The channel sends an interrupt to the processor when the transfer of the last frame starts.

  last_ie = 0: The channel does not interrupt the processor when the transfer of the last frame starts.

- BLOCK_IE: End block interrupt enable (end of block)

  block_ie = 1: The channel sends an interrupt to the processor when the transfer of the block completes.

  block_ie = 0: The channel does not interrupt the processor when the transfer of the block completes.

The DMA channel status register is written by the DMA to reflect the channel status. It can be read by the processor by polling or after an interrupt. After a functional read, all the DMA_CSR bits are automatically cleared. However, DMA_CSR bits are not cleared after emulation read.

Table 56. DMA Channel Status Register (DMA_CSR)

Base Address = 0xFFFE D800, Offset Address = 0x06 + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:7	RESERVED	Reserved	R	N/A
6	SYNC	Synchronization status	R	0
5	BLOCK	End block (end of block)	R	0
4	LAST	Last frame (start of last frame)	R	0
3	FRAME	Frame (end of frame)	R	0
2	HALF	Half frame (half of frame)	R	0
1	DROP	Event drop (request collision)	R	0
0	TOUT	Time-out in the channel (time-out error)	R	0

A status bit is not set if the corresponding interrupt enable bit in the DMA_CICR register = 0.

- TOUT: Time-out in the channel (time-out error)

  tout = 1: Time-out occurred in channel.
  tout = 0: No time-out error occurred in channel.

- DROP: Event drop (request collision)

  drop = 1: Synchronization event drop occurred during the transfer.
  drop = 0: No event drop occurred during the transfer.

- HALF: Half frame (half of frame)

  half = 1: First half of the current frame has been transferred.
  half = 0: First half of the current frame has not finished transferring yet.

- FRAME (end of frame)

  frame = 1: A complete frame has been transferred.
  frame = 0: Transfer of the current frame is still in progress.

- LAST: Last frame (start of last frame)

  last = 1: The transfer of the last frame has started.
  last = 0: Last frame has not started yet.

- BLOCK: End block (end of block)

  block = 1: The current transfer in the channel has been finished, but another one may have started if DMA_CCR2[AUTO_INIT] = 1.

  block = 0: Current transfer has not finished yet.

❏ SYNC: Synchronization status

Set to 1 when a DMA request is made in a synchronized channel. When there is a TIPB read access to DMA_CSR register, this bit returns to zero.

Sync = 1: Logical channel is servicing synchronized DMA request.

Sync = 0: Logical channel is not servicing a synchronized channel, or DMA request has not been scheduled.

Table 57. DMA Channel Source Start Address Lower Bits Register (DMA_CSSA_L)

Base Address = 0xFFFE D800, Offset Address = 0x08 + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	SSAL	Source start address, lower bits	R/W	N/A

❏ Source start address, lower bits

Lower bits of the source start address, expressed in bytes. The source start address generated by the DMA is up to a 32-bit address, made of the concatenation of DMA_CSSA_U and DMA_CSSA_L.

Table 58. DMA Channel Source Start Address Upper Bits Register (DMA_CSSA_U)

Base Address = 0xFFFE D800, Offset Address = 0x0A + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	SSAU	Source start address, upper bits	R/W	N/A

❏ Source start address, upper bits

Upper bits of the source start address, expressed in bytes. The source start address generated by the DMA is a byte address, made of concatenation of DMA_CSSA_U and DMA_CSSA_L.

Table 59. DMA Channel Destination Start Address Lower Bits Register (DMA_CDSA_L)

Base Address = 0xFFFE D800, Offset Address = 0x0C + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	DSAL	Destination start address, lower bits	R/W	N/A

❏ Destination start address, lower bits

Lower bits for the destination start address, expressed in bytes. The destination start address is up to a 32-bit address, made of the concatenation of DMA_CDSA_U and DMA_CDSA_L.

Table 60. DMA Channel Destination Start Address Upper Bits Register (DMA_CDSA_U)

Base Address = 0xFFFE D800, Offset Address = 0x0E + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	DSAU	Destination start address, upper bits	R/W	N/A

❑ Destination start address, upper bits

Upper bits for the source start address, expressed in bytes. The destination start address is made of the concatenation of DMA_CDSA_U and DMA_CDSA_L.

Table 61. DMA Channel Element Number Register (DMA_CEN)

Base Address = 0xFFFE D800, Offset Address = 0x10 + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	EN	Channel element number	R/W	N/A

❑ Element number

Number of elements within a frame (unsigned). The maximum element number is 65535.

Table 62. DMA Channel Frame Number Register (DMA_CFN)

Base Address = 0xFFFE D800, Offset Address = 0x12 + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	FN	Channel frame number	R/W	N/A

❑ Frame number

Number of frames within the block to be transferred (unsigned). The maximum frame number is 65535.

The size in bytes of the data block to transfer is data_block_in_bytes = DMA_CFN x DMA_CEN x DMA_CSDP[data_type].

Table 63. DMA Channel Source Frame Index Register (DMA_CSFI)

Base Address = 0xFFFE D800, Offset Address = 0x14 + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	SFI	Channel source frame index	R/W	N/A

- Channel source frame index

  Contains the channel source frame index, expressed as signed value in bytes, which is used to compute addresses when double indexed addressing mode is used in DMA source port.

Table 64. DMA Channel Source Element Index Register (DMA_CSEI)

Base Address = 0xFFFE D800, Offset Address = 0x16 + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	SEI	Channel source element index	R/W	N/A

- Channel source element index

  Contains the channel source element index, expressed as signed value in bytes, which is used to compute the addresses when single index addressing mode is used.

Table 65. DMA Channel Source Element Index Register (DMA_CSAC)

Base Address = 0xFFFE D800, Offset Address = 0x18 + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	SAC	Source element/frame address 16 LSB	R	N/A

This register can be used to monitor the progress of a DMA transfer on channel source port:

- It is a snapshot of the source address generated by the channel source address counter, which is scheduled in the channel source port.
- It is incremented on each access made on the channel source port (S8, S16 or S32).

Note that in OMAP3.0/3.1 the address used by this register was dedicated to DMA_CPC register, which does not exist anymore.

Table 66. DMA Channel Destination Address Counter Register (DMA_CDAC)

Base Address = 0xFFFE D800, Offset Address = 0x1A + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	DAC	Destination element/frame address 16 LSB	R	N/A

This register can be used to monitor the progress of a DMA transfer on channel destination port:

- It is a snapshot of the destination address generated by the channel destination address counter, which is scheduled in the channel destination port.
- It is incremented on each access made on the channel destination port (S8, S16 or S32).

Table 67. DMA Channel Destination Element Index Register (DMA_CDEI)

Base Address = 0xFFFE D800, Offset Address = 0x1C + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	DEI	Channel destination element index	R/W	N/A

- Channel destination element index

  Contains the channel destination element index, expressed as a signed value in bytes, which is used to compute the addresses, when single/ double indexed addressing mode is used.

Note:

When DMA_ CCR[10] = 1, then destination_element_index = DMA_CDEI.

When DMA_ CCR[10] = 0, then destination_element_index = DMA_CSEI.

Table 68. DMA Channel Destination Frame Index Register (DMA_CDFI)

Base Address = 0xFFFE D800, Offset Address = 0x1E + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	DFI	Channel destination frame index	R/W	N/A

- Channel destination frame index

  Contains the channel destination frame index, expressed as a signed value in bytes, which is used to compute the addresses when double indexed addressing mode is used.

Note:

When DMA_CCR[10] = 1, then destination_frame_index = DMA_CDFI

When DMA_CCR[10] = 0, then destination_frame_index = DMA_CSFI

Table 69. DMA COLOR Parameter Register (DMA_COLOR_L)

Base Address = 0xFFFE D800, Offset Address = 0x20 + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	FCL	Channel BLT foreground color (LSW)	R/W	N/A

This register can be used to provide parameter for DMA constant fill and transparent copy features. It must be configured in big endian format.

- If DMA_CCR2[Constant_Fill_Enable] = 1, then it defines the parameter for constant filling.

  If data_type = 8 bit, then

  DMA_COLOR_L[7:0]: it defines parameter for constant filling

  If data_type = 16 bit, then

  DMA_COLOR_L[15:0]: it defines parameter for constant filling

  If data_type = 32 bit, then

  DMA_COLOR_L[15:0]: it defines parameter[15:0] (LSW) for constant filling

  DMA_COLOR_U[15:0]: it defines parameter[31:16] (MSW) for constant filling

- If DMA_CCR2[Transparent_Copy_Enable] = 1, then it defines color key parameter for transparent copy.

  If data_type = 8 bit, then

  DMA_COLOR_L[7:0]: it defines color key for transparent copy

  If data_type = 16 bit, then

  DMA_COLOR_L[15:0]: it defines color key for transparent copy

  If data_type = 32 bit, then

  DMA_COLOR_L[15:0]: it defines color key[15:0] (LSW) for transparent copy

  DMA_COLOR_U[15:0]: it defines color key[31:16] (MSW) for transparent copy

Table 70. DMA COLOR Parameter Register (DMA_COLOR_U)

Base Address = 0xFFFE D800, Offset Address = 0x22 + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	FCU	Channel BLT foreground color (MSW)	R/W	N/A

For more details, see Table 69, *DMA COLOR Parameter Register.*

Table 71. DMA Channel Control Register 2 (DMA_CCR2)

Base Address = 0xFFFE D800, Offset Address = 0x24 + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:3	RESERVED	Reserved	R/W	N/A
2	BS	Block Synchronization	R/W	N/A
1	TCE	Transparent copy enable	R/W	N/A
0	CFE	Constant fill enable	R/W	N/A

- BS**:** Block synchronization

  This bit and the fs bit in the DMA_CCR register are used to program the way that a DMA_request is serviced in a synchronized transfer.

- Transparent copy enable

  1: Transparent copy operation is enabled. During transparent copy operation, any source data type that matches the DMA_COLOR_U/L registers is not written to the destination.

  0: Transparent copy operation is disabled. During transparent copy operation, any source pixel is written to the destination (No mask, brush are supported).

- Constant fill enable

  1: Constant fill operation is enabled. During constant fill operation, it writes destination with DMA_COLOR_U/L, instead of data from source.

  0: Constant fill operation is disabled. During BLT operation, any source data is written to the destination without constant fill.

Table 72. DMA Logical Channel Link Control Register (DMA_CLNK_CTRL)

Base Address = 0xFFFE D800, Offset Address = 0x28 + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15	EL	Enable_Lnk	R/W	0
14	SL	Stop_Lnk	R/W	0
13:5	RESERVED	Reserved	R/W	N/A
4	NID	Next LCh_ID	R/W (read as 0)	0
3:0	NID	Next LCh_ID	R/W	0000

Used to control logical channel linked queue.

- NEXTLCH_ID: Defines the NEXTLCH_ID, which is used to build logical channel chaining queue. The LOGICAL_CHANNEL_I is enabled, after the current logical channel finishes transfer (i = 0-15).
- STOP_LNK: Disables the logical channel on channel linked queue.

  1: The logical channel, defined by NEXTLCH_ID, is disabled, and ENABLE_LNK is disabled.

  0: No logical channel in chained is disabled.
- ENABLE_LNK: Defines the logical channel that is on channel linked queue.

  1: The logical channel, defined by NEXTLCH_ID, is enabled after the current channel finishes transferring.

  0: No logical channel is chained after the current logical channel.

Table 73. DMA Logical Channel Control Register (DMA_LCH_CTRL)

Base Address = 0xFFFE D800, Offset Address = 0x2A + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15	LID	LCH Interleave Disable	R/W	0
14:4	RESERVED	Reserved	R/W	N/A
3:0	LT	LCH Type	R/W	0000

Used to control logical channel access.

- LCH_Type: Defines the logical channel assignment relationship to physical channel and associated features.
  - 0000: LCh-2D dynamically shares PCh-0 and -1
  - 0001: LCh-G dynamically shares PCh-0 and -1
  - 0010: LCh-P dynamically shares PCh-0 and -1
  - 0111: LCh-PD PCh-2

  All other configurations result in undefined effects.
- LCH_Interleave_Disable: Defines synchronized logical channel interleave mode enable.

  0: Synchronized logical channel interleave mode enabled. The logical channel transfer can be interleaved at the end of DMA request transfers.

  1: A synchronized logical channel interleave mode is disabled. The logical channel takes control of the PCh until the entire DMA data has been transferred, regardless of DMA requests. The LCh/PCh cannot be preempted when interleaving is disabled, regardless of the LCh priority setting.

3.3.3 LCD Channel Dedicated Registers

Table 74 lists the DMA channel dedicated registers. Table 75 through Table 96 describe the register bits.

Table 74. DMA Logical Channel Configuration Registers

Base Address = FFFE EC00			
Name	**Description**	**R/W**	**Offset†**
DMA_LCD_CSDP	DMA LCD channel source destination parameters	R/W	0xC0
DMA_LCD_CCR	DMA LCD channel control	R/W	0xC2
DMA_LCD_CTRL	DMA LCD control	R/W	0xC4
TOP_B1_L	DMA LCD top address B1 L	R/W	0xC8
TOP_B1_U	DMA LCD top address B1 U	R/W	0xCA
BOT_B1_L	DMA LCD bottom address B1 L	R/W	0xCC
BOT_B1_U	DMA LCD bottom address B1 U	R/W	0xCE
TOP_B2_L	DMA LCD top address B2 L	R/W	0xD0
TOP_B2_U	DMA LCD top address B2 U	R/W	0xD2
BOT_B2_L	DMA LCD bottom address B2 L	R/W	0xD4
BOT_B2_U	DMA LCD bottom address B2 U	R/W	0xD6
DMA_LCD_SRC_EI_B1	DMA LCD source element index B1	R/W	0xD8
DMA_LCD_SRC_FI_B1_L	DMA LCD source frame index B1 L	R/W	0xDA
DMA_LCD_SRC_FI_B1_U	DMA LCD source frame index B1 U	R/W	0xF4
DMA_LCD_SRC_EI_B2	DMA LCD source element index B2	R/W	0xDC
DMA_LCD_SRC_FI_B2_L	DMA LCD source frame index B2 L	R/W	0xDE
DMA_LCD_SRC_FI_B2_U	DMA LCD source frame index B2 U	R/W	0xF6
DMA_LCD_SRC_EN_B1	DMA LCD source element number B1	R/W	0xE0
DMA_LCD_SRC_FN_B1	DMA LCD source frame number B1	R/W	0xE4
DMA_LCD_SRC_EN_B2	DMA LCD source element number B2	R/W	0xE2

† Note: Some offsets are not in numerical order to facilitate the relationship between the upper and lower words of some registers.

Table 74. DMA Logical Channel Configuration Registers (Continued)

Base Address = FFFE EC00			
Name	**Description**	**R/W**	**Offset†**
DMA_LCD_SRC_FN_B2	DMA LCD source frame number B2	R/W	0xE6
DMA_LCD_LCH_CTRL	DMA LCD logical channel control	R/W	0xEA

† Note: Some offsets are not in numerical order to facilitate the relationship between the upper and lower words of some registers.

Table 75. DMA LCD Channel Source Destination Parameters Register (DMA_LCD_CSDP)

Base Address = 0xFFFE E300, Offset Address = 0xC0				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:14	BURST_EN_B2	Burst enable for block 2. (Tie off = 10)	R/W	00
13	PACK_EN_B2	Pack enable for block 2	R/W	0
12:11	DATA_TYPE_B2	Data type for block 2. (Tie off = 10)	R/W	00
10:9	RESERVED	Reserved	R/W	ND
8:7	BURST_EN_B1	Burst enable for block 1. (Tie off = 10)	R/W	00
6	PACK_EN_B1	Pack enable for block 1	R/W	0
5:2	RESERVED	Reserved	R/W	ND
1:0	DATA_TYPE_B1	Data type for block 1. (Tie off = 10)	R/W	00

The OMAP_3.1_mode tie-off values are the values given to the bits in OMAP3.1 compatible mode, since the DMA_LCD_CCR, DMA_LCH_CTRL, and DMA_LCD_CSDP registers do not exist in the compatible mode. These values are tied off/on in hardware.

Used to control the LCD channel dual/single block transferring.

- data_type_b1/data_type_b2: Defines the type of data moved in the LCD channel from source port for block_1/block_2

DATA_TYPE [15:14] or [8:7]	type
00	s8: 8 bits scalar
01	s16: 16 bits scalar
10	s32: 32 bits scalar
11	illegal value

Start_Address must be aligned on the boundary of the type of data moved. For example, if data_type is s32 the source start address must be aligned on a word32. If data_type is s8, source start_address can have any value.

It is the software's responsibility to make sure that start address is aligned with channel data_type.

- PACK_EN_B1/PACK_EN_B2: packing enable for block_1/block_2 transfer

  The DMA ports can have a data bus width different from the type of data moved by the DMA channel. For example, s8 data_type can be read on a 32-bit DMA port. The DMA channel has the capacity to pack four consecutive s8 data reads in a single 32-bit read access in order to increase transfer bandwidth.

  Pack_en_b1/Pack_en_b2 = 1: The source port makes packed accesses.

  Pack_en_b1/Pack_en_b2 = 0: The source port never makes packed accesses.

- BURST_EN_B1 / BURST_EN_B2: burst enable for block_1 / block_2 transfer

  Used to enable bursting on the source port. When bursting is enabled, the source port performs bursts 4 x src_width access. When bursting is disabled, the source port performs single accesses of src_width access.

SRC_BURST_EN[1:0]	Burst Type
00	Single access
01	Reserved
10	Burst 4x port_width (Support 4x32bit accesses)
11	Reserved

If the source port of the channel has no burst access capability, this field is ignored.

Table 76. DMA LCD Channel Control Register (DMA_LCD_CCR)

Base Address = 0xFFFE E300, Offset Address =0xC2				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:14	SRC_AMODE_B2	Source addressing mode for block 2. (Tie off: 01)	R/W	00
13:12	SRC_AMODE_B1	Source addressing mode for block 2. (Tie off 01)	R/W	00
11	end_prog	End of programming status bit. (Tie off = 0)	R/W rst	00

Table 76. DMA LCD Channel Control Register (DMA_LCD_CCR) (Continued)

Bit	Name	Function	R/W	Reset
10	OMAP3_1_Compatible_ disable	OMAP3.1 channel compatibility control.	R/W	0
9	REPEAT	Repetitive operation. (Tie off = 1)	R/W	0
8	AUTOINIT	Autoinitialize at the end of the transfer. (Tie off = 1)	R/W	0
7	ENABLE	Enable transfer. (Tie off = 1)	R/W rst	0
6	PRIO	Channel priority. (Tie off = 1)	R/W	0
5	RESERVED	Reserved	R/W	ND
4	BS	Block synchronize	R/W	ND
3:0	RESERVED	Reserved	R/W	ND

The OMAP_3.1_mode tie-off values are the values given to the bits in OMAP3.1 compatible mode, since the DMA_LCD_CCR, DMA_LCH_CTRL, and DMA_LCD_CSDP registers do not exist in the compatible mode. These values are tied off/on in hardware.

- bs: block synchronize

  If the external LCD controller is the destination and bs = 1:

  Then the DMA LCD channel is synchronized on blocks. This means a block transfer is started each time the LCh is enabled and a hardware synchronization signal is received from the external LCD controller.

  - One DMA LCD channel request triggers one block transfer, in both single and dual block modes.
  - Two DMA LCD channel requests are required to trigger two block transfers, even if it is in dual block mode.

  If the external LCD controller is enabled and bs = 0

  Then the LCD channel is a software-triggered channel. This means a transfer can start as soon as the LCh is enabled.

  If the OMAP LCD controller is the destination, the bs bit is ignored.

  If auto init and repeat and/or end prog are set, a hardware request for successive block transfers is required.

- Prio_0 : Channel priority

  Prio_0 = 1: The channel has the high-priority level.

  Prio_0 = 0: The channel has the low-priority level.

- Enable:

  This bit is used to enable/disable the transfer in the DMA LCD channel when OMAP external LCD controller is selected.

  Enable = 1: The transfer starts.

  Enable = 0: The transfer stops and is reset.

  This bit is automatically cleared by the DMA hardware once the transfer is completed. However, if AUTO_INIT = 1 and REPEAT = 1, the channel continues without being disabled. Clearing of this bit by the DMA has priority over a write by the processor. If both occur simultaneously, the configuration write is discarded.

Note:

If the software must use this bit as a flag, it must do a back to back read to make sure that correct value of this enable bit is latched. This is because the enable bit can be asynchronously reset at the end of channel transfer, and a wrong value may send back when this risk condition happens.

- AUTO_INIT: Autoinitialization at the end of the transfer

  auto_init = 1: LCD channel is in autoinitialization mode. Once the current transfer is complete, the channel automatically reinitializes itself and starts a new transfer, if REPEAT or END_PROG bits is set. Else, if neither REPEAT nor END_PROG equals 1, wait for the END_PROG to be set to 1 to reload the channel with a new context and let the transfer restart.

  auto_init = 0: LCD channel is in non-autoinitialization mode. The DMA channel stops at the end of the current transfer.

- REPEAT: repetitive operation

  Repeat = 1: When LCD channel is in autoinitialization mode, once the current transfer is complete the channel automatically reinitializes itself and starts a new transfer with the previous or a new context depending on the end_prog bit (0/1).

  Repeat = 0: When LCD channel is in autoinitialization mode, once the current transfer is complete the channel automatically reinitializes itself and starts a new transfer only if END_PROG = 1.

  The repeat bit is not considered if AUTO_INIT = 0.

- OMAP3_1_COMPATIBLE_DISABLE: Omap3.1 channel compatibility control.

  This bit is used to set DMA LCD channel programming model.

  omap3_1_compatible_disable = 1: The LCD channel is in OMAP3.2 compatible mode.

  omap3_1_compatible_disable = 0: The LCD channel is in OMAP3.1 compatible mode.

- END_PROG: end of programming status bit

  end_prog = 1: When LCD channel is in autoinitialization mode, it allows the channel to reinitialize itself with a new channel context (the program register set is copied to the active register set) after the current DMA channel transfer has been finished. The end_prog bit automatically resets itself when the new context has been loaded.

  end_prog = 0: When LCD channel is in autoinitialization mode, and REPEAT bit is 1, DMA continues LCD next transfer with the same channel context. If REPEAT = 0, the channel does not reinitialize itself and does not start a new transfer.

  The end_prog bit is not considered if AUTO_INIT = 0.

- SRC_AMODE_B1/SRC_AMODE_B2: Source addressing mode for block1 or block2 transfer. This field is used to choose the addressing mode on the source port of the channel.

  src_amode_b1/b2 = 00: Reserved

  src_amode_b1/b2 = 01: Post-incremented address

  src_amode_b1/b2 = 10: Single index (element index)

  src_amode_b1/b2 = 11: Double index (element and frame indexes)

There is no need for a dst_amode_b1/dst_amode_b2 because there is no write address to a destination port in the LCD channel case.

Table 77. DMA LCD Control Register (DMA_LCD_CTRL)

Base Address = 0xFFFE E300, Offset Address = 0xC4				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:9	RESERVED	Reserved.	R/W	ND
8	LDP	LCD destination port (OMAP or external LCD controller).	R/W	0
7:6	LSP	LCD source port. Memory source for the LCD channel.	R/W	00

Table 77. DMA LCD Control Register (DMA_LCD_CTRL) (Continued)

Base Address = 0xFFFE E300, Offset Address = 0xC4				
Bit	**Name**	**Function**	**R/W**	**Reset**
5	BUS_ERROR_IT_ COND	Bus error interrupt condition.	R/W (read as 0)	0
4	BLOCK_2_IT_COND	Block 2 interrupt condition.	R/W	0
3	BLOCK_1_IT_COND	Block 1 interrupt condition.	R/W	0
2	BUS_ERROR_IT_IE	Buss error interrupt enable.	R/W	0
1	BLOCK_IT_IE	Block interrupt enable.	R/W	0
0	BLOCK_MODE	Type of block mode used for the LCD transfer.	R/W	0

The DMA LCD control register contains nine bit-fields, which control the LCD channel operation. There are two cases of interrupt: end block buffer or abort on the bus (bus error). Bits block_it_ie and bus_error_iie (interrupt enable) enable the generation of the interrupt.

If the status bits (xxx_cond bits and the corresponding interrupt enable bits xxx_ie bits) are all set, an interrupt signal is sent from the DMA LCD channel to the CPU. The CPU reads this register to find the cause of the interrupt.

Users must write to this register to clear the status bits.

- BLOCK_MODE: Type of block mode used for the LCD transfer

  0: One block buffer; only registers relative to block 1 are used.

  1: Two block buffers; the LCD channel reads alternatively top_block_1 and top_block_2.

- BLOCK_IT_IE: end block interrupt enable

  0: Interrupt disabled.

  1: Interrupt enabled.

  This bit enables an end of block interrupt for either block 1 or 2 when dual block mode is selected.

- BUS_ERROR_IT_IE: bus error interrupt enable

  0: Interrupt disabled.

  1: Interrupt enabled.

- BLOCK_1_IT_COND: status LCD channel bit. Users must write to this register to clear the status bits.

  0: No end of block 1 interrupt detected.

  1: End of block 1 interrupt detected.

- BLOCK_2_IT_COND: status LCD channel bit. Users must write to this register to clear the status bits.

  0: No end of block 2 interrupt detected.

  1: End of block 2 interrupt detected.

- BUS_ERROR_IT_COND: status LCD channel bit. Users must write to this register to clear the status bits.

  0: No bus error interrupt detected.

  1: Bus error interrupt detected.

- LCD_SOURCE_PORT: memory source for the LCD channel

  This bit indicates which memory source is selected for the next LCD transfer.

  00: Memory source is SDRAM.

  01: Memory source is L3_OCP_T1 (Camera).

  10: Memory source is L3_OCP_T2 (Test SRAM).

  11: Reserved.

- LCD_DESTINATION_PORT: LCD controller for the LCD channel

  This bit indicates which LCD controller is selected for the next LCD transfer.

  0: OMAP controller is connected to the DMA LCD channel.

  1: External LCD controller is connected to the DMA LCD channel.

Note:

To enable the external LCD controller, users must have the external LCD clock active (even if it is not ready to send an image). The following must be set to accomplish this:

DMA_GSCR.OMAP31_MAPPING_DISABLE = 1

DMA_LCD_CTRL.ICD_DESTINATION_PORT = 1

DMA_LCD_CCR.OMAP31_COMPATIBLE_DISABLE = 1

In order to enter deep idle mode, users must set DMA_LCD_CTRL.LCD_DESTINATION_PORT = 0. Indeed, DMA_LCD_CTRL.LCD_DESTINATION_PORT = 1 is a condition that prevents OMAP3.2 from going to idle state because the clock request corresponding to external LCD controller clock is kept active. Consequently, deep idle mode cannot be initiated.

DMA LCD Top Address B1 Registers

The LCD top address B1 registers are two 16-bit registers, which contain the start address for the video RAM buffer 1. The 32-bit address is obtained by the concatenation of the two word16 as described here:

LCD_TOP_B1 = DMA_LCD_TOP_B1_U & DMA_LCD_TOP_B1_L

Note:

The LSB of the word32 is equal to zero. Address of video buffer must always be even.

Table 78. DMA LCD Top Address B1 L Register (TOP_B1_L)

Base Address = 0xFFFE E300, Offset Address = 0xC8				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:1		LCD TOP address for block buffer 1 lower bits	R/W	ND
0		0 (always tied to 0)	R	0

Table 79. DMA LCD Top Address B1 U Register (TOP_B1_U)

Base Address = 0xFFFE E300, Offset Address = 0xCA				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0		LCD TOP address for block buffer 1 upper bits	R/W	ND

DMA LCD Bottom Address B1 Registers

The LCD bottom address B1 registers are two 16-bit registers that contain the bottom address for the video RAM buffer 1. The 32-bit address is obtained by the concatenation of the two word16 as described here:

LCD_BOTTOM_B1 = DMA_LCD_BOT_B1_U & DMA_LCD_BOT_B1_L

Note:

The LSB of the word32 is equal to zero. Address of video buffer must always be even.

Table 80. DMA LCD Bottom Address B1 L Register (BOT_B1_L)

Base Address = 0xFFFE E300, Offset Address = 0xCC				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:1		LCD BOT address for block buffer 1 lower bits	R/W	ND
0		0 (always tied to 0)	R	0

Table 81. DMA LCD Bottom Address B1 U Register (BOT_B1_U)

Base Address = 0xFFFE E300, Offset Address = 0xCE				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0		LCD BOT address for block buffer 1 upper bits	R/W	ND

DMA LCD Top Address B2 Registers

The LCD top address B2 registers are two 16-bit registers that contain the start address for the video RAM buffer 2. The 32-bit address is obtained by the concatenation of the two word16 as described here:

LCD_TOP_B2 = DMA_LCD_TOP_B2_U & DMA_LCD_TOP_B2_L

Note:

The LSB of the word32 is equal to zero. Address of video buffer must always be even.

Table 82. DMA LCD Top Address B2 L Register (TOP_B2_L)

Base Address = 0xFFFE E300, Offset Address = 0xD0				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:1		LCD TOP address for block buffer 2 lower bits	R/W	ND
0		0 (always tied to 0)	R	0

Table 83. DMA LCD Top Address B2 U Register (TOP_B2_U)

Base Address = 0xFFFE E300, Offset Address = 0xD2				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0		LCD TOP address for block buffer 2 upper bits	R/W	ND

DMA LCD Bottom Address B2 Registers

The LCD bottom B2 address registers are two 16-bit registers that contain the bottom address for the video RAM buffer 2. The 32-bit address is obtained by the concatenation of the two word16 as described here:

LCD_BOTTOM_B2 = DMA_LCD_BOT_B2_U & DMA_LCD_BOT_B2_L

Note:

The LSB of the word32 is equal to zero. Address of video buffer must always be even.

Table 84. DMA LCD Bottom Address B2 L Register (BOT_B2_L)

Base Address = 0xFFFE E300, Offset Address = 0xD4				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:1		LCD BOT address for block buffer 2 lower bits	R/W	ND
0		0 (always tied to 0)	R	0

Table 85. DMA LCD Bottom Address B2 U Register (BOT_B2_U)

Base Address = 0xFFFE E300, Offset Address = 0xD6				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0		LCD BOT address for block buffer 2 upper bits	R/W	ND

Table 86. DMA LCD Source Element Index B1 Register (DMA_LCD_SRC_EI_B1)

Base Address = 0xFFFE E300, Offset Address = 0xD8				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0		LCD source element index for block 1	R/W	ND

- LCD channel source element index for block 1

  Contains the channel source element index for LCD video RAM buffer 1, expressed as signed value in bytes, which is used to compute the addresses when single or double-indexed addressing mode is used.

Table 87. DMA LCD Source Frame Index B1 Register L (DMA_LCD_SRC_FI_B1_L)

Base Address = 0xFFFE E300, Offset Address = 0xDA				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0		LCD source frame index for block 1 L	R/W	ND

Table 88. DMA LCD Source Frame Index B1 Register U (DMA_LCD_SRC_FI_B1_U)

Base Address = 0xFFFE E300, Offset Address = 0xF4				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0		LCD source frame index for block 1 U	R/W	ND

❑ LCD channel source frameindex for block 1

These registers contain the channel source frame index for video RAM buffer 1, expressed as a signed value in bytes, which is used to compute addresses when double-indexed addressing mode is used.

Table 89. DMA LCD Source Element Index B2 Register (DMA_LCD_SRC_EI_B2)

Base Address = 0xFFFE E300, Offset Address = 0xDC				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0		LCD source element index for block 2	R/W	ND

❑ LCD channel source element index for block 2

Contains the channel source element index for LCD video RAM buffer 2, expressed as signed value in bytes, which is used to compute the addresses when single or double-indexed addressing mode is used.

Table 90. DMA LCD Source Frame Index B2 Register L (DMA_LCD_SRC_FI_B2_L)

Base Address = 0xFFFE E300, Offset Address = 0xDE				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0		LCD source frame index for block 2 L	R/W	ND

Table 91. DMA LCD Source Frame Index B2 Register U (DMA_LCD_SRC_FI_B2_U)

Base Address = 0xFFFE E300, Offset Address = 0xF6				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0		LCD source frame index for block 2 U	R/W	ND

❑ LCD channel source frame index for block 2

Contains the channel source frame index for video RAM buffer 2, expressed as signed value in bytes, which is used to compute addresses when double-indexed addressing mode is used.

Table 92. DMA LCD Source Element Number B1 Register (DMA_LCD_SRC_EN_B1)

Base Address = 0xFFFE E300, Offset Address = 0xE0				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0		LCD channel source element number for block 1	R/W	ND

❑ LCD channel source element number for block 1

Number of elements within a frame (unsigned) for the video RAM buffer 1. The maximum element number is 65535.

Table 93. DMA LCD Source Frame Number B1 Register (DMA_LCD_SRC_FN_B1)

Base Address = 0xFFFE E300, Offset Address = 0xE4				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0		LCD channel source frame number for block 1	R/W	ND

❑ LCD channel source frame number for block 1

Number of frames within a block (unsigned) for the video RAM buffer 1. The maximum frame number is 65535.

The size in bytes of the data block to transfer is:

data_block_in_bytes = DMA_LCD_SRC_FN_B1 x
DMA_LCD_SRC_EN_B1 x
DMA_LCD_CDSP[DATA_TYPE_B1]

Table 94. DMA LCD Source Element Number B2 Register (DMA_LCD_SRC_EN_B2)

Base Address = 0xFFFE E300, Offset Address = 0xE2				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0		LCD channel source element number for block 2	R/W	ND

❑ LCD channel source element number for block 2

Number of elements within a frame (unsigned) for the video RAM buffer 2. The maximum element number is 65535.

Table 95. DMA LCD Source Frame Number B2 Register (DMA_LCD_SRC_FN_B2)

Base Address = 0xFFFE E300, Offset Address = 0xE6				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0		LCD channel source frame number for block 2	R/W	ND

❑ LCD channel source frame number for block 2

Number of frames within a block (unsigned) for the video RAM buffer 2. The maximum frame number is 65535. The size in bytes of the data block to transfer is:

data_block_in_bytes = DMA_LCD_SRC_FN_B2 x
DMA_LCD_SRC_EN_B2 x
DMA_LCD_CSDP[DATA_TYPE_B2]

Table 96. DMA LCD Logical Channel Control Register (DMA_LCD_LCH_CTRL)

Base Address = 0xFFFE E300, Offset Address = 0xEA				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:4	RESERVED	Reserved	R/W	ND
3:0	LCH_TYPE	Logical channel type.	R/W	0000

Used to control logical channel access. The OMAP_3.1_mode tie-off values are those values given to the bits in OMAP 3.1 compatible mode, because the DMA_LCD_CCR, DMA_LCH_CTRL, and DMA_LCD_CSDP registers do not exist in the compatible mode. These values are tied off/on in hardware. (For LCH_TYPE, the tie-off value is 0000).

- LCH_Type:

  Defines logical channel assignment relationship to associated features and physical channel.

  The only possible configuration is: 0100: LCh-D PCh-D. This bit field must be configured as above to secure forward compatibility.

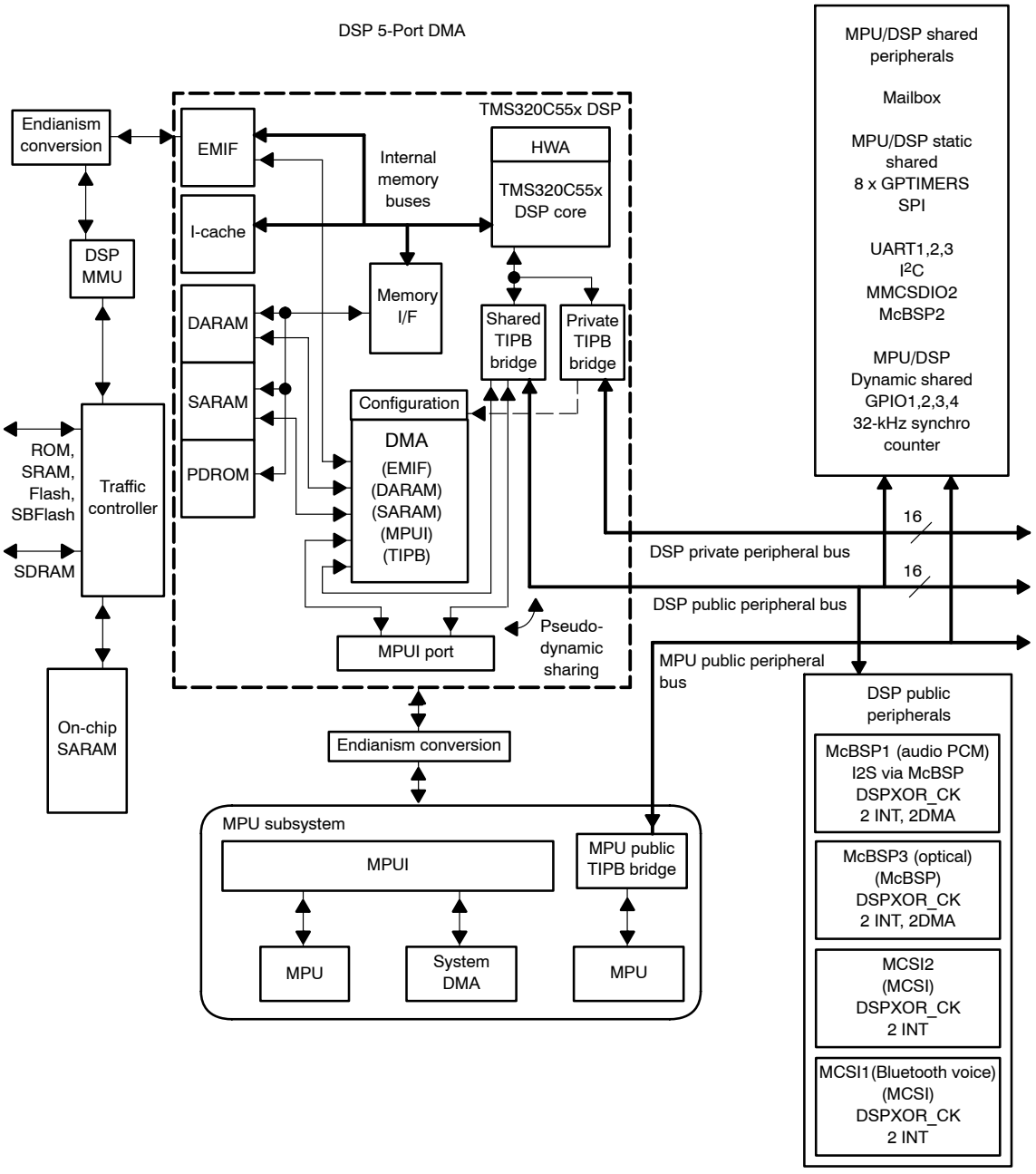
4 DSP DMA

Acting in the background of MPU operation, the DSP DMA controller can:

- Transfer data among internal memory, external memory, and peripherals residing on the DSP public peripheral bus
- Transfer data between the MPUI and internal memory

Figure 14 shows the ports serviced by the DMA controller within the context of the DSP subsystem.

Figure 14. DSP DMA and Ports



4.1 DSP DMA Controller Features

The DSP DMA controller has the following features:

- Operation independent of the MPU.
- Four standard ports, one for each data resource: DARAM, SARAM, external memory via EMIF, and peripherals via the shared TIPB bridge.
- An auxiliary port to enable certain transfers between the MPUI and memory.
- Six logical channels, which allow the DMA controller to keep track of the context of six independent block transfers plus a seventh logical channel for MPUI transfers.
- Bits for assigning each channel a low priority or a high priority. For details, see Section 4.4, *Service Chain*.
- Event synchronization. DMA transfers in each channel can be made dependent on the occurrence of selected events. For details, see Section 4.13, *Synchronizing Channel Activity*.
- An interrupt for each channel. Each channel can send an interrupt to the CPU on completion of certain operational events (see Section 4.16, *Monitoring Channel Activity*).
- Software-selectable options for updating addresses for the sources and destinations of data transfers.

The DMA EMIF does not have 8-bit read support. Internally, 8-bit DMA transfers from external to internal memory convert into 16-bit element reads. The appropriate byte is fetched from this read and placed in internal memory.

The DSP DMA controller performs data transfers between the following source and destination ports:

- Single-access RAM — SARAM port
- Dual-access RAM — DARAM port
- External memory — EMIF port
- TI peripheral bus — PERIPH port
- MPUI interface — MPUI port

Data transfers among the SARAM, DARAM, EMIF, and PERIPH ports can occur in six independent DMA channels (see Section 4.2, *Channels and Port Accesses*). Transfers between the MPUI port and memory ports (SARAM, DARAM, and EMIF) occur on a unique seventh channel dedicated to MPU operations. Transfers between the MPU and DSP peripherals are supported by a direct connection that does not involve the DSP DMA controller. Each channel is controlled by a set of configuration registers, where software sets up the transfer parameters, such as length, source address, and destination address. These registers are accessed in I/O space by the DSP via the TIPB bridge.

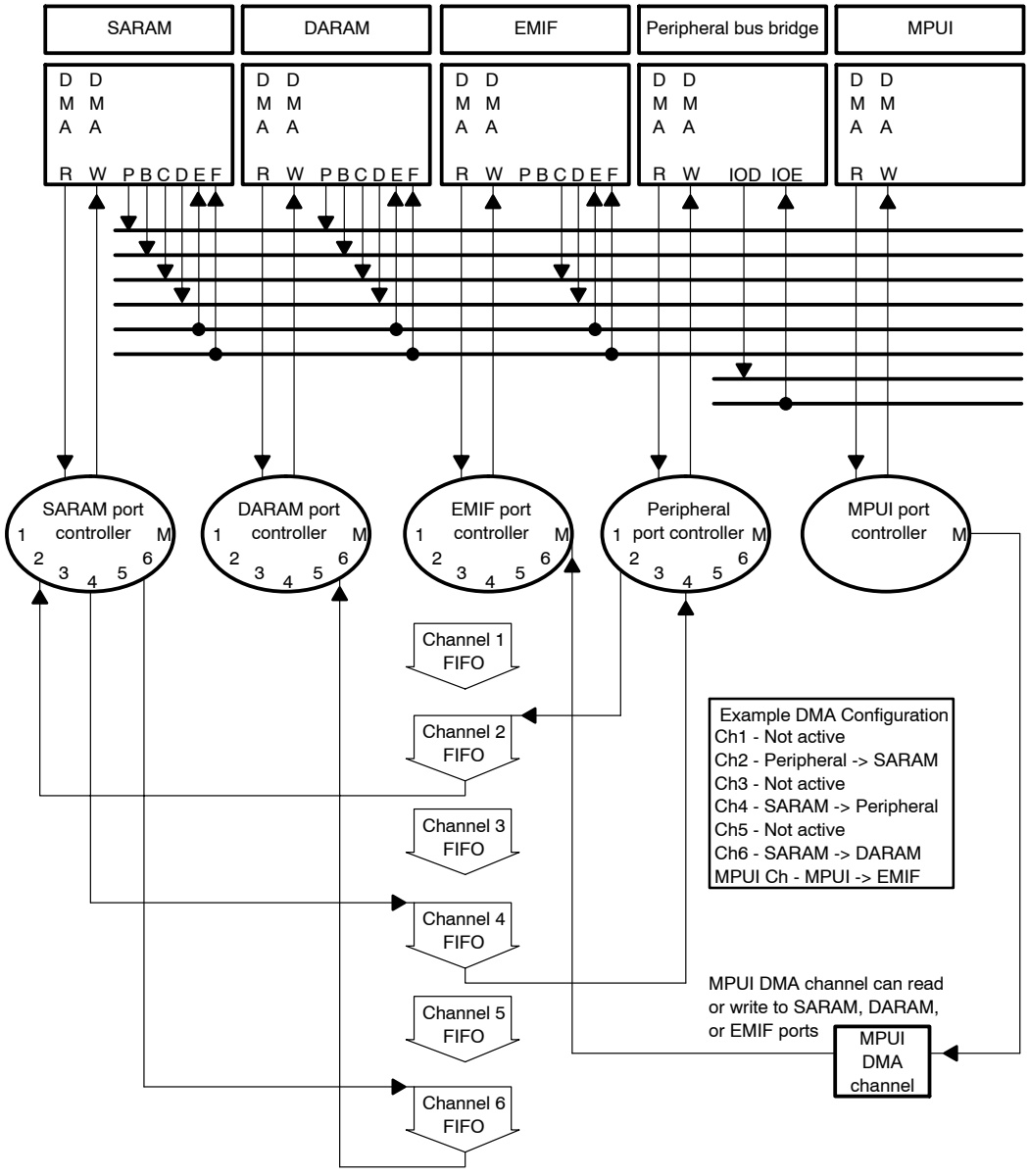
It is possible for multiple channels (or for one or more channels and the MPUI) to request access to the same standard port at the same time (see Table 97 and Figure 15). To arbitrate simultaneous requests, the DMA controller has one programmable service chain that is used by each of the standard ports (see Section 4.4, *Service Chain*).

Table 97. Possible DMA Transfers

SRC\DST	SARAM	DARAM	EMIF	Peripheral	MPUI
SARAM	Chan 0-5	Chan 0-5	Chan 0-5	Chan 0-5	MPU Chan
DARAM	Chan 0-5	Chan 0-5	Chan 0-5	Chan 0-5	MPU Chan
EMIF	Chan 0-5	Chan 0-5	Chan 0-5	Chan 0-5	MPU Chan
Peripheral	Chan 0-5	Chan 0-5	Chan 0-5	Chan 0-5	Direct
MPUI	MPU Chan	MPU Chan	MPU Chan	Direct	NA

Note: MPU transfers data to and from DSP peripherals via direct connection.

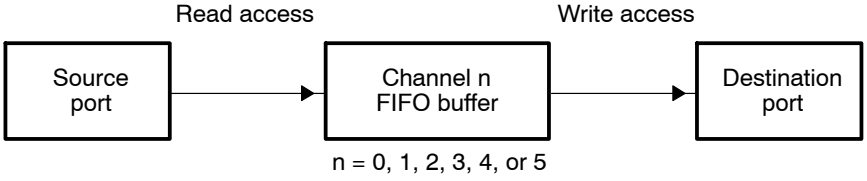
Figure 15. Example of DMA Configuration



4.2 Channels and Port Accesses

The DSP DMA controller has six paths, or channels, to transfer data among the four standard ports (for DARAM, SARAM, external memory, and peripherals). Each channel reads data from one port (the source) and writes data to that same port or another port (the destination). Each channel has a first-in, first-out (FIFO) buffer that allows the data transfer to occur in two stages: port read access transfer of data from the source port to the channel FIFO buffer and port write access transfer of data from the channel FIFO buffer to the destination port (see Figure 16).

Figure 16. The Two Parts of a DMA Transfer



The set of conditions under which transfers occur in a channel is called the channel context. Each of the six channels contains a register structure for programming and updating the channel context (see Figure 17). User code modifies the configuration registers. To transfer data, the contents of the configuration registers are copied to the working registers, and the DMA controller uses the working register values to control channel activity. The copy from the configuration registers to the working registers occurs whenever user code enables the channel (EN = 1 in DMA_CCR). In addition, if the autoinitialization mode is on (AUTO_INIT = 1 in DMA_CCR), the copy occurs between block transfers.

The DMA configuration registers consist of the following set:

- DMA_CSSA_L
- DMA_CSSA_U
- DMA_CDSA_L
- DMA_CDSA_U
- DMA_CEN
- DMA_CFN
- DMA_CSFI
- DMA_SCEI
- DMA_CDFI
- DMA_ CDEI

The following registers immediately impact the DMA channel operation and must only be modified while the channel is disabled:

- DMA_CSDP
- DMA_CCR
- DMA_CICR
- DMA_CSR

4.2.1 Channel Autoinitialization

Three control bits within the DMA_CCR register configure automatic reinitialization of the DMA channel (see Table 98):

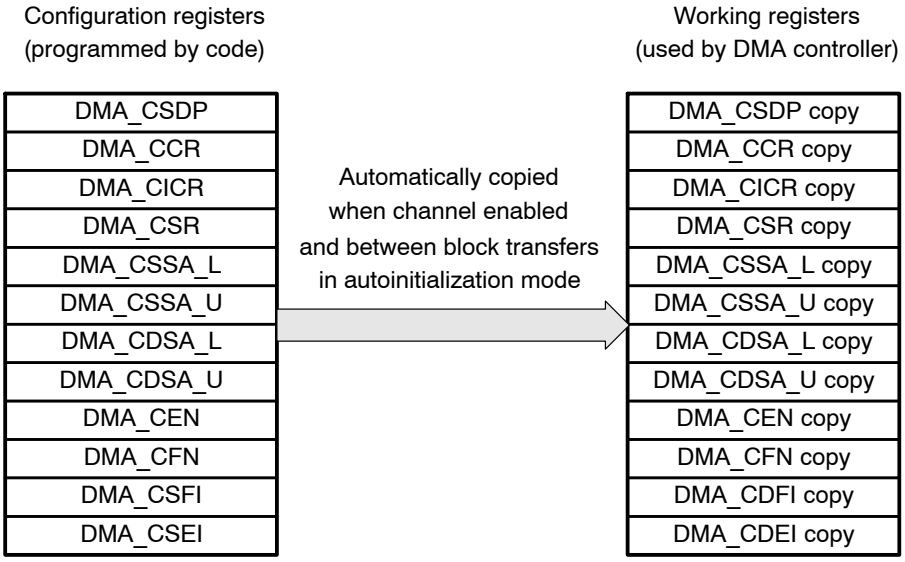
- The AUTOINIT bit: If AUTOINIT is set, then under certain conditions the channel reinitializes at the end-of-block by copying the configuration registers into the working set.
- The END_PROG bit: reinitializes the channel using software updated context registers. The program updates the DMA transfer context between one block transfer and the next (that is, different source address, different destination address, and so on). To ensure that the next block transfer does not start before software has finished updating the channel context registers, the DMA can be set to wait until the end of programming as indicated by the END_PROG bit in the DMA_CCR register. Software clears the END_PROG bit, reprograms the DMA channel context, and then sets the END_PROG bit back to 1. This ensures that the context is completely updated before a new transfer begins.
- REPEAT bit: reinitializes the channel using the same context registers as the prior block. Here the configuration registers are copied to the working set as soon as the block transfer completes. There is no need to indicate end of programming, so the END_PROG bit is not tested. To use this mode of operation, set the repeat bit in the DMA_CCR register.

Table 98. Autoinitialization Bits

AUTO_INIT	END_PROG	REPEAT	Autoinitialization Behavior
0	Don't care	Don't care	No autoinitialization
1	0	0	At end of current transfer, channel waits until END_PROG=1 to load the configuration register set into its working register set.
1	0	1	At the end of current transfer, channel immediately loads the configuration register set into its working register set.
1	1	Don't care	At the end of current transfer, channel immediately loads the configuration register set into its working register set.

Figure 17 shows the registers for controlling channel context.

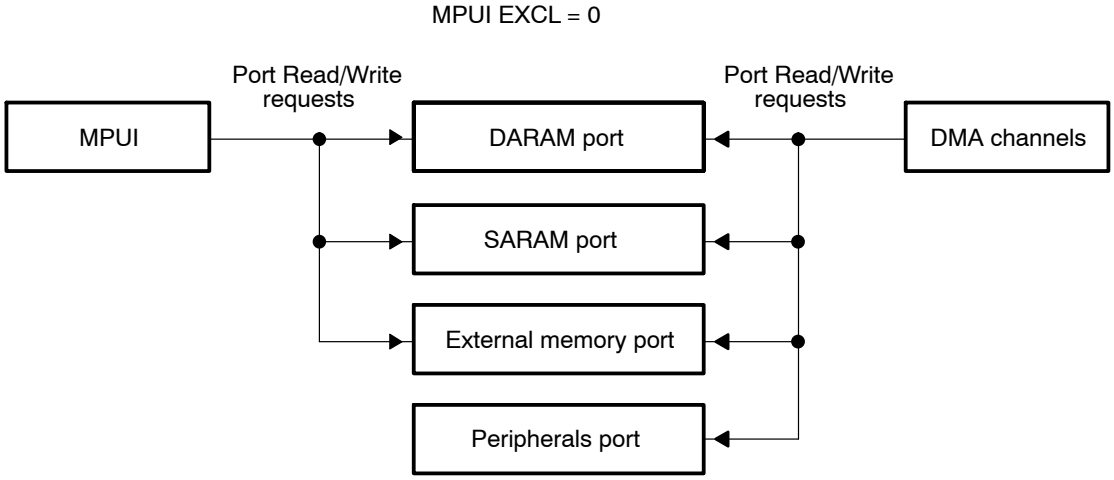
Figure 17. Registers for Controlling Channel Context



4.3 MPUI Access Configurations

As shown in Figure 18, the MPUI_EXCL bit in DMA_GCR determines the relationship between the MPUI and the DMA channels. When MPUI_EXCL = 0, the MPUI shares memory with the channels. When MPUI_EXCL = 1 , the MPUI cannot access external memory, but it can access internal RAM without interruptions from the channels. When MPUI_EXCL = 1, the DARAM port and the SARAM port operate as if all the channels were disconnected from the service chain. See Section 4.4, *Service Chain*.

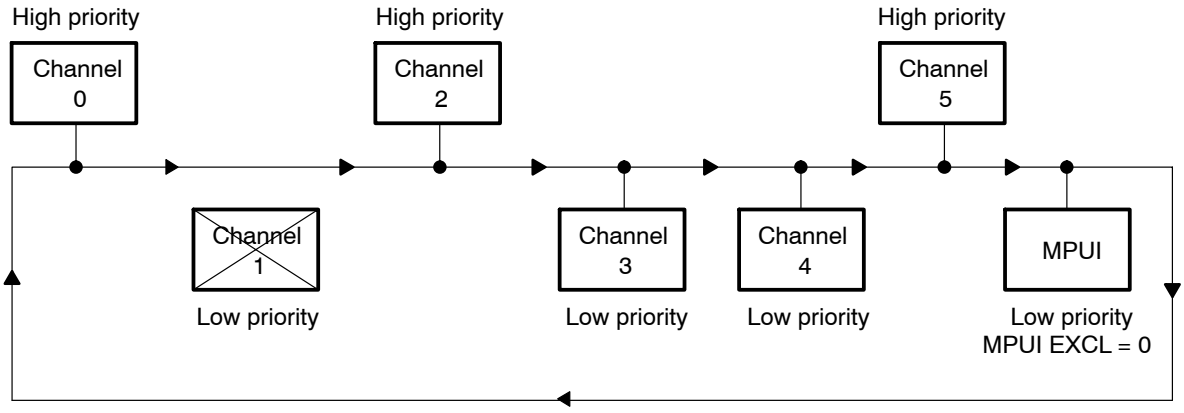
Figure 18. MPUI Access Configurations



4.4 Service Chain

Each of the standard ports can arbitrate simultaneous access requests sent by the six DMA channels and the MPUI. Each of the standard ports has an independently functioning service chain– a software-controlled and a hardware-controlled scheme for servicing access requests. Although the four standard port service chains (SARAM, DARAM, EMIF, PERIPH) function independently, they share a common configuration. For example, if channel 2 is disabled, it is disabled in the service chain of all four ports. If channel 4 is made a high priority, it is a high priority in all four ports. One possible configuration for the service chains is shown in Figure 19 (example of chain configuration applied to three ports).

Figure 19. Possible Configuration for Service Chain



The channels and the MPUI have a programmable priority level. Each channel has a priority bit in DMA_CCR for selecting a high priority or a low priority. The MPUI is assigned a high priority or a low priority with the MPUI_PRIO bit in DMA_GCR. The DMA controller services the low-priority items only when all of the high-priority items are done or stalled. After a DSP reset, all channels and the MPUI are low priority. In Figure 19, channels 0, 2, and 5 are high priority (in each of these channels, PRIO = 1). DMA channels 1, 3, and 4 and the MPUI are low priority (in each of these channels, PRIO = 0 and, for the MPUI, MPUI PRIO = 0).

The channels and the MPUI have fixed positions in the service chain. Regardless of the programmed priorities, the port checks the channels and the MPUI in a repeating circular sequence: 0, 1, 2, 3, 4, 5, MPUI, 0, 1, 2, 3, 4, 5, MPUI, and so on. At each position in the service chain, the port checks whether the channel/MPUI is ready and able to be serviced. If so, it is serviced; otherwise, the port skips to the next position. After a DSP reset, the port restarts its circular sequence, beginning with channel 0.

The channels can be individually connected or disconnected from the service chain through software. If a channel is enabled (EN = 1 in DMA_CCR), it is connected to the service chain; if it is disabled (EN = 0), it is disconnected. After a DSP reset, all channels are disconnected. In Figure 19, only channel 1 is disconnected. As a port checks the channels and the MPUI in its repeating circular sequence, it keeps skipping channel 1 until the channel is reconnected.

The MPUI cannot access the peripherals port. The peripherals port operates as though the MPUI were disconnected from the service chain.

By writing a 1 to the MPUI_EXCL bit in DMA_GCR, the MPUI is given exclusive access to the DARAM and SARAM ports. Then the DARAM and SARAM ports operate as if only the MPUI is connected to the service chain (as if none of the channels were connected, regardless of whether the channels are enabled). For more details, see Section 4.3, *MPUI Access Configurations*. The MPUI shares the RAM ports with the channels.

If a channel is tied to a synchronization event, the channel does not generate a DMA request (and, therefore, cannot be serviced) until the synchronization event occurs. To avoid long response times to synchronization events, always assign synchronized channels a high priority.

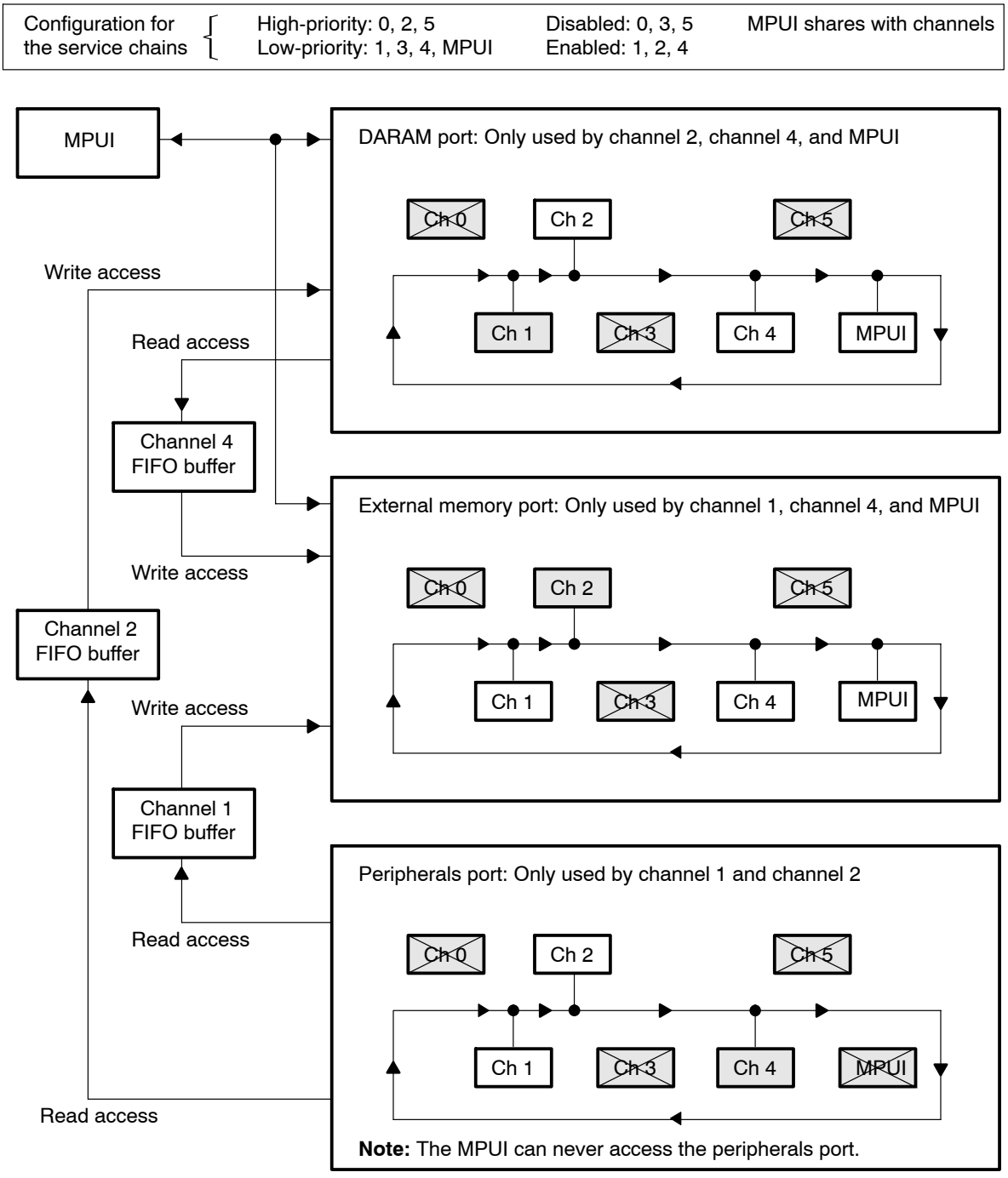
4.4.1 Service Chain Example

Figure 20 shows a DMA service chain applied to the DARAM port, the external memory port, and the peripherals port. The service chain has the following programmed characteristics. A list of activity in the ports is provided after Figure 20.

- Channels 0, 2, and 5 are high priority (PRIO = 1 in DMA_CCR). Channels 1, 3, and 4 are low priority (PRIO = 0).
- Channels 1, 2, and 4 are enabled (EN = 1 in DMA_CCR). Channels 0, 3, and 5 are disabled (EN = 0).
- The MPUI shares the internal memory with the channels (MPUI_EXCL = 0 in DMA_GCR) and is treated like a low-priority channel (MPUI_PRIO = 0 in DMA_GCR). The MPUI is shown as disconnected in the peripherals port, because the MPUI cannot access the peripherals port.

Table 99 summarizes the activity at the ports in Figure 20.

Figure 20. Service Chain Applied to Three DMA Ports



In this example Ch1: PERIPH -> External memory, Ch2: PERIPH -> DARAM, Ch4: DARAM -> Ext mem.

Table 99. Activity Shown in Figure 20

Port	This Port Arbitrates
DARAM	Write access requests from channel 2 Read access requests from channel 4 Read or write access requests from the MPUI
External memory	Write access requests from channel 1 Write access requests from channel 4 Read or write access requests from the MPUI
Peripherals	Read access requests from channel 1 Read access requests from channel 2

For each port in Figure 20, there is a channel that is connected to the service chain but does not use the port. For example, the peripherals port is not used by channel 4. If channel 4 is redefined to include the peripherals port as source or destination, the port handles channel 4 according to its position and priority in the service chain.

4.5 Units of Data

Four levels of data granularity are available when referring to a DMA transfer:

- Byte: An 8-bit value. A byte is the smallest unit of data transferred in a DMA channel.
- Element: One or more bytes to be transferred as a unit. Depending on the programmed data type, an element is an 8-bit, 16-bit, or a 32-bit value. An element transfer cannot be interrupted: all of its bytes are transferred to a port before another channel or the MPUI can take control of the port.
- Frame: One or more elements to be transferred as a unit. A frame transfer can be interrupted between element transfers.
- Block: One or more frames to be transferred as a unit. Each channel can transfer one block of data (once or multiple times). A block transfer can be interrupted between frame transfers and element transfers.

For each of the six DMA channels, you can define the number of frames in a block (with DMA_CFN), the number of elements in a frame (with DMA_CEN), and the number of bytes in an element (with the DATA_TYPE bits in DMA_CSDP). For descriptions of DMA_CFN, DMA_CEN, DMA_CSDP, and other registers of the DMA controller, see Section 4.22, *Control Configuration Registers*.

4.6 Start Addresses in a Channel

During a data transfer in a DMA channel, the first address at which data is read is called the source start address. The first address to which the data is written is called the destination start address. These are byte addresses. From the standpoint of the DMA controller, every 8 bits in memory or I/O space has its own address. Each channel contains registers for specifying the start addresses, as shown in Table 100.

Table 100. Registers Used to Define the Start Addresses for a DMA Transfer

Channel Address Register
DMA_CSSA_L source start address (lower part)
DMA_CSSA_U source start address (upper part)
DMA_CDSA_L destination start address (lower part)
DMA_CDSA_U destination start address (upper part)

The following sections explain how to load the start address registers for memory accesses and I/O accesses. The DMA controller can access all of the internal and external memory and all of I/O space (which contains registers for the DSP peripherals).

4.7 Start Address

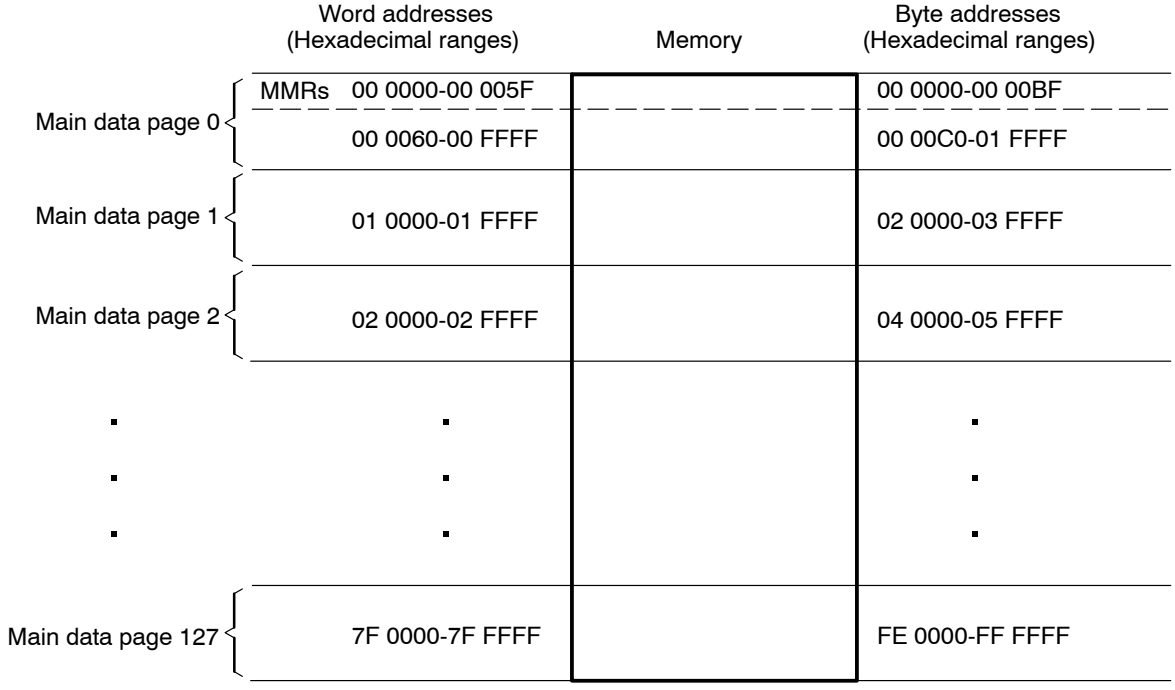
Figure 21 shows a high-level memory map for the DSP subsystem, showing both the word addresses (23-bit addresses) used by the CPU and the byte addresses (24-bit addresses) used by the DMA controller. To load the source/destination start address registers:

1) Identify the correct start address. Check for any alignment constraint for the data type; see the description for the data type bits in Section 4.23.2, *Channel Source Start Address.* If you have a word address, shift it left by 1 bit to form a byte address with 24 bits. For example, convert word address 02 4000h to byte address 04 8000h.

2) Load the 16 least significant bits (LSBs) of the byte address into DMA_CSSA_L (for source) or DMA_CDSA_L (for destination).

3) Load the eight most significant bits (MSBs) of the byte address into the eight LSBs of DMA_CSSA_U (for source) or DMA_CDSA_U (for destination).

Figure 21. High-Level Memory Map for DSP



Word addresses 00 0000h-00 005Fh (which correspond to byte addresses 00 0000h-00 00BFh) are reserved for the memory-mapped registers (MMRs) of the DSP CPU.

4.8 Start Address in I/O Space

Figure 22 shows an I/O space map for the DSP subsystem. The diagram shows both the word addresses (16-bit addresses) used by the CPU and byte addresses (17-bit addresses) used by the DMA controller. To load the source/destination start address registers:

1) Identify the correct start address. Check for any alignment constraint for the data type; see the description for the data type bits in Section 4.23.2, *Channel Source Start Address.* If you have a word address, shift it left by 1 bit to form a byte address with 17 bits. For example, convert word address 8000h to byte address 10000h.

2) Load the 16 LSBs of the byte address into DMA_CSSA_L (for source) or DMA_CDSA_L (for destination).

3) Load the MSB of the byte address into the LSB of DMA_CSSA_U (for source) or DMA_CDSA_U (for destination).

Figure 22. High-Level I/O Map for DSP

4.9 Updating Addresses in a Channel

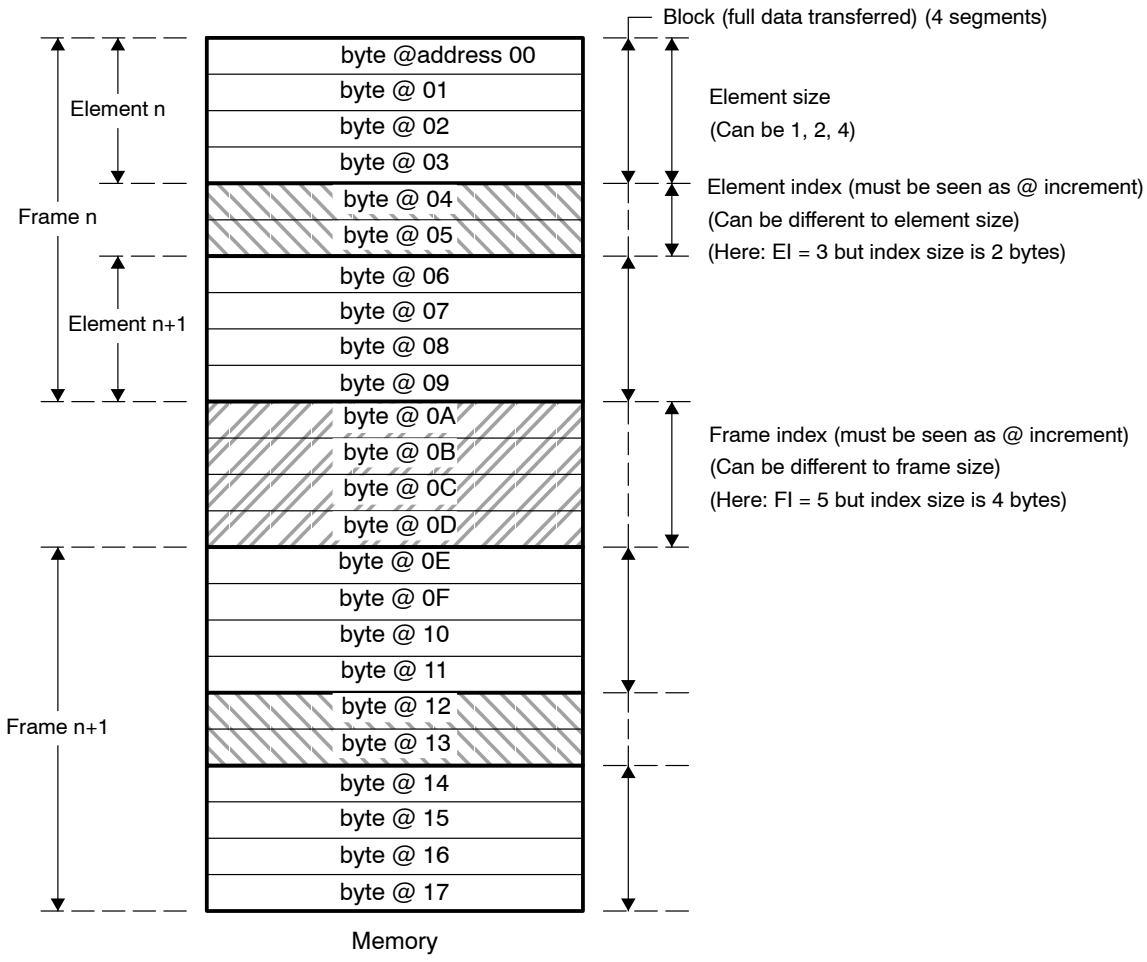
During data transfers in a DMA channel, the DMA controller begins its read and write accesses at the start addresses you specify (as described in Section 4.6, *Start Addresses in a Channel*). In many cases these addresses must be updated after a data transfer has begun so that data is read and written at consecutive or indexed locations. The address updates can be configured at two levels:

- Block-level address updates. In the autoinitialization mode (AUTO_INIT = 1 in DMA_CCR), block transfers can occur one after another until autoinitialization is turned off or the channel is disabled. To start different addresses for the block transfers, update the start addresses between the block transfers.
- Element-level address updates. The DMA controller can update the source address and/or the destination address after each element transfer. Ensure that the source address points to the start of the next element, and that the element is precisely positioned at the destination. Choose an addressing mode for the source with the SRC_AMODE bits in DMA_CCR. Choose an addressing mode for the destination with the DST_AMODE bits in DMA_CCR.

4.9.1 Addressing Modes

Figure 23 illustrates the address index management.

Figure 23. Memory Representation



An addressing mode is an address computation algorithm a DMA channel can use to determine where to access data. The system DMA has four addressing modes: constant, post-incremented, indexed, and double-indexed.

The amount of data (block size) to transfer is programmed in bytes. This size can be odd or even. The start address for a transfer is a byte address and can be odd (not word-aligned). The data block to transfer is split into frames and elements.

The byte size of this data block is:

BS = FN x EN x ES

where:

BS is the block size in bytes.

FN is the number of frames in the block, 1 . FB . 65535.

EN is the number of elements per frame, 1 . EN . 65535.

ES is the number of bytes per element, ES . {1, 2, 4}.

An element can be:

- 8-bit scalar data, s8
- 16-bit scalar data, s16
- 32-bit scalar data, s32

s8, s16, and s32 are the types of the data transferred in a channel. FN, EN, and ES (or DATA_TYPE) are extracted from the configuration registers of the channel.

To set up a channel for a transfer, the software must program two addressing modes:

- Source addressing
- Destination addressing

The modes work independently. For example, to transfer data from a TIPB serial port to internal memory, the source-addressing mode is constant (for example, when the read operation must be done at a unique register address) and the destination addressing mode is post-incremented.

The number of frames, the number of elements, and the element size are the same for source and destination. Each of the following algorithms describes address computation for each byte of the transfer.

4.9.2 Constant Addressing Mode

Address remains constant.

a(i) = SA, 0 . i . BS – 1

where:

a(i) is the address of the byte number i within the transfer.

SA is the start address of the transfer.

BS is the block size in bytes.

4.9.3 Post-Incremented Addressing Mode

Address is always incremented by 1.

a(0) =SA

a(i) = a(i – 1) + 1, 1 . i . BS –1

where:

a(i) is the address of the byte number i within the transfer.

SA is the start address of the transfer.

BS is the block size in bytes.

4.9.4 Single-Indexed Addressing Mode

Address is incremented by 1 if the end of the current element is not reached.

Address is incremented by an element index if the end of the current element is reached.

a(0) = SA

a(i) = a(i – 1) + 1 if (i mod ES) = 0, 1 . i . BS – 1

a(i) = a(i – 1) + EI if (i mod ES) = 0, 1 . i . BS – 1

where:

a(i) is the address of the byte number i within the transfer.

SA is the start address of the transfer.

BS is the block size in bytes.

ES is the element size in bytes, 1 . ES . 2.

EI is the element index in bytes, specified in a configuration register, –32768. EI . 32767.

4.9.5 Double-Indexed Addressing Mode

Address is incremented by a frame index if the end of the current frame is reached.

Address is incremented by an element index if the end of the current element is reached and end of frame is not reached.

Address is incremented by one if the end of the current element and the end of current frame are not reached.

a(0) = SA

a(i) = a(i – 1) + 1 if (I mod ES) = 0 and (i mod FS) $\overline{0}$, 1 . i . BS – 1

a(i) = a(i – 1) + EI if (i mod ES) = 0 and (i mod FS) $\overline{0}$, 1 . i . ℜS – 1

a(i) = a(i – 1) + FI if (i mod FS) = 0, 1 . i . BS – 1

where:

a(i) is the address of the byte number i within the transfer.

SA is the start address of the transfer.

BS is the block size in bytes.

ES is the element size in bytes.

EI is the element index in bytes, specified in a configuration register, –32768. EI . 32767.

FS is the frame size in bytes, FS = ES x EN.

FI is the frame index in bytes, specified in a configuration register, –32768. FI . 32767.

4.10 Data Packing

The five DMA controller ports have various widths and support various sizes of data accesses as listed in Table 101.

Table 101. DMA Controller Ports

DMA Port	Port Width	Access Sizes Supported (bytes)
SARAM	32	2, 4
DARAM	32	2, 4
EMIF	32	1, 2, 4
Peripheral	16	2
MPUI	16	2

A DMA channel has the ability to:

- Pack several consecutive element transfers into wider accesses. For example, if the element size is 16 bits, the 32-bit-wide SARAM port can pack two accesses so that 4 bytes at a time are written into the channel FIFO. Packing effectively reduces the frequency at which that channel must be serviced by the port service chain. This can reduce overhead and improve channel throughput in some cases. Packing options are determined by port access capabilities and element size. Packing at the source or destination port can be disabled by software control.

- Split a single word transfer into several byte accesses. This occurs when DMA port size is less than the size of the element type. For example, when the element size is programmed as 32 bits and the destination port is 16 bits wide, the transfer write operation is split into two 16-bit write operations.

The value programmed for channel element size determines the width of the read access at the source port and write access at the destination port. If the element size is smaller than the source port width and source packing is enabled, the source port controller receives 4 bytes per service-chain request. This reduces overhead because four times as much data is written into the channel FIFO per service-chain cycle. Similarly, if packing is enabled at the destination port, 4 bytes are written per iteration of the destination port service chain.

The DMA performs packing as shown in Table 102.

Table 102. DMA Data Packing

Data Type	Port Bus Size	Data Packing
8-bit	16-bit	Two data values packed into 16 bits
8-bit	32-bit	Four data values packed into 32 bits
16-bit	32-bit	Two data values packed into 32 bits

4.11 Bursting

Bursting is an extension of the packing concept. A burst is defined as 16 bytes. When bursting is enabled at the source port, 16 bytes of data at a time are read into the channel FIFO before the source port controller advances to the next position in the service chain. Likewise, when bursting is enabled at the destination port, it accepts 16 bytes of data from the channel FIFO before the service chain can advance to the next position.

Although packing and bursting can improve throughput of a particular channel, they can affect latencies of other active DMA channels.

Example 5. Packing 2 x s16 => 32

- A channel is set up for a transfer with the following parameters for its source:
  - Number of frames in the block: FN = 2
  - Number of elements per frame: EN = 5
  - Type of data is s16

- Frame index in bytes: FI = 13
- Element index in bytes: EI = 1
- Source start address: SA = 2
- The source port is a 32-bit port with byte word16 and word32 access capability.
- Bursts are disabled.

The memory block to transfer is as identified in Table 103 (element i, j is the element number j of frame i).

Table 103. Data Block to Transfer

Address	Byte 0	Byte 1	Byte 2	Byte 3
0			Element 1, 1	
4	Element 1, 2		Element 1, 3	
8	Element 1, 4		Element 1, 5	
12				
16				
20				
24	Element 2, 1		Element 2, 2	
28	Element 2, 3		Element 2, 4	
32	Element 2, 5			
36				
40				

The computed addresses and access types are identified in Table 104.

Table 104. Address and Access Types

Clock Cycle	Frame Number j	Element Number i	Address	Access
0	1	1	2	16 bits
1	1	2	4	32 bits
2	1	4	8	32 bits
3	2	1	24	32 bits
4	2	3	28	32 bits
5	2	5	32	16 bits
6	End of transfer			

4.12 Data Alignment

During a transfer, all the addresses computed by the DMA must be aligned on the type of data transferred:

- If the data type is s8 (8 bits scalar data), addresses can have any value.
- If the data type is s16 (16 bits scalar data), addresses must be aligned on 16-bit word boundary (the least bit of the address is always 0).
- If the data type is s32 (32 bits scalar data), addresses must be aligned on 32-bit word boundary (the two least bits of the address are always 00).

4.13 Synchronizing Channel Activity

Activity in a channel can be synchronized to an event in a DSP peripheral or to an event signaled by the driving of an external interrupt pin. Use the synchronization bits of DMA_CCR to specify which synchronization event (if any) triggers activity.

Each channel also has an FS bit in DMA_CCR that enables choosing between two synchronization modes:

- Element synchronization mode (FS = 0) requires one event per element transfer. When the selected synchronization event occurs, a read access request is sent to the source port, and then a write access request is sent to the destination port. When all the bytes of the current element are transferred, the channel makes no more requests until the next occurrence of the synchronization event.
- Frame synchronization mode (FS = 1) requires one event to trigger an entire frame of elements. When the event occurs, the channel sends a read access request and a write access request for each element in the frame. When all the elements are transferred, the channel makes no more requests until the next occurrence of the event. If a synchronization event is specified, the source port does not receive an access request until the event occurs. Once the request is received, it is handled according to the predefined position and the programmed priority of the channel in the DMA service chain (see Section 4.4, *Service Chain)*. To avoid long delays, it is best to give all synchronized channels a high priority. If the choice is made not to synchronize the channel (synchronization = 00000b), the channel sends an access request to the source port as soon as the channel is enabled (EN = 1 in DMA_CCR).

4.13.1 Read Synchronization vs. Write Synchronization

When a DMA channel is configured for synchronization, the synchronization event is tied to the element read operation or the element write operation, depending on the source and destination ports. There are three general cases (see Table 105):

- Case 1: Source port is peripheral; destination port is SARAM, DARAM, EMIF, or MPUI.

  The channel waits for the synchronization event before reading from the peripheral port into the channel FIFO. Once the FIFO has filled, the DMA channel begins writing to the destination port to empty the FIFO (source synchronization).

- Case 2: Source port is SARAM, DARAM, EMIF, or MPUI; destination port is peripheral.

  As soon as the channel is enabled (EN bit set) a read from SARAM port is performed to feed the channel FIFO. The FIFO writes to the peripheral port to not begin until the synchronization event is detected. When the channel is operating in frame-synchronization mode (DMA_CCR_FS = 1), several prereads can occur to the point of filling the FIFO while the channel is awaiting the synchronization event (destination synchronization).

- Case 3: Source port is SARAM, DARAM, EMIF, or MPUI; destination port is SARAM, DARAM, EMIF, or MPUI.

  The channel waits for the synchronization event before reading from the source port into the channel FIFO. Once the FIFO has filled, the DMA channel begins writing to the destination port to empty the FIFO (source synchronization).

Table 105. Read/Write Synchronization

Channel Synchronization Set by DMA_CCR sync[4:0] Not Equal to 00000	Source Port	Destination Port	Synchronization Event Triggers
No	X	X	No synchronization
Yes	Peripheral port	SARAM,DARAM, EMIF, or MPUI port	Source read
Yes	SARAM, DARAM, EMIF, or MPUI port	Peripheral port	Destination write
Yes	SARAM,DARAM, EMIF, or MPUI port	SARAM,DARAM, EMIF, or MPUI port	Source read

4.14 Checking the Synchronization Status

Each channel has a synchronization flag (synchronization) in its status register, DMA_CSR. When the synchronization event occurs, the DMA controller sets the flag (SYNCHRONIZATION = 1). The flag is cleared (SYNCHRONIZATION = 0) when the DMA controller has completed the first read access (transfer from source port to channel buffer) after receiving synchronization.

4.15 Dropped Synchronization Events

If synchronization events occur before the DMA controller is done servicing a currently active one (before the DMA controller clears the synchronization bit in DMA_CSR), the second new synchronization event is dropped. The DMA controller responds to an event drop as follows:

- After the current element transfer, the DMA controller disables the channel (EN = 0 in DMA_CCR); activity in the channel stops after the current element transfer.
- If the corresponding interrupt enable bit is set (DROP_IE = 1 in DMA_CICR), the DMA controller also sets the event drop status bit (DROP = 1 in DMA_CSR) and sends an interrupt request to the CPU.

4.16 Monitoring Channel Activity

The DMA controller can send an interrupt to the CPU in response to the operational events listed in Table 106. Each channel has interrupt enable (IE) bits in the interrupt control register (DMA_CICR) and some corresponding status bits in the status register (DMA_CSR). If one of the operational events in the table occurs, the DMA controller checks the corresponding IE bit and acts accordingly:

- If the IE bit is 1 (the interrupt is enabled), the DMA controller sets the corresponding status bit and sends the associated interrupt request to the CPU. DMA_CSR is automatically cleared if the program reads the register.
- If the IE bit is 0, no interrupt is sent and the status bit is not affected.

DMA_CSR also has a synchronization bit that is used when a synchronization event is chosen for the channel. This bit indicates when the selected synchronization event has occurred (SYNCHRONIZATION = 1) and when it has been serviced (SYNCHRONIZATION = 0). For details, see Section 4.13, *Synchronizing Channel Activity*.

All interrupts generated by the DSP DMA controller are level-sensitive interrupts, that is, the interrupt line is held active low for two DSP clock cycles after the CPU reads the associated channel status register (see Table 106).

Table 106. DMA Controller Operational Events and Associated Bits/Interrupts

Operational Event	Interrupt Enable Bit	Status Bit	Associated Interrupt
Block transfer is complete.	BLOCK_IE	BLOCK	Channel interrupt
Last frame transfer has started.	LAST_IE	LAST	Channel interrupt
Frame transfer is complete.	FRAME_IE	FRAME	Channel interrupt
First half of current frame has been transferred.	HALF_IE	HALF	Channel interrupt
Synchronization event has been dropped.	DROP_IE	DROP	Channel interrupt
Time-out error has occurred.	TIMEOUT_IE	TIMEOUT	Bus-error interrupt

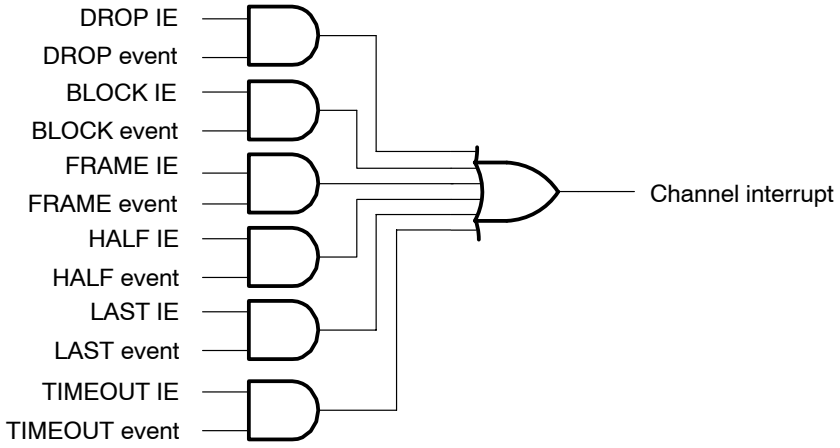
4.17 Channel Interrupt

Each of the six channels has its own interrupt. As shown in Figure 24, the channel interrupt is the logical OR of all the enabled operational events except the time-out event (the time-out event generates a bus-error interrupt request). Choose any combination of these five events by setting or clearing the appropriate interrupt enable (IE) bits in the interrupt control register (DMA_CICR) for the channel. Determine which event(s) caused the interrupt by reading the bits in the status register (DMA_CSR) for the channel.

Note:

The DMA interrupt status bits are set by hardware and cleared by a software read operation to DMA_CSR. A subsequent DMA interrupt cannot be issued until a program read of DMA_CSR has cleared the interrupt status bits. For the ongoing operation of the DMA channel, the ISR must read the DMA_CSR after each DMA interrupt.

Figure 24. Triggering a Channel Interrupt Request



For example, suppose you are monitoring activity in channel 1. In DMA_CICR:

- ❏ DROP_IE = 1
- ❏ HALF_IE = 0
- ❏ FRAME_IE = 1
- ❏ LAST_IE = 0
- ❏ BLOCK_IE = 0

If a synchronization event is dropped or if the current frame transfer is done, the channel 1 interrupt request is sent to the CPU. No other event can generate the channel 1 interrupt. To determine whether one or both of the events triggered the interrupt, read the drop and frame bits in DMA_CSR.

The channel 1 interrupt sets its corresponding flag bit in an interrupt flag register of the CPU. The CPU can respond to the interrupt or ignore the interrupt.

4.18 Time-Out Conditions

A time-out condition exists when a memory access has been stalled for too many cycles**.** Each of the four standard ports of the DMA controller is supported by hardware to detect a time-out condition:

- ❏ DARAM port: A time-out counter in the DARAM port keeps track of how many cycles have passed since a request was made to access the DARAM. When the counter reaches 255 DSP clock cycles, the DARAM port generates a time-out signal.
- ❏ SARAM port: A time-out counter in the SARAM port keeps track of how many cycles have passed since a request was made to access the SARAM. When the counter reaches 255 DSP clock cycles, the SARAM port generates a time-out signal.

- External memory port: A time-out counter in the external memory interface (EMIF) keeps track of how many cycles the external ready pin has been sampled low. The external memory map is divided into four memory spaces, each of which has a programmable time-out value up to 255 DSP cycles. When the counter reaches the time-out value, the EMIF sends a time-out signal to the DMA controller.
- Peripherals port: A time-out counter in the peripheral bus controller counts how many cycles have passed since a request was made to access a peripheral. When the counter reaches 127 DSP cycles, the peripheral bus controller sends a time-out signal to the DMA controller. The default time-out values can be changed in the TIPB CMR register.

In response to a time-out signal, the DMA controller disables the channel (EN = 0 in DMA_CCR); activity in the channel stops after the current element transfer. If the corresponding interrupt enable bit is set (TOUT_IE = 1 in DMA_CICR), the DMA controller also sets the time-out status bit (TOUT = 1 in DMA_CSR) and sends the time-out signal to the CPU as an interrupt request. The interrupt request sets the bus-error interrupt flag bit in the CPU. The CPU can respond to the interrupt request or ignore the interrupt request.

4.19 DMA Transfer Latency

Each element transfer in a channel contains a read access (a transfer from the source location to the channel buffer) and a write access (a transfer from the channel buffer to the destination location). The time to complete this activity depends on factors such as:

- The selected frequency of the CPU clock signal. This signal, as propagated to the DMA controller, determines the timing for all DMA transfers.
- Wait states or other extra cycles added by or resulting from an interface.
- Competition from other channels. The DMA controller divides cycles among all enabled channels according to their position and priority level in the service chain (see Section 4.4, *Service Chain*). If fewer channels are enabled, more cycles are allotted per channel during a given interval of time.
- Competition from the MPU access via the MPUI. If the MPUI is sharing internal RAM with the channels, the DMA controller allocates cycles to the MPUI like it does to channels. If the MPUI is given exclusive access to the internal RAM, no channels can access the internal RAM until the MPUI access configuration is changed (see Section 4.3, *MPUI Access Configurations*).

- The timing of synchronization events (if the channel is synchronized). The DMA controller cannot service a synchronized channel until the synchronization event has occurred. For details, see Section 4.13, *Synchronizing Channel Activity*.

4.20 DMA Power Reduction

The DSP is divided into idle domains that can be programmed to be idle or active. The state of all five idle domains is called the idle configuration. Any idle configuration that disables the clock generator domain and/or the DMA domain stops the DMA clock and, therefore, stops activity in the DMA controller. For more details, see *Multimedia Processor OMAP3.2 Subsystem Reference Guide (SPRU749)* and *Multimedia Processor Power Management Reference Guide (SPRU753)*.

4.21 Emulation Modes

The FREE bit of DMA_GCR controls the behavior of the DMA controller when a breakpoint is encountered in the debugger software. If FREE = 0 (the reset value), a breakpoint suspends DMA transfers. If FREE = 1, DMA transfers are not interrupted by a breakpoint.

4.22 DMA Controller Configuration Registers

Table 107 lists the DMA controller configuration registers. Table 108 through Table 127 describe the register bits.

MPU Base Address(byte): 0xE100 3000 (note, Table 107 lists word offsets)

DSP Base Address (word): 0x00 1800

Table 107. DMA Controller Configuration Registers

Register	Description	Word Address
DMA_GCR	Global control	0E00h
DMA_GTCR	Global time-out control	0E01h
DMA_GSCR	Global software incompatible control	0E02h
	Channel 0	
DMA_CSDP0	Channel 0 source destination parameters	0C00h
DMA_CCR0	Channel 0 control	0C01h
DMA_CICR0	Channel 0 interrupt control	0C02h

Table 107. DMA Controller Configuration Registers (Continued)

Register	Description	Word Address
	Channel 0 (Continued)	
DMA_CSR0	Channel 0 status	0C03h
DMA_CSSA_L0	Channel 0 source start address, lower bits	0C04h
DMA_CSSA_U0	Channel 0 source start address, upper bits	0C05h
DMA_CDSA_L0	Channel 0 destination start address, lower bits	0C06h
DMA_CDSA_U0	Channel 0 destination start address, upper bits	0C07h
DMA_CEN0	Channel 0 element number	0C08h
DMA_CFN0	Channel 0 frame number	0C09h
DMA_CSFI0	Channel 0 source frame index	0C0Ah
DMA_CSEI0	Channel 0 source element index	0C0Bh
DMA_CSAC0	Channel 0 source address counter	0C0Ch
DMA_CDAC0	Channel 0 destination address counter	0C0Dh
DMA_CDEI0	Channel 0 destination element index	0C0Eh
DMA_CDFI0	Channel 0 destination frame index	0C0Fh
	Channel 1	
DMA_CSDP1	Channel 1 source destination parameters	0C20h
DMA_CCR1	Channel 1 control	0C21h
DMA_CICR1	Channel 1 interrupt control	0C22h
DMA_CSR1	Channel 1 status	0C23h
DMA_CSSA_L1	Channel 1 source start address, lower bits	0C24h
DMA_CSSA_U1	Channel 1 source start address, upper bits	0C25h
DMA_CDSA_L1	Channel 1 destination start address, lower bits	0C26h
DMA_CDSA_U1	Channel 1 destination start address, upper bits	0C27h
DMA_CEN1	Channel 1 element number	0C28h
DMA_CFN1	Channel 1 frame number	0C29h
DMA_CSFI1	Channel 1 source frame index	0C2Ah

Table 107. DMA Controller Configuration Registers (Continued)

Register	Description	Word Address
	Channel 1 (Continued)	
DMA_CSEI1	Channel 1 source element index	0C2Bh
DMA_CSAC1	Channel 1 source address counter	0C2Ch
DMA_CDAC1	Channel 1 destination address counter	0C2Dh
DMA_CDEI1	Channel 1 destination element index	0C2Eh
DMA_CDFI1	Channel 1 destination frame index	0C2Fh
	Channel 2	
DMA_CSDP2	Channel 2 source destination parameters	0C40h
DMA_CCR2	Channel 2 control	0C41h
DMA_CICR2	Channel 2 interrupt control	0C42h
DMA_CSR2	Channel 2 status	0C43h
DMA_CSSA_L2	Channel 2 source start address, lower bits	0C44h
DMA_CSSA_U2	Channel 2 source start address, upper bits	0C45h
DMA_CDSA_L2	Channel 2 destination start address, lower bits	0C46h
DMA_CDSA_U2	Channel 2 destination start address, upper bits	0C47h
DMA_CEN2	Channel 2 element number	0C48h
DMA_CFN2	Channel 2 frame number	0C49h
DMA_CSFI2	Channel 2 frame index	0C4Ah
DMA_CSEI2	Channel 2 element index	0C4Bh
DMA_CSAC2	Channel 2 source address counter	0C4Ch
DMA_CDAC2	Channel 2 destination address counter	0C4Dh
DMA_CDEI2	Channel 2 destination element index	0C4Eh
DMA_CDFI2	Channel 2 destination frame index	0C4Fh
	Channel 3	
DMA_CSDP3	Channel 3 source destination parameters	0C60h
DMA_CCR3	Channel 3 control	0C61h

Table 107. DMA Controller Configuration Registers (Continued)

Register	Description	Word Address
	Channel 3 (Continued)	
DMA_CICR3	Channel 3 interrupt control	0C62h
DMA_CSR3	Channel 3 status	0C63h
DMA_CSSA_L3	Channel 3 source start address, lower bits	0C64h
DMA_CSSA_U3	Channel 3 source start address, upper bits	0C65h
DMA_CDSA_L3	Channel 3 destination start address, lower bits	0C66h
DMA_CDSA_U3	Channel 3 destination start address, upper bits	0C67h
DMA_CEN3	Channel 3 element number	0C68h
DMA_CFN3	Channel 3 frame number	0C69h
DMA_CSFI3	Channel 3 source frame index	0C6Ah
DMA_CSEI3	Channel 3 source element index	0C6Bh
DMA_CSAC3	Channel 3 source address counter	0C6Ch
DMA_CDAC3	Channel 3 destination address counter	0C6Dh
DMA_CDEI3	Channel 3 destination element index	0C6Eh
DMA_CDFI3	Channel 3 destination frame index	0C6Fh
	Channel 4	
DMA_CSDP4	Channel 4 source destination parameters	0C80h
DMA_CCR4	Channel 4 control	0C81h
DMA_CICR4	Channel 4 interrupt control	0C82h
DMA_CSR4	Channel 4 status	0C83h
DMA_CSSA_L4	Channel 4 source start address, lower bits	0C84h
DMA_CSSA_U4	Channel 4 source start address, upper bits	0C85h
DMA_CDSA_L4	Channel 4 destination start address, lower bits	0C86h
DMA_CDSA_U4	Channel 4 destination start address, upper bits	0C87h
DMA_CEN4	Channel 4 element number	0C88h
DMA_CFN4	Channel 4 frame number	0C89h

Table 107. DMA Controller Configuration Registers (Continued)

Register	Description	Word Address
	Channel 4 (Continued)	
DMA_CSFI4	Channel 4 source frame index	0C8Ah
DMA_CSEI4	Channel 4 source element index	0C8Bh
DMA_CSAC4	Channel 4 source address counter	0C8Ch
DMA_CDAC4	Channel 4 destination address counter	0C8Dh
DMA_CDEI4	Channel 4 destination element index	0C8Eh
DMA_CDFI4	Channel 4 destination frame index	0C8Fh
	Channel 5	
DMA_CSDP5	Channel 5 source destination parameters	0CA0h
DMA_CCR5	Channel 5 control	0CA1h
DMA_CICR5	Channel 5 interrupt control	0CA2h
DMA_CSR5	Channel 5 status	0CA3h
DMA_CSSA_L5	Channel 5 source start address, lower bits	0CA4h
DMA_CSSA_U5	Channel 5 source start address, upper bits	0CA5h
DMA_CDSA_L5	Channel 5 destination start address, lower bits	0CA6h
DMA_CDSA_U5	Channel 5 destination start address, upper bits	0CA7h
DMA_CEN5	Channel 5 element number	0CA8h
DMA_CFN5	Channel 5 frame number	0CA9h
DMA_CSFI5	Channel 5 frame index	0CAAh
DMA_CSEI5	Channel 5 element index	0CABh
DMA_CSAC5	Channel 5 source address counter	0CACh
DMA_CDAC5	Channel 5 destination address counter	0CADh
DMA_CDEI5	Channel 5 destination element index	0CAEh
DMA_CDFI5	Channel 5 destination frame index	0CAFh

Table 108. Global Control Register (DMA_GCR)

Bit	Name	Function	Type	Reset
31:4	Reserved			
3	AUTOGATING_ON	DMA autoidle bit. Controls whether DMA circuits are automatically idled when DMA is inactive. 0: DMA clocks are free running. 1: Allows the DMA clocks to autoidle when inactive.	RW	1
2	FREE	Emulation mode bit FREE controls the behavior of the DMA controller when an emulation breakpoint is encountered. 0: A breakpoint suspends DMA transfers. 1: DMA transfers continue uninterrupted on a breakpoint.	RW	0
1	MPUI_EXCL	MPUI exclusive access bit. MPUI_EXCL determines whether the MPUI has exclusive access to the internal SARAM and DARAM of the DSP subsystem. 0: MPUI shares DARAM and SARAM ports with other DMA. In this mode, the MPUI can access all internal memory, external memory, or peripherals. 1: MPUI channel has exclusive access to internal RAM. If any other DMA channel attempts to access the DARAM or the SARAM port, activity in that channel suspends. In this configuration, the MPUI can access only the DARAM port and SARAM port. It cannot access the external memory port. **Note:** When MPUI has exclusive access to the DARAM and SARAM ports (MPUI_EXCL =1), the MPUI priority is irrelevant at these ports because none of the DMA channels can access the DARAM and SARAM ports.	RW	0
0	MPUI_PRIO	MPUI priority bit. Assigns the MPUI a high or low priority in the service chain of the DMA controller. 0: MPUI channel has low priority. 1: MPUI channel has high priority.	RW	0

Table 109. Global Time-Out Control Register (DMA_GTCR)

Bit	Name	Function	Type	Reset
31:2	NC			
1	DARAM_TE	DARAM time-out enable 0: DARAM port time-out counter is disabled. 1: DARAM port time-out counter is enabled.	RW	0
0	SARAM_TE	SARAM time-out enable 0: SARAM port time-out counter is disabled. 1: SARAM port time-out counter is enabled.	RW	0

Table 110. Global Software Incompatible Control Register (DMA_GSCR)

Bit	Name	Function	Type	Reset
31:1	NC			
0	DMA_ENHANCED_ INDEXING	DMA destination indexing enhancement: Software-incompatible control. 0: Destination element index = DMA_CSEI register, destination frame index = DMA_CSFI register This mode permits software compatibility with previous versions of the DMA controller hardware. The source indexing registers are also used for destination indexing. This is the power1-up default state. 1: Destination element index = DMA_CDEI register, destination frame index = DMA_CDFI register This mode of operation has separate destination element and frame indexes, which are configured by the channel-destination element index and the channel-destination frame index, respectively.	RW	0

4.23 DMA Channel Configuration Registers

The register bit descriptions in Table 111 through Table 125 are generic in that the register sets for each of the six DMA channels (0-5) are identical.

Table 111. Channel Source Destination Parameters Registers (DMA_CSDP0...DMA_CSDP5)

Bit	Name	Function	Type	Reset
31:16	NC			
15:14	DST_BURST_EN	Destination burst enable A burst in the DMA controller refers to four consecutive 32-bit accesses at a DMA port. DST_BURST_EN determines whether the DMA controller performs a burst at the destination port of the channel. 00: Single access 01: Single access 10: Burst 16 bytes 11: Illegal If the destination port of the channel has no burst access capability, this field is ignored.	RW	00
13	DST_PACK	Destination packing The DMA controller can perform data packing to double or quadruple the amount of data passed to the destination. For example, if an 8-bit data type is selected and the destination port has a 32-bit data bus, four 8-bit pieces of data can be packed into 32 bits before being sent to the destination. The packing is performed in respect to endianism. 0: The destination port never makes packed accesses. 1: The destination port makes packed accesses when possible.	RW	0
12:9	DST	Transfer destination A unique identifier is given to each port. This field indicates which port is the destination of the channel transfer operation. 0000: SARAM 0001: DARAM 0010: EMIF 0011: TIPB 0100: MPUI Others : Illegal	RW	0000

Table 111. Channel Source Destination Parameters Registers (DMA_CSDP0...DMA_CSDP5) (Continued)

Bit	Name	Function	Type	Reset
8:7	SRC_BURST_EN	Source burst enable A burst in the DMA controller refers to four consecutive 32-bit accesses at a DMA port. SRC_BURST_EN determines whether the DMA controller performs a burst transfer at the source port of the channel. 00: Single access 01: Single access 10: Burst 16 bytes 11: Illegal	RW	00
6	SRC_PACK	Source packing The DMA controller can perform data packing to double or quadruple the amount of data passed to the destination. For example, if an 8-bit data type is selected and the source port has a 32-bit data bus, four 8-bit pieces of data can be packed into 32 bits before being sent through the channel. The packing is made with respect to endianism. 0: The source port never makes packed accesses. 1: The source port makes packed accesses when possible.	RW	0
5:2	SRC	Transfer source A unique identifier is given to each port. This field indicates which port is the source of the channel transfer operation. 0000: SARAM 0001: DARAM 0010: EMIF 0011: TIPB 0100: MPUI Others: Illegal	RW	0000
1:0	DATA_TYPE	For details, see Section 4.23.1.	RW	00

4.23.1 DATA_TYPE Bit

- DATA_TYPE: Defines the type of the data moved in the channel; used for endianism adaptation.
  - 00b 8-bit

    The DMA controller makes 8-bit accesses at the source and at the destination of the channel. The source and destination start addresses have no alignment constraint, as follows:

    Start address: XXXX XXXX XXXX XXXXb (X can be 0 or 1). If the automatic post-incremented addressing mode is chosen at the source or the destination, the corresponding address is updated by an increment of 1 after each element transfer.
  - 01b 16-bit

    The DMA controller makes 16-bit accesses at the source and at the destination. The source and destination start addresses must each be on an even 2-byte boundary; the least significant bit (LSB) must be 0 as follows:

    Start address: XXXX XXXX XXXX XXX0b (X can be 0 or 1). If the automatic post-incremented addressing mode is chosen at the source or the destination, the address is updated by an increment of 2 after each element transfer.
  - 10b 32-bit

    The DMA controller makes 32-bit accesses at the source and at the destination. The source and destination start addresses must be on an even 4-byte boundary; the 2 LSBs must be 0 as follows:

    Start address: XXXX XXXX XXXX XX00b (X can be 0 or 1). If the automatic post-incremented addressing mode is chosen at the source or the destination, the address is updated by an increment of 4 after each element transfer.
  - 11b Reserved (do not use)

Programmers Responsibility

It is the responsibility of the programmer to ensure that the start address is aligned with the channel DATA_TYPE.

Table 112. Channel Control Registers (DMA_CCR0...DMA_CCR5)

Bit	Name	Function	Type	Reset
31:16	Reserved			
15:14	DST_AMODE	Destination addressing mode This field determines the addressing mode used by the DMA controller when it writes to the source port of the channel. 00: Constant address. The same address is used for each element transfer. 01: Automatic post-incremented. After each element transfer, the address is incremented according to the selected data type: ❏ 8-bit data type address = address+1 ❏ 16-bit data type address = address+2 ❏ 32-bit data type address = address+4 10: Single index (element index). After each element transfer, the address is incremented by the programmed element index register contents. 11: Single index (element index). After each element transfer, the address is incremented by the programmed element index register contents. Address = address + element index (DMA_CSEI if DMA_GSCR-bit0 = 0, DMA_CDEI if DMA_GSCR-bit0 = 1) Double index (element index and frame index). After each element transfer, the address is incremented by the appropriate index amount: If there are more elements to transfer in the current frame, then: address = address + element index (DMA_CSEI if DMA_GSCR-bit0 = 0, DMA_CDEI if DMA_GSCR-bit0 = 1) If the last element in the frame has been transferred, then: address = address + frame index (DMA_CSFI if DMA_GSCR-bit0 = 0, DMA_CDFI if DMA_GSCR-bit0 = 1)	00	RW

Table 112. Channel Control Registers (DMA_CCR0...DMA_CCR5) (Continued)

Bit	Name	Function	Type	Reset
13:12	SRC_AMODE	Source addressing mode This field determines the addressing mode used by the DMA controller when it reads from the source port of the channel. 00: Constant address. The same address is used for each element transfer. 01: Automatic post-incremented. After each element transfer, the address is incremented according to the selected data type: ❑ 8-bit data type address = address+1 ❑ 16-bit data type address = address+2 ❑ 32-bit data type address = address+4 10: Single index (element index). After each element transfer, the address is incremented by the programmed element index register contents. Address = address + element index (DMA_CSEI) 11: Double index (element index and frame index). After each element transfer, the address is incremented by the appropriate index amount: If there are more elements to transfer in the current frame, then: address = address + element index (DMA_CSEI) If the last element in the frame has been transferred, then: address = address + frame index (DMA_CSFI)	00	RW

Table 112. Channel Control Registers (DMA_CCR0...DMA_CCR5) (Continued)

Bit	Name	Function	Type	Reset
11	END_PROG	End of programming bit. Each DMA channel has two sets of registers: configuration registers and working registers. When block transfers occur repeatedly because of autoinitialization (AUTO_INIT = 1), change the context for the next DMA transfer by writing to the configuration registers during the current block transfer. At the end of the current transfer, the contents of the configuration registers are copied into the working registers and the DMA controller begins the next transfer using the new context. For proper autoinitialization, the CPU must finish programming the configuration registers before the DMA controller copies their contents. To ensure that autoinitialization waits for the CPU, follow this procedure: 1) Make autoinitialization wait for END_PROG = 1 by clearing the repeat bit (REPEAT = 0). 2) Clear END_PROG (END_PROG = 0) to indicate that programming of the configuration registers is in progress. 3) Program the configuration registers. 4) Set END_PROG (END_PROG = 1) to indicate the end of programming. 0: When DMA channel is in autoinitialization mode and REPEAT bit is 0, DMA does not reinitialize the channel until END_PROG bit is set to 1. 1: When DMA channel is in autoinitialization mode and END_PROG = 1, the DMA reinitializes after the current DMA channel transfer completes.	0	RW
10	NC			

Table 112. Channel Control Registers (DMA_CCR0...DMA_CCR5) (Continued)

Bit	Name	Function	Type	Reset
9	REPEAT	Repeat condition bit. If autoinitialization is selected for a channel (AUTO_INIT = 1), REPEAT specifies one of two special repeat conditions: 0: Repeat only if END_PROG = 1. Once the current DMA transfer is complete, autoinitialization occurs only if the end of the programming bit is set (END_PROG = 1). 1: –init regardless of END_PROG. Once the current DMA transfer is complete, autoinitialization occurs regardless of whether END_PROG is 0 or 1.	0	RW
8	AUTO_INIT	Autoinitialization bit. The DMA controller supports autoinitialization, which is the automatic reinitialization of the channel between DMA transfers. Use autoinitialization to enable or disable this feature. 0: DMA autoinitialization is disabled. Activity in the channel stops at the end of the current block transfer. To stop a transfer immediately, clear the channel enable bit (EN). 1: DMA autoinitialization is enabled. Once the current transfer is complete, the DMA controller reinitializes the channel and starts a new block transfer. There are two options for stopping activity in the channel: ❑ To stop activity immediately, clear the channel enable bit (EN=0). ❑ To stop activity after the current block transfer, clear AUTO_INIT (AUTO_INIT=0). **Note:** The autoinitialization operation is also affected by setting the REPEAT and END_PROG bits. See the following descriptions.	0	RW

Table 112. Channel Control Registers (DMA_CCR0...DMA_CCR5) (Continued)

Bit	Name	Function	Type	Reset
7	EN	Channel enable bit Use EN to enable or disable transfers in the channel. The DMA controller clears EN once a block transfer in the channel is complete. 0: Channel is disabled. The channel cannot be serviced by the DMA controller. If a DMA transfer is already active in the channel , the DMA controller stops the transfer and resets the channel. 1: Channel is enabled. The channel can be serviced by the DMA controller at the next available time slot. Clearing of this bit by the DMA because of block completion has priority over a write by the DSP. If both occur simultaneously**,** the DSP write is discarded. **Note:** Under certain conditions the DMA controller can asynchronously reset the EN bit relative to the DSP clock. Therefore, a metastable condition can result if the DSP reads the bit coincidentally with the DMA clear. In this case the DSP might perceive the EN bit to be set for an extra polling period.	0	RW
6	PRIO	Channel priority bit All six DMA channels are given a fixed position and programmable priority level on the service chain of the DMA controller. PRIO determines whether the associated channel has a high priority or a low priority. High-priority channels are serviced before low-priority channels. 0: The channel has the low-priority level. 1: The channel has the high-priority level.	0	RW

Table 112. Channel Control Registers (DMA_CCR0...DMA_CCR5) (Continued)

Bit	Name	Function	Type	Reset
5	FS	Frame synchronization This bit determines whether the synchronization event initiates the transfer of an element or an entire frame of data. 0: When the selected synchronization event occurs, one element is transferred in the channel. Each element waits for the synchronization event. 1: When the selected synchronization event occurs, an entire frame is transferred in the channel. Each frame waits for the synchronization event.	0	RW
4:0	SYNC	Synchronization control This field is used to specify which event in the DSP (for example, timer1 countdown) initiates a DMA transfer in this channel. There are 20 possible choices, and multiple channels can share the same synchronization event. That is, one event can initiate the transfer in multiple DMA channels. 0000: Transfer not synchronized i: Transfer synchronized on DMA_REQUEST[i]. The behavior of DMA synchronized transfers is described in Section 4.13, *Synchronizing Channel Activity*. Table 113 shows the DSP DMA mapping.	00000	RW

Table 113. DSP DMA Mapping

DSP Request Source	DSP DMA Request Line	Synchronization [4:0] Settings
MCSI1 TX	DMA_REQ_01	00001
MCSI1 RX	DMA_REQ_02	00010
MCSI2 TX	DMA_REQ_03	00011
MCSI2 RX	DMA_REQ_04	00100
Ext_nDMA_req_0 (MPUIO2)	DMA_REQ_05	00101
Ext_nDMA_req_1 (MPUIO4)	DMA_REQ_06	00110
FREE	DMA_REQ_07	00111
McBSP1 TX	DMA_REQ_08	01000
McBSP1 RX	DMA_REQ_09	01001
McBSP3 TX	DMA_REQ_10	01010
McBSP3 RX	DMA_REQ_011	01011
UART1.TX	DMA_REQ_012	01100
UART1.RX	DMA_REQ_013	01101
UART2.TX	DMA_REQ_014	01110
UART2.RX	DMA_REQ_015	01111
FREE	DMA_REQ_016	10000
FREE	DMA_REQ_017	10001
UART3.TX	DMA_REQ_018	10010
UART3.RX	DMA_REQ_019	10011

Table 114. Channel Interrupt Control Registers (DMA_CICR0...DMA_CICR5)

Bit	Name	Function	Type	Reset
31:6	Reserved			
5	BLOCK_IE	End block interrupt enable. BLOCK_IE determines how the DMA controller responds when all of the current block has been transferred from the source port to the destination port. 0: Do not record the end of block event. 1: Set the block bit in DMA_CSR and send channel interrupt to the DSP CPU.	RW	0

Table 114. Channel Interrupt Control Registers (DMA_CICR0...DMA_CICR5) (Continued)

Bit	Name	Function	Type	Reset
4	LAST_IE	Last frame interrupt enable. LAST_IE determines how the DMA controller responds when it starts transferring the last frame from the source port to the destination port. 0: Do not record the last frame event. 1: Set the last bit in DMA_CSR and send channel interrupt request to the DSP CPU.	RW	0
3	FRAME_IE	Whole frame interrupt enable. FRAME_IE determines how the DMA controller responds when the first half of the current frame has been transferred from the source port to the destination port. 0: Do not record the frame complete event. 1: Set the frame bit in DMA_CSR and send channel interrupt request to the DSP.	RW	0
2	HALF_IE	Half frame interrupt enable. Determines how the DMA controller responds when the first half of the current frame has been transferred from the source port to the destination port: 0: Do not record the half frame event. 1: Set the half bit in DMA_CSR and send the channel interrupt request to the DSP CPU.	RW	0
1	DROP_IE	Synchronization event drop interrupt enable If two DMA synchronization event occurs again before the DMA controller is done servicing the previous DMA request, an error has occurred (a synchronization event drop). DROP_IE determines how the DMA controller responds when a synchronization event drop occurs in the channel: 0: Do not record the event drop. 1: Set the drop bit in DMA_CSR and send the channel interrupt request to the DSP CPU.	RW	1
0	TOUT_IE	Time-out interrupt enable. Determines how the DMA controller responds to a time-out error at the source port or the destination port of the channel. 0: Do not record the the time-out error. 1: Set the time-out bit in DMA_CSR and send the bus error interrupt request to the DSP CPU.	RW	1

Each channel has an interrupt control register DMA_CICR that specifies one or more DMA controller events triggering an interrupt. If an event occurs and its interrupt enable (IE) bit is 1, an interrupt request is sent to the DSP CPU where it can be serviced or ignored. Each channel has its own interrupt line to the CPU and one set of flags and enable bits in the CPU.

There are two classes of events that can generate interrupts:

- Status events: For example, new frame start or end of block
- Error events: For example, time-out condition or event drop

The bus-error interrupt also has a set of flag and enable bits in the CPU.

Each time an event occurs, if the corresponding interrupt enable bit is set, the channel sends an interrupt to the processor. Simultaneously, the corresponding status bit is set in DMA_CSR (DMA channel status register).

Note:

A status bit in DMA_CSR is not set if the corresponding interrupt enable bit in the DMA_CICR equals 0.

Table 115. Channel Status Registers (DMA_CSR0...DMA_CSR5)

Bit	Name	Function	Type	Reset
31:7	Reserved			
6	SYNC	Synchronization event status bit. The DMA controller updates synchronization to indicate when the synchronization event for the channel has occurred and when the synchronized channel has been serviced: 0: The DMA controller has finished servicing the previous access request. 1: The synchronization event has occurred. Channel is waiting for synchronized DMA request to be scheduled. In response to the event, the synchronized channel submits an access request to its source port. If a synchronization event occurs again before the DMA controller finishes servicing the previous DMA request, an error has occurred and the DMA controller disables the channel (see Section 4.15, *Dropped Synchronization Event*). You can track this type of error using the DROP_IE bit and the DROP bit. To select a synchronization event for a channel, use the synchronization bits of DMA_CCR.	R	0

Table 115. Channel Status Registers (DMA_CSR0...DMA_CSR5) (Continued)

Bit	Name	Function	Type	Reset
5	BLOCK	Whole block status bit. The DMA controller sets BLOCK only if BLOCK_IE=1 in the DMA_CICR and all of the current block has been transferred from the source port to the destination port. 0: Current block transfer has not finished yet. 1: The whole block has been transferred. A channel interrupt has been sent to the DSP CPU (another one may have started, if DMA_CCR2.AUTOINIT = 1).	R	0
4	LAST	Last frame status bit. The DMA controller sets LAST only if LAST_IE = 1 in the DMA_CICR and the DMA controller has started transferring the last frame from the source port to the destination port. 0: Last frame has not started yet. 1: The DMA has started transferring the last frame. A channel interrupt has been sent to the DSP CPU.	R	0
3	FRAME	Whole frame status bit. The DMA controller sets FRAME only if FRAME_IE = 1 in DMA_CICR and the current frame has been transferred from the source port to the destination port. 0: Transfer of the current frame is still in progress. 1: A complete frame has been transferred. A channel interrupt has been sent to the DSP CPU.	R	0
2	HALF	Half frame status bit. The DMA controller sets HALF only if HALF_IE = 1 in DMA_CICR and the first half of the current frame has been transferred from the source port to the destination port: 0: First half of the current frame has not finished transferring yet. 1: First half of the current frame has been transferred. A channel interrupt request has been sent to the DSP CPU.	R	0

Table 115. Channel Status Registers (DMA_CSR0...DMA_CSR5) (Continued)

Bit	Name	Function	Type	Reset
1	DROP	Synchronization event drop status bit. If a DMA synchronization event occurs again before the DMA controller is done servicing the previous DMA request, an error has occurred (a synchronization event drop). The DMA controller sets DROP only if DROP_IE = 1 in DMA_CICR and a synchronization event drop has occurred in the channel. 0: No event drop occurred or the the DROP bit has been cleared. 1: A synchronization event drop has occurred. A channel interrupt request has been sent to the CPU.	R	0
0	TOUT	Time-out status bit. The DMA controller sets TOUT only if TOUT_IE = 1 in DMA_CICR and time-out error has occurred at the source port or the destination port of the channel: 0: A time-out error has not occurred, or TOUT has been cleared. 1: A time-out error has occurred. A bus-error interrupt request has been sent to the DSP CPU.	R	0

This register is written by the DMA controller to reflect the channel status. It can be read by the processor to monitor which events have occurred. All status bits other than SYNC must be individually enabled in DMA_CICR for their status to be registered. The SYNC bit is always set after its synchronization event occurs. After a program read of DMA_CSR, all bits are automatically cleared within two DSP clock cycles. The bits are not cleared if read via hardware emulator/debugger.

4.23.2 Channel Source Start Address

Each channel has two source start address registers: DMA_CSSA_L and DMA_CSSA_U. For the first access to the source port of the channel, the DMA controller generates a byte address by concatenating the contents of the two I/O mapped registers.

Source address DMA_CSSA_U:DMA_CSSA_L

Note:

Load the source start address registers with a byte address. For a word address, shift left by 1 before loading the registers.

For a 16-bit or 32-bit data type, the start address must be aligned properly. See the description of the data type bits of DMA_CSDP.

Table 116. Channel Source Start Address, Lower Bits Registers (DMA_CSSA_L0...DMA_CSSA_L5)

Bit	Name	Function	Type	Reset
31:16	Reserved			
15:0	Source start address, lower bits	Lower bits of the source start address are expressed in bytes. The source start address generated by the DMA is up to a 32-bit byte address, made of the concatenation of DMA_CSSA_U and DMA_CSSA_L.	RW	Undefined

Table 117. Channel Source Start Address, Upper Bits Registers (DMA_CSSA_U0...DMA_CSSA_U5)

Bit	Name	Function	Type	Reset
31:16	Reserved			
15:0	Source start address, upper bits	Upper bits of the source start address are expressed in bytes. The source start address generated by the DMA is a byte address, made up of the concatenation of DMA_CSSA_U and DMA_CSSA_L.	RW	Undefined

4.23.3 Channel Destination Start Address

Each channel has two destination start address registers: DMA_CDSA_L and DMA_CDSA_U. For the first access to the destination port of the channel, the DMA controller generates a byte address by concatenating the contents of the two I/O mapped registers.

Source address DMA_CDSA_U:DMA_CDSA_L

Note:

You must load the source start address registers with a byte address. If you have a word address, shift left by 1 before loading the registers. If you have a 16-bit or 32-bit data type, the start address must be aligned properly. See the description of the data type bits of DMA_CSDP.

Table 118. Channel Destination Start Address, Lower Bits Register (DMA_CDSA_L0...DMA_CDSA_L5)

Bit	Name	Function	Type	Reset
31:16	Reserved			
15:0	Destination start address, lower bits	Lower bits for the destination start address are expressed in bytes. The destination start address is up to a 32-bit byte address, made of the concatenation of DMA_CDSA_U and DMA_CDSA_L.	RW	Undefined

Table 119. Channel Destination Start Address, Upper Bits Registers (DMA_CDSA_U0...DMA_CDSA_U5)

Bit	Name	Function	Type	Reset
31:16	Reserved			
15:0	Destination start address, upper bits	Upper bits for the source start address are expressed in bytes. The destination start address is made of the concatenation of DMACDSA_U and DMA_CDSA_L. Each channel has an element number register. Load DMA_CEN with the number of elements you want in each frame for the channel. The number of elements specified can range from 1 to 65535.	RW	Undefined

Table 120. Channel Element Number Registers (DMA_CEN0...DMA_CEN5)

Bit	Name	Function	Type	Reset
31:16	Reserved			
15:0	Channel element number	Element number Number of elements within a frame. The maximum element number is 65535. Each channel has a frame number register. Load DMA_CFN with the number of frames you want per block in each channel. The number of frames specified can range from 1 to 65535.	RW	Undefined

Table 121. Channel Frame Number Registers (DMA_CFN0...DMA_CFN5)

Bit	Name	Function	Type	Reset
31:16	Reserved			
15:0	Channel frame number	Frame number Number of frames within the block to be transferred. The maximum frame number is 65535. The size in bytes of the data block to transfer is DMA_CFN x DMA_CEN x DMA_CES, where DMA_CES is (1, 2, or 4 bytes) as set by DMA_CSDP bits 1:0. This size is programmed in bytes to allow transfer of an odd byte number and to accommodate the requirements of different access sizes on source and destination ports. Each DMA channel has a source frame index register (DMA_CSFI) and a source element index register (DMA_CSEI). Load DMA_CSFI with the frame index you want to use for the double-index addressing mode. Load DMA_CSEI with the element index you want for single- or double-index addressing mode. Select single- or double-index addressing mode separately for the source and destination ports by using the SRC_AMODE bits and the DST_AMODE bits, respectively, in the DMA_CCR.	RW	Undefined

Table 122. Channel Source Frame Index Registers (DMA_CSFI0...DMA_CSFI5)

Bit	Name	Function	Type	Reset
31:16	Reserved			
15:0	Source frame index	Channel source frame index Contains the channel source frame index, expressed in bytes, which is used to compute the addresses when double-index addressing mode is used.	RW	Undefined

Table 123. Channel Source Element Index Registers (DMA_CSEI0...DMA_CSEI5)

Bit	Name	Function	Type	Reset
31:16	Reserved			
15:0	Source element index	Element index Contains the channel source element index, expressed in bytes, which is used to compute the addresses when single-indexed addressing mode is used. Each DMA channel has a destination frame index register (DMA_CDFI) and a source element index register (DMA_CDEI). Load DMA_CDFI with the frame index for the double-index addressing mode. Load DMA_CDEI with the element index for single or double-index addressing mode. Select single- or double-index addressing mode separately for the source and destination ports by using the SRC_AMODE bits and the DST_AMODE bits, respectively, in the DMA_CCR.	RW	Undefined

Table 124. Channel Source Address Counter Registers (DMA_CSAC0...DMA_CSAC5)

Bit	Name	Function	Type	Reset
31:16	Reserved			
15:0	Source element/frame address 16 LSB	This register can be used to monitor the progress of a DMA transfer on channel source port. It is a snapshot of the current source address generated by the channel source address counter. The counter is incremented on each element transfer from the channel source port (s8, s16, or s32).	R	Undefined

Table 125. Channel Destination Address Counter Registers (DMA_CDAC0...DMA_CDAC5)

Bit	Name	Function	Type	Reset
31:16	Reserved			
15:0	Destination element/frame address 16 LSB	This register can be used to monitor the progress of a DMA transfer on channel destination port. It is a snapshot of the current destination address generated by the channel destination address counter. The counter is incremented on each element transfer to the channel destination port (s8, s16 or s32). Each DMA channel has a source frame index register DMA_CSFI and source element index register DMA_CSEI. Load DMA_CSFI with the frame index for the double-index addressing mode. Load DMA_CSEI with the element index for single- or double-index addressing mode. Select single or double-index addressing mode separately for the source and destination ports by using the SRC_AMODE bits and the DST_AMODE bits, respectively, in the DMA_CCR.	R	Undefined

Table 126. Channel Destination Element Index Registers (DMA_CDEI0...DMA_CDEI5)

Bit	Name	Function	Type	Reset
31:16	Reserved			
15:0	Destination element index	Channel destination element index Contains the channel source element index, expressed in bytes, which is used to compute the addresses when single-/double-indexed addressing mode is used. **Note:** When GSCR[0]=1, DESTINATION_ELEMENT_INDEX = DMA_CDEI. When GSCR[0]=0, DESTINATION_ELEMENT_INDEX = DMA_CSEI. For double-/single-index addressing mode.	RW	Undefined

Table 127. Channel Destination Frame Index Registers (DMA_CDFI0...DMA_CDFI5)

Bit	Name	Function	Type	Reset
31:16	Reserved			
15:0	Destination frame index	Channel destination frame index Contains the channel source element index, expressed in bytes, which is used to compute the addresses when double-indexed addressing mode is used. **Note:** When GSCR[0] = 1, DESTINATION_FRAME_INDEX= DMA_CDFI. When GSCR[0] = 0, DESTINATION_FRAME_INDEX= DMA_CSFI. For double-index addressing mode.	RW	Undefined

4.24 DSP DMA Programming Guidelines

This section discusses the DSP DMA programming guidelines.

4.24.1 Transfer Source and Destination

In the DSP, the DMA is able to transfer data between the MPUI, the EMIF (and thus external memories), the TIPB bridge, the SARAM, and the DARAM. It includes five ports and six independent channels. This means that up to six DMA transfers can run simultaneously. Each channel has a set of dedicated registers and a global register.

4.24.2 Transfer Start

There are two ways to start a DMA transfer:

- Software start (software request): After setting up the configuration registers of a DMA channel, the processor activates the transfer in this channel by writing the DMA_CCR.EN bit of this channel. The transfer immediately begins.
- Hardware start (hardware request): The processor sets up a DMA channel and designates it as a synchronized transfer (demand-driven by DMA requests from the external DMA). The channel then waits for a DMA request to start transferring. Each time the DMA requests this channel, an amount of data is transferred. This amount of data can be:
  - A complete element transferred in response to a DMA request
  - A complete frame of several elements transferred in response to a DMA request

All of the transfers can be synchronized to DMA requests, whatever their sources and destinations.

The DMA requests come from signals PENDMAREQ(19:0) located on the DSP boundary. One DMA request can trigger several channels at the same time.

4.24.3 Autoinitialization

A DMA channel (synchronized or not) can operate in two modes: Single transfer mode and autoinitialization mode.

- In single transfer mode, the channel stops when the current transfer finishes.
- In autoinitialization mode, the channel loads a new configuration and automatically restarts a new transfer when the current one finishes.

A DMA channel has two sets of configuration registers: the programming set and the active set. Only the programming set is accessible to users (through the TIPB bus). When a channel is enabled for the first time, or when a channel autoinitializes, the programming set is copied in the active set of registers. Users can then program the programming set to configure the next transfer while the current one is running.

This feature can be used in two ways:

- Continuous operation: Users can change the programming registers while the current configuration is executed. The next transfer is performed with a new context, but without stopping the DMA.

- Repetitive operation: Users never modify the programming registers. The same context is always used.

The programming set includes the following registers:

- DMA_CSSA_L
- DMA_CSSA_U
- DMA_CDSA_L
- DMA_CDSA_U
- DMA_CEN
- DMA_CFN
- DMA_CFI
- DMA_CEI

The following registers are part of the working set and are accessible to the user. They always have an effect on the current transfer.

- DMA_CSDP
- DMA_CCR
- DMA_CICR
- DMA_CSR

An error can occur if the CPU accesses a channel to program it while it is reloading. To avoid this error, the CPU can use the DMA_CCR.END_PROG bit as follows:

- The CPU must set the bit to 1 once the channel setup is finished.
- The DMA clears the bit when the programming set of registers is loaded in the working set of registers.

When a transfer finishes, the DMA channel waits until END_PROG is 1 before proceeding to reload. Once the reload is done, the DMA clears the END_PROG bit.

For repetitive operations (reloading of the same context), the DMA_CCR.REPEAT bit can disable the effect of the END_PROG bit on the reload behavior. If this bit is set to 1, the DMA channel always reloads its working set of registers, whatever the END_PROG value is. Table 128 describes the effects of the DMA_CCR.AUTOINIT, END_PROG, and REPEAT bits.

Table 128. DSP DMA DMA_CCR.AUTOINIT, END_PROG and REPEAT Bit Effects

AUTOINIT	END_PROG	REPEAT	Autoinitialization Behavior
0	x	x	No autoinitialization.
1	0	0	At the end of current transfer, the channel waits until end prog = 1 to load the programming register set to its working register set.
1	1	0	At the end of current transfer, channel immediately loads the programming register set in its working register set.
1	x	1	

4.24.4 Addressing Modes

The DMA has four addressing modes: Constant, post-incremented, indexed, and double-indexed.

The amount of data (block size) to transfer is programmed in bytes. This size can be odd or even.

The data block to transfer is split into frames and elements. The byte size of this data block is:

BS = FN x EN x ES

where:

BS is the block size in bytes.

FN is the number of frames in the block, $1 \leq FB \leq 65535$.

EN is the number of elements per frame, $1 \leq EF \leq 65535$.

ES is the number of bytes per element, ES $\in$ {1, 2, 4 }. An element can be:

- An 8-bit scalar data, s8
- A 16-bit scalar data, s16
- A 32-bit scalar data, s32

The types of data transferred in a channel are s8, s16, and s32.

FN, EN, and ES (or data type) are configuration registers of the channel.

To set up a channel for a transfer, the software needs to program two addressing modes:

- Source addressing mode
- Destination addressing mode

Both modes are independent. For example, to transfer data from a TIPB serial port to internal memory, the source addressing mode is constant and the destination one is post-incremented.

The number of frames, the number of elements, and the element size are the same for the source and destination.

4.24.5 Data Packaging and Bursting

A DMA channel has the capacity to:

- Pack several consecutive byte accesses in a single word16, word32, or burst4 access. This increases the transfer rate. For a channel, the decision to pack or burst accesses to its source port is taken by the source address unit and depends on source port access capability. The decision to pack accesses to its destination port is taken by the destination address unit and depends on destination port access capability. Packing and bursting are done only if allowed by the software.
- Split a single word transfer into several byte accesses. This is done if the DMA port data size is less than the size (or type) of the data moved.

Burst pipelining is not supported. There is one wait cycle between each burst.

4.24.6 Data Alignment

During a transfer, all of the addresses computed by the DMA must be aligned on the type of data transferred if:

- Data type is s8 (8-bit scalar data); addresses can have any value.
- Data type is s16 (16-bit scalar data); addresses must be aligned on 16-bit word boundary (the lowest bit of the address is always 0).
- Data type is s32 (32-bit scalar data); addresses must be aligned on 32 bit-word boundary (the 2 lowest bit of the address always 00).

When using the indexed addressing modes (element index and/or frame index), all of the addresses computed must be aligned on the data type.

The DMA forces the address alignment by hardware on the type of data transferred.

4.24.7 Interrupt Generation

Each DMA channel can generate an interrupt to the processor to reflect the transfer status. Each DMA channel has a dedicated interrupt line to the processor. For the processor to take a DMA interrupt into account, the signals PENINTDMA(5:0), available on the DSP boundary, must be connected to some of the peripheral interrupt lines (PENINTPERH(20:0)).

The following interrupt sources can be programmed for every DMA channel:

- End of block: The last byte of the transfer has been written to destination.
- End of frame: The last byte of the current frame has been written to destination.
- Half of frame: The middle byte of the current frame has been written to destination.
- Start of last frame: The first word of the last frame has been written to destination.
- DMA request collision: A new DMA request occurred before the end of service of the previous one.
- Time-out: An access (to the source or the destination) was timed out.

To prevent a channel from definitively locking a memory or peripheral, all of the DMA ports to memory/peripheral requests are monitored by a time-out:

- When the DMA sends a request to transfer data in a channel, a time-out counter is triggered.

- When the request is acknowledged, the time-out counter is stopped.

- If the time-out counter reaches its threshold before the request is acknowledged, the request is discarded and an error is reported in the DMA channel by setting the relevant bit in DMA_CSR (channel status register) and sending an interrupt to the processor. The channel is stopped.

For a channel, all of the sources are combined to generate one interrupt. When an interrupt is issued by a channel, its status register (DMA_CSR) is set to record the interrupt cause. The processor interrupt service routine (ISR) can read this channel status register to determine the source of the interrupt. The status bits are automatically cleared after they are read.

4.24.8 Memory Space Issues

To set up a transfer, the software specifies:

- A source port with an address that must hit in the source port memory space.
- A destination port with an address that must hit in the destination port memory space.

If the software specifies a port with an address that does not hit this port memory space (example: source port = SARAM with a DARAM start address specified), the transfer goes on and memory can be corrupted. No address space check is performed by the DMA and outside the DMA.

Programmers must ensure coherency between the source port and source start address, and between the destination port and destination start address.

4.24.9 DMA Operation in Power-Down Mode

The DMA has five clock domains per channel and one per port.

The DMA stops all clocks in a channel when the channel is inactive, and partially stops the clocks (14% of the logic remains active) when the channel is active but blocked (waiting for DMA request or wait states during transfer).

The clock in a port is stopped when a channel does not use this port or when it is blocked (because of wait states).

Index

A

C

D

I

M

N

R

S

T

OMAP5912 Multimedia Processor Memory Interfaces Reference Guide

Literature Number: SPRU756A
March 2004

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.

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
		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 chapter describes the memory interfaces of the OMAP5912 multimedia processor.

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.

Related Documentation From Texas Instruments

The following documents describe the OMAP5910 device and related peripherals. Copies of these documents are available on the Internet at www.ti.com. *Tip:* Enter the literature number in the search box provided at www.ti.com.

OMAP5912 Multimedia Processor Device Overview and Architecture Reference Guide (literature number SPRU748) introduces the setup, components, and features of the OMAP5912 multimedia processor and provides a high-level view of the device architecture.

OMAP5912 Multimedia Processor OMAP 3.2 Subsystem Reference Guide (literature number SPRU749) introduces and briefly defines the main features of the OMAP3.2 subsystem of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor DSP Sybsystem Reference Guide (literature number SPRU750) describes the OMAP5912 multimedia processor DSP subsystem. The digital signal processor (DSP) subsystem is built around a core processor and peripherals that interface with: 1) The ARM926EJS via the microprocessor unit interface (MPUI); 2) Various standard memories via the external memory interface (EMIF); 3) Various system peripherals via the TI peripheral bus (TIPB) bridge.

OMAP5912 Multimedia Processor Clocks Reference Guide (literature number SPRU751) describes the clocking mechanisms of the OMAP5912 multimedia processor. In OMAP5912, various clocks are created from special components such as the digital phase locked loop (DPLL) and the analog phase-locked loop (APLL).

OMAP5912 Multimedia Processor Initialization Reference Guide (literature number SPRU752) describes the reset architecture, the configuration, the initialization, and the boot ROM of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Power Management Reference Guide (literature number SPRU753) describes power management in the OMAP5912 multimedia processor. The ultralow-power device (ULPD) generates and manages clocks and reset signals to OMAP3.2 and to some peripherals. It controls chip-level power-down modes and handles chip-level wake-up events. In deep sleep mode, this module is still active to monitor wake-up events.This book describes the ULPD module and outline architecture.

OMAP5912 Multimedia Processor Security Features Reference Guide (literature number SPRU754) describes the security features of teh OMAP5912 multimedia processor. The OMAP5912 security scheme relies on the OMAP3.2 secure mode. The distributed security on the OMAP3.2 platform is a Texas Instruments solution to address m-commerce and security issues within a mobile phone environment. The OMAP3.2 secure mode was developed to bring hardware robustness to the overall OMAP5912 security scheme.

OMAP5912 Multimedia Processor Direct Memory Access (DMA) Support Reference Guide (literature number SPRU755) describes the direct memory access support of the OMAP5912 multimedia processor. The OMAP5912 processor has three DMAs:

- The system DMA is embedded in OMAP3.2. It handles DMA transfers associated with MPU and shared peripherals.
- The DSP DMA is embedded in OMAP3.2. It handles DMA transfers associated with DSP peripherals.
- The generic distributed DMA (GDD) is an OMAP5912 resource attached to the SSI peripheral. It handles only DMA transfers associated with the SSI peripheral.

OMAP5912 Multimedia Processor Memory Interfaces Reference Guide (literature number SPRU756) describes the memory interfaces of the OMAP5912 multimedia processor.

- SDRAM (external memory interface fast, or EMIFF)
- Asynchronous and synchronous burst memory (external memory interface slow, or EMIFS)
- NAND flash (hardware controller or software controller)
- CompactFlash on EMIFS interface
- Internal static RAM

OMAP5912 Multimedia Processor Interrupts Reference Guide (literature number SPRU757) describes the interrupts of the OMAP5912 multimedia processor. Three level 2 interrupt controllers are used in OMAP5912:

- One MPU level 2 interrupt handler (also referred to as MPU interrupt level 2) is implemented outside of OMAP3.2 and can handle 128 interrupts.
- One DSP level 2 interrupt handler (also referred to as DSP interrupt level 2.1) is instantiated outside of OMAP3.2 and can handle 64 interrupts.
- One OMAP3.2 DSP level 2 interrupt handler (referenced as DSP interrupt level 2.0) can handle 16 interrupts.

OMAP5912 Multimedia Processor Peripheral Interconnects Reference Guide (literature number SPRU758) describes various periperal interconnects of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Timers Reference Guide (literature number SPRU759) describes various timers of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Serial Interfaces Reference Guide (literature number SPRU760) describes the serial interfaces of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Universal Serial Bus (USB) Reference Guide (literature number SPRU761) describes the universal serial bus (USB) host on the OMAP5912 multimedia processor. The OMAP5912 processor provides several varieties of USB functionality. Flexible multiplexing of signals from the OMAP5912 USB host controller, the OMAP5912 USB function controller, and other OMAP5912 peripherals allow a wide variety of system-level USB capabilities. Many of the OMAP5912 pins can be used for USB-related signals or for signals from other OMAP5912 peripherals. The OMAP5912 top-level pin multiplexing

controls each pin individually to select one of several possible internal pin signal interconnections. When these shared pins are programmed for use as USB signals, the OMAP5912 USB signal multiplexing selects how the signals associated with the three OMAP5912 USB host ports and the OMAP5912 USB function controller can be brought out to OMAP5912 pins.

OMAP5912 Multimedia Processor Multi-channel Buffered Serial Ports (McBSPs) Reference Guide (literature number SPRU762) describes the three multi-channel buffered serial ports (McBSPs) available on the OMAP5912 device. The OMAP5912 device provides multiple high-speed multichannel buffered serial ports (McBSPs) that allow direct interface to codecs and other devices in a system.

OMAP5912 Multimedia Processor Camera Interface Reference Guide (literature number SPRU763) describes two camera inerfaces implemented in the OMAP5912 multimedia processor: compact serial camera port and camera parallel interface.

OMAP5912 Multimedia Processor Display Interface Reference Guide (literature number SPRU764) describes the display interface of the OMAP5912 multimedia processor.

- LCD module
- LCD data conversion module
- LED pulse generator
- Display interface

OMAP5912 Multimedia Processor Multimedia Card (MMC/SD/SDIO) (literature number SPRU765) describes the multimedia card (MMC) interface of the OMAP5912 multimedia processor. The multimedia card/secure data/secure digital IO (MMC/SD/SDIO) host controller provides an interface between a local host, such as a microprocessor unit (MPU) or digital signal processor (DSP), and either an MMC or SD memory card, plus up to four serial flash cards. The host controller handles MMC/SD/SDIO or serial port interface (SPI) transactions with minimal local host intervention.

OMAP5912 Multimedia Processor Keyboard Interface Reference Guide (literature number SPRU766) describes the keyboard interface of the OMAP5912 multimedia processor. The MPUIO module enables direct I/O communication between the MPU (through the public TIPB) and external devices. Two types of I/O can be used: specific I/Os dedicated for 8 x 8 keyboard connection, and general-purpose I/Os.

OMAP5912 Multimedia Processor General-Purpose Interface Reference Guide (literature number SPRU767) describes the general-purpose in-

terface of the OMAP5912 multimedia processor. There are four GPIO modules in the OMAP5912. Each GPIO peripheral controls 16 dedicated pins configurable either as input or output for general purposes. Each pin has an independent control direction set by a programmable register. The two−edge control registers configure events (rising edge, falling edge, or both edges) on an input pin to trigger interrupts or wake−up requests (depending on the system mode). In addition, an interrupt mask register masks out specified pins. Finally, the GPIO peripherals provide the set and clear capabilities on the data output registers and the interrupt mask registers. After detection, all event sources are merged and a single synchronous interrupt (per module) is generated in active mode, whereas a unique wake−up line is issued in idle mode. Eight data output lines of the GPIO3 are ORed together to generate a global output line at the OMAP5912 boundary. This global output line can be used in conjunction with the SSI to provide a CMT−APE interface to the OMAP5912.

OMAP5912 Multimedia Processor VLYNQ Serial Communications Interface Reference Guide (literature number SPRU768) describes the VLYNQ of the OMAP5912 multimedia processor.

VLYNQ is a serial communications interface that enables the extension of an internal bus segment to one or more external physical devices. The external devices are mapped into local, physical address space and appear as if they are on the internal bus of the OMAP 5912. The external devices must also have a VLYNQ interface. The VLYNQ module serializes bus transactions in one device, transfers the serialized data between devices via a VLYNQ port, and de-serializes the transaction in the external device.

OMAP5912 includes one VLYNQ module connected on OCPT2 target port and OCPI initiator port. These connections are configured via a static switch, which selects either SSI or VLYNQ module. This switch, forbids the simultaneous use of GDD/SSI and VLYNQ. The switch is controlled by the VLYNQ_EN bit in the OMAP5912 configuration control register (CONF_5912_CTRL).

OMAP5912 Multimedia Processor Pinout Reference Guide (literature number SPRU769) provides the pinout of the OMAP5912 multimedia processor. After power-up reset, the user can change the configuration of the default interfaces. If another interface is available on top of the default, it is possible to enable a new interface for each ball by setting the corresponding 3-bit field of the associated FUNC_MUX_CTRL register. It is also possible to configure on-chip pullup/pulldown. This document

also describes the various power domains so that the user can apply the different interfaces seamlessly with external components.

OMAP5912 Multimedia Processor Window Tracer (WT) Reference Guide (literature number SPRU770) describes the window tracer module used to capture the memory transactions from four interfaces: EMIFF, EMIFS, OCP-T1, and OCP-T2. This module is located in the OMAP3.2 traffic controller (TC).

OMAP5912 Multimedia Processor Real-Time Clock Reference Guide (literature number SPRUxxx) describes the real-time clock of the OMAP5912 multimedia processor. The real-time clock (RTC) block is an embedded real-time clock module directly accessible from the TIPB bus interface.

Trademarks

Contents

Figures

Tables

Memory Interfaces

This document describes the memory interfaces of the OMAP5912 multimedia processor.

1 Introduction

This document describes the following interfaces:

- SDRAM (external memory interface fast, or EMIFF)
- Asynchronous and synchronous burst memory (external memory interface slow, or EMIFS)
- NAND flash (hardware controller or software controller)
- CompactFlash on EMIFS interface
- Internal static RAM

1.1 SDRAM Interface

The following serves to give a brief overview of the EMIFF. Please see Chapter 2 for a detailed description of the EMIFF. The OMAP EMIFF is an SDRAM controller that manages access by the various initiators of an OMAP-based system. It can support one 16-bit device or two 8-bit devices. The external interface data bus width is always 16 bits.

The EMIFF supports the following devices:

- Standard single-data-rate SDRAM
- Low-power single-data-rate SDRAM
- Standard double-data-rate SDRAM
- Mobile double-data-rate SDRAM

In terms of capacity and organization of memory components, the EMIFF can handle:

- 1G-bit, 512M-bit, 256M-bit, 128M-bit, 64M-bit, and 16M-bit devices
- Two-bank 16M-bit devices, two-bank or four-bank devices for 64M-bit devices, four-bank devices only for any other capacity
- x8 (two devices) or x16 (single device) data bus configuration, except for the 1G-bit device: the EMIFF supports only a x16-1G-bit device (single device). The maximum external SDRAM configuration is 128 megabytes.

Program the SDRAM_TYPE field of the EMIFF interface SDRAM configuration register to specify the physical configuration of the devices.

The SDRAM type selection is the first action required from the software driver, using the SDRAM_TYPE field of the EMIF SDRAM operation register.

The SDRAM controller supports:

- The self-refresh mode (idle), autorefresh, and other operating modes (HPHB, LPLB, and POM0 modes)
- MRS command and extended MRS command for:
  - DDR SDRAM and low-power SDRAM, sent via the SDRAM request manager
  - SDR SDRAM, all burst sizes, between 1 and 32 consecutive accesses
  - DDR SDRAM, only bursts of 8
- Two pipelined levels of request from the SDRAM request manager to enable page interleave timing and reduce overhead cycles by the burst interruption mechanism.

  For example, the following SDR SDRAMs from Samsung are supported:

  - K4S64163LF (4M X 16)
  - K4S28163LD (8M X 16)

The SDR interface voltage support is 1.8 V (1.65 V–1.95 V), 2.75 V (2.5 V–3.0 V), and 3.3 V (3.0 V–3.6 V).

This interface can also support Samsung mobile SDRAM with DDR capability at both 128M bits and 256M bits.

The DDR voltage support is 1.8 V (1.65 V – 1.95 V) only.

1.1.1 OMAP1612 Stacked DDR Support

The OMAP1612 device includes a stacked mobile DDR (dual-data-rate) SDRAM. This device is directly connected to the EMIFF interface, and that interface is not available for connecting external memories in the OMAP1612 device.

The OMAP1612 stacked DDR SDRAM has the following characteristics:

- Elpida ECK2516CBCZ−10 part
- 1.8-V power supply
- 1.8-V I/O power

- ❏ 16Mx16 mobile DDR SRAM
- ❏ Four-bank operation
- ❏ Differential clock inputs
- ❏ MRS cycle with address key programs
  - CAS latency (3)
  - Burst length (2, 4, 8)
  - Burst type (Sequential or Interleave)
  - Partial self refresh type (1, 2, 4 banks)
  - Temperature compensated self refresh

1.1.2 EMIFF Configuration for OMAP1612 Stacked DDR

The recommended configuration sequence for using the OMAP1612 stacked DDR is as follows. Note that Step 2 and Step 5 (refresh counter value and delay loop) depend on the system frequency.

The refresh counter value must be calculated with the usual formula. The total refresh time for the whole stacked DDR is 64 µs.

1) Set SDRAM_OPERATION_REG to 0x7 (high bandwidth mobile DDR operation).
2) Set EMIFF_SDRAM_CONFIG register to 0x000xxxF6 (enable auto-refresh; SDRAM type = 4 banks, 256 Mbits; Refresh counter= depends on clock frequency).
3) Set SDRAM_MANUAL_CMD_REG register to 0x7 (set CKE high command).
4) Set SDRAM_MANUAL_CMD_REG register to 0x0 (NOP command).
5) Add a delay loop (> 500 ns).
6) Set SDRAM_MANUAL_CMD_REG register to 0x1 (PRECHARGE command).
7) Set SDRAM_MANUAL_CMD_REG register to 0x2 (auto-refresh command).
8) Set EMIFF_MRS_NEW register to 0x33 (burst length = 8; CAS latency = 3).
9) Set EMIFF_EMRS1 register to 0x0 (refresh all banks; max 70° C).
10) Set DLL_URD_CONTROL register to 0x06 (URD DLL enabled, at 90° C).

11) Set DLL_LRD_CONTROL register to 0x06 (LRD DLL enabled, at 90° C).

12) Set DLL_WRT_CONTROL register to 0x06 (LRD DLL enabled, at 90° C).

1.2 Asynchronous and Synchronous Burst Memory Interface (EMIFS)

The synchronous/asynchronous external memory interface slow (EMIFS) supports most common memory interface protocols through a flexible programming and timing signals control. The following serves to give a brief overview of the EMIFS. Please see chapter 2 for a detailed description of the EMIFS.

The EMIFS can control up to six devices without adding any external logic through six independent chip-selects and through dedicated memory interface control signals. Two configurable options are supported:

- Four chip-selects. Each can support up to 64M bytes of addressable memory:
  - CS0 from 0000:0000 to 03FF:FFFF
  - CS1 from 0400:0000 to 07FF:FFFF
  - CS2 from 0800:0000 to 0BFF:FFFF
  - CS3 from 0C00:0000 to 0FFF:FFFF
- Six chip-selects. Two can support up to 64M bytes of addressable memory, and four can support up to 32M bytes of addressable memory:
  - CS0 from 0000:0000 to 03FF:FFFF
  - CS1_a from 0400:0000 to 05FF:FFFF
  - CS1_b from 0600:0000 to 07FF:FFFF
  - CS2_a from 0800:0000 to 09FF:FFFF
  - CS2_a from 0A00:0000 to 0BFF:FFFF
  - CS3 from 0C00:0000 to 0FFF:FFFF

The EMIFS supports common external memory control signals:

- OE
- WE
- ADV
- BE[0–1]
- BE[2–3]
- BAA
- Ready
- Device CLK
- RST
- WP

Each chip-select (CS) controls an address range with dedicated configuration registers to ensure compliance with the protocol and timing constraints of the external device associated with it. Each chip-select configuration register supports dynamic configuration.

The EMIFS can support 16-bit and 32-bit external device width. Based on the CS configuration, the interface adjusts the access size (splitting word32) according to the external device attached to the CS.

An 8-bit device width is not supported without adding external logic.

The EMIFS can control multiplexed address and data memory devices without adding external logic based on CS configuration. The multiplexing scheme is supported for synchronous and asynchronous access mode.

Both multiplexed and nonmultiplexed devices can be supported with the same integrated circuit (IC) on different chip-selects (embedded IC non-multiplexed memories and external multiplexed devices).

The EMIFS behavior conforms to the little-endian protocol. It supports 8-,16-, or 32-bit asynchronous and synchronous read, 4- x 32-bit synchronous burst read, and 8-, 16-, or 32-bit asynchronous write.

The EMIFS is a multimaster memory interface. It supports flexible and programmable arbitration protocol (LRU priority ordering or dynamic time-based priority ordering).

At boot time or at run time, CS0 and CS3 address mapping can be swapped.

The EMIFS includes a programmable time-out to prevent the system from hanging with nonresponding devices. Automatic access completion with interrupt and status logging are issued on time-out events.

The EMIFS supports dynamic local idle mode control. The EMIFS also supports IC deep power-down mode request synchronization.

OMAP5912 can also support CompactFlash devices through the EMIFS.

2 Memory Interfaces for the EMIFS

There are a number of different memory types that can connect with the EMIFS interface. These memory types will share the same pins on the device, but their functionalities and controlling logic may differ.

The memory types and their associated controlling logic is:

- ❏ Non-multiplexed NOR flash—controlled by EMIFS directly
- ❏ Address/Data multiplexed NOR flash—controlled by EMIFS directly
- ❏ 8−bit NAND flash—controlled by either the EMIFS directly (software NAND flash controller) or by the hardware NAND flash controller (NFC)

- 16−bit NAND flash—controlled by EMIFS directly (software NAND flash controller)
- CompactFlash—controlled by the compact flash controller (CFC)

Some of these controllers can be used simultaneously. See Figure 23, Figure 24, and Figure 25 for details on simultaneous connections of these memory types. The different combinations are:

- Address/Data multiplexed NOR on EMIFS and 8-bit NAND on NFC
- Non-multiplexed NOR on EMIFS and 16−bit NAND on EMIFS
- Non-multiplexed NOR on EMIFS and CFC

The following sections describe the different external memory controllers that use the EMIFS pins.

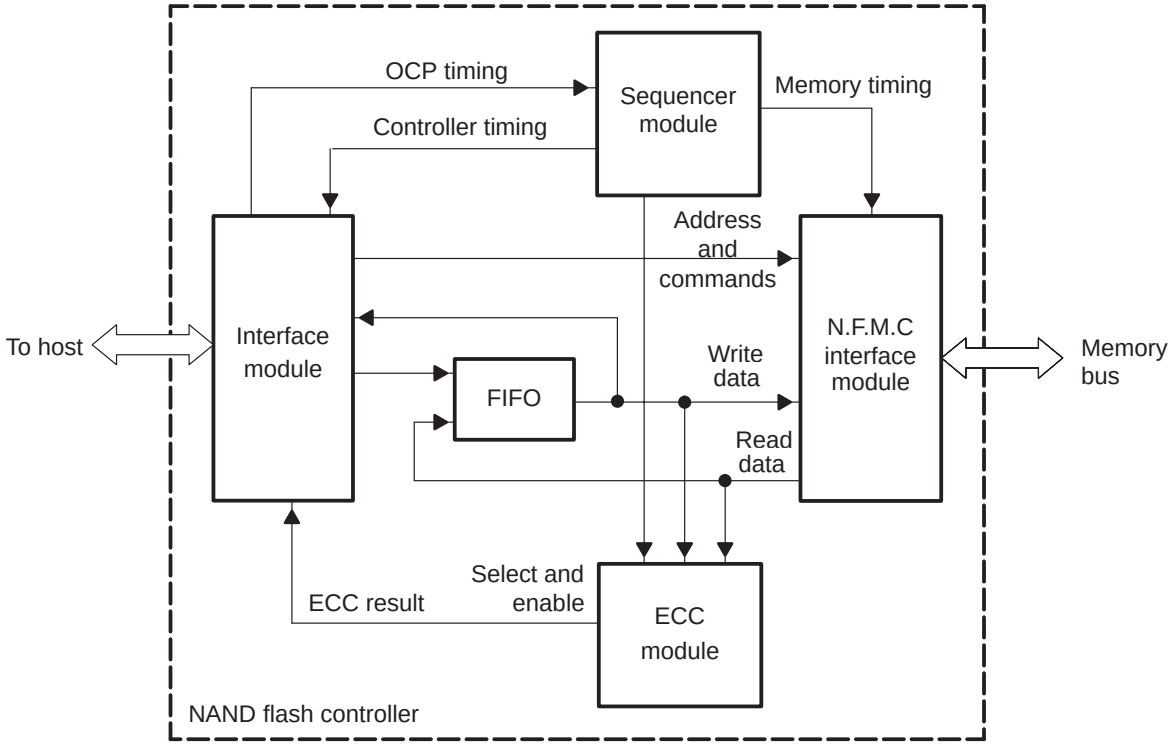
2.1 Hardware NAND Flash Controller

The NAND flash controller (NFC) is the interface between the host processor and the external NAND flash memory core (NFMC). The NFC (see Figure 1) supports the following NFMC configurations:

- 1G-bit
- 512M-bit
- 256M-bit
- 128M-bit
- 64M-bit
- 32M-bit

Only 8-bit bus widths are supported. See Figure 1 for an overview of the NAND flash controller.

Figure 1. NAND Flash Controller Overview



The external NAND flash (NFMC) is an 8-bit interface (byte addressable). From the host processor, the NFC can access data in 8, 16, or 32 bits.

The NFC supports prefetch and postwrite modes implemented around a FIFO buffer for latency optimization.

The NFC supports DMA transfers on a page-in-read and program operations, and has the following general features:

- ❏ Flexible architecture to support different sizes of NFMCs
- ❏ 8-bit interface to NAND flash
- ❏ 8-bit, 16-bit, or 32-bit interface from the host processor
- ❏ Error code correction (ECC) for maximum data integrity
- ❏ DMA support on one NAND flash memory page for read and program operations
- ❏ Ablity to support from one to four NFMCs

Accessing an NFMC requires an 8-bit bus, which is multiplexed among data, address, and commands. Table 1 lists the command operations.

Table 1. Command Operations

Function	First Cycle	Second Cycle	Third Cycle
Read 1 (lower half-page or Area A)	0x00	-	-
Read 1 (upper half-page or Area B)	0x01	-	-
Read 2 (spare or Area C)	0x50	-	-
Read ID	0x90	-	-
Reset	0xFF	-	-
Page program	0x80	0x10	-
Multiple-page program	0x80	0x11	-
Copy-back program	0x00	0x8A	0x10
Multiple copy-back program	0x03	0x8A	0x10/0x11
Block erase	0x60	0xD0	-
Multiplane block erase	0x60−0x60	0xD0	-
Read status	0x70	-	-
Read multiplane status	0x71	-	-

Associated with this 8-bit bus are qualifiers: CLE (command latch enable) which characterizes that the data on the bus is a command, and ALE (address latch enable), which characterizes that the value on the bus is an address. When CLE is low and ALE is low, the value on the bus is a data. The combination CLE high and ALE high is not permitted. Reading data requires a negative pulse on signal RE_. Writing command/address/data requires a negative pulse on signal WE_.

For program operation, data is latched into the NFMC on the rising edge of WE_. For read operation, data is latched on the falling edge of RE_. The NFMC signal Ready/Busy_ (R/B_) indicates the status of the ongoing operation. When low, R/B_ indicates that a program read or erase is in process and returns to high state upon completion. When returning high, the NFC also asserts an interrupt so that the local host can be aware that the ongoing operation is completed.

The core memory check page reference inside the core is organized by a page of 512 bytes and a spare area of 16 bytes for the most recent NFMC, as shown in Figure 18, *Single Page Program DMA in Postwrite Mode*.

The 528-byte page is itself divided into two main areas, A (from 0 to 255) and B (from 256 to 511), and a C or spare area (from 512 to 527). To access area

A, the command 0x00 is sent to the flash core. To access area B, the command 0x01 is sent. To access the spare area, the command 0x50 is sent (see Table 2).

Table 2. Pointer Operation

Command	Accessed Address	Area
0x00	0-255	Area A (lower half-page)
0x01	256-511	Area B (upper half-page)
0x50	512-527	Area C (spare)

The erase operation performs on a block basis. For the most recent NFMCs, one block equals 32 pages, or 16K bytes. On 512M-bit and 1G-bit NFMCs, there is also a logical array division by plane:

- The 512M-bit has four planes, each with 1024 blocks of 32 pages per block.
- The 1G-bit has eight planes, each with 1024 blocks of 32 pages per block.

To be generic, most control must be done in software. Commands, for instance, are not hard-coded because they can change in different versions of NFMC, or new commands can be added.

There are two types of commands: one must be followed by an address (for instance, read, program, and erase); the other does not need to be followed by an address (for instance, read status, end of data (0x10 or 0x11) in the case of program operation). An address register is necessary to store the starting address (32 bits). An access to the first type of command register places the command on the 8-bit NFMC bus and the data located in the address register also is driven on the bus. Writing data in the second type of command register does not issue an access to the address register. For read operation, after the address is transmitted to the NFMC, there is a latency time to wait for the data to be ready (typically 12 μs). The wait can be done either by an interrupt or by polling the R/B_.

2.1.1 Read Operation

First, the command 0x00, 0x01 or 0x50 is driven on the bus with the qualifier CLE being high. Then the start address is driven byte-by-byte with the least significant byte first, with the qualifier ALE being high. The NFMC drives its Ready/Busy_ (R/B_) to low. When R/B_ goes back to high, an interrupt is asserted and data is ready to be sampled by negative pulse on RE_. An access on the NAND controller access register (NND_ACCESS) triggers the negative

pulse on RE_ and data is read and stored in the NAND controller access register (NND_ACCESS). The interrupt bit must be cleared by software.

Another solution to waiting for data to be ready is to poll the ready bit in the NAND controller ready register (NND_READY). This bit is the sampled R/B_ pin of the NFMC. The read status command can also be sent and the NFMC status register tested. In that case, bit 6 of the NFMC status register reflects the R/B_ bit, as shown in Table 7 and Table 8. When R/B_ returns to 1, data is ready.

If the readiness of the NFMC was tested by accessing the NFMC status register, the command code 0x00 must be sent because the NFMC stays in read status mode when no new command is sent.

Reading 32 (16) bits is possible because it is decomposed on 4 (2) 8-bit access. Returned data is preserved in a 4 x 8-bit buffer until the current operation is completed. The data is then latched in a register in the NFC before being driven on the output sdata bus. When no operation is selected, the NFC drives 0x00000000 to the sdata bus (see Table 3).

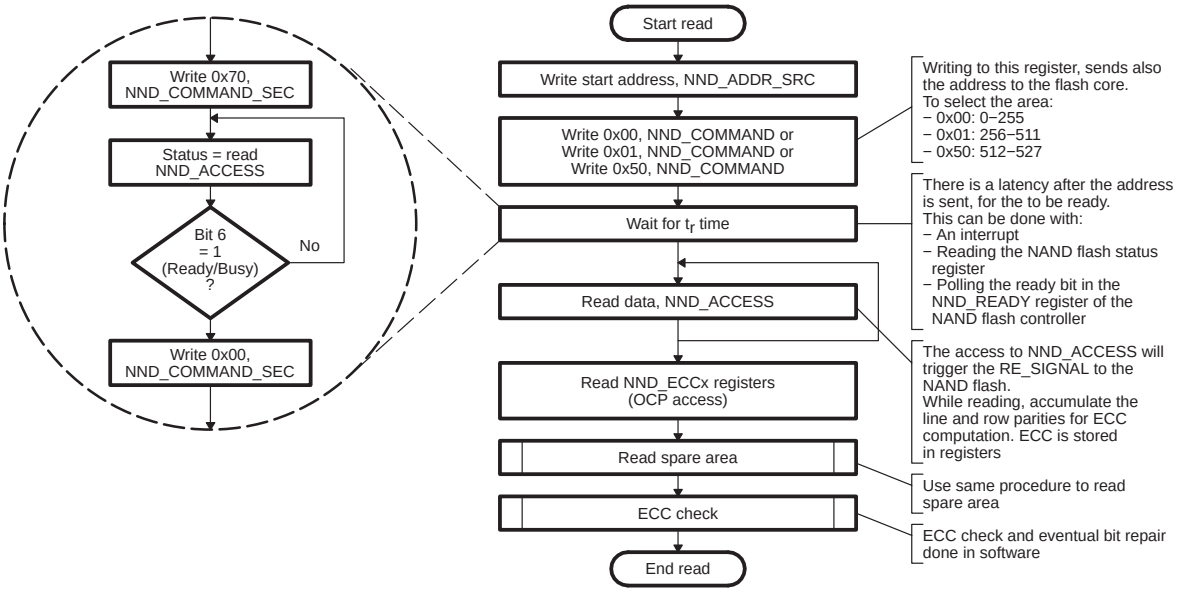
Table 3. Sent Address Function of Core Sizes

Size of Flash Core (MegaBits)	Address[31-24]	Address[23-16]	Address[15-8]	Address[7-0]
32	00000000	000-A21-A17	A16-A9	A7-A0
64	00000000	00-A22-A17	A16-A9	A7-A0
128	00000000	0-A23-A17	A16-A9	A7-A0
256	00000000	A24-A17	A16-A9	A7-A0
512	0000000-A25	A24-A17	A16-A9	A7-A0
1024	000000-A26-A25	A24-A17	A16-A9	A7-A0

The pointer 0x00, 0x01, or 0x50 selects which area to access. For instance, to access data at address 256 in the 528-byte page of the NAND flash, the command 0x01 must be sent with an address of 0x00. When the bit A8 of the NND_CTRL of the NFC is set to 0, the NFC does not send the bit 8 of the NND_ADDR_SRC register to the NFMC The address that is written in the NND_ADDR_SRC must be formatted with a dummy bit at location 8 of the address because the NFC hardware does not send this bit to the NFMC.

The flow chart shown in Figure 2 shows how to read one or multiple bytes in a page of the NFMC.

Figure 2. Read Operation

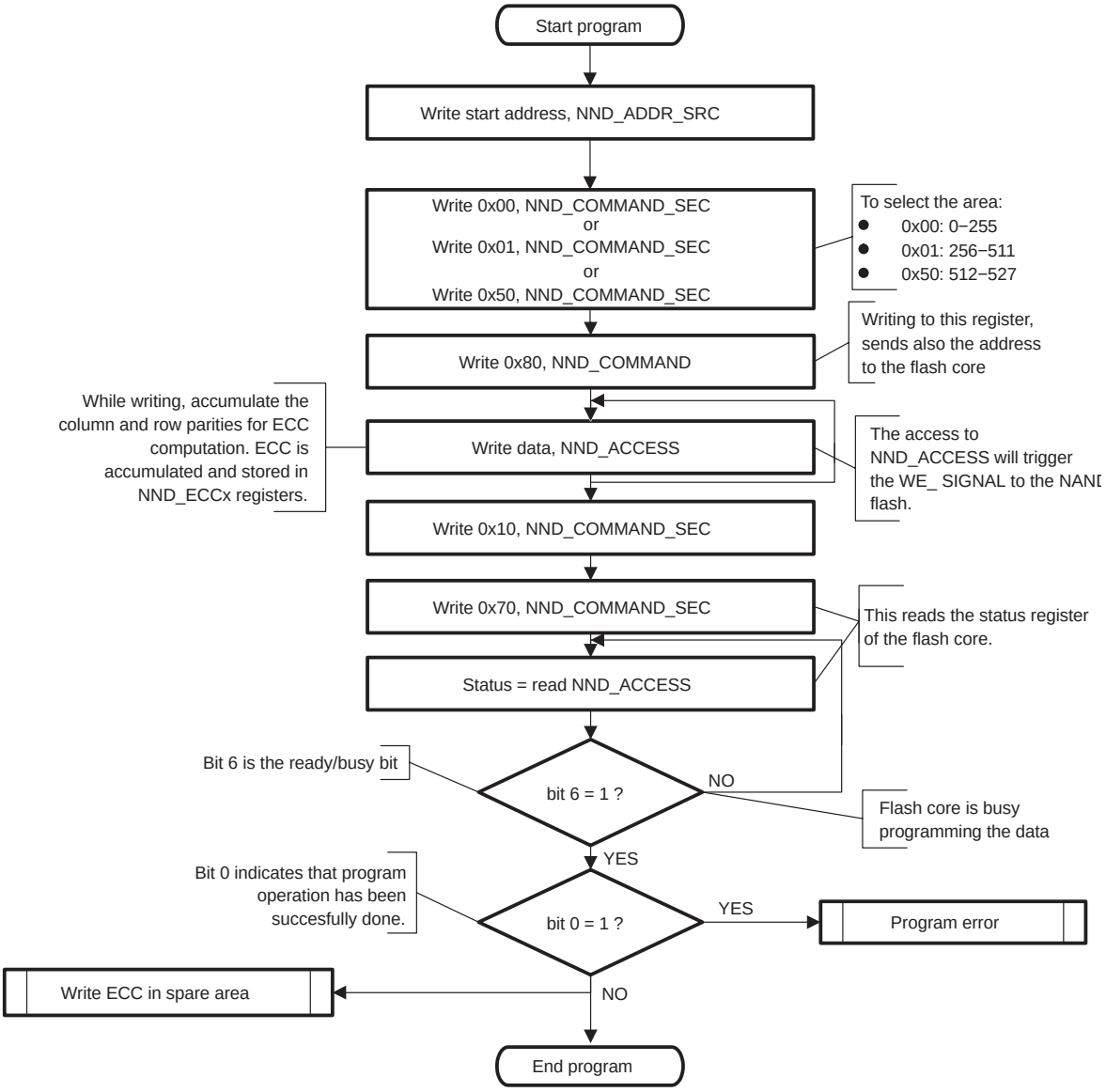


2.1.2 Write Operation

First, the command 0x00 or 0x01 or 0x50 is driven on the bus with qualifier CLE high. This is to select which area (A, B, or C) to program. Then, the command 0x80 is driven on the bus with the qualifier CLE being high. Last, the start address is driven byte-by-byte, least significant byte first, with the qualifier ALE being high (see Table 3).

Writing data in the NAND controller access register (NND_ACCESS) places the data on the bus and asserts a negative pulse on WE_ (CLE and ALE are low). After the last data has been driven, a command 0x10 is sent to terminate the flow of data. The NFMC drives R/B_ to low during the programming of NFMC (from its internal page register to memory cell) and R/B_ goes back to high, asserting an interrupt. This interrupt bit is cleared by software using the scheme shown in Table 30. The NFMC is now ready for the next operation. The result of the operation can be checked with a read status operation (sending the command 0x70 or 0x71) to the NFMC

Figure 3. Write Operation



2.1.3 Multiplane Page Program

The multiplane page program is an extension of the page program operation in which up to four pages (and for each page, up to 528 bytes) are programmed. This operation is available only on 512M-bit and 1G-bit NFMCs.

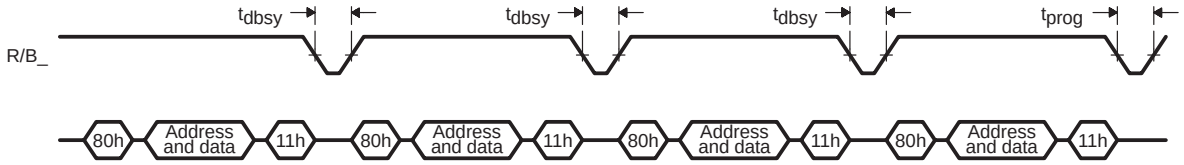
For the 1G-bit NFMC, multiplane page programming accesses pages in plane 0, 1, 2, and 3 or plane 4, 5, 6, and 7.

Restriction in Addressing With Multiplane Page Program

Although any block in each plane can be addressable for the multiplane page program, the page address in the selected block must be the same. This means that, given four addresses (one for each plane), bits 9 to 13 of those addresses must be the same because bits 9 to 13 select the address of the page in a block (see Figure 4).

It is impossible to start loading data in area B (upper half-page). The NFC does not check the validity of addresses.

Figure 4. Multi-Page Program Operation



The command to mark the end of data is 0x10 for the last operation and 0x11 for the first accesses. If the program operation fails, the software takes the responsibility to program the data in another page and mark the block (where the page belongs) as invalid. After each address and data sent, there is a latency time (T_{dbsy}). The actual programming of the flash occurs after the last address and data (T_{prog}). Typically, T_{dbsy} and T_{prog} are, respectively ,1 µs and 200 µs.

2.1.4 Erase Operation

The erase operation (see Figure 5) is possible only at block level. Command 0x60 is driven with CLE high, then the address of the block to erase is driven with ALE high, and the command 0xD0 is written with CLE high. During the physical erase, the flash core drives R/B_ to low. When the erase operation is completed, R/B_ returns to high. The command is repeatable up to four times (depending on the NFMC, because this operation is valid on 1G bit and 512M bits, as shown in Table 13, *Supported Operations on NFMCs.)* The status of the erase operation can be checked with a read status operation (command 0x70 or 0x71).

Table 4. Programming Address for Erase Operation

Size of Flash Core (MB)	Block Address (sent)	Address Page in Block (sent but ignored)	Start Address (not sent)	Erasable Block Size
32	A21-A13	A12-A9	A7-A0	8K

64	A22-A13	A12-A9	A7-A0	8K
128	A23-A14	A13-A9	A7-A0	16K
256	A24-A14	A13-A9	A7-A0	16K
512	A25-A14	A13-A9	A7-A0	16K
1024	A26-A14	A13-A9	A7-A0	16K

The address of the erase block must be properly formatted in the address register NN_ADDR_SRC. Bits that correspond to the address page in the selected block must be sent, but they are discarded by the internal logic of the NFMC (see Table 5).

Table 5. Formatting of Erase Address in Address Register

Size of Flash Core (MB)	Address[31-24]	Address[23-16]	Address[15-8]	Address[7-0]
32	00000000	00000000	000-A21-A17	A16-A9
64	00000000	00000000	00-A22-A17	A16-A9
128	00000000	00000000	0-A23-A17	A16-A9
256	00000000	00000000	A24-A17	A16-A9
512	00000000	0000000-A25	A24-A17	A16-A9
1024	00000000	000000-A26-A25	A24-A17	A16-A9

If the erase operation fails, software must mark the block as invalid by writing a value different from 0xFF in the invalid block location (6th byte) in the spare area.

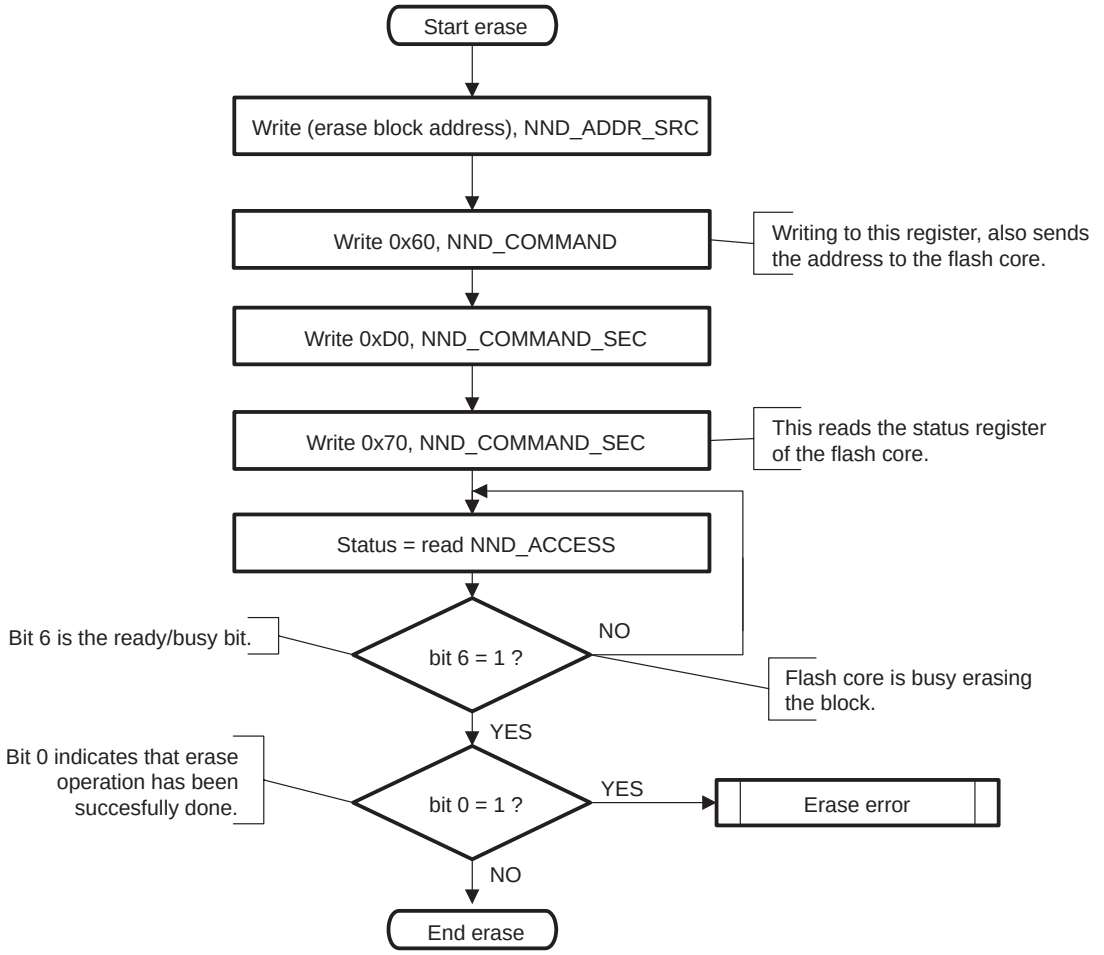
If the bit A8 of NND_CTRL is 0, then the erase address must be formatted with a dummy bit at location 8 of the erase address, because the NFC does not send the bit 8 to the NFMC (see Table 6).

Table 6. Formatting of Erase Address in Address Register With Bit A8 Not Sent

Size of Flash Core (MB)	Address[31-24]	Address[23-16]	Address[15-8]	Address[7-0]
32	00000000	00000000	00-A21-A17-**0**	A16-A9
64	00000000	00000000	0-A22-A17-**0**	A16-A9
128	00000000	00000000	A23-A17-**0**	A16-A9
256	00000000	0000000-A24	A23-A17-**0**	A16-A9

512	00000000	000000-A25-A24	A23-A17-**0**	A16-A9
1024	00000000	00000-A26-A24	A23-A17-**0**	A16-A9

Figure 5. Erase Operation



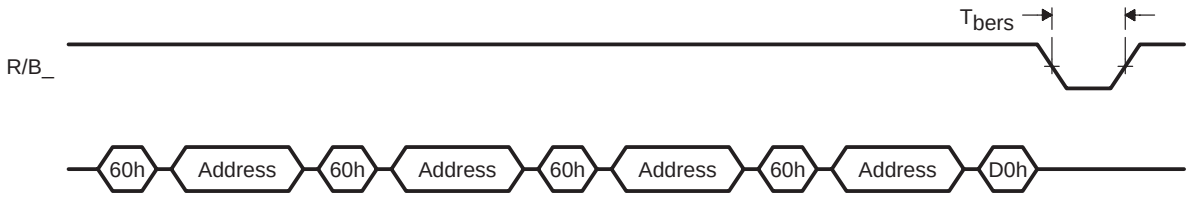
2.1.5 Multiplane Block Erase Operation

The multiplane block erase operation is identical to the multiplane page program. Up to four blocks, one from each plane, can be simultaneously erased. Standard block erase command sequences are repeated up to four times for erasing. Only one block must be selected from each plane. The NFC does not check the validity of these erase addresses (see Figure 6).

The erase confirm command (0xD0) initiates the erasing process and R/B_ goes low for T_{bers} time (typically 2 ms). The completion is detected by testing

bit 6 (Ready/Busy_) of the NFMC status register. When the erase is completed, the pass/fail status of each block is examined by reading the extended pass/fail status of the NFMC status register (bit 1 to bit 4) using the read
multi-plane status command (0x71). If not masked, an interrupt is asserted when R/B_ returns to high state.

Figure 6. Multiplane Block Erase Operation



Copy-Back Program Operation

The copy-back program quickly and efficiently rewrites data stored in one page within the plane to another page within the same plane, without using an external memory. Because the time-consuming sequentially reading and reloading cycles are removed, system performance is improved. A normal read operation with 0x00 command and the address of the source page moves the whole 528-byte data into an NFMC internal buffer. As soon as the NFMC returns to the ready state, command 0x8A with the address cycles of the destination page is sent to the NFMC. The confirm command (0x10) is required to begin the programming operation.

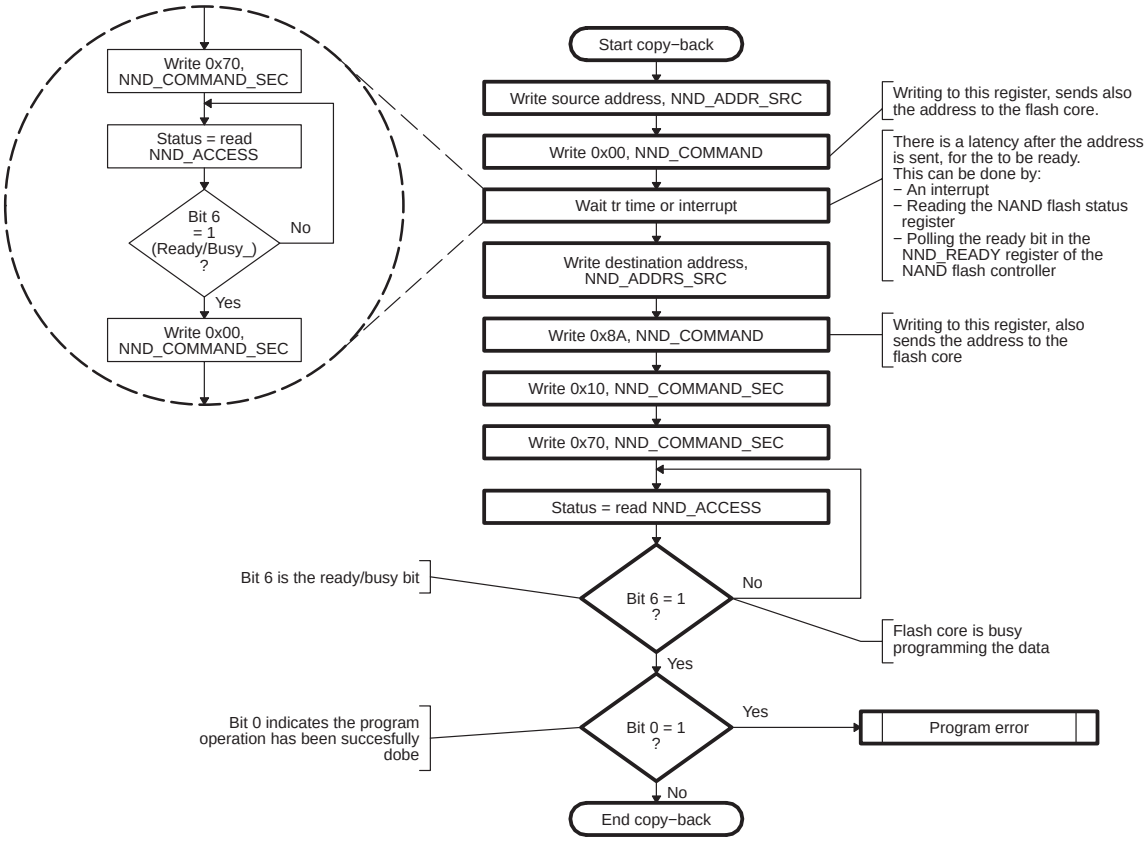
There is some restriction because the copy-back program operation is allowed only within the same memory plane. Although a full address is sent, bits 7 to 0 of the address are discarded. *The software must check the validity of page addresses.*

The source page and the destination page must be in the same plane:

- For 512M bits, bit 14 and bit 15 of the address must be the same.
- For 1G bit, bit 14, bit 15, and bit 26 of the address must be the same.

If the operation fails, software marks the block where the page belongs as invalid (see Figure 7).

Figure 7. Copy-Back Operation



2.1.6 Multiplane Copy-Back Program Operation

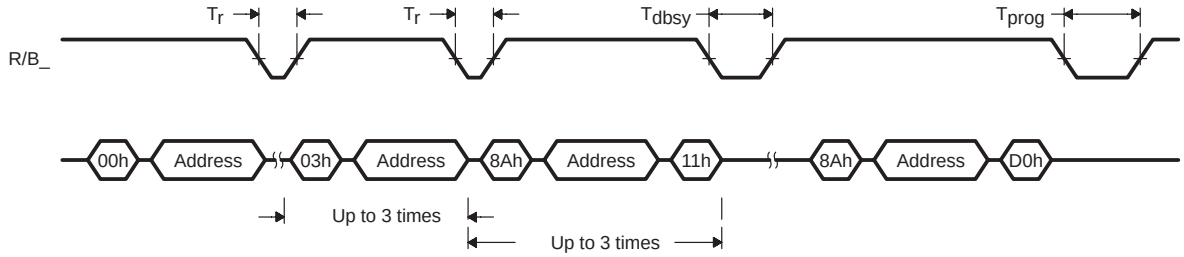
The multiplane copy-back program is an extension of the one-page copy-back program and is available on 512M-bit and 1G-bit NFMCs (see Figure 8).

First, the command for read operation (0x00) is sent, followed by the source address. Any further read operation (up to three) for transferring the addressed pages to the corresponding page register must be executed with command code 0x03 instead of a 0x00 command. Up to four pages can be addressed.

There is a latency time Tr (typically 12 µs), indicated by the R/B_ line going low, between each source address. Data is transferred from memory cells to the NFMC internal buffer. After the input of commands for reading the source pages, the same procedure as multiplane page programming occurs, except that the command operation is 0x8A instead of 0x80. Because no programming process is involved during data loading at the destination plane

address, R/B_ remains in a busy state for a short period of time. When the last command operation 0x10 is sent, thel programming of the NFMC starts and the NFMC drives its R/B_ signal low. A read status operation (command 0x70 or 0x71) can be issued to find out when the NFMC returns to ready state by polling the Ready/Busy_ bit in the status register (bit 6). Once the multiplane copy-back program is finished, any additional partial-page programming into the copied pages is prohibited before an erase operation. If not masked, an interrupt is asserted each time R/B_ returns to high state. Typically, T_r, T_{dbsy}, and T_{prog} are 12µs, 1µs, and 200µs, respectively (see Figure 8).

Figure 8. Multiplane Copy-Back Operation



2.1.7 Read Status and Read Multiplane Status Operations

The NFMC has a status register, which can be accessed to determine whether a program or erase operation was completed, and whether the ongoing operation was completed successfully. After writing command 0x70 or 0x71 to the NND_COMMAND_SEC register, an access to the NAND controller access register (NND_ACCESS) drives the content of the NFMC status register on the multiplexed 8-bit bus.

The NFMC remains in read status mode until further commands are issued to it. Therefore, if the status register is read during a random read cycle, a read command (0x00) must be issued before a sequential-page read cycle.

For the read status of a multiplane program or multi-erase operations, the read multiplane status command (71h) must be used rather than 70h. (See Table 7 and Table 8).

Table 7. Status Register Mapping for 32, 64, 128, and 256 Megabits

Bit	Definition	Read Status (70h)
7	Write protect	0: Protected
		1: Not protected
6	Device operation	0: Busy
		1: Ready

5	Reserved	0
4	Reserved	0
3	Reserved	0
2	Reserved	0
1	Reserved	0
0	Program/erase	0: Pass
		1: Fail

Table 8. Status Register Mapping for 512, 1024 Megabits

Bit	Definition	Read Status (70h)	Read Multiplane Status (71h)
7	Write protect	0: Protected	0: Protected
		1: Not protected	1: Not protected
6	Device operation	0: Busy	0: Busy
		1: Ready	1: Ready
5	Reserved	0	0
4	Plane 3 pass/fail	0	0: Pass
			1: Fail
3	Plane 2 pass/fail	0	0: Pass
			1: Fail
2	Plane 1 pass/fail	0	0: Pass
			1: Fail
1	Plane 0 pass/fail	0	0: Pass
			1: Fail
0	Program/erase (all operations)	0: Pass	0: Pass
		1: Fail	1: Fail

2.1.8 Reset Operation

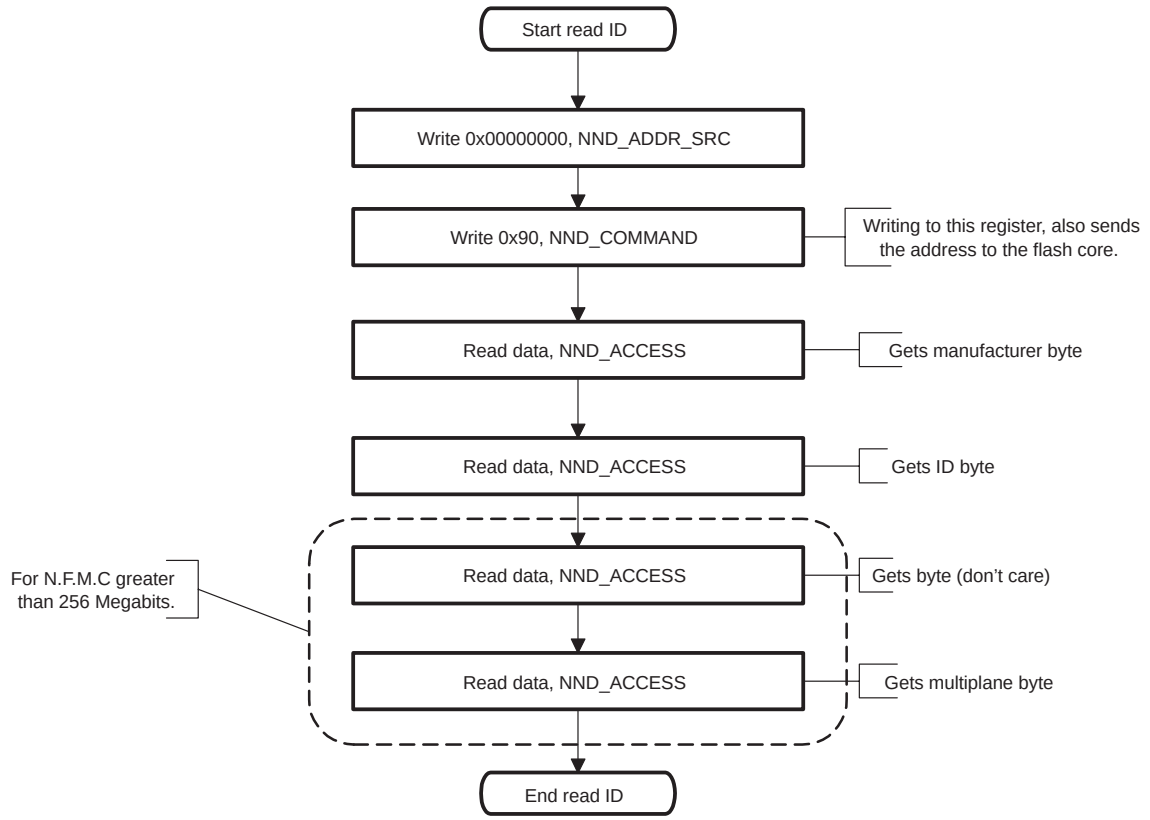
The NFMC has a reset feature executed by writing 0xFF to the command register (NND_COMMAND_SEC) because there is no address to be sent. When the NFMC is in a busy state during the random read program or erase mode, the reset operation aborts these operations. The contents of memory cells being altered are no longer valid, because the data is partially programmed or erased and cannot be ensured. The command register of the NFMC is cleared to wait for the next command. The R/B_ pin goes to low for a certain time depending on the operation (typically 5, 10, or 500 µs for

read/program/erase) after the reset command is written. If the NFMC is already in reset state, the flash memory command register does not accept a new reset command. It is recommended, after a hardware reset, to start any access to the NFMC by sending a reset command.

2.1.9 Read ID Operation

The flash core has a product identification code. To read this code, the command 0x90 is used, followed by an address input of 00h. Depending on the size of flash core, the code can be composed of either 2 or 4 bytes (for the 512M bits and 1G bit). Reading the NFMC sequentially outputs the NMFC manufacturer byte, the NFMC code or ID byte (a unique byte that represents the size of the NFMC), a reserved byte, and the multiplane byte, respectively. By reading the ID of the NFMC, the software determines which features are supported by the NFMC (see Table 12, *Characteristics of Supported NFMCs*). The memory core remains in read ID operation unless a new command code is sent.

Figure 9. Read ID Operation



2.1.10 Error Code Correction

To better protect data, the NFC provides an error code correction (ECC) logic. The algorithm for ECC is the one that Samsung recommends (see Figure 10). It can be used on 256 bytes or 512 bytes (selectable by a control bit in the NND_CTRL register), and can detect errors and correct one bit error. The ECC logic can be used either for read or program operation.

Bit checking and eventual bit repair is done by software after either half-page or page read.

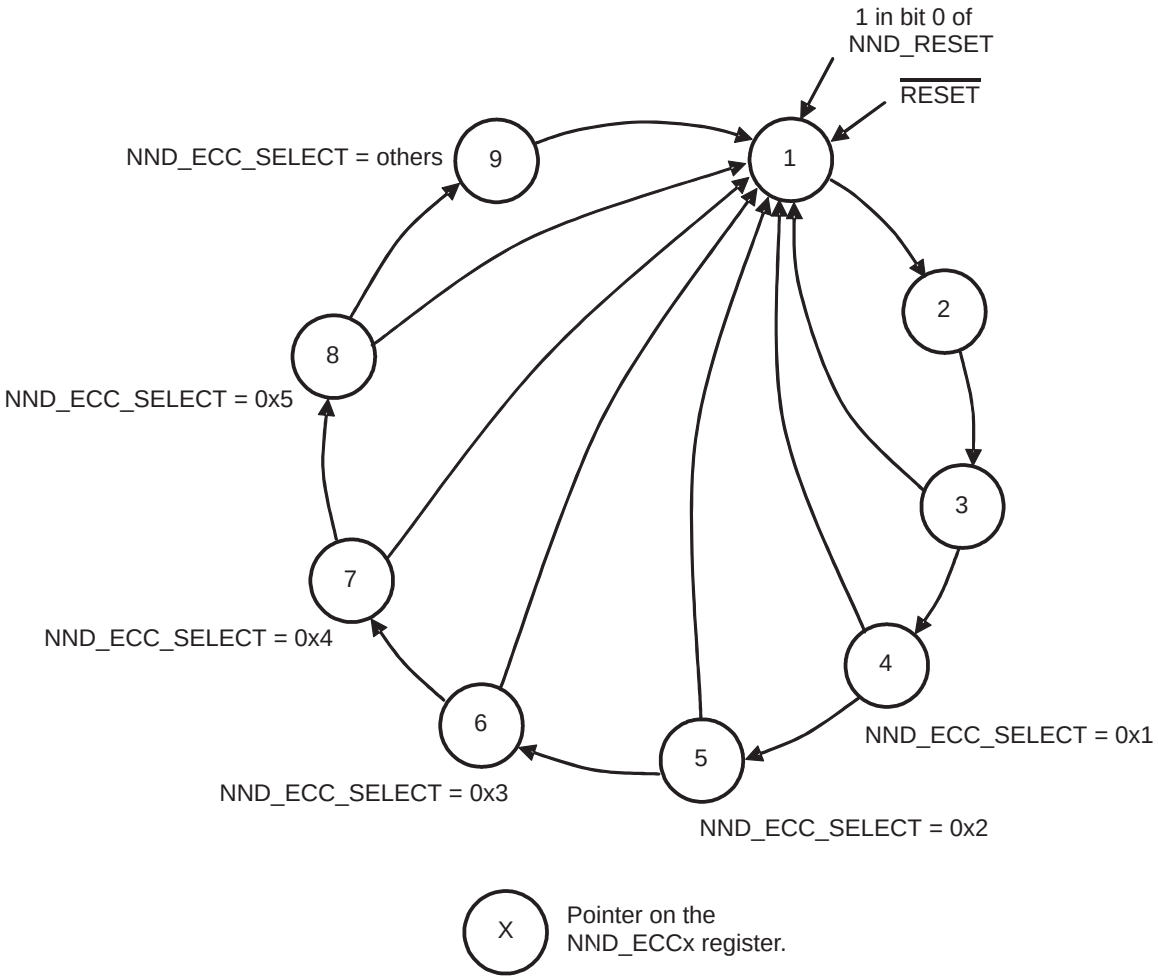
When program operation is used, ECC is normally accumulated. It is the responsibility of the software to read the ECC through dedicated registers (NND_ECCx) via host access and to write them back to the NFMC in the spare area.

For read operation, the ECC is accumulated after each data read. At least a half-page needs to be accessed, and the starting address is the beginning of the half-page. The calculated ECC can be compared with the one that was previously stored in the spare area when the last program operation in that page was performed. The NFC does not assume the physical address of the ECC bytes in the spare byte area.

ECC can be disabled by resetting a bit in the control register (NND_CTRL) of the NFC.

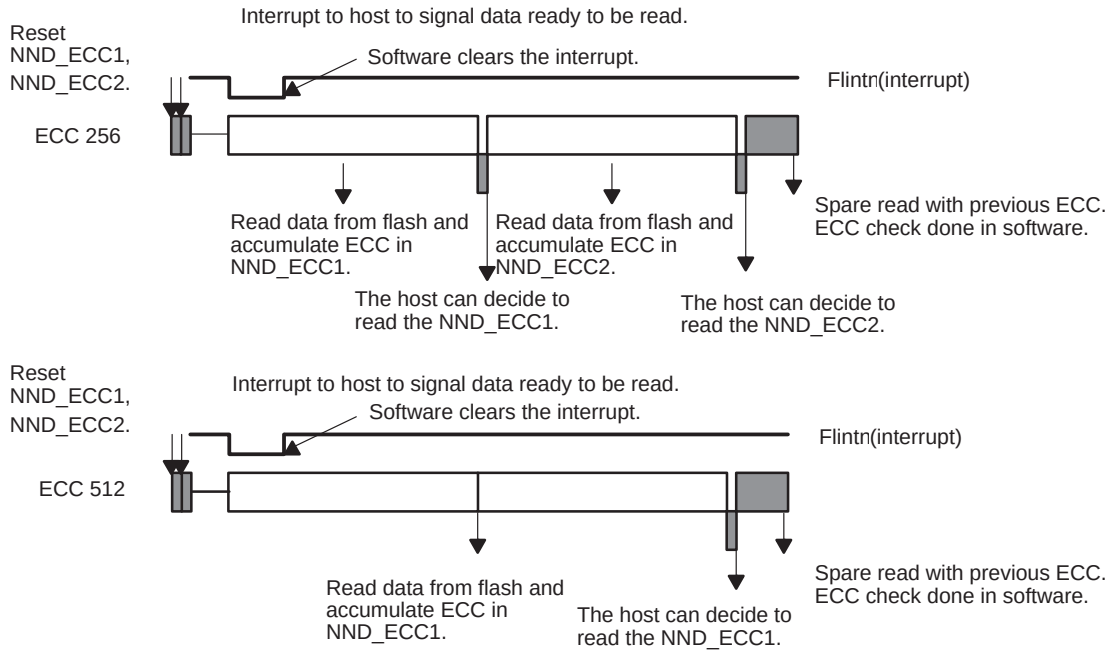
ECC registers are reset when the RESETN pin at the NFC boundary goes low or by writing a bit in the NND_RESET register. The management of the ECC pointer is done automatically by hardware. After a reset, the pointer is set to register 1 (NND_ECC1). Then, when the number of read/writes exceeds the ECC length coded by ECC_256 in the NND_CTRL register of the NFC, the pointer shifts to register 2 (NND_ECC2), next to register 3, and then moves back to the NND_ECC1 register. During ECC computation, if bit ECC_ON is reset, the ECC registers are not updated anymore. When ECC_ON is set to 1, the computation resumes. After a write in the NND_RESET register, the pointer for the ECC is set to NND_ECC1.

Figure 10. ECC Pointer Management



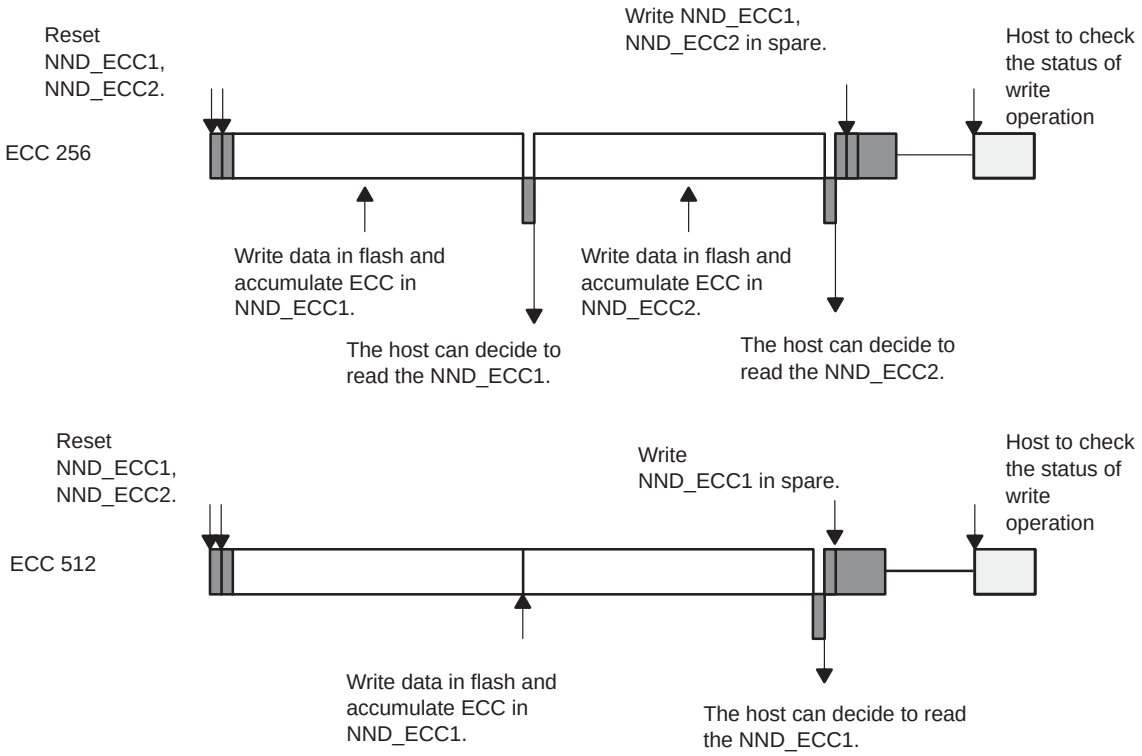
The pointer transition from one state to another is made after either 256 or 512 data has been read (control bit ECC_256 in the NND_STATUS register). After a reset **(**RESETN going low, soft reset, or writing 1 to bit 0 of NND_RESET), the pointer is set to NND_ECC1 register.

Figure 11. Single Page Read Host Mode



Note: In the case of ECC 256, the reading of ECC can also be performe d when the full page has been read.

Figure 12. Single-Page Write Host Mode



Note: In the case of ECC 256, the reading of ECC can also be performed when the full page has been written, and before the ECC registers in the spare area.

It is the responsibility of the software, when program operation is in effect, to read back the ECC registers and to save them in the spare area. The software can also calculate the ECC on a chunk of data. The reading of the ECC registers can be done once a full page has been written.

Once all the parities have been computed, they are stored in the ECC registers under the mapping shown in Table 9.

Table 9. Bit Mapping of ECC Registers

Register Name	7	6	5	4	3	2	1	0
NND_ECCx[31:24]	Reserved	Reserved	Reserved	Reserved	P2048o	P1024o	P512o	P256o
NND_ECCx[23:16]	P128o	P64o	P32o	P16o	P8o	P4o	P2o	P1o
NND_ECCx[15:8]	Reserved	Reserved	Reserved	Reserved	P2048e	P1024e	P512e	P256e
NND_ECCx[7:0]	P128e	P64e	P32e	P16e	P8e	P4e	P2e	P1e

2.1.11 Invalid Block Management

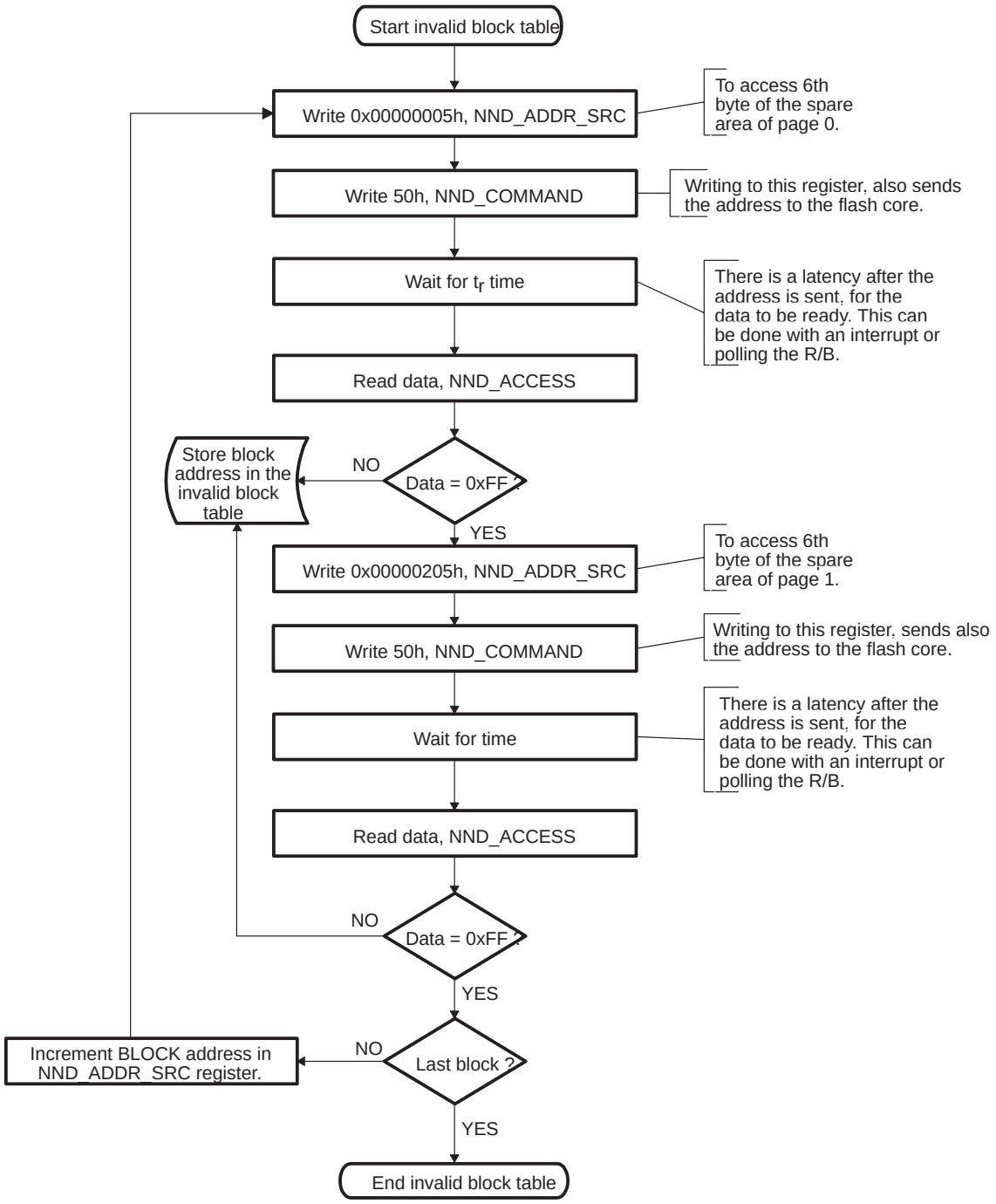
Over time, reading/programming/erasing NFMC can introduce errors in data. ECC can protect the data, but if there is more than one error, ECC cannot correct any errors. Invalid blocks are defined as blocks that contain one or more invalid bits whose reliability is not ensured by the NFMC manufacturer. All NFMC locations are erased (0xFF), except locations where the invalid block information is written before shipping. The invalid block status is defined by the 6th byte in the spare area in page 0 and page 1 (the first 2 pages) of a block. Because the invalid block information is also erasable, it is impossible to recover the information once it has been erased. Therefore, the system must be able to recognize the invalid block based on the original invalid block information and create an invalid block table. If, after an operation, a block becomes corrupted, it must be added to the invalid block table. This table management is done by software. When a page or block becomes corrupted, a value other than 0xFF must be written in the invalid block byte in the spare area. *The invalid block location can depend on the NFMC. For instance, on the 1G-bit mono-die, the invalid block status is defined by the 1st byte of the spare area* (see Figure 13).

If an operation fails, a decision must be made, as shown in Table 10.

Table 10. Decision Table When Operation Fails

Read/Write	Failure	Decision
Write	Erase failure	Write data to a new page. Mark block as invalid.
	Program failure	Write data to a new page. Mark block as invalid.
Read	Single-bit failure	ECC correction
	Multiple-bit failure	Uncorrectable error Mark block as invalid.

Figure 13. Invalid Block Mapping



2.1.12 FIFO (Prefetch and Postwrite)

A FIFO with prefetch and postwrite functions prevents stalling the processor for too many cycles when reading or programming the NFMC. The host either writes or reads this FIFO directly; the NFC state machine handles the actual programming or reading of the NFMC.

Two bits in the NND_CTRL register of the NFC select prefetch or postwrite mode (see Table 11).

Table 11. Prefetch/Postwrite Mode

Prefetch-Postwrite	Selected Mode
0 – 0	Host mode
0 – 1	Postwrite mode
1 – 0	Prefetch mode
1 – 1	Host mode

The prefetch/postwrite can read/write in advance 1, 2, 4, 8, or 16 bytes. The FIFO_SIZE field of the NND_FIFOCTRL register holds one of these values.

The BLOCK_COUNT field of NND_FIFOCTRL register also holds the number of (FIFO_SIZE) chunks of data to read/write from the NFMC. An internal counter is loaded from the NND_FIFOCTRL under certain conditions, but is not accessible from software.

The software must assure coherence between the FIFO_SIZE and BLOCK_COUNT values stored in the NND_FIFOCTRL.

For instance, if FIFO_SIZE equals 16, then the BLOCK_COUNT must be programmed with 32 or 33 (if spare bytes are included).

2.1.13 Prefetch

After the host programs the read address of the page, the read command is sent, and the host enables the prefetch bit. The internal counter is loaded with the BLOCK_COUNT value of the NND_FIFOCTRL register. Interrupt is asserted only if the different contributions of events (READY_EVENT, FIFO_FULL, and so on) are not masked (see Figure 14).

The NFMC enters its busy mode as indicated by the R/B_ pin. When the NFMC is ready, the NFC starts fetching one FIFO_SIZE byte(s) of data and begins to fill the NFC internal FIFO. While the FIFO is being loaded with data from NFMC, any access to data, address, and command registers is stalled.

When the FIFO is full with FIFO_SIZE byte(s), the NFC signals it by asserting low an interrupt (event FIFO_FULL is 1 and MSK_FULL is 1) and the counter is decremented. The host can read the FIFO through the register NND_FIFO access(es). Any read past the last byte of the FIFO returns the last byte of the FIFO.

For instance, if the FIFO_SIZE is 16 and the pointer is on the 14th position of the FIFO, a 32-bit read of the FIFO returns:

		[31:24]	[23:16]	[15:8]	[7:0]
Data read	=	B15	B15	B15	B14

Where B14 was the byte at position 14 in the FIFO, and B15 was the byte at position 15 in the FIFO.

When the FIFO is fully read by the host, the NFC state machine can fill the FIFO again. Active events are cleared by software by writing to the NND_STATUS register. When the internal counter reaches zero, an interrupt is asserted (event COUNT_ZERO is 1 and MSK_COUNT is 1), and the prefetch is stopped. The host can also decide at any time to stop the prefetch by writing a 0 to the prefetch bit.

- ❏ NFC fetches the data and fills the FIFO. (Access to FIFO in that case, is stalled.)
- ❏ When the FIFO is full, the NFC asserts the interrupt (FIFO_FULL event pending and MSK_FULL equals 1) and the internal counter is decremented.
- ❏ By host accesses to NND_FIFO, read the FIFO. The FIFO_FULL event must be cleared by writing to NND_STATUS or a future FIFO_FULL event will not be seen.
- ❏ When the FIFO is empty and the counter is not 0, the NFMC triggers a new prefetch.
- ❏ If the counter is zero, the event COUNT_ZERO is set and interrupt is asserted if MSK_COUNT is set and the prefetch is stopped.
- ❏ The host clears the pending event(s) by writing to NND_STATUS of the NFC.

When prefetch goes from 0 to 1:

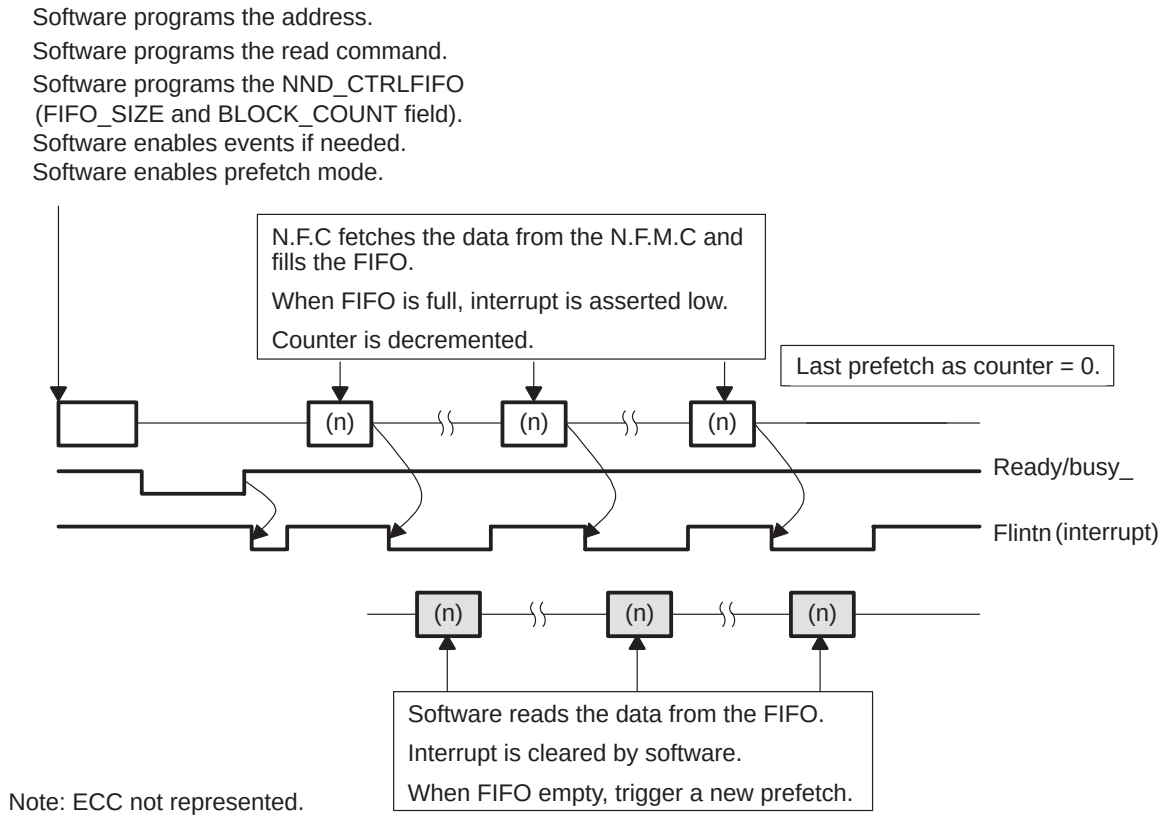
- ❏ The internal counter is loaded with the BLOCK_COUNT value of the NND_FIFOCTRL register.
- ❏ The FIFO is flushed. (FIFO_EMPTY event is set. Depending on the value of MSK_EMPTY, the interrupt is asserted or not.)

When prefetch goes from 1 to 0:

- Prefetch is aborted.

Before accessing a new page, software must successively write 0 and then 1 in the prefetch bit. At reset, the prefetch bit is set to 0.

Figure 14. Single Page Read in Prefetch Mode



Note: ECC not represented.

2.1.14 Postwrite

After the host programs the address of the page to program, the command program is sent, and the host enables the postwrite bit. The internal counter is loaded with the BLOCK_COUNT value of the NND_FIFOCTRL register (see Figure 15).

The software can write a FIFO_SIZE byte(s) in the FIFO of the NFC by accessing NND_FIFO. When the FIFO is full, the NFC sends the data to the NFMC, asserts the interrupt (if event FIFO_FULL is unmasked), and the counter is decremented. When all of the data is sent, the FIFO is empty and

the NFC asserts low the interrupt (if event FIFO_EMPTY is unmasked). The CPU refills the FIFO and clears the interrupt. When the counter reaches zero, the interrupt is also asserted (if event COUNT_ZERO is unmasked). To finish programming the NFMC, the command 0x10 must be sent.

- The host writes the data in the FIFO of the NFC. Once the FIFO is full, all access is stalled.
- When the FIFO is full, the event FIFO_FULL is set, the NFC sends the data from the FIFO to the NFMC, and the internal counter is decremented.
- When the FIFO is empty, the event FIFO_EMPTY is set, and interrupt is asserted if unmasked (active low).
- When the FIFO is empty and the counter is not zero, the host writes a new batch of data in the FIFO of the NFC.
- If the counter is zero, the event COUNT_ZERO is set and the interrupt is asserted if unmasked, the NFC sends the data from the FIFO to the NFMC, and the postwrite mechanism is stopped.
- The host clears the pending event(s) by writing to NND_STATUS of the NFC.

While the FIFO is being filled, access to command, data, and address registers is stalled.

When postwrite goes from 0 to 1:

- The FIFO is flushed. (FIFO_EMPTY event is set. Depending on the value of MSK_EMPTY, the interrupt is asserted or not.)
- The internal counter is loaded with the BLOCK_COUNT value of the NND_FIFOCTRL register.

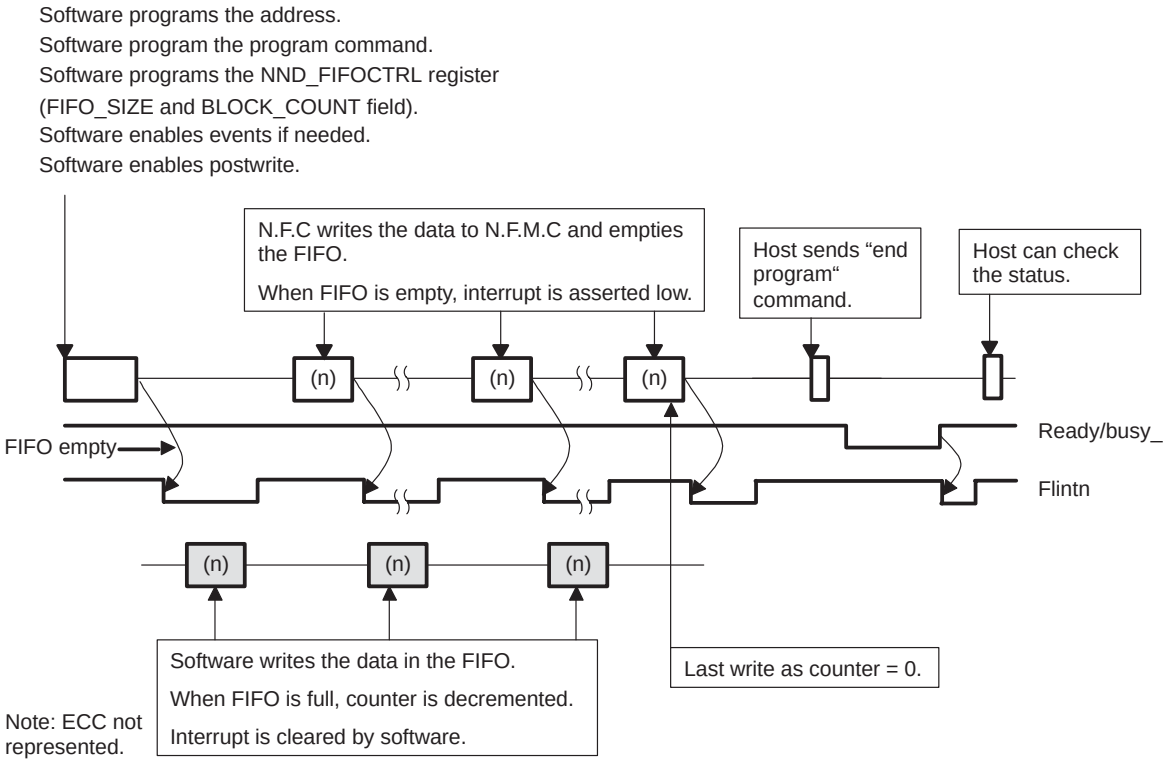
When the postwrite goes from 1 to 0:

- Postwrite is aborted.

Any write past the last byte of FIFO is discarded. For instance, if the FIFO_SIZE is 16 and the pointer is on the 14th position of the FIFO, a 32-bit write of a data equal to X3X2X1X0 is placed on the FIFO :

	13th	**14th**	**15th**	**16th**
Be = 0	data	X0	X1	X2
Be = 1	data	X3	X2	X1

Figure 15. Single-Page Program in Postwrite Mode



2.1.15 DMA Support

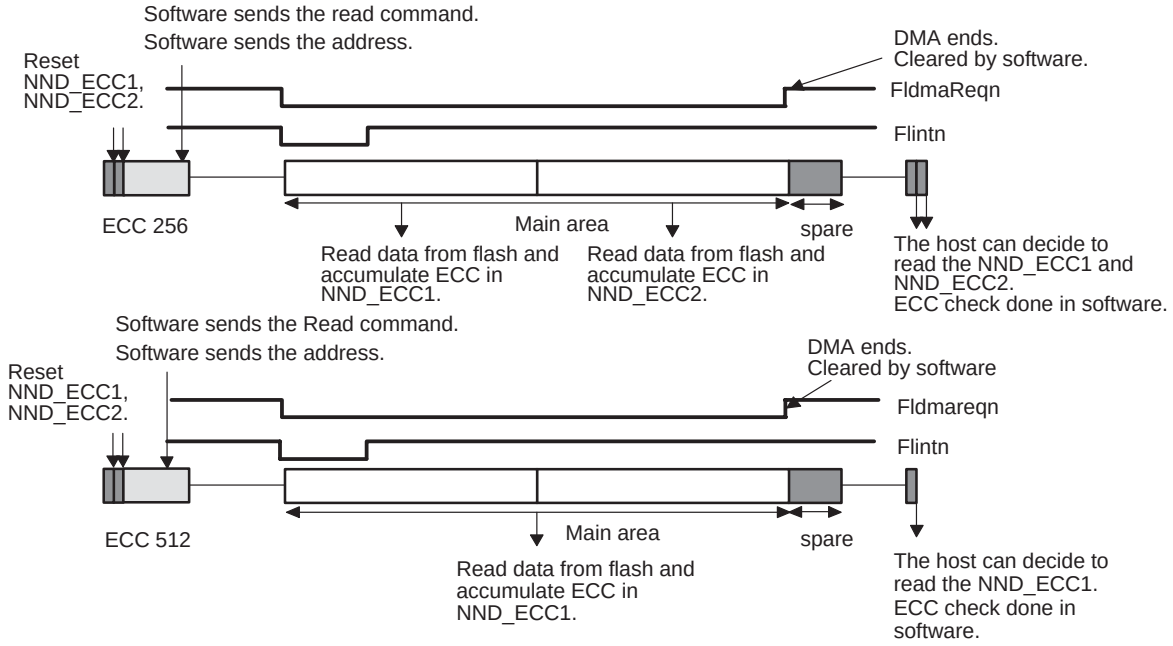
The NFC supports DMA read and write on one page in the host and FIFO prefetch and postwrite) mode.

2.1.16 Host Mode

Read Operation

The DMA request is asserted low after the latency time necessary for the NFMC to transfer data from the memory cell to its internal register. Ready/Busy_ goes low and transitions to high to indicate data ready. By writing to the ResetDMASynchro of the NND_RESET register, the DMA request is asserted high (see Figure 16).

Figure 16. Single Page Read DMA



Write Operation

No DMA signal is generated in DMA program operation.

2.1.17 FIFO mode

Prefetch

After the host programs the address of the page to read, and the command read is sent, the host enables the FIFO prefetch bit and enables DMA in system DMA. The internal counter is loaded with the BLOCK_COUNT value of the NND_FIFOCTRL register, and the DMA request is driven high (see Figure 17).

The NFMC enters its busy mode as indicated by the R/B_ terminal. When the NFMC is ready, the NFC starts fetching one FIFO_SIZE byte(s) of data and begins to fill the NFC internal FIFO. While the FIFO is being loaded with data from NFMC, access to data, address, and command registers is stalled.

When the FIFO is full with FIFO_SIZE byte(s), the NFC signals it by asserting low the DMA request, and the internal counter is decremented by one. The DMA can read the FIFO through access(es) to the NND_FIFO register. Any

read past the last byte of the buffer returns the last byte of the FIFO. When the FIFO is fully read, the DMA request is asserted high and the NFC state machine can fill the FIFO again. When the internal counter reaches zero, the FIFO is filled a last time, and when the FIFO is full, the DMA request is asserted low. DMA can read the FIFO a last time, and when it is empty, the DMA request is asserted high. Because the counter has reached zero, however, a new prefetch is not triggered. The host can also decide at any time to stop the prefetch by writing a 0 to the prefetch bit.

- The host enables the prefetch bit. DMA request is asserted high.
- NFC fetches the data and fills the FIFO (access to the FIFO, in that case, is stalled)
- When the FIFO is full, the NFC asserts low the DMA request and the internal counter is decremented.
- The DMA reads the FIFO.
- When the FIFO is read and is empty, the DMA request is asserted high. If the counter is not zero, a new prefetch is triggered.
- If the counter is zero, the prefetch is stopped.
- The prefetch bit is disabled if necessary.

When prefetch goes from 0 to 1:

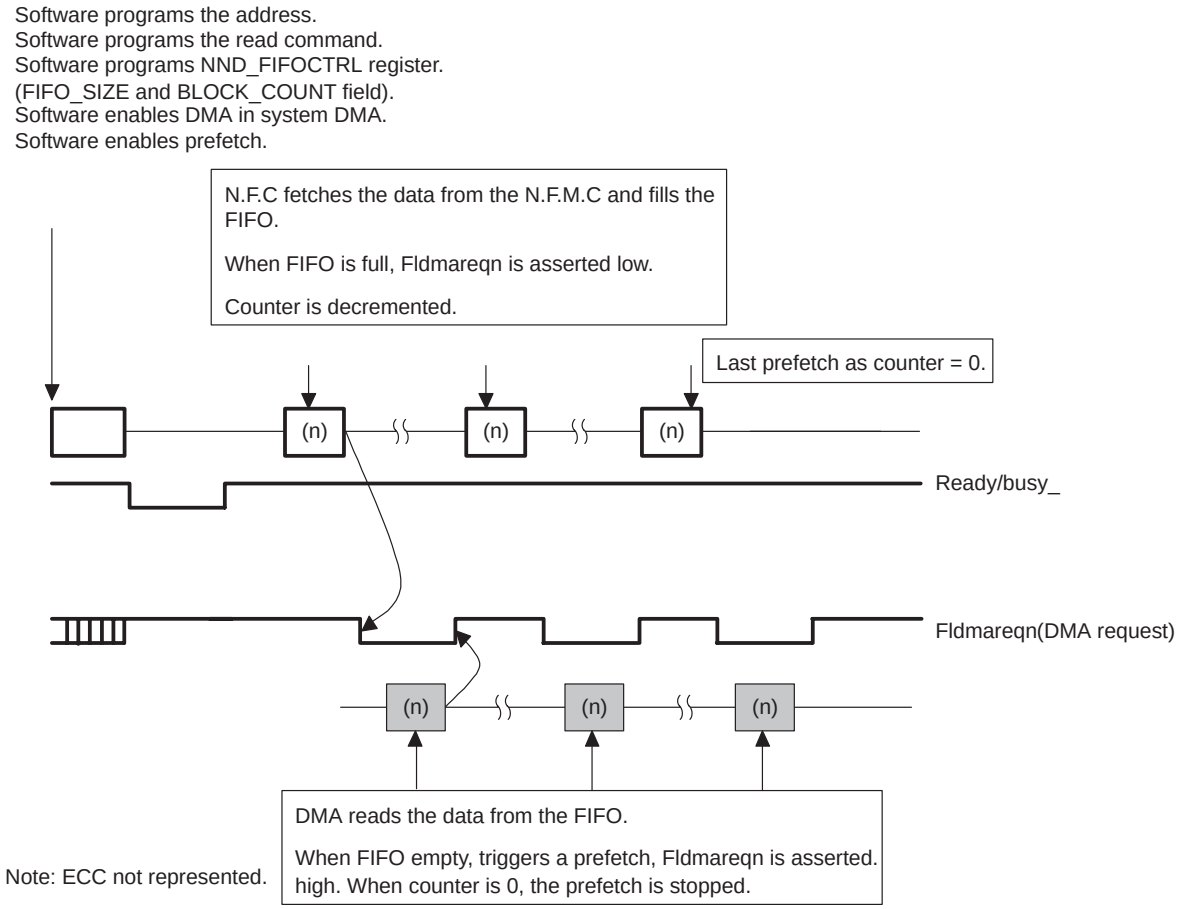
- The internal counter is loaded with the BLOCK_COUNT field of the NND_FIFOCTRL.
- The FIFO is flushed.
- The DMA request is asserted to high.

When prefetch goes from 1 to 0:

- Prefetch is aborted.

Before accessing a new page, software must successively write 0 and then 1 in the prefetch bit. At reset, the prefetch bit is set with 0.

Figure 17. Single-Page Read DMA in Prefetch Mode



Postwrite

After the host programs the address of the page to program, the program command is sent, and the host enables the FIFO postwrite mode and DMA in system DMA. The internal counter is loaded with the BLOCK_COUNT value of the NND_FIFOCTRL register and the DMA request is driven low (see Figure 18).

The DMA can write a FIFO_SIZE byte(s) in the FIFO of the NFC. When the FIFO is full, the DMA request is asserted high and the counter is decremented by one. The NFC sends the data to the NFMC. When all of the data is sent to the NFMC, the FIFO is empty, and the NFC asserts low the DMA request. The DMA can refill the FIFO. When the counter is zero, the DMA request is not asserted any longer, and the NFC sends the data from the FIFO to the NFMC

for the last time. To finish the programming of the NFMC, the command 0x10 must be sent.

- DMA writes the data in the FIFO: Once the FIFO is full, any access is stalled.
- When the FIFO is full, the internal counter is decremented, and the NFC sends the data from the FIFO to the NFMC.
- When the FIFO is empty, the DMA request is asserted low.
- When the FIFO is empty and the counter is not zero, the DMA writes a new batch of data in the FIFO.
- If the counter is 0, the NFC sends the data from the FIFO to the NFMC for the last time and the postwrite mechanism is stopped. No DMA request is asserted.

While the FIFO is being filled, access to command, data, and address registers is stalled.

When postwrite goes from 0 to 1:

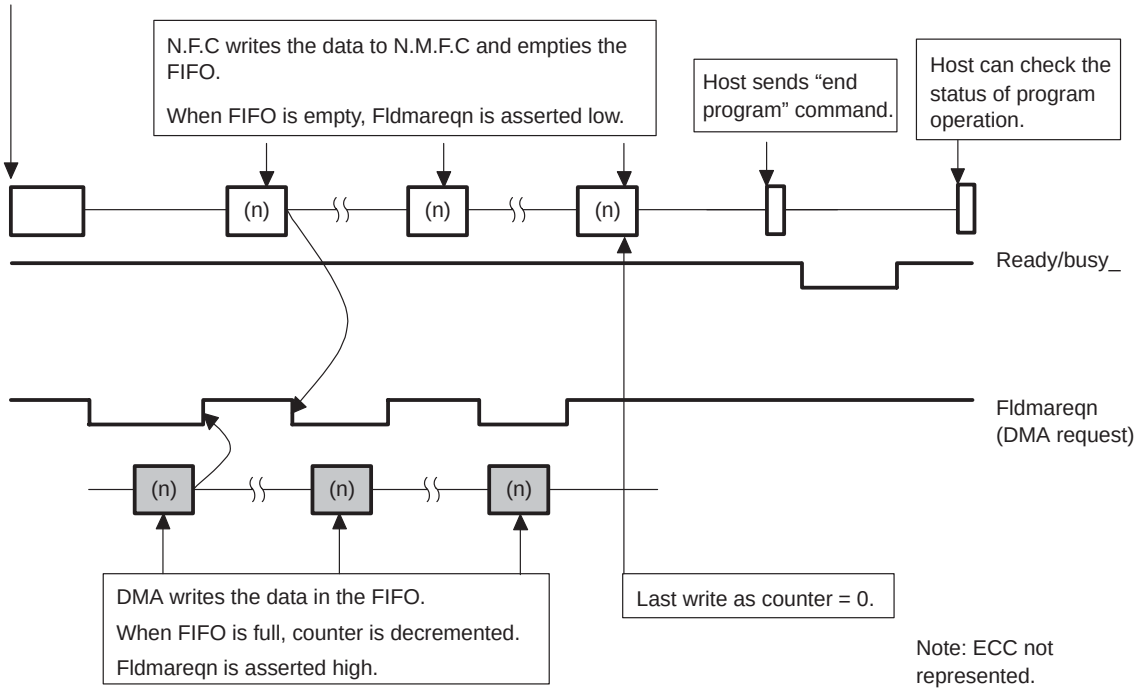
- The DMA request is asserted low.
- The FIFO is flushed.
- The internal counter is loaded with the BLOCK_COUNT value of the NND_FIFOCTRL register.

When the postwrite goes from 1 to 0:

- The next postwrite is aborted.
- The DMA request is asserted to high.

Figure 18. Single-Page Program DMA in Postwrite Mode

Software programs the address.
Software programs the program command.
Software programs the NND_FIFOCTRL register (FIFO_SIZE and BLOCK_COUNTER field).
Software enables events if needed.
Software enables DMA in system DMA.
Software enables postwrite.



2.1.18 NAND Flash Memory Core Support

Table 12 summarizes the characteristics of supported NFMCs.

Table 12. Characteristics of Supported NFMCs

	1Gb/ 128 MB	512Mb/ 64 MB	256Mb/ 32 MB	128Mb/ 16 MB	64Mb/ 8 MB	32Mb/ 4 MB
Number of pages	262144	131072	65536	32768	16384	8192
Page size+ spare (bytes)	512+16	512+16	512+16	512+16	512+16	512+16
Number of blocks	8192	4096	2048	1024	1024	512

Table 12. Characteristics of Supported NFMCs (Continued)

	1Gb/ 128 MB	512Mb/ 64 MB	256Mb/ 32 MB	128Mb/ 16 MB	64Mb/ 8 MB	32Mb/ 4 MB
Number of pages per block	32	32	32	32	16	16
Block size (bytes)	16K+512	16K+512	16K+512	16K+512	8K+256	8K+256
Addresses	4 cycles	4 cycles	3 cycles	3 cycles	3 cycles	3 cycles
	a7-a0	a7-a0	a7-a0	a7-a0	a7-a0	a7-a0
	a16-a9	a16-a9	a16-a9	a16-a9	a16-a9	a16-a9
	a24-a17	a24-a17	a24-a17	a23-a17	a22-a17	a21-a17
	a25-a26	a25				
ID	79	76	75	73	E6	E3
Samsung code	EC	EC	EC	EC	EC	EC
Toshiba code	98	98	98	98	98	98
Fujitsu code	04	04	04	04	04	04

Table 13 summarizes the type of operations supported by the NFMCs.

Table 13. Supported Operations on NFMCs

Supported Operations	1Gb/ 128MB	512Mb/ 64MB	256Mb/ 32MB	128Mb/ 16MB	64Mb/ 8MB	32Mb/ 4MB
Read 1	P	P	P	P	P	P
Read 2	P	P	P	P	P	P
Read ID	P	P	P	P	P	P
Reset	P	P	P	P	P	P
Page program	P	P	P	P	P	P
Page program multiple	P	P	-	-	-	-
Copy-back program	P	P	P	-	-	-
Copy-back program multiple	P	P	-	-	-	-
Block erase	P	P	P	P	P	P

Table 13. Supported Operations on NFMCs (Continued)

Supported Operations	1Gb/ 128MB	512Mb/ 64MB	256Mb/ 32MB	128Mb/ 16MB	64Mb/ 8MB	32Mb/ 4MB
Block erase multiple	P	P	-	-	-	-
Read status	P	P	P	P	P	P
Read status multiple	P	P	-	-	-	-

Note: Note: This table summarizes the supported operations for the current NFMCs. These operations may change in the future when new NFMCs are introduced in the market. To program new NFMCs, refer to the NFMC specifications from vendors.

2.1.19 NAND Flash Registers

For the mapping of the register, the representation below is used. Each register is 32 bits wide, and for a register the most significant word16 is above the least significant word16.

Table 14. Register Mapping

Addr	Name	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	Reg1	Reg1_value (most significant word16)															
0x00	Reg1	Reg1_value (least significant word16)															
0x04	Reg2	Reg2_value (most significant word16)															
0x04	Reg2	Reg2_value (least significant word16)															

Table 15 lists the NAND flash registers. Table 16 through Table 46 describe the register bits.

Table 15. NAND Flash Registers

Base Address = 0xFFFB CC00		
Register	**Description**	**Offset**
NND_REVISION	Indicates the current revision number of the NFC	0x00
NND_ACCESS	Indicates the location where a read or program(write) operation is to be performed	0x04
NND_ADDR_SRC	Contains the byte address of the location in the NFMC from which data is read or written by accessing the NAND controller access register (NND_ACCESS).	0x08
RESERVED	Reserved	0x0C

Note: All reserved bits must be written with 0.

Table 15. NAND Flash Registers (Continued)

Register	Description	Offset
NND_CTRL	NAND controller	0x10
NND_MASK	Used to mask event sources	0x14
NND_STATUS	Used to mask event sources	0x18
NND_READY	Used to poll the readiness of the NFMC	0x1C
NND_COMMAND	Used to write a specific command to the NFMC	0x20
NND_COMMAND_SEC	Writing to this register does not send any address to the NFMC	0x24
NND_ECC_SELECT	Defines the number of NND_ECC registers per enable	0x28
NND_ECC1 to NND_ECC9	These registers hold the ECC code calculated while reading/writing the NFMC	0x2C to 0x4C
NND_RESET	NAND controller reset register	0x50
NND_FIFO	Used to access the FIFO when prefetch or postwrite mode is selected	0x54
NND_FIFOCTRL	Holds the size of the FIFO and the number of blocks of (FIFO_SIZE) bytes to fetch/write to access a full page	0x58
NND_PSC_CLK	The NFC uses the 4-bit value in this register to divide the interface clock to adjust the timing for the signals on the NFMC.	0x5C
NND_SYSTEST	Used to test some features of the NFC	0x60
NND_SYSCFG	NAND controller system configuration	0x64
NND_SYSSTATUS	NAND controller system status	0x68
NND_FIFOTEST1	NAND controller FIFO test register 1	0x6C
NND_FIFOTEST2	NAND controller FIFO test register 2	0x70
NND_FIFOTEST3	NAND controller FIFO test register 3	0x74
NND_FIFOTEST4	NAND controller FIFO test register 4	0x78

Note: All reserved bits must be written with 0.

Table 16. NAND Controller Revision Register (NND_REVISION)

Bit	Name	Description
31-8	Reserved	Reserved
7-0	NND_REVISION	Revision number

The NND_REVISION field indicates the current revision number of the NFC. This value is fixed by hardware.

- The 4 LSB indicate a minor revision.
- The 4 MSB indicate a major revision.
  - 0x01: Revision 0.1
  - 0x02: Revision 0.2
  - 0x21: Revision 2.1

Hardware reset and software reset do not act on this register.

Table 17. NAND Controller Access Register (NND_ACCESS)

Bit	Name	Description	Reset Value	Type
31-0	Data	Data read/write access to external NAND flash	0	RW

An access to this location performs the read or program(write) operation, depending on the value of the MCMD signal. For program operation, data present on the MDATA bus is written in the NFMC.

For read operation, data is fetched from the NFMC and driven on the sdata bus. 8-bit, 16-bit, and 32-bit accesses are supported, but bytes are serialized from/to NFC to/from NFMC. A 4 x 8-bit buffer in the NFC packs the bytes coming from/to the NFMC.

When no operation is selected, the value 0x00000000 is driven on the SDATA bus.

Table 18. NAND Controller Address Register (NND_ADDR_SRC)

Bit	Name	Description	Reset Value	Type
31-0	Adress	External NAND flash start address	0	RW

This register contains the byte address of the location in the NFMC. From this location, the data is read or written by accessing the NAND controller access register (NND_ACCESS). The address in this register represents the starting address and thus is never incremented. For multiple reading and writing, the NFMC has a pointer that is incremented.

This register also holds the block erase address, which must be properly formatted before being written in this register (see Table 5). Because an access to NND_COMMAND also sends this address to the NFMC, the software first must write the new address in the NND_ADDR_SRC, and then write a command through NND_COMMAND. This address remains unchanged even if a read and a program operation are performed. Note that the address formatting depends on whether bit A8 of NND_CTRL is set or not. The command codes 0x00, 0x01, or 0x50 select which area of the NFMC to access.

Table 19. Address Decomposition

Size of Flash Core (MB)	Block Address	Page Address in Block	Start Address in Page
32	A21-A13	A12-A9	A7-A0
64	A22-A13	A12-A9	A7-A0
128	A23-A14	A13-A9	A7-A0
256	A24-A14	A13-A9	A7-A0
512	A25-A14	A13-A9	A7-A0
1024	A26-A14	A13-A9	A7-A0

When a page is selected, sending the command 0x00, 0x01, or 0x50 distinguishes among areas A, B, or C (spare).

Table 20. NAND Controller Control Register (NND_CTRL)

Bit	Name	Description
31-18	Reserved	Reserved.
17	PREFETCH	When 1, prefetch mode is enabled.
16	POSTWRITE	When 1, postwrite mode is enabled.
15	WRITEPROT3	WriteProtect3. When 0, the voltage generator is reset.
14	CHIPEN3	ChipEnable3. When 0, NFMC device is selected.
13	WRITEPROT2	WriteProtect2. When 0, the voltage generator is reset.
12	CHIPEN2	ChipEnable2. When 0, NFMC device is selected.
11	WriteProt1	WriteProtect1. When 0, the voltage generator is reset.
10	CHIPEN1	ChipEnable1. When 0, NFMC device is selected.
9	WRITEPROT0	WriteProtect0. When 0, the voltage generator is reset.

Table 20. NAND Controller Control Register (NND_CTRL) (Continued)

Bit	Name	Description
8	CHIPEN0	ChipEnable0. When 0, NFMC device is selected.
7	Reserved	Reserved
6	ADRCNT[1]	Address counter for sending bytes to NFMC
5	ADRCNT[0]	Address counter for sending bytes to NFMC
4	A8	When 0, bit A8 of address register is not sent to NFMC.
3	BE	Switch big/little endian. 1: big endian, 0: little endian
2	Reserved	Reserved
1	ECC_256	If 1, ECC is calculated on 256 bytes, else 512 bytes.
0	ECC_ON	If 1, ECC logic is enabled.

Bit 0: When set, the ECC logic is enabled and for read or program operation, ECC is computed and stored in the NND_ECCx registers. When ECC_ON is set, it is impossible to write to the ECC_256 bit to avoid corrupting the ECC logic.

Table 21. ECC Bits Operation

ECC_ON	ECC_256	Effect
0	Don’t care	No change (previous value)
1	1	ECC calculated on 256 bytes
1	0	ECC calculated on 512 bytes

Bit 1: When set, ECC is accumulated on a chunk of 256 bytes or 512 bytes. The algorithm is the same for both; only two more parities (P2048 and P2048’) are calculated. This bit must be set before enabling the ECC logic. Depending on the value of ECC_ON (bit 0), two accesses may be needed to change the size of the ECC calculation.

Bit 2: Reserved

Bit 3: To select between big and little endian. Depending on the access (32-/16-/8-bit) type and this bit, the NFC packs bytes differently. For example, assume that the NAND flash memory core contains:

- B0 at address n
- B1 at address n+1
- B2 at address n+2
- B3 at address n+3

Table 22 shows the resulting read data from the NAND flash memory core.

Table 22. Byte Packing Function of MBYTEEN and Little/Big Endianism (NND_ACCESS/NND_FIFO)

MBYTEEN	BE = 0				BE = 1			
	[31:24]	[23:16	[15:8]	[7:0]	[31:24]	[23:16	[15:8]	[7:0]
1111	B3	B2	B1	B0	B0	B1	B2	B3
0011	0	0	B1	B0	0	0	B0	B1
1100	B1	B0	0	0	B0	B1	0	0
0001	0	0	0	B0	0	0	0	B0
0010	0	0	B0	0	0	0	B0	0
0100	0	B0	0	0	0	B0	0	0
1000	B0	0	0	0	B0	0	0	0

For the NND_ACCESS/NND_FIFO register (program operation):

Other combinations of MBYTEEN are not permitted.

For all other registers:

The BE bit has no effect on other NFC registers. Only MBYTEEN bits apply.

Table 23. Byte Packing Function of MBYTEEN for Registers (Except NND_ACCESS/NND_FIFO).

MBYTEEN	Registers			
	[31:24]	[23:16]	[15:8]	[7:0]
1111	DATA3	DATA2	DATA1	DATA0
0011	0	0	DATA1	DATA0
1100	DATA3	DATA2	0	0
0001	0	0	0	DATA0
0010	0	0	DATA1	0
0100	0	DATA2	0	0
1000	DATA3	0	0	0

Bit 4: For the current NFMCs, bit 8 of the address is not sent to NFMC, because its functionality is replaced by a pointer set by command 0x00, 0x01, or 0x50.

Future NFMCs have a different page addressing configuration and bit 8 is part of the address. When control bit A8 is 0, bit 8 of NND_ADDR_SRC is not sent to NFMC. When control bit A8 is 1, bit 8 of NND_ADDR_SRC is sent to NFMC.

Table 24. 1 Gigabit Dual-Die Layout of Bytes Sent

	I/O 7	I/O 6	I/O 5	I/O 4	I/O 3	I/O 2	I/O 1	I/O 0
1st cycle	A7	A6	A5	A4	A3	A2	A1	A0
2nd cycle	A16	A15	A14	A13	A12	A11	A10	A9
3rd cycle	A24	A23	A22	A21	A20	A19	A18	A17
4th cycle	0	0	0	0	0	0	A26	A25

Table 25. 1 Gigabit Mono-Die Layout of Bytes Sent

	I/O 7	I/O 6	I/O 5	I/O 4	I/O 3	I/O 2	I/O 1	I/O 0
1st cycle	A7	A6	A5	A4	A3	A2	A1	A0
2nd cycle	0	0	0	0	A11	A10	A9	A8
3rd cycle	A19	A18	A17	A16	A15	A14	A13	A12
4th cycle	A27	A26	A25	A24	A23	A22	A21	A20

Table 26. NND_ADDR_SRC Decompostion of Flash Bus Function of Control Bit A8

Control Bit A8	Cycle	NND_ADDR_SRC Decomposition on Flash Bus
0	1st cycle	NND_ADDR_SRC[7:0]
	2nd cycle	NND_ADDR_SRC[16:9]
	3rd cycle	NND_ADDR_SRC[24:17]
	4th cycle	0-NND_ADDR_SRC[31:25]
1	1st cycle	NND_ADDR_SRC[7:0]
	2nd cycle	NND_ADDR_SRC[15:8]
	3rd cycle	NND_ADDR_SRC[23:16]
	4th cycle	NND_ADDR_SRC[31:24]

Bits 6-5: This counter sends bytes from the NND_ADDR_SRC to the NFMC. Extra address bytes are usually discarded by the NFMC logic, but this is a protection in case the NFMC logic does not do it.

Table 27. Address Counter for Sending Bytes

ADRCNT[1]-ADRCNT[0]	Number of Byte Address Sent
0-0	4 bytes sent (with least significant byte first)
0-1	3 bytes sent (with least significant byte first)
1-0	2 bytes sent (with least significant byte first)
1-1	1 byte sent (with least significant byte first)

Bit 7: Reserved

Bit 8: The ChipEn0 directly controls the NFMC selection control. When ChipEn0 is high, access to the NFMC is not allowed.When ChipEn0 goes high during a read operation, the NFMC returns to standby mode. However, when the NFMC is in a busy state during a program or an erase operation, the NFMC ignores ChipEn0 and does not return to standby mode. *Because software has direct access to the ChipEn0 signal, it must be set to low when access is performed and not set back to high before all operations to the NFMC are terminated.*

Bit 9: The WriteProt0 bit provides inadvertent write/erase protection during power transitions. The NFMC internal high voltage generator is reset when the WriteProt0 signal is low. This signal is low during reset and software must write a 1 in this location before any access to the NFMC.

Bit 10: The ChipEn1 directly controls the NFMC selection control. When ChipEn1 is high, access to the NFMC is impossible. When ChipEn1 goes high during a read operation, the NFMC returns to standby mode. However, when the NFMC is in a busy state during a program or an erase operation, the NFMC ignores ChipEn1 and does not return to standby mode. *Because software has direct access to ChipEn1 signal, it must be set to low when access is performed and not set back to high before all operations to the NFMC are terminated.*

Bit 11: The WriteProt1 bit provides inadvertent write/erase protection during power transitions. The NFMC internal high voltage generator is reset when the WriteProt1 signal is low. This signal is low during reset, and software must write a 1 in this location before any access to the NFMC.

Bit 12: The ChipEn2 directly controls the NFMC selection control. When ChipEn2 is high, access to the NFMC is impossible. When ChipEn2 goes high during a read operation, the NFMC returns to standby mode. However, when the NFMC is in a busy state during a program or an erase operation, the NFMC ignores ChipEn2 and does not return to standby mode. *Because software has*

direct access to ChipEn2 signal, it must be set to low when access is performed and not set back to high before all operations to the NFMC are terminated.

Bit 13: The WriteProt2 bit provides inadvertent write/erase protection during power transitions. The NFMC internal high voltage generator is reset when the WriteProt2 signal is low. This signal is low during reset, and software must write a 1 in this location before any access to the NFMC.

Bit 14: The ChipEn3 directly controls the NFMC selection control. When ChipEn3 is high, access to the NFMC is impossible. When ChipEn3 goes high during a read operation, the NFMC returns to standby mode. However, when the NFMC is in a busy state during a program or an erase operation, the NFMC ignores ChipEn3 and does not return to standby mode. *Because software has direct access to ChipEn3 signal, it must be set to low when access is performed and not set back to high before all operations to the NFMC are terminated.*

Bit 15: The WriteProt3 bit provides inadvertent write/erase protection during power transitions. The NFMC internal high voltage generator is reset when the WriteProt3 signal is low. This signal is low during reset, and software must write a 1 in this location before any access to the NFMC.

Bit 16: Postwrite bit. Writing 1 to this bit enables the postwrite mechanism. Access to the internal FIFO is possible. The host or DMA writes data in the FIFO, and the NFC is responsible for unloading the FIFO to the NFMC.

Bit 17: Prefetch bit. Writing 1 to this bit enables the prefetch mechanism. Access to the internal FIFO is possible. The NFC fetches data from the NFMC and fills the FIFO. The host or DMA reads data from the FIFO.

Bits 31-18: Reserved

Table 28. NAND Controller Mask Register (NND_MASK)

Bit	Name	Description
31-4	Reserved	Reserved
3	MSK_EMPTY	Mask. When 0, event for FIFO empty is masked.
2	MSK_FULL	Mask. When 0, event for FIFO full is masked.
1	MSK_COUNT	Mask. When 0, event for counter reaching zero is masked.
0	MSK_READY	Mask. When 0, event for data ready is masked.

This register masks event sources. At reset, all event sources are masked. To unmask an event, a 1 must be written in the bit of the event to be unmasked.

These masks are ANDed with the corresponding bits in the NND_STATUS register. Then, all of these contributions are ORed again to form the global interrupt at the boundary of the NFC.

Table 29. NAND Controller Status Register (NND_STATUS)

Bit	Name	Description
31-4	Reserved	Reserved
3	FIFO_EMPTY	When FIFO is empty, event is pending and this bit is 1.
2	FIFO_FULL	When FIFO is full, event is pending and this bit is 1.
1	COUNT_ZERO	When internal counter reaches zero, event is pending and this bit is 1.
0	READY_ EVENT	When R/B_ goes from 0 to 1, event is pending and this bit is 1.

All pending events sources are active high. Before they can be active, events must be unmasked by writing a 1 in the corresponding mask in the NND_MASK. For example, if READY_EVENT is 1 (pending interrupt) and MSK_READY is 1, then the interrupt at the boundary of the NFC is active low. A reset (resetn going low) or a soft reset clears all interrupts. To clear a pending event, a 1 must be written in the corresponding event source. When there is an interrupt, the software can read this register to determine which event is active.

- Bit 0: This READY_EVENT bit can be masked with the corresponding MSK_EVENT bit in the NND_MASK register. This bit reflects the status of an active event in different cases:
  - For a read operation after the address has been sent to the NFMC, there is a latency time before the data is ready. This READY_EVENT bit is set when the Ready/Busy_ pin goes back to high, indicating that data can be read.
  - For a program operation, the READY_EVENT bit is set when the programming is finished.
  - For an erase operation, the READY_EVENT bit is set when the NFMC is finished erasing the selected block.

It is the responsibility of the software to clear this bit, as described in Table 30.

Table 30. How to Clear Pending Event

Previous Event State	Write	Next Event State
0	1 or 0	0
1	1	0
1	0	1

- Bit 1: When the NFC is either in prefetch or postwrite mode, an interrupt is generated when the internal counter reaches zero. This interrupt is cleared by software according to the scheme shown in Table 30. The interrupt is active if unmasked.
- Bit 2: When the FIFO is full, the FIFO_FULL event goes to 1 and an interrupt is asserted if unmasked. The interrupt is cleared by software according to the scheme shown in Table 30.
- Bit 3: When the FIFO is empty, the FIFO_EMPTY goes to 1 and an interrupt is asserted if unmasked. The interrupt is cleared by software according to the scheme shown in Table 30.
- Bits 31-4: Reserved

Table 31. NAND Controller Ready Register (NND_READY)

Bit	Name	Description	Reset	Type
31-1	Reserved	Reserved		
0	READY	Ready. When 1, NFMC is ready for next operation.		

- Bits 31-1: Reserved
- Bit 0: This bit is the copy of the R/B_ signal coming from the NFMC resynchronized inside the NFC. This bit polls the readiness of the NFMC. *It is impossible to write to this bit.* Its value at reset is the sampled value of the R/B_ pin.

Table 32. NAND Controller Command Register (NND_COMMAND)

Bit	Name	Description		
31-8	Reserved	Reserved.		
7-0	COMMAND	Command operation code.		
		Read 1 (lower)	00000000	00

	Read 1 (upper)	00000001	01
	Read 2 (spare)	01010000	50
	Read ID	10010000	90
	Page program/page program (multiple)	10000000	80
	Copy-back program (source address)	00000000	00
	Copy-back program (destination address)	10001010	8A
	Multiple copy-back program	00000011	03
	Block erase/multiplane block erase	01100000	60
	No action	others	

This register writes a specific command to the NFMC. Writing to this register also sends the address contained in NND_ADDR_SRC to the NFMC. There is another type of command register, which does not need to send an address. See the register NND_COMMAND_SEC in Table 33. The NFMC ignores bit combinations other than those stated in Table 32.

Table 33. NAND Controller Second Command Register (NND_COMMAND_SEC)

Bit	Name	Description			Reset	Type
31-8	Reserved	Reserved.				
7-0	COMMAND_SEC	Command operation code.				
		Program 1 (lower)	00000000	00		
		Program 1 (upper)	00000001	01		
		Program 2 (spare)	01010000	50		
		Reset	11111111	FF		
		Page program end	00010000	10		
		Multiple page program end	00010001	11		
		Copy-back program end	00010000	10		
		Block erase/multiplane block erase end	11010000	D0		
		Read status	01110000	70		

Table 33. NAND Controller Second Command Register (NND_COMMAND_SEC) (Continued)

Bit	Name	Description			Reset	Type
		Read multiplane status	01110001	71		
		No action	others			

Writing to this register does not send any address to the NFMC. The command operation codes are primarily to indicate the end of the current operation. For instance, for program operation, after the last data is sent, a delimiter is needed, so command 0x10 (page program end) is sent through the NND_COMMAND_SEC register because there is no need to send the address again. The reset and read status operations also use this register. The NFMC ignores bit combinations other than those stated in Table 33.

Table 34. NAND Controller EEC Bank Selection Register (NND_ECC_SELECT)

Bit	Name	Description
31-3	Reserved	Reserved
2-0	ECC_ SELECT	Defines how many NND_ECC registers to enable.

Depending on the size of an NFMC and the size of ECC computation (256/512 bytes), ECC registers are necessary. For a page of 528 bytes and for an ECC computation on 256 bytes, three 32-bit ECC registers are necessary:

- First register for the data from address 0 to 255
- Second register for data from address 256 to 511
- Third register for data from address 511 to 527

For a future NFMC, the page size can be up to 2048 bytes with a spare of 64 bytes. This means that for an ECC of 256 bytes, nine 32-bit ECC registers are necessary. The first three 32-bit ECC registers are always enabled. By writing to this register, more ECC registers can be enabled.

Table 35. Legal Values for NND_ECC_SELECT Registers

NND_ECC_SELECT[2:0]	ECC Selected
000	NND_ECC1, NND_ECC2, NND_ECC3
001	All registers above + NND_ECC4
010	All registers above + NND_ECC5

Table 35. Legal Values for NND_ECC_SELECT Registers (Continued)

NND_ECC_SELECT[2:0]	ECC Selected
011	All registers above + NND_ECC6
100	All registers above + NND_ECC7
101	All registers above + NND_ECC8
Any other value	All registers above + NND_ECC9

Table 36. NAND Controller ECC Registers (NND_ECC1...NND_ECC9)

Bit	Name	Description	Type	Reset
31-28	Reserved	Reserved	R	0
27	P2048o	Holds ECC code parities accumulated on row	R	0
26	P1024o	Holds ECC code parities accumulated on row	R	0
25	P512o	Holds ECC code parities accumulated on row	R	0
24	P256o	Holds ECC code parities accumulated on row	R	0
23	P128o	Holds ECC code parities accumulated on row	R	0
22	P64o	Holds ECC code parities accumulated on row	R	0
21	P32o	Holds ECC code parities accumulated on row	R	0
20	P16o	Holds ECC code parities accumulated on row	R	0
19	P8o	Holds ECC code parities accumulated on row	R	0
18	P4o	Holds ECC code parities accumulated on column	R	0
17	P2o	Holds ECC code parities accumulated on column	R	0
16	P1o	Holds ECC code parities accumulated on column	R	0
15-12	Reserved	Reserved	R	0
11	P2048e	Holds ECC code parities accumulated on row	R	0
10	P1024e	Holds ECC code parities accumulated on row	R	0
9	P512e	Holds ECC code parities accumulated on row	R	0
8	P256e	Holds ECC code parities accumulated on row	R	0
7	P128e	Holds ECC code parities accumulated on row	R	0
6	P64e	Holds ECC code parities accumulated on row	R	0
5	P32e	Holds ECC code parities accumulated on row	R	0

Table 36. NAND Controller ECC Registers (NND_ECC1...NND_ECC9) (Continued)

Bit	Name	Description	Type	Reset
4	P16e	Holds ECC code parities accumulated on row	R	0
3	P8e	Holds ECC code parities accumulated on row	R	0
2	P4e	Holds ECC code parities accumulated on column	R	0
1	P2e	Holds ECC code parities accumulated on column	R	0
0	P1e	Holds ECC code parities accumulated on column	R	0

These registers hold the ECC code calculated while reading/writing the NFMC. The Pxxxx{o,e (from 2048 to 8) are the parities accumulated on row, while P4{o,e}, P2{o,e}, and P1{o,e} are the parities columns. P2048e and P2048o are necessary only when ECC is computed on 512 bytes. For 256 bytes, those bit locations are left to 0.

Table 37. NAND Controller Reset Register (NND_RESET)

Bit	Name	Description
31-8	Reserved	Reserved
7	RESETDMASYNCHRO	Writing 1 asserts high the DMA request signal.
6-1	Reserved	Reserved
0	RESET_ECC	When 1, Reset NND_ECCx (x from 0 to 9) registers.

- Bits 31-8: Reserved
- Bit 7: When in host mode (no prefetch or postwrite enabled), writing a 1 in this bit asserts high the DMA request. This bit is cleared automatically.
- Bits 6-1: Reserved
- Bit 0: Writing a 1 in the RESET_ECC bit resets the NND_ECCx registers. After the reset is done, RESET_ECC is reset automatically to 0. The NND_ECCx registers are also reset when the resetn pin at the NFC boundary goes low. After a RESET_ECC, the next ECC computation uses the NND_ECC1 register.

Table 38. NAND Controller FIFO Access Register (NND_FIFO)

Bit	Name	Description	Type	Reset
31-28	Data		RW	0

This register is used to access the FIFO when prefetch or postwrite mode is selected. Any access to this register is stalled when FIFO is busy. Access to this register in host mode has no effect; the return value in case of a read is 0x00000000.

Table 39. NAND Controller FIFO Control Register (NND_FIFOCTRL)

Bit	Name	Description	Type	Reset
31-24	FIFO_SIZE	Holds the size in bytes of the FIFO	RW	0
23-16	Reserved	Reserved	RW	0
15-0	BLOCK_COUNT	Holds the block count of (FIFO_SIZE) bytes to read/write in advance	RW	0

This register holds the size of the FIFO and how many blocks of (FIFO_SIZE) bytes to fetch/write to access a full page. For instance, if FIFO_SIZE is 4 bytes, then to read/write a 512 bytes (528 bytes with spare), the BLOCK_COUNT field is loaded with 128 (132 with spare). This register is the seed for the internal NFC counter.

The permitted values for the FIFO_SIZE are 1, 2, 4, 8, or 16 bytes, as shown in Table 40.

Table 40. Permitted Values for FIFO_SIZE

Value	FIFO_SIZE (bytes)
00000001	1(reset value)
00000010	2
00000100	4
00001000	8
Others	16

Table 41. NAND Controller Clock Prescale Register (NND_PSC_CLK)

Bit	Name	Description	Type	Reset
31-4	Reserved	Reserved	RW	0
3-0	PSC_CLK	Prescale sampling clock divider value	RW	0x1

The NFC uses this 4-bit value to divide the interface clock to adjust the timing for the signals on the NFMC, as shown in Table 42.

Table 42. Values for Divider

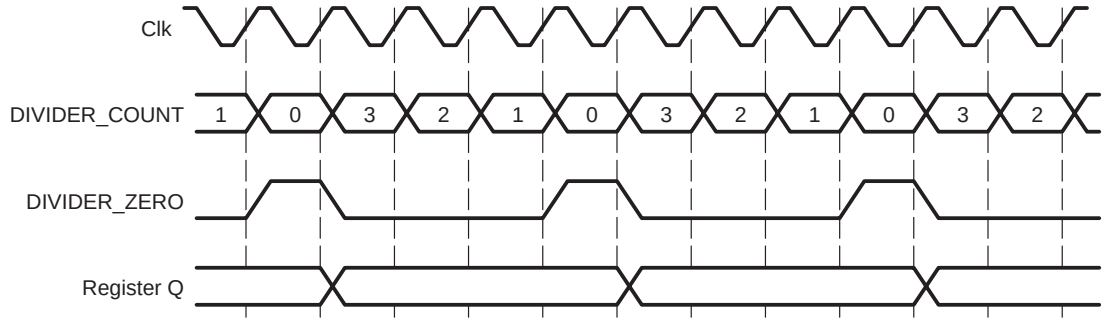
PSC Value	Division	PSC Value	Division
0x0	Divided by 1	0x8	Divided by 9
0x1	Divided by 2	0x9	Divided by 10
0x2	Divided by 3	0xA	Divided by 11
0x3	Divided by 4	0xB	Divided by 12
0x4	Divided by 5	0xC	Divided by 13
0x5	Divided by 6	0xD	Divided by 14
0x6	Divided by 7	0xE	Divided by 15
0x7	Divided by 8	0xF	Divided by 16

This prescale register is used to generate an enable pulse at the divided rate. This pulse is used to enable the NFC sequencer state machine registers.

When it counts down and reaches zero, it generates DIVIDER_ZERO, the enable pulse, and reloads with the value in the NND_PSC_CLK register. All registers that operate at the divided frequency use this pulse as an enable, allowing their outputs to change, as shown by the register Q signal (see Figure 19).

Figure 19 shows the generation of the divided clock for a divisor of 4: there is no actual clock at the divided rate.

Figure 19. Clock Divider Timing



Only as the counter reloads, and when it completes a period, truncated periods are never generated. The new value from the NND_PSC_CLK register is not loaded until the end of the current period.

Table 43. NAND Controller System Test Register (NND_SYSTEST)

Bit	Name	Description	Type	Reset
31-16	Reserved	Reserved	RW	0
15	TEST-UNLOCK	To unlock test features	RW	0
14-3	Reserved	Reserved	RW	0
2	MAP	When 1, the internal FIFO is mapped as registers.	RW	0
1	ACCESS	If 1, unlock registers for read/write access	RW	0
0	ALLOW_INT	If 1, can initiate an interrupt	RW	0

This register tests some features of the NFC. There is a special scheme to access it.

- Bits 31-16: Reserved
- Bit 15: This bit must be set initially, and then for each other access be kept to 1 by the local host, to allow the local host to set/clear other register-bit positions in this test register on subsequent accesses. Hence, to perform a successful set in another bit position, two successive accesses are required: the 1st access to set the TEST_UNLOCK bit to 1, and the 2nd and subsequent accesses to set another bit position to 0/1 while keeping TEST_UNLOCK to 1. If TEST_UNLOCK is 0, the full sequence needs to take place again.
- Bits 14-3: Reserved
- Bit 2: This bit, when 1, maps the FIFO to the addressable space of registers. The FIFO is mapped from address 0x64 to 0x70. Those registers are accessible only in test mode; in normal mode, accessing these registers returns an error.
- Bit 1: When set, all registers become accessible in read and in write (except the NND_REV register). All pins to NFMC are inactive. It is possible to write a value up to 32 bits in the buffer, thus generating an interrupt, and reading back the NAND controller access register (NND_ACCESS).
- Bit 0: Normally, the pending interrupt bits of the NND_STATUS register are set to 1 only when an interrupt is generated; the software cannot initiate an interrupt. By writing a 1 to the ALLOW_INT bit, the software can write a 1 in the interrupt bit of the NND_STAUS, thereby initiating an interrupt.

Table 44. NAND Controller System Configuration Register (NND_SYSCFG)

Bit	Name	Description	Type	Reset
31-2	Reserved	Reserved	RW	0
1	SOFTRESET	When 1, software reset sequence starts.	RW	0
0	AUTOIDLE	Controls clock activity	RW	0

- Bits 31-2: Reserved
- Bit 1: This bit resets the NFC. This is a software reset because the resetn pin is not used for this soft reset. By writing a 1 to this bit, the soft reset sequence starts. This bit is automatically reset to 0 and reads always 0.
- Bit 0: If set to 0, the normally enabled clock is free-running. When 1, the module is in power-saving mode. The local or internal clock runs only when the NFMC is accessed or an operation is ongoing.

Table 45. NAND Controller System Status Register (NND_SYSSTATUS)

Bit	Name	Description	Type	Reset
31-1	Reserved	Reserved	R	0
0	ResetDone	Internal reset monitoring	R	?

- Bits 31-1: Reserved
- Bit 0: When 1, the NFC has finished its reset. When 0, the NFC is resetting.

Table 46. NAND Controller FIFO Test Register (NND_FIFOTEST)

Bit	Name	Description	Type	Reset
31-0	DATA		RW	0

When the NND_SYSTEST is used and the map bit is 1, the internal FIFO is mapped to addressable range. It is then possible to directly read or write to the FIFO, bypassing the NND_FIFO register. In normal mode, an access to these registers is denied and the returned value is 0x00000000.

2.2 Software NAND Flash Controller

This section describes how to connect OMAP5912 to a NAND flash using only the EMIFS logic (referred to throughout this chapter as the Software NAND flash controller). The software NAND flash controller can connect to either 8-bit or 16-bit NAND flashes.

2.2.1 NAND Flash Software Controller Overview

The features of the system are as follows:

- The NAND flash device is mapped on one of the chip-selects of the EMIFS interface of the OMAP5912 device.
- One 8-bit- or 16-bit-wide interface NAND flash device is supported on EMIFS. Either the MPU or DSP (hereafter called the processor) can access and control the NAND flash device.
- The processor manages the command sequence required for block erase, write, read, and block invalidation and management. To reduce processor overhead, either the system DMA (GDMA) or the DSP DMA can be used to write or read blocks of data to/from the NAND flash device.
- There are two options for ECC calculation:
  - The MPU or DSP calculates the ECC by software.To reduce processor overhead, the NAND flash controller peripheral module can calculate the ECC with the support of the DMA for moving the data.
  - Using the NAND flash controller peripheral, an ECC can be calculated on up to nine blocks of 256 bytes at one time before it is required to read the ECC calculation results.Section 2.2.3 explains the write sequence example that is required to use a NAND flash device with the OMAP5912.

Some NAND flash devices require that CS be low during the read access time (tR); hereafter, these devices are called NAND CE Care. Thus, a standard OMAP5912 chip-select cannot be used for the NAND flash chip-select. However, some NAND flash devices do not require that CS be low during tR; these devices are called NAND CE Don't Care. The OMAP5912 chip-select can be used directly for them.

2.2.2 EMIFS Interface With NAND CE Care Flash Device Option

The interface of a NAND CE Care flash device to the OMAP5912 is possible by using $\overline{\text{FLASH.CS2UOE}}$ (output enable) and $\overline{\text{FLASH.CS2UWE}}$ (write enable). The only exception to this policy is that several signals that are used for the NAND flash interface are muxed on signals that are needed for support of synchronous burst flash memories**.**

If both NAND and synchronous burst flash memories are required in the system, two solutions are available:

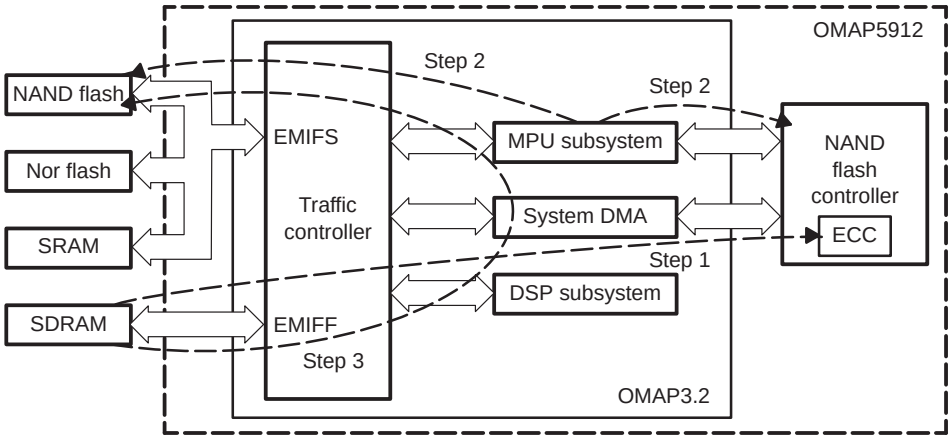
- Generate the NAND flash interface signals by GPIOs at some loss of system performance

- Use a NAND flash device that is NAND CE Don't Care. Be careful to purchase an appropriate NAND flash device (most of the NAND flash devices manufactured are NAND CE Don't Care-compliant).

2.2.3 Write Data Sequence Example

The high-level sequence of operations that is needed to write data to the NAND flash device is shown in Figure 20. The assumption is that the source data to be written to the NAND flash device resides in external SDRAM memory, but this is not a requirement. The source data can reside in any memory space available in the system. The MPU is the processor controlling the write operations.

Figure 20. NAND Flash Write Sequence



Step 1: Calculate ECC

The MPU programs the system DMA as follows:

- ❏ Transfers the data in 8-, 16-, or 32-bit word format from SDRAM into the NAND flash controller
- ❏ Transfer mode (single mode= > channel stop when current transfer finishes)
- ❏ The DMA creates an interrupt on completion of the transfer.
- ❏ Transfer start (hardware start => on DMA request)

The processor MPU configures the NAND flash controller peripheral as follows:

- ❏ Selects the number of blocks to calculate ECC on (1-9) NND_ECC_SELECT register
- ❏ Selects the type of ECC (256/512 byte blocks) by programming the NND_CTRL register
- ❏ Enables the read and program operation: ECC logic-enable by programming bit 0 of the NND_CTRL register
- ❏ Enables the clock in the NND_SYSCFG register bit 0.
- ❏ Sets 1 in NND_PSC_CLK (the module runs full-speed for minimum delay in the ECC calculation)
- ❏ Configures the NAND flash FIFO access
- ❏ Resets the ECC calculation by writing to NND_RESET

Note:

To optimize the transfer from SDRAM to NAND flash controller, keep the RDY/BUSY signal of NAND flash controller at 1.

- ❏ The processor initiates a write command to the NAND flash device (in the NAND flash controller) NND_COMMAND.
- ❏ The NAND flash controller sends a DMA request (FIFO mode).
- ❏ The data is written into the NAND flash controller peripheral by the DMA to the NAND flash FIFO.
- ❏ ECC is calculated.

- ❏ The DMA interrupts the processor when the transfer is complete.
- ❏ The processor reads the ECC calculation from the NAND flash controller peripheral.
- ❏ The calculated ECC results are written in SDRAM.

Step 2: Configure EMIFS and NAND Flash for a Write

- ❏ The associated EMIFS CS chip-select control register is programmed.
- ❏ The configuration register multiplexes the chip-select with a GPIO. The GPIO module drives the GPIO active low when access to the NAND flash device is needed.
- ❏ The configuration register programs the FLASH.RDY signal to be a GPIO input for the generation of interrupts on the rising edge of the NAND flash device R/B signal.
- ❏ The processor programs and enables an interrupt on the rising edge of the GPIO multiplexed on the R/B signal to signify when the NAND flash has completed programming.
- ❏ The processor initiates a write command (80h= serial data input) and an address write to the NAND flash device.
- ❏ The NAND flash device is now ready to receive a block of data.

Step 3: Write Data to NAND Flash Device

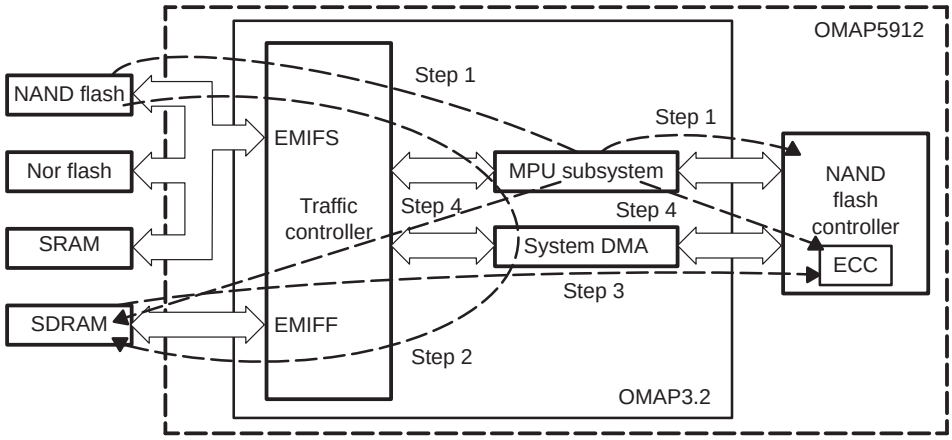
- ❏ The MPU programs the system DMA:
  - ■ Transfer the data in 8-, 16-, or 32-bit word format from SDRAM to the NAND flash peripheral.
  - ■ Transfer mode (single mode= > channel stop when current transfer finishes)
  - ■ The DMA creates an interrupt on completion of the transfer.
  - ■ Transfer start (software request)
- ❏ Data are written into the NAND flash device.
- ❏ When all of the data for a block are written into the device, the processor initiates an auto program command (10h = auto program).
- ❏ The R/B signal goes low to signify that the NAND flash is busy with the internal programming operation (t_{PROG}).
- ❏ The R/B signal goes high, creating an interrupt to the processor that signifies that the block write has been completed.

- ❏ The processor initiates a status read command (70h= status read).
- ❏ The processor reads the NAND flash status register (status read command is sent) and takes the necessary action based on the programming status (if there is an error, block management is required).
- ❏ One block write is completed; further block writes can continue from step #2.

2.2.4 Read Data Sequence Example

The high-level sequence of operations that are needed to read data from the NAND flash device is shown in Figure 21. The assumption is that the destination data read from the NAND flash device is saved in external SDRAM memory, but this is not a requirement. Data can be saved in any memory space available in the system. The MPU is the processor controlling the read operations.

Figure 21. NAND Flash Read Sequence



Step 1: Configure EMIFS, ECC, and NAND Flash for a Read

- ❏ The associated EMIFS chip-select control register is programmed.
- ❏ The configuration register multiplexes the associated CS chip-select with a GPIO. The GPIO module drives the GPIO active low when an access is needed to the NAND flash device.
- ❏ The configuration register programs the FLASH.RDY signal to be a GPIO input for the generation of interrupts on the rising edge of the NAND flash device R/B signal.

- The MPU programs the system DMA:
  - Transfer the data in 8-,16-, or 32-bit word format from theNAND_FLASH device to SDRAM.
  - Transfer mode (single mode= > channel stop when current transfer finishes)
  - The DMA creates an interrupt on completion of the transfer.
- The processor MPU configures the NAND flash controller peripheral as follows:
  - Selects the number of blocks to calculate ECC on (1-9) the NND_ECC_SELECT register
  - Selects the type of ECC (256/512 byte blocks) by programming the NND_CTRL register
  - Enables the read and program operation: ECC logic-enable by programming bit 0 of the NND_CTRL register
  - Enables the clock in the NND_SYSCFG register bit 0
  - Configures NAND flash FIFO access (NND_CTRL)
  - Sets 1 in NND_PSC_CLK (the module runs full speed for minimum delay in the ECC calculation)
  - Resets ECC calculation by writing to NND_RESET

Note:

To optimize the transfer from SDRAM to the NAND flash controller, keep the RDY/BUSY signal of the NAND flash controller at 1.

Step 2: Read Data From NAND Flash Device

- The processor initiates a read command.
- The R/B signal goes low to signify that the NAND flash is busy with the internal programming operation (t_{PROG}).
- The R/B signal goes high, creating an interrupt to the processor that signifies that the NAND flash device is ready to send data.
- The MPU programs the system DMA:
  - Transfer start (software request)
- Data are transmitted from the NAND flash device to SDRAM.
- The DMA interrupts the processor when the transfer is complete:
  - The calcuation of ECC can begin.

- ❏ The R/B signal goes low to signify that the NAND flash is busy with the internal operation.
- ❏ The R/B signal goes high, creating an interrupt to the processor that signifies that the block read has been completed (new block can be read).

Step 3: Calculate ECC

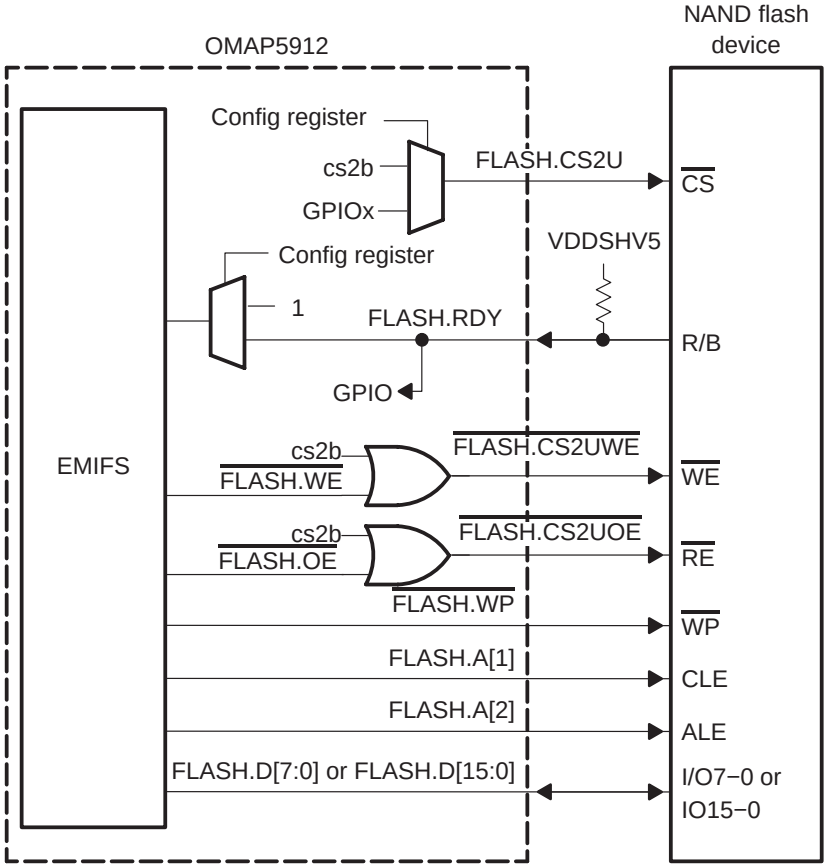
- ❏ The MPU programs the system DMA:
  - ■ Transfer the data in 8-, 16-, or 32-bit word format from SDRAM into the NAND flash controller.
  - ■ Transfer mode (single mode= > channel stop when current transfer finishes)
  - ■ The DMA creates an interrupt on completion of the transfer.
  - ■ Transfer start (hardware start => on DMA request)
- ❏ The processor initiates a write command to the NAND flash device (in the NAND flash controller) NND_COMMAND.
- ❏ The data is written into the NAND flash controller peripheral by the DMA in FIFO.
- ❏ ECC is calculated.
- ❏ The DMA interrupts the processor when the transfer is complete.

Step 4: Postprocessing

- ❏ The processor reads the ECC calculation from the NAND flash controller peripheral.
- ❏ The processor checks the ECC result with the one save in SDRAM (spare area)

Figure 22 shows the interface between the OMAP161X and a NAND flash device.

Figure 22. NAND Flash Device Interface Schematic



- CS: Most NAND flash devices require that CS be low during the read access time (t_R). For example, the Samsung K9K1G08U0M device requires CS to be low during t_R. Thus, a standard OMAP161X chip-select cannot be used for the NAND flash chip-select. To resolve this issue, a GPIO is multiplexed on this pin so that the processor can directly control the state of the NAND flash chip-select during accesses. However, some NAND flash devices, such as the Samsung K9F1G08Q0M and K9F1G16U0M, do not require that CS be low during t_R. For these types of flash devices, the OMAP161X chip-select can be used directly.

- R/B: During read or write operations the NAND flash device R/B signal goes low to indicate that the device is busy and that other operations must wait until completion. To signal the OMAP161X that the operation has completed, the R/B signal of the NAND flash device can be connected to

the FLASH.RDY that is multiplexed with a GPIO. The GPIO is programmed to create an interrupt on the rising edge of R/B.

Note: FLASH.RDY Signal

The FLASH.RDY signal is primarily intended for use with synchronous burst flash devices. If the system does have a synchronous burst flash device and the FLASH.RDY signal is required, then the NAND flash R/B must be connected on some other GPIO of the OMAP161X device. If another GPIO is used, the interface voltage range must be considered between the open drain output of the NAND flash and the OMAP5912. Otherwise, it is possible to remove this input requirement by the use of timers to create the delay and/or the use polling of the NAND flash device status register.

- $\overline{WE}$ and $\overline{RE}$: As explained before, during t_R some NAND flash devices require that $\overline{CS}$ be low and during this time $\overline{WE}$ and $\overline{RE}$ must not toggle. Thus, it is necessary to gate the OMAP161X chip-select, cs2b, with the $\overline{FLASH.WE}$ and $\overline{FLASH.0E}$ signals to create the $\overline{FLASH.CS2UWE}$ and $\overline{FLASH.CS2UOE}$, respectively.

  If the muxed signals are needed in the system, it is possible to generate $\overline{WE}$ and $\overline{RE}$ by GPIO with some performance impact and a more complicated programming model.

- CLE and ALE: The command latch enable (CLE) and address latch enable (ALE) signals are generated by the address pins of OMAP161X FLASH.A[1] and FLASH.A{2}, respectively, as shown in Table 47.

Table 47. CLE and ALE

CLE: FLASH.A[1]	ALE: FLASH.A[2]	System Address (cs2b)	Function
L	L	0x0A00 0000	Data read or write access
H	L	0x0A00 0002	Command write access
L	H	0x0A00 0004	Address write access
H	H	0x0A00 0006	Non-valid access condition

- I/O7:0 or I/015:0 : NAND flash devices have either 8- or 16-bit-wide data buses. The FLASH.D[15] signals of the OMAP161X connect directly to the NAND flash I/O signals.

ac Specifications and Issues

A review of the Samsung K9K1G08U0M NAND flash device ac specifications versus the EMIFS interface has resulted in the following findings:

- All timings required for write accesses are acceptable. WP ($\overline{WE}$ pulse width) minimum is 25 ns, which implies a maximum rate of 40 MHz.
- All timings required for read accesses are acceptable for interface with EMIFS with the note that t_{AR2} min of 50 ns (ALE to $\overline{RE}$ delay-read cycle) must be observed. This can be controlled by the new OE control added to OMAP 3.2.

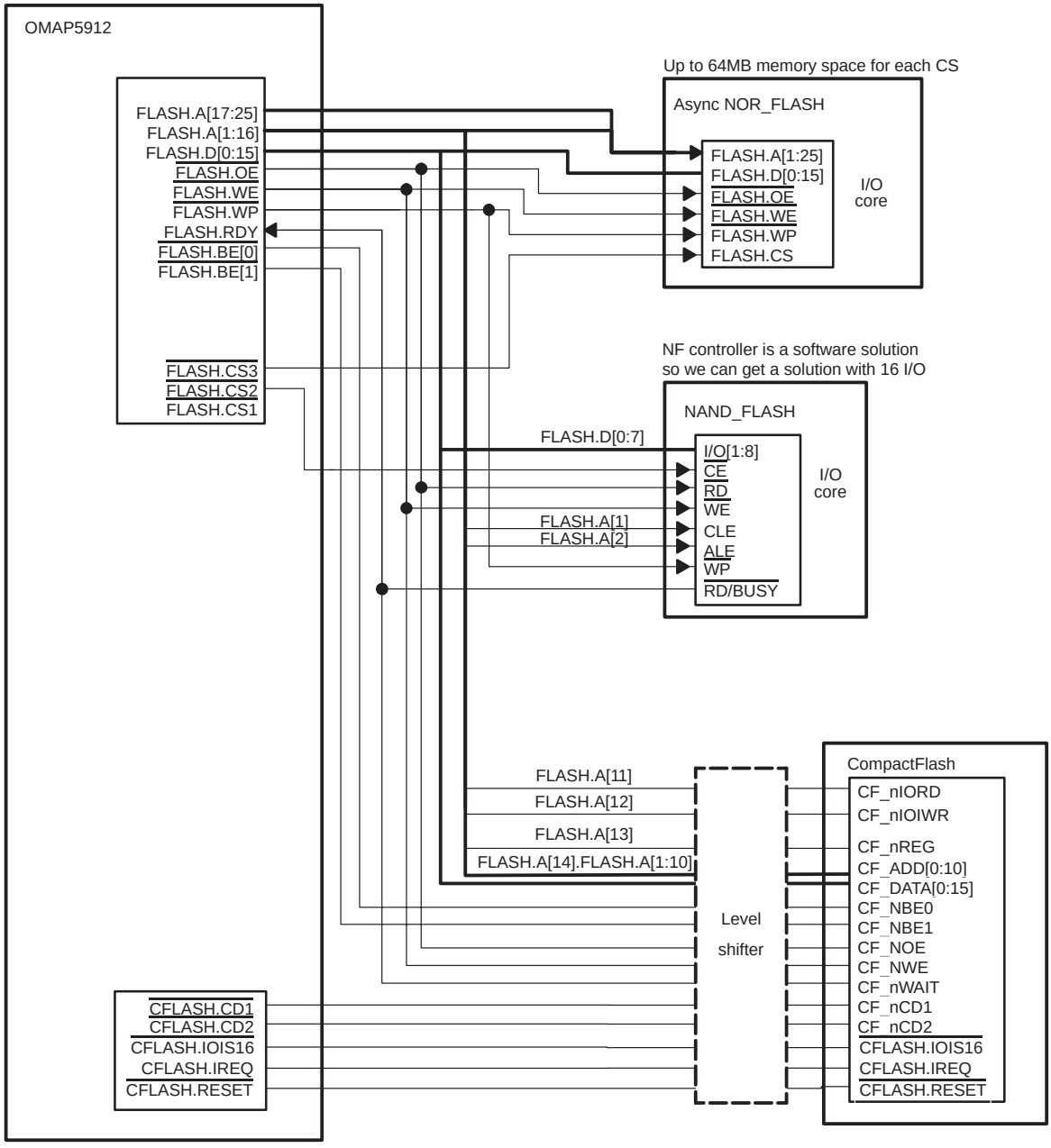
2.2.5 EMIFS Interface With NAND CE Don't Care Flash Device Option

In this case, there is no dedicated signal to interface the NAND flash device. Instead, the standard NOR flash interface is used. Non-multiplexed NOR flash and CompactFlash can also be connected at the same time.

The procedure to follow is the same as that described for NAND CE Don't Care devices.

Figure 23 shows how CompactFlash, NAND flash, and an asynchronous NOR flash can be connected.

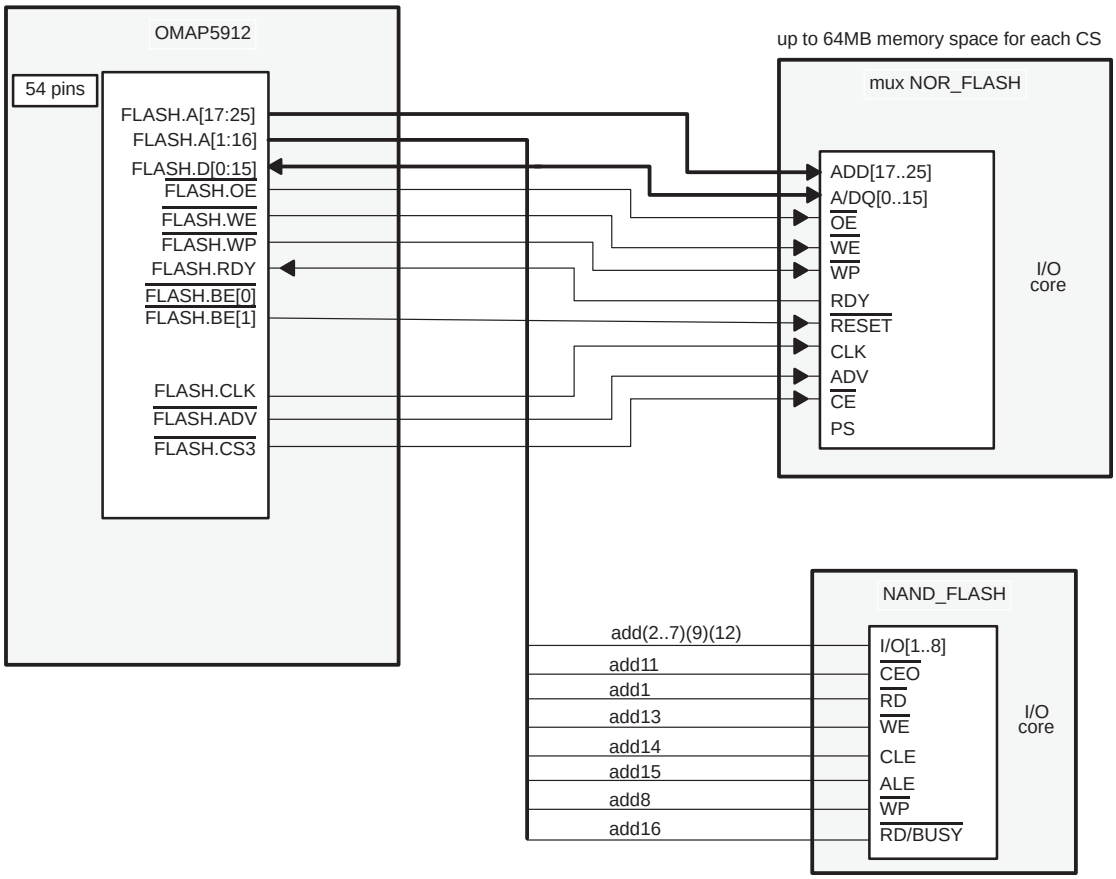
Figure 23. Software NFC with Non-multiplexed NOR and CFC



2.3 Hardware NFC with Multiplexed NOR Flash Add-On Option

If this solution is chosen, the NOR flash interface must be configured with ADDRESS and DATA multiplexed. This solution reuses ADD(1:16) for the hardware NAND flash controller.

Figure 24. Hardware NFC with Multiplexed NOR Flash Add On Option



2.4 CompactFlash Controller

The CompactFlash controller (CFC) interfaces a CompactFlash and a classical memory interface. Control signals from memory interface are processed through the CFC to drive a CompactFlash card, and control signals from

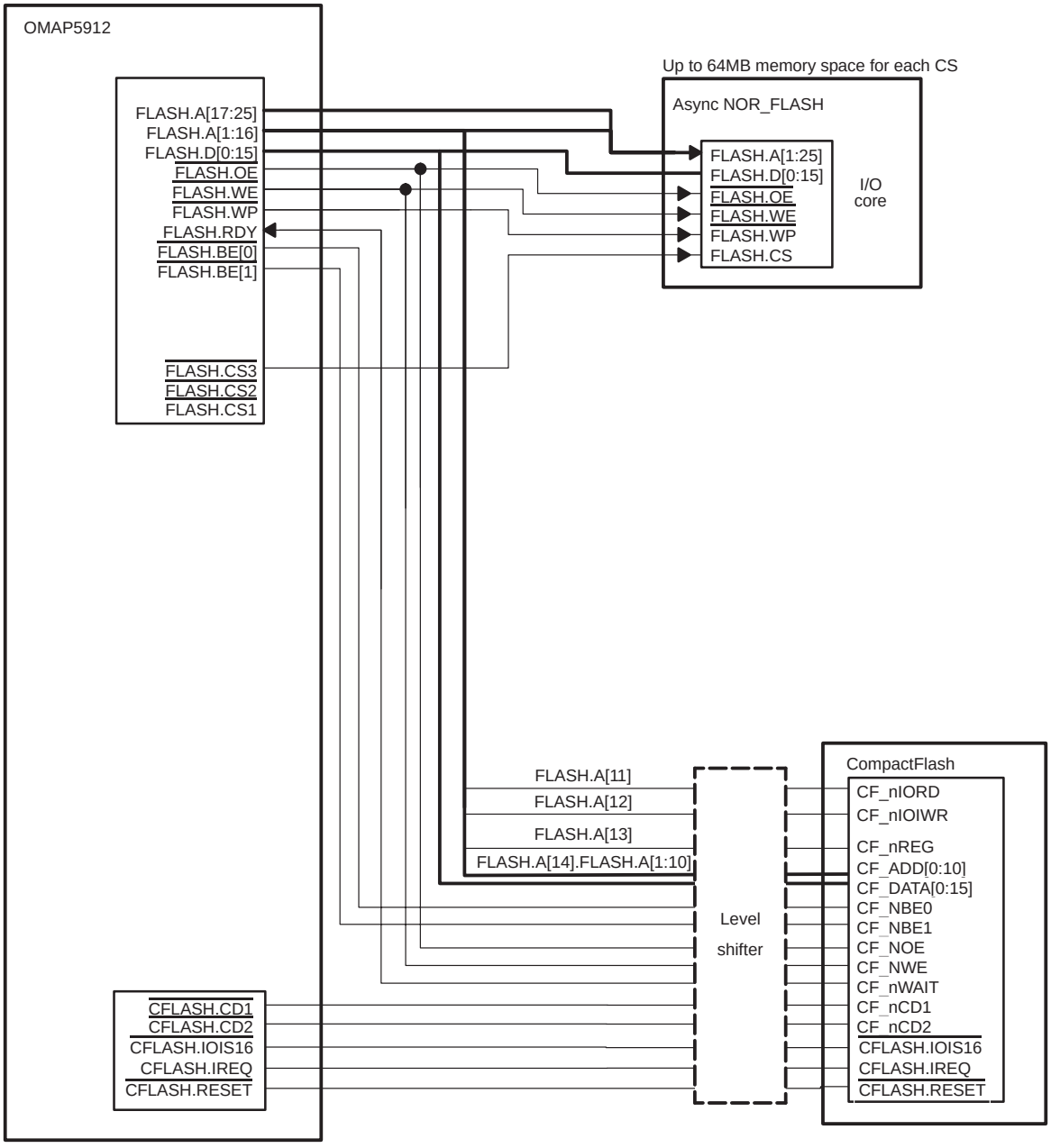
CompactFlash are processed to perform a data transfer to the memory interface.

2.4.1 CFC Connection

The CFC interfaces the slow memory interface block and memories. When CompactFlash access is required, the CFC generates CompactFlash signals and processes the CompactFlash protocol.

Figure 25 shows how to connect external compact flash to CFC.

Figure 25. Software NFC with Non-Multiplexed NOR and CFC



2.4.2 Signal Connections

Table 48 lists the CFC signal connections.

Table 48. CFC Signal Connections

Name	I/O	CompactFlash Name	CompactFlash Description
FLASH.A[10-1]	OUT	A10-A1	Address bus
FLASH.A[14]	OUT	A0	Address bus
FLASH.A[13]	OUT	REG	Attribute memory select
FLASH.A[12]	OUT	IOWR	I/O data write
FLASH.A[11]	OUT	IORD	I/O data read
FLASH.RDY	IN	WAIT	Wait
$\overline{\text{FLASH.WE}}$	OUT	WE	Strobe
$\overline{\text{FLASH.BE[1]}}$	OUT	CE2	Card enable
$\overline{\text{FLASH.BE[0]}}$	OUT	CE1	Card enable
FLASH.D[15:0]	I/O	D15-D0	Data bus
$\overline{\text{FLASH.OE}}$	OUT	OE	Output enable
$\overline{\text{CFLASH.IOIS16}}$	IN	WP IOIS16	Write protect 8/16 bits selection
$\overline{\text{CFLASH.RESET}}$	OUT	RESET	Reset
$\overline{\text{CFLASH.CD1}}$	IN	CD1	Card detect
$\overline{\text{CFLASH.CD2}}$	IN	CD2	Card detect
CFLASH.IREQ	IN	RDY/BSY IREQ	Ready for new data

The I/O values are signals from OMAP5912.

2.4.3 Memory Access Mode Selection

The CompactFlash card supports the following access modes:

- ❏ Common memory
- ❏ Attribute memory
- ❏ I/O

The controller manages access to the different modes of the CompactFlash (access protocol in these modes follows the CompactFlash standard).

Table 49 lists the memory mapping used to select the access mode.

Table 49. CFC Memory Mapping

Space Name	Start Address and CS Address	Stop Address	Size
CF memory space	0000:0000	0000:07FF	2K bytes
CF attribute space	0000:0800	0000:0FFF	2K bytes
CF I/O space	0000:1000	0000:17FF	2K bytes
CF true IDE	0000:1800	0000:1FFF	2K bytes

The CFC does not support the true IDE mode yet. However, memory space is reserved for future expansion.

2.4.4 Interface Registers

The CFC contains a control register to reset the CompactFlash card, a status register, and a configuration register.

2.4.5 CompactFlash Controller Registers

Table 50 lists the CFC controller registers. Table 51 through Table 53 describe the register bits.

Table 50. CFC Controller Registers

Register	Description	Address	Size	HW Reset Value
CF_STATUS_REG	CFC status register	0xFFFB:E000	16 bits Written by the CFC Read by TIPB bus	1111 1111 1111 111b
CF_CFG_REG_I	CFC configuration register 1	0xFFFB:E002	16 bits Written by the CFC Read by TIPB bus	0xFFFF
CF_CONTROL_REG	CFC control register	0xFFFB:E004	16 bits Written by the CFC Read by TIPB bus	0x0000

Table 51. CFC Status Register (CF_STATUS_REG)

Bit	Name	Function	Reset
15:3		Unused	0x1FFF
2	Last read access	Bad read access (active low)	0x1
		When a CompactFlash read access is performed and the CompactFlash card is not correctly connected (Card Detect is high), the CFC asserts this bit low.	
1	Last write access	Bad write access (active low)	0x1
		When a CompactFlash write access is performed and the CompactFlash card is not correctly connected (Card Detect is high), the CFC asserts this bit low.	
0	Card detect	CompactFlash correctly connected	Undefined
		When CompactFlash is correctly connected, the card detect is low and this bit is asserted low.	

The status register represents the connected or disconnected states of the CompactFlash. When a CompactFlash is written, an interrupt request is generated.

After this interruption, the status register must be accessed through the TIPB bus following an interrupt subroutine. The register contains data about the CompactFlash connection (connected or not) and current data transfer (read access, write access, or no access).

If a CompactFlash access is attempted while the CompactFlash card is not correctly connected, the last access direction is written in the status register to inform the system that the last transfer has not been completed.

Table 52. CFC Configuration Register (CF_CFG_REG)

Bit	Name	Function	Reset
15:4		Unused	0xFFF
3	Chip-select 3 configuration	Chip-select is connected on chip-select 3 (active low).	0x1
		Any transfer to CS_3 is a CompactFlash access.	
2	Chip-select 2 configuration	Chip-select is actually connected on chip-select 2 (active low).	0x1
		Any transfer to CS_2 is a CompactFlash access.	

Table 52. CFC Configuration Register (CF_CFG_REG) (Continued)

Bit	Name	Function	Reset
1	Chip-select 1 configuration	Chip-select is actually connected on chip-select 1 (active low). Any transfer to CS_1 is a CompactFlash access.	0x1
0	Chip-select 0 configuration	Chip-select is actually connected on chip-select 0 (active low) Any transfer to CS_0 is a CompactFlash access.	0x1

The configuration register indicates whether or not CompactFlash is connected, and on which chip-select.

This register runs through a CFC TIPB interface and must be configured using the TIPB bus.

The CompactFlash card is reset from the CFC by a CF_RESET read in the CFC control register, which can only be written through the TIPB bus.

Table 53. CFC Control Register (CF_CONTROL_REG)

Bit	Name	Function	Reset
15:1		Unused	0x0000
0	CF_CARD_ RESET	CompactFlash card reset (active low)	0x0

3 Frame Buffer

The frame buffer is used to store video frame before emission to an internal or external LCD controller.

Table 54. Test RAM Mapping

	L3 OCP T1			
	Start Address	End Address	Size	Supported Access
SRAM	2000 0000	2003 E7FF	250Kbytes	8/16/32-bit Ex/R/W

The frame buffer is connected to the L3 OCP-T1 port. It is accessible from the following systems:

- MPU subsystem (ARM926EJS processor)
- DSP subsystem (C55x DSP processor via the DSP MMU)

- System DMA (DMA OCPT1 port)

The frame buffer only supports READ/WRITE/IDLE accesses: an error is generated in other cases or if the address is incorrect.

The frame buffer only supports 8-, 16-, and 32-bit access in little endian.

Access Size	System Address	Data Bus			
		[31:24]	[23:16]	[15:8]	[7:0]
8	xx00	-	-	-	D[7:0]
	xx01	-	-	D[7:0]	-
	xx10	-	D[7:0]	-	-
	xx11	D[7:0]		-	-
16	xx00	-	-	D[15:8]	D[7:0]
	xx10	D[15:8]	D[7:0]	-	-
32	xx00	D[31:24]	D[23:16]	D[15:8]	D[7:0]

The module is clocked up to 100 MHz with the TC2_CK on OCP T2 port.

The frame buffer only supports incrementing burst accesses.

Any invalid access is logged into the OCPT2_ATYPER register within OMAP3.2. The OCP bus error access corresponds to OCPT2_ATYPER[3] status bit.

Peripheral Reset

The software must ensure that no access is performed (including DMA access) while the MPU peripheral reset is asserted. Whenever the peripheral reset is active, the OCP interface is held in reset.

Index

P

R

S

T

W

OMAP5912 Multimedia Processor Interrupts Reference Guide

Literature Number: SPRU757B
October 2004

OMAP™
TEXAS INSTRUMENTS TECHNOLOGY

TEXAS INSTRUMENTS

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.

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
		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 document describes the interrupts of the OMAP5912 multimedia processor.

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.

Related Documentation From Texas Instruments

Documentation that describes the OMAP5912 device, related peripherals, and other technical collateral, is available in the OMAP5912 Product Folder on TI's website: www.ti.com/omap5912.

Trademarks

OMAP and the OMAP symbol are trademarks of Texas Instruments.

Contents

Figures

Tables

Interrupts

This document describes the interrupts of the OMAP5912 multimedia processor.

1 Interrupt Overview

Three level 2 interrupt controllers are used in OMAP5912:

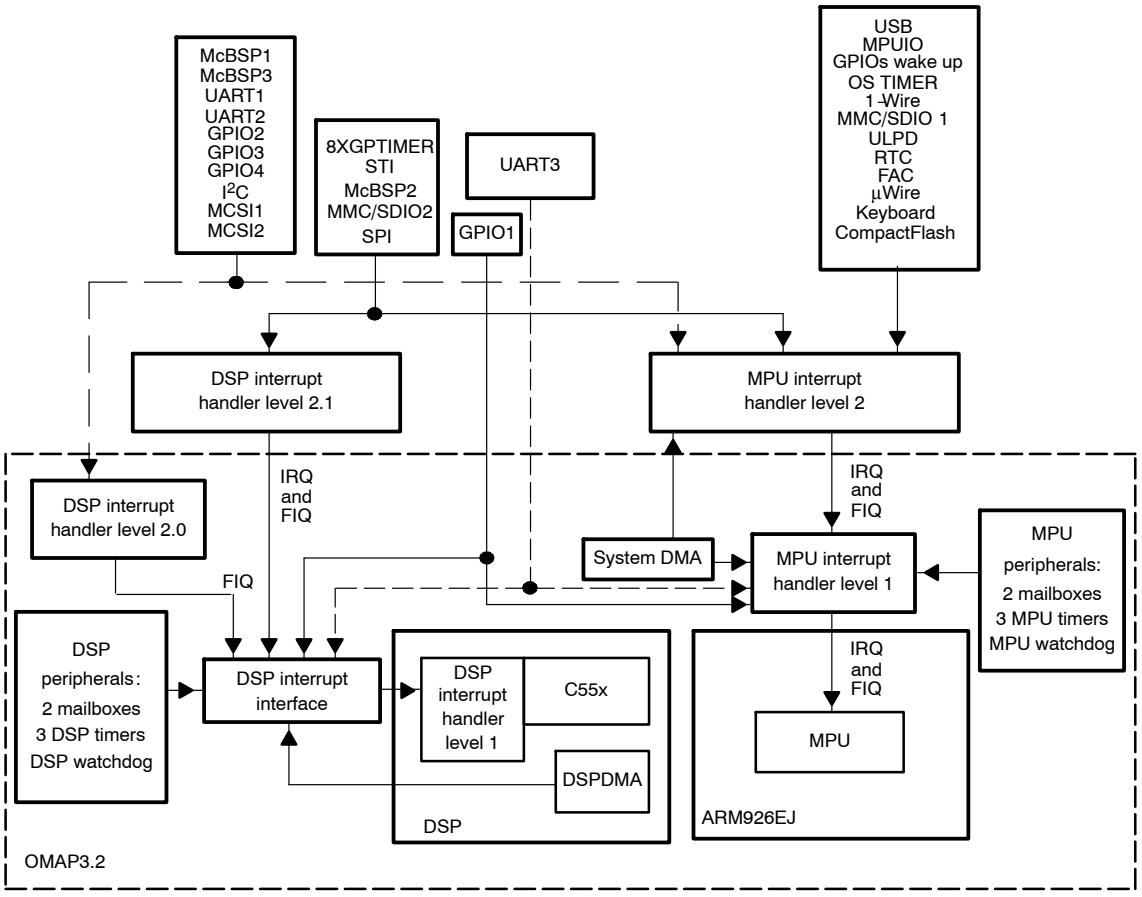
- One MPU level 2 interrupt handler (also referred to as MPU interrupt level 2) is implemented outside of OMAP3.2 and can handle 128 interrupts.
- One DSP level 2 interrupt handler (also referred to as DSP interrupt level 2.1) is instantiated outside of OMAP3.2 and can handle 64 interrupts.
- One OMAP3.2 DSP level 2 interrupt handler (referenced as DSP interrupt level 2.0) can handle 16 interrupts.

There are two level 1 interrupt controllers in OMAP3.2 and the DSP:

- One MPU level 1 interrupt handler that can handle 31 interrupts (OMAP3.2).
- One DSP level 1 interrupt handler that can handle 16 interrupts (DSP).

Figure 1 shows which peripheral module can trigger an interrupt in each processor. Some peripherals can generate interrupts to both processors. There are 160 interrupts on the MPU and 98 interrupts on the DSP side.

Figure 1. Interrupt Interconnect



Two DSP level 2 interrupt handlers connect directly to the DSP interrupt interface and then to the DSP level 1 interrupt handler. One MPU level 2 interrupt handler connects in parallel to the MPU level 1 interrupt handler. For more detail on the architecture and programming model of the level 2 interrupt handler, see Section 2, *Interrupt Controller*.

There are four groups of shared peripherals. Most of the shared peripherals connect to DSP/MPU level 2 interrupt handlers. Two exceptions are the GPIO1 and UART3, which connect to the MPU level 1 handler and the DSP level 1 handler.

Table 1 through Table 5 include the default priority and the required sensitivity to be programmed by software per interrupt line for each interrupt handler (L1 and L2). When the sensitivity is not precise**,** it must be considered as edge. The sensitivity depends on the peripheral type.

1.1 DSP Interrupt Mapping

On the DSP side are 98 interrupt input lines, 18 on the first level, and 80 on the second level.

Table 1. DSP Level 1 Interrupt Mapping

Priority	DSP Soft Interrupt	DSP Hard Interrupt	Location Vectors (Hex/Byte)	Function
1	SINT0	INT0	0	Reset
2	SINT1	INT1	8	Nonmaskable interrupt
3	SINT2	INT2	10	Emulator/test interrupt
5	SINT3	INT3	18	Level 2.0 interrupt handler
6	SINT4	INT4	20	TC_ABORT interrupt
7	SINT5	INT5	28	MAILBOX 1
9	SINT6	INT6	30	Level 2.1 interrupt handler FIQ
10	SINT7	INT7	38	IRQ2_GPIO1
11	SINT8	INT8	40	DSP timer 3 interrupt
13	SINT9	INT9	48	DMA_channel_1 interrupt
14	SINT10	INT10	50	MPUI interrupt
15	SINT11	INT11	58	Reserved
17	SINT12	INT12	60	UART3
18	SINT13	INT13	68	DSP watchdog interrupt
21	SINT14	INT14	70	DMA_channel_4 interrupt
22	SINT15	INT15	78	DMA_channel_5 interrupt
4	SINT16	INT16	80	STIO interrupt
8	SINT17	INT17	88	Level 2.1 interrupt handler IRQ
12	SINT18	INT18	90	DMA_channel_0 interrupt
16	SINT19	INT19	98	MAILBOX 2
19	SINT20	INT20	A0	DMA_channel_2 interrupt
20	SINT21	INT21	A8	DMA_channel_3 interrupt
23	SINT22	INT22	B0	DSP timer 2 interrupt

Table 1. DSP Level 1 Interrupt Mapping (Continued)

Priority	DSP Soft Interrupt	DSP Hard Interrupt	Location Vectors (Hex/Byte)	Function
24	SINT23	INT23	B8	DSP timer 1 interrupt
25	SINT24	INT24	C0	Bus error interrupt # 25 BERR
26	SINT25	INT25	C8	Emulator interrupt # 26 DLOG
27	SINT26	INT26	D0	Emulator interrupt # 27 RTOS
28	SINT27	INT27	D8	Software interrupt #28
29	SINT28	INT28	E0	Software interrupt #29
30	SINT29	INT29	E8	Software interrupt #30
31	SINT30	INT30	F0	Software interrupt #31
32	SINT31	INT31	F8	Software interrupt #32

1.1.1 DSP Level 2 Interrupt Handler

The system has two DSP level 2 interrupt handlers. One is in the OMAP3.2 gigacell (the DSP level 2.0 interrupt handler), and the other is outside the OMAP3.2 gigacell (the DSP level 2.1 interrupt handler):

- The DSP level 2.0 interrupt handler handles 16 interrupt lines.
- The DSP level 2.1 interrupt handler handles 64 interrupt lines.

For more detail on the architecture and programming model of the level 2 interrupt handler, see Section 2, *Interrupt Controller*.

1.1.2 DSP Level 2 Interrupt Mapping

The default interrupt priority for the DSP interrupt handler level 1 can be remapped by MPU software only when DSP is held in reset. All level 2 interrupt lines have the same priority by default, which DSP software can modify through the configuration register in the level 2 interrupt controller.

Table 2. DSP Level 2 Interrupt Mapping

Level 2.0 Interrupt Line	Mapping	Default Sensitivity Configuration
IRQ_0	McBSP3 TX	Level
IRQ_1	McBSP3 RX	Level
IRQ_2	McBSP1 TX	Level
IRQ_3	McBSP1 RX	Level
IRQ_4	UART2	Level
IRQ_5	UART1	Level
IRQ_6	MCSI1 TX	Level
IRQ_7	MCSI1 RX	Level
IRQ_8	MCSI2 TX	Level
IRQ_9	MCSI2 RX	Level
IRQ_10	MCSI1 frame error	Level
IRQ_11	MCSI2 frame error	Level
IRQ_12	IRQ2_GPIO2	level
IRQ_13	IRQ2_GPIO3	level
IRQ_14	IRQ2_GPIO4	level
IRQ_15	I^2C	Level

Table 3. DSP Level 2.1 Interrupt Mapping

Level 2.1 Interrupt Line	Mapping	Default Sensitivity Configuration
IRQ_0	NAND flash interrupt	Level
IRQ_1	GPTIMER1	Level
IRQ_2	GPTIMER2	Level
IRQ_3	GPTIMER3	Level
IRQ_4	GPTIMER4	Level
IRQ_5	GPTIMER5	Level
IRQ_6	GPTIMER6	Level

Table 3. DSP Level 2.1 Interrupt Mapping (Continued)

Level 2.1 Interrupt Line	Mapping	Default Sensitivity Configuration
IRQ_7	GPTIMER7	Level
IRQ_8	GPTIMER8	Level
IRQ_9	Reserved	–––––
IRQ_10	McBSP2 TX	Level
IRQ_11	McBSP2 RX	Level
IRQ_12	MCSI1_RST_INT	Level
IRQ_13	MCSI2_RST_INT	Level
IRQ_14	MMC/SDIO2	Level
IRQ_15	SPI	Level
IRQ_16–IRQ_47	Reserved	–––––
IRQ_48 … IRQ_63	Free	–––––

For each interrupt, the user must configure the SENS_nEDGE bit in the corresponding interrupt level register (ILR) according to whether the interrupt is edge- or level-sensitive. See Table 13 for more details.

1.1.3 MPU Interrupt Mapping

The interrupt priority is not hard-coded on the MPU side. Thus, the software must define all interrupt priorities for both level interrupt handlers. (See Table 4.)

Table 4. MPU Level 1 Interrupt Mapping

Level 1 Interrupt Line	OMAP 5912 Mapping	Sensitivity
IRQ_0	Level 2 interrupt handler IRQ	Level
IRQ_1	Camera IF	Level
IRQ_2	Level 2 interrupt handler FIQ	Level
IRQ_3	External FIQ	User-defined

† These IRQs are available only when the DMA is in OMAP3.2 mode (i.e. not in OMAP3.1 compatibility mode). See the Multimedia Processor Direct Memory Access (DMA) Support Reference Guide (literature number SPRU755) for more information.
‡ These IRQs are available only when the DMA is in OMAP3.1 compatibility mode. See the Multimedia Processor Direct Memory Access (DMA) Support Reference Guide (literature number SPRU755) for more information.

Table 4. MPU Level 1 Interrupt Mapping (Continued)

Level 1 Interrupt Line	OMAP 5912 Mapping	Sensitivity
IRQ_4	McBSP2 TX	Edge
IRQ_5	McBSP2 RX	Edge
IRQ_6	IRQ_RTDX (emulation event)	Edge
IRQ_7	IRQ_DSP_MMU_ABORT	Level
IRQ_8	IRQ_HOST_INT	Edge
IRQ_9	IRQ_ABORT	Level
IRQ_10	IRQ_DSP_MAILBOX1	Level
IRQ_11	IRQ_DSP_MAILBOX2	Level
IRQ_12	IRQ_LCD_LINE	Level
IRQ_13	Reserved	–––––
IRQ_14	IRQ1_GPIO1	Level
IRQ_15	UART3	Level
IRQ_16	IRQ_TIMER3	Edge
IRQ_17	GPTIMER1	Level
IRQ_18	GPTIMER2	Level
IRQ_19	IRQ_DMA_CH0†/IRQ_DMA_CH0_CH6‡	Level
IRQ_20	IRQ_DMA_CH1†/IRQ_DMA_CH1_CH7‡	Level
IRQ_21	IRQ_DMA_CH2†/IRQ_DMA_CH2_CH8‡	Level
IRQ_22	IRQ_DMA_CH3	Level
IRQ_23	IRQ_DMA_CH4	Level
IRQ_24	IRQ_DMA_CH5	Level
IRQ_25	IRQ_DMA_CH_LCD	Level
IRQ_26	IRQ_TIMER1	Edge
IRQ_27	IRQ_WD_TIMER	Edge

† These IRQs are available only when the DMA is in OMAP3.2 mode (i.e. not in OMAP3.1 compatibility mode). See the Multimedia Processor Direct Memory Access (DMA) Support Reference Guide (literature number SPRU755) for more information.

‡ These IRQs are available only when the DMA is in OMAP3.1 compatibility mode. See the Multimedia Processor Direct Memory Access (DMA) Support Reference Guide (literature number SPRU755) for more information.

Table 4. MPU Level 1 Interrupt Mapping (Continued)

Level 1 Interrupt Line	OMAP 5912 Mapping	Sensitivity
IRQ_28	Public TIPB abort	Level
IRQ_29	Reserved	–––––
IRQ_30	IRQ_TIMER2	Edge
IRQ_31	IRQ_LCD_CTRL	Level

† These IRQs are available only when the DMA is in OMAP3.2 mode (i.e. not in OMAP3.1 compatibility mode). See the Multimedia Processor Direct Memory Access (DMA) Support Reference Guide (literature number SPRU755) for more information.
‡ These IRQs are available only when the DMA is in OMAP3.1 compatibility mode. See the Multimedia Processor Direct Memory Access (DMA) Support Reference Guide (literature number SPRU755) for more information.

1.1.4 ARM926EJS Level 2 Interrupt Mapping

DSP and MPU share most of the peripherals. Therefore, almost all of the DSP level 2 interrupts also go to the MPU level 2 interrupt handlers. The MPU level 2 interrupt handlers are enabled to process 128 more interrupt lines outside the OMAP gigacell, for a total of 160 interrupt lines. (See Table 5.)

Table 5. MPU Level 2 Interrupt Mapping

Level 2 Interrupt Line	Mapping	Sensitivity
IRQ_0	FAC	Level
IRQ_1	Keyboard	Edge
IRQ_2	μWIRE TX	Level
IRQ_3	μWIRE RX	Level
IRQ_4	I^2C	Level
IRQ_5	MPUIO	Level
IRQ_6	USB HHC 1	Level
IRQ_7	USB HHC 2	Level
IRQ_8	USB_OTG	Level
IRQ_9	Reserved	–––––
IRQ_10	McBSP3 TX	Edge
IRQ_11	McBSP3 RX	Edge

† These IRQs are available only when the DMA is in OMAP3.2 mode (i.e. not in OMAP3.1 compatibility mode). See the Multimedia Processor Direct Memory Access (DMA) Support Reference Guide (literature number SPRU755) for more information.

Table 5. MPU Level 2 Interrupt Mapping (Continued)

Level 2 Interrupt Line	Mapping	Sensitivity
IRQ_12	McBSP1 TX	Edge
IRQ_13	McBSP1 RX	Edge
IRQ_14	UART1	Level
IRQ_15	UART2	Level
IRQ_16	MCSI1 combined TX/RX/Frame error/RST	Level
IRQ_17	MCSI2 combined TX/RX/Frame error/RST	Level
IRQ_18	Free	–––––
IRQ_19	Reserved	–––––
IRQ_20	USB W2FC Geni it	Level
IRQ_21	1-Wire	Level
IRQ_22	OS timer	Edge
IRQ_23	MMC/SDIO1	Level
IRQ_24	32-kHz gauging IRQ/USB client wakeup IRQ	Level/Edge
IRQ_25	RTC periodical timer	Edge
IRQ_26	RTC alarm	Level
IRQ_27	Reserved	–––––
IRQ_28	DSP_MMU_IRQ	Level
IRQ_29	USB W2FC IRQ_ISO_ON	Level
IRQ_30	USB W2FC IRQ_NON_ISO_ON	Level
IRQ_31	McBSP2 RX OVERFLOW	Level
IRQ_32–IRQ_33	Reserved	–––––
IRQ_34	GPTIMER3	Level
IRQ_35	GPTIMER4	Level
IRQ_36	GPTIMER5	Level

† These IRQs are available only when the DMA is in OMAP3.2 mode (i.e. not in OMAP3.1 compatibility mode). See the Multimedia Processor Direct Memory Access (DMA) Support Reference Guide (literature number SPRU755) for more information.

Table 5. MPU Level 2 Interrupt Mapping (Continued)

Level 2 Interrupt Line	Mapping	Sensitivity
IRQ_37	GPTIMER6	Level
IRQ_38	GPTIMER7	Level
IRQ_39	GPTIMER8	Level
IRQ_40	IRQ1_GPIO2	Level
IRQ_41	IRQ1_GPIO3	Level
IRQ_42	MMC/SDIO2	Level
IRQ_43	CompactFlash	Edge
IRQ_44	COMMRX (emulation event)	Level
IRQ_45	COMMTX (emulation event)	Level
IRQ_46	Peripheral wake up	Level
IRQ_47	Free	Level
IRQ_48	IRQ1_GPIO4	Level
IRQ_49	SPI	Level
IRQ_50–IRQ_52	Reserved	–––––
IRQ_53	IRQ_DMA_CH6†	Level
IRQ_54	IRQ_DMA_CH7†	Level
IRQ_55	IRQ_DMA_CH8†	Level
IRQ_56	IRQ_DMA_CH9†	Level
IRQ_57	IRQ_DMA_CH10†	Level
IRQ_58	IRQ_DMA_CH11†	Level
IRQ_59	IRQ_DMA_CH12†	Level
IRQ_60	IRQ_DMA_CH13†	Level
IRQ_61	IRQ_DMA_CH14†	Level
IRQ_62	IRQ_DMA_CH15†	Level

† These IRQs are available only when the DMA is in OMAP3.2 mode (i.e. not in OMAP3.1 compatibility mode). See the Multimedia Processor Direct Memory Access (DMA) Support Reference Guide (literature number SPRU755) for more information.

Table 5. MPU Level 2 Interrupt Mapping (Continued)

Level 2 Interrupt Line	Mapping	Sensitivity
IRQ_63	Reserved	–––––
IRQ_64	Reserved (USB HHC2 suspend)	–––––
IRQ_65	Reserved	–––––
IRQ_66	Free	––––
IRQ_67–IRQ_90	Reserved	–––––
IRQ_91	SHA-1/MD5	Level
IRQ_92	RNG	Level
IRQ_93	RNGIDLE	Level
IRQ_94–IRQ_102	Reserved	–––––
IRQ_103 … IRQ_127	Free	–––––

† These IRQs are available only when the DMA is in OMAP3.2 mode (i.e. not in OMAP3.1 compatibility mode). See the Multimedia Processor Direct Memory Access (DMA) Support Reference Guide (literature number SPRU755) for more information.

For each interrupt, the user must configure the SENS_nEDGE bit in the corresponding interrupt level register (ILR) according to whether the interrupt is edge or level sensitive. See Table 13 for more details.

2 Interrupt Controllers (MPU Level 2 and DSP Level 2.1)

The MPU level 2 and DSP level 2.1 interrupt controllers have the same programming model. The only difference between the two is the number of interrupts each can handle (128 for MPU level 2 and 64 for DSP level 2.1). Throughout this section, both interrupt controllers are referred to collectively as the interrupt controller.

The MPU level 2 interrupt controller functional clock source is ARM_CK or ARM_CK/2, according to the ARM_CKCTL.ARM_INTHCK_SEL bit.

The DSP level 2.1 interrupt controller functional clock source is the DSPPER_CK.

The interrupt controller is able to handle interrupts coming from different functional blocks, prioritize them, and route them to a host. It generates one IRQ and one FIQ signal to the host. Both signals are active low-level interrupts, synchronous with the interrupt controller functional clock.

The interrupt controller can be programmed to assign different priorities and mask each interrupt independently. Each interrupt line can be programmed to be either level sensitive or edge triggered. The interrupt handler also provides an asynchronous signal to the host in order to have a way to wake up the system, in case an interrupt occurs when the clocks are turned off (system in idle state). See Figure 2 for the interrupt controller block diagram.

2.1 Functional Description

The interrupt controller provides prioritized and maskable interrupts to the host.

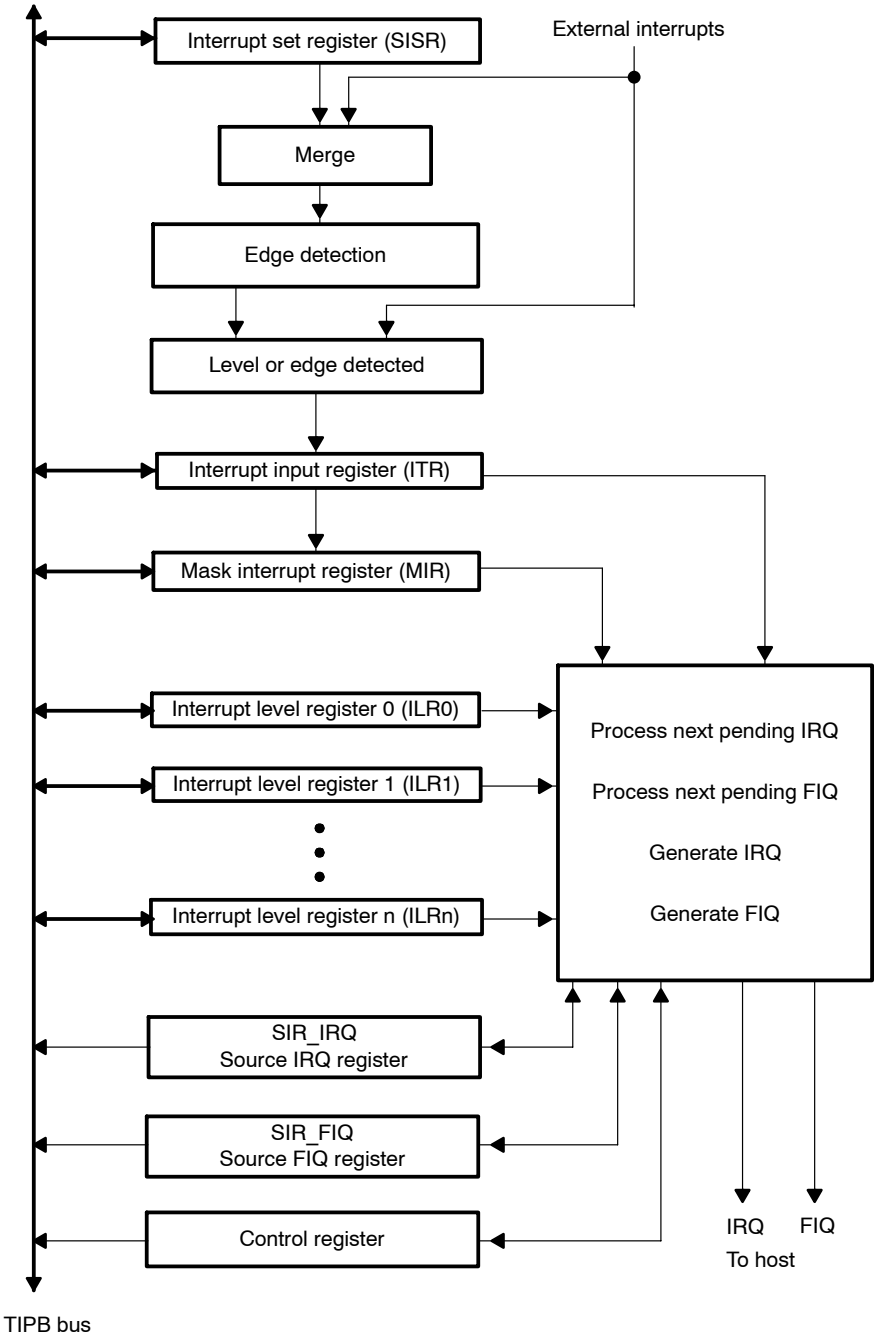
One interrupt level register (ILR) is associated with each incoming interrupt. It assigns a priority to the corresponding interrupt, determines whether it is to be level or edge sensitive, and selects which interrupt (FIQ or IRQ) it is to generate. If several interrupts have the same priority level, they are serviced in a predefined order (see Table 13).

For test purposes, the interrupt controller provides a set of software interrupt set registers (SISR). Each bit of these registers corresponds to an incoming interrupt line. By writing a1 to the targeted bit, an interrupt is generated if the corresponding ILR is set to edge sensitive. External interrupt requests and internal software requests are merged before being sent to the interrupt controller state machine.

Each incoming interrupt is routed either to the FIQ or the IRQ interrupt. The IRQ or FIQ outputs from the interrupt controller are reset by writing a 1 to the corresponding bit of the control register. Both the IRQ and FIQ are synchronous to the interrupt controller functional clock.

The interrupt handler can wake up the host asynchronously even if the functional clocks are turned off (host in idle state).

Figure 2. Interrupt Controller



2.1.1 Interrupt Processing Sequence

Although only the IRQ is discussed here, the sequence is the same for FIQ. The sequence depends, however, on the interrupt sensitivity (level or edge).

2.1.2 Edge-Triggered Interrupts

1) The interrupt controller module receives incoming interrupts and registers them in the ITR register.

2) The interrupt controller asserts the IRQ signal and begins priority calculation. When the highest priority interrupt is known, the source IRQ register (SIR_IRQ) is updated with the current interrupt number.

   The only way to deassert the IRQ that has been sent to the host is to set the NEW_IRQ_AGR bit in the control register.

3) The host, when it recognizes the interrupt, jumps to the interrupt routine.

4) Within the interrupt routine, the host reads the SIR_IRQ register to determine which interrupt line caused the interrupt. Reading the SIR_IRQ resets the interrupt bit in the ITR register. The IRQ/FIQ line, however, stays asserted. Based on the content of SIR_IRQ, the host executes specific code.

5) Before jumping out of the interrupt routine, the host must set the NEW_IRQ_AGR bit of the control register. Setting this bit deasserts the IRQ line and enables the interrupt controller to process any other pending interrupts.

2.1.3 Level-Sensitive Interrupts

1) The interrupt controller receives incoming interrupts. Level-sensitive interrupts are not registered. The interrupt controller assumes that a peripheral asserting a level-sensitive interrupt does not deassert it until the software directs it to do so.

2) The interrupt controller asserts the IRQ signal and begins priority calculation. When the highest priority interrupt is known, the SIR_IRQ register is updated to the current interrupt number.

   The only way to deassert the IRQ that has been sent to the host is to set the NEW_IRQ_AGR bit in the control register.

3) The host, when it recognizes the interrupt, jumps to the interrupt routine.

4) Within the interrupt routine, the host reads the SIR_IRQ register in the interrupt controller to determine which interrupt line caused the interrupt. Reading the SIR_IRQ has no effect on the ITR register for the level-sensitive interrupt, because the interrupt is still asserted at this point. Based on the content of SIR_IRQ, the host executes specific code.

5) Before jumping out of the interrupt routine, the software must:

   - Ensure that the peripheral that asserted the interrupt deasserts it.
   - Set the NEW_IRQ_AGR bit of the control register. Setting this bit deasserts the IRQ/FIQ line and enables the interrupt controller to process any pending interrupts.

2.1.4 Interrupt Latency

To minimize interrupt latency, an IRQ (or FIQ) is generated whenever an incoming interrupt is detected. Owing to internal resynchronization, the IRQ/FIQ generation requires three interrupt controller functional clock cycles to be generated.

The interrupt source calculation (priority handling and SIR register updating) is made in the background. The interrupt handler stalls any access to the SIR register until the interrupt source calculation is done.

To avoid unnecessary stalling, the interrupt source calculation does not depend on the number of incoming interrupts active at the same time. It depends instead on the total number of incoming interrupts. For the MPU interrupt handler, the interrupt source calculation requires 10 cycles, regardless of the number of active incoming interrupts. For the DSP side, the interrupt source calculation requires 6 cycles.

2.1.5 Interrupt Handler Sleep Mode

The MPU can shut off the system clock to save power. The system is awakened by an asynchronous event, which can be an interrupt. For proper system operation, the interrupt controller must ensure that no interrupt is generated when the system is going to power-saving (big or deep sleep) mode.

Before going to idle, the host sends an idle request (IDLE_REQ) to the interrupt controller. When the interrupt controller is ready to go to sleep, it sends back an idle acknowledge (IDLE_ACK) to signal that its functional clock can be safely shut down. IDLE_REQ/IDLE_ACK handshake is used to prevent IRQ/FIQ from occurring when the system goes into idle mode. Two different protocols manage the handshake: FORCE_WAKEUP and SMART_IDLE. The protocol used is defined by the value of the IDLE_MODE field in the OCP_CFG register (see Table 18).

2.1.6 Going to Sleep

The procedure used for going to sleep depends on the IDLE_MODE value.

Smart Idle Mode

In SMART_IDLE mode, when the IDLE_REQ signal is asserted (high level), the interrupt handler goes into an idle state on the next functional clock cycle, if no IRQ/FIQ is currently pending. If incoming interrupts are pending, they are processed before going into idle state. As long as the interrupt controller is in idle state, all incoming interrupts are ignored but still stored into the ITR register.

As soon as it is ensured that no IRQ/FIQ will be generated, the interrupt controller asserts the IDLE_ACK signal.

Upon receiving the IDLE_ACK signal, the power management is aware that no more interrupts are currently being processed and that clocks can safely be turned off.

Force Wake-up Mode

In force wake-up mode, when the IDLE_REQ signal is asserted, the interrupt controller asserts IDLE_ACK on the next OCP clock cycle regardless of the state of the current incoming interrupt and of any interrupt possibly being processed. It immediately masks all incoming interrupts and deasserts the IRQ/FIQ signal.

Waking Up

When the system wakes up, power management turns the clocks on again, and then deasserts the IDLE_REQ signal. At that moment, the interrupt controller becomes aware that its clocks are turned on. It deasserts IDLE_ACK and comes out of idle state. From this point on, IRQ/FIQ generation becomes possible again.

2.1.7 External Interrupt Asynchronous Path

When the system is asleep, the interrupt controller functional clock is shut off, the interrupt controller is in idle state, and IRQ/FIQ generation is no longer possible. However, in this case, a dedicated asynchronous path is activated in the interrupt controller to propagate interrupts to the clock manager.

Whenever the interrupt controller is idle, and an unmasked interrupt occurs, the interrupt controller asynchronously asserts a wakeup signal. Basically, this signal is an OR of all unmasked ITR bits. This signal notifies the clock manager that an external wake-up event has occurred.

When the interrupt controller functional clock is eventually turned on, the interrupt controller goes out of idle state and generation of IRQ/FIQ is enabled again. The interrupt waking up the system is *not* lost (although reading the SIR register does not necessarily give the waking interrupt number, if another higher priority interrupt was pending when the system was awakened).

2.1.8 Interrupt Global Masking

To avoid interrupting the software during the execution of critical routines, a global masking mechanism is implemented, controlled by the control register GLOBAL_MASK bit (see Figure 3) and the interrupt controller output signal IRQ_SECURE_MASK_N.

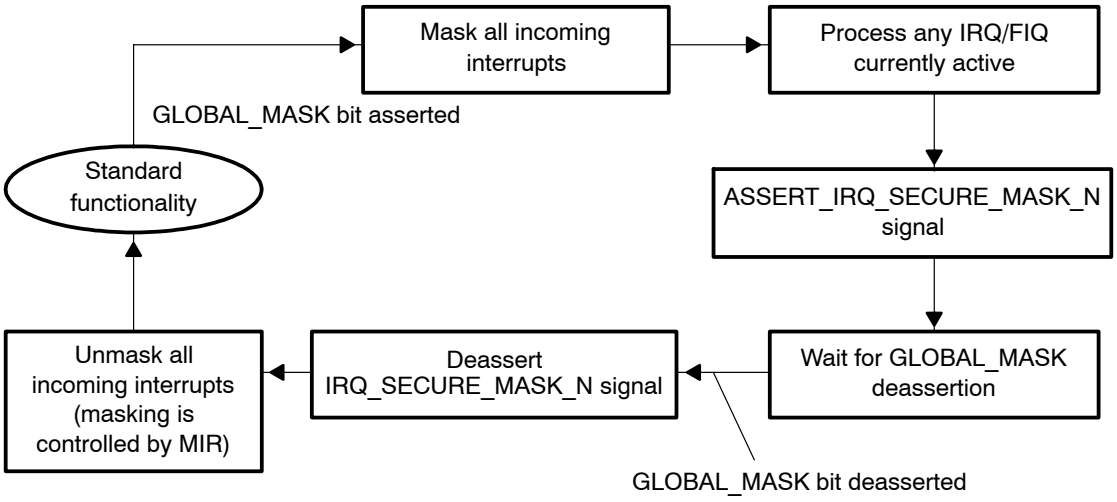
As long as the GLOBAL_MASK bit is set, all incoming interrupts are registered into ITR, but not processed. If IRQ or FIQ is asserted, the interrupt controller waits for the current IRQ/FIQ routine to complete.

When both IRQ and FIQ are deasserted, the interrupt controller asserts (sets to 0) the IRQ_SECURE_MASK_N signal. As long as this signal is asserted the interrupt controller must not assert IRQ or FIQ. Incoming interrupts must still be stored in the ITR register.

Upon deassertion of the GLOBAL_MASK bit, the interrupt controller first deasserts the IRQ_SECURE_MASK_N signal, then releases global masking (masking goes back under the control of the MIR registers), and returns to normal operation mode.

The IRQ_SECURE_MASK_N signal is synchronous of the interrupt controller functional clock.

Figure 3. Global Mask Bit Effect



2.1.9 Interrupt Spying

The immediate value of the ITR register, whose number is binary coded on the SPY_ITR_SEL input port asynchronously, is put on the SPY_ITR_OUT output port, regardless of the interrupt controller state.

2.1.10 Interrupt Controller Registers

All addresses in this section are byte addresses.

Regardless of register width, all addresses are aligned on 32-bit boundaries. This means that the address mapping does not depend on register width and that there are holes in the mapping for widths smaller than 32.

The mapping below describes only the basic set of registers. This mapping is duplicated according to the host width and number of incoming interrupts.

To ease software development and hardware implementation, duplication occurs at offset 0x100. This means that 64 locations stay unused at the end of each register section. Given that there are 11 address bits, this allows for a maximum of 8 sections, or 256 interrupts, if the data path is 32 bits wide.

The registers SIR_IRQ, SIR_FIQ, CONTROL, STATUS, OCP_CFG and INTH_REV are not duplicated, and are only available at offset 0x10, 0x14, 0x18, 0xA0, 0xA4, and 0xA8 (in the first section). These register addresses are reserved in the other sections. Writing at these locations has no effect, and reading them returns 0.

All register accesses are little endian.

All unused register bits are read as 0.

The OMAP programming model ensures that no posted write occurs in any writeable register.

Table 6 lists the interrupt controller registers. Table 7 through Table 20 describe the register bits.

Table 6. Interrupt Controller Registers

Register	Description	R/W	Offset
ITR	Interrupt register	R/W	0x00
MIR	Interrupt mask register	R/W	0x04
RESERVED	Reserved	R	0x08
RESERVED	Reserved	R	0x0C
SIR_IRQ	Interrupt encoded source register (IRQ)	R	0x10
SIR_FIQ	Interrupt encoded source register (FIQ)	R	0x14
CONTROL	Interrupt control register	R/W	0x18
	ILR Registers		
ILR0	Interrupt priority level register bit 0	R/W	0x1C
ILR1	Interrupt priority level register bit 1	R/W	0x20
ILR2	Interrupt priority level register bit 2	R/W	0x24
ILR3	Interrupt priority level register bit 3	R/W	0x28
ILR4	Interrupt priority level register bit 4	R/W	0x2C
ILR5	Interrupt priority level register bit 5	R/W	0x30
ILR6	Interrupt priority level register bit 6	R/W	0x34
ILR7	Interrupt priority level register bit 7	R/W	0x38
ILR8	Interrupt priority level register bit 8	R/W	0x3C
ILR9	Interrupt priority level register bit 9	R/W	0x40

Table 6. Interrupt Controller Registers (Continued)

Register	Description	R/W	Offset
	ILR Registers (Continued)		
ILR10	Interrupt priority level register bit 10	R/W	0x44
ILR11	Interrupt priority level register bit 11	R/W	0x48
ILR12	Interrupt priority level register bit 12	R/W	0x4C
ILR13	Interrupt priority level register bit 13	R/W	0x50
ILR14	Interrupt priority level register bit 14	R/W	0x54
ILR15	Interrupt priority level register bit 15	R/W	0x58
ILR16	Interrupt priority level register bit 16	R/W	0x5C
ILR17	Interrupt priority level register bit 17	R/W	0x60
ILR18	Interrupt priority level register bit 18	R/W	0x64
ILR19	Interrupt priority level register bit 19	R/W	0x68
ILR20	Interrupt priority level register bit 20	R/W	0x6C
ILR21	Interrupt priority level register bit 21	R/W	0x70
ILR22	Interrupt priority level register bit 22	R/W	0x74
ILR23	Interrupt priority level register bit 23	R/W	0x78
ILR24	Interrupt priority level register bit 24	R/W	0x7C
ILR25	Interrupt priority level register bit 25	R/W	0x80
ILR26	Interrupt priority level register bit 26	R/W	0x84
ILR27	Interrupt priority level register bit 27	R/W	0x88
ILR28	Interrupt priority level register bit 28	R/W	0x8C
ILR29	Interrupt priority level register bit 29	R/W	0x90
ILR30	Interrupt priority level register bit 30	R/W	0x94

Table 6. Interrupt Controller Registers (Continued)

Register	Description	R/W	Offset
	ILR Registers (Continued)		
ILR31	Interrupt priority level register bit 31	R/W	0x98
	Status and ID Register		
SISR	Software interrupt set register	W	0x9C
STATUS	Status register	R	0xA0
OCP_CFG	OCP configuration register	R/W	0xA4
INTH_REV	Interrupt controller revision ID	R	0xA8

2.1.11 Implementation

The MPU level 2 interrupt controller can handle 128 interrupts and has 32 bits wide registers. The DSP level 2.1 interrupt controller can handle 64 interrupts, and has 16 bits registers.

Both interrupt controllers have four full sets of registers.

In each set, the MPU interrupt controller has 32 ILR, mapped in the ranges 0x1C–0x98, 0x11C–0x198, 0x21C–0x298, and 0x31C–0x398. Each of these registers has nine relevant bits (read on the 9 LSB). In ILR, register 0 (@0x1C) controls interrupt 0, and register 127 (@0x398) controls interrupt 127.

In each set, the DSP interrupt controller has 16 ILR, mapped in the ranges 0x1C–0x58, 011C–0x158, 0x21C–0x258, and 0x31C–0x358. Each of these registers has eight relevant bits (read on the 8 LSB).

Table 7. Interrupt Input Register (ITR)

	DW-1†	**DW-2**	**DW-3**	**DW-4**	**DW-5**	….	**2**	**1**	**0**
@0x00	IRQ-DW-1						IRQ_2	IRQ_1	IRQ_0
Access	RW	RW	RW	RW	RW	RWs	RW	RW	RW
Default	0	0	0	0	0	0s	0	0	0

† DW = 32 for the MPU interrupt handler; DW =16 for the DSP interrupt handler.

If the incoming interrupt line number *n* is detected, the corresponding bit is set to 1. If the number of incoming interrupts is lower than DW, the MSB of this register is set to 0.

In case of an edge-sensitive interrupt, when the host accesses the SIR_IRQ or SIR_FIQ register, the bit corresponding to the currently active interrupt is reset. For level-sensitive interrupts, this bit is simply the interrupt line current state (after resynchronization).

The host can also clear each bit individually. To do this, it must write a 0 to the corresponding bit. Write access to this register is stalled as long as the actual register bit (on the functional clock domain) is not 0. Writing a 1 to any ITR bit has no effect.

This register is a status register that gives the state of every incoming interrupt regardless of which interrupt is currently being processed. Coherency between this register and the SIR_IRQ/SIR_FIQ registers is not ensured.

Table 8. Mask Interrupt Register (MIR)

	DW-1†	**DW-2**	**DW-3**	**DW-4**	**DW-5**	….	**2**	**1**	**0**
@0x04	MIR-DW-1						MIR_2	MIR_1	MIR_0
Access	RW	RW	RW	RW	RW	RWs	RW	RW	RW
Default	1	1	1	1	1	1s	1	1	1

† DW = 32 for the MPU interrupt handler; DW = 16 for the DSP interrupt handler.

This register masks each incoming interrupt by setting the corresponding bit to 1.

MIR operates after ITR. This means that occurrences of incoming interrupts are always stored in ITR.

Masking a particular interrupt does not deassert either IRQ or FIQ if it is active owing to this interrupt (providing that no other interrupts are active at the same time), regardless of whether the interrupt is level- or edge-sensitive. It only prevents subsequent incoming interrupts from being taken into account.

The write into this register must be done while there are no pending interrupts.

Table 9. Source IRQ Register (SIR_IRQ)

	CODE_NB_IT–1 … **0**†
@0x10	Active IRQ number (hexadecimal)
Access	R
Default	0

† CODE_NB_IT = 7 for the MPU interrupt handler; CODE_NB_IT = 6 for the DSP interrupt handler.

This register stores the currently active IRQ interrupt number (in hexadecimal). Reading this register clears the corresponding bit in the ITR register if the interrupt is set as edge sensitive. Unused bits are read as 0.

In case a priority calculation is ongoing for IRQ, read to this register is stalled until priority calculation completion.

Table 10. Source FIQ Register (SIR_FIQ)

	CODE_NB_IT−1 … 0
@0x14	Active FIQ number (hexadecimal)
Access	R
Default	0

This register stores the currently active FIQ interrupt number (in hexadecimal). Reading this register clears the corresponding bit in the ITR register if the interrupt is set as edge sensitive. Unused bits are read as 0.

In case a priority calculation is ongoing for FIQ, read to this register is stalled until priority calculation completion.

Table 11. Control Register

	DW-1…3	**2**	**1**	**0**
@0x18	Reserved	GLOBAL_MASK	NEW_FIQ_AGR	NEW_IRQ_AGR
Access	R	RW	W	W
Default	0	0	0	0

Table 12. Control Register Bit Descriptions

Name	Function
GLOBAL_MASK	Setting this bit to 1 has the following effect (in this order): All incoming or software generated (through SISR) interrupts are still stored in ITR, but neither FIQ nor IRQ are. When this bit is set *and* IRQ and FIQ are inactive *and* no IRQ/FIQ are to occur, the IRQ_SECURE_MASK_N signal is asserted. Resetting this bit has the following effect (in this order): IRQ_SECURE_MASK_N signal is deasserted. FIQ and IRQ generation become possible again (and all pending interrupts are processed normally). This behavior is shown in Figure 3.
NEW_FIQ_AGR	New FIQ agreement Writing a 1 resets FIQ output and enables a new FIQ generation. The FIQ output is reset if the corresponding bit of the ITR of the treated interrupt has been cleared or masked. Write access to this register bit is stalled until FIQ output is cleared. This bit is write only. Reading it always returns 0.
NEW_IRQ_AGR	New IRQ agreement Writing a 1 resets IRQ output and enables a new IRQ generation. The IRQ output is reset if the corresponding bit of ITR of the treated interrupt has been cleared or masked. Write access to this register bit is stalled until IRQ output is cleared. This bit is write only. Reading it always returns 0.

To allow for future evolution, when the software writes into this register all reserved bits must be written as 0.

Table 13. Interrupt Level Register (ILR)

	CODE_NB_IT+1 … 2†	**1**	**0**
@0x1C–0x98	PRIORITY	SENS_nEDGE	FIQ_nIRQ
Access	RW	RW	RW
Default	0	0	0

† CODE_NB_IT = 7 for the MPU interrupt handler; CODE_NB_IT = 6 for the DSP interrupt handler.

Table 14 describes the interrupt level register bits.

Table 14. Interrupt Level Register Bit Descriptions

Name	Function
PRIORITY	Defines the priority level when the corresponding interrupt is routed to IRQ or FIQ. 0 is the highest priority level. All bits set to 1 is the lowest priority level.
SENS_nEDGE	0: The corresponding interrupt is falling-edge sensitive. 1: The corresponding interrupt is low-level sensitive.
FIQ_nIRQ	0: The corresponding interrupt is routed to IRQ. 1: The corresponding interrupt is routed to FIQ.

There is one ILR per incoming interrupt.

Assuming that all interrupts have the same priority level and are active at the same time, the default order is IRQ_N, IRQ_N−1 , IRQ_N−2, … , IRQ_0.

Writes into this register must be done while there are no pending interrupts.

Table 15. Software Interrupt Set Register (SISR)

	DW-1†	**DW-2**	**DW-3**	**DW-4**	**DW-5**	….	**2**	**1**	**0**
@0x9C	$SISR_{DW-1}$						$SISR_2$	$SISR_1$	$SISR_0$
Access	W	W	W	W	W	Ws	W	W	W
Default	0	0	0	0	0	0s	0	0	0

† DW = 32 for the MPU interrupt handler; DW = 16 for the DSP interrupt handler.

Writing a 1 to any bit generates an interrupt to the host if the corresponding ILR register is set as edge trigger. Otherwise, no interrupt is generated.

A zero is always returned from a read to this register. External interrupts are merged (ORed) with the software interrupts before masking occurs.

Table 16. Status Register

	DW-1… **2**†	**0**
@0xA0	Reserved	RESET_DONE
Access	R	R
Default	0	1

† DW = 32 for the MPU interrupt handler; DW = 16 for the DSP interrupt handler.

Table 17. Status Register Bit Description

Name	Function
RESET_DONE	0: Reset has not occurred (functional logic is currently being reset). 1: Reset has occurred. This bit concerns only the functional and OCP clock domains. When OCP clock domain is being reset, OCP accesses are stalled and this bit cannot be read.

Table 18. OCP_CFG Register

	DW-1… 5†	4	3	2	1	0
@0xA4	RESERVED	IdleMode		RE-SERVED	SOFT RESET	AUTOIDLE
Access		RW		R	W	R/W
Default	0	0	0	0	0	0

† DW = 32 for the MPU interrupt handler; DW = 16 for the DSP interrupt handler.

Table 19. OCP_CFG Register Bit Descriptions

Name	Function
IDLEMODE	Power management REQ/ACK control (see Section 2.1.5) 00: Force wake-up. An idle request is acknowledged unconditionally. 01: Reserved. Do not use. 10: Smart idle. Acknowledgement of the idle request occurs only if no incoming interrupts are pending. 11: Reserved. Do not use.
SOFTRESET	Writing 1 to this bit generates a software reset of the module. Both OCP and functional clock domain are reset. This bit is write only. Reading it always returns 0. To check reset completion, use the RESET_DONE bit in the status register.
AUTOIDLE	Internal OCP clock gating strategy. 0: OCP clock is free-running. 1: Automatic OCP clock gating is applied based on OCP interface activity.

To allow for future evolution, when the software writes into this register all reserved bits must be written as 0. These bits are currently read-only and are always read as 0, no matter what the value written. Future evolution, however, may have one or more of them as read/write.

Table 20. Interrupt Revision Register (INTH_REV)

	DW-1… 8†	7	6	5	4	3	2	1	0
@0xA8	Reserved	MAJOR_REV				MINOR_REV			
Access	R	R				R			
Default	0	Revision dependent				Revision dependent			

† DW = 32 for the MPU interrupt handler; DW = 16 for the DSP interrupt handler.

This register provides the revision number of the interrupt controller block.

3 Level 1 MPU Interrupt Handler

3.1 Description

The MPU interrupt handler allows up to 32 hosts that generate interrupts to connect to the MPU, which can accept only two interrupts: fast interrupt request (FIQ) and low-priority interrupt request (IRQ). You can also program the interrupt handler to assign different priorities and mask each interrupt, and you can program each interrupt line to be either edge-triggered or level-sensitive.

For power management, the clock manager can turn off the interrupt handler functional clock. A handshaking protocol is defined for the clock and reset module to idle or wake up the interrupt handler.

A key difference between the level 1 and level 2 MPU interrupt handler is that there is only one set of registers, some of which differ from the level 2 handlers.

3.1.1 Interrupt Control and Configuration

If an interrupt occurs, the ITR register stores the incoming interrupt in the corresponding bit. When there are several incoming interrupts, the MPU interrupt handler compares the priority level of the interrupts before sending an IRQ or FIQ to the MPU core. The selected interrupt's number is stored in SIR_IRQ or SIR_FIQ for the MPU to determine which interrupt service routine to execute. Reading either of these registers by the MPU resets the corresponding bit in ITR. The MPU can also clear each bit individually in ITR by writing a 0 to the corresponding bits. Writing a 1 keeps its previous value.

Each incoming interrupt can be masked individually by setting the corresponding bit in MIR to 1.

One interrupt level register (ILR) is associated with each incoming interrupt. ILR determines whether the interrupt is to be edge-triggered or level-sensitive and assigns it a priority level: 0 (the highest priority), 1, ... 30, 31 (the lowest priority). If several interrupts have the same priority level assigned, they are serviced in a predefined order: IRQ_31, IRQ_30, ..., IRQ_1, IRQ_0. ILR also allows routing each of the 32 interrupts to either FIQ or IRQ.

The IRQ or FIQ outputs can be reset by writing a 1 to the corresponding bit of the CONTROL_REG to enable new IRQ or FIQ generation. The writing also clears the SIR_IRQ or SIR_FIQ register. The corresponding bit in the ITR must be cleared before writing to the CONTROL_REG.

3.1.2 Software Interrupt

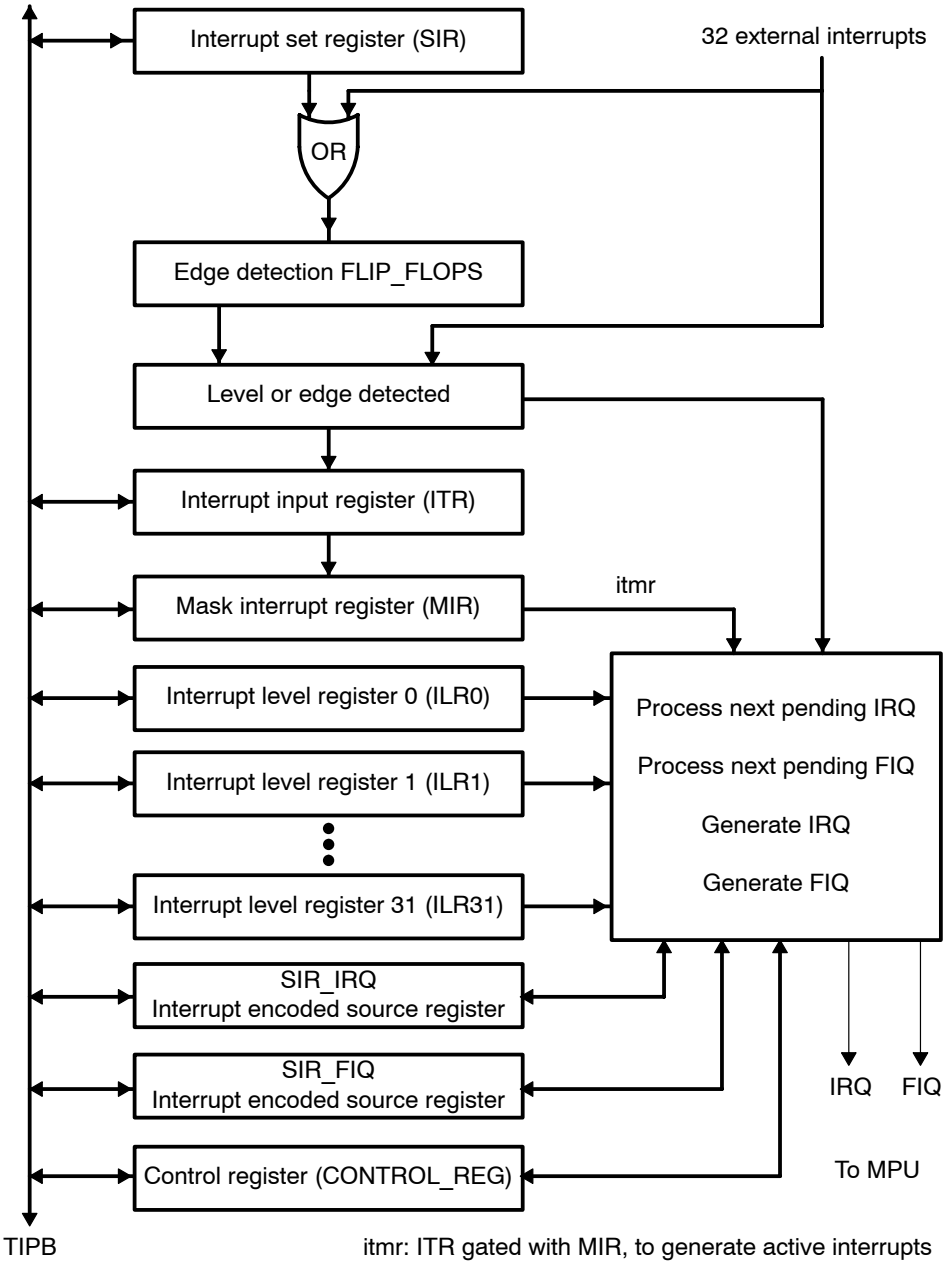
The interrupt handler also provides a 32-bit software interrupt register (SIR), which corresponds to the same 32-bit external interrupt lines. Writing a 0 followed by a 1 to the targeted bit generates an interrupt if the corresponding ILR is set to edge-sensitive; otherwise, no interrupt is generated.

An external interrupt request and an internal software request are merged before being sent to the interrupt handler to be serviced. The software interrupt register is always read as 0. You can use this software interrupt mechanism to simulate an external interrupt and test the corresponding interrupt driver as long as the interrupt line is programmed as edge-sensitive.

All internal interrupts are brought to the OMAP 3.2 subchip level to provide maximum flexibility for system integration. You can reorganize, regroup, add, or delete the interrupt inputs for your applications.

Figure 4 shows the MPU interrupt handler.

Figure 4. MPU Interrupt Handler



3.1.3 Interrupt Sequence

Table 21 shows the MPU interrupt sequence for an IRQ interrupt only. The FIQ interrupt sequence is identical.

Table 21. MPU Interrupt Sequence

Step	Interrupt Handler Action	MPU Action
One or several interrupts occur that set the corresponding bits in ITR	If one active interrupt occurs and the IRQ is not already active, the interrupt handler sends an IRQ. If several active interrupts occur, the interrupt handler must locate the interrupt with the highest priority. If an IRQ is not already active, the interrupt handler sends an IRQ.	
Processing the interrupt	When the IRQ is sent, SIR_IRQ is updated and the priority resolver is reset.	The MPU must read SIR_IRQ to determine the interrupt line being serviced. Then the MPU runs the corresponding subroutine.
Finish the interrupt		1. The MPU must first clear the interrupt bit in ITR (by writing a 0 in the corresponding bit or by reading SIR_IRQ. 2. For a level-sensitive interrupt, the level must be removed from the source that keeps the interrupt request low (active) for the next interrupt to occur. 3. Sets CONTROL_REG.NEW_IRQ_AGR to reset IRQ output and SIR_IRQ, thus allowing a new IRQ generation.

3.1.4 Interrupt Handler Software

To process edge-triggered and level-sensitive interrupts correctly, the following sequences must occur in the system. Only the IRQ treatment is described here. The FIQ treatment is exactly identical.

3.1.5 Edge-Triggered Interrupts

1) The interrupt handler module receives incoming one or more interrupts from outside OMAP and registers it in the interrupt register (ITR).
2) If there are several active incoming interrupts, the interrupt handler determines the highest priority interrupt and puts it in the N_IRQ register, which is invisible to the software programmer.

3) If the IRQ (interrupt from interrupt handler to the MPU) is not active, the interrupt handler sends the interrupt in N_IRQ register to the MPU as an IRQ signal. Then the source IRQ register (SIR_IRQ) is updated with contents of the N_IRQ register (the SIR_IRQ contains encoded information that conveys the interrupt line number of the IRQ).

4) The MPU recognizes the interrupt and jumps to the interrupt service routine (ISR) code.

5) Within the ISR code, the MPU reads the SIR_IRQ in the interrupt handler to determine which interrupt line caused the interrupt. The MPU executes specific code appropriately.

6) When MPU reads the SIR_IRQ, the corresponding bit is reset in ITR of interrupt handler module. The IRQ is still active.

7) The MPU, when it is about to exit the ISR routine, writes a 1 to CONTROL_REG.NEW_IRQ_AGR to deassert the IRQ going to MPU and to enable a new IRQ generation.

8) The MPU exits the ISR and continues its normal code execution.

9) When CONTROL_REG.NEW_IRQ_AGR is written, the process jumps to Step 2.

3.1.6 Level-Sensitive Interrupts

1) The interrupt handler module receives one or more incoming interrupts from outside OMAP. Level-sensitive interrupts are not registered, but are used in the logic as is. The interrupt handler assumes that the peripheral will not deassert the level-sensitive incoming interrupts until it is told to do so by the MPU.

2) The interrupt handler determines the highest priority interrupt and puts it in the N_IRQ register.

3) If the IRQ (interrupt from interrupt handler to the MPU) is not active, the interrupt handler sends the highest priority interrupt (interrupt in N_IRQ register) to the MPU as IRQ signal. Then the SIR_IRQ is updated with contents of N_IRQ register (the SIR_IRQ register contains encoded information that conveys the interrupt line number of IRQ). If the IRQ is active, which means CONTROL_REG.NEW_IRQ_AGR has not been set by MPU, the incoming IRQ has to wait until IRQ is not active.

4) The MPU recognizes the interrupt and jumps to the ISR code.

5) Within the ISR code, the MPU reads the SIR_IRQ in the interrupt handler to determine which interrupt line caused the interrupt.

6) The ISR code must be capable of doing one of the following things:

   - Letting the peripheral know that the interrupt generated by it has been serviced so the peripheral can deassert the interrupt request
   - Writing to interrupt handler mask interrupt register (MIR) to mask the level-sensitive interrupt

     Here the peripheral has to deassert the interrupt before the mask to the interrupt can be removed, so that the next interrupt can be recognized.

     If the peripheral deasserts the interrupt before the code in ISR tells it to, then the behavior is unpredictable and the interrupt may be lost.

7) The MPU must write a 1 to CONTROL_REG.NEW_IRQ_AGR when it is about to exit the ISR routine to deassert the IRQ going to MPU and to enable a new IRQ generation.

8) The MPU exits the ISR and continues its normal code execution.

9) When CONTROL_REG.NEW_IRQ_AGR is written into by MPU, the process jumps to Step 2.

3.1.7 Registers

Table 22 lists the registers available to handle interrupts. Table 23 through Table 30 provide register bit descriptions. All these registers are 32 bits wide and are controlled directly by the private TIBP bus. To determine the base address of these registers, see the Applications Processor Data Manual (SPRS231).

Table 22. Interrupt Registers

Name	Description	R/W	Offset
ITR	Interrupt register	R/W	0x00
MIR	Interrupt mask register	R/W	0x04
SIR_IRQ	Interrupt encoded source register for IRQ	R	0x10
SIR_FIQ	Interrupt encoded source register for FIQ	R	0x14
CONTROL_REG	Interrupt control register	R/W	0x18
ILRx	Interrupt level register	R/W	0x1C + 0x4 * x
SIR	Software interrupt set register	R/W	0x9C
GMR	Global mask interrupt register	R/W	0xA0

Table 23. Interrupt Register (ITR)

Offset: 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:0	ACT_IRQ	Sets corresponding bit in ITR for edge-sensitive and level-sensitive interrupts.	R/W	0x0000 0000

When the MPU accesses SIR_IRQ or SIR_FIQ, the ITR bit corresponding to the interrupt that has requested MPU action is reset. The MPU can also clear each bit individually by writing a 0 to the corresponding bits at the ITR address. Writing a 1 to a bit keeps its previous value. You can use the individual clearing just before the MPU unmasks some interrupts and thus ignore some interrupt occurrences.

Table 24. Mask Interrupt Register (MIR)

Offset: 0x04				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:0	IRQ_MSK	Masks each incoming interrupt individually	R/W	0xFFFF FFFF

You can mask each incoming interrupt individually with this register by setting the corresponding bit to 1. This register operates after the interrupt register, which means that occurrences of incoming interrupts are always stored in the interrupt register.

Table 25. Interrupt Encoded Source Register for IRQ (SIR_IRQ)

Offset: 0x10				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:5	Reserved			
4:0	IRQ_NUM	Indicates encoded interrupt number that has an IRQ request. Reading this register clears the corresponding bit in the ITR register if the interrupt is set as edge-sensitive.	R	00000

Table 26. Interrupt Encoded Source Register for FIQ (SIR_FIQ)

Offset: 0x14				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:5	Reserved			
4:0	FIQ_NUM	Indicates the encoded interrupt number that has an FIQ request. Reading this register clears the corresponding bit in the ITR register if the interrupt is set as edge-sensitive.	R	00000

Table 27. Interrupt Control Register (CONTROL_REG)

Offset: 0x18				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:2	Reserved			
1	NEW_FIQ_AGR	New FIQ agreement. Writing a 1 resets the FIQ output, clears the SIR_FIQ, and enables a new FIQ generation. The corresponding bit of the ITR must be cleared first. Writing 0 has no effect.	R/W	0
0	NEW_IRQ_AGR	New IRQ agreement. Writing a 1 resets the IRQ output, clears the SIR_IRQ, and enables a new IRQ generation. The corresponding bit of the ITR must be cleared first. Writing 0 has no effect.	R/W	0

Table 28. Interrupt Level Register for Interrupt Number x (0 to 31) (ILRx)

Offset: 0x1C + 0x4 * Interrupt number				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:7	Reserved			
6:2	PRIORITY	Defines the priority level when the corresponding interrupt is routed to IRQ or FIQ. 0 is the highest priority level. 31 is the lowest priority level.	R/W	00000
1	SENS_LEVEL	0: The corresponding interrupt is falling-edge sensitive. 1: The corresponding interrupt is low-level sensitive.	R/W	0
0	FIQ	0: The corresponding interrupt is routed to IRQ. 1: The corresponding interrupt is routed to FIQ.	R/W	0

Table 29. Software Interrupt Set Register (SIR)

Offset: 0x9C				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:0	SIR	See below.	R/W	0x0000 0000

The user can program the SIR register to emulate the interrupt generation. The corresponding ILR must be programmed as edge-sensitive while using SIR register. The procedure to generate an edge-sensitive interrupt is: SIR(bit i) = 0 – SIR(bit i) = 1 (rising edge generated). SIR will not be cleared automatically (user must program it). A zero is always returned from a read to this register. External interrupts are merged with the software interrupts before they are sent to the MIR mask register for interrupt masking.

Table 30. Global Mask Interrupt Register (GMR)

Offset: 0xA0				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:1	Reserved			
0	GLOBAL_MASK	When 1, the interrupt handler module and the output signal INT_DIS are set to 1. (INT_DIS is the output signal that indicates the interrupt handler has been disabled.) For power management, CLK&RST can turn off the interrupt handler functional clock. A handshaking protocol is defined for the CLK&RST module to idle or wake up the interrupt handler.	R/W	0

4 DSP Level 1 and Level 2.0 Interrupt Handler and Interface

4.1 Description

DSP interrupts are handled through two cascaded interrupt controllers:

- The DSP interrupt interface (level 1 handler) handles 24 interrupts.
- The DSP interrupt handler (level 2.0 handler) handles 16 level 2 interrupts that are routed to the DSP interrupt interface through low-priority interrupt request (IRQ) and fast-priority interrupt request (FIQ).

4.1.1 Interrupt Control and Configuration

If an interrupt occurs, the DSP_ITR register stores the incoming interrupt in the corresponding bit. When there are several incoming interrupts, the DSP interrupt handler compares the priority level of the interrupts before sending an IRQ or FIQ to the DSP core. The selected interrupt number is stored in DSP_SIR_IRQ or DSP_SIR_FIQ for the DSP to determine which interrupt service routine to execute. Reading either of these registers by the DSP resets the corresponding bit in DSP_ITR. The DSP can also individually clear each bit in DSP_ITR by writing a 0 to the corresponding bits. Writing a 1 keeps its previous value.

Each incoming interrupt can be masked individually by setting the corresponding bit in DSP_MIR to 1.

One interrupt level register (DSP_ILR) is associated with each incoming interrupt. The DSP_ILR sets whether the interrupt is to be edge-triggered or level-sensitive and assigns it a priority level: 0 (the highest priority level), 1, ... 14, 15 (the lowest priority level). If several interrupts have the same priority level assigned, they are serviced in a predefined order: IRQ_15, IRQ_14, ..., IRQ_1, IRQ_0. The DSP_ITR also allows routing of each of the 16 interrupts to either of the two DSP core input interrupts: FIQ or IRQ.

The IRQ or FIQ outputs can be reset by writing a 1 to the corresponding bit of DSP_CONTROL_REG to enable new IRQ or FIQ generation. The writing also clears DSP_SIR_IRQ or DSP_SIR_FIQ. The corresponding bit in DSP_ITR must be cleared before writing to the DSP_CONTROL_REG.

4.1.2 Software Interrupt

The interrupt handler also provides a 16-bit software interrupt register (DSP_SISR), which corresponds to the same 16-bit external interrupt lines. Writing a 0 followed by a 1 to the targeted bit generates an interrupt if the corresponding DSP_ILR is set to edge-sensitive; otherwise, no interrupt is generated.

An external interrupt request and an internal software request are merged before being sent to the interrupt handler to be serviced. The software interrupt register is always read back with a 0. You can use this software interrupt mechanism to simulate an external interrupt and test the corresponding interrupt driver as long as the interrupt line is programmed as edge-sensitive.

4.1.3 Latency

The DSP interrupt handler resides on the 16-bit DSP private TI peripheral bus (TIPB) and runs at half the frequency of the DSP clock. The latency from an incoming interrupt to FIQ/IRQ generation depends on the number of interrupts arriving at the same time. If there is only one, the latency is 5 clock cycles. If more than one interrupt become active at the same time and are routed to the same FIQ/IRQ, the latency can reach 3 + N*2 cycles, where N is the number of incoming interrupts.

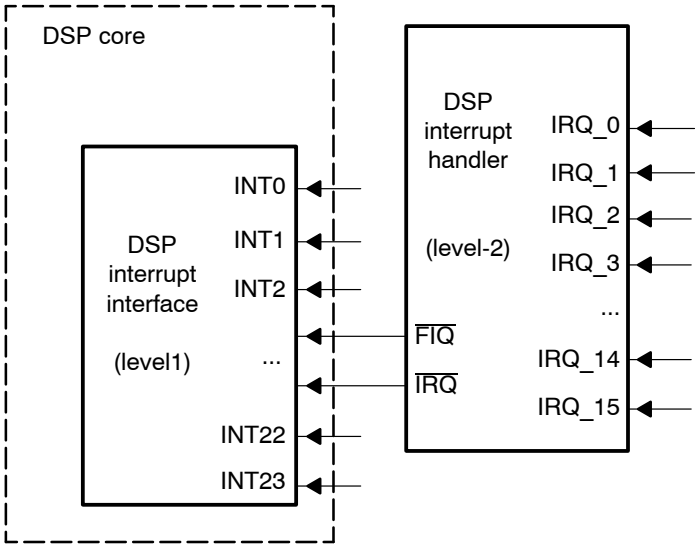
All interrupts for the DSP subsystem and OMAP subsystem are brought to the OMAP subchip boundary to provide maximum flexibility for system integration.

4.1.4 Interrupt Interface

The DSP interrupt interface augments DSP interrupt handler capability by enabling you to define edge-triggered or level-sensitive implementations for each of external interrupt lines. The DSP interrupt interface allows you to program the edge- or level-sensitivity of the two level 1 interrupts where IRQ and FIQ are routed.

Figure 14–1 shows an example of interrupt handling.

Figure 5. An Example of DSP Interrupt Handling



4.1.5 Interrupt Sequence

Table 31 shows the DSP interrupt sequence for an IRQ interrupt only. The FIQ interrupt sequence is identical.

Table 31. DSP Interrupt Sequence

Step	Interrupt Handler Action	DSP Action
One or several interrupts occur that set the corresponding bits in DSP_ITR register	If one active interrupt occurs and the IRQ is not already active, the interrupt handler sends an IRQ. If several active interrupts occur, the interrupt handler must locate the interrupt with the highest priority. If an IRQ is not already active, the interrupt handler sends an IRQ.	
Processing interrupt	When the IRQ is sent, the DSP_SIR_IRQ is updated and the priority resolver is reset.	The DSP must read the DSP_SIR_IRQ to determine the interrupt line being serviced. Then the DSP runs the corresponding subroutine.
Finish the interrupt		The DSP must first clear the interrupt bit in the DSP_ITR (by writing a 0 in the corresponding bit or by reading the DSP_SIR_IRQ). For a level-sensitive interrupt, the level must be removed for the next interrupt to occur. Sets the NEW_IRQ_AGR of the DSP_CONTROL_REG to reset IRQ output and the DSP_SIR_IRQ, thus allowing a new IRQ generation.

4.1.6 Interrupt Handler Software

To process edge-triggered and level-sensitive interrupts correctly, the following sequences must occur in the system. Here the sequence is described only for the IRQ interrupt; the FIQ sequence is exactly identical.

4.1.7 Edge-Triggered Interrupts

1) The DSP interrupt handler module receives one or more incoming interrupts from outside OMAP and registers it in the DSP interrupt register (DSP_ITR).

2) The DSP interrupt handler determines the highest priority interrupt and puts it in the N_IRQ register, which is not seen by software.

3) If the IRQ (interrupt from DSP interrupt handler to the DSP) is not active, the DSP interrupt handler sends the highest priority interrupt (interrupt in N_IRQ register) to the DSP as an IRQ signal. Then the DSP_SIR_IRQ register is updated with contents of the N_IRQ register (the DSP_SIR_IRQ contains encoded information that conveys the interrupt line number of the IRQ).

4) The DSP recognizes the interrupt and jumps to the interrupt service routine (ISR) code.

5) Within the ISR code, the DSP reads the DSP_SIR_IRQ in the DSP interrupt handler to determine which interrupt line caused the interrupt. The DSP executes specific code appropriately.

6) When the DSP reads the DSP_SIR_IRQ, the corresponding bit is reset in the DSP_ITR of the DSP interrupt handler module. The IRQ is still active.

7) The DSP writes a 1 to new IRQ agreement bit (NEW_IRQ_AGR) in the DSP_CONTROL_REG when it is about to exit the ISR routine to deassert the IRQ going to the DSP and to enable a new IRQ generation.

8) The DSP exits the ISR and continues its normal code execution.

9) When the NEW_IRQ_AGR bit is written, the process jumps to Step 2.

4.1.8 Level-Sensitive Interrupts

1) The DSP interrupt handler module receives one or more incoming interrupts from outside OMAP. Level-sensitive interrupts are not registered, but are used in the logic as is. The DSP interrupt handler assumes that the peripheral will not deassert the level-sensitive incoming interrupts until it is told to do so by the DSP.

2) The DSP interrupt handler determines the highest priority interrupt and puts it in the N_IRQ register.

3) If the IRQ (interrupt from DSP interrupt handler to the DSP) is not active, the DSP interrupt handler sends the highest priority interrupt (interrupt in N_IRQ register) to the DSP as IRQ signal. Then the DSP_SIR_IRQ is updated with contents of N_IRQ register (the DSP_SIR_IRQ register contains encoded information that conveys the interrupt line number of IRQ).

4) The DSP recognizes the interrupt and jumps to the ISR code.

5) Within the ISR code, the DSP reads the DSP_SIR_IRQ in the DSP interrupt handler to determine which interrupt line caused the interrupt.

6) The ISR code must be capable of doing one of the following things:

   - Letting the peripheral know that the interrupt generated by it has been serviced so the peripheral can deassert the interrupt request
   - Writing to the DSP mask interrupt register (DSP_MIR) to mask the level-sensitive interrupt

     Here the peripheral must deassert the interrupt before the mask to the interrupt can be removed, so that the next interrupt can be recognized.

     If the peripheral deasserts the interrupt before the code in ISR tells it to do so, then the behavior is unpredictable and the Interrupt may be lost.

7) The DSP, when it is about to exit the ISR routine, must write a 1 to the NEW_IRQ_AGR in the DSP_CONTROL_REG to deassert the IRQ going to DSP and to enable a new IRQ generation.

8) The DSP exits the ISR and continues its normal code execution.

9) When the NEW_IRQ_AGR in the DSP_CONTROL_REG is written into by DSP, the process jumps to Step 2.

4.1.9 Registers

This section describes the DSP interrupt interface and DSP interrupt handler registers for the OMAP 3.2 hardware engine. DSP software configures these registers through the DSP private TIPB. To determine the base addresses for these registers, see the Multimedia Processor Interrupts Reference Guide (literature number SPRU757).

4.1.10 DSP Interrupt Interface

Table 32 lists the DSP interface registers. Table 33 and Table 34 provide register bit descriptions. All these registers are 16 bits wide and are controlled directly by the DSP private TIPB.

Table 32. DSP Interrupt Interface Registers

Name	Description	R/W	Offset
EDGE_EN_HI	Incoming interrupt high register	R/W	0x00
EDGE_EN_LO	Incoming interrupt low register	R/W	0x02

Table 33. Incoming Interrupt High Register (EDGE_EN_HI)

Offset: 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:8	Reserved			
7	HOST_INTERRUPT	Defines whether the host interrupt from DSP to MPU is edge-triggered or level-sensitive: 0: HOST_INTERRUPT is level-sensitive. 1: HOST_INTERRUPT is edge-sensitive.	R/W	0
6	NMI	Defines whether the nonmaskable interrupt is edge or level sensitive: 0: NMI is level-sensitive. 1: NMI is edge-sensitive.	R/W	0
5:0	CHx	Defines channel CHx as edge-triggered or level-sensitive, where CHx corresponds to interrupt channels: 0: CHx is level-sensitive. 1: CHx is edge-sensitive.	R/W	000000

Table 34. Incoming Interrupt Low Register (EDGE_EN_LO)

Offset: 0x02				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	CHx	This bit defines channel CHx as edge-triggered or level-sensitive, where CHx corresponds to interrupt channels: 0: CHx is level-sensitive. 1: CHx is edge-sensitive.	R/W	0x0000

4.1.11 DSP Interrupt Handler

Table 22 lists the DSP registers available to handle interrupts. Table 36 through Table 42 describe the register bits. All these registers are 16 bits wide and are controlled directly by the DSP private TIPB.

Table 35. DSP Interrupt Registers

Name	Description	R/W	Offset
DSP_ITR	DSP interrupt	R/W	0x00
DSP_MIR	DSP mask interrupt	R/W	0x02
DSP_SIR_IRQ	Interrupt encoded source for IRQ	R	0x04
DSP_SIR_FIQ	Interrupt encoded source for FIQ	R	0x06
DSP_CONTROL_REG	DSP interrupt control	R/W	0x08
DSP_SISR	DSP software interrupt set	R/W	0x0A
DSP_ILRx	DSP interrupt level for interrupt number x	R/W	0x0C + 0x2 * x

Table 36. Interrupt Register (DSP_ITR)

Offset: 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	ACT_IRQ	If edge-sensitive interrupt occurs, it stores the active line. The DSP can individually clear each bit by writing a 0 to the corresponding bit. Writing a 1 keeps its previous value.	R/W	0x0000

Table 37. Mask Interrupt Register (DSP_MIR)

Offset: 0x02				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	IRQ_MSK	Writing a 1 masks the corresponding interrupt. Each bit corresponds to one interrupt.	R/W	0xFFFF

Table 38. Interrupt Encoded Source Register for IRQ (DSP_SIR_IRQ)

Offset: 0x04				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:4	Reserved			
3:0	IRQ_NUM	Indicates the encoded interrupt number that has an IRQ request. Reading this register clears the corresponding bit in the DSP_ITR if the interrupt is set as edge-sensitive.	R	0000

Table 39. Interrupt Encoded Source Register for FIQ (DSP_SIR_FIQ)

Offset: 0x06				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:4	Reserved			
3:0	FIQ_NUM	Indicates the encoded interrupt number that has an FIQ request. Reading this register clears the corresponding bit in the DSP_ITR if the interrupt is set as edge-sensitive.	R	0000

Table 40. Interrupt Control Register (DSP_CONTROL_REG)

Offset: 0x08				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:2	Reserved			
1	NEW_FIQ_AGR	Writing a 1 resets the FIQ output, clears the source FIQ register, and enables a new FIQ generation, reset by internal logic. The corresponding bit of DSP_ITR must be cleared first.	R/W	0
0	NEW_IRQ_AGR	Writing a 1 resets an IRQ output, clears the source IRQ register, and enables a new IRQ generation, reset by internal logic. The corresponding bit of DSP_ITR must be cleared first.	R/W	0

Table 41. Software Interrupt Set Register (DSP_SISR)

Offset: 0x0A				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	DSP_SISR	Writing a 0 followed by a 1 to any bit generates an interrupt to the DSP if the corresponding DSP_ILR register is set as edge-triggered; otherwise, no interrupt is generated.	R/W	0x0000

Table 42. Interrupt Level Register for Interrupt Number x [0−15] (DSP_ILRx)

Offset: 0x0C				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:6	Reserved			
5:2	PRIORITY	Defines the priority level when the corresponding interrupt is routed to IRQ or FIQ. 0 is the highest priority level. 15 is the lowest priority level.	R/W	0000
1	SENS_LEVEL	0: The corresponding interrupt is rising-edge sensitive. 1: The corresponding interrupt is high-level sensitive.	R/W	0
0	FIQ	0: The corresponding interrupt is routed to IRQ. 1: The corresponding interrupt is routed to FIQ.	R/W	0

Index

OMAP5912 Multimedia Processor Peripheral Interconnects Reference Guide

Literature Number: SPRU758A
March 2004

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.

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
		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 document describes various peripheral interconnects of the OMAP5912 multimedia processor.

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.

Related Documentation From Texas Instruments

The following documents describe the OMAP5910 device and related peripherals. Copies of these documents are available on the Internet at www.ti.com. *Tip:* Enter the literature number in the search box provided at www.ti.com.

OMAP5912 Multimedia Processor Device Overview and Architecture Reference Guide (literature number SPRU748) introduces the setup, components, and features of the OMAP5912 multimedia processor and provides a high-level view of the device architecture.

OMAP5912 Multimedia Processor OMAP 3.2 Subsystem Reference Guide (literature number SPRU749) introduces and briefly defines the main features of the OMAP3.2 subsystem of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor DSP Sybsystem Reference Guide (literature number SPRU750) describes the OMAP5912 multimedia processor DSP subsystem. The digital signal processor (DSP) subsystem is built around a core processor and peripherals that interface with: 1) The ARM926EJS via the microprocessor unit interface (MPUI); 2) Various standard memories via the external memory interface (EMIF); 3) Various system peripherals via the TI peripheral bus (TIPB) bridge.

OMAP5912 Multimedia Processor Clocks Reference Guide (literature number SPRU751) describes the clocking mechanisms of the OMAP5912 multimedia processor. In OMAP5912, various clocks are created from special components such as the digital phase locked loop (DPLL) and the analog phase-locked loop (APLL).

OMAP5912 Multimedia Processor Initialization Reference Guide (literature number SPRU752) describes the reset architecture, the configuration, the initialization, and the boot ROM of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Power Management Reference Guide (literature number SPRU753) describes power management in the OMAP5912 multimedia processor. The ultralow-power device (ULPD) generates and manages clocks and reset signals to OMAP3.2 and to some peripherals. It controls chip-level power-down modes and handles chip-level wake-up events. In deep sleep mode, this module is still active to monitor wake-up events.This book describes the ULPD module and outline architecture.

OMAP5912 Multimedia Processor Security Features Reference Guide (literature number SPRU754) describes the security features of teh OMAP5912 multimedia processor. The OMAP5912 security scheme relies on the OMAP3.2 secure mode. The distributed security on the OMAP3.2 platform is a Texas Instruments solution to address m-commerce and security issues within a mobile phone environment. The OMAP3.2 secure mode was developed to bring hardware robustness to the overall OMAP5912 security scheme.

OMAP5912 Multimedia Processor Direct Memory Access (DMA) Support Reference Guide (literature number SPRU755) describes the direct memory access support of the OMAP5912 multimedia processor. The OMAP5912 processor has three DMAs:

- The system DMA is embedded in OMAP3.2. It handles DMA transfers associated with MPU and shared peripherals.
- The DSP DMA is embedded in OMAP3.2. It handles DMA transfers associated with DSP peripherals.
- The generic distributed DMA (GDD) is an OMAP5912 resource attached to the SSI peripheral. It handles only DMA transfers associated with the SSI peripheral.

OMAP5912 Multimedia Processor Memory Interfaces Reference Guide (literature number SPRU756) describes the memory interfaces of the OMAP5912 multimedia processor.

- SDRAM (external memory interface fast, or EMIFF)
- Asynchronous and synchronous burst memory (external memory interface slow, or EMIFS)
- NAND flash (hardware controller or software controller)
- CompactFlash on EMIFS interface
- Internal static RAM

OMAP5912 Multimedia Processor Interrupts Reference Guide (literature number SPRU757) describes the interrupts of the OMAP5912 multimedia processor. Three level 2 interrupt controllers are used in OMAP5912:

- One MPU level 2 interrupt handler (also referred to as MPU interrupt level 2) is implemented outside of OMAP3.2 and can handle 128 interrupts.
- One DSP level 2 interrupt handler (also referred to as DSP interrupt level 2.1) is instantiated outside of OMAP3.2 and can handle 64 interrupts.
- One OMAP3.2 DSP level 2 interrupt handler (referenced as DSP interrupt level 2.0) can handle 16 interrupts.

OMAP5912 Multimedia Processor Peripheral Interconnects Reference Guide (literature number SPRU758) describes various periperal interconnects of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Timers Reference Guide (literature number SPRU759) describes various timers of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Serial Interfaces Reference Guide (literature number SPRU760) describes the serial interfaces of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Universal Serial Bus (USB) Reference Guide (literature number SPRU761) describes the universal serial bus (USB) host on the OMAP5912 multimedia processor. The OMAP5912 processor provides several varieties of USB functionality. Flexible multiplexing of signals from the OMAP5912 USB host controller, the OMAP5912 USB function controller, and other OMAP5912 peripherals allow a wide variety of system-level USB capabilities. Many of the OMAP5912 pins can be used for USB-related signals or for signals from other OMAP5912 peripherals. The OMAP5912 top-level pin multiplexing

controls each pin individually to select one of several possible internal pin signal interconnections. When these shared pins are programmed for use as USB signals, the OMAP5912 USB signal multiplexing selects how the signals associated with the three OMAP5912 USB host ports and the OMAP5912 USB function controller can be brought out to OMAP5912 pins.

OMAP5912 Multimedia Processor Multi-channel Buffered Serial Ports (McBSPs) Reference Guide (literature number SPRU762) describes the three multi-channel buffered serial ports (McBSPs) available on the OMAP5912 device. The OMAP5912 device provides multiple high-speed multichannel buffered serial ports (McBSPs) that allow direct interface to codecs and other devices in a system.

OMAP5912 Multimedia Processor Camera Interface Reference Guide (literature number SPRU763) describes two camera inerfaces implemented in the OMAP5912 multimedia processor: compact serial camera port and camera parallel interface.

OMAP5912 Multimedia Processor Display Interface Reference Guide (literature number SPRU764) describes the display interface of the OMAP5912 multimedia processor.

- LCD module
- LCD data conversion module
- LED pulse generator
- Display interface

OMAP5912 Multimedia Processor Multimedia Card (MMC/SD/SDIO) (literature number SPRU765) describes the multimedia card (MMC) interface of the OMAP5912 multimedia processor. The multimedia card/secure data/secure digital IO (MMC/SD/SDIO) host controller provides an interface between a local host, such as a microprocessor unit (MPU) or digital signal processor (DSP), and either an MMC or SD memory card, plus up to four serial flash cards. The host controller handles MMC/SD/SDIO or serial port interface (SPI) transactions with minimal local host intervention.

OMAP5912 Multimedia Processor Keyboard Interface Reference Guide (literature number SPRU766) describes the keyboard interface of the OMAP5912 multimedia processor. The MPUIO module enables direct I/O communication between the MPU (through the public TIPB) and external devices. Two types of I/O can be used: specific I/Os dedicated for 8 x 8 keyboard connection, and general-purpose I/Os.

OMAP5912 Multimedia Processor General-Purpose Interface Reference Guide (literature number SPRU767) describes the general-purpose in-

terface of the OMAP5912 multimedia processor. There are four GPIO modules in the OMAP5912. Each GPIO peripheral controls 16 dedicated pins configurable either as input or output for general purposes. Each pin has an independent control direction set by a programmable register. The two−edge control registers configure events (rising edge, falling edge, or both edges) on an input pin to trigger interrupts or wake−up requests (depending on the system mode). In addition, an interrupt mask register masks out specified pins. Finally, the GPIO peripherals provide the set and clear capabilities on the data output registers and the interrupt mask registers. After detection, all event sources are merged and a single synchronous interrupt (per module) is generated in active mode, whereas a unique wake−up line is issued in idle mode. Eight data output lines of the GPIO3 are ORed together to generate a global output line at the OMAP5912 boundary. This global output line can be used in conjunction with the SSI to provide a CMT−APE interface to the OMAP5912.

OMAP5912 Multimedia Processor VLYNQ Serial Communications Interface Reference Guide (literature number SPRU768) describes the VLYNQ of the OMAP5912 multimedia processor.

VLYNQ is a serial communications interface that enables the extension of an internal bus segment to one or more external physical devices. The external devices are mapped into local, physical address space and appear as if they are on the internal bus of the OMAP 5912. The external devices must also have a VLYNQ interface. The VLYNQ module serializes bus transactions in one device, transfers the serialized data between devices via a VLYNQ port, and de-serializes the transaction in the external device.

OMAP5912 includes one VLYNQ module connected on OCPT2 target port and OCPI initiator port. These connections are configured via a static switch, which selects either SSI or VLYNQ module. This switch, forbids the simultaneous use of GDD/SSI and VLYNQ. The switch is controlled by the VLYNQ_EN bit in the OMAP5912 configuration control register (CONF_5912_CTRL).

OMAP5912 Multimedia Processor Pinout Reference Guide (literature number SPRU769) provides the pinout of the OMAP5912 multimedia processor. After power-up reset, the user can change the configuration of the default interfaces. If another interface is available on top of the default, it is possible to enable a new interface for each ball by setting the corresponding 3-bit field of the associated FUNC_MUX_CTRL register. It is also possible to configure on-chip pullup/pulldown. This document

also describes the various power domains so that the user can apply the different interfaces seamlessly with external components.

OMAP5912 Multimedia Processor Window Tracer (WT) Reference Guide (literature number SPRU770) describes the window tracer module used to capture the memory transactions from four interfaces: EMIFF, EMIFS, OCP-T1, and OCP-T2. This module is located in the OMAP3.2 traffic controller (TC).

OMAP5912 Multimedia Processor Real-Time Clock Reference Guide (literature number SPRUxxx) describes the real-time clock of the OMAP5912 multimedia processor. The real-time clock (RTC) block is an embedded real-time clock module directly accessible from the TIPB bus interface.

Trademarks

Contents

Figures

Tables

Peripheral Interconnects

This document describes various peripheral interconnects of the OMAP5912 multimedia processor.

1 Shared Peripherals

Outside OMAP 3.2, the OMAP5912 includes several peripherals. They can be considered shared by the MPU and the DSP, or can be DSP or MPU private.

Privacy is mainly driven by the performance requested by the peripheral. OMAP provides four peripheral buses:

- 16-bit data bus DSP private
- 16-bit data bus DSP shared
- 32-bit data bus MPU private
- 32-bit data bus MPU shared

OMAP supports one bus protocol: the TIPB protocol.

The TIPB protocol enables unidirectional asynchronous transfer from or to peripherals to or from CPUs/DMAs.

Both shared buses ((MPU + system DMA) and (DSP + DSPDMA)) are multiplexed in the L4 interface either dynamically or semistatically to communicate with all the shared peripherals.

The L4 interface groups a set of wrappers (or protocol converter), a dynamic switch, and a static switch. They enable interface with all the peripherals, regardless of their interface protocol with the MPU and DSP peripheral buses.

The DSP peripherals are also accessible from the MPU via the DSP MPUI port (see Figure 1).

Figure 1. Peripherals Access

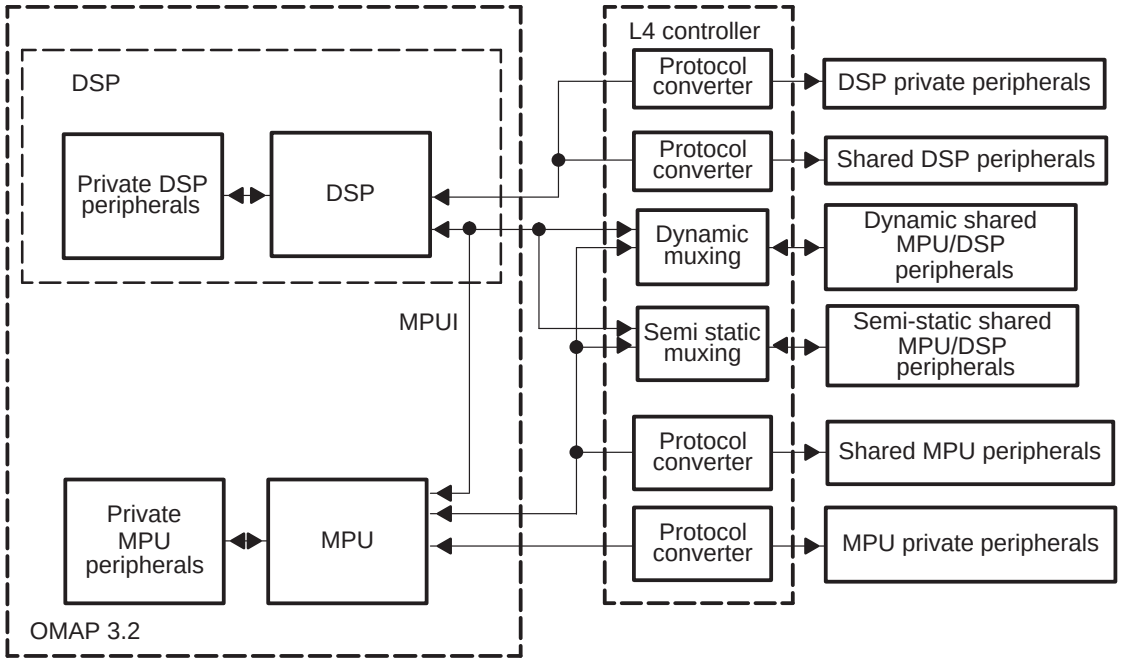


Table 1 lists the following for each module:

- On which bus the module resides
- Whether a DMA can initiate a transaction to the peripheral
- Whether the module can be accessed dynamically by each processor or the corresponding static switch must be configured

Table 1. MPU/DSP Peripheral Access

	MPU Domain			DSP Domain			
Module Name	**MPU Start**	**MPU End**	**MPU TIPB Bus Type**	**L4 Controler Switch**	**DSP Start**	**DSP End**	**DSP TIPB Bus Type**
DPLL1	FFFE CF00	FFFE CFFF	Private				
CLKM	FFFE CE00	FFFE CEFF	Private				
ULPD	FFFE 0800	FFFE 0FFF	Private				

Note: The SSI and the GDD modules are on the L3-OCP2 port and thus are seen as part of memory port interface.

Table 1. MPU/DSP Peripheral Access (Continued)

MPU Domain				DSP Domain			
Module Name	**MPU Start**	**MPU End**	**MPU TIPB Bus Type**	**L4 Controler Switch**	**DSP Start**	**DSP End**	**DSP TIPB Bus Type**
OMAP 5912 configuration	FFFE 1000	FFFE 17FF	Private				
32-kHz synchronization counter	FFFB C400	FFFB C7FF	Shared	Dynamic	E101 C400	E101 C7FF	Shared
Secure watchdog	FFFE A800	FFFE AFFF	Private				
32-kHz watchdog	FFFE B000	FFFE B7FF	Private				
RTC	FFFB 4800	FFFB 4FFF	Shared				
RNG	FFFE 5000	FFFE 57FF	Private				
DES/3DES	FFFE 4000	FFFE 47FF	Private				
SHA-1/MD5	FFFE 4800	FFFE 4FFF	Private				
OMAP MPU timer1	FFFE C500	FFFE C5FF	Private				
OMAP MPU timer2	FFFE C600	FFFE C6FF	Private				
OMAP MPU timer3	FFFE C700	FFFE C7FF	Private				
OMAP watchdog timer	FFFE C800	FFFE C8FF	Private				
OS timer	FFFB 9000	FFFB 93FF	Shared				
L3 OCP initiator	FFFE C300	FFFE C3FF	Private				

Note: The SSI and the GDD modules are on the L3-OCP2 port and thus are seen as part of memory port interface.

Table 1. MPU/DSP Peripheral Access (Continued)

MPU Domain				DSP Domain			
Module Name	**MPU Start**	**MPU End**	**MPU TIPB Bus Type**	**L4 Controler Switch**	**DSP Start**	**DSP End**	**DSP TIPB Bus Type**
L3 OCP T1 interface	FFFE C100	FFFE C1FF	Private				
OMAP5912 TIPB switch	FFFB C800	FFFB CBFF	Shared	Semi-static	E101 C800	E101 CBFF	Shared
TIPB bridge (internal)	FFFE CA00	FFFE CAFF	Private				
TIPB bridge2 (external)	FFFE D300	FFFE D3FF	Private				
MPUI interface	FFFE C900	FFFE C9FF	Private				
DSP MMU	FFFE D200	FFFE D2FF	Private				
Traffic controller	FFFE CC00	FFFE CCFF	Private				
Gigacell mailbox	FFFB F000	FFFB F7FF	Shared	Semi-static	E101 F000	E101 F7FF	Shared
Mailbox	FFFC F000	FFFC F7FF	Shared				
DSP TIPB					E100 0000	E100 07FF	Shared
MGS3 MPUI control register					E102 0000	E102 0003	Shared
System DMA	FFFE D800	FFFE DFFF	Private				
System DMA	FFFE E000	FFFE E7FF	Private				
MPU level 1 interrupt handler	FFFE CB00	FFFE CBFF	Private				

Note: The SSI and the GDD modules are on the L3-OCP2 port and thus are seen as part of memory port interface.

Table 1. MPU/DSP Peripheral Access (Continued)

MPU Domain				DSP Domain			
Module Name	**MPU Start**	**MPU End**	**MPU TIPB Bus Type**	**L4 Controler Switch**	**DSP Start**	**DSP End**	**DSP TIPB Bus Type**
MPU level 2 interrupt handler	FFFE 0000	FFFE 07FF	Private				
MPUIO	FFFB 5000	FFFB 57FF	Shared				
GPIO1	FFFB E400	FFFB E7FF	Shared	Dynamic	E101 E400	E101 E7FF	Shared
GPIO2	FFFB EC00	FFFB EFFF	Shared	Dynamic	E101 EC00	E101 EFFF	Shared
GPIO3	FFFB B400	FFFB B7FF	Shared	Dynamic	E101 B400	E101 B7FF	Shared
GPIO4	FFFB BC00	FFFB BFFF	Shared	Dynamic	E101 BC00	E101 BFFF	Shared
PWL	FFFB 5800	FFFB 5FFF	Shared				
PWT	FFFB 6000	FFFB 67FF	Shared				
GP timer 1	FFFB 1400	FFFB 17FF	Shared	Semi-static	E101 1400	E101 17FF	Shared
GP timer 2	FFFB 1C00	FFFB 1FFF	Shared	Semi-static	E101 1C00	E101 1FFF	Shared
GP timer 3	FFFB 2400	FFFB 27FF	Shared	Semi-static	E101 2400	E101 27FF	Shared
GP timer 4	FFFB 2C00	FFFB 2FFF	Shared	Semi-static	E101 2C00	E101 2FFF	Shared
GP timer 5	FFFB 3400	FFFB 37FF	Shared	Semi-static	E101 3400	E101 37FF	Shared
GP timer 6	FFFB 3C00	FFFB 3FFF	Shared	Semi-static	E101 3C00	E101 3FFF	Shared
GP timer 7	FFFB 7400	FFFB 77FF	Shared	Semi-static	E101 7400	E101 77FF	Shared

Note: The SSI and the GDD modules are on the L3-OCP2 port and thus are seen as part of memory port interface.

Table 1. MPU/DSP Peripheral Access (Continued)

MPU Domain					DSP Domain		
Module Name	**MPU Start**	**MPU End**	**MPU TIPB Bus Type**	**L4 Controler Switch**	**DSP Start**	**DSP End**	**DSP TIPB Bus Type**
GP timer 8	FFFB D400	FFFB D7FF	Shared	Semi-static	E101 D400	E101 D7FF	Shared
LCDCONV	FFFE 3000	FFFE 37FF	Private				
LCD controller	FFFE C000	FFFE C0FF	Private				
LPG1	FFFB D000	FFFB D3FF	Shared				
LPG2	FFFB D800	FFFB BFFF	Shared				
McBSP1					E101 1800	E101 1BFF	Shared
McBSP2	FFFB 1000	FFFB 13FF	Shared	Semi-static	E101 1000	E101 13FF	Shared
McBSP3					E101 7000	E101 73FF	Shared
MCSI1					E101 2800	E101 2BFF	Shared
MCSI2					E101 2000	E101 23FF	Shared
Memory Stick	FFFB 8000	FFFB 83FF	Shared				
MMC/SDIO 1	FFFB 7800	FFFB 7BFF	Shared				
MMC/SDIO 2	FFFB 7C00	FFFB 7FFF	Shared	Semi-static	E101 7C00	E101 7FFF	Shared
Compact flash controller	FFFE 2800	FFFE 2FFF	Private				
NAND flash controller	FFFB CC00	FFFB CFFF	Shared				

Note: The SSI and the GDD modules are on the L3-OCP2 port and thus are seen as part of memory port interface.

Table 1. MPU/DSP Peripheral Access (Continued)

MPU Domain				DSP Domain			
Module Name	**MPU Start**	**MPU End**	**MPU TIPB Bus Type**	**L4 Controler Switch**	**DSP Start**	**DSP End**	**DSP TIPB Bus Type**
SoSSI	FFFB AC00	FFFB AFFF	Shared				
SPI	FFFB 0C00	FFFB 0FFF	Shared	Semi-static	E101 0C00	E101 0FFF	Shared
UART1	FFFB 0000	FFFB 03FF	Shared	Semi-static	E101 0000	E101 07FF	Shared
UART2	FFFB 0800	FFFB 0BFF	Shared	Semi-static	E101 0800	E101 0BFF	Shared
UART3	FFFB 9800	FFFB 9BFF	Shared	Semi-static	E101 9800	E101 9BFF	Shared
FAC	FFFB A800	FFFB ABFF	Shared				
USB client	FFFB 4000	FFFB 43FF	Shared				
USB host	FFFB A000	FFFB A3FF	Shared				
USB OTG	FFFB 0400	FFFB 07FF	Shared				
I^2C multi-master	FFFB 3800	FFFB 3BFF	Shared	Semi-static	E101 3800	E101 3BFF	Shared
µWire	FFFB 3000	FFFB 33FF	Shared				
HDQ 1-Wire	FFFB C000	FFFB C3FF	Shared				
STI (reserved)	FFFE A000	FFFE A7FF	Private	Dynamic	E101 A400	E101 A7FF	Shared
OMAP5912 JTAG (test)	FFFE 5800	FFFE 5FFF	Private				
BCM (test)	FFFE 3800	FFFE 3FFF	Private				
Production ID (test)	FFFE 2000	FFFE 27FF	Private				

Note: The SSI and the GDD modules are on the L3-OCP2 port and thus are seen as part of memory port interface.

Table 1. MPU/DSP Peripheral Access (Continued)

MPU Domain					DSP Domain		
Module Name	**MPU Start**	**MPU End**	**MPU TIPB Bus Type**	**L4 Controler Switch**	**DSP Start**	**DSP End**	**DSP TIPB Bus Type**
Die ID (test)	FFFE 1800	FFFE 1FFF	Private				
Test block (PSA test)	FFFE D400	FFFE D4FF	Private				
DSP trace					E100 4000	E100 47FF	Private

Note: The SSI and the GDD modules are on the L3-OCP2 port and thus are seen as part of memory port interface.

2 Layer 4 Interconnect

The layer 4 interface manages accesses to OMAP5912 peripherals through the MPU shared TIPB bridge, DSP shared TIPB bridge, MPU private TIPB bridge, and DSP private TIPB bridge. DSP TIPB bridges are inside the C55x DSP. It is composed of a set of dynamic and static switches and wrappers dedicated to the access protocol of each peripheral. The MPU TIPB bridge is already a merge of MPU and system DMA signals for peripheral accesses. The DSP TIPB bridge is already a merge of DSP and system DMA signals for peripheral accesses (see Figure 2).

Two different switches are implemented:

- ❏ Static TIPB/OCP: OCP/TIPB switch (configuration register included)
- ❏ Dynamic TIPB/OCP: OCP/TIPB switch with resynchronization on MPU and DSP accesses

Two different wrappers are provided:

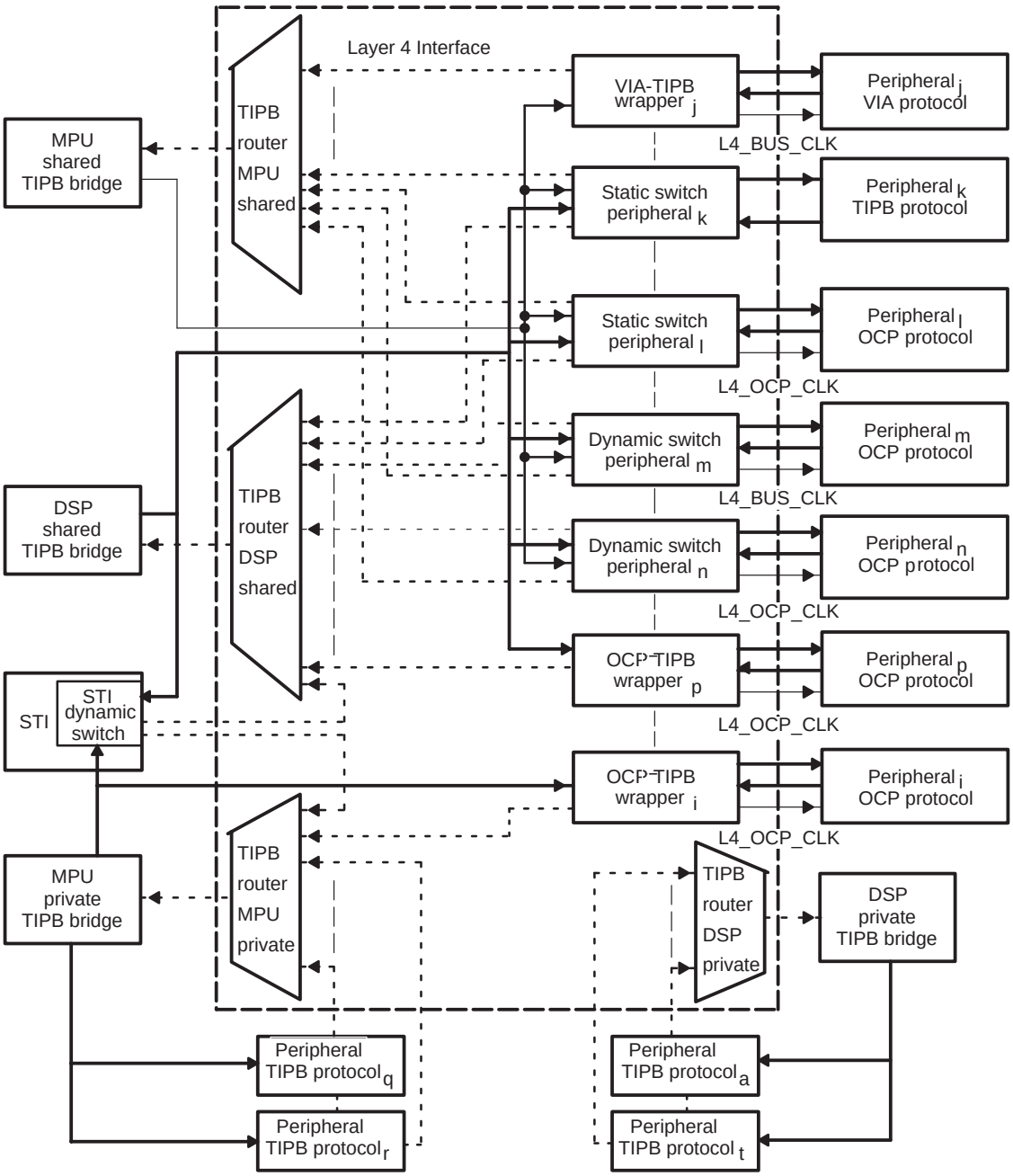
- ❏ TIPB/VIA: VIA/TIPB wrapper with resynchronization
- ❏ MPU (or DSP) TIPB/OCP: OCP/TIPB wrapper

The common protocol on the MPU/DSP side is the TIPB bus protocol. For each TIPB bridge, a dedicated router is implemented to allow the multiplexing of all the return paths from the peripherals.

- ❏ MPU shared TIPB router to MPU shared TIPB bridge
- ❏ DSP shared TIPB router to DSP shared TIPB bridge

- ❏ MPU private TIPB router to MPU private TIPB bridge
- ❏ DSP private TIPB router to DSP private TIPB bridge

Figure 2. Layer 4 Interface



2.1 TIPB Router Connections

Table 2. Private Peripherals

OMAP 5912 Peripheral	Interface	TIPB Router
DSP level 2 interrupt handler	Wrapper OCP	Private DSP
ULPD (clock and reset)	TIPB	Private MPU
OMAP5912 configuration	TIPB	Private MPU
Secure watchdog	Wrapper OCP	Private MPU
32-kHz watchdog	Wrapper OCP	Private MPU
MPU level 2 interrupt handler	Wrapper OCP	Private MPU
SHA_1 accelerator	Wrapper OCP	Private MPU
RNG random generator	Wrapper OCP	Private MPU
DES/3DES	Wrapper OCP	Private MPU
System DMA handler	Wrapper OCP	Private MPU

Table 3. MPU Peripherals Connected to MPU Shared TIPB Bridge

OMAP 5912 Peripheral	Interface	TIPB Router
HDQ/1-Wire	TIPB	Shared MPU
µWIRE	TIPB	Shared MPU
MMCSD/IO1	Wrapper OCP	Shared MPU
MPUIO	TIPB	Shared MPU
2 x LPG	TIPB	Shared MPU
SoSSI	Wrapper VIA	Shared MPU
RTC	TIPB	Shared MPU
Memory Stick	TIPB	Shared MPU
PWL	TIPB	Shared MPU
PWT	TIPB	Shared MPU
FAC	TIPB	Shared MPU
OS timer	TIPB	Shared MPU
USBOTG	Wrapper OCP	Shared MPU

Table 4. DSP Peripherals Connected to DSP Shared TIPB Bridge

OMAP 5912 Peripheral	Interface	TIPB Router
2 x MCSI	TIPB	Shared DSP
2 x McBSP	Wrapper OCP	Shared DSP

Table 5. DSP and MPU Shared Peripherals

OMAP 5912 Peripheral	Interface	TIPB Router	
		MPU	DSP
MMCSD/IO2	Static OCP	Shared	Shared
STI (reserved)	Dynamic OCP	Private	Shared
32-kHz synchronization timer	Dynamic OCP	Shared	Shared
4 x GPIO	Dynamic OCP	Shared	Shared
GPIO	Dynamic OCP	Shared	Shared
3 x UART	Static OCP	Shared	Shared
McBSP	Static OCP	Shared	Shared
I^2C	Static OCP	Shared	Shared
SPI	Static OCP	Shared	Shared
8 x GP timers	Static OCP	Shared	Shared
NAND flash control	Static OCP	Shared	Shared
Static switch configuration	TIPB	Shared	Shared

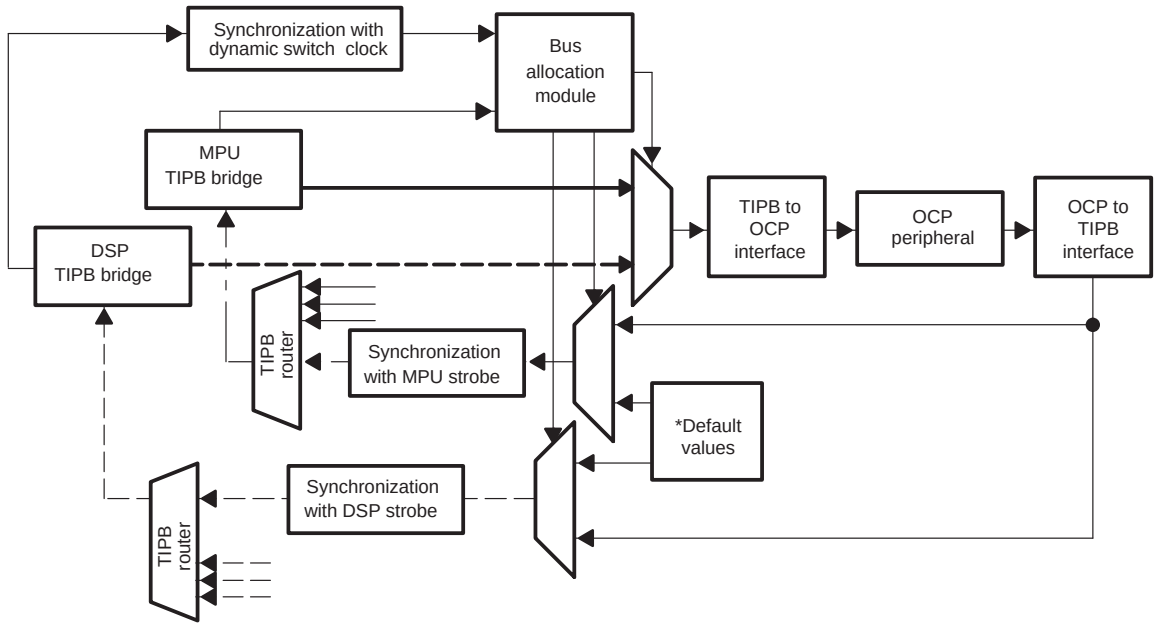
2.2 Protocols

The layer 4 interface is connected to several peripherals, which use different protocols:

- ❏ TI peripheral bus or TIPB bus
- ❏ Versatile interconnection architecture designated also as VIA bus or VIA protocol
- ❏ Open core protocol designated as OCP

2.3 Dynamic Switch

Figure 3. Dynamic Switch for OCP Peripheral



Note: Default value: To simplify the TIPB router implementation and reduce the toggling on buses, a default value is returned to the host, which does not access the peripheral.

2.4 Functional Description

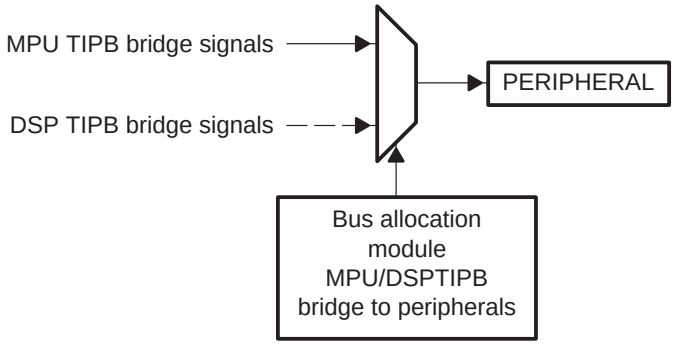
2.4.1 Bus Allocation

MPU/DSP TIPB Bridge to Peripherals

The peripheral TIPB buses are shared between the MPU TIPB bridge and the DSP TIPB bridge. Both bridges can access the same peripheral. Therefore, a bus allocation module is required to decide which one can access to the peripheral bus in case of conflict. This arbitration occurs after a synchronization of the DSP enable signal with the dynamic switch clock (ARMPER_CLK).

If no more than one of the two masters requests the bus, the bus is allocated to the requesting master. Otherwise, the MPU TIPB bridge takes control of the bus. DSP TIPB bridge accesses the peripheral after the data has been returned to the MPU TIPB bridge. During the transfer, the access size can be either 16-bit for the DSP or 8-, 16-, or 32-bit for the MPU.

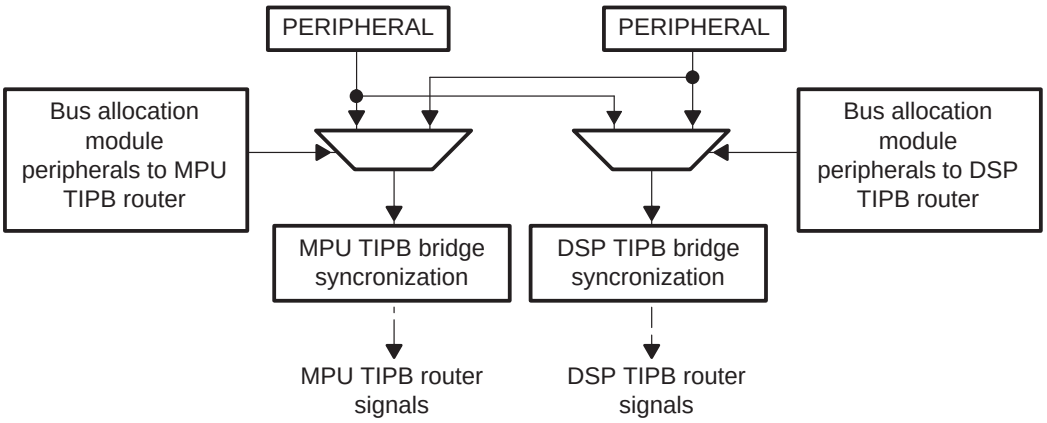
Figure 4. Transfer from Host to Peripheral



Peripherals to MPU/DSP TIPB Bridge

All peripherals can send data back on a dedicated bus (one bus per peripheral—8-, 16-, or 32-bit according to each peripheral requirement), either to the MPU or to the DSP.

Figure 5. Transfer from Peripheral to Host



Note: Default value: To simplify the TIPB router implementation and reduce the toggling on buses, a default value is returned to the host that does not access the peripheral.

2.4.2 16-bit Accesses on 32-Bit Atomic Registers

There is no protection inside the dynamic switch to ensure the atomic reading or writing of one 32-bit register with two 16-bit accesses. The nonoverlapping requests from the two hosts must be controlled in the software.

2.4.3 Conflicting Transaction

When one host requires an access to the peripheral, an enable host signal is set (synchronously with the host strobe). With the DSP TIPB bridge strobe, a DSP TIPB bridge enable is generated, and with the MPU TIPB bridge strobe an MPU TIPB bridge enable is generated. DSP TIPB bridge enable and MPU TIPB bridge enable are then resynchronized with the dynamic switch clock to perform arbitration. This arbitration takes place during one dynamic switch clock cycle. In case of conflict, the MPU TIPB bridge has the priority and the DSP TIPB bridge waits until the ready has been sent to the MPU TIPB bridge. Then the DSP TIPB bridge performs its access.

2.4.4 Abort Transaction

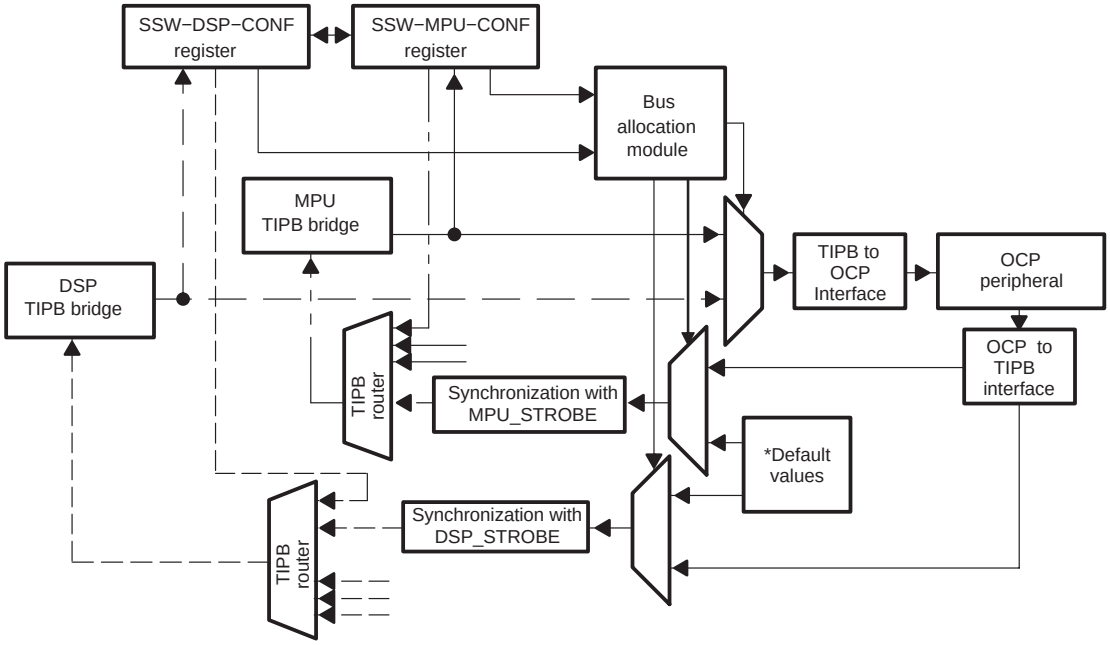
After a predetermined number of strobe cycles (DSP_nStrobe or MPU_nStrobe), if a peripheral does not activate its ready signal, the TIPB bridge generates an abort. The dynamic switch uses this abort to kill the current access. The maximum number of strobe cycles is programmable within the DSP TIPB bridge for DSP_nStrobe, and within the MPU TIPB bridge for MPU_nStrobe.

2.4.5 Reset Methodology

An MPU initiator reset is always used in the dynamic switch to reset an eventual MPU ongoing transaction. A DSP initiator reset is also used into the dynamic switch to reset an eventual DSP ongoing transaction.

2.5 Static Switch

Figure 6. Static Switch for OCP Peripheral



Note: Default value: To simplify the TIPB router implementation and reduce the toggling on buses, a default value is returned to the host which does not access the peripheral.

The static switch module allows the switching of a given peripheral between two TIPB host buses. This module enables static sharing between the DSP and the MPU processors. The switch between the two TIPB buses is implemented through a basic protocol. The software must control carefully the accessibility of each shared peripheral by the desired host. The programmer must ensure that the ongoing DMA transfers from/to the peripheral are completed before updating the peripheral ownership. There is no protection from the static switch hardware.

If the static switch is not well programmed before an access to a shared peripheral register, the corresponding ready signal is not sent back to the host. The result is a time-out error generated by the host TIPB bridge, depending on the TIPB bridge programming.

Two registers control the accessibility from the host to each statically sharedperipheral:

- ❏ SSW_MPU_CONF in the MPU peripheral address space
- ❏ SSW_DSP_CONF in the DSP peripheral address space

To perform an MPU access, the MPU software must first set (write 1) the MPU_SWITCH bit in the SSW_MPU_CONF register. Then all the MPU TIPB bus signals are connected either to the TIPB bus peripheral signals (for a TIPB peripheral), or to the dedicated wrapper TIPB bus interface (for OCP peripheral).

To perform a DSP access, the MPU software must first authorize this access by resetting the MPU_SWITCH bit in the SSW_MPU_CONF register (write 0); then the DSP software must set the DSP_SWITCH in the SSW_DSP_CONF register.

If the MPU wants to access the SSW_MPU_CONF register when the DSP_SWITCH bit is set, the access is completed but a write access has no effect on the register.

Symmetrically, if the DSP wants to access the SSW_DSP_CONF register when the MPU_SWITCH bit is set, the access is completed but a write access has no effect on the register.

The following programming sequence enables DSP access to the peripheral:

1) Check that the MPU_SWITCH bit is reset (no MPU access is ongoing on the same peripheral).
2) Set the DSP_SWITCH bit in the SSW_DSP_CONF register (DSP software: write 1)
3) Check that the DSP_SWITCH is set (in case of priority conflict with the MPU software).
4) Perform the shared peripheral access through the DSP TIPB bus.
5) After the completion of this access, reset the DSP_SWITCH bit in the SSW_DSP_CONF register (write 0).

The following programming sequence enables MPU access to the peripheral:

6) Check that DSP_SWITCH bit is reset (no DSP access is on going on the same peripheral).
7) Set the MPU_SWITCH bit in the SSW_MPU_CONF register (MPU software: write 1).
8) Check that the MPU_SWITCH is set (in case of priority conflict with the DSP software).
9) Perform the shared peripheral access through the MPU TIPB bus.
10) After the completion of this access, reset the MPU_SWITCH bit in the SSW_MPU_CONF register (write 0)

Depending on the host interface protocol for the peripheral, a dedicated wrapper is connected between the host and the peripheral bus. Two different peripheral protocols are supported:

- TIPB
- OCP

2.5.1 Simultaneous Access

Setting both the DSP_SWITCH bit and MPU_SWITCH bits is impossible, by design. An access control and arbitration phase in the ARMPER_CLK clock domain ensures that this programming conflict never occurs.

2.5.2 TIPB to OCP Interface

A full resynchronization process between the MPU TIPB interface, the DSP TIPB interface, and the peripheral interface OCP clock is implemented. The TIPB bus is stalled until the decoding of the peripheral acknowledge by the wrapper. The decoded acknowledge is then synchronized with the TIPB strobe to generate the ready (end of TIPB transaction).

2.6 Abort Transaction

After a predetermined number of strobe cycles (DSP_nStrobe or MPU_nStrobe), if a peripheral did not complete the transaction, the TIPB bridge generates an abort. The static switch uses this abort to kill the current access. The maximum number of strobe cycles is programmable, within the DSP TIPB bridge for DSP_nStrobe, and within the MPU TIPB bridge for MPU_nStrobe configuration registers.

2.7 Peripheral Static Switch Configuration Registers

The registers listed in Table 6 are accessible through the DSP shared TIPB bridge (16-bit access) or through the MPU shared TIPB bridge (16- or 32-bit accesses).

Table 6. Static Switch Configuration Registers Offset Addresses

Register Name	# Bits	Offset	Bus
UART1_SSW_MPU_CONF	16/32	0x000	MPU
UART1_SSW_DSP_CONF	16		DSP
UART2_SSW_MPU_CONF	16/32	0x020	MPU

Table 6. Static Switch Configuration Registers Offset Addresses (Continued)

Register Name	# Bits	Offset	Bus
UART2_SSW_DSP_CONF	16		DSP
UART3_SSW_MPU_CONF	16/32	0x040	MPU
UART3_SSW_DSP_CONF	16		DSP
McBSP_SSW_MPU_CONF	16/32	0x090	MPU
McBSP_SSW_DSP_CONF	16		DSP
I2C_SSW_MPU_CONF	16/32	0x0A0	MPU
I2C_SSW_DSP_CONF	16		DSP
SPI_SSW_MPU_CONF	16/32	0x0B0	MPU
SPI_SSW_DSP_CONF	16		DSP
TIMER1_SSW_MPU_CONF	16/32	0x0C0	MPU
TIMER1_SSW_DSP_CONF	16		DSP
TIMER2_SSW_MPU_CONF	16/32	0x0D0	MPU
TIMER2_SSW_DSP_CONF	16		DSP
TIMER3_SSW_MPU_CONF	16/32	0x0E0	MPU
TIMER3_SSW_DSP_CONF	16		DSP
TIMER4_SSW_MPU_CONF	16/32	0x0F0	MPU
TIMER4_SSW_DSP_CONF	16		DSP
TIMER5_SSW_MPU_CONF	16/32	0x100	MPU
TIMER5_SSW_DSP_CONF	16		DSP
TIMER6_SSW_MPU_CONF	16/32	0x110	MPU
TIMER6_SSW_DSP_CONF	16		DSP
TIMER7_SSW_MPU_CONF	16/32	0x130	MPU
TIMER7_SSW_DSP_CONF	16		DSP
TIMER8_SSW_MPU_CONF	16/32	0x140	MPU
TIMER8_SSW_DSP_CONF	16		DSP
NFCtrl_SSW_MPU_CONF	16/32	0x150	MPU

Table 6. Static Switch Configuration Registers Offset Addresses (Continued)

Register Name	# Bits	Offset	Bus
NFCtrl_SSW_DSP_CONF	16		DSP
MMCSD2_SSW_MPU_CONF	16/32	0x160	MPU
MMCSD2_SSW_DSP_CONF	16		DSP

For each static-shared peripheral, two registers are defined (Table 7 and Table 8): one is dedicated to MPU accesses, and the other to DSP accesses.

Table 7. Peripheral MPU Static Switch Configuration Register (xxx_SSW_MPU_CONF)

Base Address = FFFB C800, Offset Address = see Table 6				
Bit	**Name**	**Description**	**Access**	**Reset**
31:2	–	Reserved	R	0x00000000
1	DSP_SWITCH	0: No DSP TIPB bridge access required 1: DSP TIPB bridge access required	R	0
0	MPU_SWITCH	0: No MPU TIPB bridge access required 1: MPU TIPB bridge access required	RW	1

This register is only accessible through the MPU shared TIPB bridge.

It is used to perform MPU access on the peripheral.

Table 8. Peripheral DSP Static Switch Configuration Register (xxx_SSW_DSP_CONF)

Base Address = FFFB C800, Offset Address = see Table 6				
Bit	**Name**	**Description**	**Access**	**Reset**
15:2	–	Reserved	R	0x00000000
1	DSP_SWITCH	0: No DSP TIPB bridge access required 1: DSP TIPB bridge access required	RW	0
0	MPU_SWITCH	0: No MPU TIPB bridge access required 1: MPU TIPB bridge access required	R	1

This register is accessible only through the DSP shared TIPB bridge.

2.7.1 Reset Methodology

An MPU or a DSP initiator reset is always used internally by the static switch.

A reset on the static switch configuration register resets the DSP_SWITCH bit and sets the MPU_SWITCH bit, which turns the static switch in the MPU position.

2.8 TIPB1-VIA/VIA-TIPB Wrapper

The synchronous VIA protocol requires a free-running clock and synchronous signals relative to this clock.

In addition to the TIPB bus signals, DMA requests need to be adapted.

Two successive DMA requests must be separated by at least two system DMA clock cycles.

2.8.1 Reset Methodology

The internal state machine of the wrapper is reset by the MPU peripheral reset.

For the SoSSI peripheral, two resets must be provided to the peripheral:

- L4_VIA_Reset: synchronous with the CK_DPLL1OUT
- L4_DMA_Reset: synchronous with the DMA_LCDFREE_CLK

In both cases, the reset source is the MPU peripheral reset.

For the CCP peripheral, one reset is generated to the peripheral; L4_VIA_RESET is synchronous with the ARMPER_CLK.

2.9 TIPB-OCP/OCP-TIPB Wrapper

The synchronous OCP protocol requires a free-running clock and synchronous signals realtive to this clock.

The free-running clock is ARMPER_CLK.

2.9.1 Reset Methodology

The NO_RESET signal comes from the wrapper to the connected peripheral.

2.9.2 DSP TIPB to OCP

This wrapper is sensitive to either DSP watchdog reset or DSP peripheral reset in order to clear the interface state machine.

2.9.3 MPU TIPB to OCP

This wrapper is sensitive to MPU watchdog reset or MPU peripheral reset in order to clear the internal state machine.

2.10 TIPB-OCP/OCP-TIPB Wrapper for USBOTG

The synchronous OCP protocol requires a free-running clock and synchronous signals relative to this clock.

The free-running clock is ARMPER_CLK.

This wrapper is used for transmissions with USB client, USB host, and OTG.

Depending on the access, an increment is applied to the address:

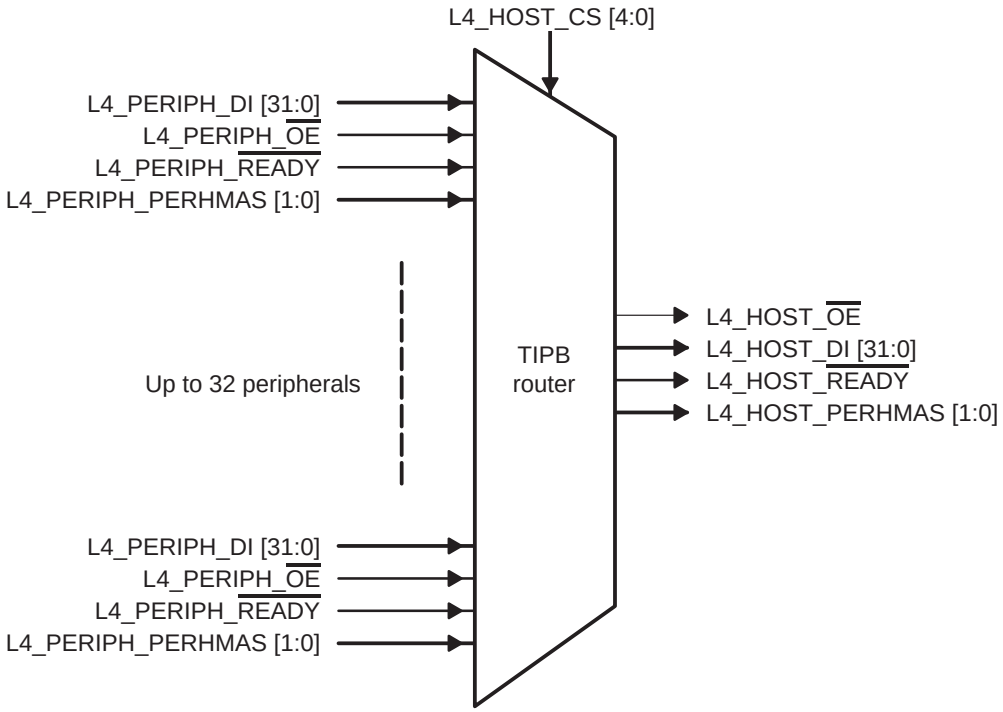
- If USB host is accessed, the address remains unchanged.
- If USB client is accessed, the address is incremented by 0x200h.
- If OTG is accessed, the address is incremented by 0x300h.

2.10.1 Reset Methodology

This wrapper is sensitive to MPU peripheral reset in order to reset the internal state machine.

2.11 TIPB Router

Figure 7. TIPB Router Block Diagram



The TIPB router is a simple multiplexer that returns information coming from peripherals to the host (DSP or MPU). The information concerns a current READ or WRITE access on a peripheral. This TIPB router supports up to 32 peripheral connections.

2.12 Peripheral Instantiation

Table 9. Private Peripherals Instantiation

OMAP 5912 Peripheral	Interface	Instantiation					TIPB Router
				Access Size			
		Address Bus Alignment†	Data	8b	16b	32b	
3 DSP level 2 interrupt handler	TIPB	Byte	16	–	R	–	Private DSP
ULPD (clock and reset)	TIPB	Byte	16	–	R	–	Private MPU
OMAP5912 configuration	TIPB	Byte	32	–	–	R	Private MPU
Secure watchdog	Wrapper OCP	Byte	32	–	R	R	Private MPU
32-kHz watchdog	Wrapper OCP	Byte	32	–	R	R	Private MPU
2 MPU level 2 interrupt handlers	TIPB	Byte	32	–	–	R	Private MPU
SHA_1 accelerator	Wrapper OCP	Byte	32	–	–	R	Private MPU
RNG random generator	Wrapper OCP	Byte	32	–	–	R	Private MPU
DES/3DES	Wrapper OCP	Byte	32	–	–	R	Private MPU
System DMA handler	Wrapper OCP	Byte	32	–	–	R	Private MPU

† Address bus alignment:
Byte: The least significant bit of the peripheral address bus corresponds to a byte address in the peripheral.
16-bit: The least significant bit of the peripheral address bus corresponds to a 16-bit address in the peripheral.
32-bit: The least significant bit of the peripheral address bus corresponds to a 32-bit address in the peripheral.

Table 10. MPU Peripherals Connected to the MPU Shared TIPB Bridge Instantiation

OMAP 5912 Peripheral	Interface	Instantiation					TIPB Router
		Address Bus Alignment†	Data	Access Size 8b	16b	32b	
HDQ/1-Wire	TIPB	Byte	8	R	–	–	Shared MPU
µWIRE	TIPB	Byte	16	–	R	–	Shared MPU
MMCSD/IO1	Wrapper OCP	16b	16	–	R	–	Shared MPU
MPUIO	TIPB	16b	16	–	R	–	Shared MPU
2 x LPG	TIPB	Byte	8	R	–	–	Shared MPU
SoSSI	Wrapper VIA	32b	32	–	–	R	Shared MPU
RTC	TIPB	Byte	8	R	–	–	Shared MPU
Memory Stick	TIPB	32b	32	–	–	R	Shared MPU
PWL	TIPB	Byte	8	R	–	–	Shared MPU
PWT	TIPB	Byte	8	R	–	–	Shared MPU
FAC	TIPB	Byte	16	–	R	–	Shared MPU
OS timer	TIPB	Byte	32	–	R	R	Shared MPU
USBOTG	Wrapper OCP	Byte	16	–	R	–	Shared MPU

† Address bus alignment:
Byte: The least significant bit of the peripheral address bus corresponds to a byte address in the peripheral.
16-bit: The least significant bit of the peripheral address bus corresponds to a 16-bit address in the peripheral.
32-bit: The least significant bit of the peripheral address bus corresponds to a 32-bit address in the peripheral.

Table 11. DSP Peripherals Connected to the DSP Shared TIPB Bridge Instantiation

OMAP 5912 Peripheral	Interface	Instantiation					TIPB Router
		Address Bus alignment	Data	Access Size 8b	16b	32b	
2 x MCSI	TIPB	Byte	16	–	R	–	Shared DSP
2 x McBSP	Wrapper OCP	Byte	16	–	R	–	Shared DSP

Table 12. DSP and MPU Shared Peripherals Instantiation

OMAP 5912 Peripheral	Interface	Instantiation					TIPB Router	
		Address Bus Alignment	Data	Access Size 8b	16b	32b	MPU	DSP
MMCSD/IO2	Static OCP	Byte	16	–	R	–	Shared	Shared
STI (reserved)	Dynamic OCP	Byte	32	–	R	R	Private	Shared
32-kHz synchronization timer	Dynamic OCP	Byte	32	–	R	R	Shared	Shared
4 x GPIO	Dynamic OCP	Byte	16	–	R	R	Shared	Shared
GPIO pin control	Dynamic OCP	Byte	16	–	R	R	Shared	Shared
3 x UART	Static TIPB	Byte	8	R	–	–	Shared	Shared
McBSP	Static OCP	Byte	16	–	R	–	Shared	Shared
I^2C	Static OCP	Byte	16	R	R	–	Shared	Shared
SPI	Static OCP	Byte	32	–	R	R	Shared	Shared
8 x GP timers	Static OCP	Byte	32	–	R	R	Shared	Shared
NAND flash control	Static OCP	Byte	32	R	R	R	Shared	Shared
Static switch configuration	TIPB	Byte	32	R	R	R	Shared	Shared

3 OCP Interconnect

The OCP, SSI interconnects, and static switches manage data transfers between OMAP3.2 and the SSI or VLYNQ and USB peripherals.

The SSI is a full-duplex interface, using a synchronous serial interconnect protocol (SSI). This protocol consists of a transmitter called (SST) in SSI, which is in charge of transmitting information, and a receiver called (SSR), which is in charge of receiving information

SSI DMA transfers are handled by the generic distributed DMA (GDD).

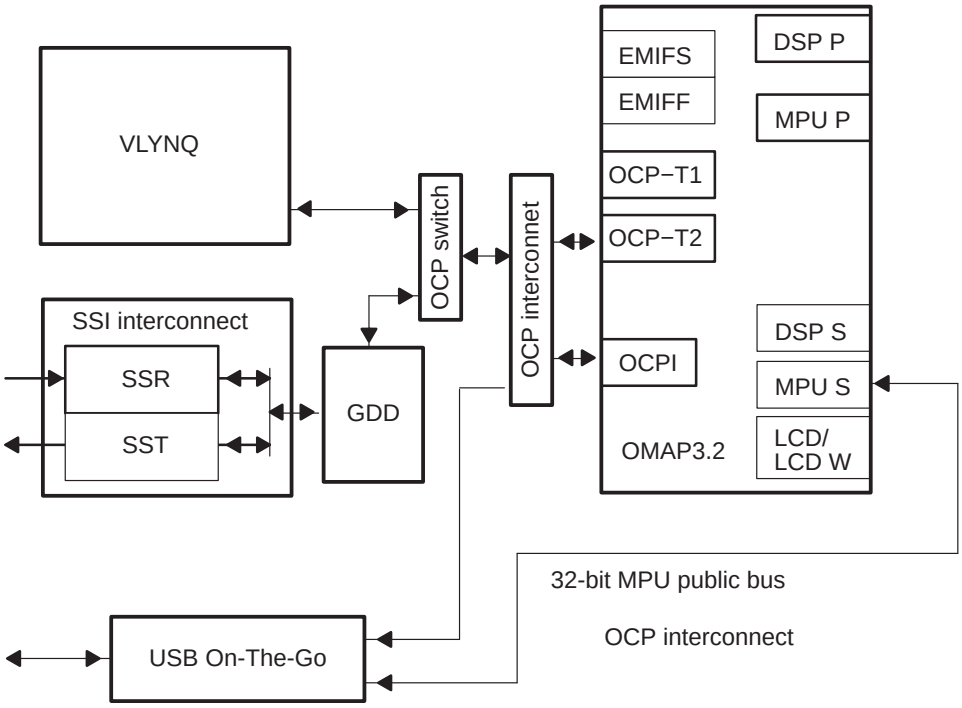
Because the SSI uses a proprietary bus interface (versatile interconnection architecture (VIA)), bridges are implemented in the SSI interconnect to perform data transfers between the SSI VIA bus and the OMAP3.2 OCP bus.

VLYNQ is also a full duplex interface, using a VBUS 1.0 protocol. A wrapper around the VLYNQ core performs the protocol conversion VBUS to OCP and OCP to VBUS.

In addition, data transfer between the USB host and OMAP3.2 are done through the OCP interconnect and the OCPI port.

The GDD has priority over the USB.

Figure 8. OCP and SSI Interconnects



3.1 OCP Introduction

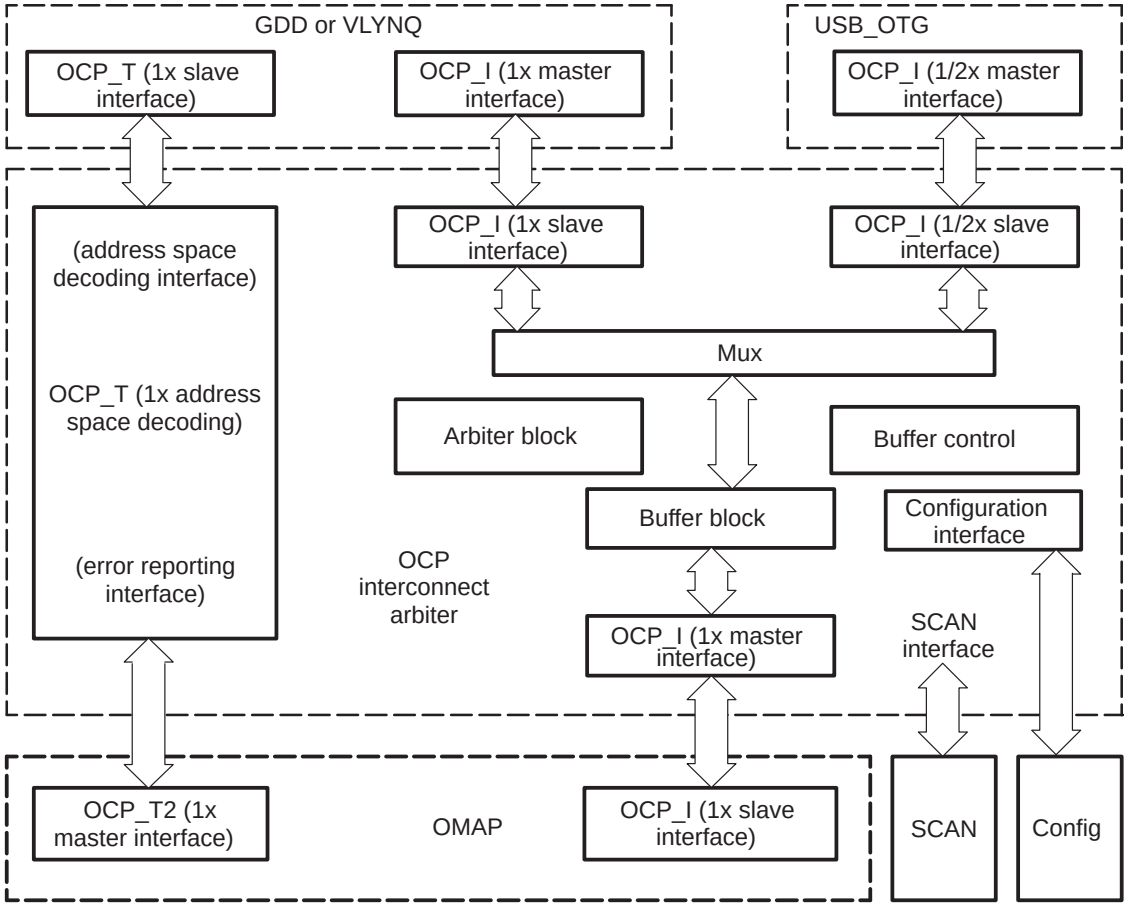
The OCP arbiter must interconnect two OCP master devices (peripherals), such as GDD or VLYNQ and USB_OTG peripheral, to a single OMAP OCP_I slave port. GDD or VLYNQ OCP_I port has priority over USB_OTG OCP_I port.

All interfaces are 32-bit data width and support burst and nonburst operations.

Because the OMAP OCP_I port does not support split bursts, the buffer block translates split burst (master can insert wait states between valid commands during burst) into a simple burst action (the master cannot insert wait states between valid commands during burst).

The OCP arbiter interconnect also provides address space decoding and standard OCP error signaling if the address is out of GDD/SSI or VLYNQ address space.

Figure 9. OCP Interconnect



3.2 Module Features

The OCP interconnect arbiter provides the following features:

- One OCP master port 32-bit width, 1x balanced clock, 4x32 (FOUR-TWO-TWO-LAST incremented burst only) nonsplit burst and nonburst accesses (OMAP OCP_I interconnect)
- One OCP slave port 32-bit width, 1x balanced clock, 4x32 split burst and nonburst accesses (GDD OCP_I interconnect)
- One OCP slave port 32-bit width, 1/2x balanced clock, 4x32 split burst and nonburst accesses (USB_OTG OCP_I interconnect)
- Translate split burst to nonsplit burst accesses for OMAP OCP_I slave interface (buffer for burst transaction)
- Offer high priority to OCP_I 1x master (GDD OCP_I has the priority)
- Address space decoding for GDD/SSI and standard error report to the OCP_T2 OMAP port when address is out of GDD/SSI address space.

3.3 1x OCP Slave Port

The OCP interconnect arbiter 1x OCP slave port has the following features:

- OCP interface (slave)
- Synchronous
- Balanced 100-MHz clock
- 32-bit data width only
- Support 4x32 burst and nonburst accesses

This port is directly connected to the GDD memory port.

3.4 1/2x OCP Slave Port

The OCP interconnect arbiter 1/2x OCP slave port has the following features:

- OCP interface (slave)
- Synchronous
- Balanced 100-MHz clock using a 50-MHz phase indication
- 32-bit data width only
- Support 4x32 burst and nonburst accesses

This port is directly connected to the USB_OTG OCP master interface.

3.5 1x OCP Master Port

The OCP interconnect arbiter 1x OCP master port has the following features:

- OCP interface (master)
- Synchronous
- Balanced 100-MHz clock
- 32-bit data width only
- Support 4x32 burst (4−2−2−LAST only) and non-burst accesses
- Non-split burst only

This port is directly connected to the OMAP OCP_I slave interface.

3.6 Address Space

The address space for the GDD/SSI and VLYNQ is shown in Table 13 (and is fixed; no configuration registers are needed).

Table 13. GDD/SSI Address Space

Block Name	Start Address	End Address	Size
SSI mapping	0x3000 0000	0x3000 0FFF	4K bytes
GDD mapping	0x3000 1000	0x3000 1FFF	4K bytes
VLYNQ conf mapping	0x3000 2000	0x3000 21FF	512 bytes
VLYNQ window mapping	0x3100 0000	0x34FF FFFF	64 Mbytes

3.7 Arbiter Block

The arbiter block controls the priority between GDD or VLYNQ and USB conflicting requests to the OMAP OCP-I port. The arbitration scheme is simple. If both 1x OCP and 1/2x OCP ports request at the same time, priority is given to the 1x port. In other words, the GDD or VLYNQ always has the highest priority.

3.8 Buffer Block

The buffer block acts as a buffer between the input and output registers. This block provides the buffering required to have all outputs come from the register (no combinatorial output). Also, because the OMAP OCP_I interface does not have the capability to handle split bursts, this buffer is used to provide burst transaction without wait states (nonsplit burst) when bursts are initiated by the USB OCP_I interface. The buffer block contains four registers to handle the 4 x 32 burst.

The following configurations are possible for the buffer block in case of a burst request transaction**:**

- GDD or VLYNQ side, send as soon as possible
- GDD or VLYNQ side, wait for buffer mode
- USB side, send as soon as possible mode
- USB side, wait for buffer mode

For all USB, GDD, and VLYNQ simple read or write transactions, the request is sent directly to the OMAP OCP_I interface.

4 SSI Interconnect

4.1 Introduction

The SSI interconnect manages accesses to OMAP5912 peripherals through GDD, acting as a bridge between external serial peripherals (using VIA buses) and the internal parallel bus (OCP). On one side it is connected to SSI, using two VIA buses, and on the other to GDD, using an OCP bus (see Figure 9). The SSI module is composed of two blocks, SSR and SST, each using a different VIA bus.

SSI interconnect is composed of five blocks:

- Decode

  The decode module dispatches incoming transactions either to OCP to the VIA asynchronous bridge or to the synchronous bridge of OCP to VIA.

- OCP to the VIA synchronous bridge

  OCP to the VIA synchronous bridge connects SST to GDD via the OCP bus. Because the SST module is synchronous with the OCP bus, resynchronization is not required and a faster synchronous bridge can be used.

- OCP to the VIA asynchronous bridge

  OCP to the VIA asynchronous bridge connects SSR to the OCP bus. Because the SSR module works on a clock that is asynchronous and faster than the OCP bus, resynchronization is required.

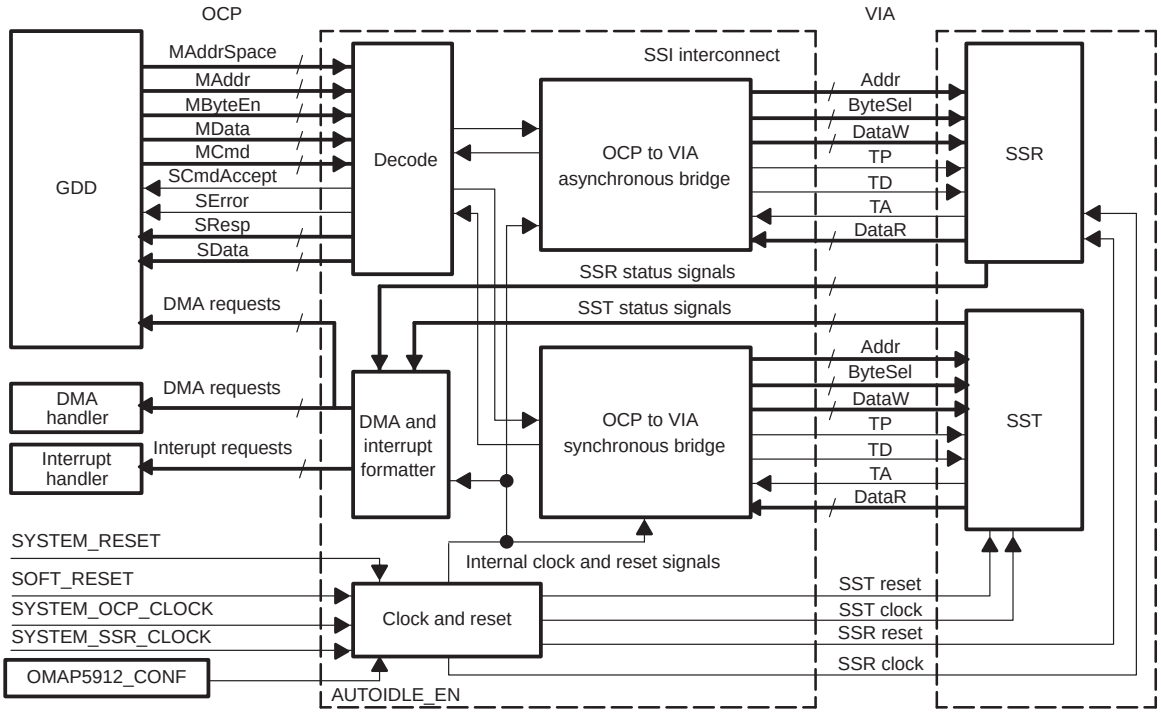
- DMA and interrupt formatter

  DMA and interrupt formatter is responsible for generating DMA and interrupt requests from the SSR and SST status signals.

- Clock and reset module

  The clock and reset module manages clock and reset signals for the SSI interconnect module and the SSI (SSR and SST) module. This module re-synchronizes reset signals and implements the AUTOIDLE mechanism for the SSI interconnect module.

Figure 10. SSI Interconnect



4.2 OCP to VIA Asynchronous Bridge

OCP to the VIA asynchronous bridge is a connection bridge between an OCP bus and a VIA bus. This submodule is a target on the OCP bus and a master on the VIA bus. On the OCP, this bridge provides only basic functionality signals plus SError (slave error). The slave error signal is needed because only basic read and basic write transactions are supported.

Burst transactions are treated as a sequence of basic read and write accesses and do not cause SError to be asserted. When unsupported commands occur, SError is asserted.

4.3 OCP to VIA Synchronous Bridge

OCP to the VIA synchronous bridge is a connection bridge between an OCP bus and a VIA bus. This submodule is a target on the OCP bus and a master on the VIA bus.

On the OCP part, this bridge provides only basic functionality signals plus slave error.

4.4 Decode

This submodule decodes the address received from the OCP initiator. If the current OCP command is for SSR or SST, the decode block forwards this command to that block while keeping the other one inactive.

For SSI, the allocated address space begins at 0x30000000 and ends at 0x30000FFF, in case the address is generated by GDD directly. When the address is provided by the GDD configuration port, the allocated address space begins at 0x000 and ends at 0xFFF. If the address does not belong to one of these two address spaces, SError is asserted. There are 4K bytes of memory allocated for SSI, divided in the following mode: 2K bytes for SST and 2K bytes for SSR. For basic compatibility with the previous version of the chip, the SST address space is assigned the lower address range, and the SSR address space uses the higher addresses.

Table 14. SSI Address Spaces

	Start Address	End Address	MAddrSpace	Address Width
SSI address space by GDD	0x3000_0000	0x3000_0FFF	0000	32 bits
SST address space	0x3000_0000	0x3000_07FF	0000	32 bits
SSR address space	0x3000_0800	0x3000_0FFF	0000	32 bits
SSI address space by OMAP	0x000	0xFFF	0001	12 bits
SST address space	0x000	0x7FF	0001	12 bits
SSR address space	0x800	0xFFF	0001	12 bits

4.5 DMA and Interrupt Formatter

The DMA and interrupt block formats status signals provided by SSR and SST and generates interrupts and DMA requests.

When programming SSR or SST in interrupt mode, take care to avoid reading the buffer before the interrupt signal is asserted.

4.6 Clock and Reset Module

The clock and reset module manages the clock and reset signals for the SSI interconnect module and the SSI (SSR and SST) module. This module resynchronizes reset signals and implements the AUTOIDLE mechanism for the SSI interconnect module.

4.7 Programming Model

4.7.1 OCPI REGISTERS (MPU Address: FFFE:C320)

The OCPI interface allocates memory spaces to the initiator in secure mode and logs unauthorized data transfers and aborts.

See the *OMAP3.2 Technical Reference Manual* (SWPU019E) for additional information.

4.7.2 OCPT REGISTERS (MPU Address: FFFE:CC00)

The OCP-T2 interface manages priority between concurrent data transfers and logs unauthorized data transfers and aborts.

See the *OMAP3.2 Technical Reference Manual* (SWPU019E) for additional information.

4.7.3 Power Saving Modes

To avoid free-running clocks in the OCP and the SSI interconnects, it is possible to shut off the clocks when no data transfer is in progress. This feature is called autoidle.

Bit [25] of MOD_CONF_CTRL_1 in OMAP5912 configuration enables the OCP interconnect to use its autoidle features.

Reset value is 0x1.

Bit [26] of MOD_CONF_CTRL_1 in OMAP5912 configuration enables the SSI interconnect to use its autoidle features.

Reset value is 0x1.

5 On-Chip/Off-Chip Memory and Peripheral Access Latencies

Table 15 and Table 16 list latencies for accesses either to off-chip memories or on-chip memories and peripherals.

All accesses are calculated assuming a 200-MHz MPU and DSP clocks.

Table 15 lists the OMAP3.2 timing parameters.

Table 15. OMAP3.2 Timing Parameters

OMAP3.2			
MPU clock cycle (ns)	5		
DSP clock cycle (ns)	5		
TC clock cycle	10		
MCU 50-MHz TIPB internal peripheral bus cycle (ns)	20		
100-MHz dynamic and static switch cycle (ns)	10		
DSP 33-MHz TIPB internal peripheral bus (ns)	30		
Memories			
Flash initial latency (TC cycle)	10	FLASH_INIT (ns)	100
Flash access clock cycle (TC cycle) @50-MHz device	2	FLASH_BURST (ns)	20
SDRAM RAS latency (TC cycle)	3	RAS_T (ns()	30
SDRAM CAS latency (TC cycle) func(memory type)	3	CAS_T (ns)	30†/20‡
SDRAM access clock cycle (TC cycle) @100-MHz device	1	BURSTCYCLE_T (ns)	5†/10‡
Secure RAM write latency (TC cycle) @ 100 MHz	3		
Secure ROM and RAM initial read latency (TC cycles) @100 MHz	3		
Secure ROM and RAM access clock cycles in (TC cycles) @ 100 MHz	1		
Datum per cycle func (memory type)	2†/1‡		
OMAP5912 Peripherals			
GDD functional clock (ns)	10		
SSR functional/interface clock (ns)	5		
SST functional/interface clock (ns)	10		
VLYNQ clock (ns)	10		
Computing			
Memory type = MDDR/SDR	Mddr†/Sdr‡		
Assumed address offset for I fetch miss (1−4)	1		
Flash width	16		
SDRAM width	16		

† Timing parameters are calculated for a mobile DDR.
‡ Timing parameters are calculated for a standard SDRAM.

Table 15. OMAP3.2 Timing Parameters (Continued)

DMA		
System DMA clock (ns)	10	
DSP DMA clock (ns)	5	
CCP		
CCP clock (ns)	20	CCP cycle

† Timing parameters are calculated for a mobile DDR.
‡ Timing parameters are calculated for a standard SDRAM.

Table 16 lists access time calculations for different class of peripherals.

Table 16. Peripheral Access Time Calculations

		Burst Read				**Single Read**		**Single Write**	
		First Data		**Line Fill**					
		ns	Initiator cycles	ns	Initiator cycles	ns	Initiator cycles	ns	Initiator cycles
MPU to boot ROM*	Burst read/write, 32b read, 16/8b read	70	14	185	37	70	14		
MPU to secure ROM*	Burst read/write, 32b read, 16/8b read	70	14	185	37	70	14		
MPU to secure RAM*	Burst read/write, 32b read/write, 16/8b read/write	70	14	185	37	70	14	70	14

† This latency value includes the following:
− Initial latency: from enable to 1st transaction
− Pipeline latency: from source to destination
− Close latency: from last acknowledge to release of the channel
− For the normal and dedicated P-channel case
‡ Page open: external SDRAM is a mobile DDR
§ 51 equivalent TC cycles = 19 TC cycles plus 16 CCP interface cycles

Table 16. Peripheral Access Time Calculations (Continued)

		Burst Read				Single Read		Single Write	
		First Data		Line Fill					
		ns	initiator cycles	ns	initiator cycles	ns	initiator cycles	ns	initiator cycles
OCP-T2 Peripherals									
MPU to SSR	Read					155	31		
MPU to SST	Write							110	22
OCP-I Interconnect									
MPU to GDD	Read/write					70	14	70	14
MPU Public Peripheral									
MPU to UART	Read/write					135	27	155	31
GDD Accesses									
GDD access to SSR	DMA request to read/write					125	12.5		
GDD access to SST	DMA request to read/write							50	5
GDD access to SDRAM	Request to acknowledge read/write	175†/200‡	17.5†/ 20‡	145†/ 170‡	14.5†/ 17‡	140†/ 130‡	14†/ 13‡	110†/ 100‡	11†/10‡
MPU Private Peripherals									
MPU access to level1 int handler register	Read/write					80	16	100	20

† This latency value includes the following:
− Initial latency: from enable to 1st transaction
− Pipeline latency: from source to destination
− Close latency: from last acknowledge to release of the channel
− For the normal and dedicated P-channel case
‡ Page open: external SDRAM is a mobile DDR
§ 51 equivalent TC cycles = 19 TC cycles plus 16 CCP interface cycles

Table 16. Peripheral Access Time Calculations (Continued)

		Burst Read				Single Read		Single Write	
		First Data		Line Fill					
GDMA Accesses									
GDMA pipeline latency (1)	GDMA pipeline latency(initial, intermediate, close)					230	23	200	20
GDMA peripheral latency	From request to 4*32-bit read from CCP FIFO§	510	51						
GDMA EMIFF latency	DMA read/write from/to EMIFF	145†/170‡	14.5†/17‡	135†/160‡	13.5†/16‡	110†/100‡	11†/10‡	100†/90‡	10†/9‡
DSP Public Peripherals									
DSP to McBSP access	16 bits read/write					400	80	400	80
DSP Private Peripherals									
DSP to level 2.1 interrupt handler access	16 bits read/write					75	15	90	18
DSP_DMA Accesses									
DSP_DMA to EMIFF access	16 bits read/write (2)					130†/120‡	26†/24‡	120†/110‡	24†/22‡

† This latency value includes the following:
− Initial latency: from enable to 1st transaction
− Pipeline latency: from source to destination
− Close latency: from last acknowledge to release of the channel
− For the normal and dedicated P-channel case
‡ Page open: external SDRAM is a mobile DDR
§ 51 equivalent TC cycles = 19 TC cycles plus 16 CCP interface cycles

Table 16. Peripheral Access Time Calculations (Continued)

		Burst Read		Single Read		Single Write	
		First Data	Line Fill				
Other							
DSP to GDD access	OCPT2 :16 bits read/write from/to GDD			80	16	80	16
DSP to SSI access	OCPT2: 16 bits read/write from/to SSI			90	18	90	18
GDMA peripheral latency, NAND	From request to 4*32-bit read from NAND flash controller	710	71				
MPU Public Peripheral (OCP)							
MPU to NAND controller	MPU 32-bit read from NAND flash controller	50	10				

† This latency value includes the following:
− Initial latency: from enable to 1st transaction
− Pipeline latency: from source to destination
− Close latency: from last acknowledge to release of the channel
− For the normal and dedicated P-channel case
‡ Page open: external SDRAM is a mobile DDR
§ 51 equivalent TC cycles = 19 TC cycles plus 16 CCP interface cycles

Index

P

S

T

OMAP5912 Multimedia Processor
Timers
Reference Guide

Literature Number: SPRU759B
October 2004

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.

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
		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 document describes various timers of the OMAP5912 multimedia processor.

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.

Related Documentation From Texas Instruments

Documentation that describes the OMAP5912 device, related peripherals, and other technical collateral, is available in the OMAP5912 Product Folder on TI's website: www.ti.com/omap5912.

Trademarks

OMAP and the OMAP symbol are trademarks of Texas Instruments.

Contents

Figures

Tables

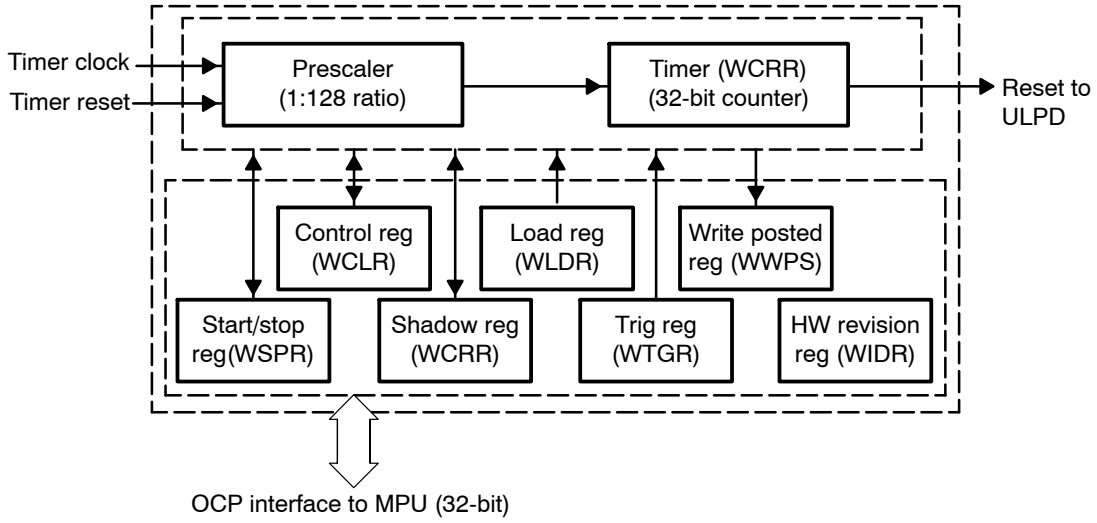
Timers

This document describes various timers of the OMAP5912 multimedia processor.

1 32-Bit Watchdog Timer General Overview

The 32-bit watchdog timer (see Figure 1) is an upward counter that generates a reset to the ultra-low power device (ULPD) module upon an overflow condition. This counter can be accessed, loaded, and cleared by accessing registers through a 16-/32-bit open-core protocol (OCP) interface.

Figure 1. 32-Bit Watchdog



The 32-bit watchdog is an upward 32-bit counter coupled with a prescaler stage. The prescaler ratio can be set between 1 and 128 clock ratios by accessing the PTV field and the PRE field of the watchdog control register (WCLR). 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 specific start/stop sequence applied to the watchdog start/stop register (WSPR) can start/stop the watchdog.

1.1 Posted and Nonposted Writes

There are two clock domains in the watchdog: the timer clock and the interface clock. Table 1 lists these clocks and their frequencies.

Table 1. Watchdog Timer Clock Domains

Device	Timer Clock	Interface Clock
32-kHz watchdog	CLK32K_IN: 32 kHz	MPU peripheral clock: 96 MHz†

† The MPU peripheral clock is typically at half the frequency of the MPU clock frequency.

- The counter and its related registers (WCLR, WCRR, WLDR, WTGR and WSPR registers) are clocked by the timer clock.
- The read/write interface (designated as the OCP interface) which manages the data flow to and from the MPU is clocked by the interface clock. Timer registers WDIR, WD_SYSCONFIG, WD_SYSSTATUS and WWPS are also clocked by the interface clock

For the 32-kHz watchdog timer, any write to a register located in the timer clock domain is done in posted mode.

In effect, the write is acknowledged after it has been resynchronized to the 32-kHz clock. You can perform concurrent writes to the timer clock domain registers and verify that they were acknowledged by reading the status bit in the write posted status register (WWPS). In the case where a register associated write posted status bit is set, a write from the MPU is not acknowledged (see Table 2).

Table 2. Write Posted Status Register (WWPS)

Bit	Name	Function
4	W_PEND_WSPR	When equal to 1, a write is pending to the WSPR register.
3	W_PEND_WTGR	When equal to 1, a write is pending to the WTGR register.
2	W_PEND_WLDR	When equal to 1, a write is pending to the WLDR register.
1	W_PEND_WCRR	When equal to 1, a write is pending to the WCRR register.
0	W_PEND_WCLR	When equal to 1, a write is pending to the WCLR register.

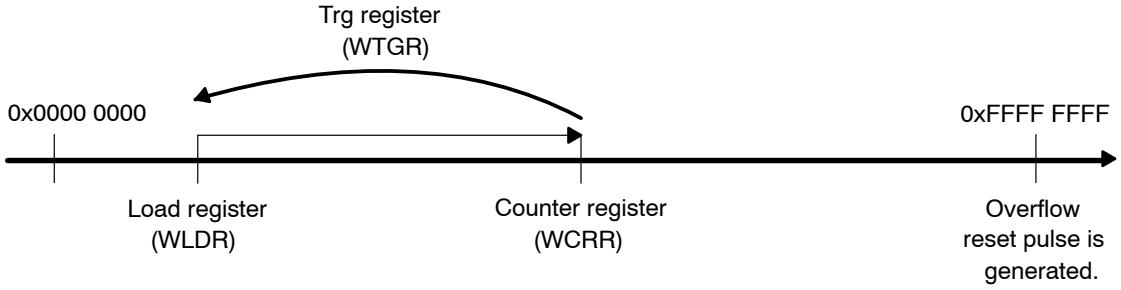
1.2 Reset Context

After reset release, the 32-bit watchdog is on. To get the reset values, software must read the PTV field of WCLR and the 32-bit register to catch the static configuration of the module.

1.3 Overflow/Reset Generation

When the watchdog counter register (WTCR) reaches overflow, an active-low pulse is generated to the ULPD. After reset generation, the counter is automatically reloaded with the watchdog load register (WLDR) and the prescaler counter is reset. The prescaler ratio remains unchanged. Then, after the reset pulse has been generated, the prescaler counter and timer counter are incremented again. Figure 2 shows the 32-bit watchdog functional overview.

Figure 2. 32-Bit Watchdog- General Functional Overview



1.4 Triggering New Reload

To reload the timer counter and reset the prescaler counter values without reaching overflow, a reload command can be executed by accessing the watchdog trigger register (WTGR) using a specific reload sequence.

The specific reload sequence is performed when the written value on the watchdog trigger register (WTGR) is different from its previous value. In this case, reload is executed in the same way as overflow autoreload, without a reset pulse generation. The timer is loaded with the watchdog load register (WLDR) value and the prescaler counter is reset.

1.5 Prescaler Value/Timer Reset Frequency

The 32-bit watchdog is composed of a prescaler stage and a timer counter. The prescaler stage is clocked with the timer clock and acts as a clock divider for the timer counter stage. The ratio can be managed by accessing the ratio definition field of the control register (PTV of the watchdog control register (WCLR)) and enabled with the PRE field of WCLR. Table 3 provides the divisor ratios.

Table 3. Prescaler Clock Ratios

PRE Bit (WCLR)	PTV Bits (WCLR)	Divisor (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 estimated by using:

Reset Period = (0xFFFF FFFF – WLDR + 1) x Timer Clock Period x Clock Divider (PS),

where Timer Clock = 1/Timer clock frequency and PS = $2^{(PTV)}$.

For example, if we consider a timer clock input of 32 kHz 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 4.

Table 4. 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

1.6 Accessing Watchdog Registers

All registers are 32-bit wide, accessible via OCP interface with 16-bit or 32-bit OCP access (read/write).

1.7 Modifying WCRR, WLDR, and Prescaler Ratios

To modify the timer counter value (WCRR), prescaler ratio (PTV of WCLR), or load value (WLDR), the 32-bit watchdog must be disabled by using a specific start/stop sequence (WSPR).

In this case, the load register value and prescaler ratio are updated registers, but new values are taken into account only after a new overflow context or a new trigger command (WTGR).

1.8 WCRR Access Restriction

Because the WCRR register 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 an 32-bit read command or an 16-bit LSB read command. To be sure that a coherent value is read inside WCRR, use 32-bit reads, or, in case of $2 \times$ 16-bit reads, be sure the least significant 16-bit read is performed first (followed by the most significant 16-bit read).

1.9 Start/Stop Command

To start/stop the 32-bit watchdog, access must be made through the start/stop register (WSPR) using a specific sequence.

To disable the 32-bit watchdog, follow this sequence:

1) Write 0xXXXX AAAA in WSPR.
2) Write 0xXXXX 5555 in WSPR.

To enable the 32-bit watchdog, follow this sequence:

1) Write 0xXXXX BBBB in WSPR.
2) Write 0xXXXX 4444 in WSPR.

Other write sequences on WSPR do not have any effect on the start/stop feature of the module.

1.10 Watchdog Module Under Emulation

During emulation mode, the 32-bit watchdog can/cannot continue to run according to the value of the EMUFREE bit of the system configuration register (WD_SYSCONFIG).

If EMUFREE is 1, timer execution is not stopped in emulation mode and a reset pulse is still generated when overflow is reached.

If EMUFREE is 0, counters (prescaler/timer) are frozen and increment starts again after exit from emulation mode.

1.11 Watchdog Timer Registers

Address of register: Start address + Address offset
Registers width: 32 bits (can be accessed as 32 bits or as 2 x 16 bits – OCP access)

Table 5 lists the 32-bit watchdog timer registers. Table 6 through Table 14 describe the register bits.

Table 5. Watchdog Timer Registers

Base Address = 0xFFFE B000			
Name	**Description**	**R/W**	**Offset**
WIDR	Watchdog HW revision (ID)	R	0x00
RESERVED†	Reserved	-	0x04
RESERVED†	Reserved	-	0x08
RESERVED†	Reserved	-	0x0C
WD_SYSCONFIG	Watchdog system configuration	R/W	0x10
WD_SYSSTATUS	Watchdog status	R	0x14
RESERVED†	Reserved	-	0x18
RESERVED†	Reserved	-	0x1C
RESERVED†	Reserved	-	0x20
WCLR	Watchdog control	R/W	0x24
WCRR	Watchdog counter	R/W	0x28
WLDR	Watchdog load	R/W	0x2C
WTGR	Watchdog trigger	R/W	0x30

† To keep mapping compatibility between 32-bit watchdog and dual-mode timer modules.

Table 5. Watchdog Timer Registers (Continued)

Base Address = 0xFFFE B000			
Name	**Description**	**R/W**	**Offset**
WWPS	Watchdog write pending	R	0x34
RESERVED†	Reserved	-	0x38
RESERVED†	Reserved	-	0x3C
RESERVED†	Reserved	-	0x40
RESERVED†	Reserved	-	0x44
WSPR	Watchdog start/stop	R/W	0x48

† To keep mapping compatibility between 32-bit watchdog and dual-mode timer modules.

Table 6. Watchdog HW Revision (ID) Register (WIDR)

Base Address = 0xFFFE B000, Offset = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:8	RESERVED	Reserved	R	0x000000
7:0	WD_REV	Module HW revision number indicates the revision number of the current timer module. This value is fixed in hardware. Least significant 4 bits indicate a minor revision. Most significant 4 bits indicate a major revision. A reset has no effect on the value returned.	R	0x10

This read-only, 32-bit register (accessible in 16-bit mode) contains the hardware revision number of the module. A write to this register has no effect.

Table 7. Watchdog System Configuration Register (WD_SYSCONFIG)

Base Address = 0xFFFE B000, Offset = 0x10				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:6	RESERVED	Write 0s for future compatibility Read returns 0	R/W	0x0000000
5	EMUFREE	Enable sensitivity to suspend signal (emulation) 0: The module freezes its internal logic upon suspend assertion. 1: The module ignores the suspend input.	R/W	0
4:2	RESERVED	Write 0s for future compatibility Read returns 0	R/W	0x0
1	SOFTRESET	Software reset. Set this bit to 1 to trigger a module reset. This bit is automatically reset by hardware. During read, it always returns 0. 0: Normal mode 1: The module is reset.	R/W	0
0	AUTOIDLE	Internal OCP gating strategy 0: Interface clock is free-running. 1: Interface clock gating is active.	R/W	0

Table 8. Watchdog System Status Register (WD_SYSSTATUS)

Base Address = 0xFFFE B000, Offset = 0x14				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:8	RESERVED	Reserved	R	0x0000000
7:1	RESERVED	Reserved	R	0x00
0	RESETDONE	Internal reset monitoring 0: Internal module reset is ongoing. 1: Reset completed	R	0

This register provides system status information about the module.

Table 9. Watchdog Control Register (WCLR)

Base Address = 0xFFFE B000, Offset = 0x24				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:6	RESERVED	Reserved	R/W	0x00000
5	PRE	0: The timer clock input pin clocks the counter. 1: The divided input pin clocks the counter.	R/W	1
4:2	PTV	Prescaler ratio value:	R/W	0x0
0:1	RESERVED	Reserved	R/W	0x0

This 32-bit register is accessible in 16-bit mode.

Table 10. Watchdog Counter Register (WCRR)

Base Address = 0xFFFE B000, Offset = 0x28				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:0	TIME_COUNTER	Value of timer counter	R/W	0xFFF0 0000

This 32-bit register is accessible in 16-bit mode.

The WCRR register is a 32-bit register (16-bit addressable). The MPU can perform either a 32-bit access or two 16-bit accesses to the register, whereas the DSP performs two consecutive 16-bit transactions. Because the interface clock is completely asynchronous with the timer clock, some synchronization is done to ensure that the WCRR value is not read while it is incremented.

In 16-bit-mode, the following sequence must be followed to read the WCRR register properly:

1) Read the lower 16 bits of the WCRR register (offset = 0x08). When the WCRR is read, the lower 16 bits (LSBs) are read, and the upper 16 bits (MSW) of the WCRR register are stored in a temporary register (shadow register).

2) Read the upper 16 bits of the WCRR register (offset = 0x0A). The value of the upper 16 bits (MSW), which has been stored in the temporary register, is read.

To read the value of WCRR correctly, the first OCP read access must be to the lower 16-bits (that is, offset = 0x08), followed by OCP read access to the upper 16-bits (that is, offset = 0x0A).

WCRR value can be modified by writing on WCRR register and only if watchdog has been disabled (using WSPR register).

Table 11. Watchdog Load Register (WLDR)

Base Address = 0xFFFE B000, Offset = 0x2C				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:0	TIME_LOAD	Timer counter value loaded on overflow in autoreload mode or on WTGR write access: 32-kHz watchdog	R/W	0xFFF0 0000

This 32-bit register is accessible in 16-bit mode.

This register specifies the load value of the timer counter. This load value is effective (loaded inside WCRR) after an overflow context or by triggering a new reload via WTGR register.

Table 12. Watchdog Trigger Register (WTGR)

Base Address = 0xFFFE B000, Offset = 0x30				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:0	TTGR_VALUE	Trigger value. Triggers a reload of the watchdog module by writing a value different from the previous value inside WTGR	R/W	0x0000 0000

This 32-bit register is accessible in 16-bit mode.

A write to the watchdog trigger register (WTGR) reloads the value contained in the WLDR register value to the watchdog counter register (WCRR).

This command is performed by writing a value inside WTGR different from the previous value inside WTGR.

Table 13. Watchdog Write Pending Register (WWPS)

Base Address = 0xFFFE B000, Offset = 0x34				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:5	RESERVED	Reserved		0x000000
4	W_PEND_WSPR	When equal to one, a write is pending to the WSPR register.	R	0
3	W_PEND_WTGR	When equal to one, a write is pending to the WTGR register.	R	0
2	W_PEND_WLDR	When equal to one, a write is pending to the WLDR register.	R	0

Table 13. Watchdog Write Pending Register (WWPS) (Continued)

Base Address = 0xFFFE B000, Offset = 0x34				
Bit	**Name**	**Function**	**R/W**	**Reset**
1	W_PEND_WCRR	When equal to one, a write is pending to the WCRR register.	R	0
0	W_PEND_WCLR	When equal to one, a write is pending to the WCLR register.	R	0

This 32-bit register is accessible in 16-bit mode.

In write-posting mode, the software must read the pending write status bits (watchdog write-posted status register bits [4:0]) to ensure that the following write access is not discarded because of an ongoing write synchronization process. These bits are automatically cleared by internal logic when the write to the corresponding register is acknowledged.

Table 14. Watchdog Start/Stop Register (WSPR)

Base Address = 0xFFFE B000, Offset = 0x48				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:0	WSPR_VALUE	WSPR value switches on/off watchdog module with specific protocol.	R/W	0x0000 0000

This 32-bit register is accessible in 16-bit mode.

The 32-bit watchdog (timer counter + prescaler counter) can be started/stopped by accessing WSPR with a specific start/stop sequence. To disable the 32-bit watchdog, follow this sequence:

- ❏ Write 0x XXXX AAAA in WSPR.
- ❏ Write 0x XXXX 5555 in WSPR.

To enable the 32-bit watchdog, follow this sequence:

- ❏ Write 0x XXXX BBBB in WSPR
- ❏ Write 0x XXXX 4444 in WSPR.

A read on this register (WSPR) returns the last data written in WSPR.

2 32-kHz Watchdog Timer

The software can access the 32-kHz watchdog at any time.

All writes to the watchdog register file are write posted, so the watchdog timer write command is granted before the actual write in the timer clock domain is performed.

This mode allows software to perform concurrent writes on 32-bit watchdog registers and to manage them at software level by reading the status of posted writes (by reading status bit of the write posted status register (WWPS)).

- ❏ The prescaler value is forced to 1 at power up reset.
- ❏ The WLDR value is forced to 0xFFF00000 at power up reset.
- ❏ The timer input clock is 32 kHz.

Table 15. Time-Out for 32-kHz Watchdog Timer

Clock Frequency (kHz)	**32**
PTV	0
WLDR value	0xFFF0 0000
Reset period(s)	33
Default Configuration at Power-Up Reset	
PTV	0
WLDR value	0xFFF0 0000

The time-out value generates a reset that is logged into the ULPD reset status register (Bit [3]).

After power-up reset, the time-out period can be changed by programming either the prescaler (PTV) or the timer-load value (WLDR).

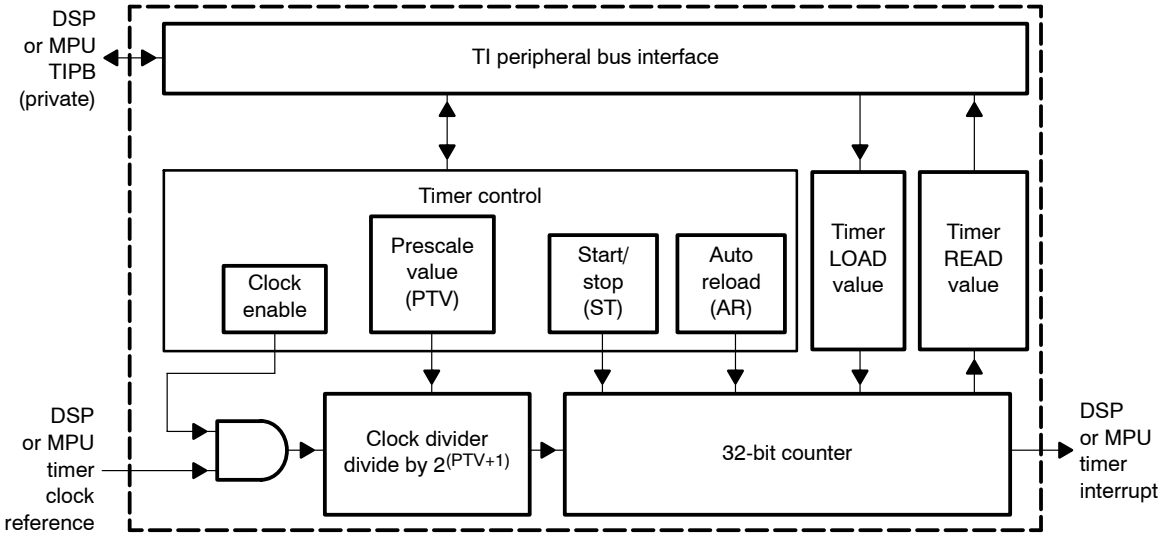
3 OMAP3.2 Operating System Timer

There are six timers for the OS and general-purpose housekeeping functions inside the hardware engine. Three are in the MPU subsystem (controlled by the MPU), and the other three are controlled by the DSP._Figure 3 shows the timer block diagram.

The functionality for all timers is identical, except the MPU timer interface to the 32-bit MPU private TI peripheral bus (TIPB) and the DSP timer interface to the 16-bit DSP private TIPB.

The timer is configured either in autoreload or in one-shot mode with on-the-fly read capability. The timer generates an interrupt to the MPU or DSP when the value is equal to zero.

Figure 3. Timer Block Diagram



3.1 DSP and MPU OS Timer Start and Stop

An OS timer can be started by setting the ST bit of the DSP_CNTL_TIMER or the MPU_CNTL_TIMER register. At start, the timers are loaded with the values programmed to the DSP_LOAD_TIMER_HI, DSP_LOAD_TIMER_LO, and MPU_LOAD_TIMER registers. The timer is stopped by resetting the ST bit. When stopped, the timer value is frozen at the current value.

3.2 DSP and MPU OS Timer

If the autoreload (AR) bit is set in DSP_CNTL_TIMER or MPU_CNTL_TIMER, a new value (from the load register) is loaded into the timer when it passes through zero.

If the AR bit is reset, the timer decrements from the loaded value to zero and then stops.

3.3 Reading OS Timer Values

The timer values can be read from the respective DSP or MPU read timer registers either on-the-fly or after the timer is stopped.

- For the MPU, MPU_READ_TIMER is a 32-bit register and can be read directly.
- By contrast, the data width of the DSP is only 16 bits. To read the value of the DSP OS timers correctly, the first access must be to the upper 16 bits.

  When the upper read occurs, the lower 16 bits are simultaneously stored in a temporary register. An access to the lower 16 bits provides the content of this temporary register as the read result. A read from DSP_READ_TIMER_HI followed by a read from DSP_READ_TIMER_LO provides the correct DSP OS timer read result.

3.4 DSP and MPU OS Timer Input Clocks

This section describes clocking for the DSP and MPU OS timers.

3.4.1 Configuration of the Input Clock for DSP 32-Bit OS Timers

The input clock reference for the DSP timer modules is controlled from the clock generation and reset module. Programming the TIMXO bit of the DSP prescaler selection register DSP_CKCTL in the clock module selects between two possible clock sources:

- Programming the TIMXO bit to 0 selects the main input reference clock as the clock reference for the DSP timers.
- Programming the TIMXO bit to 1 selects the output from the DPLL1 module as the source for the DSP 32-bit OS timers.

The default is referenced from DPLL1. When operating from DPLL1, it is recommended to stop the DSP OS timer before programming a change to the DPLL1 frequency divisor. See *Clock Generation and Reset Management* in the Multimedia Processor OMAP 3.2 Subsystem Reference Guide (SPRU749) for details on DSP_CKCTL and on programming the DPLL1 module.

3.4.2 Configuration of the Input Clock for MPU 32-Bit OS Timers

The input clock reference for the MPU timer modules is controlled from the clock generation and reset module. Programming the ARM_TIMXO bit of the MPU prescaler selection register ARM_CKCTL: in the clock module selects between two possible clock sources.

- Programming the ARM_TIMXO bit to 0 selects the main input reference clock as the clock reference for the MPU timers.
- Programming the ARM_TIMXO bit to 1 selects the output from the DPLL1 module as the source for the MPU 32-bit OS timers.

The default is referenced from DPLL1. When operating from DPLL1, it is recommended to stop the MPU OS timer before programming a change to the DPLL1 frequency divisor. See *Clock Generation and Reset Management* in the Multimedia Processor OMAP 3.2 Subsystem Reference Guide (SPRU749) for details of ARM_CKCTL: and on programming the DPLL1 module.

3.4.3 Input Clock Enable

Input reference clocks for the DSP and MPU OS timer modules are ANDed with a clock enable signal to gate the clock from timer internal logic. The clock is enabled to the timers when the CLOCK_ENABLE bit is set in DSP_CNTL_TIMER and in MPU_CNTL_TIMER.

3.5 Timer Interrupts

An interrupt occurs when the corresponding timer decrements to zero.

DSP and MPU OS timer interrupt periods are defined by:

1) The programmed value for the DSP and MPU OS timer input clock references (see Section 3.4.1 and Section 3.4.2*).*
2) The value of the prescaler bit field, PTV, in DSP_CNTL_TIMER and in MPU_CNTL_TIMER. Table 16 lists valid entries for PTV.
3) The value of the load registers, DSP_LOAD_TIMER_HI, DSP_LOAD_TIMER_LO, and MPU_LOAD_TIMER.

Table 16. Valid Prescaler Values for OS Timer Configuration

PTV Bit Field in DSP and MPU OS Timer Control Register	Input Clock Divisor
000	2
001	4
010	8
011	16
100	32
101	64
110	128
111	256

The following equations are used to determine the timer interrupt periods.

- For the DSP OS timer:

$$T_{DSP\_interrupt} = T_{DSP\_ref\_clk} \times (< DSP\_LOAD\_TIMER\_HI, LO > + 1) \times 2^{(PTV+1)}$$

- For the MPU OS timer

$$T_{MPU\_interrupt} = T_{MPU\_ref\_clk} \times (< MPU\_LOAD\_TIMER > + 1) \times 2^{(PTV+1)}$$

Where $T_{DSP\_ref\_clk}$ and $T_{MPU\_ref\_clk}$ are clock periods of the DSP and MPU OS timer input clocks.

Based on these equations, examples for various MPU OS timer interrupt periods are calculated in Table 17 for a hypothetical MPU OS timer input clock reference frequency of 100 MHz ($T_{MPU\_ref\_clk}$ = 10 ns).

Table 17. Example MPU OS Timer Interrupt Periods

MPU_LOAD_TIMER	PTV = 000	PTV = 111
0x0 (granularity)	20 ns	2.56 µs
0xFFFFFFFF(max period)	85.9 s	10995 s (3 hr 3 min 15 s)

3.6 Timer Programming

To activate the DSP and MPU OS timers, perform the following sequence:

1) Program the load register.

   The content of the load register is loaded into the timer read register when the ST bit is set.

2) Write to DSP_CNTL_TIMER or to MPU_CNTL_TIMER to configure and start the timer.

   - Set the PTV value as the dividing factor for the timer clock.
   - Set the autoreload feature with the AR bit.
   - Set the ST bit to start the timer.

3) You can stop the timer at any time by resetting the ST bit (bit 0) of the DSP_CNTL_TIMER or MPU_CNTL_TIMER.

Note:

Only the ST or CLOCK_ENABLE bits of DSP_CNTL_TIMER or MPU_CNTL_TIMER can be written while the timer is running. Undefined results occur if the prescaler (PTV) or autoreload (AR) bits in the control register or timer load registers are written while the timer is running.

3.7 OS Timer Registers

The following sections describe the DSP and MPU operating system timer registers.

Note:

The register descriptions and base address offsets given here apply identically to each of the three DSP or three MPU OS timers. See the Applications Processor Data Manual (SPRS231), to determine the base address for each timer.

3.7.1 DSP 32-Bit OS Timers

Accesses to the DSP 32-bit OS timer registers are controlled by the DSP private TIPB. Table 18 lists the DSP OS timer registers. Table 19 through Table 23 provide register bit descriptions.

Table 18. DSP OS Timer Registers

Base Address = 0xE100 5000, 0xE100 5800, 0xE100 6000			
Name	**Description**	**R/W**	**Offset**
DSP_CNTL_TIMER	DSP control timer	R/W	0x00
DSP_LOAD_TIMER_LO	DSP load timer value, 1/2 LSW	W	0x04
DSP_LOAD_TIMER_HI	DSP load timer value, 1/2 MSW	W	0x06
DSP_READ_TIMER_LO	DSP read timer value, 1/2 LSW	R	0x08
DSP_READ_TIMER_HI	DSP read timer value, 1/2 MSW	R	0x0A

Table 19. DSP Control Timer Register (DSP_CNTL_TIMER)

Base Address = 0xE100 5000, 0xE100 5800, 0xE100 6000, Offset = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:7	RESERVED			0x000
6	FREE	Specifies what action the timer takes upon receiving a suspend indication from the TIPB bridge. (For example, suspend is indicated if processor execution has been suspended at an emulation breakpoint.) 0: Stop counting on suspend indication (even when ST=1). 1: A suspend indication has no effect on the count.	R/W	0
5	CLOCK_ENABLE	Enables input reference clock to the DSP OS timer module 0: Disable 1: Enable	R/W	0
4:2	PTV	Prescale timer input reference clock 000: Divide input reference clock by 2 001: Divide input reference clock by 4 010: Divide input reference clock by 8 011: Divide input reference clock by 59120: Divide input reference clock by 32 101: Divide input reference clock by 64 110: Divide input reference clock by 128 111: Divide input reference clock by 256	R/W	000

Table 19. DSP Control Timer Register (DSP_CNTL_TIMER) (Continued)

Base Address = 0xE100 5000, 0xE100 5800, 0xE100 6000, Offset = 0x00				
1	AR	0: One-shot mode	R/W	0
		1: Autoreload mode		
0	ST	0: Stop timer value decrement	R/W	0
		1: Start timer value decrement		
		If one-shot mode is selected (AR = 0), this bit is automatically reset by internal logic when the timer value is equal to 0.		

Table 20. DSP Load Timer Value, 1/2 LSW (DSP_LOAD_TIMER_LO)

Base Address = 0xE100 5000, 0xE100 5800, 0xE100 6000, Offset = 0x04				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	DSP_LOAD_TIMER_HI	This value is loaded when the timer passes through 0 or when it starts.	W	Undefined

Table 21. DSP Load Timer Value, 1/2 MSW (DSP_LOAD_TIMER_HI)

Base Address = 0xE100 5000, 0xE100 5800, 0xE100 6000, Offset = 0x06				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	DSP_LOAD_TIMER_LO	This value is loaded when the timer passes through 0 or when it starts.	W	Undefined

Table 22. DSP Read Timer Value, 1/2 LSW (DSP_READ_TIMER_LO)

Base Address = 0xE100 5000, 0xE100 5800, 0xE100 6000, Offset = 0x08				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	DSP_READ_TIMER_LO	Value of the timer bits (15:0): To read a correct value for DSP 32-bit OS timer, the upper 16 bits (from DSP_READ_TIMER_HI) must read prior to reading this register.	R	Undefined

Table 23. DSP Read Timer Value, 1/2 MSW (DSP_READ_TIMER_HI)

Base Address = 0xE100 5000, 0xE100 5800, 0xE100 6000, Offset = 0x0A				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	DSP_READ_ TIMER_HI	Value of the timer bits (31:16): To read correct value for DSP 32-bit OS timer, this register must be read first, followed by lower 16 bits of DSP_READ_TIMER_LO.	R	Undefined

3.7.2 MPU 32-Bit OS Timers

Accesses to the MPU 32-bit OS timer configuration registers are controlled by the MPU private TIPB.

Table 24 lists the MPU OS timer registers. Table 25 through Table 27 provide register bit descriptions.

Table 24. MPU OS Timer Registers

Base Address = 0xFFFE C500, 0xFFFE C600, 0xFFFE C700			
Name	**Description**	**R/W**	**Offset**
MPU_CNTL_TIMER	MPU control timer	R/W	0x00
MPU_LOAD_TIMER	MPU load timer	W	0x04
MPU_READ_TIMER	MPU read timer	W	0x08

Table 25. MPU Control Timer Register (MPU_CNTL_TIMER)

Base Address = 0xFFFE C500, 0xFFFE C600, 0xFFFE C700, Offset = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:7	RESERVED			0x0000000
6	FREE	Specifies what action the timer takes upon receiving a suspend indication from the TIPB bridge. (For example, suspend is indicated if processor execution has been suspended at an emulation breakpoint.) 0: Stop counting on suspend indication (even when ST = 1). 1: Suspend indication has no effect on the count.	R/W	0

Table 25. MPU Control Timer Register (MPU_CNTL_TIMER) (Continued)

Base Address = 0xFFFE C500, 0xFFFE C600, 0xFFFE C700, Offset = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
5	CLOCK_ENABLE	Enable input reference clock to the MPU OS timer module	R/W	0
4:2	PTV	Prescale timer input reference clock	R/W	000
1	AR	0: One-shot mode 1: Autoreload mode	R/W	0
0	ST	0: Stop timer value decrement 1: Start timer value decrement If one-shot mode is selected (AR = 0), this bit is automatically reset by internal logic when the timer value is equal to 0.	R/W	0

Table 26. MPU Load Timer Register Value (MPU_LOAD_TIMER)

Base Address = 0xFFFE C500, 0xFFFE C600, 0xFFFE C700, Offset = 0x04				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:0	MPU_LOAD_TIMER	This value is loaded when the timer passes through 0 or when it starts.	W	UD

Table 27. MPU Read Timer Register Value (MPU_READ_TIMER)

Base Address = 0xFFFE C500, 0xFFFE C600, 0xFFFE C700, Offset = 0x08				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:0	MPU_READ_TIMER	Value of the timer	R	UD

4 Dual-Mode Timer

This peripheral is a 32-bit timer with the following features:

- Counter timer with compare and capture modes
- Autoreload mode
- Start-stop mode
- Programmable divider clock source
- 16-/32-bit addressing
- On the fly read/write registers
- Interrupts generated on overflow, compare, and capture
- Interrupt enable
- Wake-up enable
- Write posted mode
- Dedicated input trigger for capture mode and dedicated output trigger/PWM signal

The timer module contains a free-running upward counter with autoreload capability on overflow. The timer counter can be read and written on the fly (while counting). The timer module includes compare logic to allow interrupt event on programmable counter matching value. A dedicated output signal can be pulsed or toggled on overflow and match event. This offers timing stamp trigger signal or PWM (pulse width modulation) signal sources. A dedicated input signal can be used to trigger automatic timer counter capture and interrupt event, on 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 timer implemented in OMAP5912, there are three possible clock sources:

- The 32-kHz clock
- The system clock
- An external clock source

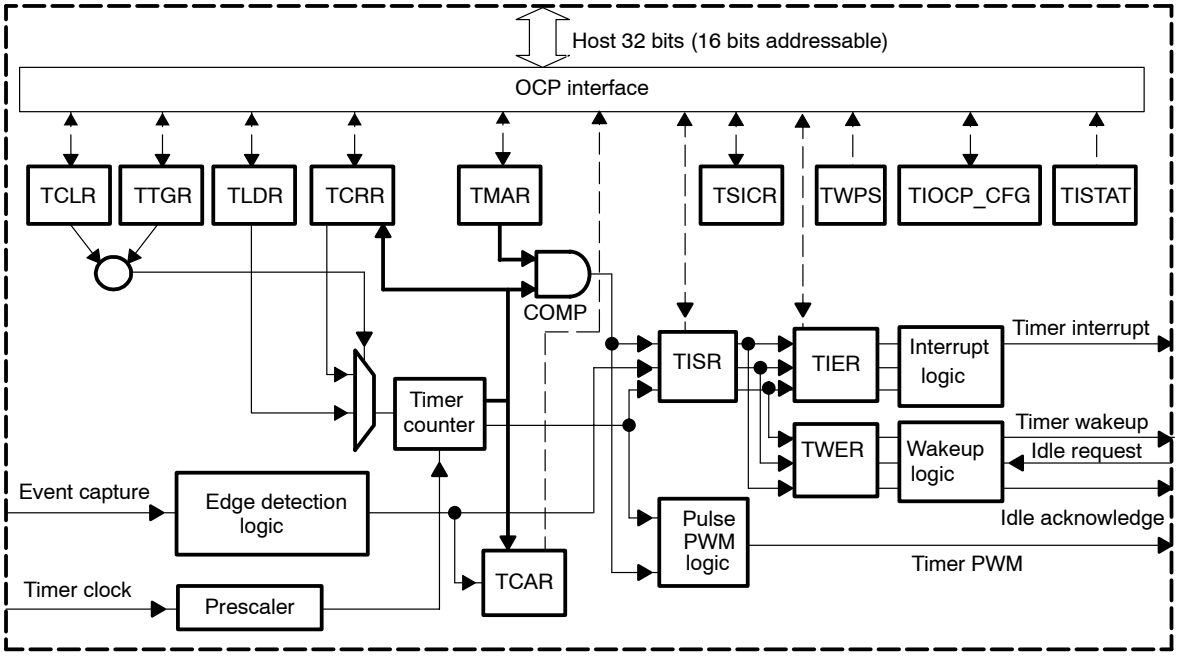
The selection is done in configuration. Refer to the MOD_CONF_CTRL_1 description for additional information.

Table 28 lists the access types for various clock selections, and Figure 4 shows the dual-mode timer.

Table 28. Access Types (Posted Mode vs Nonposted Mode) for Various Clock Selections

Timer Clock Selection	CONF_MOD_GPTIMER_CLK_SEL = 00		CONF_MOD_GPTIM-ER_CLK_SEL = 10	CONF_MOD_GPTIM-ER_CLK_SEL = 01
Timer clock frequency	System clock = 12 MHz	System clock = 19.2 MHz	External timer clock = 20 MHz	32 kHz
Access type	Posted mode	Nonposted mode	Nonposted mode	Posted mode
Action to be taken by software before next write to timer	Software must read the pending write status bits (timer write posted status register bits [4:0]) to ensure that following write access is not discarded because of ongoing write synchronization process.	None	None	Software must read the pending write status bits (timer write posted status register bits [4:0]) to ensure that following write access is not discarded because of ongoing write synchronization process.

Figure 4. Dual-Mode Timer Block Diagram



4.1 Description

The general-purpose timer is an upward counter. It supports three functional modes:

- Timer mode
- Capture mode
- Compare mode

By default, the capture and compare modes are disabled after core reset.

4.2 Mode Functionality

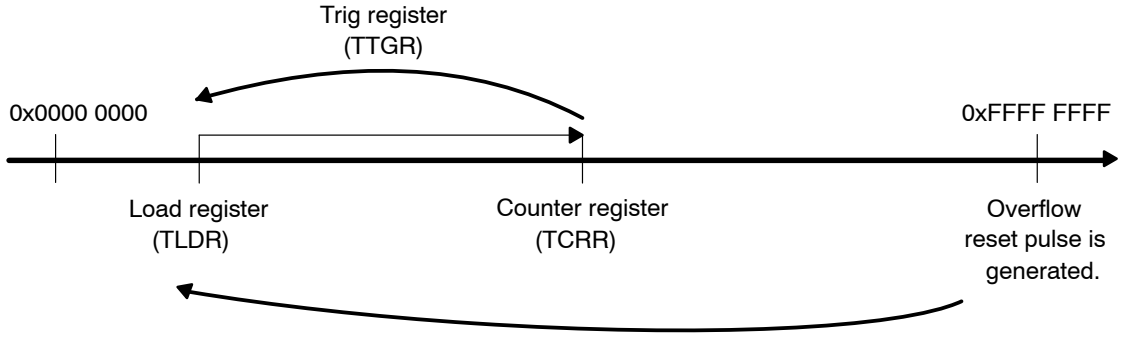
The timer is an upward counter that can be started and stopped at any time through the timer control register (TCLR ST bit). The timer counter register (TCRR) can be loaded when stopped or on the fly (while counting). TCRR can be loaded directly by a TCRR write access with the new timer value. TCRR can also be loaded with the value held in the timer load register TLDR by a trig register (TTGR) write access. The TCRR loading is done regardless of the TTGR written value. The timer counter register TCRR value 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 modules reset is asserted. The timer is maintained in stop after reset is released. When the timer is stopped, TCRR is frozen and it can be restarted from the frozen value, unless TCRR has been reloaded with a new value.

In one-shot mode (TCLR AR bit =0), the counter is stopped after counting overflow (counter value remains at zero).

When autoreload mode is enabled (TCLR AR bit =1), the TCRR is reloaded with the timer load register (TLDR) value after a counting overflow. An interrupt can be issued on overflow if the overflow interrupt enable bit is set in the timer interrupt enable register (TIER OVF_IT_ENA bit =1). A dedicated output pin (timer PWM) can be programmed through TCLR (TRG and PT bits) to generate one positive pulse (prescaler duration) or to invert the current value (toggle mode) when an overflow occurs.

Figure 5 shows the TCRR timing value.

Figure 5. TCRR Timing Value



4.3 Capture Mode Functionality

The timer value in TCRR can be captured and saved in TCAR when a transition is detected on the module input pin (event capture).

Rising edge, falling edge, or both can be selected in TCLR (TCM) to trigger the timer counter capture. The module sets the TISR (TCAR_IT_FLAG bit) when an active edge is detected; at the same time the counter value TCRR is stored in the timer capture register TCAR. If there are several consecutive capture events, the first one is saved in TCAR.

The edge detection logic is reset when either the TCAR_IT_FLAG is cleared by writing a 1 to it, or when the TCM bits in the TCLR register transition from the no-capture mode detection to any other mode. The timer functional clock (input to prescaler) is used to sample the input pin (event capture). Input negative or positive pulses 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 timer interrupt enable register TIER (TCAR_IT_ENA bit).

4.4 Compare Mode Functionality

When the compare enable TCLR (CE bit) is set to 1, the timer value (TCRR) is continuously compared to the value held in the timer match register (TMAR). TMAR value can be loaded at any time (timer counting or stopped). When the TCRR and the TMAR values match, an interrupt is issued if the TIER (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 and a match occur.

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 2^n division ratio value (PTV) can be configured in the TCLR register.

The prescaler counter is reset when the timer counter is stopped or reloaded on the fly.

Table 29 lists the prescaler/timer reload values versus contexts.

Table 29. Prescaler/Timer Reload Values Versus Contexts

Contexts	Prescaler Counter	Timer Counter
Overflow (when autoreload on)	Reset	TLDR
TCRR write	Reset	TCRR
TTGR write	Reset	TLDR
Stop	Reset	Frozen

4.6 Pulse-Width Modulation

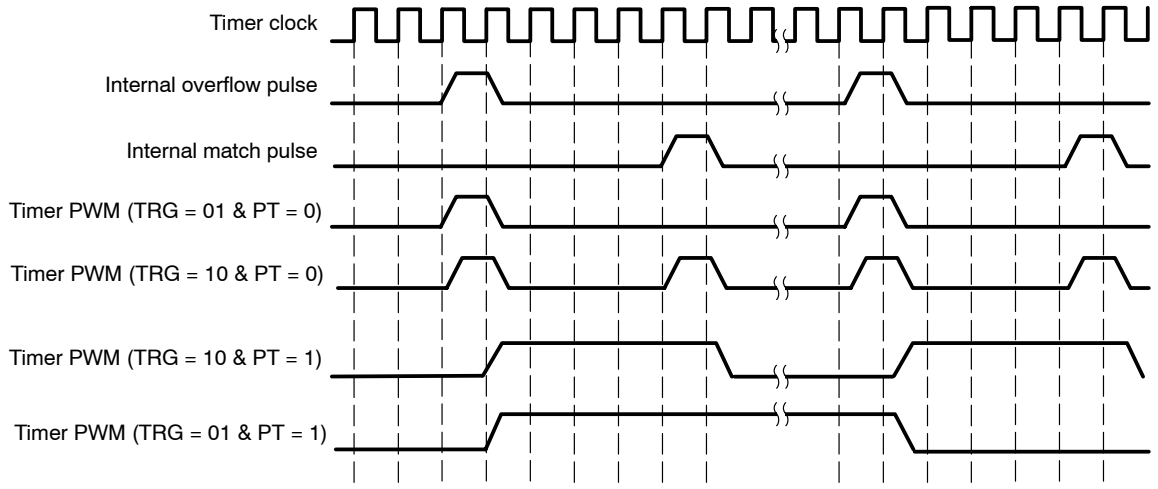
The timer can be configured to provide a programmable pulse-width modulation (timer PWM) output. The timer PWM output pin can be configured to toggle on an event. TCLR (TRG bits) determine on which register value the timer PWM pin toggles. Either overflow or match can be used to toggle the timer PWM pin when a compare condition occurs. The TCLR (SCPWM bit) can only be programmed to set or clear the timer PWM output signal while the counter is stopped or the trigger is off. This allows the output pin to be set to a known state before modulation starts. The modulation is synchronously stopped when TRG bit is cleared and overflow occurred. This allows the output pin to be set to a known state when modulation is stopped.

In Figure 6, the internal overflow pulse is set each time the (0xFFFF FFFF – 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 6 TCLR (SCPWM bit) is set to 0.

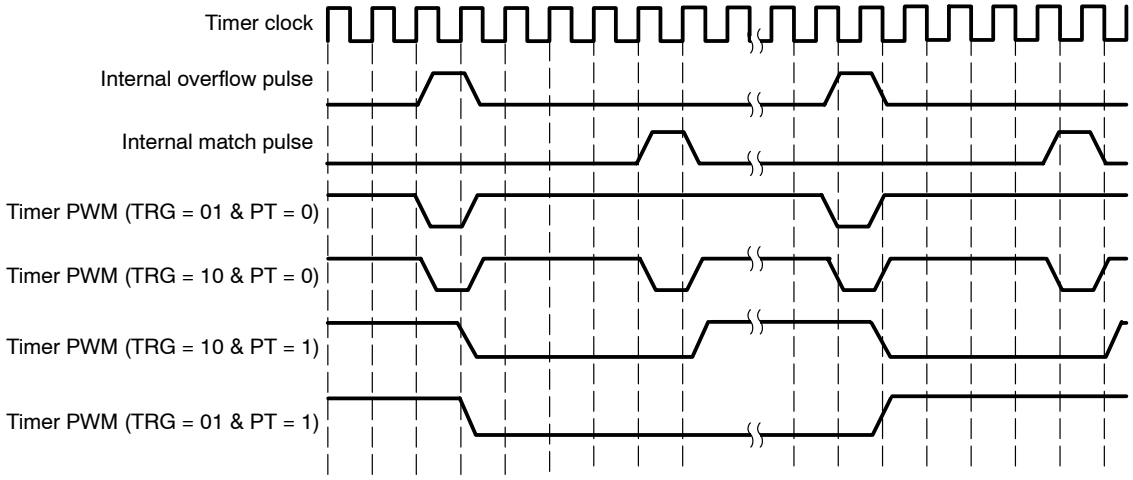
The TLDR and TMAR registers must keep values smaller than the overflow value (0xFFFFFFFF) with at least two units. In case the PWM trigger events are both overflow and match, the difference between the values kept in TMAR register and the value in TLDR must be at least two units. When match event is used, the compare mode TCLR (CE) must be set.

Figure 6. Timing Diagram of Pulse-Width Modulation With SCPWM Bit = 0



In Figure 7 TCLR (SCPWM bit) is set to 1.

Figure 7. Timing Diagram of Pulse-Width Modulation With SCPWM Bit = 1



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 has been issued, the associated interrupt status bit is set in the timer status register (TISR). The pending interrupt event is reset when the set status bit is overwritten by a 1. Reading the interrupt status register and writing the value back allows fast interrupt acknowledge.

4.8 Sleep Mode Request and Acknowledge

Upon a sleep mode request issued by the host processor, the timer module goes into sleep mode depending on the IDLEMODE field of the system configuration register (see TIOCP_CFG).

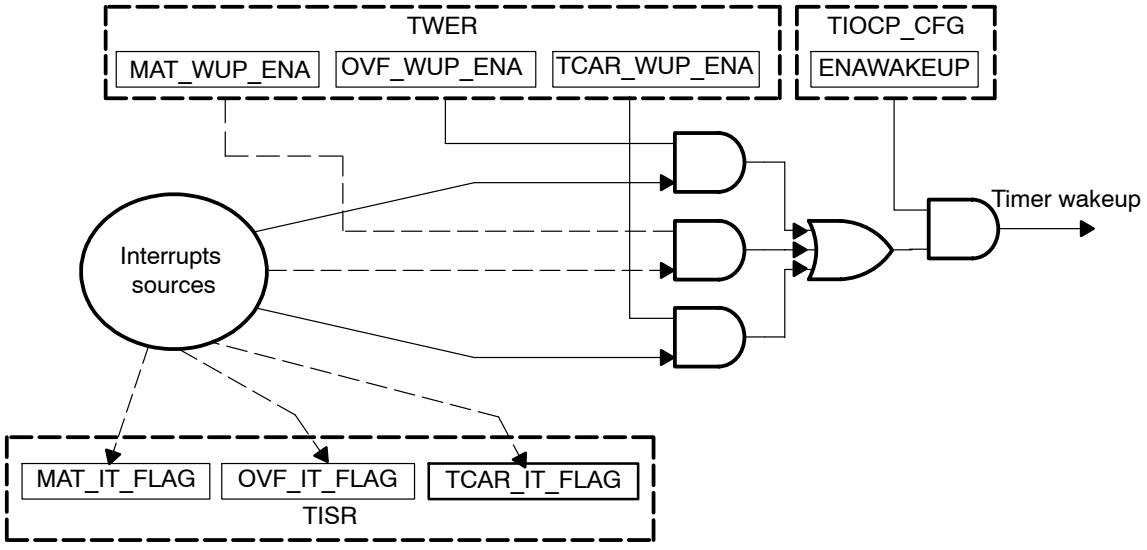
If the IDLEMODE field sets no-idle mode, the timer does not go into sleep mode and the idle acknowledge signal is never asserted.

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 8 shows the wake-up request generation.

Figure 8. Wake-Up Request Generation



4.8.1 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 then can read the corresponding bit in TISR to find out 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 a 1 in the corresponding bit of the TISR register.

4.9 Timer Counting Rate

The dual-mode timer is composed of a prescaler stage and a timer counter. The prescaler stage is clocked with the timer clock and acts as a clock divider for the timer counter stage.

The timer rate is defined by:

- Value of the prescaler fields (PRE and PTV of TCLR register)
- Value loaded into the timer load register (TLDR)

Table 30 lists prescaler clock ratios values.

Table 30. Prescaler Clock Ratios 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

timer rate = (0xFFFF FFFF – TLDR + 1) x timer clock period x clock divider (PS)

With timer clock period = 1/ timer clock frequency and PS = $2^{(PTV + 1)}$.

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 31.

Table 31. 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

4.10 Dual-Mode Timer Under Emulation

During emulation mode, the dual-mode timer continues to run according to the value of the EMUFREE bit of the timer OCP configuration register (TIOCP_CFG).

If EMUFREE is 1, timer execution is not stopped in emulation mode and the interrupt assertion 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 asynchronous input pin is internally synchronized on two timer-clock rising edges.

4.11 Accessing Registers

All registers are 32 bits wide, accessible via OCP interface with 16-bit or 32-bit OCP access (read/write).

4.12 Programming Timer Registers

The host uses the OCP bus protocol to write the TLDR, TCRR, TIER, TISR, TCLR, TIOCP_CFG, TWER, TTGR, TSICR, and TMAR registers synchronously with the timer interface clock.

Because the timer registers are 32-bit wide, 16-bit wide access mode requires two consecutive write operations (16 LSBs followed by 16 MSBs).

4.13 Reading Timer Registers

In 16-bit access mode, reading the LSBs from the timer counter register (TCRR) captures the current timer counter value. This must be followed by reading the 16 MSBs.

4.14 Writing Timer Registers

4.14.1 Software Write Posting Synchronization Mode

This mode is used if TSCR (POSTED bit) is set to 1 in the timer control register.

It uses a posted write scheme for updating any internal register. This means that the write transaction is immediately acknowledged on the OCP 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 CPU that requested the write transaction is stalled. For each register, a status bit is provided that is set if there is a pending write access to the register.

In this mode, it is mandatory that the CPU check this status bit prior to any write access. In case a write is attempted to a register with a previous access pending, the previous access is discarded without notice.

There is one status bit per register, accessible in the timer write-posted status register.

The timer module updates the timer counter register value synchronously with the OCP 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(host peripheral clock)/4

4.14.2 Software Non-Write Posting Synchronization Mode

This mode is used if TSCR (POSTED bit) is set to 0 in the timer control register.

This mode uses a non-posted write scheme for updating any internal register. This means that the write transaction is not acknowledged on the OCP interface until the effective write operation occurs, after the resynchronization in the timer clock domain. The drawback is that both the interconnect and the CPU are stalled during this period.

- The CPU cannot serve interrupts, increasing the interrupt latency.
- An interconnect including time-out logic to detect erroneous transactions can generate an unwanted system abort event.

This stall period can be quantified as follows:

- T (stall) = 3 MPU peripheral clocks + 5 timer clocks

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 OCP interface frequency and the functional clock frequency.

4.14.3 Write Mode Selection

The choice between the synchronization modes must consider the frequency ratio and the stall periods that can be supported by the system without affecting the global performance.

4.15 Dual-Mode Timer Registers

Table 32 lists the 32-bit (or 2 x 16-bit) dual-mode timer registers. Table 33 through Table 46 describe the individual register bits. The registers are accessible in 16-bit mode and use little-endian addressing.

Table 32. Dual-Mode Timer Registers

Base Address = 0xFFFB 1400 (n * 0x800, MPU), E101 1400 (n * 0x800, DSP); n = 0...7				
Name	**Description**	**R/W**	**Offset LSB**	**Offset MSB**
TIDR	Timer identification	R	0x00	0x02
RESERVED	Reserved		0x04	0x06
RESERVED	Reserved		0x08	0x0A
RESERVED	Reserved		0x0C	0x0E
TIOCP_CFG	Timer OCP configuration	R/W	0x10	0x12
TISTAT	Timer system status	R	0x14	0x16
TISR	Timer status	R/W	0x18	0x1A
TIER	Timer interrupt enable	R/W	0x1C	0x1E
TWER	Timer wake-up enable	R/W	0x20	0x22
TCLR	Timer control	R/W	0x24	0x26
TCRR	Timer counter	R/W	0x28	0x2A
TLDR	Timer load	R/W	0x2C	0x2E
TTGR	Timer trigger	R/W	0x30	0x32
TWPS	Timer write pending status	R	0x34	0x36
TMAR	Timer match	R/W	0x38	0x3A
TCAR	Timer capture	R	0x3C	0x3E
TSICR	Timer synchronization interface control	R/W	0x40	0x42

Table 33. Timer Identification Register (TIDR)

Base Address = 0xFFFB 1400 (n * 0x800, MPU), E101 1400 (n * 0x800, DSP); n = 0...7, Offset = 0x00 (LSB), 0x02 (MSB)				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:8	RESERVED			0x000000
7:0	TID_REV	Module HW revision number of the current timer module: value set by hardware. Four LSBs of TID_REV indicate a minor revision. Four MSBs of TID_REV indicate a major revision. A reset has no effect on value returned	R	HW ID revision

Table 34. Timer OCP Configuration Register (TIOCP_CFG)

Base Address = 0xFFFB 1400 (n * 0x800, MPU), E101 1400 (n * 0x800, DSP); n = 0...7, Offset = 0x10 (LSB), 0x12 (MSB)				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:6	RESERVED			0x000 0000
5	EMUFREE	0: The timer is frozen in emulation mode. 1: The timer runs free.	R/W	0
4:3	IdleMode	Power management, request/acknowledge 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 timer (see Section 4.8, *Sleep Mode Request and Acknowledge*). 11: Reserved	R/W	0x0
2	ENAWAKEUP	Wake-up feature control 0: Wake-up is disabled. 1: Wake-up is enabled.	R/W	0

Table 34. Timer OCP Configuration Register (TIOCP_CFG) (Continued)

Base Address = 0xFFFB 1400 (n * 0x800, MPU), E101 1400 (n * 0x800, DSP); n = 0...7, Offset = 0x10 (LSB), 0x12 (MSB)				
Bit	**Name**	**Function**	**R/W**	**Reset**
1	SOFTRESET	Software reset. Set this bit to 1 to trigger a module reset. The bit is automatically reset by hardware. During reads, it always returns 0. 0: Normal mode 1: Reset the OCP and the functional domain	R/W	0
0	AUTOIDLE	Interface clocks gating strategy 0: Interface clock is free-running. 1: Automatic interface gating strategy is applied, based on the interface activity.	R/W	0

This register allows controlling various parameters of the OCP interface.

Table 35. Timer System Status Register (TISTAT)

Base Address = 0xFFFB 1400 (n * 0x800, MPU), E101 1400 (n * 0x800, DSP); n = 0...7, Offset = 0x14 (LSB), 0x16 (MSB)				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:1	RESERVED		R	0x000 0000
0	RESETDONE	Internal global reset monitoring: 0: Reset is ongoing 1: Reset completed	R	1

This register monitors the internal global reset status. This status bit is set to one when all clock domains have been reset. This status can be monitored by the software to check if the module is ready-to-use following a reset (either hardware or software reset).

Table 36. Timer Status Register (TISR)

Base Address = 0xFFFB 1400 (n * 0x800, MPU), E101 1400 (n * 0x800, DSP); n = 0...7, Offset = 0x18 (LSB), 0x1A (MSB)				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:3	RESERVED		R	0x000 0000
2	TCAR_IT_FLAG	0: No capture interrupt request 1: Capture interrupt request	R/W	0
1	OVF_IT_FLAG	0: No overflow interrupt request 1: Overflow interrupt pending	R/W	0
0	MAT_IT_FLAG	0: No compare interrupt request 1: Compare interrupt pending	R/W	0

The timer status register is used to determine which of the timer events requested an interrupt. Bit 0 corresponds to the compare result of TCRR and TMAR, and is set when the compare register matches the counter value. Bit 1 corresponds to the TCRR overflow; and bit 2 indicates that an external pulse transition of the correct polarity is detected on the external event capture pin. If the value is 1, then that timer event is requesting the interrupt. If the user wants to reset the status bit, then a 1 must be written to the appropriate bit. Writing a 1 to the bit TCAR_IT_FLAG resets the edge detection logic. However, the user cannot generate an interrupt by writing a 1 to the timer status register bits. If the user writes a 0 to a bit in the timer status register bits, the value remains unchanged.

Table 37. Timer Interrupt Enable Register (TIER)

Base Address = 0xFFFB 1400 (n * 0x800, MPU), E101 1400 (n * 0x800, DSP); n = 0...7, Offset = 0x1C (LSB), 0x1E (MSB)				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:3	RESERVED		R	0x000 0000
2	TCAR_IT_ENA	Capture interrupt enable 0: Interrupt disabled 1: Interrupt enabled	R/W	0
1	OVF_IT_ENA	Overflow interrupt enable 0: Interrupt disabled 1: Interrupt enabled	R/W	0
0	MAT_IT_ENA	Match interrupt enable 0: Interrupt disabled 1: Interrupt enabled	R/W	0

The timer interrupt enable register allows the user to enable certain timer events for generating an interrupt request. Bit 0 determines whether or not the compare register flag MAT_IT_FLAG can generate an interrupt. Bit 1 determines whether or not the overflow counter flag OVF_IT_FLAG can generate an interrupt. Bit 2 determines whether the edge detection flag TCAR_IT_FLAG can generate an interrupt.

Table 38. Timer Wake-Up Enable Register (TWER)

Base Address = 0xFFFB 1400 (n * 0x800, MPU), E101 1400 (n * 0x800, DSP); n = 0...7, Offset = 0x20 (LSB), 0x22 (MSB)				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:3	RESERVED		R/W	0x0000 0000
2	TCAR_WUP_ENA	0: Wake-up generation is disabled. 1: Wake-up generation is enabled.	R/W	0
1	OVF_WUP_ENA	0: Wake-up generation is disabled. 1: Wake-up generation is enabled.	R/W	0
0	MAT_WUP_ENA	0: Wake-up generation is disabled. 1: Wake-up generation is enabled.	R/W	0

The timer wake-up enable register (TWER) allows the user to mask the expected source of a wake-up event that generates a wake-up request. The TWER is programmed synchronously with the interface clock before any idle mode request coming from the host processor.

Table 39. Timer Control Register (TCLR)

Base Address = 0xFFFB 1400 (n * 0x800, MPU), E101 1400 (n * 0x800, DSP); n = 0...7, Offset = 0x24 (LSB), 0x26 (MSB)				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:13	RESERVED			0x00000
12	PT	Pulse or toggle mode on timer PWM output pin 0: Pulse 1: Toggle	R	0
11:10	TRG	Trigger output mode on timer PWM output pin 00: No trigger 01: Trigger on overflow 10: Trigger on overflow and match 11: Reserved	R/W	0x0
9:8	TCM	Transition capture mode on event capture input pin 00: No capture 01: Capture on low to high transition 10: Capture on high to low transition 11: Capture on both edge transition	R/W	0x0
7	SCPWM	This bit must be set or cleared while the timer is stopped or the trigger is off. 1: Set the timer PWM output pin and select negative pulse for pulse mode. 0: Clear the timer PWM output pin and select positive pulse for pulse mode.	R/W	0
6	CE	1: Compare mode is enabled. 0: Compare mode is disabled.	R/W	0
5	PRE	Prescaler enable 0: The timer clock input pin clocks the counter. 1: The divided input pin clocks the counter.	R/W	0
4:2	PTV	Prescale clock timer value	R/W	0x0

Table 39. Timer Control Register (TCLR) (Continued)

Base Address = 0xFFFB 1400 (n * 0x800, MPU), E101 1400 (n * 0x800, DSP); n = 0...7, Offset = 0x24 (LSB), 0x26 (MSB)				
Bit	**Name**	**Function**	**R/W**	**Reset**
1	AR	1: Autoreload timer 0: One-shot timer	R/W	0
0	ST	1: Start timer 0: Stop timer: Only the counter is frozen In case of one-shot mode selected (AR = 0), this bit is automatically reset by internal logic when the counter overflows.	R/W	0

Table 40. Timer Counter Register (TCRR)

Base Address = 0xFFFB 1400 (n * 0x800, MPU), E101 1400 (n * 0x800, DSP); n = 0...7, Offset = 0x28 (LSB), 0x2A (MSB)				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:0	TIME_COUNTER	Value of timer counter	R/W	0x000 0000

The TCRR register is a 32-bit register (16-bit addressable). Therefore, the MPU can perform one 32-bit access or two 16-bit accesses to the register while the DSP performs two consecutive 16-bit transactions. Because the timer interface clock is completely asynchronous with the timer clock, some synchronization is done to ensure that the TCRR value is not read while it is being incremented.

In 16-bit mode, the following sequence must be followed to read the TCRR register properly:

1) Read the lower 16-bits of the TCRR register (offset = 0x28). When the TCRR is read, the lower 16-bit LSB are read, and the upper 16-bits of the TCRR MSB register are stored in a temporary register.

2) Read the upper 16 bits of the TCRR register (offset = 0x2A). During this read, the value of the upper 16-bit MSB that has been stored in the temporary register is read.

Therefore, to read the value of TCRR correctly, the first OCP read access must be to the lower 16-bits (that is, offset = 0x28), followed by OCP read access to the upper 16-bits (that is, offset= 0x2A).

Table 41. Timer Load Register (TLDR)

Base Address = 0xFFFB 1400 (n * 0x800, MPU), E101 1400 (n * 0x800, DSP); n = 0...7, Offset = 0x2C (LSB), 0x2E (MSB)				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:0	TIME_VALUE	Timer counter value loaded on overflow in autoreload mode or on TTGR write access	R/W	0x0000 0000

Table 42. Timer Trigger Register (TTGR)

Base Address = 0xFFFB 1400 (n * 0x800, MPU), E101 1400 (n * 0x800, DSP); n = 0...7, Offset = 0x30 (LSB), 0x32 (MSB)				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:0	TTGR_VALUE	Writing in the TTGR register, TCRR is loaded from TLDR and prescaler counter is cleared. Reload is done regardless the AR field value of TCLR register.	R/W	0xFFFF FFFF

The read value of this register is always 0xFFFF FFFF.

Table 43. Timer Write Pending Status Register (TWPS)

Base Address = 0xFFFB 1400 (n * 0x800, MPU), E101 1400 (n * 0x800, DSP); n = 0...7, Offset = 0x34 (LSB), 0x36 (MSB)				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:5	RESERVED		R	0x000 0000
4	W_PEND_TMAR	When equal to one, a write is pending to the TMAR register.	R	0
3	W_PEND_TTGR	When equal to one, a write is pending to the TTGR register.	R	0
2	W_PEND_TLDR	When equal to one, a write is pending to the TLDR register.	R	0
1	W_PEND_TCRR	When equal to one, a write is pending to the TCRR register.	R	0
0	W_PEND_TCLR	When equal to one, a write is pending to the TCLR register.	R	0

In posting mode, the software must read the pending write status bits (timer write posted status register bits [4:0]) to ensure that following write access is not discarded because of an ongoing write synchronization process. These bits are automatically cleared by internal logic when the write to the corresponding register is acknowledged. The TISR and TIER registers interface only with the timer interface clock, so they do not need a write pending status bit.

Table 44. Timer Match Register (TMAR)

Base Address = 0xFFFB 1400 (n * 0x800, MPU), E101 1400 (n * 0x800, DSP); n = 0...7, Offset = 0x38 (LSB), 0x3A (MSB)				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:0	COMPARE VALUE	Value to be compared to the timer counter	R/W	0x000 0000

The compare logic consists of a 32-bit wide, read/write data TMAR register and logic to compare the counter current value with the value stored in the TMAR register.

Table 45. Timer Capture Register (TCAR)

Base Address = 0xFFFB 1400 (n * 0x800, MPU), E101 1400 (n * 0x800, DSP); n = 0...7, Offset = 0x3C (LSB), 0x3E (MSB)				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:0	CAPTURED VALUE	Timer counter value captured on an external event trigger	R	0x0000 0000

When the appropriate transition (rising, falling, or both) is detected in the edge detection logic, the current counter value is stored to the TCAR register.

Table 46. Timer Synchronization Interface Control Register (TSICR)

Base Address = 0xFFFB 1400 (n * 0x800, MPU), E101 1400 (n * 0x800, DSP); n = 0...7, Offset = 0x40 (LSB), 0x42 (MSB)				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:2	RESERVED	Reserved	R	0x0000 0000
2	POSTED	1: Posted mode active (clocks ratio needs to fit the MPU peripheral clock > 4 timer clock frequency requirement) 0: Posted mode inactive	R/W	1
1	SFT	This bit resets the all of the functional parts of the module. During reads, it always returns 0. 1: Software reset is enabled. 0: Software reset is disabled.	R/W	0
0	RESERVED	Reserved	R	0

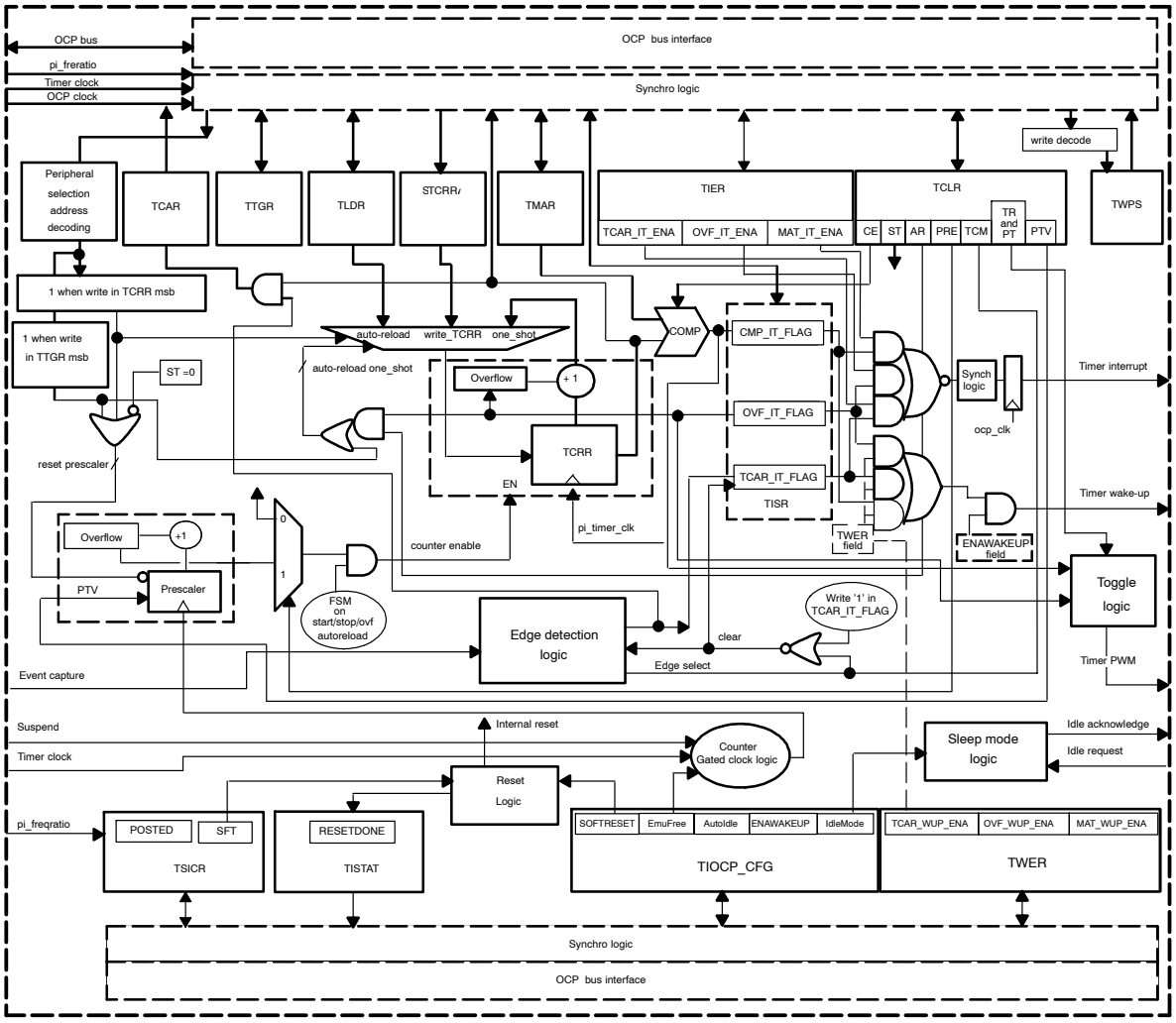
The timer clock has different sources. Based on the timer clock source selection, the user can override the reset value for POSTED to perform nonposted read/write accesses.

Typically, if the timer clock source is 20 MHz and the timer interface clock is 50 MHz, it is recommended to clear the POSTED bit.

4.16 Implementation

Figure 9 is an overview of the implementation and interaction of the registers according to the functional description and the input/output pins.

Figure 9. Dual-Mode Timer Design Implementation Overview



5 32-kHz Synchronized Timer

5.1 Functional Description

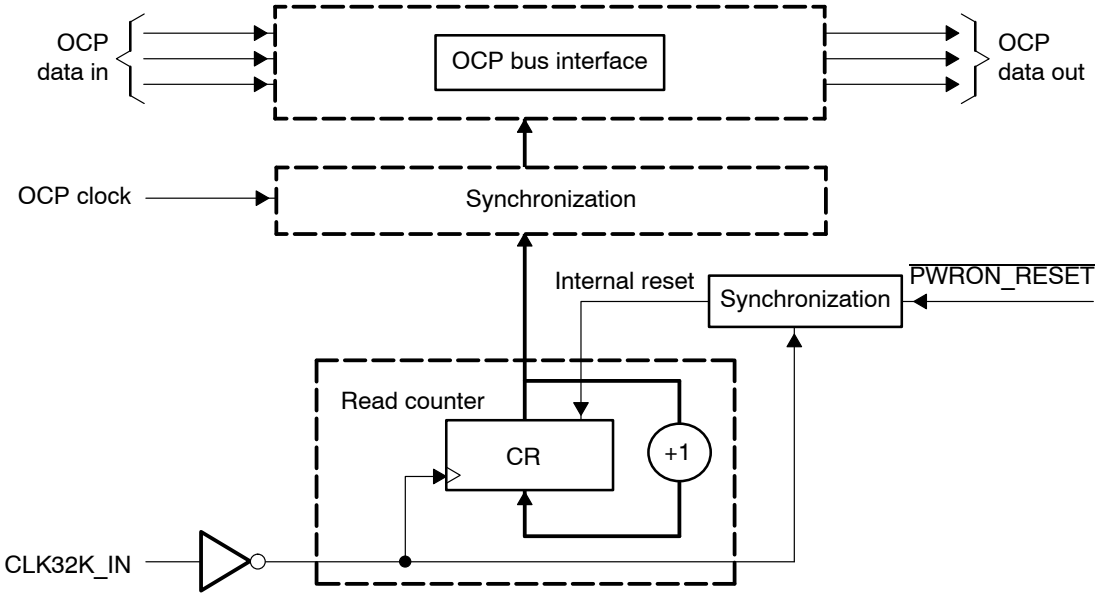
The 32-kHz synchronization counter is a 32-bit counter clocked by the falling edge of the 32-kHz clock.

It is reset with external asynchronous power-up reset ($\overline{\text{PWRON\_RESET}}$).

When $\overline{\text{PWRON\_RESET}}$ 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 clock.

When the highest value is reached, it wraps back to zero and starts running again.

Figure 10. 32-kHz SYNCH Timer



Synchronization ensures the read transaction correctness by synchronizing the counter register read access on the OCP clock signal. The $\overline{\text{PWRON\_RESET}}$ input pin resets the counter register (CR) and the inverted CLK32K_IN clocks it.

5.1.1 Reading the Timer

The counter register is 32 bits wide and a 16-bit capture is done on the 16-bit LSB first to allow LSB16 + MSB16 capture.

Internal synchronization logic allows the counter value to be read while the counter is running. The time latency to read a synchronized register is one OCP clock period.

5.2 32-kHz Synchronization Timer Registers

Table 47 lists the 32-bit, 32-kHz synchronization registers, all of which are accessible in 16-bit mode and use little-endian addressing. The address of a register is the start address plus the offset address. Table 48 through Table 49 describe the register bits.

Table 47. 32-kHz Synchronization Timer Registers

Base Address = 0xFFFB C400 (MPU), 0xE101 C400 (DSP)				
Name	**Description**	**R/W**	**Offset LSB**	**Offset MSB**
32KSYNCNT_REV	Revision identification	R	0x00	0x02
RESERVED	Reserved	R	0x04	0x06
RESERVED	Reserved	R	0x08	0x0A
RESERVED	Reserved	R	0x0C	0x0E
CR	Read counter	R	0x10	0x12

Table 48. Identification Register (32KSYNCNT_REV)

Base Address = 0xFFFB C400 (MPU), 0xE101 C400 (DSP)Offset = 0x00 (LSB), 0x02 (MSB)				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:8	RESERVED	Reads return 0		0x000000
7:0	CID_REV	Module HW revision number of the current timer module: value set by hardware. Four LSBs of TID_REV indicate a minor revision. Four MSBs of TID_REV indicate a major revision.	R	HW ID revision

Table 49. Read Counter Register (CR)

Base Address = 0xFFFB C400 (MPU), 0xE101 C400 (DSP)Offset = 0x10 (LSB), 0x12 (MSB)				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	COUNTER_LO	Value of 32-kHz SYNCH counter (16-bit LSB) Same as CR[15:0]	R	0x0003

The CR register is a read-only 32-bit register. Therefore, the MPU performs a 32-bit access on the register while the DSP has to perform two consecutive 16-bit transactions.

In 16-bit mode, the following sequence must be followed to read the CR register properly:

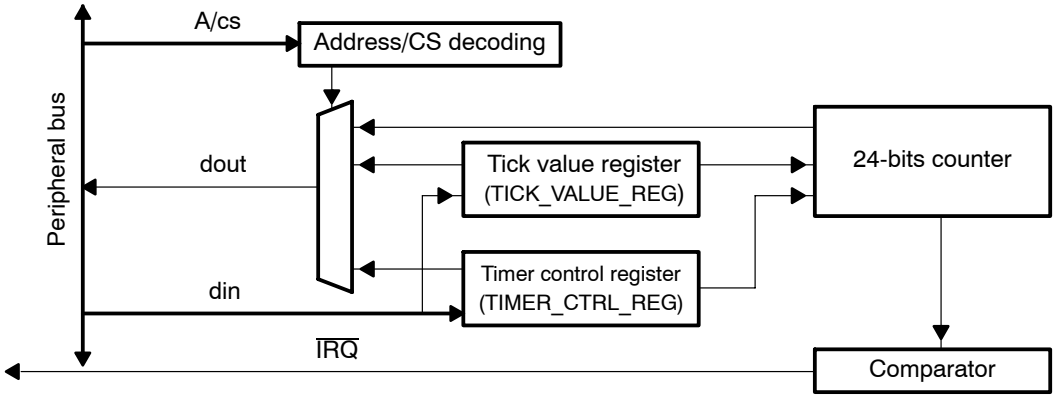
1) Read the lower 16 bits of the CR register (offset = 10).
2) When the CR is read and synchronized, the lower 16-bit LSB are read, and the upper 16-bits of the CR MSB register are stored in a temporary register.
3) Read the upper 16 bits of the CR register (offset = 12).
4) During this read, the value of the upper 16 bits that have been temporarily stored is read.

Therefore, to read the value of CR correctly, the first read access must be to the lower 16 bits (that is, offset = 10), followed by read access to the upper 16 bits (that is, offset = 12).

6 Operating System Timer

The operating system (OS) timer is a 24-bit timer with the 32-kHz clock. Figure 11 shows the operating system timer.

Figure 11. OS Timer



6.1 Countdown Operation

Canonical operation of the OS timer works as follows:

1) The host MPU writes the counter initial value to the tick value register (TICK_VALUE_REG), where, by default, all bits are set to 1.

2) The host MPU sets the timer reload bit (TRB) inside the timer control register (TIMER_CTRL_REG).

3) MPU polls the TRB bit. When it is cleared, the timer internal counter has been loaded with the the tick value register (TICK_VALUE_REG) value.

4) MPU then launches the timer by setting the timer start stop bit (TSS) inside the timer control register (TIMER_CTRL_REG). The timer starts to count down to zero and generates a negative edge-sensitive interrupt (low level 15-µs-wide pulse) to the interrupt controller.

5) On the next cycle, the counter is reloaded from the tick value register (TICK_VALUE_REG) and then starts to down count again, unless the ARL bit of the timer control register (TIMER_CTRL_REG) is low: one-shot mode.

6.2 Overriding Normal Counting

Normal operation can be overridden by using two bits in the timer control register (TIMER_CTRL_REG):

- The timer reload bit (TRB), causes the counter to be reloaded on the next 32-kHz clock cycle (whether or not the timer is counting).
- The timer start stop bit (TSS) causes the counter to be stopped on the next 32-kHz clock cycle. Then, the content of the counter is frozen.

6.3 Loading/Autorestart

Loading the counter in the timer can be done in two ways :

- Writing a 1 to the TRB bit in the timer control register (TIMER_CTRL_REG).
- Waiting until the counter reaches zero. It then is reloaded from the tick value register (TICK_VALUE_REG) if the autorestart bit (ARL) in the control register is set to 1. If not, the timer is stopped.

6.4 Timer Interrupt Period

The timer interrupt period is defined by the value loaded into the tick value register (TICK_VALUE_REG).

Setting the counter to value 0 is not allowed and leads to unpredictable results.

The timer interrupt rate is as follows:

$$\text{IRQ rate} = \frac{u}{32768},$$

where *u* is the sum of the TVR contents plus one.

Table 50. Timer Interrupt Period According to TVR Value

TICK_VALUE_REG	Interrupt Period
0x00000001	61 µs
0x0000028F	19.9 ms
0x000007FF	62.5 ms
0x0000FFFF	2 s
0x00FFFFFF	512 s

6.5 Peripheral Alignment and Data Width

The TIPB interface data path is 32 bits wide. To keep software compatible with earlier 16-bit timer versions, the access width is taken into account when writing to the registers. This means that writing 8 bits (or 16 bit) sets 24 MSB (or 16 MSB) to 0.

For read accesses, all 32 bits are always output.

All unused register bits remain at 0.

6.6 OS Timer Registers

Table 51 lists the OS timer registers. Table 52 through Table 55 describe the register bits. Synchronization of reads and writes to the 32-kHz clock is performed in a different way for each register, which leads to slight restrictions concerning register access (see Table 52).

Table 51. Operating System Registers

Base Address = 0xFFFB 9000			
Name	**Description**	**R/W**	**Offset**
TICK_VALUE_REG	Tick value	R/W	0x00
TICK_COUNTER_REG	Tick counter	R	0x04
TIMER_CTRL_REG	Timer control	R/W	0x08

Table 52. Timer Registers Access Timing Constraints

Register Name	Read	Write
TIMER_CTRL_REG	Can be read anytime. The value read is the last value written.	Two consecutive writes must be separated by at least 1 CLK32 period (31.25 μs). If this is not the case, the value written is not assured.
TICK_COUNTER_REG	Reads are resynchronized on a high-speed clock to the prevent peripheral bus from timing out. Can be read anytime, providing high-speed clock is running. If not, the value is not assured.	Writing to this register has no effect.
TICK_VALUE_REG	Can be read anytime. The value read is the last value written.	Two consecutive writes must be separated by at least 1 CLK32 period (31.25 μs). If this is not the case, the value written is not assured.

Table 53. Tick Value Register (TICK_VALUE_REG)

Base Address = 0xFFFB 9000, Offset = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:24	RESERVED	Reserved	R	0x00
23:0	TICK_VALUE_REG	Value is loaded when the timer reaches 0 or when it starts.	R/W	0xFFFFFF

Writes to this register must be separated by at least one 32-kHz clock period.

Setting TICK_VALUE_REG to 0, and then loading the counter with this value leads to unpredictable results.

Table 54. Tick Counter Register (TICK_COUNTER_REG)

Base Address = 0xFFFB 9000, Offset = 0x04				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:24	RESERVED	Reserved	R	0x0
23:0	TICK_COUNTER_REG	Value of timer	R	0xFFFFFF

When reading this register, the high-speed clock must be running. If not, the value read is not ensured.

Table 55. Timer Control Register (TIMER_CTRL_REG)

Base Address = 0xFFFB 9000, Offset = 0x08				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:4	RESERVED	Reserved	R	0x0000 000
3	ARL	Autoreload/start: 1: Autorestart mode 0: One-shot mode: when counter reaches zero, an interrupt is generated and timer is stopped	R/W	1
2	IT_ENA	Interrupt enable: 0: Interrupt disabled 1: Interrupt enabled	R/W	0
1	TRB	Timer reload bit: Setting bit to 1 reloads the counter at next 32 kHz-clock rising edge. Once the counter is reloaded, the bit is reset.	R/W	0
0	TSS	Timer start/stop: 0: Stop timer 1: Start timer If timer is in one-shot mode (ARL = 0), the bit is automatically reset when the timer = 0.	R/W	0

Writes to this register must be separated by at least one 32-kHz clock period.

This page is intentionally left blank.

Index

N

O

P

R

S

T

W

OMAP5912 Multimedia Processor Serial Interfaces Reference Guide

Literature Number: SPRU760B
October 2004

OMAP™
TEXAS INSTRUMENTS TECHNOLOGY

TEXAS INSTRUMENTS

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.

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
		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 document describes the serial interfaces of the OMAP5912 multimedia processor.

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.

Related Documentation From Texas Instruments

Documentation that describes the OMAP5912 device, related peripherals, and other technical collateral, is available in the OMAP5912 Product Folder on TI's website: www.ti.com/omap5912.

Trademarks

OMAP and the OMAP symbol are trademarks of Texas Instruments.

Contents

Figures

Tables

This page is intentionally left blank.

Serial Interfaces

This document describes the serial interfaces of the OMAP5912 multimedia processor.

1 SPI Master/Slave

1.1 Functional Description

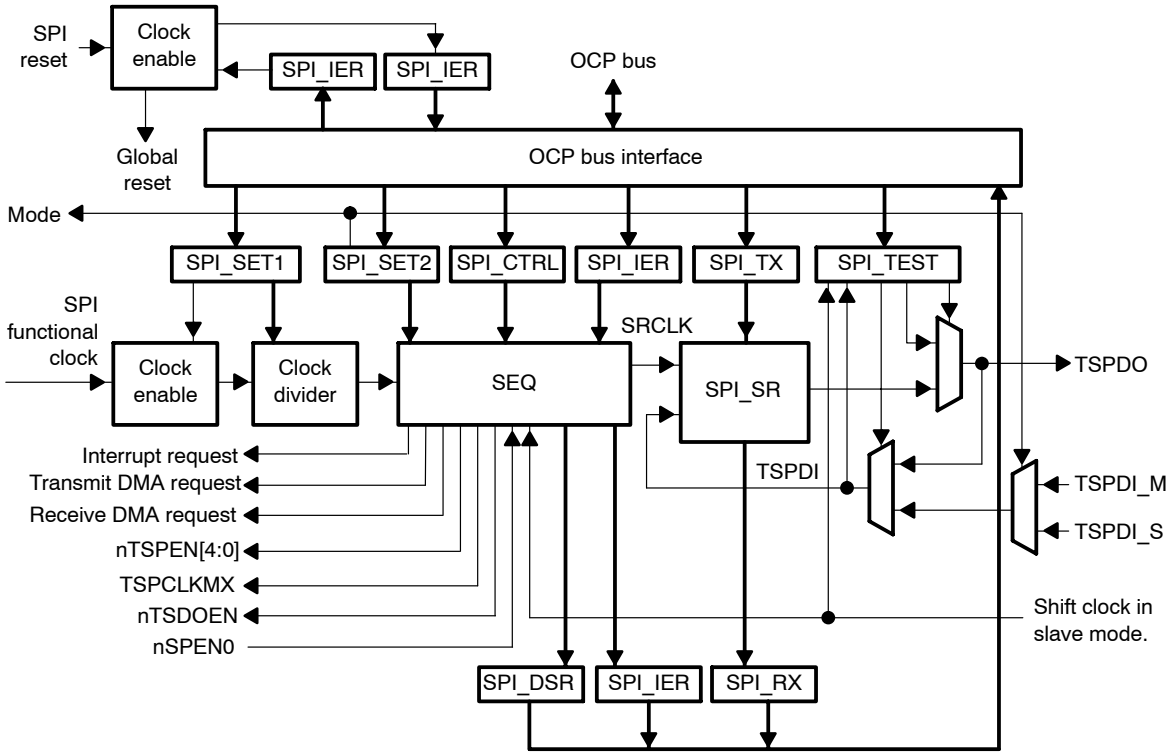
The serial interface is a bidirectional, four-line interface dedicated to the transfer of data to and from external devices offering a four-line serial interface. The four-line interface consists of:

- The clock used to shift data in and out
- The device enable
- The data input
- The data output

This serial port is based on a looped shift register, thus allowing both transmit (PISO) and receive (SIPO) modes. It can operate in master or in slave mode using MCU-DSP or DMA protocol. It supports up to five serial devices. The serial data is sent MSB first.

The serial port is fully controlled by the OCP bus (data write, data read, and activation of serialization operations).

Figure 1. SPI Master/Slave Block Diagram



1.2 Interface

Table 1 describes the SPI interface.

Table 1. SPI Interface

SPI Integration		
SPIF.SCK	Shift register clock Output in master mode Input in slave mode	I/O
SPIF.DIN	Serial data Input in master mode Serial data output in slave mode	I/O
SPIF.DOUT	Serial data output in master mode Serial data Input in slave mode Note: this signal is always driven by OMAP5912 regardless of master or slave mode.	I/O
$\overline{\text{SPIF.CS[0]}}$	SPI chip-select 0 Configured in output in master mode Configured in input in slave mode	I/O
$\overline{\text{SPIF.CS[3:1]}}$	External SPI chip-selects in master mode	O

1.3 SPI Registers

Start address: FFFB 0C00

Address of one register: Start address + offset address

Access supported: 16 or 32 bits

SPI uses little-endian addressing scheme.

The SPI offers input and output registers for loading data to serialize (transmit) and reading received data (receive).

Table 2. SPI Registers

Base Address = 0xFFFB 0C00				
Mnemonic	**Register Name**	**Size (Bits)**	**Access**	**Offset**
SPI_REV	Identification register	32	R	0x000
	Reserved			0x004
	Reserved			0x008
	Reserved			0x00C
SPI_SCR	System configuration register	32	R/W	0x010
SPI_SSR	System status register	32	R	0x014
SPI_ISR	Interrupt status register	32	R/W	0x018
SPI_IER	Interrupt enable register	32	R/W	0x01C
	Reserved			0x020
SPI_SET1	Set up 1 register	32	R/W	0x024
SPI_SET2	Set up 2 register	32	R/W	0x028
SPI_CTRL	Control register	32	R/W	0x02C
SPI_DSR	Data status register	32	R	0x030
SPI_TX	Transmit register	32	R/W	0x034
SPI_RX	Receive register	32	R	0x038
SPI_TEST	Test register	32	R/W	0x03C

Table 3. Identification Register Bit Description (SPI_REV—0x000)

Base Address = 0xFFFB 0C00, Offset = 0x00			
Bit	**Name**	**Function**	**Access**
31:8	Reserved	A read access returns 0.	R
7:0	REV	Revision number The 4-bit LSB indicates a minor revision. The 4-bit MSB indicates a major revision. Ex: 0x10 → version 1.0.	R

A write to this register has no effect.

A reset has no effect on the value returned.

Table 4. System Configuration Register Bit Description (SPI_SCR—0x010)

Base Address = 0xFFFB 0C00, Offset = 0x10				
Bit	**Name**	**Function**	**Access**	**Reset**
31:5	Reserved	A read access returns 0.	R	0x0000000
4:3	IDLEMODE	Power management, req/ack control 00: Force-idle. An idle request is acknowledged unconditionally. 01: No idle. An idle request is never acknowledged. 10: Smart idle. An idle request is acknowledged based on the internal activity of the module. 11: Reserved: Do not use.	R/W	00
2	EN AWAKEUP	Wake-up feature control 0: Wake-up is disabled. 1: Wake-up capability is enabled.	R/W	0
1	SOFTRESET	Software reset. Set this bit to 1 to trigger an SPI reset. This bit is automatically reset by hardware. Writing a 0 has no effect. During reads, it always returns 0. 0: Normal mode. 1: The module is reset.	R/W	0
0	AUTOIDLE	Internal OCP clock gating strategy 0: OCP clock is free-running. 1: Automatic OCP clock gating strategy is applied, based on the OCP interface activity.	R/W	0

This register allows control of the OCP interface parameters.

Table 5. System Status Register Bit Description (SPI_SSR—0x014)

This register provides status information about the reset.

Base Address = 0xFFFB 0C00, Offset = 0x14				
Bit	**Name**	**Function**	**Access**	**Reset**
31:1	Reserved	A read access returns 0.	R	0x00000000
0	RESETDONE	Internal reset monitoring 0: Internal module reset is ongoing. 1: Reset completed.	R	0

Note: Before accessing or using the module the local host must ensure that internal reset is released by reading the system status register (SPI_SSR).

Table 6. Interrupt Status Register Bit Description (SPI_ISR—0x018)

Base Address = 0xFFFB 0C00, Offset = 0x18				
Bit	**Name**	**Function**	**Access**	**Reset**
31:5	Reserved	A read access returns 0.	R	0x0000000
4	WAKEUP	Wake-up: Active high.	R/W	0
3	TX_UNDERFLOW	Transmit underflow: Active high.	R/W	0
2	RX_OVERFLOW	Receive overflow: Active high.	R/W	0
1	WE	Write end: Active high. Serialization complete.	R/W	0
0	RE	Read end: Active high. Receive register updated.	R/W	0

The interrupt status register is used to qualify the interrupt. Writing a 1 to the corresponding status bit releases the interrupt. Writing a 0 has no effect.

Table 7. Interrupt Enable Register Bit Description (SPI_IER—0x01C)

Base Address = 0xFFFB 0C00, Offset = 0x1C				
Bit	**Name**	**Function**	**Access**	**Reset**
31:5	Reserved	A read access returns 0.	R	0x0000000
4	MSK4	Enable interrupt when wake up 0: Interrupt disabled. 1: Interrupt active.	R/W	0
3	MSK3	Enable interrupt when TX underflow 0: Interrupt disabled. 1: Interrupt active.	R/W	0
2	MSK2	Enable interrupt when RX overflow 0: Interrupt disabled. 1: Interrupt active.	R/W	0
1	MSK1	Enable interrupt for write cycle 0: Interrupt disabled. 1: Interrupt active.	R/W	0
0	MSK0	Enable interrupt for read cycle 0: Interrupt disabled. 1: Interrupt active.	R/W	0

Table 8. Set Up SPI 1 Register Bit Description (SPI_SET1—0x024)

Base Address = 0xFFFB 0C00, Offset = 0x24				
Bit	**Name**	**Function**	**Access**	**Reset**
31:6	Reserved	A read access returns 0.	R	0x0000000
5	DMA_EN	Defines the protocol 0: MCU-DSP protocol. 1: DMA protocol.	R/W	0
4:1	PTV	The prescale time value (2^{PTV}) generates the shift register clock in master mode. (TSPCLKMX). The following scale values are supported: 0000: 1 0001: 2 0010: 4 0011: 8 0100: 16 0101: 32 0110: 64 0111: 128 1000: 256 1001: 512 1010: 1024 1011: 2048 Others: 4096	R/W	0000
0	EN_CLK	SPI functional clock enable 0: Clock is shut off. 1: Clock is running.	R/W	0

Note: A write access to this register during a transaction does not affect the register and activates an OCP error.

SPI_SET1 is dedicated to the configuration of the serial port interface.

SPI_SET2 is dedicated to the configuration of the serial port interface.

Table 9. Set Up SPI 2 Register Bit Description (SPI_SET2—0x028)

Base Address = 0xFFFB 0C00, Offset = 0x28				
Bit	**Name**	**Function**	**Access**	**Reset**
31:16	Reserved	A read access returns 0.	R	0x0000
15	MODE	0: Slave mode 1: Master mode	R/W	0
14:10	CP	Clock phase The shift register clock begins toggling at: 0: The middle of the data transfer 1: The beginning of the data transfer Bit 10 qualifies the access on device 0. Bit 11 qualifies the access on device 1. Bit 12 qualifies the access on device 2. Bit 13 qualifies the access on device 3. Bit 14 qualifies the access on device 4.	R/W	00000
9:5	CE	Shift register enable Active level of the shift register enable 0: The active level is low. 1: The active level is high. Bit 5 qualifies the access on device 0. Bit 6 qualifies the access on device 1. Bit 7 qualifies the access on device 2. Bit 8 qualifies the access on device 3. Bit 9 qualifies the access on device 4.	R/W	00000
4:0	CI	Clock invert Inactive edge of the shift register clock 0: The inactive state of the clock is low. 1: The inactive edge of the clock is high. Bit 0 qualifies the access on device 0. Bit 1 qualifies the access on device 1. Bit 2 qualifies the access on device 2. Bit 3 qualifies the access on device 3. Bit 4 qualifies the access on device 4.	R/W	00000

Notes:
1) In slave mode, the end of a transaction is detected based on CE field configuration. If CEi = 0, the end of a transaction is detected on the rising edge of nSPEN0. If CEi = 1, the end of a transaction is detected on the falling edge of nSPEN0. For more information on the different configurations, see Section 14.1.4.4.
2) In slave mode, all CIi bits must have the same value. This value depends on the inactive edge of the master clock.
3) In slave mode, all CEi bits must have the same value. This value depends on the active level of the master enable.
4) In slave mode, all CPi bits must have the same value. This value depends on the master clock phase.
5) A write access to this register during a transaction does not affect the register and activates an OCP error. The delay between a write in the SET2 register and the CI/CE bits taking effect is 2 x ARMXOR_CK cycles.

Table 10. Control SPI Register Bit Description (SPI_CTRL—0x02C)

Base Address = 0xFFFB 0C00, Offset = 0x2C				
Bit	**Name**	**Function**	**Access**	**Reset**
31:10	Reserved	A read access returns 0.	R	0x000000
9:7	AD	Index of the addressed device 000: Enable device 0 001: Enable device 1 010: Enable device 2 011: Enable device 3 100: Enable device 4 Others: Undefined	R/W	000
6:2	NB	Number of bits to receive/transmit 00000: 1 bit receive/transmit 11111: 32 bits receive/transmit	R/W	00000
1	WR	Write process activation– active high	R/W	0
0	RD	Read and write process activation– active high	R/W	0

Note: A write access to this register during a transaction does not affect the register and activates an OCP error.

This register is dedicated to the activation of the serial port interface. It defines:

- ❏ Read activation of the serial port
- ❏ Write activation of the serial port
- ❏ Number of bits to transfer (in the range 1 to 32)
- ❏ External device address (up to 5)

Table 11. Shift Status Register Bit Description (SPI_DSR—0x030)

Base Address = 0xFFFB 0C00, Offset = 0x30				
Bit	**Name**	**Function**	**Access**	**Reset**
31:2	Reserved	A read access returns 0.	R	0x0000000
1	TX_EMPTY	Shift register is empty: Active high This bit is cleared when the transmit register (SPI_TX) has been written (in functional or emulation mode).	R	1
0	RX_FULL	Receive register is full: Active high This bit is cleared when the receive register (SPI_RX) has been read (not cleared from debugger read).	R	0

The data to transmit are loaded to the SPI_TX register.

Table 12. Transmit Register Bit Description (SPI_TX—0x034)

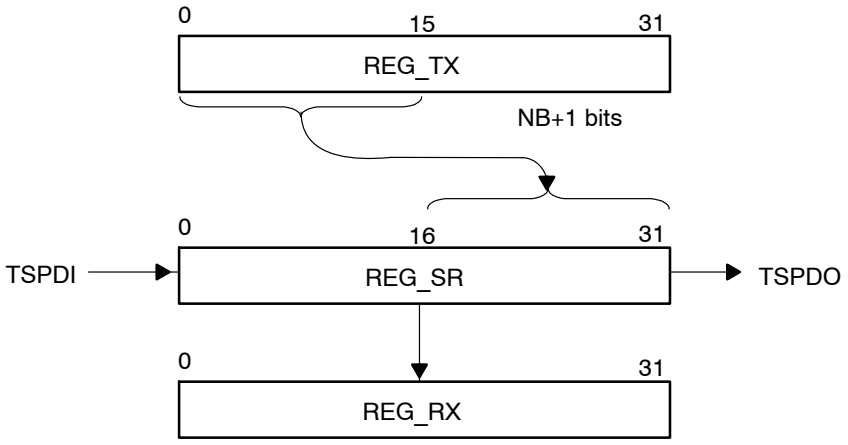
Base Address = 0xFFFB 0C00, Offset = 0x34				
Bit	**Name**	**Function**	**Access**	**Reset**
31:0	SPI_TX	Data to transmit	R/W	0x00000000

Notes: 1) The bits to transmit have to be aligned on the LSB of the SPI_TX register.

2) If the number of bits to transmit is 8, SPI_TX [7] is transmitted first, then SPI_TX [6], and so on.

3) If the number of bits to transmit is 32, SPI_TX [31] is transmitted first, then SPI_TX [30], and so on.

Figure 2. Transmission Example



Note: The number of bits of the word to be transmitted is programmed through NB bit field in the SPI_CTRL register. The transmit register (SPI_TX) data loading must be completed according to the transmitted word bit length and in the proper sequence register access. SPI_TX register is considered as updated (transmit register full with new data) when the most-significant byte part of the transmitted word has been written. If the number of bits of the transmitted word is not aligned on a byte boundary, the value of the unused bits is considered as *don't care*.

- 0 ≤ NB ≤ 7: MSB = SPI_TX[7:0]
- 8 ≤ NB ≤ 15: MSB = SPI_TX[15:8]
- 16 ≤ NB ≤ 23: MSB = SPI_TX[23:16]
- 24 ≤ NB ≤ 31: MSB = SPI_TX[31:24]

The SPI_TX register is a 32-bit wide register that is 16-bit, or 32-bit addressable. Partial register update with successive 16-bit accesses can be used to load the transmit register. In that case, the LSB must be updated before the MSB part of the transmitted word.

The received data are accessible on the OCP bus through SPI_RX register.

Table 13. Receive Register Bit Description (SPI_RX—0x038)

Base Address = 0xFFFB 0C00, Offset = 0x38				
Bit	**Name**	**Function**	**Access**	**Reset**
31:0	SPI_RX	Receive data	R	0x00000000

Note: The number of bits of the word to be received is programmed through the NB bit field in SPI_CTRL register. Receive register (SPI_RX) data read must be completed according to the received word bit length and in the proper sequence register access.

The SPI_RX register is considered to be empty if the most-significant byte part of the received word has been read (in functional mode only and not in emulation mode). If the number of bits of the received word is not aligned on a byte boundary, the unused bits are read as undefined value.

The SPI_RX register is a 32-bit register that is 16-bit, or 32-bit addressable. Partial register reads with successive 16-bit accesses can be used to read the receive register. In this case, the LSB must be read before the MSB part of the received word.

Table 14. Test Register Bit Description (SPI_TEST—0x03C)

Base Address = 0xFFFB 0C00, Offset = 0x3C				
Bit	**Name**	**Function**	**Access**	**Reset**
31:11	Reserved	A read access returns 0.	R	0x000000
10:6	RTSPEN	Read value of TSPEN	R	0
5	RCV	Read clock value	R	0
4	WCV	Write clock value	R/W	0
3	RTV	Read test value (spy TSPDI)	R	0
2	WTV	Write test value (force TSPDO)	R/W	0
1	FDO	Force TSPDO to read value from WTV bit: Active high	R/W	0
0	TMODE	Test mode enable: Active high	R/W	0

When the test mode is selected by setting the TMODE configuration bit in the SPI_TEST register, it enables the following features:

- ❏ TSPDO is fed back to TSPDI.
- ❏ It is possible to control and monitor the TSPDO, TSPDI, and CLK_S pins.

When the test mode is not active, a read to the SPI_TEST register always returns 0.

- FDO: This control bit enables forcing TSPDO to read the value of the WTV bit, allowing control of TSPDO.
- WTV: This bit enables the forcing the TSPDO value (test purposes).
- RTV: This bit is directly connected to TSPDI. It assumes that the input signal is static.
- WCV: This bit enables forcing the CLK output (in master mode only) to provide control of the CLK output.
- RCV: This bit is directly connected to CLK input. It enables testing connectivity of CLK_S in feedback mode only.
- RTSPEN: These bits are directly connected to TSPEN outputs when the test mode is enabled.

1.4 Protocol Description

The serial port interface must be configured via the setup registers.

A read process is always simultaneous with a write process, because the internal shift register is based on a loop (FIFO principle). However, the concurrent write process can be a dummy write if there is no data to transmit.

Depending on the mode selection (master or slave mode), the shift register clock can be derived internally from the CLK_M, or it can come directly from the CLK_S input.

The external transfer of a packet starts as soon as one of the transmit clocks is generated.

The received/transmitted data packet is shifted in/shifted out on the rising or falling edge of the shift register clock (SRCLK).

The loading of the packet is then validated on the deactivation of the enable signal (rising or falling edge).

SPI_ISR is updated at the end of a transaction in MCU-DSP and DMA modes (master or slave), and an interrupt request can be generated depending on SPI_IER bits.

Sections 1.4.1 to 1.4.3 present the steps describing the MCU-DSP and DMA protocols. These steps must be followed in order to have a correct behavior.

In MCU-DSP protocol, the interrupt request must not be masked. Thus, the SPI can issue an interrupt to inform the host that the transaction is finished.

1.4.1 MCU-DSP Protocol

The host is either the MCU or the DSP.

MCU-DSP Transmit Protocol in Master Mode

The protocol has several steps:

Step 1: MCU-DSP writes to the setup registers (SPI_SET1 and SPI_SET2).

Step 2: MCU-DSP writes to the transmit register (SPI_TX).

Step 3: MCU-DSP writes to the control register (SPI_CTRL).

Once the WR bit is set:

- ❏ The transmit register (SPI_TX) is copied into the shift register (SPI_SR).
- ❏ The device enable goes low (nTSPENi), if CEi = 0 in SPI_SET2.
- ❏ The shift register clock is activated (SRCLK) and the transmission starts.

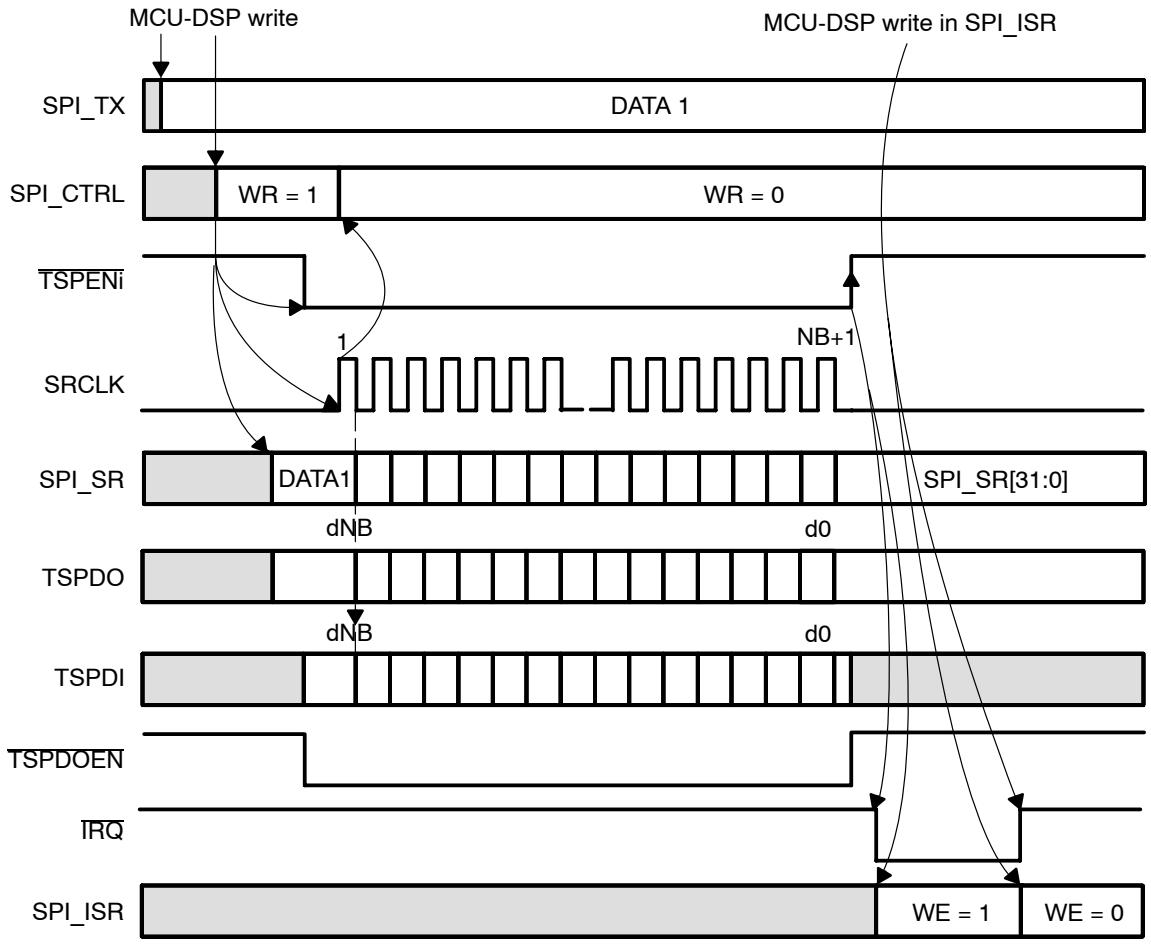
One SRCLK cycle later, the WR bit is reset.

Step 4: When the transmission is completed:

- ❏ The device enable (nTSPENi) goes high if CEi = 0 in SPI_SET2.
- ❏ The WE bit is set in the interrupt status register (SPI_ISR).
- ❏ An interrupt is generated, if MSK1 is set in the interrupt enable register (SPI_IER).

Step 5: Once the MCU-DSP clears the WE status bit (SPI_ISR), the interrupt request is released.

Figure 3. MCU-DSP Transmit Protocol in Master Mode With CIi = 0, CEi = 0 and CPi = 0



MCU-DSP Receive/Transmit Protocol in Master Mode

The protocol has several steps:

Step 1: MCU-DSP writes to the setup registers (SPI_SET1 and SPI_SET2).

Step 2: MCU-DSP writes to the transmit register (SPI_TX) (optional). This is necessary only when you want to perform a transmission at the same time.

Step 3: MCU-DSP writes to the control register (SPI_CTRL).

Once the RD bit is set:

- ❏ The transmit register is copied into the shift register (SPI_SR).
- ❏ The device enable goes low (nTSPEN[i]), if CEi = 0 in SPI_SET2.
- ❏ The shift register clock is activated (SRCLK) and the transmission and reception start.
- ❏ One SRCLK cycle later, the RD bit is reset.

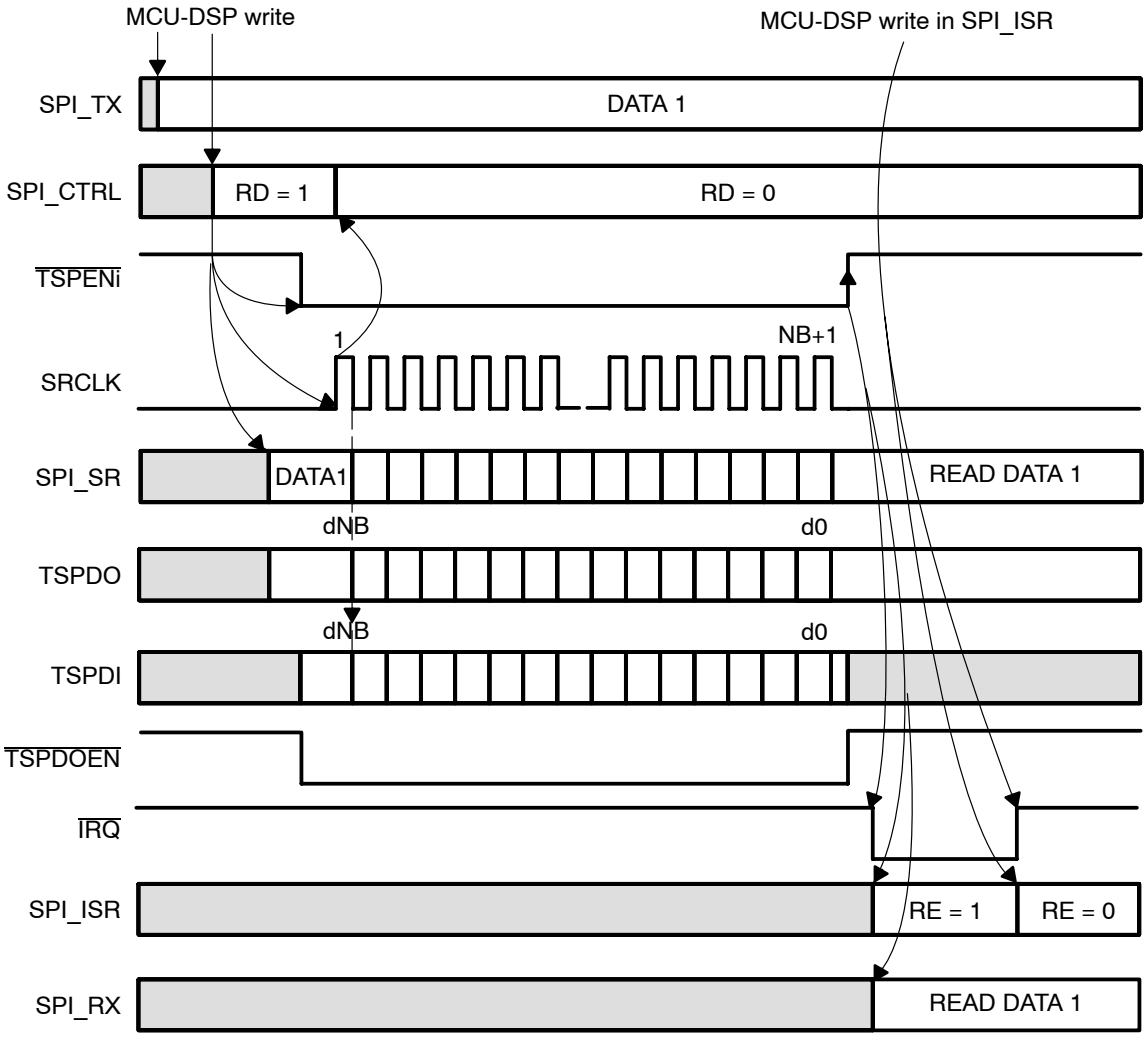
The WR bit has no effect on behavior. This bit is reset like RD, if it has been set.

When the reception is completed:

- ❏ The device enable goes high (nTSPEN[i]), if CEi = 0 in SPI_SET2.
- ❏ The RE bit is set in the interrupt status register (SPI_ISR).
- ❏ The shift register (SPI_SR) is copied into the receive register (SPI_RX).
- ❏ An interrupt is generated if MSK0 is set in the interrupt enable register (SPI_IER).

Step 4: Once the MCU-DSP has read the receive register (SPI_RX) and has cleared the RE status bit (SPI_ISR), the interrupt request is released.

Figure 4. MCU-DSP Receive Transmit Protocol in Master Mode With CIi = 0, CEi = 0, and CPi = 0



MCU-DSP Transmit Protocol in Slave Mode

The protocol has several steps:

Step 1: MCU-DSP writes to the setup registers (SPI_SET1 and SPI_SET2).

Step 2: MCU-DSP writes to the transmit register (SPI_TX).

Step 3: MCU-DSP writes to the control register (SPI_CTRL).

Once the WR bit is set, the transmit register (SPI_TX) is copied into the shift register (SPI_SR).

If CEi = 0 in SPI_SET2, the transmission starts as soon as the slave device enable goes low (nSPEN0) and the shift register clock is activated (SRCLK).

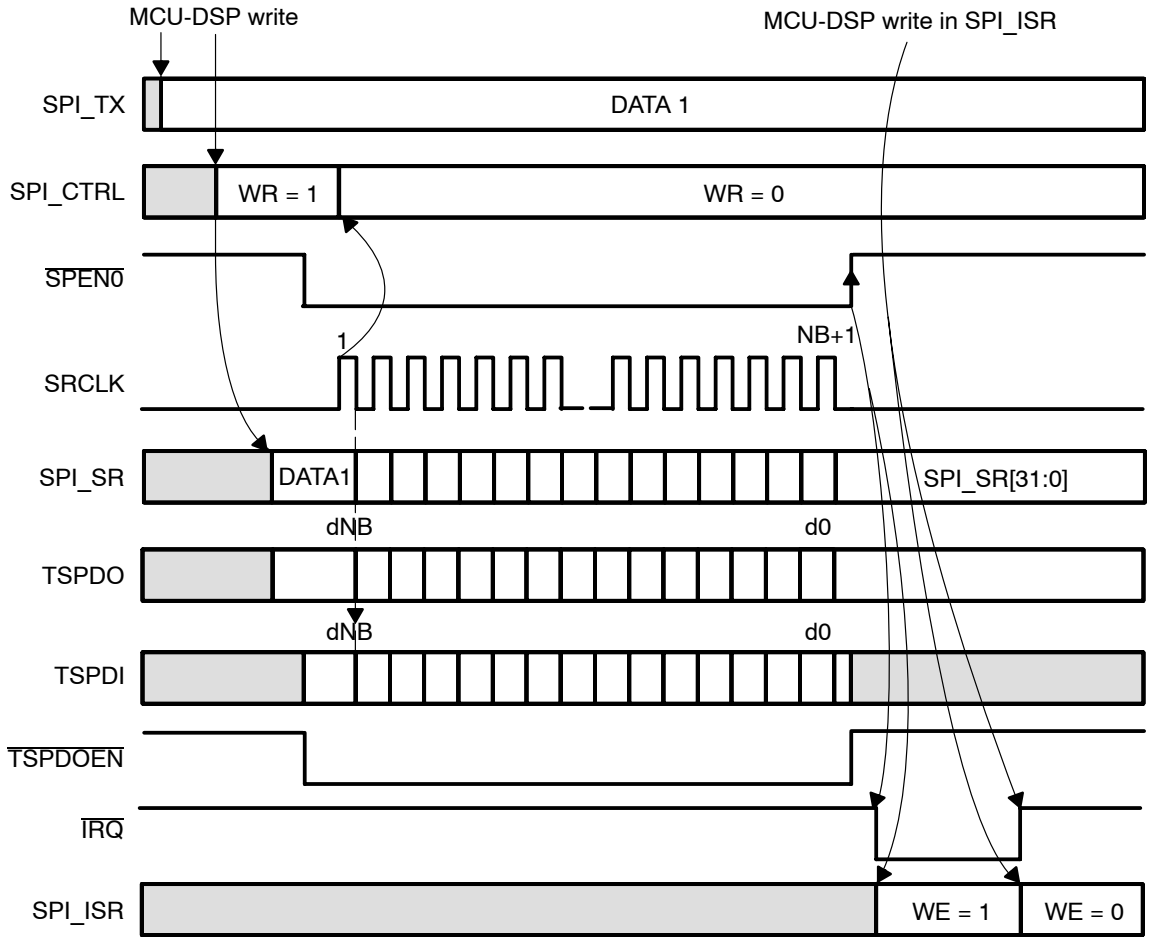
One SRCLK cycle later, the WR bit is reset.

Step 4: When the transmission is completed:

- ❏ The device enable goes high (nSPEN0) if CEi = 0 in SPI_SET2.
- ❏ The WE bit is set in the interrupt status register (SPI_ISR).
- ❏ An interrupt is generated if MSK1 is set in the interrupt enable register (SPI_IER).

Step 5: Once the MCU-DSP clears the WE status bit (SPI_ISR), the interrupt request is released.

Figure 5. MCU-DSP Transmit Protocol in Slave Mode With CIi = 0, CEi = 0 and CPi = 0



MCU-DSP Receive/Transmit Protocol in Slave Mode

The protocol is made up of several steps:

Step 1: MCU-DSP writes to the setup registers (SPI_SET1 and SPI_SET2).

Step 2: MCU-DSP writes to the transmit register (SPI_TX) (optional).

This is necessary only when you want to perform a transmission at the same time.

Step 3: MCU-DSP writes to the control register (SPI_CTRL).

Once the RD bit is set, the transmit register (SPI_TX) is copied into the shift register (SPI_SR).

If CEi = 0 in SPI_SET2, the transmission and reception start as soon as the slave device enable goes low (nSPEN0) and the shift register clock is activated (SRCLK).

One cycle later, the RD bit is reset.

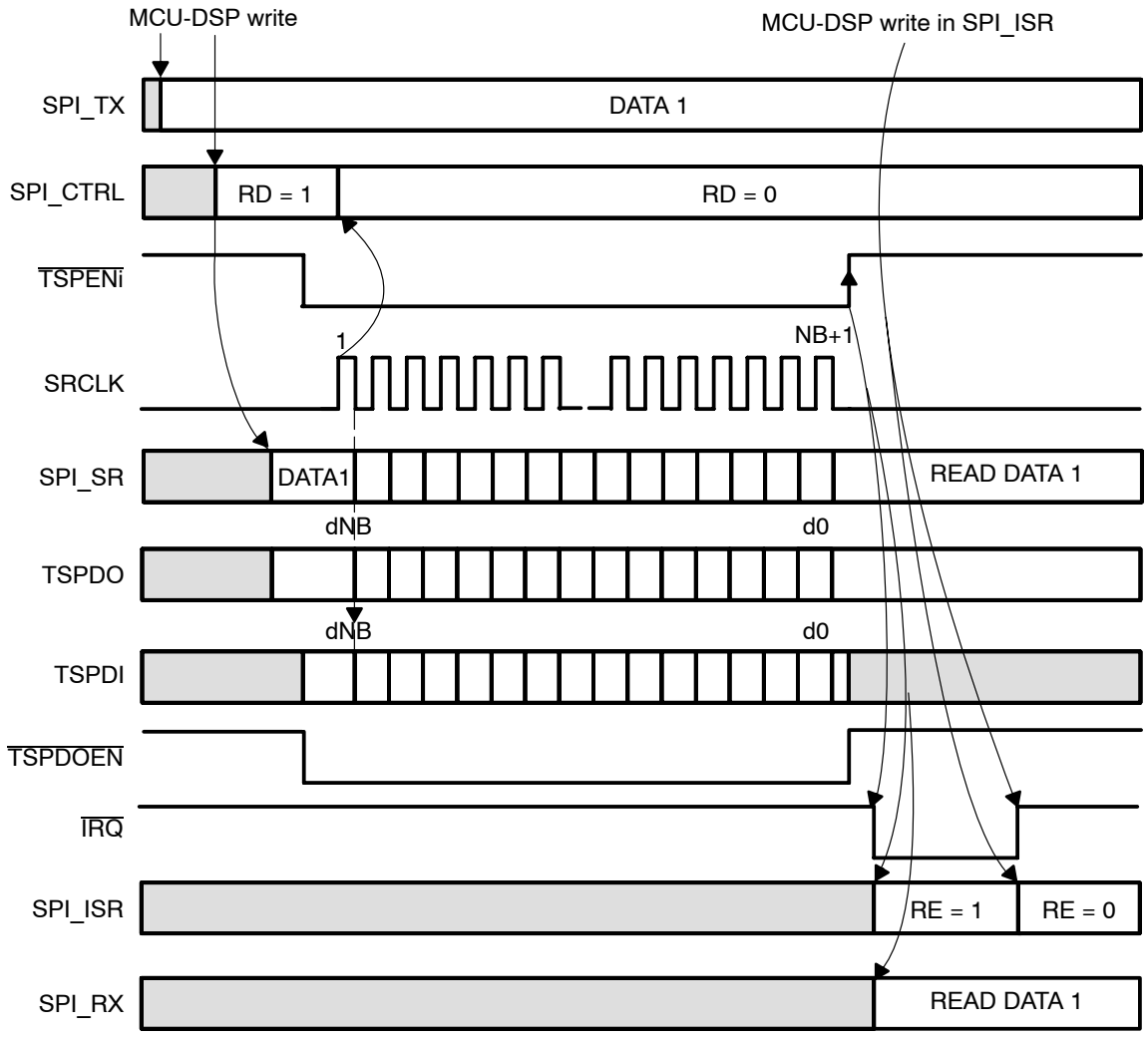
The WR bit has no effect on the behavior. This bit is reset like RD, if it has been set.

Step 4: When the reception is completed:

- ❏ The device enable goes high (nSPEN0) if CEi = 0 in SPI_SET2.
- ❏ RE status bit is set in the interrupt status register (SPI_ISR).
- ❏ The shift register (SPI_SR) is copied into the receive register (SPI_RX).
- ❏ An interrupt is generated if MSK0 is set in the interrupt enable register (SPI_IER).

Step 5: Once the MCU-DSP has read the receive register (SPI_RX) and has cleared the RE status bit (SPI_ISR), the interrupt request is released.

Figure 6. MCU-DSP RX/TX Protocol in Slave Mode With CIi = 0, CEi = 0 and CPi = 0



1.4.2 DMA Protocol

The interrupt (nIRQ) waveform is the same as the one in MCU-DSP protocol.

DMA Transmit Protocol in Master Mode

The protocol has several steps:

Step 1: MCU-DSP writes to the setup and control registers (SPI_SET1, SPI_SET2, and SPI_CTRL).

When DMA_EN is set, a transmit DMA request is generated, once the WR bit is set.

Step 2: The DMA writes the data to the transmit register (SPI_TX). Once the data is written:

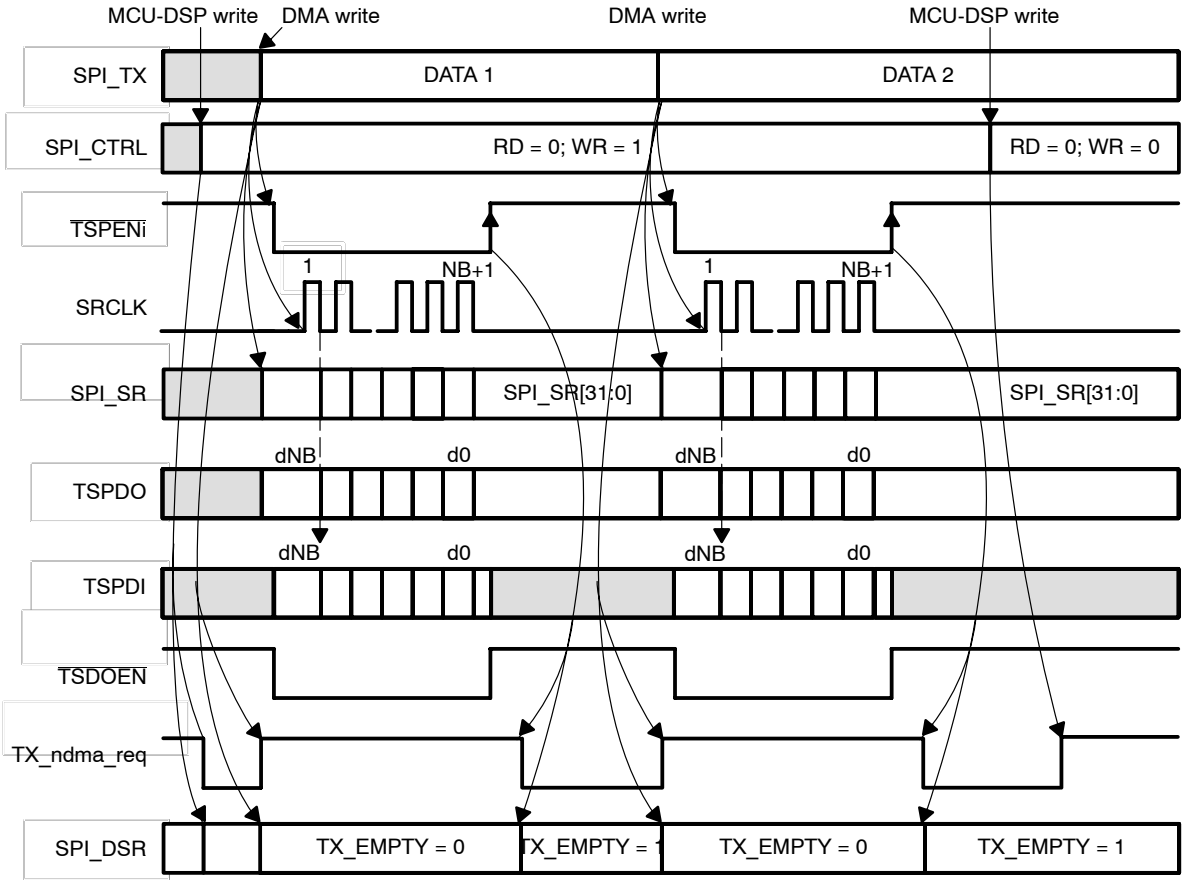
- ❏ The transmit DMA request is cleared.
- ❏ The TX_EMPTY status bit is reset in the data status register (SPI_DSR).
- ❏ The transmit register (SPI_TX) is copied into the shift register (SPI_SR).
- ❏ The device enable goes low (nTSPENi), if CEi = 0 in SPI_SET2.
- ❏ The shift register clock is activated (SRCLK) and the transmission starts.

Step 3: When the transmission is completed:

- ❏ The device enable goes high (nTSPENi) if CEi = 0 in SPI_SET2.
- ❏ TX_EMPTY is set in the data status register (SPI_DSR).
- ❏ A transmit DMA request is generated.
- ❏ Another transmission is able to start (Step2 → Step3 → Step2 → Step3…).

To stop the process, the MCU-DSP must reset the WR bit in the control register (SPI_CTRL).

Figure 7. DMA TX Protocol in Master Mode With CIi = 0, CEi = 0 and CPi = 0



DMA Receive Protocol in Master Mode

The protocol has several steps:

Step 1: MCU-DSP writes to the setup registers (SPI_SET1 and SPI_SET2).

Step 2: MCU-DSP writes to the control register (SPI_CTRL). Once the RD bit is set:

- ❏ The device enable goes low (nTSPENi) if CEi = 0 in SPI_SET2.
- ❏ The shift register clock is activated (SRCLK) and the reception starts.

Step 3: When the reception is completed:

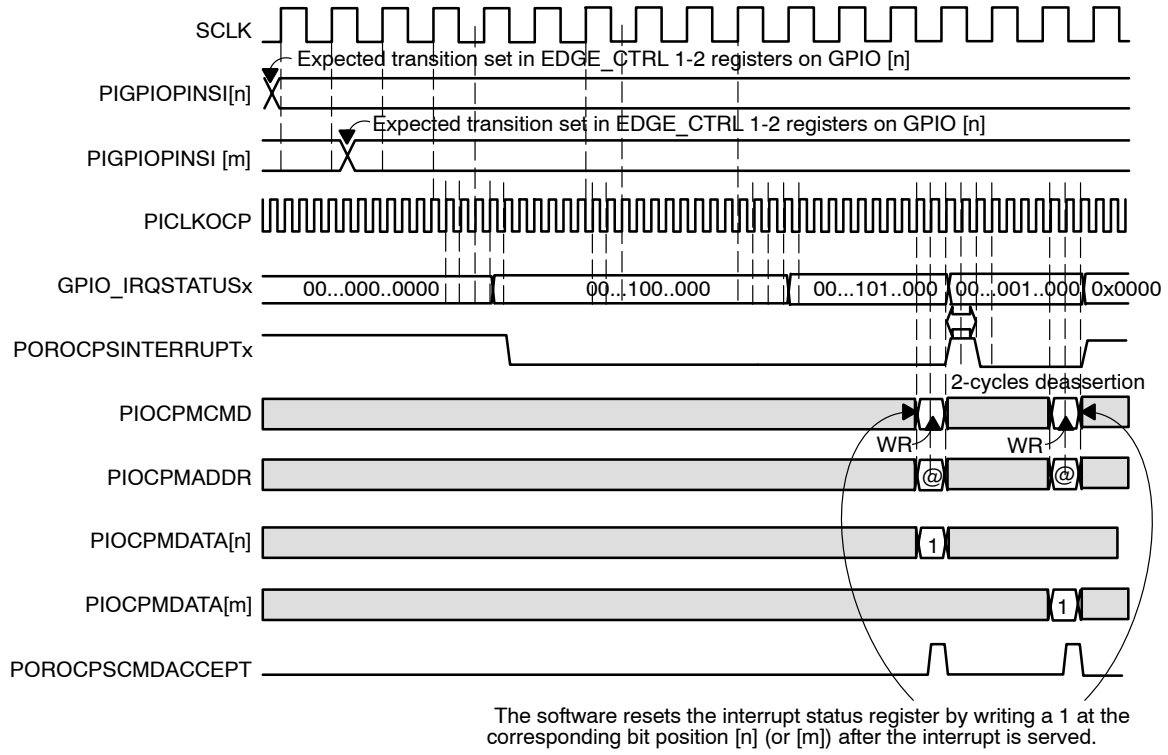
- ❏ The device enable goes high (nTSPENi) if CEi = 0 in SPI_SET2.
- ❏ The RX_FULL status bit is set in the data status register (SPI_DSR).
- ❏ The shift register (SPI_SR) is copied into the receive register (SPI_RX).
- ❏ A receive DMA request is generated.

Step 4: Once the DMA reads the receive register (SPI_RX):

- ❏ The device enable goes low (nTSPENi) if CEi = 0 in SPI_SET2.
- ❏ The RX_FULL status bit is reset in the data status register (SPI_DSR).
- ❏ The receive DMA request is released.
- ❏ Another reception starts (Step3 → Step4 → Step3 → Step4…).

To stop the process, the MCU-DSP must reset the RD bit in the control register (SPI_CTRL).

Figure 8. DMA Receive Protocol in Master Mode With CIi = 0, CEi = 0 and CPi = 0



DMA Transmit and Receive Protocol in Master Mode

The protocol has several steps:

Step 1: MCU-DSP writes to the setup and control registers (SPI_SET1, SPI_SET2, and SPI_CTRL).

When DMA_EN is set, a transmit DMA request is generated, once the RD and WR bits are set.

Step 2: The DMA writes the data in the transmit register (SPI_TX). Once the data is written:

- ❏ The transmit DMA request is released.
- ❏ TX_EMPTY is reset in the data status register (SPI_DSR).
- ❏ The transmit register (SPI_TX) is copied into the shift register (SPI_SR).
- ❏ The device enable goes low (nTSPENi), if CEi = 0 in SPI_SET2.
- ❏ The shift register clock (SRCLK) is activated and the transmission and reception start.

Step 3: When the transmission and the reception are completed:

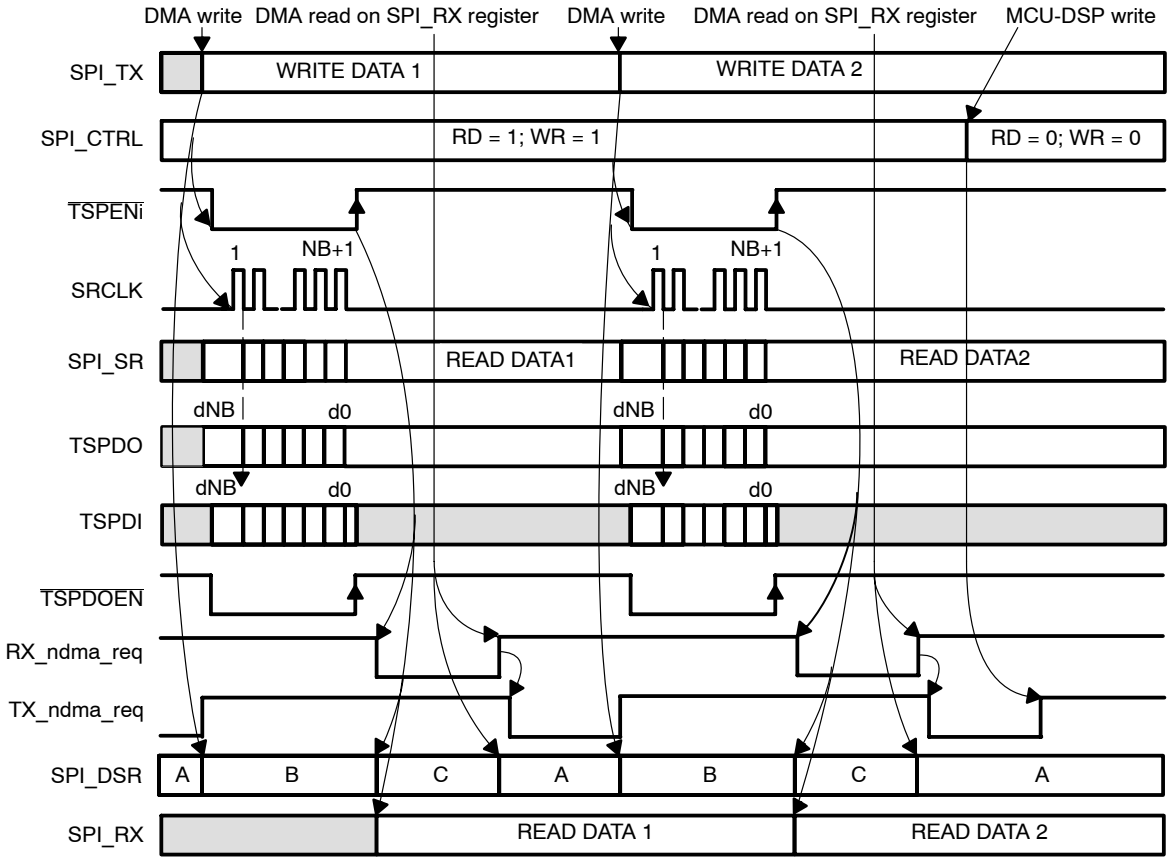
- ❏ The device enable goes high (nTSPENi), if CEi = 0 in SPI_SET2.
- ❏ The shift register (SPI_SR) is copied into the receive register (SPI_RX).
- ❏ RX_FULL and TX_EMPTY are set in the data status register (SPI_DSR).
- ❏ A receive DMA request is generated.

Step 4: Once the DMA reads the receive register (SPI_RX):

- ❏ The RX_FULL status bit is reset in the data status register (SPI_DSR).
- ❏ The receive DMA request is released.
- ❏ A transmit DMA request is generated.
- ❏ Another transmission and reception can start (Step 2 → Step 3 → Step 4 → Step 2 → Step 3 → Step4 →…).

To stop the process, the MCU-DSP has to reset the RD and WR bits in the control register (SPI_CTRL).

Figure 9. DMA Transmit and Receive Protocol in Master Mode With CIi = 0, CEi = 0 and CPi = 0



A: TX_EMPTY = 1 and RX_FULL = 0

B: TX_EMPTY = 0 and RX_FULL = 0

C: TX_EMPTY = 1 and RX_FULL = 1

DMA Transmit Protocol in Slave Mode

The protocol has several steps:

Step 1: MCU-DSP writes to the setup and control registers (SPI_SET1, SPI_SET2, and SPI_CTRL).

When DMA_EN is set, a transmit DMA request is generated, once the WR bit is set.

Step 2: The DMA writes the data in the transmit register (SPI_TX). Once the data is written:

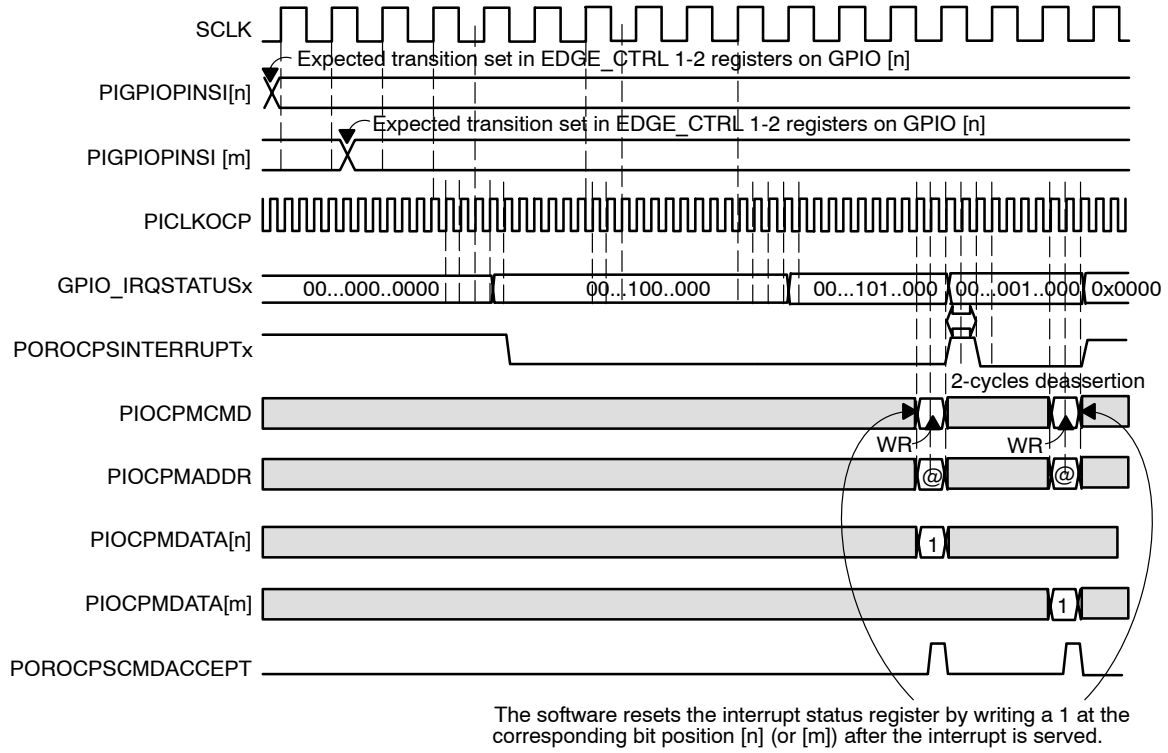
- ❏ The transmit DMA request is cleared (TX_NDMA_REQ goes high).
- ❏ TX_EMPTY is reset in the data status register (SPI_DSR).
- ❏ The transmit register (SPI_TX) is copied into the shift register (SPI_SR).
- ❏ If CEi = 0 in SPI_SET2, the transmission starts as soon as the slave device enable goes low (nSPEN0) and the shift register clock is activated (SRCLK).

Step 3: When the transmission is completed:

- ❏ TX_EMPTY is set in the data status register (SPI_DSR).
- ❏ A transmit DMA request is generated.
- ❏ Another transmission can start (Step2 → Step3 → Step2 → Step3…).

To stop the process, the MCU-DSP has to reset the WR bit in the control register (SPI_CTRL).

Figure 10. DMA Transmit Protocol in Slave Mode With CIi = 0, CEi = 0 and CPi = 0



DMA Receive Protocol in Slave Mode

The protocol has several steps:

Step 1: MCU-DSP writes to the setup and control registers (SPI_SET1, SPI_SET2, and SPI_CTRL).

- ❏ If CEi = 0 in SPI_SET2 and the RD bit is set, the reception starts as soon as the slave device enable goes low (nSPEN0) and the shift register clock is activated (SRCLK).

Step 2: When the reception is completed:

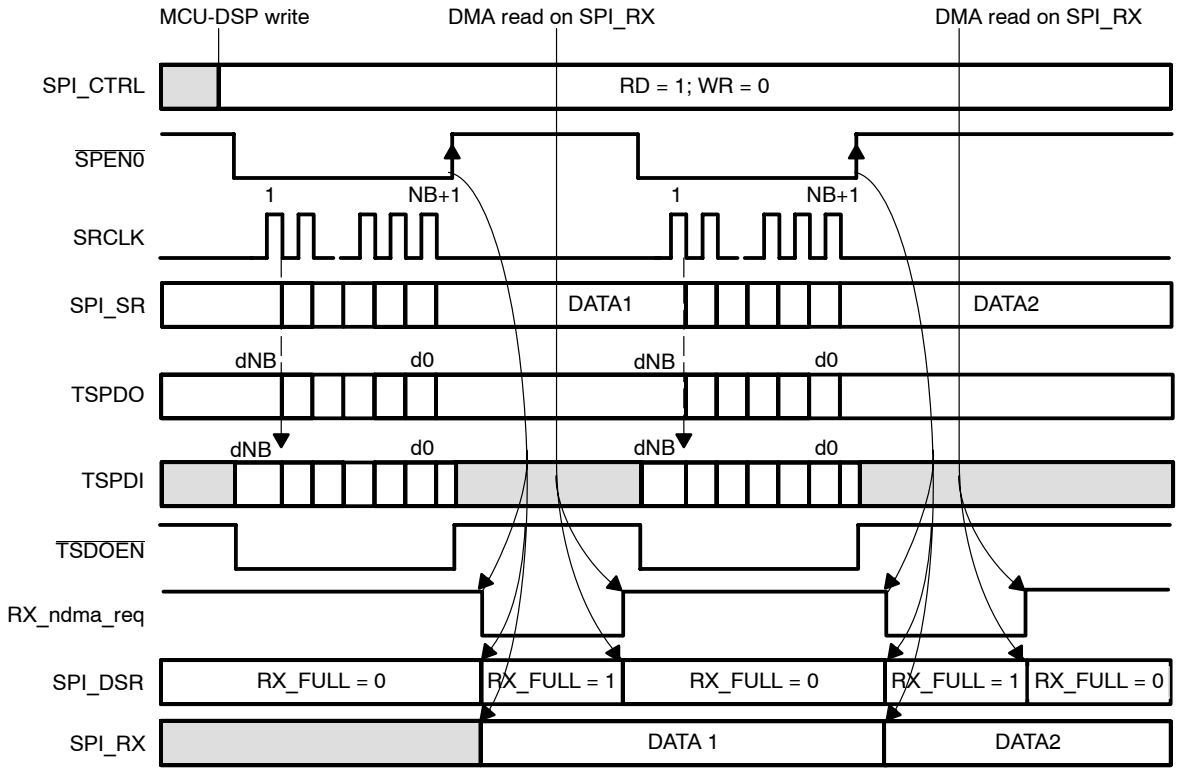
- ❏ The device enable goes high (nSPEN0), if CEi = 0 in SPI_SET2.
- ❏ The RX_FULL bit is set in the data status register (SPI_DSR).
- ❏ The shift register (SPI_SR) is copied into the receive register (SPI_RX).
- ❏ A receive DMA request is generated.

Step 3: Once the DMA reads the receive register (SPI_RX):

- ❏ The RX_FULL status bit is reset in the data status register (SPI_DSR).
- ❏ The receive DMA request is released.
- ❏ Another reception can start.

To stop the process, the MCU-DSP must reset the RD bit in the control register (SPI_CTRL).

Figure 11. DMA Receive Protocol in Slave Mode With CIi = 0, CEi = 0 and CPi = 0



DMA Transmit and Receive Protocol in Slave Mode

The protocol has several steps:

Step 1: MCU-DSP writes to the setup and control registers (SPI_SET1, SPI_SET2, and SPI_CTRL).

When the DMA_EN and WR bits are set, a transmit DMA request is generated.

Step 2: The DMA writes the data in the transmit register (SPI_TX). Once the data is written:

- ❏ The transmit DMA request is released.
- ❏ The TX_EMPTY status bit is reset in the data status register (SPI_DSR).
- ❏ The transmit register (SPI_TX) is copied into the shift register (SPI_SR).

If CEi = 0 in SPI_SET2, the transmission and reception start as soon as the slave device enable goes low (nSPEN0) and the shift register clock is activated (SRCLK).

Step 3: When the transmission and the reception are completed:

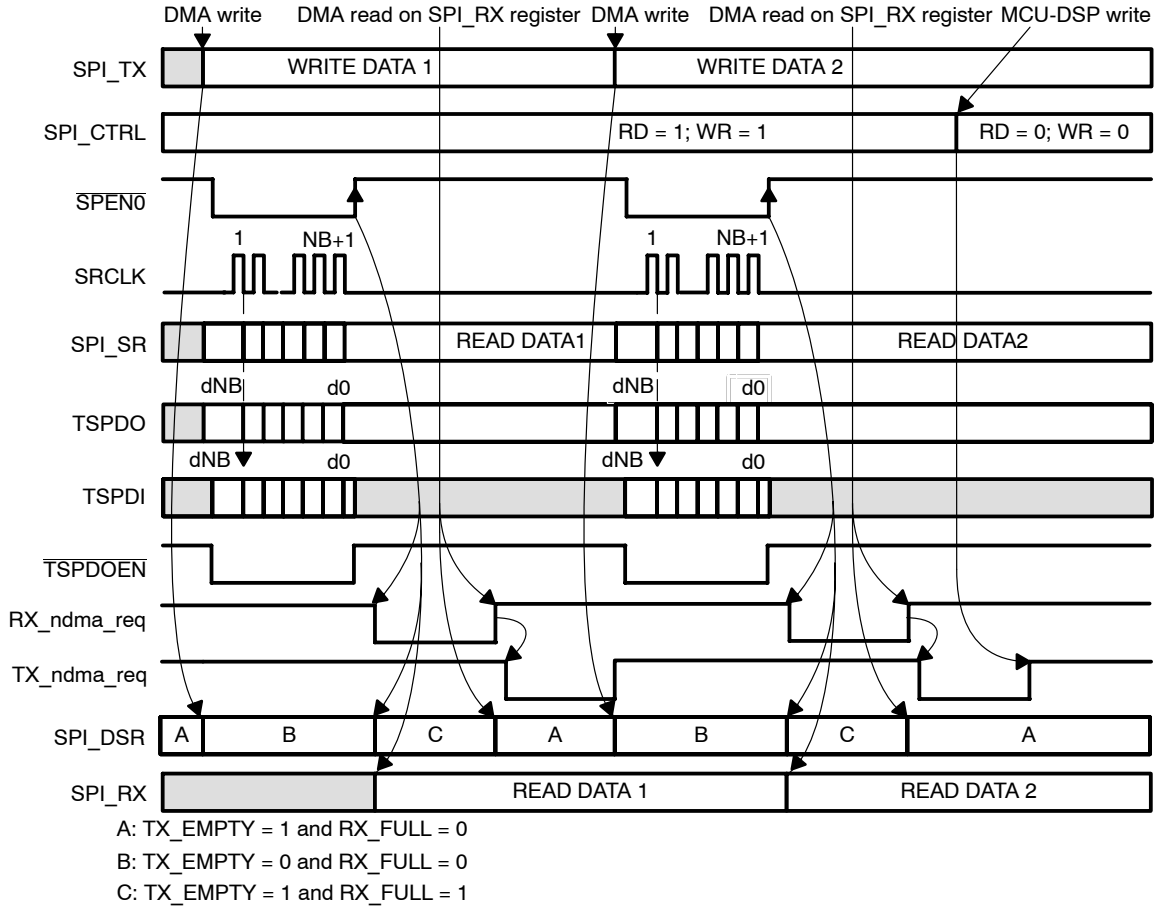
- ❏ The shift register (SPI_SR) is copied into the receive register (SPI_RX).
- ❏ The RX_FULL and TX_EMPTY bits are set in the data status register (SPI_DSR).
- ❏ A receive DMA request is generated.

Step 4: Once the DMA reads the receive register (SPI_RX):

- ❏ The RX_FULL status bit is reset in the data status register (SPI_DSR).
- ❏ The receive DMA request is released.
- ❏ A transmit DMA request is generated.
- ❏ Another transmission and reception can start.

To stop the process, the MCU-DSP must reset the RD and WR bits in the control register (SPI_CTRL).

Figure 12. DMA Transmit and Receive Protocol in Slave Mode With CIi = 0, CEi = 0 and CPi = 0



1.4.3 Overflow/Underflow Interrupts

In slave mode, whether the functional mode is MCU DSP or DMA, the possibility exists that the receive register (SPI_RX) is overflowing or/and the transmit register (SPI_TX) is underflowing. In either case, an interrupt is generated and sent to the host. It is up to the host to take the right action, according to the received interrupt.

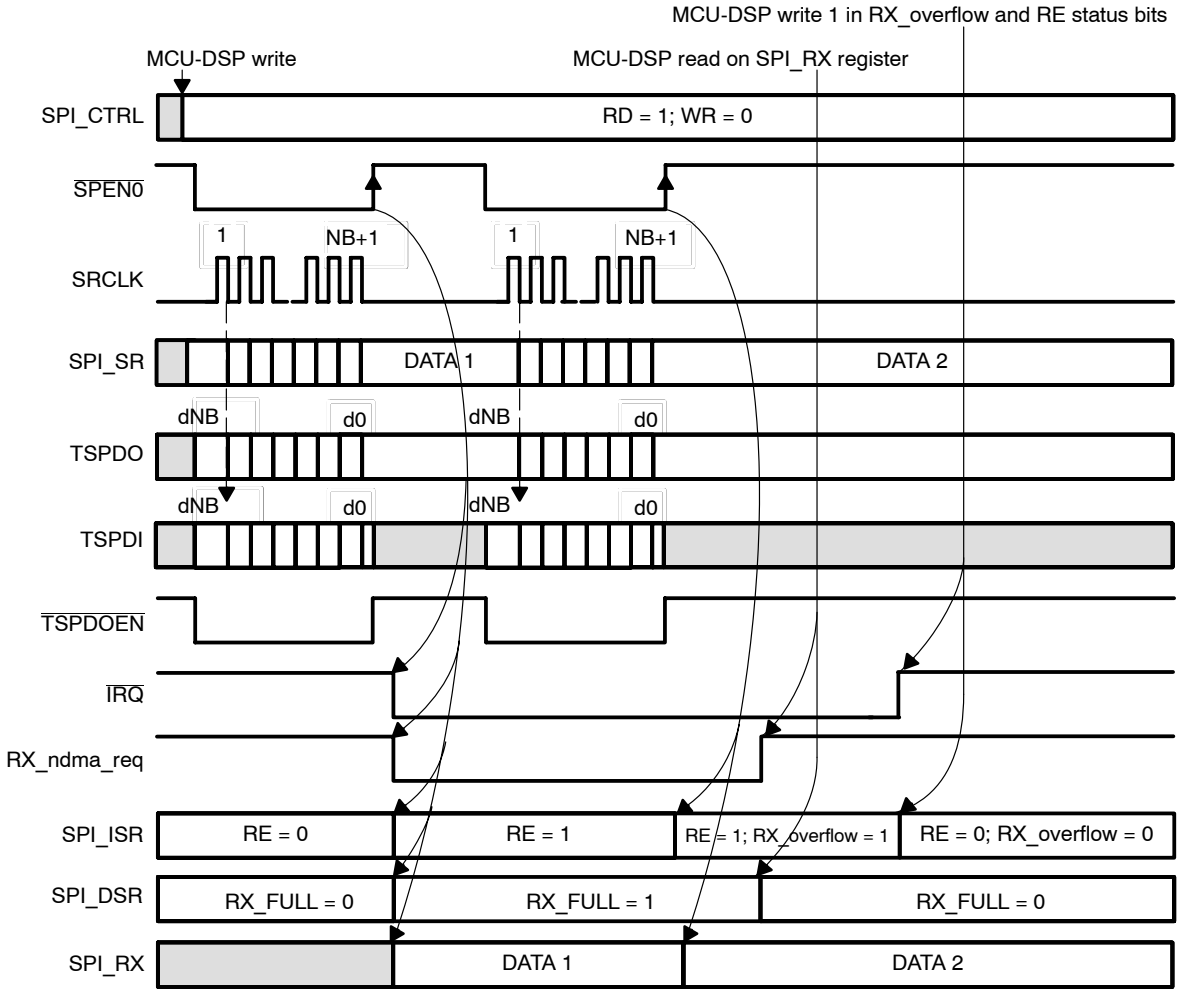
Overflow Interrupt Generation

To generate an overflow interrupt, the SPI must be in the following state:

- ❏ SPI is configured in slave mode (SPI_SET2 [15] = 0).
- ❏ Enable for overflow interrupt is active (SPI_IER [2] = 1).
- ❏ The receive register (SPI_RX) has not been read between two receptions.

To release the interrupt (nIRQ) activated by the RX overflow bit (SPI_ISR [2]), the user has to clear the RX overflow status bit by writing a 1 in SPI_ISR [2].

Figure 13. Example of an Overflow Generation With CIi = 0, CEi = 0 and CPi = 0



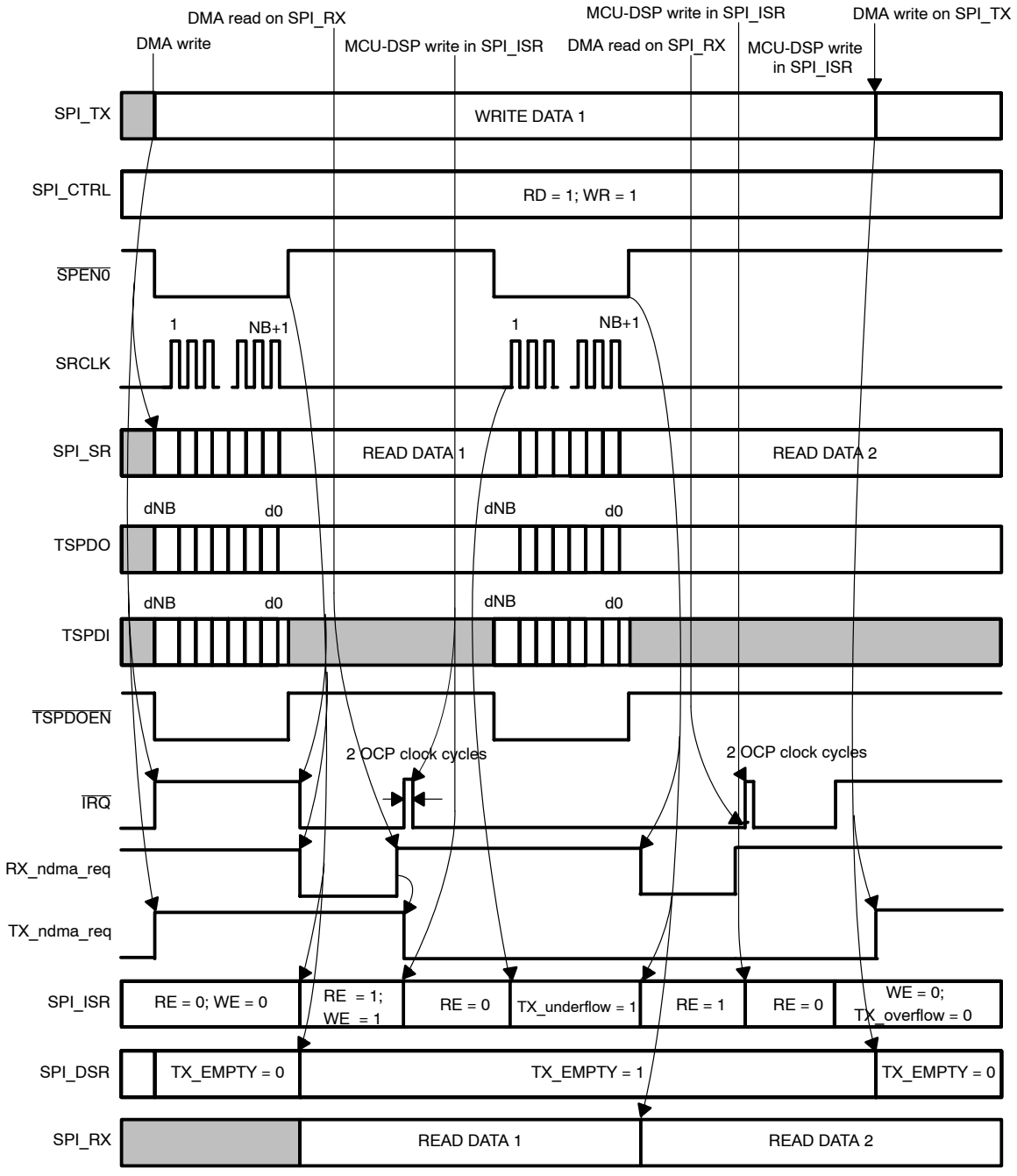
Underflow Interrupt Generation

To generate an underflow interrupt, the SPI has to be in the following state:

- ❏ SPI is configured in slave mode (SPI_SET2 [15] = 0).
- ❏ Enable for underflow interrupt is active (SPI_IER [3] = 1).
- ❏ The transmit register (SPI_TX) has not been updated between two transmissions.

To release the interrupt (nIRQ) activated by the TX underflow bit (SPI_ISR [3]), the user has to clear the Tx_underflow status bit by writing a 1 to SPI_ISR [3].

Figure 14. Example of an Underflow Generation With CIi = 0, CEi = 0 and CPi = 0



1.4.4 Transmission Modes

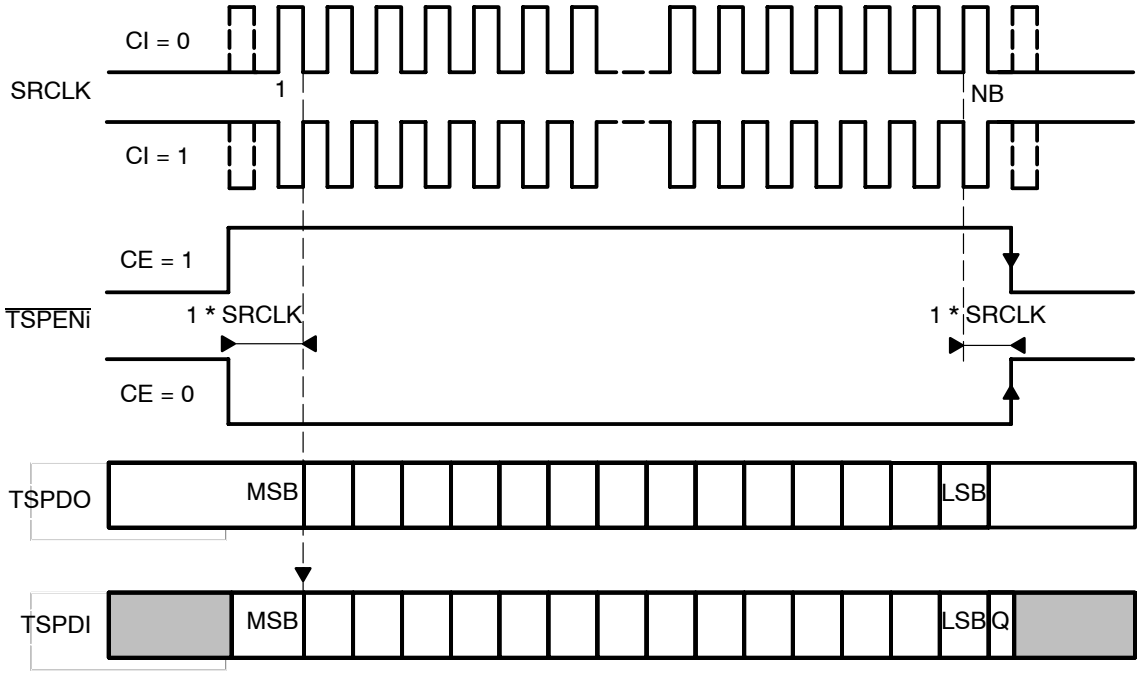
The serial interface is active as soon as the shift register clock is activated.

nSPEN0 behaves the same as nTSPENi, as shown in Figure 15 and Figure 16.

The transmitted data packet is shifted out on the rising or falling edge of SRCLK, whereas the received data packet is captured on the falling or rising edge of SRCLK (complementary edge).

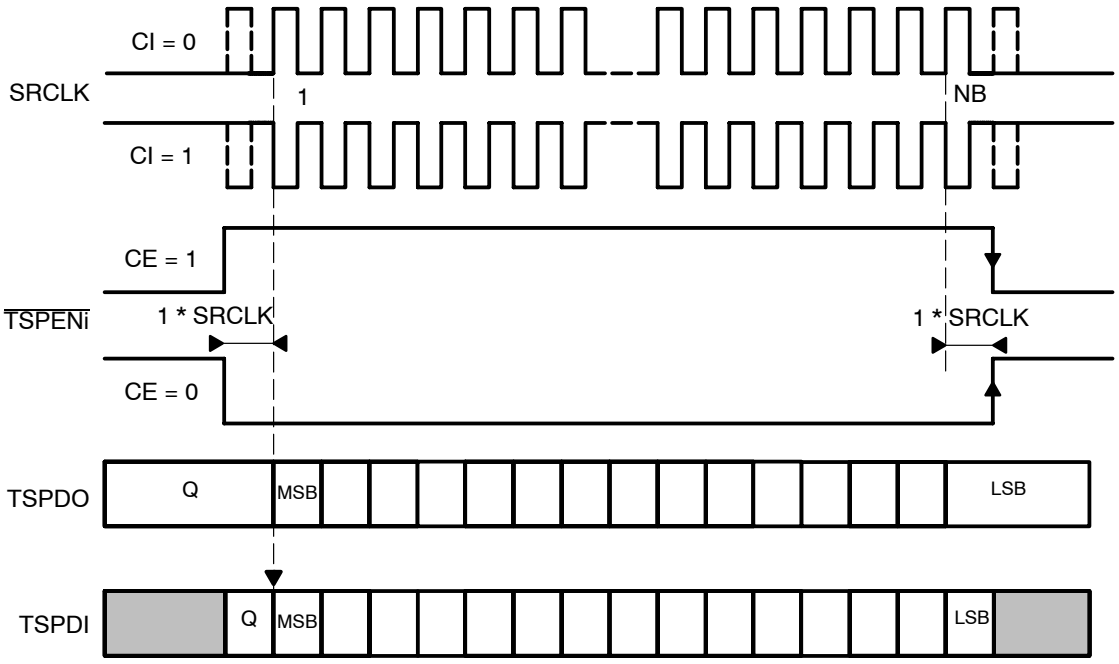
When CPi = 0, the first edge (rising or falling) of SRCLK is used to capture the data and the second edge (falling or rising) is used to shift the data.

Figure 15. Example of a Transmission With CPi = 0



When CPi = 1, the first edge (rising or falling) of SRCLK is used to shift the data and the second edge (falling or rising) is used to capture the data.

Figure 16. Example of a Transmission With CPi = 1



1.5 Idle and Wake-Up Feature

When the host processor issues an idle request, the SPI goes into idle mode (in this mode, no clock is provided to the SPI), according to the IDLEMODE field of the system configuration register SPI_SCR as described in Table 4.

If the IDLEMODE field is set to no-idle, the SPI does not go to idle mode.

If the IDLEMODE field is set to force-idle, the SPI goes into idle mode and acknowledges the request unconditionally. In this mode, SPI does not execute any transaction.

If the IDLEMODE field is set to smart-idle, the SPI module evaluates its internal capability to have the functional and interface clocks switched off. Once there is no more internal activity, the request signal is acknowledged and the SPI enters the sleep mode.

In this mode, the module issues a wake-up request when the following conditions are met:

- SPI is configured in slave mode (SPI_SET2 [15] = 0).
- The slave device enable (nSPEN0) becomes inactive after a transaction.

The ENAWAKEUP bit of the system configuration register (SPI_SCR) controls this wake-up request.

When the system wakes up, the following actions are executed:

- The idle request goes low (inactive).
- The wake-up request is deasserted (in smart-idle mode only).
- The WAKEUP status bit is set in the interrupt status register (SPI_ISR).
- The idle acknowledge goes low (inactive).
- An interrupt is generated if MSK4 is set in the interrupt enable register (SPI_IER).

Once the SPI acknowledges the idle request, the functional and interface clocks can be stopped one cycle later.

Some restrictions apply on the module functionality when a wake-up request is generated (in Smart-Idle mode only). The following conditions must be met to ensure the right behavior of the requested transaction:

- The SPI must be in DMA mode: DMA_EN set in the set up register (SPI_SET1 [5] = 1).
- The requested transaction that wakes up the module must be a receive: RD set and WR reset in the control register (SPI_CTRL [1:0] = 01).
- The functional clock (CLK_M) must be active if the bit ENAWAKEUP (SPI_SCR [2]) is set, in order to generate a wake-up request and to copy the received data into the SPI_RX register. If the bit ENAWAKEUP is reset, SPI does not work when receiving a read transaction.

The AUTOIDLE bit of the system configuration register (SPI_SCR [0]) can be set in order to save power. This bit controls the internal OCP clock activity:

- When this bit is cleared, the internal OCP clock is free-running.
- When this bit is set, the internal OCP clock becomes inactive if the OCP command is in IDLE state.

1.6 Emulation Mode

In emulation mode, SPI has a slightly different functionality: a read of SPI_RX does not clear the bit RX_FULL (SPI_DSR [0]). Thus, SPI_RX is still considered as not read for SPI.

Otherwise, SPI behavior in emulation mode is the same as in functional mode.

A read of the bits EMUSOFT and EMUFREE (SPI_SCR [6:5]) always gives 01. A write has no effect on these bits.

1.7 Reset

Before accessing or using the module, the local host must ensure that internal reset is released by reading the system status register (SPI_SSR).

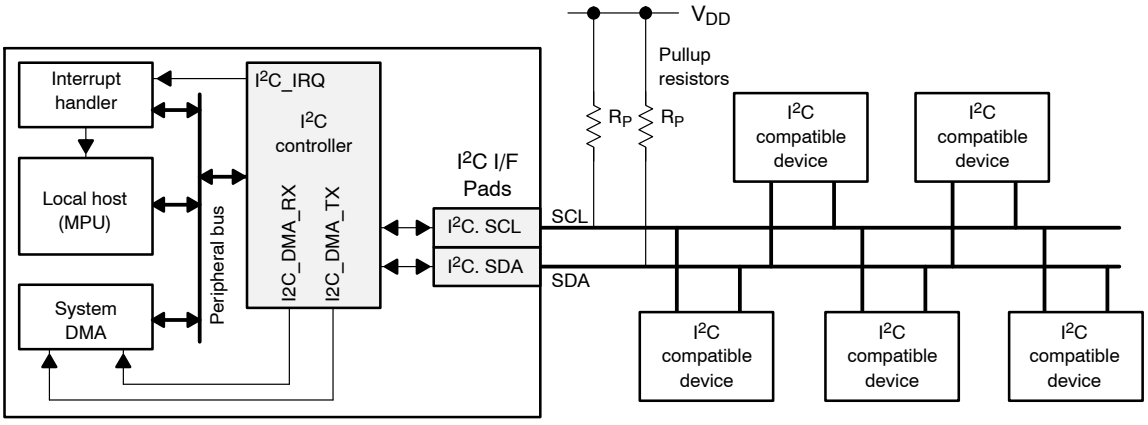
2 I²C Multimaster Peripheral

2.1 Overview

The multimaster I²C peripheral provides an interface between a local host (LH) such as an MPU, MIPS, or DSP processor and any I²C-bus-compatible device that connects via the I²C serial bus. External components attached to the I²C bus can serially transmit/receive up to 8-bit data to/from the LH device through the two-wire I²C interface.

This I²C peripheral supports any slave or master I²C-compatible device. Figure 17 shows the example of a system with multiple I²C compatible devices in which the I²C serial ports are connected together for a two-way transfer from one device to other devices.

Figure 17. I²C System Overview



2.2 Functional Overview

The I²C bus is a multimaster bus. The I²C controller supports the multimaster mode that allows more than one device capable of controlling the bus to be connected to it. Each I²C device, including the DSP, is recognized by a unique address and can operate as either transmitter or receiver, according to the function of the device. In addition to being a transmitter or receiver, a device connected to the I²C bus can also be considered as master or slave when performing data transfers. A master device is a device that 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.

2.3 I²C Controller Features

The main features of the I²C controller are:

- ❏ Compliance with Philips I²C specification version 2.1, January 2000
- ❏ Standard mode (up to 100K bits/s) and fast mode (up to 400K bits/s) support
- ❏ 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
- ❏ Low-power design
- ❏ Two DMA channels
- ❏ Wide-interrupt capability

The current I²C does not support:

- ❏ High-speed (HS) mode for transfer up to 3.4M bits/s
- ❏ C-bus-compatibility mode

2.4 I²C Master/Slave Controller Signal Pads

Data are communicated to devices interfacing with the I²C via the serial data line (SDA) and the serial clock line (SCL). These two wires carry information between the DSP or MPU device and other devices connected to the I²C bus. The I²C is a shared peripheral that can be allocated to the MCU or the DSP. Both the SDA and SCL are bidirectional pins. They must be connected to a positive supply voltage via 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 15. Signal Pads

Name	Type	Reset Value	Description
I2C_SCL	In/ Out(OD)	Input	I^2C serial CLK line. Open-drain output buffer. Requires external pull-up resistor (Rp).
I2C_SDA	In/ Out(OD)	Input	I^2C serial data line. Open-drain output buffer. Requires external pull-up resistor (Rp).

2.5 Operational Details

2.5.1 I^2C Reset

The I^2C module can be reset in the following three ways:

- A system bus reset (RESET_ = 0). A device reset causes the system bus reset.
- A software reset by setting the SRST bit in the I2C_SYSC register. This bit has the same action on the module logic as the system bus reset.
- The I2C_EN bit in the I2C_CON register can also reset a part of the I^2C module. When the system bus reset is released (RESET_ = 1), I2C_EN = 0 keeps the functional part of the I^2C module in reset state and all configuration registers can be accessed.

Table 16. Reset State of I^2C Signals

Pin	I/O/Z	System Reset	I^2C Reset (I2C_EN = 0)
SDA	I/O/Z	High impedance	High impedance
SCL	I/O/Z	High impedance	High impedance

2.5.2 I^2C Bit Transfer

The master device generates one clock pulse for each data bit transferred. Because of the variety of technology devices (CMOS, NMOS, bipolar) that can be connected to the I^2C bus, the levels of logical 0 (low) and 1 (high) are not fixed and depend on the associated level of VDD. See Table 17 for electrical specifications.

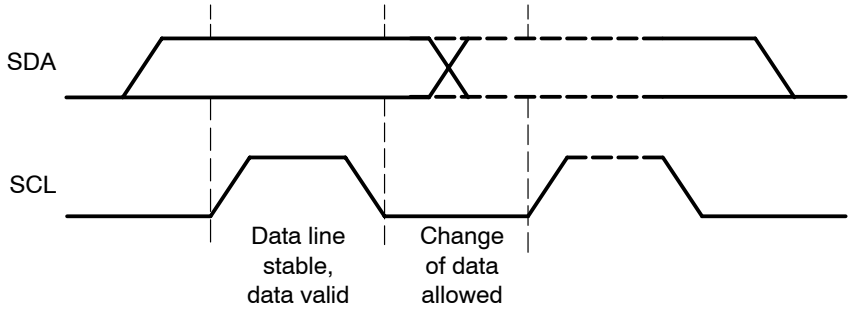
Table 17. Electrical Specification of the Input/Output

	Parameter	Standard Mode Devices		Fast-Mode Devices		Unit
		Min	Max	Min	Max	
VIL	Low-level input voltage:					
	Fixed input levels	−0.5	1.5	n/a	n/a	V
	VDD-related input levels	−0.5	0.3VDD	−0.5	0.3VDD	
VIH	High-level input voltage:					
	Fixed input levels	3.0	VDDmax+0.5	n/a	n/a	V
	VDD-related input levels	0.7VDD	VDDmax+0.5	0.7VDD	VDDmax+0.5	
	Low-level output voltage:					
	VDD>2V					
VOL1	At 3mA sink current	0	0.4	0	0.4	V
VOL2	At 6mA sink current	n/a	n/a	0	0.6	
	VDD<2V					
VOL3	At 3mA sink current	n/a	n/a	0	0.2VDD	

2.5.3 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. Bit Transfer on the I^2C Bus



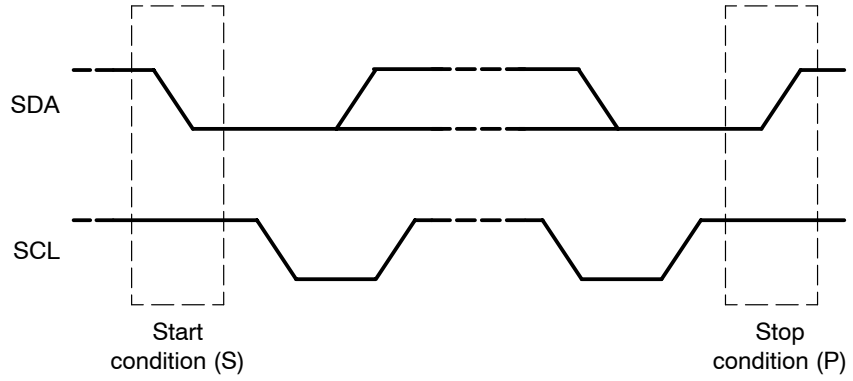
2.5.4 START and STOP Conditions

The I²C module generates start and stop conditions when it is configured as a master:

- START condition is a high-to-low transition on the SDA line while SCL is high.
- STOP condition is a low-to-high transition on the SDA line while SCL is high.

The bus is considered to be busy after the START condition (BB = 1) and free after the STOP condition (BB = 0).

Figure 19. Start and Stop Condition Events

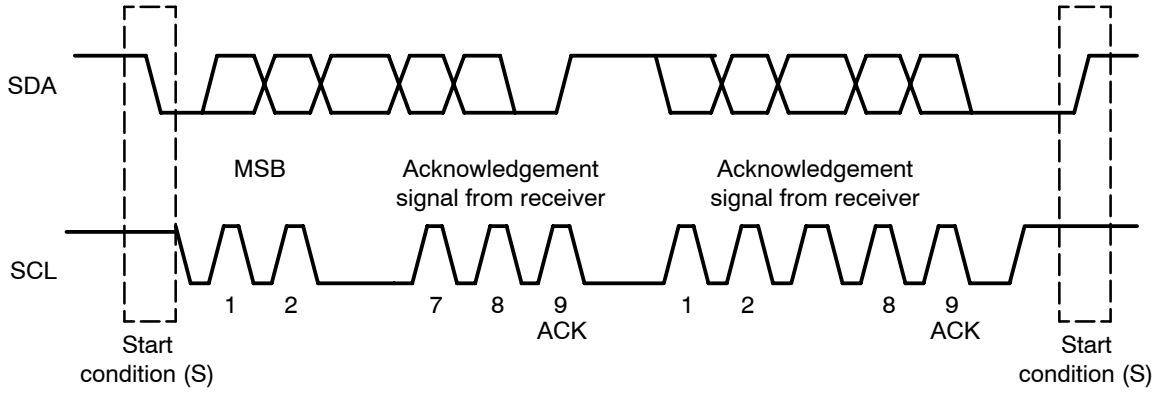


2.6 I²C Operation

Serial Data Formats

The I²C 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 are transferred with the most-significant bit (MSB) first. Each byte is followed by an acknowledge bit from the I²C module, if it is in receiver mode. The I²C controller supports endianism.

Figure 20. I2C Data Transfer



The I2C module supports two data formats, as shown in Figure 21:

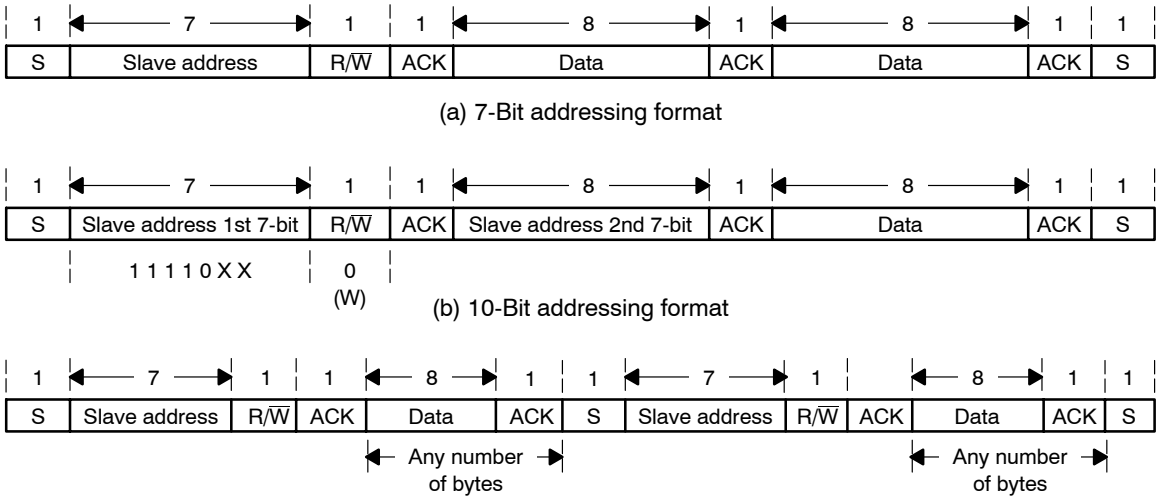
- 7-bit/10-bit addressing format
- 7-bit/10-bit addressing format with repeated start condition

The first byte after a start condition (S) always consists of 8 bits. In the 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 of 7 MSB slave-address bits and 1 LSB R/W_ bit. In the addressing formats with 10-bit addresses, the first byte has 7 MSB slave address bits, such as 11110XX where XX is the two MSB of the 10-bit addresses and 1 LSB R/W_ bit, which is 0 in this case.

The least-significant R/W_ of the address byte indicates the direction of the transmission of the following data bytes. If R/W_ is 0, the master writes data into the selected slave; if it is 1, the master reads data out of the slave.

Figure 21. I²C Data Transfer Formats



Master Transmitter

In this mode, data assembled in one of the previously described data formats is shifted out on the serial data line SDA in synch with the self-generated clock pulses on the serial clock line SCL. The clock pulses are inhibited and SCL is held low when the intervention of the processor is required (XUDF) after a byte has been transmitted.

Master Receiver

This mode can be entered only from the master transmitter mode. With any of the address formats (Figure 21 (a), (b), and (c)), the master receiver is entered after the slave address byte and bit R/W_ have been transmitted, if R/W_ is high. Serial data bits received on bus line SDA are shifted in synch with the self-generated clock pulses on SCL. The clock pulses are inhibited and SCL held low when the intervention of the processor is required (ROVR) after a byte has been transmitted. At the end of a transfer, it generates the stop condition.

Slave Transmitter

This mode can be entered only from the slave receiver mode. With any of the address formats (Figure 21 (a), (b), and (c)), the slave transmitter is entered if the slave address byte is the same as its own address and bit R/W_ has been transmitted, if R/W_ is high. The slave transmitter shifts the serial data out on the data line SDA in synch 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 the local host is required to intervene (XUDF).

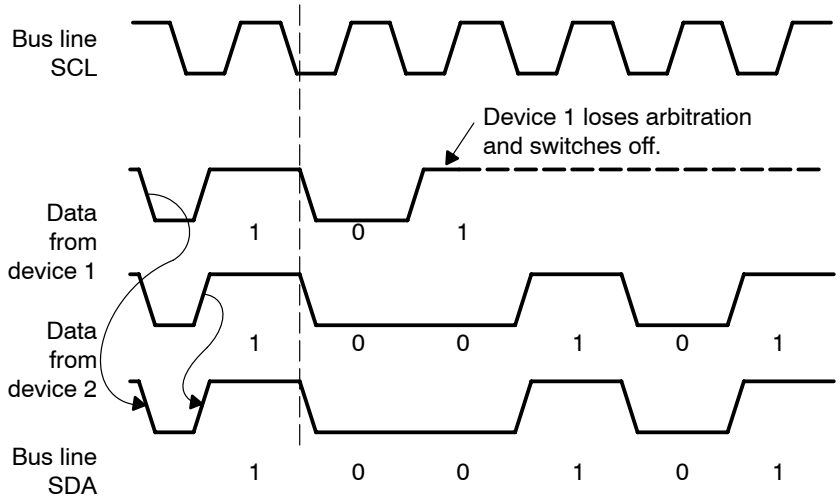
Slave Receiver

In this mode, serial data bits received on the bus line SDA are shifted-in in synch with the clock pulses on SCL that are generated by the master device. It does not generate the clock, but it can hold the clock line SCL low while the local host is required to intervene (ROVR) following the reception of a byte.

2.6.1 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, it switches to the slave receiver mode, sets the arbitration lost (AL) flag, and generates the arbitration lost interrupt. Figure 22 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. Should two or more devices send identical first bytes, arbitration continues on the subsequent bytes.

Figure 22. Arbitration Procedure Between Two Master Transmitters

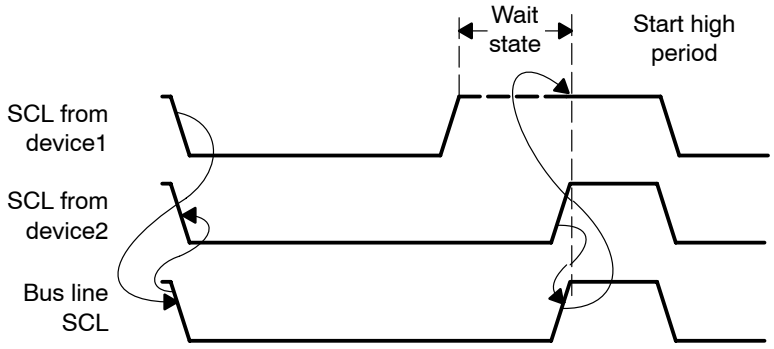


2.6.2 I²C Clock Generation and I²C Clock Synchronization

Under normal conditions, only one master device generates the clock signal, SCL. During the arbitration procedure, however, 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 generation of 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 achieved, where the slowest device determines the length of the low period and the fastest 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 23 illustrates the clock synchronization.

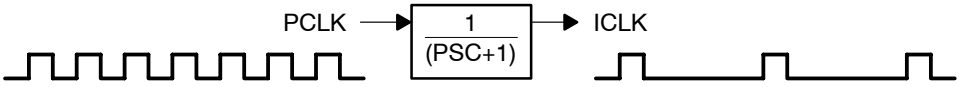
Figure 23. Synchronization of Two I²C Clock Generators



2.6.3 Prescaler (SCLK/ICLK)

The I²C module is operated with an internal approximately 12-MHz clock (ICLK). This clock is generated via the I²C prescaler block. The prescaler consists of an 8-bit register. I²C _PSC is used for dividing down the system clock (SCLK) to obtain an approximately 12-MHz clock for the I²C module.

Figure 24. Synchronization of Two I²C Clock Generators



0x0:	Divide by 1
0x1:	Divide by 2
↓	↓
0xFF:	Divide by 256

Values after reset are low (All 8 bits).

2.6.4 Noise Filter

The noise filter suppresses any noise that is 50 ns or less. It is designed to suppress noise with 1 ICLK, assuming the lower and upper limits of ICLK are 8 MHz and 16 MHz, respectively.

2.6.5 I²C Interrupts

The I²C module generates six types of interrupts: Arbitration-lost, no-acknowledge, general call, registers-ready-for-access, receive, and transmit. These six interrupts are accompanied by six interrupt masks and flags defined in the I2C_IE and I2C_STAT registers, respectively.

- Arbitration lost interrupt (AL): Generated when the I²C arbitration procedure is lost.

- No-acknowledge interrupt (NACK): Generated when the master I²C does not receive an acknowledge from the receiver.
- General call interrupt (GC): Generated when the device detects the address of all zeros (8 bits).
- Registers-ready-for-access interrupt (ARDY): Generated by the I²C when the previously programmed address, data, and command have been performed and the status bits have been updated. This interrupt is used to let the LH know that the I²C registers are ready for access.
- Receive interrupt/status (RRDY): Generated when there is received data ready to be read by the LH from the I2C_DATA register. The LH can poll this bit to read the received data from the I2C_DATA register.
- Transmit interrupt/status (XRDY): Generated when the LH needs to 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.

When the interrupt signal is activated, the local host must read the I2C_STAT register to define the type of interrupt, process the request, and then write into this register the right value to clear the interrupt flag.

2.6.6 DMA Events

The I²C module can generate two DMA requests events, read (I2C_DMA_RX) and write (I2C_DMA_TX), that the DMA controller can use to synchronously read received data from the I2C_DATA and write transmitted data to the I2C_DATA register. The DMA read and write requests are generated in a similar manner as RRDY and XRDY, respectively.

The I²C 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.

2.7 Register Map

Table 18 lists the revision, interrupt enable, I²C status, system status, buffer configuration, data counter, and data access registers. Table 19 through Table 26 describe the register bits.

Start address: FFFB 3800

Table 18. Register Map

Name	Addr Offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I2C_REV	0x00	REV															
I2C_IE	0x04											GC_IE	XRDY_IE	RRDY_IE	ARDY_IE	NACK_IE	AL_IE
I2C_STAT	0x08	SBD			BB	ROVR	XUDF	AAS				GC	XRDY	RRDY	ARDY	NACK	AL
Reserved	0x0C																
I2C_SYSS	0x10																RDONE
I2C_BUF	0x14	RDMA_EN								XDMA_EN							
I2C_CNT	0x18	DCOUNT															
I2C_DATA	0x1C	DATA															
I2C_SYSC	0x20															SRST	
I2C_CON	0x24	I2C_EN	BE			STB	MST	TRX	XA							STP	STT
I2C_OA	0x28	OA															
I2C_SA	0x2C	SA															
I2C_PSC	Ox30	PSC															
I2C_SCLL	0x34	SCLL															
I2C_SCLH	0x38	SCLH															
I2C_SYSTEST	0x3C	ST_EN	FREE	TMODE		SSB								SCL_I	SCL_O	SDA_I	SDA_O

All bits defined as reserved must be written by software with zeros to preserve future compatibility. When read, any reserved bit returns 0. Also, note that it is good software practice to use complete mask patterns for setting or testing bit fields individually within a register.

Table 19. Module Revision Register(I2C_REV)

Bit	Name	Description
15:8	–	Reserved
7:0	REV	Module version number

This read-only register contains the hard-coded revision number of the module. A write to this register has no effect.

Module Revision Number (REV)

This 8-bit field indicates the revision number of the current I^2C controller module. Its value is fixed by hardware and corresponds to the RTL revision of this module.

The 4 LSBs indicate a minor revision. The 4 MSBs indicate a major revision.

For example:

- 0x20: Revision 2.0
- 0x21: Revision 2.1

A reset has no effect on the value returned.

Note:

I^2C controller with interrupt using interrupt vector register (I2C_IV) is revision 1.x.

I^2C controller with interrupt using status register bits (I2C_STAT) is revision 2.x.

Table 20. Interrupt Enable Register(I2C_IE)

Bit	Name	Description
15:6	–	Reserved
5	GC_IE	General call interrupt enable
4	XRDY_IE	Transmit data ready interrupt enable
3	RRDY_IE	Receive data ready interrupt enable
2	ARDY_IE	Register access ready interrupt enable
1	NACK_IE	No acknowledgment interrupt enable
0	AL_IE	Arbitration lost interrupt enable

This R/W register controls the interrupts mask/unmask function.

The following are common to all bits:

When the local host sets a bit location to 1 , an interrupt is signaled to the local host if the corresponding bit location in I2C_STAT (status register) is asserted to 1 by the core of the I^2C controller. If it is set to 0, the interrupt is masked and is not signaled to the local host.

- 0: Interrupt disabled
- 1: Interrupt enabled

Value after reset is low (all bits).

Table 21. Status Register(I2C_STAT)

Bit	Name	Description
15	SBD	Single byte data
14:13	–	Reserved
12	BB	Bus busy
11	ROVR	Receive overrun
10	XUDF	Transmit underflow
9	AAS	Address as slave
8:6	–	Reserved
5	GC	General call
4	XRDY	Transmit data ready
3	RRDY	Receive data ready
2	ARDY	Register access ready
1	NACK	No acknowledgment interrupt enable
0	AL	Arbitration lost interrupt enable

This register is composed of read-only and read-/clear-only registers. It provides core status information for interrupt handling and other I^2C control management.

Single Byte Data (SBD)

This read-only bit is set to 1 in slave-receive or master-receive modes when the last byte that was read from the I2C_DATA register contains a single valid byte.

The core clears this bit to 0 when the local host clears the register access ready interrupt flag.

Note:

When SBD = 1, in little-endian data format (I2C_CON:BE = 0), the MSB reads as 0x00, and in big-endian format (I2C_CON:BE = 1), the LSB reads as 0x00.

Whenever the number of bytes to be received is unknown (for example, slave receiver), the LH must poll this bit before clearing the register access ready interrupt flag.

- 0: No action
- 1: Single valid byte in last 16-bit data read

Value after reset is low.

Bus Busy (BB)

This read-only bit indicates the state of the serial bus.

In slave mode, on reception of a start condition, the device sets BB to 1. BB is cleared to 0 after reception of a stop condition.

In master mode, the software controls BB. To start a transmission with a start condition, MST, TRX, and STT must be set to 1 in the I2C_CON register. To end a transmission with a stop condition, STP must be set to 1 in the I2C_CON register. When BB = 1, and STT is set to a 1, a restart condition is generated.

- 0: Bus is free
- 1: Bus is occupied

Value after reset is low.

Receive Overrun (ROVR)

Receive mode only.

This read-only bit indicates whether the receiver has experienced overrun. Overrun occurs when the shift register is full and the receive FIFO is full. An overrun condition does not result in a data loss; the peripheral is just holding the bus (low on SCL) to prevent others bytes from being received.

ROVR is set to 1 when the I^2C recognizes an overrun.

ROVR is clear when reading the I2C_DATA register, or when resetting the I^2C (I2C_CON:I2C_EN = 0).

- ❏ 0: Normal operation
- ❏ 1: Receiver overrun

Value after reset is low.

Transmit Underflow (XUDF)

Transmit mode only.

This read-only bit indicates whether the transmitter has experienced underflow.

In the master transmit mode, underflow occurs when the shift register is empty, the transmit FIFO is empty, and there are still some bytes to transmit (DCOUNT ≠ 0).

In the slave transmit mode, underflow occurs when the shift register is empty, the transmit FIFO is empty, and there are still some bytes to transmit (read request from external I^2C master).

XUDF is set to 1 when the I^2C recognizes an underflow. The core holds the line till the underflow cause disappears.

XUDF is clear when writing the I2C_DATA register or resetting the I^2C (I2C_CON:I2C_EN = 0).

- ❏ 0: Normal operation
- ❏ 1: Transmit underflow

Value after reset is low.

Address As Slave (AAS)

The device sets this bit to 1 when it recognizes its own slave address or an address of all zeros (8 bits). The AAS bit is reset to 0 by restart or stop.

- ❏ 0: No action
- ❏ 1: Address as slave

Value after reset is low.

General Call (GC)

The device sets this read-/clear-only bit to1 if it detects the address of all zeros (8 bits), that is, general call.

When the core sets this bit to 1, an interrupt is signaled to the local host if the interrupt was enabled.

The LH is able to clear this bit only by writing a 1 into this register. Writing 0 has no effect.

When this bit is set to 1, AAS also reads as 1.

- 0: No action
- 1: General call

Value after reset is low.

Transmit Data Ry (XRDY)

Transmit mode only.

This read-/clear-only bit (XRDY) is set to 1 when the I^2C peripheral is a master or slave transmitter, the LH is able to write new data into the I2C_DATA register, and the transmitter still requires new data. A master transmitter requests new data if DCOUNT $\neq 0$, and a slave transmitter requests new data if a read request is received from external master.

The transmitter requests two bytes to be written even if only a single byte is needed. In this case, the other byte needs to be filled with a dummy 0x00 value that is not transmitted over the I^2C line.

When the core sets this bit to 1, an interrupt is signaled to the local host if the interrupt was enabled. The LH can also poll this bit to write new transmitted data into the I2C_DATA register.

The LH is able to clear this bit only by writing a 1 into this register. Writing 0 has no effect.

If the DMA transmit mode is enabled, this bit is not set. Instead, a DMA TX request to the main DMA controller of the system is generated.

- 0: Transmit buffer full (or receiver mode)
- 1: Transmit data ready (for write) and byte is needed

Value after reset is low.

Receive Data Ry (RRDY)

This read-/clear-only RRDY is set to 1 when the LH is able to read new data from the I2C_DATA register. When the core sets this bit to 1, an interrupt is signaled to the local host if the interrupt was enabled. The LH can also poll this bit to read the received data in I2C_DATA register.

The LH is able to clear this bit only by writing a 1 into this register. Writing 0 has no effect.

In interrupt mode, the LH needs to poll this bit after each read to I2C_DATA to ensure that there is no other DATA on the FIFO waiting to be read. Indeed, the RRDY needs to be cleared to 0 in order to receive a new RRDY interrupt.

If the DMA receive mode is enabled, this bit is not set. Instead, a DMA RX request to the main DMA controller of the system is generated.

- 0: Receive buffer empty
- 1: Receive data ready (for read)

Value after reset is low.

Register Access Ry (ARDY)

When set to 1, this read-/clear-only bit indicates that the previously programmed data and command (receive or transmit, master or slave) has been performed and the status bit has been updated. The LH uses this flag to let it know that the I^2C registers are ready to be accessed again (see Table 22).

Table 22. ARDY Set Conditions

Mode	Others	ARDY Set Conditions
Master transmit	STP = 1	DCOUNT = 0
Master receive	STP = 1	DCOUNT = 0 and receiver FIFO empty
Master transmit or receive	STP = 0	DCOUNT passed 0
Slave transmit	–	Stop or restart condition received from master
Slave receive	–	Stop or restart condition and receiver FIFO empty

The LH is able to clear this bit only by writing a 1 into this register. Writing 0 has no effect.

- 0: No action
- 1: Access ready

Value after reset is low.

No Acknowledgment (NACK)

The read-/clear-only NO_ACKNOWLEDGE flag bit is set when the hardware detects NO_ACKNOWLEDGE has been received.

The LH is able to clear this bit only by writing a 1 into this register. Writing 0 has no effect.

- 0: Normal/no action required
- 1: NACK

Value after reset is low.

Arbitration_Lost (AL)

The read-/clear-only ARBITRATION_LOST flag bit is set to 1 when the device in the master transmitter mode senses it has lost an arbitration. This occurs when two or more transmitters start a transmission almost simultaneously, or when the I^2C attempts to start a transfer while BB (bus busy) is 1.

When this is set to 1 due to arbitration lost, the core automatically clears the MST/STP bits in the I2C_CON register and the I^2C becomes a slave receiver.

The LH is able to clear this bit only by writing a 1 to this register. Writing 0 has no effect.

- 0: Normal/no action required
- 1: Arbitration lost

Value after reset is low.

Table 23. System Status Register(I2C_SYSS)

Bit	Name	Description
15:1	–	Reserved
0	RDONE–	Reset Done

Reset Done (RDONE)

This read-only bit indicates the state of the reset in case of hardware reset, global software reset (I2C_SYSC.SRST), or partial software reset (I2C_CON.I2C_EN).

The module must receive all its clocks before it can grant a reset-completed status.

- 0: Internal module reset is ongoing or partially held in reset
- 1: Reset completed

Value after reset is low.

Table 24. Buffer Configuration Register(I2C_BUF)

Bit	Name	Description
15	RDMA_EN	Receive DMA channel enable
14:8	–	Reserved
7	XDMA_EN	Transmit DMA channel enable
6:0	–	Reserved

This R/W register enables DMA transfers.

Receive DMA Channel Enable (RDMA_EN)

When this bit is set to 1, the receive DMA channel is enabled and the core forces the receive data ready status bit (I2C_STAT:RRDY) to 0.

- ❏ 0: Receive DMA channel disabled
- ❏ 1: Receive DMA channel enabled

Value after reset is low.

Transmit DMA Channel Enable (XDMA_EN)

When this bit is set to 1, the transmit DMA channel is enabled and the core forces the transmit data ready status (I2C_STAT:XRDY) bit to 0.

- ❏ 0: Transmit DMA channel disabled
- ❏ 1: Transmit DMA channel enabled

Value after reset is low.

Table 25. Data Counter Register(I2C_CNT)

Bit	Name	Description
15:0	DCOUNT	Data count

This R/W register controls the numbers of bytes in the I^2C data payload.

Data Count (DCOUNT)

Master modes only (receive or transmit).

This 16-bit countdown counter decrements by 1 for every byte received or sent. A write initializes DCOUNT to a saved initial value. A read returns the number of bytes that are yet to be received or sent. A read into DCOUNT returns the initial value only before a start condition and after a stop condition.

When DCOUNT reaches 0, the core generates a stop condition if a stop condition was specified (I2C_CON:STP = 1), and the ARDY status flag is set to 1 in the I2C_STAT register.

If I2C_CON:STP = 0, then the I^2C asserts SCL = 0 when DCOUNT reaches 0. The LH can then reprogram DCOUNT to a new value and resume sending or receiving data with a new start condition (restart). This process repeats until the LH sets the STP to 1 in the I2C_CON register.

The ARDY flag is set each time DCOUNT reaches 0 and DCOUNT is reloaded to its initial value.

In slave mode (receive or transmit), DCOUNT is not used.

0x0: Data counter = 65536 bytes (2^{16})

0x1: Data counter = 1 bytes

↓ ↓

0xFFFF: Data counter = 65535 bytes ($2^{16} - 1$)

DCOUNT value after reset is 0x0000.

Table 26. Data Access Register(I2C_DATA)

Bit	Name	Description
15:0	DATA	Transmit/Receive FIFO data

This register is the entry point for the local host to read data from or write data to the FIFO buffer. The FIFO size is 2 x 16 bits (4 bytes). Bytes within a word are stored and read in little-endian format (I2C_CON:BE = 0) or big-endian format (I2C_CON:BE = 1).

Transmit/Receive FIFO Data Value (DATA)

When read, this register contains the received I^2C data packet (1 or 2 bytes). This register must be accessed 16-bit-wise by the LH. In the case of an odd number of bytes received to read, the upper byte (with I2C_CON.BE = 0) or the lower byte (with I2C_CON.BE = 1) of the last access always reads as 0x00. The LH must check the SBD status bit in I2C_STAT register in order to flush this null byte.

When written, this register contains the byte(s) value(s) to transmit over the I^2C data line (1or 2 bytes). This register must be accessed 16-bit-wise, 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-wise access. In the 16-bit-wise access, the module transmits only the relevant byte, based on the byte counter (I2C_CNT). This feature is useful for DMA access, which supports only one word size per channel.

In SYSTEST loopback mode (I2C_SYSTEST:TMODE = 11), this register is also the entry/receive point for the data.

Values after reset are low (all 16-bits).

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 local host performs a 16-bit access if the buffer contains a single byte.

Table 27. I^2C System Configuration Register(I2C_SYSC)

Bit	Name	Description
15:2	–	Reserved
1	SRST	Soft Reset
0	–	Reserved

Soft Reset (SRST)

When this bit is set to 1, all of the module is reset, as for the hardware reset. The core automatically sets this bit to 0, and it is only reset by the hardware reset. During reads, it always returns 0.

- ❏ 0: Normal mode
- ❏ 1: The module is reset

Values after reset is low.

Table 28. I²C Configuration Register(I2C_CON)

Bit	Name	Description
15	I2C_EN	I²C module enable
14	BE	Big endian mode
13:12	–	Reserved
11	STB	Start byte mode (master mode only)
10	MST	Master/slave mode
9	TRX	Transmitter/Receiver mode (master mode only)
8	XA	Expand address
7:2	–	Reserved
1	STP	Stop condition (master mode only)
0	STT	Start condition (master mode only)

Active Transfer Phase

During an active transfer phase (STT has been set to 1), no modification must be done in this register. Changing it may result in an unpredictable behavior.

CAUTION

I²C Module Enable (I2C_EN)

When this bit is set to 0, the I²C controller is not enabled and reset. When 0, the receive and transmit FIFOs are cleared and all status bits are set to their default values. None of the configuration registers (I2C_IE, I2C_BUF, I2C_CNT, I2C_CON, I2C_OA, I2C_SA, I2C_PSC, I2C_SCLL and I2C_SCLH) are reset, all keep their initial values, and all can be accessed. The LH must set this bit to 1 for normal operation.

- ❑ 0: I²C controller in reset
- ❑ 1: I²C module enabled

Value after reset is low.

Big Endian (BE)

When this bit is 0 (default), the FIFO is accessed in little-endian format. In transmit mode, the LSB (I2C_DATA [7:0]) is transmitted first, and the MSB (I2C_DATA [15:8]) is transmitted in second position over the I^2C line. Conversely, in receive mode, the first, or odd, byte received (1,3, 5…) is stored in the LSB position, and the second, or even, byte received is stored in the MSB position.

When the LH sets this bit to a 1, the FIFO is accessed in big-endian format. In transmit mode, the MS byte (I2C_DATA [15:8]) is transmitted first and the LSB (I2C_DATA [7:0]) is transmitted in second position over the I^2C line. Conversely, in receive mode, the first, or odd, byte received (1,3, 5…) is stored in the MS byte position, and the second, or even, byte received is stored in the LSB position.

- ❏ 0: Little-endian mode
- ❏ 1: Big-endian mode

Value after reset is low.

Start Byte (STB)

Master mode only.

The local host sets the start-byte mode bit to 1 to configure the I^2C in start byte mode (I2C_SA = 00000001). See the Philips I^2C specification Version 2.1 for more details.

- ❏ 0: Normal mode
- ❏ 1: Start byte mode

Value after reset is low.

Master/Slave Mode (MST)

When this bit is cleared, the I^2C controller is in the slave mode and the serial clock (SCL) is received from the master device.

When this bit is set, the I^2C controller is in the master mode and it generates the serial clock.

This bit is automatically cleared at the end of the transfer on a detected stop condition and in case of arbitration lost.

- ❏ 0: Slave mode
- ❏ 1: Master mode

Value after reset is low.

Transmitter/Receiver Mode (TRX)

Master mode only.

When this bit is cleared, the I^2C controller is in the receiver mode, and data on data line SDA is shifted into the receiver FIFO and can be read from the I2C_DATA register.

When this bit is set, the I^2C controller is in the transmitter mode, and the data written in the transmitter FIFO via I2C_DATA is shifted out on data line SDA.

- 0: Receiver mode
- 1: Transmitter mode

Value after reset is low. The operating modes are defined as shown in Table 29.

Table 29. I^2C Controller Transmitter/Receiver Operating Modes

MST	TRX	Operating Modes
0	x	Slave receiver
0	x	Slave transmitter
1	0	Master receiver
1	1	Master transmitter

Expand Address (XA)

When set, this bit expands the address to 10-bit.

- 0: 7-bit address mode
- 1: 10-bit address mode

Value after reset is low.

Stop Condition (STP)

Master mode only.

The local host is able to set this bit to a 1 to generate a stop condition. The hardware resets it to 0 after the stop condition has been generated. The stop condition is generated when DCOUNT passes 0.

When this bit is not set to 1 before the end of the transfer (DCOUNT = 0), the stop condition is not generated and the SCL line is held to 0 by the master, which can restart a new transfer by setting the STT bit to 1.

- 0: No action or stop condition detected
- 1: Stop condition queried

Value after reset is low.

Start Condition (STT)

Master mode only.

The local host is able to set this bit to a 1 to generate a start condition. The hardware resets it to 0 after the start condition has been generated. The start/stop bits can be configured to generate different transfer formats.

- 0: No action or start condition generated
- 1: Start

Value after reset is low.

Table 30. STT Register Values

STT	STP	Conditions	Bus Activities
1	0	Start	S–A–D
0	1	Stop	P
1	1	Start–Stop (DCOUNT = n)	S–A–D..(n)..D–P
1	0	Start (DCOUNT = n)	S–A–D..(n)..D

DCOUNT is the data count value in the I2C_CNT register.

Table 31. I^2C Own Address Register(I2C_OA)

Bit	Name	Description
15:10	–	Reserved
9:0	OA	Own address

This register specifies the module I^2C 7-bit or 10-bit address (own address).

2.7.1 Own Address (OA)

This field specifies either:

- A 10-bit address coded on OA [9:0] when XA (expand address, I2C_CON [8]) is set to 1.
- A 7-bit address coded on OA [6:0] when XA (expand address, I2C_CON [8]) is set to 0. In this case, application software must set OA [9:7] bits to 000.

Values after reset is low (all 10 bits).

Table 32. I2C_SA: I²C Slave Address Register

Bit	Name	Description
15:10	–	Reserved
9:0	SA	Slave address

This register specifies the addressed I²C module 7-bit or 10-bit address (slave address).

Slave Address (SA)

This field specifies either:

- A 10-bit address coded on SA [9:0] when XA (expand address, I2C_CON [8]) is set to 1.
- A 7-bit address coded on SA [6:0] when XA (expand address, I2C_CON [8]) is set to 0. In this case, application software must set SA [9:7] bits to 000.

Values after reset are high (all 10 bits).

Table 33. I²C Clock Prescaler Register(I2C_PSC)

Bit	Name	Description
15:8	–	Reserved
7:0	PSC	Prescale sampling clock divider value

This register specifies the internal clocking of the I²C peripheral core.

I2C_PSC [7–0]: Prescale sampling clock divider value (PSC)

The core uses this 8-bit value to divide the system clock (SCLK) and generate its own internal sampling clock (ICLK). The core logic is sampled at the clock rate of the system clock for the module, divided by (PSC+1).

0x0: Divide by 1

0x1: Divide by 2

↓ ↓

0xFF: Divide by 256

Values after reset are low (all 8 bits).

Table 34. I²C SCL Low Time Control Register(I2C_SCLL)

Bit	Name	Description
15:8	–	Reserved
7:0	SCLL	SCL low time

This register determines the SCL low time value when master.

SCL Low Time (SCLL)

Master mode only.

This 8-bit value generates the SCL low time value (t_{LOW}) when the peripheral is operated in master mode.

The SCL low time depends on the I2C_PSC value and the ICLK time period (internal sampling clock rate):

- When I2C_PSC = 0, t_{LOW} = (SCLL+7) * ICLK time period
- When I2C_PSC = 1, t_{LOW} = (SCLL+6) * ICLK time period
- When I2C_PSC > 1, t_{LOW} = (SCLL+5) * ICLK time period

The different values to compute the SCL low time are due to the synchronization stage and noise filter on the SCL line.

Values after reset are low (all 8 bits).

Table 35. I²C SCL High Time Control Register(I2C_SCLH)

Bit	Name	Description
15:8	–	Reserved
7:0	SCLH	SCL high time

This register determines the SCL high time value when master.

SCL High Time (SCLH)

Master mode only.

This 8-bit value generates the SCL high-time value (t_{HIGH}) when the peripheral is operated in master mode.

The SCL high time depends on the I2C_PSC value and the ICLK time period (internal sampling clock rate):

- When I2C_PSC = 0, t_{HIGH} = (SCLH+7) * ICLK time period
- When I2C_PSC = 1, t_{HIGH} = (SCLH+6) * ICLK time period
- When I2C_PSC > 1, t_{HIGH} = (SCLH+5) * ICLK time period

The different values to compute the SCL high time are due to the synchronization stage and noise filter on the SCL line.

Values after reset are low (all 8 bits).

Table 36. System Test Register (I2C_SYSTEST)

Bit	Name	Description
15	ST_EN	System test enable
14	FREE	Free running mode (on breakpoint)
13:12	TMODE	Test mode select
11	SSB	Set status bits
10:4	–	Reserved
3	SCL_I	SCL line sense input value
2	SCL_O	SCL line drive output value
1	SDA_I	SDA line sense input value
0	SDA_O	SDA line drive output value

System Test Register

Never set this register for normal I^2C operation.

This register facilitates system-level tests by overriding some of the standard functional features of the peripheral. It can permit the test of SCL counters, control the signals that connect to I/O pins, or create digital loopback for self-test when the module is configured in system test (SYSTEST) mode. It also provides stop/no-stop functionality in debug mode.

System Test Enable (ST_EN)

This bit must be set to 1 to permit other system test registers bits to be set.

- 0: Normal mode
- 1: System test enabled

Value after reset is low.

Free Running Mode After Breakpoint (FREE)

This bit determines the state of the I^2C controller when a breakpoint is encountered in the HLL debugger.

This bit can be set independently of the ST_EN value.

FREE = 0: If the I^2C controller is a master, it stops immediately after the completion of the ongoing bit transfer. If the I^2C controller is a slave, it stops during the data phase transfer when one byte is completely transmitted/ received.

FREE = 1: The I^2C runs free.

- 0: Stop mode (on breakpoint condition)
- 1: Free running mode

Value after reset is low.

Test Mode Select (TMODE)

In normal functional mode (ST_EN = 0), these bits are *don't care*. They always read as 00 and a write is ignored.

In system test mode (ST_EN = 1), these bits can be set according to the following table to permit various system tests (see Table 37).

Table 37. Test Mode Select

TMODE	Mode
00	Functional mode (default)
01	Reserved
10	Test of SCL counters (SCLL, SCLH, PSC)
11	Loop back mode select + SDA/SCL IO mode select

Values after reset is low (2 bits).

- SCL counter test mode: In this mode, the SCL pin is driven with a permanent clock as if mastered, with the parameters set in the I2C_PSC, I2C_SCLL, and I2C_SCLH registers.
- Loopback mode: In the master transmit mode only, data transmitted out of the I2C_DATA register (write action) is received in the same I2C_DATA register via an internal path through the 1-deep FIFO buffers. The DMA and interrupt requests are normally generated if enabled.
- SDA/SCL IO mode: In this mode, the SCL IO and SDA IO are controlled via the I2C_SYSTEST [3:0] register bits.

Set Status Bits (SSB)

Writing 1 into this bit also sets the six read/clear-only status bits contained in the I2C_STAT register (bits 5:0) to 1.

Writing 0 into this bit does not clear status bits that are already set; only writing 1 into a set status bit can clear it. This bit must be cleared before attempting to clear a status bit.

- 0: No action
- 1: Set all six bits in I2C_STAT [5:0] to 1

Value after reset is low.

SCL Line Sense Input Value (SCL_I)

In normal functional mode (ST_EN = 0), this read-only bit always reads 0.

In system test mode (ST_EN = 1 & TMODE = 11), this read-only bit returns the logical state taken by the SCL line (either 1 or 0).

Value after reset is low.

SCL Line Drive Output Value (SCL_O)

In normal functional mode (ST_EN = 0), this bit is *don't care*. It always reads 0 and a write is ignored.

In system test mode (ST_EN = 1 & TMODE = 11), a 0 forces a low level on the SCL line, and a 1 puts the I^2C output driver to a high impedance state.

- 0: Forces 0 on the SCL data line
- 1: SCL output driver in HiZ state

Value after reset is low.

SDA Line Sense Input Value (SDA_I)

In normal functional mode (ST_EN = 0), this read-only bit always reads 0.

In system test mode (ST_EN = 1 & TMODE = 11), this read-only bit returns the logical state taken by the SDA line (either 1 or 0).

Value after reset is low.

SDA Line Drive Output Value (SDA_O)

In normal functional mode (ST_EN = 0), this bit is *don't care*. It reads as 0 and a write is ignored.

In system test mode (ST_EN = 1 & TMODE = 11), a 0 forces a low level on the SDA line and a 1 puts the I^2C output driver to a high impedance state.

- 0: Forces 0 on the SDA data line
- 1: SDA output driver in Hi-Z state

Value after reset is low.

2.8 Programming Guidelines

2.8.1 Main Program

Module Configuration Before Enabling the Module

Step 1: Program the prescaler to obtain an approximately 12-MHz I^2C module clock (I2C_PSC = x; this value is to be calculated and is dependent on the system clock frequency).

Step 2: Program the I^2C clock to obtain 100K bps or 400K bps (I2C_SCLL = x and I2C_SCLH = x; these values are to be calculated and are dependent on the system clock frequency).

Step 3: Configure its own address (I2C_OA = x).

Step 4: Take the I^2C module out of reset (I2C_CON:I2C_EN = 1).

Initialization Procedure

Step 1: Configure the I^2C mode register (I2C_CON) bits.

Step 2: If using 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.

Configure Slave Address and Data Counter Registers

In master mode, configure the slave address (I2C_SA = x) and the number of bytes associated with the transfer (I2C_CNT = x).

Initiate a Transfer

Poll the bus busy (BB) bit in the I^2C status register (I2C_STAT). If it is cleared to 0 (bus not busy), configure START/STOP (I2C_CON:STT / I2C_CON:STP) condition to initiate a transfer.

Poll Receive Data

Poll the receive data ready interrupt flag bit (RRDY) in the I^2C status register (I2C_STAT). Use the RRDY interrupt or the DMA to read the receive data in the data receive register (I2C_DATA).

Poll Transmit Data

Poll the transmit data ready interrupt flag bit (XRDY) in the I^2C status register (I2C_STAT). Use the XRDY interrupt or the DMA to write data into the data transmit register (I2C_DATA).

2.9 Interrupt Subroutines

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.

2.10 Flow Diagrams

Figure 25. Setup Procedure

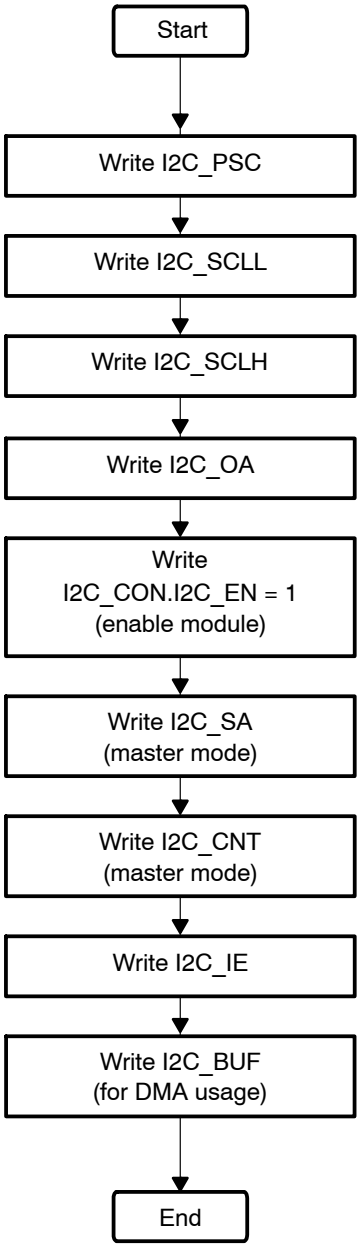


Figure 26. Master Transmitter Mode, Polling

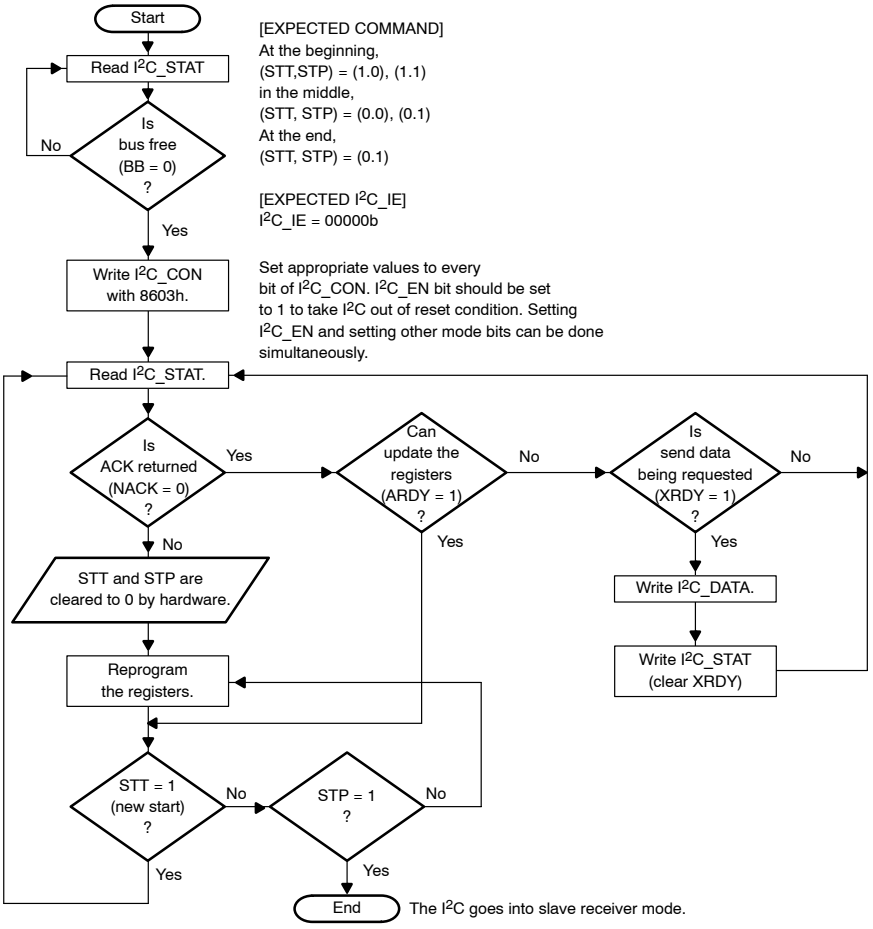


Figure 27. Master Receiver Mode, Polling

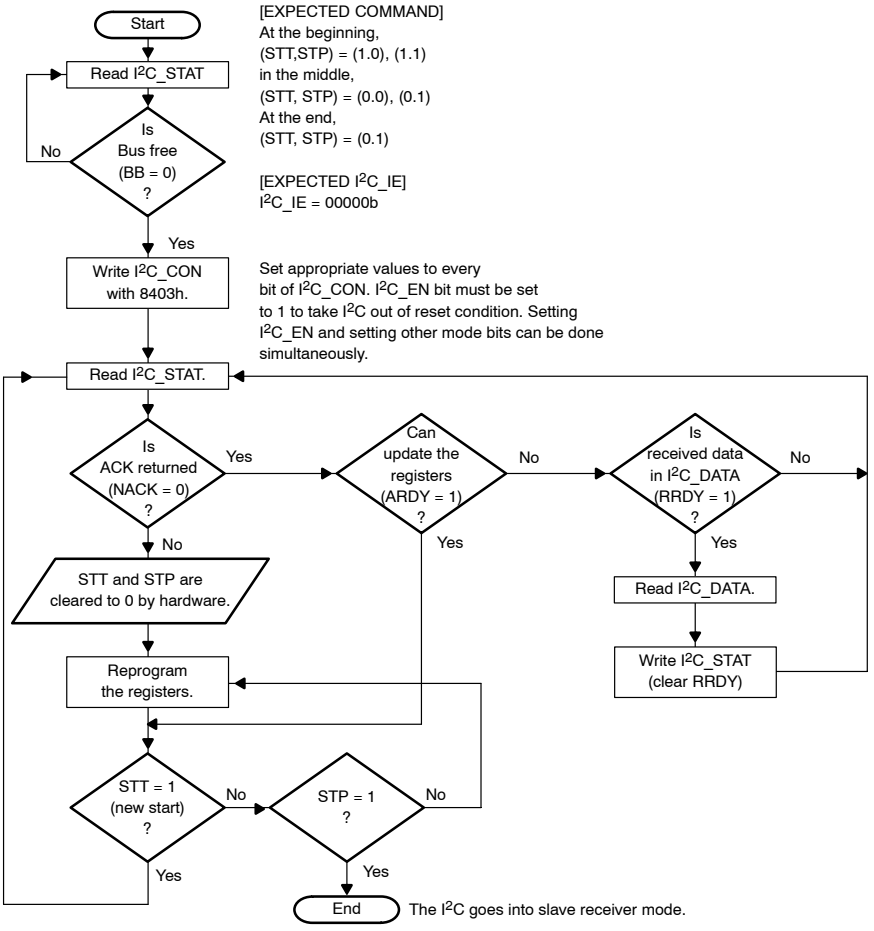


Figure 28. Master Transmitter Mode, Interrupt

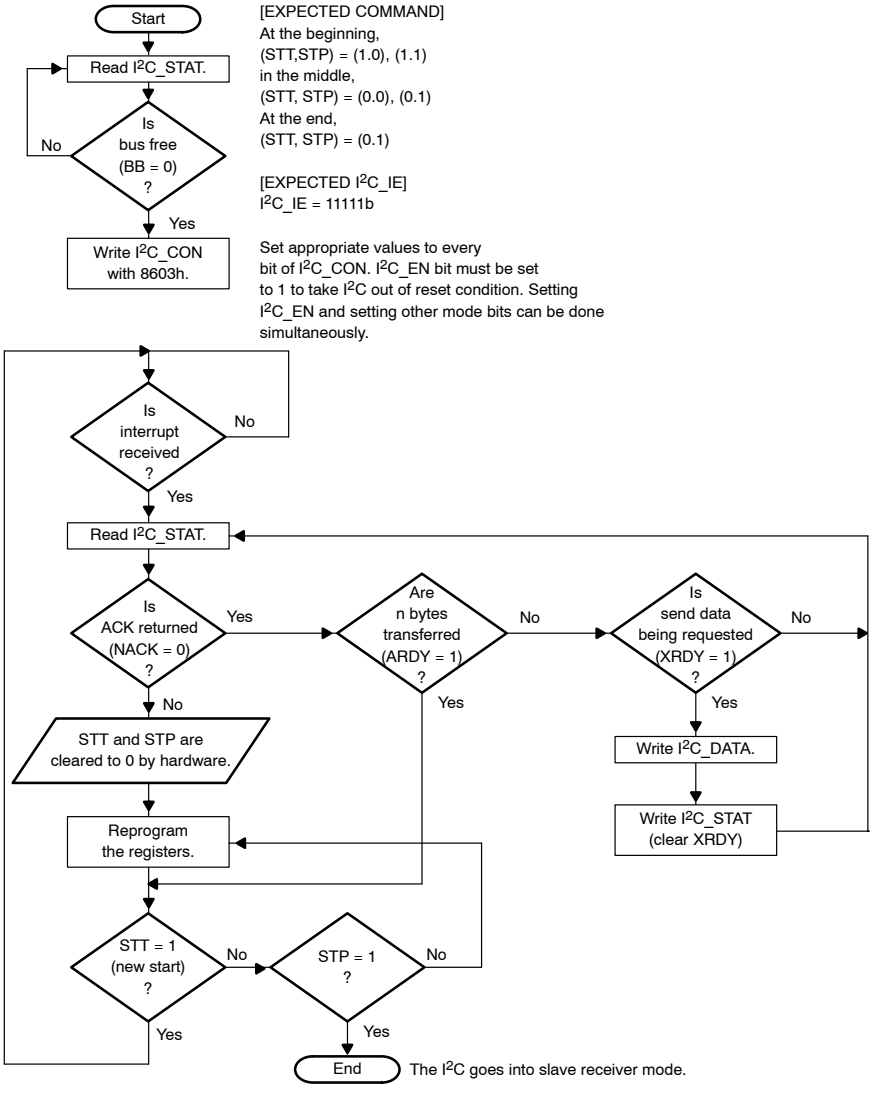


Figure 29. Master Receiver Mode, Interrupt

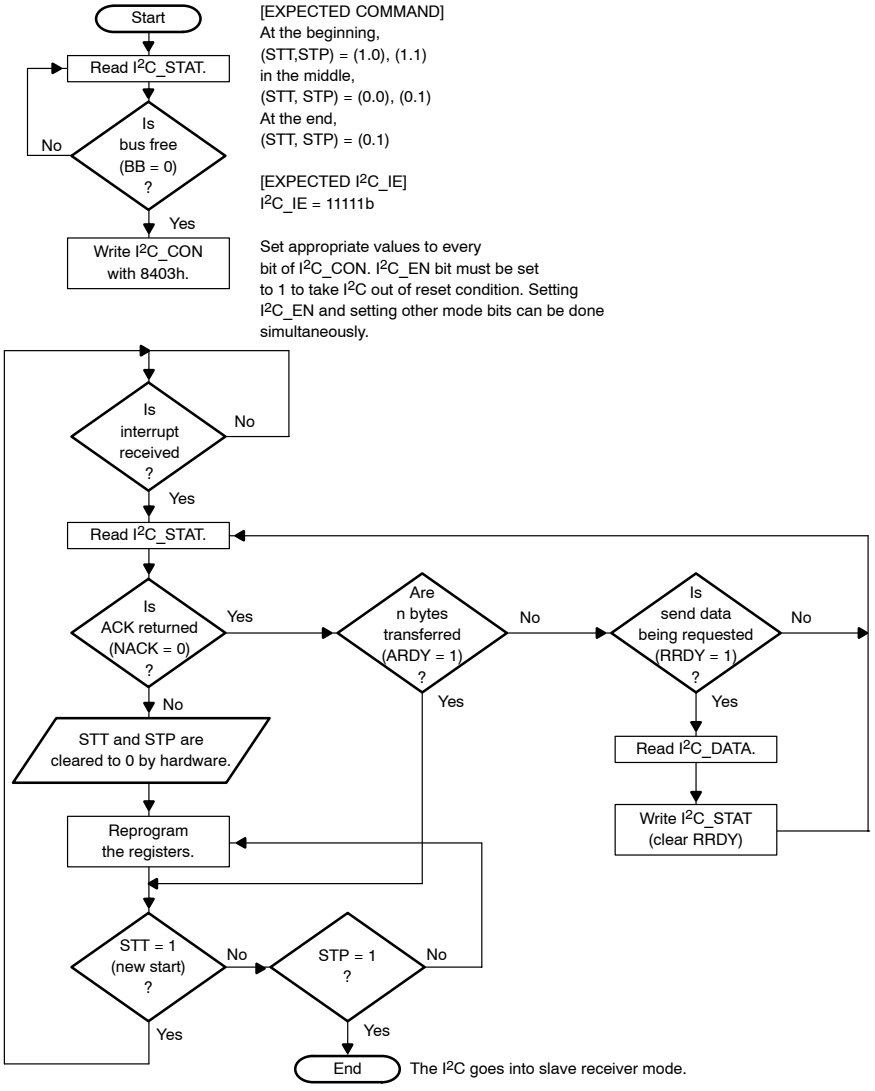


Figure 30. Master Transmitter Mode, DMA

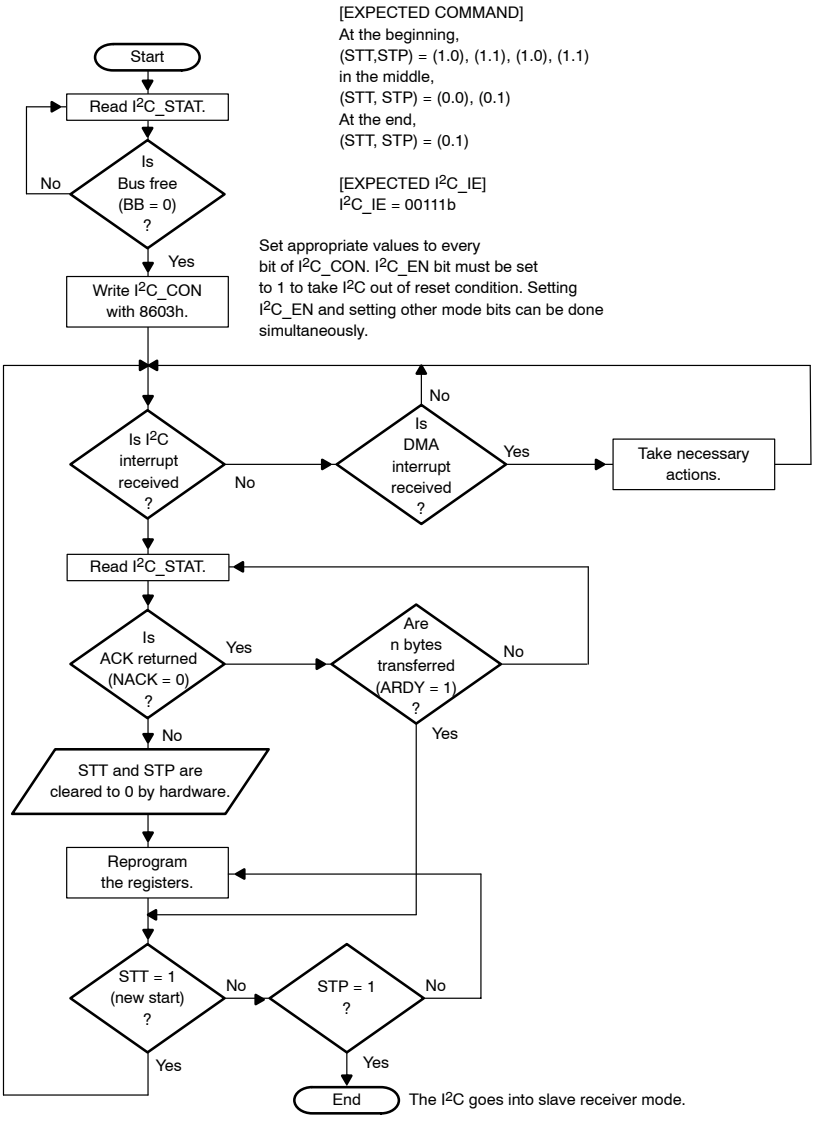


Figure 31. Master Receiver Mode, DMA

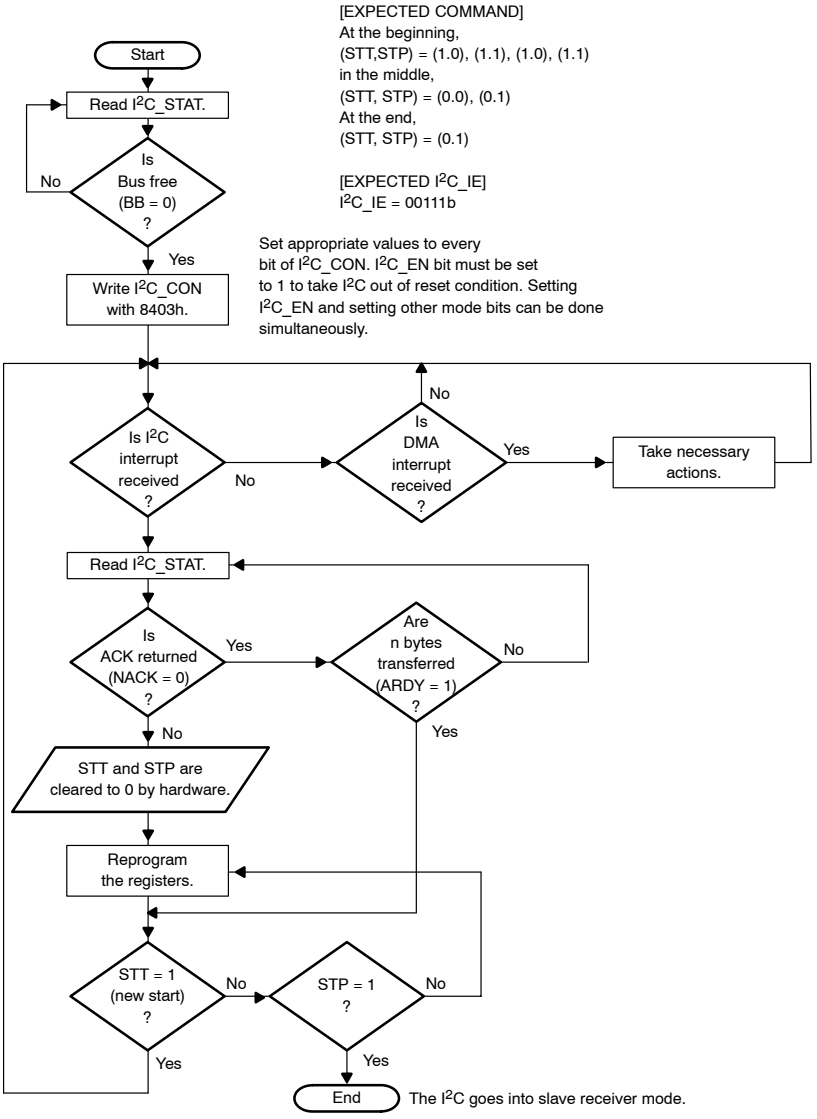


Figure 32. Slave Transmitter/Receiver Mode, Polling

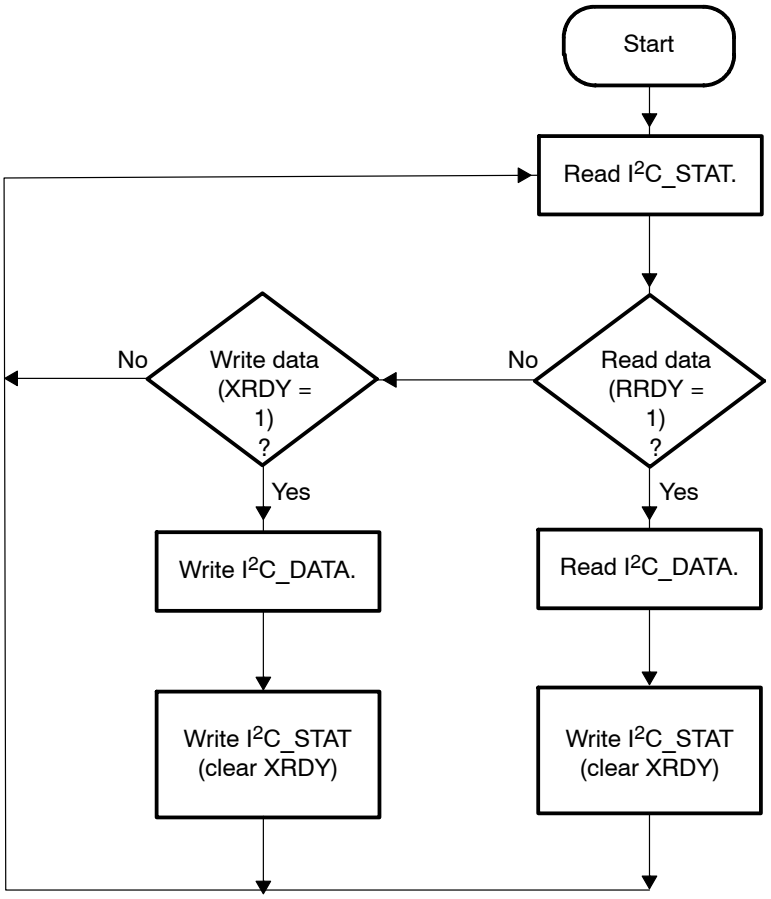
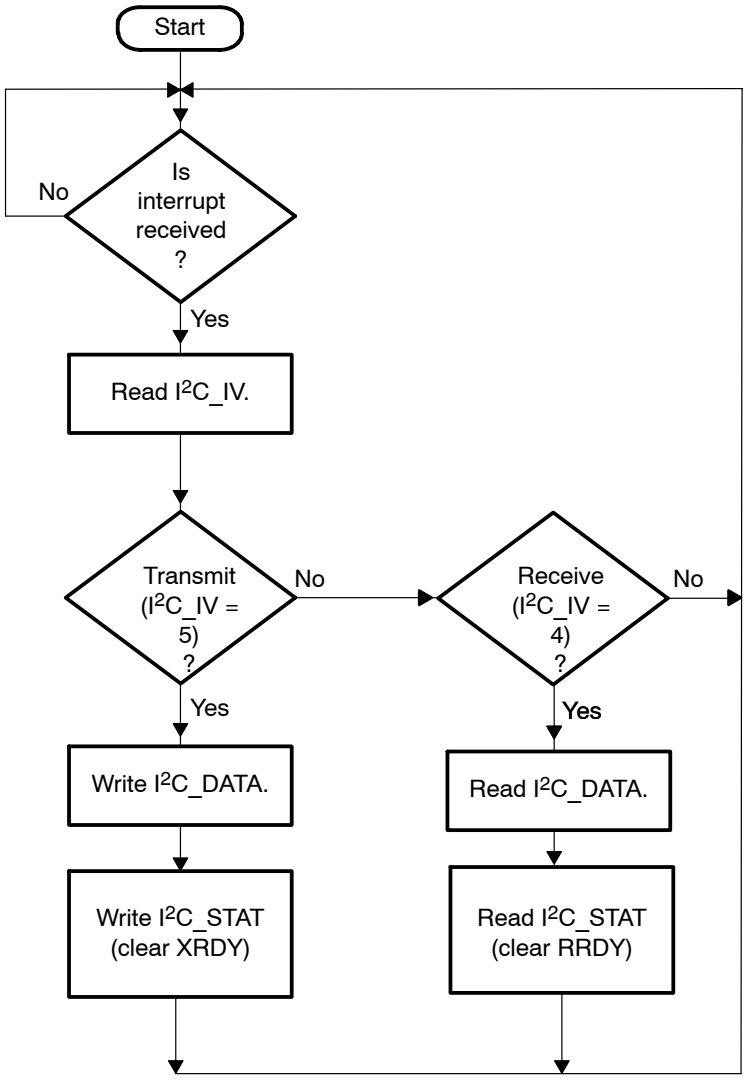


Figure 33. Slave Transmitter/Receiver Mode, Interrupt

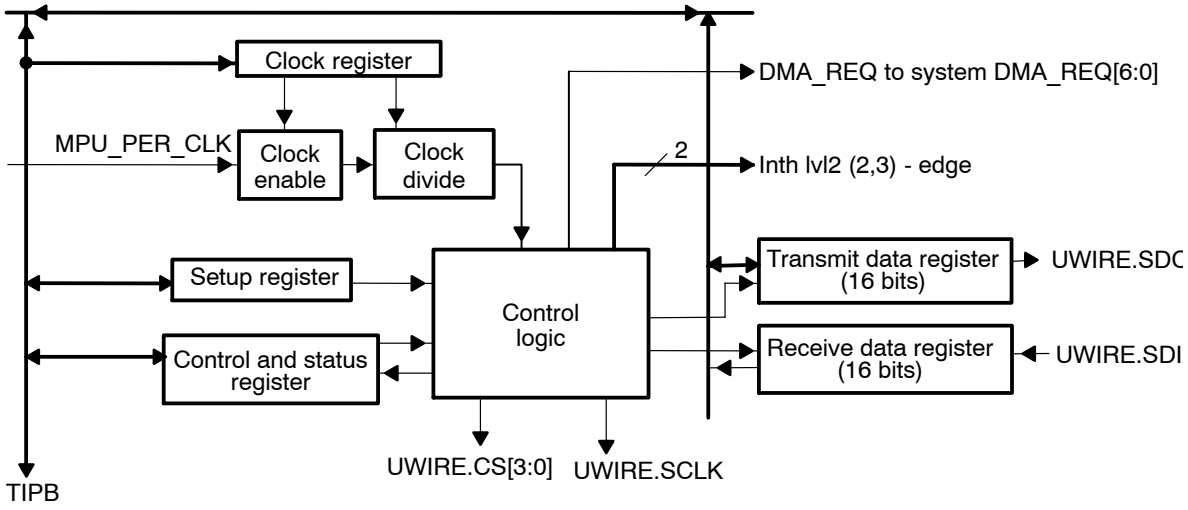


3 MicroWire Interface

This serial synchronous interface can drive up to four serial external components. For the external devices, this interface is compatible with the µWire standard and is seen as the master (see Figure 34).

A transmit DMA mode is available.

Figure 34. Block Diagram



3.1 MicroWire Registers

Start address in the peripheral range (hex): FFFB:3000

Table 38 lists the MicroWire registers. Table 39 through Table 46 describe the individual registers.

Table 38. MicroWire Registers

Register	Description	R/W	Size	Address	Offset
TDR	Transmit data	W	16 bits	FFFB:3000	0x00
RDR	Receive data	R	16 bits	FFFB:3000	0x00
CSR	Control and status	R/W	16 bits	FFFB:3000	0x04
SR1	Setup 1	R/W	16 bits	FFFB:3000	0x08
SR2	Setup 2	R/W	16 bits	FFFB:3000	0x0C
SR3	Setup 3	R/W	16 bits	FFFB:3000	0x10
SR4	Setup 4	R/W	16 bits	FFFB:3000	0x14
SR5	Setup 5	R/W	16 bits	FFFB:3000	0x18

Table 39. Transmit Data Register (TDR)

Base Address = 0xFFFB 3000, Offset = 0x00			
Bit	**Name**	**Function**	**Reset**
15:0	TD	Data to transmit	Undefined

Note: MSB (bit 15) is the first transmitted bit.

Whatever its size, the word is aligned on the most-significant bit (MSB) side.

Table 40. Receive Data Register (RDR)

Base Address = 0xFFFB 3000, Offset = 0x00			
Bit	**Name**	**Function**	**Reset**
15:0	RD	Received data	Undefined

Note: LSB (bit 0) is the last received bit.

Whatever its size, the word is aligned on the least-significant bit (LSB) side.

Table 41. Control and Status Register (CSR)

Base Address = 0xFFFB 3000, Offset = 0x04			
Bit	**Name**	**Function**	**Reset**
15	RDRB	RDRB bit at 1 indicates that the receive (RDR) is full. When the controller reads the content of the RDR, this bit is cleared. This bit is read only.	0
14	CSRB	CSRB bit at 0 indicates that the control and status (CSR) is ready to receive new data. After starting a µWire transfer with the CSR, this bit is set to 1. When the corresponding action has been done, CSRB is reset. This bit is controlled by a µWire internal state machine running on the F_INT internal clock (12 MHz/N). If the CSR is read just after being written, and the MPU is running at high frequency (60 MHz or 120 MHz, for instance) compared to the internal clock, the CSRB status bit may still be low for the first read access. The CSRB latency is 0 if the transfer was initiated by modifying the CS_CMD bit, but it can be 0 – 3 cycles if initiated by the START bit. Suggested workarounds are a) to have a few NOPs between initiating a µWire transfer and checking CSRB status or, b) to check that CSRB first has a high value on an initial read before it goes low on a subsequent read. This bit is read only.	0

Table 41. Control and Status Register (CSR) (Continued)

Base Address = 0xFFFB 3000, Offset = 0x04			
Bit	**Name**	**Function**	**Reset**
13	START	1: Start a write and/or a read process. This bit is automatically reset by internal logic when a write or a read process is activated. Send NB_BITS_WR bits (contained in TDR) to the serial output DO. If NB_BITS_WR is equal to zero, then the write process is not started. Receive NB_BITS_RD bits from the serial input DI and store them in RDR.	0
12	CS_CMD	1: Set the chip-select of the selected device to its active level.	0
11:10	INDEX	Index of the external device 00: CS0 01: CS1 10: CS2 11: CS3	Undefined
9:5	NB_BITS_WR	Number of bits to transmit	Undefined
4:0	NB_BITS_RD	Number of bits to receive	Undefined

Table 42. Setup Register 1 (SR1)

Base Address = 0xFFFB 3000, Offset = 0x08			
Bit	**Name**	**Function**	**Reset**
11	CS1_CHK	Idem CS0_CHK. Used when the CS1 is selected.	0xX (undefined)
10:9	CS1_FRQ	Defines the frequency of the serial clock SCLK when CS1 is selected 00 : F_INT/2 01 : F_INT/4 10 : F_INT/8 11 : undefined	0xX (undefined)
8	CS1CS_LVL	Defines the active level of the CS1 chip-select.	0x0
7	CS1_EDGE_WR	Idem CS0_EDGE_WR when CS1 is selected.	0xX (undefined)
6	CS1_EDGE_RD	Idem CS0_EDGE_RD when CS1 is selected.	0xX (undefined)

Note: Content of this register must not be changed when a read or write process is running.

Table 42. Setup Register 1 (SR1) (Continued)

Base Address = 0xFFFB 3000, Offset = 0x08			
Bit	**Name**	**Function**	**Reset**
5	CS0_CHK	Before activating a write process, checks if external device is ready. 0: No check is done and the write process is immediately executed. 1: If DI signal is low, the interface considers the external component busy; if DI is high, the interface considers the first external component ready and starts the write process. Used when CS0 is selected.	Undefined
4:3	CS0_FRQ	Defines the frequency of the serial clock SCLK when CS0 is selected (F_INT is the frequency of the internal clock). 00: F_INT/2 01: F_INT/4 10: F_INT/8 11: Undefined	Undefined
2	CS0CS_LVL	Defines the active level of the chip-select by CS0	0
1	CS0_EDGE_WR	When CS0 is selected, defines the active edge of the serial clock SCLK used to write data to the serial input D0. (Output data is generated on this edge) 0: Falling (serial clock not inverted) 0: Rising (when serial clock inverted) 1: Rising (serial clock not inverted) 1: Falling (when serial clock inverted)	Undefined
0	CS0_EDGE_RD	When CS0 is selected, defines the active edge of the serial clock SCLK used to read data from the serial input DI. (Input data is strobed on this edge) 0: Falling (serial clock not inverted) 0: Rising (when serial clock inverted) 1: Rising (serial clock not inverted) 1: Falling (when serial clock inverted)	Undefined

Note: Content of this register must not be changed when a read or write process is running.

Table 42 sets up the serial interface for the first and second external components.

Table 43. Setup Register 2 (SR2)

Base Address = 0xFFFB 3000, Offset = 0x0C			
Bit	**Name**	**Function**	**Reset**
11	CS3_CHK	Same as CS0_CHK. Used when CS3 is selected.	Undefined
10:9	CS3_FRQ	Defines the frequency of the serial clock SCLK when CS3 is selected 00: F_INT/2 01: F_INT/4 10: F_INT/8 11: Undefined	Undefined
8	CS3CS_LVL	Defines the active level of the CS3 chip-select	0
7	CS3_EDGE_WR	Same as CS0_EDGE_WR when CS3 is selected	Undefined
6	CS3_EDGE_RD	Same as CS0_EDGE_RD when CS3 is selected	Undefined
5	CS2_CHK	Idem CS0_CHK. Used when the CS2 is selected.	0xX (undefined)
4:3	CS2_FRQ	Defines the frequency of the serial clock SCLK when CS2 is selected (F_INT is the frequency of the internal clock): 00 : F_INT/2 01 : F_INT/4 10 : F_INT/8 11 : Undefined	0xX (undefined)
2	CS2CS_LVL	Defines the active level of the CS2 chip-select.	0x0
1	CS2_EDGE_WR	Idem CS0_EDGE_WR when CS2 is selected.	0xX (undefined)
0	CS2_EDGE_RD	Idem CS0_EDGE_RD when CS2 is selected.	0xX (undefined)

Note: Content of this register must not be changed when a read or write process is running.

Table 43 sets up the serial interface for the first and second external components.

Table 44. Setup Register 3 (SR3)

This register sets up the serial interface for the internal clock.

Base Address = 0xFFFB 3000, Offset = 0x10			
Bit	**Name**	**Function**	**Reset**
2:1	CK_FREQ	Defines the frequency of the internal clock, F_INT, when CLK_EN = 1. All the internal logic is controlled by F_INT (F is the frequency of the external input clock). 00: F/2 01: F/4 10: F/7 11: F/10	00
0	CLK_EN	Switch off the clock if 0. Switch on the clock if 1.	0

Note: Content of this register must not be changed when a read or write process is running.

Table 45. Setup Register 4 (SR4) (R/W)

This register sets up the serial clock polarity.

Base Address = 0xFFFB 3000, Offset = 0x14			
Bit	**Name**	**Function**	**Reset**
0	CLK_IN	Serial clock is not inverted if 0. Serial clock is inverted if 1.	0

Note: Content of this register must not be changed when a read or write process is running.

Table 46. Setup Register 5 (SR5) (R/W)

Base Address = 0xFFFB 3000, Offset = 0x18			
Bit	**Name**	**Function**	**Reset Value**
3	CS_TOGGLE_TX_EN	CS_TOGGLE_TX_EN is possible only in autotransmit mode. When in autotransmit mode with CS_TOGGLE_TX_EN inactive, the CS does not go to its active level automatically. Control the CS with the CS CMD bit of the control and status register (CSR) in the software. CS_toggle transmit mode is disabled if 0. CS_toggle transmit mode is enabled if 1.	0
2	AUTO_TX_EN	In autotransmit mode, the CS_CMD and START bits of the control and status register (CSR) are not used. A hardware state machine detects a TXD write and automatically sets the programmed CS to its active value, and then starts the transmission. The CS CMD and the START bits in the control and status register (CSR) are not updated during autotransmit. Autotransmit mode is disabled if 0. Autotransmit mode is enabled if 1.	0
1	IT_EN	In IT mode, an interrupt is generated each time a word has been transferred or received. This interrupt is a low-level interrupt. A status register (IST) allows the CPU to know which interrupt (receive and/or transmit) occurred. IT mode is disabled if 0. IT mode is enabled if 1.	0
0	DMA_TX_EN	DMA transmit mode is disabled if 0. DMA transmit mode is enabled if 1.	0

Note: Content of this register must not be changed when a read or write process is running.

Set up the DMA, IT, AUTO_TX, and CS_TOGGLE modes in this register.

In DMA mode, a DMA request is initiated each time a transmission slot is available.

The maximum word size in DMA mode is 16 bits.

Note:

You cannot use another CS in normal or DMA modes when a DMA mode is active on one specific CS.

Note:

To use the µWire in DMA transmit mode, DMA_EN and AUTO_TX_EN must be enabled, and IT_EN is best disabled. The AUTO_TX_EN can be active when DMA_EN is disabled.

3.2 Protocol Description

The serial port must be configured in order to use the setup registers.

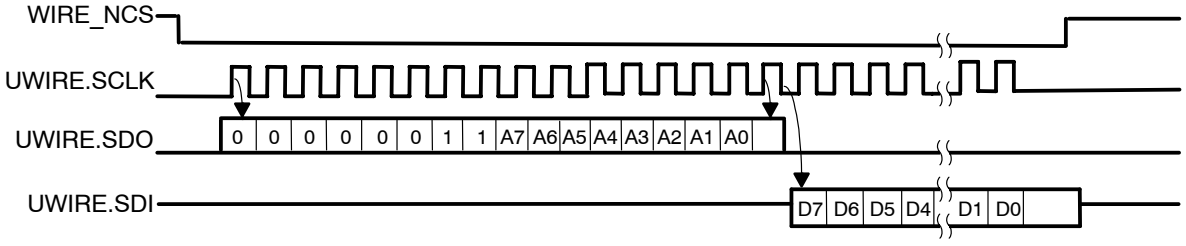
This interface drives only one device at a given time. Therefore, the chip-select of the selected device must be set to its active level before starting any read or write process.

After loading the transmit data register (TDR), the write process is activated by setting the START bit to 1 and by writing a value different from zero to the NB_BITS_WR field.

A read process is always simultaneous with a write process, which means that at every serial clock (SCLK) cycle, data is read. After having finished a write process (if necessary), a number (defined by NB_BITS_RD) of SCLK cycles is generated to allow storage of data from the serial input DI.

The transmitted data word is shifted out on the rising or falling edge of the serial clock, according to the value of the *_EDGE_WR bits of the setup registers. The received data word is shifted in on the falling or rising edge of the serial clock, according to the value of the *_EDGE_RD bits of the setup registers. When *_EDGE_WR and *_EDGE_RD bits have the same value, it is assumed that the device behavior is the one shown in Figure 35. Otherwise, the required behavior of the external device is as shown in Figure 36.

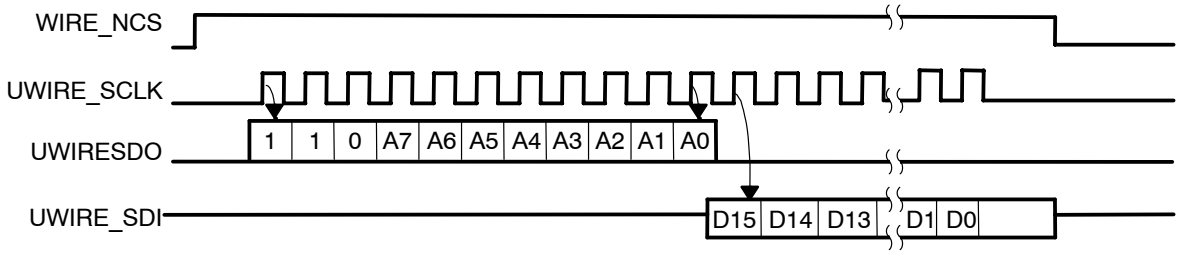
Figure 35. Behavior of an X25C02 EEPROM Read Cycle



On the DO line, data is generated from the µWire interface on SLCK falling edge and read by the EEPROM interface on SCLK rising edge.

On the DI line, data is generated from the EEPROM interface on SCLK falling edge and read by the µWire interface on SCLK falling edge.

Figure 36. Behavior of an XL93LC66 EEPROM Read Cycle



On the DO line, data is generated from the µWire interface on SLCK falling edge and read by the EEPROM interface on SCLK rising edge.

On the DI line, data is generated from the EEPROM interface on SCLK rising edge and read by the µWire interface on SCLK rising edge.

3.3 Example of Protocol Using a Serial EEPROM (XL93LC66)

Set up the interface by writing the following values in setup 1 register (SR1):

- ❏ CS_EDGE_RD = 1
- ❏ CS_EDGE_WR = 0
- ❏ CSCS_LVL = 1
- ❏ CS_FRQ = 00
- ❏ CS_CHK = 1

In this example, only two cycles (read and write) are described.

3.3.1 Read Cycle

1) Set the following fields of the control and status register (CSR):
   - NB_BITS_RD: 0
   - NB_BITS_WR: 0
   - INDEX: 00
   - CS_CMD: 1
   - START: 0
2) Load the transmit data register (TDR) with:
   - 1 1 0 A7 A6 A5 A4 A3 A2 A1 A0 x x x x x: *Don't care*
   - A7 ... A0: Address of the selected memory register

3) Wait for the CSRB bit of CSR to be reset.

4) Set the following fields of the control and status register (CSR):
   - NB_BITS_RD: 16 (decimal)
   - NB_BITS_WR: 11 (decimal)
   - INDEX: 00
   - CS_CMD: 1
   - START: 1

5) Wait until CSRB = 0 and RDRB = 1 (status bits of CSR).

6) Read the content of receive data register (RDR).

7) To continue reading data external component, the EEPROM, go to step 8. Else go to step 9.

8) Set the following fields of the control and status register (CSR):
   - NB_BITS_RD: 16 (decimal)
   - NB_BITS_WR: 0 (decimal)
   - INDEX: 00
   - CS_CMD: 1
   - START: 1
   - Go to step 5.

9) Set the following fields of the control and status register (CSR):

❑ INDEX: 00
   - CS_CMD: 0
   - START: 0

3.3.2 Write Cycle

1) Set the following fields of the control and status register (CSR):
   - NB_BITS_RD: 0
   - NB_BITS_WR: 0
   - INDEX: 00
   - CS_CMD: 1
   - START: 0

2) Load the transmit data register (TDR) with:
   - 1 0 1 A7 A6 A5 A4 A3 A2 A1 A0 x x x x x x: *Don't care*
   - A7 ... A0: Address of the selected memory register
3) Wait for the CSRB bit of the control and status register (CSR) to be reset.
4) Set the following fields of the control and status register (CSR):
   - NB_BITS_RD: 0
   - NB_BITS_WR: 11 (decimal)
   - INDEX: 00
   - CS_CMD: 1
   - START: 1
5) Wait for the CSRB bit of the control and status register (CSR) to be reset.
6) Load the transmit data register (TDR) with:
   - D15 D14 ... D0
   - D15 ... D0: Data
7) Set the following fields of the control and status register (CSR):
   - NB_BITS_RD: 0
   - NB_BITS_WR: 16 (decimal)
   - INDEX: 00
   - CS_CMD 1
   - START: 1
8) Wait for the CSRB bit of CSR to be reset.
9) Set the following fields of the control and status register (CSR):
   - INDEX: 00
   - CS_CMD: 0
   - START: 0

3.4 Example of Protocol Using an LCD Controller (COP472-3)

Set up the interface by writing in setup 1 register (SR1) the following value:

- ❑ CS_EDGE_RD = 1
- ❑ CS_EDGE_WR = 0
- ❑ CSCS_LVL = 0
- ❑ CS_FRQ = 10
- ❑ CS_CHK = 0

This example describes a loading sequence to drive a four-digit display.

3.4.1 Loading Sequence

1) Set the following fields of the control and status register (CSR):
   - NB_BITS_RD: 0
   - NB_BITS_WR: 0
   - INDEX: 01
   - CS_CMD: 1
   - START: 0
2) Wait for the CSRB bit of the control and status register (CSR) to be reset.
3) Load the transmit data register (TDR) with:
   - D7d1...D0d1 D7d2...D0d2 D7d1...D0d1: Data for digit 1
   - D7d2...D0d2: Data for digit 2
4) Set the following fields of the control and status register (CSR):
   - NB_BITS_RD: 0
   - NB_BITS_WR: 16 (decimal)
   - INDEX: 01
   - CS_CMD: 1
   - START: 1
5) Wait for the CSRB bit of the control and status register (CSR) to be reset.
6) Load the transmit data register (TDR) with:
   - D7d3...D0d3 D7d4...D0d4 D7d3...D0d3: Data for digit 3
   - D7d4...D0d4: Data for digit 4
7) Set the following fields of the control and status register (CSR):
   - NB_BITS_RD: 0
   - NB_BITS_WR: 16 (decimal)
   - INDEX: 01
   - CS_CMD: 1
   - START: 1
8) Wait for the CSRB bit of the control and status register (CSR) to be reset.
9) Load the transmit data register (TDR) with:
   - D7...D0 x x x x x x x x: *Don't care*
   - D7...D0: Data for special segment and control function

10) Set the following fields of the control and status register (CSR):
    - NB_BITS_RD: 0
    - NB_BITS_WR: 8 (decimal)
    - INDEX: 01
    - CS_CMD: 1
    - START: 1
11) Wait for CSRB to go low, which indicates the CSR is ready to receive new data. It is advised that you read the bit before and after every write access to CSR to check the status.
12) Set the following fields of the control and status register (CSR):
    - INDEX: 01
    - CS_CMD: 0
    - START: 0

3.5 Example of Protocol Using Autotransmit Mode

The setup 5 register (SR5) controls the autotransmit mode. The following example configures μWire for a read access on CS0 with serial clock out inverted, CS autotoggle enabled, DMA request disabled, and interrupt enabled:

1) SR5 = DMA_TX_EN: 0

   IT_EN: 1

   AUTO_TX_EN: 1

   CS_TOGGLE_TX_EN: 1

2) SR1 = CS0_EDGE_RD: 0

   CS0_EDGE_WR: 1

   CS0CS_LVL: 0

   CS0_FREQ: 00

   CS0_CHK: 1

Note:

Data out is latched on the falling edge of the serial clock. Data in is sampled on the rising edge.

3) SR3 = CLK_EN: 1

   CK_FREQ: 00 (must wait for 1 ARMXOR_CK + 1 F_INT cycle before any other register access)

4) SR4 = CLK_IN: 1

5) Set the following fields of the control and status register (CSR):

- ❑ NB_BITS_RD: 5
- ❑ NB_BITS_WR: 7
- ❑ INDEX: 00
- ❑ CS_CMD: 0
- ❑ START: 0

6) Wait for the CSRB = 0 of the control and status register (CSR).

7) Load the transmit data register (TDR) with:

   - A6 A5 A4 A3 A2 A1 A0 x x x x x x x x x: *Don't care*
   - A6 ... A0: Address of the selected memory register

   Transfer is automatically started.

8) Wait until CSRB = 0 and RDRB = 1 (status bits of CSR).

9) Read the content of receive data register (RDR).

10) To continue reading data external component, go to 5 else go to 11.

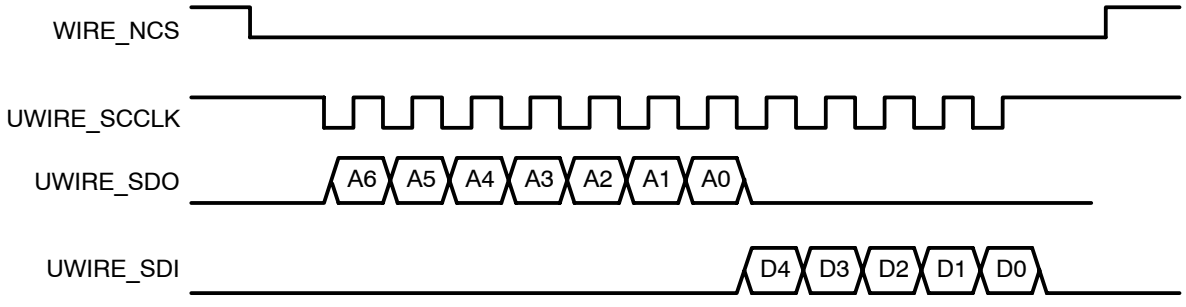
11) 20

12) Release autotransmit mode: SR5 = AUTO_TX_EN: 0.

13) END

The corresponding behavior of the serial interface is described in Figure 37.

Figure 37. Read Cycle in Autotransmit Mode



3.6 Example of Autotransmit Mode With DMA Support

The setup 5 register (SR5) controls the autotransmit mode and DMA mode. The following example configures μWire for a 16-bit write access on CS1 with serial clock out not inverted, CS autotoggle enabled, DMA request enabled, and interrupt disabled:

1) Set up and enable the DMA channel.

2) Program the configuration registers SR1, SR3, and SR4.

3) Check CSRB status to ensure that the peripheral is ready to receive (low).

4) Program the control and status register (CSR) as follows:

- ❏ NB_BITS_RD = 0
- ❏ NB_BITS_WR = 16
- ❏ INDEX = 00
- ❏ CS_CMD: = 1
- ❏ START = 0

5) Write to the setup register (SR5) to configure and initiate the transfer:

- ❏ DMA_TX_EN = 1
- ❏ IT_EN = 0
- ❏ AUTO_TX_EN = 1
- ❏ CS_TOGGLE_TX_EN = 1 (Note that in AUTO TX mode, setting the DMA_TX_EN bit to 1 starts the transfer)

6) When the DMA transfer is complete, check the status of CSRB to determine whether μWire has finished the serial data transfer.

7) Write to the setup register (SR5) to disable DMA and AUTO TX mode:

- ❏ DMA_TX_EN = 0
- ❏ IT_EN = 0
- ❏ AUTO_TX_EN = 0
- ❏ CS_TOGGLE_TX_EN = 0

Using Autostart and Autotoggle CS Mode

You must wait for a minimum of 3 x F_INT clock cycles after the end of transfer (transition 1 to 0 detected on CSRB) before setting the SR3 register to turn off the internal clock.

4 Multichannel Serial Interfaces

Multichannel serial interfaces (MCSIs) have multichannel transmission capability. MCSIs expand the parallel interface of a DSP to connect to external devices such as codecs and GSM system simulators.

The two public MCSIs on the device provide full duplex transmission and master or slave clock control. All transmission parameters are configurable to cover the maximum number of operating conditions:

- Master or slave clock control (transmission clock and frame synchronization pulse)
- Programmable transmission clock frequency
- Single-channel or multichannel (x16) frame structure
- Programmable word length: 3 to 16 bits
- Full-duplex transmission
- Programmable frame configuration
  - Continuous or burst transmission
  - Normal or alternate framing
  - Normal or inverted frame polarity
  - Short or long frame pulse
  - Programmable oversize frame length
  - Programmable frame length
- Programmable interrupt occurrence time (TX and RX)
- Error detection with interrupt generation on wrong frame length
- DMA support for both TX and RX data transfers

4.1 Communication Protocol

4.1.1 Configuration Parameters

The configuration parameters can be modified only if the MCSI is disabled (CONTROL_REG[0] = 0).

Slave/Master Control

Using the control bit, the interface can be configured in one of two ways:

- In master mode, with the transmission clock and the frame synchronization pulse generated by the interface
- In slave mode, with the transmission clock and the frame synchronization pulse generated from an external device

Control bit:
MAIN_ PARAMETERS _REG(6) = MCSI_MODE
1: Master
0: Slave

Single-Channel/Multichannel

The frame structure can be either single-channel-based (one channel per frame), or multichannel-based with the number of channels fixed at 16.

Control bit:
MAIN_ PARAMETERS _REG(7) = MULTI
1: Multichannel
0: Single-channel

Short/Long Framing

The frame-synchronization pulse duration can be either short, with a pulse duration equal to the bit duration, or long, with a pulse duration equal to the channel duration.

The long frame is active only during transmission on channel 0.

Control bit:
MAIN_ PARAMETERS _REG(8) = FRAME_SIZE
1: Long
0: Short

Normal/Alternate Frame Synchronization

The frame-synchronization pulse position is either normal, with the frame-synchronization pulse starting one bit before channel 0, or alternates with the frame-synchronization pulse starting with the first bit of channel 0.

Control bit:
MAIN_ PARAMETERS _REG(9) = FRAME_POSITION
1: Alternate
0: Normal

Continuous/Burst Mode

The frame mode is either continuous with one frame-synchronization pulse at the first frame, or bursts with one frame-synchronization pulse at each frame.

Control bit:
MAIN_ PARAMETERS _REG(5) = FRAME_MODE
1: Continuous
0: Burst

Normal/Inverted Clock

The polarity of the clock can be either normal, with writing on the positive edge clock and reading on the negative edge clock , or inverted, with writing on the negative edge clock and reading on the positive edge clock.

Control bit:
MAIN_ PARAMETERS _REG(4) = CLOCK_POLARITY
1: Inverted
0: Normal

Normal/Inverted Frame Synchronization

The polarity of the frame-synchronization pulse can be either normal, with a positive pulse, or inverted, with a negative pulse.

Control bit:
MAIN_ PARAMETERS _REG(10) = FRAME_POLARITY
1: Inverted
0: Normal

Channel Used

To enable a channel in multimode, set bit n for the desired channel n.

Word Size

To choose the size of the word, set its size minus one into the main parameters registers.

Control bit:
MAIN_PARAMETERS_REG(3:0) = WORD_ SIZE
(2 <= WORD_ SIZE <= 15)

The MCSI transmits and receives the most significant bit first. For example, if the word_size equals 11, the upper 12 bits of the TX registers are transmitted, the upper 12 bits of the RX registers contain the received data, and the lower 4 bits are zeros.

Frame Size

To add any overhead bits at the end of each frame, set the number of desired overhead bits in the over_size_register.

Control bit:
OVER_CLOCK_REG(9:0) = OVER_CLK (0<=OVER_CLK <=1023)

Transmission Clock Frequency

In master mode, the clock frequency is derived from the 12-MHz master clock and can be programmed from 5.8 kHz to 6 MHz in increments of 83 ns.

Control bit:
CLOCK_FREQUENCY_REG(10:0) = CLK_FREQ
(2<=CLK_FREQ <= 2047)

with
($t_{CLK} = t_{12MHz} * CLK\_FREQ$)

4.1.2 Sample Setup for Communication μ-Law Interface Using Interrupts

MCSI Configuration

An example of communication μ-law interface setup using interrupts follows.

- ❏ DSP_Write(0x0000) = CONTROL_REG (disable MCSI for setup)
- ❏ DSP_Write(0x0007) = MAIN_PARAMETERS_REG (set up MCSI per configuration below)
  - ■ Bit 15-14 (00b): No DMA
  - ■ Bit 10 (0b): Positive polarity for frame
  - ■ Bit 9 (0b): Normal synchronization mode
  - ■ Bit 8 (0b): Short framing
  - ■ Bit 7 (0b): Single channel
  - ■ Bit 6 (0b): Slave mode
  - ■ Bit 5 (0b): Burst mode
  - ■ Bit 4 (0b): Positive edge for clock
  - ■ Bit 3-0 (0111b): 8-bit data
- ❏ DSP_Write(0x0700) = INTERRUPTS_REG (all interrupts are enabled)
- ❏ DSP_Write(0x0000) = OVER_CLOCK_REG
- ❏ DSP_Write(0x0001) = CONTROL_REG (start MCSI)

Transmit Data Loading (TX_INT ISR)

- ❏ DSP_Write = TX_REG

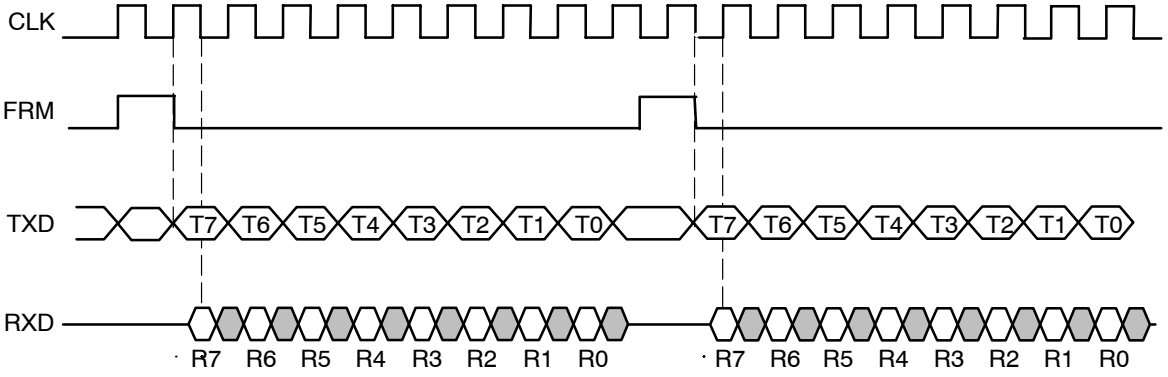
Received Data Loading (RX_INT ISR)

- ❏ DSP_Read = RX_REG

Stop MCSI

- ❏ DSP_Write(0x0000) = CONTROL_REG (disable MCSI clock)
- ❏ DSP_Write(0x0002) = CONTROL_REG (reset MCSI registers)

Figure 38. Communication μ-Law Interface Interrupts Waveform Example



4.1.3 Interface Management

Interrupts Generation

Three physical interrupts are available for real-time management of the MCSI by the DSP:

- ❏ RX_INT (data receive interrupt)
- ❏ TX_INT (data transmit interrupt)
- ❏ FERR_INT (frame duration error interrupt)

RX_INT, TX_INT, and FERR_INT are maskable with dedicated programmable control bits of the interrupt register INTERRUPTS_REG.

- ❏ RX_INT is masked when MASK_IT_RX = 0.
- ❏ TX_INT is masked when MASK_IT_TX = 0.
- ❏ FERR_INT is masked when MASK_IT_ERROR = 0.

Each interrupt is associated with a flag bit in the STATUS_REG register that is set to 1 when the interrupt is generated. To acknowledge the interrupt and release the corresponding physical signal, the DSP must write a 1 at the bit location in the status register. The following list provides interrupt/flag bit associations:

- ❏ RX_INT (RX_READY flag and acknowledge bit)
- ❏ TX_INT (TX_READY flag and acknowledge bit)
- ❏ FERR_INT (FRAME _ERROR flag and acknowledge bit)

Receive Interrupt

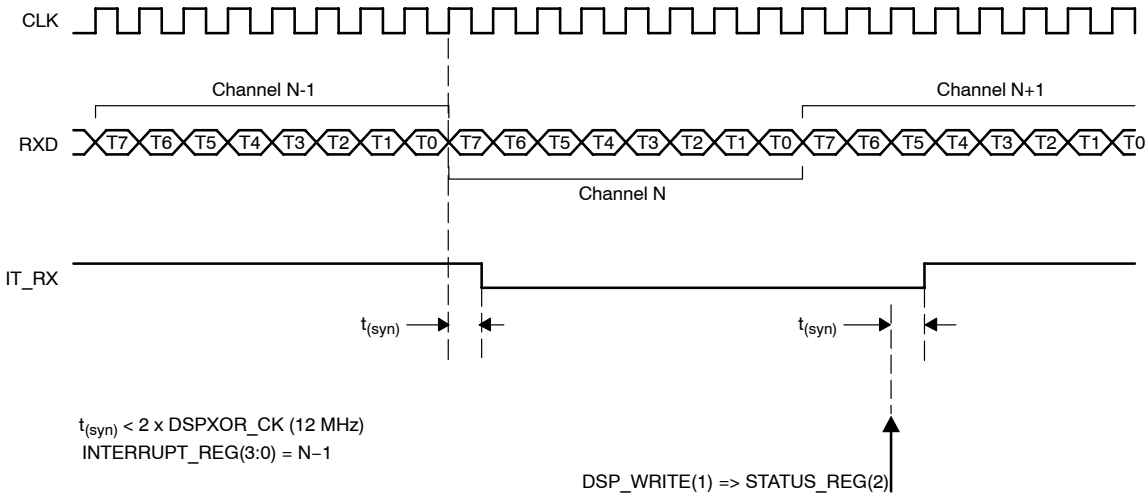
The receive interrupt is generated every frame after the completion of the reception of a data word:

- In single-channel mode, the interrupt is generated one half-clock period (plus a synchronization delay) after the reception of the word.
- In multichannel mode, the interrupt is generated one half-clock period (plus a synchronization delay) after the reception of the word of the channel whose number is defined by the NB_CHAN_IT_RX parameter of INTERRUPTS_REG register.

Note:

If MCSI is in slave mode, the clock must be driven after valid data reception until the interrupt is generated and must not be gated before then, because the interrupt is generated on the MCSI interface clock.

Figure 39. Receive Interrupt Timing Diagram

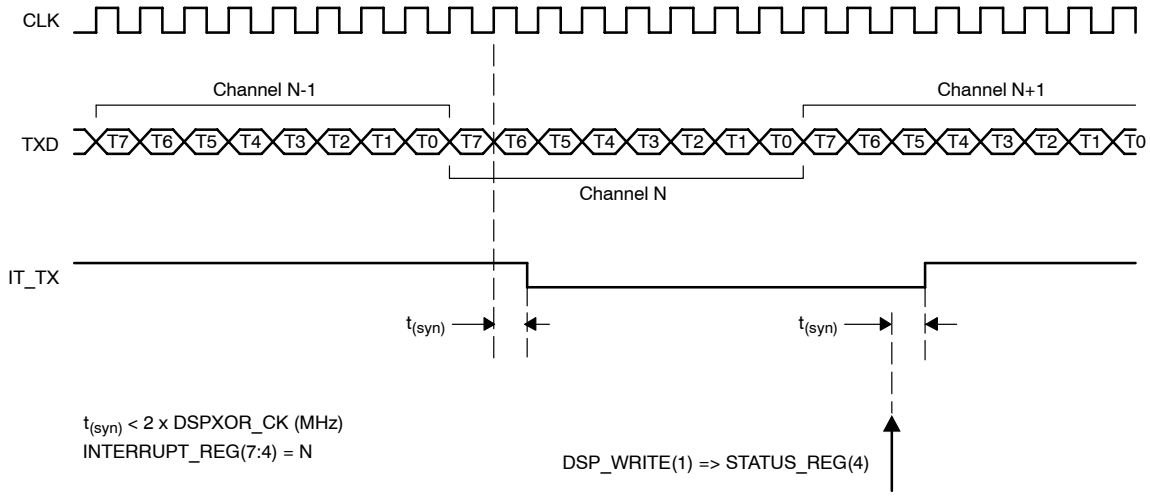


Transmit Interrupt

The transmit interrupt is generated every frame after the start of the transmission of a data word.

- In single-channel mode, the interrupt is generated one clock period after the beginning of the transmission of the word.
- In multichannel mode, the interrupt is generated one clock period after the transmission of the word of the channel whose number is defined by the NB_CHAN_IT_RX parameter of INTERRUPTS_REG register.

Figure 40. Transmit Interrupt Timing Diagram



Frame Duration Error Interrupt

The frame duration error interrupt is only generated when:

- ❏ The interface is configured in burst mode (CONTINUOUS = 0).
- ❏ The frame duration is smaller or longer than the expected value.

Namely, expected frame duration = [(channels number) * (word size)] + (over-size number) in clock periods units with over-size number defined in OVER_SIZE_REG register.

If the frame duration is longer than the expected value, then the interrupt is generated one clock period after the number of the over_size clock periods, as defined in OVER_CLOCK parameter.

If the frame duration is smaller than the expected value, then the interrupt is generated one clock period after the occurrence of the next frame pulse (first active edge).

Figure 41. Frame Duration Error—Too Many (Long)

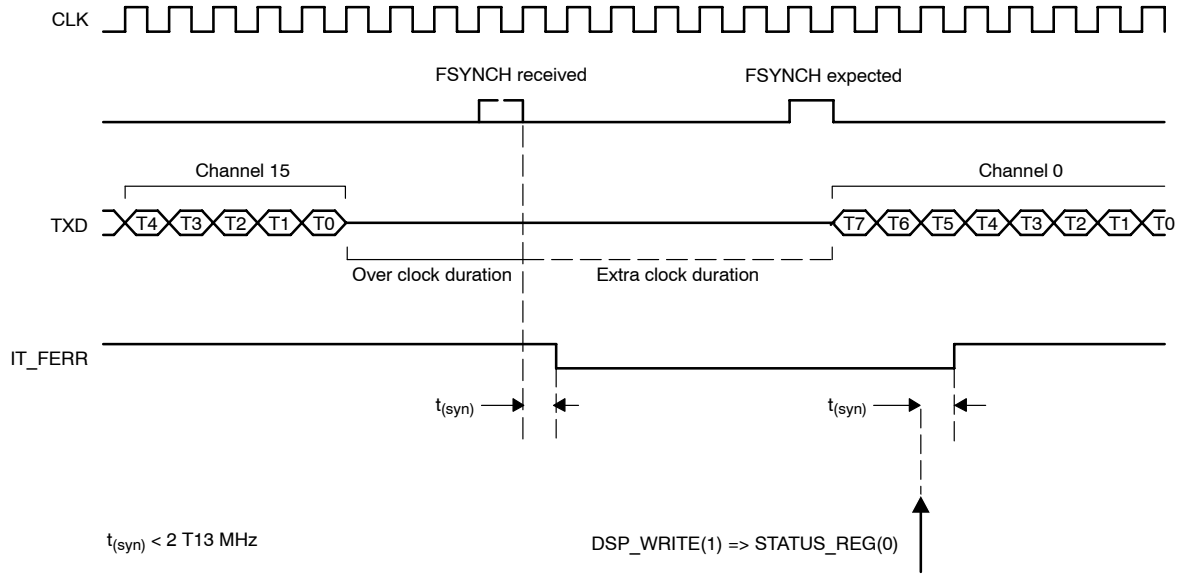
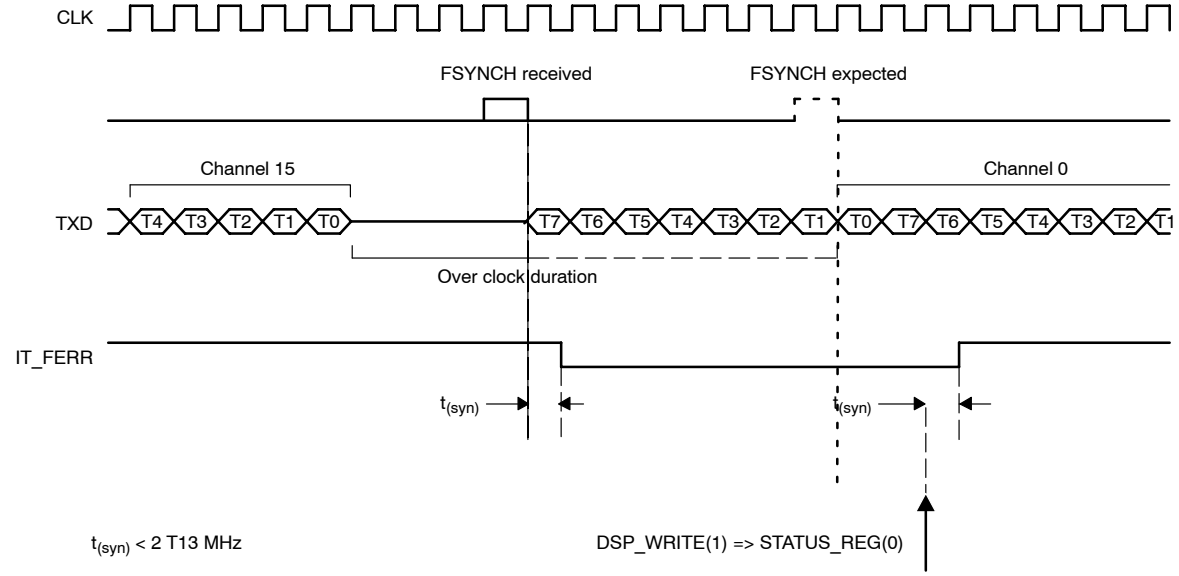


Figure 42. Frame Duration Error—Too Few (Short)



4.1.4 Interrupt Programming

At module reset, RX_INT, TX_INT, and FERR_INT are masked.

To validate an interrupt:

If in multichannel mode, the RX and TX interrupts can be configured to occur in a dedicated channel of the frame [1-16].

- DSP_WRITE**(**channel_nb**)** = INTERRUPTS_REG(3:0) for RX_INT
- INTERRUPTS_REG(7:4) for TX_INT

Unmask the interrupt:

- DSP_WRITE(1) =
  - INTERRUPTS_REG(8) for RX_INT
  - INTERRUPTS_REG(9) for TX_INT
  - INTERRUPTS_REG(10) for FERR_INT

On interrupt occurrence:

- DSP_READ =
  - STATUS_REG(0) for FERR_INT occurrence
  - STATUS_REG(2) for RX_INT occurrence
  - STATUS_REG(3) for RX character overflow
  - STATUS_REG(4) for TX_INT occurrence
  - STATUS_REG(5) for TX character underflow

Then, to release the interrupt signal and reset the corresponding status bits:

- DSP_WRITE(1**)** =
  - STATUS_REG(0) for FERR_INT release
  - STATUS_REG(2) for RX_INT release
  - STATUS_REG(4) for TX_INT release

4.1.5 DMA Channel Operation

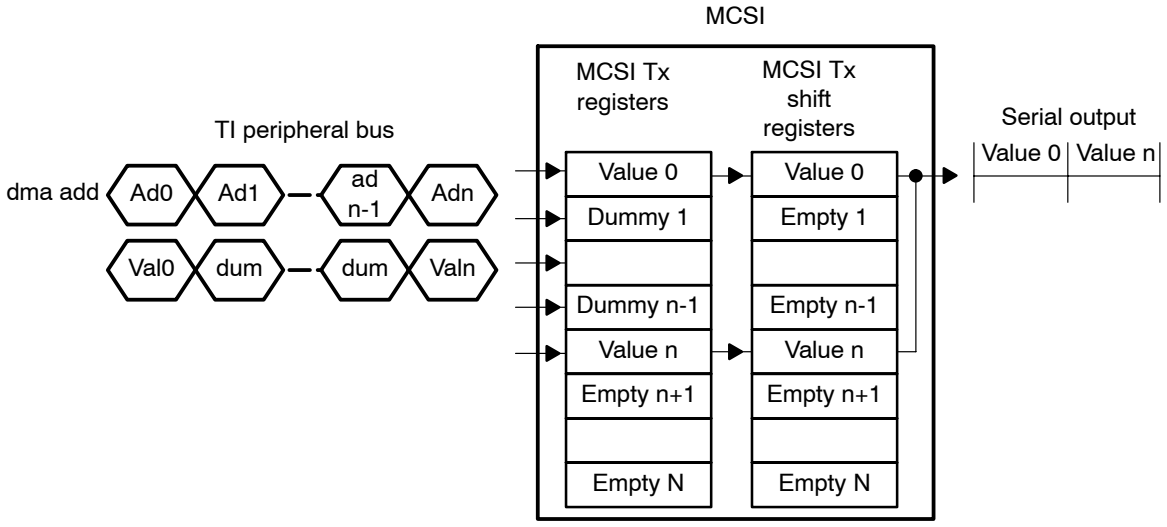
Both transmit and receive operations can be supported by DMA. DMA support is enabled by control bits in the MAIN_PARAMETERS_REG:

- MAIN_PARAMETERS_REG(15:14) = DMA_ENABLE(1:0)
  - TX_DMA_REQ enabled when DMA_ENABLE(0) = 1
  - TX_DMA_REQ disabled when DMA_ENABLE(0) = 0
  - RX_DMA_REQ enabled when DMA_ENABLE(1) = 1
  - RX_DMA_REQ disabled when DMA_ENABLE(1) = 0

Transmit DMA Transfers

A new transmit DMA transfer is initiated during the transmission of the last channel of a frame, at which time all data in the transmit registers (TX_REGs) has been moved to shift registers; the TX_REGs are now ready to be rewritten. If N channels are used, the DMA controller successively accesses all consecutive registers between TX_REG(0) and TX_REG(N-1). If some channels between TX_REG(0) and TX_REG(N-1) are not used, the DMA controller writes dummy values when addressing these unused registers (see Figure 43).

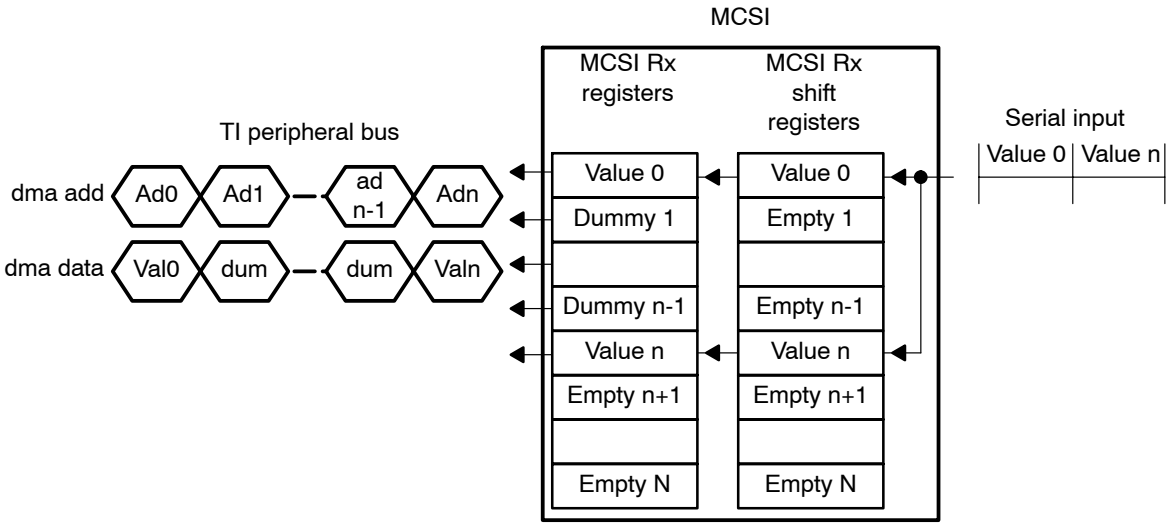
Figure 43. Transmit DMA Transfers



Receive DMA Transfers

A receive DMA transfer is initiated after the reception of the last channel of a frame, at which time all receive registers RX_REG have been updated and are ready to be read. If N channels are used, the DMA controller successively accesses all consecutive registers between RX_REG(0) and RX_REG(N-1). If some channels between RX_REG(0) and RX_REG(N-1) are not used, the DMA controller reads dummy values when addressing these unused registers (see Figure 44).

Figure 44. Receive DMA Transfers



A multichannel application cannot use DMA for some channels and interrupt servicing for others. RX/TX interrupts are not generated when DMA RX/TX transfers are enabled.

4.1.6 Interface Activation

Start Sequence

A typical sequence to start the interface is:

1) MCSI configuration:
   a) DSP_WRITE(0x0000)= CONTROL_REG in order to remove the write protection on the control registers
   b) DSP_WRITE(0x….)= MAIN_PARAMETERS_REG
   c) DSP_WRITE(0x….)= INTERRUPTS_REG
   d) DSP_WRITE(0x….)= CHANNEL_USED_REG
   e) DSP_WRITE(0x….)= CLOCK_FREQUENCY_REG
   f) DSP_WRITE(0x….)= OVER_CLOCK_REG
2) Transmit data loading for selected channels:
   a) DSP_WRITE(0x….)= TX_REG[channel index]
3) Enable MCSI clock:
   a) DSP_WRITE(0x0001)= CONTROL_REG

Stop Sequence

A typical sequence to stop the interface is:

1) Disable MCSI clock: DSP_WRITE(0x0000) = CONTROL_REG

   The status register keeps its content even after the stop of the transmission. The control registers can now be modified.

2) Software reset: DSP_WRITE(0x0002) = CONTROL_REG

   The software reset initializes the status register.

Software Reset

The MCSI software reset is activated with the SW_RESET bit of the control register (CONTROL_REG) (see Table 52, *Activity Control Register*).

This reset is limited to the control and status registers, the internal state machine, and the PISO and SIPO logic. The parameters registers are not affected by this software reset.

On the software reset, the MCSI reference clock is disabled, thus halting the execution of any current operating mode.

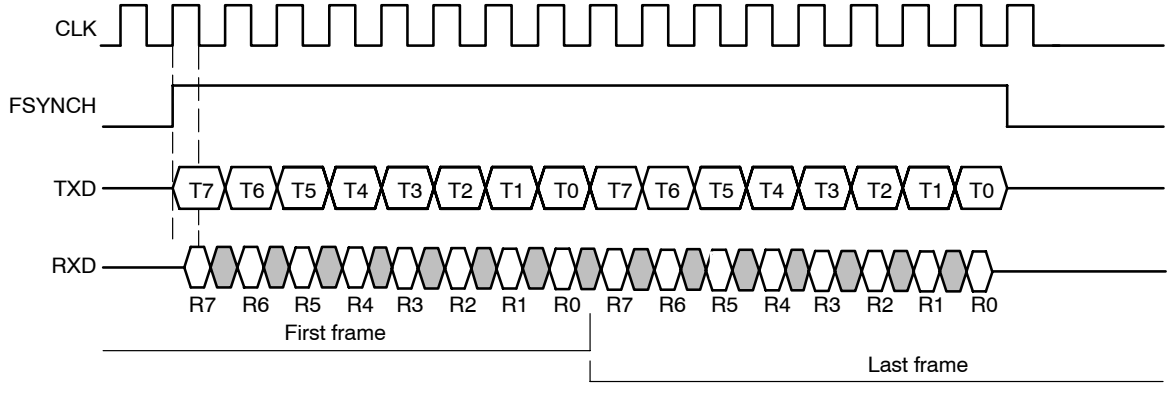
4.1.7 Functional Mode Timing Diagrams

The following timing diagrams are based on a positive clock polarity with parameter CLOCK_POL = 0.

(Transmit on rising edge/receive on falling edge.)

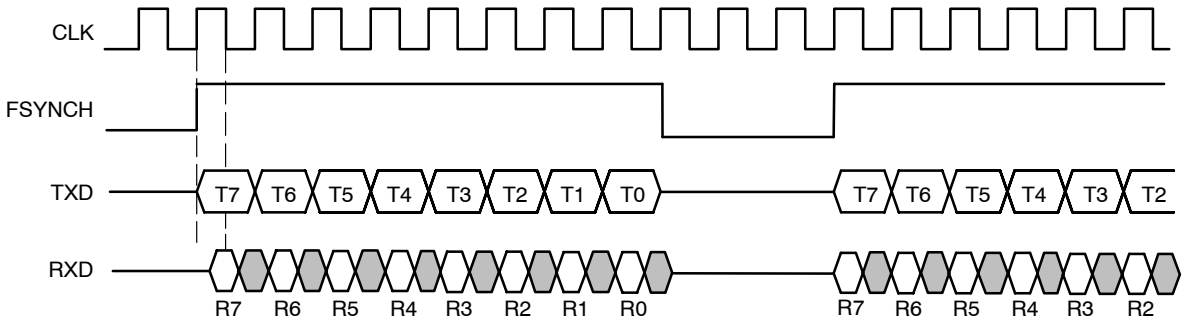
Single-Channel/Alternate Long Framing

Figure 45. Single-Channel/Alternate Long Framing



Single-Channel/Alternate Long Framing/Burst

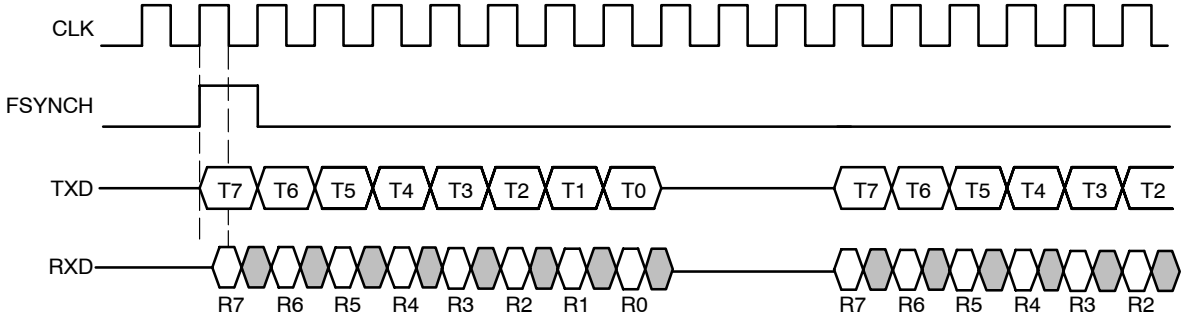
Figure 46. Single-Channel/Alternate Long Framing/Burst



OVER_CLOCK_REG = 0x0003

Single-Channel/Alternate Short Framing/Continuous/Burst

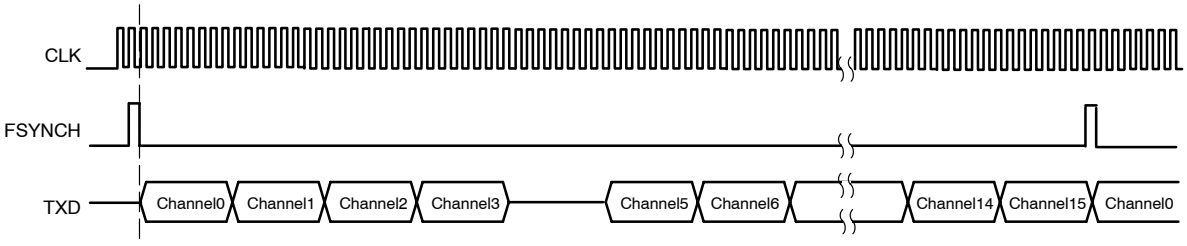
Figure 47. Single-Channel/Alternate Short Framing/Continuous/Burst



OVER_CLOCK_REG = 0x0003

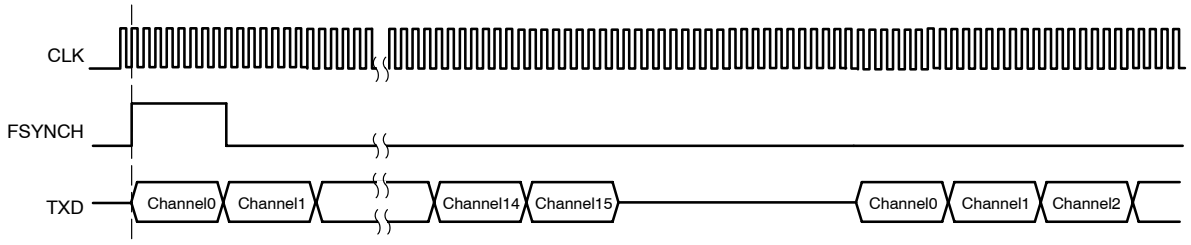
Multichannel/Normal Short Framing/Channel4 Disable

Figure 48. Multichannel/Normal Short Framing/Channel4 Disable



Multichannel/Alternate Long Framing/Continuous/Burst

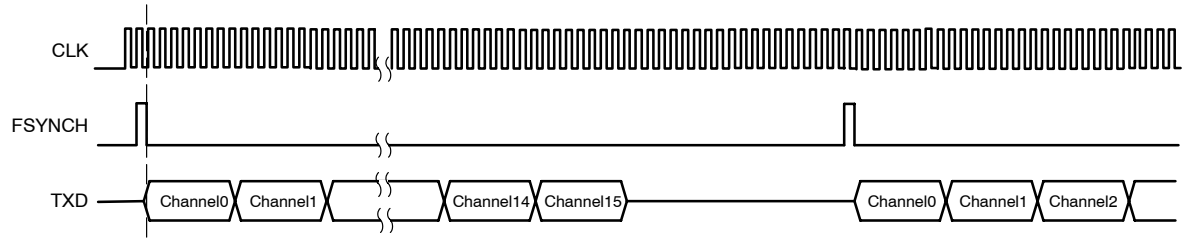
Figure 49. Multichannel/Alternate Long Framing/Continuous/Burst



OVER_CLOCK_REG = 0x0013

Multichannel/Normal Short Framing/Burst

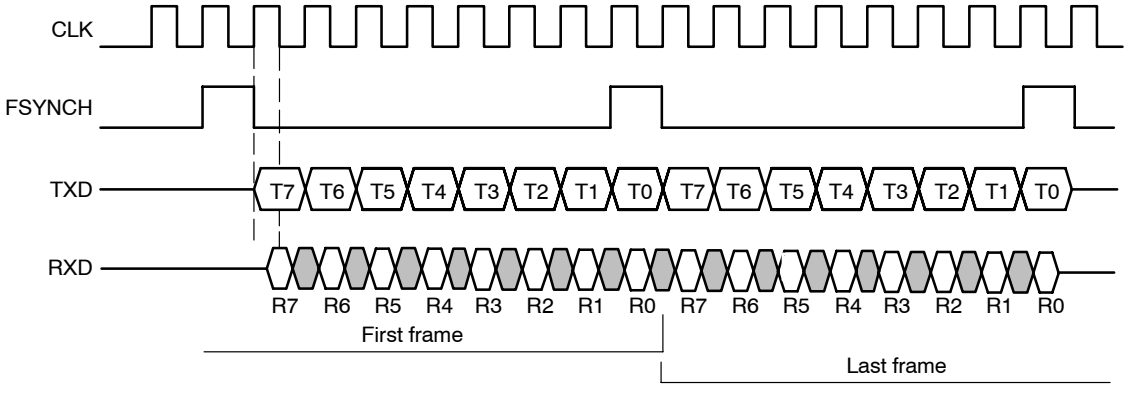
Figure 50. Multichannel/Normal Short Framing/Burst



OVER_CLOCK_REG = 0x0013

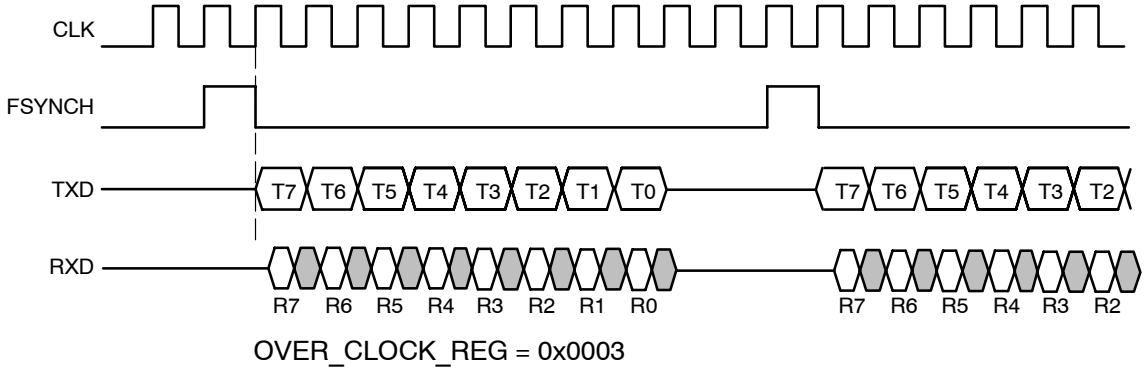
Single-Channel/Normal Short Framing

Figure 51. Single-Channel/Normal Short Framing



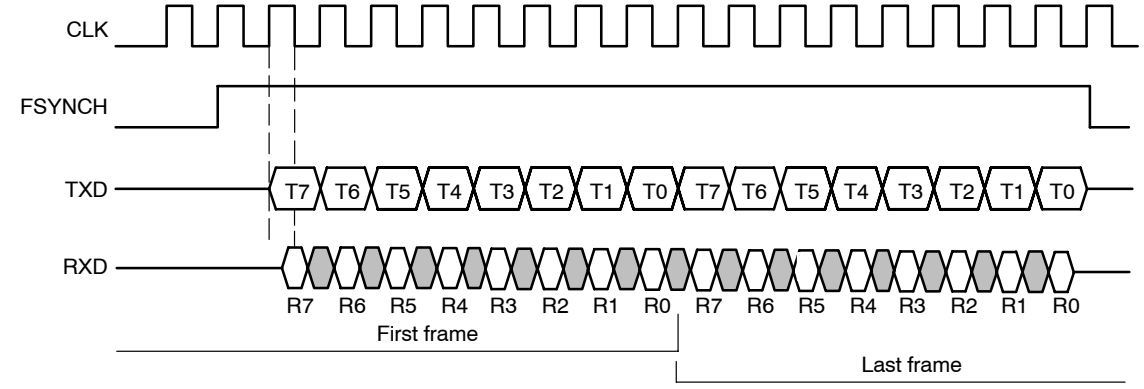
Single-Channel/Normal Short Framing/Burst

Figure 52. Single-Channel/Normal Short Framing/Burst



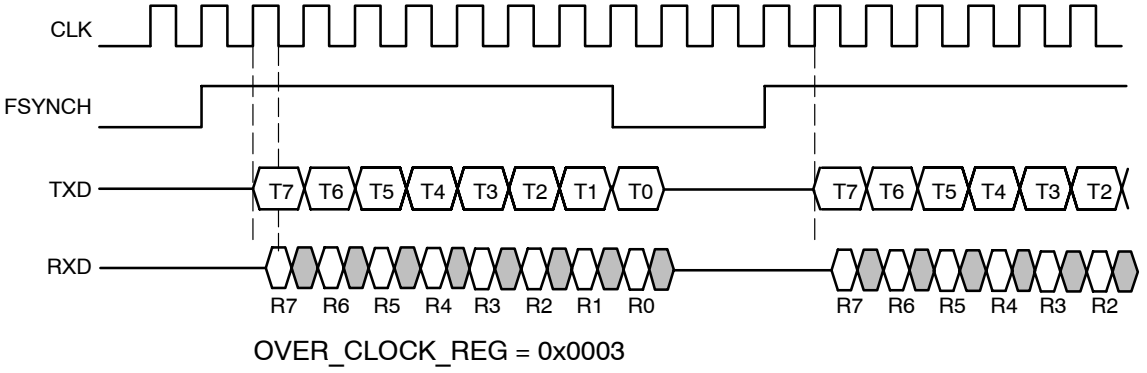
Single-Channel/Normal Long Framing

Figure 53. Single-Channel/Normal Long Framing



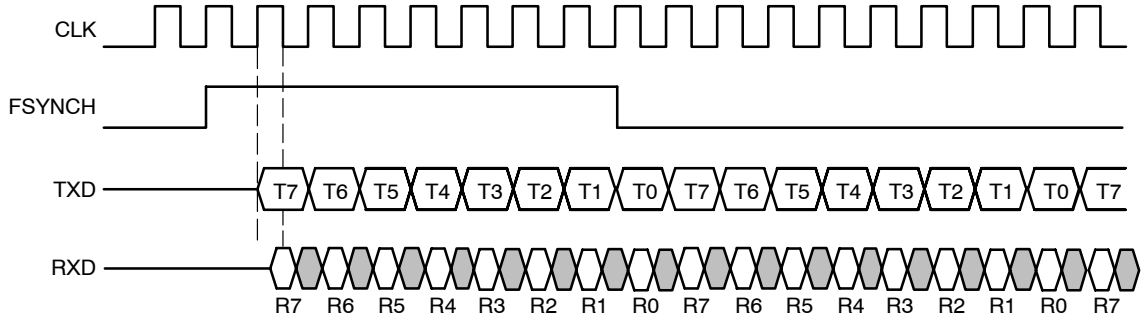
Single-Channel/Normal Long Framing/Burst

Figure 54. Single-Channel/Normal Long Framing/Burst



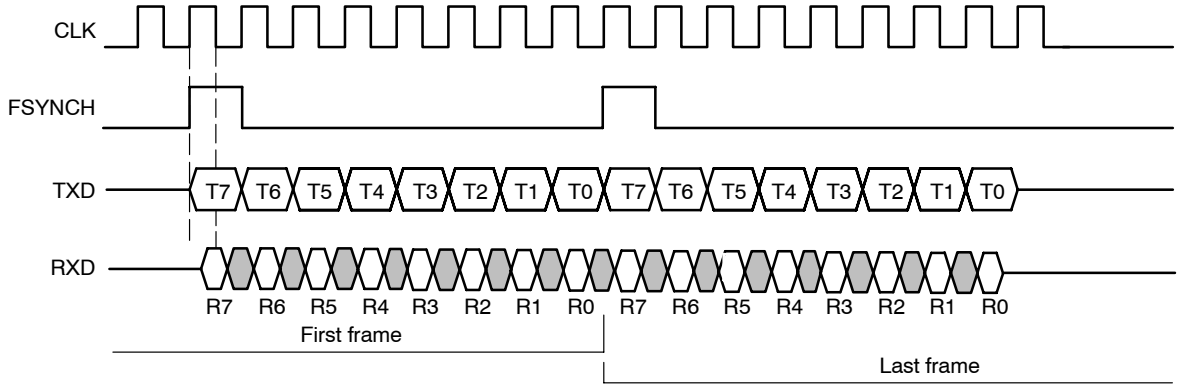
Single-Channel/Normal Long Framing/Continuous

Figure 55. Single-Channel/Normal Long/Continuous



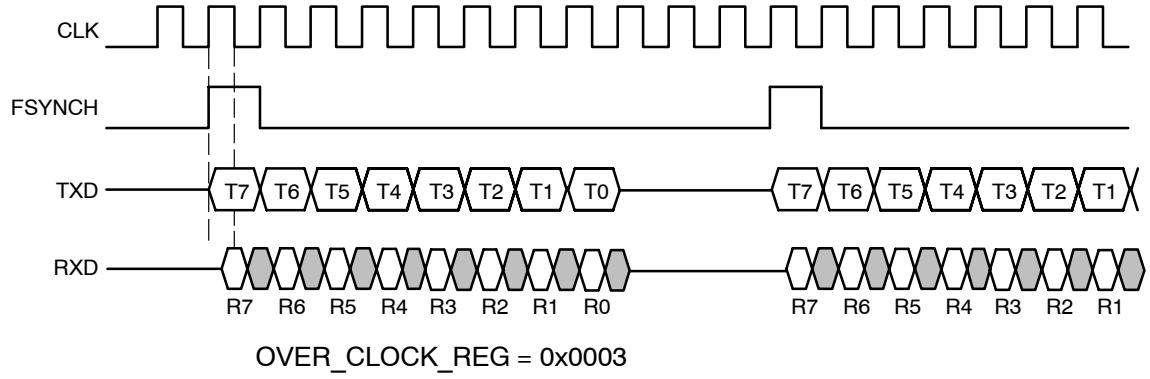
Single-Channel/Alternate Short Framing

Figure 56. Single-Channel/Alternate Short Framing



Single-Channel/Alternate Short Framing/Burst

Figure 57. Single-Channel/Alternate Short Framing/Burst



4.2 MCSI Register Descriptions

Table 47 through Table 55 describe the MCSI registers. The CHANNEL_USED_REG, CLOCK_FREQUENCY_REG, OVER_CLOCK_REG, INTERRUPTS_REG, and MAIN_PARAMETERS_REG registers are write protected if the MCSI is enabled (CONTROL_REG[0] = 1).

The channel selection register is only used in multichannel mode.

Table 47. Channel Selection Register (CHANNEL_USED_REG)

Bit	Name	Access	Hardware Reset
15	USE_CH15	R/W	0
14	USE_CH14	R/W	0
13	USE_CH13	R/W	0
12	USE_CH12	R/W	0
11	USE_CH11	R/W	0
10	USE_CH10	R/W	0
9	USE_CH9	R/W	0
8	USE_CH8	R/W	0
7	USE_CH7	R/W	0
6	USE_CH6	R/W	0
5	USE_CH5	R/W	0
4	USE_CH4	R/W	0
3	USE_CH3	R/W	0
2	USE_CH2	R/W	0
1	USE_CH1	R/W	0
0	USE_CH0	R/W	0

USE_CH[i] selects channel [i] for data transmission (active high).

Table 48. Clock Frequency Register (CLOCK_FREQUENCY_REG)

The clock frequency register is used only in master mode when the interface generates the serial clock.

Bit	Name	Description	Access	Hardware Reset
15:11	Unused		R	0000 0
10:0	CLK_FREQ	Division factor of 12-MHz reference clock (2<=CLK_FREQ<= 2047) In master mode, this register defines the transmission baud rate from a frequency ratio based on a 12-MHz reference clock. The transmission clock frequency can be programmed from 5.8 kHz to 6 MHz in steps or increments of 83 ns. Clock frequency = 12 MHz/CLK_FREQ with 2 <= CLK_FREQ <= 2047.	R/W	000 0000 0000

CLK_FREQ: division factor of 12-MHz reference clock (2<=CLK_FREQ<= 2047)

In master mode, this register defines the transmission baud rate from a frequency ratio based on a 12-MHz reference clock. The transmission clock frequency can be programmed from 5.8 kHz to 6 MHz in steps or increments of 83 ns.

Clock frequency = 12 MHz / CLK_FREQ with 2 <= CLK_FREQ <= 2047.

Table 49. Oversized Frame Dimension Register (OVER_CLOCK_REG)

Bit	Name	Description	Access	Hardware Reset
15:10	Unused		R	0000 00
9:0	OVER_CLOCK	Overhead clock periods in frame duration (0 = OVER_CLOCK = 1023)	R/W	00 0000 0000

Table 50. Interrupt Masks Register (INTERRUPTS_REG)

Bit	Name	Description	Access	Hardware Reset
15:11	Unused		R	0000 0
10	MASK_IT_ERROR	Mask of frame duration error interrupt (active at 0)	R/W	0
9	MASK_IT_TX	Mask of transmit interrupt (active at 0)	R/W	0

Table 50. Interrupt Masks Register (INTERRUPTS_REG) (Continued)

Bit	Name	Description	Access	Hardware Reset
8	MASK_IT_RX	Mask of receive interrupt (active at 0)	R/W	0
7:4	Number channel for IT_TX	Channel number for transmit interrupt generation (0 <=NB_CHAN <= 15)	R/W	0000
3 – 0	Number channel for IT_RX	Channel number for receive interrupt generation (0 <= NB_CHAN <=15)	R/W	0000

Table 51. Main Parameters Register (MAIN_PARAMETERS__REG)

Bit	Name	Value	Description	Access	Hardware Reset
15:14	DMA enable		Enable bits for DMA:	R/W	00
		00	Normal mode (No DMA)		
		01	DMA transmit mode, normal receive mode		
		10	Normal transmit mode, DMA receive mode		
		11	DMA transmit and receive mode		
13:11	Reserved		Reserved bits. These bits must always be written as 0.	R/W	000
10	FSYNCH_POLARITY		Frame-synchronization pulse polarity	R/W	0
		0	Positive		
		1	Negative		
9	FSYNCH_MODE		Frame-synchronization pulse position	R/W	0
		0	Normal		
		1	Alternate		
8	FSYNCH_SIZE		Frame-synchronization pulse shape	R/W	0
		0	Short		
		1	Long		
7	Multi/single		Frame structure	R/W	0

Table 51. Main Parameters Register (MAIN_PARAMETERS__REG) (Continued)

Bit	Name	Value	Description	Access	Hardware Reset
		0	Single		
		1	Multi		
6	MCSI mode		Interface transmission mode	R/W	0
		0	Slave		
		1	Master		
5	Continuous/burst		Frame mode	R/W	0
		0	Burst		
		1	Continuous		
4	CLOCK_POLARITY		Clock edge selection	R/W	0
		0	Positive		
		1	Negative		
3:0	Word size		Word size in bits number (2 <= size <= 15) with 2 for 3 bits and 15 for 16 bits.	R/W	0000

Table 52. Activity Control Register (CONTROL_REG)

Bit	Name	Value	Description	Access	Hardware Reset	Software Reset
15:3	Reserved		Reserved bits. These bits must always be written as 0.	R	0000 0000 0000 0	0000 0000 0000 0
2	Reserved		Reserved bits. These bits must always be written as 0.	R/W	0	0
1	MCSI software reset		Asynchronous reset of MCSI module	R/W	0	1
		0	Disable			
		1	Enable			

Table 52. Activity Control Register (CONTROL_REG) (Continued)

Bit	Name	Value	Description	Access	Hardware Reset	Software Reset
0	MCSI clock enable		Enable clock of MCSI module	R/W	0	0
		0	Disable			
		1	Enable			

Note:

The software reset is applied as long as the MCSI software reset bit is set to 1. A software reset disables the MSCI (the MCSI clk enable bit is cleared) and initializes the status register. It does not modify the other registers.

To clear an interrupt on the MCSI, the DSP must write to the MCSI status register with the bit corresponding to the interrupt set to 1. The MCSI status register has a two-cycle latency when writing into it, so the interrupt line is cleared two cycles after a write. To prevent clearing the interrupt handler before the interrupt line is cleared, the interrupt routine must be at least two cycles long.

Table 53. Interface Status Register (STATUS_REG)

Bit	Name	Value	Description	Access	Hardware Reset	Software Reset
15:7	Reserved		Reserved bits. These bits must always be written as 0.	R	0000 0000 0	0000 0000 0
6	Reserved		Reserved bits. These bits must always be written as 0.	R/W	0	0
5	TX underflow		Transmit underflow	R	0	0
		0	No under			
		1	Under			
4	TX ready		Flag for transmit interrupt occurrence	R/W	0	0
		0	No interrupt			
		1	Interrupt			

Table 53. Interface Status Register (STATUS_REG) (Continued)

Bit	Name	Value	Description	Access	Hardware Reset	Software Reset
3	RX overflow		Receive overflow	R	0	0
		0	No over			
		1	Over			
2	RX ready		Flag for receive interrupt occurrence	R/W	0	0
		0	No interrupt			
		1	Interrupt			
1	Error type few/many		Too short (few) or too long frame (many) status	R	0	0
		0	Short			
		1	Long			
0	Frame error		Error flag when wrong frame duration	R/W	0	0
		0	Correct			
		1	Bad			

This register is cleared by a software reset.

Table 54. Receive Word Register (RX_REG[15:0])

Bit	Name	Access	Hardware Reset
15	b15	R	U
14	b14	R	U
13	b13	R	U
12	b12	R	U
11	b11	R	U
10	b10	R	U
9	b9	R	U

Note: The MCSI receives the most significant bit first. For example, if the word_size equals 11, the upper 12 bits of the RX registers contain the received data, and the lower 4 bits are zeros.

Table 54. Receive Word Register (RX_REG[15:0]) (Continued)

Bit	Name	Access	Hardware Reset
8	b8	R	U
7	b7	R	U
6	b6	R	U
5	b5	R	U
4	b4	R	U
3	b3	R	U
2	b2	R	U
1	b1	R	U
0	b0	R	U

Note: The MCSI receives the most significant bit first. For example, if the word_size equals 11, the upper 12 bits of the RX registers contain the received data, and the lower 4 bits are zeros.

Table 55. Transmit Word Register (TX_REG[15:0])

Bit	Name	Access	Hardware Reset
15	b15	R/W	U
14	b14	R/W	U
13	b13	R/W	U
12	b12	R/W	U
11	b11	R/W	U
10	b10	R/W	U
9	b9	R/W	U
8	b8	R/W	U
7	b7	R/W	U
6	b6	R/W	U
5	b5	R/W	U
4	b4	R/W	U

Note: The MCSI transmits the most significant bit first. For example, if the word_size equals 11, the upper 12 bits of the TX registers are transmitted.

Table 55. Transmit Word Register (TX_REG[15:0]) (Continued)

Bit	Name	Access	Hardware Reset
3	b3	R/W	U
2	b2	R/W	U
1	b1	R/W	U
0	b0	R/W	U

Note: The MCSI transmits the most significant bit first. For example, if the word_size equals 11, the upper 12 bits of the TX registers are transmitted.

5 MCSI1 and MCSI2

This section provides information specific to MCSI1 and MCSI2 on the device.

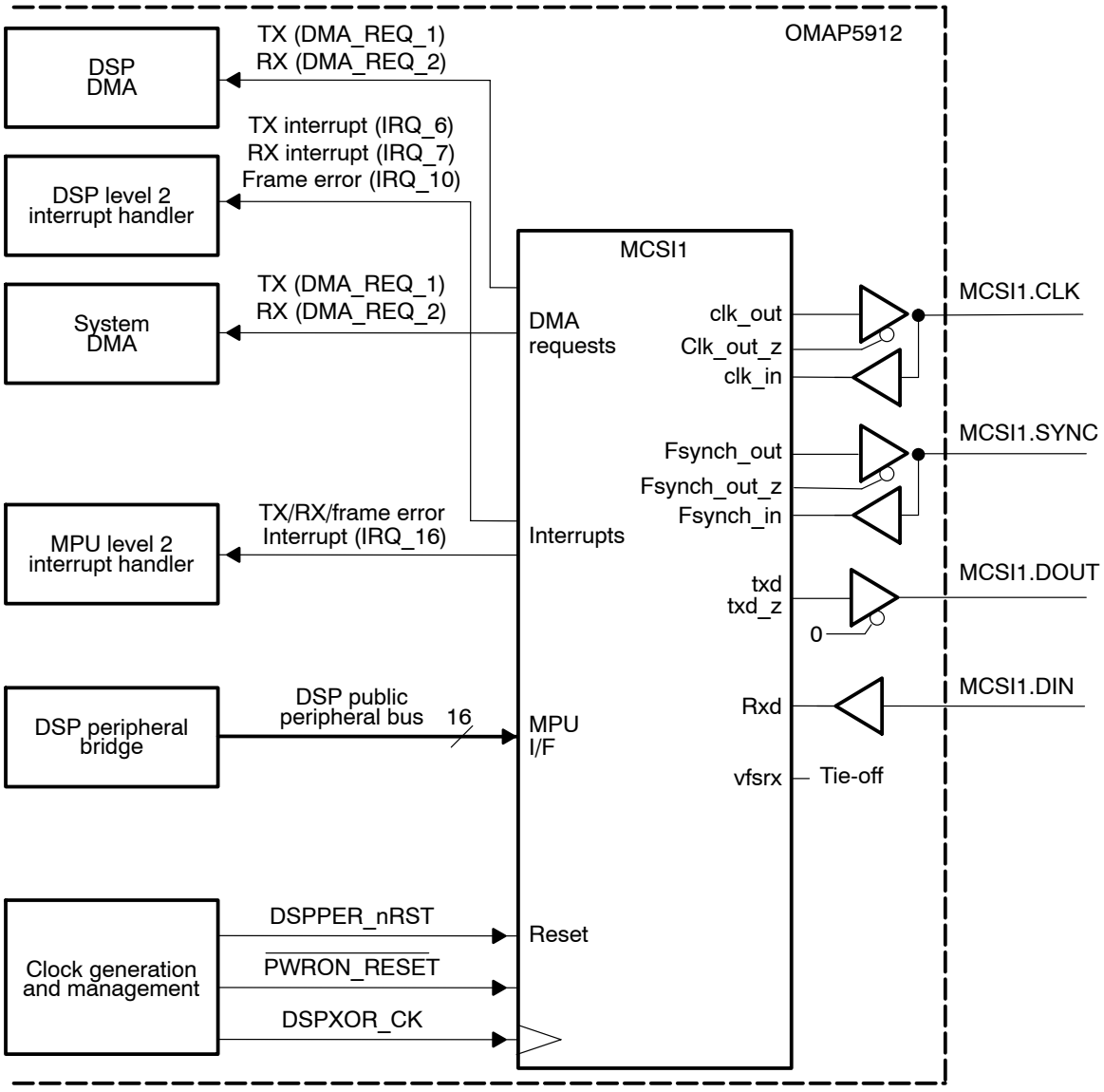
5.1 MCSI1 Pin Description

Table 56 identifies the MCSI1 I/O pins. Figure 58 shows the MCSI1 interface.

Table 56. MCSI1 Pin Descriptions

Pin	I/O Direction	Description
MCSI1.DIN	In	Data input
MCSI1.DOUT	Out	Data output
MCSI1.CLK	In/out	Bit clock
MCSI1.SYNC	In/out	Frame synchronization

Figure 58. MCSI1 Interface



5.2 MCSI1 Interrupt Mapping

Table 57 identifies the MCSI1 interrupt mappings. MCSI1 generates level-2 interrupts for both the DSP and the MPU. Only one MPU MCSI1 interrupt covers TX, RX, and frame error conditions; software must check the MCSI1 status register to determine the interrupt source.

Table 57. MCSI1 Interrupt Mapping

Incoming Interrupts	Level 2 DSP Interrupt	Level 2 MPU Interrupt
MCSI1 TX interrupt	IRQ_06	IRQ_16
MCSI1 RX interrupt	IRQ_07	IRQ_16
MCSI1 frame error	IRQ_10	IRQ_16

5.3 MCSI1 DMA Request Mapping

Table 58 identifies MCSI1 DMA request lines.

Table 58. TDMA Request Mapping—MCSI1

DMA Request Source	DMA Request Line—DSP	DMA Request Line—MPU
MCSI1 TX	DMA_REQ_01	DMA_REQ_01
MCSI1 RX	DMA_REQ_02	DMA_REQ_02

5.4 MCSI2 Pin Description

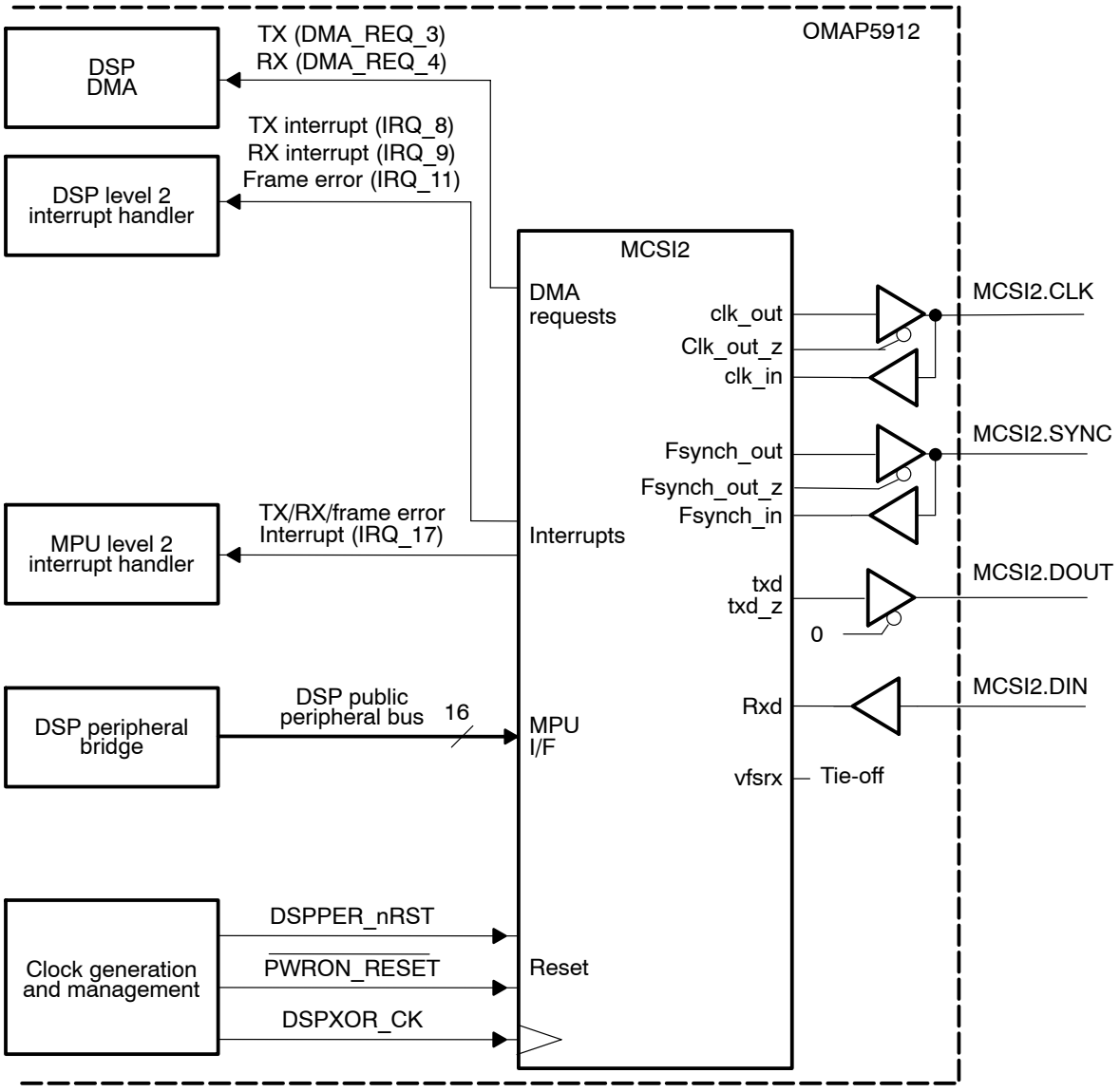
This section provides information specific to MCSI2 on the device.

Table 59 identifies the MCSI2 I/O pins. Figure 59 shows the MCSI2 interface.

Table 59. MCSI2 Pin Descriptions

Pin	I/O Direction	Description
MCSI2.DIN	In	Data input
MCSI2.DOUT	Out	Data output
MCSI2.CLK	In/out	Bit clock
MCSI2.SYNC	In/out	Frame synchronization

Figure 59. MCSI2 Interface



5.5 MCSI2 Interrupt Mapping

Table 60 identifies the MCSI2 interrupts. MCSI2 generates level-2 interrupts for both the DSP and the MPU. Only one MPU MCSI2 interrupt covers TX, RX, and frame error conditions; software must check the MCSI2 status register to determine the interrupt source.

Table 60. MCSI2 Interrupt Mapping

Incoming Interrupts	Level 2 DSP Interrupt	Level 2 MPU Interrupt
MCSI2 TX interrupt	IRQ_08	IRQ_17
MCSI2 RX interrupt	IRQ_09	IRQ_17
MCSI2 frame error	IRQ_11	IRQ_17

5.6 MCSI2 DMA Request Mapping

Table 61 identifies MCSI2 DMA request lines. Only the DSP DMA controller can transfer MCSI2 data; there is no MPU DMA capability.

Table 61. DMA Request Mapping—MCSI2

DMA Request Source	DMA Request Line—DSP	DMA Request Line—MPU
MCSI2 TX	DMA_REQ_03	–
MCSI2 RX	DMA_REQ_04	–

6 UARTs

There are three nearly identical UART modules on this device. The modules are identical except for the fact that UART2 does not support infrared data association (IrDA). UART1 and UART3 support IrDA. This section describes all 3 UART modules. Note that discussions regarding IrDA apply only to UART1 and UART3. This section includes a register description and a module configuration example. It also shows the the basic UART IrDA module pins.

Figure 60. UART IrDA Signals

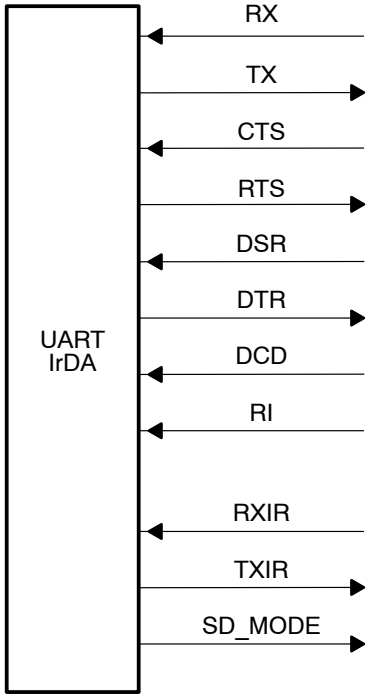
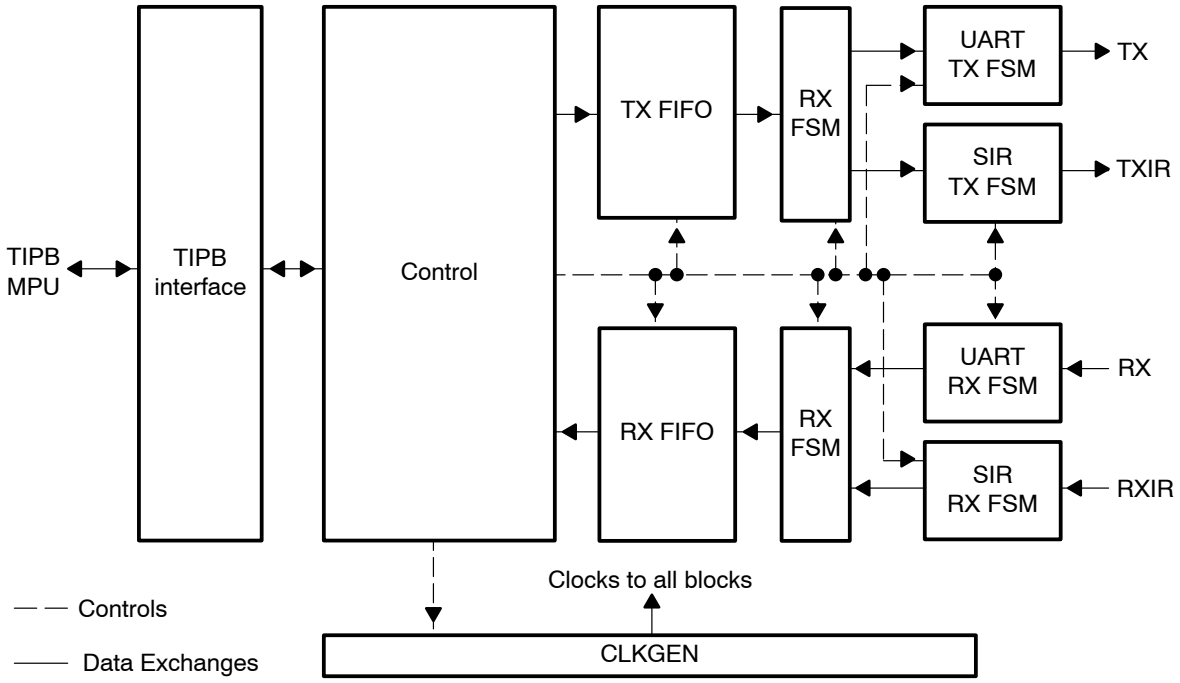


Figure 61. Functional Block Diagram



6.1 Main Features

- ❏ Selectable UART/IrDA modes
- ❏ Dual 64 entry FIFOs for received and transmitted data
- ❏ Programmable and selectable transmit and receive FIFO trigger levels for DMA and interrupt generation
- ❏ Programmable sleep mode
- ❏ Complete status reporting capabilities in both normal and sleep modes
- ❏ Frequency prescaler values from 0 to 16383 to generate the appropriate baud rates
- ❏ Single 48-MHz clock reference for baud setting
- ❏ Two DMA requests and one interrupt request to the system

6.1.1 UART/Modem Functions

- Baud-rate from 300 bits/s up to 3.6864M bits/s
- Autobaud between 1200 bits/s and 115.2K bits/s
- Software/hardware flow control
  - Programmable Xon/Xoff characters
  - Programmable auto-$\overline{\text{RTS}}$ and auto-$\overline{\text{CTS}}$
- Programmable serial interface characteristics
  - 5-, 6-, 7-, or 8-bit characters
  - Even, odd, mark (always = 1), space (always = 0) or no parity (non parity bit frame) bit generation and detection
  - 1, 1.5, or 2 stop-bit generation
- False start bit detection
- Line break generation and detection
- Fully prioritized interrupt system controls
- Internal test and loopback capabilities
- Modem control functions ($\overline{\text{CTS}}$, $\overline{\text{RTS}}$, $\overline{\text{DSR}}$, $\overline{\text{DTR}}$, $\overline{\text{RI}}$ and $\overline{\text{DCD}}$)

6.1.2 IrDA Functions

- Slow infrared (SIR 115.2 KBAUD), medium infrared (MIR 0.576 MBAUD) and fast infrared (FIR 4.0 MBAUD) operations. Very fast infrared (VFIR) is not supported.
- Framing error, cyclic redundancy check (CRC) error, illegal symbol (FIR), abort pattern (SIR, MIR) detection.
- 8-entry status FIFO (with selectable trigger levels) available to monitor frame length and frame errors.

In Table 62, the module I/O description is at the module level.

Table 62. I/O Description

Signal	I/O	Description	Reset
		UART/MODEM Signals (UART 1,2,3)	
RX	I	Serial data input.	Unknown
TX	O	Serial data output.	1
$\overline{CTS}$	I	Clear to send. Active-low modem status signal. Reading bit 4 of the modem status register checks the condition of $\overline{CTS}$. Reading bit 0 of that register checks a change of state of $\overline{CTS}$ since the last read of the modem status register. $\overline{CTS}$ is used in auto-$\overline{CTS}$ mode to control the transmitter.	Unknown
$\overline{RTS}$	O	Request to send. When active (low), the module is ready to receive data. Setting modem control register bit 1 activates $\overline{RTS}$. It becomes inactive as a result of a module reset, loopback mode, or by clearing the MCR[1]. In auto-$\overline{RTS}$ mode, it becomes inactive as a result of the receiver threshold logic.	1
$\overline{DSR}$	I	Data set ready. Active-low modem status signal. Reading bit 5 of the modem status register checks the condition of $\overline{DSR}$. Reading bit 1 of that register checks a change of state of $\overline{DSR}$ since the last read of the modem status register.	Unknown
$\overline{DTR}$	O	Data terminal ready. Active-low modem control signal. Reading bit 0 of the modem control register checks the condition of $\overline{DTR}$.	1
$\overline{DCD}$	I	Data carrier detect. Active-low modem status signal. The condition of $\overline{DCD}$ is checked by reading bit 7 of the modem status register, and any change in its state can be detected by reading bit 3 of that register.	Unknown
$\overline{RI}$	I	Reading indicator. Active-low modem status signal. The condition of $\overline{RI}$ is checked by reading bit 6 of the modem status register, and any change in its state is detected by reading bit 2 of that register.	Unknown

Table 62. I/O Description (Continued)

Signal	I/O	Description	Reset
		IrDA Signals (UART1 and UART3 only)	
RXIR	I	Serial data input.	Unknown
TXIR	O	Serial data output.	0
SD	O	Signal used to configure transceivers.	1

6.2 Control and Status Registers Description

Each register is selected using a combination of address and some LCR register bit(s) settings as shown in the following Table 63.

6.2.1 UART IrDA Registers Mapping

The local host can access the following registers at address = module base address + address offset. The module base address is the module start address. Register address offsets depend on the module address alignment at the system top level. The address offsets (0x13 x S) and (0x18 x S to 0x31 x S (inclusive)) are reserved and must be read as 0x00 at all times.

- S = 1 for 8-bit aligned addresses
- 2 for 16-bit aligned addresses
- 4 for 32-bit aligned addresses

All UART registers are 8-bit. Start addresses:

- UART1: FFFB 0000
- UART2: FFFB 0800
- UART3: FFFB 9800

Table 63. UART IrDA Registers

Address Offset	Registers					
	LCR[7] = 0		LCR[7] = 1 and LCR[7:0] is not 0xBF		LCR[7:0] = 0xBF	
	READ	WRITE	READ	WRITE	READ	WRITE
0x00 x S	RHR	THR	DLL	DLL	DLL	DLL
0x01 x S	IER §	IER §	DLH	DLH	DLH	DLH
0x02 x S	IIR	FCR‡	IIR	FCR‡	EFR	EFR
0x03 x S	LCR	LCR	LCR	LCR	LCR	LCR
0x04 x S	MCR‡	MCR‡	MCR‡	MCR‡	XON1/ADDR1	XON1/ADDR1
0x05 x S	LSR	–	LSR	–	XON2/ADDR2	XON2/ADDR2
0x06 x S	MSR/TCR†	TCR†	MSR/TCR†	TCR†	XOFF1/TCR†	XOFF1/TCR†
0x07 x S	SPR/TLR†	SPR/TLR†	SPR/TLR†	SPR/TLR†	XOFF2/TLR†	XOFF2/TLR†
0x08 x S	MDR1	MDR1	MDR1	MDR1	MDR1	MDR1
0x09 x S	MDR2	MDR2	MDR2	MDR2	MDR2	MDR2
0x0A x S	SFLSR	TXFLL	SFLSR	TXFLL	SFLSR	TXFLL
0x0B x S	RESUME	TXFLH	RESUME	TXFLH	RESUME	TXFLH
0x0C x S	SFREGL	RXFLL	SFREGL	RXFLL	SFREGL	RXFLL
0x0D x S	SFREGH	RXFLH	SFREGH	RXFLH	SFREGH	RXFLH
0x0E x S	BLR	BLR	UASR	–	UASR	–
0x0F x S	ACREG	ACREG	–	–	–	–
0x10 x S	SCR	SCR	SCR	SCR	SCR	SCR
0x11 x S	SSR	–	SSR	–	SSR	–
0x12 x S	EBLR	EBLR	–	–	–	–
0x14 x S	MVR	–	MVR	–	MVR	–
0x15 x S	SYSC	SYSC	SYSC	SYSC	SYSC	SYSC

† In UART modes, IER[7:4] can only be written when EFR[4] = 1. In IrDA modes, EFR[4] has no effect on access to IER[7:4].
‡ MCR[7:5] and FCR[5:4] can only be written when EFR[4] = 1.
§ Transmission control register (TCR) and trigger level register (TLR) are accessible only when EFR[4] = 1 and MCR[6] = 1.

Table 63. UART IrDA Registers (Continued)

Address Offset	Registers					
	LCR[7] = 0		LCR[7] = 1 and LCR[7:0] is not 0xBF		LCR[7:0] = 0xBF	
	READ	WRITE	READ	WRITE	READ	WRITE
0x16 x S	SYSS	SYSS	SYSS	SYSS	SYSS	SYSS
0x17 x S	WER	WER	WER	WER	WER	WER

† In UART modes, IER[7:4] can only be written when EFR[4] = 1. In IrDA modes, EFR[4] has no effect on access to IER[7:4].
‡ MCR[7:5] and FCR[5:4] can only be written when EFR[4] = 1.
§ Transmission control register (TCR) and trigger level register (TLR) are accessible only when EFR[4] = 1 and MCR[6] = 1.

Offset Address (hex): 0x00 x S and LCR[7] = 0 and read

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 0 of the FIFO is used to store the single data character.

If an overflow occurs the data in the RHR is not overwritten.

Table 64. Receive Holding Register (RHR)

Bit	Name	Function	R/W	Reset
7:0	RHR	Receive holding register	R	Unknown

Offset Address (hex): 0x00 x S and LCR[7] = 0 and write

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 0 of the FIFO is used to store the data.

Table 65. Transmit Holding Register (THR)

Bit	Name	Function	R/W	Reset
7:0	THR	Transmit holding register	W	Unknown

Offset Address (hex): 0x02 x S and LCR not 0xBF (and EFR[4] = 1 for FCR[5:4]) and write.

Table 66. FIFO Control Register (FCR)

Bit	Name	Function	R/W	Reset
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 If SCR[7] = 0 and TLR[7:4] non-0, 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	00
5:4	TX_FIFO_TRIG	Sets the trigger level for the TX FIFO: If SCR[6] = 0 and TLR[3:0] = 0000: 00: 8 spaces 01: 16 spaces 10: 32 spaces 11: 56 spaces If SCR[6] = 0 and TLR[3:0] non-0, 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.	W	00
3	DMA_MODE	0: DMA_MODE 0 (No DMA) 1: DMA_MODE 1 (UART_nDMA_REQ[0] in TX, UART_nDMA_REQ[1] in RX) This register is considered if SCR[0] = 0.	W	0
2	TX_FIFO_CLEAR	0: No change. 1: Clears the transmit FIFO and resets its counter logic to 0. Returns to 0 after clearing FIFO.	W	0
1	RX_FIFO_CLEAR	0: No change. 1: Clears the receive FIFO and resets its counter logic to 0. Returns to 0 after clearing FIFO.	W	0
0	FIFO_EN	0: Disables the transmit and receive FIFOs. 1: Enables the transmit and receive FIFOs.	W	0

Notes: 1) Bits 4 and 5 can only be written to when EFR[4] = 1.

2) Bits 0 to 3 can be changed only when the baud clock is not running (DLL and DLH set to 0).

3) See for FCR[5:4] setting restriction when SCR[6] = 1.

4) See for FCR[7:6] setting restriction when SCR[7] = 1.

Offset Address (hex): 0x10 x S

Table 67. Supplementary Control Register (SCR)

Bit	Name	Function	R/W	Reset
7	RX_TRIG_GRANU1	0: Disables the granularity of 1 for trigger RX level. 1: Enables the granularity of 1 for trigger RX level.	R/W	0
6	TX_TRIG_GRANU1	0: Disables the granularity of 1 for trigger TX level. 1: Enables the granularity of 1 for trigger TX level.	R/W	0
5	DSR_IT	0: Disables $\overline{DSR}$ interrupt. 1: Enables $\overline{DSR}$ interrupt.	R/W	0
4	RX_CTS_DSR_ WAKE_UP_ENABLE	0: Disables the wake-up interrupt and clears SSR[1]. 1: Waits for a falling edge of pins RX, $\overline{CTS}$ or $\overline{DSR}$ to generate an interrupt.	R/W	0
3	TX_EMPTY_CTL_IT	0: Normal mode for THR interrupt (See UART mode interrupts table.) 1: The THR interrupt is generated when TX FIFO and TX shift register are empty.	R/W	0
2:1	DMA_MODE_2	Used to specify the DMA mode valid if SCR[0] = 1 00: DMA mode 0 (no DMA) 01: DMA mode 1 (UART_nDMA_REQ[0] in TX, UART_nDMA_REQ[1] in RX) 10: DMA mode 2 (UART_nDMA_REQ[0] in RX) 11: DMA mode 3 (UART_nDMA_REQ[0] in TX)	R/W	00
0	DMA_MODE_CTL	0: The DMA_MODE is set with FCR[3]. 1: The DMA_MODE is set with SCR[2:1].	R/W	0

Bit 4 enables the wake-up interrupt, but this interrupt is not mapped into the IIR register. Therefore, when an interrupt occurs and there is no interrupt pending in the IIR register, the SSR[1] bit must be checked. To clear the wake-up interrupt, bit SCR[4] must be reset to 0.

Offset Address (hex): 0x03 x S

LCR[6:0] defines parameters of the transmission and reception.

Table 68. Line Control Register (LCR)

Bit	Name	Function	R/W	Reset
7	DIV_EN	0: Normal operating condition. 1: Divisor latch enable. Allows access to DLL, DLH, and other registers (refer to the registers' mapping).	R/W	0
6	BREAK_EN	Break control bit 0: Normal operating condition. 1: Forces the transmitter output to go low to alert the communication terminal.	R/W	0
5	PARITY_TYPE2		R/W	0
4	PARITY_TYPE1	0: Odd parity is generated (if LCR[3] = 1). 1: Even parity is generated (if LCR[3] = 1).	R/W	0
3	PARITY_EN	0: No parity. 1: A parity bit is generated during transmission and the receiver checks for received parity.	R/W	0
2	NB_STOP	Specifies the number of stop bits: 0: 1 stop bits (word length = 5, 6, 7, 8) 1: 1.5 stop bits (word length = 5) 1–2 stop bits (word length = 6, 7, 8)	R/W	0
1:0	CHAR_LENGTH	Specifies the word length to be transmitted or received 00: 5 bits 01: 6 bits 10: 7 bits 11: 8 bits	R/W	00

Offset Address (hex): 0x05 x S and LCR is not 0xBF and read

Table 69. Line Status Register (LSR) (UART Mode)

Bit	Name	Function	R/W	Reset
7	RX_FIFO_STS	0: Normal operation. 1: At least one parity error, framing error, or break indication in the receiver FIFO. Bit 7 is cleared when no more errors are present in the FIFO.	R	0
6	TX_SR_E	0: Transmitter hold and shift registers are not empty. 1: Transmitter hold and shift registers are empty.	R	1
5	TX_FIFO_E	0: Transmit hold register is not empty. 1: Transmit hold register is empty. The processor can now load up to 64 bytes of data into the THR if the TX FIFO is enabled.	R	1
4	RX_BI	0: No break condition. 1: A break was detected while the data being read from the RX FIFO was being received (that is, RX input was low for one character time frame).	R	0
3	RX_FE	0: No framing error in data being read from RX FIFO. 1: Framing error occurred in data being read from RX FIFO (received data did not have a valid stop bit).	R	0
2	RX_PE	0: No parity error in data being read from RX FIFO. 1: Parity error in data being read from RX FIFO.	R	0
1	RX_OE	0: No overrun error. 1: Overrun error has occurred. Set when the character held in the receive shift register is not transferred to the RX FIFO. This case occurs only when receive FIFO is full.	R	0
0	RX_FIFO_E	0: No data in the receive FIFO. 1: At least one data character in the RX_FIFO.	R	0

When the LSR is read, LSR[4:2] reflects the error bits [BI, FE, 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, PE] (See Table 109, *UART Mode Interrupts*.)

LSR [7] is set when there is an error anywhere in the RX FIFO, and is cleared only when there are no more errors remaining in the FIFO.

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 LSR clears [OE] if set (See Table 109, UART Mode Interrupts.)

Table 70. Line Status Register (LSR) (IR Mode)

Bit	Name	Function	R/W	Reset
7	THR_EMPTY	0: Transmit holding register is not empty. 1: Transmit holding register is empty. The processor can now load up to 64 bytes of data into the THR if the TX FIFO is enabled.	R	1
6	STS_FIFO_FULL	0: Status FIFO not full. 1: Status FIFO full.	R	0
5	RX_LAST_BYTE	0: The RX FIFO does not contain the last byte of the frame to be read. 1: The RX FIFO contains the last byte of the frame to be read. This bit is set only when the last byte of a frame is available to be read. It is used to determine the frame boundary. It is cleared on a single read of the LSR register.	R	0
4	FRAME_TOO_LONG	0: No frame-too-long error in frame. 1: 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
3	ABORT	0: No abort pattern error in frame. 1: Abort pattern is received. SIR & MIR: Abort pattern FIR: Illegal symbol 0000	R	0
2	CRC	0: No CRC error in frame. 1: CRC error in the frame at the top of the STATUS FIFO (next character to be read).	R	0

Table 70. Line Status Register (LSR) (IR Mode) (Continued)

Bit	Name	Function	R/W	Reset
1	STS_FIFO_E	0: Status FIFO not empty. 1: Status FIFO empty.	R	1
0	RX_FIFO_E	0: At least one data character in the RX_FIFO. 1: No data in the receive FIFO.	R	1

When the LSR is read, LSR[4:2] reflects the error bits [FL, CRC, ABORT] of the frame at the top of the STATUS FIFO (next frame status to be read). See Table 110, *IrDA Mode Interrupts*.

Offset Address (hex): 0x11 x S and read

Table 71. Supplementary Status Register (SSR)

Bit	Name	Function	R/W	Reset
7:2	–	Reserved	R	000000
1	RX_CTS_DSR_WAKE_UP_STS	0: No falling edge event on RX, $\overline{\text{CTS}}$ and $\overline{\text{DSR}}$. 1: A falling edge occurred on RX, $\overline{\text{CTS}}$ or $\overline{\text{DSR}}$.	R	0
0	TX_FIFO_FULL	0: TX FIFO not full. 1: TX FIFO full.	R	0

Bit 1 is reset only when SCR[4] is reset to 0.

Offset Address (hex): 0x04 x S and LCR is not 0xBF (and EFR[4] = 1 for MCR[7:5])

MCR[3:0] controls the interface with the modem, data set, or peripheral device that is emulating the modem.

Table 72. Modem Control Register (MCR)

Bit	Name	Function	R/W	Reset
7	–	Reserved.	R	0
6	TCR_TLR	0: No action. 1: Enables access to the TCR and TLR registers.	R/W	0
5	XON_EN	0: Disable the *XON any* function. 1: Enable the *XON any* function.	R/W	0

Table 72. Modem Control Register (MCR) (Continued)

Bit	Name	Function	R/W	Reset
4	LOOPBACK_EN	0: Normal operating mode. 1: 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.	R/W	0
3	CD_STS_CH	0: In loopback, forces $\overline{DCD}$ input high and IRQ outputs to inactive state. 1: In loopback, forces $\overline{DCD}$ input low and IRQ outputs to inactive state.	R/W	0
2	RI_STS_CH	0: In loopback, forces $\overline{RI}$ input high. 1: In loopback, forces $\overline{RI}$ input low.	R/W	0
1	RTS	0: Force $\overline{RTS}$ output to inactive (high). 1: Force $\overline{RTS}$ output to active (low). In loopback, controls MSR[4] If auto-RTS is enabled, the $\overline{RTS}$ output is controlled by hardware flow control.	R/W	0
0	DTR	0: Force $\overline{DTR}$ output to inactive (high). 1: Force $\overline{DTR}$ output to active (low).	R/W	0

Bits 5 and 6 can be written only when EFR[4] = 1.

Offset Address (hex): 0x06 x S and LCR is not 0xBF and (EFR[4] = 0 or MCR[6] = 0) and read.

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.

Table 73. Modem Status Register (MSR)

Bit	Name	Function	R/W	Reset
7	NCD_STS	This bit is the complement of the $\overline{DCD}$ input. In loopback mode it is equivalent to MCR[3].	R	Unknown
6	NRI_STS	This bit is the complement of the $\overline{RI}$ input. In loopback mode it is equivalent to MCR[2].	R	Unknown
5	NDSR_STS	This bit is the complement of the $\overline{DSR}$ input. In loopback mode, it is equivalent to MCR[0].	R	Unknown

Table 73. Modem Status Register (MSR) (Continued)

Bit	Name	Function	R/W	Reset
4	NCTS_STS	This bit is the complement of the $\overline{CTS}$ input. In loopback mode it is equivalent to MCR[1].	R	Unknown
3	DCD_STS	Indicates that $\overline{DCD}$ input (or MCR[3] in loopback) has changed. Cleared on a read.	R	0
2	RI_STS	Indicates that $\overline{RI}$ input (or MCR[2] in loopback) has changed state from low to high. Cleared on a read.	R	0
1	DSR_STS	1: Indicates that $\overline{DSR}$ input (or MCR[0] in loop-back) has changed state. Cleared on a read.	R	0
0	CTS_STS	1: Indicates that $\overline{CTS}$ input (or MCR[1] in loopback) has changed state. Cleared on a read.	R	0

6.3 Interrupt Enable Register (IER)

Offset Address (hex): 0x01 x S and LCR[7] = 0 (and EFR[4] = 1 for IER[7:4]—UART modes only)

UART Modes IER

The interrupt enable register (IER) can be programmed to enable/disable any interrupt. This mode has seven types of interrupt: Receiver error, RHR interrupt, THR interrupt, XOFF received, and $\overline{CTS}$/$\overline{RTS}$ change of state from low to high. Each interrupt can be enabled/disabled individually. The IER also has a sleep-mode enable bit.

Table 74. Interrupt Enable Register (IER) (UART Mode)

Bit	Name	Function	R/W	Reset
7	CTS_IT	0: Disables the $\overline{CTS}$ interrupt. 1: Enables the $\overline{CTS}$ interrupt.	R/W	0
6	RTS_IT	0: Disables the $\overline{RTS}$ interrupt. 1: Enables the $\overline{RTS}$ interrupt.	R/W	0
5	XOFF_IT	0: Disables the XOFF interrupt. 1: Enables the XOFF interrupt.	R/W	0
4	SLEEP_MODE	0: Disables sleep mode. 1: Enables sleep mode (stop baud rate clock when the module is inactive).	R/W	0

Table 74. Interrupt Enable Register (IER) (UART Mode) (Continued)

Bit	Name	Function	R/W	Reset
3	MODEM_STS_IT	0: Disables the modem status register interrupt. 1: Enables the modem status register interrupt.	R/W	0
2	LINE_STS_IT	0: Disables the receiver line status interrupt. 1: Enables the receiver line status interrupt.	R/W	0
1	THR_IT	0: Disables the THR interrupt. 1: Enables the THR interrupt.	R/W	0
0	RHR_IT	0: Disables the RHR interrupt and time-out interrupt. 1: Enables the RHR interrupt and time-out interrupt.	R/W	0

Bits 4, 5, 6, and 7 can only be written when EFR[4] = 1.

IrDA Modes IE

These modes have eight types of interrupts: received EOF, LSR interrupt, TX status, status FIFO interrupt, RX overrun, last byte in RX FIFO, THR interrupt, and RHR interrupt. All can be enabled/disabled individually.

Table 75. Interrupt Enable Register (IER) (IrDA Mode)

Bit	Name	Function	R/W	Reset
7	EOF_IT	0: Disables the received EOF interrupt. 1: Enables the received EOF interrupt.	R/W	0
6	LINE_STS_IT	0: Disables the receiver line status interrupt. 1: Enables the receiver line status interrupt.	R/W	0
5	TX_STATUS_IT	0: Disables the TX status interrupt. 1: Enables the TX status interrupt.	R/W	0
4	STS_FIFO_TRIG_IT	0: Disables status FIFO trigger level interrupt. 1: Enables status FIFO trigger level interrupt.	R/W	0
3	RX_OVERRUN_IT	0: Disables the RX overrun interrupt. 1: Enables the RX overrun interrupt.	R/W	0
2	LAST_RX_BYTE_IT	0: Disables the last byte of frame in RX FIFO interrupt. 1: Enables the last byte of frame in RX FIFO interrupt.	R/W	0

Table 75. Interrupt Enable Register (IER) (IrDA Mode) (Continued)

Bit	Name	Function	R/W	Reset
1	THR_IT	0: Disables the THR interrupt. 1: Enables the THR interrupt.	R/W	0
0	RHR_IT	0: Disables the RHR interrupt. 1: Enables the RHR interrupt.	R/W	0

The TX_STATUS_IT interrupt reflects two possible conditions. The MDR2[0] must be read to determine the status in the event of this interrupt.

Offset Address (hex): 0x02 x S and LCR is not 0xBF and read.

The IIR is a read-only register that provides the source of the interrupt in a prioritized manner.

Table 76. Interrupt Identification Register (IIR) (UART Mode)

Bit	Name	Function	R/W	Reset
7:6	FCR_MIRROR	Mirror the contents of FCR[0] on both bits	R	00
5:1	IT_TYPE		R	00000
0	IT_PENDING	0: An interrupt is pending (UART_nIRQ active). 1: No interrupt is pending (UART_nIRQ inactive).	R	1

The UART_nIRQ output is activated whenever one of the eight interrupts is active.

Table 77. IrDA Mode Register (IIR)

Bit	Name	Function	R/W	Reset
7	EOF_IT	0: Received EOF interrupt inactive. 1: Received EOF interrupt active.	R	0
6	LINE_STS_IT	0: Receiver line status interrupt inactive. 1: Receiver line status interrupt active.	R	0
5	TX_STATUS_IT	0: TX status interrupt inactive. 1: TX status interrupt active.	R	0
4	STS_FIFO_IT	0: Status FIFO trigger level interrupt inactive. 1: Status FIFO trigger level interrupt active.	R	0

Table 77. IrDA Mode Register (IIR) (Continued)

Bit	Name	Function	R/W	Reset
3	RX_OE_IT	0: RX overrun interrupt inactive. 1: RX overrun interrupt active.	R	0
2	RX_FIFO_LAST_BYTE_IT	0: Last byte of frame in RX FIFO interrupt inactive. 1: Last byte of frame in RX FIFO interrupt active.	R	0
1	THR_IT	0: THR interrupt inactive. 1: THR interrupt active.	R	0
0	RHR_IT	0: RHR interrupt inactive. 1: RHR interrupt active.	R	0

Offset Address (hex): 0x02 x S and LCR = 0xBF

This register enables or disables enhanced features. Most of the enhanced functions apply only to UART modes, but EFR[4] enables write access to FCR[5:4], the TX trigger level, which is also used in IrDA modes.

Table 78. Enhanced Feature Register (EFR)

Bit	Name	Function	R/W	Reset
7	AUTO_CTS_EN	Auto-CTS enable bit 0: Normal operation. 1: Auto-CTS flow control is enabled, that is, transmission is halted when the $\overline{CTS}$ pin is high (inactive).	R/W	0
6	AUTO_RTS_EN	Auto-RTS enable bit 0: Normal operation. 1: Auto-RTS flow control is enabled. $\overline{RTS}$ 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.	R/W	0
5	SPECIAL_CHAR_DETECT	0: Normal operation. 1: Special character detect enable. Received data is compared with XOFF2 data. If a match occurs the received data is transferred to FIFO, and IIR bit 4 is set to 1 to indicate that a special character has been detected.	R/W	0

Table 78. Enhanced Feature Register (EFR) (Continued)

Bit	Name	Function	R/W	Reset
4	ENHANCED_EN	Enhanced functions write enable bit 0: Disables writing to IER bits 4–7, FCR bits 4–5, and MCR bits 5–7. 1: Enables writing to IER bits 4–7, FCR bits 4–5, and MCR bits 5–7.	R/W	0
3:0	SW_FLOW_ CONTROL	Combinations of software flow control can be selected by programming bits 3–0. See Table 79.	R/W	0000

Table 79. Software Flow Control Options(EFR[0–3])

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
X	X	0	0	No receive flow control
X	X	1	0	Receiver compares XON1, XOFF1
X	X	1	1	Receiver compares XON2, XOFF2
X	X	1	1	Receiver compares XON1, XON2: XOFF1, XOFF2†

† XON1 and XON2 must be set to different values if software flow control is enabled.

Offset Address (hex): 0x04 x S and LCR = 0xBF

Table 80. XON1/ADDR1 Register

Bit	Name	Function	R/W	Reset
7:0	XON_WORD1	Used to store the 8-bit XON1 character in UART modes and ADDR1 address 1 for IrDA modes	R/W	0x00

Offset Address (hex): 0x05 x S and LCR = 0xBF

Table 81. XON2/ADDR2 Register

Bit	Name	Function	R/W	Reset
7:0	XON_WORD2	Used to store the 8-bit XON2 character in UART modes and ADDR2 address 2 for IrDA modes	R/W	0x00

Offset Address (hex): 0x06 x S and LCR = 0xBF and (EFR[4] = 0 or MCR[6] = 0)

Table 82. XOFF1 Register

Bit	Name	Function	R/W	Reset
7:0	XOFF_WORD1	Used to store the 8-bit XOFF1 character used in UART modes	R/W	0x00

Offset Address (hex): 0x07 x S and LCR = 0xBF and (EFR[4] = 0 or MCR[6] = 0)

Table 83. XOFF2 Register

Bit	Name	Function	R/W	Reset
7:0	XOFF_WORD2	Used to store the 8-bit XOFF2 character used in UART modes	R/W	0x00

Offset Address (hex): 0x07 x S and LCR is not 0xBF and (EFR[4] = 0 or MCR[6] = 0)

This R/W register does not control the module in any way. It is a scratchpad register the programmer can use to hold temporary data.

Table 84. Scratchpad Register (SPR)

Bit	Name	Function	R/W	Reset
7:0	SPR_WORD	Scratchpad register	R/W	0x00

6.3.1 Divisor Latches (DLL, DLH)

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.

DLL and DLH can only be written to before sleep mode is enabled, that is, before IER[4] is set.

Offset Address (hex): 0x00 x S and LCR[7] = 1

Table 85. Divisor Latches LowRegister (DLL)

Bit	Name	Function	R/W	Reset
7:0	CLOCK_LSB	Used to store the 8-bit LSB divisor value	R/W	0x00

Offset Address (hex): 0x01 x S and LCR[7] = 1

Table 86. Divisor Latches High Register (DLH)

Bit	Name	Function	R/W	Reset
7:6	–	Reserved	R	00
5:0	CLOCK_MSB	Used to store the 6-bit MSB divisor value	R/W	000000

Offset Address (hex): 0x06 x S and EFR[4] = 1 and MCR[6] = 1

This register stores the receive FIFO threshold levels to start/stop transmission during hardware/software flow control.

Table 87. Transmission Control Register (TCR)

Bit	Name	Function	R/W	Reset
7:4	RX_FIFO_TRIG_START	RCV FIFO trigger level to RESTORE transmission (0 – 60)	R/W	0x0
3:0	RX_FIFO_TRIG_HALT	RCV FIFO trigger level to HALT transmission (0 – 60)	R/W	0xF

Notes: 1) Trigger levels from 0 – 60 bytes are available with a granularity of 4. (Trigger level = 4 x [4-bit register value])

2) The programmer must ensure that TCR[3:0] > TCR[7:4] whenever auto-RTS or software flow control is enabled, to prevent device malfunction.

3) In FIFO interrupt mode with flow control, the programmer also must ensure that the trigger level to HALT transmission is greater than or equal to the receive FIFO trigger level (either TLR[7:4] or FCR[7:6]). Otherwise, the FIFO operation stalls. This problem does not exist in FIFO DMA mode with flow control because a DMA request is sent each time a byte is received.

Offset Address (hex): 0x07 x S and EFR[4] = 1 and MCR[6] = 1

This register stores the programmable transmit and receive FIFO trigger levels used for DMA and IRQ generation.

Table 88. Trigger Level Register (TLR)

Bit	Name	Function	R/W	Reset
7:4	RX_FIFO_TRIG_DMA	RCV FIFO trigger level	R/W	0x0
3:0	TX_FIFO_TRIG_DMA	Transmit FIFO trigger level	R/W	0x0

Table 89 and Table 90 summarize the different ways to set the trigger levels for the transmit FIFO and the receive FIFO, respectively.

Table 89. TX FIFO Trigger Level Setting Summary

SCR[6]	TLR[3:0]	TX FIFO Trigger Level
0	0000	Defined by FCR[5:4] (either 8,16,32, 56 spaces)
0	Non-zero	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 (min 1 space required). All zeros result in unpredictable behavior.

Table 90. RX FIFO Trigger Level Setting Summary

SCR[7]	TLR[7:4]	RX FIFO Trigger Level
0	0000	Defined by FCR[7:6] (either 8,16,56, 60 characters)
0	Non-zero	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 (min 1 character required). All zeros result in unpredictable behavior.

Offset Address (hex): 0x08 x S

The mode of operation is programmed by writing to MDR1[2:0]. Therefore, the MDR1 must be programmed on start-up after configuring registers DLL, DLH, and LCR. The value of MDR1[2:0] must not be changed again during normal operation.

Table 91. Mode Definition Register 1 (MDR1)

Bit	Name	Function	R/W	Reset
7	FRAME_END_MODE	0: Frame-length method.	R/W	0
		1: Set EOT bit method.		
6	SIP_MODE	MIR/FIR modes only	R/W	0
		0: Manual SIP mode: SIP is generated with the control of ACREG[3].		
		1: Automatic SIP mode: SIP is generated after each transmission.		

Table 91. Mode Definition Register 1 (MDR1) (Continued)

Bit	Name	Function	R/W	Reset
5	SCT	Store and control the transmission	R/W	0
		0: Starts the IrDA transmission as soon as a value is written to THR.		
		1: Starts the IrDA transmission with the control of ACREG[2].		
4	SET_TXIR	Used to configure the IrDA transceiver	R/W	0
		0: No action.		
		1: TXIR pin output is forced high.		
3	IR_SLEEP	0: IrDA sleep mode disabled.	R/W	0
		1: IrDA sleep mode enabled.		
2:0	MODE_SELECT	000: UART 16x mode	R/W	111
		001: SIR mode		
		010: UART 16x autobaud		
		011: UART 13x mode		
		100: MIR mode		
		101: FIR mode		
		110: Reserved		
		111: Disable (default state)		

Offset Address (hex): 0x09 x S

IrDA modes only.

MDR2[0] describes the status of the interrupt in IIR[5]. The IRTX_UNDERRUN bit should 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].

Table 92. Mode Definition Register 2 (MDR2)

Bit	Name	Function	R/W	Reset
7:3	–	Reserved	R	00000
2:1	STS_FIFO_TRIG	Frame status FIFO threshold select: 00: 1 entry 01: 4 entries 10: 7 entries 11: 8 entries	R/W	00
0	IRTX_UNDERRUN	IRDA transmission status interrupt When the IIR[5] interrupt occurs, the meaning of the interrupt is: 0: IRTX last bit of the frame has been transmitted successfully without error. 1: IRTX 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

Offset Address (hex): 0x0E x S and LCR[7] = 1 and read

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.

Table 93. UART Autobauding Status Register (UASR)

Bit	Name	Function	R/W	Reset
7:6	PARITY_TYPE	00: No parity identified 01: Parity space 10: Even parity 11: Odd parity	R	00
5	BIT_BY_CHAR	0: 7 bits character identified. 1: 8 bits character identified.	R	0
4:0	SPEED	Used to report the speed identified 00000: No speed identified 00001: 115 200 bauds 00010: 57 600 bauds 00011: 38 400 bauds 00100: 28 800 bauds 00101: 19 200 bauds 00110: 14 400 bauds 00111: 9 600 bauds 01000: 4 800 bauds 01001: 2 400 bauds 01010: 1 200 bauds	R	00000

This register sets up transmission according to characteristics of previous reception instead of the LCR, DLL, and DLH registers used when UART is in autobauding mode.

To reset the autobauding hardware (to start a new AT detection) or to set the UART in standard mode (no autobaud), MDR1[2:0] must be set to reset state 111, and then to the UART in autobaud mode 010 or UART in standard mode 000.

Usage limitation:

- Only 7- and 8-bit character (5 and 6 bits not supported)
- 7-bit character with space parity not supported
- Baud rate between 1200 and 115200 bp/s (10 possibilities)

6.4 Transmit Frame Length Register (TXFLL, TXFLH)

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.

Offset Address (hex): 0x0A x S and write

Table 94. Transmit Frame Length Low Register (TXFLL)

Bit	Name	Function	R/W	Reset
7:0	TXFLL	LSB register used to specify the frame length.	W	0x00

Offset Address (hex): 0x0B x S and write

Table 95. Transmit Frame Length High Register (TXFLH)

Bit	Name	Function	R/W	Reset
7:5	–	Reserved	R	000
4:0	TXFLH	MSB register used to specify the frame length.	W	00000

6.4.1 Received Frame Length Register (RXFLL, RXFLH)

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).

Offset Address (hex): 0x0C x S and write

Table 96. Received Frame Length Low Register (RXFLL)

Bit	Name	Function	R/W	Reset
7:0	RXFLL	LSB register used to specify the frame length in reception	W	0x00

Offset Address (hex): 0x0D x S and write

Table 97. Received Frame Length High Register (RXFLH)

Bit	Name	Function	R/W	Reset
7:4	–	Reserved	R	0x0
3:0	RXFLH	MSB register used to specify the frame length in reception	W	0x0

Offset Address (hex): 0x0A x S and read

IrDA modes only.

Reading this register in effect 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).

Table 98. Status FIFO Line Status Register (SFLSR)

Bit	Name	Function	R/W	Reset
7:5	–	Reserved	R	000
4	OE_ERROR	1: Overrun error in RX FIFO when frame at top of FIFO was received.	R	Unknown
3	FRAME_TOO_LONG_ ERROR	1: Frame-length too long error in frame at top of FIFO.	R	Unknown
2	ABORT_DETECT	1: Abort pattern detected in frame at top of FIFO.	R	Unknown
1	CRC_ERROR	1: CRC error in frame at top of FIFO.	R	Unknown
0	–	Reserved	R	0

Offset Address (hex): 0x0B x S and read

IrDA modes only.

This register clears internal flags that 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.

Table 99. Resume Register (RESUME)

Bit	Name	Function	R/W	Reset
7:0	RESUME	Dummy read to restart the TX or RX	R	0x00

6.4.2 Status FIFO Register (SFREGL, SFREGH)

IrDA modes only. The frame lengths of received frames are written into the status FIFO. This information can be read from 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.

Offset Address (hex): 0x0C x S and read

Table 100. Status FIFO Register Low (SFREGL)

Bit	Name	Function	R/W	Reset
7:0	SFREGL	LSB part of the frame length	R	Unknown

Offset Address (hex): 0x0D x S and read

Table 101. Status FIFO Register High (SFREGH)

Bit	Name	Function	R/W	Reset
7:4	–	Reserved	R	0x0
3:0	SFREGH	MSB part of the frame length	R	Unknown

Offset Address (hex): 0x0E x S and LCR[7] = 0

IrDA modes only. 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, the start pattern is always 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.

Table 102. BOF Control Register (BLR)

Bit	Name	Function	R/W	Reset
7	STS_FIFO_RESET	Status FIFO reset. This bit is self-clearing.	R/W	0
6	XBOF_TYPE	SIR xBOF select 0: 0xFF. 1: 0xC0.	R/W	1
5:0	–	Reserved	R	000000

Offset Address (hex): 0x12 x S and LCR[7] = 0

IrDA modes only.

This register specifies the number of BOF + xBOFs to transmit in IrDA SIR operation. The value set into this register must take into account the BOF character. 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 IrDA MIR mode, this register specifies the number of additional start flags (MIR protocol mandates a minimum of two start flags).

Table 103. BOF Length Register (EBLR)

Bit	Name	Function	R/W	Reset
7:0	EBLR	This register allows definition up to 176 xBOFs, the maximum required by IrDA specification.	R/W	0x00

Offset Address (hex): 0x0F x S and LCR[7] = 0

IrDA modes only.

Table 104. Auxiliary Control Register (ACREG)

Bit	Name	Function	R/W	Reset
7	PULSE_TYPE	SIR pulse width select 0: 3/16 of baud-rate pulse width. 1: 1.6 µs.	R/W	0
6	SD_MOD	Primary output used to configure transceivers. Connected to the SD/MODE input pin of IrDA transceivers 0: SD pin is set to high. 1: SD pin is set to low.	R/W	0
5	DIS_IR_RX	0: Normal operation (Note: RXIR input automatically disabled during transmit but enabled outside of transmit operation.) 1: Disables RXIR input (permanent state–independent of transmit). Hence, RX_IR is disabled when either TX is active or ACREG[5] = 1	R/W	0
4	DIS_TX_UNDERRUN	0: Long stop bits cannot be transmitted, and TX underrun is enabled. 1: Long stop bits can be transmitted, and TX underrun is disabled.	R/W	0

Table 104. Auxiliary Control Register (ACREG) (Continued)

Bit	Name	Function	R/W	Reset
3	SEND_SIP	MIR/FIR modes only Send serial infrared interaction pulse (SIP) 0: No action. 1: Send SIP pulse. If this bit is set during a MIR/FIR transmission, the SIP is sent at the end of it. This bit is automatically cleared at the end of the SIP transmission.	R/W	0
2	SCTX_EN	Store and controlled 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.	R/W	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.	R/W	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 set-EOT bit frame closing method. This bit is automatically cleared when the LH writes to the THR (TX FIFO).	R/W	0

Offset Address (hex): 0x14 x S and read

Table 105. Module Version Register (MVR)

Bit	Name	Function	R/W	Reset
7:4	MAJOR_REV	Major revision number of the module	R	1
3:0	MINOR_REV	Minor revision number of the module	R	–

Note: 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 (**this specification)**.

Offset Address (hex):0x15 x S

The autoidle bit controls a power-saving technique to reduce the logic power consumption of the OCP interface. That is, 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.

Table 106. System Configuration Register (SYSC)

Bit	Name	Function	R/W	Reset
7:5	–	Reserved	R	000
4:3	IdleMode	Power management request/acknowledge 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. Ref: OCP design guidelines version 1.1	R/W	00
2	EnaWakeUp	Wake-up feature control 0: Wake up is disabled. 1: Wake-up capability is enabled.	R/W	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. 0: Normal mode. 1: The module is reset.	R/W	0
0	AutoIdle	Internal OCP clock gating strategy 0: Clock is running. 1: Automatic OCP clock gating strategy is applied, based on the OCP interface activity.	R/W	0

Offset Address (hex): 0x16 x S

Table 107. System Status Register (SYSS)

Bit	Name	Function	R/W	Reset
7:1	–	Reserved	R	0000000
0	ResetDone	Internal reset monitoring 0: Internal module reset is ongoing. 1: Reset completed.	R	0

Offset Address (hex): 0x17 x S

The UART wake-up enable register masks and unmasks a UART event that would subsequently notify the system. Such events are any activity in the logic that can cause an interrupt and/or any that require the system to wake up. Even if the wake-up is disabled for certain events, if these events also interrupt the UART, the UART still registers the interrupt as such.

Table 108. Wake-Up Enable Register (WER)

Bit	Name	Function	R/W	Reset
7	–	Reserved	R	0
6	Event 6 Receiver line status interrupt	0: Event is not allowed to wake up the system. 1: Event can wake up the system.	W/R	1
5	Event 5 RHR interrupt	0: Event is not allowed to wake up the system. 1: Event can wake up the system.	W/R	1
4	Event 4 RX/ RXIR activity	0: Event is not allowed to wake up the system. 1: Event can wake up the system.	W/R	1
3	Event 3 DCD (CD) activity	0: Event is not allowed to wake up the system. 1: Event can wake up the system.	W/R	1
2	Event 2 RI activity	0: Event is not allowed to wake up the system. 1: Event can wake up the system.	W/R	1
1	Event 1 DSR activity	0: Event is not allowed to wake up the system. 1: Event can wake up the system.	W/R	1
0	Event 0 CTS activity	0: Event is not allowed to wake up the system. 1: Event can wake up the system.	W/R	1

6.5 Different Modes of Operation

The UART/IRDA module can operate in six different modes:

1) UART 16x mode (≤230.4K bits/s)
2) UART 16x mode with autobauding (≥1200 bits/s and ≤115.2K bits/s)
3) UART 13x mode (≥460.8K bits/s)
4) IrDA SIR mode (≤115.2K bits/s)
5) IrDA MIR mode (0.576 and 1.152M bits/s)
6) IrDA FIR mode (4M bits/s)

The module performs serial-to-parallel conversion on received data characters, and parallel-to-serial conversion on data characters the processor transmits. The complete status of each channel of the module and each received character/frame can be read at any time during functional operation via the line status register (LSR).

The module can be placed in an alternate mode (FIFO mode), relieving the processor of excessive software overhead by buffering received/transmitted characters. Both the receiver and transmitter FIFOs store up to 64 bytes of data (plus 3 additional bits of error status per byte for the receiver FIFO) and have selectable trigger levels.

Both interrupts and DMA are available to control the data flow between the LH and the module.

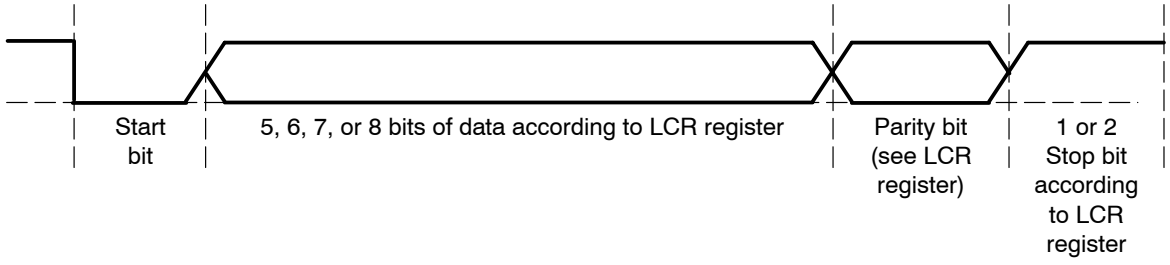
6.5.1 UART Modes

The UART uses a wired interface for serial communication with a remote device.

The module can use hardware or software flow control to manage transmission/ reception. Hardware flow control uses the RTS output and CTS input signals to automatically control serial data flow. This significantly reduces software overhead and increases system efficiency. Software flow control uses programmable XON/XOFF characters to automatically control data flow.

The UART modem module is enhanced with an autobauding functionality, which, in control mode, allows for automatic setting of the speed, number of bits per character, and parity.

Figure 62. UART Data Format



6.5.2 SIR Mode

In slow infrared (SIR) mode, data transfer takes place between the LH and peripheral devices at speeds up to 115200 bauds. A SIR transmit frame begins 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 ends with a stop flag (0xC1).

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 determine 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) first, 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 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 in the received frame.

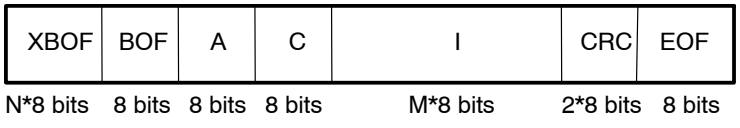
The module can transfer data both ways, but when the device is transmitting, hardware automatically disables the IR RX circuitry. Refer to Table 104, Auxiliary Control Register Bit 5, for a description of the logical operation of all three modes, SIR, MIR, and FIR.

The infrared output in SIR mode can be either 1.6 µs or 3/16 encoding, selected by the PULSE_TYPE bit of the auxiliary control register (ACREG[7]). In 1.6 µs encoding, the infrared pulse width is 1.6 µs, and 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 2 start flags at the start of each frame for back-to-back frames. Reception supports variable-length stop bits.

Frame Format

Figure 63. IrDA SIR Frame Format



The CRC is applied on the address (A), control (C), and information (I) bytes.

The two words of 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 BOF and EOF). For each byte equal to 0xC0 (BOF), 0xC1 (EOF), or 0x7D (control escape), it does the following.

In transmission:

- Inserts a control escape (CE) byte preceding the byte.
- Complements bit 5 of the byte (that is, exclusive ORs the byte with 0x20).

The byte sent for the CRC computation is the initial byte written in the TX FIFO (before the XOR with 0x20).

In reception:

For the A, C, I, and CRC fields:

- Compares the byte with the CE byte. If not equal, sends it to the CRC detector and stores it in the RX FIFO.
- If equal to CE, discards the CE byte.
- Complements bit 5 of the byte following the CE.
- Sends the complemented byte to the CRC detector and stores it in the RX FIFO.

Abort Sequence

The transmitter may decide to prematurely close a frame. The transmitter aborts by sending the 0x7DC1 sequence. The abort pattern closes the frame without a CRC field or an ending flag.

It is possible to abort a transmission frame by programming the ABORT_EN bit of the auxiliary control register (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 is received without transparency.

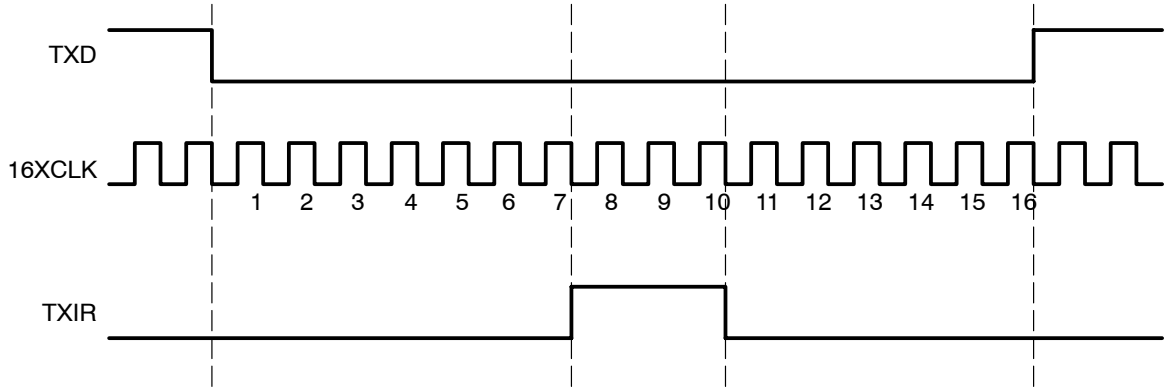
Pulse Shaping

In SIR mode both the 3/16 and the 1.6 ìs pulse duration methods are supported. ACREG[7] selects the pulse-width method in transmit mode.

Encoder

Serial data from the transmit state machine is encoded to transmit data to the optoelectronics. While the serial data input to the (TXD) is high, the output (TXIR) is always low, and the counter used to form a pulse on TXIR is continuously cleared. After TXD resets to 0, TXIR rises on the falling edge of the 7th 16XCLK. On the falling edge of the 10th 16XCLK pulse, TXIR falls, creating a 3-clock-wide pulse. While TXD stays low, a pulse is transmitted during the 7th to the 10th clocks of each 16-clock bit cycle.

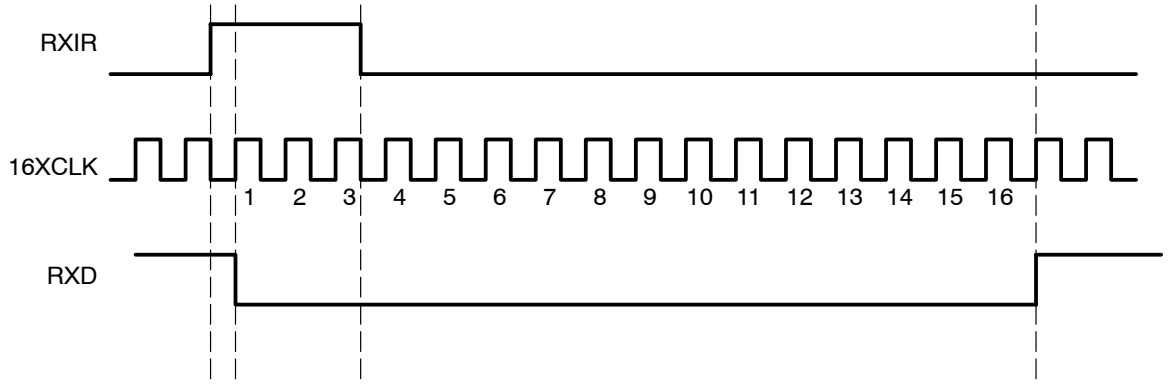
Figure 64. IrDA Encoder Mechanism



Decoder

After reset, RXD is high and the 4-bit counter is cleared. When a rising edge is detected on RXIR, 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 RXIR, RXD remains high.

Figure 65. IrDA Decoder Mechanism



The reception of RXIR input is disabled with DIS_IR_RX bits of the auxiliary control register (ACREG[5]).

IR Address Checking

In all IR modes, if address checking has been 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 multi-point infrared environment. It is possible to program two frame addresses that the UART IrDA receives with XON1/ADDR1 and XON2/ADDR2 registers.

Selecting address1 checking is done by setting EFR[0] to 1, and 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.

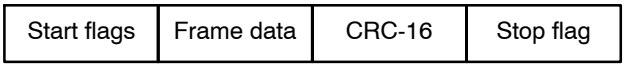
6.5.3 MIR Mode

In medium infrared (MIR) mode, data transfer takes place between the LH and peripheral devices at 0.576 or 1.152M bits/s speed. A MIR transmit frame begins with start flags (at least 2), followed by a frame data, CRC−16, and ends with a stop flag.

6.5.4 MIR Transmit Frame Format

On transmit, the MIR state machine attaches start flags, CRC−16, and stop flags. It also looks for 5 consecutive 1s in the frame data and automatically inserts 0 after them. (This is called bit stuffing.)

Figure 66. MIR Transmit Frame Format



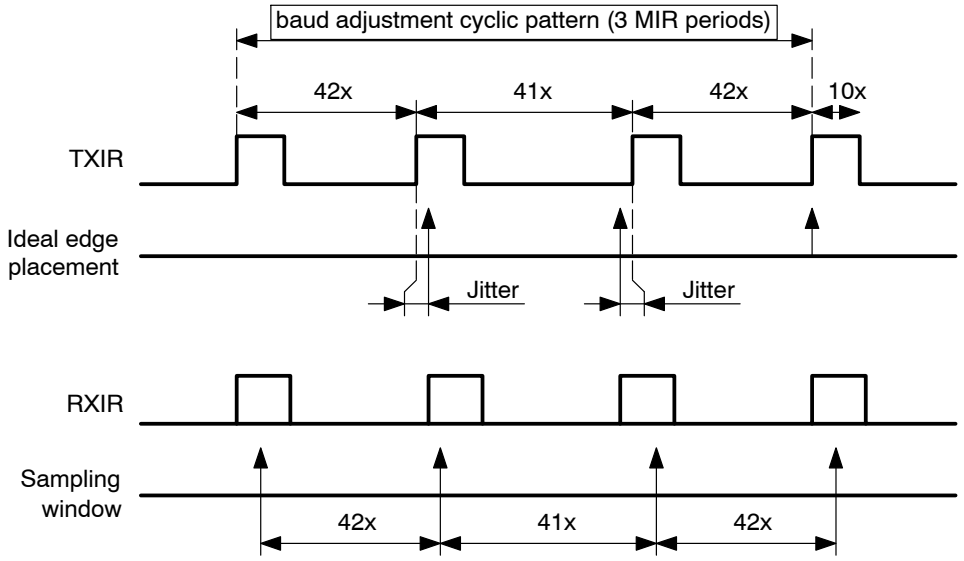
On receive, the MIR receive state machine recovers the receive clock, removes the start flags, destuffs the incoming data, and determines the frame boundary with reception of the stop flag. It also checks for errors such as frame abort, CRC error, or frame-length error. At the end of a frame reception, the LH reads the line status register (LSR) to discover possible errors in the received frame.

The module transfers data both ways, but when the device is transmitting, hardware automatically disables the IR RX circuitry. Refer to Table 104, Auxiliary Control Register Bit 5, for a description of the logical operation of all three modes, SIR, MIR, and FIR.

MIR Encoder/Decoder

In order to meet MIR baud-rate tolerance of ±0.1% 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. The jitter created this way is within MIR tolerances. The pulse width is not exactly 1/4 but within tolerances defined by the IrDA specifications.

Figure 67. MIR Baud-Rate Adjustment Mechanism

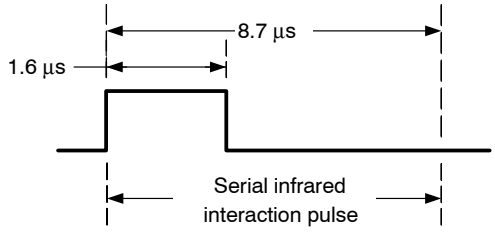


SIP Generation

In MIR and FIR operation modes, the transmitter needs to 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 currently occupied.

The SIP pulse is shown below in Figure 68.

Figure 68. SIP Pulse

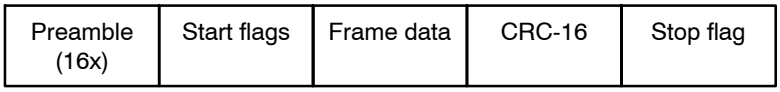


When the SIP_MODE bit of mode definition register 1 = 1 (MDR1[6]), the TX state machine always sends 1 SIP at the end of a transmission frame. But when MDR1[6] = 0, the transmission of the SIP depends on the SEND_SIP bit of the auxiliary control register (ACREG[3]). The local host can set ACREG[3] at least once every 500 ms. The advantage of this approach over the default is that the TX state machine does not need to send the SIP at the end of each frame, which may reduce the overhead required.

6.5.5 FIR Mode

In fast infrared mode (FIR), data transfer takes place between the LH and peripheral devices at 4M bits/s. A FIR transmit frame starts with a preamble, followed by a start flag, frame data, CRC−32, and ends with a stop flag.

Figure 69. FIR Transmit Frame Format



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 serial infrared interaction pulse (SIP).

On receive, the FIR receive state machine recovers the receive clock, removes the start flag, decodes the 4PPM 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 line status register (LSR) to discover possible errors in the received frame.

The module transfers data both ways, but when the device is transmitting, hardware automatically disables the IR RX circuitry. Refer to Table 104, Auxiliary Control Register Bit 5, for a description of the logical operation for all three modes, SIR, MIR, and FIR.

6.6 Functional Description

6.6.1 Trigger Levels

The UART provides programmable trigger levels for both receiver and transmitter DMA and interrupt generation. After reset, both transmitter and receiver FIFOs are disabled, so in effect the trigger level is the default value of 1 byte. The programmable trigger levels are an enhanced feature available via the trigger level register (TLR).

6.6.2 Interrupts

The UART/IrDA module generates interrupts on the UART_nIRQ output pin. All interrupts are enabled/disabled by writing to the appropriate bit in the interrupt enable register (IER). The interrupt status of the device can be checked at any time by reading the interrupt identification register (IIR).

The UART and IrDA modes have different interrupts in the UART/IrDA module, and therefore different IER and IIR mappings according to the selected mode.

UART Mode Interrupts

The UART modes have seven possible interrupts. These interrupts are prioritized to six different levels.

When an interrupt is generated, the interrupt identification register (IIR) indicates that an interrupt is pending by bringing IIR[0] to 0, and provides the type of interrupt through IIR[5−1]. It also summarizes the interrupt control functions.

Table 109. 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
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

Table 109. UART Mode Interrupts (Continued)

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, DSR	RTS pin, CTS pin or DSR pin change state from active (low) to inactive (high).	Read IIR

It is important to note that for the receiver line status interrupt, RX_FIFO_STS bit (LSR[7]) generates the interrupt.

For the XOFF interrupt, if an XOFF flow character detection caused the interrupt, an XON flow character detection clears the interrupt. If special character detection caused the interrupt, a read of the IIR clears the interrupt.

IrDA Mode Interrupts

IrDA modes have eight possible interrupts. The UART_nIRQ output is activated when any of the eight interrupts is generated (there is no priority).

Table 110. IrDA 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	Last byte in RX FIFO	Last byte of frame in RX FIFO	Read IIR
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

Table 110. IrDA Mode Interrupts (Continued)

5	TX status	1. THR empty before EOF sent. Last bit of transmission of the IRDA frame has occurred but with an underrun error. OR 2. Transmission of the last bit of the IRDA frame is finished successfully.	1. Read RESUME register. OR 2. Read IIR
6	Receiver line status interrupt	CRC, ABORT or frame-length error is written into STATUS FIFO	Read STATUS FIFO [Read until empty—max 8 reads required]
7	Received EOF	Received end-of-frame.	Read IIR

For IIR[5] the interrupt source 1 is used with interrupt reset method 1. The interrupt source 2 is used with interrupt reset method 2.

Wake-Up Interrupt

Wake-up interrupt is a special interrupt, not designed the same as the previous ones. It is enabled when the RX_CTS_DSR_WAKE_UP_ENABLE bit of the supplementary control register (SCR[4]) is set to 1. The IIR register 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.

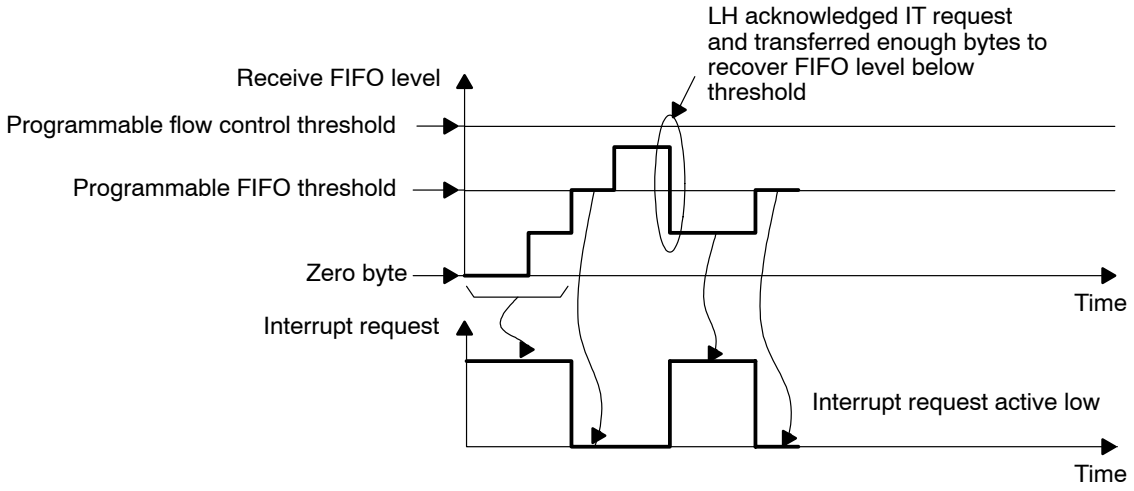
6.6.3 FIFO Interrupt Mode Operation

In FIFO interrupt mode (FIFO control register FCR[0] = 1, relevant interrupts enabled via IER), an interrupt signal (UART_nIRQ) informs the processor of the receiver and transmitter status. These interrupts are raised when receive/transmit FIFO thresholds (respectively, TLR[7:4] and TLR[3:0], or FCR[7:6] and FCR[5:4]) are reached. The interrupt signals instruct the local host 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).

Note that when the UART flow control is enabled along with the interrupt capabilities, the user must ensure that the UART flow control FIFO threshold (TCR[3:0]) is greater than, or equal to, the receive FIFO threshold.

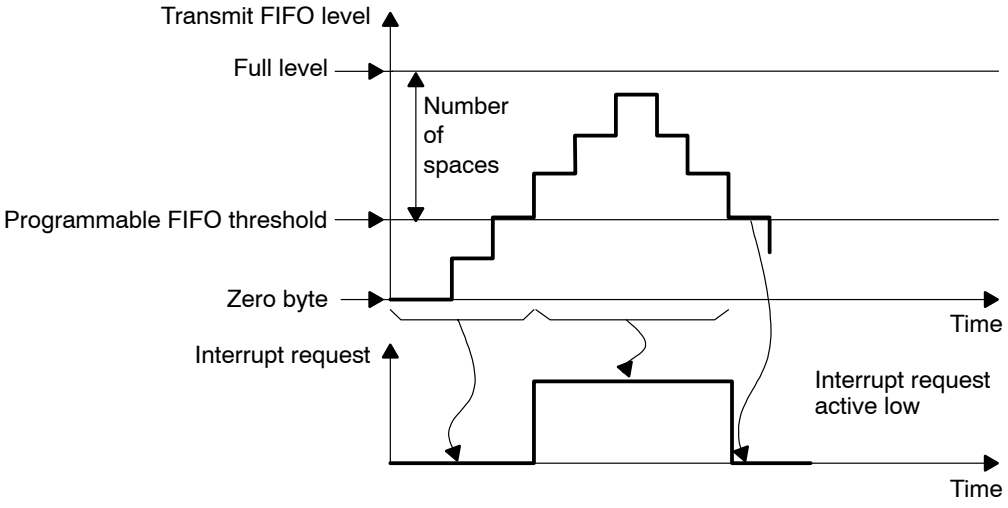
Figure 70 and Figure 71 respectively depict receive and transmit operations.

Figure 70. Receive FIFO IT Request Generation



In receive, no interrupt is generated until receive FIFO reaches its threshold. Once low, the interrupt can only be deasserted when the local host has handled enough bytes to make the FIFO level below threshold. The flow control threshold is set at a higher value than FIFO threshold.

Figure 71. Transmit FIFO IT Request Generation



In transmit mode, an interrupt request is automatically asserted when FIFO is empty. This request is deasserted when the FIFO crosses the threshold level. The interrupt line is deasserted until a sufficient number of elements has been transmitted to go below FIFO threshold.

6.6.4 FIFO Polled Mode Operation

In FIFO polled mode (FCR [0] = 0, relevant interrupts disabled via interrupt enable register (IER)), the status of the receiver and transmitter are checked by polling the line status register (LSR). This mode is an alternative to the FIFO interrupt mode of operation in which the status of the receiver and transmitter is automatically known by means of interrupts sent to the LH.

6.6.5 FIFO DMA Mode Operation

DMA Signaling

The four modes of DMA operation, DMA modes 0/1/2/3, are 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] determine DMA mode 0 to 3 according to supplementary control register (SCR) description.

For example:

- 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: Either set 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).

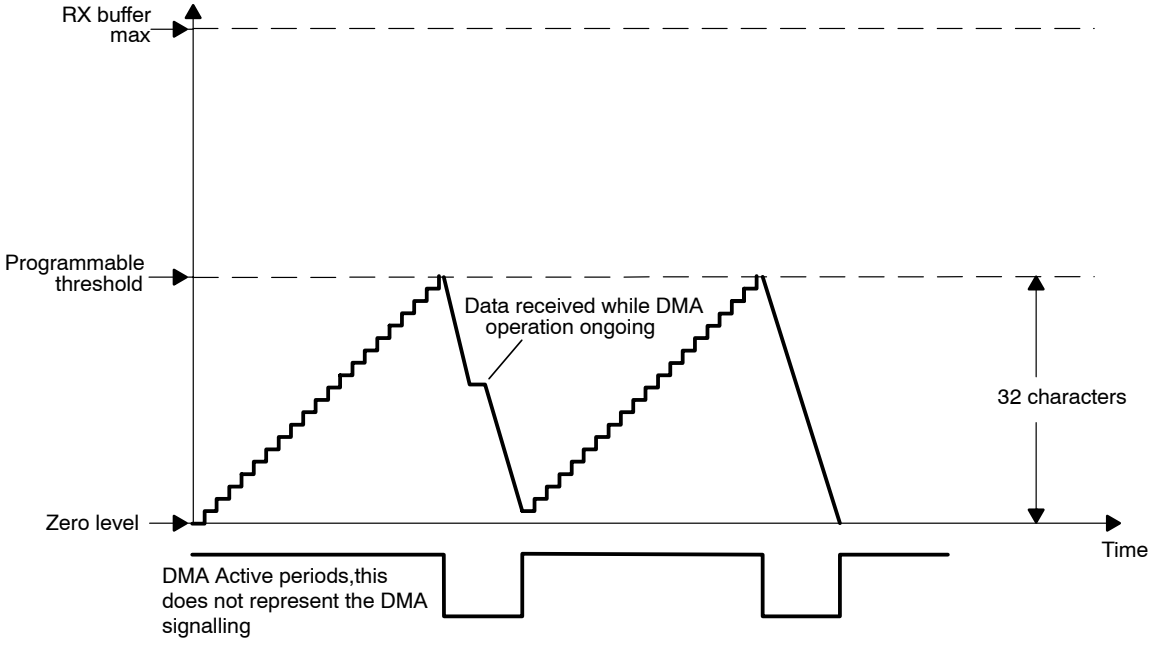
If the FIFOs are disabled (FCR[0] = 0), DMA occurs in single-character transfers.

When DMA mode 0 has been programmed, the signals associated with DMA operation are not active.

DMA Transfers (DMA Mode 1, 2, or 3)

Figure 72 through Figure 75 show the supported DMA operations.

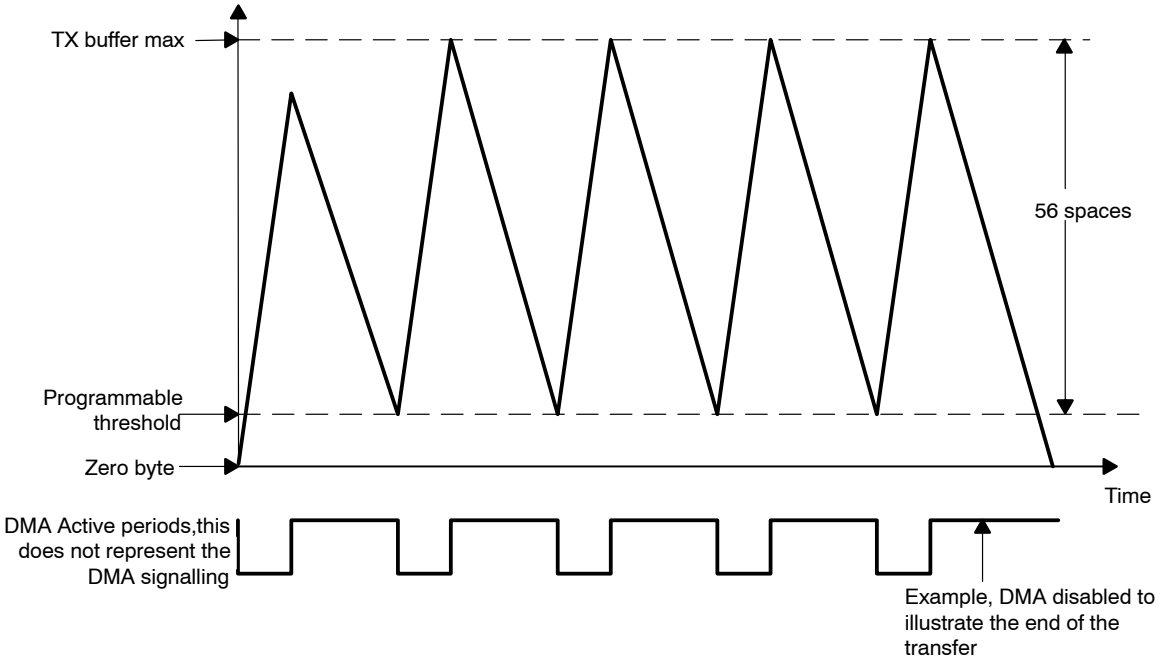
Figure 72. Receive FIFO DMA Request Generation (32 Characters)



In receive mode, a DMA request is generated as soon as the receive FIFO reaches its threshold level as defined in the trigger level register (TLR). (See Table 88.) This request is deasserted when the system DMA reads the number of bytes defined by the threshold level.

In transmit mode, a DMA request is automatically asserted when the FIFO is empty. This request is deasserted when the system DMA writes the number of bytes defined by the number of spaces in the trigger level register (TLR). If an insufficient number of characters is written, the DMA request remains active.

Figure 73. Transmit FIFO DMA Request Generation (56 Spaces)

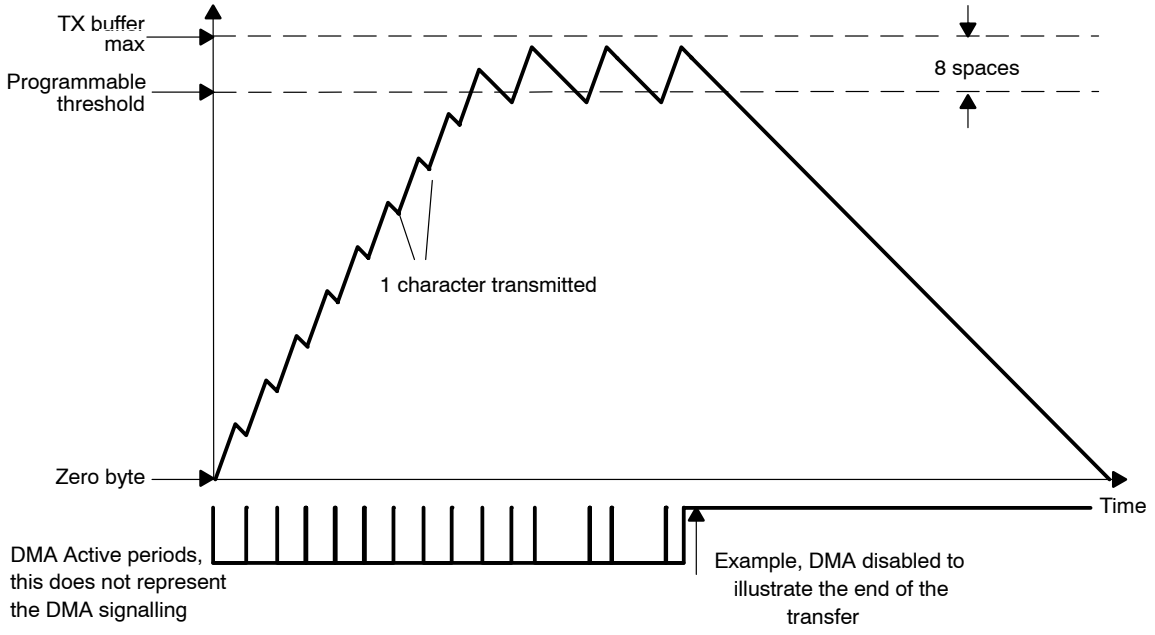


The DMA request is again asserted if the FIFO is able to receive the number of bytes defined by the TLR register. (See Table 88.)

The threshold can be programmed in a number of ways. See Figure 73 for an example of a DMA transfer that operates with a space setting of 56, which could arise from the use of the autosettings in the FCR[5:4] or the use of the TLR[3:0] concatenated with the FCR[5:4]. The setting of 56 spaces in the UART_IrDA should correlate with settings of the system DMA so that the buffer does not overflow (program the DMA request size of the local host controller to be equal to the number of spaces value in the UART).

Figure 74 shows another example with 8 spaces, to illustrate the buffer level crossing the space threshold. Again, the local host DMA controller settings must correspond to those of the UART_IrDA.

Figure 74. Transmit FIFO DMA Request Generation (8 Spaces)

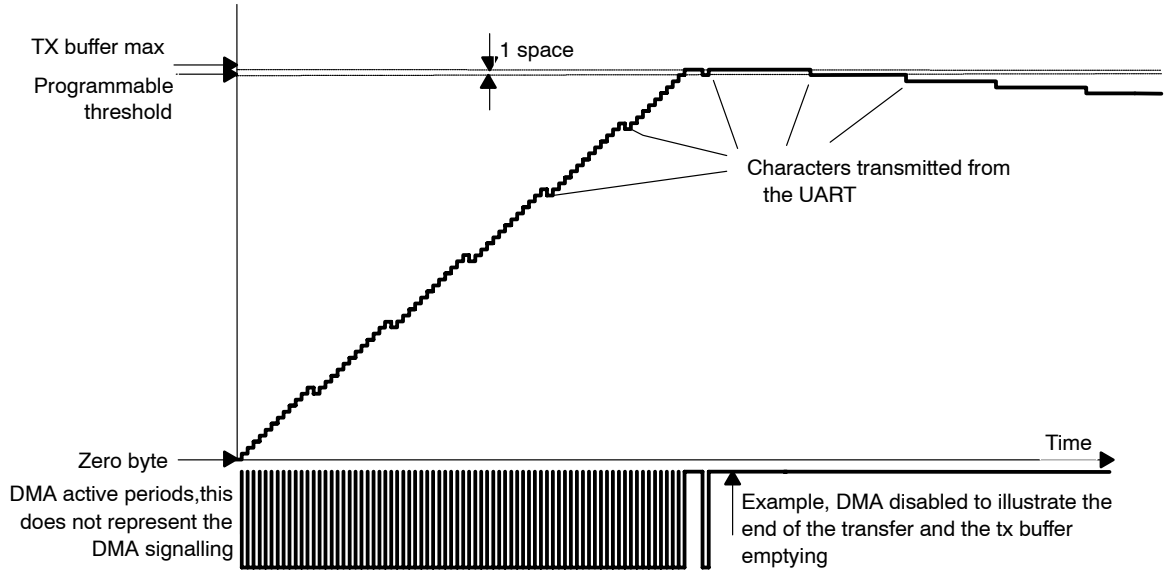


The final example in Figure 75 illustrates the setting of 1 space that uses the DMA for each transfer of 1 character to the transmit buffer. The buffer is filled at a faster rate than the BAUD rate transmits data to the TX pin. Eventually, the buffer is completely full and the DMA operation stops transferring data to the transmit buffer.

The buffer holds the maximum amount of data words on 2 occasions. Shortly after that the DMA is disabled to illustrate the slower transmission of the data words to the TX pin. Eventually, the buffer is emptied at the rate specified by the baud-rate settings of the DLL and DLH registers.

Again, the DMA settings must correspond to the local host DMA controller settings to ensure the correct operation of this logic.

Figure 75. Transmit FIFO DMA Request Generation (1 Space)



6.6.6 Sleep Mode

UART Modes

In UART modes, sleep mode is enabled by writing a 1 to IER[4] (when EFR[4] = 1).

Sleep mode is entered when

- ❏ The serial data input line, RX, is idle.
- ❏ The TX FIFO and TX shift register are empty.
- ❏ The RX FIFO is empty.
- ❏ There are no interrupts pending except THR interrupts.

Sleep mode is a good way to lower power consumption of the UART, but this state can be achieved only when the UART is set in modem mode. Therefore, even if the UART has no functional key role, it must be initialized in a functional mode to take advantage of sleep mode.

In sleep mode, the module clock and baud rate clock are stopped internally. Because most registers are clocked using these clocks, the 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, it occurs with any change in the state of the modem input pins. An interrupt is generated on a wake-up event by setting SCR[4] to 1.

Note:

Writing to the divisor latches, DLL and DLH, to set the baud clock, BCLK, must not be done during sleep mode. Therefore, it is advisable to disable sleep mode using IER[4] before writing to DLL or DLH.

IrDA Modes

In IrDA modes, sleep mode is enabled by writing a 1 to MDR1[3].

Sleep mode is entered when

- The serial data input line, RXIR, is idle.
- The TX FIFO and TX shift register are empty.
- The RX FIFO is empty.
- There are no interrupts pending except THR interrupts.

The module wakes up when any change is detected on the RXIR line, if data is written to the TX FIFO.

6.6.7 Idle Modes

Sleep and autoidle modes are embedded power-saving features. At the system level, power reduction techniques are applied by shutting down certain internal clock and power domains of the device.

The UART supports REQ_IDLE ACK handshaking protocol. This protocol is used at system level to shut down UART clocks 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 (Refer to SYSC[2] and WER.)

For a software programming guide, refer to the *OCP Design Guidelines for Idle Mode Control*.

6.6.8 Break and Time-Out Conditions

Time-Out Counter

An RX idle condition is detected when the receiver line, RX, 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 timeout interrupt, the counter only counts when there is data in the RX FIFO. 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 TX line is pulled low. A break condition is activated by setting LCR[6]. 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:

- ❏ Reset transmit FIFO (if enabled).
- ❏ Wait for transmit shift register to become empty (LSR[6] = 1).
- ❏ Take a guard time according to stop bit definition.
- ❏ Set LCR[6] to 1.

Break condition is asserted as long as LCR[6] is set to 1.

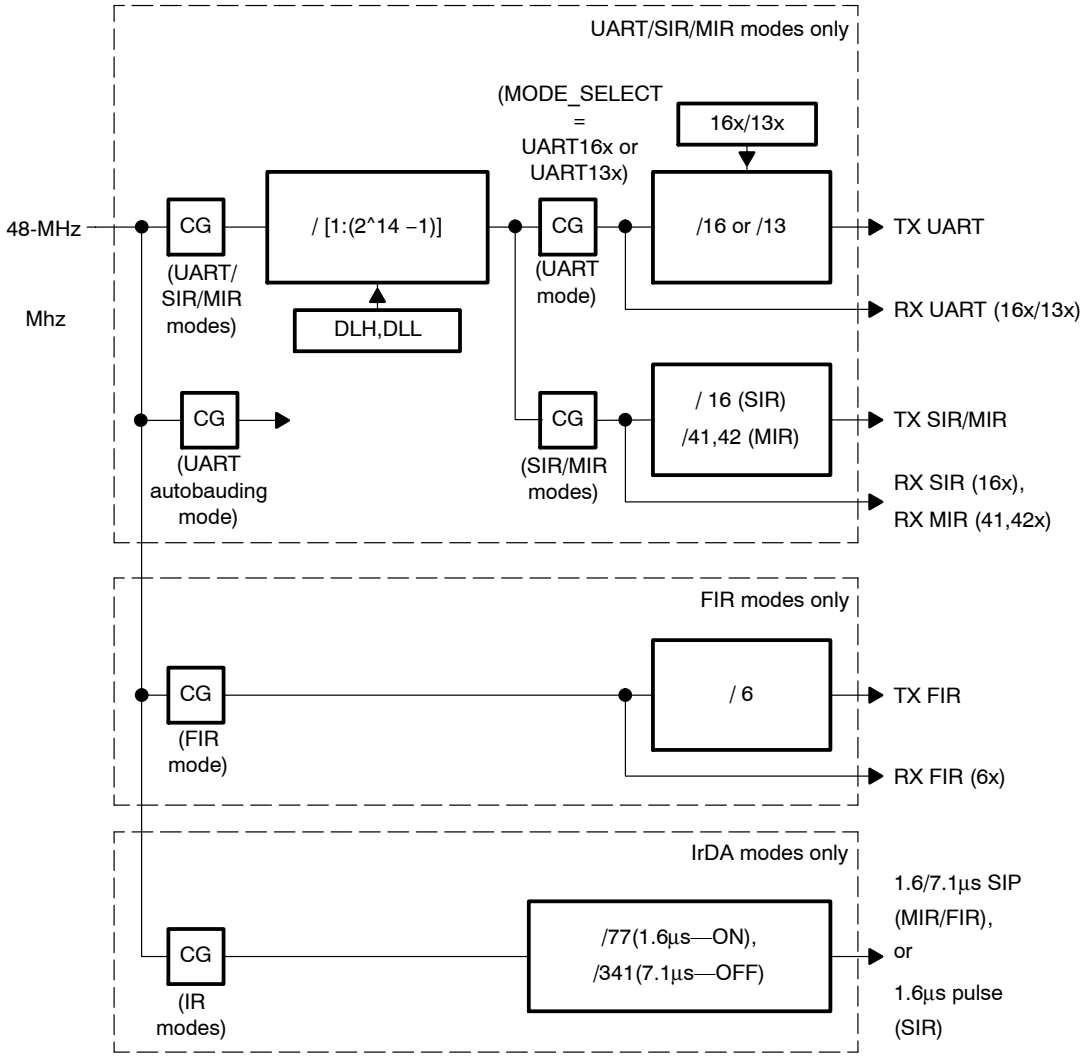
The above functionality (time-out counter and break condition) applies only to the UART modem operation and does not extend to the UART IrDA modes of operation.

6.6.9 Programmable Baud Rate Generator

The UART/IrDA module contains a programmable baud generator and a set of fixed dividers that take the 48-MHz clock input and divide it down to the expected baud rate.

The baud rate generator and associated controls are depicted in Figure 76.

Figure 76. Baud Rate Generator



Before Modifying Clock Parameters

It is recommended that MODE_SELECT = DISABLE (MDR1[2:0] = 111) be set before attempting to initialize or modify clock parameters controls (DLH, DLL). Nonobservance of this rule may result in an unpredictable behavior of the module.

Choosing the appropriate divisor value:

- ❏ UART 16x mode: Divisor value = Operating Freq/(16x baud rate)
- ❏ UART 13x mode: Divisor value = Operating Freq/(13x baud rate)
- ❏ SIR mode: Divisor value = Operating Freq/(16x baud rate)
- ❏ MIR mode: Divisor value = Operating Freq/(41x/42x baud rate)
- ❏ FIR mode: Divisor value = none.

Table 111. UART BAUD Rate Settings (48-MHz Clock)

Baud Rate (b/s)	Baud Multiple	DLH,DLL (Decimal)	DLH,DLL (Hex)	Actual Baud Rate (b/s)	Error (%)
0.3 K	16 x	10000	0x27, 0x10	0.3 K	0
0.6 K	16 x	5000	0x13, 0x88	0.6 K	0
1.2 K	16 x	2500	0x09, 0xC4	1.2 K	0
2.4 K	16 x	1250	0x04, 0xE2	2.4 K	0
4.8 K	16 x	625	0x02, 0x71	4.8 K	0
9.6 K	16 x	313	0x01, 0x39	9.6153 K	+0.16
14.4 K	16x	208	0x00, 0xD0	14.423 K	+0.16
19.2 K	16 x	156	0x00, 0x9C	19.231 K	+0.16
28.8 K	16 x	104	0x00, 0x68	28.846 K	+0.16
38.4 K	16 x	78	0x00, 0x4E	38.462 K	+0.16
57.6 K	16 x	52	0x00, 0x34	57.692 K	+0.16
115.2 K	16 x	26	0x00, 0x1A	115.38 K	+0.16
230.4 K	16 x	13	0x00, 0x0D	230.77 K	+0.16
460.8 K	13 x	8	0x00, 0x08	461.54 K	+0.16
921.6 K	13 x	4	0x00, 0x04	923.08 M	+0.16
1.8342 M	13 x	2	0x00, 0x02	1.1846 M	+0.16
3.6864 M	13 x	1	0x00, 0x01	3.6923 M	+0.16

Table 112. IrDA Baud Rate Settings (48-MHz Clock)

Baud Rate (b/s)	IR Mode	Baud Multi-ple	En-coding	DLH, DLL	Actual Baud Rate (* = Avg) (b/s)	Error (%)	Source Jitter (%)[1]	Pulse Duration
2.4 K	SIR	16x	3/16	1250	2.4 K	0	0	78.1 µs
9.6 K	SIR	16x	3/16	312	9.6153 K	+0.16	0	19.5 µs
19.2 K	SIR	16x	3/16	156	19.231 K	+0.16	0	9.75 µs
38.4 K	SIR	16x	3/16	78	38.462 K	+0.16	0	4.87 µs
57.6 K	SIR	16x	3/16	52	57.692 K	+0.16	0	3.25 µs
115.2 K	SIR	16x	3/16	26	115.38 K	+0.16	0	1.62 µs
0.576 M	MIR	41x/42 x	1/4	2	0.5756 M*	0	+1.63/ −0.80	416 ns
1.152 M	MIR	41x/42 x	1/4	1	1.1511 M*	0	+1.63/ −0.80	208 ns
4 M	FIR	6x	4 PPM	–	4 M	0	0	125 ns

6.6.10 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, $\overline{\text{CTS}}$ must be active before the module can transmit data.

Auto-RTS only activates the $\overline{\text{RTS}}$ output when there is enough room in the FIFO to receive data, and deactivates the $\overline{\text{RTS}}$ output when the RX FIFO is sufficiently full. The HALT and RESTORE trigger levels in the TCR determine the levels at which $\overline{\text{RTS}}$ is activated/deactivated.

If both auto-CTS and auto-RTS are enabled, data transmission does not occur unless the receiver FIFO has empty space. Thus, overrun errors are eliminated during hardware flow control. If they are not enabled, overrun errors occur when the transmit data rate exceeds the receive FIFO latency.

Auto-RTS

Auto-RTS data flow control originates in the receiver block (see Figure 61, Functional Block Diagram). The receiver FIFO trigger levels used in auto-RTS are stored in the TCR. $\overline{\text{RTS}}$ 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, $\overline{\text{RTS}}$ 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 $\overline{\text{RTS}}$ until it has begun sending the additional byte. $\overline{\text{RTS}}$ is automatically reasserted once the receiver FIFO reaches the RESUME trigger level programmed via TCR[7:4]. This reassertion requests the sending device to resume transmission.

Auto-CTS

The transmitter circuitry checks $\overline{\text{CTS}}$ before sending the next data byte. When $\overline{\text{CTS}}$ is active, the transmitter sends the next byte. To stop the transmitter from sending the following byte, $\overline{\text{CTS}}$ 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 $\overline{\text{CTS}}$ 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.

6.6.11 Software Flow Control

The enhanced feature register (EFR) and the modem control register (MCR) enable software flow control. Different combinations of software flow control are enabled by setting different combinations of EFR[3:0].

Two other enhanced features relate to software flow control:

- XON any function (MCR[5]): The operation will resume after receiving any character, after recognizing the XOFF character.

  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. A read of the IIR clears the XOFF interrupt. The special character is transferred to the RX FIFO.

Receive (RX)

When the 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 after completing transmission of the current character. XOFF detection also sets IIR[4] (if enabled via IER[5]) and causes UART_nIRQ to go low.

To resume transmission an XON1/2 character must be received (in certain cases XON1 and XON2 must be received sequentially). When the correct XON characters are received, IIR[4] is cleared and the XOFF interrupt disappears.

If a parity, framing, or break error occurs while receiving a software flow control character, the 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, in EFR[1:0] = 10, if the 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] = 11), 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 programmed trigger level TCR[3:0].

XON1: Two characters are transmitted when the RX FIFO reaches the programmed trigger level via TCR[7:4].

If an XOFF character has been sent, software flow control is disabled and the module transmits XON characters automatically to enable normal transmission to proceed. The transmission of XOFF/XON (s) follows the same protocol as transmission of an ordinary byte from the FIFO. This means that even if the word length is set to be 5, 6, or 7 characters, then the 5, 6, or 7 least-significant bits of XOFF1,2/XON1,2 are transmitted. Note that the transmission of 5, 6, or 7 bits of a character is seldom done, 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.

6.6.12 Autobauding Mode

In autobaud mode, UART extracts transfer characteristics (speed, length and parity) from an AT command. These characteristics are used to receive data following an *at* and to send data.

Here are valid AT commands:

AT DATA <CR>
AT DATA <CR>
A/
a

The 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 have to be recognized because of their special meaning. They 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* (AT) is received, it and all subsequent data are saved into FIFO, including final <CR> (0x0D). Then, the autobaud state machine waits for the next valid AT command. If an *a/* (AI) is received, it is saved into FIFO and the state machine waits for next valid AT command.

Upon the first successful detection of the baud rate, the UART activates an interrupt to signify that the AT(upper- or lowercase) sequence has been detected. The UASR register reflects the correct settings for the baud rate that has been detected. The interrupt activity continues in this fashion each time a subsequent character is received. Therefore, it is recommended that the software enable the RHR interrupt when using the autobaud mode.

The following settings are detected in autobaud mode with a module clock of 48 MHz:

- Speed:115.2K bauds, 57.6K bauds, 38.4K bauds, 28.8K bauds, 19.2K bauds, 14.4K bauds, 9.6K bauds, 4.8K bauds, 2.4K bauds, or 1.2K bauds
- Length: 7 or 8 bits
- Parity: Odd, even, or space

Note:

The combination of 7-bit character + space parity is not supported.

The method used to identify the speed is:

1) Detect the transition 1 → 0 on the received data. This happens as soon as a stop-to-start bit transition occurs. The transition is valid after a majority vote on 3 sampling periods.

2) Sample the start bit duration with 115 200 *16-Hz clock frequency as long as there is no rising edge. A transition 0 → 1 is considered as valid after a majority vote on 3 sampling periods.

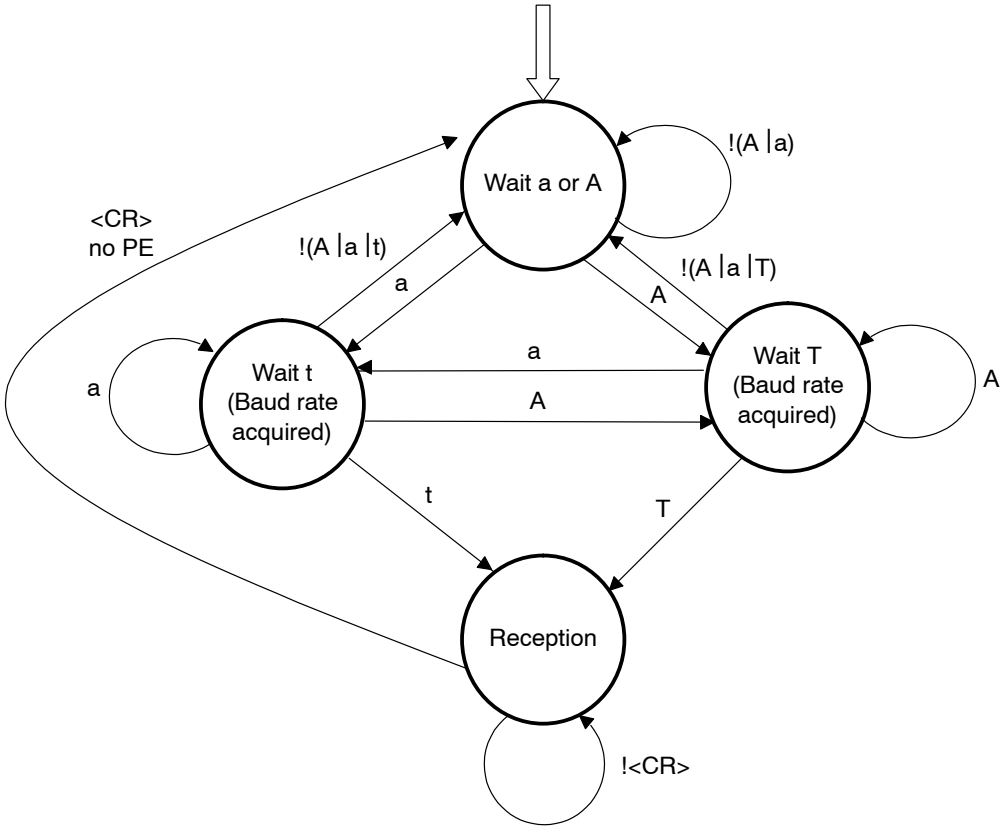
3) Compare the sampled value with a table. If the sampled value is outside a valid range, an error is reported (no speed identified) and the hardware goes back to the first state (1).

4) Else store the first data bit in the received register (for serial to parallel conversion) and go to frame format identification.

The next received bits are sampled according to the programmed baud rate. However, after 7 bits reception, the speed identification must be restarted because several *a* or *A* characters may have been received before a valid *t* or *T* character.

The autobauding mode is selected when MDR1[2:0] = 010. In the UART autobauding mode, DLL, DLH, and LCR[5:0] settings are not used. Instead, UASR is updated with the configuration detected by the autobauding logic.

Figure 77. Autobaud State Machine



6.6.13 Frame Closing

A transmission frame is properly terminated in one of two ways:

- Frame-length method: The frame-length method is selected when MDR1[7] = 0. The local host writes the frame-length value to the TXFLH and TXFLL registers. The device automatically attaches end flags to the frame once the number of bytes transmitted becomes equal to the frame-length value.
- Set-EOT bit method: The Set-EOT bit method is selected when MDR1[7] = 1. The local host writes 1 to ACREG[0] (EOT bit) just before it writes the last byte to the TX FIFO. When the local host 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.

6.6.14 Store and Controlled Transmission (SCT)

In SCT the local host 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 a 1 to ACREG[2] (deferred TX start) to start transmission. SCT is enabled when MDR1[5] = 1. This method of transmission differs from the normal mode, where transmission of data starts immediately after data is written to the TX FIFO. SCT is useful for sending short frames without TX underrun.

6.6.15 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 causes an internal flag to be set, which disables further transmission. Before the next frame can be transmitted the system (LH) must:

- ❏ Reset the TX FIFO.
- ❏ Read the RESUME register. This clears the internal flag.

This functionality is disabled with ACREG[4], compensated by the extension of the stop bit in transmission in case the TX FIFO is empty.

6.6.16 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 local host with IIR[3] and discards the remaining portion of the frame. Overrun also causes an internal flag to be set, which disables further reception. Before the next frame can be received the system (LH) must:

- ❏ Reset the RX FIFO.
- ❏ Read the RESUME register. This clears the internal flag.

6.6.17 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 it are written into the status FIFO.

The frame length and error status can be determined by reading SFREGL/H and SFLSR. Reading the SFLSR causes the read pointer to be incremented. The status FIFO is eight entries deep and therefore 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 screens bad frames using the error-status information and later requests the sender to resend only the bad frames.

This status FIFO is used effectively in DMA, as the LH need not be interrupted each time a frame is received but only when the programmed status FIFO trigger level is reached.

6.7 UART Configuration Example

This section outlines the programming stages for operating one UART module with FIFO, interrupt, and no DMA capabilities. This is a three-step procedure that ensures a quick start of these modules (obviously, it does not cover every UART module feature). The first stage covers the software reset of the module (interrupts, status, and controls). The second stage deals with FIFO configuration and enable. The last stage with deals with the baud rate data and stop configuration. The procedure below is programming language agnostic.

6.7.1 UART Software Reset

Goal:

To clear IER and MCR registers, remove UART breaks (LCR[6] = 0) and put module in reset (MDR1[2:0] = 0x07).

Procedure:

To write into both the IER and MCR register EFR[4] must first be set to 1.

To be able to access the EFR register, 0xBF must be first be written to LCR register. Hence,

1) LCR = 0xBF; First write to the LCR register.
2) EFR[4] = 1; When LCR = 0xBF, enable the enhanced feature register.
3) LCR[7] = 0; Here, access to IER and MCR is allowed.
4) IER = 0x00; Disable interrupt.
5) MCR = 0x00; Force control signals inactive.
6) LCR[6] = 0; Here, remove UART breaks.
7) MDR1 = 0x07; Here, UART is in reset or disabled.

Alternatively, the SYSC[1] can be set to one to instigate a hardware reset from the generic synchronous reset module. The reset progress can be monitored via the SYSS[0]. Once complete, the above sequence should ensure that the UART is in the equivalent disabled mode with reference to MDR1[2:0].

6.7.2 UART FIFO Configuration

Goal:

To set trigger level for halt/restore (TCR register), set trigger level for transmit/receive (TLR register), and configure FIFO (FCR register).

Procedure:

To write into both the TLR and TCR registers, EFR[4] must be set to 1 and MCR[6] to 1. To write into FCR, EFR[4] must be set to 1. Note that EFR[4] = 1 was already done in the previous section. Therefore, a simple write to MCR[6] is necessary.

MCR[6] = 1; Sets 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

6.7.3 Baud Rate Data and Stop Configuration

Goal:

To configure UART data, stop (LCR register) 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 DLH and DLL registers

- ❏ Set DLH and DLL;

  LCR[7] = 0; Removes access to DLH and DLL registers

- ❏ Set IER to desired value. Sets interrupts.

  MDR1[2:0] = 0; Enables UART without autobauding

The UART module is operational.

7 HDQ and 1-Wire Protocols

This module implements the hardware protocol of the master function of the HDQ™ and the 1-Wire™ protocol.

This module works off a command structure that is programmed into transmit command registers. The received data is in the receive data register. The firmware is responsible for performing correct sequencing in the command registers. The module only implements the hardware interface layer of the protocols.

The HDQ and the 1-Wire modes are selectable in software and must be chosen before any transmit and receive from the module is performed. The mode is assumed static during operation of the device. The 1-Wire and the HDQ protocols both use HDQ timing.

7.1 Functional Description

The module works with both HDQ and the 1-Wire protocols. The protocols use a single wire to communicate between the master and the slave. The protocols employ a return-to-1 mechanism, where after any command, the line is pulled up to a high. This requires an external pullup.

An open-drain configuration is used on the wire. The DX is connected to the GZ pin of an external 3-state output driver, and the input pin is connected to a zero level.

A control bit determines whether the HDQ or the 1-Wire protocol is to be used. Although this bit can be modified at any point, it is recommended that it be modified only as part of boot-up configuration. For the design, the bit is assumed static. By default, the configuration complies with the HDQ spec.

7.1.1 Receive and Transmit Operation

The receive and transmit operations are performed according to the timing specified in the HDQ protocol. This is done to keep the hardware interface section compatible between the two devices. In essence, the 1-Wire mode runs at slower speeds than the capabilities of the mode. The differences between the protocol at the hardware layer are described in the following subsections.

7.1.2 HDQ Mode (Default)

In HDQ mode, the firmware does not require the host to create an initialization pulse to the slave. However, the slave can be reset using an initialization pulse (also referred to as a break pulse). The pulse is created by setting the appropriate bit in the control and status register. The slave does not respond with a presence pulse, as it does in the 1-Wire protocol.

In a typical write to the slave, 2 bytes of data are sent to the slave. This is the command/address byte followed by the data that must be written. In a typical read, 1 command/address byte is sent to the slave, and the slave returns 1 byte of data.

The master implementation is a byte engine. The firmware is responsible for sending the ID, command/address, and data. The master engine provides only one data TX register.

HDQ is a return-to-1 protocol. This means that after a data byte (either command/address + write data for writes, or just command/address for reads) is sent to the slave, the host pulls the line high. This is accomplished in the device by setting the line to high (with an external pullup). The slave pulls the line low to initiate a transaction. This is the case when a read occurs, and the slave must send the read data back to the host.

If the host initiates a read and data is not received in a specified interval (the slave does not pull the line low within this time), a time-out status bit is set. This indicates that a read was not successfully completed. On successful completion, the time-out bit is cleared. The bit remains set or cleared until the next transaction by the host.

An interrupt condition indicates either a TX complete, RX complete, or time-out condition. The read of the interrupt status register clears all of the interrupt conditions. Only one interrupt signal is sent to the microcontroller, and only an overall mask bit exists for the enabling and disabling of the interrupt. The interrupt conditions cannot be individually masked.

The programmer must perform the following sequence for the reads and writes to the slave:

Write operation:

1) Write the command or data value to the TX write register.
2) Write 0 to the R/W bit of the control and status register to indicate a write.
3) Write 1 to the go bit of the control and status register to start the actual transmit. This step and the above step can be done at the same time.
   - The hardware sends the byte from the TX data register.
   - The time-out bit is always cleared in a write, because the hardware has no acknowledge mechanism from the slave.
   - The completion of the operation sets the TX complete flag in the interrupt status register. If interrupts are masked, no interrupt is generated. The interrupt status register is always cleared at the beginning of any read or write operation.
   - At the end of the write, the go bit is cleared.

4) Software must read the interrupt status register to clear the interrupt.

5) Repeat for each successive byte.

Read operation:

1) Write the command value to the TX write register.

2) Write 0 to R/W bit, 1 to the go bit, and wait for TX complete interrupt.

3) Write 1 to the R/W bit of the control and status register to indicate a read.

4) Write 1 to the go bit of the control and status register to start the actual read. This step and the above step can be done at the same time.

   - The hardware detects a low-going edge of the line (created by the slave) and receives 8 bits of data in the RX receive buffer register. The first bit that is received from the slave is the LSB and the last bit is the MSB of the byte. 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, a time-out bit is set in the control and status register.
   - The completion of the operation sets the RX complete flag in the interrupt status register. If interrupts are masked, no interrupt is generated. The interrupt status register is always cleared at the beginning of either a read or a write operation.
   - At the end of the read, the go bit is cleared. It is also cleared if a time-out is detected.

5) Software must read the interrupt status register to determine whether RX was complete or whether a time-out occurred.

6) Software does a read of the RX buffer register to retrieve the read data from slave.

7) Repeat for each successive byte.

In HDQ16 mode, the address/command is only written once to the slave. However, after the first byte is received, if an RX complete interrupt is received, the software must initiate the read of the second byte by writing the go bit of the control and status register. The first byte that was received is shadowed and provided to the software while the hardware is fetching the second byte of data.

7.2 1-Wire Mode (SDQ)

This section highlights the primary differences between the HDQ and the 1-Wire protocols.

In the 1-Wire mode, the firmware must send an initialization pulse to the multiple slaves that can be connected on the interface. If any slave is present, the slave responds with a presence pulse.

The initialization pulse is sent by setting the appropriate bit in the control and status register. A presence detect is indicated in the appropriate bit of the register. If no presence is received, a time-out bit is set in the status register. The initialization bit is cleared at the end of the initialization pulse. Also, the presence detect and the time-out bits are cleared at the end of the initialization pulse, if a presence detect is received. The time-out bit has no other significance in this mode; that is, unlike in HDQ mode, it is always cleared during a read operation.

1-Wire mode is a bit-by-bit protocol for a read. Unlike HDQ, which sends eight bits of data on a read, the slave must be clocked by the host in 1-Wire protocol for each bit. At the end of the command/address byte, the line is pulled high and the host creates a low-going edge to initiate a bit read from the slave. The host then pulls the line high, and the slave either pulls the line low to indicate a 0 or does not drive the line to indicate a 1. The host repeats the operation for the next bit that need to be read.

The first bit that is received is the LSB and the last bit is the MSB in the RX data register.

An interrupt condition indicates either a TX complete, RX complete, or time-out condition. The read of the interrupt status register clears all of the interrupt conditions. Only one interrupt signal is sent to the microcontroller, and only an overall mask bit exists for the enabling and disabling of the interrupt. The interrupt conditions cannot be individually masked.

The programmer must perform the following sequence for the reads and writes to the slave:

Write operation:

1) Write the ID, command, or data value to the TX write register.

2) Write 0 to the R/W bit of the control and status register to indicate a write.

3) Write 1 to the go bit of the control and status register to start the actual transmit. This step and the above step can be done at the same time.

- The hardware sends the one byte of the TX write data register.
- The time-out bit is always cleared in a write.
- The completion of the operation sets the TX complete flag in the interrupt status register. If interrupts are masked, no interrupt is generated. The interrupt status register is always cleared at the beginning of any read or write operation.
- At the end of the write, the go bit is cleared.

4) If interrupt is enabled, software must read the interrupt status register to clear the interrupt.

5) Repeat for each successive byte.

Read operation:

1) Write the ID value to the TX write register.

2) Write 0 to R/W bit and 1 to the go bit and wait for TX complete interrupt.

3) Write the command value to the TX write register.

4) Write 0 to R/W bit and 1 to the GO bit and wait for TX complete interrupt.

5) Write 1 to the R/W bit of the control and status register to indicate a read.

6) Write 1 to the go bit of the control and status register to start the actual read. This step and the above step can be done at the same time.
   - The hardware creates a low-going edge of the line (created by the slave), and clocks 8 bits of data into the RX receive buffer register. The first bit that is received from the slave is the LSB and the last bit is the MSB of the byte.
   - The time-out bit is always cleared in a read.
   - The completion of the operation sets the RX complete flag in the interrupt status register. If interrupts are masked, no interrupt is generated. The interrupt status register is always cleared at the beginning of any read or write operation.
   - At the end of the read, the go bit is cleared. It is also cleared if a time-out is detected.

7) If interrupt is enabled, software must read the interrupt status register to determine if RX was completed or whether there was a time-out.

8) Software does a read of the RX buffer register to retrieve the read data from the slave.

9) Repeat for each successive byte.

7.3 1-Wire Bit Mode Operation

A single-bit mode can be entered by writing to the appropriate bit in the control and status register. In this mode, only one bit of data is received each time from the slave. After the bit is received, an RX complete interrupt is generated. Bit 0 of the receive buffer is updated each time a bit is received.

The mode has no effect in HDQ mode, as HDQ does not support single-bit protocol.

7.3.1 Timing Diagrams

Figure 78 shows the timing diagram for the read, reset, and write. In the HDQ, the reset pulse contains only the initialization and not the presence pulse. The timing required for the various signals are specified in *Single-Wire Advanced Battery Monitor IC for Cellular and PDA Applications* (SLUS480).

The master works at the timing of the HDQ interface, which encompasses the HDQ and the 1-Wire timing. Therefore, in 1-Wire mode, the master runs slower than the full performance capability of the protocol.

Figure 78. Read Timing Diagram

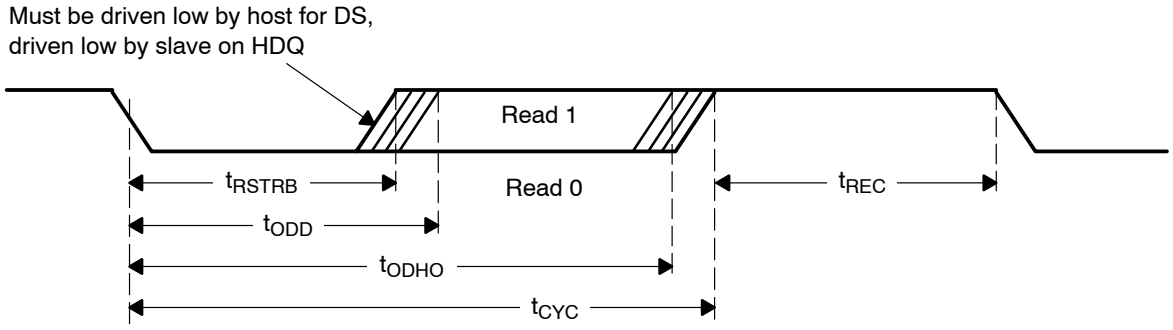


Figure 79. Reset Timing Diagram

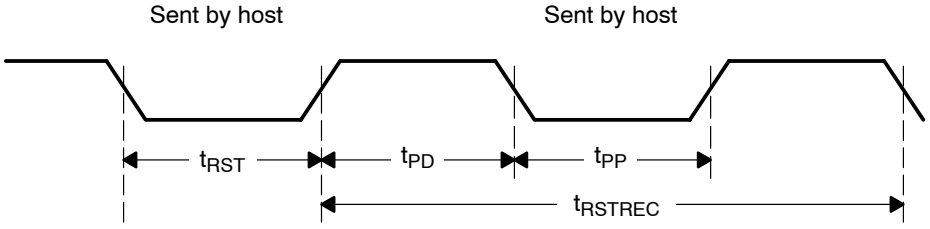
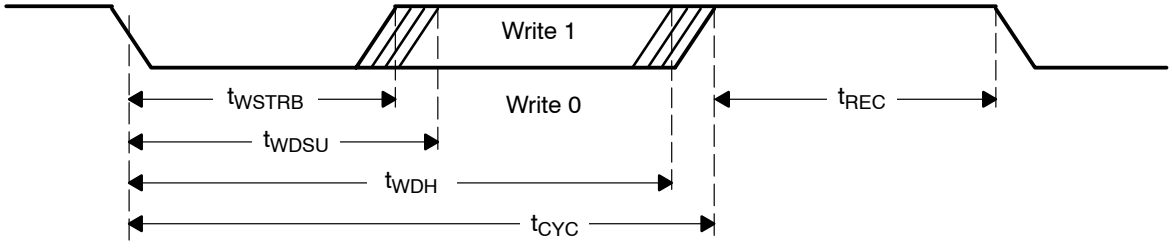
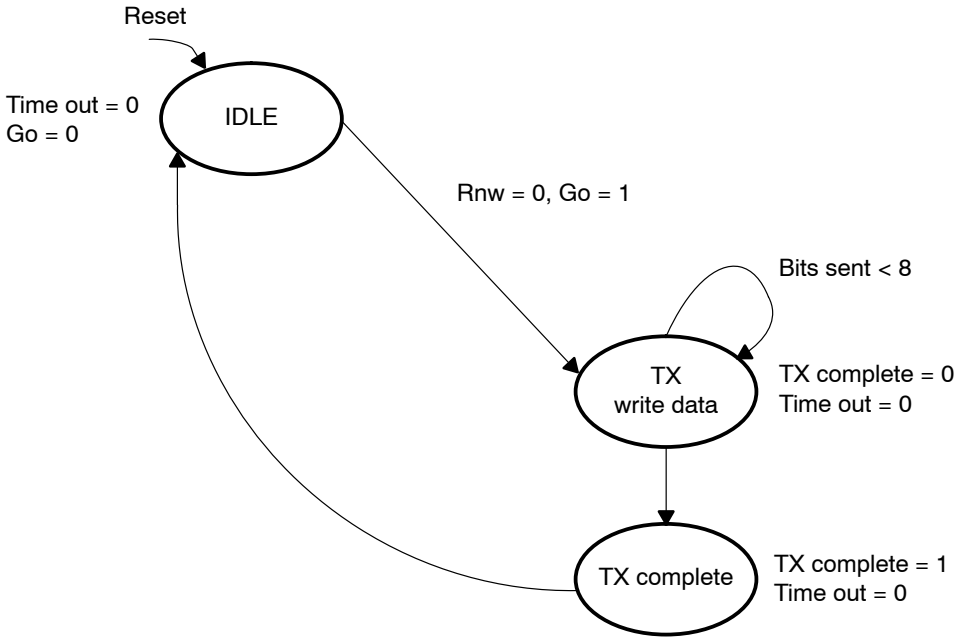


Figure 80. Write Timing Diagram



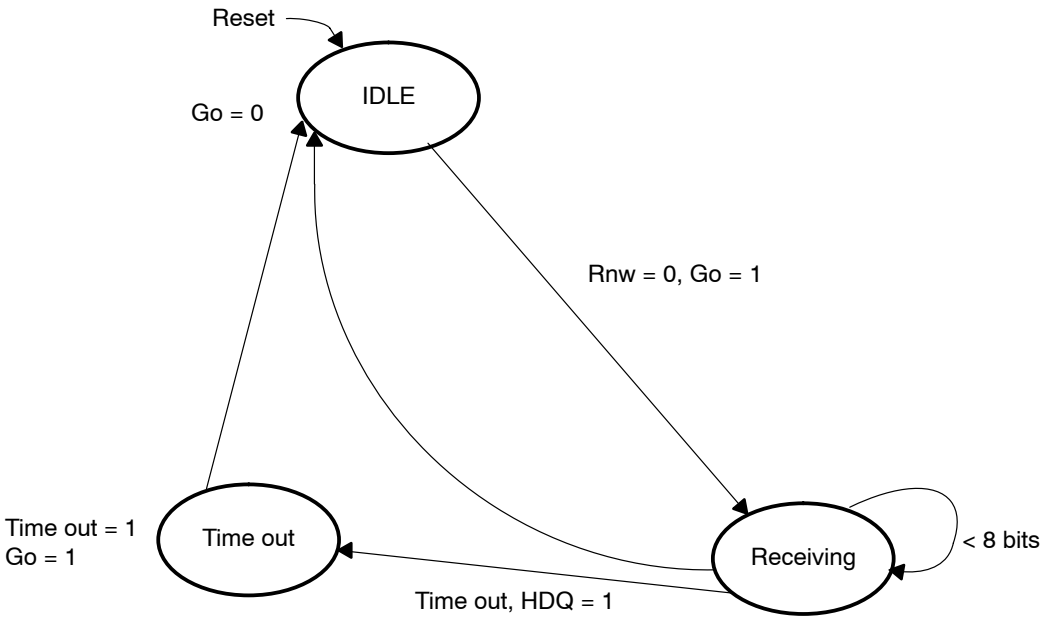
7.3.2 Write State Diagram

Figure 81. Write State Machine #1



7.3.3 Read State Diagram

Figure 82. Read State Machine #1



7.3.4 Status Flags

The status register contains status flags from the transmitter, the receiver, and the presence detect logic.

The presence-condition-detected status flag is contained in the status register. This is valid only in 1-Wire mode. It is cleared when the host sends an initialization pulse and then is set to 1 if a pulse is received. Otherwise, it stays cleared at 0.

7.3.5 Interrupts

The module provides the following interrupt status:

- Transmitter complete

  A write of one byte was completed. Successful or unsuccessful completion is not indicated, because there is no acknowledge from the slave in either HDQ or 1-Wire mode. Cleared at beginning of write command.

- Read complete

  Indicates successful completion of a byte read in both modes. Cleared at beginning of read command.

- Presence detect/time-out
  - In 1-Wire mode, it indicates that it is now valid to check the presence detect received bit. Cleared at beginning of initialization sequence.
  - In HDQ mode, it indicates that after a read command was issued by the host, the slave did not pull the line low within the specified time. In HDQ mode, the bit is cleared at beginning of the read command.

Only one interrupt is generated to the microcontroller, based on any of the above interrupt status conditions. A read to the interrupt status register clears all of the status bits that have been set.

The interrupt can be masked by setting the appropriate bit in the control and status register.

A read of the interrupt status register clears the interrupt. If there is a pending interrupt, the interrupt line stays low and no low-high-low transition is created. The interrupt therefore must be handled as a level interrupt (where a low-going edge is not needed) in an upstream interrupt handler (or processor).

7.4 Power-Down Mode

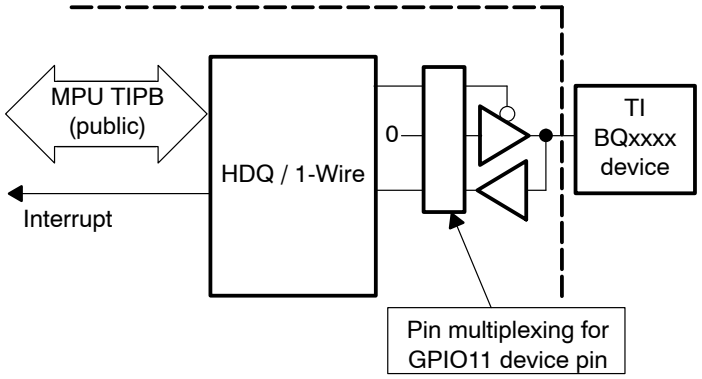
Writing to the appropriate bit in the control and status register shuts the clock to the state machine. The state machines are reset when the clock is disabled, and if any transaction is being performed, it is aborted into the reset state. The register values are not affected by disabling the clock. No register access must be performed to the module registers after the software puts the module in power-down mode by setting bit 5 of the control and status register to 0, other than a write to the power-down bit to take it out of power-down mode.

7.5 HDQ and 1-Wire Battery Monitoring Serial Interface

The HDQ and 1-Wire battery monitoring serial interface module implements the hardware protocol of the master function of the Benchmarq HDQ and the Dallas Semiconductor 1-Wire™ protocol. The module works off a command structure that is programmed into transmit command registers. The received data is in the received data register. The firmware is responsible for correct sequencing in the command registers. The module implements only the hardware interface layer of the protocols.

The HDQ and the 1-Wire mode are selectable in software, which must be done before any transmit and receive from the module is performed. The mode is assumed static during operation of the device.

Figure 83. HDQ and 1-Wire Overview



7.6 Software Interface

Mapping the registers to the TI peripheral bus (TIPB) address signals is shown in Table 113. The memory map identifies the 2K space associated with the peripheral.

The hardware provides no synchronization between the register clock domain and the state machine domain. This means that during a read, the hardware has the capability to modify the receive buffer. It is also possible that any access to the transmit write data register corrupts the data that is being sent if a TX is being performed.

However, these hazards can be avoided in software by observing the following limitations:

- A read is not performed from the interrupt status register or receive buffer register unless the processor has been interrupted by the peripheral.
- After the release of the go bit in the control and status register, no access to the TX write data buffer or the control and status registers is performed until the processor has been interrupted by the peripheral.
- Software is not allowed to poll the interrupt status register to determine whether an interrupt was generated.
- No register access can be done to the module registers after the software puts the module in power-down mode (by setting bit 5 of the control and status register to 0), except to reenable the clock.

Table 113. HDQ and 1-Wire Registers

Base Address = 0xFFFB C000			
Register	**Description**	**Type**	**Address**
HDQ1W_TX	TX write data	R/W	FFFB:C000
HDQ1W_RX	RX receive buffer	R	FFFB:C004
HDQ1W_CTRL	Control and status	R/W	FFFB:C008
HDQ1W_INTS	Interrupt status, read to clear	R	FFFB:C00C

Table 114. HDQ/1−Wire TX Write Data Register (HDQ1W_TX)

Base Address = 0xFFFB C000, Offset Address = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:24	Reserved	Reserved − read aliased to bit 7:0, writes ignored	R/W	0x00
23:16	Reserved	Reserved − read aliased to bit 7:0, writes ignored	R/W	0x00
15:8	Reserved	Reserved − read aliased to bit 7:0, writes ignored	R/W	0x00
7:0	WD	Write data (used in both HDQ and 1−Wire modes)	R/W	0x00

Table 115. HDQ/1−Wire RX Receive Buffer Register (HDQ1W_RX)

Base Address = 0xFFFB C000, Offset Address = 0x04				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:24	Reserved	Reserved − read aliased to bit 7:0, writes ignored	R	U
23:16	Reserved	Reserved − read aliased to bit 7:0, writes ignored	R	U
15:8	Reserved	Reserved − read aliased to bit 7:0, writes ignored	R	U
7:0	RD	Next received character	R	U

Table 116. HDQ/1−Wire Control Register (HDQ1W_CTRL)

Base Address = 0xFFFB C000, Offset Address = 0x08				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:24	Reserved	Reserved − read aliased to bit 7:0, writes ignored	R/W	0x00
23:16	Reserved	Reserved − read aliased to bit 7:0, writes ignored	R/W	0x00
15:8	Reserved	Reserved − read aliased to bit 7:0, writes ignored	R/W	0x00
7	SBM	Single-bit mode for 1-Wire	R/W	0
6	IM	Interrupt mask (1: Enable, 0: Disable interrupts)	R/W	0
5	PDM	Power-down mode (1: Enable clocks, 0: Disable clocks)	R/W	0
4	GB	Go bit Write 1 to send the appropriate commands. Bit returns to 0 after the command is complete.	R/W	0
3	PD	Presence detect received, 1-Wire mode only. 0: Not detected. 1: Detected.	R	0
2	IP	Write 1 to send Initialization pulse. Bit returns to 0 after pulse is sent.	R/W	0
1	RWB	R/W bit (determines if next command is read or write) 0: Write. 1: Read.	R/W	0
0	MODE	Set mode 0: HDQ. 1: 1-Wire.	R/W	0

Table 117. HDQ/1−Wire Interrupt Status Register (HDQ1W_INTS)

Base Address = 0xFFFB C000, Offset Address = 0x0C				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:24	Reserved	Reserved − read aliased to bit 7:0, writes ignored	R	0x00
23:16	Reserved	Reserved − read aliased to bit 7:0, writes ignored	R	0x00
15:8	Reserved	Reserved − read aliased to bit 7:0, writes ignored	R	0x00
7:3	Reserved	Reserved − reads 0, writes ignored	R	0x0
2	TC	TX completed	R/C	0

Table 117. HDQ/1−Wire Interrupt Status Register (HDQ1W_INTS) (Continued)

Base Address = 0xFFFB C000, Offset Address = 0x0C				
Bit	**Name**	**Function**	**R/W**	**Reset**
1	RC	Read complete	R/C	0
0	DTO	Presence detect/time-out: In 1-Wire mode this is due to presence detect, and in HDQ mode this is due to time-out on read	R/C	0

Note: R/C − read clears bit

8 Frame Adjustment Counter

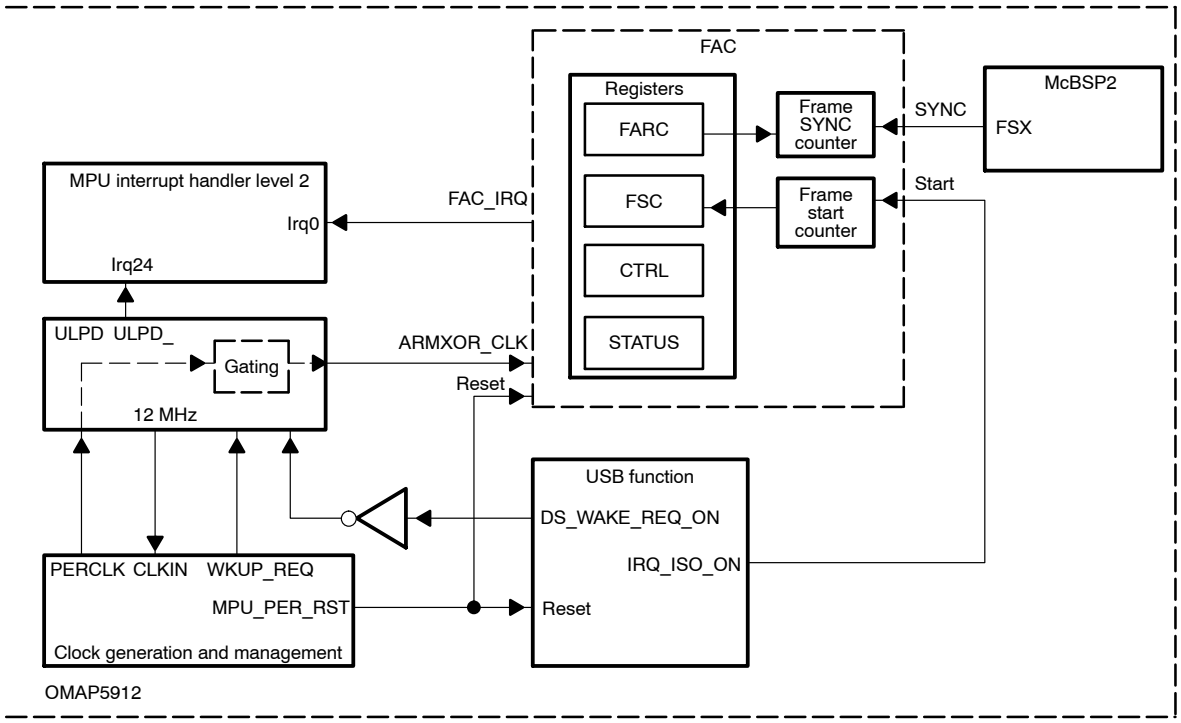
The frame adjustment counter counts the number of rising edges of one signal (start of frame interrupt of the USB function) during a programmable number of rising edges of a second signal (transit frame synchronization of McBSP2). System-level software uses this count value to adjust the duration of the two time domains with respect to each other to reduce overflow and underflow. If the data being transferred is audio data, this module can be part of a solution that reduces pops and clicks.

8.1 Features

The frame adjustment counter (FAC) is a module that consists of a frame-synchronization capture pin and a frame-start capture pin. System-level software uses the respective count values to adjust the duration of the two time domains with respect to each other to reduce the overflow and underflow.

A frame-adjustment reference count (FARC) register is programmed with the number of frame-synchronization pulses over which the frame-start pulses are to be counted. A 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. In halt mode, a new count is initiated upon software servicing the interrupt by reading the FSC register. The RUN bit in the control and configuration register can enable and disable the counters. If the RUN bit is set to 0, the counters are reset immediately, even though the count is not finished. The software can use this bit as a software reset. Additionally, a status register (STATUS) containing a FSC_FULL bit indicates to the system software whether FSC has been read subsequent to the last FSC update.

Figure 84. FAC Top-Level Diagram



The main FAC features are:

- ❏ Frame-synchronization capture pin (SYNC)
- ❏ Frame-start capture pin (START)
- ❏ Programmable frame-adjustment reference count register (FARC)
- ❏ Read-only frame-start count register (FSC)
- ❏ Interrupt generation logic
- ❏ Configuration and control register (CTRL)
- ❏ Status register (STATUS)

8.2 Synchronization and Counter Control

Because the frame-start and frame-synchronization signals are from different time domains, the FAC module synchronizes them to the system clock domain and uses the synchronized signals as the count enables. The actual counters for frame synchronization and frame start are clocked by the system clock.

8.2.1 Synchronization

The synchronization mechanism is based on the assumption that the system clock is running at least eight times faster than frame synchronization and frame start. Figure 85 and Figure 86 show the synchronization logic and the counter hookup.

Figure 85. FAC Module Counters and Clock Synchronization

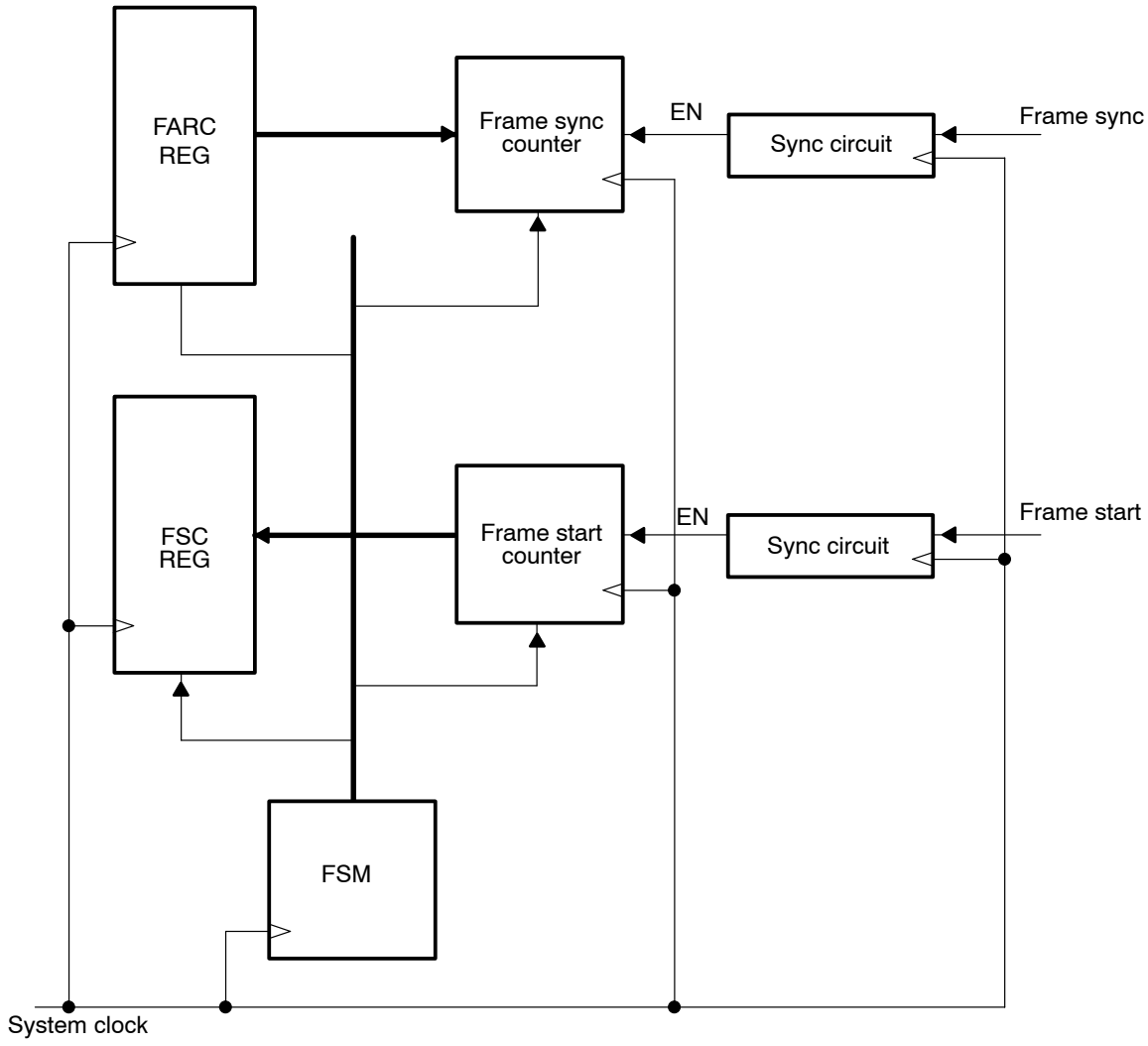


Figure 86. Synchronization Circuit for Frame Synchronization and Frame Start Signals

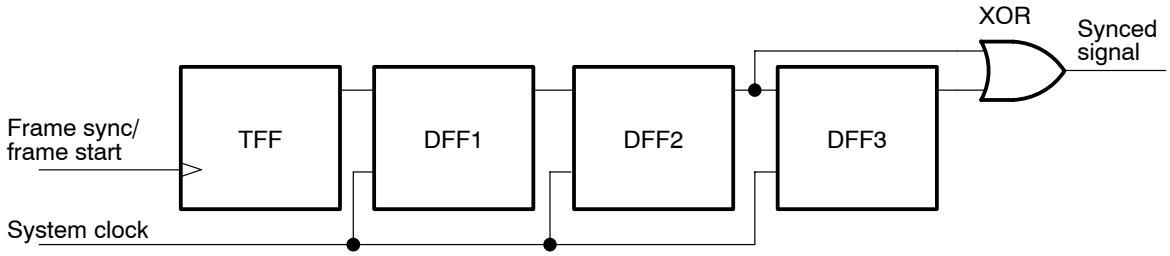
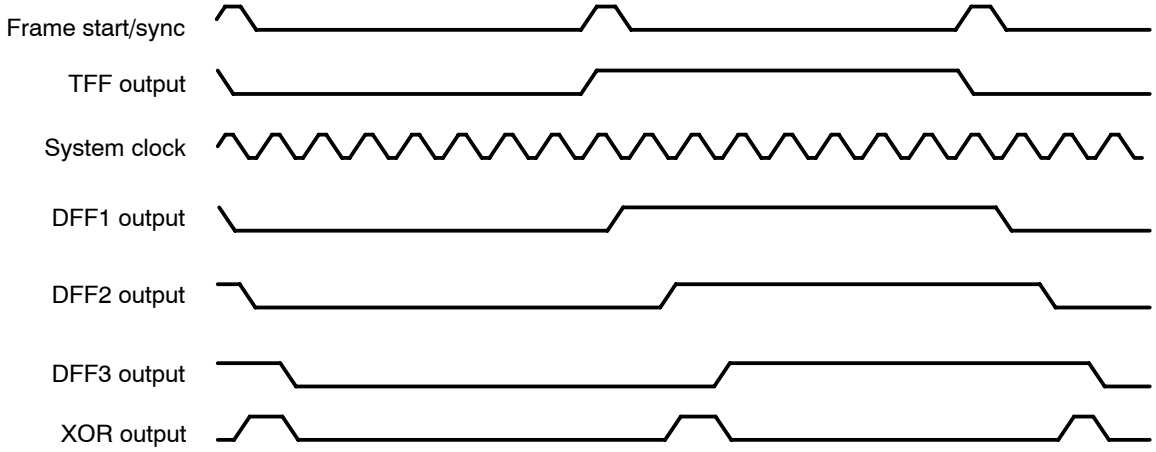


Figure 87 shows the actual waveforms at the output of each flip-flop and the XOR output.

Figure 87. Synchronization Circuit Waveforms



8.3 FAC Interrupt

The FAC generates one interrupt, FAC_IRQ (in halt mode when the FARC value is met), connected to the MPU level 2 interrupt handler, line 0 (level-sensitive).

8.4 FAC Clocks and Reset

The FAC works with a clock (PCLK) provided by the ULPD from a request generated by the USB function (DS_WAKE_REQ_ON).

The DS_WAKE_REQ_ON request does not wake up the system itself.

The ULPD module uses this request to generate an interrupt (ULPD_nIrq) to the MPU, which wakes up the system via its wake-up request (WKUP_REQ).

Once the system is awakened (12 MHz provided to the MPU), the MPU programmable peripheral clock (PERCLK) is used as the source clock for the FAC clock.

The MPU TIPB reset (MPU_PER_RST) resets the FAC.

8.5 Software Interface

Table 118 lists the FAC registers. Table 119 through Table 122 describe the register bits.

Table 118. FAC Registers

Register	Description	Type	Address
FARC	Frame adjustment reference count	R/W	FFFB:A800
FSC	Frame start count	R	FFFB:A804
CTRL	Control and configuration	R/W	FFFB:A808
STATUS	Status	R	FFFB:A80C
SYNC CNT	Frame synchronization counter	R	FFFB:A810
START CNT	Frame start counter	R	FFFB:A814

The FAC module is a word16 module with 32-bit aligned addresses.

The frame-adjustment counter register (FARC) is programmed with the number of frame synchronization counts over which the frame start pulses are counted. This is a 16-bit programmable fixed reference in the range of 0-65536. A value of 0 disables the count operation.

Table 119. Frame Adjustment Reference Count Register (FARC)

Base Address = 0xFFFB A800, Offset Address = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	FARC	16-bit value in the range 0-65536: 0—disable counting	R/W	0x0000

The frame-start count (FSC) register is a 16-bit read-only register that contains the number of frame-start rising edges that occur during the programmable FARC period. The frame-start counting can be in one of two modes. When the CNT bit in the control and configuration (CTRL) register is set to 1, the counting is in continuous mode and this register is updated periodically (every time the frame adjustment reference count is met) with the new count value. If CNT is 0, the counting is in halt mode. The frame-start count register is updated when the frame adjustment reference count is met, and the counting halts until the software reads the FSC register.

A level-sensitive interrupt can be generated to indicate that the frame-start counting is finished, and the FSC register is loaded with a new count value. The interrupt is controlled by the INT_ENABLE bit in the control and configuration (CTRL) register. If this bit is set to 1, an interrupt is generated when the FSC register is updated. Because the interrupt is level-sensitive, the interrupt signal is kept low until the software reads the FSC register, or the RUN bit in the control register is set to 0. When the FSC is read or RUN bit in control register is set to 0, the interrupt signal is reset to 1. When the IN_ENABLE bit is set to 0, no interrupt is generated. The interrupt can be enabled or disabled for both continuous mode and halt mode.

Table 120. Frame Start Count Register (FSC)

Base Address = 0xFFFB A800, Offset Address = 0x04				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	FS	16-bit value	R	0x0000

The control and configuration (CTRL) register is a R/W register used to configure the module. The RUN bit enables the frame-start counter. If this bit is set to 0, the frame-start counting is disabled immediately. The software can use this bit as a software reset for the FAC module by setting the RUN bit to 0. When the RUN bit is set to 0, the frame-start counter, the frame-synchronization counter, and the FSC register are reset to 0. The software reset also clears the status register FSC_FULL bit to 0. If an interrupt has been generated and the FAC module is waiting for an FSC register read, a software reset puts the counter control back to idle state. This means that after the software has been reset the counter starts counting again, regardless of whether the FSC register has been read or not.

Table 121. FAC Control and Configuration Register (CTRL)

Base Address = 0xFFFB A800, Offset Address = 0x08				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:3	Reserved	Reserved	R	0x0000
2	INT_ENABLE	The INT_ENABLE bit is independent of the CNT bit. The interrupt can be enabled or disabled in either continuous mode or halt mode. 0: No interrupt is generated. 1: An interrupt is generated when FSC is updated.	R/W	0
1	RUN	Enables operation of the counter. 0: The frame start counter, the frame-synchronization counter, and the FSC are reset to 0. Any pending interrupt also is cleared when RUN is set to 0. 1: Enables the frame start counter.	R/W	0
0	CNT	0: Halt mode– updates FSC value when the frame-adjustment reference count is met and halts operation until FSC is read. 1: Continuous mode– periodically updates FSC value each time the frame-adjustment reference count is met.	R/W	0

Table 122. FAC Status Register (STATUS)

Base Address = 0xFFFB A800, Offset Address = 0x0C				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:1	Reserved	Reserved	R	0x0000
0	FSC_FULL	This bit is set to a 1 when FSC is updated. This bit is set back to a 0 when the FSC has been read or RUN bit in control is 0.	R/C	0

The status register (STATUS) contains an interrupt status bit.

Table 123. SYNC Counter Register (SYNC_CNT)

Base Address = 0xFFFB A800, Offset Address = 0x10				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	SC	Sync count value	R	0x0000

Table 124. Start Counter Register (START_CNT)

Base Address = 0xFFFB A800, Offset Address = 0x14				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	SC	Start count value	R	0x0000

This page is intentionally left blank.

Index

M

N

P

R

S

T

W

OMAP5912 Multimedia Processor Universal Serial Bus (USB) Reference Guide

Literature Number: SPRU761A
March 2004

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.

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
		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 document describes the universal serial bus (USB) host on the OMAP5912 multimedia processor.

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.

Related Documentation From Texas Instruments

The following documents describe the OMAP5910 device and related peripherals. Copies of these documents are available on the Internet at www.ti.com. *Tip:* Enter the literature number in the search box provided at www.ti.com.

OMAP5912 Multimedia Processor Device Overview and Architecture Reference Guide (literature number SPRU748) introduces the setup, components, and features of the OMAP5912 multimedia processor and provides a high-level view of the device architecture.

OMAP5912 Multimedia Processor OMAP 3.2 Subsystem Reference Guide (literature number SPRU749) introduces and briefly defines the main features of the OMAP3.2 subsystem of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor DSP Sybsystem Reference Guide (literature number SPRU750) describes the OMAP5912 multimedia processor DSP subsystem. The digital signal processor (DSP) subsystem is built around a core processor and peripherals that interface with: 1) The ARM926EJS via the microprocessor unit interface (MPUI); 2) Various standard memories via the external memory interface (EMIF); 3) Various system peripherals via the TI peripheral bus (TIPB) bridge.

OMAP5912 Multimedia Processor Clocks Reference Guide (literature number SPRU751) describes the clocking mechanisms of the OMAP5912 multimedia processor. In OMAP5912, various clocks are created from special components such as the digital phase locked loop (DPLL) and the analog phase-locked loop (APLL).

OMAP5912 Multimedia Processor Initialization Reference Guide (literature number SPRU752) describes the reset architecture, the configuration, the initialization, and the boot ROM of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Power Management Reference Guide (literature number SPRU753) describes power management in the OMAP5912 multimedia processor. The ultralow-power device (ULPD) generates and manages clocks and reset signals to OMAP3.2 and to some peripherals. It controls chip-level power-down modes and handles chip-level wake-up events. In deep sleep mode, this module is still active to monitor wake-up events.This book describes the ULPD module and outline architecture.

OMAP5912 Multimedia Processor Security Features Reference Guide (literature number SPRU754) describes the security features of teh OMAP5912 multimedia processor. The OMAP5912 security scheme relies on the OMAP3.2 secure mode. The distributed security on the OMAP3.2 platform is a Texas Instruments solution to address m-commerce and security issues within a mobile phone environment. The OMAP3.2 secure mode was developed to bring hardware robustness to the overall OMAP5912 security scheme.

OMAP5912 Multimedia Processor Direct Memory Access (DMA) Support Reference Guide (literature number SPRU755) describes the direct memory access support of the OMAP5912 multimedia processor. The OMAP5912 processor has three DMAs:

- The system DMA is embedded in OMAP3.2. It handles DMA transfers associated with MPU and shared peripherals.
- The DSP DMA is embedded in OMAP3.2. It handles DMA transfers associated with DSP peripherals.
- The generic distributed DMA (GDD) is an OMAP5912 resource attached to the SSI peripheral. It handles only DMA transfers associated with the SSI peripheral.

OMAP5912 Multimedia Processor Memory Interfaces Reference Guide (literature number SPRU756) describes the memory interfaces of the OMAP5912 multimedia processor.

- SDRAM (external memory interface fast, or EMIFF)
- Asynchronous and synchronous burst memory (external memory interface slow, or EMIFS)
- NAND flash (hardware controller or software controller)
- CompactFlash on EMIFS interface
- Internal static RAM

OMAP5912 Multimedia Processor Interrupts Reference Guide (literature number SPRU757) describes the interrupts of the OMAP5912 multimedia processor. Three level 2 interrupt controllers are used in OMAP5912:

- One MPU level 2 interrupt handler (also referred to as MPU interrupt level 2) is implemented outside of OMAP3.2 and can handle 128 interrupts.
- One DSP level 2 interrupt handler (also referred to as DSP interrupt level 2.1) is instantiated outside of OMAP3.2 and can handle 64 interrupts.
- One OMAP3.2 DSP level 2 interrupt handler (referenced as DSP interrupt level 2.0) can handle 16 interrupts.

OMAP5912 Multimedia Processor Peripheral Interconnects Reference Guide (literature number SPRU758) describes various periperal interconnects of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Timers Reference Guide (literature number SPRU759) describes various timers of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Serial Interfaces Reference Guide (literature number SPRU760) describes the serial interfaces of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Universal Serial Bus (USB) Reference Guide (literature number SPRU761) describes the universal serial bus (USB) host on the OMAP5912 multimedia processor. The OMAP5912 processor provides several varieties of USB functionality. Flexible multiplexing of signals from the OMAP5912 USB host controller, the OMAP5912 USB function controller, and other OMAP5912 peripherals allow a wide variety of system-level USB capabilities. Many of the OMAP5912 pins can be used for USB-related signals or for signals from other OMAP5912 peripherals. The OMAP5912 top-level pin multiplexing

controls each pin individually to select one of several possible internal pin signal interconnections. When these shared pins are programmed for use as USB signals, the OMAP5912 USB signal multiplexing selects how the signals associated with the three OMAP5912 USB host ports and the OMAP5912 USB function controller can be brought out to OMAP5912 pins.

OMAP5912 Multimedia Processor Multi-channel Buffered Serial Ports (McBSPs) Reference Guide (literature number SPRU762) describes the three multi-channel buffered serial ports (McBSPs) available on the OMAP5912 device. The OMAP5912 device provides multiple high-speed multichannel buffered serial ports (McBSPs) that allow direct interface to codecs and other devices in a system.

OMAP5912 Multimedia Processor Camera Interface Reference Guide (literature number SPRU763) describes two camera inerfaces implemented in the OMAP5912 multimedia processor: compact serial camera port and camera parallel interface.

OMAP5912 Multimedia Processor Display Interface Reference Guide (literature number SPRU764) describes the display interface of the OMAP5912 multimedia processor.

- LCD module
- LCD data conversion module
- LED pulse generator
- Display interface

OMAP5912 Multimedia Processor Multimedia Card (MMC/SD/SDIO) (literature number SPRU765) describes the multimedia card (MMC) interface of the OMAP5912 multimedia processor. The multimedia card/secure data/secure digital IO (MMC/SD/SDIO) host controller provides an interface between a local host, such as a microprocessor unit (MPU) or digital signal processor (DSP), and either an MMC or SD memory card, plus up to four serial flash cards. The host controller handles MMC/SD/SDIO or serial port interface (SPI) transactions with minimal local host intervention.

OMAP5912 Multimedia Processor Keyboard Interface Reference Guide (literature number SPRU766) describes the keyboard interface of the OMAP5912 multimedia processor. The MPUIO module enables direct I/O communication between the MPU (through the public TIPB) and external devices. Two types of I/O can be used: specific I/Os dedicated for 8 x 8 keyboard connection, and general-purpose I/Os.

OMAP5912 Multimedia Processor General-Purpose Interface Reference Guide (literature number SPRU767) describes the general-purpose in-

terface of the OMAP5912 multimedia processor. There are four GPIO modules in the OMAP5912. Each GPIO peripheral controls 16 dedicated pins configurable either as input or output for general purposes. Each pin has an independent control direction set by a programmable register. The two−edge control registers configure events (rising edge, falling edge, or both edges) on an input pin to trigger interrupts or wake−up requests (depending on the system mode). In addition, an interrupt mask register masks out specified pins. Finally, the GPIO peripherals provide the set and clear capabilities on the data output registers and the interrupt mask registers. After detection, all event sources are merged and a single synchronous interrupt (per module) is generated in active mode, whereas a unique wake−up line is issued in idle mode. Eight data output lines of the GPIO3 are ORed together to generate a global output line at the OMAP5912 boundary. This global output line can be used in conjunction with the SSI to provide a CMT−APE interface to the OMAP5912.

OMAP5912 Multimedia Processor VLYNQ Serial Communications Interface Reference Guide (literature number SPRU768) describes the VLYNQ of the OMAP5912 multimedia processor.

VLYNQ is a serial communications interface that enables the extension of an internal bus segment to one or more external physical devices. The external devices are mapped into local, physical address space and appear as if they are on the internal bus of the OMAP 5912. The external devices must also have a VLYNQ interface. The VLYNQ module serializes bus transactions in one device, transfers the serialized data between devices via a VLYNQ port, and de-serializes the transaction in the external device.

OMAP5912 includes one VLYNQ module connected on OCPT2 target port and OCPI initiator port. These connections are configured via a static switch, which selects either SSI or VLYNQ module. This switch, forbids the simultaneous use of GDD/SSI and VLYNQ. The switch is controlled by the VLYNQ_EN bit in the OMAP5912 configuration control register (CONF_5912_CTRL).

OMAP5912 Multimedia Processor Pinout Reference Guide (literature number SPRU769) provides the pinout of the OMAP5912 multimedia processor. After power-up reset, the user can change the configuration of the default interfaces. If another interface is available on top of the default, it is possible to enable a new interface for each ball by setting the corresponding 3-bit field of the associated FUNC_MUX_CTRL register. It is also possible to configure on-chip pullup/pulldown. This document

also describes the various power domains so that the user can apply the different interfaces seamlessly with external components.

OMAP5912 Multimedia Processor Window Tracer (WT) Reference Guide (literature number SPRU770) describes the window tracer module used to capture the memory transactions from four interfaces: EMIFF, EMIFS, OCP-T1, and OCP-T2. This module is located in the OMAP3.2 traffic controller (TC).

OMAP5912 Multimedia Processor Real-Time Clock Reference Guide (literature number SPRUxxx) describes the real-time clock of the OMAP5912 multimedia processor. The real-time clock (RTC) block is an embedded real-time clock module directly accessible from the TIPB bus interface.

Trademarks

Contents

Figures

Tables

Universal Serial Bus (USB)

This document describes the universal serial bus (USB) host on the OMAP5912 multimedia processor.

1 Overview

The OMAP5912 processor provides several varieties of USB functionality, including:

- USB host: OMAP5912 provides a three-port *USB Specification Revision 1.1*-compliant host controller, which is based on the *OHCI Specification for USB Release 1.0a*.
- USB device: OMAP5912 provides a full-speed USB device.
- USB On-The-Go (OTG): OMAP5912 acts as an OTG dual-role device; the USB device functionality and one port of the USB host controller act in concert to provide an OTG port.

Flexible multiplexing of signals from the OMAP5912 USB host controller, the OMAP5912 USB function controller, and other OMAP5912 peripherals allow a wide variety of system-level USB capabilities. Many of the OMAP5912 pins can be used for USB-related signals or for signals from other OMAP5912 peripherals. The OMAP5912 top-level pin multiplexing controls each pin individually to select one of several possible internal pin signal interconnections. When these shared pins are programmed for use as USB signals, the OMAP5912 USB signal multiplexing selects how the signals associated with the three OMAP5912 USB host ports and the OMAP5912 USB function controller can be brought out to OMAP5912 pins.

2 USB Host Controller

The OMAP5912 USB host controller (HC) is a three-port controller that communicates with USB devices at the USB low-speed (1.5M bit-per-second maximum) and full-speed (12M bit-per-second maximum) data rates. It is compatible with the *Universal Serial Bus Specification Revision 2.0* and the *Open HCI—Open Host Controller Interface Specification for USB,* Release 1.0a*,* available through the Compaq Computer Corporation web site, and hereafter called the *OHCI Specification for USB.* It is assumed that users of

the OMAP5912 USB host controller are already familiar with the *USB Specification* and *OHCI Specification for USB*.

The OMAP5912 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 OMAP5912 controls the OTG link, and the USB function controller acts as the downstream device when OMAP5912 acts as an OTG downstream device. The OMAP5912 OTG controller is described in Section 15.4, *USB OTG Controller*.

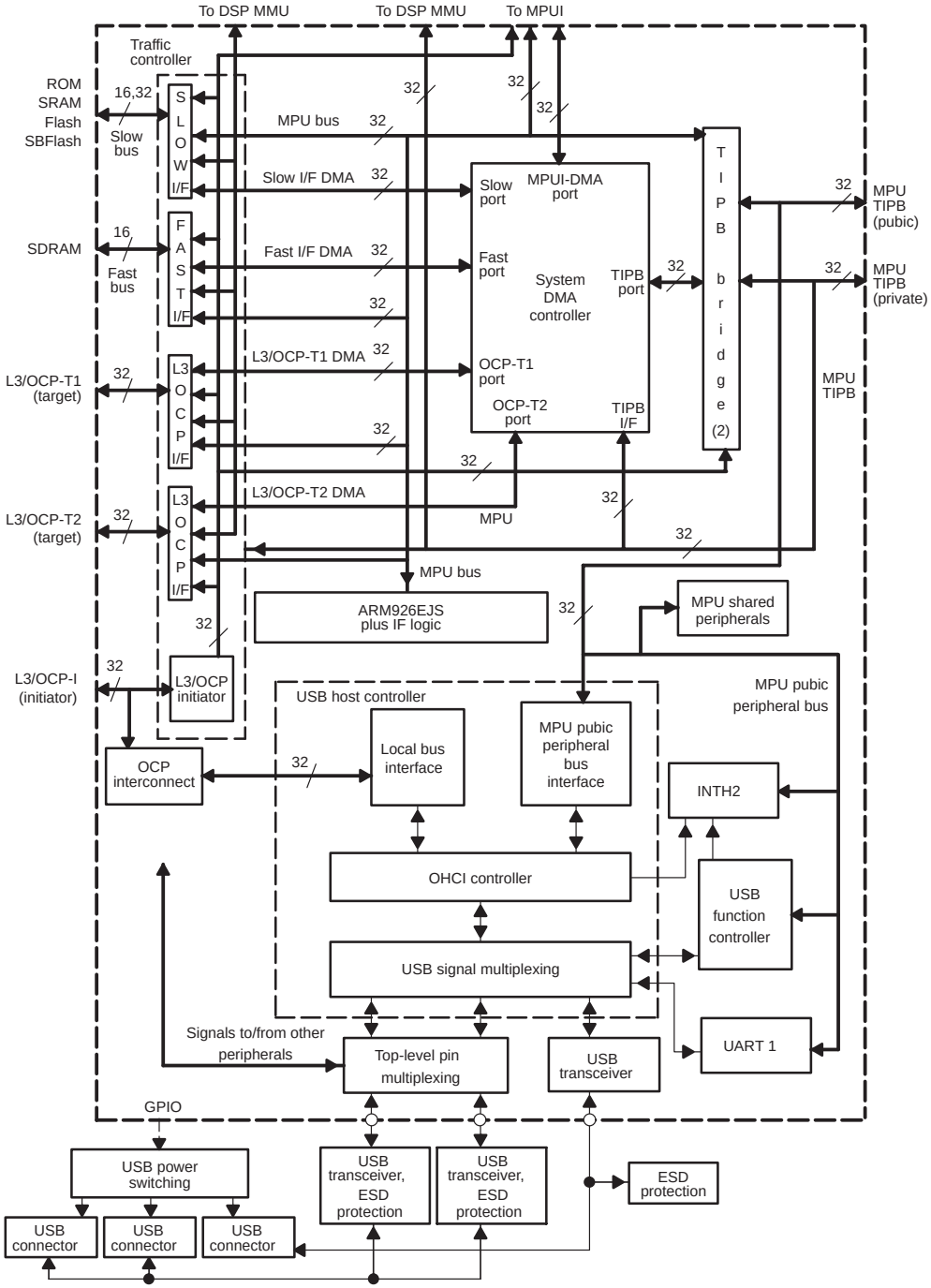
The OMAP5912 USB host controller implements the register set and makes use of the memory data structures defined in the *OHCI Specification for USB*. These registers and data structures are the mechanism by which a USB host controller driver software package can control the OMAP5912 USB host controller. The *OHCI Specification for USB* also defines how the USB host controller implementation must interact with those registers and data structures in system memory. The OMAP5912 MPU accesses these registers via the OMAP5912 MPU public peripheral bus.

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 OMAP5912 USB host controller accesses these data structures without direct intervention by the processor using the OMAP5912 open-core protocol (OCPI) bus. These data structures and data buffers can be located in internal or external system RAM.

The USB host controller provides an interrupt to the MPU level 2 interrupt handler to signal certain hardware events to the host controller driver software.

Figure 1 shows the OMAP5912 USB host controller.

Figure 1. USB Host Controller



2.1 USB Open Host Controller Interface Functionality

2.1.1 OHCI Controller Overview

The *Open HCI—Open Host Controller Interface Specification for USB*, Release 1.0a, 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 *Universal Serial Bus Specification Version 2.0*, defines most of the USB functionality that the OMAP5912 USB host controller provides.

The *OHCI Specification for USB* 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. Also discussed are issues such as interrupt generation, USB host controller state, USB frame management, and the methods that 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 for USB* or the *USB Specification*. OMAP5912 USB host controller users can refer to the *USB Specification* and the *OHCI Specification for USB* for detailed discussions of USB requirements and OHCI controller operation.

2.2 OMAP5912 USB Host Controller Differences From OHCI Specification for USB

The OMAP5912 USB host controller implementation does not implement every aspect of the functionality defined in the *OHCI Specification for USB*. The differences focus on power switching, overcurrent reporting, and the OHCI ownership change interrupt. Other restrictions are imposed by the effects of OMAP5912 pin multiplexing options.

2.2.1 Power Switching Output Pins Not Supported

The OMAP5912 device does not provide pins that can be controlled directly by the USB host controller OHCI port power control features. The OHCI $\overline{\text{RHPORTSTATUS}}$ register port power control bits can be programmed by the USB host controller driver software, but this does not have any direct effect on any VBUS switching implemented on the board.

Users can use software control of GPIO pins or other implementation-specific control mechanisms to control VBUS switching.

2.2.2 Overcurrent Protection Input Pins Not Supported

The OMAP5912 device does not provide any pins that allow the USB host controller OHCI $\overline{\text{RHPORTSTATUS}}$ overcurrent protection status bits to be directly controlled by external hardware.

Users can use software monitoring of GPIO pins or other implementation-specific control mechanisms to report port overcurrent information to the USB host controller driver.

2.2.3 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 OMAP5912 pins. The OHCI RHDESCRIPTORA register always reports three available USB host ports, regardless of the HMC_MODE setting (see Table 20) or top-level pin multiplexing settings. When the HMC_MODE setting disables a USB host controller port, the USB host controller sees that port as unattached.

When OMAP5912 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.

2.2.4 No Ownership Change Interrupt

The OMAP5912 USB host controller does not implement the OHCI ownership change interrupt.

2.3 OMAP5912 Implementation of *OHCI Specification for USB*

2.3.1 Isochronous Transmit Descriptor (TD) OFFSETX/PSWX Values

The OMAP5912 USB host controller implements the *OHCI Specification for USB* optional feature of 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.

2.3.2 OMAP5912 USB Host Controller Endpoint Descriptor (ED) List Head Pointers

The OHCI *Specification for USB* 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 OMAP5912 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 OMAP5912 USB host controller attempts to access using the physical address of 0, which causes an unrecoverable error to be signaled. HOSTUEADDR, HOSTUESTATUS, and HOSTTIMEOUTCTRL registers are updated in this case.

2.3.3 OHCI USB Suspend State

The OMAP5912 USB host controller ignores upstream traffic from downstream devices for about 3 ms after the host controller state (HCCONTROL.HCFS) 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 that system software must 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 packets are to be sent on the bus segment. It is recommended that system software disable all list enable bits (HCCONTROL.PLE, HCCONTROL.IE, HCCONTROL.CLE, and HCCONTROL.BLE) and then wait for at least 1 ms before setting the host controller into USB suspend state (via HCCONTROL.HCFS). When restoring from suspend, 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 has been placed into the USB suspend state under software control, but is brought out by a remote wake-up, system software must monitor the HCRHPORTSTATUS[x].PSS and HCRHPORTSTATUS[x].PSSC bits. The HCRHPORTSTATUS[x].PSS bit changes to 0 only after completion of resume signaling on the bus segment completes 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 so 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 HCRHPORTSTATUS[n].PSSC bit = 1 and

HCRHPORTSTATUS[n].PSS = 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 mode. While the port is in suspend or being resumed, system software must not enable any EDs for any devices downstream of the suspended port. Once the root hub status change interrupt

occurs as a result of the suspended port PSS bit changing to 0, EDs can be enabled for devices downstream of the port that is now operational.

2.4 USB Host Controller Registers

Most of the OMAP5912 host controller (HC) registers are the OHCI operational registers, which are defined by the *OHCI Specification for USB*. Four additional registers not specified by the *OHCI Specification for USB* provide additional information about the USB host controller state. USB host controller registers can be accessed in user and supervisor modes.

The OMAP5912 USB host controller registers are listed in Table 1. Table 2 through Table 29 describe specific register bits.

Table 1. USB Host Controller Registers

Name	Description	R/W	Size†	Address
HCREVISION	OHCI revision number	R	32	FFFB:A000h
HCCONTROL	HC operating mode	R/W	32	FFFB:A004h
HCCOMMANDSTATUS	HC command and status	R/W	32	FFFB:A008h
HCINTERRUPTSTATUS	HC interrupt status	R/W	32	FFFB:A00Ch
HCINTERRUPTENABLE	HC interrupt enable	R/W	32	FFFB:A010h
HCINTERRUPTDISABLE	HC interrupt disable	R/W	32	FFFB:A014h
HCHCCA	Physical address of HCCA‡	R/W	32	FFFB:A018h

† Access to these registers must be by 32-bit reads or 32-bit writes. Use of other access sizes may result in undefined operation.
‡ Restrictions apply to the physical addresses used in these registers. See Section 2.9, *Physical Addressing*.
§ This register provides control and status for the OMAP5912 pins normally associated with USB port 0 (the integrated USB transceiver) for some HMC_MODE values.
¶ This register provides control and status for the OMAP5912 pins normally associated with USB port 1 for some HMC_MODE values.
# This register provides control and status for the OMAP5912 pins normally associated with USB port 2 for some HMC_MODE values.

Table 1. USB Host Controller Registers (Continued)

Name	Description	R/W	Size†	Address
HCPERIODCURRENTED	Physical address of current periodic endpoint descriptor‡	R/W	32	FFFB:A01Ch
HCCONTROLHEADED	Physical address of head of control endpoint descriptor list‡	R/W	32	FFFB:A020h
HCCONTROLCURRENTED	Physical address of current control endpoint descriptor‡	R/W	32	FFFB:A024h
HCBULKHEADED	Physical address of head of bulk endpoint descriptor list‡	R/W	32	FFFB:A028h
HCBULKCURRENTED	Physical of current bulk endpoint descriptor‡	R/W	32	FFFB:A02Ch
HCDONEHEAD	Physical address of head of list of retired transfer descriptors‡	R	32	FFFB:A030h
HCFMINTERVAL	HC frame interval	R/W	32	FFFB:A034h
HCFMREMAINING	HC frame remaining	R	32	FFFB:A038h
HCFMNUMBER	HC frame number	R	32	FFFB:A03Ch
HCPERIODICSTART	HC periodic start	R/W	32	FFFB:A040h
HCLSTHRESHOLD	HC low speed threshold	R/W	32	FFFB:A044h
HCRHDESCRIPTORA	HC root hub A	R, R/W	32	FFFB:A048h
HCRHDESCRIPTORB	HC root hub B	R/W	32	FFFB:A04Ch
HCRHSTATUS	HC root hub status	R, R/W	32	FFFB:A050h
HCRHPORTSTATUS1	HC port 1 control and status§	R, R/W	32	FFFB:A054h
HCRHPORTSTATUS2	HC port 2 control and status¶	R, R/W	32	FFFB:A058h
HCRHPORTSTATUS3	HC port 3 control and status#	R, R/W	32	FFFBA05Ch

† Access to these registers must be by 32-bit reads or 32-bit writes. Use of other access sizes may result in undefined operation.
‡ Restrictions apply to the physical addresses used in these registers. See Section 2.9, *Physical Addressing*.
§ This register provides control and status for the OMAP5912 pins normally associated with USB port 0 (the integrated USB transceiver) for some HMC_MODE values.
¶ This register provides control and status for the OMAP5912 pins normally associated with USB port 1 for some HMC_MODE values.
# This register provides control and status for the OMAP5912 pins normally associated with USB port 2 for some HMC_MODE values.

Table 1. USB Host Controller Registers (Continued)

Name	Description	R/W	Size†	Address
Reserved	Reserved	None		FFFB:A060h to FFFB:A0DFh
HOSTUEADDR	Host UE address	R	32	FFFB:A0E0h
HOSTUESTATUS	Host UE status	R	32	FFFB:A0E4h
HOSTTIMEOUTCTRL	Host time-out control	R/W	32	FFFB:A0E8h
HOSTREVISION	Host revision	R	32	FFFB:A0ECh
Reserved	Reserved	None		FFFB:A0F0h to FFFB:AFFFh

† Access to these registers must be by 32-bit reads or 32-bit writes. Use of other access sizes may result in undefined operation.
‡ Restrictions apply to the physical addresses used in these registers. See Section 2.9, *Physical Addressing*.
§ This register provides control and status for the OMAP5912 pins normally associated with USB port 0 (the integrated USB transceiver) for some HMC_MODE values.
¶ This register provides control and status for the OMAP5912 pins normally associated with USB port 1 for some HMC_MODE values.
# This register provides control and status for the OMAP5912 pins normally associated with USB port 2 for some HMC_MODE values.

The OCHI revision number (Table 2) register reports the revision number of the *OHCI Specification for USB* upon which the USB host controller is based.

Table 2. OHCI Revision Number Register (HCREVISION)

Bit	Name	Description	Type	Reset
31:8	Reserved	Reserved		0x00 0000
7:0	REV	OHCI specification revision—the OHCI revision number upon which the USB host controller is based. Write has no effect.	R	0x10

The HC operating mode register (Table 3) controls the operating mode of the USB host controller.

Table 3. HC Operating Mode Register (HCCONTROL)

Bit	Name	Value	Description	Type	Reset
31:11	Reserved		Reserved		
10	RWE		Remote wake-up enable. This bit has no effect in OMAP5912. The OMAP5912 USB host controller does not provide a processor wake-up mechanism.	R/W	0

Table 3. HC Operating Mode Register (HCCONTROL)(Continued)

Bit	Name	Value	Description	Type	Reset
9	RWC		Remote wake-up connected. This bit has no effect in OMAP5912. The OMAP5912 USB host controller does not provide a processor wake-up mechanism.	R/W	0
8	IR		Interrupt routing. The OMAP5912 USB host controller does not provide an SMI interrupt. This bit must be 0 to allow the USB host controller interrupt to propagate to the MPU level 2 interrupt controller.	R/W	0
7:6	HCFS		Host controller functional state:	R/W	00
		00	USB reset		
		01	USB resume		
		10	USB operational		
		11	USB suspend		
			A transition to USB operational causes SOF generation to begin in 1 ms. The USB host controller can automatically transition from USB suspend to USB resume if a downstream resume is received. The USB host controller enters USB suspend after a software reset. The USB host controller enters USB reset after a hardware reset. The USB reset state resets the root hub and causes downstream signaling of USB reset.		
5	BLE		Bulk list enable:	R/W	0
		0	Bulk ED list not processed in the next 1-ms frame. Host controller driver can modify the list. If driver removes the ED pointed to by the HCBULKCURRENTED from the ED list, it must update HCBULKCURRENTED to point to an ED still on the list before it re-enables the bulk list.		
		1	Enables processing of bulk ED list. HCBULKHEADED must be 0 or point to a valid ED before setting this bit. HCBULKCURRENTED must point to a valid ED or be 0 before setting this bit.		
4	CLE		Control list enable:	R/W	0

Table 3. HC Operating Mode Register (HCCONTROL)(Continued)

Bit	Name	Value	Description	Type	Reset
		0	Control ED list is not processed in the next 1-ms frame. Host controller driver can modify the control ED list. If driver removes the ED pointed to by the HCCONTROLCURRENTED from the ED list, it must update HCCONTROLCURRENTED to point to an ED still on the list before it re-enables the control list.		
		1	Enables processing of the control ED list. HCCONTROLHEADED must be 0 or point to a valid ED before setting this bit. HCCONTROLCURRENTED must be 0 or point to a valid ED before setting this bit.		
3	IE		Isochronous enable	R/W	0
		0	Isochronous EDs are not processed. The USB host controller checks this bit every time it finds an isochronous ED in the periodic list. When this bit is written to 1, processing of isochronous EDs can be enabled in the next frame, if not in the current frame.		
		1	Enables processing of isochronous EDs		
2	PLE		Periodic list enable	R/W	0
		0	The periodic ED lists are not processed. When written to 0, periodic list processing is disabled beginning with the next frame.		
		1	Enables processing of the periodic ED lists. When written to 1, periodic list processing begins in the next frame.		
1:0	CBSR		Control/bulk service ratio Specifies the ratio between control and bulk EDs processed in a frame	R/W	00
		00	One control ED per bulk ED		
		01	Two control EDs per bulk ED		
		10	Three control EDs per bulk ED		
		11	Four control EDs per bulk ED		

The HC command and status register shows the current state of the host controller and accepts commands from the host controller driver.

Table 4. HC Command and Status Register (HCCOMMANDSTATUS)

Bit	Name	Description	Type	Reset
31:18	Reserved	Reserved		
17:16	SOC	Scheduling overrun count	R	00
		Counts the number of times a scheduling overrun occurs. This count is incremented even if the host controller driver has not acknowledged any previous pending scheduling overrun interrupt.		
15:4	Reserved	Reserved		
3	OCR	Ownership change request	R/W	0
		This bit is set by the host controller driver to gain ownership of the host controller.		
		OMAP5912 does not support SMI interrupts, so no ownership change interrupt occurs.		
2	BLF	Bulk list filled	R/W	0
		The host controller driver must set this bit if it modifies the bulk list to include new TDs. If HCBULKCURRENTED is 0, the USB host controller does not begin processing bulk list EDs unless this bit is set. When the USB host controller sees this bit set and begins processing the bulk list, it clears this bit.		
1	CLF	Control list filled	R/W	0
		The host controller driver must set this bit if it modifies the control list to include new TDs. If HCCONTROLHEADED is 0, the USB host controller does not begin processing control list EDs unless this bit is set. When the USB host controller sees this bit set and begins processing the control list, it clears this bit.		
0	HCR	Host controller reset	R/W	0
		Write of 0 has no effect.		
		1: This bit initiates a software reset of the USB host controller. This transitions the USB host controller to the USB suspend state. This resets most USB host controller OHCI registers. OHCI register accesses must not be attempted until a read of this register returns a 0. A write of 1 to this bit does not reset the root hub and does not signal USB reset to downstream USB functions.		

The HC interrupt status register reports the status of the USB host controller internal interrupt sources.

Table 5. HC Interrupt and Status Register (HCINTERRUPTSTATUS)

Bit	Name	Description	Type	Reset
31	Reserved	Reserved		
30	OC	Ownership change The OMAP5912 USB host controller does not implement ownership change interrupts.	R	0
29:7	Reserved	Reserved		
6	RHSC	Root hub status change When 1, indicates a root hub status change has occurred. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0
5	FNO	Frame number overflow When 1, indicates a frame number overflow has occurred. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0
4	UE	Unrecoverable error When 1, indicates that an unrecoverable error has occurred on the OCPI bus or that an isochronous TD PSW field condition code was not set to *Not Accessed* when the USB host controller attempted to perform a transfer using that PSW/offset pair. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0
3	RD	Resume detected When 1, indicates that a downstream device has issued a resume request. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0
2	SF	Start of frame When 1, indicates that a SOF has been issued. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0

Table 5. HC Interrupt and Status Register (HCINTERRUPTSTATUS) (Continued)

Bit	Name	Description	Type	Reset
1	WDH	Write done head When 1, indicates that the USB host controller has updatedthe HCDONEHEAD register. Write of 0 has no effect. Write of 1 clears this bit. The host controller driver must read the value from HCDONEHEAD before writing 1 to this bit.	R/W	0
0	SO	Scheduling overrun When 1, indicates that a scheduling overrun has occurred. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0

The HC interrupt enable register (Table 6) enables various OHCI interrupt sources to generate interrupts to the OMAP5912 level 2 interrupt handler.

Table 6. HC Interrupt Enable Register (HCINTERRUPTENABLE)

Bit	Name	Description	Type	Reset
31	MIE	Master interrupt enable When 1, allows other enabled OHCI interrupt sources to propagate to the OMAP5912 level 2 interrupt controller. When 0, OHCI interrupt sources are ignored and no USB host controller interrupts are propagated to the OMAP5912 level 2 interrupt controller. A write of 0 has no effect on this bit. A write of 1 sets this bit.	R/W	0
30	OC	Ownership change This bit has no effect on OMAP5912.	R	0
29:7	Reserved	Reserved		
6	RHSC	Root hub status change When 1 and MIE is 1, allows root hub status change interrupts to propagate to the OMAP5912 level 2 interrupt controller. When 0, or when MIE is 0, root hub status change interrupts do not propagate. A write of 0 has no effect on this bit. A write of 1 sets this bit.	R/W	0

Table 6. HC Interrupt Enable Register (HCINTERRUPTENABLE) (Continued)

Bit	Name	Description	Type	Reset
5	FNO	Frame number overflow When 1 and MIE is 1, allows frame number overflow interrupts to propagate to the OMAP5912 level 2 interrupt controller. When 0, or when MIE is 0, frame number overflow interrupts do not propagate. A write of 0 has no effect on this bit. A write of 1 sets this bit.	R/W:	0
4	UE	Unrecoverable error When 1 and MIE is 1, allows unrecoverable error interrupts to propagate to the OMAP5912 level 2 interrupt controller. When 0, or when MIE is 0, unrecoverable error interrupts do not propagate. A write of 0 has no effect on this bit. A write of 1 sets this bit.	R/W	0
3	RD	Resume detected When 1 and MIE is 1, allows resume detected interrupts to propagate to the OMAP5912 level 2 interrupt controller. When 0, or when MIE is 0, resume detected interrupts do not propagate. A write of 0 has no effect on this bit. A write of 1 sets this bit.	R/W	0
2	SF	Start of frame When 1 and MIE is 1, allows start of frame interrupts to propagate to the OMAP5912 level 2 interrupt controller. When 0, or when MIE is 0, start of frame interrupts do not propagate. A write of 0 has no effect on this bit. A write of 1 sets this bit.	R/W	0

Table 6. HC Interrupt Enable Register (HCINTERRUPTENABLE) (Continued)

Bit	Name	Description	Type	Reset
1	WDH	Write done head When 1 and MIE is 1, allows write done head interrupts to propagate to the OMAP5912 level 2 interrupt controller. When 0, or when MIE is 0, write done head interrupts do not propagate. A write of 0 has no effect on this bit. A write of 1 sets this bit.	R/W	0
0	SO	Scheduling overrun When 1 and MIE is 1, allows scheduling overrun interrupts to propagate to the OMAP5912 level 2 interrupt controller. When 0, or when MIE is 0, scheduling overrun interrupts do not propagate. A write of 0 has no effect on this bit. A write of 1 sets this bit.	R/W	0

The HC interrupt disable register is used to clear bits in the HCINTERRUPTENABLE register.

Table 7. HC Interrupt Disable Register (HCINTERRUPTDISABLE)

Bit	Name	Description	Type	Reset
31	MIE	Master interrupt enable Read always returns 0. Write of 0 has no effect. Write of 1 clears the HCINTERRUPTENABLE MIE bit.	R/W	0
30	OC	Ownership change This bit has no effect on OMAP5912.	R	0
29:7	Reserved	Reserved		
6	RHSC	Root hub status change Read always returns 0. Write of 0 has no effect. Write of 1 clears the HCINTERRUPTENABLE RHSC bit.	R/W	0

Table 7. HC Interrupt Disable Register (HCINTERRUPTDISABLE) (Continued)

Bit	Name	Description	Type	Reset
5	FNO	Frame number overflow Read always returns 0. Write of 0 has no effect. Write of 1 clears the HCINTERRUPTENABLE FNO bit.	R/W	0
4	UE	Unrecoverable error Read always returns 0. Write of 0 has no effect. Write of 1 clears the HCINTERRUPTENABLE UE bit.	R/W	0
3	RD	Resume detected Read always returns 0. Write of 0 has no effect. Write of 1 clears the HCINTERRUPTENABLE RD bit.	R/W	0
2	SF	Start of frame Read always returns 0. Write of 0 has no effect. Write of 1 clears the HCINTERRUPTENABLE SF bit.	R/W	0
1	WDH	Write done head Read always returns 0. Write of 0 has no effect. Write of 1 clears the HCINTERRUPTENABLE WDH bit.	R/W	0
0	SO	Scheduling overrun Read always returns 0. Write of 0 has no effect. Write of 1 clears the HCINTERRUPTENABLE SO bit.	R/W	0

The HCAA address register defines the physical address of the beginning of the HCCA.

Table 8. HC HCAA Address Register (HCHCCA)

Bit	Name	Description	Type	Reset
31:8	HCCA	Physical address of the beginning of the HCCA	R/W	0x000000
7:0	Reserved	Reserved	R	x00

The HC current periodic register defines the physical address of the next endpoint descriptor (ED) on the periodic ED List.

Table 9. HC Current Periodic Register (HCPERIODCURRENTED)

Bit	Name	Description	Type	Reset
31:4	PCED	Physical address of current ED on the periodic ED list. This field represents bits 31:4 of the physical address of the next ED on the periodic ED List. EDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. See Section 2.9, *Physical Addressing*, for the restrictions on physical addresses.	R	0x0000000
3:0	Reserved	Reserved	R	0x0

The HC head control register defines the physical address of the head ED of the control ED list.

Table 10. HC Head Control Register (HCCONTROLHEADED)

Bit	Name	Description	Type	Reset
31:4	CHED	Physical address of head ED on the control ED list This field represents bits 31:4 of the physical address of the head ED on the control ED list. EDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. See Section 2.9, *Physical Addressing*, for the restrictions on physical addresses.	R/W	0x0000000
3:0	Reserved	Reserved	R	0x0

The HC current control register defines the physical address of the next ED on the control ED list.

Table 11. HC Current Control Register (HCCONTROLCURRENTED)

Bit	Name	Description	Type	Reset
31:4	CCED	Physical address of current ED on the control ED list This field represents bits 31:4 of the physical address of the next ED on the control ED list. EDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. See Section 2.9, *Physical Addressing*, for the restrictions on physical addresses. A value of 0x0000000 indicates that the USB host controller has reached the end of the control ED list without finding any transfers to process. This register is automatically updated by the USB host controller.	R/W	0x0000000
3:0	Reserved	Reserved	R	0x0

The head bulk register defines the physical address of the head ED on the bulk ED list.

Table 12. HC Head Bulk Register (HCBULKHEADED)

Bit	Name	Description	Type	Reset
31:4	BHED	Physical address of head ED on the bulk ED list This field represents bits 31:4 of the physical address of the head ED on the bulk ED list. EDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. See Section 2.9, *Physical Addressing*, for the restrictions on physical addresses.	R/W	0x0000000
3:0	Reserved	Reserved	R	0x0

The current bulk register defines the physical address of the next ED on the bulk ED list.

Table 13. HC Current Bulk Register (HCBULKCURRENTED)

Bit	Name	Description	Type	Reset
31:4	BCED	Physical address of current ED on the bulk ED list This field represents bits 31:4 of the physical address of the next ED on the bulk ED list. EDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. See Section 2.9, *Physical Addressing*, for the restrictions on physical addresses. A value of 0x0000000 indicates that the USB host controller has reached the end of the bulk ED list without finding any transfers to process. The USB host controller automatically updates this register.	R/W	0x0000000
3:0	Reserved	Reserved	R	0x0

The head done register defines the physical address of the current head of the done TD queue.

Table 14. HC Head Done Register (HCDONEHEAD)

Bit	Name	Description	Type	Reset
31:4	DH	Physical address of the last TD that has added to the done queue. This field represents bits 31:4 of the physical address of the top TD on the done TD queue. TDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. A value of 0x00000000 indicates that there are no TDs on the done queue. The USB host controller automatically updates this register.	R	0x0000000
3:0	Reserved	Reserved	R	0x0

The frame interval register defines the number of 12-MHz clock pulses in each USB frame.

Table 15. HC Frame Interval Register (HCFMINTERVAL)

Bit	Name	Description	Type	Reset
31	FIT	Frame interval toggle The host controller driver must toggle this bit any time it changes the frame interval field.	R/W	0
30:16	FSMPS	Largest data packet Largest data packet size allowed for full-speed packets, in bit times.	R/W	0x0000
15:14	Reserved	Reserved		
13:0	FI	Frame interval Number of 12-MHz clocks in the USB frame. Nominally, this is set to 11,999, to give a 1-ms frame. The host controller driver can make minor changes to this field to attempt to manually synchronize with another clock source.	R/W	0x2EDF

The HC frame remaining register reports the number of full-speed bit times remaining in the current frame.

Table 16. HC Frame Remaining Register (HCFMREMAINING)

Bit	Name	Description	Type	Reset
31	FRT	Frame remaining toggle This bit is loaded with the frame interval toggle bit every time the USB host controller loads the frame interval field into the frame remaining field.	R	0
30:14	Reserved	Reserved		
13:0	FR	Frame remaining The number of full-speed bit times remaining in the current frame. This field is automatically reloaded with the frame interval field value at the beginning of every frame.	R	0x0000

The HC frame number register reports the current USB frame number.

Table 17. HC Frame Number Register (HCFMNUMBER)

Bit	Name	Description	Type	Reset
31:16	Reserved	Reserved		
15:0	FN	Frame number	R	0x0000
		This field reports the current USB frame number. It is incremented when the frame remaining field is reloaded with the frame interval field value. Frame number automatically rolls over from 0xFFFF to 0x0000.		
		After frame number is incremented, its new value is written to the HCCA and the USB host controller sets the SOF interrupt status bit and begins processing the ED lists.		

The HC periodic start register defines the position within the USB frame where EDs on the periodic list have priority over EDs on the bulk and control lists.

Table 18. HC Periodic Start Register (HCPERIODICSTART)

Bit	Name	Description	Type	Reset
31:14	Reserved	Reserved		
13:0	PS	Periodic start	R/W	0x0000
		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.		

The HC low-speed threshold register defines the latest time in a frame that the USB host controller can begin a low-speed packet.

Table 19. HC Low-Speed Threshold Register (HCLSTHRESHOLD)

Bit	Name	Description	Type	Reset
31:14	Reserved	Reserved		
13:0	LST	Low-speed threshold	R/W	0x0628
		This field defines the number of full-speed bit times in the frame after which the USB host controller cannot start an 8-byte low-speed packet. The USB host controller only begins a low-speed transaction if the frame remaining field is greater than the low-speed threshold.		
		The host controller driver must set this field to a value that ensures that an 8-byte low-speed TD completes before the end of the frame. When set, the host controller driver must not change the value.		

The HC root hub A register defines several aspects of the USB host controller root hub functionality.

Table 20. HC Root Hub A Register (HCRHDESCRIPTORA)

Bit	Name	Value	Description	Type	Reset
31:24	POTPG		Power-on to power-good time This field 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. This field has no effect on USB host controller operation. After turning on power to a port, the USB host controller driver must delay the amount of time implied by POTPG before attempting to reset an attached downstream device. The required amount of time is implementation-specific and must be calculated based on the amount of time the VBUS supply takes to provide valid VBUS to a worst-case downstream USB function controller. The implementation-specific value must be computed and then written to this register before the USB host controller driver is initialized. Because OMAP5912 does not provide a direct control from the USB host controller to switch VBUS on and off, this value must take into account any delays caused by other methods of controlling VBUS externally. This bit has no relationship to the OTG controller register bits that relate to VBUS. System software can update this register to simplify host controller driver and/or OTG driver coding.	R/W	0xA
23:13	Reserved		Reserved		
12	NOCP		No overcurrent protection When 1, this bit indicates that the USB host controller does not implement overcurrent protection inputs. OMAP5912 does not provide signals to allow connection of external overcurrent indication signals to the USB host controller, so this bit defaults to 1. This bit has no relationship to the OTG controller register bits that relate to VBUS.	R/W	1

Table 20. HC Root Hub A Register (HCRHDESCRIPTORA) (Continued)

Bit	Name	Value	Description	Type	Reset
11	OCPM		Overcurrent protection mode	R/W	0
			OMAP5912 does not provide host controller overcurrent protection input signals, so this bit has no effect. This bit has no relationship to the OTG controller register bits that relate to VBUS.		
10	DT		Device type	R	0
			This bit is always 0, which indicates that the USB host controller implemented is not a compound device.		
9	NPS		No power switching	R/W	1
		0	Indicates that VBUS power switching is supported and is either per-port or all-port switched per the power switching mode field.		
		1	Indicates that VBUS power switching is not supported and that power is available to all downstream ports when the USB host controller is powered.		
			Because OMAP5912 does not provide connections from the USB host controller to control external VBUS switching, this bit defaults to 1.		
			This bit has no relationship to the OTG controller register bits that relate to VBUS. System software can update this register to simplify host controller driver and/or OTG driver coding.		
8	PSM		Power switching mode	R/W	0
		0	Indicates that all ports are powered at the same time		
		1	Indicates that individual port power switching is supported		

Table 20. HC Root Hub A Register (HCRHDESCRIPTORA) (Continued)

Bit	Name	Value	Description	Type	Reset
			Because OMAP5912 does not provide signals from the USB host controller to control external VBUS switching, this bit defaults to 0.		
7:0	NDP		Number of downstream ports This register defaults to 3 to indicate three downstream ports. The USB signal multiplexing mode and OMAP5912 top-level pin multiplexing features can place the OMAP5912 device in a mode where 0, 1, 2, or 3 of the USB host controller downstream ports are usable. This register reports three ports, regardless of USB signal multiplexing mode and OMAP5912 top-level pin multiplexing mode.	R	0x03

The HC root hub B register defines several aspects of the USB host controller root hub functionality.

Table 21. HC Root Hub B Register (HCRHDESCRIPTORB)

Bit	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. If set, per-port power control is implemented for the corresponding port. If clear, global power control is implemented for the corresponding port. PPCM bit 0 is reserved. PPCM bit 1 is the port power control mask for downstream port 1. PPCM bit 2 is the port power control mask for downstream port 2. PPCM bit 3 is the port power control mask for downstream port 3. PPCM bits 4 through 15 are reserved. OMAP5912 does not provide connections from the USB host controller to pins to provide external port power switching. Systems that implement port power switching must use other mechanisms to control port power. System software can update these bits to simplify host controller driver and/or OTG driver coding.	R/W	0x0000
15:0	DR	Device removable Each bit defines whether a corresponding downstream port has a removable or non-removable device. A cleared bit indicates the corresponding port may have a removable device attached. A set bit indicates that the corresponding port has a non-removable device attached. DR bit 0 is reserved. DR bit 1 is the device removable bit for downstream port 1. DR bit 2 is the device removable bit for downstream port 2. DR bit 3 is the device removable bit for downstream port 3. DR bits 4 through 15 are reserved.	R/W	0x0000

The HC root hub status register reports the USB host controller root hub status.

Table 22. HC Root Hub Status Register (HCRHSTATUS)

Bit	Name	Description	Type	Reset
31	CRWE	Clear remote wake-up enable Write of 0 has no effect. Write of 1 clears the device remote wake-up enable bit.	R/W	0
30:18	Reserved	Reserved		
17	OCIC	Overcurrent indication change This bit is automatically set when the overcurrent indicator bit changes. Write of 0 has no effect. Write of 1 clears this bit. This bit has no relationship to the OTG controller register bits that relate to VBUS. System software can update this register to simplify host controller driver and/or OTG driver coding.	R/W	0
16	LPSC	Local power status change This bit defaults to 0 because the root hub does not support the local power status feature. Write of 0 has no effect. Write of 1 sets the port power status bits for all ports if power switching mode is 0. A write of 1 sets port power status bits for ports with their corresponding port power control mask bits cleared if power switching mode is 1. This bit has no relationship to the OTG controller register bits that relate to VBUS. System software can update this register to simplify host controller driver and/or OTG driver coding.	R/W	0
15	DRWE	Device remote wake-up enable When 1, this bit enables a connect status change event to be treated as a resume event, which causes a transition from USB suspend to USB resume state and sets the resume detected interrupt status bit. When 0, connect status change events do not cause a transition from USB suspend to USB resume state and the resume detected interrupt is not changed. Write of 0 has no effect. Write of 1 sets the device remote wake-up enable bit.	R/W	0
14:2	Reserved	Reserved		

Table 22. HC Root Hub Status Register (HCRHSTATUS) (Continued)

Bit	Name	Description	Type	Reset
1	OCI	Overcurrent indicator This bit reports global overcurrent indication if global overcurrent reporting is selected. When 1, this bit indicates that an overcurrent condition has been sensed. When 0, no overcurrent condition has been sensed. Because OMAP5912 does not provide signals for external hardware to report overcurrent status to the USB host controller, this bit is always 0. This bit has no relationship to the OTG controller register bits that relate to VBUS.	R	0
0	LPS	Local power status The root hub does not support the local power status feature. This bit always reads as 0. Write of 0 has no effect. Write of 1 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), a write of 1 turns off power to those ports whose corresponding port power control mask bit is 0. Because OMAP5912 does not provide signals from the USB host controller to external VBUS switching circuitry, this bit has no effect. This bit has no relationship to the OTG controller register bits that relate to VBUS. System software can update this register to simplify host controller driver and/or OTG driver coding.	R/W	0

The HC port 1 status and control register reports and controls the state of USB host port 1. HCRHPORTSTATUS1 can provide status and control for the host controller port that is usually associated with the OMAP5912 integrated USB transceiver and the associated OMAP5912 pins:

- ❏ USB.DP
- ❏ USB.DM

Alternately, OMAP5912 top-level pin multiplexing can also configure the host controller port associated with this register to provide status and control for a USB host port connected to the following pins:

- MCSI2.DOUT/USB0.TXEN
- UART2.TX/USB0.TXD
- UART2.RTS/USB0.SE0
- UART2.CTS/USB0.RCV.
- MCSI2.DIN/USB0.VP
- UART2.RX/USB0.VM
- MCSI2.SYNC/USB0.SPEED
- MCSI2.CLK/USB0.SUSP

The host controller port associated with this register can also be held in a state where it always appears to be disconnected, depending on the HMC_MODE value and top-level pin multiplexing. See Table 22 and 4.3, *Pin Multiplexing*.

HC port 1 can act as the host portion of an OTG controller. See Section 4, *USB OTG Controller*.

Table 23. HC Port 1 Status and Control Register (HCRHPORTSTATUS1)

Bit	Name	Description	Type	Reset
31:21	Reserved	Reserved		
20	PRSC	Port 1 reset status change This bit indicates, when 1, that the port 1 port reset status bit has changed. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0
19	OCIC	Port 1 overcurrent indicator change This bit indicates, when 1, that the port 1 port overcurrent indicator has changed. Write of 0 has no effect. Write of 1 clears this bit. The OMAP5912 does not provide inputs for signaling external overcurrent indication to the USB host controller. Overcurrent monitoring, if required, must be handled through some other mechanism. This bit has no relationship to the OTG controller register bits that relate to VBUS.	R/W	0

Table 23. HC Port 1 Status and Control Register (HCRHPORTSTATUS1) (Continued)

Bit	Name	Description	Type	Reset
18	PSSC	Port 1 suspend status change This bit indicates, when 1, that the port 1 port suspend status has changed. Suspend status is considered to have changed only after the resume pulse, low-speed EOP, and 3-ms synchronization delays have been completed. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0
17	PESC	Port 1 enable status change This bit indicates, when 1, that the port 1 port enable status changed. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0
16	CSC	Port 1 connect status change This bit indicates, when 1, that the port 1 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, then this bit is set. A write of 1 clears this bit. A write of 0 has no effect. If the HCRHDESCRIPTORB.DR[1] bit is set to indicate a non-removable USB device on port 1, this bit is set only after a root hub reset to inform the system that the device is attached.	R/W	0
15:10	Reserved	Reserved		
9	LSDA/CPP	Port 1 low-speed device attached/clear port power This bit indicates, when read as 1, that a low-speed device is attached to port 1. A 0 in this bit indicates a full-speed device. This bit is valid only when port 1 current connect status is 1. The host controller driver can write a 1 to this bit to clear the port 1 port power status. A write of 0 to this bit has no effect. OMAP5912 USB host controller does not control external port power using OHCI mechanisms, so, if required, USB host port power must be controlled through other means. This bit has no relationship to the OTG controller register bits that relate to VBUS. System software can update this register to simplify host controller driver and/or OTG driver coding.	R/W	0

Table 23. HC Port 1 Status and Control Register (HCRHPORTSTATUS1) (Continued)

Bit	Name	Description	Type	Reset
8	PPS/SPP	Port 1 port power status/set port power This bit indicates, when read as 1, that the port 1 power is enabled. When read as 0, port 1 power is not enabled. The OMAP5912 does not provide signals from the USB host controller to control external port power, so if required, USB host port power control signals must be controlled through other means. Software can track the current power state using the port power status bit and other power control bits, but those bits have no direct effect on external port power control. This bit has no relationship to the OTG controller register bits that relate to VBUS. System software can update this register to simplify host controller driver and/or OTG driver coding. A write of 1 to this bit sets the port 1 port power status bit. A write of 0 has no effect. This bit has no relationship to the OTG controller register bits that relate to VBUS. System software can update this register to simplify host controller driver and/or OTG driver coding.	R/W	1
7:5	Reserved	Reserved		
4	PRS/SPR	Port 1 port reset status/set port reset When read as 1, indicates that port 1 is signalling the USB reset. When read as 0, USB reset is not being sent to port 1. A write of 1 to this bit sets the port 1 port reset status bit and causes the USB host controller to begin signaling USB reset to port 1. A write of 0 to this bit has no effect.	R/W	0
3	POCI/CSS	Port 1 port overcurrent indicator/clear suspend status When read as 1, indicates a port 1 port overcurrent condition has occurred. When 0, no port 1 port overcurrent condition has occurred. OMAP5912 does not provide inputs for signaling external overcurrent indication to the USB host controller. Overcurrent monitoring, if required, must be handled through some other mechanism. A write of 1 to this bit when port 1 port suspend status is 1 causes resume signaling on port 1. A write of 1 when port 1 port suspend status is 0 has no effect. A write of 0 has no effect.	R/W	0

Table 23. HC Port 1 Status and Control Register (HCRHPORTSTATUS1) (Continued)

Bit	Name	Description	Type	Reset
2	PSS/SPS	Port 1 port suspend status/set port suspend When read as 1, indicates that port 1 is in the USB suspend state or is in the resume sequence. When 0, indicates that port 1 is not in the USB suspend state. This bit is cleared automatically at the end of the USB resume sequence and also at the end of the USB reset sequence. If port 1 current connect status is 1, a write of 1 to this bit sets the port 1 port suspend status bit and places port 1 in USB suspend state. If current connect status is 0, a write of 1 instead sets connect status change to inform the USB host controller driver software of an attempt to suspend a disconnected device. A write of 0 to this bit has no effect.	R/W	0
1	PES/SPE	Port 1 port enable status/set port enable When read as 1, indicates that port 1 is enabled. When read as 0, this bit indicates that port 1 is not enabled. 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. A write of 1 to this bit when port 1 current connect status is 1 sets the port 1 port enable status bit. A write of 1 when port 1 current connect status is 0 has no effect. A write of 0 has no effect.	R/W	0
0	CCS/CPE	Port 1 current connection status/clear port enable When read as 1, indicates that port 1 currently has a USB device attached. When 0, indicates that no USB device is attached to port 1. This bit is set to 1 after root hub reset if the HCRHDESCRIPTORB.DR[1] bit is set to indicate a non-removable device on port 1. A write of 1 to this bit clears the port 1 port enable bit. A write of 0 to this bit has no effect.	R/W	0

The HC port 2 status register reports and controls the state of USB host port 2. Depending on HMC_MODE (see Table 23) value, HCRHPORTSTATUS1 can provide status and control for the host controller port that is usually associated with the following OMAP5912 pins:

- MCBSP.CLKX/USB1.TXEN
- MCSI1.DOUT/USB1.TXD
- RST_HOST_OUT/USB1.SE0
- MCSI1.DIN/USB1.RCV
- MCSI1.SYNC/USB1.VP
- MCSI1.CLK/USB1.VM
- CLK32K_OUT/USB1.SPEED
- MPU_BOOT/USB1.SUSP

The host controller port associated with this register can also be held in a state where it always appears disconnected, depending on the HMC_MODE value and top-level pin multiplexing (see Section 4.3, *Pin Multiplexing*).

HC port 2 can act as the host portion of an OTG controller. See Section 4, *USB OTG Controller*.

Table 24. HC Port 2 Status and Control Register (HCRHPORTSTATUS2)

Bit	Name	Description	Type	Reset
31:21	Reserved	Reserved		
20	PRSC	Port 2 reset status change This bit indicates, when 1, that the port 2 port reset status bit has changed. A write of 1 clears this bit. A write of 0 has no effect.	R/W	0
19	OCIC	Port 2 overcurrent indicator change This bit indicates, when 1, that the port 2 port overcurrent indicator has changed. A write of 1 clears this bit. A write of 0 has no effect. The OMAP5912 does not provide inputs for signaling external overcurrent indication to the USB host controller. Overcurrent monitoring, if required, must be handled through some other mechanism. This bit has no relationship to the OTG controller register bits that relate to VBUS.	R/W	0
18	PSSC	Port 2 suspend status changed This bit indicates, when 1, that the port 2 port suspend status has changed. Suspend status is considered to have changed only after the resume pulse, low-speed EOP, and 3-ms synchronization delays have been completed. A write of 1 clears this bit. A write of 0 has no effect.	R/W	0

Table 24. HC Port 2 Status and Control Register (HCRHPORTSTATUS2) (Continued)

Bit	Name	Description	Type	Reset
17	PESC	Port 2 enable status change This bit indicates, when 1, that the port 2 port enable status changed. A write of 1 clears this bit. A write of 0 has no effect.	R/W	0
16	CSC	Port 2 connect status change This bit indicates, when 1, that the port 2 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, then this bit is set. A write of 1 clears this bit. A write of 0 has no effect. If the HCRHDESCRIPTORB.DR[2] bit is set to indicate a nonremovable USB device on port 2, this bit is set only after a root hub reset to inform the system that the device is attached.	R/W	0
15:10	Reserved	Reserved		
9	LSDA/CPP	Port 2 low-speed device attached/clear port power This bit indicates, when read as 1, that a low-speed device is attached to port 2. A 0 in this bit indicates a full-speed device. This bit is valid only when port 2 current connect status is 1. The host controller driver can write a 1 to this bit to clear the port 2 port power status. A write of 0 to this bit has no effect. The OMAP5912 USB host controller does not control external port power using OHCI mechanisms, so, if required, USB host port power must be controlled through other means.	R/W	0

Table 24. HC Port 2 Status and Control Register (HCRHPORTSTATUS2) (Continued)

Bit	Name	Description	Type	Reset
8	PPS/SPP	Port 2 port power status/set port power This bit indicates, when read as 1, that the port 2 power is enabled. When read as 0, port 2 power is not enabled. The OMAP5912 does not provide signals from the USB host controller to control external port power, so, if required, USB host port power control signals must be controlled through other means. Software can track the current power state using the port power status bit and other power control bits, but those bits have no direct effect on external port power control. A write of 1 to this bit sets the port 2 port power status bit. A write of 0 has no effect. This bit has no relationship to the OTG controller register bits that relate to VBUS. System software can update this register to simplify host controller driver and/or OTG driver coding.	R/W	1
7:5	Reserved	Reserved		
4	PRS/SPR	Port 2 port reset status/set port reset When read as 1, indicates that port 2 is sending a USB reset. When read as 0, USB reset is not being sent to port 2. A write of 1 to this bit sets the port 2 port reset status bit and causes the USB host controller to begin signaling USB reset to port 2. A write of 0 to this bit has no effect.	R/W	0
3	POCI/CSS	Port 2 port overcurrent indicator/clear suspend status When read as 1, indicates that a port 2 port overcurrent condition has occurred. When 0, no port 2 port overcurrent condition has occurred. The OMAP5912 does not provide inputs for signaling external overcurrent indication to the USB host controller. Overcurrent monitoring, if required, must be handled through some other mechanism. A write of 1 to this bit when port 2 port suspend status is 1 causes resume signaling on port 2. A write of 1 when port 2 port suspend status is 0 has no effect. A write of 0 has no effect. This bit has no relationship to the OTG controller register bits that relate to VBUS.	R/W	0

Table 24. HC Port 2 Status and Control Register (HCRHPORTSTATUS2) (Continued)

Bit	Name	Description	Type	Reset
2	PSS/SPS	Port 2 port suspend status/set port suspend When read as 1, indicates that port 2 is in the USB suspend state, or is in the resume sequence. When 0, indicates that port 2 is not in the USB suspend state. This bit is cleared automatically at the end of the USB resume sequence and also at the end of the USB reset sequence. If port 2 current connect status is 1, a write of 1 to this bit sets the port 2 port suspend status bit and places port 2 in USB suspend state. If current connect status is 0, a write of 1 instead sets connect status change to inform the USB host controller driver software of an attempt to suspend a disconnected device. A write of 0 to this bit has no effect.	R/W	0
1	PES/SPE	Port 2 port enable status/set port enable When read as 1, indicates that port 2 is enabled. When read as 0, this bit indicates that port 2 is not enabled. 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. A write of 1 to this bit when port 2 current connect status is 1 sets the port 2 port enable status bit. A write of 1 when port 2 current connect status is 0 has no effect. A write of 0 has no effect.	R/W	0
0	CCS/CPE	Port 2 current connection status/clear port enable When read as 1, indicates that port 2 currently has a USB device attached. When 0, indicates that no USB device is attached to port 2. This bit is set to 1 after root hub reset if the HCRHDESCRIPTORB.DR[2] bit is set to indicate a non-removable device on port 2. A write of 1 to this bit clears the port 2 port enable bit. A write of 0 to this bit has no effect.	R/W	0

The HC port 3 status and control register reports and controls the state of USB host port 3. Depending on HMC_MODE (see Table 24) value and top-level pin multiplexing settings, HCRHPORTSTATUS2 can provide status and control for the host controller port that is usually associated with the following OMAP5912 USB pins:

- MCSI2.DOUT/USB2.TXEN
- UART2.TX/USB2.TXD
- UART2.RTS/USB2.SE0
- UART2.CTS/USB2.RCV
- MCSI2.DIN/USB2.VP
- UART2.RX/USB2.VM
- MCSI2.SYNC/USB2.SPEED
- MCSI2.CLK/USB2.SUSP

The host controller port associated with this register can also be held in a state where it always appears to be disconnected, depending on the HMC_MODE value and top-level pin multiplexing (see Section 4.3, *Pin Multiplexing*).

HC port 3 is not capable of acting as the host portion of an OTG controller. OTG functionality is only available through HC port 1 and HC port 2. See Section 4, *USB OTG Controller*.

Table 25. HC Port 3 Status and Control Register (HCRHPORTSTATUS3)

Bit	Name	Description	Type	Reset
31:21	Reserved	Reserved		
20	PRSC	Port 3 reset status change This bit indicates, when 1, that the port 3 port reset status bit has changed. A write of 1 clears this bit. A write of 0 has no effect.	R/W	0
19	OCIC	Port 3 overcurrent indicator change This bit indicates, when 1, that the port 3 port overcurrent indicator has changed. A write of 1 clears this bit. A write of 0 has no effect. The OMAP5912 does not provide inputs for signaling external overcurrent indication to the USB host controller. Overcurrent monitoring, if required, must be handled through some other mechanism. This bit has no relationship to the OTG controller register bits that relate to VBUS. System software can update this register to simplify host controller driver and/or OTG driver coding.	R/W	0

Table 25. HC Port 3 Status and Control Register (HCRHPORTSTATUS3) (Continued)

Bit	Name	Description	Type	Reset
18	PSSC	Port 3 suspend status change This bit indicates, when 1, that the port 3 port suspend status has changed. Suspend status is considered to have changed only after the resume pulse, low-speed EOP, and 3-ms synchronization delays have been completed. A write of 1 clears this bit. A write of 0 has no effect.	R/W	0
17	PESC	Port 3 enable status change This bit indicates, when 1, that the port 3 port enable status changed. A write of 1 clears this bit. A write of 0 has no effect.	R/W	0
16	CSC	Port 3 connect status change This bit indicates, when 1, that the port 3 current connect status has changed because of 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, then this bit is set. A write of 1 clears this bit. A write of 0 has no effect. If the HcRhDescriptorB.DR[3] bit is set to indicate a non-removable USB device on Port 3, this bit is set only after a root hub reset to inform the system that the device is attached.	R/W	0
15:10	Reserved	Reserved		
9	LSDA/CPP	Port 3 low-speed device attached/clear port power This bit indicates, when read as 1, that a low-speed device is attached to port 3. A 0 in this bit indicates a full-speed device. This bit is only valid when port 3 current connect status is 1. The host controller driver can write a 1 to this bit to clear the port 3 port power status. A write of 0 to this bit has no effect. OMAP5912 USB host controller does not control external port power using OHCI mechanisms, so, if required, USB host port power must be controlled through other means. This bit has no relationship to the OTG controller register bits that relate to VBUS. System software can update this register to simplify host controller driver and/or OTG driver coding.	R/W	0

Table 25. HC Port 3 Status and Control Register (HCRHPORTSTATUS3) (Continued)

Bit	Name	Description	Type	Reset
8	PPS/SPP	Port 3 power status/set port power This bit indicates, when read as 1, that the port 3 power is enabled. When read as 0, port 3 power is not enabled. The OMAP5912 does not provide signals from the USB host controller to control external port power, so, if required, USB host port power control signals must be controlled through other means. Software can track the current power state using the port power status bit and other power control bits, but those bits have no direct effect on external port power control. A write of 1 to this bit sets the port 3 power status bit. A write of 0 has no effect. This bit has no relationship to the OTG controller register bits that relate to VBUS. System software can update this register to simplify host controller driver and/or OTG driver coding.	R/W	1
7:5	Reserved	Reserved		
4	PRS/SPR	Port 3 reset status/set port reset When read as 1, indicates that port 3 is sending a USB reset. When read as 0, USB reset is not being sent to port 3. A write of 1 to this bit sets the port 3 reset status bit and causes the USB host controller to begin signaling USB reset to port 3. A write of 0 to this bit has no effect.	R/W	0
3	POCI/CSS	Port 3 overcurrent indicator/clear suspend status When read as 1, indicates that a port 3 overcurrent condition has occurred. When 0, no port 3 overcurrent condition has occurred. The OMAP5912 does not provide inputs for signaling external overcurrent indication to the USB host controller. Overcurrent monitoring, if required, must be handled through some other mechanism. A write of 1 to this bit when port 3 suspend status is 1 causes resume signaling on port 3. A write of 1 when port 3 suspend status is 0 has no effect. A write of 0 has no effect. This bit has no relationship to the OTG controller register bits that relate to VBUS.	R/W	0

Table 25. HC Port 3 Status and Control Register (HCRHPORTSTATUS3) (Continued)

Bit	Name	Description	Type	Reset
2	PSS/SPS	Port 3 port suspend status/set port suspend When read as 1, indicates that port 3 is in the USB suspend state, or is in the resume sequence. When 0, indicates that port 3 is not in the USB suspend state. This bit is cleared automatically at the end of the USB resume sequence and also at the end of the USB reset sequence. If port 3 current connect status is 1, a write of 1 to this bit sets the port 3 suspend status bit and places port 3 in USB suspend state. If current connect status is 0, a write of 1 instead sets connect status change to inform the USB host controller driver software of an attempt to suspend a disconnected device. A write of 0 to this bit has no effect.	R/W	0
1	PES/SPE	Port 3 enable status/set port enable When read as 1, indicates that port 3 is enabled. When read as 0, this bit indicates that port 3 is not enabled. 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. A write of 1 to this bit when port 3 current connect status is 1 sets the port 3 enable status bit. A write of 1 when port 3 current connect status is 0 has no effect. A write of 0 has no effect.	R/W	0
0	CCS/CPE	Port 3 current connection status/clear port enable When read as 1, indicates that port 3 currently has a USB device attached. When 0, indicates that no USB device is attached to port 3. This bit is set to 1 after root hub reset if the HCRHDESCRIPTORB.DR[3] bit is set to indicate a non-removable device on port 3. A write of 1 to this bit clears the port 3 enable bit. A write of 0 to this bit has no effect.	R/W	0

The host UE address register reports the physical address of the last OCPI bus access that caused an unrecoverable error (UE). This register has no meaning until an unrecoverable error has occurred. It also has no meaning if the USB host controller issues an unrecoverable error because the offset checking fault occurred while processing an isochronous TD. This register is not defined by the OHCI specification.

Table 26. Host UE Address Register (HOSTUEADDR)

Bit	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. This information, along with the information in HOSTUESTATUS, can help a developer determine why the USB host issued an OCPI access to a physical address that resulted in an unrecoverable error.	R	0x0000 0000

The host UE status register reports the OCPI bus cycle-type for the last unrecoverable error that occurred. This register has no meaning until an unrecoverable error has occurred. It also has no meaning if the USB host controller issues an unrecoverable error because the offset checking fault occurred while processing an isochronous TD. This register is not defined by the OHCI specification.

Table 27. Host UE Status Register (HOSTUESTATUS)

Bit	Name	Description	Type	Reset
31:1	Reserved	Reserved	R	xxxxxxxx
0	UEAccess	Access type when unrecoverable error occurred When an unrecoverable error occurs because of time-out of a OCPI bus write, this bit is set. When an unrecoverable error occurs because of time-out of a OCPI bus read, this bit is cleared. This bit has no meaning before an unrecoverable error occurs. This information, along with the information in HOSTUEADDR, can help a developer determine why the USB host issued an OCPI bus access that resulted in an unrecoverable error.		0

The host time-out control register controls the USB host controller OCPI bus time-out mechanism. This register is not defined by the OHCI specification.

Table 28. Host Time-out Control Register (HOSTTIMEOUTCTRL)

Bit	Name	Description	Type	Reset
31:1	Reserved	Reserved		
0	TO_DIS	OCPI bus time-out disable When 1, 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. When 0, 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 a 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.	R/W	0

The host revision register returns the revision number for the OMAP5912 USB host controller. This register is not defined by the OHCI specification.

Table 29. Host Revision Register (HOSTREVISION)

Bit	Name	Description	Type	Reset
31:8	Reserved	Reserved	R	xxxxxx
7:4	MAJORREV	Major revision number MAJORREV indicates the major revision number of the USB host controller. The original OMAP5912 USB host controller version implements a major revision number of 0.	R	xx
3:0	MINORREV	Minor revision number MINORREv indicates the minor revision number of the USB host controller. The original OMAP5912 USB host controller version implements a minor revision number of 0.	R	xx

2.5 USB Host Controller Reserved Registers and Reserved Bit Fields

To enhance code reusability with possible future versions of the USB host controller, reads and writes to reserved USB host controller register addresses are to be avoided. Unless otherwise specified, when writing registers that have reserved bits, read-modify-write operations must be used so that the reserved bits are written with their previous values.

2.6 USB Host Controller Registers, USB Reset, and USB Clocking

When the USB host controller clock is disabled, or when the ULPD does not provide 48 MHz to the USB host controller, reads from and writes to the USB host controller registers do not occur correctly. To properly access the USB host controller registers, the USB host controller must be clocked and must be out of reset.

USB host controller clock enable is controlled as described in Table 30**.**

Table 30. USB Host Controller Clock Control

OTG_SYS-CON_ 2 OTG_ PADEN	MOD_CONF_CTRL_0. CONF_MOD_USB_HOST_ HHC_UHOST_EN_R	OTG_ SYSCON2 UHOST_EN	USB Host Clock Enabled?	USB Host in Reset?
0	Don’t care	0	No	Yes
0	Don’t care	1	Yes	No
1	0	Dont care	No	Yes
1	1	Dont care	Yes	No

The USB host controller hardware reset is controlled by the ARM_RSTCT2.PER_EN bit and the OTG_SYSCON_1.SOFT_RESET bit. The USB host controller is held in reset whenever ARM_RSTCT2.PER_EN is 0 or OTG_SYSCON_1.SOFT_RESET is 1. The USB host controller can transition out of reset whenever the clock is enabled and ARM_RSTCT2.PER_EN is 1 and OTG_SYSCON_1.SOFT_RESET is 0.

The USB host controller completes its reset within about 72 cycles of the 48-MHz clock after the host controller clock is transitioned from disabled to enabled using the mechanisms shown in Table 30 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 register and the HCHCCA register return their correct reset default values.

2.7 OHCI Interrupts

The OMAP5912 USB host controller provides an interrupt output to the MPU level 2 interrupt handler on its IRQ_06 interrupt input. This is a level-sensitive interrupt signal, and the MPU level 2 interrupt handler IRQ_06 must be programmed as a level-sensitive input.

2.7.1 OHCI Scheduling Overrun Interrupt

The OHCI scheduling overrun interrupt is supported as described in the *OHCI Specification for USB*.

2.7.2 OHCI HCDONEHEAD Writeback Interrupt

The OHCI HCDONEHEAD writeback interrupt is supported as described in the *OHCI Specification for USB*.

2.7.3 OHCI Start Of Frame Interrupt

The OHCI start of frame interrupt is supported as described in the *OHCI Specification for USB*.

2.7.4 OHCI Resume Detect Interrupt

The OHCI resume detect interrupt is supported as described in the *OHCI Specification for USB*.

2.7.5 OHCI Unrecoverable Error Interrupt

The OHCI unrecoverable error interrupt is supported as described in the *OHCI Specification for US*B. This interrupt occurs if the USB host controller is unable to complete an OCPI bus read or OCPI write within 4096 OCPI bus clocks when the USB host OCPI bus time-out feature is enabled [see Table 15-28, *Host Time-out Control Register (HOSTTIMEOUTCTRL*)]. When an OCPI bus time-out causes an unrecoverable error, HOSTUEADDR and HOSTUESTATUS 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.

2.7.6 OHCI Frame Number Overflow

The OHCI frame number overflow interrupt is supported as described in the *OHCI Specification for USB*.

2.7.7 OHCI Root Hub Status Change

The OHCI root hub status change interrupt is supported as described in the *OHCI Specification for USB*. The OMAP5912 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.

2.7.8 OHCI Ownership Change Interrupt

The optional OHCI ownership change interrupt is not supported.

2.8 USB Host Controller Access to System Memory

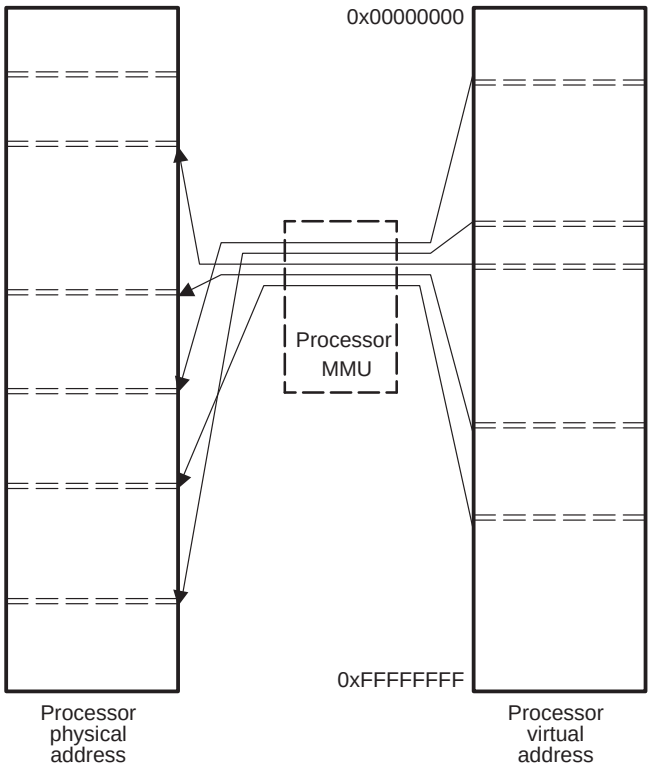
The USB host controller must have access to system memory to read and write the OHCI data structures and data buffers associated with USB traffic. The OMAP5912 OCPI bus allows the USB host controller to access OMAP5912 system memory, as shown in Figure 1.

2.9 Physical Addressing

Transactions on the OMAP5912 OCPI bus use physical addresses, so all system memory accesses initiated by the USB host controller must use physical addresses. The OMAP5912 MPU 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 the appropriate virtual address to physical address and physical address to virtual address conversions when manipulating the USB host controllers data structures and pointers to those data structures.

Figure 2 shows the MPU virtual address to physical address conversion.

Figure 2. Relationships Between Virtual Address Physical Address



2.10 Cache Coherency in OHCI Data Structures and Data Buffers

The OMAP5912 traffic controller does not provide mechanisms to flush (or writeback) the MPU cache when a DMA controller or OCPI access to system memory occurs. Because there is no forced coherency mechanism, the system implementation must ensure that the OMAP5912 USB host controller can access the correct data from system memory and that the MPU accesses that same data. This requires that any system memory accessed by the USB host controller be allocated in non-cached 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 can read from, and, if in writeback mode, write to the cache; but 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 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 non-cacheable system memory. In this case, every MPU access directly accesses physical memory, so there is no coherency issue. Configuration of cacheable portions of the MPU virtual address space is provided via the MPU memory management unit. See the description of the MPU MMU in the *OMAP3.2 Hardware*
Engine Reference Guide (SWPU019C).

2.11 OCPI Bus 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 processor 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 within 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. This list is pointed to by the HCDONEHEAD register, which contains a physical address that points to the most recent transfer

descriptor that has been 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 for proper address conversion depends on the settings used in the MPU MMU, so it is system-dependent. The conversion functions are described in general terms in the following subsections.

2.11.1 MPUVAtoPA()—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.

2.11.2 PAtoMPUVA()—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.

2.12 NULL Pointers

The *OHCI Specification for USB* 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.

2.13 OMAP5912 OCPI Bus and the USB Host Controller

The OMAP5912 OCPI bus provides a mechanism whereby OMAP5912 peripheral modules can access system memory. Unlike peripherals that use

the DMA controller, peripherals connected to the OCPI bus can generate the address, byte enables, and read/write indication for each OCPI access. This functionality allows the USB host controller to implement a scatter-gather engine to access the OHCI data structures from system memory in an efficient manner.

2.14 OCPI Registers

The USB host controller connects to an OCPI arbiter, which then connects to the transfer controller OCP-I port. There are no OCPI arbiter register settings that affect USB host controller access to system memory, but the transfer controller OCP-I port must be properly configured to allow access to system memory.

Because the USB host must be able to access system memory, the OMAP5912 security features must be properly configured to allow USB host controller access to system memory.

2.15 USB Host Controller Clock Control

The OMAP5912 clock generation and system reset management module (ULPD) provides a 48-MHz clock to the USB OTG controller, USB device controller, and USB host controller. This clock can be stopped by software to reduce USB OTG controller, USB device controller, and USB host controller power consumption when USB functionality is not needed.

The ULPD controls the 48-MHz clock to the USB host controller via:

- ❏ CLOCK_CTRL_REG.USB_MCLK_EN
- ❏ SOFT_REQ_REG.SOFT_USB_OTG_DPLL_REQ
- ❏ SOFT_DISABLE_REQ_REG.DIS_USB_HOST_DPLL_REQ

When these ULPD register bits are properly configured, the USB host controller receives a clock when the UHOST_EN signal is active. See discussions of OT_SYSCON_2.OTG_PADEN and OTG_SYSCON_2.HOST_EN in Section 4.

The USB host controller is held in reset and receives no clock at any time that UHOST_EN is inactive.

2.16 USB Host Controller Hardware Reset

The ULPD module provides reset of the USB host controller. The PER_EN bit in the MPU reset control 2 register controls the reset to many OMAP5912

peripherals, including the USB host controller. When held in reset, the USB host controller does not generate any USB activity on its USB ports. The USB host controller requires that its 48-MHz clock input (from the OMAP5912 ULPD module) be active and that UHOST_EN be set (see Table 64, *OTG System Configuration Register 2 (OTG_SYSCON_2)*) in order to complete its reset sequence.

There is a delay of approximately 72 cycles of the ULPD USB host controller 48-MHz clock before the USB host controller is successfully reset. This delay starts at the latest of the PER_EN bit set, UHOST_EN set, or 48-MHz clock start. 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 both HCREVISION and HCHCCA are not correct, reset the values and continue reading until the proper reset values are seen.

The UHOST_EN bit, when cleared, 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 UHOST_EN 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, that the ULPD 48-MHz clock for the USB host controller is enabled, and that UHOST_EN is set. It must then wait until reads of both the HCREVISION register and the HCHCCA register return their correct reset default values.

2.17 USB Host Controller OHCI Reset

The *OHCI Specification for USB* provides the HCR bit in the HCCOMMANDSTATUS register, which resets the OHCI controller. This reset can be used to reset the OHCI functionality and has no effect on the USB host controller OCPI and the MPU public peripheral bus interfaces. The OHCI reset does not affect the USB host controller clock.

2.18 USB Host Controller Power Management

Power management of the OMAP5912 USB host controller is limited to disabling the clock to the USB host by clearing UHOST_EN (see (see Table 64, *OTG System Configuration Register 2 (OTG_SYSCON_2)*). When UHOST_EN is 0, the USB host controller clocks are disabled and the USB host controller is held in reset. The USB signal multiplexing controlled by

HMC_MODE is not affected, so a HMC_MODE setting that multiplexes USB function controller and/or UART1 signals to OMAP5912 top-level multiplexing can still make use of the USB function controller and/or UART1.

When the OMAP5912 host controller 48-MHz clock is disabled or UHOST_EN is 0, all USB host controller OHCI registers and the HOSTUEADDR, HOSTUESTATUS, HOSTTIMEOUTCTRL, and HOSTREVISION registers are inaccessible.

2.19 OCPI Clocking

The OCPI clock must be active to allow USB host controller access to system memory. The OCPI clock is controlled by the ARM_IDLECT3.EN_OCPI_CK and IDLOCPI_ARM bits.

3 USB Device Controller

The USB device controller supports the implementation of a full-speed device fully compliant with the USB 1.1 standard.

It provides an interface between the MPU 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 are for each endpoint, 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.

3.1 USB Device Controller Registers

Table 31 lists the USB device controller registers. Table 32 through Table 54 describe the register bits.

Table 31. USB Device Controller Registers

Register	Description	R/W	Address
REV	Revision		0xFFFB4000
	Endpoint		
EP_NUM	Endpoint selection		0xFFFB4004

Table 31. USB Device Controller Registers(Continued)

Register	Description	R/W	Address
DATA	Data		0xFFFB4008
CTRL	Control		0xFFFB400C
STAT_FLG	Status		0xFFFB4010
RXFSTAT	Receive FIFO status		0xFFFB4014
SYSCON1	System configuration 1		0xFFFB4018
SYSCON2	System configuration 2		0xFFFB401C
DEVSTAT	Device status		0xFFFB4020
SOF	Start of frame		0xFFFB4024
IRQ_EN	Interrupt enable		0xFFFB4028
DMA_IRQ_EN	DMA interrupt enable		0xFFFB402C
IRQ_SRC	Interrupt source		0xFFFB4030
EPN_STAT	Non-ISO endpoint interrupt enable		0xFFFB4034
DMAN_STAT	Non-ISO DMA interrupt enable		0xFFFB4038
Reserved			0xFFFB403C
	DMA Configuration		
RXDMA_CFG	DMA receive channels configuration		0xFFFB4040
TXDMA_CFG	DMA transmit channels configuration		0xFFFB4044
DATA_DMA	DMA FIFO data		0xFFFB4048
Reserved			0xFFFB404C
TXDMA0	Transmit DMA control 0		0xFFFB4050
TXDMA1	Transmit DMA control 1		0xFFFB4054
TXDMA2	Transmit DMA control 2		0xFFFB4058
Reserved			0xFFFB405C
RXDMA0	Receive DMA control 0		0xFFFB4060
RXDMA1	Receive DMA control 1		0xFFFB4064
RXDMA2	Receive DMA control 2		0xFFFB4068

Table 31. USB Device Controller Registers(Continued)

Register	Description	R/W	Address
Reserved			0xFFFB406C...407F
	Endpoint Configuration		
EP0	Endpoint configuration 0		0xFFFB4080
EP1_RX...EP15RX	Receive endpoint configuration 1...15		0xFFFB4084...40BC
Reserved			0xFFFB40C0
EP1_TX...EP15TX	Transmit endpoint configuration 1...15		0xFFFB40C4...40FC

- ❑ 16-bit access: all bits of the register are accessed.
- ❑ 8-LSB bit access: the 8 least significant bits are accessed.
- ❑ 8-MSB bit access: the 8 most significant bits are accessed.

Table 32. Revision Register (REV)

Bit	Name	Description
15:8	-	Reserved
7:0	REV_NB	This 8-bit field indicates revision number of current USB device controller module—value fixed by hardware: 0x01: Revision 0.1 0x02: Revision 0.2 0x21: Revision 2.1

This read-only register contains the revision number of the module. A write to this register is denied.

MPU and USB reset have no effect on this register.

Table 33. Endpoint Selection Register (EP_NUM)

Bit	Name	Description
15:7	–	Reserved
6	SETUP_SEL	The setup FIFO select bit is set by the USB device controller to access the status (STAT_FLG, RXFSTAT) and data (DATA) registers for the endpoint selected. If EP_NUM.EP_DIR bit is set to 0, the MPU can read data from endpoint RX FIFO by reading DATA register. If EP_NUM.EP_DIR bit is set to 1, the MPU can write data into endpoint TX FIFO by writing into the DATA register. After each access to an endpoint during interrupt handling, the USB device controller must absolutely clear this bit: 0: No access 1: Access permitted Note: When the USB device controller sets this bit, it must set SETUP_SEL bit to 0. After having accessed the endpoint FIFO either for read or for write access, the USB device controller must clear this bit by writing a 0 to it. The value after MPU or USB reset is low.
5	EP_SEL	The TX/RX FIFO select bit is set by the USB device controller in response to a set-up general USB interrupt, to access the EP0 read-only setup FIFO when reading DATA register. Setting this bit clears the IRQ_SRC.SETUP interrupt bit. When this bit is set, the value of other EP_NUM register bits must be 0. 0: No access 1: Access permitted Note: After having read the setup FIFO, the USB device controller must clear this bit by writing a 0 to it. Value after MPU or USB reset is low.

Table 33. Endpoint Selection Register (EP_NUM) (Continued)

Bit	Name	Description
4	EP_DIR	The endpoint direction bit gives the direction associated with the endpoint number selected in EP_NUM.EP_NUM: 0: OUT endpoint 1: IN endpoint Value after MPU or USB reset is low.
3:0	EP_NUM	The endpoint number binary encoded in these four bits, associated with the direction given by EP_NUM.EP_DIR bit, is the current endpoint selected. All reads and writes to the endpoint status and the control and data locations are for this endpoint. 0000: EP0 0001: EP1 … 1111: EP15 Value after MPU or USB reset is low.

This read/write register selects and enables the endpoint that can be accessed by the USB device controller.

Table 34. Data Register (DATA)

Bit	Name	Description
15:0	DATA†	Transmit/receive FIFO data: EP_NUM.EP_DIR = 1: This register contains the data written by the USB device controller to be sent to the USB host during the next IN transaction. Data can be written successfully only 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 be read successfully only if the EP_NUM.EP_SEL bit is asserted, or if EP_NUM.SETUP_SEL bit is asserted (for setup data).

† A write into the data register (DATA) when EP_NUM.EP_DIR = 0 and a read from the register when EP_NUM.EP_DIR = 1 are denied.

This register is the entry point to write into a selected TX endpoint or to read data from a selected RX endpoint, or to read data from setup FIFO. If selected endpoint direction is OUT, this register is read-only and a write into it is denied. If selected endpoint direction is IN, this register is write-only and a read to this register is denied.

Table 35. Control Register (CTRL)

Bit	Name	Description
15:8	–	Reserved
7	CLR_HALT†	The clear halt endpoint (non-ISO) bit only concerns non-ISO endpoints—used by the USB device controller to clear an endpoint halt condition: 0: No action 1: Clear halt condition Always read 0
6	SET_HALT†	The set halt endpoint (non-ISO) only concerns non-ISO endpoints—used by the USB device controller to halt the selected endpoint. The halted endpoint returns STALL handshakes to the USB host. The USB device controller can disable the endpoint interrupt if it does not want to be informed of STALL handshakes. Note: If the endpoint to halt is used by a DMA channel, the USB device controller must disable the DMA channel before setting halt condition for this endpoint. 0: No action 1: Halt endpoint Always read 0
5:3	–	Reserved

† It is not required to set EP_NUM.EP_SEL bit before setting this bit. If this bit is set during the handling of an interrupt to the endpoint, however, the USB device controller must not set EP_NUM.EP_SEL bit before setting CTRL.CLR_HALT bit, in order to avoid a possible effect on interrupts . The USB device controller must check that FIFO is empty before setting the halt feature for the endpoint. A STALL transaction clears the FIFO.

Table 35. Control Register (CTRL) (Continued)

Bit	Name	Description
2	SET_FIFO_EN†	The set FIFO enable (non-ISO) bit only concerns non-ISO endpoints. If the selected endpoint direction is IN, the USB device controller uses this bit to enable the USB device to transmit data from the FIFO at the next valid IN token. If the selected endpoint direction is OUT, the USB device controller uses this bit to enable the USB device to receive data from the USB host at the next valid OUT transaction. If not, setting the device returns a NAK handshake. ISO endpoints FIFO are always enabled. Note: The USB device controller must never enable endpoint 0 FIFO out of control transfers. For bulk and interrupt endpoints, FIFO must never be enabled when the halt feature is set or when RX FIFO is not empty. Furthermore, during EP interrupts handling, the USB device controller must have cleared the interrupt bit before setting CTRL.SET_FIFO_EN bit (to avoid masked ACK interrupts). 0: No action 1: FIFO enabled Always read 0
1	CLR_EP	Clear endpoint: the USB device controller sets this bit to clear the selected endpoint FIFO pointers and flags. This bit resets the FIFO pointers, the FIFO empty status bit is set, and the FIFO enable bit and other FIFO flags are cleared upon completion of the FIFO reset. It also clears the previous transaction handshake status. For ISO endpoints or non-ISO double-buffered endpoints, both foreground and background FIFO are cleared. 0: No action 1: Clear endpoint Always read 0
0	RESET_EP	The endpoint reset (non-control) bit only concerns non-control endpoints.The USB device controller sets this bit to reset the selected endpoint. It forces an interrupt or a bulk endpoint data PID to DATA0, clears the halt condition, and clears the FIFO (both foreground and background if endpoint is double-buffered) and previous transactions handshake status. For an ISO endpoint, it only clears the FIFO (both foreground and background). 0: No action 1: Reset endpoint Always read 0

† It is not required to set EP_NUM.EP_SEL bit before setting this bit. If this bit is set during the handling of an interrupt to the endpoint, however, the USB device controller must not set EP_NUM.EP_SEL bit before setting CTRL.CLR_HALT bit, in order to avoid a possible effect on interrupts . The USB device controller must check that FIFO is empty before setting the halt feature for the endpoint. A STALL transaction clears the FIFO.

This set-only register controls the FIFO and status of the selected endpoint. A read access to this register always returns 0.

Endpoint 0 setup FIFO is always enabled and ready to accept setup data. No control register CTRL is implemented for this FIFO because the USB device controller cannot control it.

Table 36. Status Register (STAT_FLG)

Bit	Name	Description
15	NO_RXPACKET	The isochronous no packet received (ISO OUT) bit only concerns ISO OUT endpoints.This bit notifies the USB device controller that the core has not received any isochronous packet on the endpoint. This bit is updated on a SOF (start of frame). 0: The endpoint received a packet on the previous frame. 1: The endpoint did not receive a packet on the previous frame. Value after MPU or USB reset is high.
14	MISS_IN	The isochronous missed IN token for the previous frame (ISO IN) bit only concerns ISO IN endpoints.This bit notifies the USB device controller that the core missed a valid ISO IN token during the previous frame and that TX data were flushed from the FIFO instead of being transmitted to the USB host. This bit is updated on a SOF (start of frame). 0: The endpoint received an IN token the previous frame. 1: The endpoint did not receive an IN token the previous frame and TX data were flushed. Value after MPU or USB reset is low.
13	DATA_FLUSH	The isochronous receive data flush (ISO OUT) bit only concerns ISO OUT endpoints. When set, this bit indicates that data were flushed from the isochronous FIFO that was moved from the foreground to the background. This happens when the USB device controller does not read all of the data from the foreground FIFO in a frame. This bit is updated in every frame. 0: Not significant 1: Data was flushed. Value after MPU or USB reset is low.

Table 36. Status Register (STAT_FLG) (Continued)

Bit	Name	Description
12	ISO_ERR	The isochronous receive data error (ISO OUT) bit only concerns ISO OUT endpoints. When set, this bit indicates that the ISO data packet was received incorrectly. This happens when the core detects an error in the data packet (CRC, bit stuffing, PID check) or when there is an overrun condition in the FIFO. When this bit is set, the FIFO contents are automatically flushed by the core and the FIFO status is empty. This bit is updated in every frame. 0: Not significant 1: ISO packet received with errors Value after MPU or USB reset is low.
11:10	–	Reserved
9	ISO_FIFO_EMPTY	The ISO FIFO empty bit only concerns ISO endpoints.This bit is set when the FIFO for the 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. 0: ISO FIFO is not empty. 1: ISO FIFO is empty. Value after MPU or USB reset is high (FIFO empty).
8	ISO_FIFO_FULL	The ISO FIFO full bit only concerns ISO endpoints.This bit is set when the FIFO for the selected 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 USB device controller or the USB host). 0: ISO FIFO is not full. 1: ISO FIFO is full. Value after MPU or USB reset is low (FIFO empty).
7	-	Reserved
6	EP_HALTED	The endpoint halted flag (non-ISO) bit only concerns non-ISO endpoints.This bit when asserted 1 indicates that the selected endpoint is halted. The endpoint can be put into the halt state only by the USB device controller writing the endpoint halt control bit (in response to a SET_FEATURE request, for instance). 0: The selected endpoint is not halted. 1: The selected endpoint is halted. Value after MPU or USB reset is low.

Table 36. Status Register (STAT_FLG) (Continued)

Bit	Name	Description
5	STALL	The transaction stall (non-ISO) bit only concerns non-ISO endpoints.This status bit is 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 the current buffer (see Section 3.11, *Important Note on USB Device Interrupts*). The core automatically returns a STALL packet if a valid IN token is received by a halted TX endpoint, if a valid OUT transaction is received by a halted RX endpoint, or if there is a request error (endpoint 0). The bit is cleared when the USB device controller has finished handling the corresponding interrupt (at EP_NUM.EP_SEL bit deselection). 0: No STALL handshake was returned. 1: A STALL handshake packet was returned. Value after MPU or USB reset is low.
4	NAK	The transaction non-acknowledge (non-ISO) bit only concerns non-ISO endpoints with SYSCON1.NAK_EN bit asserted.This status bit is 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 the 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. The bit is cleared when the USB device controller has finished handling the corresponding interrupt (at EP_NUM.EP_SEL bit deselection). 0: No NAK handshake was returned (SYSCON1.NAK_EN bit is set). 1: A NAK handshake packet was returned and SYSCON1.NAK_EN bit is set. Value after MPU or USB reset is low.
3	ACK	The transaction acknowledge (non-ISO) bit only concerns non-ISO endpoints.This bit is 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. The bit is cleared when the USB device controller has finished handling the corresponding interrupt (at EP_NUM.EP_SEL bit deselection). 0: No ACK handshake packet was returned. 1: An ACK handshake packet was returned. Value after MPU or USB reset is low.

Table 36. Status Register (STAT_FLG) (Continued)

Bit	Name	Description
2	FIFO_EN	The FIFO enable status (non-ISO) bit only concerns non-ISO endpoints.This bit is asserted when CTRL.SET_FIFO_EN is set to 1 and is cleared automatically after a transaction completes with an ACK, STALL. 0: The non-ISO endpoint FIFO is disabled. 1: The non-ISO endpoint FIFO is enabled. Value after MPU or USB reset is low.
1	NON_ISO_FIFO_EMPTY	The non-ISO FIFO empty bit only concerns non-ISO endpoints.This bit is 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. 0: Non-ISO FIFO is not empty. 1: Non-ISO FIFO isempty. Value after MPU or USB reset is high (FIFO empty).
0	NON_ISO_FIFO_FULL	The non-ISO FIFO full bit only concerns non-ISO endpoints.This bit is 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 USB device controller or the USB host). 0: Non-ISO FIFO is not full. 1: Non-ISO FIFO is full. Value after MPU or USB reset is low (FIFO empty).

This read-only register provides a status of the FIFO and the results of the transactions handshakes for the selected endpoint. The 8 MSB are reserved for ISO endpoints, whereas the 8 LSB are reserved for non-ISO endpoints. This register cannot be read if EP_NUM.EP_SEL bit is not asserted for the endpoint. No status flag exists for the read-only set-up FIFO, which is always enabled.

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 SYSCON1.NAK_EN is asserted to 1. An ERR handshake or a NAK handshake when SYSCON1.NAK_EN is 0 is considered transparent.

A write to this register has no effect.

Table 37. Receive FIFO Status Register (RXFSTAT)

Bit	Name	Description
15:10	–	Reserved
9:0	RXF_COUNT	The receive FIFO byte count 10-bit field indicates the number of bytes currently in the receive FIFO. Values after MPU or USB reset is low (all 10 bits).

This read-only register indicates how many bytes are in the receive FIFO for the selected endpoint. A write to this register has no effect. The USB device controller cannot read this register if the EP_NUM.EP_SEL bit is not set for the endpoint. No receive FIFO status exists for the setup FIFO, because 8 bytes are always expected.

Table 38. System Configuration Register 1 (SYSCON1)

Bit	Name	Description
15:9	-	Reserved
8	CFG_LOCK	Device configuration locked bit: after the USB device controller has entered the device configuration (registers 0x20 to 0x3F), it must set the CFG_LOCK bit so that the device can be used. When the device configuration is not locked, the device is not ready to be used: 0: Device configuration is not locked. Device is not ready. 1: Device configuration is locked. The value after the USB device controller hardware reset is low; after USB reset, it is unchanged (keep previous configuration).

Table 38. System Configuration Register 1 (SYSCON1) (Continued)

Bit	Name	Description
7	DATA_ENDIAN	The data endian bit can be set by the USB device controller to select little- or big-endian format on data access (read or write): 0: Little endian 1: Big endian The value after USB device controller hardware reset is low; after USB reset, it is unchanged (keep previous configuration).
6	DMA_ENDIAN	The DMA data endian bit can be set by the USB device controller to select little- or big-endian format on data access (read or write): 0: Little endian 1: Big endian The value after USB device controller hardware reset is low; after USB reset, it is unchanged (keep previous configuration).
5	-	Reserved
4	NAK_EN	The NAK enable bit can be set by the USB device controller to be signaled for NAK transaction handshake response. When this bit is set, STAT_FLG.NAK is set on a NAK handshake if no non-handled interrupt is pending on the endpoint, and the endpoint interrupt is asserted. In the normal mode, when cleared, NAK handshake response to the USB host is made transparent to the USB device controller and no interrupt is asserted: 0: NAK is disabled. 1: NAK is enabled. The value after MPU or USB reset is low. Note: If the USB device controller sets this bit, it must wait for a NAK interrupt before selecting TX endpoint to write TX data.
3	AUTODEC_DIS	The autodecode process disabled can be set to force software to handle all EP0 transactions (except the SET_ADDRESS transaction): 0: Autodecode process is activated (see Table 55, *Autodecoded Versus Non-Autodecoded Control Requests*). 1: Autodecode process is deactivated. The value after USB device controller hardware reset is low; after USB reset, it is unchanged.

Table 38. System Configuration Register 1 (SYSCON1) (Continued)

Bit	Name	Description
2	SELF_PWR	The self-powered bit indicates to the USB host whether the device is bus-powered or self-powered. This information is needed for an autodecoded request. The USB device controller must update this bit after a SET_CONFIGURATION according to the self-powered bit D6 given in the configuration descriptor (see *USB 1.1 Specification*). 0: Bus powered 1: Self-powered The value after USB device controller hardware reset is low; after USB reset, it is unchanged.
1	SOFF_DIS	When the shutoff disable bit is set, it disables the power shutoff circuitry: 0: Power shutoff circuitry is enabled. 1: Power shutoff circuitry is disabled. The value after USB device controller hardware reset is low; after USB reset, it is unchanged.
0	PULLUP_EN	The 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 USB traffic when the device is not ready: 0: Pull-up is disabled. USB host cannot detect the device. In this mode, the 48-MHz USB clock is forced off. 1: Pull-up is enabled. The value after USB device controller hardware reset is low; after USB reset, it is high; and after detach, it is unchanged.

This read/write register provides control functions for power management and miscellaneous control for the core.

Values depend on whether the reset action comes from the USB device controller (MPU) or the USB host.

Figure 3. Little Endian and Big Endian Formats

Table 39. System Configuration Register 2 (SYSCON2)

Bit	Name	Description
15:7	–	Reserved
6	RMT_WKP	The set-only remote wake-up 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. Reading this bit always returns 0. Writing 0 into this bit has no effect. To generate a resume, the software must check the remote wake-up enable value before initiating any wake-up sequence: 0: No action 1: Initiates the remote wake-up sequence Always read 0
5	STALL_CMD	The 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. 0: No action 1: Stall current USB command Always read 0
4	Reserved	Reserved
3	DEV_CFG	If the USB device controller receives a SET_CONFIGURATION with a valid configuration value, and the device is in addressed state, it must write a 1 to the device configured (DEV_CFG) bit to inform the command decode that the device moves to configured state. The core sets the DEVSTAT.CFG bit to 1. If the device is already configured when the SET_CONFIGURATION request is received, the USB device controller does not have to set this bit. If the new configuration value is 0, USB device controller must set the SYSCON2.CLR_CFG bit to move to addressed state. Reading this bit always returns 0. Writing 0 into this bit has no effect. 0: No action 1: Allows DEVSTAT.CFG to be set Always read 0

Table 39. System Configuration Register 2 (SYSCON2) (Continued)

Bit	Name	Description
2	CLR_CFG	If the USB device controller receives a SET_CONFIGURATION with a configuration value of 0, and if the device is configured, it must write a 1 to the clear configured (CLR_CFG) bit to inform the command decode that the device becomes deconfigured (moves to addressed state). The core clears the DEVSTAT.CFG bit. Reading this bit always returns 0. Writing 0 into this bit has no effect. 0: No action 1: Allows DEVSTAT.CFG to be cleared Always read 0
1:0	Reserved	Reserved

This set-only register provides miscellaneous controls for the function. A read to this register always returns 0.

Table 40. Device Status Register (DEVSTAT)

Bit	Name	Description
15:10	Reserved	Reserved
9	B_HNP_ENABLE	The HNP enable for B-device bit is used for On-The-Go feature selection purposes. See the *On-The-Go Supplement* to the *USB 2.0 Specification*. 0: Not significant 1: HNP enable for B-device Value after MPU or USB reset is low.
8	A_HNP_SUPPORT	The B-device is directly connected to an A-device port that supports HNP. This A_HNP_SUPPORT bit is used for On-The-Go feature selection purposes. See the *On-The-Go Supplement* to the *USB 2.0 Specification*. 0: Not significant 1: B-device connection to HNP capable A-device Value after MPU or USB reset is low.

Table 40. Device Status Register (DEVSTAT) (Continued)

Bit	Name	Description
7	A_ALT_HNP_SUPPORT	The B-device is connected to an A-device port that is not capable of HNP, but the A-device has an alternate port that is capable of HNP. This A_ALT_HNP_SUPPORT bit is used for On-The-Go feature selection purposes. See the *On-The-Go Supplement* to the *USB 2.0 Specification*. 0: Not significant 1: B-device connection to not HNP capable A-device Value after MPU or USB reset is low.
6	R_WK_OK	The remote wake-up granted bit is 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 grants the function the ability to assert remote wake-up. 0: Not significant 1: Remote wake-up granted Value after MPU or USB reset is low.
5	USB_RESET	USB reset signaling is active. The USB_RESET bit returns 1 when the USB host resets the USB bus. A valid USB reset resets the endpoint FIFOS, the other control register bits, except for SYSCON1.CFG_LOCK and the associated configuration registers (0x20 to 0x3F), IRQ_EN.DS_CHG_IE and IRQ_SRC.DS_CHG, and forces the device to the default state for DEVSTAT (this register). This bit is cleared at the end of reset. This bit, as well as other DEVSTAT bits, is double-buffered. If a pending interrupt has not been handled when a USB reset occurs and is handled only when USB reset is finished, the USB device controller does not see the USB_RESET bit going high and then low. 0: Device is not being reset by USB host. 1: Device is being reset by USB host. The value after the USB device controller hardware reset is low, and during USB reset is high (low after USB reset).

Table 40. Device Status Register (DEVSTAT) (Continued)

Bit	Name	Description
4	SUS	Suspended state bit: the device in suspended state is, at a minimum, attached to the USB, powered, has been reset by the USB host, and has not seen bus activity for 5 ms. It can also have a unique address and be configured for use. Because the device is suspended, however, the host cannot use the device function. This bit returns 1 when the USB device is in a suspended state. 0: Not suspended 1: Suspended Value after MPU or USB reset is low.
3	CFG	Configured state bit: the device is attached to the USB, powered, has been reset, has a unique address, and is configured. The host can use the function provided by the device. This bit returns 1 when the USB device has been configured after a set SYSCON2.DEV_CFG = 1. This bit remains set to 1 until the device becomes deconfigured. This bit is cleared when the core receives a valid SET_CONFIGURATION request and the USB device controller sets the SYSCON2.CLR_CFG bit. During the time this bit is not set to 1, transactions that are not for control EP0 are ignored. A GET_ENDPOINT_STATUS to a non-control endpoint is stalled. 0: Not configured 1: Configured Value after MPU or USB reset is low.
2	ADD	Addressed state bit: the device is attached to the USB, powered, has been reset, and a unique device address has been assigned. This bit (ADD) returns 1 after an ET_ADDRESS standard request. This bit remains set to 1 until the device becomes de-addressed. 0: Not addressed 1: Addressed Value after MPU or USB reset is low.

Table 40. Device Status Register (DEVSTAT) (Continued)

Bit	Name	Description
1	DEF	The default state bit returns 1 when the USB device is attached to the USB and powered, and has been reset. This bit remains set to 1 until the device becomes de-powered. The device moves into default state as soon as the USB reset is effective. 0: Not in default 1: Default The value after MPU is low, and after USB reset is high.
0	ATT	The attached state bit returns 1 when the device is attached to the USB and is powered. This bit remains set to 1 until the device powers down. 0: Not attached 1: Attached The value after USB device controller hardware reset is low (unattached) or high (attached), and after USB reset is high.

This read-only register provides a status reflecting the visible device states as defined in *USB1.1 Specification*. A write to this register has no effect.

Note:

This register is double buffered. If IRQ_EN.DS_CHG_IE 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.

The values depend on whether the reset action comes from the USB device controller (MPU) or USB host. They also depend on whether the device is attached or not when the USB device controller hardware reset event occurs.

Table 41. Start of Frame Register (SOF)

Bit	Name	Description
15:13	Reserved	Reserved
12	FT_LOCK	Frame timer locked bit: the USB host sends out a start of frame (SOF) packet every millisecond. When the device does not receive a SOF packet because of a bus error, a local start of-frame that is used for ISO FIFO switch is generated. Once the core receives two valid SOF, separated by TF (time frame), it sets SOF.FT_LOCK to 1, only if TF is in frame interval IF allowed by the 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 has been received in an interval of TF since 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 locked to TF. When SOF.FT_LOCK is cleared, a local SOF is generated after the 12036 USB bit time, if no valid SOF has been received, and SOF.FT_LOCK remains 0. 0: Frame timer is not locked. 1: Frame timer is locked. The value after MPU or USB reset is low. Note: Each time the core receives a valid SOF out of the allowed interval IF, a local SOF is generated and the ISO FIFO switches.
11	TS_OK	The timestamp OK bit indicates that the timestamp in SOF.TS is valid for the current frame. It returns a 1 if a valid SOF packet was received from the USB host and a 0 otherwise. 0: Timestamp is invalid. 1: Timestamp is valid. Value after MPU or USB reset is low.
10:0	TS	The timestamp number field returns the timestamp from the last USB host-valid SOF packet. The frame number is valid if SOF.TS_OK is 1. In case of an SOF miss, this value is not updated and TS_OK is cleared. The value after MPU or USB reset is low (all 11 bits).

This read-only register provides a frame timer status for use in ISO communications. A write to this register is denied.

Table 42. Interrupt Enable Register (IRQ_EN)

Bit	Name	Description
15:8	Reserved	Reserved
7	SOF_IE	Start of frame interrupt enable
6	Reserved	Reserved
5	EPn_RX_IE	Receive endpoint n interrupt enable (non-ISO)
4	EPn_TX_IE	Transmit endpoint n interrupt enable (non-ISO)
3	DS_CHG_IE	Device state changed interrupt enable
2:1	Reserved	Reserved
0	EP0_IE	EP0 transactions interrupt enable

This read/write register enables all non-DMA interrupts (control, state changed, ISO, and non-ISO).

The values depend on whether the reset action comes from the USB device controller (MPU) or the USB host.

When the USB device controller sets a bit position to 1, an interrupt is signaled to the USB device controller if the corresponding IRQ_SRC bit is asserted to 1 by the core for any IRQ_SRC bit controlled by this bit. If reset to 0, the interrupt is masked and not signaled to the USB device controller.

- 0: Interrupt is disabled.
- 1: Interrupt is enabled.

The value after MPU or USB reset is low for all bits except IRQ_EN.DS_CHG_IE bit, which remains unchanged after a reset from the USB host.

Table 43. DMA Interrupt Enable Register (DMA_IRQ_EN)

Bit	Name	Description
15:11	Reserved	Reserved
10	TX2_DONE_IE	Transmit DMA channel 2 done interrupt enable
9	RX2_CNTt_IE	Receive DMA channel 2 transactions count interrupt enable
8	RX2_EOT_IE	Receive DMA channel 2 end of transfer interrupt enable
7	Reserved	Reserved

Table 43. DMA Interrupt Enable Register (DMA_IRQ_EN) (Continued)

Bit	Name	Description
6	TX1_DONE_IE	Transmit DMA channel 1 done interrupt enable
5	RX1_CNT_IE	Receive DMA channel 1 transactions count interrupt enable
4	RX1_EOT_IE	Receive DMA channel 1 end of transfer interrupt enable
3	Reserved	Reserved
2	TX0_DONE_IE	Transmit DMA channel 0 done interrupt enable
1	RX0_CNT_IE	Receive DMA channel 0 transactions count interrupt enable
0	RX0_EOT_IE	Receive DMA channel 0 end of transfer interrupt enable

This read/write register enables all DMA interrupts.

When the USB device controller sets a bit position to 1, an interrupt is signaled to the USB device controller if the corresponding bit in the IRQ_SRC register is asserted to 1 by the core. If reset to 0, the interrupt is masked and not signaled to the USB device controller.

- 0: Interrupt is disabled.
- 1: Interrupt is enabled.

The value after MPU or USB reset is low for all bits.

Table 44. Interrupt Source Register (IRQ_SRC)

Bit	Name	Description
15:11	Reserved	Reserved
10	TXn_DONE	The transmit DMA channel n done interrupt flag (non-ISO) bit is only for non-ISO DMA transfer. This bit is never set for ISO DMA transfer. The core automatically sets this bit when a transmit DMA channel has completed the programmed transfer by servicing the last IN transaction from the USB host. This is when TXDMAn.TXn_TSC (transfer size counter) equals 0, and the last IN transaction completes with an ACK. When this bit is asserted, the USB device controller must read the DMAN_STAT register to identify the endpoint number for which the transfer completed. The endpoint interrupt IRQ_SRC.EPn_TX is never set for the assigned endpoint to TX DMA channel n. 0: No action 1: Non-ISO transmit DMA transfer for a channel has ended. The value after MPU or USB reset is low.

Table 44. Interrupt Source Register (IRQ_SRC) (Continued)

Bit	Name	Description
9	RXn_CNT	The receive DMA channel n transactions count interrupt flag (non-ISO) bit is only for non-ISO DMA transfer. This bit is never set for ISO DMA transfer and never set if the interrupt enable bit associated with it is not set. It means that no poll operation is possible on the DMA. The core automatically sets this bit during an active Receive DMA transfer each time RXDMAn.RXn_TC equals 0 after an OUT transaction with ACK status. This bit is set after the RX DMA data have been read (end of DMA request). When this bit is asserted, the USB device controller must read DMAN_STAT register to identify the endpoint number for which the transfer completed. 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. 0: No action 1: Non-ISO receive DMA transfer for a channel has reached transactions count level. The value after MPU or USB reset is low.
8	RXn_EOT	The receive DMA channel n end of transfer interrupt flag (non-ISO) bit is for non-ISO DMA transfer only. This bit is never set for ISO DMA transfer and never set if the interrupt enable bit associated with it is not set. It means that no poll operation is possible on the DMA. The core automatically sets this bit when a receive DMA channel detects an end-of-transfer (EOT) packet during the last OUT transaction from the USB host. This bit is set after the 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 USB device controller can grant DMA transfer again to the same endpoint by enabling the FIFO again (STAT_FLG.FIFO_EN = 1). An end of transfer is detected when the core receives a data packet that is less than the configured endpoint FIFO size (or is empty), or when RXDMAn.RXn_TC equals 0 after an OUT transaction with ACK status and the RXDMAn.RXn_STOP bit is set. When this bit is asserted, the USB device controller must read DMAN_STAT.DMAn_RX_IT_SRC to identify the endpoint number for which the transfer completed, and DMAN_STAT.DMAn_RX_SB must be informed of an odd number of bytes received during the last transaction (useful for 16-bit read access from the DATA_DMA register). The endpoint interrupt IRQ_SRC.EPn_RX is never set to RX DMA channel for the assigned endpoint. 0: No action 1: Non-ISO receive DMA transfer for a channel has ended. The value after MPU or USB reset is low.

Table 44. Interrupt Source Register (IRQ_SRC) (Continued)

Bit	Name	Description
7	SOF	Start of frame interrupt flag bit: every millisecond, the USB host outputs a start-of-frame packet to the functions. The SOF bit reflects when a new SOF is received. Writing a 1 in the SOF bit location clears the flag. Writing a 0 has no effect. In accordance with the USB1.1 specification, if SOF is received corrupted or is not received, the core still sets this flag at the same rate (if bit SOF.FT_LOCK = 1) or after 12043 USB bit time (if bit SOF.FT_LOCK = 0). 0: No action 1: Start-of-frame packet received (or internal SOF) The value after MPU or USB reset is low.
6	Reserved	Reserved
5	EPn_RX	The EPn OUT transactions interrupt flag bit only concerns non-ISO endpoints. The core automatically sets the EPn_RX bit when a handshake sequence occurs for an OUT transaction to an interrupt of the bulk endpoint (NAK with SYSCON1.NAK_EN set, ACK or STALL). The USB device controller must read the EPn_STAT register to identify the endpoint causing the interrupt. 0: No action 1: OUT transaction detected on an endpoint The value after MPU or USB reset is low.
4	EPn_TX	The EPn IN transactions interrupt flag bit only concerns non-ISO endpoints. The core automatically sets this bit 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 USB device controller must read the EPn_STAT register to identify the endpoint causing the interrupt. 0: No action 1: IN transaction detected on an endpoint value after MPU or USB reset is low.
3	DS_CHG	Device state changed interrupt flag bit: the core automatically resets the DS_CHG bit when the state of the device changes. This is when the core modifies any of the bits present in the DEVSTAT register. When this bit is cleared, the background DEVSTAT register moves into foreground position. 0: No action 1: Device state change detected Value after USB device controller hardware reset is low and after USB reset is high.

Table 44. Interrupt Source Register (IRQ_SRC) (Continued)

Bit	Name	Description
2	SETUP	Setup transaction interrupt flag bit: the core automatically sets the SETUP bit when a valid setup transaction completes on control endpoint for a non-autodecoded control request and automatically clears it when the USB device controller sets EP_NUM.SETUP_SEL bit when reading setup data. A write 1 to it has no effect. 0: No action 1: Valid setup transaction occurred on endpoint 0. Value after MPU or USB reset is low.
1	EP0_RX	EP0 OUT transactions interrupt flag bit: the core automatically sets the EP0_RX bit when a handshake sequence occurs for a non-autodecoded OUT transaction to control endpoint (NAK with SYSCON1.NAK_EN set, ACK or STALL). 0: No action 1: OUT transaction on EP0 Value after MPU or USB reset is low.
0	EP0_TX	EP0 IN transactions interrupt flag bit: the core automatically sets this bit when a handshake sequence occurs for a non-autodecoded IN transaction to control endpoint (NAK with SYSCON1.NAK_EN set, ACK or STALL). 0: No action 1: IN transaction on EP0 Value after MPU or USB reset is low.

This read/clear only register identifies and clears the source of the interrupt signaled by a set flag.

The value depends on whether the reset action comes from the USB device controller (MPU) or the USB host.

The USB device controller can clear a set bit location only by writing a 1 into the bit location (except for the setup bit, which is automatically cleared by the core). A write 0 has no effect.

When the core sets a bit location to 1, an interrupt is signaled to the USB device controller if the interrupt was enabled.

- 0: No interrupt
- 1: Interrupt signaled

The value after the MPU or USB reset is low except for the IRQ_SRC.DS_CHG bit, which is high after a USB reset.

Table 45. Non-ISO Endpoint Interrupt Status Register (EPN_STAT)

Bit	Name	Description
15:12	Reserved	Reserved
11:8	EPn_RX_IT_SRC	The receive endpoint interrupt source (non-ISO) bit only concerns non-ISO endpoints. When the IRQ_SRC.EPn_RX flag is set, the endpoint causing the interrupt condition is encoded in these four register bits. When the IRQ_SRC.EPn_RX flag is cleared, the four bits read as 0. 0000: No receive endpoint interrupt is pending. 0001: EP1 … 1111: EP15 Values after MPU or USB reset are low (all four bits).
7:4	Reserved	Reserved
3:0	EPn_TX_IT_SRC	Transmit endpoint interrupt source (non-ISO) bit only concerns non-ISO endpoints.When the IRQ_SRC.EPn_TX flag is set, the endpoint that causes this flag to be set is encoded in these four register bits. When the IRQ_SRC.EPn_TX flag is cleared, the four bits read as 0. 0000: No transmit endpoint interrupt is pending. 0001: EP1 … 1111: EP15 Values after MPU or USB reset are low (all 4 bits).

This read-only register identifies the non-ISO endpoint causing an EPn interrupt. A write into it is denied.

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 46. Non-ISO DMA Interrupt Status Register (DMAN_STAT)

Bit	Name	Description
15:11	Reserved	Reserved
12	DMAn_RX_SB	The DMA receive single byte (non-ISO) bit only concerns non-ISO endpoints only (ISO endpoints receive a constant number of bytes). This bit is set when an IRQ_SRC.RXn_EOT interrupt is asserted and the core received an odd number of bytes during the last transaction. It is used to know the exact number of bytes received in case of a 16-bit read access from DATA_DMA register. When the IRQ_SRC.RXn_EOT flag is cleared, this bit reads as 0. 0: No EOT DMA interrupt is pending or the core received an even number of bytes during the last transaction. 1: An EOT DMA interrupt is pending and an odd number of bytes was received during the last transaction. Value after MPU or USB reset is low.
11:8	DMAn_RX_IT_SRC	The DMA receive interrupt source (non-ISO) bit only concerns non-ISO endpoints. When the IRQ_SRC.EPn_RX flag is set, the endpoint causing this flag to be set is encoded in these four register bits. When the IRQ_SRC.EPn_RX flag is cleared, the four bits read as 0. 0000: No receive DMA interrupt is pending. 0001: EP1 … 1111: EP15 Values after MPU or USB reset are low (all 4 bits).
7:4	Reserved	Reserved
3:0	DMAn_TX_IT_SRC	The DMA transmit interrupt source (non-ISO) bit only concerns non-ISO endpoints only. When the IRQ_SRC.EPn_TX flag is set, the endpoint that causes this flag to be set is encoded in these four register bits. When the IRQ_SRC.EPn_TX flag is cleared, the four bits read as 0. 0000: No transmit DMA interrupt is pending. 0001: EP1 … 1111: EP15 Values after MPU or USB reset are low (all 4 bits).

This read-only register identifies the endpoint that causes a DMAn interrupt. A write into it is denied.

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 47. DMA Receive Channels Configuration Register (RXDMA_CFG)

Bit	Name	Description
15:13	Reserved	Reserved
12	RX_REQ	The RX DMA request active level or pulse bit allows the RXDMA request to be configurable level or pulse-sensitive. When pulse-sensitive, the request is active during 2 cycles. 0: RX DMA request active level 1: RX DMA request active pulse Value after MPU or USB reset is low.
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. 0000: Receive DMA channel 2 is deactivated. 0001: EP1 … 1111: EP15 Values after MPU or USB reset are low (all 4 bits).

Table 47. DMA Receive Channels Configuration Register (RXDMA_CFG) (Continued)

Bit	Name	Description
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 deactivated. Any other value automatically enables receive DMA transfer for the selected endpoint. 0000: Receive DMA channel 1 is deactivated. 0001: EP1 … 1111: EP15 Values after MPU or USB reset are low (all 4 bits).
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 deactivated. Any other value automatically enables receive DMA transfer for the selected endpoint. 0000: Receive DMA channel 0 is deactivated. 0001: EP1 … 1111: EP15 Values after MPU or USB reset are low (all 4 bits).

This read/write register enables the three possible DMA receive channels and selects the endpoint number that is assigned to each of these DMA channels. An endpoint used by an RX DMA channel must have been configured (through register EPn_RX). The RXDMA_CFG register can be filled when SYSCON1.CFG_LOCK is set.

Only one channel is serviced at a time, so it is better to configure ISO endpoints on the first channel (RXDMA0_EP). In this case, the ISO endpoint is configured on channel 2,and it can never be serviced with low software and a fast USB host.

The USB device controller transmit endpoint DMA channels 0, 1, and 2 are associated with OMASP5912 controller inputs.

CAUTION

System software must not program RXDMAx_EP to endpoint numbers that are not configured as DMA endpoints.

Table 48. DMA Transmit Channels Configuration Register (TXDMA_CFG)

Bit	Name	Description
15:13	Reserved	Reserved
12	TX_REQ	The TX DMA request active level or pulse bit allows the TXDMA request to be configurable level or pulse-sensitive. When pulse-sensitive, the request is active for two cycles. 0: TX DMA request active level 1: TX DMA request active pulse Value after MPU or USB reset is low.
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 deactivated. Any other value automatically enables transmit DMA transfer for the selected endpoint. 0000: Transmit DMA channel 2 is deactivated. 0001: EP1 … 1111: EP15 Values after MPU or USB reset are low (all four bits).

Table 48. DMA Transmit Channels Configuration Register (TXDMA_CFG) (Continued)

Bit	Name	Description
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 deactivated. Any other value automatically enables transmit DMA transfer for the selected endpoint. 0000: Transmit DMA channel 1 is deactivated. 0001: EP1 … 1111: EP15 Values after MPU or USB reset are low (all four bits).
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 deactivated. Any other value automatically enables transmit DMA transfer for the selected endpoint. 0000: Transmit DMA channel 0 is deactivated. 0001: EP1 … 1111: EP15 Values after MPU or USB reset are low (all four bits).

This read/write register enables the three possible DMA transmit channels and selects the endpoint number that is assigned to each. An endpoint used by a TX DMA channel must have been configured (through register EPn_TX). TXDMA_CFG register can be filled when SYSCON1.CFG_LOCK is set.

Only one channel is serviced at a time, so it is better to configure ISO endpoints on the first channel (TXDMA0_EP). In this case, your ISO endpoint is configured on the channel 2 and it can never be serviced with low software and a fast USB host.

USB device controller transmit endpoint DMA channels 0, 1, and 2 are associated with OMASP5912 controller inputs.

Table 49. DMA FIFO Data Register (DATA_DMA)

Bit	Name	Description
15:0	DATA_DMA	DMA FIFO data. When an RX DMA request is active for a channel (only one active at a given time), this register contains the data that the core received from the USB host OUT transaction using this channel. Data can be accessed by the main DMA controller engine (read access) in response to the DMA request for the channel. When a TX DMA request is active for a channel (only one active at a given time), this register contains the data written by the main DMA controller engine (write access) in response to a DMA request for the transmit channel to be sent to the USB host during the next IN transaction. 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 USB device controller must not access this register directly; however, there is no hardware mechanism to prevent such access. No access into this register must be made out of DMA request handling.

This register is the entry point to write or to read data into/from an endpoint used in a DMA transfer through DMA channel 0, 1 or 2.

Table 50. Transmit DMA Control Register n (TXDMAn)

Bit	Name	Description
15	TXn_EOT	Transmit DMA channel n end of transfer: the TXn_EOT bit can be either 0 or 1 for BULK DMA transfer. When the USB device controller sets it to 1, this bit signals the core that the transfer size set in TXDMAn.TXn_TSC is in bytes. A TX one interrupt (IRQ_SRC.TXn_DONE) is asserted with the last IN transaction. If the number of bytes set in TXDMAn.TXn_TSC is a multiple of the endpoint 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. 0: DMA transfer size is in buffers. 1: DMA transfer size is in bytes. Value after MPU or USB reset is low.
14	TXn_START	Transmit DMA channel n start. The USB device controller sets this bit 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, unless the USB device controller clears the endpoint in the TXDMA_CFG register. A write 0 into this bit has no effect and a read to this bit always returns 0. The IRQ_SRC.TXn_DONE interrupt bit is asserted when the DMA transfer ends. 0: No action 1: DMA transfer start Always reads 0

Table 50. Transmit DMA Control Register n (TXDMAn) (Continued)

Bit	Name	Description
13:10	Reserved	Reserved
9:0	TXn_TSC	Transmit DMA channel n transfer size counter. The binary encoded value from 0 to 1023, which the USB device controller writes into this register, corresponds to the number of bytes or number of buffer transfers (function of TXDMAn.TXn_EOT) to be transmitted by the transmit DMA channel n. When read, the register reflects the number of bytes/buffers the USB device has still to transmit. Read mode is only provided for software debug purposes. Caution: For ISO transfer, the user must verify that the set value does not exceed the ISO FIFO size for the endpoint. There is no hardware mechanism to prevent this situation. If this situation occurs, results are unpredictable. Caution: 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, 0001, 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. Values after MPU or USB reset are low (all 10 bits).

This read/write register controls the operation of the transmit DMA channel n (n = 0, 1, 2).

Table 51. Receive DMA Control Register n (RXDMAn)

Bit	Name	Description
15	RXn_STOP	Receive DMA channel n transfer stop. When this bit is set, an RXn_EOT interrupt is asserted to the USB device controller 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 USB device controller expects a given amount of data for the transfer. Caution. At end of transfer, the DMA channel is disabled and all OUT transactions received to the assigned endpoint are sent NAK by the core. The USB device controller must set CTRL.SET_FIFO_EN for the endpoint to reenable the channel. Values after MPU or USB reset are low.
14:8	Reserved	Reserved
7:0	RXn_TC	Receive DMA channel n transactions count. The USB device controller 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 transaction 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 purposes. Values after MPU or USB reset are low (all 8 bits).

This read/write register monitors incoming OUT transactions during DMA transfer on channel n (n=0,1,2).

Table 52. Endpoint 0 Configuration Register (EP0)

Bit	Name	Description
15:14	Reserved	Reserved
13:12	EP0_SIZE	Endpoint 0 FIFO size. This field contains the endpoint 0 FIFO size value and must match the value sent by the USB device controller to the USB host during the GET_DEVICE_DESCRIPTOR request preceding configuration phase. Status flags (STAT_FLG.NON_ISO_FIFO_EMPTY and STAT_FLG.NON_ISO_FIFO_FULL) and overrun and underrun conditions are based on this value for all IN and OUT transactions to endpoint 0. The USB device controller must fill this field before setting SYSCON1.CFG_LOCK. 00: 8 bytes 01: 16 bytes 10: 32 bytes 11: 64 bytes Values after USB device controller hardware reset or USB reset are unchanged (which means that value is unknown until first write access).
11	Reserved	Reserved
10:0	EP0_PTR	Endpoint 0 pointer. This field contains the address of the endpoint 0 pointer. Value 0x000 is not permitted (reserved for setup FIFO). 0x000: address = BASE (not permitted) 0x001: address = BASE + 8 bytes 0x002: address = BASE + 16 bytes 0x003: address = BASE + 24 bytes ... 0x0FF: address = BASE + 2040 bytes Values after USB device controller hardware reset or USB reset are unchanged (which means that values are unknown until first write access). Note: The 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 results in memory overlap.

This read/write register gives the device configuration for control endpoint 0.

Values depend on whether the reset action comes from USB device controller (MPU) or the USB host.

Table 53. Receive Endpoint n Configuration Register (EPn_RX)

Bit	Name	Description
15	EPn_RX_VALID	The receive endpoint n valid bit must be set by the USB device controller 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. 0: Receive endpoint n does not exist for this configuration. 1: Receive endpoint n is part of the device configuration. Values after USB device controller hardware reset are low, after USB reset is unchanged.
14	EPn_RX_SIZE/DB	Receive non-ISO (or ISO) endpoint n double-buffer (DB).This bit is only for non-ISO endpoints. For ISO endpoints, which are always double-buffered, this bit is endpoint size MSB. This bit must be set by the USB device controller to allow double buffering for receive non-ISO endpoint n. This is used to reduce number of transactions resulting in NAK handshake. 0: No double buffer for non-ISO receive endpoint n. 1: Double buffer used for non-ISO receive endpoint n. Values after USB device controller hardware reset or USB reset are unchanged (which means that values are unknown until first write access).

Table 53. Receive Endpoint n Configuration Register (EPn_RX) (Continued)

Bit	Name	Description
13:12	EPn_RX_SIZE	The 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, and STAT_FLG.ISO_FIFO_FULL) and overrun and underrun conditions are based on this value for all OUT transactions to endpoint n. Non-ISO: 00: 8 bytes ISO: 000: 8 bytes [13:12] 01: 16 bytes [14:12] 001: 16 bytes 10: 32 bytes 010: 32 bytes 11: 64 bytes 011: 64 bytes 100: 128 bytes 101: 256 bytes 110: 512 bytes 111: 1023 bytes Warning: For ISO endpoints, a size of 1023 bytes takes up the whole memory and prevents it from being programmed with a 2K-byte USB device controller. Values after USB device controller hardware reset or USB reset are unchanged (which means that values are unknown until first write access).

Table 53. Receive Endpoint n Configuration Register (EPn_RX) (Continued)

Bit	Name	Description
11	EPn_RX_ISO	The 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). 0: Receive endpoint n type is isochronous. 1: Receive endpoint n type is bulk or interrupt. Values after USB device controller hardware reset or USB reset are unchanged.
10:0	EPn_RX_PTR	The receive endpoint n pointer field contains the address of the receive endpoint n pointer. Value 0x000 is not permitted (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. 0x000: address = BASE (not permitted) 0x001: address = BASE + 8 bytes 0x002: address = BASE + 16 bytes 0x003: address = BASE + 24 bytes ... 0x0FF: address = BASE + 2040 bytes Values after USB device controller hardware reset or USB reset are unchanged (which means that values are unknown until first write access). 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 results in memory overlap.

This read/write register gives the device configuration for non-control receive endpoint n (n: 115). The endpoints size fields must match values sent by the USB device controller to the USB host in response to the GET_CONFIGURATION_DESCRIPTOR during configuration phase.

The USB device controller must fill this field before setting SYSCON1.CFG_LOCK and must not change the values once SYSCON1.CFG_LOCK is set.

Values depend on whether the reset action comes from USB device controller (MPU) or USB host.

Table 54. Transmit Endpoint n Configuration Register (EPn_TX)

Bit	Name	Description
15	EPn_TX_VALID	The transmit endpoint n valid bit must be set by the USB device controller 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. 0: Transmit endpoint n does not exist for this configuration. 1: Transmit endpoint n is part of the device configuration. Values after USB device controller hardware reset are low, after USB reset is unchanged.
14	EPn_TX_SIZE/Db	Transmit non-ISO (or ISO) endpoint n double buffer. This bit is only for non-ISO endpoints used in DMA mode. For ISO endpoints, which are always double buffered, this bit is endpoint size MSB. For non-ISO endpoints that are not used in a DMA transfer, double buffering is not provided.This bit must be set by the USB device controller to allow double buffering for transmit non-ISO endpoint n, when used in a DMA transfer. This is used to reduce the number of transactions resulting in NAK handshake. 0: No double buffer for non-ISO transmit endpoint n. 1: Double buffer used for non-ISO transmit endpoint n. Values after MPU or USB reset is unchanged. Or transmit ISO endpoint n size[2]
13:12	EPn_TX_SIZE	Transmit endpoint n size

Table 54. Transmit Endpoint n Configuration Register (EPn_TX) (Continued)

Bit	Name	Description
11	EPn_TX_ISO	The 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). 0: Transmit endpoint n type is isochronous. 1: Transmit endpoint n type is bulk or interrupt. Values after MPU or USB reset is unchanged.
10:0	EPn_TX_PTR	The transmit endpoint n pointer field 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. 0x000: address = BASE 0x001: address = BASE + 8 bytes 0x002: address = BASE + 16 bytes 0x003: address = BASE + 24 bytes ... 0x0FF: address = BASE + 2040 bytes Values after USB device controller hardware reset or USB reset are unchanged (which means that values are unknown until first write access). 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 results in memory overlap.

This read/write register gives the device configuration for non-control transmit endpoint n (n: 115). The endpoints size fields must match the values sent by the USB device controller to the USB host in response to the GET_CONFIGURATION_DESCRIPTOR during configuration phase.

Note:

The USB device controller must fill this field before setting SYSCON1.CFG_LOCK and must not change the values once SYSCON1.CFG_LOCK is set.

3.1.1 EPn_TX[14:12].EPn_TX_SIZE: Transmit Endpoint n Size

EPn_TX.[14] bit description only applies for ISO endpoints.

This field 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.

Non-ISO: [13:12]	00:	8 bytes	ISO: [14:12]	000:	8 bytes
	01:	16 bytes		001:	16 bytes
	10:	32 bytes		010:	32 bytes
	11:	64 bytes		011:	64 bytes
				100:	128 bytes
				101:	256 bytes
				110:	512 bytes
				111:	1023 bytes

ISO Endpoints

For ISO endpoints, a size of 1023 bytes takes up the whole memory and prevents programming with a 2K-byte USB device controller.

WARNING

Values after USB device controller hardware reset or USB reset are unchanged (value is unknown until first write access).

3.2 USB Device Transactions

There is an interrupt to the MPU at the end of an USB transaction if the MPU has actions to perform. Isochronous transactions are an exception because isochronous interrupt information is available at start of frame interrupts. The MPU ISR code determines which endpoint and direction caused the interrupt and acts appropriately. The following subsections describe in detail the activities surrounding USB transactions that are not part of a DMA transfer. Cases where a transaction occurs before the previous one has been handled by the MPU are not taken into account in this part. The information is organized so that each section deals with one type and direction of transaction, such as:

- ❏ Non-isochronous, non-setup OUT transactions
- ❏ Non-isochronous IN transactions
- ❏ Isochronous OUT transactions
- ❏ Isochronous IN transactions

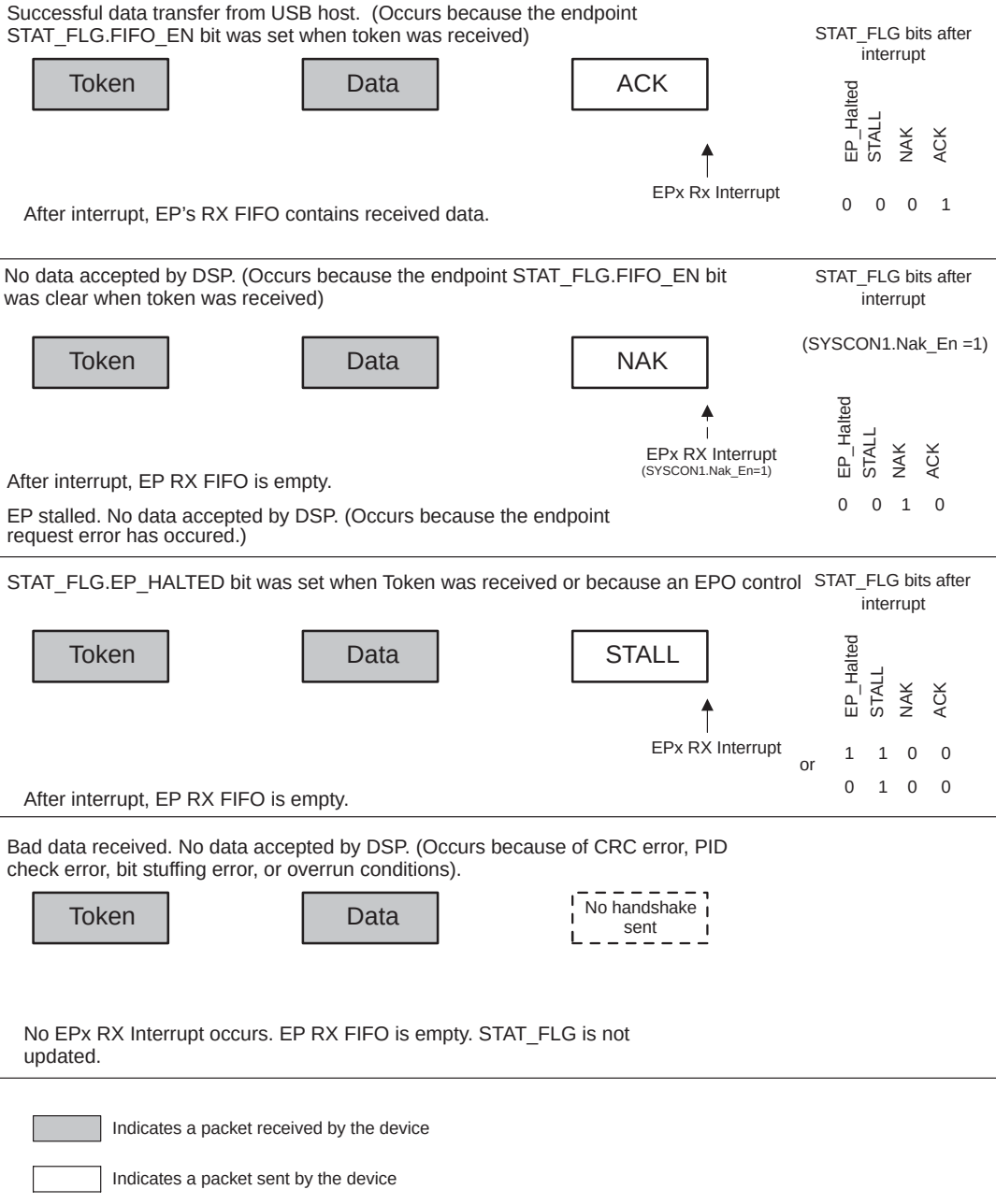
This allows each section to focus only on a specific style of transaction without adding in the confusion of special cases related to other styles.

3.3 Non-Isochronous, Non-Setup OUT (USB HOST -> MPU) Transactions

Non-isochronous, non-setup OUT transactions refer to USB transactions where data is moved from the USB host to the MPU, 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 non-setup transactions on control endpoints.

Figure 4 shows the various USB protocol conditions that can occur during non-isochronous, non-setup OUT transactions. The diagram 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 (good data received), NAK (device not ready to receive data), and STALL (device in a condition where the endpoint cannot handle OUT transactions). The last case is an abnormal instance where the token packet or the data packet was received with errors. The RX FIFO only contains valid receive data under the first (acknowledge) case.

Figure 4. Non-Isochronous, Non-Control OUT Transaction Phases and Interrupts



3.3.1 Non-Isochronous, Non-Control OUT Endpoint Handshaking Conditions

An endpoint CTRL.SET_FIFO_EN bit provides the main control for the endpoint ability to allow successful OUT transaction data reception for the endpoint. If at the beginning of an OUT transaction to an endpoint, the endpoint STAT_FLG.FIFO_EN bit is 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 (this is the top case shown in Figure 4). If, however, the endpoint STAT_FLG.FIFO_EN bit was 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 (the second case shown in Figure 4).

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 RXFSTAT.RXF_COUNT value before reading data from the RX FIFO.

At the end of an USB OUT transaction to an endpoint where the data is accepted (ACKed), the hardware clears the endpoint STAT_FLG.FIFO_EN bit. Once the MPU software has dealt with the OUT transaction data in the endpoint RX FIFO, it must re-enable the endpoint OUT transaction reception by setting the endpoint CTRL.SET_FIFO_EN bit. MPU software can use the endpoint CTRL.SET_FIFO_EN bit as a receive flow control mechanism.

Acknowledged Transactions (ACK)

At completion of an OUT transaction to an endpoint, the USB module issues an endpoint-specific interrupt to the MPU and STAT_FLG is updated. In response to the endpoint interrupt, the MPU must read EPN_STAT register to identify the endpoint causing the interrupt, then write a 1 to the interrupt bit to clear it. The MPU must then set EP_NUM.EP_NUM to the endpoint number and EP_NUM.EP_SEL to 1, and then read the endpoint status from STAT_FLG. STAT_FLG.ACK is set to indicate that the endpoint received a transaction to which the USB module signaled ACK handshaking.

If STAT_FLG.FIFO_EMPTY is cleared, the transaction sent 1 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 knows the number of bytes to read from RX FIFO by reading RXFSTAT.RXF_COUNT value. The MPU can then read RX data from DATA register. Once the MPU has read the data from the FIFO, it sets the CTRL.SET_FIFO_EN bit to allow the next USB OUT

transaction to the endpoint to be placed into the RX FIFO, and then clears the EP_NUM.EP_SEL bit. This clears the STAT_FLG.ACK bit for this endpoint to allow next transaction status to be written into the STAT_FLG register.

Non-Acknowledged Transactions (NAK)

The device can be configured via the SYSCON1.NAK_EN either to inform the MPU of a NAKed transaction or not. If SYSCON1.NAK_EN is cleared, no interrupt is asserted to the MPU if an OUT transaction completes with a NAK handshake and STAT_FLG.NAK bit is not set. If SYSCON1.NAK_EN is set, the USB module issues an endpoint-specific interrupt to the MPU at completion of an OUT transaction to an endpoint, and STAT_FLG.NAK bit is set. In response to the endpoint interrupt, the MPU must read EPN_STAT register to identify the endpoint causing the interrupt, and then write a 1 to the interrupt bit to clear it. The MPU must then set EP_NUM.EP_NUM to the endpoint number and EP_NUM.EP_SEL to 1, and then read the endpoint status from STAT_FLG. STAT_FLG.NAK is set to indicate that the endpoint received a transaction to which the USB module signaled NAK handshaking.

The MPU must set the CTRL.SET_FIFO_EN bit to allow the next USB OUT transaction to the endpoint to be placed into the RX FIFO, and then clear the EP_NUM.EP_SEL bit. This clears the STAT_FLG.NAK bit for this endpoint to allow the next transaction status to be written into the STAT_FLG register.

3.3.2 Non-Isochronous, Non-Control 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 either if the endpoint STAT_FLG.EP_HALTED bit is set or if a request error occurs (control transactions only). When an endpoint OUT transaction is given a STALL handshake, the endpoint STAT_FLG.STALL bit is set and an endpoint-specific interrupt is generated for the endpoint. STAT_FLG.FIFO_EN is of lower priority than STAT_FLG.EP_HALTED; when the STAT_FLG.EP_HALTED bit is set and transactions to the RX endpoint are stalled, regardless of the STAT_FLG.FIFO_EN value. If STAT_FLG.FIFO_EN is set, the STAT_FLG.FIFO_EN bit is automatically cleared at the end of the STALLed transaction and RX FIFO is cleared.

In response to the endpoint interrupt, the MPU must read the EPN_STAT register to identify the endpoint causing the interrupt, and then write a 1 to the interrupt bit to clear it. The MPU must then set EP_NUM.EP_NUM to the endpoint number and EP_NUM.EP_SEL to 1, and then read the endpoint

status from STAT_FLG. STAT_FLG.STALL is set to indicate that the endpoint received a transaction to which the USB module signaled STALL handshaking.

If STAT_FLG.EP_HALTED has been set by the MPU and can be removed, the MPU must set CTRL.CLR_HALT to clear the condition and set CTRL.SET_FIFO_EN to allow the next USB OUT transaction to the endpoint to be placed into the RX FIFO. If STAT_FLG.EP_HALTED has been set in response to a SET_FEATURE request sent by the USB host, or if the bit is cleared (control transaction only), the MPU has no action to perform and must clear the EP_NUM.EP_SEL bit. This clears the STAT_FLG.STALL bit for this endpoint and allows the next transaction status to be written into the STAT_FLG register.

Packet Errors

In case of a receive data error 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 (the fourth case shown in Figure 4). Additionally, the endpoint RX FIFO is not filled, and STAT_FLG.FIFO_EN bit is not cleared. If the MPU 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 expected DATA PID during an OUT transaction, the module automatically returns ACK handshake to the USB host, regardless of the STAT_FLG.FIFO_EN bit (per the USB specification). Data is ignored, and no interrupt is asserted to the MPU.

This error occurs if ACK handshake from previous OUT transaction is received corrupted by the USB host.

3.3.3 Non-Isochronous, Non-Control OUT Endpoint FIFO Error Conditions

If the USB host attempts 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. Additionally, the endpoint RX FIFO is not filled, and STAT_FLG.FIFO_EN bit is not cleared.

The MPU must not read more data from RX FIFO than the value indicated by RXFSTAT.RXF_COUNT.

3.4 Non-Isochronous IN (MPU->USB HOST) Transactions

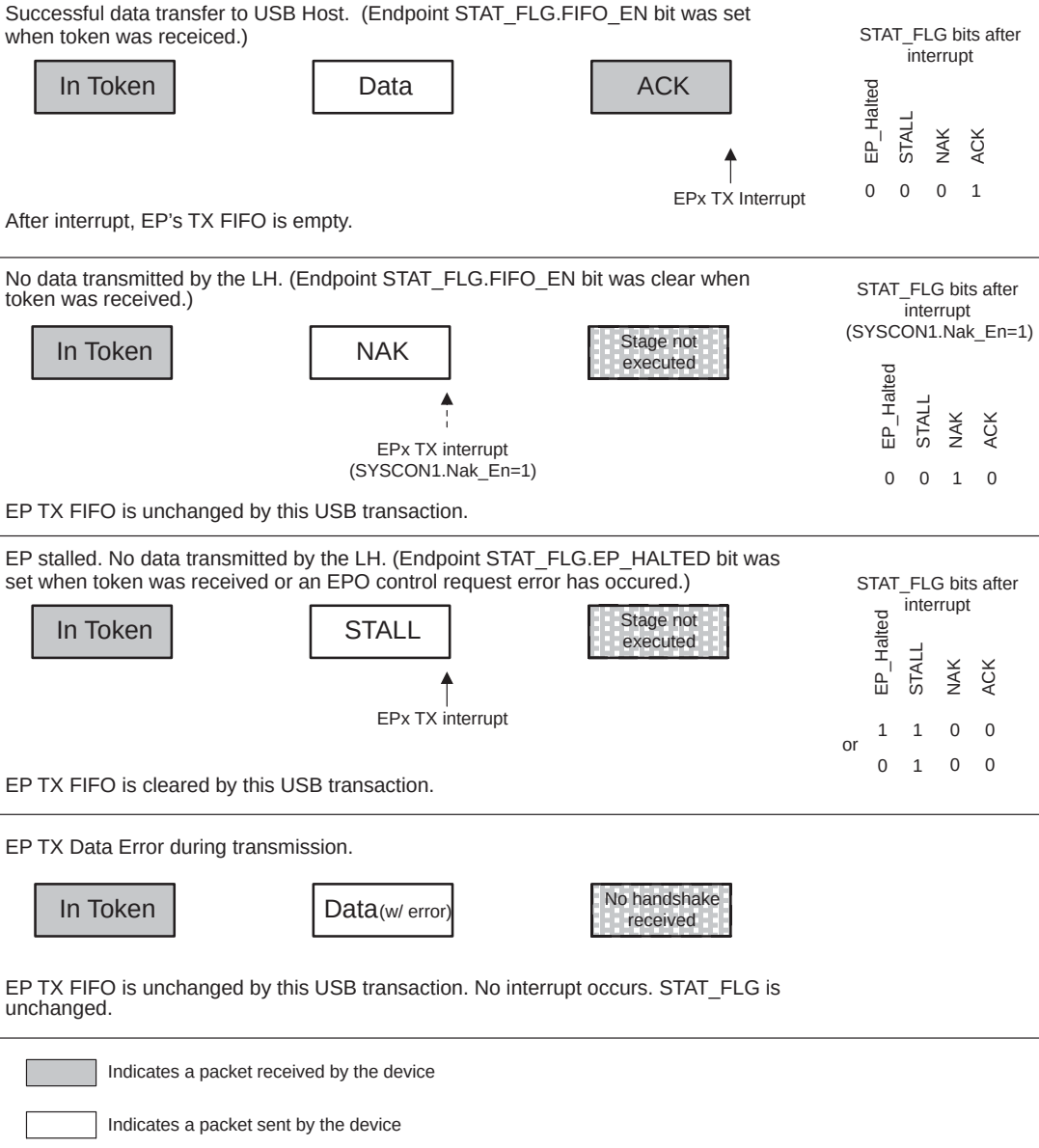
Non-isochronous IN transactions refer to USB transactions in which data is moved from the MPU to the USB host, where the USB handshaking protocols are in effect, and data transmission is ensured. These transactions are the IN transactions that occur on control, bulk, and interrupt endpoints. These transactions 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 STAT_FLG.FIFO_EN 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 STAT_FLG.FIFO_EN is set when an IN transaction to the endpoint occurs, a 0-byte data packet is sent.

Once the endpoint previous transmit activity is taken care of, the MPU code gains access to endpoint FIFO and status by setting the EP_NUM.EP_SEL bit. Then the MPU 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 has been written to the endpoint FIFO, MPU code sets the CTRL.SET_FIFO_EN bit to allow the USB to use the endpoint TX FIFO, and then clears the EP_NUM.EP_SEL bit. The data in the endpoint TX FIFO is sent to the USB host the next time an IN transaction to the endpoint occurs.

Figure 5 shows the various USB protocol conditions that can occur during non-isochronous IN transactions. It diagrams the three phases of the IN transaction, the direction of information flow for each phase, when endpoint-specific nterrupts are generated, and the resulting STAT_FLG bits for the endpoint. The top three cases show the normal USB handshaking: acknowledge (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 last 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 5. Non-Isochronous IN Transaction Phases and Interrupts



3.4.1 Non-Isochronous 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 (first case shown in Figure 5), and the second indicates that the host received a garbled data packet (last case shown in Figure 5).

Acknowledged Transactions (ACK)

When the endpoint IN transaction completes on the USB bus with an ACK handshake, the endpoint generates an endpoint-specific interrupt to the MPU (see first case in Figure 5). In response to the endpoint interrupt, the MPU must read the EPN_STAT register to identify the endpoint causing the interrupt, and then write a 1 to the interrupt bit to clear it. The MPU must then set EP_NUM.EP_NUM to the endpoint number, EP_NUM.EP_DIR to 1 (to signal an IN endpoint), and EP_NUM.EP_SEL to 1, and then read the endpoint status from STAT_FLG. STAT_FLG.ACK 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 has more data to transmit to the USB host, it must fill the TX FIFO following the process indicated above. It must then clear EP_NUM.EP_SEL bit. This clears the STAT_FLG.ACK bit for this endpoint to allow next transaction status to be written into the STAT_FLG register.

Non-Acknowledged Transactions (NAK)

For the case in which the MPU 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 via the endpoint STAT_FLG.FIFO_EN bit; when 1 it indicates that data in the TX FIFO can be sent to the USB host. When the endpoint STAT_FLG.FIFO_EN bit is 0 and an IN transaction to the endpoint occurs, NAK handshake is sent, indicating that the MPU is not ready to handle the request.

If SYSCON1.NAK_EN bit 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 is generated. If the SYSCON1.NAK_EN bit is set, when the NAK handshake is sent in the data packet portion of the transaction to the IN endpoint, the STAT_FLG.NAK bit is set and an endpoint-specific interrupt to the MPU is generated.

In response to the endpoint interrupt, the MPU must read the EPN_STAT register to identify the endpoint causing the interrupt, and then write a 1 to the interrupt bit to clear it. The MPU must then set EP_NUM.EP_NUM to the endpoint number, EP_NUM.EP_DIR to 1 (to signal an IN endpoint), and EP_NUM.EP_SEL to 1, and then read the endpoint status from STAT_FLG. STAT_FLG.NAK is set to indicate that the endpoint sent a NAK handshake to the USB host. If the MPU has data to transmit to the USB host, it must fill the TX FIFO following the process indicated above. The MPU must then clear the

EP_NUM.EP_SEL bit. That clears the STAT_FLG.NAK bit 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 still retains control of the FIFO).

Signaling NAK handshake for several endpoint transactions in a row can cause the PC host to discard the transaction, so NAK is not necessarily a good mechanism in cases where the MPU is not able to service a request for long periods of time.

3.4.2 Non-Isochronous 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 STAT_FLG.EP_HALTED flag is set, or if a request error occurs (control transaction only). A USB STALL handshake indicates that the device endpoint is in a condition in which it is not able to transfer data and instructs the USB host not to retry the transaction. The device typically requires intervention via some other mechanism to clear the condition, usually a control transfer via endpoint 0. The MPU can set the endpoint EP_HALTED bit by selecting the endpoint by writing the appropriate value in EP_NUM register, and then setting the endpoint CTRL.Set_HALT bit, and clear it by selecting the endpoint, and then setting the endpoint CTRL.Clr_HALT bit. When the endpoint EP_HALTED bit 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 STAT_FLG.STALL bit is set, and an endpoint-specific
interrupt to the MPU is generated.

In response to the endpoint interrupt, the MPU must read the EPN_STAT register to identify the endpoint causing the interrupt, and then write a 1 to the interrupt bit to clear it. The MPU must then set EP_NUM.EP_NUM to the endpoint number, EP_NUM.EP_DIR to 1 (to signal an IN endpoint), and EP_NUM.EP_SEL to 1, and then read the endpoint status from STAT_FLG. STAT_FLG.STALL is set to indicate that the endpoint sent a STALL handshake to the USB host. The MPU must then clear EP_NUM.EP_SEL bit. This clears the STAT_FLG.STALL bit 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; so for a given endpoint number, the TX can be halted when the RX is not.

Packet Errors

If an error (CRC, bit stuffing, or PID check) occurs during the token packet of a USB IN transaction to a non-isochronous endpoint, the USB block ignores the transaction. No endpoint-specific interrupt to the MPU occurs for transactions with corrupted packets. If the MPU 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 module detects after a time-out that an error has occurred. The data to transmit is still in the TX FIFO to be re-sent during next IN transaction, STAT_FLG.FIFO_EN is not cleared, and no interrupt is asserted to the MPU.

3.4.3 Non-Isochronous IN Endpoint FIFO Error Conditions

The MPU cannot write more data into the TX FIFO than the configured FIFO size.

3.5 Isochronous OUT (USB HOST-> MPU) Transactions

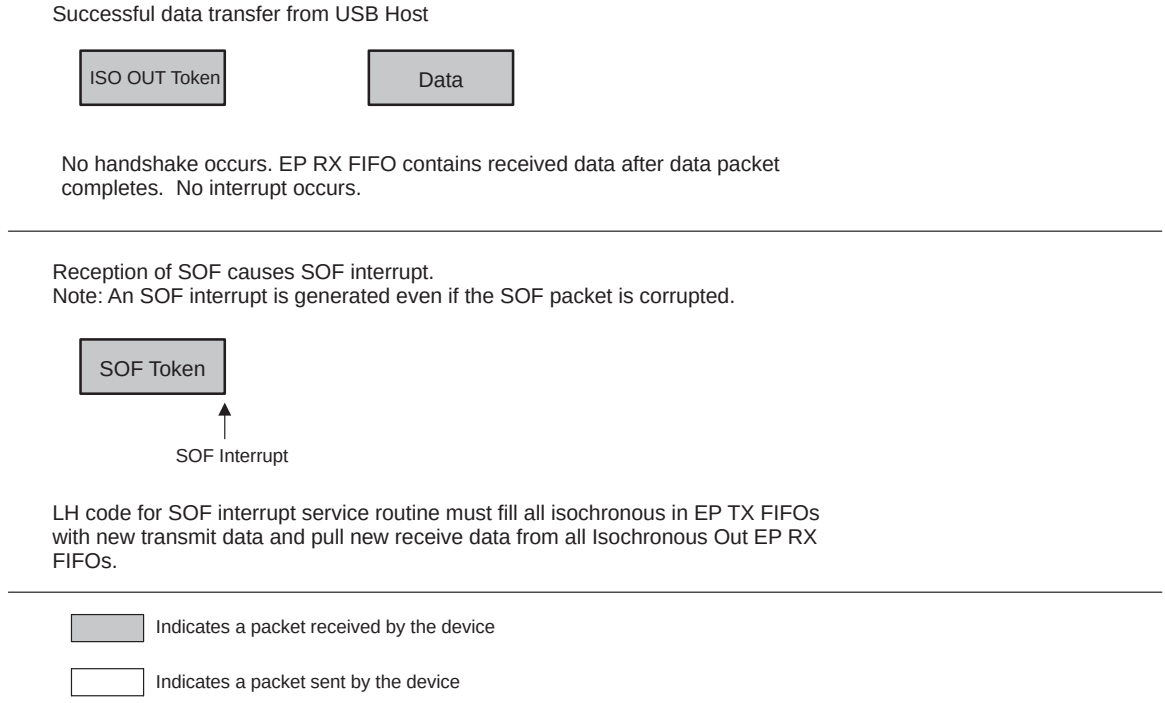
Isochronous OUT transactions are USB transactions in which a given amount of data is transferred from the USB host to the USB device controller module device every 1-ms USB frame. No USB handshaking is provided, and no endpoint-specific interrupt to the MPU is generated at completion of an isochronous OUT transaction. The MPU is responsible for handling isochronous OUT data at each start of frame (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 STAT_FLG.ISO_FIFO_EMPTY. If the RX FIFO contains data, code must read the RXFSTAT.RXF_COUNT value (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_NUM.EP_SEL bit.

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 is allowed to read to the foreground RX FIFO. The designations foreground and background are swapped at each start of frame (SOF). Isochronous endpoint FIFOs in the background are always enabled to the USB, whereas the foreground FIFOs are enabled to the MPU.

Figure 6 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 is generated for the isochronous OUT transaction. The data for isochronous endpoints are instead handled by the MPU at each start of frame (SOF) interrupt, which is shown as the second case in Figure 6.

Figure 6. Isochronous OUT Transaction Phases and Interrupts



3.5.1 Isochronous OUT Endpoint Handshaking

Because isochronous endpoint transactions have no handshake packets, the STAT_FLG.STALL, STAT_FLG.NAK, and STAT_FLG.ACK bits for isochronous endpoints always return 0. Because there is no handshake, the endpoint-specific interrupt for isochronous endpoints is not used.

3.5.2 Isochronous OUT Transaction Error Conditions

If the MPU 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 STAT_FLG.DATA_FLUSH bit 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 is 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 data error (CRC, PID check, or bit stuffing), the RX FIFO is empty at the next SOF interrupt, and the STAT_FLG.ISO_ERR bit is asserted for the duration of the next frame.

3.5.3 Isochronous OUT Endpoint FIFO Error Conditions

The MPU must never read more data than the value given by RXFSTAT.RXF_COUNT.

If the USB host sends more data than the FIFO can contain, the FIFO is cleared and the STAT_FLG.ISO_ERR 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 CTRL.CLR_EP bit is set.

3.6 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 used as the source of data for IN transactions to the ISO endpoint, and the foreground FIFO can be written to by the MPU. When an IN transaction to an ISO endpoint occurs, the USB module sends all data found in the endpoint background TX FIFO. The MPU is responsible for providing new data to the isochronous IN endpoint foreground TX FIFO at each start of frame interrupt.

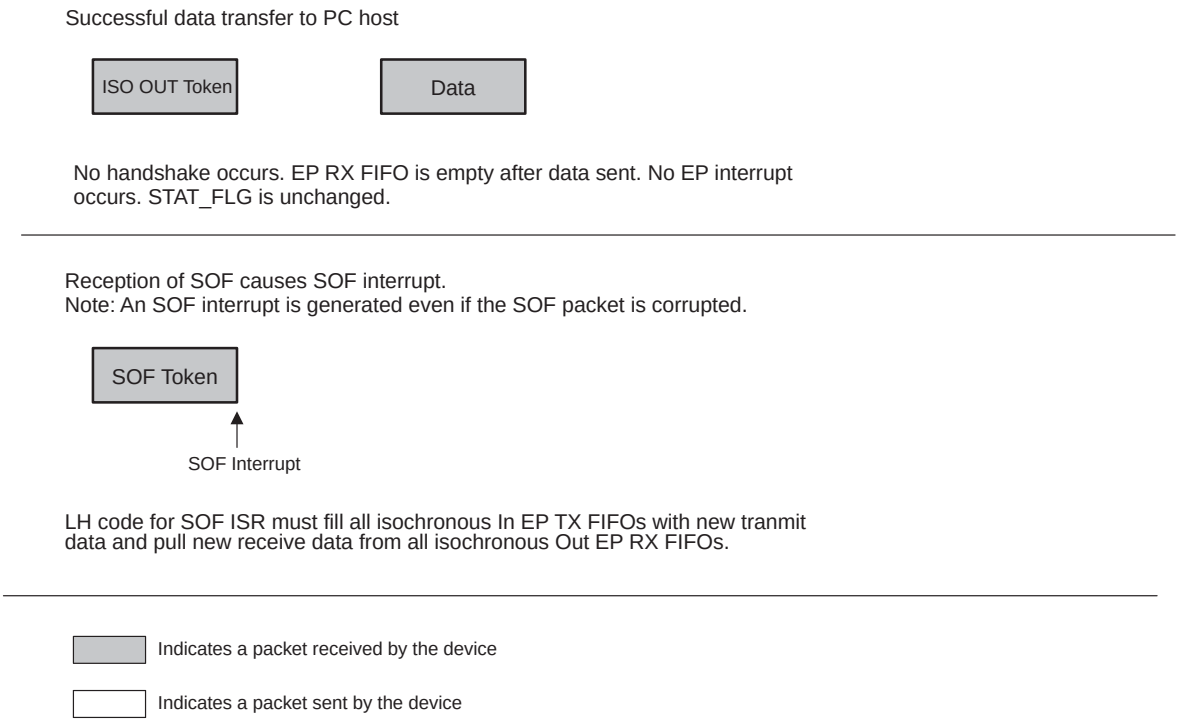
In response to the SOF interrupt, for each isochronous IN endpoint, MPU code selects the endpoint (via the EP_NUM register), and then fills the endpoint TX FIFO (via the DATA register). Once all the transmit data have been written to the FIFO, the MPU code must clear the EP_NUM.EP_SEL bit.

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 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 start of frame (SOF). Because ISO endpoints implement double-buffering, ISO endpoints do not control access to the FIFOs via a CTRL.SET_FIFO_EN bit; the CTRL.SET_FIFO_EN and the STAT_FLG.FIFO_EN bits are not implemented for ISO IN endpoints.

Figure 7 shows the transaction phases associated with isochronous IN transactions and the SOF transaction. No endpoint-specific interrupt to the MPU 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; it is assumed that the MPU refills the isochronous IN endpoint transmit FIFO at each SOF interrupt.

Figure 7. Isochronous IN Transaction Phases and Interrupts



3.6.1 Isochronous IN Endpoint Handshaking

Because isochronous endpoint transactions have no handshake packets, the STAT_FLG.STALL, STAT_FLG.NAK, and STAT_FLG.ACK bits for

isochronous endpoints always return 0. Because there is no handshake, there is no endpoint-specific interrupt to the MPU to report handshake results for isochronous endpoints.

3.6.2 Isochronous IN Transaction Error Conditions

If the USB host did not successfully complete an ISO IN transaction in the previous frame, and if data were present in TX FIFO to be sent at the IN transaction, the STAT_FLG.MISS_IN bit 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.

3.6.3 Isochronous IN Endpoint FIFO Error Conditions

If the MPU attempts to overfill the configured endpoint FIFO, data written to DATA register after the TX FIFO is full is lost, but 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 does not wish to send the data it wrote, clearing the endpoint is the only mechanism to do this.

3.7 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 is capable of autodecoding some control write and control read transfers. These operations are summarized in Figure 8 and Figure 9. An IN or an OUT transaction is received out of a control request. This transaction is automatically stalled by the core.

Figure 8. Stages and Transaction Phases of Autodecoded Control Transfers

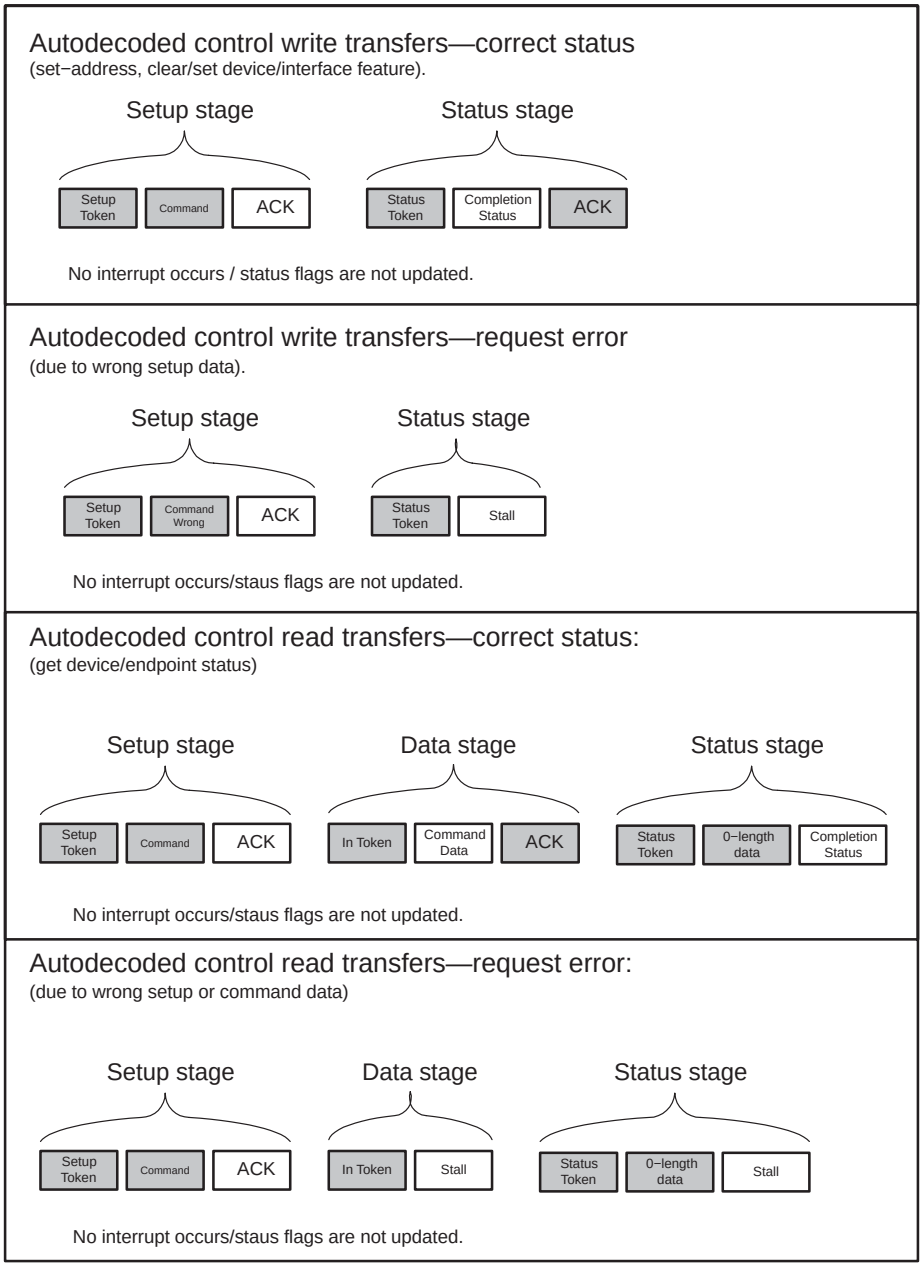
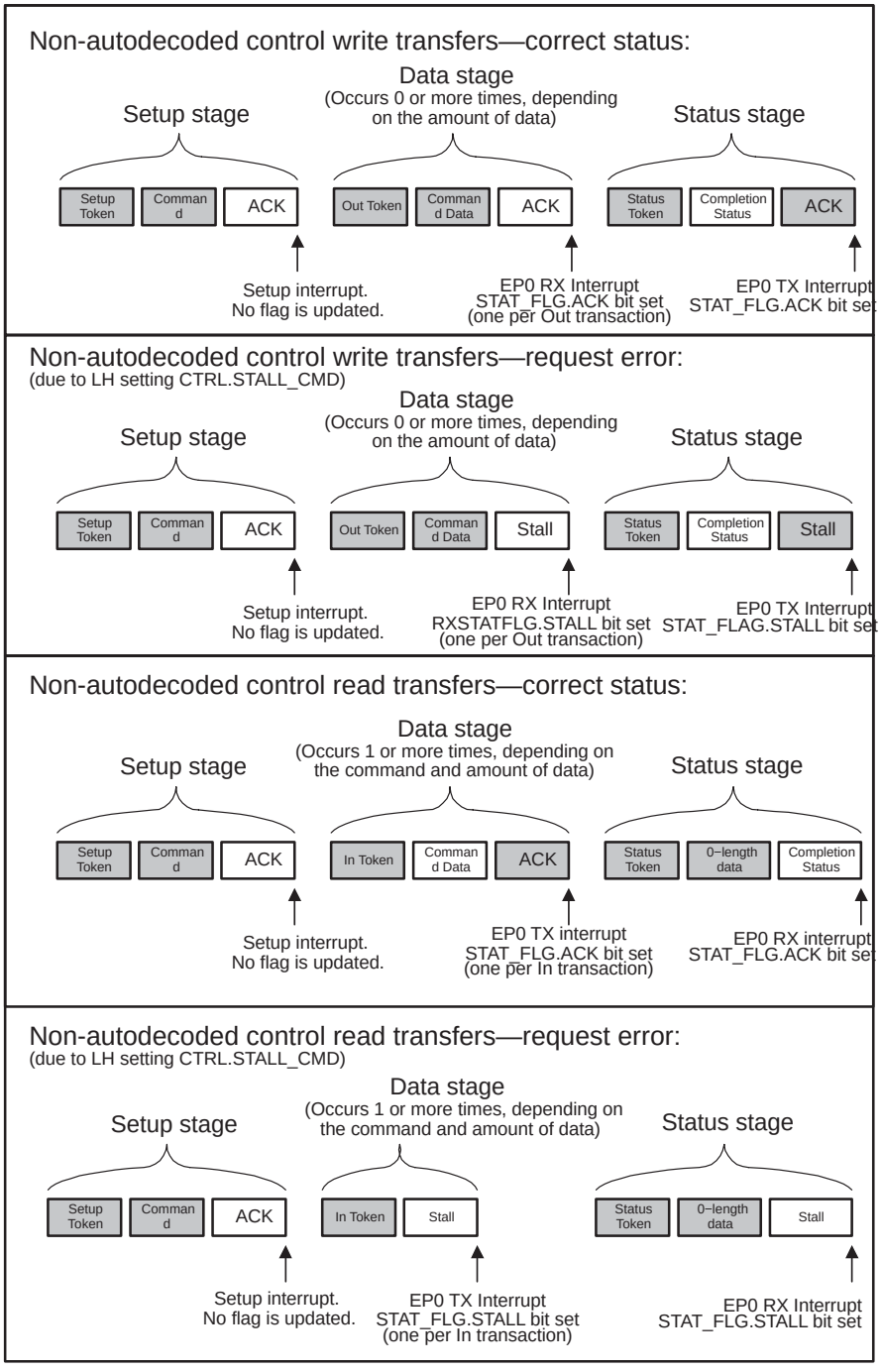


Figure 9. Stages and Transaction Phases of Non-Autodecoded Control Transfers



Non-autodecoded control read and control write transfers are sets of transactions that occur on endpoint 0 and have specific USB protocol meaning but are not 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 occurs at the end of each transaction of each stage of a control transfer. The MPU must perform the following actions to act on non-autodecoded control transfers:

- Process the setup phase setup USB interrupt. The MPU reads the control transfer command from the setup FIFO and decodes the command. For control reads, the MPU 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 only enables the endpoint 0 FIFO. MPU code also sets any flags needed for processing 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 has been 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, interpret the write data and act on the write request. After handling the last data phase interrupt, the MPU 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 provides its completion status back to the USB host during this stage, either via status in the data phase of the transaction (for control write transfers) or via 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. The USB device controller block handles all handshaking automatically and without regard to the endpoint 0 control bits that affect normal (non-control transfer) transaction handshaking. No interrupt is asserted to the MPU during autodecoded control transfers.

If no USB1.1 specification defined request is associated with the data of the setup phase, request is stalled by the core and the MPU 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 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 (if the MPU had control of the FIFO). New setup data are placed into the setup FIFO, and the setup interrupt flag is set (IRQ_SRC.setup).

In response to the setup interrupt, the MPU must select the setup FIFO by setting the EP_NUM.SETUP_SEL bit. This clears the IRQ_SRC.SETUP flag. The MPU must then read 8 bytes from the setup FIFO, clear the EP_NUM.SETUP_SEL bit, and confirm that IRQ_SRC.SETUP bit has not been reset by a new setup transaction. If the IRQ_SRC.SETUP flag is asserted, the MPU must discard the previously read data and handle the new

setup packet as explained above. Thus the MPU never misses a new occurring setup transaction (per USB 1.1 specification).

3.7.1 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 new address is different from 0, the device moves into addressed state (DEVSTAT.ADD 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 DEVSTAT.R_WK_OK bit 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).

Per the USB 1.1 specifications, 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 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, except if a corrupted packet is received, which the USB module ignores. The CTRL.SET_FIFO_EN and SYSCON2.STALL_CMD bits 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.

3.7.2 Autodecoded Control Read Transfers

Autodecoded control reads include the standard device requests GET_ENDPOINT and DEVICE_STATUS. These control read transfers access information that is kept in registers inside the USB module, so 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 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, except if a corrupted token packet is received, which the USB module ignores. The CTRL.FIFO_EN and SYSCON2.STALL_CMD bits 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. 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.

3.7.3 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 IRQ_SRC.SETUP flag set. The MPU must respond to this general USB interrupt by setting EP_NUM.SETUP_SEL bit, which clears the setup interrupt flag. The MPU must then read 8 bytes from the setup FIFO via the DATA register, clear EP_NUM.EP_SEL bit, and check the IRQ_SRC.SETUP flag. If the IRQ_SRC.SETUP flag is set, the MPU must discard the setup data it has just read and handle the new setup data packet following the same scheme. If the IRQ_SRC.SETUP flag is cleared, the MPU code interprets this request information and performs any application-specific activity needed because of the setup stage request. If there is one or more data stage for the transfer, the MPU must set the CTRL.SET_FIFO_EN bit 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 as for non-isochronous, non-control OUT endpoints applies. The MPU can cause NAK, STALL, or ACK signaling for the data stage transactions. If ACK was signaled on a given general USB interrupt, the MPU must respond by reading the data from the endpoint 0 RX FIFO and saving it for processing.

After completion of the data stage, a status stage IN transaction occurs. The USB module provides handshaking to the USB host based on the endpoint 0 handshaking control bit STAT_FLG.FIFO_EN. The MPU can delay signaling completion of the control write transfer by forcing NAK handshaking to the host during the status stage (by holding STAT_FLG.FIFO_EN 0), or causing ACK handshaking by setting the CTRL.SET_FIFO_EN bit with an empty endpoint 0 FIFO. An endpoint 0 TX general USB interrupt is sent to the MPU at completion of the status stage.

After a SET_CONFIGURATION request, the device moves in addressed or configured state as soon as the MPU sets the SYSCON2.DEV_CFG or SYSCON2.CLR_CFG bits.

Specific MPU Required Actions

If the device receives a valid set endpoint halt feature request, it must set the appropriate CTRL.SET_HALT control bit.

If the device receives a valid CLEAR_ENDPOINT halt feature request, it must set the appropriate CTRL.RESET_EP bit to clear the halt condition, FIFO flags, and reset data PID to DATA0 for the endpoint. If the specified endpoint number is 0, the MPU has only to set CTRL.CLR_HALT bit to clear the halt condition.

If the device receives a valid SET_CONFIGURATION request, it must reset all endpoints by setting the CTRL.RESET_EP control bits, set the SYSCON1.SELF_PWR bit 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 must write 1 to the SYSCON2.DEV_CFG bit to allow the device to move into the configured state (DEVSTAT.CFG bit set). If the device was configured when the SET_CONFIGURATION was received, and the new configuration value is 0, the MPU must write 1 to the SYSCON2.CLR_CFG bit to allow the device to move back into the addressed state (DEVSTAT.CFG bit cleared).

If the device receives a valid set interface request, it must reset all endpoints used by the interface set, by setting CTRL.RESET_EP control bits, and then set halt conditions for endpoints not used by this interface.

Other MPU required actions are specific to the request and not detailed in this document.

Non-Autodecoded Control Write Transfer Handshaking

Setup stage transactions that are valid are signaled ACK. Transactions with invalid setup stage token or data packets are ignored and receive no handshake packet from the USB module, and there is no interrupt generated.

Data stage handshaking for non-autodecoded control write transfers is dependent on the endpoint 0 STAT_FLG.FIFO_EN, STAT_FLG.EP_HALTED, and SYSCON2.STALL_CMD bits. The MPU can delay completion of any transaction of the data stage by signaling NAK (via CTRL.SET_FIFO_EN bit not set). 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 SYSCON2.STALL_CMD or the CTRL.SET_HALT (reflected in the STAT_FLG.EP_HALTED register bit) register bits provide this functionality. STAT_FLG.EP_HALTED does not reflect the forced STALL caused by SYSCON2.STALL_CMD; it retains its previous value.

Status stage handshaking is controlled by the endpoint 0 STAT_FLG.FIFO_EN and SYSCON2.STALL_CMD bits. Successful completion of a non-autodecoded control write transfer is indicated by the USB device controller module returning a zero 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 while dealing with the control write, which the MPU cannot deal with itself, 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 SYSCON2.STALL_CMD bit, which causes stalling of all the remaining transactions of all remaining stages of a non-autodecoded control transfer, up to the reception of the next valid SETUP command.

Error conditions are handled as for BULK/INTERRUPT transactions. If a packet is received corrupted, the core ignores the transaction and no interrupt is asserted.

3.7.4 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 USB host to USB device. At the end of the setup stage handshake, the USB module generates a MPU general USB interrupt with the IRQ_SRC.SETUP flag set. The MPU must respond to this general USB interrupt by setting the EP_NUM.SETUP_SEL bit, which clears the setup interrupt flag. The MPU must then read 8 bytes from the setup FIFO via the DATA register, clear the EP_NUM.EP_SEL bit, and check the IRQ_SRC.SETUP flag. If the IRQ_SRC.SETUP flag is set, the MPU must discard the setup data it has just read and handle the new setup data packet following the same scheme. If the IRQ_SRC.SETUP flag is cleared, the MPU code interprets this request information and then prepares data for the IN transaction that follow. This includes placing the data being requested (or the first few bytes, if more than one FIFO worth of data is being returned) into the endpoint 0 FIFO, and setting the CTRL.SET_FIFO_EN bit.

The data stage of a control read transfer consists of one or more IN transactions. Transaction handshaking and interrupt generation as for non-isochronous, non-control IN endpoints applies; the MPU 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 have been 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.

After completion of the data stage, a status stage OUT transaction occurs. The USB host sends a zero-length data packet, and the MPU code must return its completion status for the control read standard request via standard handshaking mechanisms.

Note:

In the case of returning exactly what the host requested and that request was a multiple of the maximum packet size, no zero-length packet is required. A zero-length packet is required only when the amount of data the device has to 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 is dependent on the endpoint 0 STAT_FLG.FIFO_EN, STAT_FLG.EP_HALTED, and SYSCON2.STALL_CMD bits. 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 SYSCON2.STALL_CMD and CTRL.SET_HALT (reflected through the STAT_FLG.EP_HALTED register bit) register bits provide this functionality. STAT_FLG.EP_HALTED does not reflect the forced STALL caused by SYSCON2.STALL_CMD; it retains its previous value.

The status stage is controlled by the CTRL.FIFO_EN and SYSCON2.STALL_CMD bits.

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 while dealing with the control read, which the MPU cannot deal with itself, 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 SYSCON2.STALL_CMD bit, which causes stalling of all the remaining transactions of all remaining stages of a non-autodecoded control transfer, up to the reception of the next valid SETUP command.

Error conditions are handled as for 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.

3.7.5 Autodecoded Versus Non-Autodecoded Control Requests

Table 55. 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.
	Endpoint	Autodecoded (function of AUTODEC_ DIS register bit)	None	The core automatically stalls the command if endpoint number is different from 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 spec for interface. These requests are stalled).	Command is stalled anyway.

Notes:
1) Transactions on endpoints other than zero 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 halt feature for the endpoint. This does not happen if USB host works correctly.
3) If endpoint 0 is halted, per *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 USB 1.1 when specified as such, but many device reactions are not specified by USB 1.1.
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 55. Autodecoded Versus Non-Autodecoded Control Requests (Continued)

Request	Recipient	Status	MPU Required Action	Device Behavior if Device Is Not Configured
	Endpoint	Non-autodecoded	The MPU must stall the command (via SYSCON2.STALL_CMD bit) if 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 different from 0. Addressed: device takes the new address value or moves in default state if address number is 0. Configured: request is STALLed.
GET_DESCRIPTOR	All	Non-autodecoded	The MPU must write descriptor data into endpoint 0 FIFO.	Command is passed to the MPU.

Notes: 1) Transactions on endpoints other than zero 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 halt feature for the endpoint. This does not happen if USB host works correctly.

3) If endpoint 0 is halted, per *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 USB 1.1 when specified as such, but many device reactions are not specified by USB 1.1.

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 55. Autodecoded Versus Non-Autodecoded Control Requests (Continued)

Request	Recipient	Status	MPU Required Action	Device Behavior if Device Is Not Configured
SET_ DESCRIPTOR	All	Non-autodecoded	The MPU must stall the command (via SYSCON2.STALL_ CMD bit) if it does not support set descriptor requests.	Command is passed to the MPU.
GET/SET CONFIGURA-TION	Device	Non-autodecoded	The MPU must stall the command (via 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 config nb is not 0), or set SYSCON2.CLR_CFG bit (if config nb is 0) before allowing status stage to complete. The device moves to configured state (if DEV_CFG set), or moves to addressed state (if CLR_CFG set) and a DS_CHG interrupt is asserted to the MPU.	Command is passed to the MPU.

Notes: 1) Transactions on endpoints other than zero 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 halt feature for the endpoint. This does not happen if USB host works correctly.

3) If endpoint 0 is halted, per *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 USB 1.1 when specified as such, but many device reactions are not specified by USB 1.1.

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 55. 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 (via SYSCON2.STALL_ CMD bit) if interface/setting number is not correct. If the request is SET_ INTER-FACE, 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 zero 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 halt feature for the endpoint. This does not happen if USB host works correctly.

3) If endpoint 0 is halted, per *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 USB 1.1 when specified as such, but many device reactions are not specified by USB 1.1.

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.

3.7.6 Note on 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, unexpected IN transaction is STALLed, causing STALL handshake for all remaining transactions of the transfer until 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 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. But 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 has not enabled the TX FIFO.

3.8 USB Device Initialization

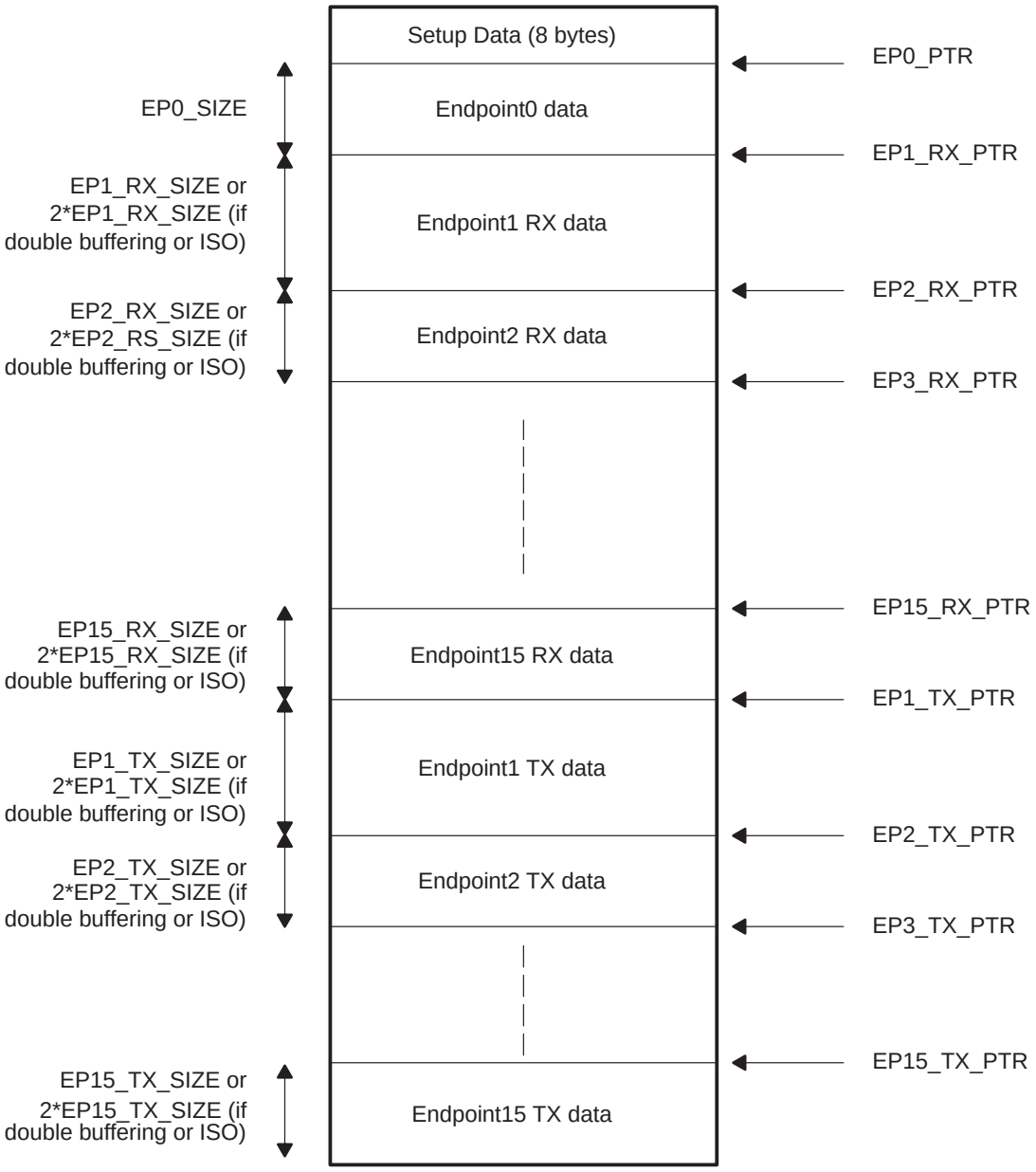
To allow communication between the device and a USB host, the MPU must configure the device by filling the configuration registers.

For each endpoint, the MPU must write on the dedicated register:

- Endpoint size
- Whether double-buffering is allowed for endpoint or not
- Endpoint type (ISO or non-ISO)
- Address of the pointer

System software must choose how to allocate the 2040 available bytes of USB device controller RAM to the USB endpoints. Receive endpoint size and type are configured using the EP1_RX through EP15_RX registers. Transmit endpoint size and type are configured using the EP1_TX through EP15_TX registers. Figure 10 shows an example of the RAM organization, obtained by following the flowchart shown in Figure 11.

Figure 10. Example of RAM Organization



Once the endpoints are configured, the MPU must set the SYSCON1.CFG_LOCK bit. If this bit is not set, all transactions are ignored by the core. Then, when the MPU is ready to communicate with the USB host, it must set the SYSCON1.PULLUP_EN bit. The MPU can wait until the DS_CHG attach interrupt has been detected and handled before setting the

SYSCON1.PULLUP_EN bit. The USB host cannot detect the device until this bit is set.

Figure 11 and Figure 12 show flowcharts for the configuration phase.

Figure 11. Device Configuration Routine

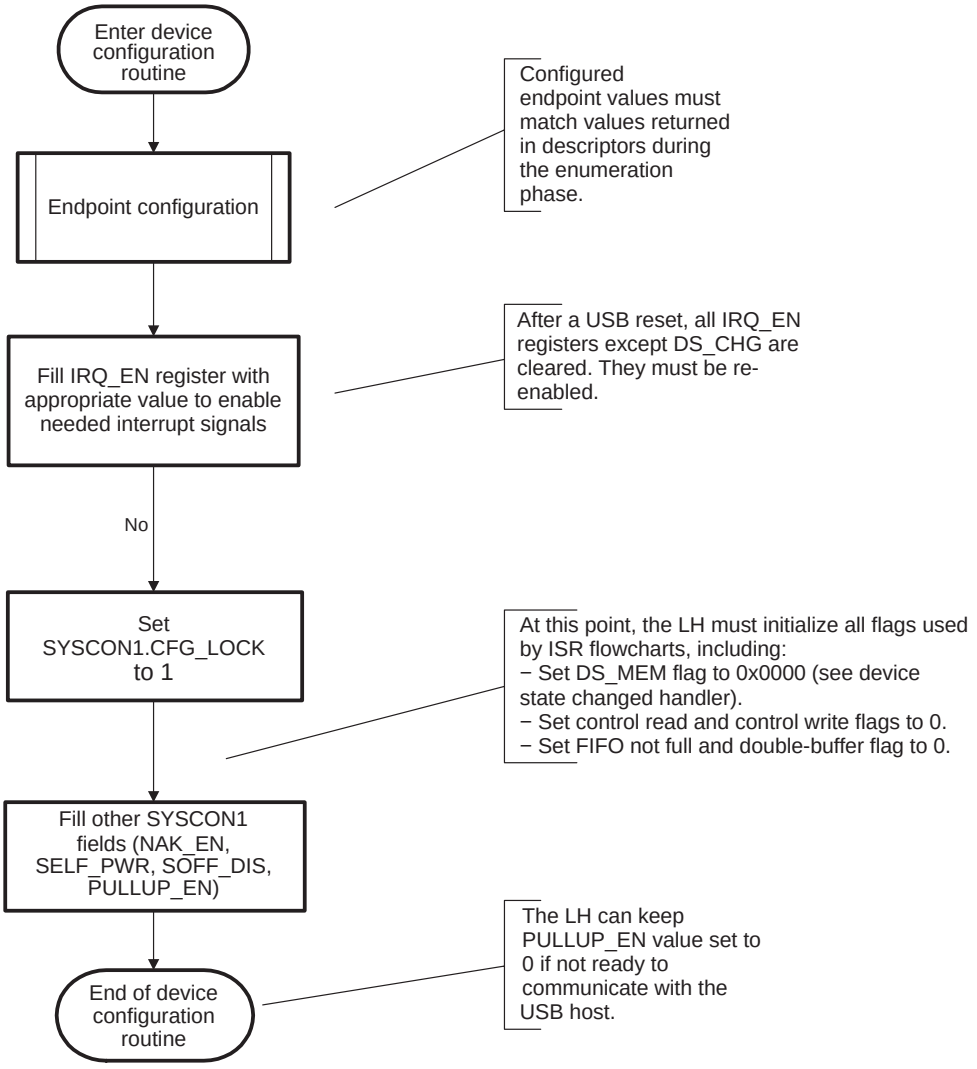
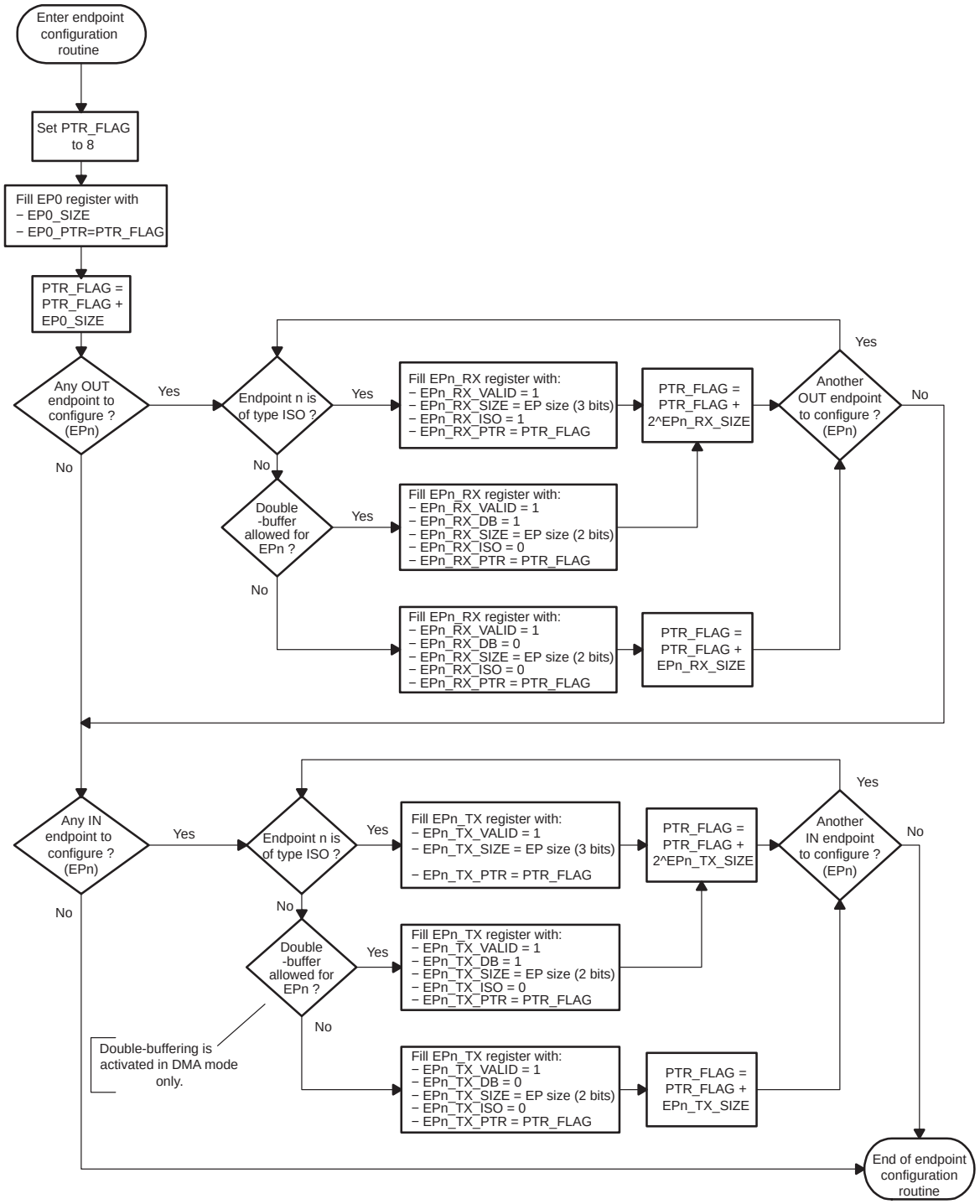


Figure 12. Endpoint Configuration Routine



3.9 Preparing for Transfers

To avoid NAK handshakes for the first transaction on an endpoint, the MPU must prepare the endpoint FIFO for receiving or transferring data. After the first transaction, the FIFO is enabled during the interrupt handling (ISR flowchart).

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 CTRL.SET_FIFO_EN bit enables both FIFOs. Therefore, it is not possible to allow a single transaction when double buffering is used.

The MPU enters the prepare for USB RX transfers routine, presented once after the enumeration phase, and then properly reacts to EP interrupts. Whether double-buffering is allowed or not is transparent to the MPU, unless both FIFOs are cleared through a CTRL.CLR_EP or CTRL.RESET_EP. In that case, and in the case where the MPU finishes to handle an interrupt without having set the CTRL.SET_FIFO_EN bit, the MPU must reenter the prepare for USB RX transfers routine.

For transmit endpoints, the MPU enters the prepare for endpoint n TX transfer routine, presented each time a new file must be transmitted from endpoint n to USB host. The MPU 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 CTRL.CLR_EP or CTRL.RESET_EP bits (see Figure 13 and Figure 14).

Note:

This does not apply to endpoint 0, which is not used before a setup interrupt occurs. At setup interrupt, the MPU reacts appropriately, and enables EP0 FIFO only if necessary.

To ensure proper usage of the module, you cannot prepare data on different endpoints at the same time. You can enter a routine once the routine is not being used by another endpoint (no parallelism); the EP_NUM register can be accessed via different routines.

Figure 13. Prepare for USB RX Transfers Routine

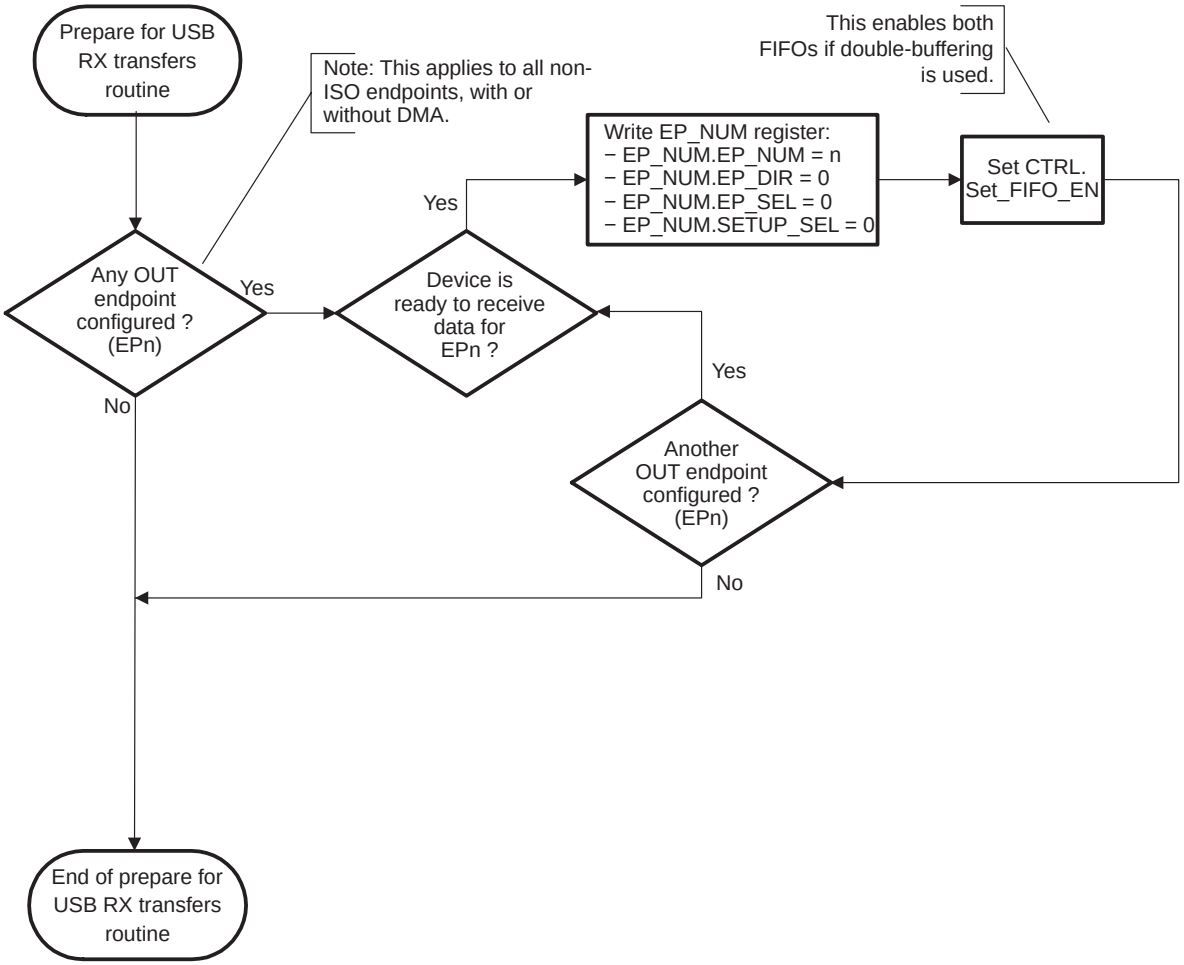
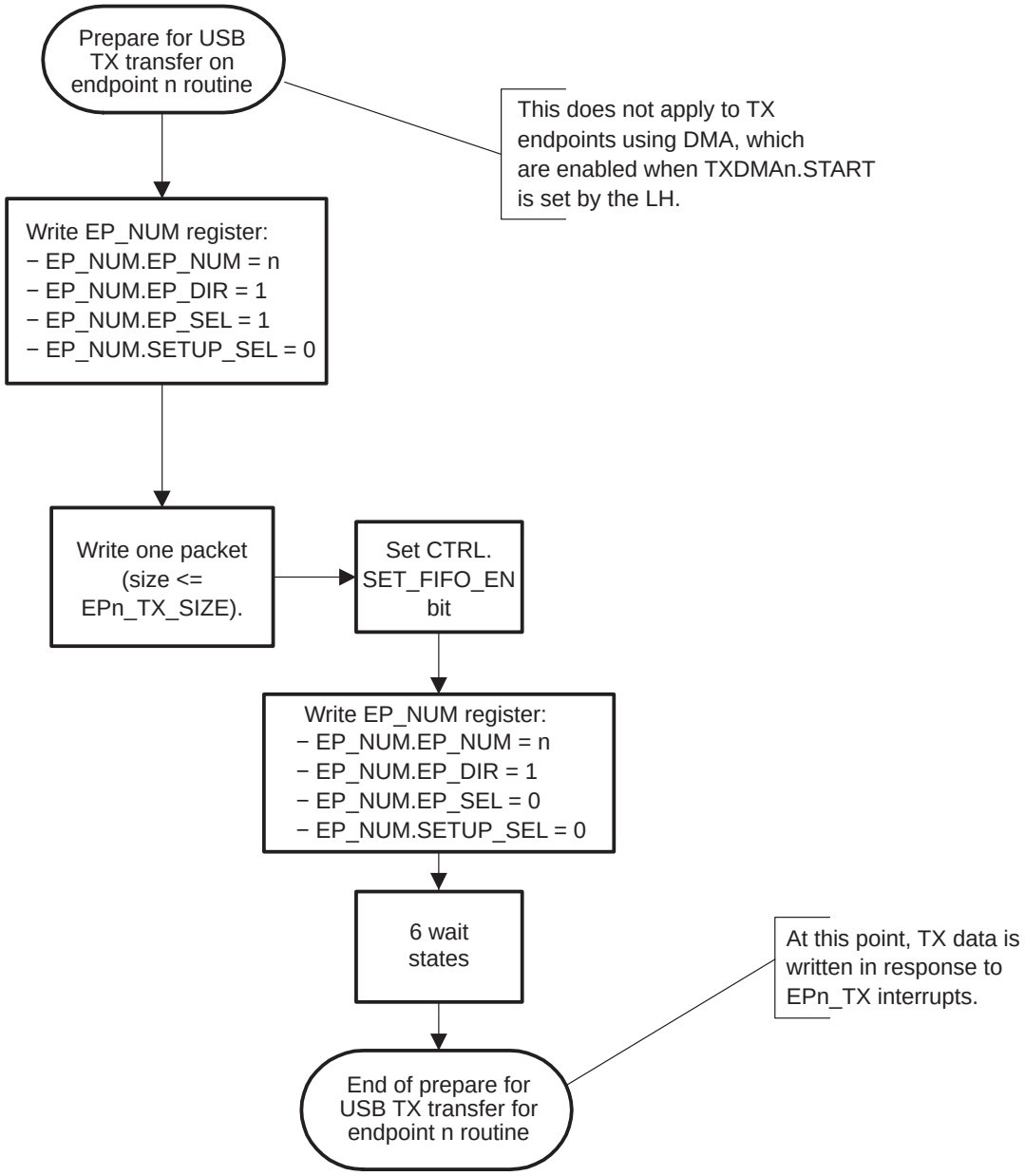


Figure 14. Prepare for TX Transfer on Endpoint n Routine



3.10 USB Device Interrupt Service Routine (ISR) Flowcharts

The flowcharts in this section give general operational guidelines for USB device ISR processing. System-architecture-specific details are left to the engineers who write the MPU and USB host code. One USB-specific interrupt register is provided (IRQ_SRC), including:

- General USB interrupts (including endpoint 0, DMA and device states interrupts) on MPU level 2 IRQ_20
- Non-ISO endpoint-specific interrupt on MPU level 2 IRQ_30
- Start of frame (SOF) interrupt for ISO transactions on MPU level 2 IRQ_29

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 non-isochronous endpoints. The SOF ISR is responsible for handling isochronous endpoints and, if needed by the application, tracking the USB frame number. Many flowcharts are presented in this chapter to provide guidelines for how to handle the interrupts related to the USB device controller module. The flowcharts in this part suppose that the SYSCON1.NAK_EN bit is cleared.

Note:

A key assumption behind the flowcharts presented here is that the application provides separate buffers for each direction of 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 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.

3.11 Important Note on USB Device Interrupts

When an endpoint interrupt is asserted, the MPU writes the EP_NUM register with the EP_NUM.EP_SEL bit set to 1. The MPU must finish the interrupt handling before clearing the EP_NUM.EP_SEL bit, 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 must not select and then unselect the endpoint without handling the interrupt, because this

clears the pending transaction status flags. The MPU does not need to set EP_NUM.EP_SEL to 1 when setting CTRL.SET_FIFO_EN, CTRL.SET_HALT, and CTRL.CLR_HALT bits.

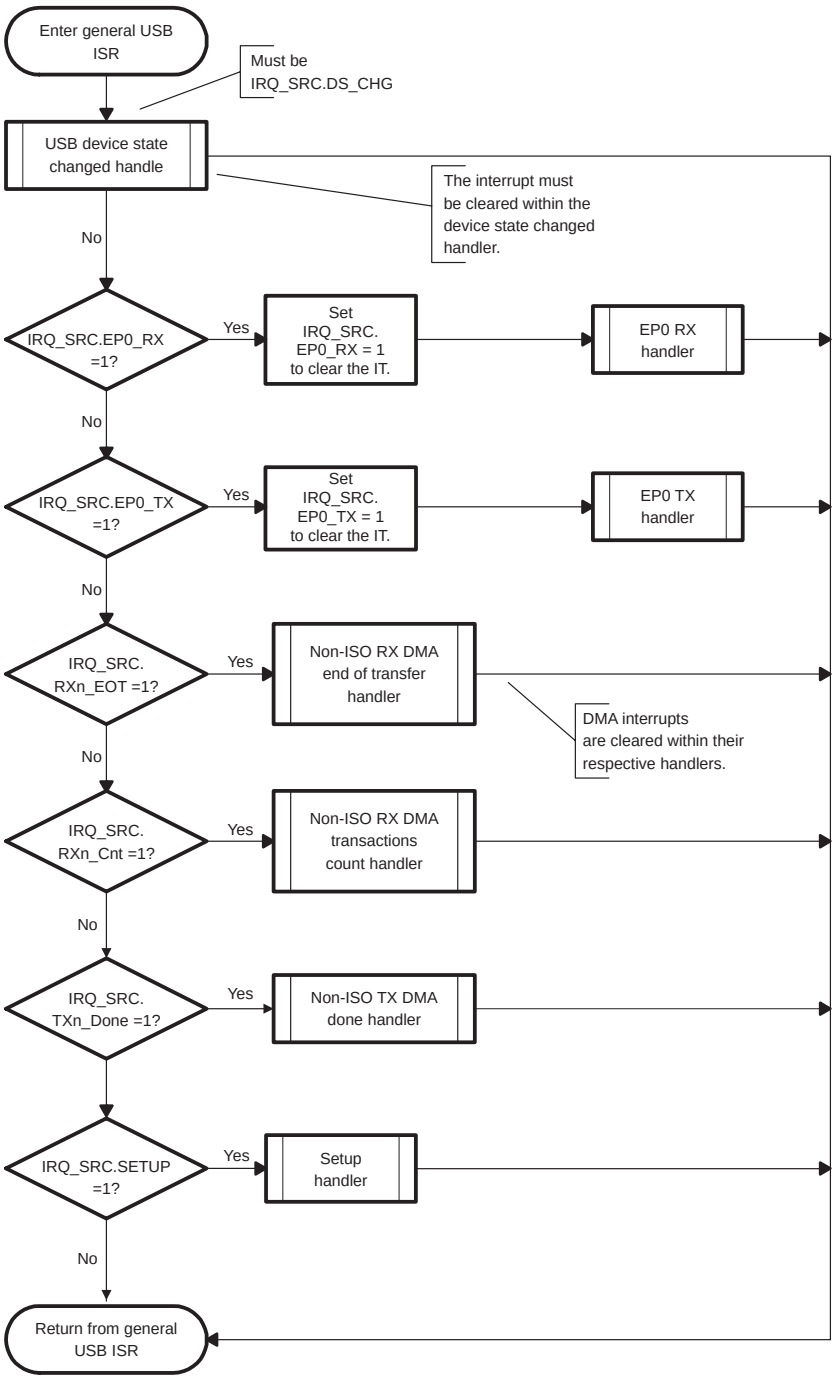
The endpoint status (STAT_FLG register) is updated at the end of each USB transaction if the previous transaction has been handled. If a pending interrupt has not been handled when a new non-transparent transaction occurs, status flags are not updated (and NAK is returned, even if FIFO was enabled, or STALLed if the EP halt feature was set), so that the MPU 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, and is not updated if there are already two interrupts pending on the endpoint.

The MPU does not need to set the SYSCON1.NAK_EN bit during normal operation. However, for debugging process, this bit can be set when the MPU finishes handling an EP interrupt without having set the corresponding CTRL.SET_FIFO_EN bit. During TX transaction, if the SYSCON1.NAK_EN bit is set, the MPU 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 was writing the TX data.

3.12 Parsing General USB Device Interrupt

The general USB interrupt (MPU level 2 IRQ_20) 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 dealt with by the ISR before returning from the ISR. Figure 15 shows an appropriate flowchart for parsing the general USB interrupts.

Figure 15. General USB Interrupt ISR Source Parsing Flowchart



3.13 Setup Interrupt Handler

A separate interrupt flag exists for setup transactions so that the MPU 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 for use in determining which USB bus activity is needed and in controlling how the MPU 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 16 is responsible for processing setup transactions occurring on endpoint 0. It calls the routine that parses the control transfer request information, shown in Figure 17 to set flags that the rest of the ISR code can use to control proper response to control transfers. Two flags are set by the setup interrupt handler, to be used during endpoint 0 interrupt handlers:

- Control read flag
- Control write flag

Figure 16. Setup Interrupt Handler

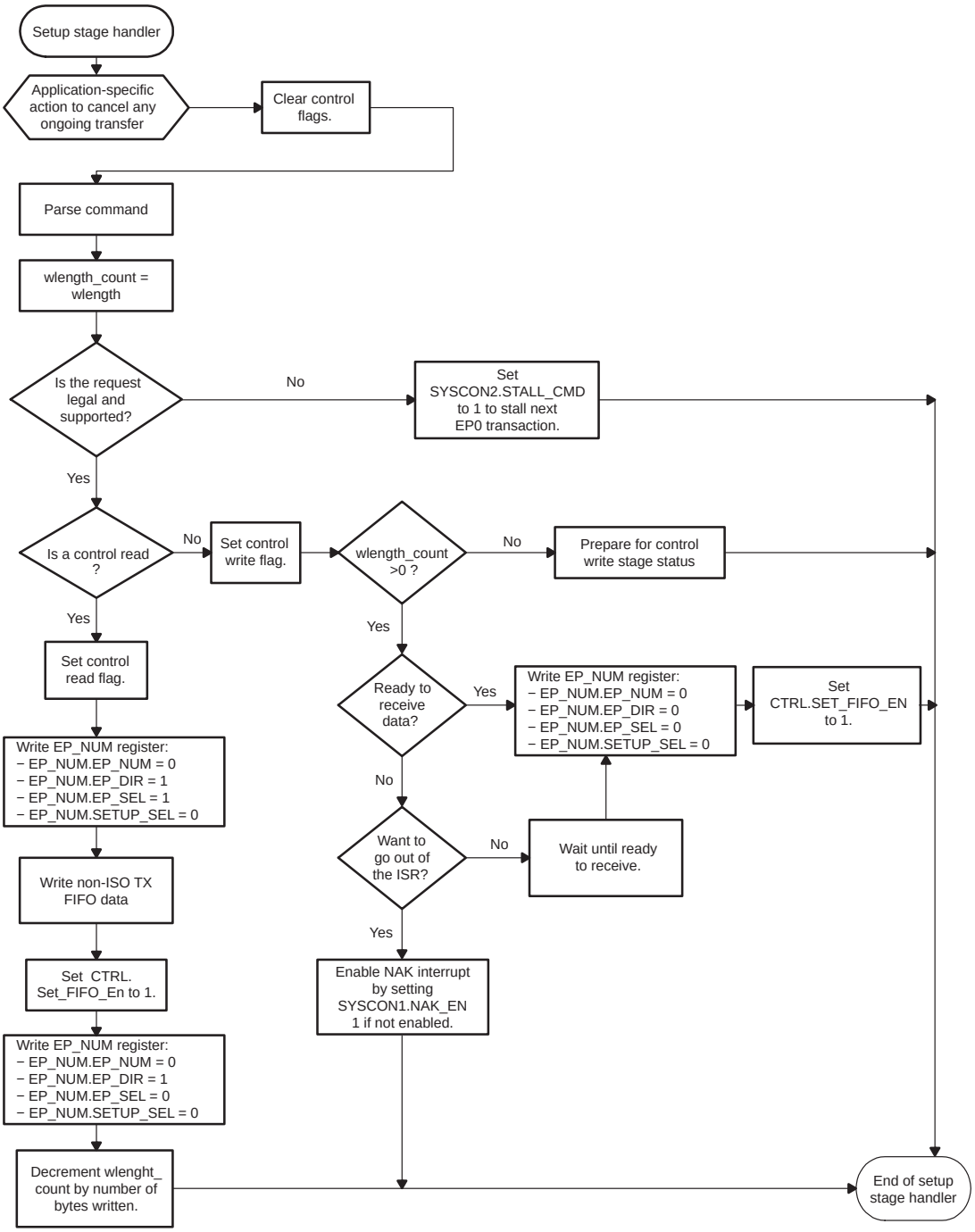
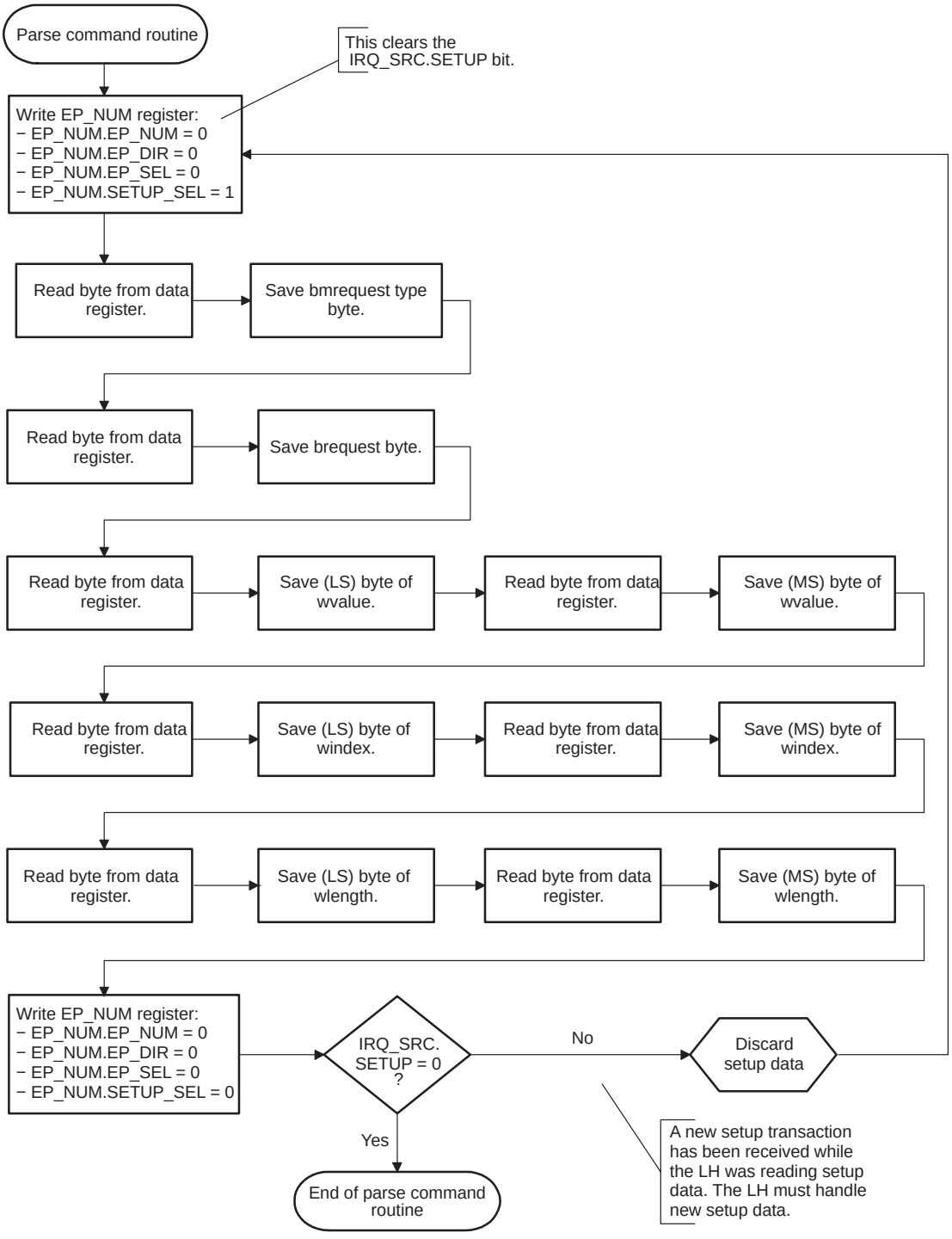


Figure 17. Parse Command Routine (Setup Stage Control Transfer Request)



3.14 Endpoint 0 RX Interrupt Handler

Figure 18 shows the endpoint 0 RX portion of the general USB interrupt handler, which must handle general USB interrupts related to control OUT transactions on endpoint 0. No EP0 interrupt is generated for autodecoded control transfers.

Figure 18. Endpoint 0 RX Interrupt Handler

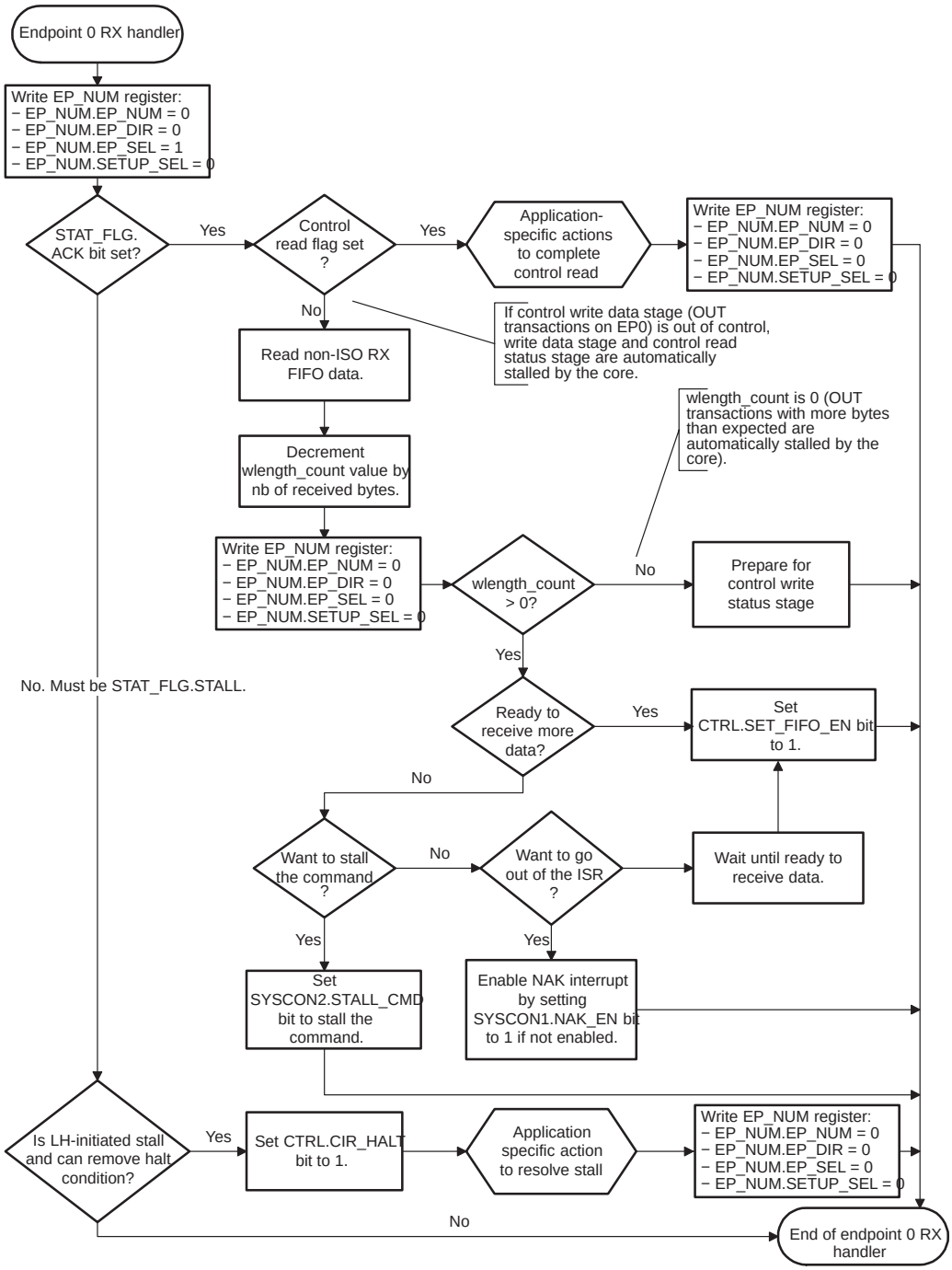
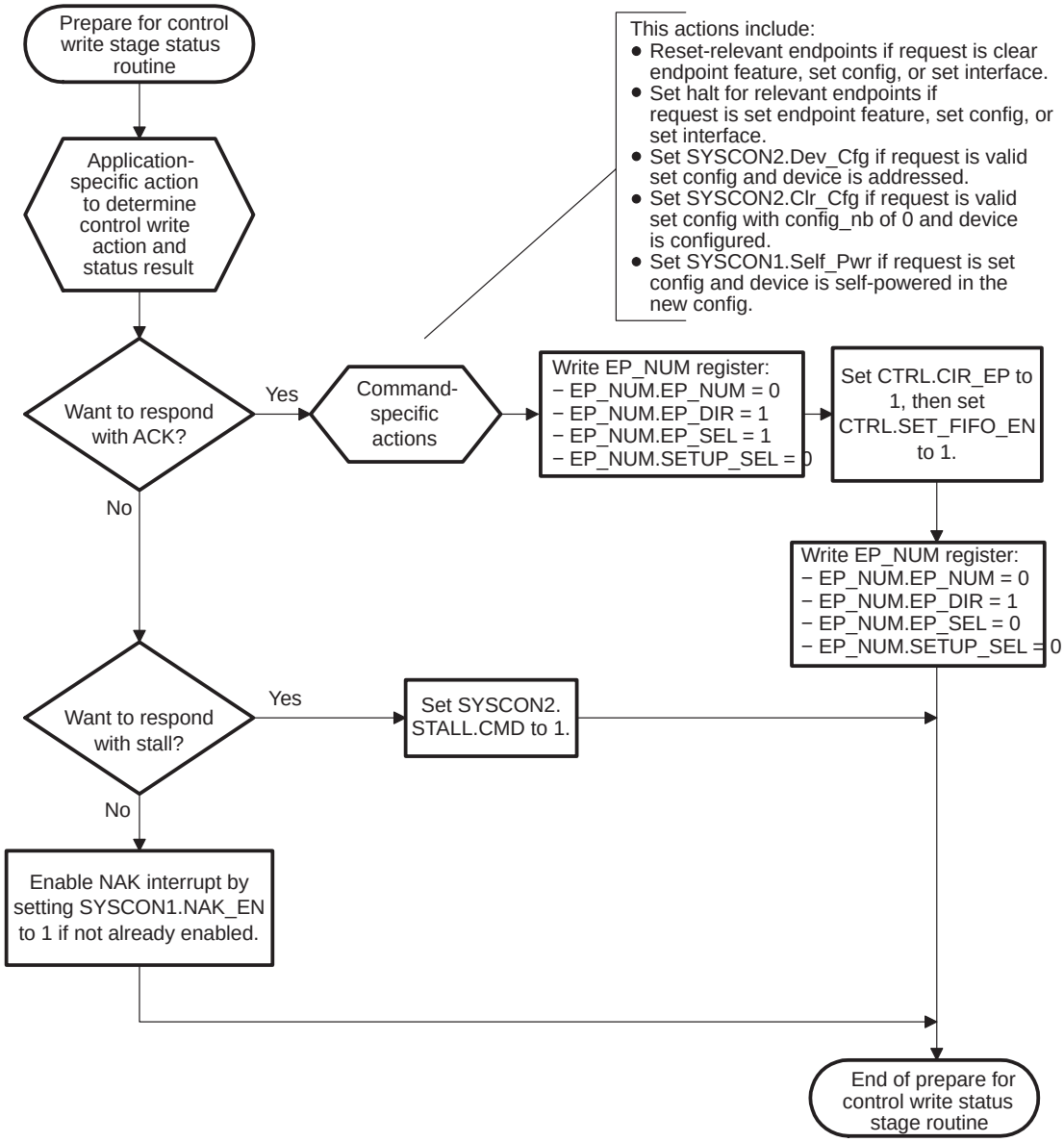


Figure 19. Prepare for Control Write Status Stage Routine



To support the SET_DESCRIPTOR command, when in commspecific actions, you must first reset all endpoints by selecting the endpoint and then clear it, unlock the configuration, charge the new one, lock the configuration, go in prepare for transfers, and enable all interrupts (see Figure 19).

3.15 Endpoint 0 TX Interrupt Handler

Figure 20 shows the TX portion of the general USB interrupt handler, which must handle 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 occurs signaling 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 21).

Figure 20. Endpoint 0 TX Interrupt Handler

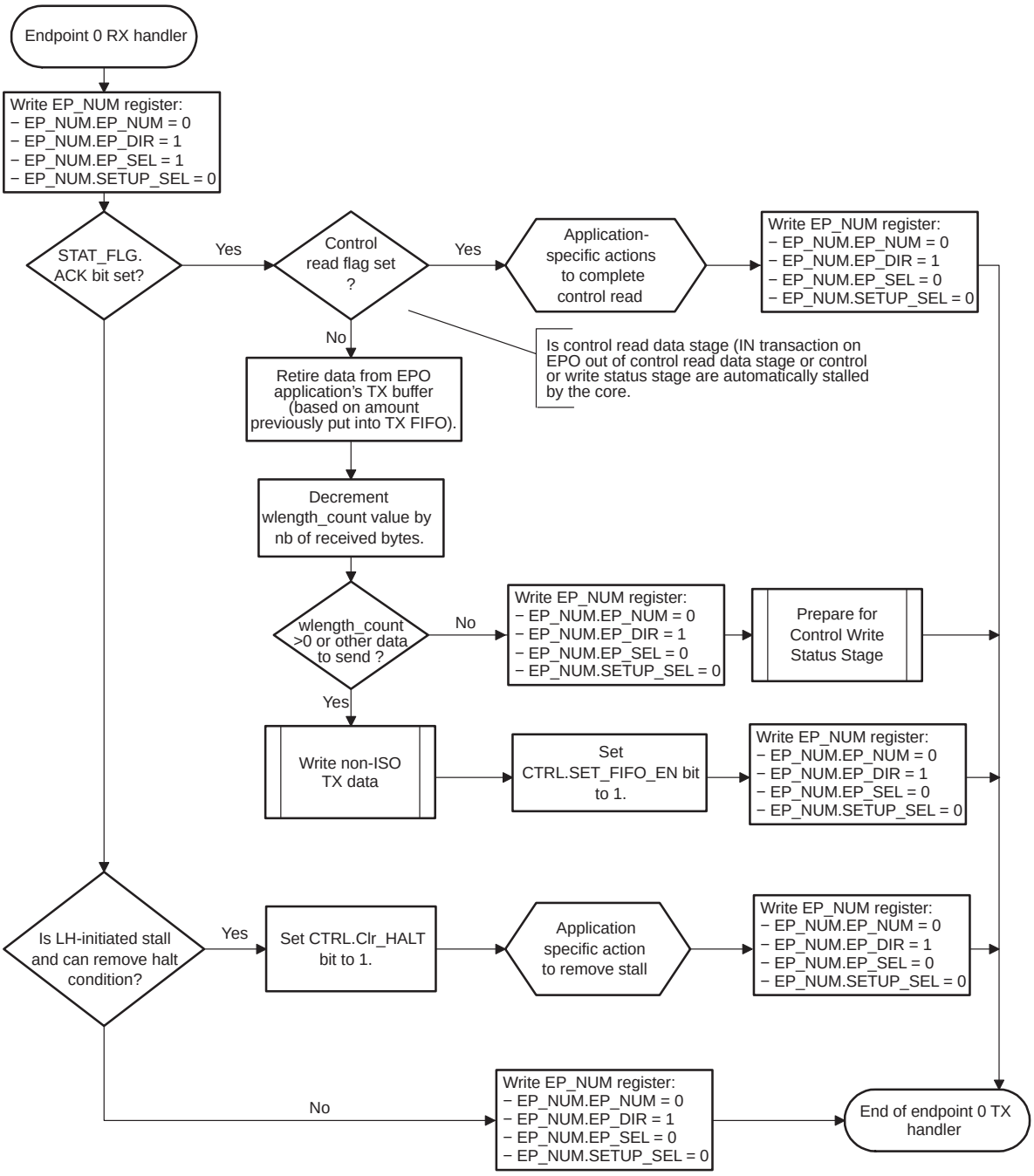
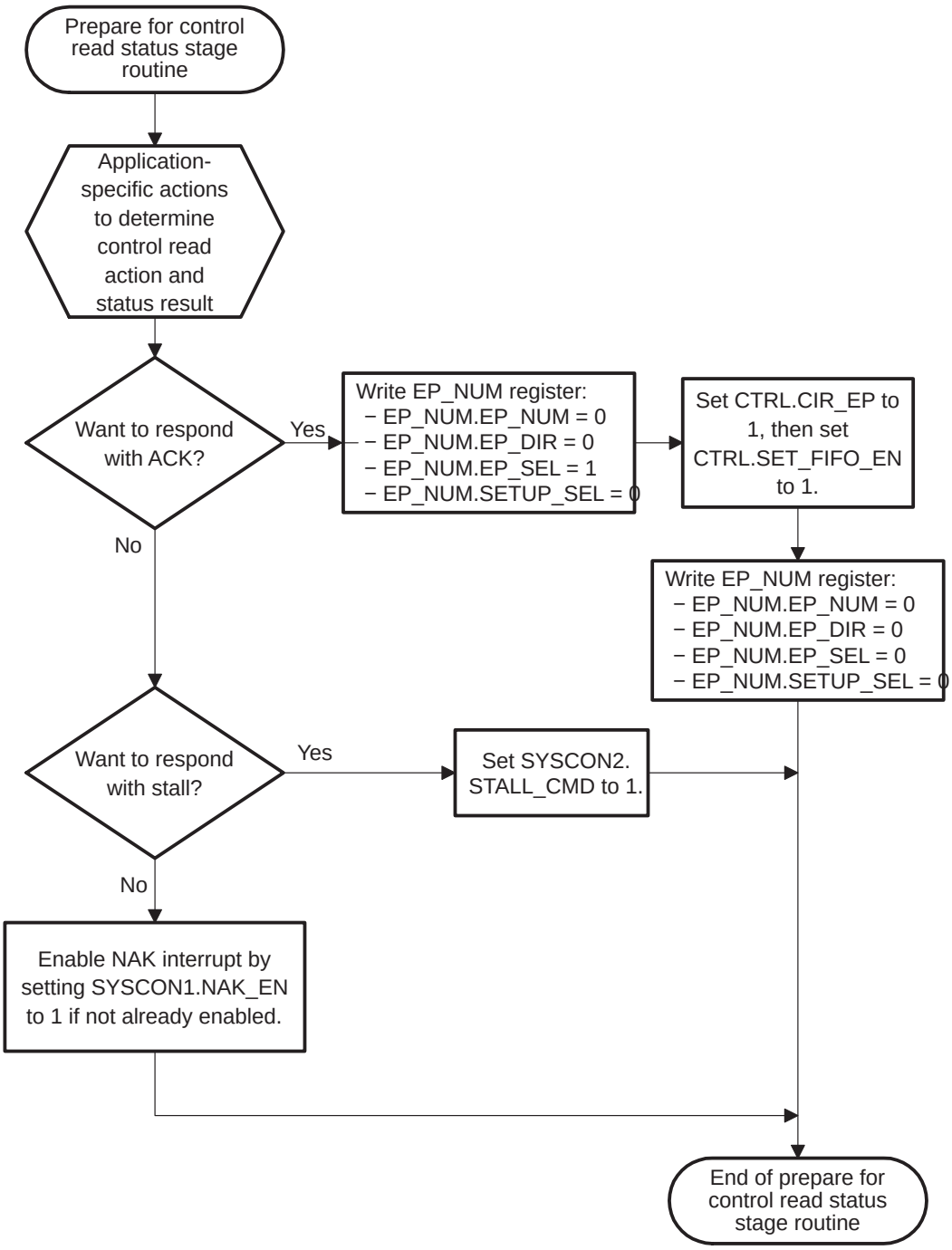


Figure 21. Prepare for Control Read Status Stage Routine



3.16 Device States Changed 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) shows how the USB device controller device moves from one state to another state with respect to the USB1.1 specification. Attach/unattach transition is not shown in the transition flow.

Because SET_CONFIGURATION is not decoded by the core, the MPU has the responsibility to distinguish a SET_CONFIGURATION with a nonvalid configuration value from other SET_CONFIGURATION requests and to set SYSCON2.DEV_CFG 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 has the responsibility to set SYSCON2.CLR_CFG 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.

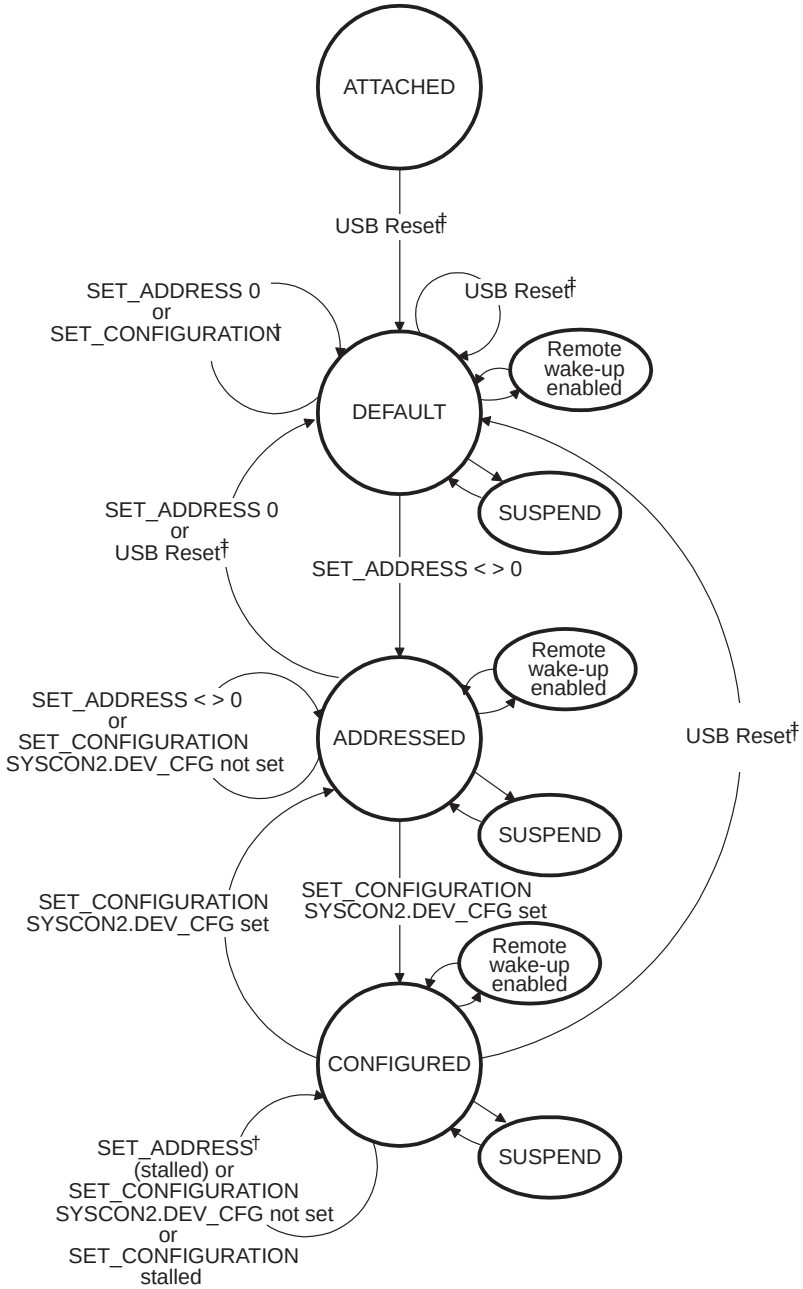
- Attached: The device is attached to the USB and powered.
- Default: The device is attached to the USB, is powered, and has been reset.
- Addressed: The device is attached to the USB, is powered, has been reset, and an address has been assigned. The device moves into the addressed state after a SET_ADDRESS request with an address number other than 0.
- Configured: The device is attached to the USB, is powered, has been reset, has an address other than 0, and is configured. The device moves into the configured state after a valid SET_CONFIGURATION request, only if the MPU has set the SYSCON2.DEV_CFG bit (meaning the configuration is valid).
- Suspended: The device is at minimum default and has not seen bus activity for 5 ms.
- Reset: When set, the device is receiving a valid USB host reset.
- R_WK_OK: 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 (IRQ_SRC.DS_CHG), 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

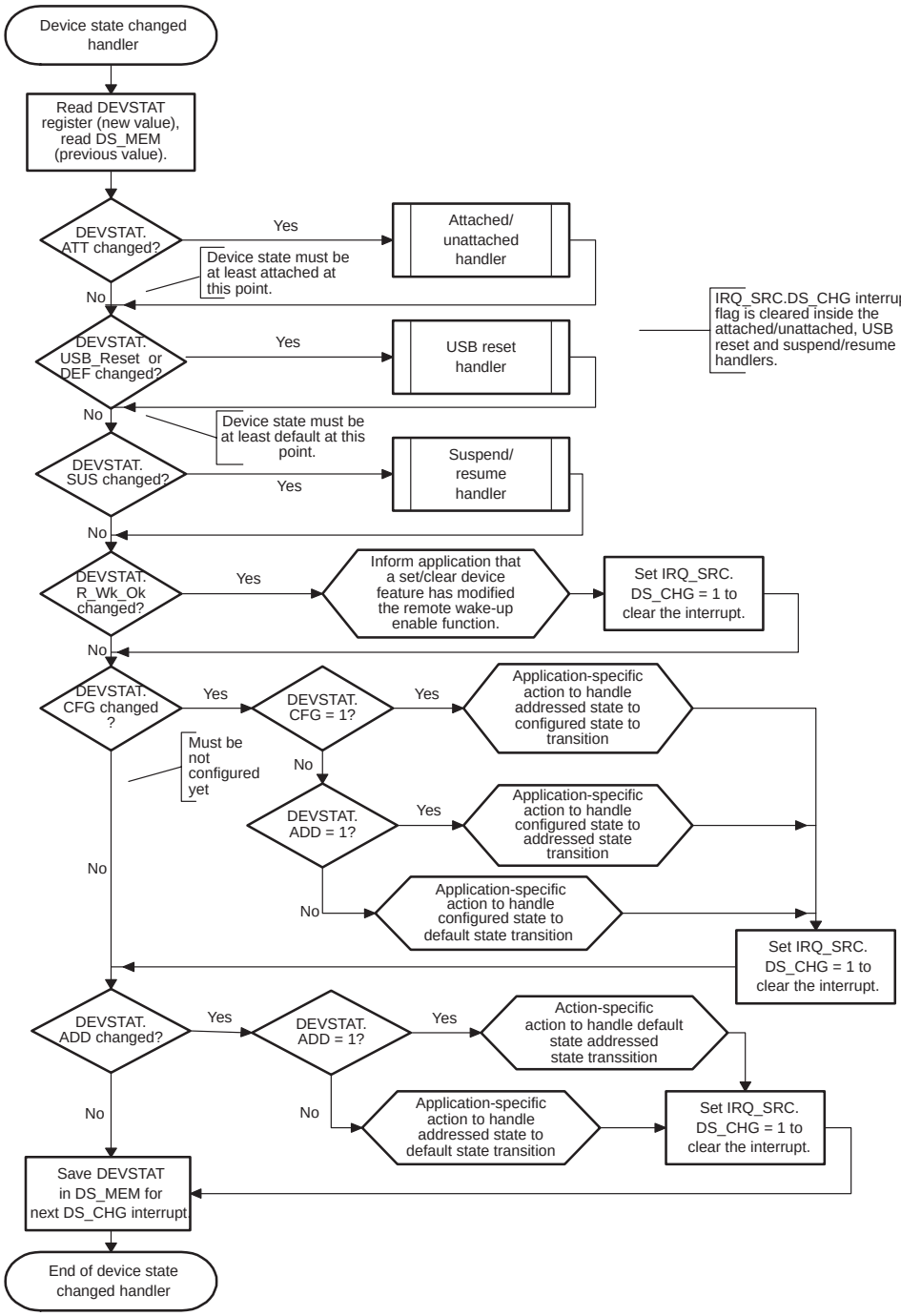
corrupted or not sent by the USB host. A SET_DEVICE_FEATURE or a CLEAR_DEVICE_FEATURE is effective after setup transaction, even if no status stage occurs. A SET_CONFIGURATION request is effective before status stage, when the MPU sets the SYSCON2.CLR_CFG or SYSCON2.DEV_CFG bit (see Figure 23).

Figure 22. USB Device Controller Device State Transitions



Behavior not specified by USB 1.1 specifications (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.

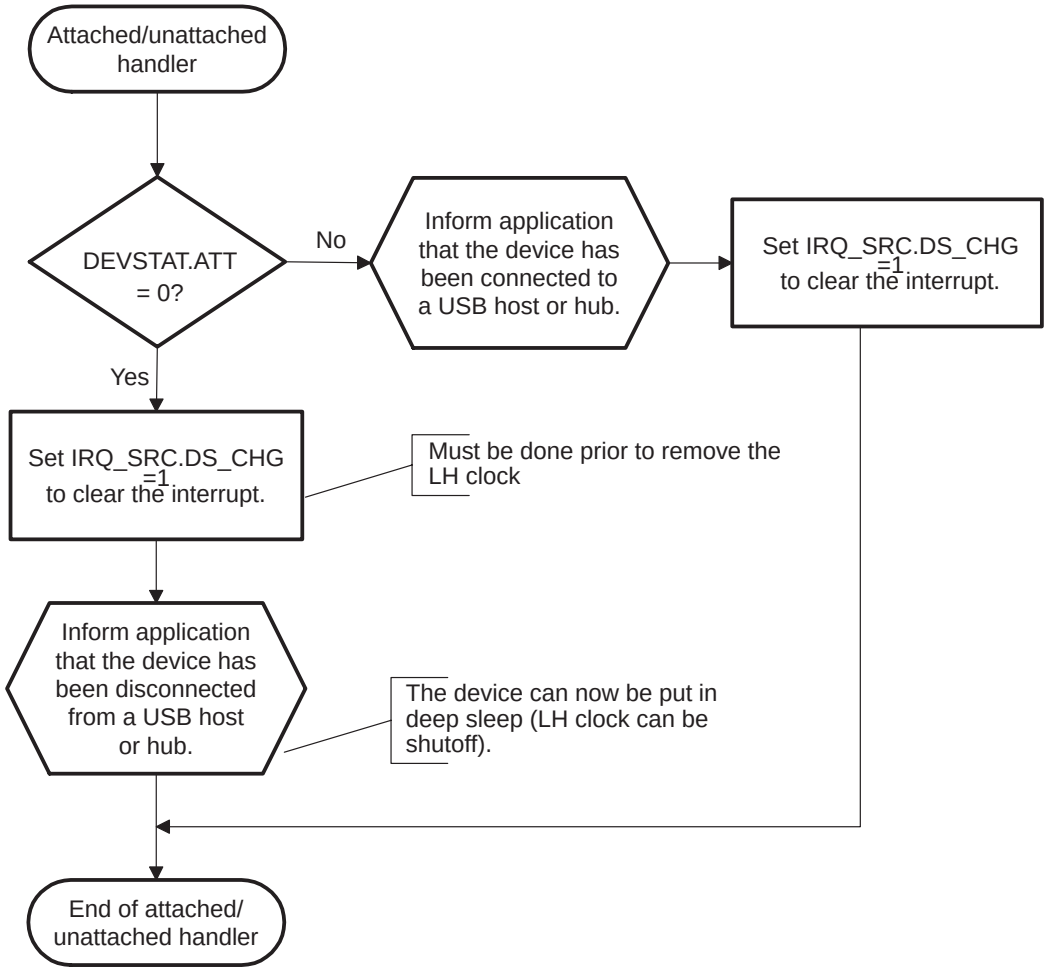
Figure 23. Typical Operation for USB Device State Changed Interrupt Handler



3.17 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 after IRQ_SRC.DS_CHG is cleared after servicing an unattached interrupt. Disabling the USB device controller clock before IRQ_SRC.DS_CHG is cleared can result in improper functionality for future USB device controller interrupts (see Figure 24).

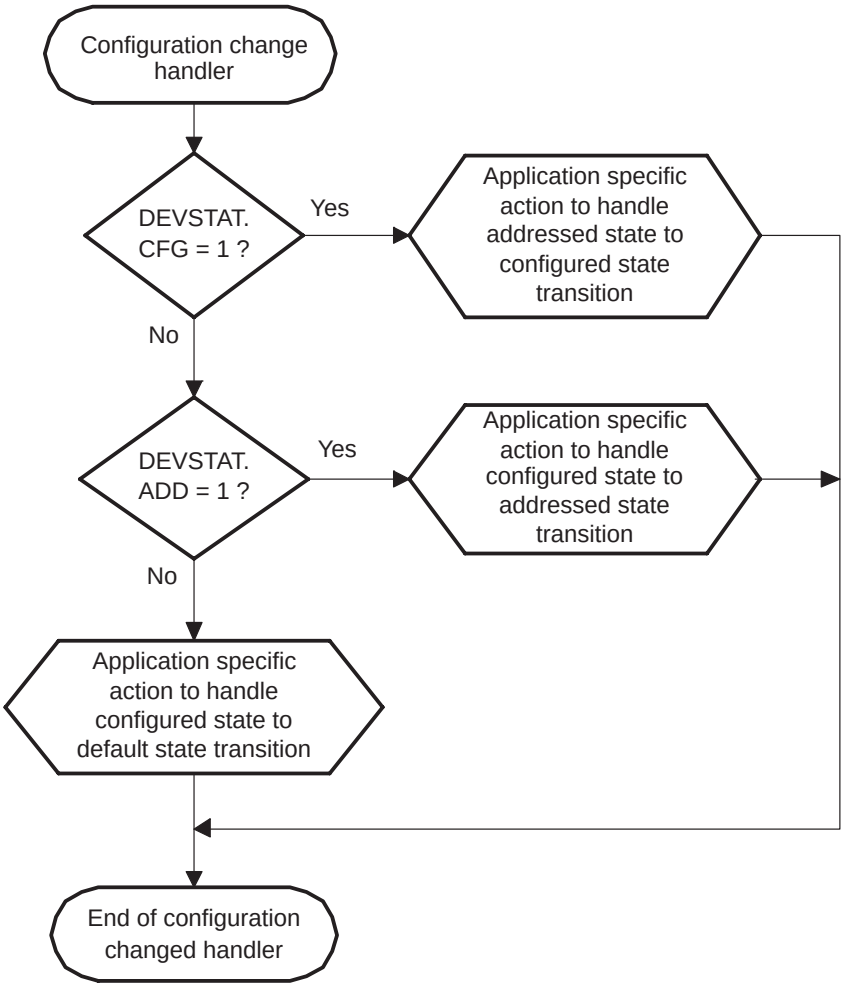
Figure 24. Attached/Unattached Handler



3.18 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 25.

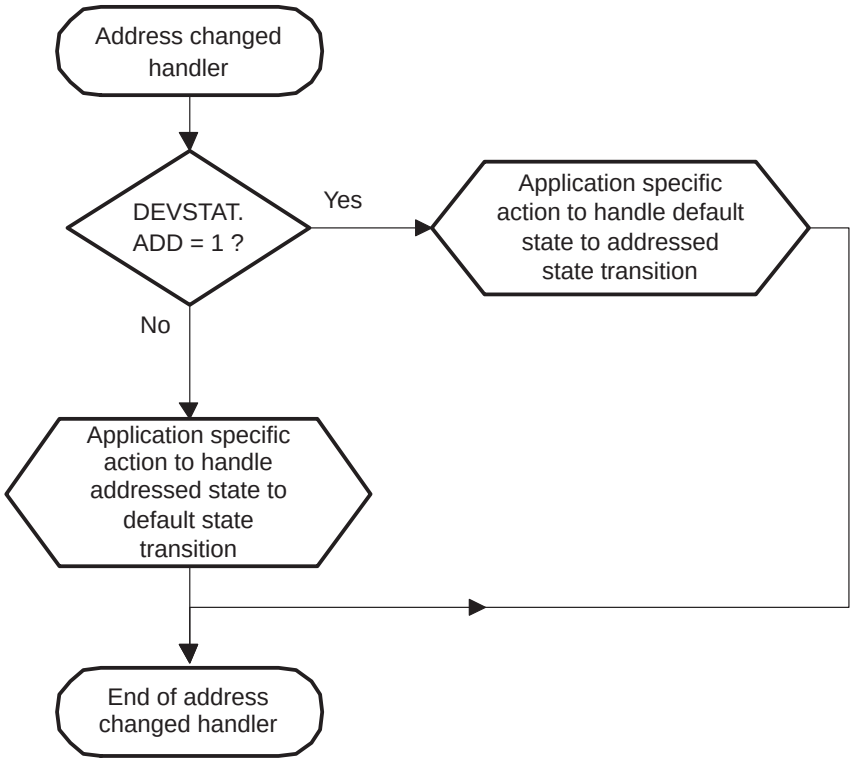
Figure 25. Configuration Changed Handler



3.19 Device State Address Changed Handler

When a 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 26.

Figure 26. Address Changed Handler

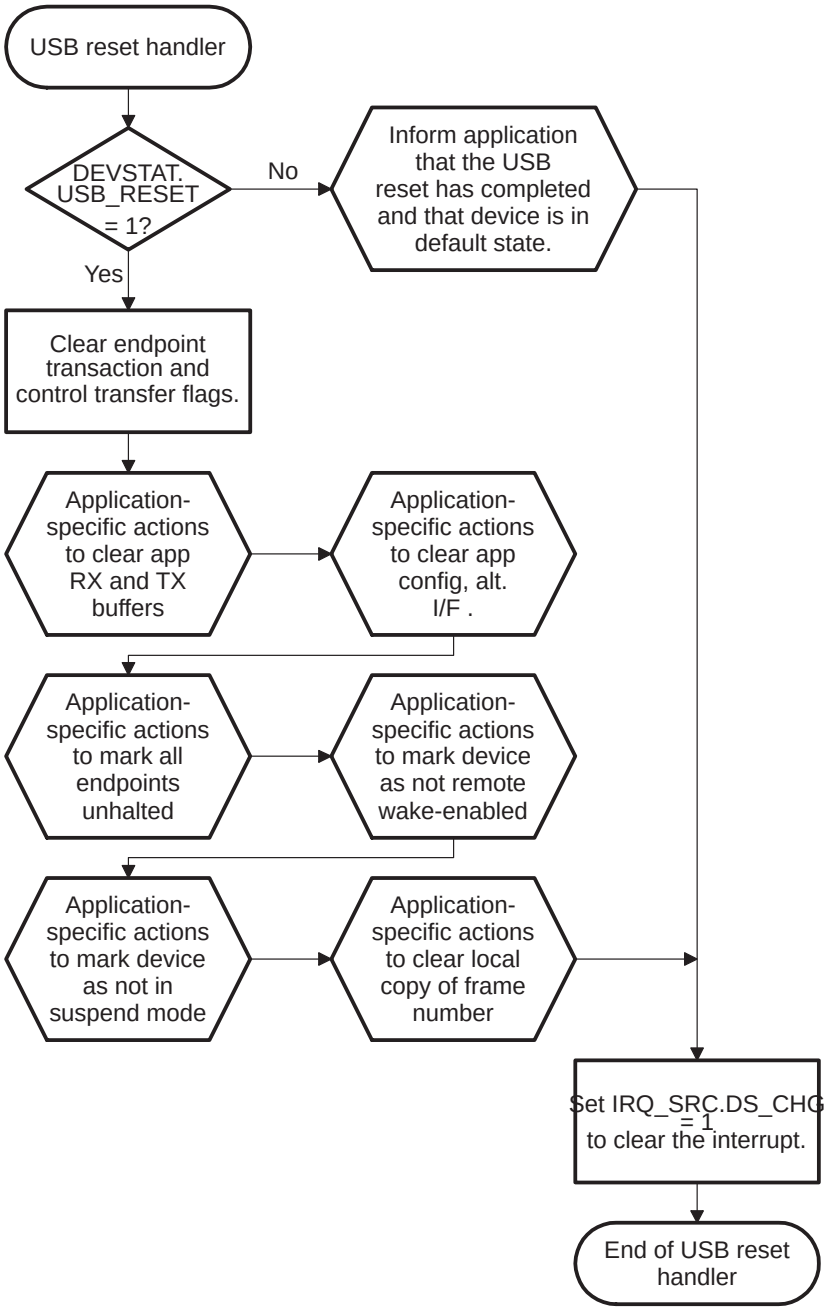


3.20 USB Device Reset Interrupt Handler

When a USB reset occurs, the USB module generates a general USB interrupt to the MPU (see Figure 27). The MPU 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 mode flag
- ❏ Clears any application-specific copy of the frame number

Figure 27. USB Device Reset Handler Flowchart



3.21 Suspend/Resume Interrupt Handler

When a USB device suspend/resume general USB interrupt occurs, the USB module has either entered or left suspend mode. The MPU code must determine which and react appropriately (see Figure 28).

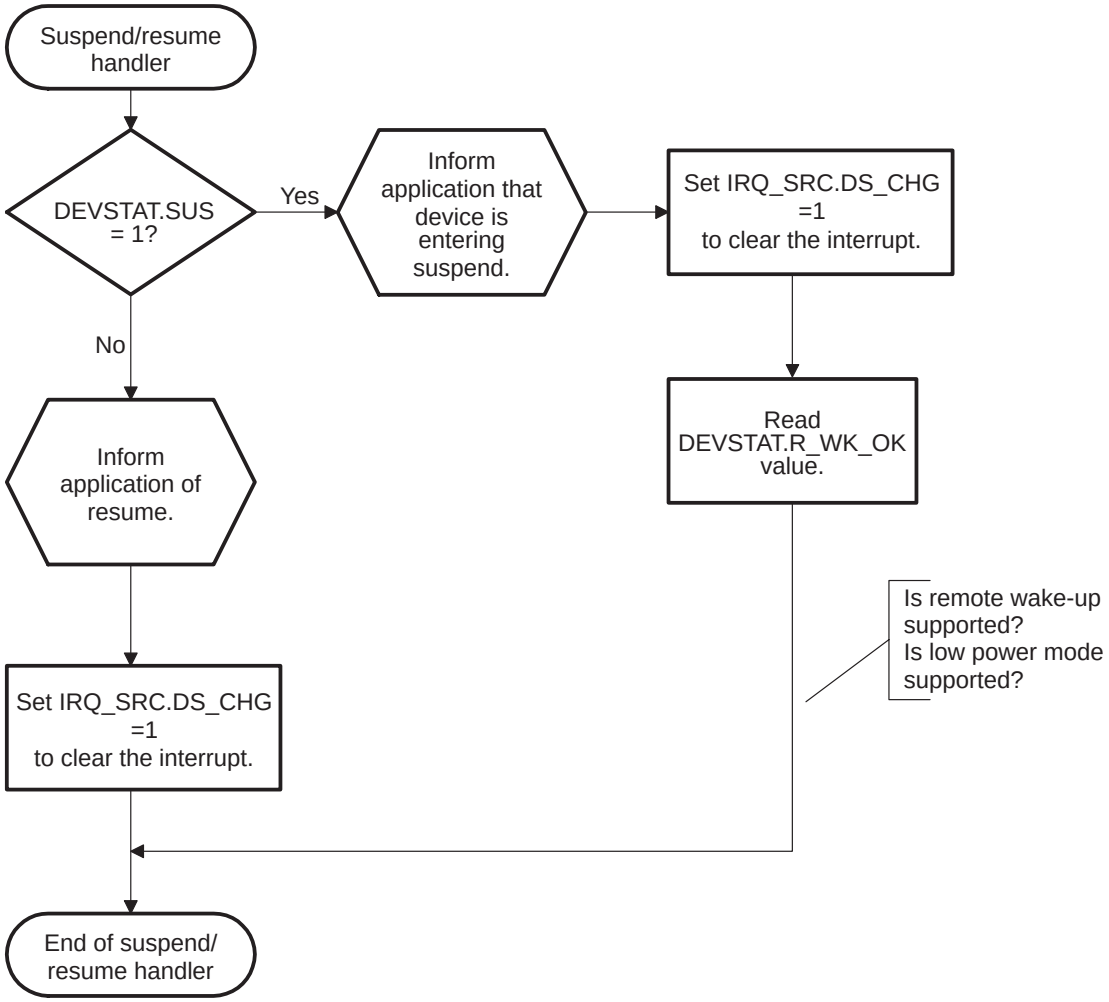
The suspend sense hardware is implemented to trigger only after 5 ms of bus idle. This forces compliance with the *USB Specification Version 1.1* T_{WTRSM} timing parameter (3 ms of IDLE to identify suspend, 2 ms before the remote device can signal resume).

If the MPU wants to wake the device from suspend mode and remote wake-up enable is set (DEVSTAT.R_WK_OK = 1), it must first turn its clock on (if stopped), and then set SYSCON2.RMT_WKP. 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 (USB host or MPU driven). Setting or not setting the SYSCON1.SOFF_DIS bit is part of the device configuration. However, the MPU can modify its value at suspend interrupt time if necessary.

A USB reset is also a valid way to exit suspend mode. But the suspend/resume handler and the USB reset handler do not have to take this into account, because three interrupts are generated in that case (1 for resume, 1 for reset, and 1 for end of reset).

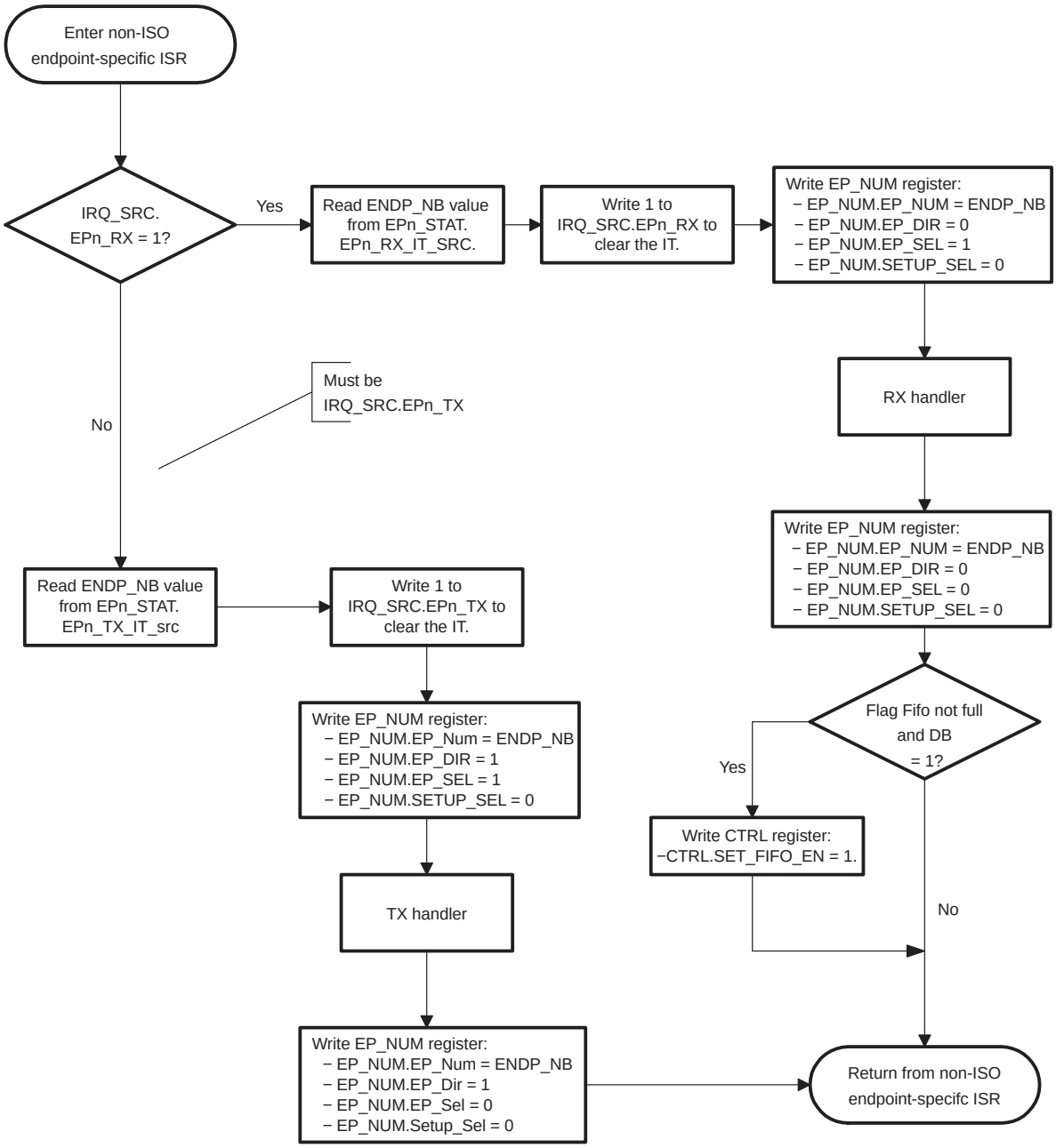
Figure 28. Typical Operation for USB Suspend/Resume General USB Interrupt Handler



3.22 Parsing Non-ISO Endpoint-Specific Interrupt

The endpoint-specific interrupt ISR (also known as the non-isochronous interrupt, on MPU Level 2 IRQ_30) must parse the interrupt identifier register IRQ_SRC to determine the interrupts that are active (IRQ_SRC.EPN_RX, IRQ_SRC.EPN_TX, 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 29).

Figure 29. Non-ISO Endpoint-Specific (Except EP 0) ISR Flowchart



3.23 Non-ISO, Non-Control OUT Endpoint Receive Interrupt Handler

Figure 30 shows the operations necessary to handle non-ISO, non-control OUT endpoint-specific receive interrupts. This flowchart shows two different RX transaction handshaking interrupts. There is a third interrupt handshaking possibility when NAK interrupts are enabled, which is not depicted here. Depending on the application-specific actions needed for various endpoints in the real system, it is possible to use one routine that is common to all of the non-ISO, non-control receive endpoints, where the only differences are in the EP_NUM register value set and the selection of the proper application RX data buffer in the read non-ISO RX FIFO data routine (see Figure 31).

This flowchart does not attempt to document control endpoint 0 receive interrupts, which are discussed separately because of the more complex three-stage transfer mechanism used for control writes.

Figure 30. Non-Isochronous Non-Control Endpoint Receive Interrupt Handler

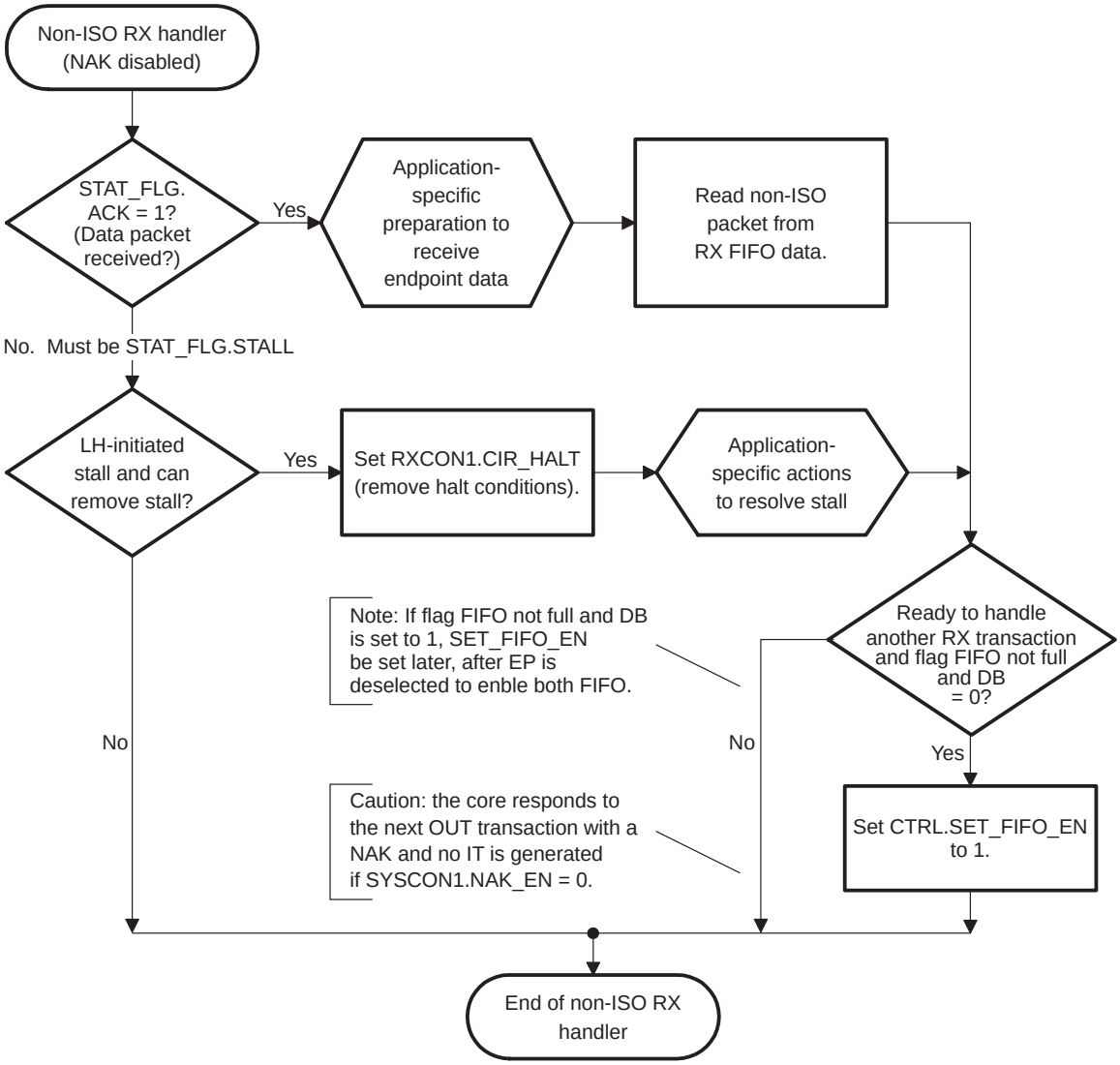
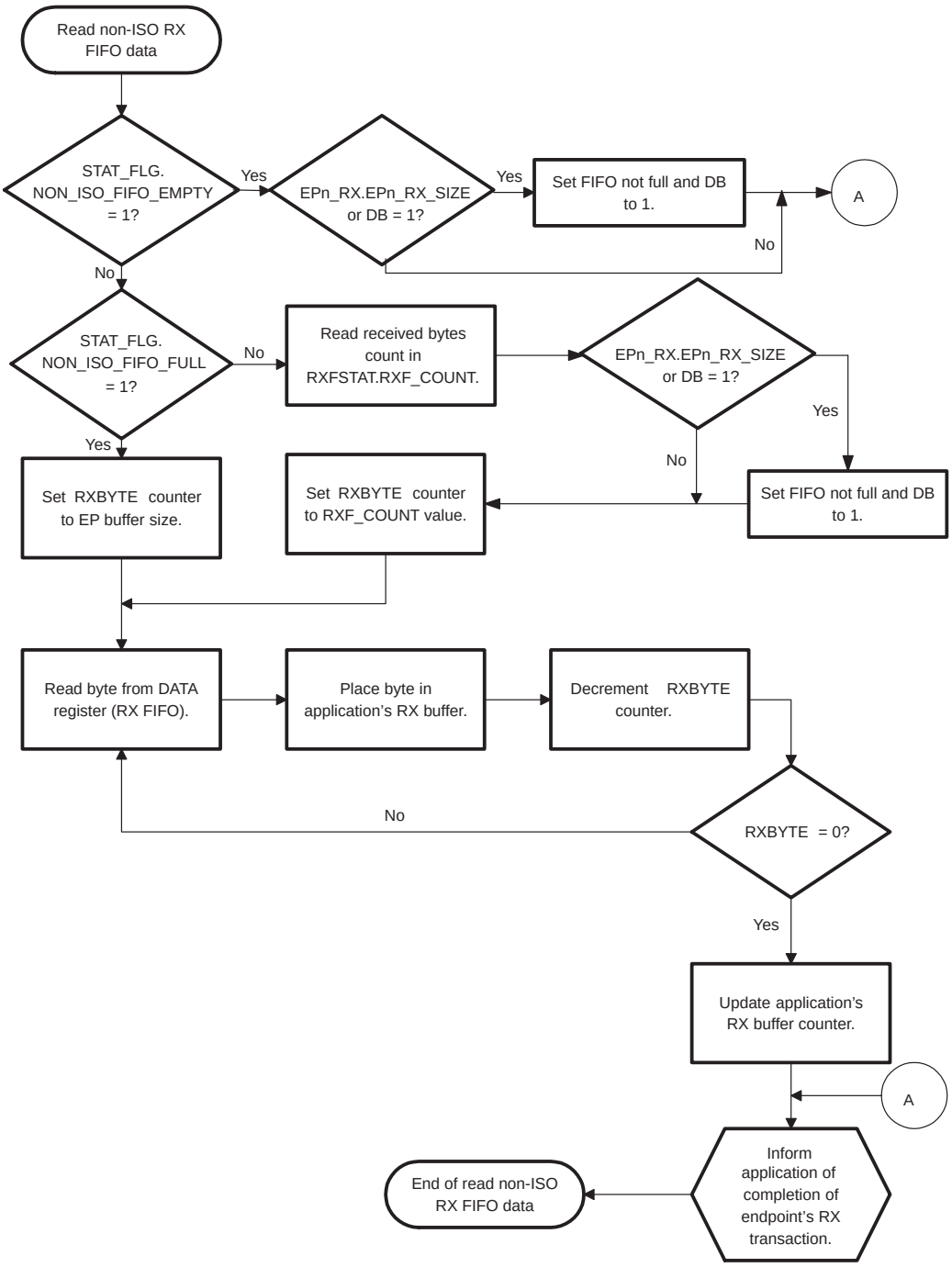


Figure 31. Read Non-Isochronous RX FIFO Data Flowchart



3.24 Non-ISO, Non-Control IN Endpoint Transmit Interrupt Handler

Figure 32 shows the operations necessary to handle non-ISO, non-control IN endpoint-specific transmit interrupts. This flowchart shows two TX transaction handshaking interrupts. There is a third interrupt handshaking possibility when NAK interrupts are enabled, which is not depicted here. Depending on the application-specific actions needed for various endpoints in the real system, it is possible to use one routine that is common to all of the non-ISO, non-control transmit endpoints, where the only differences are in the EP_NUM register value set and in the routine selection of the application TX buffer (see Figure 33).

This flowchart does not attempt to document control endpoint 0 transmit interrupts, which are discussed separately because of the more complex three-stage transfer mechanism used for control reads.

Figure 32. Non-Isochronous Non-Control Endpoint Transmit Interrupt Handler

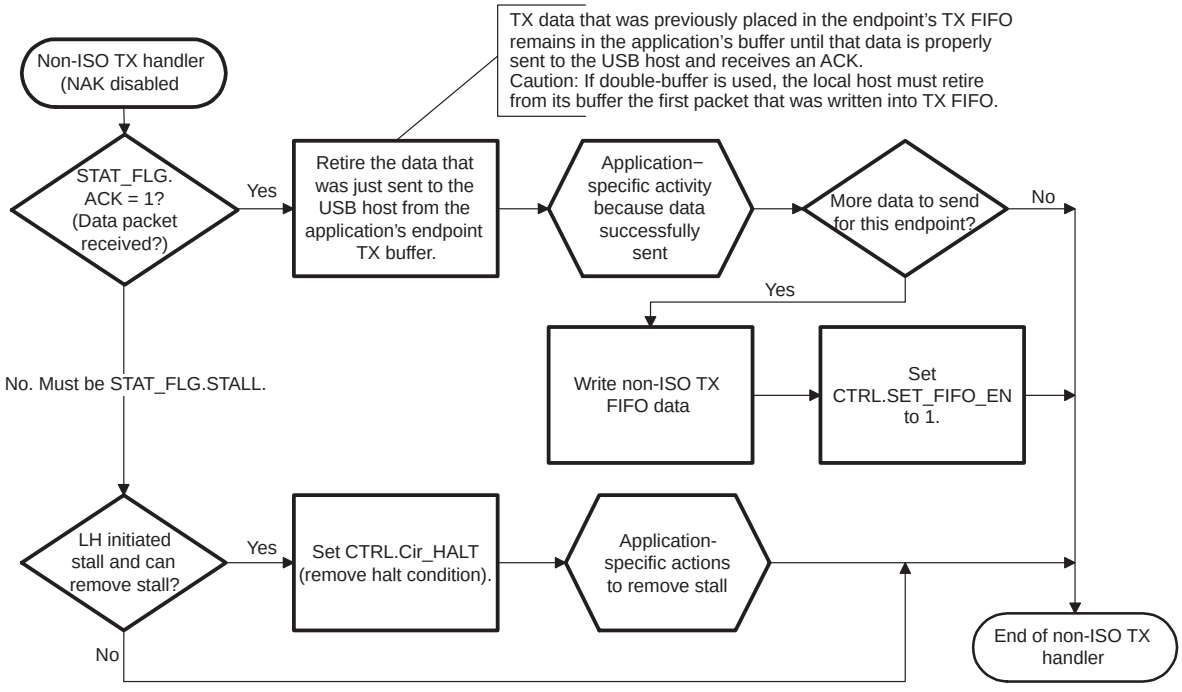
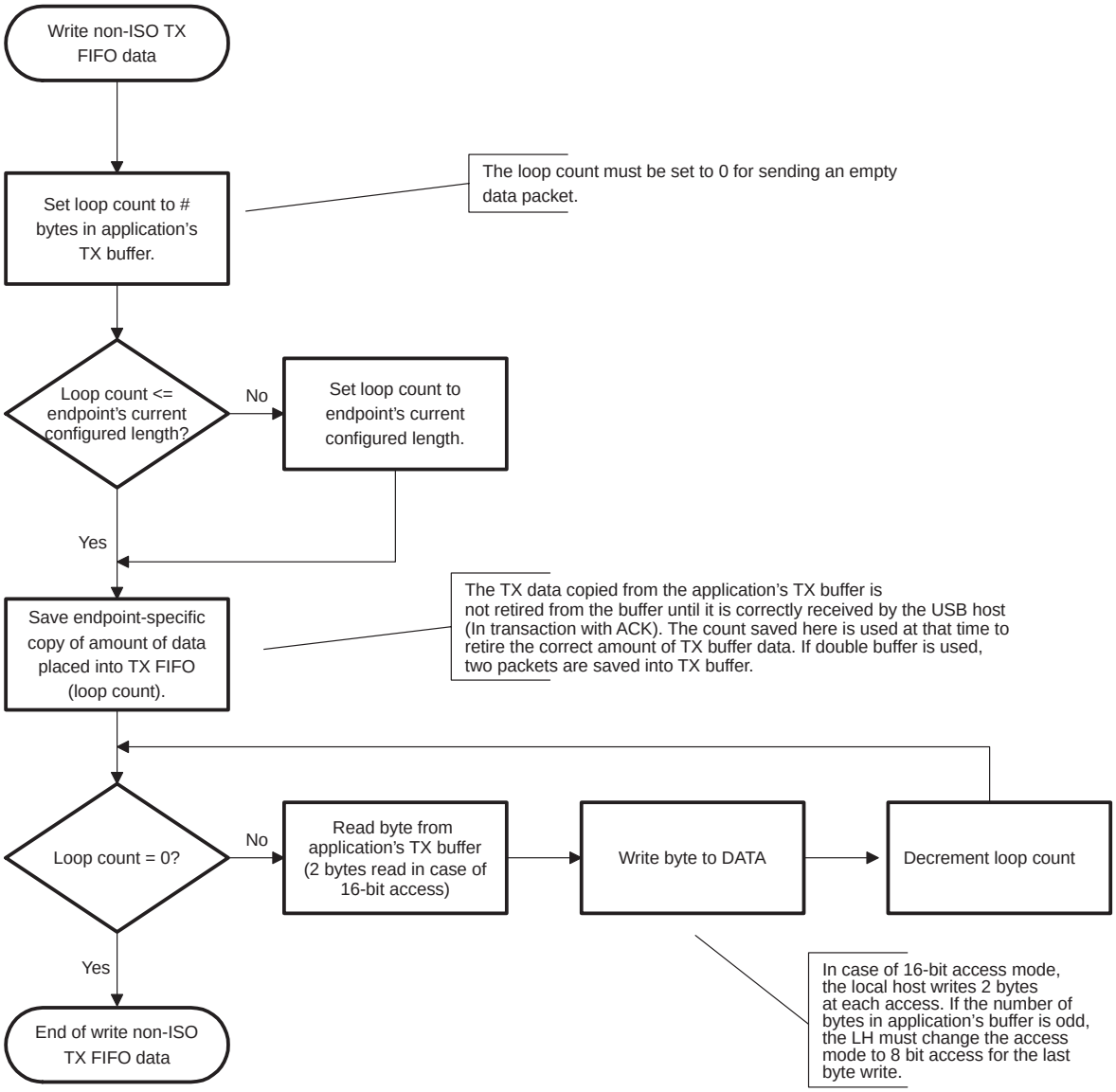


Figure 33. Write Non-Isochronous TX FIFO Data Flowchart



3.25 SOF Interrupt Handler

SOF interrupts to the MPU (also known as isochronous interrupts on MPU level 2 IRQ_29) occur once per USB frame. The MPU 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 34 shows the SOF ISR flowchart. The read ISO RX FIFO

data and write ISO TX FIFO data procedures are shown Figure 35 and Figure 36, respectively.

Figure 34. SOF Interrupt Handler Flowchart

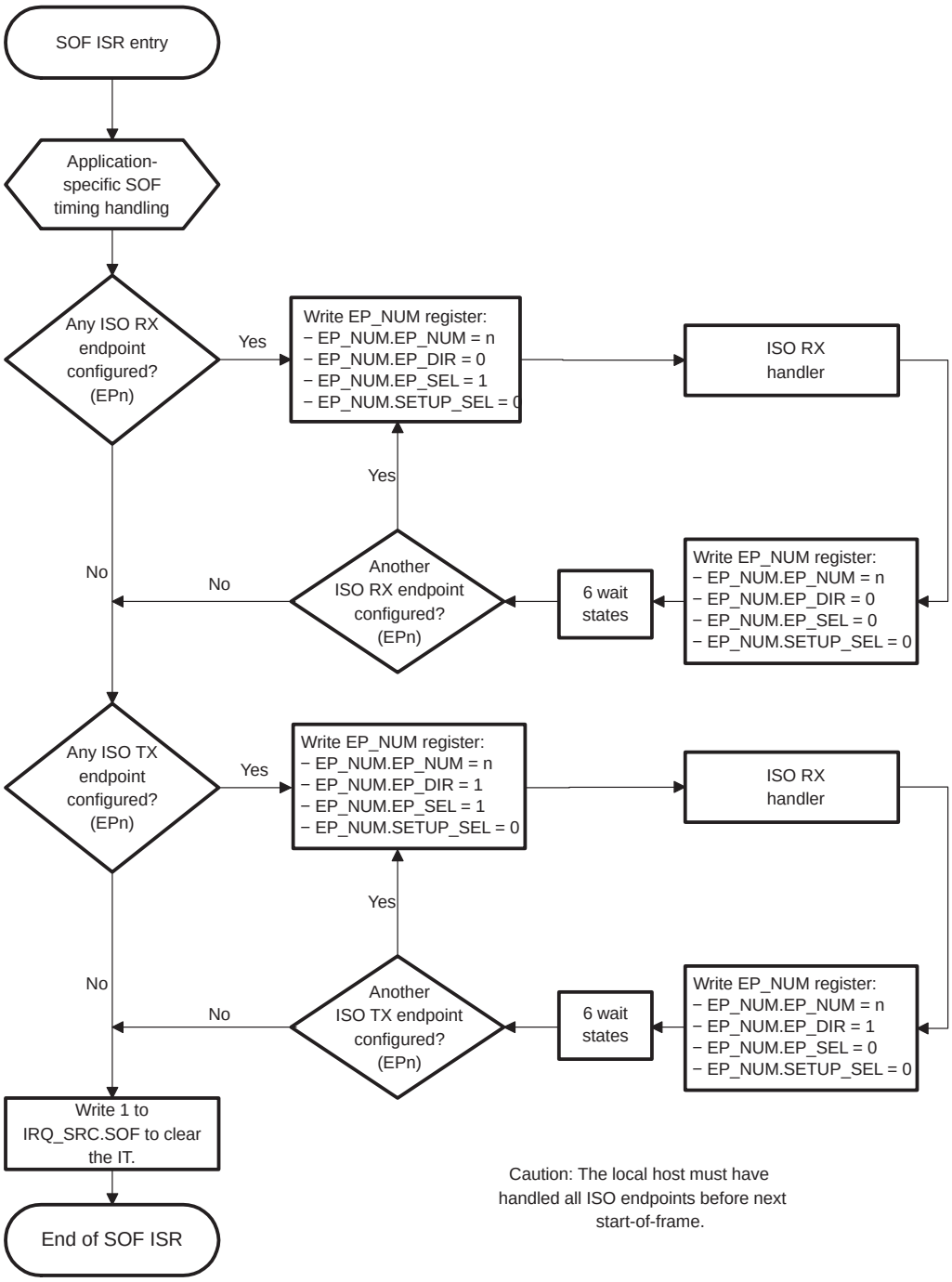


Figure 35. Read Isochronous RX FIFO Data Flowchart

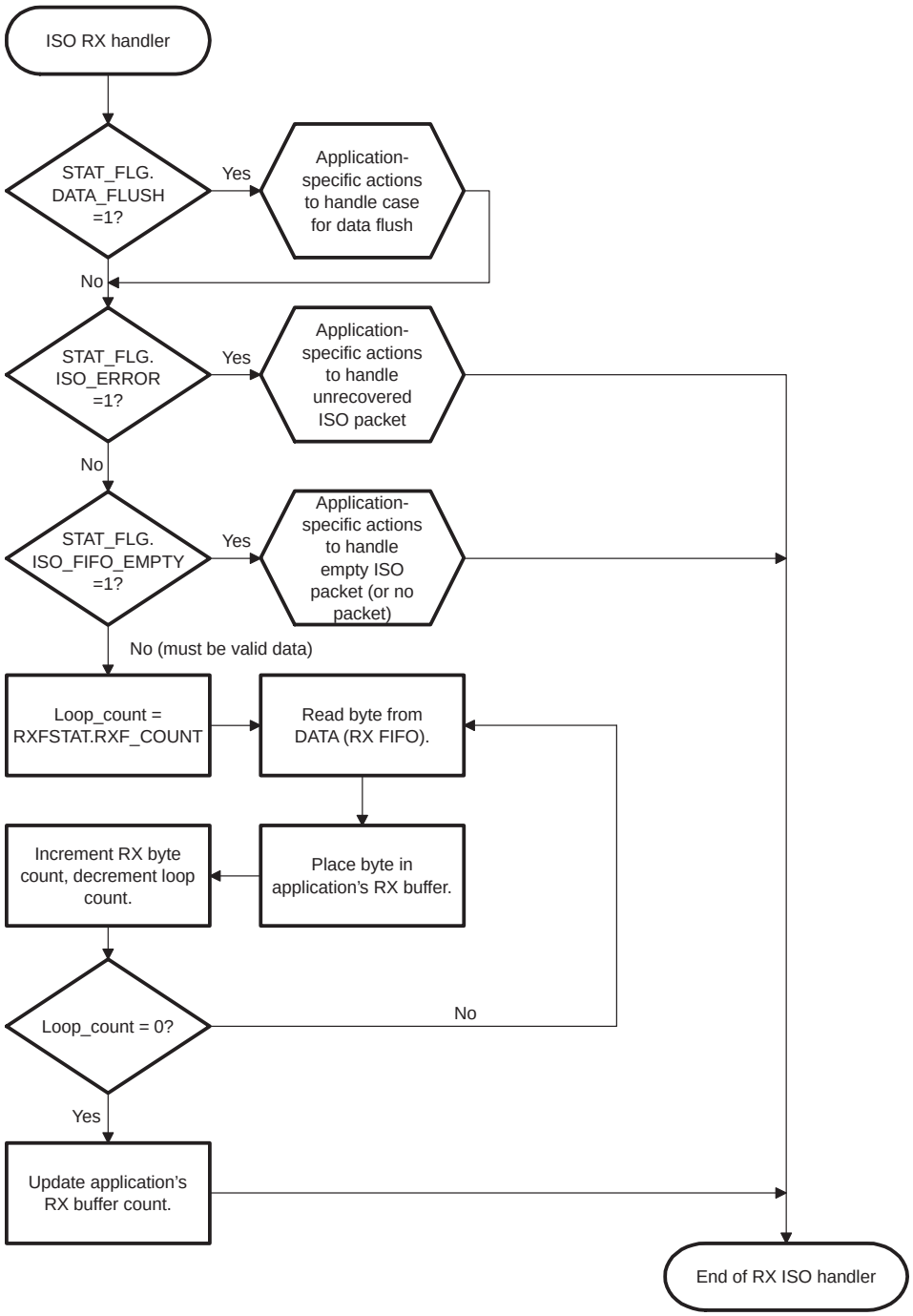
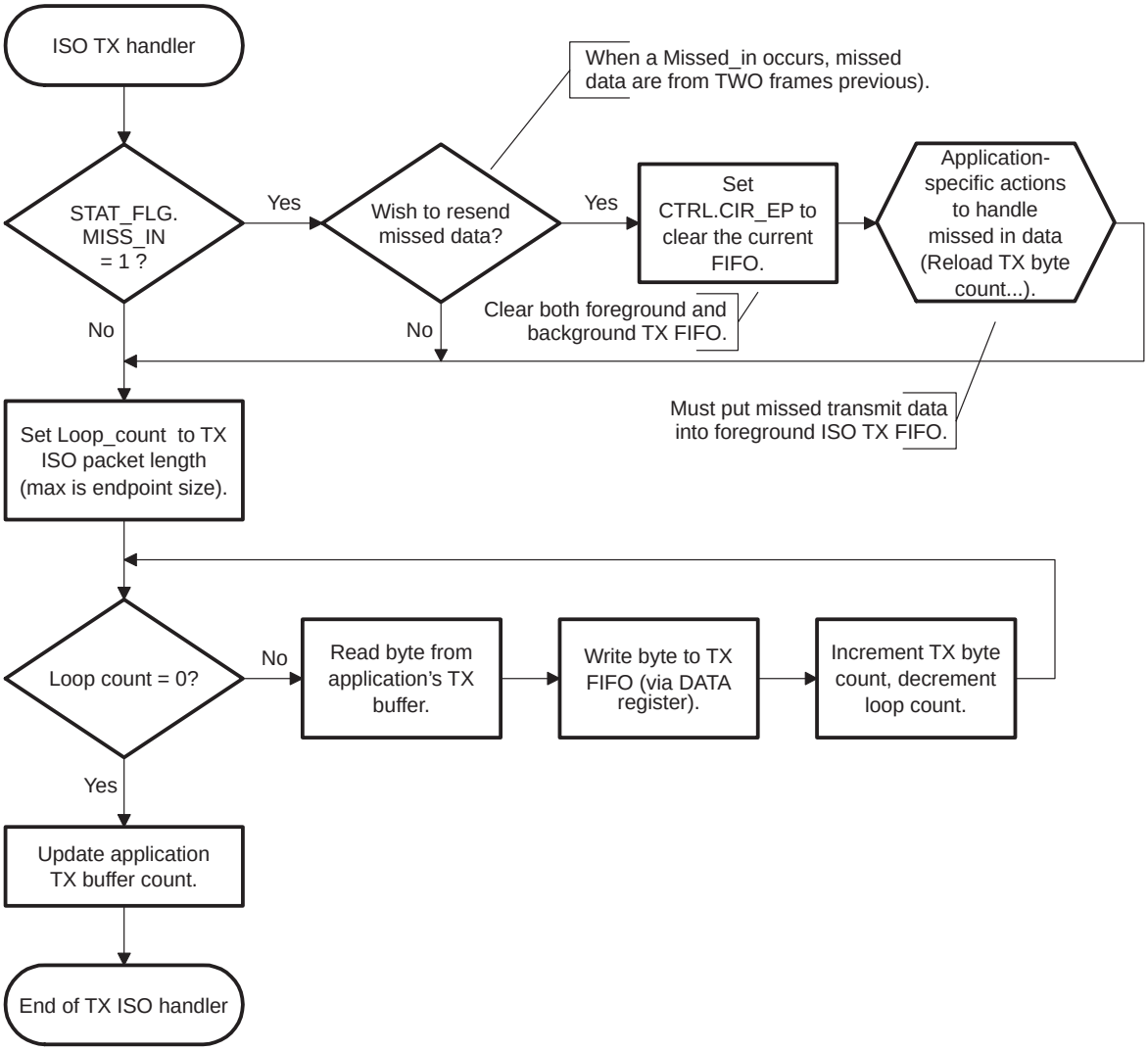


Figure 36. Write Isochronous TX FIFO Data Flowchart



3.26 Summary of USB Device Controller Interrupts

Table 56 lists the interrupt types by endpoint types.

Table 56. USB Device Controller Interrupt Type by Endpoint Type

Interrupt Type	General USB IRQs (MPU Level 2 IRQ_20)				EP-Specific IRQs (Non-ISO Interrupt on MPU Level 2 IRQ_30)		SOF ISO Interrupt on MPU Level 2 IRQ_29)
	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			

3.26.1 USB Device Controller Clock Control

The OMAP5912 clock generation and system reset management module(ULPD) provides a single 48-MHz clock to the USB OTG controller, USB device controller, and USB host controller. This clock can be stopped by software to reduce USB OTG controller, USB host, and USB device controller power consumption when USB functionality is not needed. The ULPD controls the 48-MHz clock to the USB device controller via:

- ❏ CLOCK_CTRL_REG.USB_MCLK_EN
- ❏ SOFT_REQ_REG.SOFT_USB_OTG_DPLL_REQ
- ❏ SOFT_DISABLE_REQ_REG.DIS_USB_HOST_DPLL_REQ

The OTG module allows separate control of USB device controller clocking via OTG_SYSCON_1.DEV_IDLE_EN.

3.26.2 USB Device Controller Hardware Reset

Reset of the USB device controller is provided by the ULPD module. The PER_EN bit in the MPU reset control 2 register controls the reset to many OMAP5912 peripherals, including the USB device controller. When held in reset, the USB device controller does not recognize any USB activity.

When the USB device controller is in reset, its registers have no effect on USB functionality. Software must wait until OTG_SYSCON_1.RESET_DONE is 1.

3.27 DMA Operation

The USB device controller provides support for six DMA channels. Three receive DMA channels are reserved to OUT transfers (ISO or non-ISO) and three transmit DMA channels are reserved to IN transfers (ISO or non-ISO). It is not possible to operate DMA transactions on control EP0.

The MPU must not access an endpoint used in a DMA transfer through the EP_NUM, CTRL, and STAT_FLG registers (in DMA, this remark applies after the MPU has set the CTRL.SET_FIFO_EN bit to enable the RX DMA transfer). In particular, the MPU must not set the halt feature while the endpoint is selected in the RXDMA_CFG register.

Note:

To use the DMA channels properly, you must 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 via the three RXDMA control registers. Each channel is assigned to a given endpoint number by assigning a non-zero value in RXDMA_CFG.RXDMAn_EP fields (a 0 value means the DMA channel is deselected). Received OUT data must be read when an RX DMA request is active, through the register DATA_DMA. 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 transmit DMA channels 0 through 2 are connected to OMAP5912 DMA controller requests DMA_REQ_26, DMA_REQ_27, and DMA_REQ_29, respectively.

Non-Isochronous 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 have been 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 Interrupt (IRQ_SRC.RXn_EOT)

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 RXDMAn.RXn_TC field, if the RXDMAn.RXn_STOP bit has been set by the MPU

After an end of transfer interrupt, the MPU must set CTRL.SET_FIFO_EN for the endpoint to reenable the channel.

The MPU must not initiate a new RX DMA transfer until it receives an end-of-transfer interrupt.

Transaction Count Interrupt (IRQ_SRC.RXn_CNT)

The intent of this interrupt is watermark control. It can be used by the MPU to monitor the file size of incoming transfers and take appropriate actions if, for instance, the file being received 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 (and not bytes) has reached 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. RXn_COUNT interrupt is

asserted even if RXDMAn.RXn_STOP has been set; in that case, both RXn_COUNT and RXn_EOT interrupts are asserted (see Figure 37 through Figure 40).

The transaction count watermark is programmed in the RXDMAn register.

Figure 37. Non-ISO RX DMA Transaction Example (RX_TC=2)

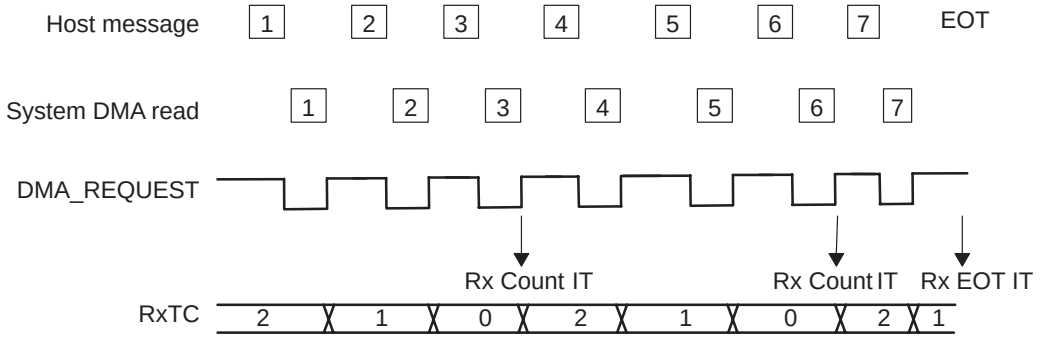


Figure 38. Non-ISO RX DMA Start Routine

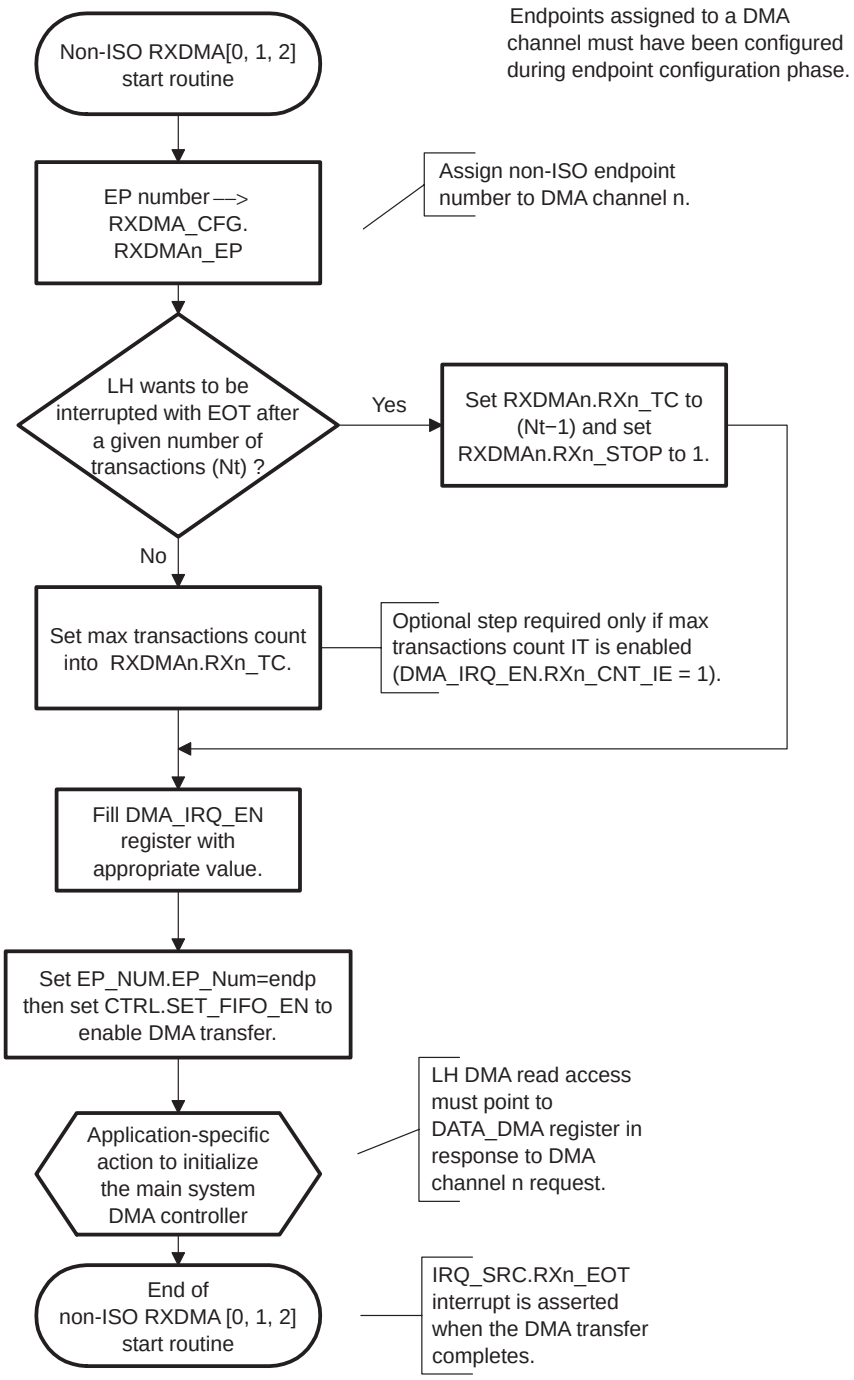


Figure 39. Non-ISO RX DMA EOT Interrupt Handler

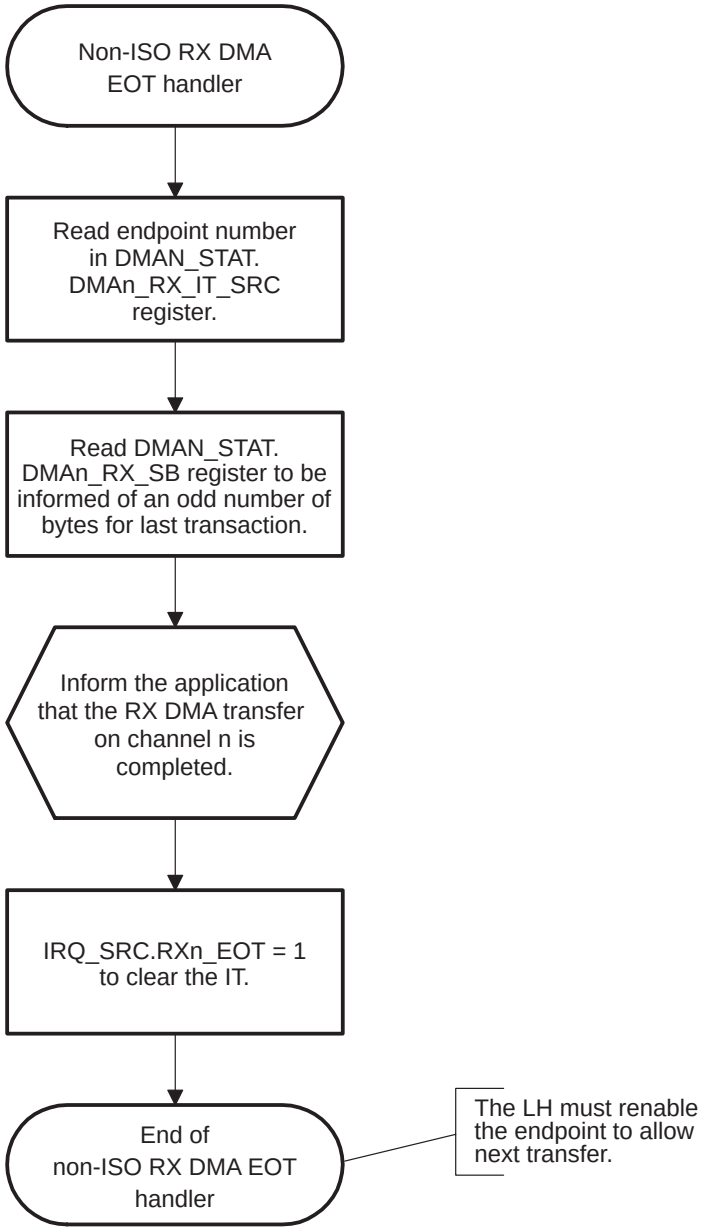
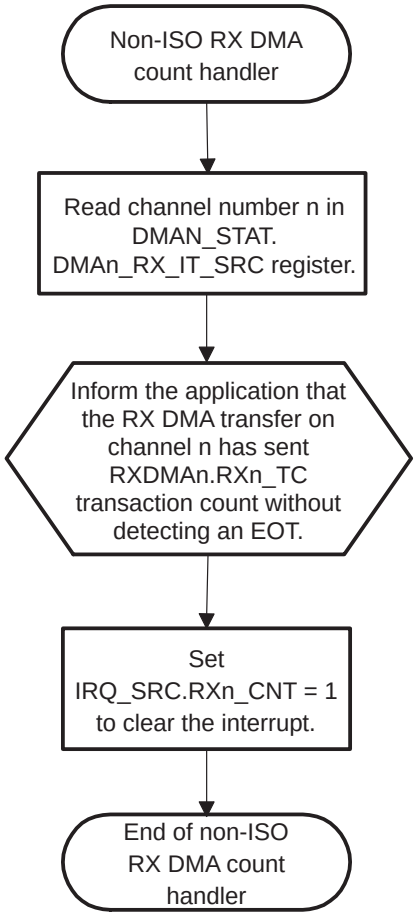


Figure 40. Non-ISO RX DMA Transactions Count Interrupt Handler



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. There is no interrupt associated with DMA transfer to ISO OUT endpoints (see Figure 41 and Figure 42).

Figure 41. ISO RX DMA Transaction

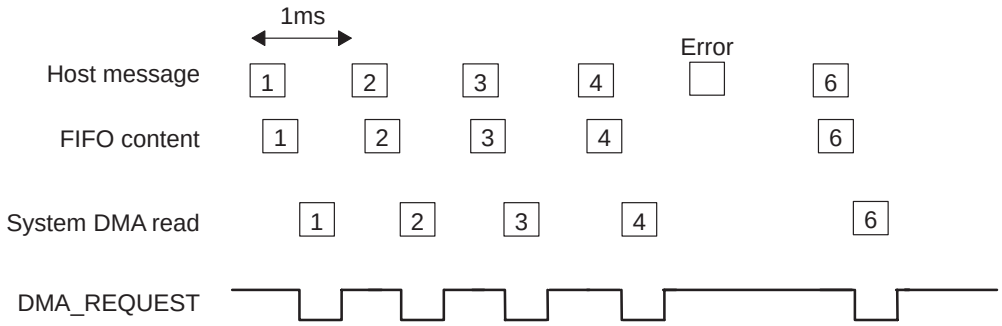
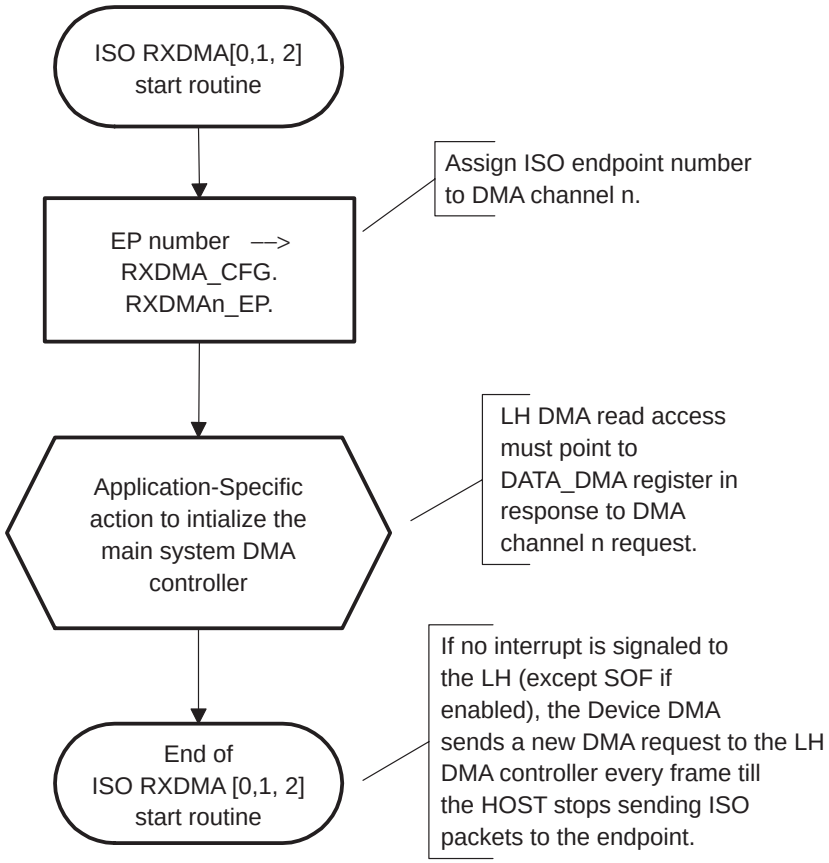


Figure 42. ISO RX DMA Start Routine



Transmit DMA Channels Overview

Transmit DMA channels are programmed via the three TXDMA control registers. Each channel can be assigned to a given endpoint number by assigning a non-zero value in TXDMA_CFG.TXDMAn_EP (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 OMAP5912 DMA controller requests DMA_REQ_29, DMA_REQ_30, and DMA_REQ_31, respectively.

Non-Isochronous IN (MPU –> USB HOST) DMA Transactions

Non-ISO (bulk) TX DMA file transfers are virtually unlimited in size. The flowcharts depicted in Figure 43 and Figure 44 show how to handle small, medium, or large file transfers.

TXDMA0, TXDMA1, and TXDMA2 registers operate for non-ISO endpoints in the following manner. The transfer size counter (TXDMAn.TXN_TSC) 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 that the START bit is set and then each time there is space free in TX FIFO for other TX packet to be written, until TXDMAn.TXn_TSC counts 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 after the last IN transaction completes successfully. This is after START bit was set and after TXDMAn.TXn_TSC equals 0 for the selected DMA channel.

The MPU 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 needs two or more DMA passes, signaled by an interrupt completion after each pass.

Figure 43 shows the necessary 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 via a DONE interrupt, whose handler is shown in Figure 44. The start routine and the associated interrupt handler are tightly coupled.

Figure 43. Non-ISO TX DMA Start Routine

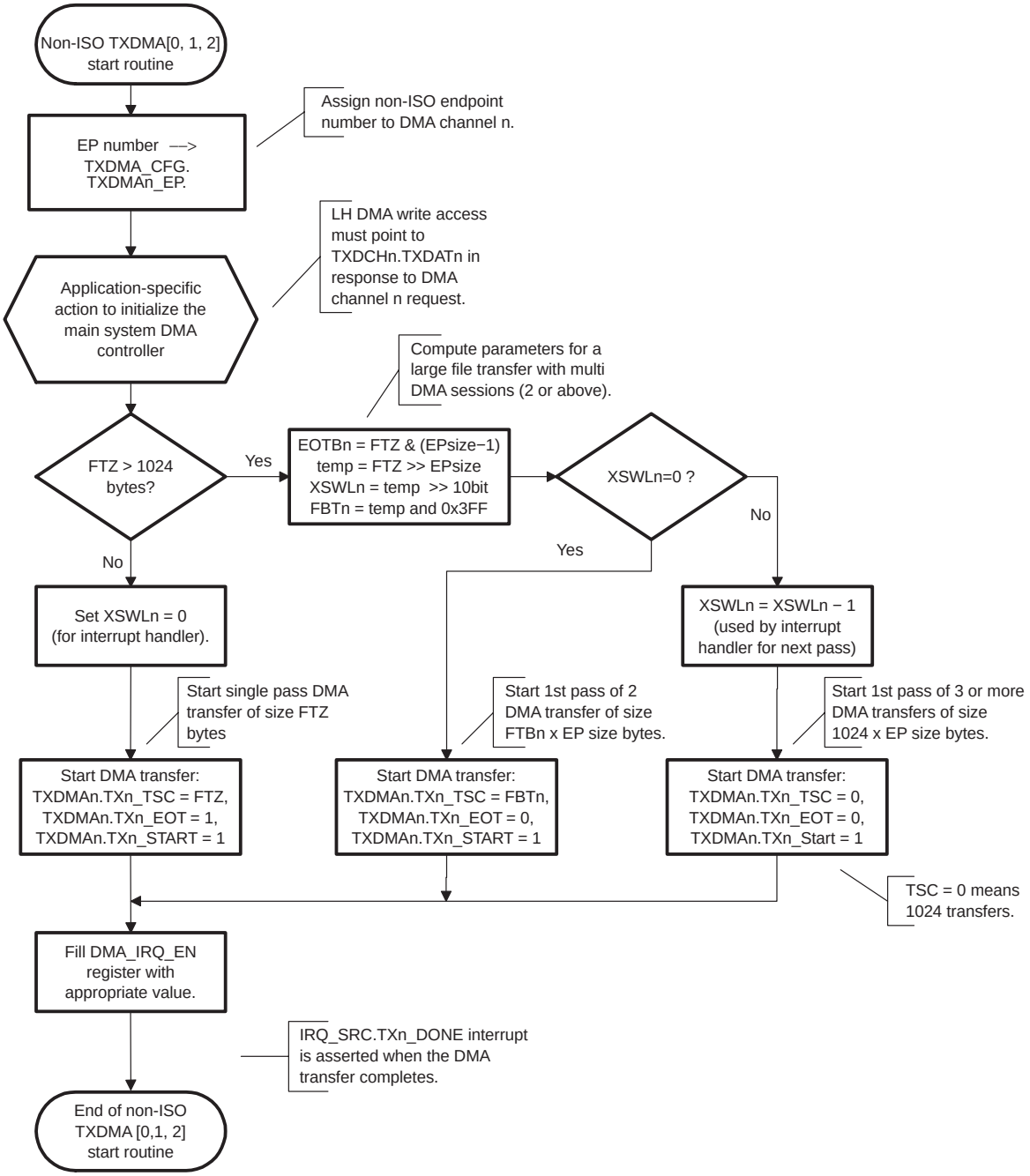
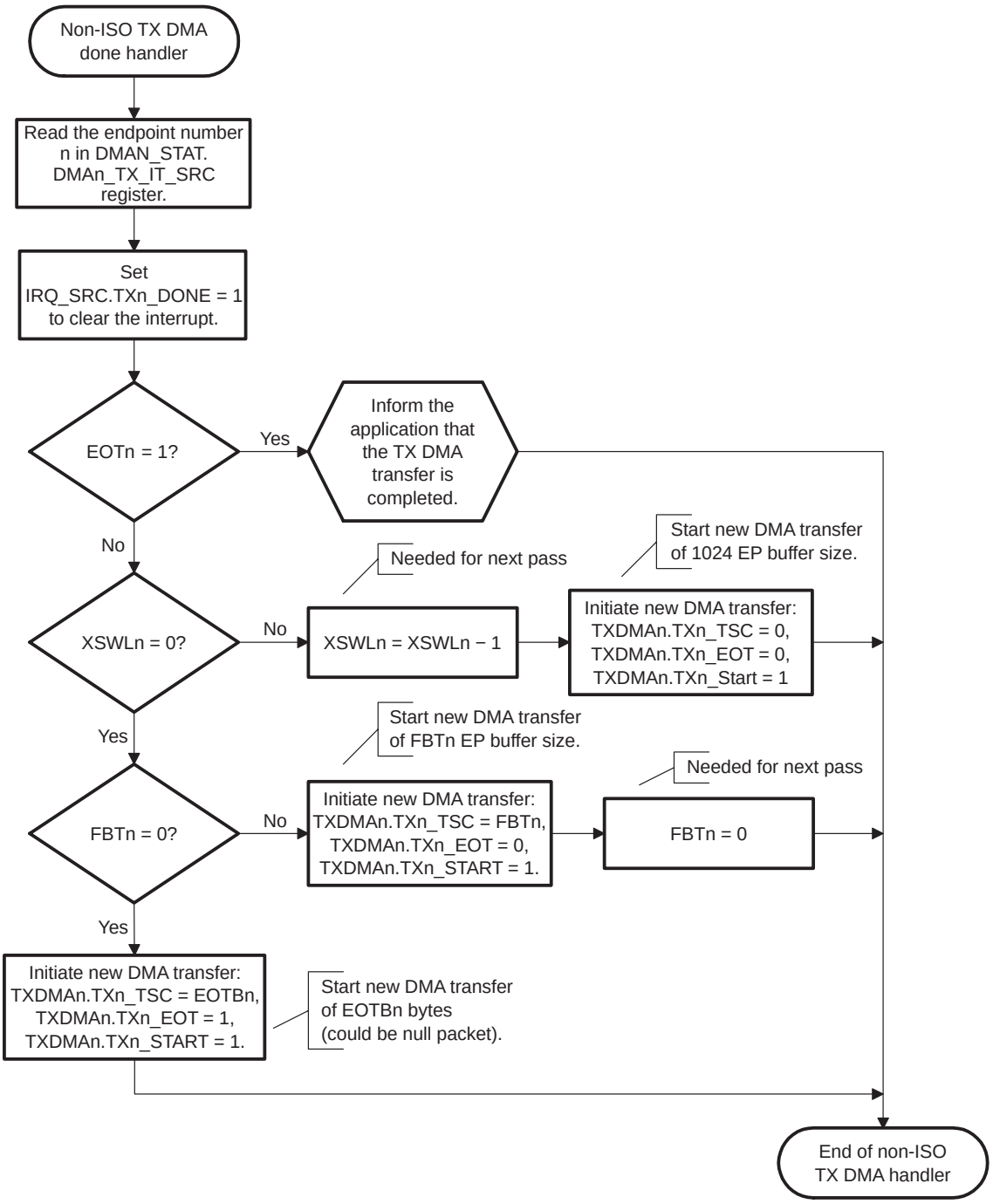


Figure 44. Non-ISO TX DMA Done Interrupt Handler



Example 1. Example: 100603 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 5 TXn_DONE interrupts, are required:

1)	EOT=0, FBT=0, loop=3	32768 bytes transferred (1024 x 32 bytes)
2)	EOT=0, FBT=0, loop=2	32768 bytes
3)	EOT=0, FBT=0, loop=1	32768 bytes
4)	EOT=0, FBT=0x47, loop=0	2272 bytes (71 x 32 bytes)
5)	EOT=1, FBT=0x1B, loop=0	27 bytes

Isochronous IN (USB HOST –> MPU) DMA Transactions

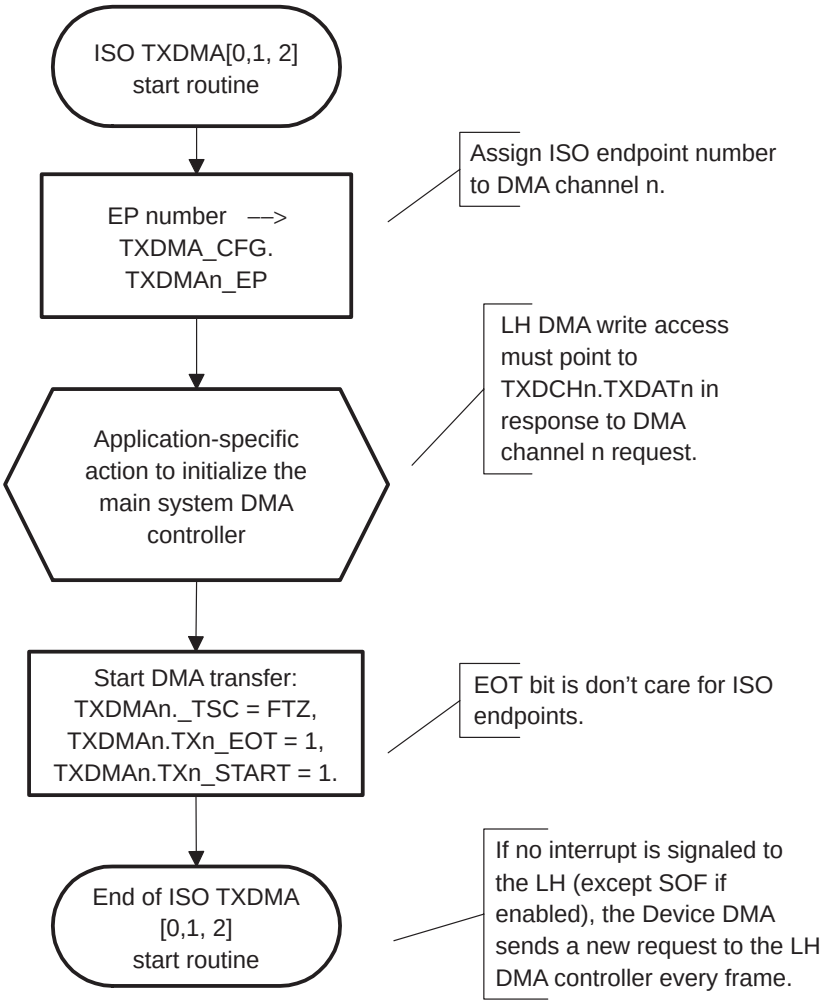
For ISO endpoints, the transfer size counter (TXDMAn.TXn_TSC) corresponds to the number of bytes to transmit. The programmed size must not
exceed the programmed buffer size of the endpoint. Otherwise, the result is unpredictable (see Figure 45).

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 TXDMAn.TXn_TSC value.

During ISO transfers to a DMA operated IN endpoint, a request to the MPU system DMA 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 during DMA operation to ISO IN endpoints.

Figure 45. ISO TX DMA Start Routine



Important Note on 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 start-of-frame if a TX DMA channel is configured for an isochronous endpoint; this
request must be serviced imperatively.

Note on DMA Channels Deconfiguration

It is recommended that the MPU wait for an EOT (RX) or a DONE (TX) interrupt before disabling the channel by writing a value 0 in the TX/RXDMA_CFG

register. However, if needed by the application, the MPU can unselect the endpoint number in the TX/RXDMA_CFG register during a DMA transfer. The resulting behavior is:

- For RX transfer:
  - If RX 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 have been read). When double-buffering is used, the deconfiguration is effective after both buffers have been read (if both buffers were not empty at unselection). An EOT interrupt is asserted if an end-of-transfer is detected.
  - If the RX DMA request is not active when unselection 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 has been handled and the TX data have been sent through an IN transaction (both buffers in case of double-buffering with both buffers full). No TX_DONE interrupt is asserted even if TXDMAn.TSC bit value 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 TX buffer(s) have been sent through IN transaction(s). If the TXDMAn_TSC value is 0 at this point, no TX_DONE interrupt is asserted. If TX_DONE interrupt had already been asserted when the endpoint is deselected, configuration is effective only after the TX_DONE interrupt handling.

TX/RXDMA_CFG.TX/RXDMAn_EP reflects the endpoint value until deconfiguration is effective. The MPU must read this register to know if the channel has been disabled yet or not. 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).

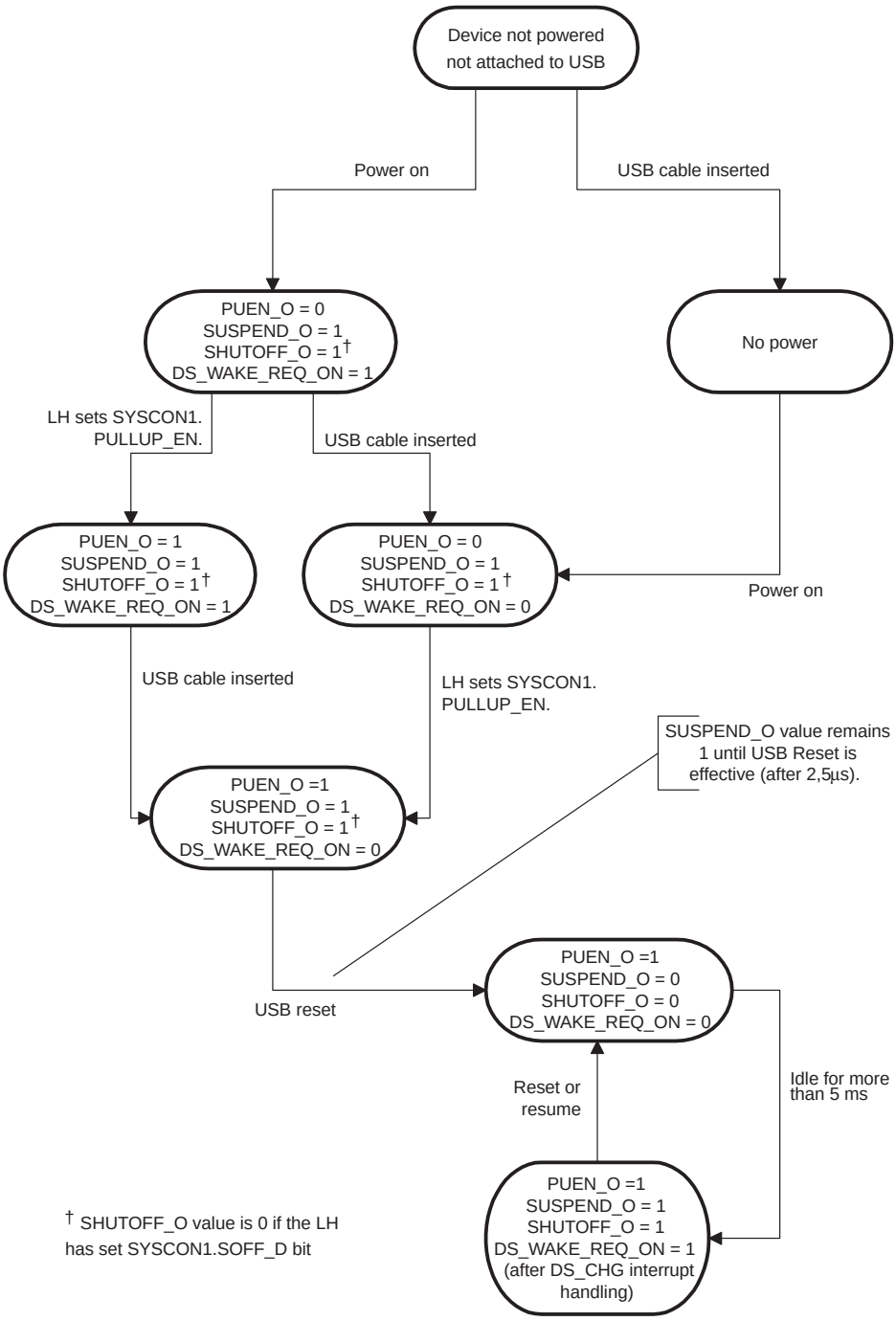
If the selected endpoint is of isochronous type, deconfiguration is effective after the TX/RX request has been serviced, and the subsequent isochronous transactions are handled at SOF interrupt through the endpoint registers (EP_NUM and STAT_FLG).

3.28 Power Management

Figure 46 shows the values assigned to the USB device controller signals concerned with power management, in the function of the device state. These signals are:

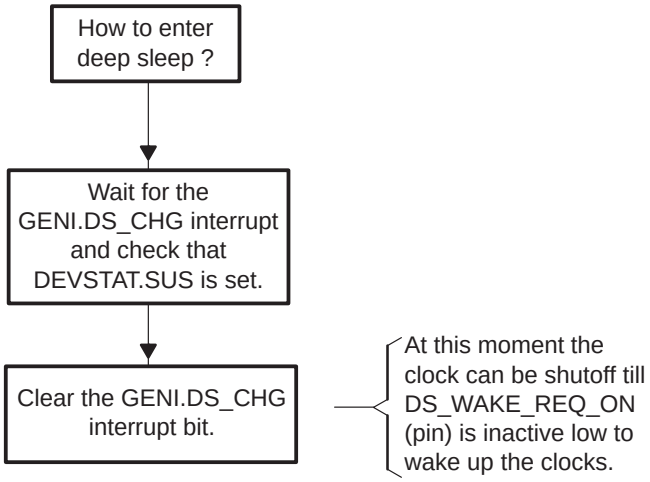
- PUEN_O: Pullup enable signal, always reflecting the SYSCON1.PULLUP_EN register bit
- SHUTOFF_O: Power circuitry shutoff signal, controlled by the core and the SYSCON1.SOFF_DIS bit
- DS_WAKE_REQ_ON: Deep-sleep wake request, asserted low when the interface clock is needed
- SUSPEND_O: Suspend signal, asserted high when the device is in suspend mode

Figure 46. Power Management Signal Values



From a software point of view, Figure 47 shows the reaction. This flowchart does not need to be implemented; it only reflects the way the module can enter deep sleep.

Figure 47. Power Management Flowchart



4 USB OTG Controller

The OMAP5912 USB OTG controller works in conjunction with the OMAP5912 USB device controller and one port of the OMAP5912 host controller to provide USB On-The-Go functionality. The OMAP5912 OTG controller includes various control and status logic that switches between the two controllers as required by the USB On-The-Go protocol. The combination allows OMAP5912 to act as a USB OTG dual-role device. The OMAP5912 OTG implementation simultaneously allows one USB OTG port and up to two additional USB host ports.

The USB OTG controller provides a dual-role device capability that is compatible with the On-The-Go supplement to the *USB 2.0 Specification* (Revision 1.0). This dual-role device can provide a 12M bit-per-second connection between OMAP5912 and other USB OTG dual-role devices. The OMAP5912 OTG solution does not support 48M bit-per-second OTG connectivity.

In addition, the USB OTG controller provides control and status information for various aspects of the OMAP5912 USB host controller and the OMAP5912 USB device controller operation when On-The-Go functionality is not enabled.

4.1 OTG Controller Features

The main features of the OMAP5912 USB OTG controller include:

- ❏ USB specification version 2.0 compatibility
- ❏ When acting as an OTG a-peripheral or an OTG b-peripheral: a 12M bit-per-second communication link with configurable data transfer type, data buffer size, single- or double-buffering for each endpoint, and 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 Rev 1.0-compatible USB host controller capable of USB full-speed (12M bits-per-second) and USB low-speed (1.5M bits-per-second) communication.

These are basic features of the OMAP5912 USB device controller and the OMAP5912 USB host controller. Additional OTG-specific features provided by the OMAP5912 OTG controller include:

- ❏ Control and status logic that allows implementation of an OTG dual-role device (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 host negation protocol (HNP) and both types of OTG session request protocol
- ❏ Support for OTG transceivers compatible with the OTG working group OTG transceiver specification

4.2 OTG Controller Registers

Table 57 lists the 32-bit OTG controller registers. Table 58 through Table 72 describe the register bits.

Table 57. OTG Controller Registers

Name	Description	R/W	Address
OTG_REV	OTG controller revision number	R/O	FFFB:0400h
OTG_SYSCON_1	OTG system configuration group 1	R/W	FFFB:0404h
OTG_SYSCON_2	OTG system configuration group 2	R/W	FFFB:0408h
OTG_CTRL	OTG control	R/W	FFFB:040Ch

Table 57. OTG Controller Registers (Continued)

Name	Description	R/W	Address
OTG_IRQ_EN	OTG interrupt enable	R/W	FFFB:0410h
OTG_IRQ_SRC	OTG interrupt source identification	R/W	FFFB:0414h
OTG_OUTCTRL	OTG interrupt source identification	R/W	FFFB:0418h
OTG_TEST	OTG interrupt source identification	R/W	FFFB:0420h
Reserved	Reserved	None	FFFB:0424h to FFFB:04FBh
VC	USB vendor code	R/O	FFFB:04FCh

All bits defined as reserved must be written by software with 0s, for preserving future compatibility. When read, any reserved bit returns 0. It is good software practice to use complete mask patterns for setting or testing bit fields individually within a register.

Table 58. OTG Revision Number Register (OTG_REV)

Bit	Name	Description
31:8		
7:0	OTG_REV_NB	OTG revision number: this 8-bit field indicates the revision number of the current OTG controller, in binary-coded digital (BCD), with the major revision number in bits 7−4, and the minor revision number in bits 3−0. This value is hardware fixed. 0x10: Revision 1.0 0x11: Revision 1.1 0x12: Revision 1.2 0x13: Revision 1.3 0x14: Revision 1.4 0x15: Revision 1.5 0x16: Revision 1.6 .... 0xF2: Revision 15.2

This read-only register reflects the OTG controller revision number.

Table 59. OTG System Configuration Register 1 (OTG_SYSCON_1)

Bit	Name	Description
31:27	Reserved	Reserved
26:24	USB2_TRX_MODE†	USB port 2 transceiver mode. These bits configure the transceiver signaling type used by the USB host, USB device, and USB OTG controllers when communicating with the transceiver connected to USB pin group 2. Transceiver signaling type depends on the signaling mode used by the transceiver. The supported transceiver signaling types are 3-pin DAT/SE0 mode bidirectional transceiver signaling, 4-pin VP/VM mode bidirectional transceiver signaling, and 6-pin DAT/SE0 unidirectional transceiver signaling (see Section 4.5, *Transceiver Signaling Types*).
23	Reserved	Reserved
22:20	USB1_TRX_MODE‡	USB port 1 transceiver mode.These bits configure the transceiver signaling type used by the USB host, USB device, and USB OTG controllers when communicating with the transceiver connected to USB pin group 1. Transceiver signaling type depends on the signaling mode used by the transceiver. The supported trans)ceiver signaling types are 3-pin DAT/SE0 mode bidirectional transceiver signaling, 4-pin VP/VM mode bidirectional transceiver signaling, and 6-pin DAT/SE0 unidirectional transceiver signaling (see Section 4.5, *Transceiver Signaling Types*).
19	Reserved	Reserved
18:16	USB0_TRX_MODE§	USB port 0 transceiver mode. These bits configure the transceiver signaling type used by the USB host, USB device, and USB OTG controllers when communicating with the transceiver connected to USB pin group 0 or USB alternate pin group 2. Transceiver signaling type depends on the signaling mode used by the transceiver. The supported transceiver signaling types are 3-pin DAT/SE0 mode bidirectional transceiver signaling, 4-pin VP/VM mode bidirectional transceiver signaling, and 6-pin DAT/SE0 unidirectional transceiver signaling (see Section 4.5, *Transceiver Signaling Types*).

† See Table 60, *Pin Group 2 Transceiver Type Selection*.
‡ See Table 61, *Pin Group 1 Transceiver Type Selection*.
§ See Table 62, *Pin Group 0 Transceiver Type Selection* and Table 63, *Alternate Pin Group 2 Transceiver Type Selection*

Table 59. OTG System Configuration Register 1 (OTG_SYSCON_1) (Continued)

Bit	Name	Description
15	OTG_IDLE_EN	OTG controller clock-gating control. This bit disables the clock to the OTG controller. 0: The OTG controller clock is not gated. Register settings in the ULPD can prevent the USB OTG controller from getting a clock even if this bit is 0. 1: The OTG controller is idled by disabling the OTG controller. Accesses to OTG controller registers are allowed, but the OTG controller clock is not gated. This register is cleared 0 by soft reset or hardware reset. For more information about the power saving circuitry, see Section 4.2.2.
14	HST_IDLE_EN	0 − Host power management circuitry disabled 1 − Host power management circuitry enabled
13	DEV_IDLE_EN	Device controller clock-gating control. This bit defines the device power-saving circuitry. 0: The USB device controller clock is not gated. Other registers in OMAP5912 can prevent the USB device controller from getting a clock even if this bit is 0. 1: The USB device controller clock is disabled if the USB device controller does not need an active clock (such as when the USB device controller is in susped). Accesses to USB device controller registers are allowed, but the USB device controller state machines does not function. If the USB device controller sees USB resume signaled when DEV_IDLE_EN is 1, a USB device controller general interrupt is generated. The interrupt service routine must clear DEV_IDLE_EN before performing the operations necessary to service the interrupt. This register is set to1 by soft reset or hardware reset. For more information about the USB device controller, see Section 3, *USB Device Controller*.
12:3	Reserved	Reserved

† See Table 60, *Pin Group 2 Transceiver Type Selection*.
‡ See Table 61, *Pin Group 1 Transceiver Type Selection*.
§ See Table 62, *Pin Group 0 Transceiver Type Selection* and Table 63, *Alternate Pin Group 2 Transceiver Type Selection*

Table 59. OTG System Configuration Register 1 (OTG_SYSCON_1) (Continued)

Bit	Name	Description
2	RESET_DONE	Reset status information: This bit reflects the reset status of the controller (hardware and software reset, internal reset monitoring of the USB device and the USB OTG controller sections). 0: Internal reset for OTG is ongoing (includes OTG controller and USB device) 1: Reset completed This register is cleared 0 by soft reset or hardware reset. This reset done information is important for proper use of the controller. After a hardware reset or a soft reset, this bit must be checked to determine that the reset has completed.This register does not reflect the reset status of the OMAP5912 USB host controller. Software must query certain USB host controller registers to determine when the USB host controller reset is complete. See Section 2.16,*USB Host Controller Hardware Reset*.
1	SOFT_RESET	Software reset for the OTG, device, and host controllers: set this bit to 1 to trigger a reset of the OTG controller, USB host controller, and USB device controller. The bit is automatically cleared by the hardware. During reads, it always returns 0. 0: Normal mode 1: The controller is reset. Software must monitor the RESET_DONE to determine when the OTG controller and the USB device controller have completed reset. Section 2.16 *USB Host Controller Hardware Reset* describes software activities required to determine when the USB host controller completes its reset.
0	Reserved	–

† See Table 60, *Pin Group 2 Transceiver Type Selection*.
‡ See Table 61, *Pin Group 1 Transceiver Type Selection*.
§ See Table 62, *Pin Group 0 Transceiver Type Selection* and Table 63, *Alternate Pin Group 2 Transceiver Type Selection*

This read-write register controls the USB transceiver interface, the USB controller clock gating, and USB controller reset generation.

Table 60. Pin Group 2 Transceiver Type Selection

USB2_TRX_MODE	OMAP5912 Pin Group 2 Transceiver Signaling
0x0	6-pin DAT/SE0 mode unidirectional signaling Normal functionality is available if USB_TRANSCEIVER_CTRL.CONF_USB2_UNI_R = 1. If USB_TRANSCEIVER_CTRL.CONF_USB2_UNI_R = 0, OMAP5912 places pins USB2.TXD and USB2_TXSE0 in high-impedance mode.

0x1	4-pin VP/VM mode bidirectional signaling
0x2	3-pin DAT/SE0 mode bidirectional signaling
0x3	6-pin DAT/SE0 mode unidirectional signaling
Other values	Reserved

This register is cleared 0x0 by soft reset or hardware reset.

When both

- ❏ USB2_TRX_MODE = 0
- ❏ USB_TRANSCEIVER_CTRL.CONF_USB2_UNI_R = 0

OMAP5912 places pins USB2.TXD and USB2.TXSE0 in high-impedance mode to prevent contention with any signals driven by the external transceiver. Mode 0 can be used for normal 6-pin DAT/SE0 mode functionality when USB_TRANSCEIVER_CTRL.CONF_USB2_UNI_R = 1.

This register does not have an effect on the transceiver signaling type when top-level pin multiplexing selects alternate pin group 2 (see the USB0_TRX_MODE subsection). When HMC_MODE selects a transceiverless link logic connection using USB pin group 2, this register must be set to 3.

Table 61. Pin Group 1 Transceiver Type Selection

USB1_TRX_MODE	OMAP5912 Pin Group 1 Transceiver Signaling
0x0	6-pin DAT/SE0 mode unidirectional signaling. Normal functionality is available if USB_TRANSCEIVER_CTRL.CONF_USB1_UNI_R = 1. If USB_TRANSCEIVER_CTRL.CONF_USB1_UNI_R = 0, OMAP5912 places pins USB1.TXD and USB1_TXSE0 in high-impedance mode.
0x1	4-pin VP/VM mode bidirectional signaling
0x2	3-pin DAT/SE0 mode bidirectional signaling
0x3	6-pin DAT/SE0 mode unidirectional signaling
Other values	Reserved

This register is cleared to 0x0 by soft reset or hardware reset.

When both:

- ❏ USB1_TRX_MODE = 0
- ❏ USB_TRANSCEIVER_CTRL.CONF_USB1_UNI_R = 0

OMAP5912 places pins USB1.TXD and USB1_TXSE0 in high-impedance mode to prevent contention with any signals driven by the external transceiver. Mode 0 can be used for normal 6-pin DAT/SE0 mode functionality when USB_TRANSCEIVER_CTRL.CONF_USB1_UNI_R = 1.

Table 62. Pin Group 0 Transceiver Type Selection

USB0_TRX_MODE	OMAP5912 Pin Group 0 Transceiver Signaling
0x0	6-pin DAT/SE0 mode unidirectional signaling.
0x1	4-pin VP/VM mode bidirectional signaling
0x2	3-pin DAT/SE0 mode bidirectional signaling
0x3	6-pin DAT/SE0 mode unidirectional signaling
Other values	Reserved

This register is cleared to 0x0 by soft reset or hardware reset.

When using the OMAP5912 integrated USB transceiver, USB0_TRX_MODE is set to 3. Software can avoid USB0_TRX_MODE = 0 to improve software compatibility with future devices.

When OMAP5912 top-level signal multiplexing selects alternate pin group 2, the USB0_TRX_MODE signal affects the transceiver type for the transceiver connected to the following OMAP5912 pins:

- MCSI2.DOUT/USB0.TXEN
- UART2.TX/USB0.TXD
- UART1.RTS/USB0.SE0
- UART2.CTS/USB0.RCV
- MCSI2.DIN/USB0.VP
- UART2.RX/USB0.VM

Table 63. Alternate Pin Group 2 Transceiver Type Selection

USB0_TRX_MODE	OMAP5912 Alternate Pin Group 2 Transceiver Signaling
0x0	6-pin DAT/SE0 mode unidirectional signaling.
0x1	4-pin VP/VM mode bidirectional signaling
0x2	3-pin DAT/SE0 mode bidirectional signaling
0x3	6-pin DAT/SE0 mode unidirectional signaling
Other values	Reserved

Table 64. OTG System Configuration Register 2 (OTG_SYSCON_2)

Bit	Name	Description
31	OTG_EN	The OTG enable bit selects the OTG controller functionality: 0: The OTG controller circuitry is not activated. The USB device controller and the USB host controller can be used, but an OTG link cannot be implemented. 1: The OTG controller circuitry is activated. The OTG state machine (HNP) can be used on one of the USB ports. The USB device controller and one of the USB host controller ports can be used to implement an OTG link. This register is cleared 0 by soft reset or hardware reset. When an OTG link is not needed, this bit can be cleared and OTG_ SYSCON_1.OTG_IDLE_EN can be set to reduce power consumption. A driver switch interrupt is internally generated whenever a 1 is written to OTG_EN. If OTG_IRQ_EN.DRIVER_SWITCH_EN is 1, this internal interrupt propagates to the MPU level 2 interrupt controller.
30	USBx_SYNCHRO	The USB output signals synchronized bit must be set to 1 to allow proper signal timing on OMAP5912 USB pins. This register is cleared 0 by soft reset or hardware reset.
29	OTG_MST16	This bit enables compatibility with the 16-bit OTG master controller, and it must be set to 0 to allow proper USB host controller access to OMAP5912 system memory. 0: Host controller is connected to system memory via a 32-bit bus. 1: Host controller is connected to system memory via a 16-bit bus. OMAP5912 does not support this setting. This register is cleared to 0 by soft reset or hardware reset.
28	SRP_GPDATA	Session request protocol generation pulse width on D+. This bit allows the OTG controller to extend the duration of the D+ pulse during the data line pulsing portion of a session request protocol activity. This bit is only used when the OTG controller is configured to act as an OTG default B device. 0: Data line pulse during SRP is 6 ms (nominal). 1: Data line pulse during SRP is 9 ms (nominal). This register is cleared 0 by soft reset or hardware reset. When the OTG controller performs the SRP process, it first generates a data line pulse SRP request and then a VBUS pulse SRP request.

† See Table 65, *OTG_PADEN Source Status*.
‡ See Table 66, *HMC_PADEN: USB Signal Multiplexing Control Source*.

Table 64. OTG System Configuration Register 2 (OTG_SYSCON_2) (Continued)

Bit	Name	Description
27	SRP_GPDVBUS	Session request protocol generation discharge VBUS enable. This bit enables the OTG controller to provide a VBUS discharge sequence as part of its SRP generation activities. This bit enables the VBUS discharge functionality after any VBUS charge process. When enabled, the duration of the VBUS discharge is defined by SRP_GPUVBUS. 0: VBUS is not discharged after the SRP VBUS pulse. 1: VBUS is discharged after the SRP VBUS pulse. This register is cleared 0 by soft reset or hardware reset.

† See Table 65, *OTG_PADEN Source Status*.
‡ See Table 66, *HMC_PADEN: USB Signal Multiplexing Control Source*.

Table 64. OTG System Configuration Register 2 (OTG_SYSCON_2) (Continued)

Bit	Name	Description
26:24	SRP_GPUVBUS	Session request protocol generation charge VBUS duration.This bit controls the duration that the OTG controller attempts to charge VBUS when acting as a default-B OTG device and requesting SRP. 0x0: OMAP5912 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 .5 ms nominal. 0x3: During SRP generation, VBUS is charged for 2 ms nominal. 0x4: During SRP generation, VBUS is charged for 4 ms nominal. 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. This register is cleared 000 by soft reset or hardware reset. The current sourcing capabilities of the hardware that drives the SRP VBUS pulse determines the required duration of the VBUS pulse. The value chosen must ensure that VBUS rises higher than 2 V within the selected VBUS pulse duration when attached to an OTG dual-role device. The value chosen must also ensure that VBUS does not rise higher than 2 V within the selected VBUS pulse duration when attached to a non-OTG port, such as a USB host port or a downstream port of a USB hub. SRP_GPUVBUS also controls the duration of a VBUS discharge pulse when SRP_GPDBVBUS is set to 1. If SRP_GPUVBUS is 0x0, the OTG controller does not request a VBUS discharge. The minimum time for SRP signalling completion is (the duration of the data line pulse defined by SRP_GPDATA) + (2 * VBUS pulse time defined by SRP_GPUVBUS). Disabling the VBUS discharge does not affect that minimum time. The interrupt OTG_IRQ_SRC:B_SRP_DONE is generated upon completion of the entire SRP.
23	Reserved	Reserved

† See Table 65, *OTG_PADEN Source Status*.
‡ See Table 66, *HMC_PADEN: USB Signal Multiplexing Control Source*.

Table 64. OTG System Configuration Register 2 (OTG_SYSCON_2) (Continued)

Bit	Name	Description
22:20	A_WAIT_VRISE	Offset for A_WAIT_VRISE timer.These bits define the maximum duration that the OTG controller must wait for VBUS to rise above the A-device VBUS valid threshold. A_WAIT_VRISE is used only when OMAP5912 acts as a default-A device. The A_WAIT_VRISE counter begins counting upon transition from the A_IDLE state. (See the *On-The-Go Supplement to the USB 2.0 Specification Revision 1.0* description of the dual-role A-device state diagram). 0x0: Maximum delay in A_WAIT_VRISE state is 200 ms nominal. 0x1: Maximum delay in A_WAIT_VRISE state is 287.04 ms nominal. 0x2: Maximum delay in A_WAIT_VRISE state is 374.08 ms nominal. 0x3: Maximum delay in A_WAIT_VRISE state is 548.16 ms nominal. This register is cleared 0x0 by soft reset or hardware reset. If the A_WAIT_VRISE counter expires before VBUS rises above the A-device VBUS valid threshold, it indicates that there is a heavy load on VBUS. The OTG controller state machine transitions to A_VBUS_ERR and generates a VBUS error interrupt.
19	Reserved	Reserved
18:16	B_ASE0_BRST	Offset for B_ASE0_BRST timer. These bits control how the USB OTG controller manages disabling the D+ pullup when acting as a default-B OTG dual-role device and transitioning from state b_peripheral to b_wait_acon, and when acting as a default-A OTG dual-role device and transitioning from state a_peripheral to a_wait_bcon. The setting of this register controls the mechanism by which the OMAP5912 OTG controller knows that the D+ pullup is disabled. Because OMAP5912 implementations control the D+ pullup using an I^2C link to an OTG transceiver, the only allowed setting is 0x4. When in this mode of operation, system software must write a 1 to OTG_CTRL.OTG_PU immediately after completion of the I^2C operation that disables the D+ pullup. This field is cleared 0x0 by soft reset or hardware reset. This field has no effect when OTG is disabled. This field must not be changed when OTG functionality is enabled (OTG_SYSCON_2.OTG_EN=1).
15	Reserved	Reserved

† See Table 65, *OTG_PADEN Source Status*.
‡ See Table 66, *HMC_PADEN: USB Signal Multiplexing Control Source*.

Table 64. OTG System Configuration Register 2 (OTG_SYSCON_2) (Continued)

Bit	Name	Description
14	SRP_DPW	Session request protocol detection—pulse width. This bit selects the width for the SRP detection when acting as a default-A OTG dual-role device. This value applies on both DATA and VBUS pulsing detections. 0: SRP pulse must be greater than 1 ms (nominal) to be sensed as a valid D+, D−, or VBUS SRP pulse. 1: SRP pulse must be greater than 167 ns (nominal) to be sensed as a valid D+, D−, or VBUS SRP pulse. This register is cleared 0 by soft reset or hardware reset. This bit has no effect when the OTG feature is disabled (OTG_SYSCON_2.OTG_EN = 0).
13	SRP_DATA	Session request protocol detection—data pulsing detection enable. This bit determines whether the OMAP5912 OTG controller considers D+ or D− pulses as SRP requests. 0: OTG controller does not consider D+ or D− pulses as SRP requests. 1: OTG controller treats a D+ or D− pulse of appropriate minimum duration as an SRP request. This register is cleared 0 by soft reset or hardware reset. This register is used only when the OMAP5912 OTG controller acts as a default-A dual-role device. SRP detection requires that either OTG_SYSCON_2.SRP_DATA or OTG_SYSCON_2.SRP_VBUS (or both) is set.
12	SRP_VBUS	Session request protocol detection—VBUS pulsing detection enable.This bit determines whether the OMAP5912 OTG controller considers VBUS pulses as SRP requests. 0: OTG controller does not consider VBUS pulses as SRP requests. 1: OTG controller treats a VBUS pulse of appropriate minimum duration as an SRP request. This register is cleared 0 by soft reset or hardware reset. This register is only used when the OMAP5912 OTG controller acts as a default-A dual-role-device. SRP detection requires that either OTG_SYSCON_2.SRP_DATA or OTG_SYSCON_2.SRP_VBUS (or both) is set.
11	Reserved	Reserved

† See Table 65, *OTG_PADEN Source Status*.
‡ See Table 66, *HMC_PADEN: USB Signal Multiplexing Control Source*.

Table 64. OTG System Configuration Register 2 (OTG_SYSCON_2) (Continued)

Bit	Name	Description
10	OTG_PADEN†	OTG transceiver control and status information selector. This bit determines how OTG_CTRL register bits ID, VBUSVLD, BSESSVLD, BSESSEND, and ASESSEND are controlled. When using OMAP5912 OTG controller functionality to implement an On-The-Go dual-role device, this bit must be set to 0. In this case, those OTG_CTRL bits are read/write bits and are controlled by system software. When using OMAP5912 USB device controller functionality without enabling OTG functionality, this bit must be set to 1. In this case, OTG_CTRL register bits ID, VBUSVLD, BSESSEND, and ASESSEND are held at 0, and OTG_CTRL bit BSESSVLD is controlled by the OMAP5912 GPIO0/USB.VBUS pin (if top-level pin multiplexing selects the USB_VBUS mode for pin GPIO0). Software writes to those OTG_CTRL bits are ignored when OTG_PADEN = 1.
9	HMC_PADEN‡	USB pin multiplexing control selector. This bit determines whether USB signal multiplexing is controlled by registers in OTG_SYSCON_2 or by registers in the OMAP5912 configuration register module. 0 = OTG_SYSCON_2 provides the controls. 1 = OMAP5912 configuration register module registers provide the controls.
8	UHOST_EN	The host USB controller enable bit controls the clock enable and hardware reset for the OMAP5912 USB host controller when HMC_PADEN is 0. When HMC_PADEN is 1, writes to this bit have no effect on the clock enable. 0: USB host controller is not clocked and is held in hardware reset. 1: USB host controller clock is not held inactive and a USB host controller hardware reset is not forced. The USB host controller is clocked if the ULPD 48-MHz clock output to the USB controllers is active and if the OMAP5912 peripheral reset is inactive. This register is cleared 1 by soft reset or hardware reset. A read of this register gives the internal value used for UHOST_EN regardless of the setting of HMC_PADEN. See Section 15.1.20 USB host controller hardware reset for information on USB host controller access restrictions relating to UHOST_EN.

† See Table 65, *OTG_PADEN Source Status*.
‡ See Table 66, *HMC_PADEN: USB Signal Multiplexing Control Source*.

Table 64. OTG System Configuration Register 2 (OTG_SYSCON_2) (Continued)

Bit	Name	Description
7	HMC_TLLSPEED	The HMC TLL SPEED configuration bit controls the way that the OMAP5912 transceiverless link logic models the speed of the transceiverless link when HMC_PADEN is 0. When HMC_PADEN is 1, this bit has no effect on the transceiverless link logic and writes to this bit have no effect. 0: Transceiverless link logic models a low-speed (1.5M bits-per-second) USB link. 1: Transceiverless link logic models a full-speed (12M bits-per-second) USB link. This register is cleared 0 by soft reset or hardware reset. 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.
6	HMC_TLLATTACH	The HMC TLL ATTACH configuration bit controls the OMAP5912 transceiverless link logic attach/detach status when HMC_PADEN is 0. When HMC_PADEN is 1, this bit has no effect on the transceiverless link logic and writes to this bit have no effect. 0: Transceiverless link logic models a detached link. 1: Transceiverless link logic models an attached link. This register is cleared 0 by soft reset or hardware reset. 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.
5:0	HMC_MODE	The HMC mode configuration bits control the OMAP USB signal multiplexing selection when HMC_PADEN is 0. When HMC_PADEN is 1 this field is unused and writes to this register have no effect. HMC_MODE values are discussed in Section 4.3, *Pin Multiplexing*. This register is cleared 0x0 by soft reset or hardware reset.

† See Table 65, *OTG_PADEN Source Status*.
‡ See Table 66, *HMC_PADEN: USB Signal Multiplexing Control Source*.

This read-write register provides control and status for various aspects of OTG controller, USB pin multiplexing, and USB host controller functionality.

Table 65. OTG_PADEN Source Status

OTG_PADEN	Value	Source
0	ID	OTG_CTRL.ID
	VBUSVLD	OTG_CTRL.VBUSVLD
	BSESSEND	OTG_CTRL.BSESSEND
	ASESSEND	OTG_CTRL.ASESSEND
	BSESSVLD	OTG_CTRL.BSESSVLD
	VBUS value to USB device controller	
1	ID	Tied to 0
	VBUSVLD	Tied to 0
	BSESSEND	Tied to 0
	ASESSEND	Tied to 0
	BSESSVLD	Follows GPIO0/USB.VBUS if top-level pin multiplexing selects the USB.VBUS mode for pin GPIO_0. Is tied to 0 if top-level pin multiplexing selects a mode other than USB.VBUS for pin GPIO0.
	VBUS value to USB device controller	BSESSVLD

This register is cleared 0 by soft reset or hardware reset.

Table 66. HMC_PADEN: USB Signal Multiplexing Control Source

HMC_PADEN	Value	Source
0	HMC_MODE	OTG_SYSCON_2.HMC_MODE
	HMC_TLLATTACH	OTG_SYSCON_2.HMC_TLLATTACH
	HMC_TLLSPEED	OTG_SYSCON_2.HMC_TLLSPEED
	UHOST_EN	OTG_SYSCON_2.UHOST_EN

Table 66. HMC_PADEN: USB Signal Multiplexing Control Source (Continued)

HMC_PADEN	Value	Source
1	HMC_MODE	MOD_CONF_CTRL_0. CONF_MOD_USB_HOST_HHC_HMC_MODE_R
	HMC_TLLATTACH	MOD_CONF_CTRL_0. CONF_MOD_USB_HOST_HHC_HMC_TLL_ATTACH_R
	HMC_TLLSPEED	MOD_CONF_CTRL_0. CONF_MOD_USB_HOST_HHC_HMC_TLL_SPEED_R
	UHOST_EN	MOD_CONF_CTRL_0.CONF_MOD_USB_HOST_HHC_UHOST_ EN_R

This register is cleared 0 by soft reset or hardware reset.

For best software compatibility with future devices, it is recommended that system software set this register to 0.

Table 67. OTG Control Register (OTG_CTRL)

Bit	Name	Description
31:21	Reserved	Reserved
20	ASESSVLD	Current A-device session valid value. When OTG is enabled (OTG_EN = 1), this bit must be programmed to reflect the OTG transceiver VBUS voltage comparator that compares against $V_{A\_SESS\_VLD}$. When VBUS is above $V_{A\_SESS\_VLD}$, ASESSVLD must be set to 1. When VBUS is below $V_{A\_SESS\_VLD}$, ASESSVLD should be set to 0. This information is only relevant when connected as an A-device (OTG_CTRL.ID = 0) and OTG is enabled (OTG_EN = 1). 0: VBUS voltage is below $V_{A\_SESS\_VLD}$. 1: VBUS voltage is above $V_{A\_SESS\_VLD}$. This register is cleared 0x0 by soft reset or hardware reset.
19	BSESSEND	Current B-device session end value. When OTG is enabled (OTG_EN = 1), this bit must be programmed to reflect the OTG transceiver VBUS voltage comparator that compares against $V_{B\_SESS\_END}$. When VBUS is below $V_{B\_SESS\_END}$, BSESSEND must be set to 1. When VBUS is above $V_{B\_SESS\_END}$, BSESSEND 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). 0: VBUS voltage is above $V_{B\_SESS\_END}$. 1: VBUS voltage is below $V_{B\_SESS\_END}$. This register is cleared 0x0 by soft reset or hardware reset.

Table 67. OTG Control Register (OTG_CTRL) (Continued)

Bit	Name	Description
18	BSESSVLD	Current B-device session valid value. When OTG is enabled (OTG_EN = 1) and OMAP5912 is connected as a dual-role B-device (OTG_CTRL.ID = 1), this bit must be programmed to reflect the OTG transceiver VBUS voltage comparator that compares against $V_{B\_SESS\_VLD}$. When VBUS is above $V_{B\_SESS\_VLD}$, BSESSVLD must be set to 1. When VBUS is below $V_{B\_SESS\_VLD}$, BSESSVLD must be set to 0. 0: VBUS voltage is below $V_{B\_SESS\_VLD}$. 1: VBUS voltage is above $V_{B\_SESS\_VLD}$. This register is cleared 0x0 by soft reset or hardware reset. 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 OMAP5912 USB device controller to function properly. 0: VBUS voltage is below $V_{A\_VBUS\_VLD}$ (if OTG is enabled), or VBUS is insufficient to allow OMAP5912 USB device controller functionality (if OTG is disabled and OTG_PADEN is 0). 1: VBUS voltage is above $V_{A\_VBUS\_VLD}$ (if OTG is enabled), or VBUS is sufficient to allow OMAP5912 USB device controller functionality (if OTG is disabled and OTG_PADEN is 0).
17	VBUSVLD	Current VBUS valid value. When OTG is enabled (OTG_EN = 1), this bit must be programmed to reflect the OTG transceiver VBUS voltage comparator that compares against $V_{A\_VBUS\_VLD}$. When VBUS is above $V_{A\_VBUS\_VLD}$, VBUSVLD must be set to 1. When VBUS is below $V_{A\_VBUS\_VLD}$, VBUSVLD must be set to 0. This bit has no meaning when acting as an OTG default-B device (OTG ID = 1 and OTG_EN = 1). This register is cleared 0x0 by soft reset or hardware reset.
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) this bit has no meaning. 0: ID pin is grounded. 1: ID pin is high-impedance. This register is cleared 0x0 by soft reset or hardware reset. 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.

Table 67. OTG Control Register (OTG_CTRL) (Continued)

Bit	Name	Description
15	DRIVER_SEL	Active controller/driver software. When OTG is enabled (OTG_EN = 1), this read-only bit determines which OMAP5912 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), this bit 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. 0: Host driver has control of the OTG link. 1: Device driver has control of the OTG link. This bit is set to 1 by soft reset or hardware reset.
14:13	Reserved	Reserved
12	A_SETB_HNPEN	B-device HNP indication (when acting as default-A device). When OTG is enabled (OTG_EN = 1) and OMAP5912 acts as a 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. This bit has no effect when OMAP5912 acts as a default-B device (OTG_CTRL.ID=1) or when OTG is disabled (OTG_EN = 0). 0: The B-device has not been enabled to issue HNP. The OMAP5912 OTG controller does not respond to HNP issued by the B-device. 1: The B-device has been enabled to issue HNP. The OMAP5912 OTG controller responds to HNP issued by the B-device. This bit is cleared 0 by soft reset or hardware reset.
11	A_BUSREQ	Bus request (when acting as default-A device). When OTG is enabled (OTG_EN = 1) and OMAP5912 is acting as a 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 OMAP5912 USB host controller. This bit has no effect if OTG_EN = 0 or if OTG_CTRL.ID = 1. 0: 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. 1: A-device requests host role on the bus. OTG state machine begins a session (if the OTG session is ended) or attempts to return to ownership of the OTG link to the OMAP5912 USB host controller as soon as possible. This bit is cleared 0 by soft reset, hardware reset, or when the A_REQ_TMROUT interrupt is set.

Table 67. OTG Control Register (OTG_CTRL) (Continued)

Bit	Name	Description
10	Reserved	Reserved
9	B_HNPEN	B-device HNP enable (when acting as default-B device). When OTG is enabled (OTG_EN = 1) and OMAP5912 is acting as a 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. This bit has no meaning when OTG_EN = 0 or when OTG_CTRL.ID = 0. 0: B-device is not HNP enabled. The OTG controller does not issue HNP. 1: B-device is HNP enabled. The OTG controller can issue HNP when B_BUSREQ is set. This bit is cleared 0 by soft reset or hardware reset.
8	B_BUSREQ	Bus request (when acting as default-B device). When OTG is enabled (OTG_EN = 1) and OMAP5912 is acting as a 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 OTG_CTRL.ID = 0. 0: System software does not currently wish to begin acting as host. 1: System software (acting as a default-B device) wishes to begin acting as host. HNP and, if necessary, SRP are issued. This bit is cleared 0 by soft reset or hardware reset. 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.
7	OTG_BUSDROP	OTG bus drop request. When OTG is enabled (OTG_EN = 1) and OMAP5912 is acting as a 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 OTG_CTRL.ID = 1. 0: System software does not require the end of the OTG session. 1: System software requires that the OTG session be ended. This bit is cleared 0 by soft reset or hardware reset.
6	Reserved	Reserved

Table 67. OTG Control Register (OTG_CTRL) (Continued)

Bit	Name	Description
5	OTG_PD	D+ pulldown enable. When OTG is enabled (OTG_EN = 1), this read-only register indicates whether the OTG transceiver applies a pulldown to D+. When OTG is disabled, this bit has no meaning. 0: OTG transceiver does not activate the D+ pulldown. 1: OTG transceiver activates the D+ pulldown. This bit is cleared 0 by soft reset or hardware reset. This register can be polled when the OTG_IRQ_EN:OPRT_CHG_EN = 0. When OTG_IRQ_EN.OPRT_CHG_EN = 1 and OTG_IRQ_SRC:OPRT_CHG = 1, OTG_PD retains its value until after OTG_IRQ_SRC.OPRT_CHG is cleared.
4	OTG_PU	D+ pullup enable. When OTG is enabled (OTG_EN = 1), this register indicates whether the OTG transceiver applies a pullup to D+. When OTG is disabled, this bit has no meaning. 0: OTG transceiver disables the D+ pullup. 1: OTG transceiver activates the D+ pullup. This bit is cleared 0 by soft reset or hardware reset. This bit can be polled when the OTG_IRQ_EN:OPRT_CHG_EN = 0. When OTG_IRQ_EN.OPRT_CHG_EN = 1 and OTG_IRQ_SRC:OPRT_CHG =1, OTG_PU retains its value until after OTG_IRQ_SRC.OPRT_CHG is cleared. When OTG_SYSCON_2.B_ASE0_RST = 4 and acting as default-B device (OTG_CTRL.ID = 1), and system software is performing HNP, the software must write a 1 to this register when it knows that the OTG controller D+ pullup is active (typically upon completion of I^2C activity).
3	OTG_DRV_VBUS	VBUS drive enable. When OTG is enabled (OTG_EN = 1), this read-only bit indicates whether the OTG transceiver drives VBUS. When OTG is disabled (OTG_EN = 0), this bit has no meaning. Driving VBUS is requested only when OMAP5912 acts as a default-A device and an OTG session is needed. 0: OTG transceiver does not drive VBUS. 1: OTG transceiver drives VBUS. This bit is cleared 0 by soft reset or hardware reset. This bit can be polled when the OTG_IRQ_EN:OPRT_CHG_EN = 0. When OTG_IRQ_EN.OPRT_CHG_EN = 1 and OTG_IRQ_SRC:OPRT_CHG = 1, OTG_DRV_VBUS retains its value until after OTG_IRQ_SRC.OPRT_CHG is cleared.

Table 67. OTG Control Register (OTG_CTRL) (Continued)

Bit	Name	Description
2	OTG_PD_VBUS	VBUS pulldown enable. When OTG is enabled (OTG_EN = 1), this read-only bit indicates whether the OTG transceiver is to discharge VBUS when generating SRP. Driving VBUS is requested only when OMAP5912 acts as a default-A device and an OTG session is needed or is in progress. When OTG is disabled (OTG_EN = 0), this bit has no meaning. Discharge of VBUS is requested only when OMAP5912 is acting as a default-B device, an OTG session needs to be requested, and OTG_SYSCON_2.SRP_GPDVBUS = 1. 0: OTG transceiver does not discharge VBUS. 1: OTG transceiver discharges VBUS. This bit is cleared 0 by soft reset or hardware reset. This bit can be polled when the OTG_IRQ_EN:OPRT_CHG_EN = 0. When OTG_IRQ_EN.OPRT_CHG_EN = 1 and OTG_IRQ_SRC:OPRT_CHG =1, OTG_PD_VBUS retains its value until after OTG_IRQ_SRC.OPRT_CHG is cleared.
1	OTG_PU_VBUS	VBUS pullup enable. When OTG is enabled (OTG_EN = 1), this read-only bit indicates whether the OTG transceiver charges VBUS. Charging VBUS is only used when OMAP5912 acts as a default-B device and is performing SRP. When OTG is disabled (OTG_EN = 0), this bit has no meaning. 0: OTG transceiver does not charge VBUS. 1: OTG transceiver charges VBUS. This bit is cleared 0 by soft reset or hardware reset. This bit can be polled when the OTG_IRQ_EN:OPRT_CHG_EN = 0. When OTG_IRQ_EN.OPRT_CHG_EN = 1 and OTG_IRQ_SRC:OPRT_CHG =1, OTG_DRV_VBUS retains its value until after OTG_IRQ_SRC.OPRT_CHG is cleared.
0	OTG_PU_ID	ID signal pullup enable. When OTG is enabled (OTG_EN = 1), this read-only bit 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 I^2C interfaces ignore this bit. 0: OTG transceiver does not apply a pullup to ID. 1: OTG transceiver applies a pullup to ID. This bit is cleared 0 by soft reset or hardware reset. This register can be used for polling when the 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.

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 bits OTG_PU_ID, OTG_PU_VBUS, OTG_PD_VBUS, OTG_DRV_VBUS, OTG_PU, and OTG_PD 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 handler that updates the OTG transceiver control registers when the OPRT_CHG interrupt occurs.

The bits ID, VBUSVLD, BSESSVLD, BSESSEND, and ASESSVLD 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 OMAP5912 GPIO input that is configured as an MPU interrupt source. The interrupt service routine for the transceiver interrupt must query the OTG transceiver interrupt and appropriate status bits via I^2C operations and update the status bits in OTG_CTRL[31:27].

The bits OTG_BUSDROP, B_BUSREQ, B_HNPEN, A_BUSREQ, and A_SETB_HNPEN control the OTG controller state machines and provide functionality defined in the OTG supplement. System software must manage these bits as described below.

The DRIVER_SEL bit shows whether the USB host controller or the USB device controller has control of the OTG link.

Table 68. OTG Interrupt Enable Register (OTG_IRQ_EN)

Bit	Name	Description
15	DRIVER_SWITCH_EN	Driver switch interrupt enable. This bit enables interrupt generation when the OTG_IRQ_SRC: DRIVER_SWITCH bit is active. 0: No interrupt is generated. 1: An interrupt is generated if the OTG_IRQ_SRC: DRIVER_SWITCH bit is set. This bit is cleared to 0 by soft reset or hardware reset.
14	Reserved	Reserved

Table 68. OTG Interrupt Enable Register (OTG_IRQ_EN) (Continued)

Bit	Name	Description
13	A_VBUS_ERR_EN	A-device Vbus error interrupt enable: This bit enables interrupt generation when the OTG_IRQ_SRC: A_VBUS_ERr bit is active. 0: No interrupt is generated. 1: An interrupt is generated if the OTG_IRQ_SRC: A_VBUS_ERR bit is set. This bit is cleared to 0 by soft reset or hardware reset.
12	A_REQ_TMROUT_EN	A-device request time-out interrupt enable. This bit enables generation of an interrupt when the OTG_IRQ_SRC: A_REQ_TMROUT bit is active. 0: No interrupt is generated. 1: An interrupt is generated if the OTG_IRQ_SRC: A_REQ_TMROUT bit is set. This bit is cleared 0 by soft reset or hardware reset.
11	A_SRP_DETECT_EN	A-device SRP detection interrupt enable. This bit enables interrupt generation when the OTG_IRQ_SRC: A_SRP_DETECT bit is active. 0: No interrupt is generated. 1: An interrupt is generated if the OTG_IRQ_SRC: A_SRP_DETECT bit is set. This bit is cleared to 0 by soft reset or hardware reset.
10	B_HNP_FAIL_EN	B-device HNP failed interrupt enable. This bit enables interrupt generation when the OTG_IRQ_SRC:B_HNP_FAIL bit is active. 0: No interrupt is generated. 1: An interrupt is generated if the OTG_IRQ_SRC:B_HNP_FAIL bit is set. This bit is cleared to 0 by soft reset or hardware reset.
9	B_SRP_TMROUT_EN	B-device SRP time-out interrupt enable. This bit enables interrupt generation when the OTG_IRQ_SRC:B_SRP_TMROUT bit is active. 0: No interrupt is generated. 1: An interrupt is generated if the OTG_IRQ_SRC:B_SRP_TMROUT bit is set. This bit is cleared to 0 by soft reset or hardware reset.

Table 68. OTG Interrupt Enable Register (OTG_IRQ_EN) (Continued)

Bit	Name	Description
8	B_SRP_DONE_EN	B-device SRP done interrupt enable.This bit enables interrupt generation when the OTG_IRQ_SRC:B_SRP_DONE bit is active. 0: No interrupt is generated. 1: An interrupt is generated if the OTG_IRQ_SRC:B_SRP_DONE bit is set. This bit is cleared 0 by soft reset or hardware reset.
7	B_SRP_STARTED_EN	B-device SRP started interrupt enable. This bit enables interrupt generation when the OTG_IRQ_SRC:B_SRP_STARTED bit is active. 0: No interrupt is generated. 1: An interrupt is generated if the OTG_IRQ_SRC:B_SRP_STARTED bit is set. This bit is cleared 0 by soft reset or hardware reset.
6:1	Reserved	Reserved
0	OPRT_CHG_EN	OTG output port status change interrupt enable: this bit enables interrupt generation when the OTG_IRQ_SRC:OPRT_CHG bit is active. 0: No interrupt is generated. 1: An interrupt is generated if the OTG_IRQ_SRC:OPRT_CHG bit is set. This bit is cleared 0 by soft reset or hardware reset.

This read/write register controls which OTG controller interrupts are passed to the MPU level 2 interrupt controller IRQ_8.

Table 69. OTG Interrupt StatusRegister (OTG_IRQ_SRC)

Bit	Name	Description
15	DRIVER_SWITCH	Driver switch interrupt status. This bit reflects the status of the driver switch interrupt source. Driver switch interrupts occur when HNP completes and control of the OTG link must switch between the OMAP5912 USB host controller driver and the OMAP5912 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, OGT_CTRL.DRIVER_SEL selects the controller driver that is appropriate, given the value of OTG_CTRL.ID. 0: Driver switch interrupt is inactive. 1: Driver switch interrupt is active. This bit is cleared either by writing a 1 or by soft reset or hardware reset. This bit has no effect when OTG_SYSCON_2.OTG_EN is 0.
14	Reserved	Reserved
13	A_VBUS_ERR	A-device Vbus error interrupt status. This bit reflects the status of the A_VBUS_ERR interrupt. VBUS errors occur when the OMAP5912 OTG controller state machine acts as a 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_CTRL.OTG_BUS_DROP clears the source of this interrupt. 0: VBUS error interrupt is inactive. 1:VBUS error interrupt is active. This bit is cleared either by writing a 1 or by soft reset or hardware reset. This bit has no effect when OTG_SYSCON_2.OTG_EN is 0.

Table 69. OTG Interrupt StatusRegister (OTG_IRQ_SRC) (Continued)

Bit	Name	Description
12	A_REQ_TMROUT	A-device request time-out interrupt status. This bit reflects the status of the A_REQ_TMROUT interrupt. This interrupt occurs when the OMAP5912 OTG controller state machine acts as a 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. 0: A-device request time-out interrupt is inactive. 1: A-device request time-out interrupt is active. This bit is cleared either by writing a 1 or by soft reset or hardware reset. This bit has no effect when OTG_SYSCON_2.OTG_EN is 0.
11	A_SRP_DETECT	A-device SRP detection interrupt status.This bit reflects the status of SRP detection when the OMAP5912 OTG controller acts as an 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. 0: SRP detection interrupt is inactive. 1: An SRP has been detected, using the VBUS pulsing and/or data pulsing detection mechanism. This bit is cleared either by writing 1 or by soft reset or hardware reset. This bit has no effect when OTG_SYSCON_2.OTG_EN is 0.
10	B_HNP_FAIL	B-device HNP failed interrupt status. This bit reflects interrupts that are generated when the OMAP5912 OTG controller acts as an OTG default-B device and an HNP it attempts 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 OMAP5912 OTG controller acting as a 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 OMAP5912 acting as a default-B device issues an HNP request, but VBUS fails before completion of the HNP. 0: B-device HNP failure interrupt is inactive. 1: B-device HNP failure interrupt is active. This bit is cleared either by writing 1 or by soft reset or hardware reset. This bit has no effect when OTG_SYSCON_2.OTG_EN is 0.

Table 69. OTG Interrupt StatusRegister (OTG_IRQ_SRC) (Continued)

Bit	Name	Description
9	B_SRP_TMROUT	B-device SRP time-out interrupt status.This bit reflects interrupts that are generated when the OMAP5912 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). 0: SRP time-out interrupt is inactive. 1: SRP time-out interrupt is active. This bit is cleared either by writing 1 or by soft reset or hardware reset. This bit has no effect when OTG_SYSCON_2.OTG_EN is 0.The SRP timer starts when the B-device bus request is activated.
8	B_SRP_DONE	B-device SRP done interrupt status. This bit reflects interrupts that are generated upon completion of an SRP when the OMAP5912 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. 0: SRP done interrupt is inactive. 1: SRP done interrupt is active. This bit is cleared either by writing 1 or by soft reset or hardware reset. This bit has no effect when OTG_SYSCON_2.OTG_EN is 0.
7	B_SRP_STARTED	B-device SRP started interrupt status. This bit reflects interrupts that are generated when the OMAP5912 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. 0: SRP begin interrupt is inactive. 1: SRP begin interrupt is active. This bit is cleared either by writing 1 or by soft reset or hardware reset. This bit has no effect when OTG_SYSCON_2.OTG_EN is 0.

Table 69. OTG Interrupt StatusRegister (OTG_IRQ_SRC) (Continued)

Bit	Name	Description
6:1	Reserved	Reserved
0	OPRT_CHG	OTG output port status change interrupt status. This bit reflects interrupts that are generated when the OMAP5912 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 I^2C operations to change the control registers in the OTG transceiver. When this bit is 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. 0: Output port status change interrupt is inactive. 1: An interrupt is generated for a change on OTG output ports status. This bit is cleared either by writing 1 or by soft reset or hardware reset. This bit has no effect when OTG_SYSCON_2.OTG_EN is 0.

This read/write register reflects all OTG interrupt status.

Table 70. OTG Output pins control register (OTG_OUTCTRL)

Bit	Name	Description
15	RESERVED	Reserved
14	OTGVPD	OTG Pull down VBUS
13	OTGVPU	OTG Pull up VBUS
12	OTGPUID	OTG Pull up ID
11	RESERVED	Reserved
10	USB2VDR	USB2 Port Vbus drive
9	USB2PDEN	USB2 Port Pull Down enabled
8	USB2PUEN	USB2 Port Pull Up enabled
7	RESERVED	Reserved
6	USB1VDR	USB1 Port Vbus drive
5	USB1PDEN	USB1 Port Pull Down enabled

Table 70. OTG Output pins control register (OTG_OUTCTRL) (Continued)

Bit	Name	Description
4	USB1PUEN	USB1 Port Pull Up enabled
3	RESERVED	Reserved
2	USB0VDR	USB0 Port Vbus drive
1	USB0PDEN	USB0 Port Pull Down enabled
0	USB0PUEN	USB0 Port Pull Up enabled

Table 71. OTG test register (OTG_TEST)

Bit	Name	Description
15	TEST_UNLOCK	Test mode unlock
14:9	RESERVED	Reserved
8	IRQ_OTG	IRQOTGON signal control
7:0	OTG_FSM_STATE	OTG FSM state

Table 72. OTG Vendor Code Register (OTG_ VC)

Bit	Name	Description
31:16	Reserved	Reserved
15:0	VC	Vendor code identifier: this read-only register reflects the Texas Instruments vendor code identifier: 0x5449 for TI in ASCII.

This register reflects the binary coded decimal equivalent of the USB vendor code identifier assigned to Texas Instruments.

4.2.1 OTG Clock and Reset Requirements

The OTG module is clocked mainly by the 48-MHz input clock from the OMAP5912 ULPD module. The ULPD module registers control the 48 MHz.

- ❏ CLOCK_CTRL_REG.USB_MCLK_EN
- ❏ SOFT_REQ_REG.SOFT_USB_OTG_DPLL_REQ
- ❏ SOFT_DISABLE_REQ_REG.DIS_USB_HOST_DPLL_REQ

When the OTG module receives a 48-MHz clock from the ULPD module, the OTG controller logic can have its clocks disabled locally via the OTG_SYSCON_1.OTG_IDLE_EN bit. The OTG module register interface is clocked separately, so OTG module registers other than OTG_SYSCON_2

and OTG_CTRL can be accessed when the 48-MHz clock input is inactive. Accesses to OTG_SYSCON_2 and OTG_CTRL require that the OTG controller 48-MHz clock be active.

The OTG controller 48-MHz clock must be enabled at both the ULPD level and the OTG module level whenever USB On-The-Go functionality is needed. It is not appropriate to disable the OTG controller 48-MHz clock when a USB On-The-Go link is established with another USB On-The-Go dual-role device.

The OTG controller uses the same 48-MHz clock input from the ULPD module to provide clocks to the USB device controller and the USB host controller. When the ULPD 48-MHz clock to the OTG module 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 module can disable the USB device controller clock using OTG_SYSCON_1.DEV_IDLE_EN. When the ULPD 48-MHz clock to the OTG module is disabled, or when OTG_SYSCON_1.DEV_IDLE_EN is 1, the USB device controller registers cannot be accessed, but 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 OTG_SYSCON_1.DEV_IDLE_EN is 1, but must enable the 48-MHz clock to the USB device controller before writing to the USB device controller registers.

Hardware reset of the USB OTG module is provided by the ULPD module. The PER_EN bit in the MPU reset control 2 register controls the reset to many OMAP5912 peripherals, including the USB OTG module. 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.

The OTG controller provides a soft reset mechanism. When OTG_SYSCON_1.SOFT_RESET is written with a 1, the OTG controller, the USB device controller, and the USB host controller are reset. Soft reset is complete when OTG_SYSCON_1.RESET_DONE is 1. OTG_SYSCON_1.SOFT_RESET automatically clears itself.

4.2.2 OTG Controller Power Management

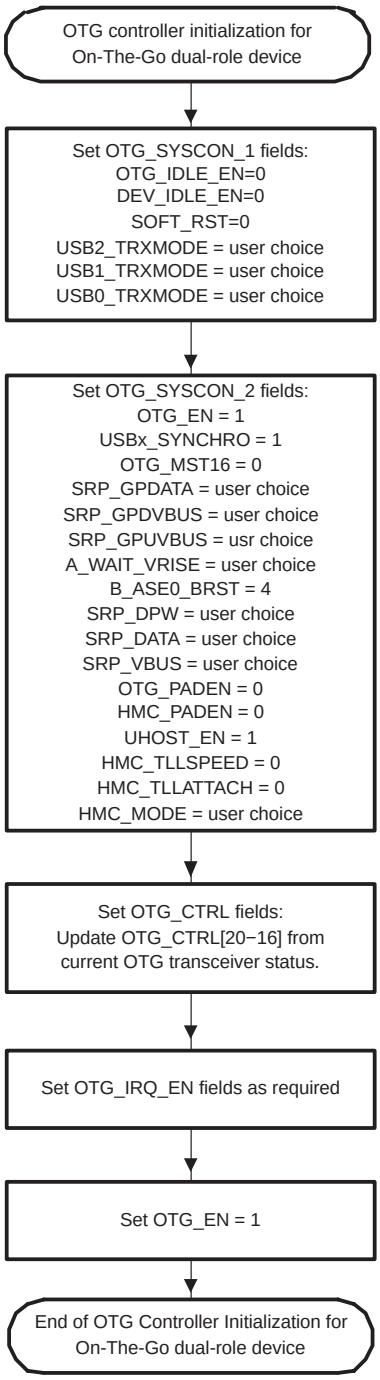
OTG controller power management is provided using the OTG_SYSCON_2.OTG_EN bit. When OTG_SYSCON_2.OTG_EN is 0, OTG controller power consumption is reduced, because the OTG controller logic is not clocked. The OTG controller cannot perform On-The-Go HNP or SRP when it is not clocked.

4.2.3 OTG Controller Initialization

OTG controller initialization operations depend on whether or not the system implements an On-The-Go dual-role device.

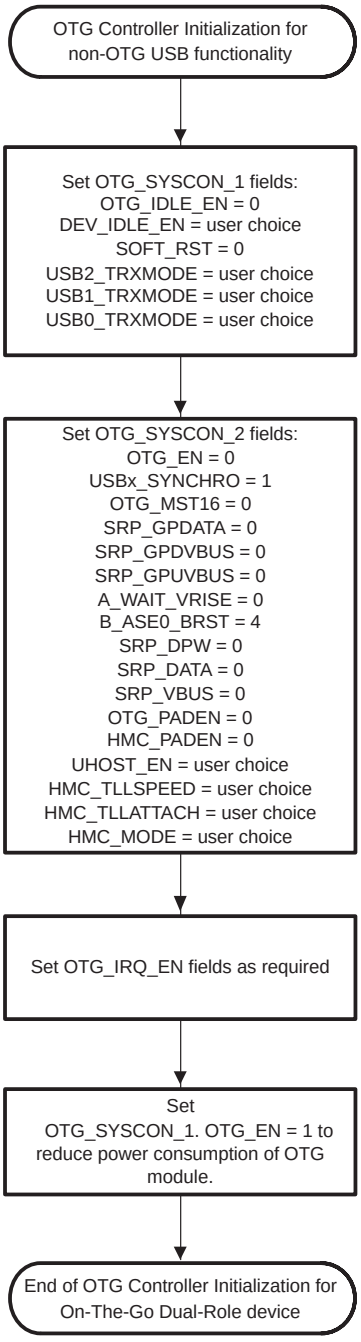
If the application implements an OTG dual-role device, follow the initialization shown in Figure 48.

Figure 48. OTG Controller Initialization When Implementing an OTG Dual-Role Device



Systems that implement USB functionality without implementing an OTG dual-role device are recommended to follow the initialization shown in Figure 49.

Figure 49. OTG Controller Initialization When Not Implementing an OTG Dual-Role Device



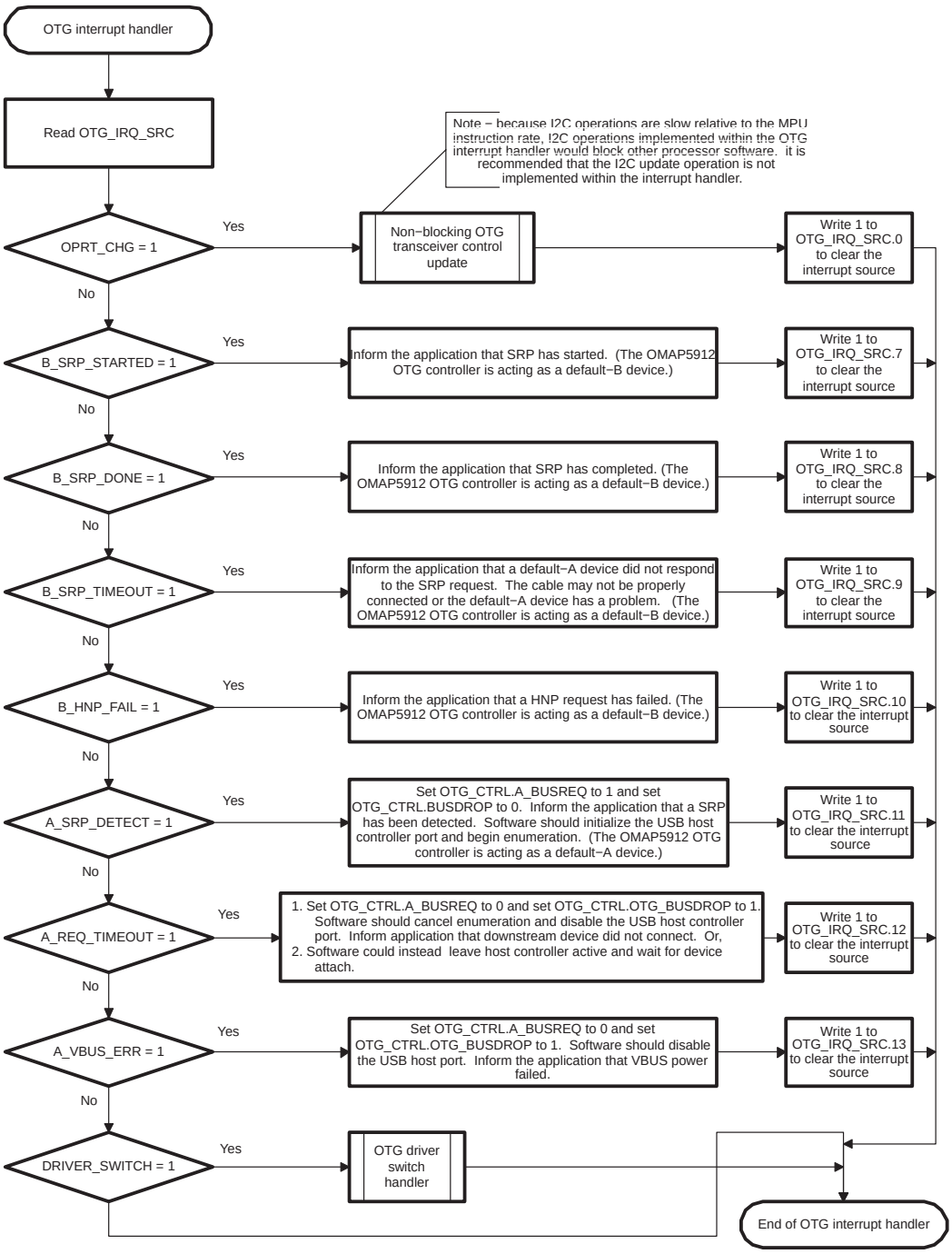
User choice values shown in Figure 48 and Figure 49 depend on the system implementation, including transceiver types being used, top-level pin multiplexing choice, and connector type. Pin multiplexing options and transceiver types are discussed in Section 4.3, *Pin Multiplexing*.

4.2.4 OTG Controller Interrupt Handler and Related Software

The OTG controller implements several interrupt sources that are separately enabled via bits in OTG_IRQ_EN. The OTG controller provides a single interrupt output that provides OTG controller interrupts to the MPU level 2 interrupt controller IRQ_8 input.

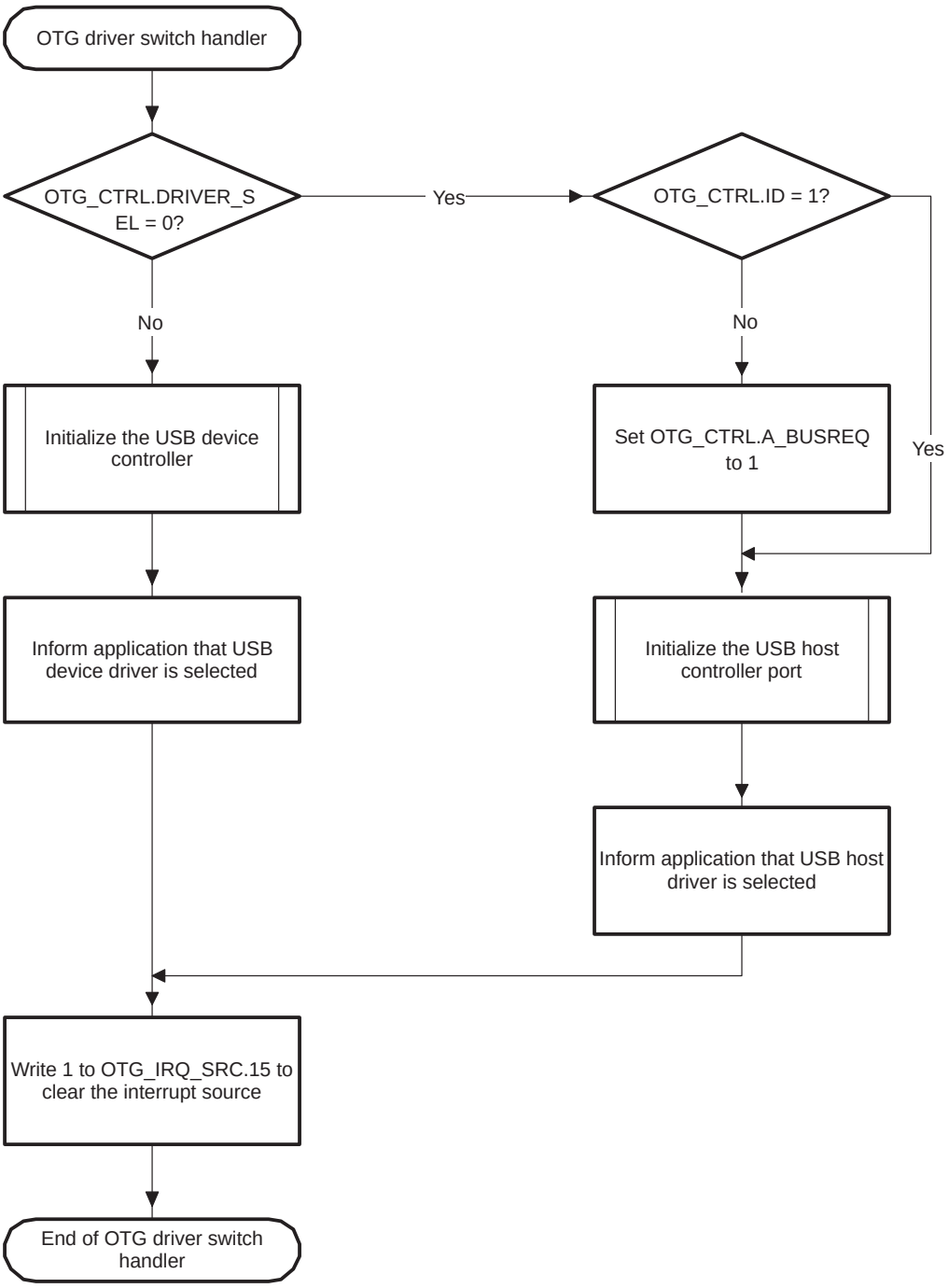
Figure 50 shows the operations required in the USB OTG controller interrupt handler. This flowchart references several other operations, which are described in flowcharts later in this section.

Figure 50. OTG Interrupt Handler



The OTG driver switch handler is shown in Figure 51. 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 can also occur when OTG_SYSCON_2.OTG_EN is set to 1.

Figure 51. OTG Driver Switch Handler



The effects of On-The-Go functionality on OMAP5912 USB device controller operation are documented in the USB device controller flowcharts. Because USB host controller operations are mainly defined in the open host controller for USB specification, some On-The-Go functionality issues are discussed here.

Most OMAP5912 USB host controller software is not affected by USB On-The-Go functionality. A host controller driver written for non-OTG functionality needs few modifications to support On-The-Go functionality. In general, On-The-Go events precede USB host controller driver operations, so the On-The-Go 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 On-The-Go 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.

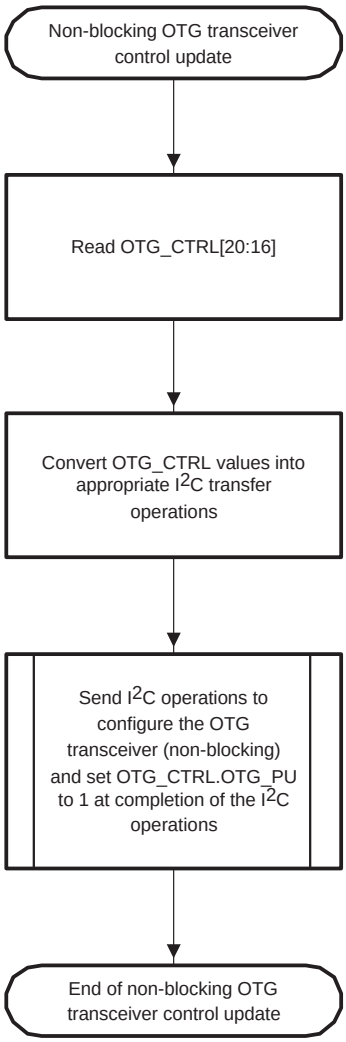
The USB host controller driver must implement a mechanism where the application can specify that the OTG port is no longer needed. The host controller driver software must respond to this message by suspending the port associated with the OTG link and cancelling all EDs and TDs associated with the OTG connection. These ED and TD removal activities are ones 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 need not remain active. If no other USB host controller ports are being 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 re-enable the USB host controller clock and re-initialize the USB host on an OTG driver switch interrupt to USB host controller control of the OTG link if the host clock is dynamically disabled.

Software to support communication between the USB OTG controller and the external OTG transceiver via I^2C is described in Figure 52. 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;

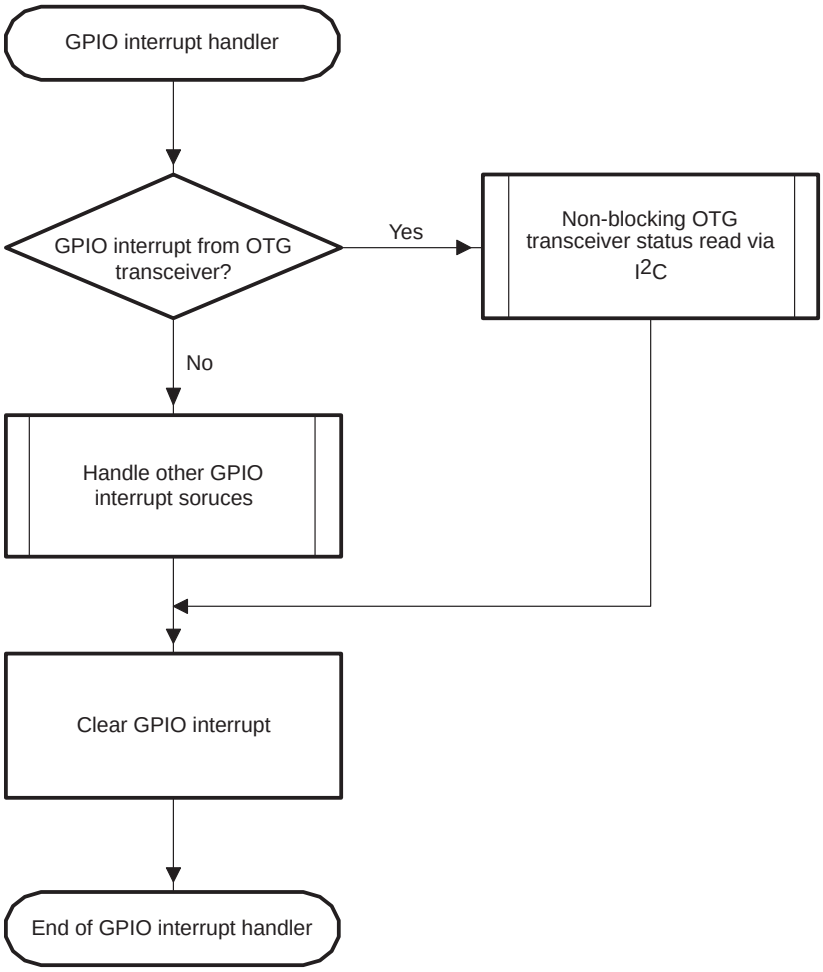
interrupt handlers block other processor software operations until the handler completes. System software must implement a non-blocking mechanism, such as a message-based system, to allow the I^2C operations to be completed outside of the OTG interrupt handler, but then allow updating of the OTG controller OTG_CTRL.OTG_PU bit on completion of the I^2C operations.

Figure 52. OTG Transceiver I^2C Control Handler



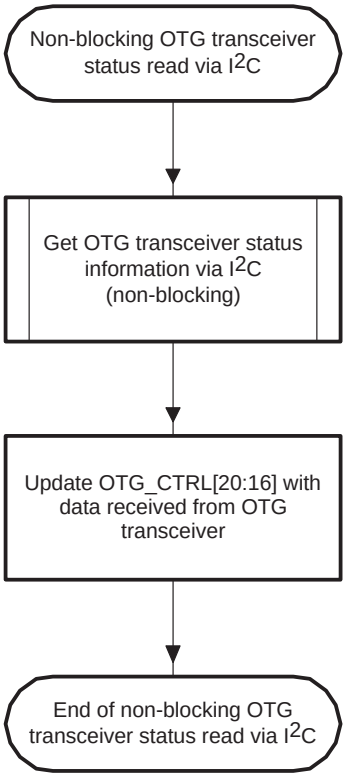
The OTG transceiver signals the need for a status transfer from the OTG transceiver to the OTG controller, using an interrupt output signal. This interrupt is generally connected to an OMAP5912 GPIO input pin, and that pin is configured to provide an interrupt to the MPU Level 2 interrupt controller. Figure 53 shows the basic GPIO interrupt handler functionality required.

Figure 53. GPIO Interrupt Handler Support for OTG Transceiver Interrupt Input



The operations in Figure 54 show the non-blocking OTG transceiver status read and update to the OTG controller registers. To reduce the effect on system performance, it is important that the I^2C read operations are non-blocking.

Figure 54. OTG Transceiver Status Read



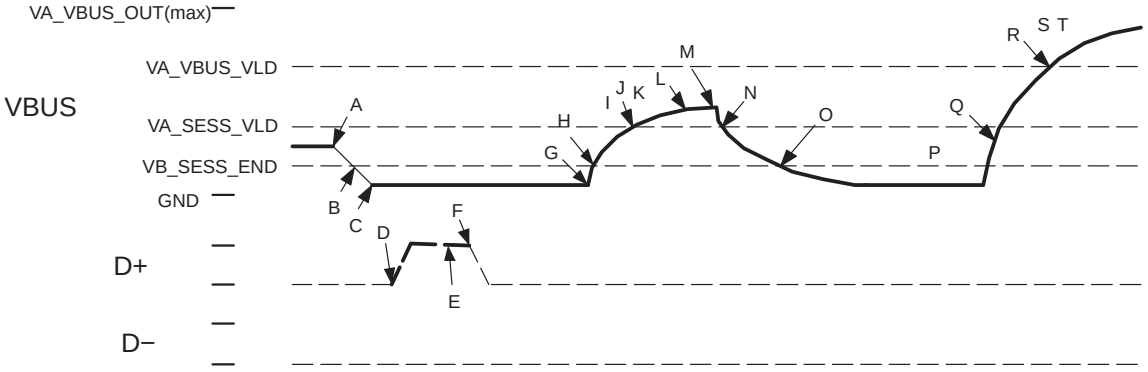
4.2.5 Typical SRP and HNP Events

This section describes the typical sequence of events for SRP and HNP events with the OMAP5912 OTG controller.

Careful system software design can optimize system performance by managing interrupt source enables at both the OTG controller and at 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 55 shows the typical events that occur when OMAP5912 acts as an OTG default-A dual-role device and the default-B device issues an SRP. Figure 55 shows that the default-B device pulses its D+ pullup resistor. The OMAP5912 OTG controller accepts D− pullup pulses as well as the D+ pullup pulse shown as SRP data line.

Figure 55. OMAP5912 OTG Controller Response To SRP When Acting as a Default-A Dual-Role OTG Device

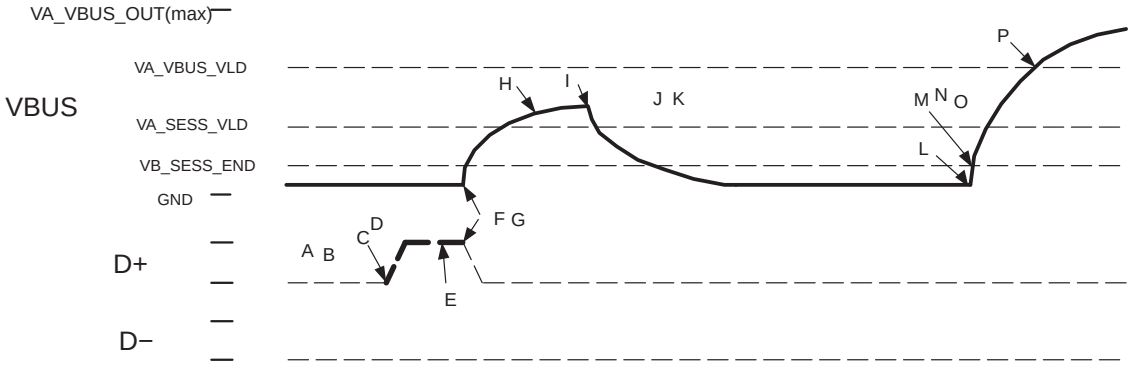


A. OTG dual-role default-B device begins discharging VBUS (optional).
B. VBUS at default-B device falls below VB_SESS_END.
C. OTG dual-role default-B device stops discharging VBUS (optional).
D. Default-B device enables SRP D+ pullup.
E. OMAP5912 OTG controller recognizes D+ pulse SRP request if D+ pulse 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 OMAP5912 OTG transceiver. OTG transceiver issues an interrupt.
J. OMAP5912 GPIO interrupt handler identifies OTG transceiver interrupt. Handler initiates I^2C operation to get transceiver interrupt source information.
K. OMAP5912 I^2C operations to get transceiver interrupt source information completes. System software sees VBUS > VA_SESS_VLD and sets OTG_CTRL.ASESSVLD to 1.
L. OMAP5912 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. OMAP5912 OTG Interrupt handler sees OPRT_CHG interrupt and sees OTG_CTRL.OTG_DRV_VBUS set to 1 and initiates I^2C 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 dual-role device.
Q. OMAP5912 I^2C operations to configure OTG transceiver to drive VBUS complete. System software writes a 1 to OTG_CTRL.OTG_PU to indicate that the I^2C operation has completed.
R. VBUS rises above VA_VBUS_VLD. OTG transceiver issues interrupt.
S. OMAP5912 GPIO interrupt handler sees OTG transceiver interrupt, initiates I^2C operations to get OTG interrupt source information.
T. OMAP5912 I2C operations complete. System software sees OTG status showing that VBUS voltage is above VA_SESS_VLD and sets OTG_CTRL.ASESSVLD to 1. OMAP5912 OTG dual-role device is now ready for default-B device to enable its pullup.

Figure 56 shows the typical events that occur when OMAP5912 acts as an OTG default-B dual-role device and issues an SRP to the default-A device. If OMAP5912 completes its SRP request but the default-A device does not drive VBUS within 5.5 seconds, the OMAP5912 OTG controller issues a B_SRP_TIMEOUT interrupt. System software must provide a message to the user that says the SRP request failed and that the user needs to check the

cable and the A-device. The OMAP5912 OTG controller only implements D+ data line SRP pulses and does not request a D− pulse as a data line SRP pulse.

Figure 56. OMAP5912 OTG Controller SRP Generation When Acting as a Default-A Dual-Role OTG Device



A. OMAP5912 is acting as a default-B device. OTG_CTRL.BSESSEND=1. Application decides that it wants to communicate with the default-B dual-role device. System software sets OTG_CTRL.B_BUSREQ.

B. OMAP5912 OTG controller sees write of 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. OMAP5912 OTG interrupt handler sees OPRT_CHG interrupt and sees OTG_PU = 1. Handler initiates I^2C operation to enable OTG transceiver D+ pullup.

D. OMAP5912 I^2C operations to enable the D+ pullup complete. OMAP5912 software sets OTG_CTRL.OTG_PU to 1 to indicate that the D+ pullup enable operation has completed.

E. OMAP5912 OTG controller changes OTG_CTRL.OTG_PU to 0 approximately 9 ms after it had set it to 1. OTG controller also sets OTG_CTRL.OTG_VBUS_PU at the same time. OTG controller issues OPRT_CHG interrupt.

F. OMAP5912 OTG interrupt handler sees OPRT_CHG interrupt and sees OTG_PU = 0 and sees OTG_PU_VBUS = 1. Handler initiates I^2C operation to disable OTG transceiver D+ pullup and to enable OTG transceiver VBUS pullup.

G. OMAP5912 I^2C operations to disable the D+ pullup and enable the VBUS pullup complete. OMAP5912 software sets OTG_CTRL.OTG_PU to 1 to indicate that the D+ pullup disable operation has completed.

H. OMAP5912 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 had set 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. OMAP5912 OTG interrupt handler sees OPRT_CHG interrupt and sees OTG_CTRL.OTG_PU_VBUS = 0 and sees current value in OTG_CTRL.OTG_PD_VBUS. Handler initiates I^2C 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. OMAP5912 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. OMAP5912 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^2C 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.

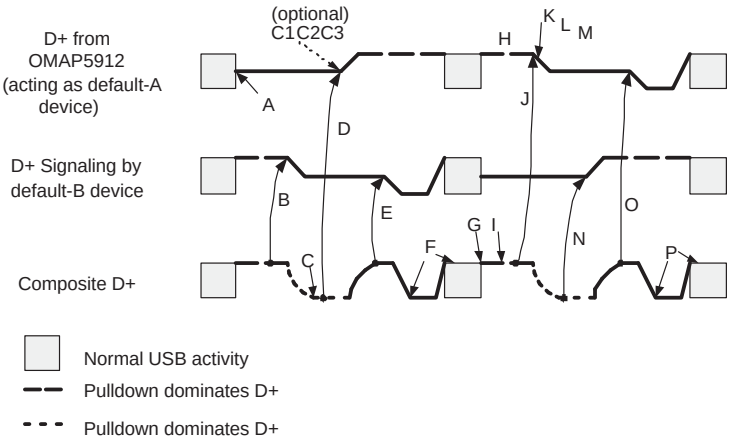
N. The OMAP5912 GPIO interrupt handler sees the OTG transceiver interrupt and initiates the I^2C operations which query the OTG transceiver interrupt status.

O. The I^2C operation completes, and OMAP5912 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 OMAP5912 USB device controller will see 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 OMAP5912 default-B device.

Figure 57 shows the typical events that occur when OMAP5912 acts as an OTG default-A dual-role device 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 57. OMAP5912 OTG Controller HNP Events When Acting as a Default-A Dual-Role OTG Device



A: OMAP5912 application has no more traffic for default-B device. System software suspends the USB host port which is being used for the OTG link, sets OTG_CTRL.A_BUSREQ to 0. If the OTG transceiver supports the autoconnect feature, system software should initiate the appropriate I^2C 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 OMAP5912 to enable its D+ pullup.

C: OMAP5912 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 will automatically enable its D+ pullup upon seeing SE0 on the bus, and issue an interrupt.

C2.OMAP5912 GPIO interrupt handler sees an OTG transceiver interrupt and issues I^2C operations to query the transceiver interrupt source.

C3. OMAP5912 I^2C operation completes and the interrupt status shows autoconnect occurred. It is not necessary to update any OTG controller registers when auto-HNP occurs.

D. OMAP5912 OTG interrupt handler sees OPRT_CHG interrupt and OTG_PU set to 1, and (if auto-HNP is not used) causes I_2C operation to enable OTG transceiver D+ pullup. OMAP5912 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 OMAP5912 dual-role device as a peripheral, and begins normal USB bus activity.

F. OMAP5912 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. OMAP5912 application decides that it wants to communicate with the default-B device. System software sets OTG_CTRL.A_BUSREQ.

I. OMAP5912 OTG controller sees suspend signaled on the OTG link, sets OTG_CTRL.OTG_PU to 0, issues OPRT_CHG interrupt.

J. OMAP5912 OTG interrupt handler sees OPRT_CHG interrupt, sees OTG_CTRL.OTG_PU set to 0, issues I^2C operations to disable the OTG transceiver D+ pullup.

K. OMAP5912 I^2C operations to disable the D+ pullup complete. OMAP5912 software sets OTG_CTRL.OTG_PU to 1 to indicate that the D+ pullup disable operation has completed.

L. OMAP5912 OTG controller sees write of 1 to OTG_CTRL.OTG_PU and issues driver switch interupt.

M. OMAP5912 OTG interrupt handler sees driver switch interrupt, begins intialization of host port.

N. Default-B device sees SE0 on the OTG link, turns on its D+ pullup.

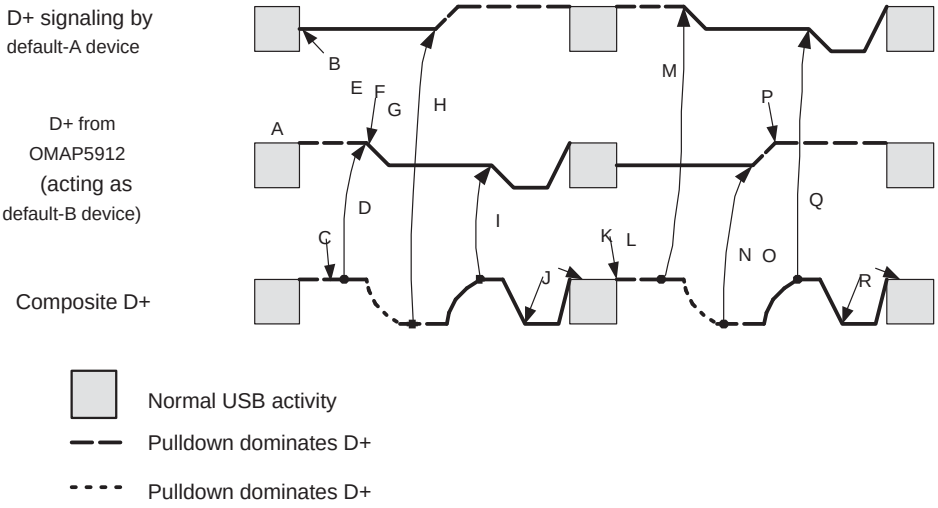
O. OMAP5912 OTG controller and USB host controller see D+ pullup as HNP operation. OMAP5912 USB host controller signals USB reset, sends the set feature signal with B_HNPEN feature enabled, and enumerates the default-B dual-role device as a peripheral, and begins normal USB bus activity.

P. Default-B device sees and responds appropriately to USB reset, the set feature signal, enumeration and normal USB bus activity.

Figure 58 shows the typical events that occur when OMAP5912 acts as an OTG default-B dual-role device 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 VB_SESS_VLD threshold during the HNP process, the OMAP5912 OTG controller issues a B_HNP_FAIL interrupt.

Figure 58. OMAP5912 OTG Controller HNP Events When Acting as a Default-B Dual-Role OTG Device



A: OMAP5912 application decides that it wishes 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 (OMAP5912). Default-A device suspends the OTG link.

C: OMAP5912 OTG controller (acting as the OTG dual-role default-B device) 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. OMAP5912 OTG interrupt handler sees OPRT_CHG interrupt and sees OTG_PU = 0. Handler initiates I^2C operation to disable OTG transceiver D+ pullup.

E. OMAP5912 I^2C operations to disable the D+ pullup complete. OMAP5912 software sets OTG_CTRL.OTG_PU to 1 to indicate that the D+ pullup disable operation has completed.

F. OMAP5912 OTG controller sees write of 1 to OTG_CTRL.OTG_PU and issues driver switch interupt.

G. OMAP5912 OTG interrupt handler sees driver switch interrupt, begins intialization of host port.

H. Default-A device sees SE0 on the OTG link, turns on its D+ pullup.

I. OMAP5912 OTG controller sees D+ pullup as HNP operation and OMAP5912 USB host contoller sees D+ pullup as an attach. OMAP5912 USB host controller signals USB reset, and enumerates the default-A dual-role device 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. OMAP5912 application decides it has no more information to communicate to the default-A device. System software suspends the USB host port which is being used for the OTG link, sets OTG_CTRL.A_BUSREQ to 0.

L. OMAP5912 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. OMAP5912 OTG interrupt handler sees OPRT_CHG interrupt, sees OTG_CTRL.OTG_PU set to 1, initiates I^2C operations to enable OTG transceiver D+ pullup.

O. OMAP5912 OTG interrupt handler sees Driver Switch and begins initialization of OMAP5912 USB device controller.

P. OMAP5912 I^2C operations to enable OTG transceiver D+ pullup complete.

Q. Default-A devcie sees D+ pullup, signals USB reset, issues Set Feature of the B_HNP_ENABLE feature, and enumerates the OMAP5912 default-B dual-role device as a peripheral, and begins normal USB bus activity.

R. OMAP5912 USB device controller sees and responds appropriately to USB reset, set feature of the B_HNP_ENABLE feature, enumeration, and normal USB bus activity.

4.2.6 System-Level OTG Considerations

The OMAP5912 USB OTG implementation allows up to two separate USB host controller ports to function in parallel with an OTG link. The USB OTG specification states that an OTG dual-role device 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 Interface Specification* can use the transceiver autoconnect feature. Enabling this OTG transceiver feature when OMAP5912 is acting as a default-A dual-role On-The-Go device eases the timing requirements on system software during an HNP transition from OMAP5912 acting as an A-host to OMAP5912 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. It is possible to connect a non-OTG USB device to the OMAP5912-based OTG system mini-AB receptacle using an adapter cable as defined in the On-The-Go supplement to the USB specification. 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.

4.3 Pin Multiplexing

OMAP5912 USB signal multiplexing determines which USB functionality is available at which OMAP5912 pins. OMAP5912 provides three pin groups that can be configured to provide up to three simultaneous USB ports. These ports can be configured to provide a USB OTG port, a USB device port, and/or up to three USB host ports. When OMAP5912 is configured to provide USB OTG functionality at one pin group, it is not possible to configure another pin group to act as a USB device, because there is only one USB device controller and it is used by the OTG functionality.

A USB transceiver is needed for each USB port used in the system. It converts between signaling appropriate for the OMAP5912 USB controllers and signaling appropriate for the USB wire. OMAP5912 USB functionality includes support for several types of USB transceivers. OMAP5912 provides one integrated USB transceiver suitable for a single USB host or USB device port, but it is not capable of acting as a USB OTG transceiver. OMAP5912 provides signaling to up to two external USB transceivers and/or two external USB-OTG-capable transceivers.

Several different types of external transceiver signaling are supported. Signaling between the OMAP5912 USB controller and the external USB

transceiver for monitoring and controlling the differential USB signal can be via a 6-wire, 4-wire, or 3-wire signaling interface, with two or more additional control signals provided either by additional signals or via an I^2C link. OTG transceivers can support the 6-wire, 4-wire, and/or 3-wire basic signaling but generally provide an interrupt output and an I^2C link for the additional control and status reporting required by OTG functionality.

4.4 Selecting and Configuring USB Connectivity

The process of selecting desired USB connectivity and configuring OMAP5912 for that connectivity can be done using the steps listed below.

4.4.1 Select Desired USB Functionality

Choose the desired USB functionality. OMAP5912 can bring up to three USB ports to device pins. Options include:

- One USB device port
- One USB OTG port
- Up to three USB host ports

Choose a maximum of three of the above. It is not possible to configure both a USB device port and a USB OTG port, because the USB OTG port must use the single USB device controller. Similarly, a USB OTG port also makes use of one of the three USB host controller ports. Some possible configurations are:

- One USB device port
- One OTG port and one host port
- Two host ports and one USB device port

Some illegal combinations are:

- 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 OMAP5912 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 OMAP5912 pins)

4.4.2 Select How USB Functionality Is Multiplexed to OMAP5912 Pins

OMAP5912 provides pin interfaces for up to three USB ports. The USB functional ports can be mapped to the different OMAP5912 USB pin groups.

- Pin group 0 is associated with the OMAP5912 integrated USB transceiver, which is connected to OMAP5912 pins USB.DP and USB.DM. The transceiver can be used as a USB host or USB device port, but is not for use as a USB OTG port without additional external hardware. Pin group 0 is related to USB signal multiplexing port 0. When USB alternate pin group 2 is used, OMAP5912 pins USB.DP and USB.DM cannot be used for USB transceiver functionality.

The other two USB-related pin groups are associated with standard CMOS input and output pins. USB connectivity that uses these pin groups must use USB external transceivers. OMAP5912 top-level pin multiplexing allows a wide variety of functionality to be provided via these pins, so selection of these pins for use as USB functionality limits designer ability to use those pins for other (non-USB) functionality.

- Pin group 1 is associated with the following OMAP5912 pins:
  - MCBSP3.CLKX/USB1.TXEN
  - MCSI1.DOUT/USB1.TXD
  - RTC_WAKE_INT/USB1.SE0
  - MCSI1.DIN/USB1.RCV
  - MCSI1.SYNC/USB1.VP
  - MCSI1.BCLK/USB1.VM
  - CLK32K_OUT/USB1.SPEED
  - MPU_BOOT/USB1.SUSP

The last pin group has two different USB-related modes:

- Pin group 2 maps the signals associated with USB signal multiplexing port 2 to OMAP5912 pins, and is associated with these OMAP5912 pins:
  - MCSI2.DOUT/USB2.TXEN
  - UART2.TX/USB2.TXD
  - UART2.RTS/USB2.SE0
  - UART2.CTS/USB2.RCV
  - MCSI2.DIN/USB2.VP
  - UART2.RX/USB2.VM
  - MCSI2.SYNC/USB2.SPEED
  - MCSI2.CLK/USB2.SUSP
- Alternate pin group 2 maps signals associated with USB signal multiplexing port 0 to the same OMAP5912 pins used in pin group 2. The pin names for alternate pin group 2 are as follows:

- MCSI2.DOUT/USB0.TXEN
- UART2.TX/USB0.TXD
- UART2.RTS/USB0.SE0
- UART2.CTS/USB0.RCV
- MCSI2.DIN/USB0.VP
- UART2.RX/USB0.VM
- MCSI2.SYNC/USB0.SPEED
- MCSI2.CLK/USB0.SUSP

The selection procedure is as follows:

1) Given the required USB functionality, the USB pin group descriptions, and any non-USB usage of the pins in the pin group descriptions, choose a USB signal multiplexing mode from Table 73.

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 OMAP5912 pin usage shown under the pin group and alternate pin group columns and find one HMC_MODE value that meets your needs for both USB and non-USB functionality.

4) Make careful note of the information in the row that determines how the top-level pin multiplexing and HMC_MODE value must be configured (see Table 73).

Table 73. USB Signal Multiplexing Modes

HMC_MODE	OTG Required?	USB Multiplexing Port 0	USB Multiplexing Port 1	USB Multiplexing Port 2	USB Pin Group 0	USB Pin Group 1	USB Pin Group 2	USB Alternate Pin Group 2
0	No	USB device	Disabled	Disabled	USBdevice: See Note 1.	Available for non-USB usage	Available for non-USB usage	N/A
					Available for non-USB usage		N/A	USB device: See Note 2.
0 (cont)	Yes	USB OTG	Disabled	Disabled	USB OTG: See Note 3.	Available for non-USB usage	Available for non-USB usage	N/A
					Available for non-USB usage		N/A	USB OTG: See Note 4.

Table 73. USB Signal Multiplexing Modes (Continued)

HMC_MODE	OTG Required?	USB Multiplexing Port 0	USB Multiplexing Port 1	USB Multiplexing Port 2	USB Pin Group 0	USB Pin Group 1	USB Pin Group 2	USB Alternate Pin Group 2
1	No	USB host	USB host	USB host	USB host: See Note 5.	USB host: See Note 6.	USB host: See Note 7.	N/A
				Disabled	Available for non-USB usage		N/A	USB host: See Note 8.
	Yes	USB OTG	USB host	USB host	USB OTG: See Note 3.	USB host: See note 6.	USB host: See Note 7.	N/A
				Disabled	Available for non-USB usage		N/A	USB OTG: See Note 4.
2	No	USB host	UART1	USB host	USB host: See Note 5.	UART1: See note 9.	USB host: See Note 7.	N/A
				Disabled	Available for non-USB usage		N/A	USB host: See Note 8.
3	No	USB host	USB device	USB host	USB host: See Note 5.	USB device: See Note 10.	USB host: See Note 7.	N/A
				Disabled	Available for non-USB usage		N/A	USBhost: See Note 8.
	Yes	USB host	USB OTG	USB host	USB OTG: See Note 3.	USB OTG: See Note 11.	USB host: See Note 7.	N/A
				Disabled	Available for non-USB usage		N/A	USB OTG: See Note 12.
4	No	USB device	USB host	USB host	USB device: See Note 1.	USB host: See Note 6.	USB host: See Note 7.	N/A
				Disabled	Available for non-USB usage		N/A	USB device: See Note 2.

Table 73. USB Signal Multiplexing Modes (Continued)

HMC_MODE	OTG Re-quired?	USB Multi-plex-ing Port 0	USB Multi-plex-ing Port 1	USB Multi-plex-ing Port 2	USB Pin Group 0	USB Pin Group 1	USB Pin Group 2	USB Alter-nate Pin Group 2
	Yes	USB OTG	USB host	USB host	USB OTG: See Note 3.	USB host: See Note 6.	USB host: See Note 7.	N/A
4 (cont)	Yes	USB OTG	USB host	Disabled	Available for non-USB usage	USB host: See Note 6.	N/A	USB OTG: See Note 4.
5	No	USB host	Disabled	USB host	USB host: See Note 5.	Available for non-USB usage	USB host: See Note 7.	N/A
				Disabled	Available for non-USB usage		N/A	USB host: See Note 8.
	Yes	USB OTG	Disabled	USB host	USB OTG: See Note 3.	Available for non-USB usage	USB host: See Note 7.	N/A
				Disabled	Available for non-USB usage		N/A	USB OTG: See Note 4.
6	No	USB device	Disabled	USB host	USB device: See Note 1.	Available for non-USB usage	USB host: See Note 7.	N/A
				Disabled	Available for non-USB usage		N/A	USB device: See Note 2.
	Yes	USB OTG	Disabled	USB host	USB OTG: See Note 3.	Available for non-USB usage	USB host: See Note 7.	N/A
				Disabled	Available for non-USB usage		N/A	USB OTG: See Note 4.
7	No	USB host	Disabled	Disabled	USB host: See Note 5.	See Note 13.		N/A
8	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 73. USB Signal Multiplexing Modes (Continued)

HMC_MODE	OTG Required?	USB Multiplexing Port 0	USB Multiplexing Port 1	USB Multiplexing Port 2	USB Pin Group 0	USB Pin Group 1	USB Pin Group 2	USB Alternate Pin Group 2
9	No	USB host	USB host	USB host with TLL	USB host: See Note 5.	USB host: See Note 6.	USB host + TLL: See Note 14.	N/A
				Disabled	Available for non-USB usage		N/A	USB host: See Note 8.
9 (cont)	Yes	USB OTG	USB host	USB host with TLL	USB OTG: See Note 3.	USB host: See Note 6.	USB host + TLL: See Note 14.	N/A
				Disabled	Available for non-USB usage		N/A	USB OTG: See Note 4.
10	No	USB host	UART1	USB host with TLL	USB host: See Note 5.	UART1: See Note 9.	USB host + TLL: See Note 14.	N/A
				Disabled	Available for non-USB usage		N/A	USB host: See Note 8.
11	No	USB host	USB device	USB host with TLL	USB host: See Note 5.	USB device: See Note 10.	USB host + TLL: See Note 14.	N/A
				Disabled	Available for non-USB usage		N/A	USB host: See Note 8.
	Yes	USB host	USB OTG	USB host with TLL	USB host: See Note 5.	USB OTG: See Note 11.	USB host + TLL: See Note 14.	N/A

Table 73. USB Signal Multiplexing Modes (Continued)

HMC_MODE	OTG Required?	USB Multiplexing Port 0	USB Multiplexing Port 1	USB Multiplexing Port 2	USB Pin Group 0	USB Pin Group 1	USB Pin Group 2	USB Alternate Pin Group 2
				Disabled	Available for non-USB usage		N/A	USB host: See Note 8.
12	No	USB device	USB host	USB host with TLL	USB device: See Note 1.	USB host: See Note 6.	USB host + TLL: See Note 14.	N/A
					Available for non-USB usage		N/A	USB device: See Note 2.
	Yes	USB OTG	USB host	USB host with TLL	USB OTG: See Note 3.	USB host: See Note 6.	USB host + TLL: See Note 14.	N/A
12 (cont)	Yes	USB OTG	USB host	USB host with TLL	Available for non-USB usage	USB host: See Note 6.	N/A	USB OTG: See Note 4.
13	No	USB host	Disabled	USB device with TLL	USB host: See Note 5.	Available for non-USB usage	USB device + TLL: See Note 15.	N/A
				Disabled	Available for non-USB usage		N/A	USB host: See Note 8.
14	No	USB device	Disabled	USB host with TLL	USB device: See Note 1.	Available for non-USB usage	USB host + TLL: See Note 14.	N/A
				Disabled	Available for non-USB usage		N/A	USB device: See Note 2.

Table 73. USB Signal Multiplexing Modes (Continued)

HMC_MODE	OTG Required?	USB Multiplexing Port 0	USB Multiplexing Port 1	USB Multiplexing Port 2	USB Pin Group 0	USB Pin Group 1	USB Pin Group 2	USB Alternate Pin Group 2
15	No	USB host	USB host	USB device with TLL	USB host: See Note 5.	USB host: See Note 6.	USB device + TLL: See Note 15.	N/A
				Disabled	Available for non-USB usage		N/A	USB host: See Note 8.
16	No	USB host	Disabled	Disabled	USB host: See Note 5.	Available for non-USB usage	Available for non-USB usage	N/A
					Available for non-USB usage		N/A	USB host: See Note 8.
	Yes	USB OTG	Disabled	Disabled	USB OTG: See Note 3.	Available for non-USB usage	Available for non-USB usage	N/A
					Available for non-USB usage		N/A	USB OTG: See Note 4.
17	No	USB host	USB host	Disabled	USB host: See Note 5.	USB host: See Note 6.	Available for non-USB usage	N/A
17 (cont)	No	USB host	USB host	Disabled	Available for non-USB usage	USB host: See Note 6.	N/A	USB host: See Note 8.
	Yes	USB OTG	USB host	Disabled	USB OTG: See Note 3.	USB host: See Note 6.	Available for non-USB usage	N/A
					Available for non-USB usage		N/A	USB OTG: See Note 4.
18	No	USB host	UART1	Disabled	USB host: See Note 5.	UART1: See Note 9.	Available for non-USB usage	N/A

Table 73. USB Signal Multiplexing Modes (Continued)

HMC_MODE	OTG Required?	USB Multiplexing Port 0	USB Multiplexing Port 1	USB Multiplexing Port 2	USB Pin Group 0	USB Pin Group 1	USB Pin Group 2	USB Alternate Pin Group 2
					Available for non-USB usage		N/A	USB host: See Note 8.
19	No	USB host	USB device	Disabled	USB host: See Note 5.	USB device: See Note 10.	Available for non-USB usage	N/A
					Available for non-USB usage		N/A	USB host: See Note 8.
	Yes	USB host	USB OTG	Disabled	USB host: See Note 5.	USB OTG: See Note 11.	Available for non-USB usage	N/A
					Available for non-USB usage		N/A	USB host: See Note 8.
20	No	USB device	USB host	Disabled	USB device: See Note 1.	USB host: See Note 6.	Available for non-USB usage	N/A
					Available for non-USB usage		N/A	USB device: See Note 2.
	Yes	USB OTG	USB host	Disabled	USB OTG: See Note 3.	USB host: See Note 6.	Available for non-USB usage	N/A
					Available for non-USB usage		N/A	USB OTG: See Note 4.
21	No	USB host	USB host port 2 internally connected to USB device via TLL. USB host port 1 disabled.		USB host: See Note 5.	Available for non-USB usage	Available for non-USB usage	N/A

Table 73. USB Signal Multiplexing Modes (Continued)

HMC_MODE	OTG Required?	USB Multiplexing Port 0	USB Multiplexing Port 1	USB Multiplexing Port 2	USB Pin Group 0	USB Pin Group 1	USB Pin Group 2	USB Alternate Pin Group 2
22	No	Disabled	Disabled	Disabled	Available for non-USB usage	Available for non-USB usage	Available for non-USB usage	N/A
23	No	USB host	USB host	USB host + TLL (TXDP/TXDM)	USB host: See Note 5.	USB host: See Note 6.	USB host + TLL: See Note 16.	N/A
				N/A	Available for non-USB usage		N/A	USB host: See Note 8.
	Yes	USB host	USB OTG	USB host + TLL (TXDP/TXDM)	USB host: See Note 5.	USB OTG: See Note 11.	USB host + TLL: See Note 16.	N/A
				N/A	Available for non-USB usage		N/A	USB host: See Note 8.
24	No	USB host	UART1	USB host + TLL (TXDP/TXDM)	USB host: See Note 5.	UART1: See Note 9.	USB host + TLL: See Note 16.	N/A
				N/A	Available for non-USB usage		N/A	USB host: See Note 8.
25	No	USB host	USB device	USB host + TXDP/TXDM	USB host: See Note 5.	USB device: See Note 10.	USB host + TLL: See Note 16.	N/A
				N/A	Available for non-USB usage		N/A	USB host: See Note 8.

Table 73. USB Signal Multiplexing Modes (Continued)

HMC_MODE	OTG Required?	USB Multiplexing Port 0	USB Multiplexing Port 1	USB Multiplexing Port 2	USB Pin Group 0	USB Pin Group 1	USB Pin Group 2	USB Alternate Pin Group 2
25 (cont)	Yes	USB host	USB OTG	USB host + TXDP/ TXDM	USB host: See Note 5.	USB OTG: See Note 11.	USB host + TLL: See Note 16.	N/A
				N/A	Available for non-USB usage		N/A	USB host: See Note 8.
Other HMC_MODE values	No	Disabled	Disabled	Disabled	Available for non-USB usage	Available for non-USB usage	Available for non-USB usage	N/A

Notes:
1) Figure 76 shows typical connectivity for USB pin group 0 acting as USB device.
2) Figure 80, Figure 81, or Figure 82 show connectivity options for USB alternate pin group 2 acting as USB device.
3) Figure Figure 59 shows typical connectivity for USB pin group 0 acting as USB OTG.
4) Figure 63, Figure 64, or Figure 65 show connectivity options for USB alternate pin group 2 acting as USB OTG.
5) Figure 66 shows typical connectivity for USB pin group 0 acting as USB host.
6) Figure 67, Figure 68, or Figure 69 show connectivity options for USB pin group 1 acting as USB host.
7) Figure 70, Figure 71, or Figure 72 show connectivity options for USB pin group 2 acting as USB host.
8) Figure 73, Figure 74, or Figure 75show connectivity options for USB alternate pin group 2 acting as USB host.
9) Figure 89 shows typical connectivity for USB pin group 1 acting as UART1.
10) Figure 77, Figure 78, or Figure 79 show connectivity options for USB pin group 1 acting as USB device.
11) Figure 60, Figure 61, or Figure 62 show connectivity options for USB pin group 1 acting as USB OTG.
12) Figure 63, Figure 64, or Figure 65 show connectivity options for USB alternate pin group 2 acting as USB OTG.
13) Figure 88 shows typical connectivity for USB pin group 1 to USB pin group 2 wrap mode.
14) See connectivity when OMAP5912 acts as the USB host on a transceiver-less link connection to an on-board USB device (Figure 85) using USB TXD/USB TXSE0 signaling.
15) See connectivity when OMAP5912 acts as the USB host on a transceiver-less link connection to an on-board USB device using USB TXD/USB TXSE0 signaling (Figure 86).
16) See connectivity when OMAP5912 acts as the USB host on a transceiver-less link connection to an on-board USB device using USB TXD/USB TXSE0 signaling (Figure 87).

USB controller ports that are not specifically mentioned for a row of Table 73 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 pin group 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 OMAP5912 device pins, but all of the pins associated with that pin group are

set for non-USB functionality, the associated USB multiplexing port sees 6-/8-wire transceiver signaling, which implies that both D+ and D- are low. If that pin group is associated with a host controller port, that host controller port appears as a disconnected USB link. If that pin group is associated with the device controller, the device controller sees USB reset signaling.

Functionality is undefined when top-level pin multiplexing selects non-USB signalling of one or more of the pins that are required by the selected HMC_MODE and transceiver signalling type. For example, if an HMC_MODE value selects USB pin group 2 as a USB host connection and a 4-wire transceiver interface is selected, but pin UART2.CTS/USB2.RCV top-level pin multiplexing selects signal UART2.CTS, the associated USB host controller port behavior is undefined.

Select USB and/or USB OTG Transceiver Type

USB connectivity on OMAP5912 USB pin group 1, USB pin group 2, and USB alternate pin group 2 require external components, except for pin group 2 when it is configured for a TLL mode. Several different types of external transceiver signaling are supported. Signaling between the OMAP5912 USB controller and the external USB transceiver for monitoring and controlling the differential USB signal can be via a 6-wire, 4-wire, or 3-wire signaling interface, with two or more additional control signals provided either by additional signals or via an I^2C link. OTG transceivers can support the 6-wire, 4-wire, and/or 3-wire basic signaling, but generally provide an interrupt output and an I^2C link for the additional control and status reporting required by OTG functionality.

The figures referenced in the USB Pin Group 0, USB Pin Group 1, USB Pin Group 2, and Alternate Pin Group 2 columns of Table 73 show external connectivity options for each of the possible selections. Carefully examine the options for the HMC_MODE you have identified.

Consider which OMAP5912 pins are required in each diagram and determine if any particular choice (3-wire, 4-wire, or 6-wire) is advantageous in terms of other non-USB functionality that is available for each pin group. 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 OMAP5912 pins that are required. This can assist in making non-USB functionality available on the USB pin groups.

Determine Proper Top-level Multiplexing Settings

Top-level pin multiplexing settings are determined primarily by which pins perform USB functions and the type of transceiver connected to the pins.

Pin multiplexing is controlled on a pin-by-pin basis via the top-level pin multiplexing. Each pin, referenced by default pin name, must be configured to provide the correct functional signal. To configure a pin for its USB functionality, determine the default pin name to be configured. For example, the default pin name for MCSI2.SYNC/USB0.SPEED is MCSI2.SYNC, and the default pin name for MCBSP3.CLKX/USB1.TXEN is MCBSP3.CLKX. See Table 74 to determine the name of the register and the name of the bit field in that register that controls multiplexing for the pin. Choose the desired USB signal name to be used on that pin, and use the associated value under pin multiplexing configuration value as the value to be programmed into the specified bit field in the specified pin multiplexing control register.

For example, to configure top-level pin multiplexing for UART2.CTS/USB0.RCV, locate the UART2.CTS in the default pin name column of Table 74. The register bit field that configures the top-level pin multiplexing for this pin is FUNC_MUX_CTRL_C.CONF_CTS2_R. Find USB0.RCV in the USB Signal Name column of the UART2.CTS row, and find that the Pin Multiplexing Configuration Value for USB2.RCV is 5. This means that writing a 5 to FUNC_MUX_CTRL_C.CONF_CTS2_R configures the UART2.CTS pin to act as USB0.RCV.

Table 74. Top-Level Pin Multiplexing Configuration for OMAP5912 USB-Related Pins

Default Pin Name	Pin Multiplexing Control Register	Bit Field	USB Signal Name	Pin Multiplexing Configuration Value
USB.DP	USB_TRANSCEIVER_CTRL	CONF_USB_PORT0_R	USB.DP	0
			USB.PUEN	7
USB.DM	USB_TRANSCEIVER_CTRL	CONF_USB_PORT0_R	USB.DM	0
			HIGH IMPEDANCE	7
USB.PUEN	FUNC_MUX_CTRL_D	CONF_USB_PUEN_R	USB.PUEN	0
			USB.CLK0	1
			USB.PUDIS	3
GPIO0	FUNC_MUX_CTRL_7	CONF_GPIO_0_R	USB.VBUS	2

Table 74. Top-Level Pin Multiplexing Configuration for OMAP5912 USB-Related Pins (Continued)

Default Pin Name	Pin Multiplexing Control Register	Bit Field	USB Signal Name	Pin Multiplexing Configuration Value
MCBSP3. CLKX	FUNC_MUX_CTRL_9	CONF_MCBSP3_ CLK_R	USB1.TXEN	2
MCSI1.DOUT	FUNC_MUX_CTRL_9	CONF_MCSI1_ DOUT_R	USB1.TXD	1
RTC_WAKE_ INT	FUNC_MUX_CTRL_9	CONF_WAKEUP_INT_ R	USB1_TXSE0	4
MCSI1.DIN	FUNC_MUX_CTRL_A	CONF_MCSI1_DIN_R	USB1.RCV	1
MCSI1.SYNC	FUNC_MUX_CTRL_A	CONF_MCSI1_SYNC_R	USB1.VP	2
MCSI1.BCLK	FUNC_MUX_CTRL_A	CONF_MCSI1_BCLK_R	USB1.VM	2
CLK32K_OUT	FUNC_MUX_CTRL_A	CONF_CLK32K_OUT_R	USB1.SPEED	5
MPU_BOOT	FUNC_MUX_CTRL_8	CONF_ARM_BOOT_R	USB1.SUSP	2
MCSI2.DOUT	FUNC_MUX_CTRL_B	CONF_MCSI2_ DOUT_R	USB2.TXEN	1
			USB0.TXEN	5
UART2.TX	FUNC_MUX_CTRL_C	CONF_TX2_R	USB2.TXD	2
			USB0.TXD	5
UART2.RTS	FUNC_MUX_CTRL_C	CONF_RTS2_R	USB2_TXSE0	2
			USB0_TXSE0	5
UART2.CTS	FUNC_MUX_CTRL_C	CONF_CTS2_R	USB2.RCV	1
			USB0.RCV	5
MCSI2.DIN	FUNC_MUX_CTRL_B	CONF_MCSI2_DIN_R	USB2.VP	1
			USB0.VP	5
UART2.RX	FUNC_MUX_CTRL_C	CONF_RX2_R	USB2.VM	1
			USB0.VM	5
MCSI2.SYNC	FUNC_MUX_CTRL_B	CONF_MCSI2_SYNC_R	USB2.SPEED	2
			USB0.SPEED	5

Table 74. Top-Level Pin Multiplexing Configuration for OMAP5912 USB-Related Pins (Continued)

Default Pin Name	Pin Multiplexing Control Register	Bit Field	USB Signal Name	Pin Multiplexing Configuration Value
MCSI2.CLK	FUNC_MUX_CTRL_B	CONF_MCSI2_CLK_R	USB2.SUSP	1
			USB0.SUSP	5

When a 3-wire or 4-wire (bidirectional) transceiver is connected to USB pin group 1, USB_TRANSCEIVER_CTRL.CONF_USB1_UNI_R must be 0. When a 6-wire or 8-wire (unidirectional) transceiver is connected to USB pin group 1, USB_TRANSCEIVER_CTRL.CONF_USB1_UNI_R must be 1.

When a 3-wire or 4-wire (bidirectional) transceiver is connected to USB pin group 2 or USB alternate pin group 2, USB_TRANSCEIVER_CTRL.CONF_USB2_UNI_R must be 0. When a 6-wire or 8-wire (unidirectional) transceiver is connected to USB pin group 2 or USB alternate pin group 2, USB_TRANSCEIVER_CTRL.CONF_USB2_UNI_R must be 1.

Determine OTG Module Control Register Settings

The OTG module 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 being used.

OTG_SYSCON_1.USB0_TRX_MODE must be 0 or 3 to allow proper operation of the integrated USB transceiver on USB pin group 0. For cases where USB alternate pin group 2 is used, OTG_SYSCON_1.USB0_TRX_MODE must be set to match the type of transceiver used on USB alternate pin group 3 (3 for a 6-wire transceiver, 1 for a 4-wire transceiver, or 2 for 3-wire transceiver).

OTG_SYSCON_1.USB1_TRX_MODE must be set to match the type of transceiver connected to USB pin group 1. When USB pin group 1 is not connected to a transceiver, those pins must be configured for non-USB operation and OTG_SYSCON_1.USB1_TRX_MODE must be set to 0.

OTG_SYSCON_1.USB2_TRX_MODE must be set to match the type of transceiver connected to USB pin group 2. When alternate USB pin group 2 is used, OTG_SYSCON_1.USB2_TRX_MODE has no effect and can be set to 0.

OTG_SYSCON_2.USBx_SYNCHRO must be set to 1 for proper transceiver operation.

OTG_SYSCON_2.OTG_PADEN must be set to 0 whenever OTG functionality is required. Writing a 1 to this bit prevents proper operation of the OTG_CTRL register. When OTG_PADEN is 0 and OTG_EN = 0, the GPIO0/USB.VBUS input is not provided to the USB device controller. In this case, software monitors GPIO0 and updates OTG_CTRL.BSESSVLD with the value from GPIO0.

When OTG_PADEN is 1 and GPIO_0/USB.VBUS is configured for USB.VBUS functionality, GPIO0/USB.VBUS propagates to the USB device controller without software intervention, but the OTG_CTRL register bits ASESSVLD, BSESSEND, BSESSVLD, VBUSVLD, and ID are read-only and are always 0. Because of this, it is impossible to implement USB-OTG functionality when OTG_PADEN is 1.

OTG_SYSCON_2.HMC_PADEN determines whether values for UHOST_EN, HMC_MODE, TLL_ATTACH, and TLL_SPEED are provided by bits in OTG_SYSCON_2 or by bits in MOD_CONF_CTRL_0, as shown in Table 75. For software compatibility with future devices, HMC_PADEN must be set to 0.

Table 75. UHOST_EN, HMC_MODE, TLL_ATTACH, TLL_SPEED Source Selection

Register	Field	Value	
OTG_SYSCON2	HMC_PADEN	0	1
MOD_CONF_CTRL_0	UHOST_EN	Don’t care	Valid
	HMC_MODE		
	HMC_TLLATTACH		
	HMC_TLLSPEED		
OTG_SYSCON_2	CONF_MOD_USB_HOST_HHC_UHOST_EN_R	Valid	Don’t care
	CONF_MOD_USB_HOST_HMC_MODE_R		
	CONF_MOD_USB_HOST_HMC_TLL_ATTACH_R		
	CONF_MOD_USB_HOST_HMC_TLL_SPEED_R		

Table 75. UHOST_EN, HMC_MODE, TLL_ATTACH, TLL_SPEED Source Selection (Continued)

Register	Field	Value	
Values used	UHOST_EN	Values from OTG_ SYSCON_2	Values from MOD_CONF_ CTRL_0
	HMC_MODE		
	TLL_ATTACH		
	TLL_SPEED		

If OTG_SYSCON_2.HMC_PADEN is 0, OTG_SYSCON_2.HMC_MODE must be programmed with the HMC_MODE value chosen from Table 73. If OTG_SYSCON_2.HMC_PADEN is 1, the chosen HMC_MODE value must be written to MOD_CONF_CTRL_0.CONF_MOD_USB_HOST_HMC_ MODE_R.

4.5 Transceiver Signaling Types

USB Transceiver Unidirectional Signaling

When a USB or USB OTG transceiver is connected to OMAP5912 and is used in unidirectional DAT/SE0 signaling mode, the signaling shown in Table 76 is used.

Table 76. Signaling Between USB Controller and Unidirectional USB Transceiver Using DAT/SE0 Signaling

Logical Signal Name(s)	OMAP5912 Pin Direction	Transceiver Pin Direction	Description				
$\overline{OE}$	Output	Input	When low, USB transceiver drives D+ and D-.				
DAT and SE0	Output	Input	Controls the values output by the USB transceiver on D+ and D- when $\overline{OE}$ is low. Ignored when $\overline{OE}$ is high.				
			OEn	DAT	SE0	D+	D-
			0	0	0	0	1
				1	0	1	0
				X	1	0	0
			1	X	X	Undriven	Undriven

Table 76. Signaling Between USB Controller and Unidirectional USB Transceiver Using DAT/SE0 Signaling (Continued)

Logical Signal Name(s)	OMAP5912 Pin Direction	Transceiver Pin Direction	Description
RCV	Input	Output	Output from transceiver differential receiver. D+ D- RCV 0 0 X 0 1 0 1 0 1 1 1 X
RCVVP	Input	Output	Output from transceiver single-ended D+ signal receiver . D+ RCVVP 0 0 1 1
RCVVM	Input	Output	Output from transceiver single-ended D- signal receiver. D- RCVVM 0 0 1 1
SPEED	Output	Input	Determines transceiver D+, D- output characteristics. 0 = Low speed. 1 = Full speed. Transceivers with I^2C interfaces can implement this functionality as a control bit rather than a pin.
SUSPEND	Output	Input	Controls transceiver powerdown modes. 0 = Active mode. 1 = Low-power mode Transceivers with I^2C interfaces can implement this functionality as a control bit rather than a pin.

OMAP5912 does not support unidirectional transceivers that implement VPO/VMO signaling.

USB Transceiver 3-Wire Bidirectional Signaling

When a USB or USB OTG transceiver is connected to OMAP5912 and is used in 3-wire bidirectional TXDAT/TXSE0 signaling mode, the signaling shown in Table 77 is used.

Table 77. Signaling Between USB Controller and 3-Wire Bidirectional USB Transceiver Using TXDAT/TXSE0 Signaling

Logical Signal Name	OMAP5912 Pin Direction	Transceiver Pin Direction	Description				
$\overline{OE}$	Output	Input	When low, USB transceiver drives D+ and D-.				
TXDAT/ RCVDAT and TXSE0/ RCVSE0			When $\overline{OE}$ is low, OMAP5912 drives TXDAT and TXSE0 and the transceiver drives D+ and D- based on the values of TXDAT and TXSE0.				
	Output	Input	OEn	TXDAT	TXSE0	D+	D-
			0	0	0	0	1
				1	0	1	0
				X	1	0	0
	Input	Output	OEn	D+	D-	RCVDAT	RCVSE0
			1	0	0	0	1
				0	1	0	0
				1	0	1	0
				1	1	Undefined	Undefined
SPEED	Output	Input	Determines transceiver D+, D- output characteristics. 0 = Low speed 1 = Full speed Transceivers with I^2C interfaces can implement this functionality as a control bit rather than a pin.				
SUSPEND	Output	Input	Controls transceiver powerdown modes. 0 = Active mode 1 = Low-power mode Transceivers with I^2C interfaces can implement this functionality as a control bit rather than a pin.				

OMAP5912 does not support 3-wire bidirectional signaling using VP/VM signals.

USB Transceiver 4-Wire Bidirectional Signaling

When a USB or USB OTG transceiver is connected to OMAP5912 and is used in 4-wire bidirectional DAT/SE0 signaling mode, the signaling shown in Table 78 is used.

Table 78. Signaling Between USB Controller and 4-Wire Unidirectional USB Transceiver Using VP/VM Signaling

Logical Signal Name	OMAP5912 Pin Direction	Transceiver Pin Direction	Description		
$\overline{OE}$	Output	Input	When low, USB transceiver drives D+ and D-.		
TXVM/ RCVVM			Value driven to or received from D-		
	Output	Input	OEn	TXVM	D-
			0	0	0
			0	1	1
	Input	Output	OEn	D-	RCVVM
			1	0	0
			1	1	1
TXVP/ RCVVP			Value driven to or received from D+		
	Output	Input	OEn	TXVP	D+
			0	0	0
			0	1	1
	Input	Output	OEn	D+	RCVVP
			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

Table 78. Signaling Between USB Controller and 4-Wire Unidirectional USB Transceiver Using VP/VM Signaling (Continued)

Logical Signal Name	OMAP5912 Pin Direction	Transceiver Pin Direction	Description
SPEED	Output	Input	Determines transceiver D+, D- output characteristics. 0 = Low speed 1 = Full speed Transceivers with I^2C interfaces can implement this functionality as a control bit rather than a pin.
SUSPEND	Output	Input	Controls transceiver power-down modes. 0 = Active mode 1 = Low-power mode Transceivers with I^2C interfaces can implement this functionality as a control bit rather than a pin.

OMAP5912 does not support 4-wire bidirectional signaling using DAT/SE0 signals.

4.6 USB OTG External Connectivity

To provide USB OTG connectivity, a dual-role device system that provides a USB OTG controller must implement certain features. These features include a USB mini-AB receptacle, VBUS monitoring and control, transient suppression, ID monitoring, controllable D+ and D- series, pullup and pulldown resistors, and a D+ and D- driver/receiver. USB OTG transceivers that implement many of these features are commercially available.

Some USB OTG transceivers implement a 6-wire connection to the OTG controller, whereas others require only 4 wires or 3 wires. The OMAP5912 device must be properly configured to support the signaling required by the attached transceiver.

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 VBUS state changes, when ID state changes, or when any number of other transceiver-dependent conditions

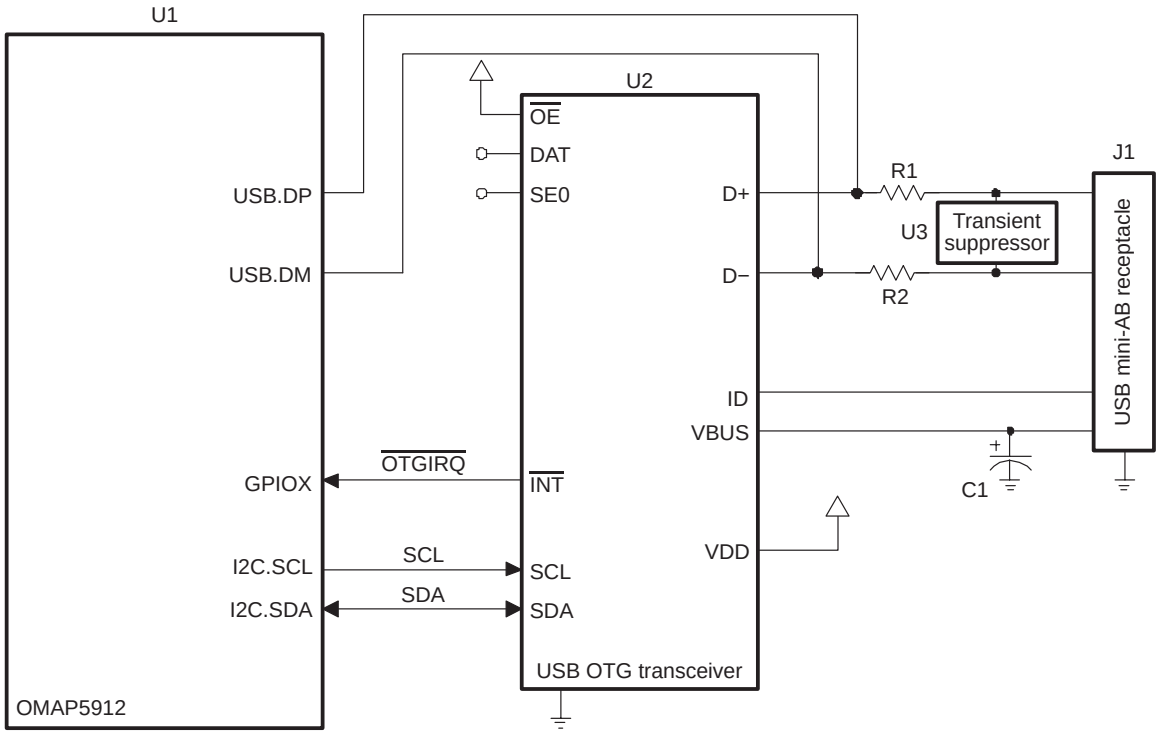
occur. When OTG transceiver interrupts occur, system software must query the OTG transceiver status using I^2C and update the OTG controller registers as appropriate.

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 USB OTG specification 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 is capable of supplying the amount of VBUS current required by the USB OTG specification, might 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.

Pin Group 0 OTG

The integrated USB transceiver on OMAP5912 USB pin group 0 does not provide the functionality specific to OTG transceivers. To implement an OTG dual-role device using OMAP5912 USB pin group 0, an external OTG transceiver is necessary to provide that additional functionality. In such a system, the external OTG transceiver provides the VBUS detect and drive, D+ and D- pullup and pulldown, and ID detection features, while the OMAP5912 integrated USB transceiver provides the D+ and D- signal drive and receive functions (see Figure 59).

Figure 59. OTG on USB Pin Group 0



R1, R2 27 Ohm 5%
C1 VBUS capacitance must meet OTG spec C_{DRD_VBUS}
U1 OMAP5912
U2 USB OTG transceiver
U3 Transient suppressor, like SN65220, SN65240, or SN75240
J1 USB mini-AB receptacle

Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for pins USB.DP and USB.DM and selection of an HMC_MODE value that routes the OTG controller port to pin group 0 (see Table 73). OTG_SYSCON_1.USB0_TRX_MODE must be set to 3 to allow proper operation of the integrated USB transceiver.

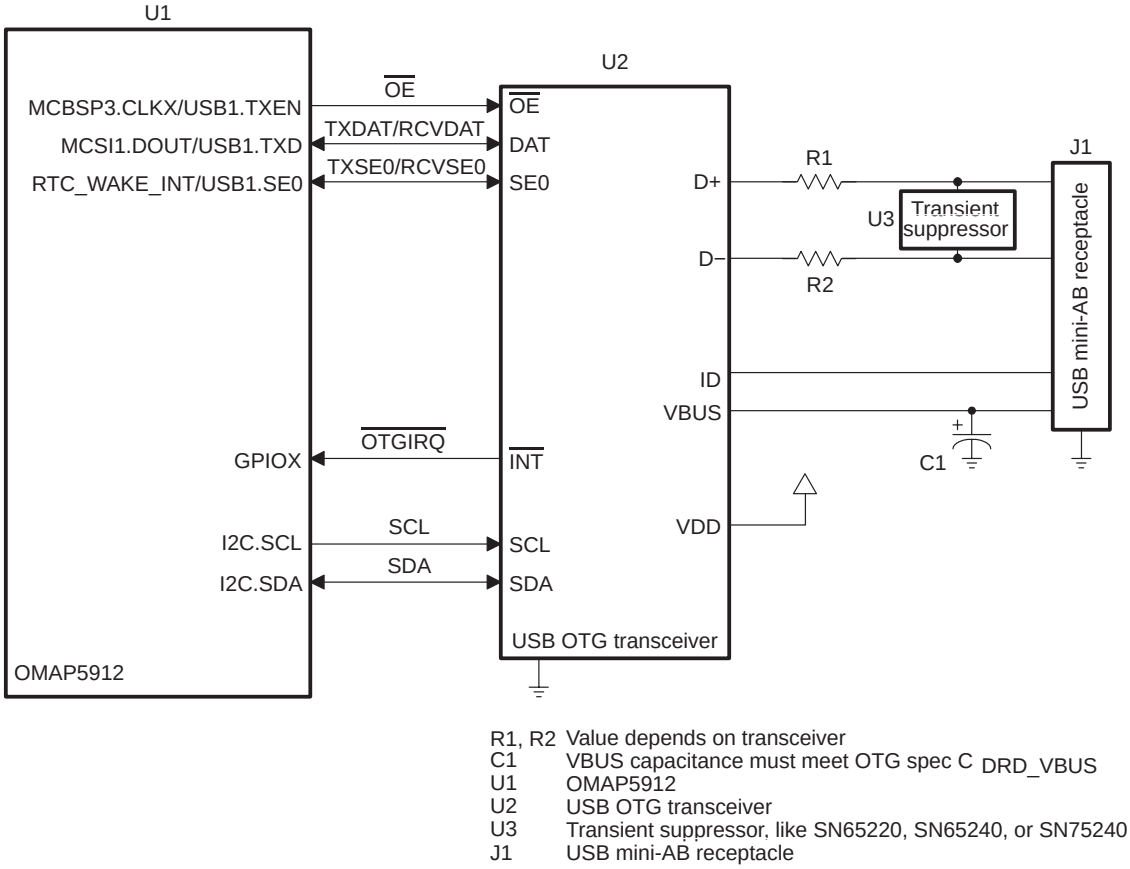
System software is responsible for transferring control signals from the OMAP5912 OTG controller to the OTG transceiver and status information from the OTG transceiver to the OMAP5912 OTG controller registers.

When acting as an OTG default-A dual-role device, system software must make use of the OTG transceiver for autoconnect mode. System software must also configure the OTG transceiver to suspend mode whenever OMAP5912 acts as a default-A device and is placing the OTG link into suspend so that the remote-B device can issue an HNP.

If you want to use the external OTG transceiver ability to provide I^2C signaling to the transceiver D+ and D- pins, it is necessary to control the OTG transceiver OEn input using an OMAP5912 GPIO pin. It is also necessary to set USB_TRANSCEIVER_CTRL.CONF_USB0_ISOLATE to 1 to ensure that the I^2C signals have no effect on the OMAP5912 USB controllers.

OTG on Pin Group 1 Using 3-Wire OTG Transceiver

Figure 60. OTG on USB Pin Group 1 Using 3-Wire OTG Transceiver



Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

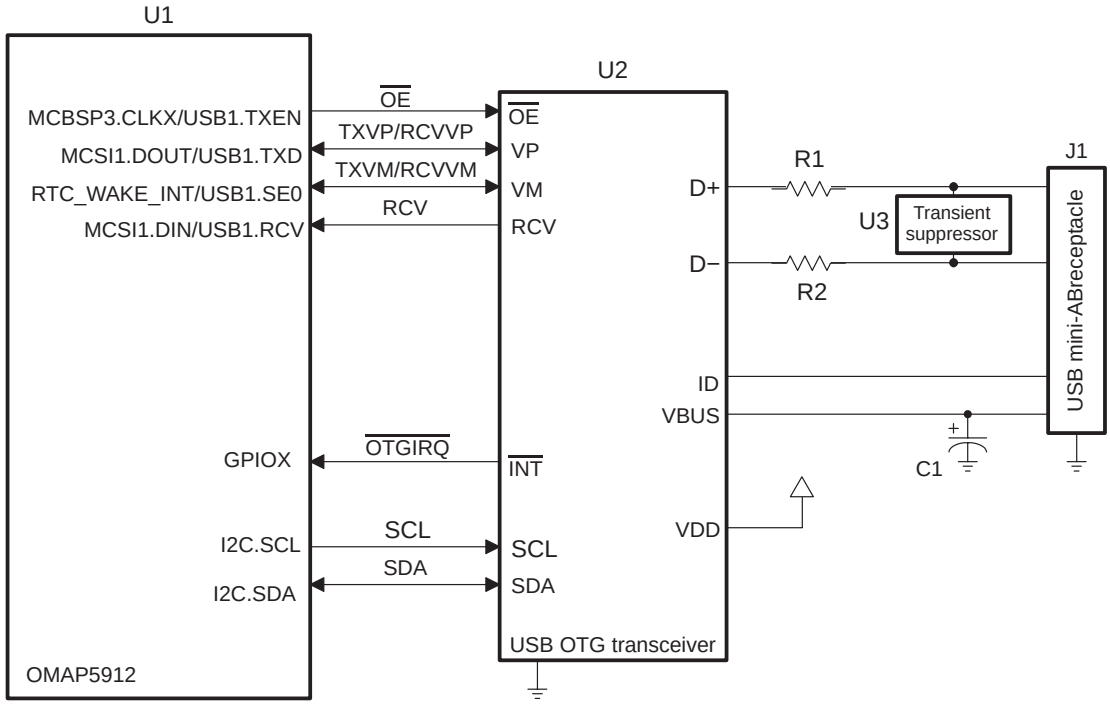
- ❏ MCBSP3.CLKX/USB1.TXEN
- ❏ MCSI1.DOUT/USB1.TXD
- ❏ RTC_WAKE_INT/USB1.SE0

See Table 74.

HMC_MODE must be set to a value that routes OTG functionality to USB pin group 1 (see Table 73). OTG_SYSCON_1.USB1_TRX_MODE must be set to 2 to allow proper bidirectional operation of the 3-wire USB transceiver. OTG_SYSCON_2.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins MCSI1.DOUT/USB1.TXD and RTC_WAKE_INT/USB1.SE0.

OTG on Pin Group 1 Using 4-Wire OTG Transceiver

Figure 61. OTG on USB Pin Group 1 Using 4-Wire OTG Transceiver



R1, R2 Value depends on transceiver
C1 VBUS capacitance must meet OTG spec C DRD_VBUS
U1 OMAP5912
U2 USB OTG transceiver
U3 Transient suppressor, like SN65220, SN65240, or SN75240
J1 USB mini-AB receptacle

Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

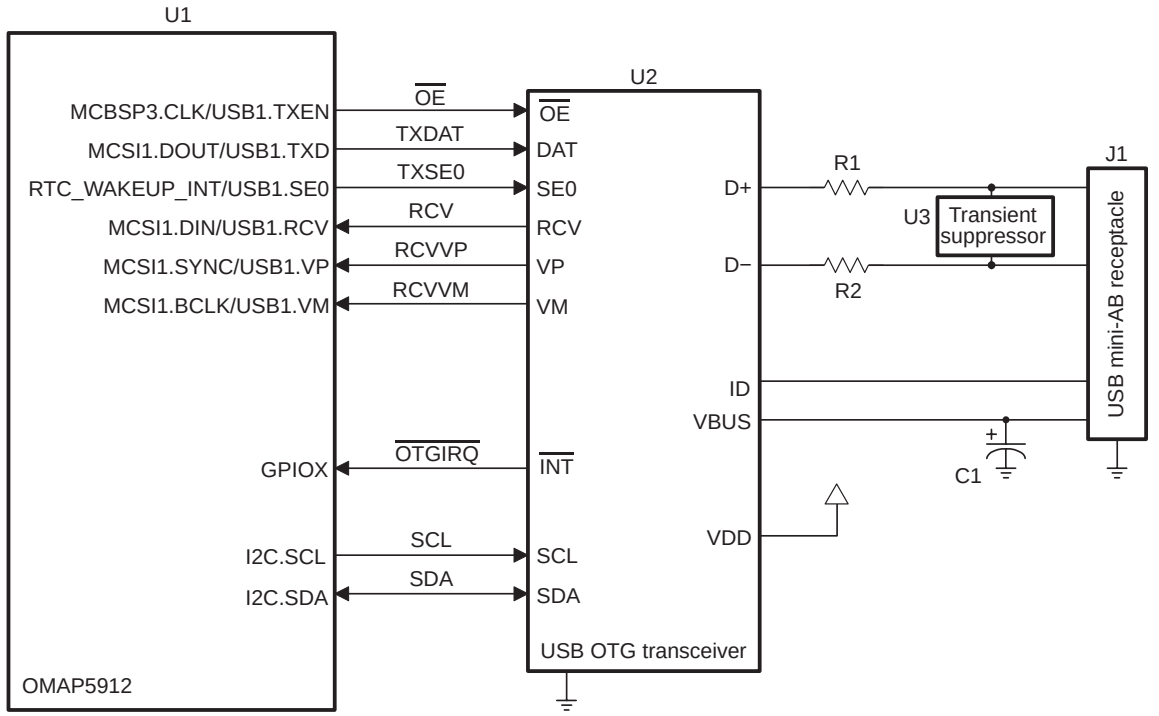
- ❏ MCBSP3.CLKX/USB1.TXEN
- ❏ MCSI1.DOUT/USB1.TXD
- ❏ RTC_WAKE_INT/USB1.SE0
- ❏ MCSI1.DIN/USB1.RCV

See Table 74.

HMC_MODE must be set to a value that provides OTG functionality on USB pin group 1 (see Table 73). OTG_SYSCON_1.USB1_TRX_MODE must be set to 1 to allow proper bidirectional operation of the 4-wire USB transceiver. OTG_SYSCON_2.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins MCSI1.DOUT/USB1.TXD and RTC_WAKE_INT/USB1.SE0.

OTG on Pin Group 1 OTG Using 6-Wire OTG Transceiver

Figure 62. OTG on USB Pin Group 1 Using 6-Wire OTG Transceiver



R1, R2 Value depends on transceiver
C1 VBUS capacitance must meet OTG spec C DRD_VBUS
U1 OMAP5912
U2 USB OTG transceiver
U3 Transient suppressor, like SN65220, SN65240, or SN75240
J1 USB mini-AB receptacle

Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

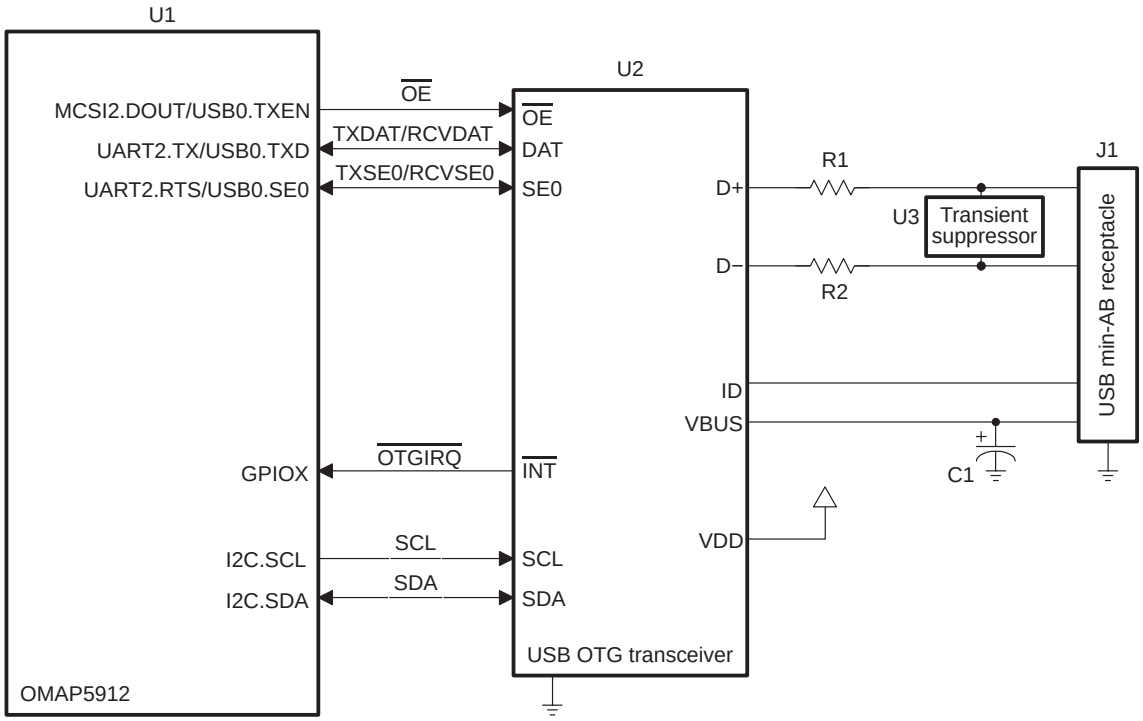
- ❏ MCBSP3.CLKX/USB1.TXEN
- ❏ MCSI1.DOUT/USB1.TXD
- ❏ RTC_WAKE_INT/USB1.SE0
- ❏ MCSI1.SYNC/USB1.VP
- ❏ MCSI1.BCLK/USB1.VM
- ❏ MCSI1.DIN/USB1.RCV

See Table 74.

HMC_MODE must be set to a value that provides OTG functionality on USB Pin group 1 (see Table 73). OTG_SYSCON_1.USB1_TRX_MODE must be set to 3 to allow proper operation of the 6-wire USB transceiver.

OTG on Alternate Pin Group 2 Using 3-Wire OTG Transceiver

Figure 63. OTG on USB Alternate Pin Group 2 Using 3-Wire OTG Transceiver



R1, R2 Value depends on transceiver
C1 VBUS capacitance must meet OTG spec C DRD_VBUS
U1 OMAP5912
U2 USB OTG transceiver
U3 Transient suppressor, like SN65220, SN65240, or SN75240
J1 USB mini-AB receptacle

Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

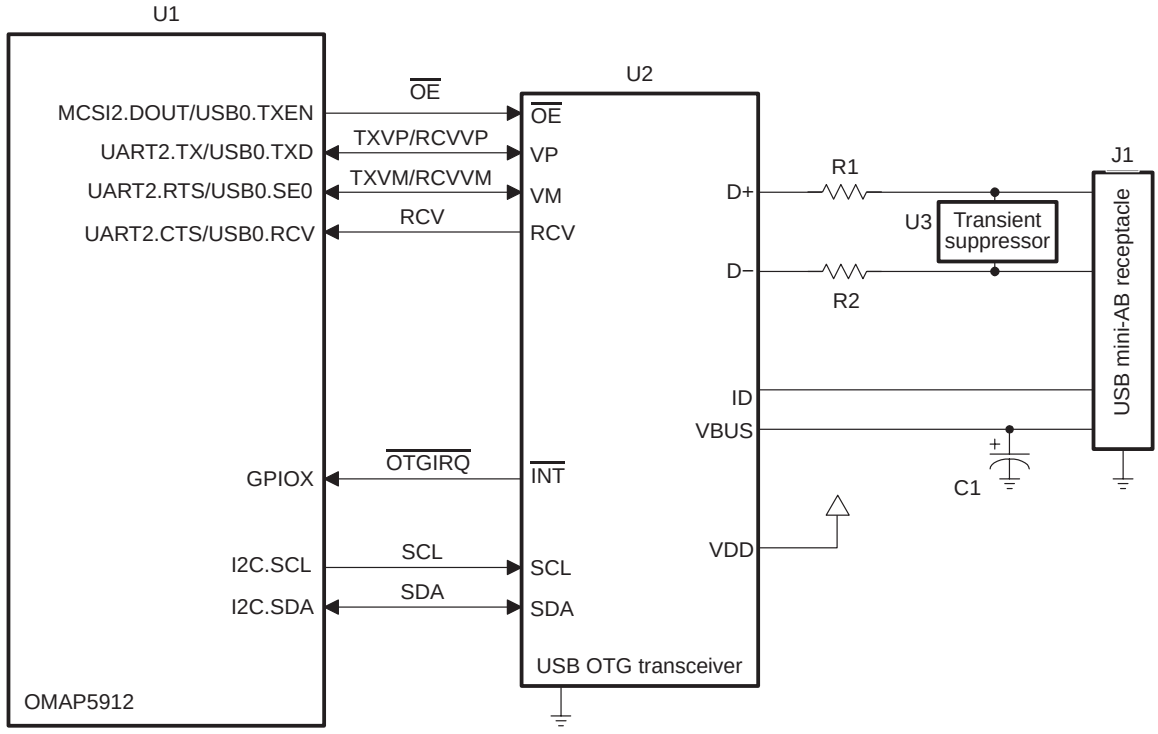
- ❏ MCSI2.DOUT/USB0.TXEN
- ❏ UART2.TX/USB0.TXD
- ❏ UART2.RTS/USB0.SE0

See Table 74.

HMC_MODE must be set to a value that provides OTG functionality on USB alternate pin group 2 (see Table 73). OTG_SYSCON_1_TRX_MODE must be set to 2 to allow proper bidirectional operation of the 3-wire USB transceiver. OTG_SYSCON_2.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins UART2.TX/USB0.TXD and UART2.RTS/USB0.SE0.

OTG on Alternate Pin Group 2 Using 4-Wire OTG Transceiver

Figure 64. OTG on USB Alternate Pin Group 2 Using 4-Wire OTG Transceiver



R1, R2 Value depends on transceiver
C1 VBUS capacitance must meet OTG spec C_{DRD_VBUS}
U1 OMAP5912
U2 USB OTG transceiver
U3 Transient suppressor, like SN65220, SN65240, or SN75240
J1 USB mini-AB receptacle

Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- ❏ MCSI2.DOUT/USB0.TXEN
- ❏ UART2.TX/USB0.TXD
- ❏ UART2.RTS/USB0.SE0
- ❏ UART2.CTS/USB0.RCV

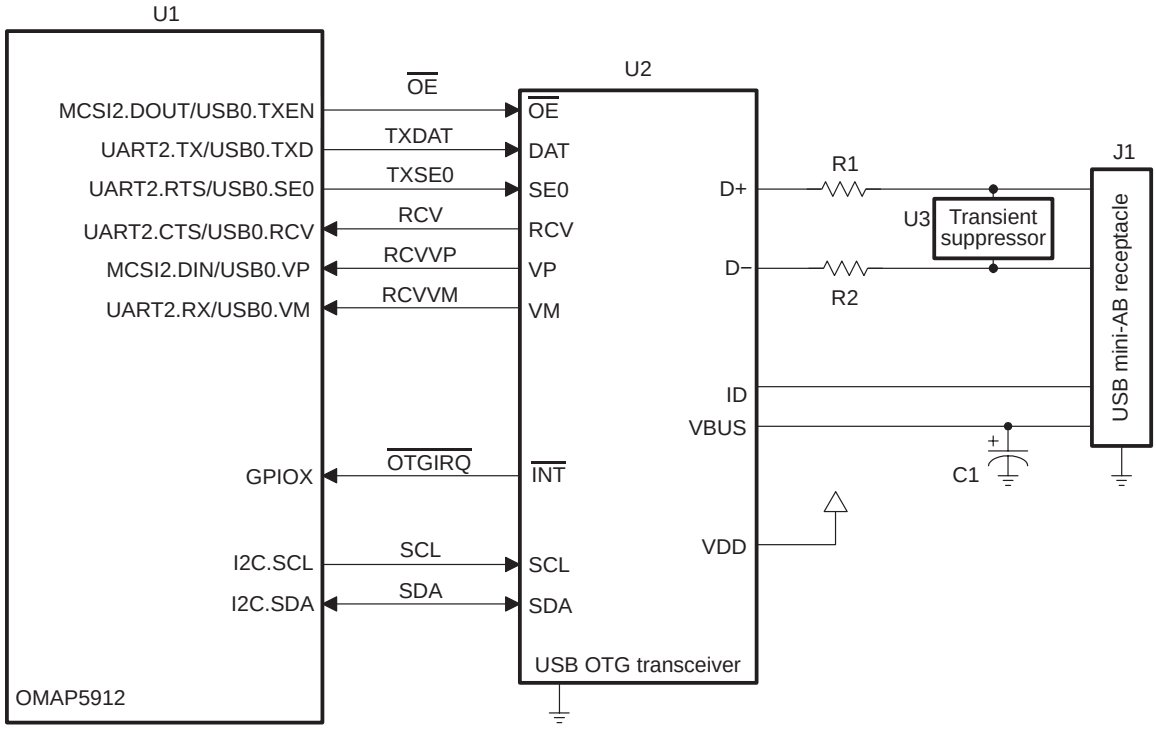
See Table 74.

HMC_MODE must be set to a value that provides OTG functionality on USB alternate pin group 2 (see Table 73).

OTG_SYSCON_1.USB2_TRX_MODE must be set to 1 to allow proper bidirectional operation of the 4-wire USB transceiver. OTG_SYSCON_2.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins UART2.TX/USB0.TXD and UART2.RTS/USB0.SE0.

OTG on Alternate Pin Group 2 Using 6-Wire OTG Transceiver

Figure 65. OTG on USB Alternate Pin Group 2 Using 6-Wire OTG Transceiver



R1, R2 Value depends on transceiver
C1 VBUS capacitance must meet OTG spec C DRD_VBUS
U1 OMAP5912
U2 USB OTG transceiver
U3 Transient suppressor, like SN65220, SN65240, or SN75240
J1 USB mini-AB receptacle

Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- ❑ MCSI2.DOUT/USB0.TXEN
- ❑ UART2.TX/USB0.TXD
- ❑ UART2.RTS/USB0.SE0
- ❑ UART2.CTS/USB0.RCV
- ❑ MCSI2.DIN/USB0.VP
- ❑ UART2.RX/USB0.VM

See Table 74.

HMC_MODE must be set to a value that provides OTG functionality on USB Alternate Pin group 2 (see Table 73). OTG_SYSCON_1.USB1_TRX_MODE must be set to 3 to allow proper operation of the 6-wire USB transceiver.

4.7 Host Controller Connectivity With USB Transceivers

To provide a robust USB solution, a system that provides a USB host controller must implement certain features. These features include a type-A receptacle, power on the VBUS signal (can be switched or unswitched power), transient suppression, pulldown resistors, and USB-compatible downstream port transceiver.

Because OMAP5912 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, OMAP5912 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.

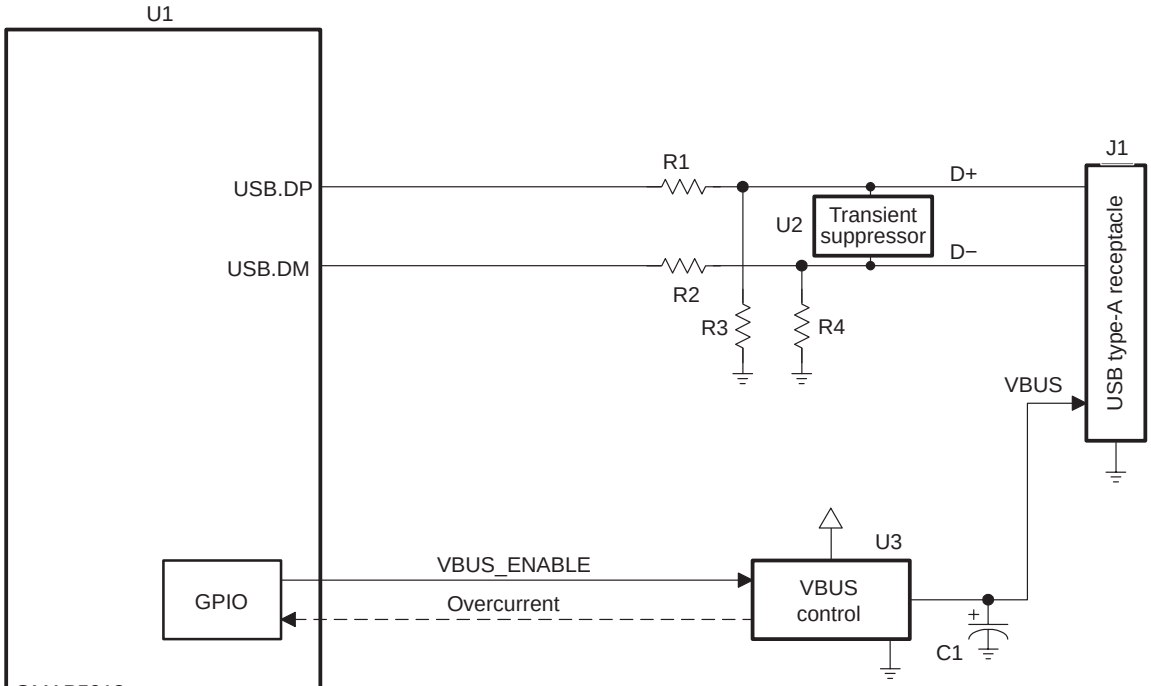
Some USB transceivers implement a 6-wire connection to the host controller, whereas others require only 4 wires or 3 wires. The OMAP5912 device must be properly configured to support the signaling required by the attached transceiver.

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.

USB Host Connections Using the OMAP5912 Integrated USB Transceiver

The OMAP5912 integrated USB transceiver can provide connectivity for a host-controller port. Figure 66 shows a way to connect the integrated USB transceiver as a host-only port.

Figure 66. USB Host Connections Using the OMAP5912 Integrated USB Transceiver



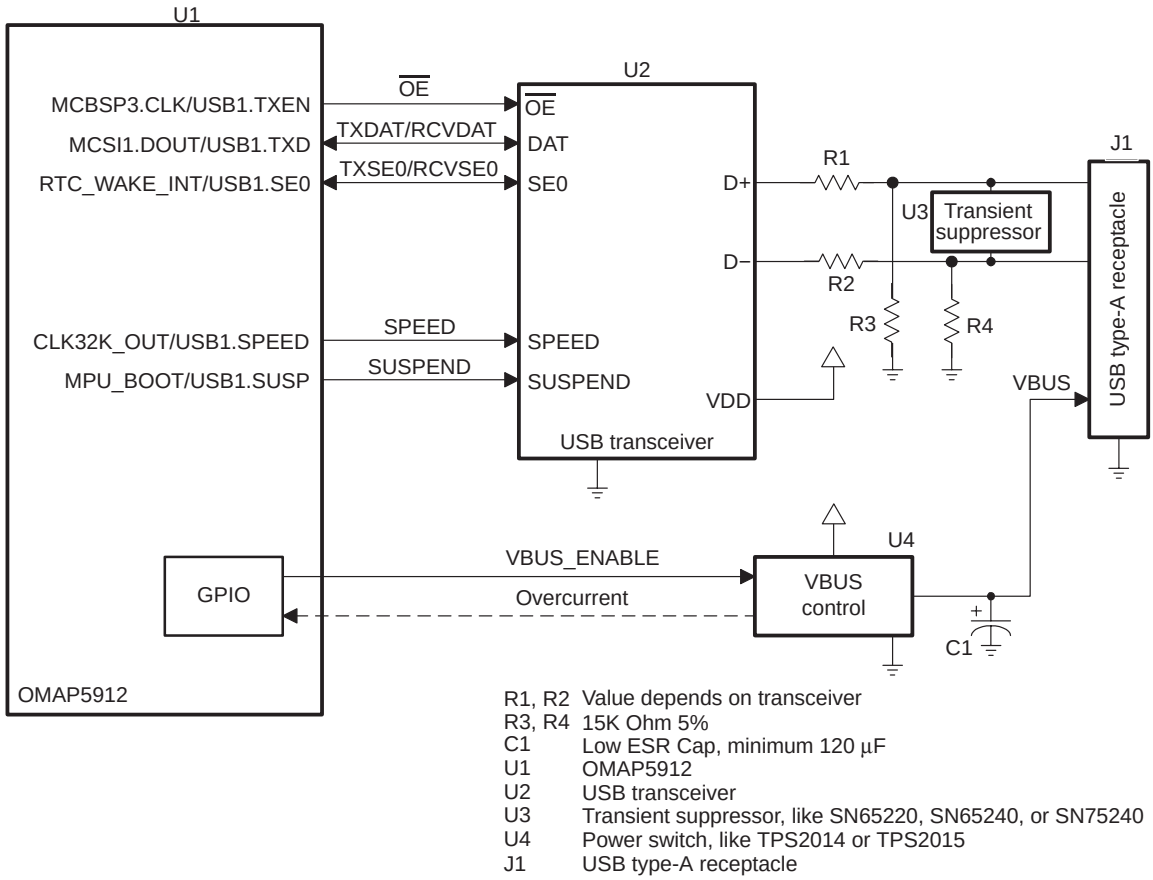
R1, R2 Value depends on transceiver
R3, R4 15K Ohm 5%
C1 Low ESR Cap, minimum 120 µF
U1 OMAP5912
U2 Transient suppressor, like SN65220, SN65240, or SN75240
U3 Power switch, like TPS2014 or TPS2015
J1 USB type-A receptacle

Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for pins USB.DP and USB.DM (see Table 74) and selection of an HMC_MODE value, which routes a host controller port to pin group 0 (see Table 73). OTG_SYSCON_1.USB0_TRX_MODE must be set to 3 to allow proper operation of the integrated USB transceiver.

OMAP5912 integrated pulldowns for pins USB.DP and USB.DM do not meet the USB specifications for D+ and D− pulldowns. The external resistors shown in Figure 66 must be implemented when using USB.DP and USB.DM for a host connection.

Pin Group 1 USB Host Using 3-Wire USB Transceiver

Figure 67. USB Host Connections On USB Pin Group 1 Using 3-Wire Transceiver



R1, R2	Value depends on transceiver
R3, R4	15K Ohm 5%
C1	Low ESR Cap, minimum 120 µF
U1	OMAP5912
U2	USB transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
U4	Power switch, like TPS2014 or TPS2015
J1	USB type-A receptacle

Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- ❏ MCBSP3.CLKX/USB1.TXEN
- ❏ MCSI1.DOUT/USB1.TXD
- ❏ RTC_WAKE_INT/USB1.SE0
- ❏ CLK32K_OUT/USB1.SPEED
- ❏ MPU_BOOT/USB1.SUSP

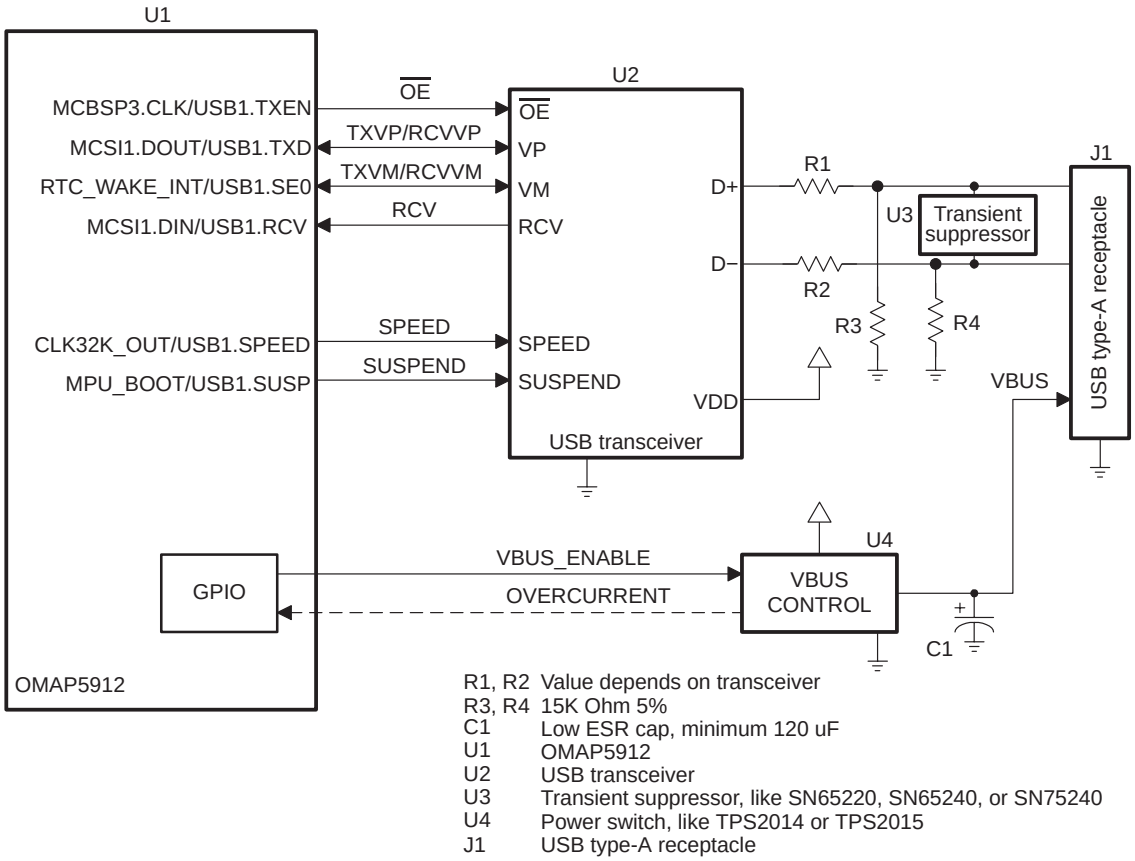
See Table 74.

HMC_MODE must be set to a value that routes a host controller port to USB pin group 1 (see Table 73). OTG_SYSCON_1.USB1_TRX_MODE must be set to 2 to allow proper bidirectional operation of the 3-wire USB transceiver.

OTG_SYSCON_2.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins MCSI1.DOUT/USB1.TXD and RTC_WAKE_INT/USB1.SE0.

Pin Group 1 USB Host Using 4-Wire USB Transceiver

Figure 68. USB Host Connections on USB Pin Group 1 Using 4-Wire Transceiver



Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

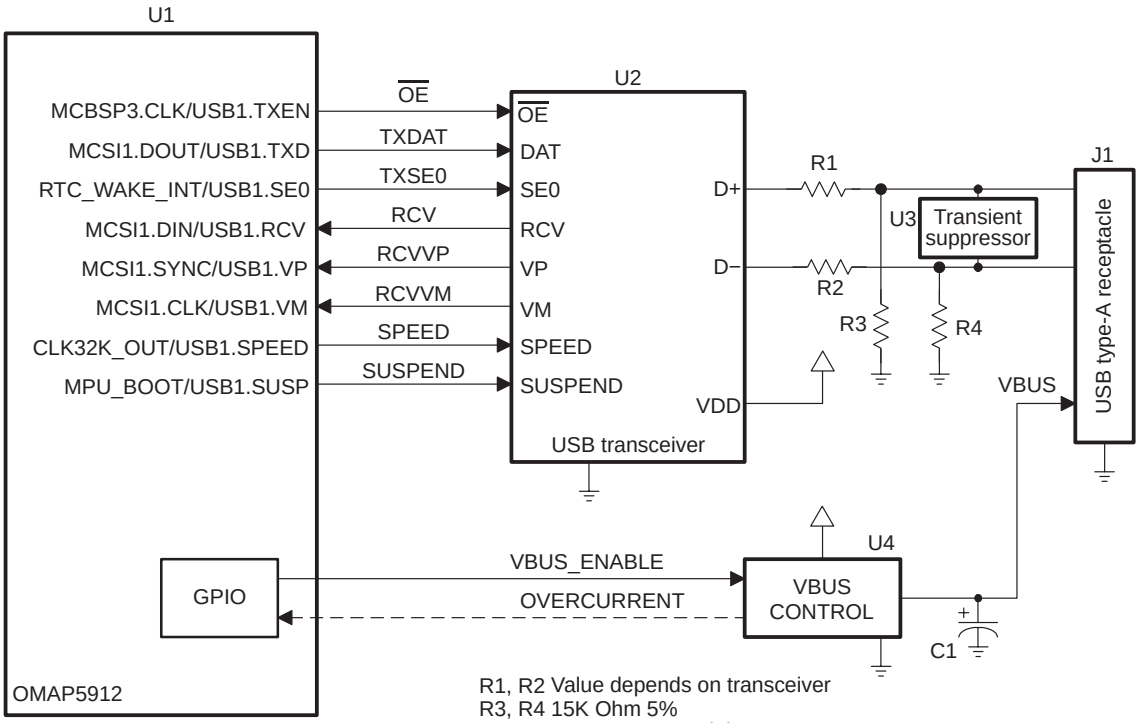
- ❏ MCBSP3.CLKX/USB1.TXEN
- ❏ MCSI1.DOUT/USB1.TXD
- ❏ RTC_WAKE_INT/USB1.SE0
- ❏ MCSI1.DIN/USB1.RCV
- ❏ CLK32K_OUT/USB1.SPEED
- ❏ MPU_BOOT/USB1.SUSP

See Table 74.

HMC_MODE must be set to a value that routes a host controller port to USB pin group 1 (see Table 73). OTG_SYSCON_1.USB1_TRX_MODE must be set to 1 to allow proper bidirectional operation of the 4-wire USB transceiver. OTG_SYSCON_2.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins MCSI1.DOUT/USB1.TXD and RTC_WAKE_INT/USB1.SE0.

Pin Group 1 USB Host Using 6-Wire USB Transceiver

Figure 69. USB Host Connections on USB Pin Group 1 Using 6-Wire OTG Transceiver



Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

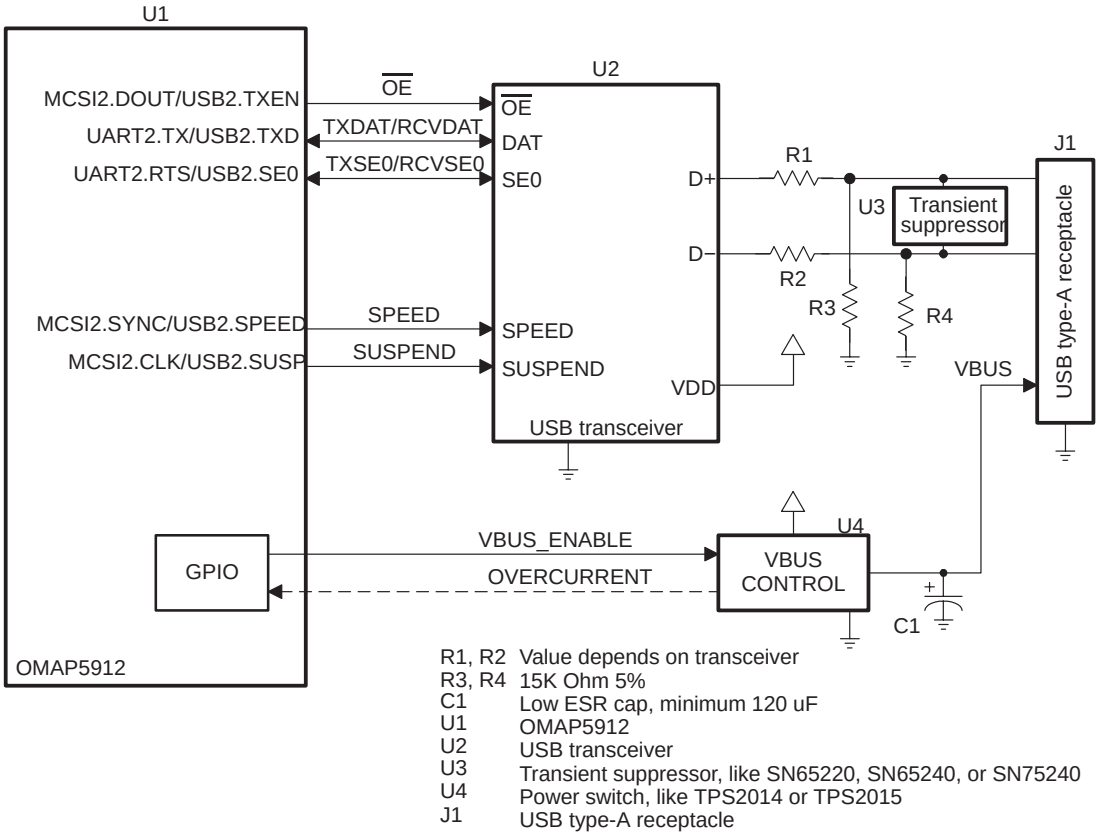
- MCBSP3.CLKX/USB1.TXEN
- MCSI1.DOUT/USB1.TXD
- RTC_WAKE_INT/USB1.SE0
- MCSI1.SYNC/USB1.VP
- MCSI1.BCLK/USB1.VM
- MCSI1.DIN/USB1.RCV
- CLK32K_OUT/USB1.SPEED
- MPU_BOOT/USB1.SUSP

See Table 74.

HMC_MODE must be set to a value that routes a host controller port to USB pin group 1 (see Table 73). OTG_SYSCON_1.USB1_TRX_MODE must be set to 3 to allow proper unidirectional operation of the 6-wire USB transceiver.

Pin Group 2 USB Host Using 3-Wire USB Transceiver

Figure 70. USB Host Connections on USB Pin Group 2 Using 3-Wire Transceiver



Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

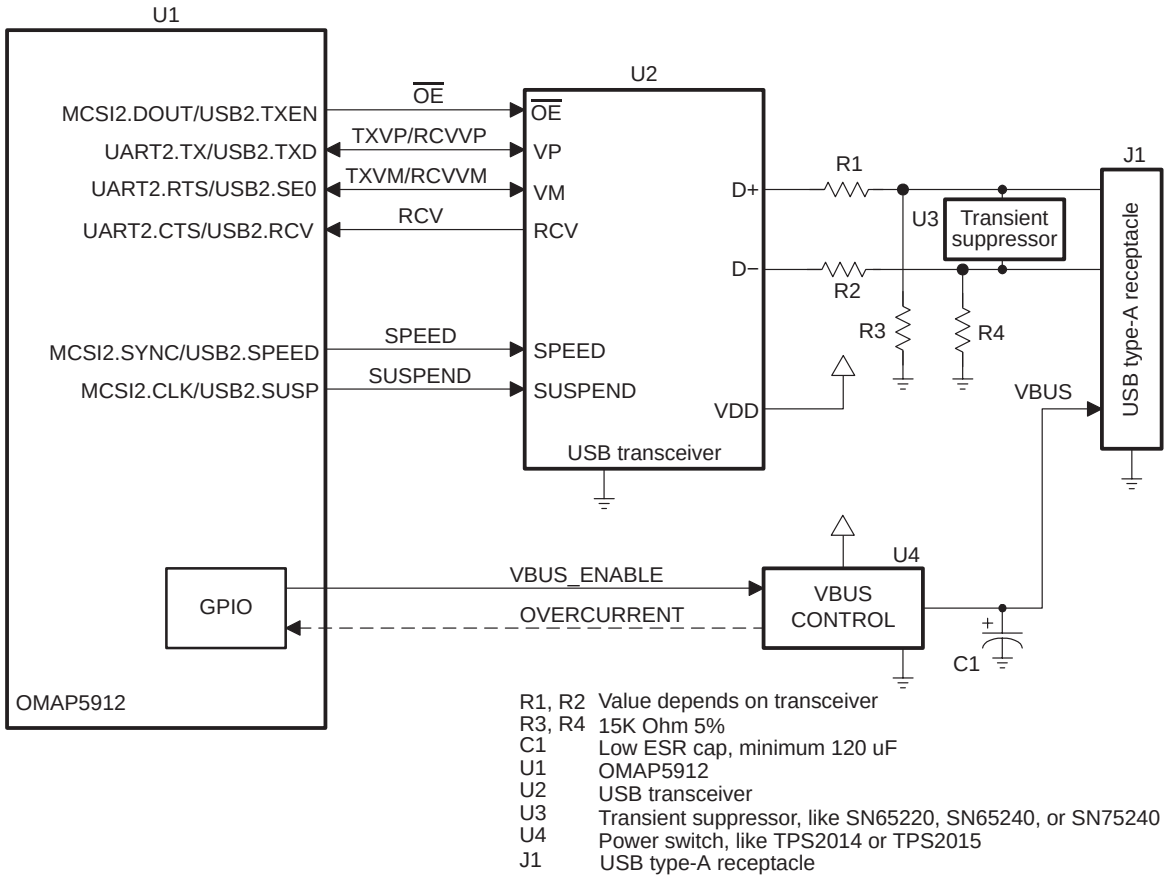
- ❏ MCSI2.DOUT/USB2.TXEN
- ❏ UART2.TX/USB2.TXD
- ❏ UART2.RTS/USB2.SE0
- ❏ MCSI2.SYNC/USB2.SPEED
- ❏ MCSI2.CLK/USB2.SUSPEND

See Table 74.

HMC_MODE must be set to a value that routes a host controller port to USB pin group 2 (see Table 73). OTG_SYSCON_1.USB2_TRX_MODE must be set to 2 to allow proper bidirectional operation of the 3-wire USB transceiver. OTG_SYSCON_2.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins UART2.TX/USB2.TXD and UART2.RTS/USB2.SE0.

Pin Group 2 USB Host Using 4-Wire USB Transceiver

Figure 71. USB Host Connections on USB Pin Group 2 Using 4-Wire Transceiver



Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

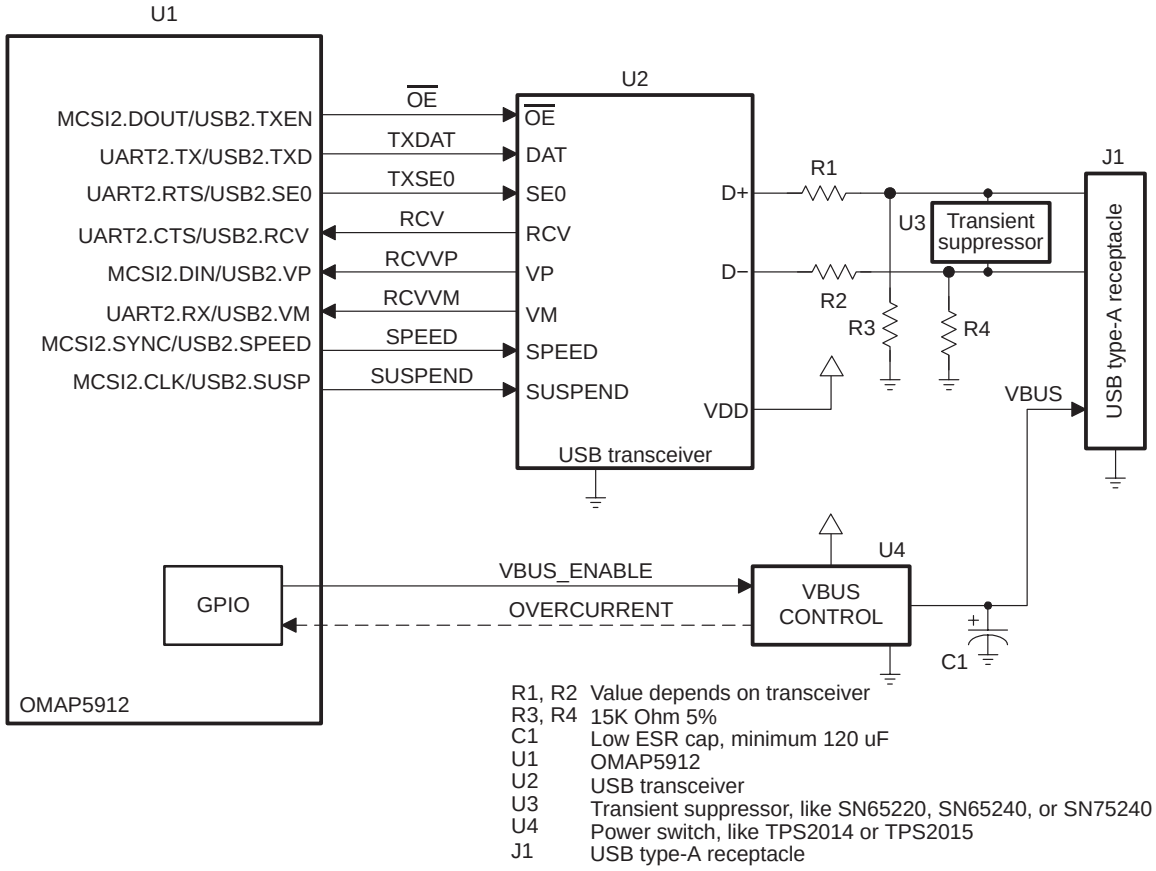
- ❏ MCSI2.DOUT/USB2.TXEN
- ❏ UART2.TX/USB2.TXD
- ❏ UART2.RTS/USB2.SE0
- ❏ UART2.CTS/USB2.RCV
- ❏ MCSI2.SYNC/USB2.SPEED
- ❏ MCSI2.CLK/USB2.SUSP

See Table 74.

HMC_MODE must be set to a value that routes a host controller port to USB pin group 2 (see Table 73). OTG_SYSCON_1.USB2_TRX_MODE must be set to 1 to allow proper bidirectional operation of the 4-wire USB transceiver. OTG_SYSCON_2.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins UART2.TX/USB2.TXD and UART2.RTS/USB2.SE0.

Pin Group 2 USB Host Using 6-Wire USB Transceiver

Figure 72. USB Host Connections on USB Pin Group 2 Using 6-Wire Transceiver



Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- MCSI2.DOUT/USB2.TXEN
- UART2.TX/USB2.TXD
- UART2.RTS/USB2.SE0
- UART2.CTS/USB2.RCV
- MCSI2.DIN/USB2.VP
- UART2.RX/USB2.VM
- MCSI2.SYNC/USB2.SPEED
- MCSI2.CLK/USB2SUSP

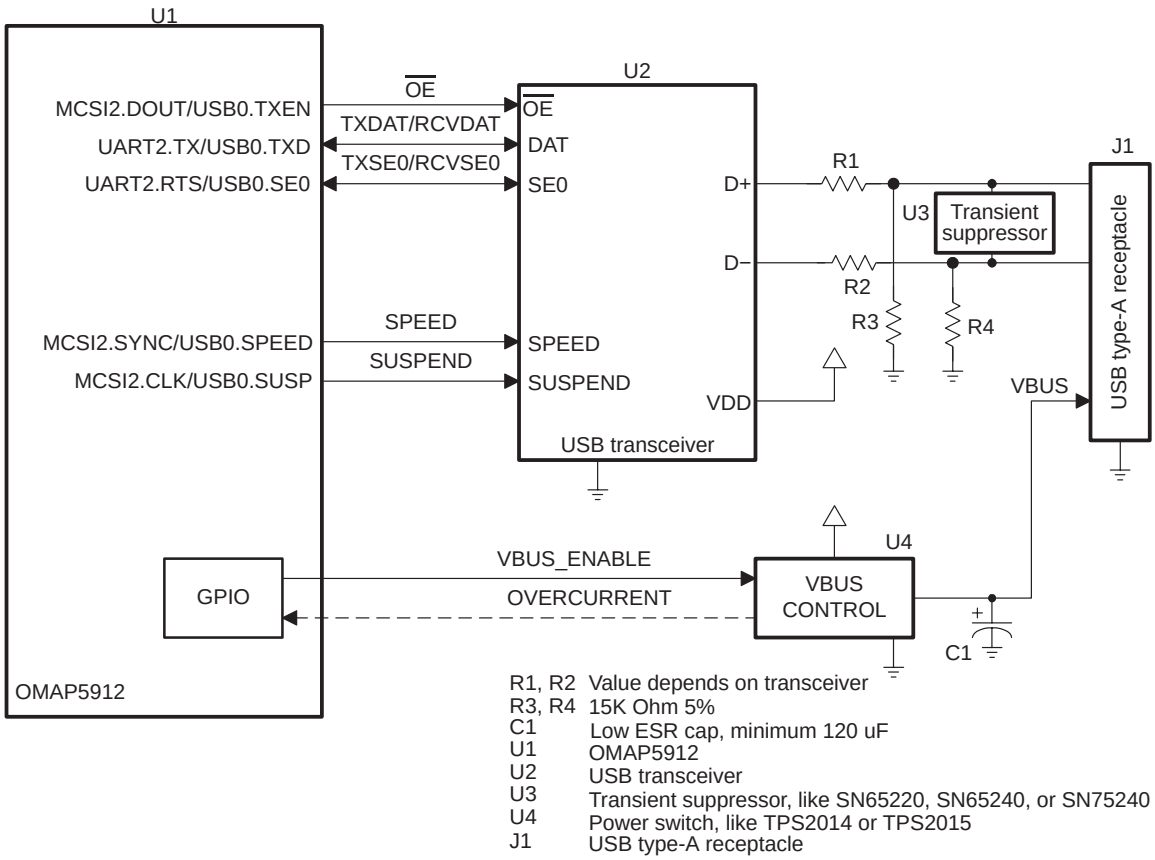
See Table 74.

HMC_MODE must be set to a value that routes a host controller port to USB pin group 2 (see Table 73). OTG_SYSCON_1.USB2_TRX_MODE must be set to 3 to allow proper unidirectional operation of the 6-wire USB transceiver.

USB Host on USB Alternate Pin Group 2, 3-Wire USB Transceiver

Table 73 shows that pin group 2 can receive either of two sets of USB-related signaling. In normal mode, pin group 2 receives the signaling associated with the controller shown under the Port 2 column. In alternate mode, pin group 2 receives signaling associated with the controller shown under port 0. Figure 73 shows USB host port connectivity when pin group 2 is configured for alternate mode and connected using a 3-wire USB transceiver. When pin group 2 is configured in alternate mode, pins USB.DP and USB.DM are not usable for USB functionality.

Figure 73. USB Host Connections on USB Alternate Pin Group 2 Using 3-Wire Transceiver



Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- MCSI2.DOUT/USB0.TXEN
- UART2.TX/USB0.TXD
- UART2.RTS/USB0.SE0
- MCSI2.SYNC/USB0.SPEED
- MCSI2.CLK/USB0.SUSP

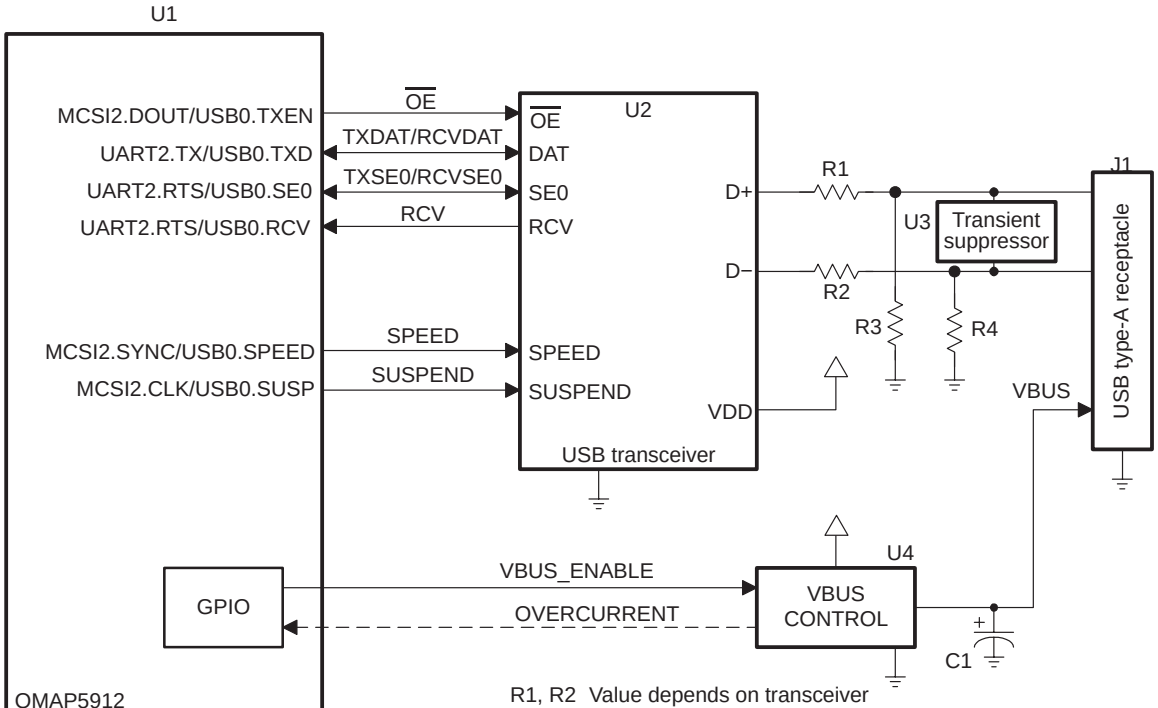
See Table 74.

HMC_MODE must be set to a value that routes a host controller port to USB pin group 0 (see Table 73). OTG_SYSCON_1.USB0_TRX_MODE must be set to 2 to allow proper bidirectional operation of the 3-wire USB transceiver. OTG_SYSCON_2.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins UART2.TX/USB0.TXD and UART2.RTS/USB0.SE0.

USB Host on USB Alternate Pin Group 2, 4-Wire USB Transceiver

Table 73 shows that pin group 2 can receive either of two sets of USB-related signaling. In normal mode, pin group 2 receives the signaling associated with the controller shown under the Port 2 column. In alternate mode, pin group 2 receives signaling associated with the controller shown under port 0. Figure 74 shows USB host port connectivity when pin group 2 is configured for alternate mode and connected using a 4-wire USB transceiver. When pin group 2 is configured in alternate mode, pins USB.DP and USB.DM are not usable for USB functionality.

Figure 74. USB Host Connections on USB Alternate Pin Group 2 Using 4-Wire Transceiver



Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for pins:

- MCSI2.DOUT/USB0.TXEN
- UART2.TX/USB0.TXD
- UART2.RTS/USB0.SE0
- UART2.CTS/USB0.RCV
- MCSI2.SYNC/USB0.SPEED
- MCSI2.CLK/USB0.SUSP

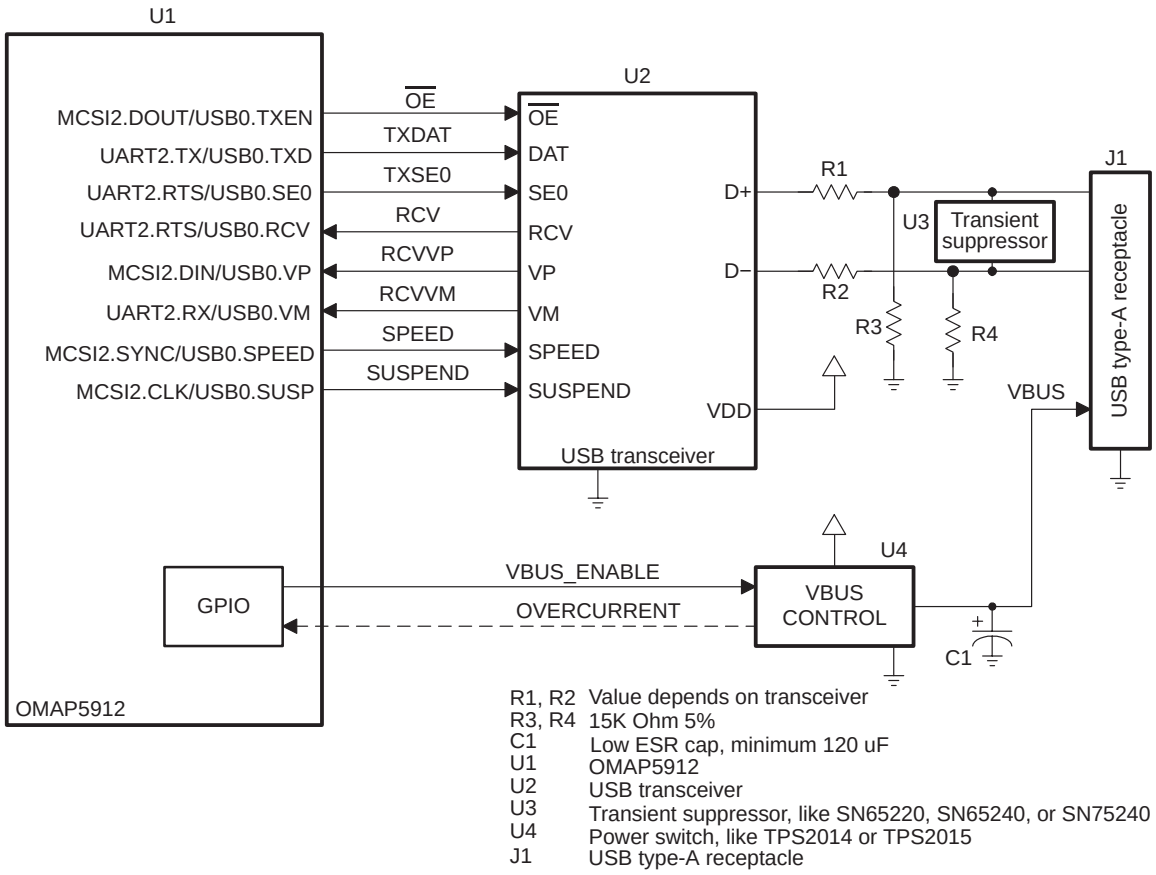
See Table 74.

HMC_MODE must be set to a value that routes a host controller port to USB alternate pin group 2 (see Table 73). OTG_SYSCON_1.USB0_TRX_MODE must be set to 1 to allow proper bidirectional operation of the 4-wire USB transceiver. OTG_SYSCON_0.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins UART2.TX/USB0.TXD and UART2.RTS/USB0.SE0.

USB Host on USB Alternate Pin Group 2, 6-Wire USB Transceiver

Table 73 shows that pin group 2 can receive either of two sets of USB-related signaling. In normal mode, pin group 2 receives the signaling associated with the controller shown under the Port 2 column. In alternate mode, pin group 2 receives signaling associated with the controller shown under port 0. Figure 75 shows USB host port connectivity when pin group 2 is configured for alternate mode and connected using a 6-wire USB transceiver. When pin group 2 is configured in alternate mode, pins USB.DP and USB.DM are not usable for USB functionality.

Figure 75. USB Host Connections on USB Alternate Pin Group 2 Using 6-Wire Transceiver



Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for pins:

- MCSI2.DOUT/USB0.TXEN
- UART2.TX/USB0.TXD
- UART2.RTS/USB0.SE0
- UART2.CTS/USB0.RCV
- MCSI2.DIN/USB0.VP
- UART2.RX/USB0.VM
- MCSI2.SYNC/USB0.SPEED
- MCSI2.CLK/USB0.SUSP

See Table 74.

HMC_MODE must be set to a value that routes a host controller port to USB alternate pin group 2 (see Table 73). OTG_SYSCON_1.USB0_TRX_MODE must be set to 3 to allow proper unidirectional operation of the 6-wire USB transceiver.

4.8 USB Function 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. These features include a USB type-B or mini-B receptacle, VBUS power detection, transient suppression, a controllable pullup resistor to the D+ or D- line, and USB–compatible upstream port transceiver. These elements are shown in Figure 76.

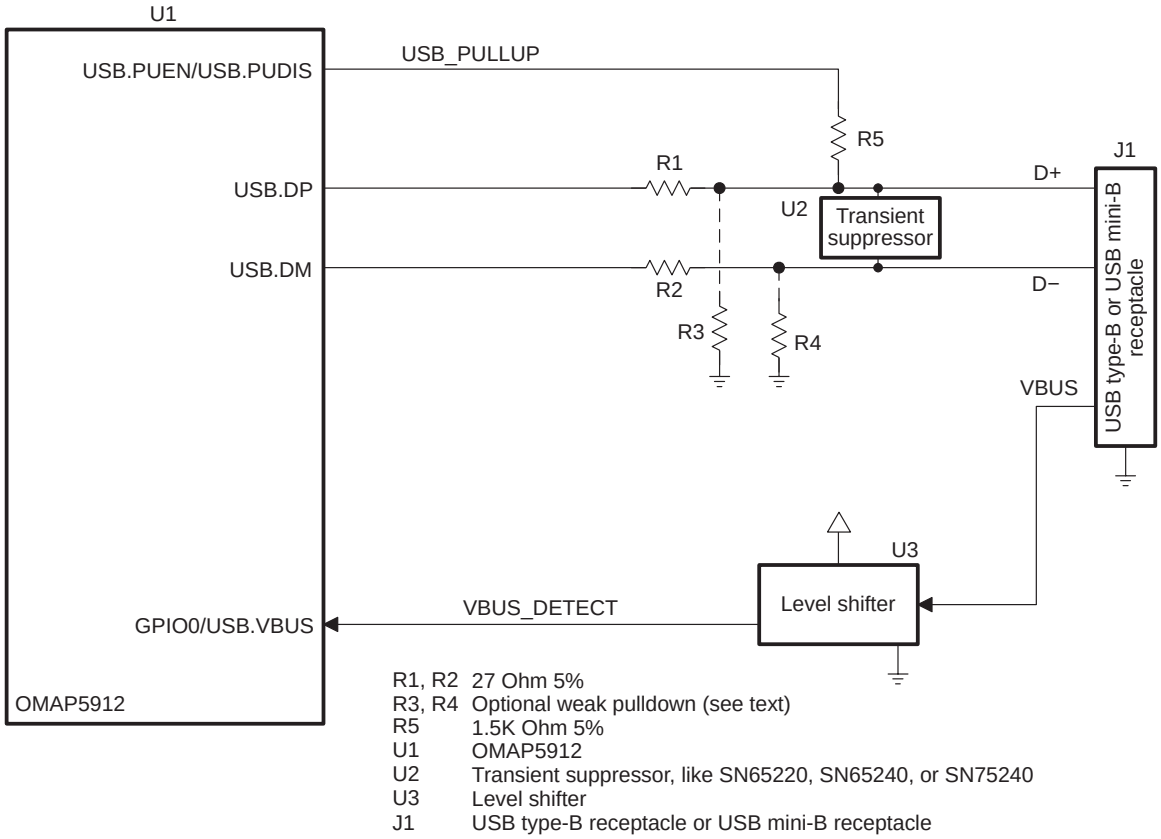
Optional weak pulldown resistors can be used to hold the D+ and D– signals at voltages below the USB transceiver VIL level when there is no USB host connected to the USB Type B connector. By keeping D+ and D– voltages below VIL, the USB transceiver IDDQ can be reduced. Choice of value for the pulldown resistors must be made carefully so that the circuit meets the requirements of the *USB Specification*.

The OMAP5912 USB function 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-wire connection to the host controller, whereas others require only 4 wires or 3 wires. Figure 76 through Figure 82 show the signal connectivity options for the integrated transceiver and external 3-, 4-, and 6-wire USB transceivers on each of the available OMAP5912 USB pin groups.

Pin Group 0 USB Device

Figure 76. USB Device Connections Using OMAP5912 Integrated USB Transceiver

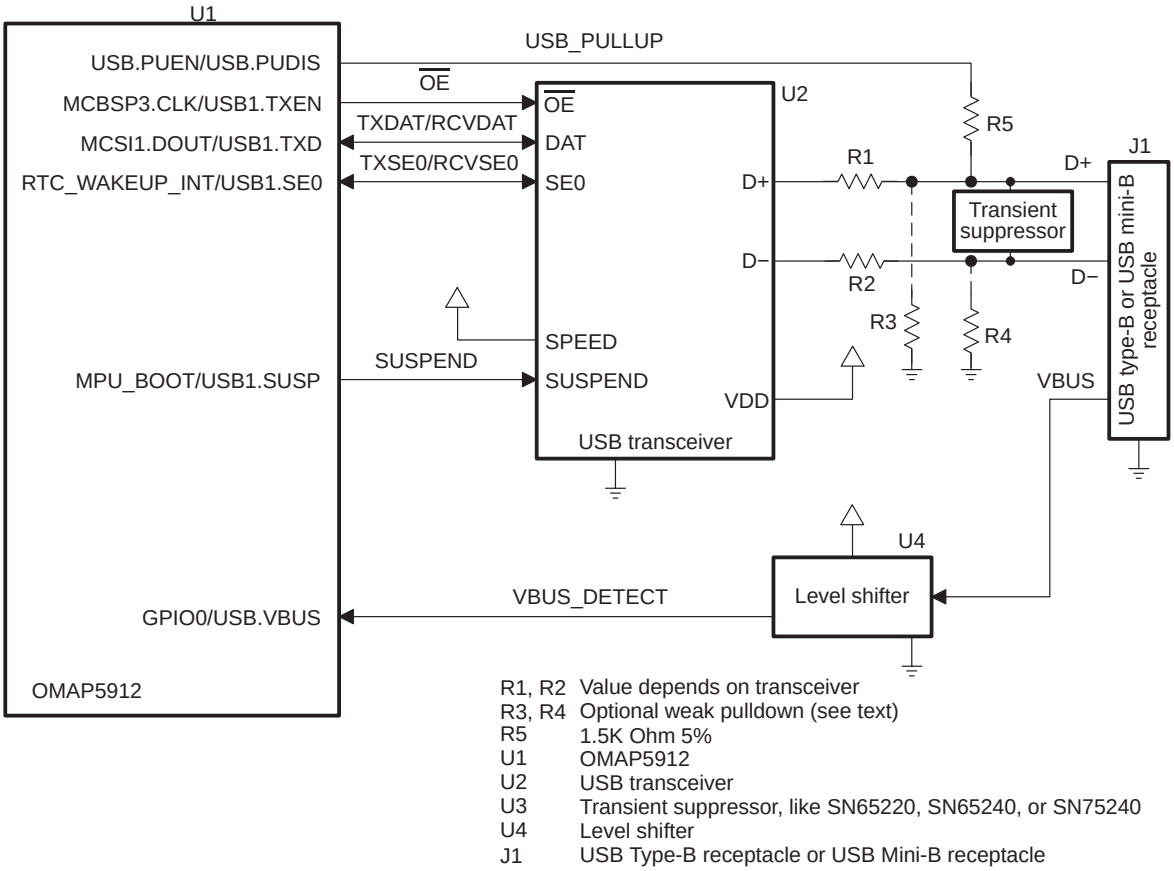


Proper initialization of the OMAP5912 to support this mode of operation requires proper setting of the top-level multiplexing for USB.DP and USB.DM (see Table 74) and selection of an HMC_MODE value that routes the USB device controller to pin group 0 (see Table 73). OTG_SYSCON_1. USB0.TRX_MODE must be set to 3 to allow proper operation of the integrated USB transceiver.

When OTG_SYSCON_2.OTG_PADEN is 1, the status of GPIO0/USB.VBUS is automatically provided to the USB device controller via hardware mechanisms. When OTG_SYSCON_2.OTG_PADEN is 0, the USB device controller only sees the VBUS status from the software-controlled register OTG_CTRL.BSESSVLD bit. In this case, software must monitor the VBUS_DETECT signal (using GPIO0 if wired as shown), and update OTG_CTRL.BSESSVLD whenever the VBUS_DETECT signal changes.

Pin Group 1 USB Device Using 3-Wire USB Transceiver

Figure 77. USB Device Connections on USB Pin Group 1 Using 3-Wire Transceiver



Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- MCBSP3.CLKX/USB1.TXEN
- MCSI1.DOUT/USB1.TXD
- RTC_WAKE_INT/USB1.SE0
- CLK32K_OUT/USB1.SPEED
- MPU_BOOT/USB1.SUSP

See Table 74.

HMC_MODE must be set to a value that routes the OMAP5912 USB device controller to USB pin group 2 (see Table 73). OTG_SYSCON_1.USB1_TRX_MODE must be set to 2 to allow proper bidirectional operation of the 3-wire USB transceiver.

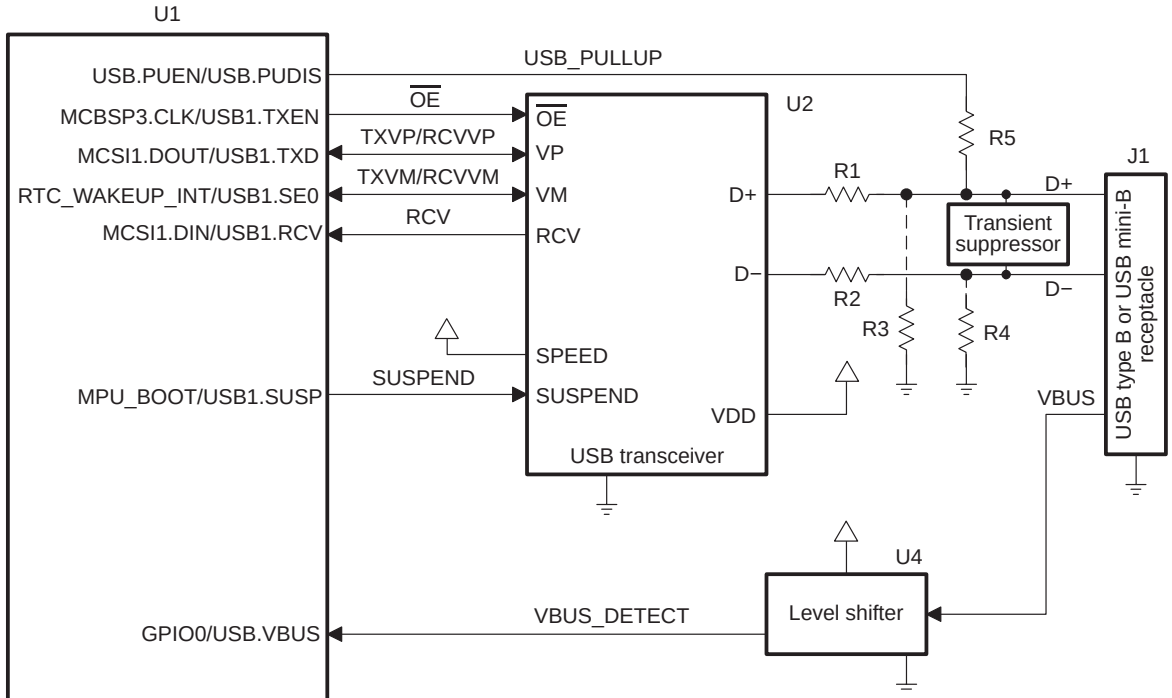
OTG_SYSCON_2.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins MCSI1.DOUT/USB1.TXD and RTC_WAKE_INT/USB1.SE0.

A USB OTG transceiver can be used to provide USB device-only functionality instead of a USB transceiver, with appropriate I^2C and interrupt connectivity. In this case, OTG_SYSCON_2.OTG_PADEN 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^2C link and updating the value sent to the device controller via OTG_CTRL.BSESSVLD. 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, those hardware connections shown can be removed when an OTG transceiver is used. System software must then control those features via the OTG transceiver I^2C register set.

When OTG_SYSCON_2.OTG_PADEN is 1, the status of GPIO0/USB.VBUS is automatically provided to the USB device controller via hardware mechanisms (when HMC_MODE provides USB device functionality to USB pin group). When OTG_SYSCON_2.OTG_PADEN is 0, the USB device controller only ses the VBUS status from the software-controlled register OTG_CTRL.BSESSVLD bit. In this case, software must monitor the VBUS_DETECT signal (using GPIO0 if wired as shown), and update OTG_CTRL.BSESSVLD whenever the VBUS_DETECT signal changes.

Pin Group 1 USB Device Using 4-Wire USB Transceiver

Figure 78. USB Device Connections on USB Pin Group 1 Using 4-Wire Transceiver



R1, R2 Value depends on transceiver
R3, R4 Optional weak pulldown (see text)
R5 1.5K Ohm 5%
U1 OMAP5912
U2 USB transceiver
U3 Transient suppressor, like SN65220, SN65240, or SN75240
U4 Level shifter
J1 USB Type-B receptacle or USB Mini-B receptacle

Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- MCBSP3.CLKX/USB1.TXEN
- MCSI1.DOUT/USB1.TXD
- RTC_WAKE_INT/USB1.SE0
- CLK32K_OUT/USB1.SPEED
- MPU_BOOT/USB1.SUSP

See Table 74.

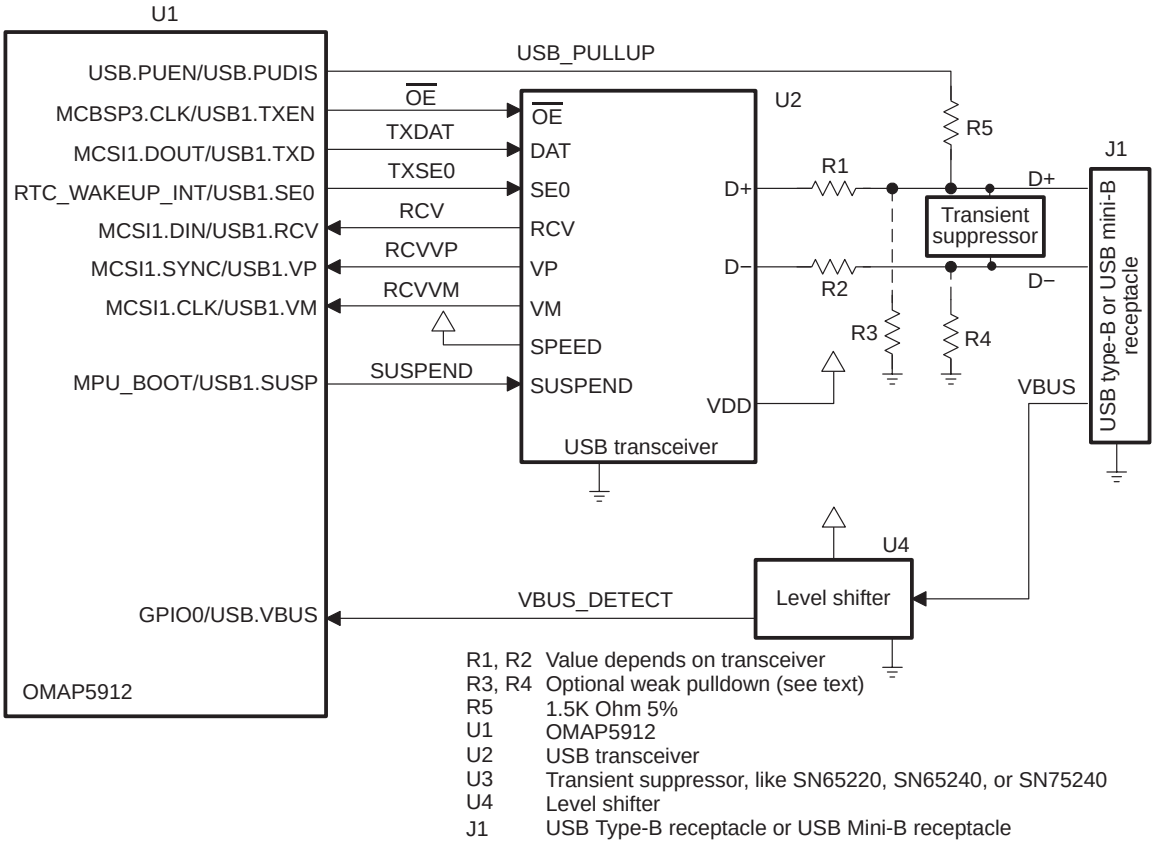
HMC_MODE must be set to a value that routes the USB device controller to USB pin group 1 (see Table 73). OTG_SYSCON_1.USB1_TRX_MODE must be set to 1 to allow proper bidirectional operation of the 4-wire USB transceiver. OTG_SYSCON_2.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins MCSI1.DOUT/USB1.TXD and RTC_WAKE_INT/USB1.SE0.

A USB OTG transceiver can be used to provide USB device-only functionality instead of a USB transceiver, with appropriate I^2C and interrupt connectivity. In this case, OTG_SYSCON_2.OTG_PADEN 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^2C link and updating the value sent to the device controller via OTG_CTRL.BSESSVLD. In this case, 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, those hardware features shown can be removed when an OTG transceiver is used. System software needs to control those features via the OTG transceiver I^2C register set.

When OTG_SYSCON_2.OTG_PADEN is 1, the status of GPIO0/USB.VBUS is automatically provided to the USB device controller via hardware mechanisms (when HMC_MODE provides USB device functionality to a USB pin group). When OTG_SYSCON_2.OTG_PADEN is 0, the USB device controller only sees the VBUS status from the software-controlled register OTG_CTRL.BSESSVLD bit. In this case, software must monitor the VBUS_DETECT signal (using GPIO0 if wired as shown), and update OTG_CTRL.BSESSVLD whenever the VBUS_DETECT signal changes.

Pin Group 1 USB Device Using 6-Wire USB Transceiver

Figure 79. USB Device Connections on USB Pin Group 1 Using 6-Wire Transceiver



Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- ❏ MCBSP3.CLKX/USB1.TXEN
- ❏ MCSI1.DOUT/USB1.TXD
- ❏ RTC_WAKE_INT/USB1.SE0
- ❏ MCSI1.SYNC/USB1.VP
- ❏ MCSI1.BCLK/USB1.VM
- ❏ MCSI1.DIN/USB1.RCV
- ❏ CLK32K_OUT/USB1.SPEED
- ❏ MPU_BOOT/USB1.SUSP

See Table 74.

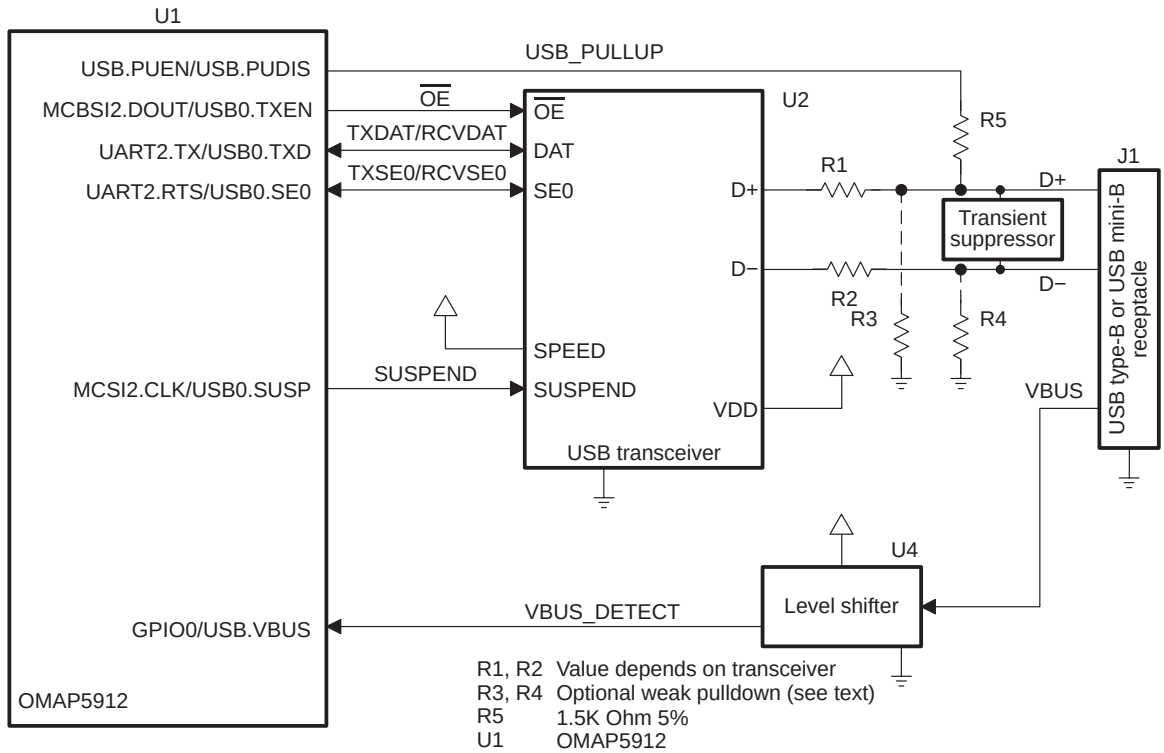
HMC_MODE must be set to a value that routes the USB device controller to USB pin group 1 (see Table 73). OTG_SYSCON_1.USB1_TRX_MODE must be set to 3 to allow proper unidirectional operation of the 6-wire USB transceiver.

A USB OTG transceiver can be used to provide USB device-only functionality instead of a USB transceiver, with appropriate I^2C and interrupt connectivity. In this case, OTG_SYSCON_2.OTG_PADEN 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^2C link and updating the value sent to the device controller via OTG_CTRL.BSESSVLD. In this case, 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 via the OTG transceiver I^2C register set.

When OTG_SYSCON_2.OTG_PADEN is 1, the status of GPIO0/USB.VBUS is automatically provided to the USB device controller via hardware mechanisms (when HMC_MODE provides USB device functionality to a USB pin group). When OTG_SYSCON_2.OTG_PADEN is 0, the USB device controller only sees the VBUS status from the software-controlled register OTG_CTRL.BSESSVLD bit. In this case, software must monitor the VBUS_DETECT signal (using GPIO0 if wired as shown), and update OTG_CTRL.BSESSVLD whenever the VBUS_DETECT signal changes.

USB Device on USB Alternate Pin Group 2, 3-Wire USB Transceiver

Figure 80. USB Device Connections on USB Alternate Pin Group 2 Using 3-Wire Transceiver



R1, R2	Value depends on transceiver
R3, R4	Optional weak pulldown (see text)
R5	1.5K Ohm 5%
U1	OMAP5912
U2	USB transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
U4	Level shifter
J1	USB Type-B receptacle or USB Mini-B receptacle

Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- MCSI2.DOUT/USB0.TXEN
- UART2.TX/USB0.TXD
- UART2.RTS/USB0.SE0
- MCSI2.SYNC/USB0.SPEED
- MCSI2.CLK/USB0.SUSP

See Table 74.

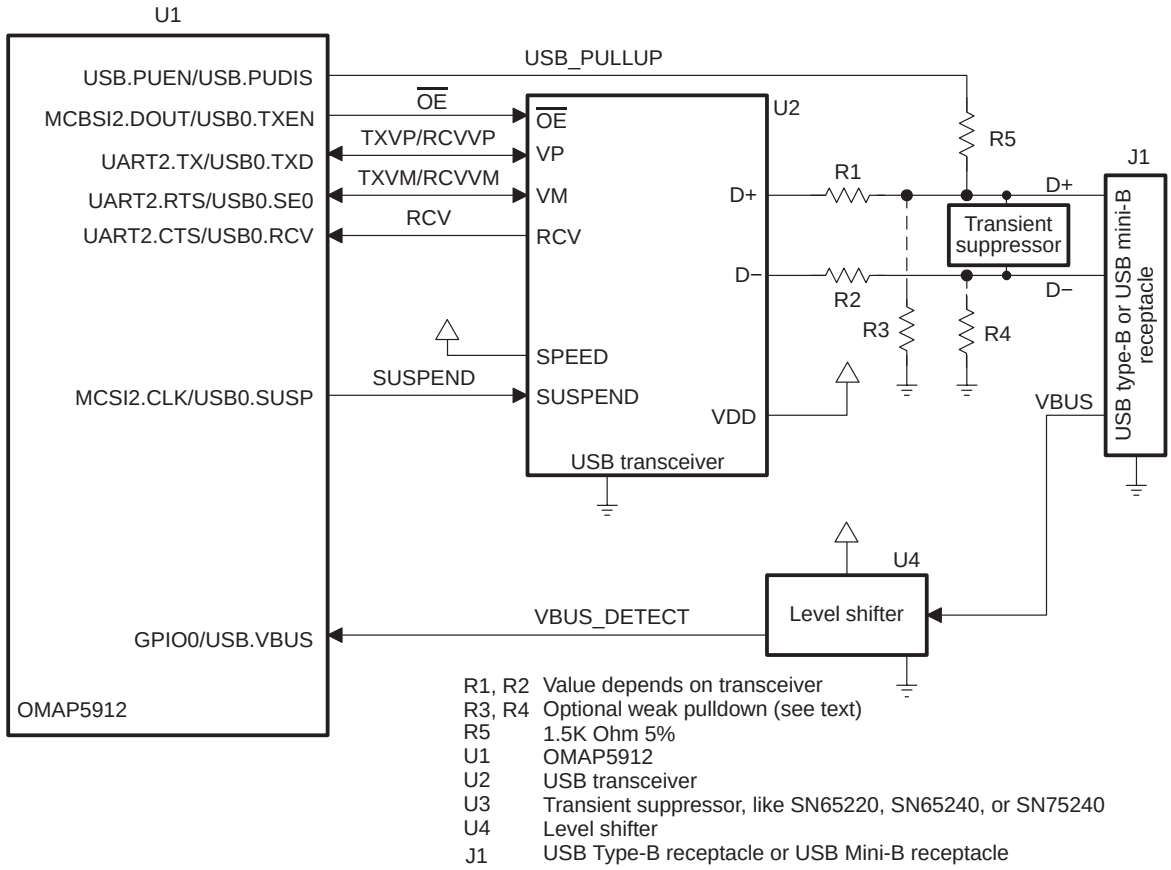
HMC_MODE must be set to a value that routes the USB device controller to USB alternate pin group 2 (see Table 73). OTG_SYSCON_1.USB0_TRX_MODE must be set to 2 to allow proper bidirectional operation of the 3-wire USB transceiver. OTG_SYSCON_2.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins UART2.TX/USB0.TXD and UART2.RTS/USB0.SE0.

A USB OTG transceiver can be used to provide USB device-only functionality instead of a USB transceiver, with appropriate I^2C and interrupt connectivity. In this case, OTG_SYSCON_2.OTG_PADEN 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^2C link and updating the value sent to the device controller via OTG_CTRL.BSESSVLD. 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, those hardware features shown can be removed when an OTG transceiver is used. System software must control those features via the OTG transceiver I^2C register set.

When OTG_SYSCON_2.OTG_PADEN is 1, the status of GPIO0/USB.VBUS is automatically provided to the USB device controller via hardware mechanisms (when HMC_MODE provides USB device functionality to a USB pin group). When OTG_SYSCON_2.OTG_PADEN is 0, the USB device controller only sees the VBUS status from the software-controlled register OTG_CTRL.BSESSVLD bit. In this case, software must monitor the VBUS_DETECT signal (using GPIO0 if wired as shown), and update OTG_CTRL.BSESSVLD whenever the VBUS_DETECT signal changes.

USB Device on USB Alternate Pin Group 2, 4-Wire USB Transceiver

Figure 81. USB Device Connections on USB Alternate Pin Group 2 Using 4-Wire OTG Transceiver



Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- ❏ MCSI2.DOUT/USB0.TXEN
- ❏ UART2.TX/USB0.TXD
- ❏ UART2.RTS/USB0.SE0
- ❏ UART2.CTS/USB0.RCV
- ❏ MCSI2.SYNC/USB0.SPEED
- ❏ MCSI2.CLK/USB0.SUSP

See Table 74.

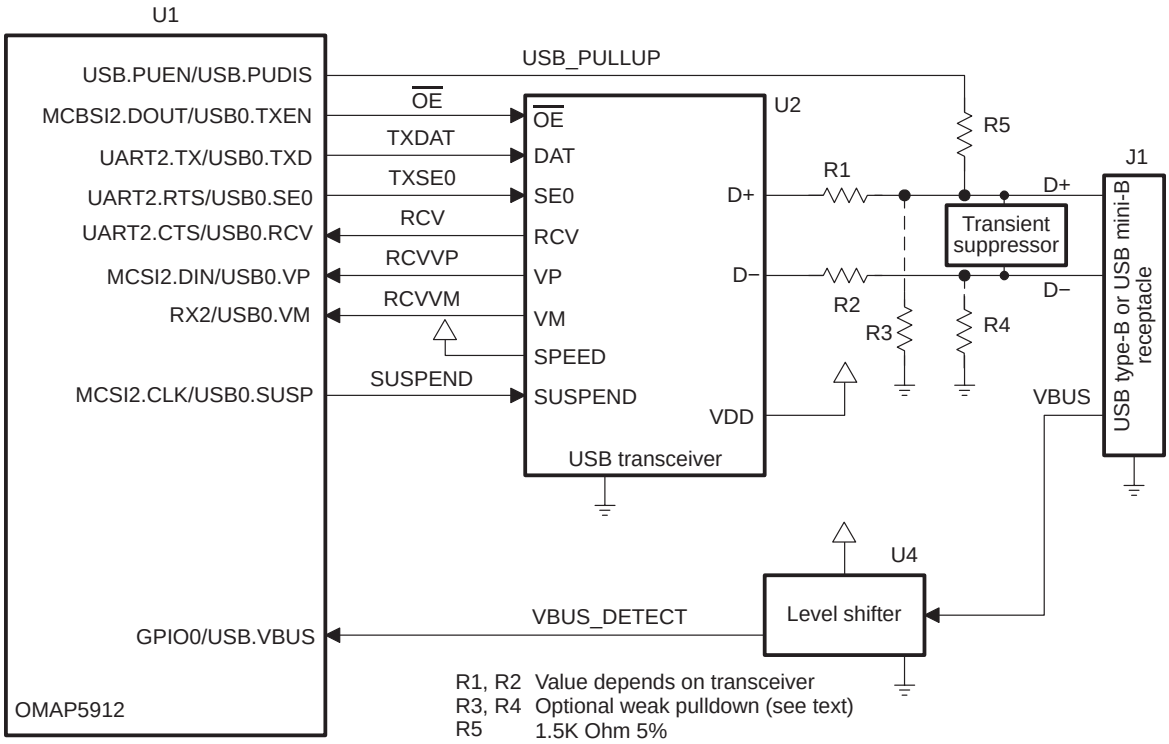
HMC_MODE must be set to a value that routes the USB device controller to USB pin group 0 (see Table 73). OTG_SYSCON_1.USB0_TRX_MODE must be set to 1 to allow proper bidirectional operation of the 4-wire USB transceiver. OTG_SYSCON_0.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins UART2.TX/USB0.TXD and UART2.RTS/USB0.SE0.

A USB OTG transceiver can be used to provide USB device-only functionality instead of a USB transceiver, with appropriate I^2C and interrupt connectivity.In this case, OTG_SYSCON_2.OTG_PADEN 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^2C link and updating the value sent to the device controller via OTG_CTRL.BSESSVLD. 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, those hardware features shown can be removed when an OTG transceiver is used. System software must control those features via the OTG transceiver I^2C register set.

When OTG_SYSCON_2.OTG_PADEN is 1, the status of GPIO0/USB.VBUS is automatically provided to the USB device controller via hardware mechanisms (when HMC_MODE provides USB device functionality to a USB pin group). When OTG_SYSCON_2.OTG_PADEN is 0, the USB device controller only sees the VBUS status from the software-controlled register OTG_CTRL.BSESSVLD bit. In this case, software must monitor the VBUS_DETECT signal (using GPIO0 if wired as shown), and update OTG_CTRL.BSESSVLD whenever the VBUS_DETECT signal changes.

USB Device on USB Alternate Pin Group 2, 6-Wire USB Transceiver

Figure 82. USB Device Connections on USB Alternate Pin Group 2 Using 6-Wire Transceiver



R1, R2	Value depends on transceiver
R3, R4	Optional weak pulldown (see text)
R5	1.5K Ohm 5%
U1	OMAP5912
U2	USB transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
U4	Level shifter
J1	USB Type-B receptacle or USB Mini-B receptacle

Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for pins:

- MCSI2.DOUT/USB0.TXEN
- UART2.TX/USB0.TXD
- UART2.RTS/USB0.SE0
- UART2.CTS/USB0.RCV
- MCSI2.DIN/USB0.VP
- UART2.RX/USB0.VM
- MCSI2.SYNC/USB0.SPEED
- MCSI2.CLK/USB0.SUSP

See Table 74.

HMC_MODE must be set to a value that routes the USB device controller to USB alternate pin group 2 (see Table 73). OTG_SYSCON_1.USB0_TRX_MODE must be set to 3 to allow proper unidirectional operation of the 6-wire USB transceiver.

A USB OTG transceiver can be used to provide USB device-only functionality instead of a USB transceiver, with appropriate I^2C and interrupt connectivity. In this case, OTG_SYSCON_2.OTG_PADEN 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^2C link and updating the value sent to the device controller via OTG_CTRL.BSESSVLD. In this case, 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, those hardware features shown can be removed when an OTG transceiver is used. System software must
control those features via the OTG transceiver I^2C register set.

When OTG_SYSCON_2.OTG_PADEN is 1, the status of GPIO0/USB.VBUS is automatically provided to the USB device controller via hardware mechanisms (when HMC_MODE provides USB device functionality to a USB pin group). When OTG_SYSCON_2.OTG_PADEN is 0, the USB device controller only sees the VBUS status from the software-controlled register OTG_CTRL.BSESSVLD bit. In this case, software must monitor the VBUS_DETECT signal (using GPIO0 if wired as shown) and update OTG_CTRL.BSESSVLD whenever the VBUS_DETECT signal changes.

4.9 Onboard Transceiverless Connection Using OMAP5912 Transceiverless Link

The transceiverless link logic feature of the OMAP5912 USB signal multiplexing enables connection of the OMAP5912 USB device controller to an external, onboard USB host controller or connection of the OMAP5912 USB host controller, without the use of USB transceivers or associated circuitry. When the transceiverless link logic is used, both of the 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 OMAP5912 transceiverless link 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 transceiverless link logic signaling system is not suitable for use across a cable. It is intended only for use when the OMAP5912 device is used with an external USB integrated circuit that is on the same board.

When using the transceiverless link logic, six of the external USB integrated circuit pins that typically connect to a USB transceiver connect instead directly to OMAP5912 device pins. Signaling on these pins use CMOS levels. Transceiverless link logic can be used only with external devices that support 6-wire transceiver connectivity.

Figure 83 and Figure 84 show how transceiverless link logic can be compared to a typical USB implementation. Figure 83 shows OMAP5912 being 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 OMAP5912 transceiverless link logic. Figure 84 shows OMAP5912 used as a USB device controller, with the top portion of the diagram showing a transceiver-based solution and the bottom portion showing a transceiverless solution using the OMAP5912 transceiverless link logic.

Figure 83. OMAP5912 USB Host Controller Connection—With and Without the OMAP5912 Transceiverless Link Logic

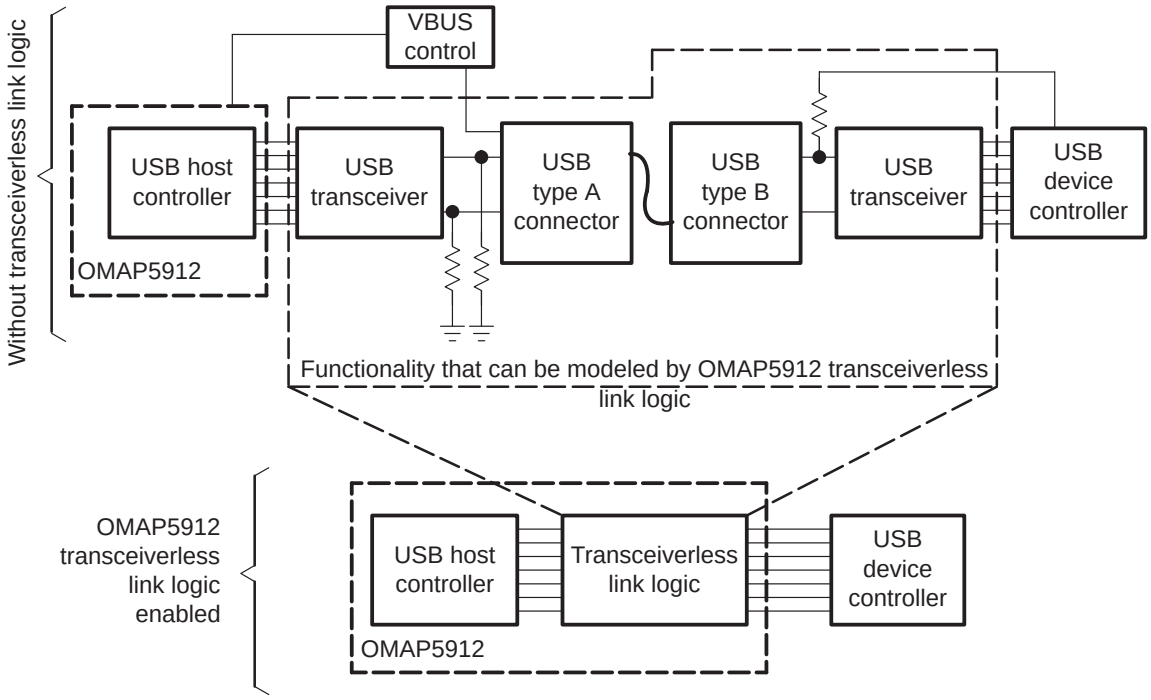
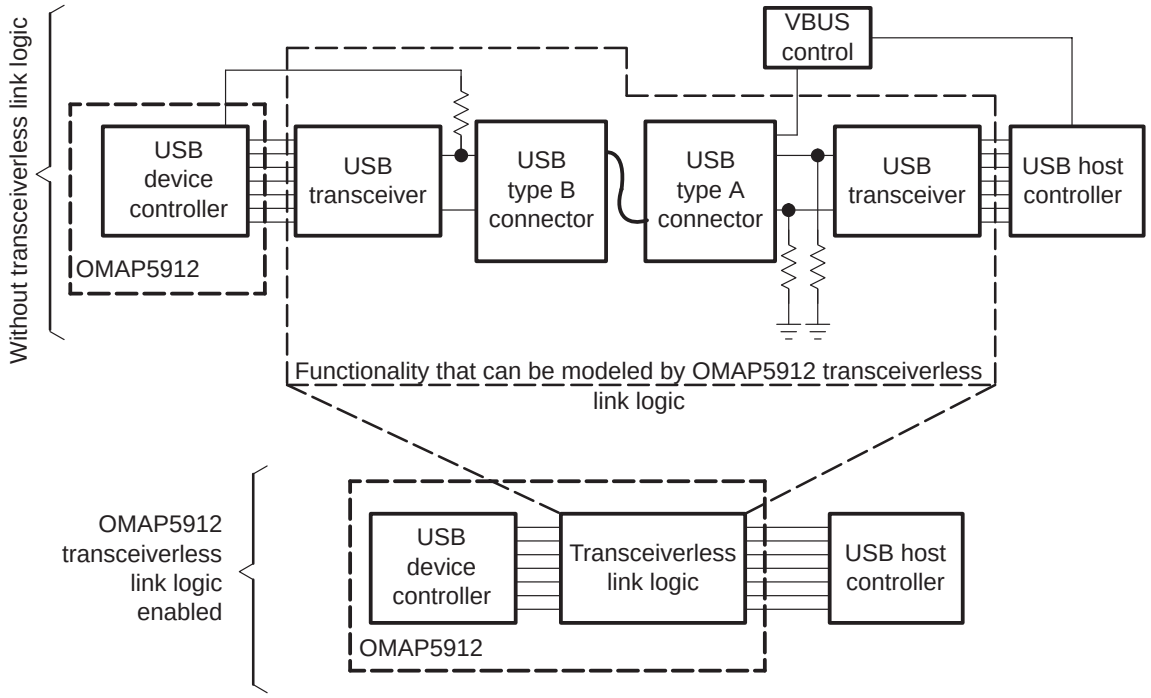


Figure 84. OMAP5912 USB Device Connection—With and Without the OMAP5912 Transceiverless Link Logic



The transceiverless link logic function in the OMAP5912 device interprets the transmit control signals from the external USB integrated circuit and similar signals from the OMAP5912 USB host controller or OMAP5912 USB device 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 OMAP5912 USB host controller or OMAP5912 USB device controller.

Two control bits help determine the proper differential pair state. TLL_ATTACH and TLL_SPEED provide these inputs and are controlled by OMAP5912 register bits (see Table 75). TLL_ATTACH is used to control when the differential pair models a pullup resistor as is implemented in a transceiver-based USB device. TLL_SPEED is used solely to determine whether the modeled pullup resistor is modeled on the internal version of D+ (for a full-speed transceiverless link) or D- (for a low-speed transceiverless link).

Caution

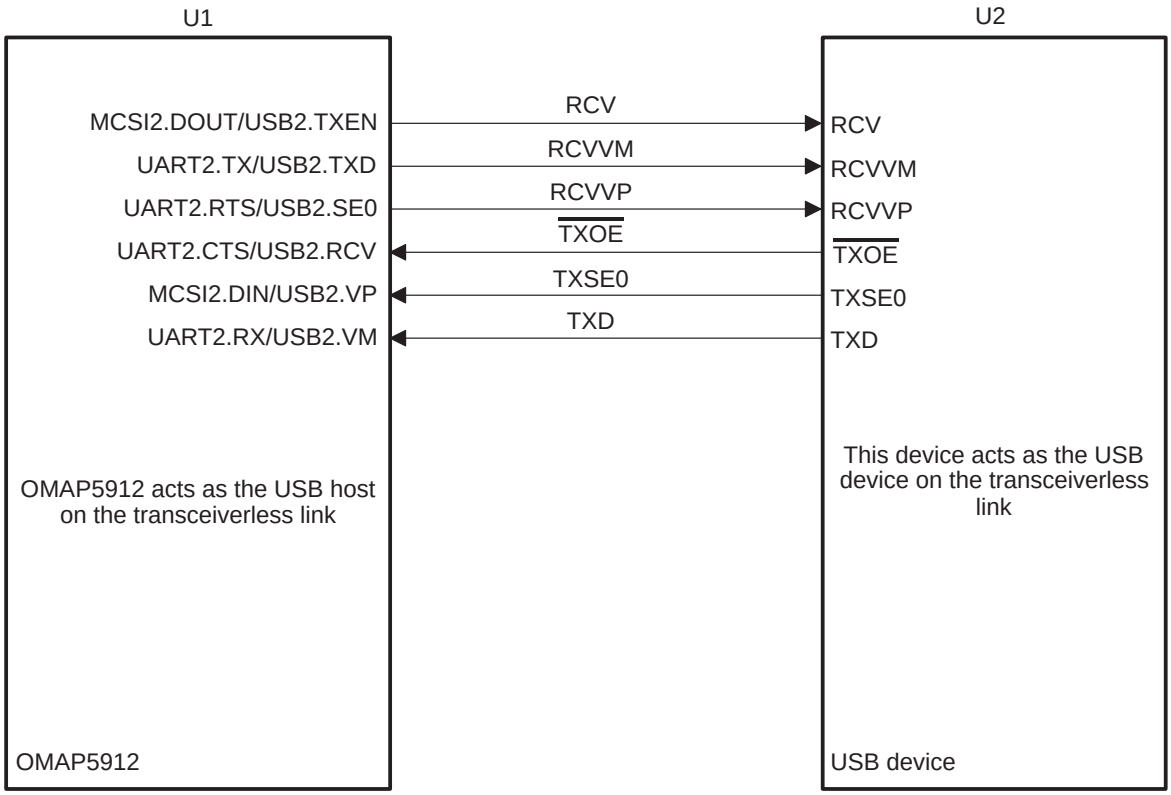
The OMAP5912 USB device cannot operate as a low-speed USB device. Do not set TLL_SPEED to 0 when HMC_MODE connects the OMAP5912 USB device controller to the transceiverless link logic.

The transceiverless link 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 transceiverless link logic connection to an on-board downstream USB device. OMAP5912 does not provide any modes where both a transceiverless link logic connection to an on-board USB host and a transceiver-based OTG connection are simultaneously
available.

USB Host With Transceiverless Link Logic (TXD/TXSE0)

Figure 85 shows connectivity when OMAP5912 acts as the USB host controller on a transceiverless link connection to an on-board USB device controller.

Figure 85. OMAP5912 as USB Host on Transceiverless Link Using TXD/TXSE0 Signaling



Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- MCSI2.DOUT/USB2.TXEN
- UART2.TX/USB2.TXD
- UART2.RTS/USB2.SE0
- UART2.CTS/USB2.RCV
- MCSI2.DIN/USB2.VP
- UART2.RX/USB2.VM

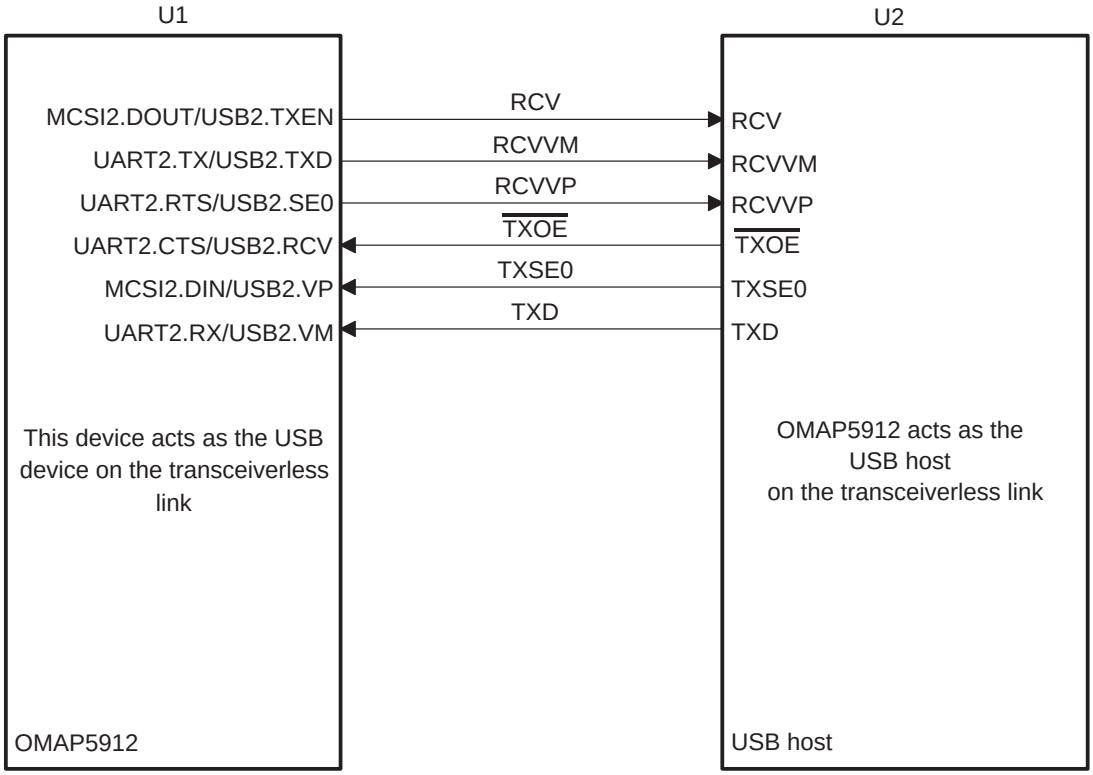
See Table 74.

HMC_MODE must be set to a value that routes a host controller port through the TLL (in TXD/TXSE0 mode) to USB pin group 2 (see Table 73). OTG_SYSCON_1.USB2_TRX_MODE must be set to 3 to allow proper operation of the transceiverless link logic connection.

USB Device With Transceiverless Link Logic (TXD/TXSE0)

Figure 86 shows connectivity when OMAP5912 acts as the USB device on a transceiverless link connection to an on-board USB host controller.

Figure 86. OMAP5912 as USB Device Controller on Transceiverless Link Using TXD/TXSE0 Signaling



Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- MCSI2.DOUT/USB2.TXEN
- UART2.TX/USB2.TXD
- UART2.RTS/USB2.SE0
- UART2.CTS/USB2.RCV
- MCSI2.DIN/USB2.VP
- UART2.RX/USB2.VM

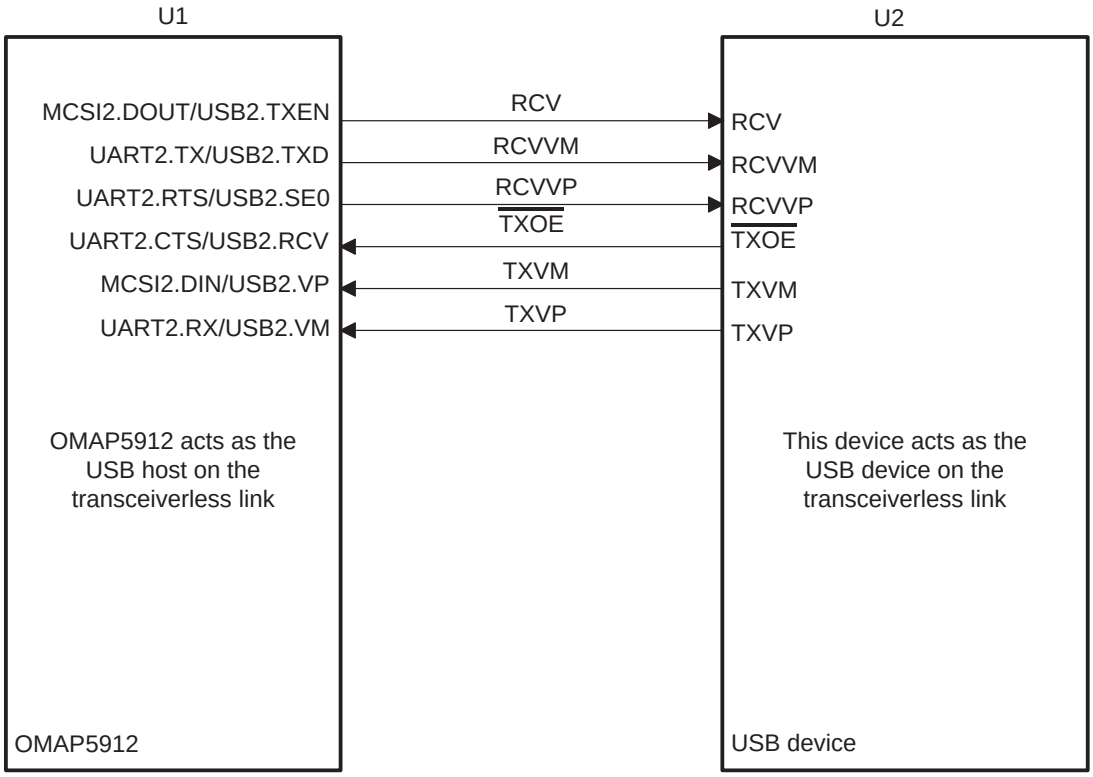
See Table 74.

HMC_MODE must be set to a value that routes the USB device controller through the TLL (in TXD/TXSE0 mode) to USB pin group 2 (see Table 73).

OTG_SYSCON_1.USB2_TRX_MODE must be set to 3 to allow proper operation of the transceiverless link logic connection. TLL_SPEED must be 1 when using the OMAP5912 USB device controller with transceiverless link logic.

USB Host With Transceiverless Link Logic (TXVP/TXVM)

Figure 87. OMAP5912 as USB Host Controller on Transceiverless Link Using TXVP/TXVM Signaling



Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- ❏ MCSI2.DOUT/USB2.TXEN
- ❏ UART2.TX/USB2.TXD
- ❏ UART2.RTS/USB2.SE0
- ❏ UART2.CTS/USB2.RCV
- ❏ MCSI2.DIN/USB2.VP
- ❏ UART2.RX/USB2.VM

See Table 74.

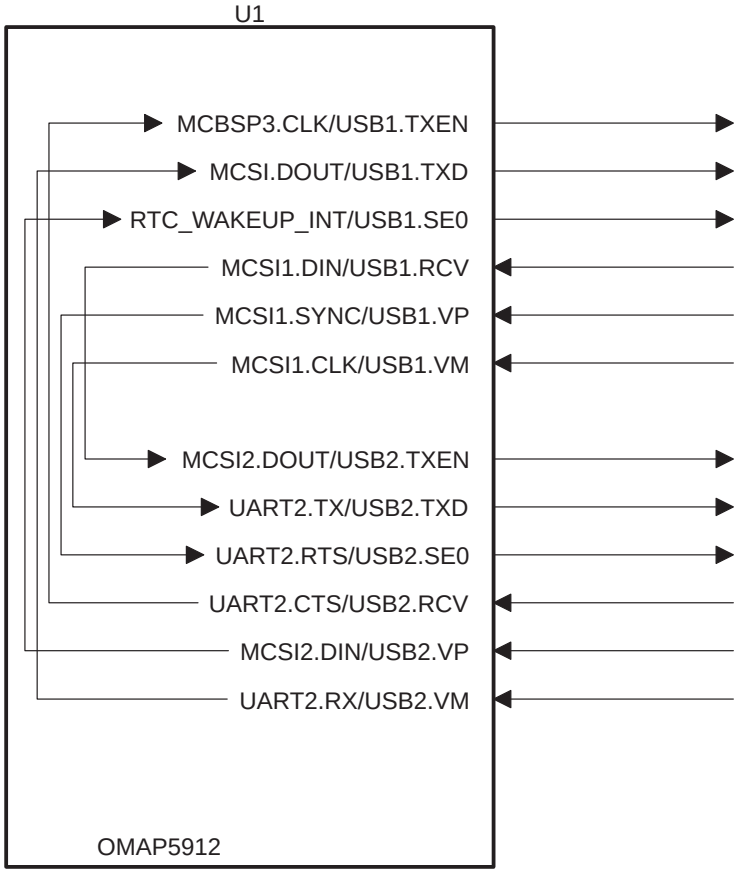
HMC_MODE must be set to a value that routes a host controller port through the TLL (in TXVP/TXVM mode) to USB pin group 2 (see Table 73). OTG_SYSCON_1.USB2_TRX_MODE must be set to 3 to allow proper operation of the transceiverless link logic connection.

4.10 Other Pin Connectivity Controlled by USB Signal Multiplexing

Pin Group 1 to Pin Group 2 Internal Loopback

Figure 88 shows how the OMAP5912 internally connects the pins when configured for HMC_MODE 7.

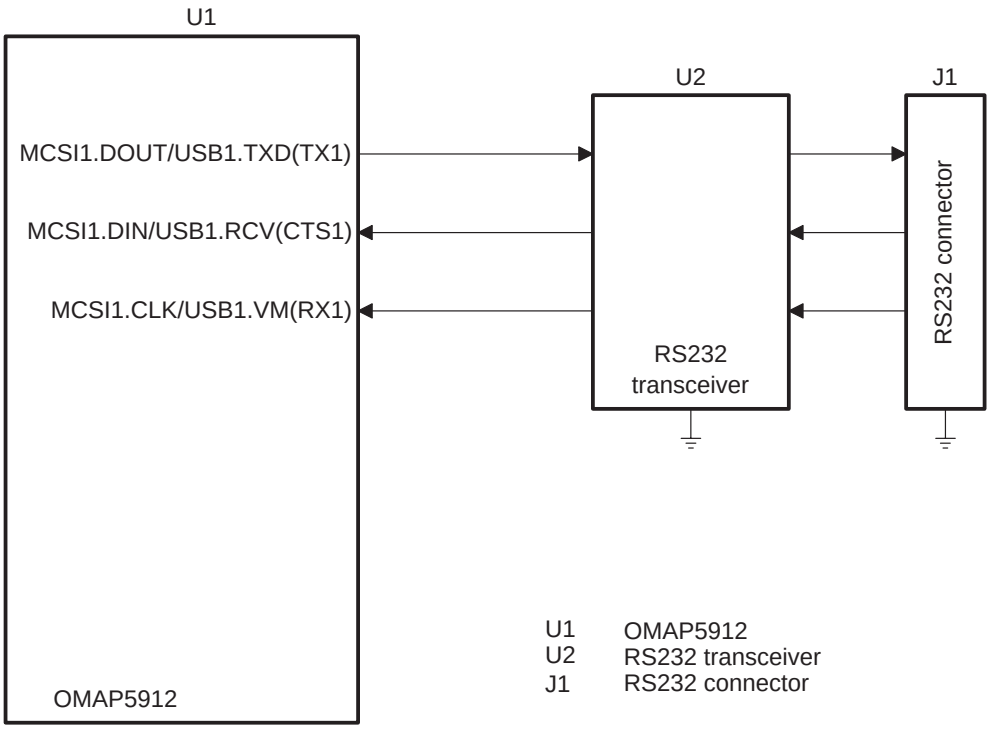
Figure 88. OMAP5912 Pin Group 1 to Group 2 Internal Loopback (HMC_MODE = 7)



UART 1 on USB Pin Group 2 Pins

When configured for some HMC_MODE values, the USB signal multiplexing mechanism routes UART1 signals to some OMAP5912 USB pin group 1 pins (see Figure 89).

Figure 89. External Connectivity When USB Pin Group 1 Configured for UART1 Signalling



Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- MCSI1.DOUT/USB1.TXD
- MCSI1.BCLK/USB1.VM
- MCSI1.DIN/USB1.RCV

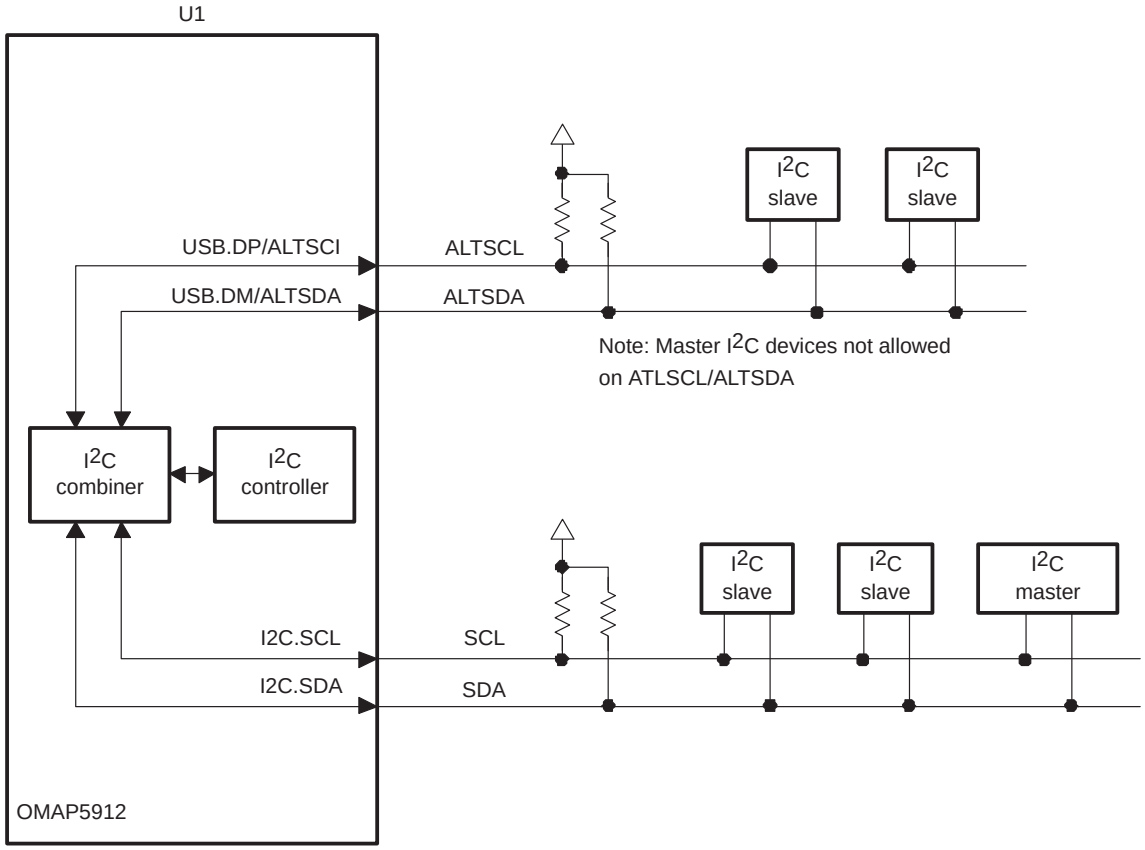
See Table 74.

HMC_MODE must be set to a value that routes the UART1 to USB pin group 1 (see Table 73). OTG_SYSCON_1.USB1_TRX_MODE must be set to 3 to allow proper operation of the 3-wire connection to the external transceiver.

I²C on Pin Group 0

OMAP5912 supports a mode in which the OMAP5912 I²C signals are brought out via USB pin group 0 pins. In this mode of operation, top-level pin multiplexing is configured to provide the SCL signal on the USB.DP pin and the SDA signal on the USB.DM pin. This is configured using the USB_TRANSCEIVER_CTRL.CONF_USB0 register, (see Figure 90). You must also set USB_TRANSCEIVER_CTRL.CONF_USB0_ISOLATE to 1 to ensure that the I²C signals have no effect on the OMAP5912 USB controllers.

Figure 90. I²C on USB Pin Group 0



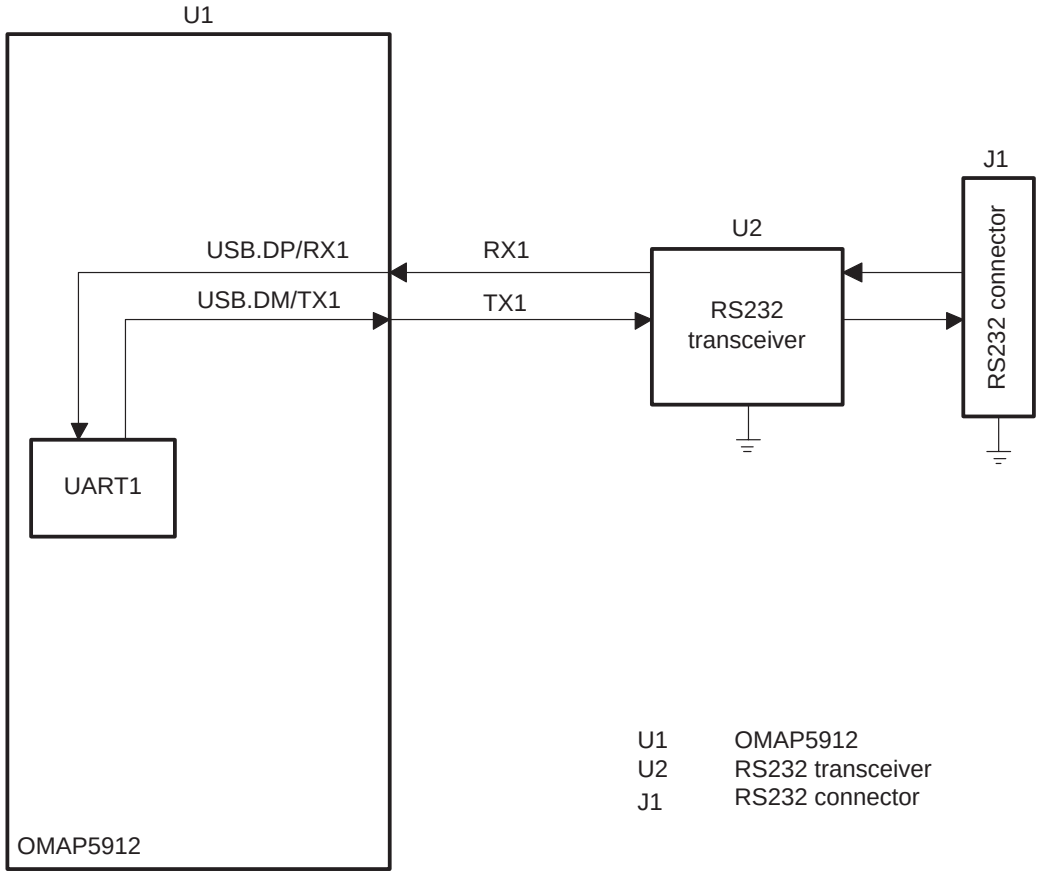
This mode of operation does not allow mastering I²C devices on the I²C link on pins USB.DP and USB.DM. Mastering I²C is allowed on SCL and SDA. A master I²C device connected externally to SCL and SDA is only able to access the I²C slaves on SCL and SDA or the OMAP5912 I²C controller slave function. An external master I²C device on SCL and SDA cannot access I²C slaves on ALTSCL and ALTSDA.

This mode of operation implements only one I^2C controller. OMAP5912 internally combines ALTSCL and SCL, and ALTSDA and SDA. This means that it is not possible to implement I^2C slaves with identical I^2C addresses on both links.

UART1 on Pin Group 0

OMAP5912 supports a mode in which the OMAP5912 UART1.TX and UART1.RX signals are brought out via USB pin group 0 pins. In this mode of operation, top-level pin multiplexing is configured to provide the RX input on USB.DP/UART1.RX and provide the TX output on USB.DM/UART1.TX. This is configured using the USB_TRANSCEIVER_CTRL.CONF_USB0 register (see Figure 91). When using this mode, set USB_TRANSCEIVER_CTRL.CONF_USB0_ISOLATE to 1 to ensure that the UART1 signals have no effect on the OMAP5912 USB controllers.

Figure 91. UART1 on USB Pin Group 0



4.11 Conflicts Between USB Signal Multiplexing and Top-Level Multiplexing

When OMAP5912 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 OMAP5912 top-level signal multiplexing is used.

When OMAP5912 top-level signal multiplexing selects non-USB functionality for a pin but the USB signal multiplexing is set to use that pin as an input, the OMAP5912 top-level signal multiplexing presents a low level to the USB signal multiplexer.

It may be useful to select a HMC_MODE value that brings some USB signals to the OMAP5912 top-level signal multiplexing, but then set the top-level signal multiplexing to ignore those USB signals.

4.12 OMAP5912 USB Hardware Considerations

4.12.1 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. Further information, including data sheets and application notes, can be found at the Texas Instruments web site.

Although an OTG transceiver has the ability to power VBUS within the OTG specification limits for an OTG dual role device, 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 (via a standard-A receptacle to mini-A plug as defined in the OTG specification), a VBUS switch as mentioned above can be used to meet those requirements.

4.12.2 Transient Suppression for USB Connectors

It is important to provide transient suppression for USB connectors. Electrostatic discharge that occurs when a user connects or disconnects a USB cable can have a dramatic effect on a system if not suppressed. Texas Instruments offers several devices for transient suppression on USB connections, such as the SN65220, SN65240, and SN75240 universal serial bus port transient suppressor devices. Further information, including data sheets and application notes, can be found at the Texas Instruments web site.

4.12.3 VBUS Monitoring for USB Function Controller

A USB function controller must be capable of monitoring the VBUS voltage provided by the upstream USB host controller. The OMAP5912 device provides the input pin USB.VBUS, which is provided to the OMAP5912 USB function controller (when HMC_MODE provides USB device functionality to a USB pin group). This input is a CMOS input that is not rated for the full VBUS range defined by the USB specification. An external signal level shifter is required to convert the VBUS signal range to a range suitable for the OMAP5912 USB.VBUS pin.

4.12.4 USB D+ Pullup Enable for USB Function Controller

When using a USB signal multiplexing mode that provides USB function controller signals to OMAP5912 pins, the OMAP5912 top-level pin multiplexing options lead to several possible USB pullup implementations.

When the USB.PUEN pin is set for top-level multiplexing configuration 0, the USB.PUEN pin is driven low when the pullup is active and is driven high when the pullup is inactive. In this mode of operation, an external inverter or an external 3-statable device can be used to provide a nominal 3.3-V signal to the supply end of the USB D+ pullup.

When the USB.PUEN pin is set for top-level multiplexing configuration 1, the USB.PUEN pin provides a clock output and cannot be used to control the USB pullup.

When the USB.PUEN pin is set for top-level multiplexing configuration 3, the USB.PUEN pin is driven high when the pullup is active and is driven low when the pullup is inactive. In this mode of operation, the pullup resistor can be connected directly between the OMAP5912 USB.PUEN and the D+ signal.

4.12.5 MPU_BOOT Signal Sharing

The OMAP5912 device implements shared functionality on the MPU_BOOT/USB1.SUSP pin. The MPU_BOOT pin is sampled at hardware reset. If low, the MPU processor boots from memory connected to CS0 on the EMIFS; if high, the MPU processor enters the boot overlay mode, causing it to boot from memory connected to CS3 on the EMIFS. After reset, the pin can be configured for other functionality, such as the USB1.SUSP output. The MPU_BOOT signal has an internal pulldown resistor that is enabled by default. The boot overlay mode requires an external pullup resister. The internal pulldown can be disabled in the OMAP configuration registers.

4.12.6 USB D+, D- Pulldown for USB Function Controller

System implementations that use the OMAP5912 USB function 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. The OMAP5912 integrated USB transceiver includes programmable weak pulldowns:

- When USB_TRANSCEIVER_CTRL.CONF_USB0_PWRDN_DN is 0, the internal weak pulldown is enabled for OMAP5912 pin USB.DM.
- When USB_TRANSCEIVER_CTRL.CONF_USB0_PWRDN_DP is 0, the internal weak pulldown is enabled for OMAP160 pin USB.DP.

Index

N

O

P

R

S

T

U

OMAP5912 Multimedia Processor Multichannel Buffered Serial Ports (McBSPs) Reference Guide

Literature Number: SPRU762B
October 2004

OMAP™
TEXAS INSTRUMENTS TECHNOLOGY

TEXAS INSTRUMENTS

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.

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
		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 document describes the three multichannel buffered serial ports (McBSPs) available on the OMAP5912 device.

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.

Related Documentation From Texas Instruments

Documentation that describes the OMAP5912 device, related peripherals, and other technical collateral, is available in the OMAP5912 Product Folder on TI's website: www.ti.com/omap5912.

Trademarks

OMAP and the OMAP symbol are trademarks of Texas Instruments.

Contents

Figures

Tables

This page is intentionally left blank.

Multichannel Buffered Serial Ports (McBSPs)

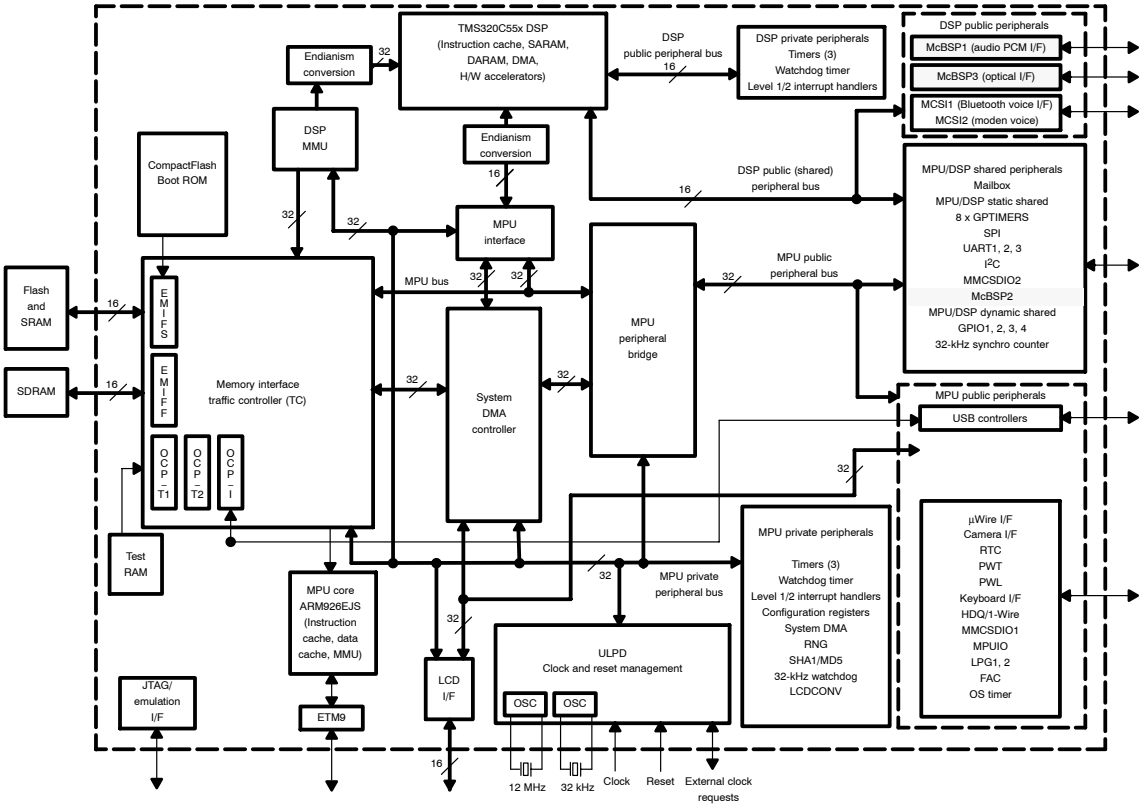
This document describes the three multichannel buffered serial ports (McBSPs) available on the OMAP5912 device.

1 Introduction to McBSPs

The device provides multiple high-speed multichannel buffered serial ports (McBSPs) that allow direct interface to codecs and other devices in a system.

Figure 1 shows the processor with the McBSP area highlighted.

Figure 1. McBSP Area



Section 1 through section 11 describe McBSP functionality common to all three McBSPs. Section 12, *McBSP in the Device,* and section 13, *McBSP Registers,* provide specific application information.

1.1 Key Features of the McBSPs

The McBSPs feature:

- Full-duplex communication
- Double-buffered transmission and triple-buffered reception, which allow a continuous data stream
- Independent clocking and framing for reception and for transmission
- The 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 enable or disable block 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
- Support for external generation of clock signals and frame-synchronization signals
- A programmable sample rate generator for internal generation and control of clock signals and frame-synchronization signals
- Programmable polarity for frame-synchronization pulses and clock signals
- Direct interface to:
  - T1/E1 framers
  - Multivendor integration protocol (MVIP) switching compatible and ST-BUS compliant devices including:
    - MVIP framers
    - H.100 framers
    - SCSA framers
  - IOM-2 compliant devices
  - I2S compliant devices
  - SPI devices
- A wide selection of data sizes: 8, 12, 16, 20, 24, and 32 bits

Note:

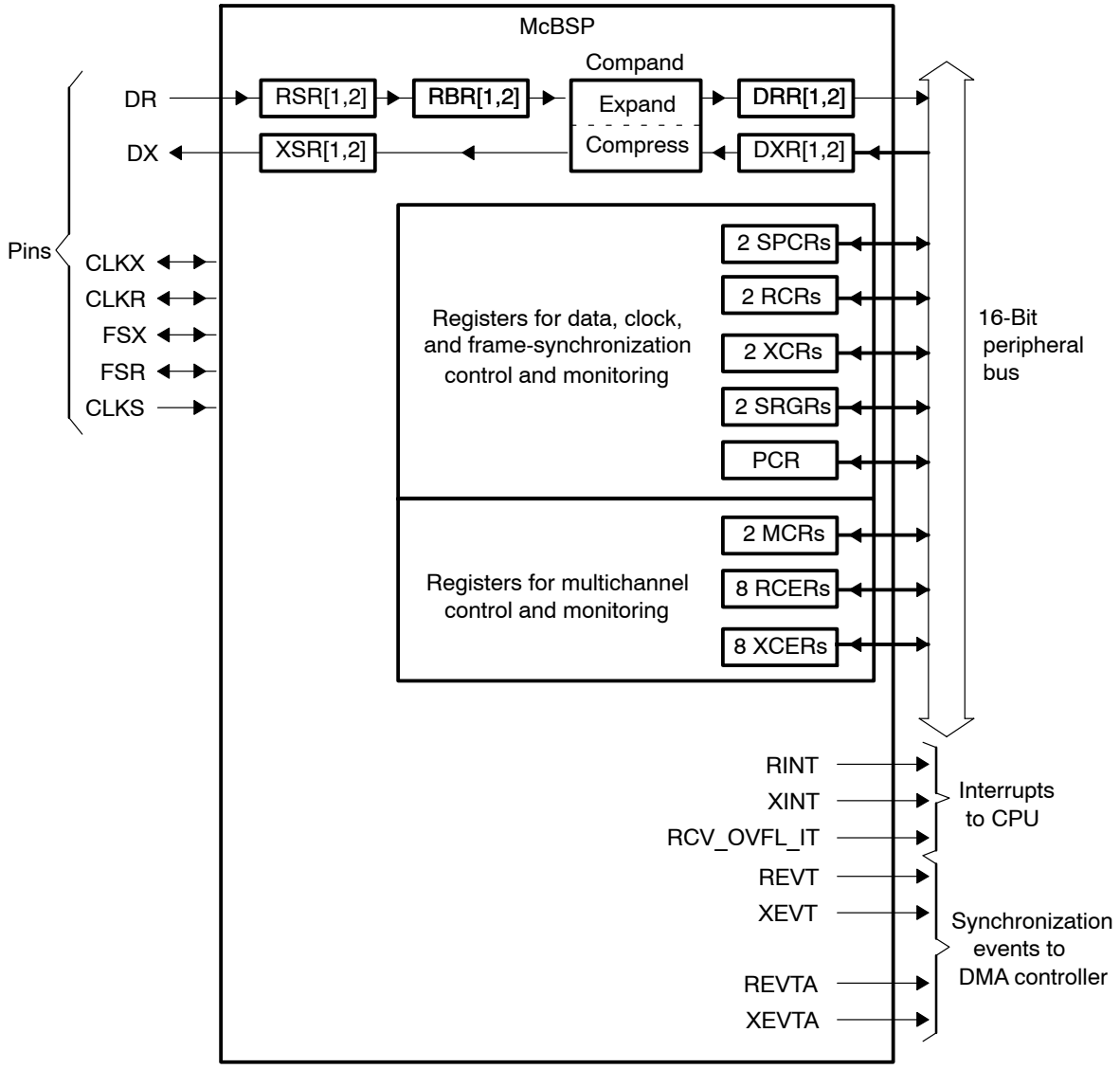
A value of the chosen data size is referred to as a *serial word* or *word* throughout the McBSP documentation. Elsewhere, *word* is used to describe a 16-bit value.

- μ-law and A-law companding
- The option of transmitting/receiving 8-bit data with the LSB first
- Status bits for flagging exception/error conditions
- The capability to use the McBSP pins as general-purpose I/O pins

1.2 McBSP Generic Block Diagram

The McBSP consists of a data-flow path and a control path connected to external devices by seven pins as shown in Figure 2. The figure and the text in this section use generic pin names. For the signal names on each McBSP, see section 1.3.

Figure 2. Conceptual Block Diagram of the McBSP



Data is communicated to devices interfaced with the McBSP via the data transmit (DX) pin for transmission and via the data receive (DR) pin for reception. Control information in the form of clocking and frame synchronization is communicated via the following pins: CLKX (transmit clock), CLKR (receive clock), FSX (transmit frame synchronization), and FSR (receive frame synchronization).

The CPU and the DMA controller communicate with the McBSP through 16-bit-wide registers accessible via the internal peripheral bus. The CPU or the DMA controller writes the data to be transmitted to the data transmit registers (DXR1, DXR2). Data written to the DXRs is shifted out to DX via the transmit shift registers (XSR1, XSR2). Similarly, receive data on the DR pin is shifted into the receive shift registers (RSR1, RSR2) and copied into the receive buffer registers (RBR1, RBR2). The contents of the RBRs is then copied to the DRRs, which can be read by the CPU or the DMA controller. This allows simultaneous movement of internal and external data communications.

DRR2, RBR2, RSR2, DXR2, and XSR2 are not used (written, read, or shifted) if the serial word length is 8 bits, 12 bits, or 16 bits. For larger word lengths, these registers are needed to hold the most significant bits.

The remaining registers in Figure 2 are registers for controlling McBSP operation.

1.3 McBSP Pins

Table 1 describes the McBSP interface pins. For information on using these pins for general-purpose input/output (GPIO), see section 9.

Table 1. McBSP Interface Pins

Pin	McBSP1	McBSP2	McBSP3	Direction	Possible Uses
CLKR		MCBSP2.CLKR		I/O	Supplying or reflecting the receive clock; supplying the input clock of the sample rate generator; general-purpose I/O
CLKX	MCBSP1.CLKX	MCBSP2.CLKX	MCBSP3.CLKX	I/O	Supplying or reflecting the transmit clock; supplying the input clock of the sample rate generator; general-purpose I/O
CLKS	MCBSP1.CLKS			I	Supplying the input clock of the sample rate generator; general-purpose input
DR	MCBSP1.DR	MCBSP2.DR	MCBSP3.DR	I	Receiving serial data; general-purpose input

Table 1. McBSP Interface Pins (Continued)

Pin	McBSP1	McBSP2	McBSP3	Direction	Possible Uses
DX	MCBSP1.DX	MCBSP2.DX	MCBSP3.DX	O	Transmitting serial data; general-purpose output
FSR		MCBSP2.FSR		I/O	Supplying or reflecting the receive frame-sync signal; controlling sample rate generator synchronization, when GSYNC = 1 (see section 3.3); general-purpose I/O
FSX	MCBSP1.FSX	MCBSP.FSX	MCBSP3.FSX	I/O	Supplying or reflecting the transmit frame-sync signal; general-purpose I/O

1.4 McBSP Register Addresses

See section 13, *McBSP Registers*, for McBSP register memory maps and detailed register descriptions.

2 McBSP Operation

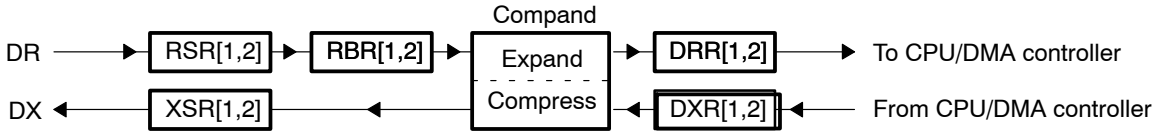
This section addresses the following topics:

- Data transfer process
- Clocking and framing data
- Frame phases
- McBSP reception
- McBSP transmission
- Interrupts and DMA events generated by McBSPs

2.1 Data Transfer Process of McBSPs

Figure 3 shows a diagram of the McBSP data transfer paths. The McBSP receive operation is triple-buffered, and transmit operation is double-buffered. The use of registers varies, depending on whether the defined length of each serial word is 16 bits.

Figure 3. McBSP Data Transfer Paths



2.1.1 Data Transfer Process for Word Length of 8, 12, or 16 Bits

If the word length is 16 bits or smaller, only one 16-bit register is needed at each stage of the data transfer paths. The registers DRR2, RBR2, RSR2, DXR2, and XSR2 are not used (written, read, or shifted).

Receive data arrives on the DR pin and is shifted into receive shift register 1 (RSR1). Once a full word is received, the content of RSR1 is copied to receive buffer register 1 (RBR1), if RBR1 is not full with previous data. RBR1 is then copied to data receive register 1 (DRR1), unless the previous content of DRR1 has not been read by the CPU or the DMA controller. If the companding feature of the McBSP is implemented, the required word length is 8 bits and receive data is expanded into the appropriate format before being passed from RBR1 to DRR1. For more details about reception, see section 2.5.

Transmit data is written by the CPU or the DMA controller to data transmit register 1 (DXR1). If there is no previous data in transmit shift register (XSR1), the value in DXR1 is copied to XSR1; otherwise, DXR1 is copied to XSR1 when the last bit of the previous data is shifted out on the DX pin. If selected, the companding module compresses 16-bit data into the appropriate 8-bit format before passing it to XSR1. After transmit frame synchronization, the transmitter begins shifting bits from XSR1 to the DX pin. For more details about transmission, see section 2.6.

2.1.2 Data Transfer Process for Word Length of 20, 24, or 32 Bits

If the word length is larger than 16 bits, two 16-bit registers are needed at each stage of the data transfer paths. The registers DRR2, RBR2, RSR2, DXR2, and XSR2 are needed to hold the most significant bits.

Receive data arrives on the DR pin and is shifted first into RSR2 and then into RSR1. Once the full word is received, the contents of RSR2 and RSR1 are copied to RBR2 and RBR1, respectively, if RBR1 is not full. Then the contents of RBR2 and RBR1 are copied to DRR2 and DRR1, respectively, unless the previous content of DRR1 has not been read by the CPU or the DMA controller. The CPU or the DMA controller must read data from DRR2 first and then from DRR1. When DRR1 is read, the next RBR-to-DRR copy occurs. For more details about reception, see section 2.5.

For transmission, the CPU or the DMA controller must write data to DXR2 first and then to DXR1. When new data arrives in DXR1, if there is no previous data in XSR1, the contents of DXR2 and DXR1 are copied to XSR2 and XSR1, respectively; otherwise, the contents of the DXRs are copied to the XSRs when the last bit of the previous data is shifted out on the DX pin. After transmit frame synchronization, the transmitter begins shifting bits from the XSRs to the DX pin. For more details about transmission, see section 2.6.

2.2 Companding (Compressing and Expanding) Data

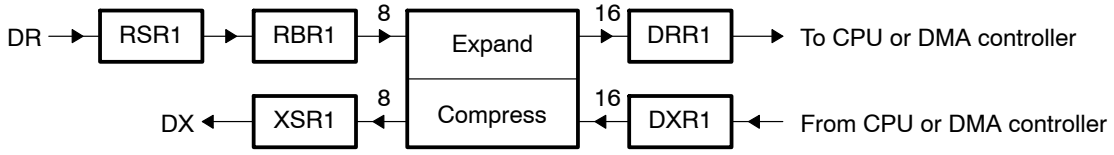
Companding (COMpressing and exPANDing) hardware allows compression and expansion of data in either μ-law or A-law format. The companding standard employed in the United States and Japan is μ-law. The European companding standard is referred to as A-law. The specifications for μ-law and A-law log PCM are part of the CCITT G.711 recommendation.

A-law and μ-law allow 13 bits and 14 bits of dynamic range, respectively. Any 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 via the CPU or DMA controller must be at least 16 bits wide.

The μ-law and A-law formats both encode data into 8-bit code words. Companded data is always 8 bits wide; the appropriate word length bits (RWDLEN1, RWDLEN2, XWDLEN1, XWDLEN2) must therefore be set to 0, indicating an 8-bit wide serial data stream. If companding is enabled and either of the frame phases does not have an 8-bit word length, companding continues as if the word length is 8 bits.

Figure 4 illustrates the companding processes. When companding is chosen for the transmitter, compression occurs during the process of copying data from DXR1 to XSR1. The transmit data is encoded according to the specified companding law (A-law or μ-law). When companding is chosen for the receiver, expansion occurs during the process of copying data from RBR1 to DRR1. The receive data is decoded to twos-complement format.

Figure 4. Companding Processes

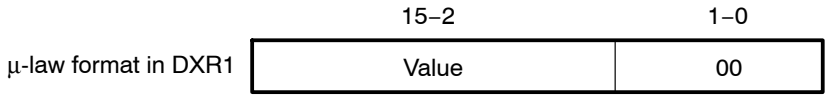


2.2.1 Companding Formats

For reception, the 8-bit compressed data in RBR1 is expanded to left-justified 16-bit data in DRR1. The receive sign-extension and justification mode specified in RJUST is ignored when companding is used.

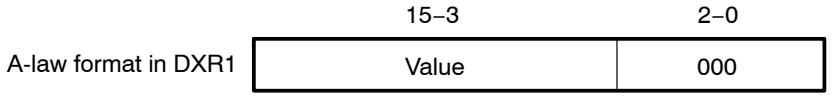
For transmission using μ-law compression, the 14 data bits must be left-justified in DXR1 and the remaining two low-order bits must be filled with 0s, as shown in Figure 5.

Figure 5. μ-Law Transmit Data Companding Format



For transmission using A-law compression, the 13 data bits must be left-justified in DXR1, with the remaining three low-order bits filled with 0s, as shown in Figure 6.

Figure 6. A-Law Transmit Data Companding Format



2.2.2 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 both XCOMPAND and RCOMPAND enable the same companding format.

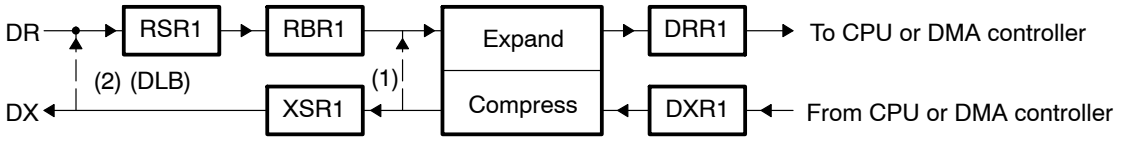
Figure 7 shows two methods by which the McBSP can compand internal data. Data paths for these two methods are used to indicate:

- When both the transmit and receive sections of the serial port are reset, DRR1 and DXR1 are connected internally through the companding logic. Values from DXR1 are compressed, as selected by XCOMPAND, and then expanded, as selected by RCOMPAND. RRDY and XRDY bits are not set. However, data is available in DRR1 within four CPU clocks after being written to DXR1.

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. DRR1 and DXR1 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 RCOMPAND and XCOMPAND. Receive and transmit interrupts (RINT when RINTM = 0 and XINT when XINTM = 0) or synchronization events (REVT and XEVT) allow synchronization of the CPU or DMA to these conversions, respectively. Here, the time for this companding depends on the serial bit rate selected.

Figure 7. Two Methods by Which the McBSP Can Compand Internal Data



2.2.3 Reversing Bit Order: Option to Transfer LSB First

Generally, the McBSP transmits or receives all data with the most significant bit (MSB) first. However, certain 8-bit data protocols that do not use companded data require the least significant bit (LSB) to be transferred first. If you set XCOMPAND = 01b in XCR2, the bit ordering of 8-bit words is reversed (LSB first) before being sent from the serial port. If you set RCOMPAND = 01b in RCR2, the bit ordering of 8-bit words is reversed 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.

2.3 Clocking and Framing Data

This section explains basic concepts and terminology important for understanding how McBSP data transfers are timed and delimited.

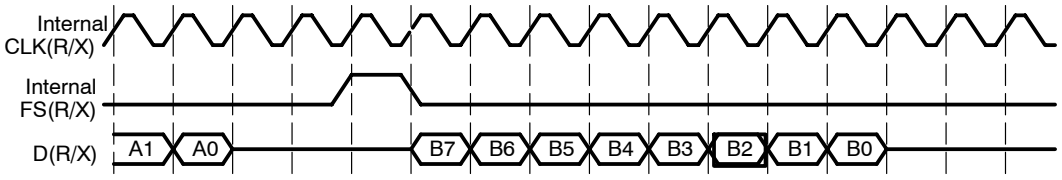
2.3.1 Clocking

Data is shifted one bit at a time from the DR pin to the RSR(s) or from the XSR(s) to the 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 DR pin to the RSR(s). The transmit clock signal (CLKX) controls bit transfers from the XSR(s) to the DX pin. CLKR or CLKX can be derived from a pin at the boundary of the McBSP or derived from inside the McBSP. The polarities of CLKR and CLKX are programmable.

In the example in Figure 8, the clock signal controls the timing of each bit transfer on the pin.

Figure 8. Example of Clock Signal Control of Bit Transfer Timing



Note:

The McBSP cannot operate at a frequency faster 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 sample rate generator for CLKX and/or CLKR, choose an appropriate input clock frequency and divide down value (CLKDV).

2.3.2 Serial Words

Bits traveling between a shift register (RSR or XSR) and a data pin (DR or DX) are transferred in a group called a serial word. You can define how many bits are in a word.

Bits coming in on the DR pin are held in RSR until RSR holds a full serial word. Only then is the word passed to RBR (and ultimately to the DRR).

During transmission, XSR does not accept new data from DXR until a full serial word has been passed from XSR to the DX pin.

In the example in Figure 8, an 8-bit word size was defined (see bits 7 through 0 of word B being transferred).

2.3.3 Frames and Frame Synchronization

One or more words are transferred in a group called a frame. You can define how many words are in a frame.

All of the words in a frame are sent in a continuous stream. However, there can be pauses between frame transfers. The McBSP uses frame-synchronization signals to determine when each frame is received/transmitted. When a pulse occurs on a frame-synchronization 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 receive frame-synchronization (FSR) signal initiate frame transfers on DR. Pulses on the transmit frame-sync (FSX) signal initiate frame transfers on DX. FSR or FSX can be derived from a pin at the boundary of the McBSP or derived from inside the McBSP.

In the example in Figure 8, a one-word frame is transferred when a frame-synchronization pulse occurs.

In McBSP operation, the inactive-to-active transition of the frame-synchronization signal indicates the start of the next frame. For this reason, the frame-synchronization signal may be high for an arbitrary number of clock cycles. Only after the signal is recognized to have gone inactive, and then active again, does the next frame synchronization occur.

2.3.4 Detecting Frame-Synchronization 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 synchronization, these interrupts can be sent in response to frame-synchronization pulses. Set the appropriate interrupt mode bits to 10b (for reception, RINTM = 10b; for transmission, XINTM = 10b).

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 and FSRP/FSXP still select the appropriate source and polarity of frame synchronization. Thus, even when the serial port is in the reset state, these signals are synchronized to the 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-synchronization pulse can be detected, and after this occurs, the CPU can take the serial port out of reset safely.

2.3.5 Ignoring Frame-Synchronization Pulses

The McBSP can be configured to ignore transmit and/or receive frame-synchronization pulses. To have the receiver or transmitter recognize frame-synchronization pulses, clear the appropriate frame-synchronization ignore bit (RFIG = 0 for the receiver, XFIG = 0 for the transmitter). To have the receiver or transmitter ignore frame-synchronization pulses until the desired frame length or number of words is reached, set the appropriate frame-synchronization ignore bit (RFIG = 1 for the receiver, XFIG = 1 for the transmitter). For more details on unexpected frame-synchronization pulses, see one of the following topics:

- *Unexpected Receive Frame-Synchronization Pulse* (see section 4.2)
- *Unexpected Transmit Frame-Synchronization Pulse* (see section 4.5)

You can also use the frame-synchronization ignore function for data packing (for more details, see section 11.2).

2.3.6 Frame Frequency

The frame frequency is determined by the period between frame-synchronization pulses and is defined as shown by Equation 1.

Equation 1. McBSP Frame Frequency

$$\text{Frame Frequency} = \frac{\text{Clock Frequency}}{\text{Number of Clock Cycles Between Frame−Sync Pulses}}$$

The frame frequency can be increased by decreasing the time between frame-synchronization pulses (limited only by the number of bits per frame). As the frame transmit frequency increases, the inactivity period between the data packets for adjacent transfers decreases to zero.

2.3.7 Maximum Frame Frequency

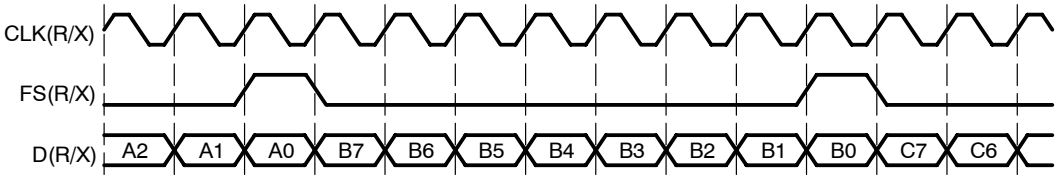
The minimum number of clock cycles between frame synchronization pulses is equal to the number of bits transferred per frame. The maximum frame frequency is defined as shown by Equation 2.

Equation 2. McBSP Maximum Frame Frequency

$$\text{Maximum Frame Frequency} = \frac{\text{Clock Frequency}}{\text{Number of Bits Per Frame}}$$

Figure 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 9. McBSP Operating at Maximum Packet Frequency



If there is a 1-bit data delay as shown in this figure, the frame-synchronization pulse overlaps the last bit transmitted in the previous frame. Effectively, this permits a continuous stream of data, making frame-synchronization pulses redundant. Theoretically, only an initial frame-synchronization pulse is required to initiate a multipacket transfer.

The McBSP supports operation of the serial port in this fashion by ignoring the successive frame-synchronization pulses. Data is clocked into the receiver or clocked out of the transmitter during every clock cycle.

Note:

For XDATDLY = 0 (0-bit data delay), the first bit of data is transmitted asynchronously to the internal transmit clock signal (CLKX). For more details, see section 8.12, *Set the Transmit Data Delay*.

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, a user 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 permits the user to compose frames for custom applications or, in general, to maximize the efficiency of data transfers.

2.4.1 Number of Phases, Words, and Bits Per Frame

Table 2 shows which bit-fields in the receive control registers (RCR1 and RCR2) and in the transmit control registers (XCR1 and XCR2) 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 2. McBSP Register Bits

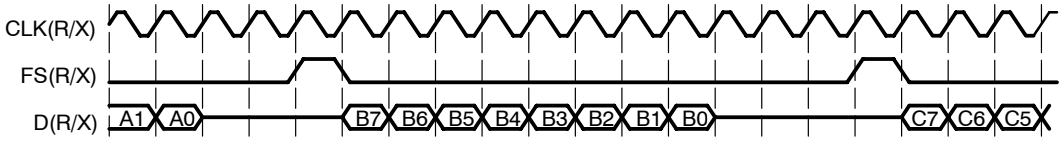
Operation	Number of Phases	Words per Frame Set With	Bits per Word Set With
Reception	1 (RPHASE = 0)	RFRLEN1	RWDLEN1
Reception	2 (RPHASE = 1)	RFRLEN1 and RFRLEN2	RWDLEN1 for phase 1 RWDLEN2 for phase 2
Transmission	1 (XPHASE = 0)	XFRLEN1	XWDLEN1
Transmission	2 (XPHASE = 1)	XFRLEN1 and XFRLEN2	XWDLEN1 for phase 1 XWDLEN2 for phase 2

2.4.2 Single-Phase Frame Example

Figure 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 DX and DR pins are available one clock cycle after FS(R/X) goes active. The figure makes the following assumptions:

- ❏ (R/X)PHASE = 0: Single-phase frame
- ❏ (R/X)FRLEN1 = 0b: 1 word per frame
- ❏ (R/X)WDLEN1 = 000b: 8-bit word length
- ❏ (R/X)FRLEN2 and (R/X)WDLEN2 are ignored
- ❏ CLK(X/R)P = 0: Receive data clocked on falling edge; transmit data clocked on rising edge
- ❏ FS(R/X)P = 0: Active-high frame-synchronization signals
- ❏ (R/X)DATDLY = 01b: 1-bit data delay

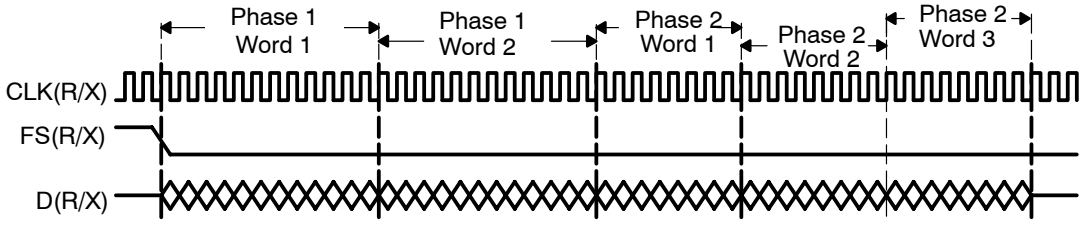
Figure 10. Single-Phase Frame for a McBSP Data Transfer



2.4.3 Dual-Phase Frame Example

Figure 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 11. Dual-Phase Frame for a McBSP Data Transfer

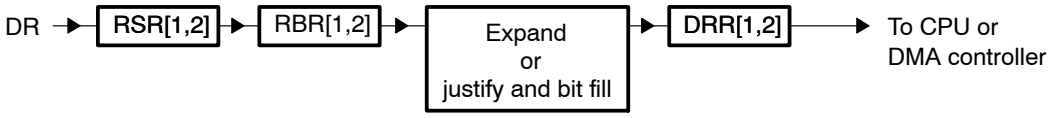


2.5 McBSP Reception

This section explains the fundamental process of reception in the McBSP. For details about how to program the McBSP receiver, see *Receiver Configuration* on page 82.

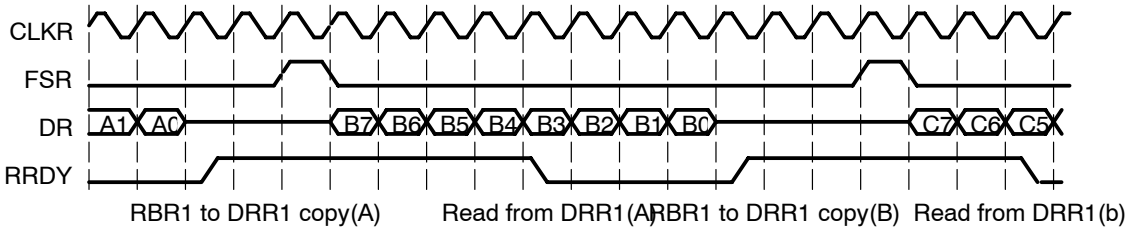
Figure 12 and Figure 13 show how reception occurs in the McBSP. Figure 12 shows the physical path for the data. Figure 13 is a timing diagram showing signal activity for one possible reception scenario. A description of the process follows the figures.

Figure 12. McBSP Reception Physical Data Path



RSR[1,2]: Receive shift registers 1 and 2
RBR[1,2]: Receive buffer registers 1 and 2
DRR[1,2]: Data receive registers 1 and 2

Figure 13. McBSP Reception Signal Activity



CLKR: Internal receive clock
FSR: Internal receive frame-synchronization signal
DR: Data on DR pin
RRDY: Status of receiver ready bit (high is 1)

The following process describes how data travels from the DR pin to the CPU or to the DMA controller:

1) The McBSP waits for a receive frame-synchronization pulse on internal FSR.

2) When the pulse arrives, the McBSP inserts the appropriate data delay that is selected with the RDATDLY bits of RCR2.

   In the preceding timing diagram (Figure 13), a 1-bit data delay is selected.

3) The McBSP accepts data bits on the 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 most significant bits. For details on choosing a word length, see section 7.8, *Set the Receive Word Length(s)*.

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 most significant bits.

5) The McBSP copies the contents of the receive buffer register(s) into the data receive register(s), provided that DRR1 is not full with previous data. When DRR1 receives new data, the receiver ready bit (RRDY) is set in SPCR1. 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 DRR1 is used. If the word length is larger than 16 bits, DRR2 and DRR1 are used and DRR2 contains the most significant bits.

   If companding is used during the copy (RCOMPAND = 10b or 11b in RCR2), the 8-bit compressed data in RBR1 is expanded to a left-justified 16-bit value in DRR1. If companding is disabled, the data copied from RBR[1,2] to DRR[1,2] is justified and bit filled according to the RJUST bits.

6) The CPU or the DMA controller reads the data from the data receive register(s). When DRR1 is read, RRDY is cleared and the next RBR-to-DRR copy is initiated.

Note:

If both DRRs are required (word length larger than 16 bits), the CPU or the DMA controller must read from DRR2 first and then from DRR1. As soon as DRR1 is read, the next RBR-to-DRR copy is initiated. If DRR2 is not read first, the data in DRR2 is lost.

When activity is not properly timed, errors can occur. See the following topics for more details:

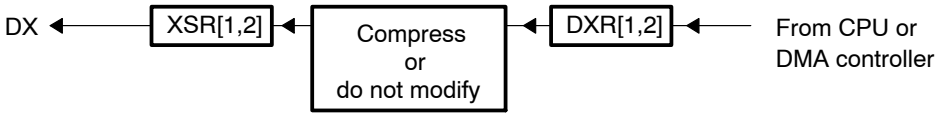
- *Overrun in the Receiver* (see section 4.1)
- *Unexpected Receive Frame-Synchronization Pulse* (see section 4.2)

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 8, *Transmitter Configuration*.

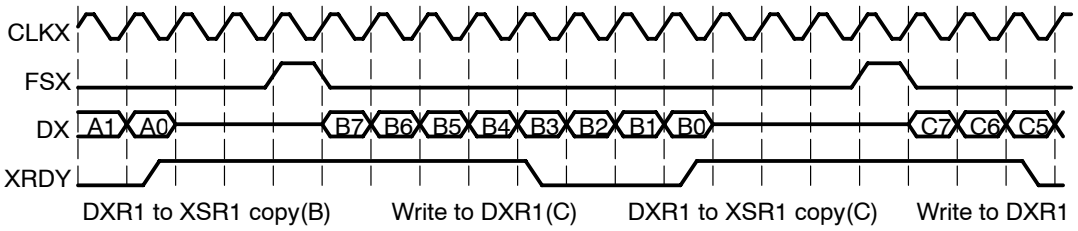
Figure 14 and Figure 15 show how transmission occurs in the McBSP. Figure 14 shows the physical path for the data. Figure 15 is a timing diagram showing signal activity for one possible transmission scenario. A description of the process follows the figures.

Figure 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 15. McBSP Transmission Signal Activity



CLKX: Internal transmit clock
FSX: Internal transmit frame-synchronization signal
DX: Data on DX pin
XRDY: Status of transmitter ready bit (high is 1)

1) The CPU or the DMA controller writes data to the data transmit register(s). When DXR1 is loaded, the transmitter ready bit (XRDY) is cleared in SPCR2 to indicate that the transmitter is not ready for new data.

   If the word length is 16 bits or smaller, only DXR1 is used. If the word length is larger than 16 bits, DXR2 and DXR1 are used and DXR2 contains the most significant bits. For details on choosing a word length, see section 8.8, *Set the Transmit Word Length(s)*.

Note:

If both DXRs are needed (word length larger than 16 bits), the CPU or the DMA controller must load DXR2 first and then load DXR1. As soon as DXR1 is loaded, the contents of both DXRs are copied to the transmit shift registers (XSRs), as described in the next step. If DXR2 is not loaded first, the previous content of DXR2 is passed to the XSR2.

2) When new data arrives in DXR1, 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 most significant bits.

   If companding is used during the transfer (XCOMPAND = 10b or 11b in XCR2), the McBSP compresses the 16-bit data in DXR1 to 8-bit data in the μ-law or A-law format in XSR1. If companding is disabled, the McBSP passes data from the DXR(s) to the XSR(s) without modification.

3) The McBSP waits for a transmit frame-synchronization pulse on internal FSX.

4) When the pulse arrives, the McBSP inserts the appropriate data delay that is selected with the XDATDLY bits of XCR2.

   In the preceding timing diagram (Figure 15), a 1-bit data delay is selected.

5) The McBSP shifts data bits from the transmit shift register(s) to the DX pin.

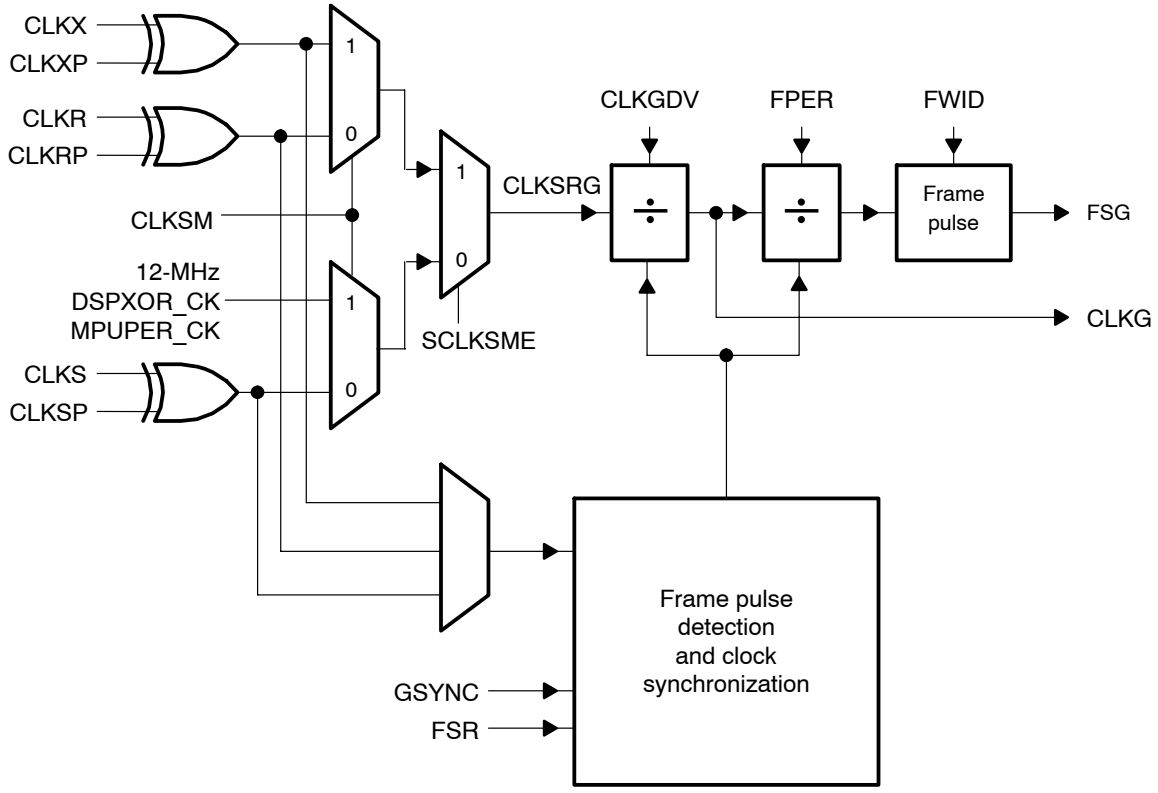
When activity is not properly timed, errors can occur. See the following topics for more details:

- *Overwrite in the Transmitter* (section 4.3)
- *Underflow in the Transmitter* (section 4.4)
- *Unexpected Transmit Frame-Synchronization Pulse* (section 4.5)

3 McBSP Sample Rate Generator

Each McBSP contains a sample rate generator that can be used to generate an internal data clock (CLKG) and an internal frame-synchronization signal (FSG). CLKG can be used for bit shifting on the data receive (DR) pin and/or the data transmit (DX) pin. FSG can be used to initiate frame transfers on DR and/or DX. Figure 16 is a conceptual block diagram of the sample rate generator.

Figure 16. Conceptual Block Diagram of the Sample Rate Generator



The source clock for the sample rate generator (labeled CLKSRG in the diagram) can be supplied by either the 12-MHz DSPXOR_CK clock, the ARMPER_CK clock, or by an external pin (CLKS, CLKX, or CLKR). The source is selected with the SCLKME bit of PCR and the CLKSM bit of SRGR2. If a pin is used, the polarity of the incoming signal can be inverted with the appropriate polarity bit (CLKSP of SRGR2, CLKXP of PCR, or CLKRP of PCR).

The sample rate generator 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 CLKGDV bits of SRGR1 to produce CLKG.
- Frame period divide-down. CLKG is divided according to the FPER bits of SRGR2 to control the period from the start of a frame-pulse to the start of the next pulse.
- Frame-synchronization pulse-width countdown. CLKG cycles are counted according to the FWID bits of SRGR1 to control the width of each frame-synchronization pulse.

Note:

The McBSP cannot operate at a frequency faster than 1/2 the clock frequency. Choose an input clock frequency and a CLKDV value such that CLKG is less than or equal to 1/2 the clock frequency.

In addition to the three-stage clock divider, the sample rate generator has a frame-synchronization pulse detection and clock synchronization module that allows synchronization of the clock divide down with an incoming frame-synchronization pulse on the FSR pin. This feature is enabled or disabled with the GSYNC bit of SRGR2.

For details on getting the sample rate generator ready for operation, see section 3.4, *Reset and Initialization Procedure*.

3.1 Clock Generation in the Sample Rate Generator

The sample rate generator can produce a clock signal (CLKG) for use by the receiver, the transmitter, or both. Use of the sample rate generator to drive clocking is controlled by the clock mode bits (CLKRM and CLKXM) in the pin control register (PCR). 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 sample rate generator 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 3. The digital loopback mode (described in section 7.4) is selected with the DLB bit of SPCR1. The clock stop mode (described in section 6) is selected with the CLKSTP bits of SPCR1.

When using the sample rate generator as a clock source, make sure the sample rate generator is enabled ($\overline{\text{GRST}}$ = 1).

Table 3. Effects of DLB and CLKSTP on Clock Modes

Mode Bit Settings		Effect
CLKRM = 1	DLB = 0 (Digital loopback mode disabled)	CLKR is an output pin driven by the sample rate generator 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 sample rate generator 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 sample rate generator.

3.1.1 Choosing an Input Clock

The sample rate generator must be driven by an input clock signal from one of the four sources selectable with the SCLKME bit of PCR and the CLKSM bit of SRGR2 (see Table 4). When CLKSM = 1, the minimum divide down value in CLKGDV bits is 1. CLKGDV is described in section 3.1.3.

Note:

The McBSP cannot operate at a frequency faster than 1/2 the CPU clock frequency. Choose an input clock frequency and a CLKDV value such that CLKG is less than or equal to 1/2 the CPU clock frequency.

Table 4. Choosing an Input Clock for the Sample Rate Generator with the SCLKME and CLKSM Bits

SCLKME	CLKSM	Input Clock for Sample Rate Generator
0	0	Signal on CLKS pin
0	1	CPU clock
1	0	Signal on CLKR pin
1	1	Signal on CLKX pin

3.1.2 Choosing a Polarity for the Input Clock

As shown in Figure 17, when the input clock is received from a pin, you can choose the polarity of the input clock. The rising edge of CLKSRG generates CLKG and FSG, but you can determine which edge of the input clock causes a rising edge on CLKSRG. The polarity options and their effects are described in Table 5.

Figure 17. Possible Inputs to the Sample Rate Generator and the Polarity Bits

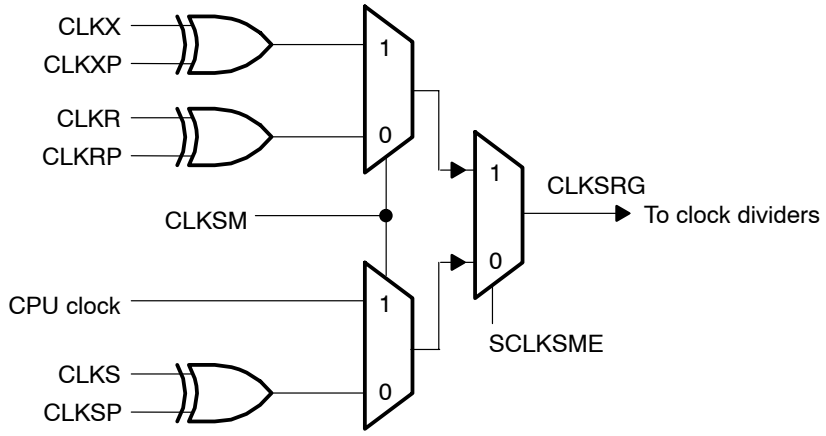


Table 5. Polarity Options for the Input to the Sample Rate Generator

Input Clock	Polarity Option	Effect
Signal on CLKS pin	CLKSP = 0 in SRGR2	Rising edge on CLKS pin generates transitions on CLKG and FSG.
	CLKSP = 1 in SRGR2	Falling edge on CLKS pin generates transitions on CLKG and FSG.
CPU clock	Always positive polarity	Rising edge of CPU clock generates transitions on CLKG and FSG.
Signal on CLKR pin	CLKRP = 0 in PCR	Falling edge on CLKR pin generates transitions on CLKG and FSG.
	CLKRP = 1 in PCR	Rising edge on CLKR pin generates transitions on CLKG and FSG.
Signal on CLKX pin	CLKXP = 0 in PCR	Rising edge on CLKX pin generates transitions on CLKG and FSG.
	CLKXP = 1 in PCR	Falling edge on CLKX pin generates transitions on CLKG and FSG.

3.1.3 Choosing a Frequency for the Output Clock (CLKG)

The input clock (CPU clock or external clock) can be divided down by a programmable value to drive CLKG. Regardless of the source to the sample rate generator, the rising edge of CLKSRG (see Figure 16) generates CLKG and FSG.

The first divider stage of the sample rate generator creates the output clock from the input clock. This divider stage uses a counter that is preloaded with the divide down value in the CLKGDV bits of SRGR1. The output of this stage is the data clock (CLKG). CLKG has the frequency represented by the following equation.

$$\text{CLKG frequency} = \frac{\text{Input clock frequency}}{(\text{CLKGDV} + 1)}$$

Thus, the 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.

Note:

The McBSP cannot operate at a frequency faster than 1/2 the CPU clock frequency. Choose an input clock frequency and a CLKDV value such that CLKG is less than or equal to 1/2 the CPU clock frequency.

3.1.4 Keeping CLKG Synchronized to an External Input Clock

When an external signal is selected to drive the sample rate generator (see section 3.1.1), the GSYNC bit in SRGR2 and the FSR pin can be used to configure the timing of the output clock (CLKG) relative to the input clock.

GSYNC = 1 ensures that the McBSP and an external device are dividing down the input clock with the same phase relationship. If GSYNC = 1, an inactive-to-active transition on the FSR pin triggers a resynchronization of CLKG and generation of FSG.

For more details about synchronization, see section 3.3.

3.2 Frame Synchronization Generation in the Sample Rate Generator

The sample rate generator can produce a frame-synchronization signal (FSG) for use by the receiver, the transmitter, or both.

If you want the receiver to use FSG for frame synchronization, make sure FSRM = 1. (When FSRM = 0, receive frame synchronization is supplied via the FSR pin.)

If you want the transmitter to use FSG for frame synchronization, you must set the following:

- FSXM = 1 in PCR: This indicates that transmit frame synchronization is supplied by the McBSP itself rather than from the FSX pin.
- FSGM = 1 in SRGR2: This indicates that when FSXM = 1, transmit frame synchronization is supplied by the sample rate generator. (When FSGM = 0 and FSXM = 1, the transmitter uses frame-synchronization pulses generated every time data is transferred from DXR[1,2] to XSR[1,2].)

In either case, the sample rate generator must be enabled ($\overline{\text{GRST}}$ = 1) and the frame-synchronization logic in the sample rate generator must be enabled ($\overline{\text{GRST}}$ = 0).

3.2.1 Choosing the Width of the Frame-Synchronization Pulse on FSG

Each pulse on FSG has a programmable width. You program the FWID bits of SRGR1, and the resulting pulse width is (FWID + 1) CLKG cycles, where CLKG is the output clock of the sample rate generator.

3.2.2 Controlling the Period Between the Starting Edges of Frame-Synchronization Pulses on FSG

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 sample rate generator:

- If the sample rate generator is using an external input clock and GSYNC = 1 in SRGR2, FSG pulses in response to an inactive-to-active transition on the FSR pin. Thus, the frame-synchronization period is controlled by an external device.
- Otherwise, you program the FPER bits of SRGR2, and the resulting frame-synchronization period is (FPER + 1) CLKG cycles, where CLKG is the output clock of the sample rate generator.

3.2.3 Keeping FSG Synchronized to an External Clock

When an external signal is selected to drive the sample rate generator (see section 3.1.1 on page 38), the GSYNC bit of SRGR2 and the FSR pin can be used to configure the timing of FSG pulses.

GSYNC = 1 ensures that the McBSP and an external device are dividing down the input clock with the same phase relationship. If GSYNC = 1, an inactive-to-active transition on the FSR pin triggers a resynchronization of CLKG and generation of FSG.

See section 3.3 for more details about synchronization.

3.3 Synchronizing Sample Rate Generator Outputs to an External Clock

The sample rate generator can produce a clock signal (CLKG) and a frame-synchronization signal (FSG) based on an input clock signal that is either the CPU clock signal or a signal at the CLKS, CLKR, or CLKX pin. When an external clock is selected to drive the sample rate generator, the GSYNC bit of SRGR2 and the FSR pin can be used to control the timing of CLKG and the pulsing of FSG relative to the chosen input clock.

Make GSYNC = 1 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 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 FPER bits of SRGR2 are ignored because the frame-synchronization period on FSG is determined by the arrival of the next frame-synchronization pulse on the FSR pin.

If GSYNC = 0, CLKG runs freely and is not resynchronized, and the frame-synchronization period on FSG is determined by FPER.

3.3.1 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 SRGR2 and FSXM = 1 in PCR). 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 sample rate generator clock drives the transmit and receive clocking (CLKRM = CLKXM = 1 in PCR). Therefore, the CLK(R/X) pin must not be driven by any other driving source.

3.3.2 Synchronization Examples

Figure 18 and Figure 19 show the clock and frame-synchronization operation with various polarities of CLKS (the chosen input clock) and FSR. These figures assume FWID = 0 in SRGR1, for an FSG pulse that is one CLKG cycle wide. The FPER bits of SRGR2 are not programmed; the period from the start of a frame-synchronization pulse to the start of the next pulse is determined by the arrival of the next inactive-to-active transition on the 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. The second figure has a slower CLKG frequency (it has a larger divide-down value in the CLKGDV bits of SRGR1).

Figure 18. CLKG Synchronization and FSG Generation When GSYNC = 1 and CLKGDV = 1

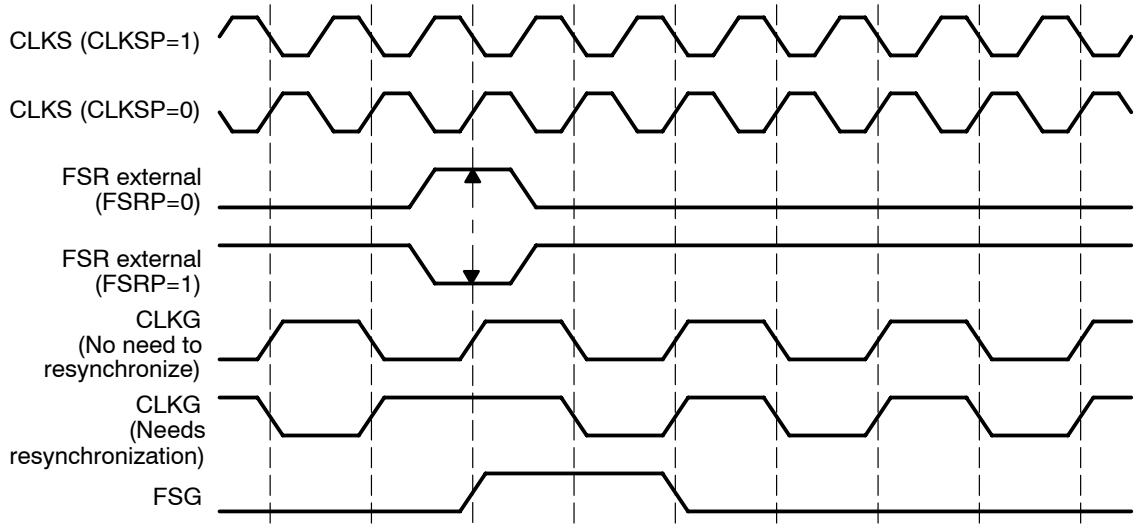
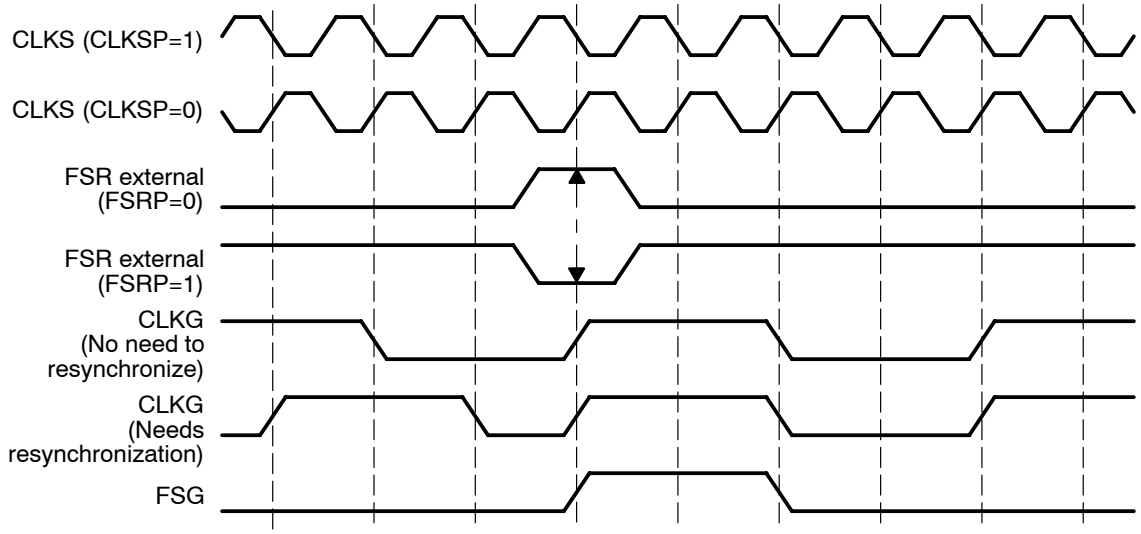


Figure 19. CLKG Synchronization and FSG Generation When GSYNC = 1 and CLKGDV = 3



3.4 Reset and Initialization Procedure for the Sample Rate Generator

To reset and initialize the sample rate generator:

Step 1: Place the McBSP/sample rate generator in reset.

During an OMAP5912 reset, the sample rate generator, the receiver, and the transmitter reset bits ($\overline{\text{GRST}}$, $\overline{\text{RRST}}$, and $\overline{\text{XRST}}$) are automatically forced to 0. Otherwise, during normal operation, the sample rate generator can be reset by making $\overline{\text{GRST}}$ = 0 in SPCR2, provided that CLKG and/or FSG is not used by any portion of the McBSP. Depending on your system you may also want to reset the receiver ($\overline{\text{RRST}}$ = 0 in SPCR1) and reset the transmitter ($\overline{\text{XRST}}$ = 0 in SPCR2).

If $\overline{\text{GRST}}$ = 0 due to an OMAP5912 reset, CLKG is driven by the CPU clock divided by 2, and FSG is driven inactive-low. If $\overline{\text{GRST}}$ = 0 due to program code, CLKG and FSG are driven low (inactive).

Step 2: Program the registers that affect the sample rate generator.

Program the sample rate generator registers (SRGR1 and SRGR2) 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 sample rate generator registers are programmed, wait two CLKSRG cycles. This ensures proper synchronization internally.

Step 3: Enable the sample rate generator (take it out of reset).

In SPCR2, make $\overline{\text{GRST}}$ = 1 to enable the sample rate generator.

After the sample rate generator is enabled, wait two CLKG cycles for the sample rate generator logic to stabilize.

On the next rising edge of CLKSRG, CLKG transitions to 1 and starts clocking with a frequency equal to

$$\text{CLKG frequency} = \frac{\text{Input clock frequency}}{(\text{CLKGDV} + 1)}$$

where the input clock is selected with the SCLKME bit of PCR and the CLKSM bit of SRGR2 in one of the configurations in the following table.

Table 6. Input Clock Selection for Sample Rate Generator

SCLKME	CLKSM	Input Clock for Sample Rate Generator
0	0	Signal on CLKS pin
0	1	CPU clock
1	0	Signal on CLKR pin
1	1	Signal on CLKX pin

Step 4: If necessary, enable the receiver and/or the transmitter.

If necessary, remove the receiver and/or transmitter from reset by setting $\overline{\text{RRST}}$ and/or $\overline{\text{XRST}}$ = 1.

Step 5: If necessary, enable the frame-synchronization logic of the sample rate generator.

After the required data acquisition setup is done (DXR[1/2] is loaded with data), set $\overline{\text{GRST}}$ = 1 in SPCR2 if an internally generated frame-synchronization pulse is required. FSG is generated with an active-high edge after the programmed number of CLKG clocks (FPER + 1) have elapsed.

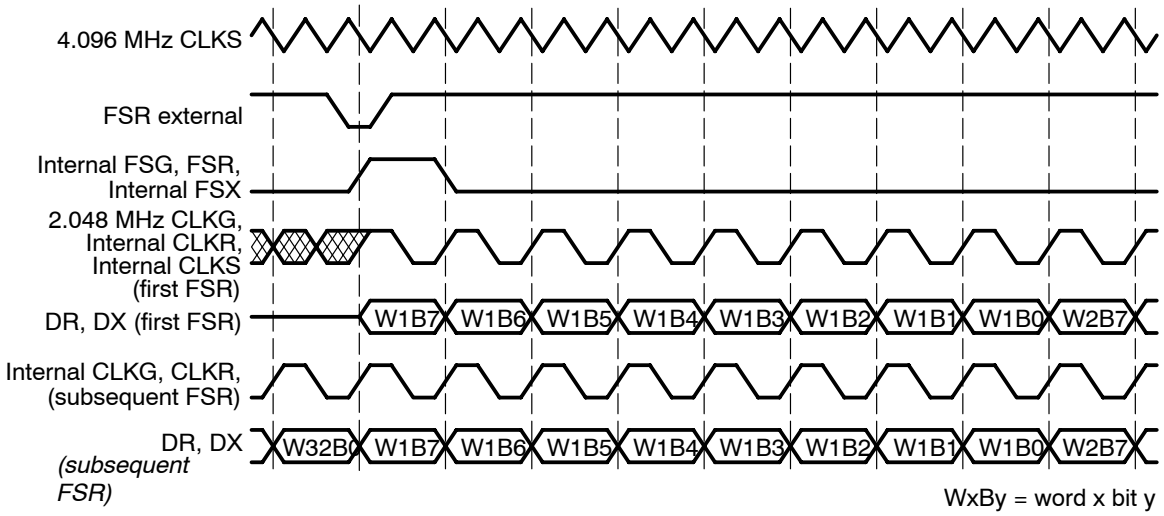
3.5 Sample Rate Generator Clocking Examples

This section shows three examples of using the sample rate generator to clock data during transmission and reception.

3.5.1 Double-Rate ST-Bus Clock

Figure 20 shows McBSP configuration to be compatible with the Mitel ST-Bus. This operation is running at maximum frame frequency.

Figure 20. ST-Bus and MVIP Clocking Example



For this McBSP configuration:

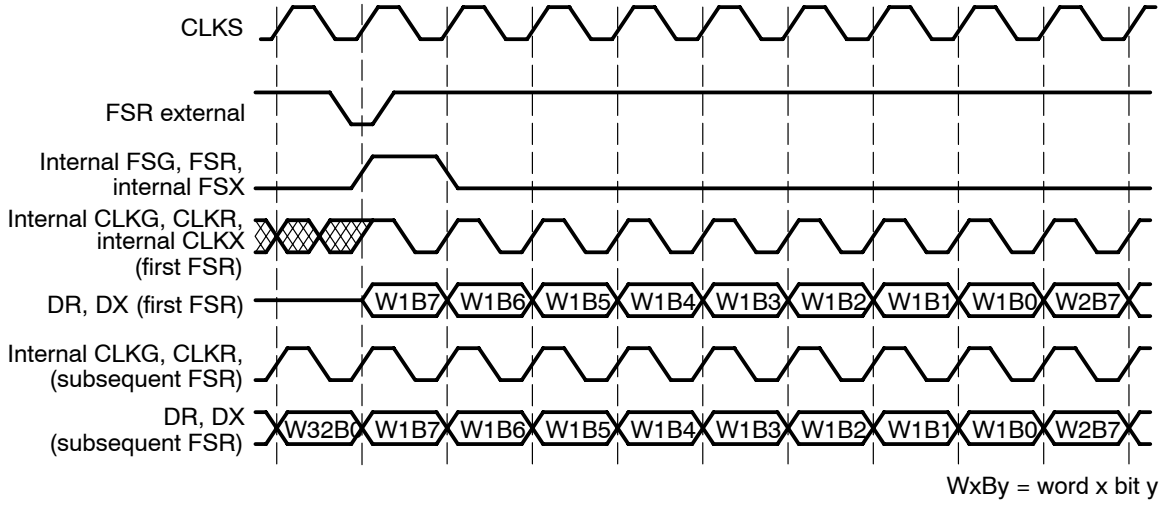
- DLB = 0: Digital loopback mode off, CLKSTP = 00b: Clock stop mode off, and CLKRM/CLKXM = 1: Internal CLKR/CLKX generated internally by sample rate generator.
- GSYNC = 1: Synchronize CLKG with external frame-synchronization signal input on FSR pin. CLKG is not synchronized until the frame-synchronization signal is active. FSR is regenerated internally to form a minimum pulse width.
- SCLKME = 0 and CLKSM = 1: External clock signal at CLKS pin drives the sample rate generator.
- CLKSP = 1: Falling edge of CLKS generates CLKG and thus internal CLK(R/X).
- CLKGDV = 1: Frequency of receive clock (shown as CLKR) is half CLKS frequency.
- FSRP/FSXP = 1: Active-low frame-synchronization pulse.
- RFRLEN1/XFRLEN1 = 11111b: 32 words per frame.
- RWDLEN1/XWDLEN1 = 0: 8 bits per word.
- RPHASE/XPHASE = 0: Single-phase frame and thus (R/X)FRLEN2 and (R/X)WDLEN2 are ignored.
- RDATDLY/XDATDLY = 0: No data delay.

3.5.2 Single-Rate ST-Bus Clock

The example in Figure 21 is the same as the double-rate ST-bus clock example in section 3.5.1 except that:

- CLKGDV = 0: CLKS drives internal CLK(R/X) without any divide down (single-rate clock).
- CLKSP = 0: Rising edge of CLKS generates CLKG and internal CLK(R/X).

Figure 21. Single-Rate Clock Example



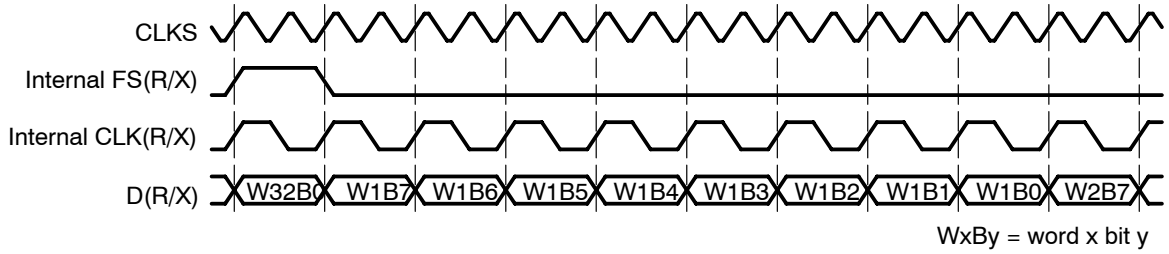
The rising edge of CLKS is used to detect the external FSR pulse, which is used to resynchronize internal McBSP clocks and generate a frame-synchronization pulse for internal use. The internal frame-synchronization pulse is generated so that it is wide enough to be detected on the falling edge of internal clocks.

3.5.3 Other Double-Rate Clock

The example in Figure 22 is the same as the double-rate ST-bus clock example in section 3.5.1 except that:

- CLKSP = 0: Rising edge of CLKS generates CLKG and thus CLK(R/X).
- CLKGDV = 1: Frequency of CLKG (and thus internal CLKR and internal CLKX) is half CLKS frequency.
- FSRM/FSXM = 0: Frame synchronization is externally generated. The frame-synchronization pulse is wide enough to be detected.
- GSYNC = 0: CLKS drives CLKG. CLKG runs freely; it is not resynchronized by a pulse on the FSR pin.
- FSRP/FSXP = 0: Active-high input frame-synchronization signal.
- RDATDLY/XDATDLY = 1: Data delay of one bit.

Figure 22. Double-Rate Clock Example



4 McBSP Exception/Error Conditions

There are five serial port events that can constitute a system error:

- Receiver overrun (RFULL = 1)

  This occurs when DRR1 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 DRR(s) and the RSR(s) are now full with another new word shifted in from DR. Therefore, RFULL = 1 indicates an error condition wherein any new data that can arrive at this time on 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 DR and DRR1 is not read. For more details about overrun in the receiver, see section 4.1.

- Unexpected receive frame-synchronization pulse (RSYNCERR = 1)

  This occurs during reception when RFIG = 0 and an unexpected frame-synchronization 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 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 has been restarted. For more details about receive frame-synchronization errors, see section 4.2.

- Transmitter data overwrite

  This occurs when the CPU or DMA controller overwrites data in the DXR(s) before the data is copied to the XSR(s). The overwritten data never reaches the DX pin. For more details about overwrite in the transmitter, see section 4.3.

- Transmitter underflow ($\overline{\text{XEMPTY}}$ = 0)

  If a new frame-synchronization signal arrives before new data is loaded into DXR1, the previous data in the DXR(s) is sent again. This procedure continues for every new frame-synchronization pulse that arrives until DXR1 is loaded with new data. For more details about underflow in the transmitter, see section 4.4.

- Unexpected transmit frame-synchronization pulse (XSYNCERR = 1)

  This occurs during transmission when XFIG = 0 and an unexpected frame-synchronization 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 DXR(s) since the last DXR-to-XSR copy, the current value in the XSR(s) is lost. For more details about transmit frame-synchronization errors, see section 4.5.

4.1 Overrun in the Receiver

RFULL = 1 in SPCR1 indicates that the receiver has experienced overrun and is in an error condition. RFULL is set when all of the following conditions are met:

1) DRR1 has not been read since the last RBR-to-DRR copy (RRDY = 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 described in the section 2.5, *McBSP Reception*, data arriving on 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 DRR1. The RRDY bit is set when new data arrives in DRR1 and is cleared when that data is read from DRR1. 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 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 DRR1 is read no later than 2.5 cycles before the end of the third word is shifted into the RSR1.

Note:

If both DRRs are needed (word length larger than 16 bits), the CPU or the DMA controller must read from DRR2 first and then from DRR1. As soon as DRR1 is read, the next RBR-to-DRR copy is initiated. If DRR2 is not read first, the data in DRR2 is lost.

After the receiver starts running from reset, a minimum of three words must be received before RFULL is set. Either of the following events clears the RFULL bit and allows subsequent transfers to be read properly:

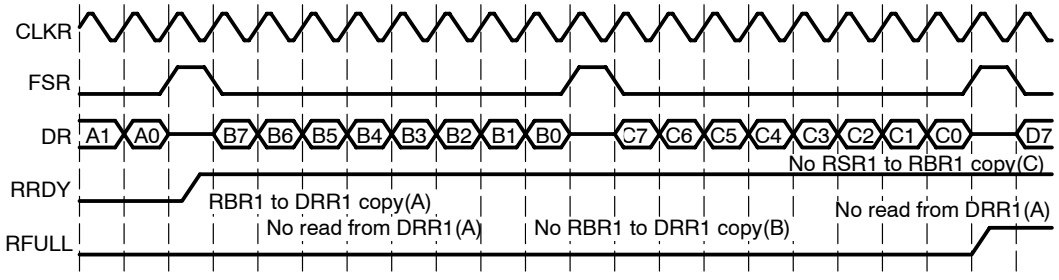
- The CPU or DMA controller reads DRR1.
- The receiver is reset individually ($\overline{\text{RRST}}$ = 0) or as part of an OMAP5912 reset.

Another frame-synchronization pulse is required to restart the receiver.

4.1.1 Example of Overrun Condition

Figure 23 shows the receive overrun condition. Because serial word A is not read from DRR1 before serial word B arrives in RBR1, B is not transferred to DRR1 yet. Another new word (C) arrives and RSR1 is full with this data. DRR1 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 DRR1 is not read in time, the next word can overwrite D.

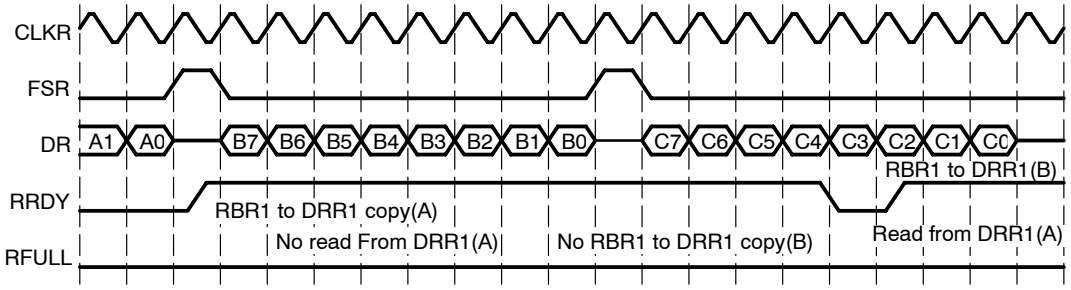
Figure 23. Overrun in the McBSP Receiver



4.1.2 Example of Preventing Overrun Condition

Figure 24 shows the case where RFULL is set, but the overrun condition is prevented by a read from DRR1 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 receiver attempts to transfer word C from RSR1 to RBR1.

Figure 24. Overrun Prevented in the McBSP Receiver



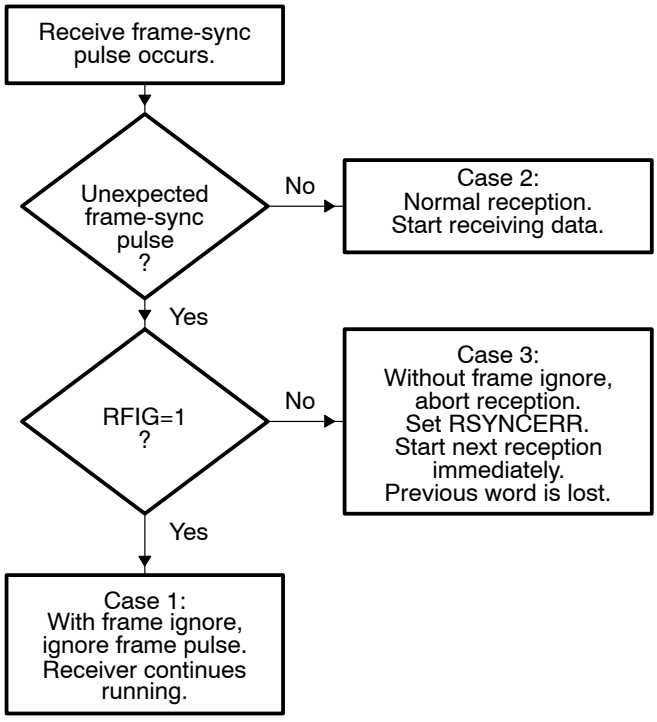
4.2 Unexpected Receive Frame-Synchronization Pulse

Section 4.2.1 shows how the McBSP responds to any receive frame-synchronization pulses, including an unexpected pulse. Sections 4.2.2 and 4.2.3 show an example of a frame-synchronization error and an example of how to prevent such an error, respectively.

4.2.1 Possible Responses to Receive Frame-Synchronization Pulses

Figure 25 shows the decision tree that the receiver uses to handle all incoming frame-synchronization pulses. The figure assumes that the receiver has been started ($\overline{\text{RRST}}$ = 1 in SPCR1). Case 3 in the figure is the case in which an error occurs.

Figure 25. Possible Responses to Receive Frame-Synchronization Pulses



Any one of three cases can occur:

- Case 1: Unexpected internal FSR pulses with RFIG = 1 in RCR2. Receive frame-synchronization pulses are ignored, and the reception continues.
- Case 2: Normal serial port reception. Reception continues normally because the frame-synchronization pulse is not unexpected. There are three possible reasons why a receive operation might *not* be in progress when the pulse occurs:
  - The FSR pulse is the first after the receiver is enabled ($\overline{\text{RRST}}$ = 1 in SPCR1).
  - The FSR pulse is the first after DRR[1,2] is read, clearing a receiver full (RFULL = 1 in SPCR1) condition.
  - The serial port is in the interpacket intervals. The programmed data delay for reception (programmed with the RDATDLY bits in RCR2) may start during these interpacket intervals for the first bit of the next word to be received. Thus, at maximum frame frequency, frame synchronization can still be received 0 to 2 clock cycles before the first bit of the synchronized frame.

- Case 3: Unexpected receive frame synchronization with RFIG = 0, frame-synchronization pulses are not ignored. Unexpected frame-synchronization pulses can originate from an external source or from the internal sample rate generator.

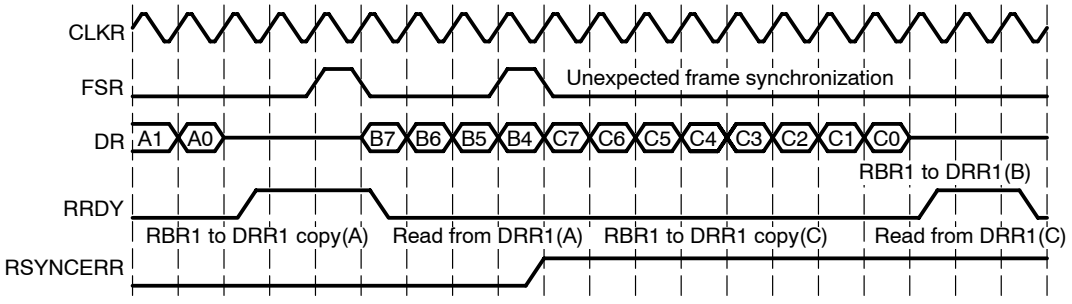
  If a frame-synchronization pulse starts the transfer of a new frame before the current frame is fully received, this pulse is treated as an unexpected frame-synchronization pulse, and the receiver sets the receive frame-synchronization error bit (RSYNCERR) in SPCR1. RSYNCERR can be cleared only by a receiver reset or by a write of 0 to this bit.

  If you want the McBSP to notify the CPU of receive frame-synchronization errors, you can set a special receive interrupt mode with the RINTM bits of SPCR1. When RINTM = 11b, the McBSP sends a receive interrupt (RINT) request to the CPU each time that RSYNCERR is set.

4.2.2 Example of Unexpected Receive Frame-Synchronization Pulse

Figure 26 shows an unexpected receive frame-synchronization pulse during normal operation of the serial port, with time intervals between data packets. When the unexpected frame-synchronization 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.

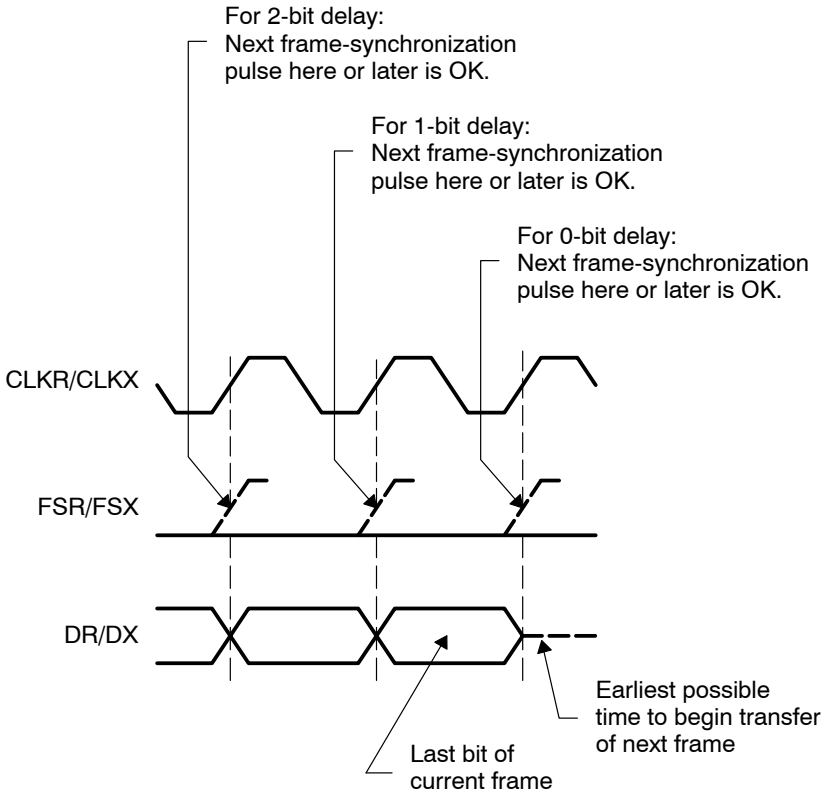
Figure 26. An Unexpected Frame-Synchronization Pulse During a McBSP Reception



4.2.3 Preventing Unexpected Receive Frame-Synchronization Pulses

Each frame transfer can be delayed by 0, 1, or 2 CLKR cycles, depending on the value in the RDATDLY bits of RCR2. For each possible data delay, Figure 27 shows when a new frame-synchronization pulse on FSR can safely occur relative to the last bit of the current frame.

Figure 27. Proper Positioning of Frame-Synchronization Pulses



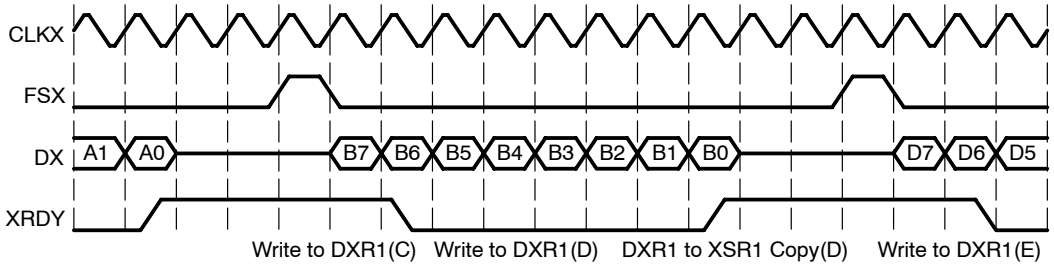
4.3 Overwrite in the Transmitter

As described in section 2.6, the transmitter must copy the data previously written to the DXR(s) by the CPU or DMA controller into the XSR(s) and then shift each bit from the XSR(s) to the DX pin. If new data is written to the DXR(s) before the previous data is copied to the XSR(s), the previous data in the DXR(s) is overwritten and thus lost.

4.3.1 Example of Overwrite Condition

Figure 28 shows what happens if the data in DXR1 is overwritten before being transmitted. Initially, DXR1 is loaded with data C. A subsequent write to DXR1 overwrites C with D before C is copied to XSR1. Thus, C is never transmitted on DX.

Figure 28. Data in the McBSP Transmitter Overwritten and Thus Not Transmitted



4.3.2 Preventing Overwrites

You can prevent CPU overwrites by making the CPU:

- Poll for XRDY = 1 in SPCR2 before writing to the DXR(s). XRDY is set when data is copied from DXR1 to XSR1 and is cleared when new data is written to DXR1.
- Wait for a transmit interrupt (XINT) before writing to the DXR(s). When XINTM = 00b in SPCR2, 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.

4.4 Underflow in the Transmitter

The McBSP indicates a transmitter empty (or underflow) condition by clearing the $\overline{\text{XEMPTY}}$ bit in SPCR2. Either of the following events activates $\overline{\text{XEMPTY}}$ ($\overline{\text{XEMPTY}}$ = 0):

- DXR1 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 DX pin.
- The transmitter is reset (by forcing $\overline{\text{XRST}}$ = 0 in SPCR2, or by an OMAP5912 reset) and is then restarted.

In the underflow condition, the transmitter continues to transmit the old data that is in the DXR(s) for every new transmit frame-synchronization signal until a new value is loaded into DXR1 by the CPU or the DMA controller.

Note:

If both DXRs are needed (word length larger than 16 bits), the CPU or the DMA controller must load DXR2 first and then load DXR1. As soon as DXR1 is loaded, the contents of both DXRs are copied to the transmit shift registers (XSRs). If DXR2 is not loaded first, the previous content of DXR2 is passed to the XSR2.

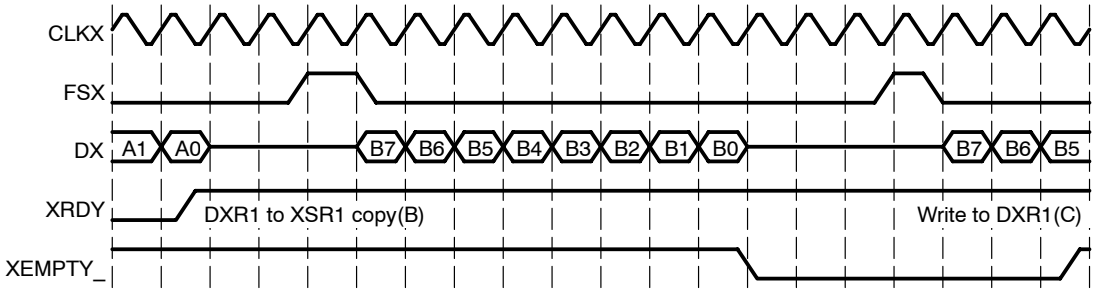
$\overline{\text{XEMPTY}}$ is deactivated ($\overline{\text{XEMPTY}}$ = 1) when a new word in DXR1 is transferred to XSR1. If FSXM = 1 in PCR and FSGM = 0 in SRGR2, the transmitter generates a single internal FSX pulse in response to a DXR-to-XSR copy. Otherwise, the transmitter waits for the next frame-synchronization pulse before sending out the next frame on DX.

When the transmitter is taken out of reset ($\overline{\text{XRST}}$ = 1), it is in a transmitter ready (XRDY = 1 in SPCR2) and transmitter empty ($\overline{\text{XEMPTY}}$ = 0) state. If DXR1 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 transmit frame-synchronization pulse is generated or detected. Alternatively, if a transmit frame-synchronization pulse is detected before DXR1 is loaded, zeros are output on DX.

4.4.1 Example of the Underflow Condition

Figure 29 shows an underflow condition. After B is transmitted, DXR1 is not reloaded before the subsequent frame-synchronization pulse. Thus, B is again transmitted on DX.

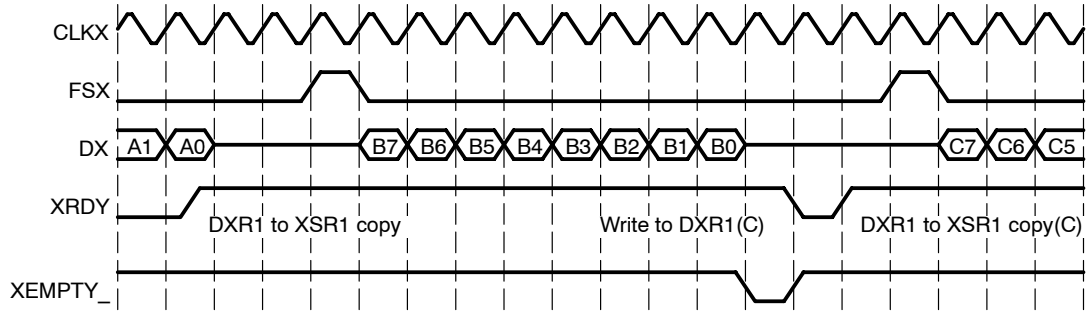
Figure 29. Underflow During McBSP Transmission



4.4.2 Example of Preventing Underflow Condition

Figure 30 shows the case of writing to DXR1 just before an underflow condition would otherwise occur. After B is transmitted, C is written to DXR1 before the next frame-synchronization pulse. As a result, there is no underflow; B is not transmitted twice.

Figure 30. Underflow Prevented in the McBSP Transmitter



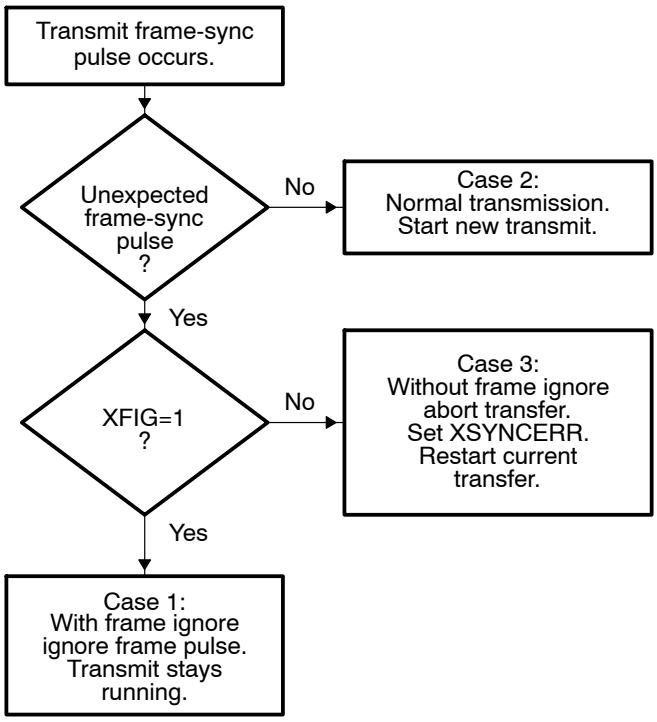
4.5 Unexpected Transmit Frame-Synchronization Pulse

section 4.5.1 shows how the McBSP responds to any transmit frame-synchronization pulses, including an unexpected pulse. Sections 4.5.2 and 4.5.3 show an example of a frame-synchronization error and an example of how to prevent such an error, respectively.

4.5.1 Possible Responses to Transmit Frame-Synchronization Pulses

Figure 31 shows the decision tree that the transmitter uses to handle all incoming frame-synchronization pulses. The figure assumes that the transmitter has been started ($\overline{\text{XRST}}$ = 1 in SPCR2). Case 3 in the figure is the case in which an error occurs.

Figure 31. Possible Responses to Transmit Frame-Synchronization Pulses



Any one of three cases can occur:

- Case 1: Unexpected internal FSX pulses with XFIG = 1 in XCR2. Transmit frame-synchronization pulses are ignored, and the transmission continues.

- Case 2: Normal serial port transmission. Transmission continues normally because the frame-synchronization pulse is not unexpected. There are two possible reasons why a transmit operations might *not* be in progress when the pulse occurs:

  This FSX pulse is the first after the transmitter is enabled ($\overline{\text{XRST}}$ = 1).

  The serial port is in the interpacket intervals. The programmed data delay for transmission (programmed with the XDATDLY bits of XCR2) may start during these interpacket intervals before the first bit of the previous word is transmitted. Thus, at maximum packet frequency, frame synchronization can still be received 0 to 2 clock cycles before the first bit of the synchronized frame.

- Case 3: Unexpected transmit frame synchronization with XFIG = 0 (frame-synchronization pulses not ignored). Unexpected frame-synchronization pulses can originate from an external source or from the internal sample rate generator.

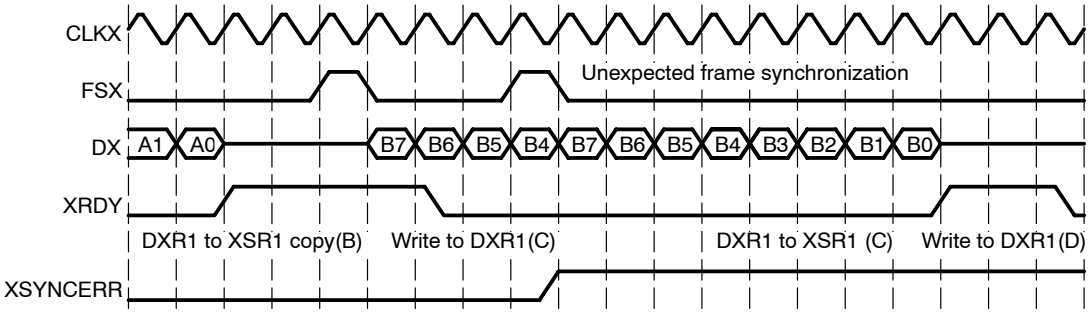
  If a frame-synchronization pulse starts the transfer of a new frame before the current frame is fully transmitted, this pulse is treated as an unexpected frame-synchronization pulse, and the transmitter sets the transmit frame-synchronization error bit (XSYNCERR) in SPCR2. XSYNCERR can be cleared only by a transmitter reset or by a write of 0 to this bit.

  If you want the McBSP to notify the CPU of frame-synchronization errors, you can set a special transmit interrupt mode with the XINTM bits of SPCR2. When XINTM = 11b, the McBSP sends a transmit interrupt (XINT) request to the CPU each time that XSYNCERR is set.

4.5.2 Example of Unexpected Transmit Frame-Synchronization Pulse

Figure 32 shows an unexpected transmit frame-synchronization pulse during normal operation of the serial port with intervals between the data packets. When the unexpected frame-synchronization 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.

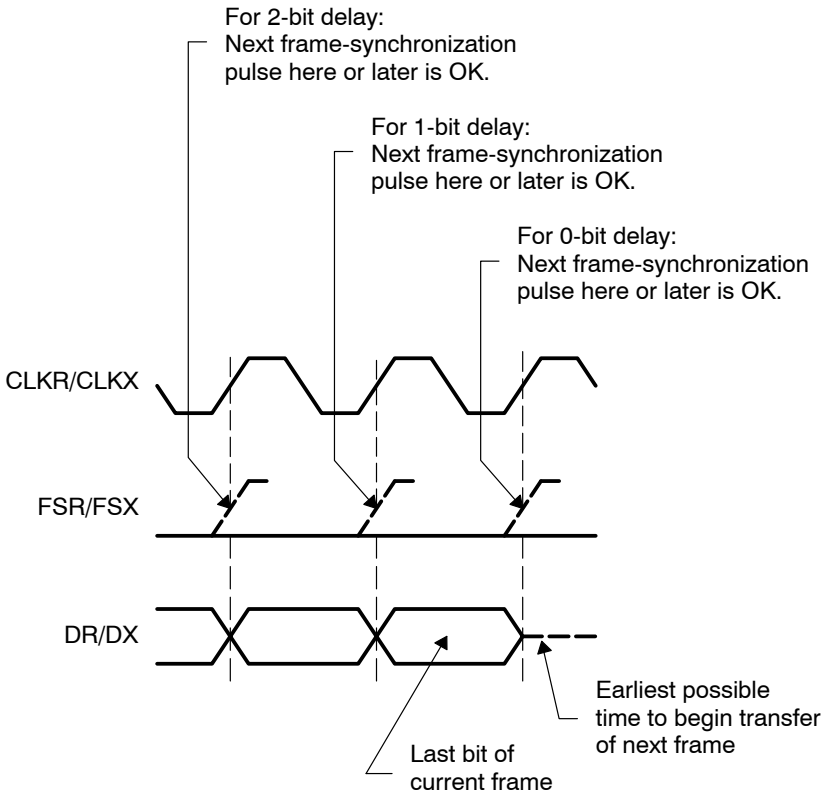
Figure 32. An Unexpected Frame-Synchronization Pulse During a McBSP Transmission



4.5.3 Preventing Unexpected Transmit Frame-Synchronization Pulses

Each frame transfer can be delayed by 0, 1, or 2 CLKX cycles, depending on the value in the XDATDLY bits of XCR2. For each possible data delay, Figure 33 shows when a new frame-synchronization pulse on FSX can safely occur relative to the last bit of the current frame.

Figure 33. Proper Positioning of Frame-Synchronization Pulses



5 Multichannel Selection Modes

This section discusses the multichannel selection modes for the McBSP.

5.1 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 4: Channels 64–79
Block 1: Channels 16–31	Block 5: Channels 80–95
Block 2: Channels 32–47	Block 6: Channels 96–111
Block 3: Channels 48–63	Block 7: Channels 112–127

The blocks are assigned to partitions according to the selected partition mode. In the two-partition mode (described in section 5.4), 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 8-partition mode (described in section 5.5), 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).

5.2 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 (described in section 5.6) and three transmit multichannel selection modes (described in section 5.7).

5.3 Configuring a Frame for Multichannel Selection

Before you enable a multichannel selection mode, make sure you properly configure the data frame:

- Select a single-phase frame (RPHASE/XPHASE = 0). Each frame represents a TDM data stream.
- Set a frame length (in RFRLEN1/XFRLEN1) 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.

5.4 Using Two Partitions

For multichannel selection operation in the receiver and/or the transmitter, you can use two partitions or eight partitions (described in section 5.5). If you choose the 2-partition mode (RMCME = 0 for reception, XMCME = 0 for transmission), McBSP channels are activated using an alternating scheme. In response to a frame-synchronization pulse, the receiver or transmitter begins with the channels in partition A and then alternates between partitions B and A until the complete frame has been transferred. When the next frame-synchronization pulse occurs, the next frame is transferred beginning with the channels in partition A.

5.4.1 Assigning Blocks to Partitions A and B

For reception, any two of the eight receive-channel blocks can be assigned to receive partitions A and B, which means up to 32 receive channels can be enabled at any given point in time. Similarly, any two of the eight transmit-channel blocks (up 32 enabled transmit channels) can be assigned to transmit partitions A and B.

For reception:

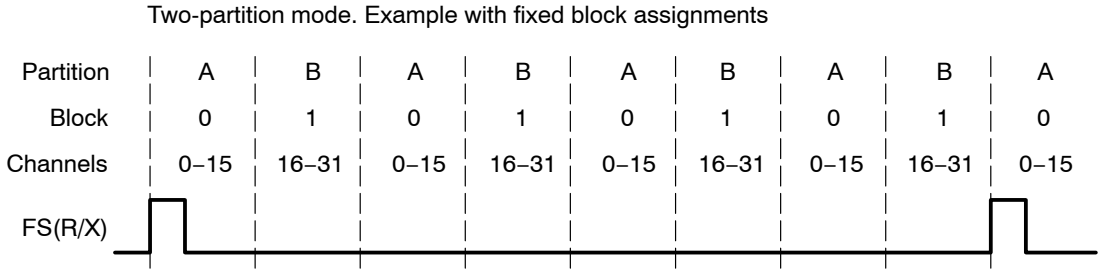
- Assign an even-numbered channel block (0, 2, 4, or 6) to receive partition A by writing to the RPABLK bits. In the receive multichannel selection mode (described in section 5.6), the channels in this partition are controlled by receive channel enable register A (RCERA).
- Assign an odd-numbered block (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 (RCERB).

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 5.7), the channels in this partition are controlled by transmit channel enable register A (XCERA).
- 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 (XCERB).

Figure 34 shows an example of alternating between the channels of partition A and the channels of partition B. Channels 0-15 have been assigned to partition A, and channels 16-31 have been assigned to partition B. In response to a frame-synchronization pulse, the McBSP begins a frame transfer with partition A and then alternates between partitions B and A until the complete frame is transferred.

Figure 34. Alternating Between the Channels of Partition A and the Channels of Partition B



As explained in section 5.4.2, you can dynamically change which blocks of channels are assigned to the partitions.

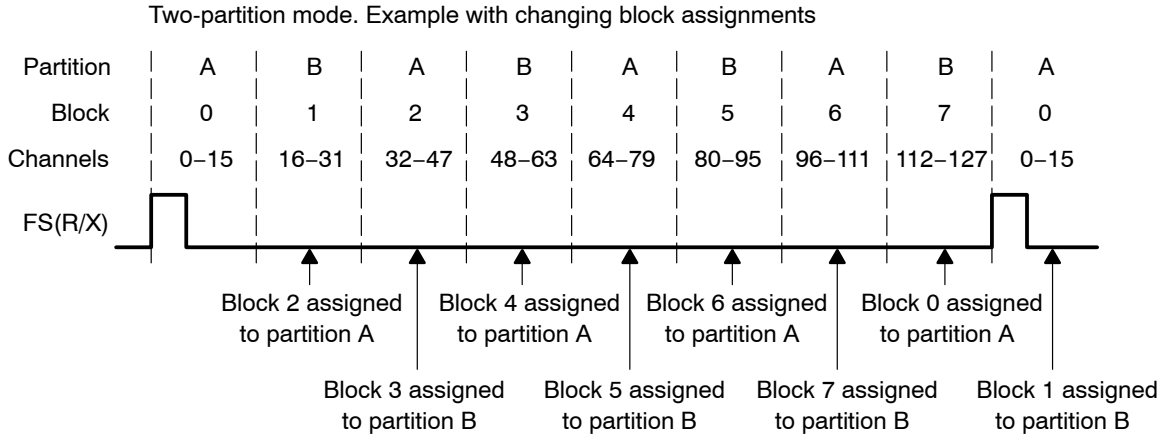
5.4.2 Reassigning Blocks During Reception/Transmission

If you want to use more than 32 channels, you can change which channel blocks are assigned to partitions A and B during the course of a data transfer. However, these changes must be carefully timed. While a partition is being transferred, its associated block assignment bits cannot be modified and its associated channel enable register cannot be modified. For example, if block 3 is being transferred and block 3 is assigned to partition A, you 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. See section 5.8, *Using Interrupts Between Block Transfers*.

Figure 35 shows an example of reassigning channels throughout a data transfer. In response to a frame-synchronization 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 35. Reassigning Channel Blocks Throughout a McBSP Data Transfer



5.5 Using Eight Partitions

For multichannel selection operation in the receiver and/or the transmitter, you can use eight partitions or two partitions (described in section 5.4). If you choose the 8-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-synchronization pulse, the receiver or transmitter begins with the channels in partition A and then continues with the other partitions in order until the complete frame has been transferred. When the next frame-synchronization pulse occurs, the next frame is transferred, beginning with the channels in partition A.

In the 8-partition mode, the (R/X)PABLK and (R/X)PBBLK bits are ignored and the 16-channel blocks are assigned to the partitions as shown in Table 7 and Table 8. These assignments cannot be changed. The tables also show the registers used to control the channels in the partitions.

Table 7. Receive Channel Assignment and Control With Eight Receive Partitions

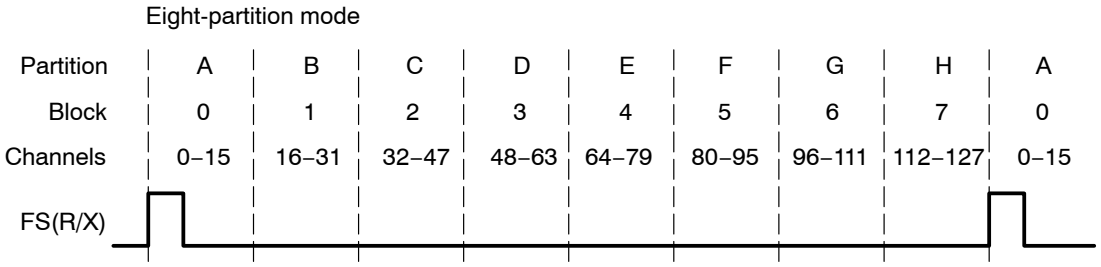
Receive Partition	Assigned Block of Receive Channels	Register Used For Channel Control
A	Block 0: channels 0 through 15	RCERA
B	Block 1: channels 16 through 31	RCERB
C	Block 2: channels 32 through 47	RCERC
D	Block 3: channels 48 through 63	RCERD
E	Block 4: channels 64 through 79	RCERE
F	Block 5: channels 80 through 95	RCERF
G	Block 6: channels 96 through 111	RCERG
H	Block 7: channels 112 through 127	RCERH

Table 8. Transmit Channel Assignment and Control When Eight Transmit Partitions Are Used

Transmit Partition	Assigned Block of Transmit Channels	Register Used For Channel Control
A	Block 0: channels 0 through 15	XCERA
B	Block 1: channels 16 through 31	XCERB
C	Block 2: channels 32 through 47	XCERC
D	Block 3: channels 48 through 63	XCERD
E	Block 4: channels 64 through 79	XCERE
F	Block 5: channels 80 through 95	XCERF
G	Block 6: channels 96 through 111	XCERG
H	Block 7: channels 112 through 127	XCERH

Figure 36 shows an example of the McBSP using the 8-partition mode. In response to a frame-synchronization 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 36. McBSP Data Transfer in the 8-Partition Mode



5.6 Receive Multichannel Selection Mode

The RMCM bit of MCR1 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 (RCERs). The way channels are assigned to the RCERs depends on the number of receive channel partitions (2 or 8), as defined by the RMCME bit of MCR1.
- 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 DRR(s), 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 DR pin in channel 0.
2) Ignores bits received in channels 1-14.
3) Accepts bits shifted in from the DR pin in channel 15.
4) Ignores bits received in channels 16-38.
5) Accepts bits shifted in from the DR pin in channel 39.

5.7 Transmit Multichannel Selection Modes

The XMCM bits of XCR2 determine whether all channels or only selected channels are enabled and unmasked for transmission. More details on enabling and masking are in section 5.7.1. The McBSP has three transmit multichannel selection modes described in the following table: XMCM = 01b, XMCM = 10b, and XMCM = 11b.

Table 9. Selecting a Transmit Multichannel Selection Mode With the XMCM Bits

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 (XCERs). If enabled, a channel in this mode is also unmasked. The XMCME bit of MCR2 determines whether 32 channels or 128 channels are selectable in XCERs.
10b	All channels are enabled, but they are masked unless they are selected in the appropriate transmit channel enable registers (XCERs). The XMCME bit of MCR2 determines whether 32 channels or 128 channels are selectable in XCERs.
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 (RCERs). Once enabled, they are masked, unless they are also selected in the appropriate transmit channel enable registers (XCERs). The XMCME bit of MCR2 determines whether 32 channels or 128 channels are selectable in RCERs and XCERs.

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 DX pin in channel 0.
2) Places the DX pin in the high impedance state in channels 1-14.
3) Shifts data to the DX pin in channel 15.
4) Places the DX pin in the high impedance state in channels 16-38.
5) Shifts data to the DX pin in channel 39.

5.7.1 Disabling/Enabling Versus Masking/Unmasking

For transmission, a channel can be:

- ❑ Enabled and unmasked (transmission can begin and can be completed).
- ❑ Enabled but masked (transmission can begin but cannot be completed).
- ❑ Disabled (transmission cannot occur).

The following definitions explain the channel control options:

Enabled channel — A channel that can begin transmission by passing data from the data transmit register(s) (DXR(s)) to the transmit shift registers (XSR(s)).

Masked channel — A channel that cannot complete transmission. The DX pin is held in the high impedance state; data cannot be shifted out on the DX pin.

In systems where symmetric transmit and receive provides software benefits, this feature allows transmit channels to be disabled on a shared serial bus. A similar feature is not needed for reception because multiple receptions cannot cause serial bus contention.

Disabled channel — A channel that is not enabled. A disabled channel is also masked.

Because no DXR-to-XSR copy occurs, the XRDY bit of SPCR2 is not set. Therefore, no DMA synchronization event (XEVT) is generated, and if the transmit interrupt mode depends on XRDY (XINTM = 00b in SPCR2), no interrupt is generated.

The $\overline{\text{XEMPTY}}$ bit of SPCR2 is not affected.

Unmasked channel — A channel that is not masked. Data in the XSR(s) is shifted out on the DX pin.

5.7.2 Activity on McBSP Pins for Different Values of XMCM

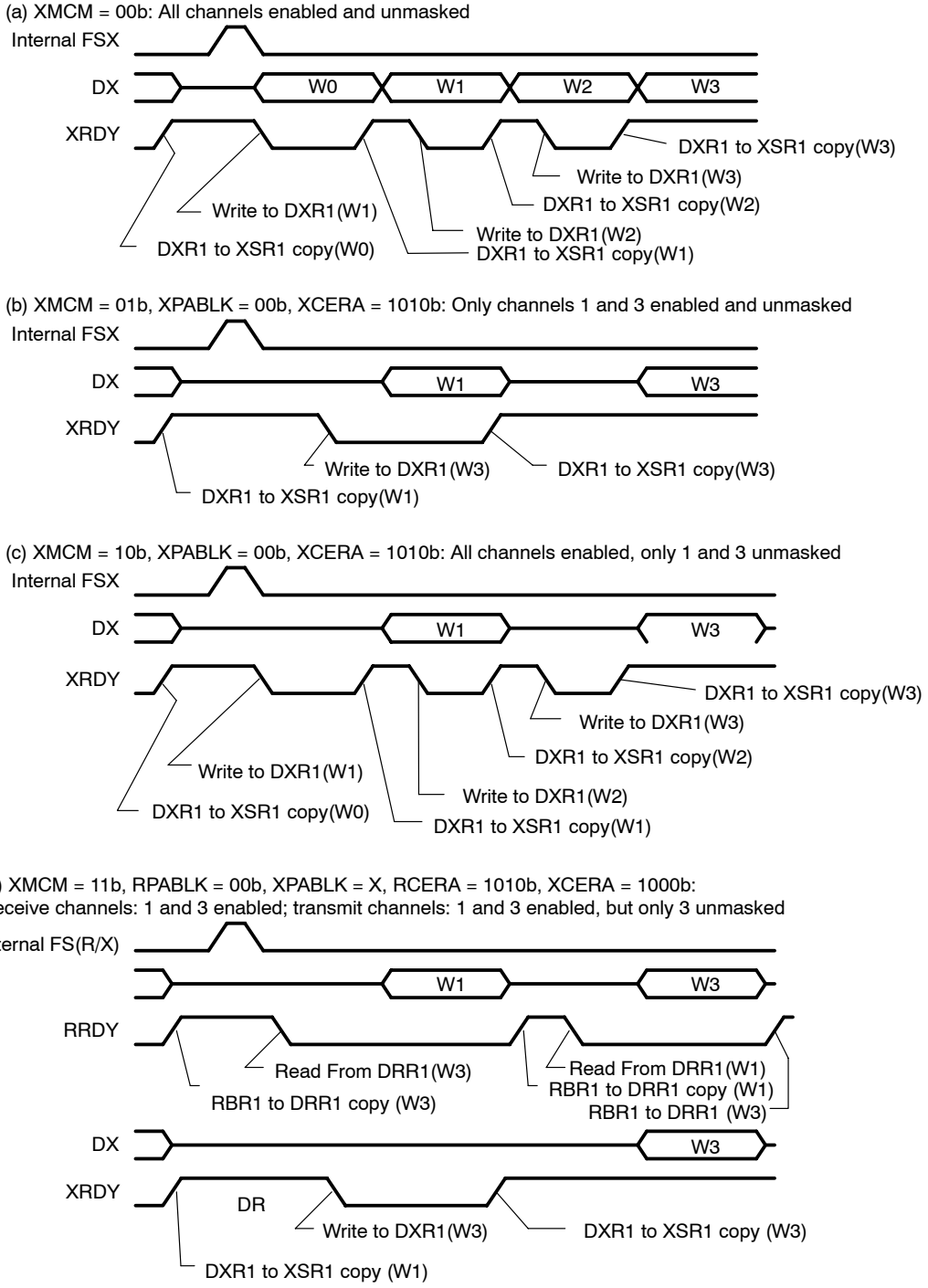
Figure 37 shows the activity on the McBSP pins for the various XMCM values. In all cases, the transmit frame is configured as follows:

- XPHASE = 0: Single-phase frame (required for multichannel selection modes)
- XFRLEN1 = 0000011b: 4 words per frame
- XWDLEN1 = 000b: 8 bits per word
- XMCME = 0: 2-partition mode (only partitions A and B used)

When 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.

In the figure, the arrows showing where the various events occur are only sample indications. Wherever possible, there is a time window in which these events can occur.

Figure 37. Activity on McBSP Pins for the Possible Values of XMCM



5.8 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 CPU clock cycles.

This type of interrupt is especially helpful if you are using the two-partition mode (described in section 5.4) and you want to know when you can assign a different block of channels to partition A or B.

6 SPI Operation Using the Clock Stop Mode

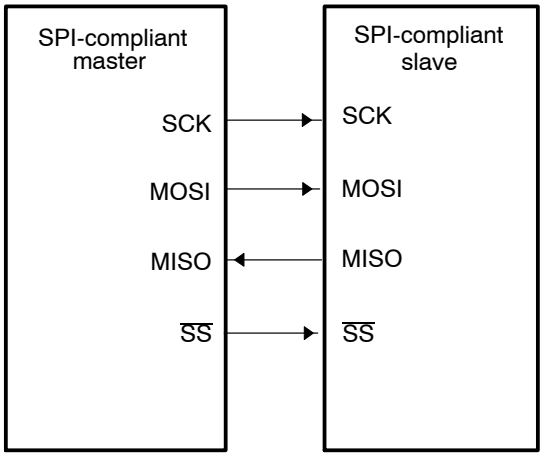
6.1 SPI Protocol

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)
- Shift-clock (also referred to as SCK)
- Slave-enable signal (also referred to as SS)

A typical SPI interface with a single slave device is shown in Figure 38.

Figure 38. 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, or the 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.

6.2 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 serial clock signal (SCK) of the SPI protocol, while the transmit frame-synchronization signal (FSX) is used as the slave-enable signal (SS_).

The receive clock signal (CLKR) and receive frame-synchronization signal (FSR) are not used in the clock stop mode because these signals are internally connected to their transmit counterparts, CLKX and FSX.

6.3 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 10. Table 11 shows 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 section 6.4 show the effects of CLKSTP, CLKXP, and CLKRP.

Table 10. Bits Used to Enable and Configure the Clock Stop Mode

Bit Field	Description
CLKSTP bits of SPCR1	Use these bits to enable the clock stop mode and to select one of two timing variations. (See also Table 11.)
CLKXP bit of PCR	This bit determines the polarity of the CLKX signal. (See also Table 11.)
CLKRP bit of PCR	This bit determines the polarity of the CLKR signal. (See also Table 11.)
CLKXM bit of PCR	This bit determines whether CLKX is an input signal (McBSP as slave) or an output signal (McBSP as master).
XPHASE bit of XCR2	You must use a single-phase transmit frame (XPHASE = 0).
RPHASE bit of RCR2	You must use a single-phase receive frame (RPHASE = 0).
XFRLEN1 bits of XCR1	You must use a transmit frame length of 1 serial word (XFRLEN1 = 0).
RFRLEN1 bits of RCR1	You must use a receive frame length of 1 serial word (RFRLEN1 = 0).
XWDLEN1 bits of XCR1	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 bits of RCR1	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 11. 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.

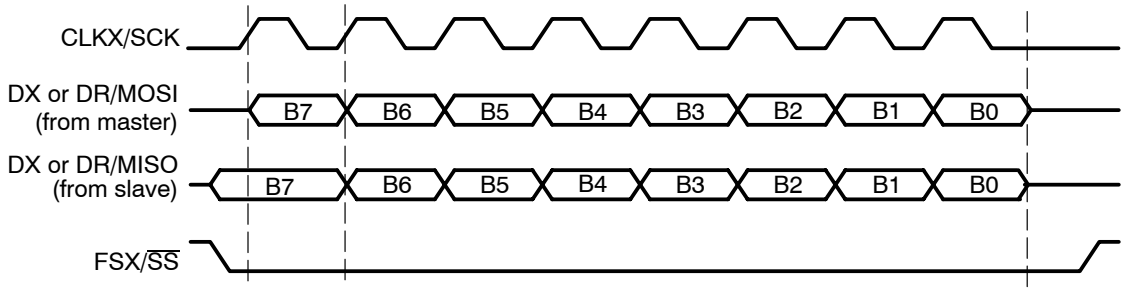
6.4 Clock Stop Mode Timing Diagrams

The timing diagrams for the four possible clock stop mode configurations are shown here. Notice that the frame-synchronization signal 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 RWDLEN1 bits of RCR1, and the transmit packet length is selected with the XWDLEN1 bits of XCR1. 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.

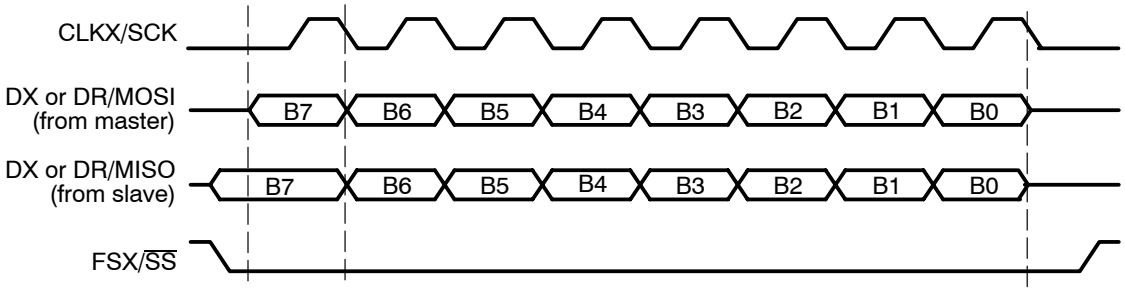
Figure 39. SPI Transfer With CLKSTP = 10b (No Clock Delay), CLKXP = 0, and CLKRP = 0



Notes: 1) If the McBSP is the SPI master (CLKXM = 1), MOSI = DX. If the McBSP is the SPI slave (CLKXM = 0), MOSI = DR.

2) If the McBSP is the SPI master (CLKXM = 1), MISO = DR. If the McBSP is the SPI slave (CLKXM = 0), MISO = DX.

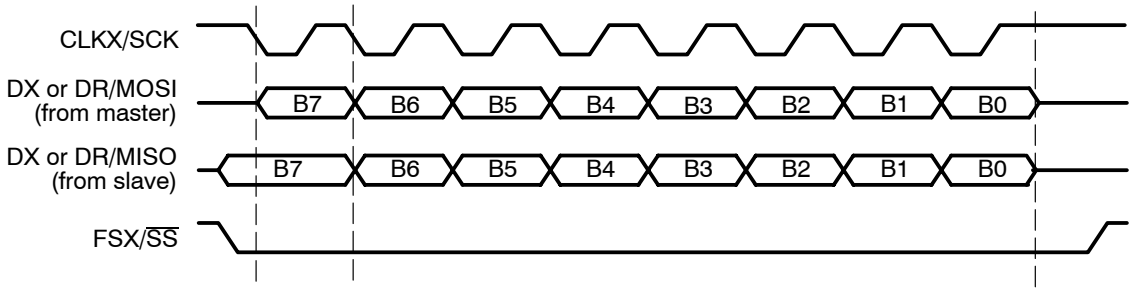
Figure 40. SPI Transfer With CLKSTP = 11b (Clock Delay), CLKXP = 0, CLKRP = 1



Notes: 1) If the McBSP is the SPI master (CLKXM = 1), MOSI = DX. If the McBSP is the SPI slave (CLKXM = 0), MOSI = DR.

2) If the McBSP is the SPI master (CLKXM = 1), MISO = DR. If the McBSP is the SPI slave (CLKXM = 0), MISO = DX.

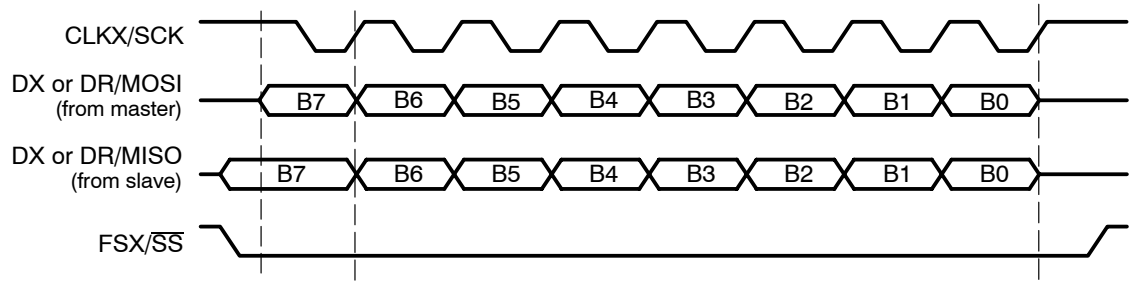
Figure 41. SPI Transfer With CLKSTP = 10b (No Clock Delay), CLKXP = 1, and CLKRP = 0



Notes: 1) If the McBSP is the SPI master (CLKXM = 1), MOSI = DX. If the McBSP is the SPI slave (CLKXM = 0), MOSI = DR.

2) If the McBSP is the SPI master (CLKXM = 1), MISO = DR. If the McBSP is the SPI slave (CLKXM = 0), MISO = DX.

Figure 42. SPI Transfer With CLKSTP = 11b (Clock Delay), CLKXP = 1, CLKRP = 1



Notes: 1) If the McBSP is the SPI master (CLKXM = 1), MOSI=DX. If the McBSP is the SPI slave (CLKXM = 0), MOSI = DR.

2) If the McBSP is the SPI master (CLKXM = 1), MISO=DR. If the McBSP is the SPI slave (CLKXM = 0), MISO = DX.

6.5 Procedure for Configuring a 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 ($\overline{\text{XRST}}$ = 0) in SPCR2 to reset the transmitter. Clear the receiver reset bit ($\overline{\text{RRST}}$ = 0) in SPCR1 to reset the receiver.

Step 2: Place the sample rate generator in reset.

Clear the sample rate generator reset bit ($\overline{\text{GRST}}$ = 0) in SPCR2 to reset the sample rate generator.

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 6.6)
- *McBSP as an SPI Slave* (section 6.7)

Step 4: Enable the sample rate generator.

To release the sample rate generator from reset, set the sample rate generator reset bit ($\overline{\text{GRST}}$ = 1) in SPCR2.

Make sure that during the write to SPCR2, you only modify $\overline{\text{GRST}}$. Otherwise, you modify the McBSP configuration you selected in the previous step.

Step 5: Enable the transmitter and receiver.

After the sample rate generator is released from reset, wait two sample rate generator 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 ($\overline{\text{XRST}}$ = 1 in SPCR2) and enable the receiver ($\overline{\text{RRST}}$ = 1 in SPCR1).

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 $\overline{\text{XRST}}$ = 1 and $\overline{\text{RRST}}$ = 1.

In either case, make sure you only change $\overline{\text{XRST}}$ and $\overline{\text{RRST}}$ when you write to SPCR2 and SPCR1. Otherwise, you modify the bit settings you selected earlier in this procedure.

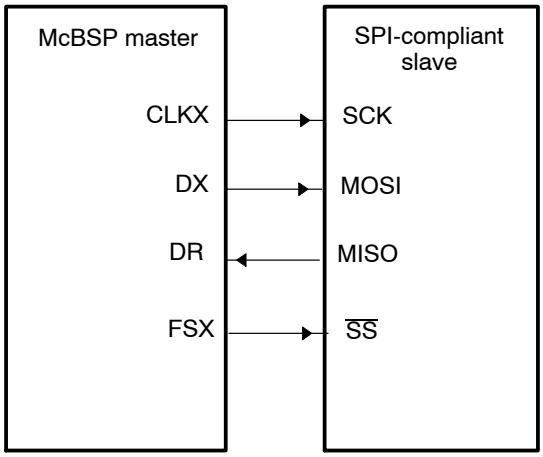
After the transmitter and receiver are released from reset, wait two sample rate generator clock periods for the McBSP logic to stabilize.

Step 6: If necessary, enable the frame-synchronization logic of the sample rate generator.

After the required data acquisition setup is done (DXR[1/2] is loaded with data), set $\overline{\text{FRST}}$ = 1 if an internally generated frame-synchronization pulse is required, that is, if the McBSP is the SPI master.

6.6 McBSP as the SPI Master

An SPI interface with the McBSP used as the master is shown in the following figure. When the McBSP is configured as a master, the transmit output signal (DX) is used as the MOSI signal of the SPI protocol and the receive input signal (DR) is used as the MISO signal.



The register bit values required to configure the McBSP as a master are listed in Table 12. After the table are more details about the configuration requirements.

Table 12. Bit 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 CLKX pin is positive (CLKXP = 0) or negative (CLKXP = 1).
CLKRP = 0 or 1	The polarity of CLKR as seen on the CLKR pin is positive (CLKRP = 0) or negative (CLKRP = 1).
CLKXM = 1	The CLKX pin is an output pin driven by the internal sample rate generator. Because CLKSTP is equal to 10b or 11b, CLKR is driven internally by CLKX.
SCLKME = 0 CLKSM = 1	The clock generated by the sample rate generator (CLKG) is derived from the CPU clock.
CLKGDV is a value from 0 to 255	CLKGDV defines the divide down value for CLKG.
FSXM = 1	The FSX pin is an output pin driven according to the FSGM bit.
FSGM = 0	The transmitter drives a frame-synchronization pulse on the FSX pin every time data is transferred from DXR1 to XSR1.
FSXP = 1	The 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 CLKX pin is enabled only during packet transfers. When packets are not being transferred, the CLKX pin remains high or low depending on the polarity used.

For SPI master operation, the CLKX pin must be configured as an output. The sample rate generator is then used to derive the CLKX signal from the CPU clock. The clock stop mode internally connects the CLKX pin to the CLKR signal so that no external signal connection is required on the 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.

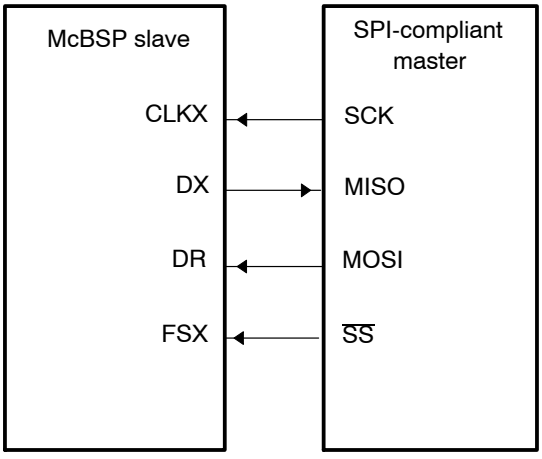
The McBSP can also provide a slave-enable signal (SS_) on the FSX pin. If a slave-enable signal is required, the FSX pin must be configured as an output and the transmitter must be configured so that a frame-synchronization pulse is generated automatically each time a packet is transmitted (FSGM = 0). The polarity of the 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-synchronization pulse width (FWID) and frame-synchronization period (FPER) are overridden, and custom frame-synchronization waveforms are not allowed. To see the resulting waveform produced on the FSX pin, see the timing diagrams in section 6.4. The signal becomes active before the first bit of a packet transfer, and remains active until the transfer of the last bit of the packet. After the packet transfer is complete, the FSX signal returns to the inactive state.

6.7 McBSP as an SPI Slave

An SPI interface with the McBSP used as a slave is shown in Figure 43. When the McBSP is configured as a slave, DX is used as the MISO signal and DR is used as the MOSI signal.

Figure 43. SPI Interface With McBSP as Slave



The register bit values required to configure the McBSP as a slave are listed in Table 13. Following the table are more details about configuration requirements.

Table 13. Bit 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 CLKX pin is positive (CLKXP = 0) or negative (CLKXP = 1).
CLKRP = 0 or 1	The polarity of CLKR as seen on the CLKR pin is positive (CLKRP = 0) or negative (CLKRP = 1).
CLKXM = 0	The 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 sample rate generator (CLKG) is derived from the CPU clock. (The sample rate generator is used to synchronize the McBSP logic with the externally-generated master clock.)
CLKGDV = 1	The sample rate generator divides the CPU clock by 2 before generating CLKG.
FSXM = 0	The FSX pin is an input pin, so that it can be driven by the SPI master.
FSXP = 1	The 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 CLKX and FSX pins must be configured as inputs. The 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 FSX pin is also internally connected to the FSR signal, and no external signal connections are required on the CLKR and FSR pins.

Although the CLKX signal is generated externally by the master and is asynchronous to the McBSP, the sample rate generator of the McBSP must be enabled for proper SPI slave operation. The sample rate generator 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.

7 Receiver Configuration

To configure the McBSP receiver, perform the following procedure:

1) Place the McBSP/receiver in reset (see section 7.2).
2) Program the McBSP registers for the desired receiver operation (see section 7.1).
3) Take the receiver out of reset (see section 7.2).

7.1 Programming the McBSP Registers for the Desired Receiver Operation

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.

It may be helpful to first photocopy the McBSP Register Worksheet (see section 14) and to fill in the photocopy of the worksheet as you read the tasks.

- ❑ 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 receive frame-synchronization 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-synchronization behavior:

- Set the receive frame-synchronization mode.
- Set the receive frame-synchronization polarity.
- Set the sample rate generator (SRG) frame-synchronization 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.

7.2 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). Table 14 describes the bits used for both of these steps.

Table 14. Register Bits Used to Reset or Enable the McBSP Receiver

Register	Bit	Name	Function		Type	Reset
SPCR1	0	$\overline{\text{RRST}}$	Receiver reset		R/W	0
			$\overline{\text{RRST}}$ = 0	The serial port receiver is disabled and in the reset state.		
			$\overline{\text{RRST}}$ = 1	The serial port receiver is enabled.		
SPCR2	6	$\overline{\text{GRST}}$	Sample rate generator reset		R/W	0
			$\overline{\text{GRST}}$ = 0	Sample rate generator is reset. If $\overline{\text{GRST}}$ = 0 due to an OMAP5912 reset, CLKG is driven by the CPU clock divided by 2, and FSG is driven low (inactive). If $\overline{\text{GRST}}$ = 0 due to program code, CLKG and FSG are both driven low (inactive).		
			$\overline{\text{GRST}}$ = 1	Sample rate generator is enabled. CLKG is driven according to the configuration programmed in the sample rate generator registers (SRGR[1,2]). If $\overline{\text{FRST}}$ = 1, the generator also generates the frame-synchronization signal FSG as programmed in the sample rate generator registers.		
SPCR2	7	$\overline{\text{FRST}}$	Frame-synchronization logic reset		R/W	0
			$\overline{\text{FRST}}$ = 0	Frame-synchronization logic is reset. The sample rate generator does not generate frame-synchronization signal FSG, even if $\overline{\text{GRST}}$ = 1.		
			$\overline{\text{FRST}}$ = 1	If $\overline{\text{GRST}}$ = 1, frame-synchronization signal FSG is generated after (FPER + 1) number of CLKG clock cycles; all frame counters are loaded with their programmed values.		

7.2.1 Reset Considerations

The serial port can be reset in the following two ways:

1) An OMAP5912 reset ($\overline{\text{RESET}}$ signal driven low) places the receiver, transmitter, and sample rate generator in reset. When the device reset is removed ($\overline{\text{RESET}}$ signal released), $\overline{\text{GRST}}$ = $\overline{\text{FRST}}$ = $\overline{\text{RRST}}$ = $\overline{\text{XRST}}$ = 0 keep the entire serial port in the reset state.

2) The serial port transmitter and receiver can be reset directly using the $\overline{\text{RRST}}$ and $\overline{\text{XRST}}$ bits in the serial port control registers. The sample rate generator can be reset directly using the $\overline{\text{GRST}}$ bit in SPCR2.

Table 15 shows the state of McBSP pins when the serial port is reset due to an OMAP5912 reset and a direct receiver/transmitter reset.

Table 15. Reset State of Each McBSP Pin

Pin	Possible State(s)	State Forced By OMAP5912 Reset	State Forced By Receiver/Transmitter Reset
			Receiver reset ($\overline{\text{RRST}}$ = 0 and $\overline{\text{GRST}}$ = 1)
DR	I	Input	Input
CLKR	I/O/Z	Input	Known state if input; CLKR running if output
FSR	I/O/Z	Input	Known state if input; FSRP inactive state if output
CLKS	I/O/Z	Input	Input
			Transmitter reset ($\overline{\text{XRST}}$ = 0 and $\overline{\text{GRST}}$ = 1)
DX	O/Z	High/Low impedance	Low impedance after transmit bit clock provided
CLKX	I/O/Z	Input	Known state if input; CLKX running if output
FSX	I/O/Z	Input	Known state if input; FSXP inactive state if output

Note: In Possible State(s) column, I = input, O = output, Z = high impedance

For more details about McBSP reset conditions and effects, see section 10.3, *Resetting and Initializing a McBSP*.

7.3 Set the Receiver Pins to Operate as McBSP Pins

The RIOEN bit, described in Table 16, determines whether the receiver pins are McBSP pins or general-purpose I/O pins.

Table 16. Register Bit Used to Set Receiver Pins to Operate as McBSP Pins

Register	Bit	Name	Function		Type	Reset
PCR	12	RIOEN	Receive I/O enable		R/W	0
			This bit is only applicable when the receiver is in the reset state ($\overline{\text{RRST}}$ = 0 in SPCR1).			
			RIOEN = 0	The DR, FSR, CLKR, and CLKS pins are configured as serial port pins and do not function as general-purpose I/O pins.		
			RIOEN = 1	The DR pin is a general-purpose input pin. The FSR and CLKR pins are general-purpose I/O pins. These serial port pins do not perform serial port operation. The CLKS pin is a general-purpose input pin if RIOEN = XIOEN = 1 and $\overline{\text{RRST}}$ = $\overline{\text{XRST}}$ = 0. For more information on using these pins as general-purpose I/O pins, see section 9.		

7.4 Enable/Disable the Digital Loopback Mode

The DLB bit determines whether the digital loopback mode is on. DLB is described in Table 17.

Table 17. Register Bit Used to Enable/Disable the Digital Loopback Mode

Register	Bit	Name	Function		Type	Reset
SPCR1	15	DLB	Digital loopback mode		R/W	0
			DLB = 0	Digital loopback mode is disabled.		
			DLB = 1	Digital loopback mode is enabled.		

7.4.1 Digital Loopback Mode

In the digital loopback mode, the receive signals are connected internally through multiplexers to the corresponding transmit signals, as shown in Table 18. This mode allows testing of serial port code with a single DSP device; the McBSP receives the data it transmits.

Table 18. Receive Signals Connected to Transmit Signals in Digital Loopback Mode

This Receive Signal	Is Fed Internally by This Transmit Signal
DR (receive data)	DX (transmit data)
FSR (receive frame synchronization)	FSX (transmit frame synchronization)
CLKR (receive clock)	CLKX (transmit clock)

7.5 Enable/Disable the Clock Stop Mode

The CLKSTP bits determine whether the clock stop mode is on. CLKSTP is described in Table 19.

Table 19. Register Bits Used to Enable/Disable the Clock Stop Mode

Register	Bit	Name	Function		Type	Reset
SPCR1	12:11	CLKSTP	Clock stop mode		R/W	00
			CLKSTP = 0Xb	Clock stop mode disabled; normal clocking for non-SPI mode.		
			CLKSTP = 10b	Clock stop mode enabled, without clock delay.		
			CLKSTP = 11b	Clock stop mode enabled, with clock delay.		

7.5.1 Clock Stop Mode

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 determines whether the starting edge of the clock on the CLKX pin is rising or falling. The CLKRP bit determines whether receive data is sampled on the rising or falling edge of the clock shown on the CLKR pin.

Table 20 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 receive frame-synchronization signal is tied internally to the transmit frame-synchronization signal.

Table 20. 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.

7.6 Enable/Disable the Receive Multichannel Selection Mode

The RMCM bit determines whether the receive multichannel selection mode is on. RMCM is described in Table 21.

Table 21. Register Bit Used to Enable/Disable the Receive Multichannel Selection Mode

Register	Bit	Name	Function		Type	Reset
MCR1	0	RMCM	Receive multichannel selection mode		R/W	0
			RMCM = 0	The mode is disabled. All 128 channels are enabled.		
			RMCM = 1	The mode is enabled. Channels can be individually enabled or disabled. The only channels enabled are those selected in the appropriate receive channel enable registers (RCERs). The way channels are assigned to the RCERs depends on the number of receive channel partitions (2 or 8), as defined by the RMCME bit.		

For more details, see section 5.6, *Receive Multichannel Selection Mode*.

7.7 Choose One or Two Phases for the Receive Frame

The RPHASE bit (see Table 22) determines whether the receive data frame has one or two phases.

Table 22. Register Bit Used to Choose One or Two Phases for the Receive Frame

Register	Bit	Name	Function	Type	Reset
RCR2	15	RPHASE	Receive phase number	R/W	0
			Specifies whether the receive frame has 1 or 2 phases.		
			RPHASE = 0 Single-phase frame		
			RPHASE = 1 Dual-phase frame		

7.8 Set the Receive Word Length(s)

The RWDLEN1 and RWDLEN2 bit fields (see Table 23) determine how many bits are in each serial word in phase 1 and in phase 2, respectively, of the receive data frame.

Table 23. Register Bits Used to Set the Receive Word Length(s)

Register	Bit	Name	Function	Type	Reset
RCR1	7:5	RWDLEN1	Receive word length 1. Specifies the length of every serial word in phase 1 of the receive frame.	R/W	000
			RWDLEN1 = 000 8 bits		
			RWDLEN1 = 001 12 bits		
			RWDLEN1 = 010 16 bits		
			RWDLEN1 = 011 20 bits		
			RWDLEN1 = 100 24 bits		
			RWDLEN1 = 101 32 bits		
			RWDLEN1 = 11X Reserved		
RCR2	7:5	RWDLEN2	Receive word length 2. If a dual-phase frame is selected, RWDLEN2 specifies the length of every serial word in phase 2 of the frame.	R/W	000
			RWDLEN2 = 000 8 bits		
			RWDLEN2 = 001 12 bits		
			RWDLEN2 = 010 16 bits		
			RWDLEN2 = 011 20 bits		
			RWDLEN2 = 100 24 bits		
			RWDLEN2 = 101 32 bits		
			RWDLEN2 = 11X Reserved		

7.8.1 Word Length Bits

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.

7.9 Set the Receive Frame Length

The RFRLEN1 and RFRLEN2 bit fields (see Table 24) determine how many serial words are in phase 1 and in phase 2, respectively, of the receive data frame.

Table 24. Register Bits Used to Set the Receive Frame Length

Register	Bit	Name	Function	Type	Reset
RCR1	14:8	RFRLEN1	Receive frame length 1	R/W	000 0000
			(RFRLEN1 + 1) is the number of serial words in phase 1 of the receive frame.		
			RFRLEN1 = 000 0000 1 word in phase 1		
			RFRLEN1 = 000 0001 2 words in phase 1		
			| |		
			| |		
			RFRLEN1 = 111 1111 128 words in phase 1		
RCR2	14:8	RFRLEN2	Receive frame length 2	R/W	000 0000
			If a dual-phase frame is selected, (RFRLEN2 + 1) is the number of serial words in phase 2 of the receive frame.		
			RFRLEN2 = 000 0000 1 word in phase 2		
			RFRLEN2 = 000 0001 2 words in phase 2		
			| |		
			| |		
			RFRLEN2 = 111 1111 128 words in phase 2		

7.9.1 Selected Frame Length

The receive frame length is the number of serial words in the receive frame. Each frame can have one or two phases, depending on value that you load 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 25 for a summary of how to calculate the frame length. This length corresponds to the number of words or logical time slots or channels per frame-synchronization pulse.

Program the RFRLEN fields with [*w 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 RFRLEN1.

Table 25. 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

7.10 Enable/Disable the Receive Frame-Synchronization Ignore Function

The RFIG bit (see Table 26) controls the receive frame-synchronization ignore function.

Table 26. Register Bit Used to Enable/Disable the Receive Frame-Synchronization Ignore Function

Register	Bit	Name	Function		Type	Reset
RCR2	2	RFIG	Receive frame-synchronization ignore		R/W	0
			RFIG = 0	An unexpected receive frame-synchronization pulse causes the McBSP to restart the frame transfer.		
			RFIG = 1	The McBSP ignores unexpected receive frame-synchronization pulses.		

7.10.1 Unexpected Frame-Synchronization Pulses and the Frame-Synchronization Ignore Function

If a frame-synchronization pulse starts the transfer of a new frame before the current frame is fully received, this pulse is treated as an unexpected frame-synchronization pulse.

When RFIG = 1, reception continues, ignoring the unexpected frame-synchronization 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-synchronization pulse occurs, the serial port:

1) Aborts the current data transfer.

2) Sets RSYNCERR in SPCR1 to 1.

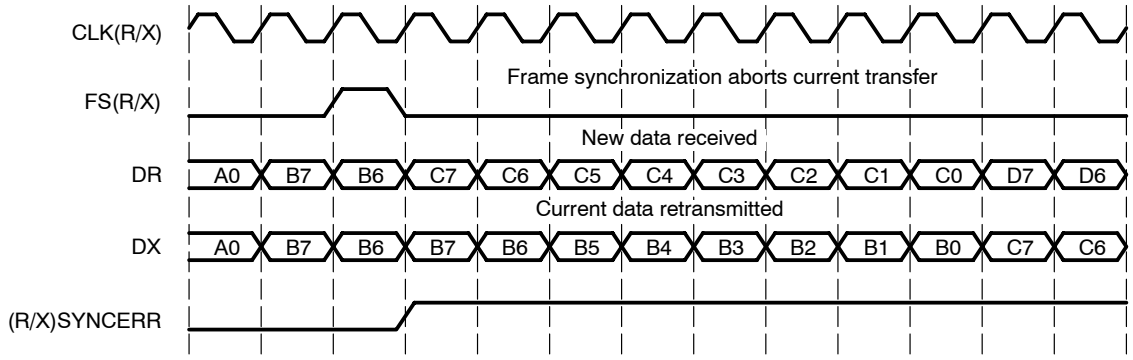
3) Begins the transfer of a new data word.

For more details about the frame-synchronization error condition, see section 4.2, *Unexpected Receive Frame-Synchronization Pulse*.

7.10.2 Examples of Effects of RFIG

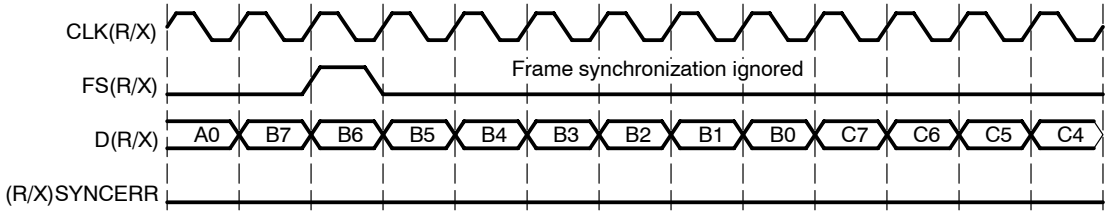
Figure 44 shows an example in which word B is interrupted by an unexpected frame-synchronization pulse when (R/X)FIG = 0. In the case of reception, the reception of B is aborted (B is lost), and a new data word (C in this example) is received after the appropriate data delay. This condition is a receive synchronization error, which sets the RSYNCERR bit.

Figure 44. Unexpected Frame-Synchronization Pulse With (R/X)FIG = 0



In contrast with Figure 44, Figure 45 shows McBSP operation when unexpected frame-synchronization signals are ignored (when (R/X)FIG = 1). Here, the transfer of word B is not affected by an unexpected pulse.

Figure 45. Unexpected Frame-Synchronization Pulse With (R/X)FIG = 1



7.11 Set the Receive Companding Mode

The RCOMPAND bits (see Table 27) determine whether companding or another data transfer option is chosen for McBSP reception.

Table 27. Register Bits Used to Set the Receive Companding Mode

Register	Bit	Name	Function	Type	Reset
RCR2	4:3	RCOMPAND	Receive companding mode	R/W	00
			Modes other than 00b are enabled only when the appropriate RWDLEN is 000b, indicating 8-bit data.		
			RCOMPAND = 00 No companding, any size data, MSB received first.		
			RCOMPAND = 01 No companding, 8-bit data, LSB received first (for details, see section 7.11.4, *Option to Receive LSB First*).		
			RCOMPAND = 10 μ-law companding, 8-bit data, MSB received first.		
			RCOMPAND = 11 A-law companding, 8-bit data, MSB received first.		

7.11.1 Companding

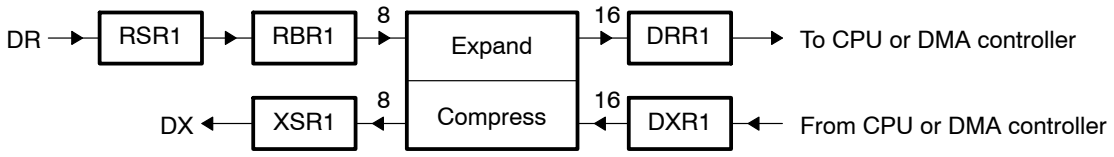
Companding (COMpressing and exPANDing) hardware allows compression and expansion of data in either μ-law or A-law format. The companding standard employed in the United States and Japan is μ-law. The European companding standard is referred to as A-law. The specifications for μ-law and A-law log PCM are part of the CCITT G.711 recommendation.

A-law and μ-law allow 13 bits and 14 bits of dynamic range, respectively. Any 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 via the CPU or DMA controller must be at least 16 bits wide.

The μ-law and A-law formats both encode data into 8-bit code words. Companded data is always 8 bits wide; the appropriate word length bits (RWDLEN1, RWDLEN2, XWDLEN1, XWDLEN2) must therefore be set to 0, indicating an 8-bit wide serial data stream. If companding is enabled and either of the frame phases does not have an 8-bit word length, companding continues as if the word length is 8 bits.

Figure 46 illustrates the companding processes. When companding is chosen for the transmitter, compression occurs during the process of copying data from DXR1 to XSR1. The transmit data is encoded according to the specified companding law (A-law or μ-law). When companding is chosen for the receiver, expansion occurs during the process of copying data from RBR1 to DRR1. The receive data is decoded to twos-complement format.

Figure 46. Companding Processes for Reception and for Transmission



7.11.2 Format of Expanded Data

For reception, the 8-bit compressed data in RBR1 is expanded to left-justified 16-bit data in DRR1. The RJUST bit of SPCR1 is ignored when companding is used.

7.11.3 Companding Internal Data

If the McBSP is otherwise unused (the serial port transmit and receive sections are reset), the companding hardware can compand internal data. See section 2.2.2, *Capability to Compand Internal Data*.

7.11.4 Option to Receive LSB First

Normally, the McBSP transmits or receives all data with the most significant bit (MSB) first. However, certain 8-bit data protocols that do not use companded data require the least significant bit (LSB) to be transferred first. If you set RCOMPAND = 01b in RCR2, the bit ordering of 8-bit words is reversed 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.

7.12 Set the Receive Data Delay

The RDATDLY bits (see Table 28) determine the length of the data delay for the receive frame.

Table 28. Register Bits Used to Set the Receive Data Delay

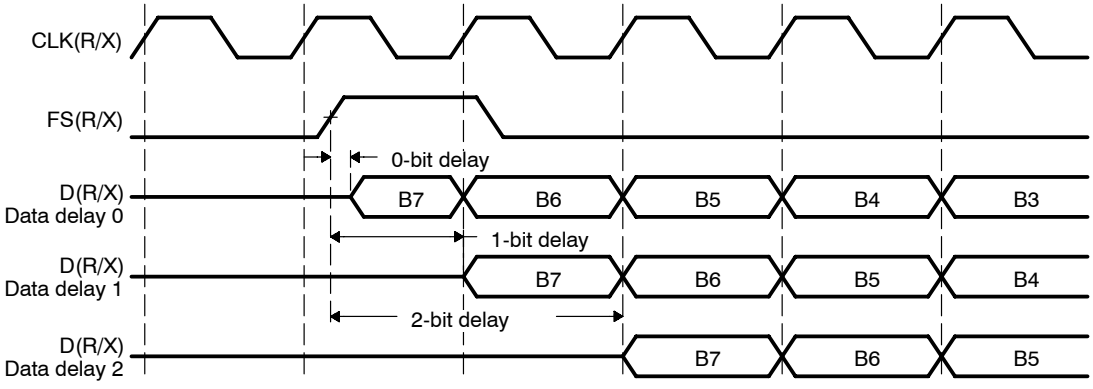
Register	Bit	Name	Function		Type	Reset
RCR2	1:0	RDATDLY	Receive data delay		R/W	00
			RDATDLY = 00	0-bit data delay		
			RDATDLY = 01	1-bit data delay		
			RDATDLY = 10	2-bit data delay		
			RDATDLY = 11	Reserved		

7.12.1 Data Delay

The start of a frame is defined by the first clock cycle in which frame synchronization is found to be active. The beginning of actual data 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 described in Table 28 and shown in Figure 47. In this figure, the data transferred is an 8-bit value with bits labeled B7, B6, B5, and so on. Typically a 1-bit delay is selected, because data often follows a 1-cycle active frame-synchronization pulse.

Figure 47. Range of Programmable Data Delay



7.12.2 0-Bit Data Delay

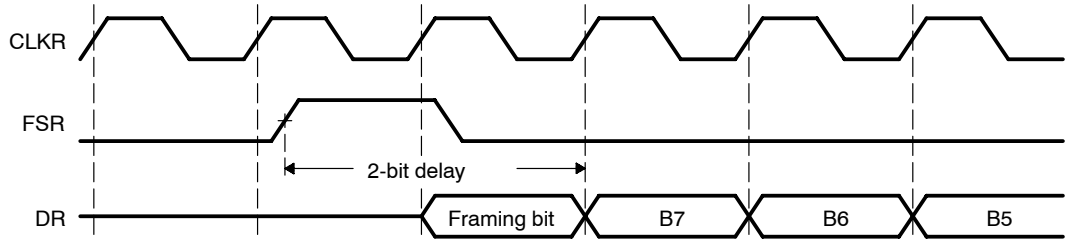
Normally, a frame-synchronization pulse is detected or sampled with respect to an edge of internal serial clock CLK(R/X). Thus, on the following cycle or later (depending on the data delay value), data may be received or transmitted. However, in the case of 0-bit data delay, the data must be ready for reception and/or transmission on the same serial clock cycle.

For reception, this problem is solved because receive data is sampled on the first falling edge of CLKR where an active-high internal FSR is detected. However, data transmission must begin on the rising edge of the internal CLKX clock that generated the frame synchronization. Therefore, the first data bit is assumed to be present in XSR1, and thus on DX. The transmitter then asynchronously detects the frame-synchronization signal (FSX) going active high and immediately starts driving the first bit to be transmitted on the DX pin.

7.12.3 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 48. In this figure, the data transferred is an 8-bit value with bits labeled B7, B6, B5, and so on.

Figure 48. 2-Bit Data Delay Used to Skip a Framing Bit



7.13 Set the Receive Sign-Extension and Justification Mode

The RJUST bits (see Table 29) determine whether data received by the McBSP is sign-extended and how it is justified.

Table 29. Register Bits Used to Set the Receive Sign-Extension and Justification Mode

Register	Bit	Name	Function		Type	Reset
SPCR1	14:13	RJUST	Receive sign-extension and justification mode		R/W	00
			RJUST = 00	Right justify data and zero fill MSBs in DRR[1,2].		
			RJUST = 01	Right justify data and sign extend it into the MSBs in DRR[1,2].		
			RJUST = 10	Left justify data and zero fill LSBs in DRR[1,2].		
			RJUST = 11	Reserved.		

7.13.1 Sign-Extension and the Justification

RJUST in SPCR1 selects whether data in RBR[1,2] is right- or left-justified (with respect to the MSB) in DRR[1,2] and whether unused bits in DRR[1,2] are filled with zeros or with sign bits.

Table 30 and Table 31 show the effects of various RJUST values. The first table shows the effect on an example 12-bit receive-data value 0xABC. The second table shows the effect on an example 20-bit receive-data value 0xABCDE.

Table 30. Example: Use of RJUST 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 31. Example: Use of RJUST 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

7.14 Set the Receive Interrupt Mode

The RINTM bits (see Table 32) determine 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. The options are set by the receive interrupt mode bits, RINTM, in SPCR1.

Table 32. Register Bits Used to Set the Receive Interrupt Mode

Register	Bit	Name	Function		Type	Reset
SPCR1	5:4	RINTM	Receive interrupt mode		R/W	00
			RINTM = 00	RINT generated when RRDY changes from 0 to 1. Interrupt on every serial word by tracking the RRDY bit in SPCR1. Regardless of the value of RINTM, RRDY can be read to detect the RRDY = 1 condition.		
			RINTM = 01	RINT 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 has been crossed within a frame and at the end of the frame. For details, see section 5.8, *Using Interrupts Between Block Transfers*. In any other serial transfer case, this setting is not applicable and, therefore, no interrupts are generated.		
			RINTM = 10	RINT generated by a new receive frame-synchronization pulse. Interrupt on detection of receive frame-synchronization pulses. This generates an interrupt even when the receiver is in its reset state. This is done by synchronizing the incoming frame-synchronization pulse to the CPU clock and sending it to the CPU via RINT.		
			RINTM = 11	RINT generated when RSYNCERR is set. Interrupt on frame-synchronization error. Regardless of the value of RINTM, RSYNCERR can be read to detect this condition. For information on using RSYNCERR, see section 4.2, *Unexpected Receive Frame-Synchronization Pulse*.		

7.15 Set the Receive Frame-Synchronization Mode

The bits described in Table 33 determine the source for receive frame synchronization and the function of the FSR pin.

Table 33. Register Bits Used to Set the Receive Frame Synchronization Mode

Register	Bit	Name	Function		Type	Reset
PCR	10	FSRM	Receive frame-synchronization mode		R/W	0
			FSRM = 0	Receive frame synchronization is supplied by an external source via the FSR pin.		
			FSRM = 1	Receive frame synchronization is supplied by the sample rate generator. FSR is an output pin reflecting internal FSR, except when GSYNC = 1 in SRGR2.		
SRGR2	15	GSYNC	Sample rate generator clock synchronization mode. If the sample rate generator creates a frame-synchronization signal (FSG) that is derived from an external input clock, the GSYNC bit determines whether FSG is kept synchronized with pulses on the FSR pin.		R/W	0
			GSYNC = 0	No clock synchronization is used: CLKG oscillates without adjustment, and FSG pulses every (FPER + 1) CLKG cycles.		
			GSYNC = 1	Clock synchronization is used. When a pulse is detected on the FSR pin: ❑ CLKG is adjusted as necessary so that it is synchronized with the input clock on the CLKS, CLKR, or CLKX pin. ❑ FSG pulses. FSG only pulses in response to a pulse on the FSR pin. The frame-synchronization period defined in FPER is ignored.		
				For more details, see section 3.3, *Synchronizing Sample Rate Generator Outputs to an External Clock*.		
SPCR1	15	DLB	Digital loopback mode		R/W	0
			DLB = 0	Digital loopback mode is disabled.		
			DLB = 1	Digital loopback mode is enabled. The receive signals, including the receive frame-synchronization signal, are connected internally through multiplexers to the corresponding transmit signals.		

Table 33. Register Bits Used to Set the Receive Frame Synchronization Mode (Continued)

Register	Bit	Name	Function		Type	Reset
SPCR1	12:11	CLKSTP	Clock stop mode		R/W	00
			CLKSTP = 0Xb	Clock stop mode disabled; normal clocking for non-SPI mode.		
			CLKSTP = 10b	Clock stop mode enabled without clock delay. The internal receive clock signal (CLKR) and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX.		
			CLKSTP = 11b	Clock stop mode enabled with clock delay. The internal receive clock signal (CLKR) and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX.		

7.15.1 Receive Frame-Synchronization Modes

Table 34 shows how you can select various sources to provide the receive frame-synchronization signal and the effect on the FSR pin. The polarity of the signal on the FSR pin is determined by the FSRP bit.

In digital loopback mode (DLB = 1), the transmit frame-synchronization signal is used as the receive frame-synchronization signal.

Also in the clock stop mode, the internal receive clock signal (CLKR) and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX.

Table 34. Select Sources to Provide the Receive Frame-Synchronization Signal and the Effect on the FSR Pin

DLB	FSRM	GSYNC	Source of Receive Frame Synchronization	FSR Pin Status
0	0	0 or 1	An external frame-synchronization signal enters the McBSP through the 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 sample rate generator frame-synchronization signal (FSG).	Output. FSG is inverted as determined by FSRP before being driven out on the FSR pin.
0	1	1	Internal FSR is driven by the sample rate generator frame-synchronization signal (FSG).	Input. The external frame-synchronization input on the 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 sample rate generator 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 transmit) frame synchronization is inverted as determined by FSRP before being driven out on the FSR pin.

7.16 Set the Receive Frame-Synchronization Polarity

The FSRP bit (see Table 35) determines whether frame-synchronization pulses are active high or active low on the FSR pin.

Table 35. Register Bit Used to Set Receive Frame-Synchronization Polarity

Register	Bit	Name	Function		Type	Reset
PCR	2	FSRP	Receive frame-synchronization polarity		R/W	0
			FSRP = 0	Frame-synchronization pulse FSR is active high.		
			FSRP = 1	Frame-synchronization pulse FSR is active low.		

7.16.1 Frame-Synchronization Pulses, Clock Signals, and Their Polarities

Receive frame-synchronization pulses can be generated internally by the sample rate generator (see section 3.2) or driven by an external source. The source of frame synchronization is selected by programming the mode bit, FSRM, in PCR. FSR is also affected by the GSYNC bit in SRGR2. For information about the effects of FSRM and GSYNC, see section 7.15, *Set the Receive Frame-Synchronization Mode*. Similarly, receive clocks can be selected to be inputs or outputs by programming the mode bit, CLKRM, in the PCR (see section 7.17, *Set the Receive Clock Mode*).

When FSR and FSX are inputs (FSXM = FSRM = 0, external frame-synchronization pulses), the McBSP detects them on the internal falling edge of clock, internal CLKR, and internal CLKX, respectively. The receive data arriving at the DR pin is also sampled on the falling edge of internal CLKR. These internal clock signals are either derived from an external source via CLK(R/X) pins, or driven by the sample rate generator clock (CLKG) internal to the McBSP.

When FSR and FSX are outputs, implying that they are driven by the sample rate generator, they are generated, or transition to their active state, on the rising edge of the internal clock, CLK(R/X). Similarly, data on the DX pin is output on the rising edge of internal CLKX.

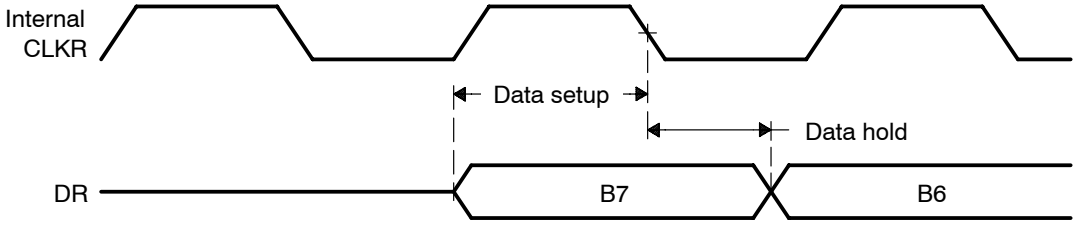
FSRP, FSXP, CLKRP, and CLKXP in the pin control register (PCR) configure the polarities of the FSR, FSX, CLKR, and CLKX signals, respectively. All frame-synchronization signals (internal FSR, internal FSX) that are internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to McBSP), and FSRP = FSXP = 1, the external active-low frame-synchronization signals are inverted before being sent to the receiver (internal FSR) and transmitter (internal FSX). Similarly, if internal synchronization is selected (FSR/FSX are output pins and GSYNC = 0), the internal active-high frame-synchronization signals are inverted before being sent to the FS(R/X) pin, if the polarity bit FS(R/X)P = 1.

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 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 CLKR pin.

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 49 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 49. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge



Set the SRG Frame-Synchronization Period and Pulse Width

7.16.2 Frame-Synchronization Period and the Frame-Synchronization Pulse Width

The sample rate generator can produce a clock signal, CLKG, and a frame-synchronization signal, FSG. If the sample rate generator is supplying receive or transmit frame synchronization, you must program the bit fields FPER and FWID.

On FSG, the period from the start of a frame-synchronization pulse to the start of the next pulse is (FPER + 1) CLKG cycles. The 12 bits of FPER allow a frame-synchronization period of 1 to 4096 CLKG cycles, which allows up to 4096 data bits per frame. When GSYNC = 1, FPER is a don't care value.

Each pulse on FSG has a width of (FWID + 1) CLKG cycles. The eight bits of FWID allow a pulse width of 1 to 256 CLKG cycles. It is recommended that FWID be programmed to a value less than the programmed word length.

The values in FPER and FWID are loaded into separate down-counters. The 12-bit FPER counter counts down the generated clock cycles from the programmed value (4095 maximum) to 0. The 8-bit FWID counter counts down from the programmed value (255 maximum) to 0.

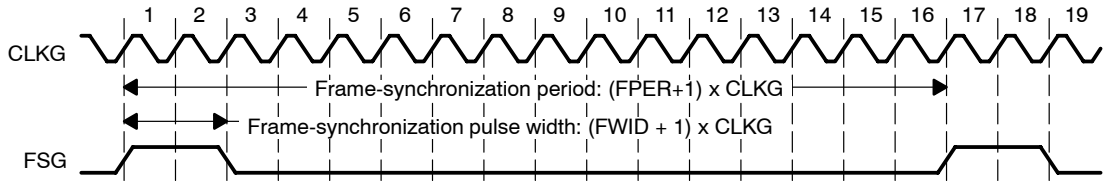
Table 36 shows settings for FPER and FWID.

Table 36. Register Bits Used to Set the SRG Frame-Synchronization Period and Pulse Width

Register	Bit	Name	Function	Type	Reset
SRGR2	11:0	FPER	Sample rate generator frame-synchronization period For the frame-synchronization signal FSG, (FPER + 1) determines the period from the start of a frame-synchronization pulse to the start of the next frame-synchronization pulse. Range for (FPER + 1): 1 to 4096 CLKG cycles	R/W	0000 0000 0000
SRGR1	15:8	FWID	Sample rate generator frame-synchronization pulse width This field plus 1 determines the width of each frame-synchronization pulse on FSG. Range for (FWID + 1): 1 to 256 CLKG cycles	R/W	0000 0000

Figure 50 shows a frame-synchronization period of 16 CLKG periods (FPER = 15 or 00001111b) and a frame-synchronization pulse with an active width of 2 CLKG periods (FWID = 1).

Figure 50. Frame of Period 16 CLKG Periods and Active Width of 2 CLKG Periods



When the sample rate generator comes out of reset, FSG is in its inactive state. Then, when $\overline{\text{GRST}}$ = 1 and FSGM = 1, a frame-synchronization pulse is generated. The frame width value (FWID + 1) is counted down on every CLKG cycle until it reaches 0, at which time FSG goes low. At the same time, the frame period value (FPER + 1) is also counting down. When this value reaches 0, FSG goes high, indicating a new frame.

7.17 Set the Receive Clock Mode

Table 37 shows the settings for bits used to set receive clock mode.

Table 37. Register Bits Used to Set the Receive Clock Mode

Register	Bit	Name	Function		Type	Reset
PCR	8	CLKRM	Receive clock mode		R/W	0
			Case 1: Digital loopback mode not set (DLB = 0) in SPCR1.			
			CLKRM = 0	The CLKR pin is an input pin that supplies the internal receive clock (CLKR).		
			CLKRM = 1	Internal CLKR is driven by the sample rate generator of the McBSP. The CLKR pin is an output pin that reflects internal CLKR.		
			Case 2: Digital loopback mode set (DLB = 1) in SPCR1.			
			CLKRM = 0	The CLKR pin is in the high impedance state. The internal receive clock (CLKR) is driven by the internal transmit clock (CLKX). Internal CLKX is derived according to the CLKXM bit of PCR.		
			CLKRM = 1	Internal CLKR is driven by internal CLKX. The CLKR pin is an output pin that reflects internal CLKR. Internal CLKX is derived according to the CLKXM bit of PCR.		
SPCR1	15	DLB	Digital loopback mode		R/W	00
			DLB = 0	Digital loopback mode is disabled.		
			DLB = 1	Digital loopback mode is enabled. The receive signals, including the receive frame-synchronization signal, are connected internally through multiplexers to the corresponding transmit signals.		

Table 37. Register Bits Used to Set the Receive Clock Mode (Continued)

Register	Bit	Name	Function		Type	Reset
SPCR1	12:11	CLKSTP	Clock stop mode		R/W	00
			CLKSTP = 0Xb	Clock stop mode disabled; normal clocking for non-SPI mode.		
			CLKSTP = 10b	Clock stop mode enabled without clock delay. The internal receive clock signal (CLKR) and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX.		
			CLKSTP = 11b	Clock stop mode enabled with clock delay. The internal receive clock signal (CLKR) and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX.		

7.17.1 Selecting a Source for the Receive Clock and a Data Direction for the CLKR Pin

Table 38 shows how you can select various sources to provide the receive clock signal and affect the CLKR pin. The polarity of the signal on the 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 receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX.

Table 38. Receive Clock Signal Source Selection

DLB in SPCR1	CLKRM in PCR	Source of Receive Clock	CLKR Pin Status
0	0	The 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 sample rate generator clock (CLKG) drives internal CLKR.	Output. CLKG, inverted as determined by CLKRP, is driven out on the CLKR pin.
1	0	Internal CLKX drives internal CLKR. To configure CLKX, see section 8.18, *Set the Transmit Clock Mode*.	High impedance.
1	1	Internal CLKX drives internal CLKR. To configure CLKX, see section 8.18, *Set the Transmit Clock Mode*.	Output. Internal CLKR (same as internal CLKX) is inverted as determined by CLKRP before being driven out on the CLKR pin.

7.18 Set the Receive Clock Polarity

Table 39. Register Bit Used to Set Receive Clock Polarity

Register	Bit	Name	Function		Type	Reset
PCR	0	CLKRP	Receive clock polarity		R/W	0
			CLKRP = 0	Receive data sampled on falling edge of CLKR		
			CLKRP = 1	Receive data sampled on rising edge of CLKR		

7.18.1 Frame Synchronization Pulses, Clock Signals, and Their Polarities

Receive frame-synchronization pulses can be generated internally by the sample rate generator (see section 3.2) or driven by an external source. The source of frame synchronization is selected by programming the mode bit, FSRM, in PCR. FSR is also affected by the GSYNC bit in SRGR2. For information about the effects of FSRM and GSYNC, see section 7.15, *Set the Receive Frame-Synchronization Mode*. Similarly, receive clocks can be selected to be inputs or outputs by programming the mode bit, CLKRM, in the PCR (see section 7.17, *Set the Receive Clock Mode*).

When FSR and FSX are inputs (FSXM = FSRM = 0, external frame-synchronization pulses), the McBSP detects them on the internal falling edge of clock, internal CLKR, and internal CLKX, respectively. The receive data arriving at the DR pin is also sampled on the falling edge of internal CLKR. These internal clock signals are either derived from external source via CLK(R/X) pins or driven by the sample rate generator clock (CLKG) internal to the McBSP.

When FSR and FSX are outputs, implying that they are driven by the sample rate generator, they are generated, or transition to their active state, on the rising edge of internal clock, CLK(R/X). Similarly, data on the DX pin is output on the rising edge of internal CLKX.

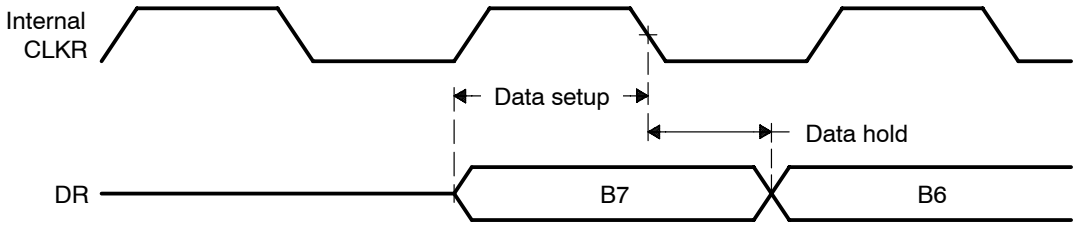
FSRP, FSXP, CLKRP, and CLKXP in the pin control register (PCR) configure the polarities of the FSR, FSX, CLKR, and CLKX signals, respectively. All frame-synchronization signals (internal FSR, internal FSX) that are internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to McBSP) and FSRP = FSXP = 1, the external active-low frame-synchronization signals are inverted before being sent to the receiver (internal FSR) and transmitter (internal FSX). Similarly, if internal synchronization (FSR/FSX are output pins and GSYNC = 0) is selected, the internal active-high frame-synchronization signals are inverted, if the polarity bit FS(R/X)P = 1, before being sent to the FS(R/X) pin.

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 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 CLKR pin.

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 51 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 51. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge



7.19 Set the SRG Clock Divide-Down Value

Table 40. Register Bits Used to Set the Sample Rate Generator (SRG) Clock Divide-Down Value

Register	Bit	Name	Function	Type	Reset
SRGR1	7:0	CLKGDV	Sample rate generator clock divide-down value The input clock of the sample rate generator is divided by (CLKGDV + 1) to generate the required sample rate generator clock frequency. The default value of CLKGDV is 1 (divide input clock by 2).	R/W	0000 0001

7.19.1 Sample Rate Generator Clock Divider

The first divider stage generates the serial data bit clock from the input clock. This divider stage utilizes a counter, preloaded by CLKGDV, that contains the divide ratio value.

The output of the first divider stage is the data bit clock, which is output as CLKG and which serves as the input for the second and third stages of the divider.

CLKG has a frequency equal to 1/(CLKGDV + 1) of sample rate generator 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.

7.20 Set the SRG Clock Synchronization Mode

Table 41. Register Bit Used to Set the SRG Clock Synchronization Mode

Register	Bit	Name	Function	Type	Reset
SRGR2	15	GSYNC	Sample rate generator clock synchronization GSYNC is used only when the input clock source for the sample rate generator is external– on the CLKS, CLKR, or CLKX pin. GSYNC = 0: The sample rate generator clock (CLKG) is free running. CLKG oscillates without adjustment, and FSG pulses every (FPER + 1) CLKG cycles. GSYNC = 1: Clock synchronization is performed. When a pulse is detected on the FSR pin: ❑ CLKG is adjusted as necessary so that it is synchronized with the input clock on the CLKS, CLKR, or CLKX pin. ❑ FSG pulses. FSG only pulses in response to a pulse on the FSR pin. The frame-synchronization period defined in FPER is ignored.	R/W	0

For more details on using the clock synchronization feature, see section 3.3, *Synchronizing Sample Rate Generator Outputs to an External Clock*.

7.21 Set the SRG Clock Mode (Choose an Input Clock)

Table 42. Register Bits Used to Set the SRG Clock Mode (Choose an Input Clock)

Register	Bit	Name	Function		Type	Reset
PCR SRGR2	7 13	SCLKME CLKSM	Sample rate generator clock mode		R/W R/W	0 1
			SCLKME = 0 CLKSM = 0	Sample rate generator clock derived from CLKS pin.		
			SCLKME = 0 CLKSM = 1	Sample rate generator clock derived from CPU clock. (This is the condition forced by an OMAP5912 reset.)		
			SCLKME = 1 CLKSM = 0	Sample rate generator clock derived from CLKR pin.		
			SCLKME = 1 CLKSM = 1	Sample rate generator clock derived from CLKX pin.		

7.21.1 SRG Clock Mode

The sample rate generator can produce a clock signal (CLKG) for use by the receiver, the transmitter, or both, but CLKG is derived from an input clock. Table 42 shows the four possible sources of the input clock. For more details on generating CLKG, see section 3.1, *Clock Generation in the Sample Rate Generator*.

7.22 Set the SRG Input Clock Polarity

Table 43. Register Bits Used to Set the SRG Input Clock Polarity

Register	Bit	Name	Function		Type	Reset
SRGR2	14	CLKSP	CLKS pin polarity. CLKSP determines the input clock polarity when the CLKS pin supplies the input clock (SCLKME = 0 and CLKSM = 0).		R/W	0
			CLKSP = 0	Rising edge on CLKS pin generates CLKG and FSG.		
			CLKSP = 1	Falling edge on CLKS pin generates CLKG and FSG.		
PCR	1	CLKXP	CLKX pin polarity. CLKXP determines the input clock polarity when the CLKX pin supplies the input clock (SCLKME = 1 and CLKSM = 1).		R/W	0
			CLKXP = 0	Rising edge on CLKX pin generates transitions on CLKG and FSG.		
			CLKXP = 1	Falling edge on CLKX pin generates transitions on CLKG and FSG.		
PCR	0	CLKRP	CLKR pin polarity. CLKRP determines the input clock polarity when the CLKR pin supplies the input clock (SCLKME = 1 and CLKSM = 0).		R/W	0
			CLKRP = 0	Falling edge on CLKR pin generates transitions on CLKG and FSG.		
			CLKRP = 1	Rising edge on CLKR pin generates transitions on CLKG and FSG.		

7.22.1 Using CLKSP/CLKXP/CLKRP to Choose an Input Clock Polarity

The sample rate generator can produce a clock signal (CLKG) and a frame-synchronization signal (FSG) for use by the receiver, the transmitter, or both. To produce CLKG and FSG, the sample rate generator must be driven by an input clock signal derived from the CPU clock or from an external clock on the CLKS, CLKX, or CLKR pin. If you use a pin, choose a polarity for that pin by using the appropriate polarity bit: CLKSP for the CLKS pin, CLKXP for the CLKX pin, and CLKRP for the CLKR pin. The polarity determines whether the rising or falling edge of the input clock generates transitions on CLKG and FSG.

8 Transmitter Configuration

To configure the McBSP transmitter, perform the following procedure:

1) Place the McBSP/transmitter in reset (see section 8.2).
2) Program the McBSP registers for the desired transmitter operation (see section 8.1).
3) Take the transmitter out of reset (see section 8.2).

8.1 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.

It may be helpful to print the McBSP Register Worksheet at the end of this document first and to fill in the worksheet as you read the tasks.

- ❑ 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 transmit frame-synchronization ignore function.
  - ■ Set the transmit companding mode.
  - ■ Set the transmit data delay.
  - ■ Set the transmit DXENA mode.
  - ■ Set the transmit interrupt mode.
- ❑ Frame-synchronization behavior:
  - ■ Set the transmit frame-synchronization mode.
  - ■ Set the transmit frame-synchronization polarity.
  - ■ Set the SRG frame-synchronization 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.

8.2 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). Table 44 describes the bits used for both of these steps.

Table 44. Register Bits Used to Place Transmitter in Reset

Register	Bit	Name	Function		Type	Reset
SPCR2	0	$\overline{\text{XRST}}$	Transmitter reset		R/W	0
			$\overline{\text{XRST}}$ = 0	The serial port transmitter is disabled and in the reset state.		
			$\overline{\text{XRST}}$ = 1	The serial port transmitter is enabled.		
SPCR2	6	$\overline{\text{GRST}}$	Sample rate generator reset		R/W	0
			$\overline{\text{GRST}}$ = 0	Sample rate generator is reset. If $\overline{\text{GRST}}$ = 0 due to an OMAP5912 reset, CLKG is driven by the CPU clock divided by 2, and FSG is driven low (inactive). If $\overline{\text{GRST}}$ = 0 due to program code, CLKG and FSG are both driven low (inactive).		
			$\overline{\text{GRST}}$ = 1	Sample rate generator is enabled. CLKG is driven according to the configuration programmed in the sample rate generator registers (SRGR[1,2]). If $\overline{\text{FRST}}$ = 1, the generator also generates the frame-synchronization signal FSG as programmed in the sample rate generator registers.		
SPCR2	7	$\overline{\text{FRST}}$	Frame-synchronization logic reset		R/W	0
			$\overline{\text{FRST}}$ = 0	Frame-synchronization logic is reset. The sample rate generator does not generate frame-synchronization signal FSG, even if $\overline{\text{GRST}}$ = 1.		
			$\overline{\text{FRST}}$ = 1	If $\overline{\text{GRST}}$ = 1, frame-synchronization signal FSG is generated after (FPER + 1) number of CLKG clock cycles; all frame counters are loaded with their programmed values.		

8.2.1 Reset Considerations

The serial port can be reset in the following two ways:

1) An OMAP5912 reset ($\overline{\text{RESET}}$ signal driven low) places the receiver, transmitter, and sample rate generator in reset. When the device reset is removed ($\overline{\text{RESET}}$ signal released), $\overline{\text{GRST}}$ = $\overline{\text{FRST}}$ = $\overline{\text{RRST}}$ = $\overline{\text{XRST}}$ = 0, keeping the entire serial port in the reset state.

2) The serial port transmitter and receiver can be reset directly using the $\overline{\text{RRST}}$ and $\overline{\text{XRST}}$ bits in the serial port control registers. The sample rate generator can be reset directly using the $\overline{\text{GRST}}$ bit in SPCR2.

Table 45 shows the state of McBSP pins when the serial port is reset due to an OMAP5912 reset and a direct receiver/transmitter reset.

Table 45. Reset State of Each McBSP Pin

Pin	Possible State(s)	State Forced by OMAP5912 Reset	State Forced by Receiver/Transmitter Reset
		Receiver Reset ($\overline{\text{RRST}}$ = 0 and $\overline{\text{GRST}}$ = 1)	
DR	I	Input	Input
CLKR	I/O/Z	Input	Known state if input; CLKR running if output
FSR	I/O/Z	Input	Known state if input; FSRP inactive state if output
CLKS	I/O/Z	Input	Input
		Transmitter Reset ($\overline{\text{XRST}}$ = 0 and $\overline{\text{GRST}}$ = 1)	
DX	O/Z	High impedance	High impedance
CLKX	I/O/Z	Input	Known state if input; CLKX running if output
FSX	I/O/Z	Input	Known state if input; FSXP inactive state if output

For more details about McBSP reset conditions and effects, see section 10.3, *Resetting and Initializing a McBSP*.

8.3 Set the Transmitter Pins to Operate as McBSP Pins

Table 46. Register Bit Used to Set Transmitter Pins to Operate as McBSP Pins

Register	Bit	Name	Function		Type	Reset
PCR	13	XIOEN	Transmit I/O enable		R/W	0
			This bit is only applicable when the transmitter is in the reset state ($\overline{XRST}$ = 0 in SPCR2).			
			XIOEN = 0	The DX, FSX, CLKX, and CLKS pins are configured as serial port pins and do not function as general-purpose I/Os.		
			XIOEN = 1	The DX pin is a general-purpose output pin. The FSX and CLKX pins are general-purpose I/O pins. These serial port pins do not perform serial port operations. The CLKS pin is a general-purpose input pin if RIOEN = XIOEN = 1, and $\overline{RRST}$ = $\overline{XRST}$ = 0. For more information on using these pins as general-purpose I/O pins, see section 9.		

8.4 Enable/Disable the Digital Loopback Mode

The DLB bit determines whether the digital loopback mode is on. DLB is described in Table 47.

Table 47. Register Bit Used to Enable/Disable the Digital Loopback Mode

Register	Bit	Name	Function		Type	Reset
SPCR1	15	DLB	Digital loopback mode		R/W	0
			DLB = 0	Digital loopback mode is disabled.		
			DLB = 1	Digital loopback mode is enabled.		

8.4.1 Digital Loopback Mode

In the digital loopback mode, the receive signals are connected internally through multiplexers to the corresponding transmit signals, as shown in Table 48. This mode allows testing of serial port code with a single DSP device; the McBSP receives the data it transmits.

Table 48. Receive Signals Connected to Transmit Signals in Digital Loopback Mode

This Receive Signal	Is Fed Internally by This Transmit Signal
DR (receive data)	DX (transmit data)
FSR (receive frame synchronization)	FSX (transmit frame synchronization)
CLKR (receive clock)	CLKX (transmit clock)

8.5 Enable/Disable the Clock Stop Mode

The CLKSTP bits determine whether the clock stop mode is on. CLKSTP is described in Table 49.

Table 49. Register Bits Used to Enable/Disable the Clock Stop Mode

Register	Bit	Name	Function		Type	Reset
SPCR1	12:11	CLKSTP	Clock stop mode		R/W	00
			CLKSTP = 0Xb	Clock stop mode disabled; normal clocking for non-SPI mode.		
			CLKSTP = 10b	Clock stop mode enabled without clock delay.		
			CLKSTP = 11b	Clock stop mode enabled with clock delay.		

8.5.1 Clock Stop Mode

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 determines whether the starting edge of the clock on the CLKX pin is rising or falling. The CLKRP bit determines whether receive data is sampled on the rising or falling edge of the clock shown on the CLKR pin.

Table 50 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 receive frame-synchronization signal is tied internally to the transmit frame-synchronization signal.

Table 50. 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.

8.6 Enable/Disable Transmit Multichannel Selection

Table 51. Register Bits Used to Enable/Disable Transmit Multichannel Selection

Register	Bit	Name	Function		Type	Reset
MCR2	1:0	XMCM	Transmit multichannel selection		R/W	00
			XMCM = 00b	No transmit multichannel selection mode is on. All channels are enabled and unmasked. No channels can be disabled or masked.		
			XMCM = 01b	All channels are disabled unless they are selected in the appropriate transmit channel enable registers (XCERs). If enabled, a channel in this mode is also unmasked. The XMCME bit determines whether 32 channels or 128 channels are selectable in XCERs.		
			XMCM = 10b	All channels are enabled, but they are masked unless they are selected in the appropriate transmit channel enable registers (XCERs). The XMCME bit determines whether 32 channels or 128 channels are selectable in XCERs.		
			XMCM = 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 (RCERs). Once enabled, they are masked unless they are also selected in the appropriate transmit channel enable registers (XCERs). The XMCME bit determines whether 32 channels or 128 channels are selectable in RCERs and XCERs.		

For more details, see section 5.7, *Transmit Multichannel Selection Modes*.

8.7 Choose One or Two Phases for the Transmit Frame

Table 52. Register Bit Used to Choose 1 or 2 Phases for the Transmit Frame

Register	Bit	Name	Function	Type	Reset
XCR2	15	XPHASE	Transmit phase number	R/W	0
			Specifies whether the transmit frame has 1 or 2 phases.		
			XPHASE = 0 Single-phase frame.		
			XPHASE = 1 Dual-phase frame.		

8.8 Set the Transmit Word Length(s)

Table 53. Register Bits Used to Set the Transmit Word Length(s)

Register	Bit	Name	Function	Type	Reset
XCR1	7:5	XWDLEN1	Transmit word length of frame phase 1	R/W	000
			XWDLEN1 = 000b 8 bits		
			XWDLEN1 = 001b 12 bits		
			XWDLEN1 = 010b 16 bits		
			XWDLEN1 = 011b 20 bits		
			XWDLEN1 = 100b 24 bits		
			XWDLEN1 = 101b 32 bits		
			XWDLEN1 = 11Xb Reserved		
XCR2	7:5	XWDLEN2	Transmit word length of frame phase 2	R/W	000
			XWDLEN2 = 000b 8 bits		
			XWDLEN2 = 001b 12 bits		
			XWDLEN2 = 010b 16 bits		
			XWDLEN2 = 011b 20 bits		
			XWDLEN2 = 100b 24 bits		
			XWDLEN2 = 101b 32 bits		
			XWDLEN2 = 11Xb Reserved		

8.8.1 Word Length Bits

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, XWDLEN1 selects the length for every serial word transmitted 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.

8.9 Set the Transmit Frame Length

Table 54. Register Bits Used to Set the Transmit Frame Length

Register	Bit	Name	Function	Type	Reset
XCR1	14:8	XFRLEN1	Transmit frame length 1	R/W	000 0000
			(XFRLEN1 + 1) is the number of serial words in phase 1 of the transmit frame.		
			XFRLEN1 = 000 0000 1 word in phase 1		
			XFRLEN1 = 000 0001 2 words in phase 1		
			| |		
			| |		
			XFRLEN1 = 111 1111 128 words in phase 1		
XCR2	14:8	XFRLEN2	Transmit frame length 2	R/W	000 0000
			If a dual-phase frame is selected, (XFRLEN2 + 1) is the number of serial words in phase 2 of the transmit frame.		
			XFRLEN2 = 000 0000 1 word in phase 2		
			XFRLEN2 = 000 0001 2 words in phase 2		
			| |		
			| |		
			XFRLEN2 = 111 1111 128 words in phase 2		

8.9.1 Selected Frame Length

The transmit frame length is the number of serial words in the transmit 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 (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 55 for a summary of how to calculate the frame length. This length corresponds to the number of words or logical time slots or channels per frame-synchronization pulse.

Note:

Program the XFRLEN 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 XFRLEN1.

Table 55. How to Calculate Frame Length

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

8.10 Enable/Disable the Transmit Frame-Synchronization Ignore Function

Table 56. Register Bit Used to Enable/Disable the Transmit Frame-Synchronization Ignore Function

Register	Bit	Name	Function		Type	Reset
XCR2	2	XFIG	Transmit frame-synchronization ignore		R/W	0
			XFIG = 0	An unexpected transmit frame-synchronization pulse causes the McBSP to restart the frame transfer.		
			XFIG = 1	The McBSP ignores unexpected transmit frame-synchronization pulses.		

8.10.1 Unexpected Frame-Synchronization Pulses and Frame-Synchronization Ignore

If a frame-synchronization pulse starts the transfer of a new frame before the current frame is fully transmitted, this pulse is treated as an unexpected frame-synchronization pulse.

When XFIG = 1, normal transmission continues with unexpected frame-synchronization signals ignored.

When XFIG = 0 and an unexpected frame-synchronization pulse occurs, the serial port:

1) Aborts the present transmission.

2) Sets XSYNCERR to 1 in SPCR2.

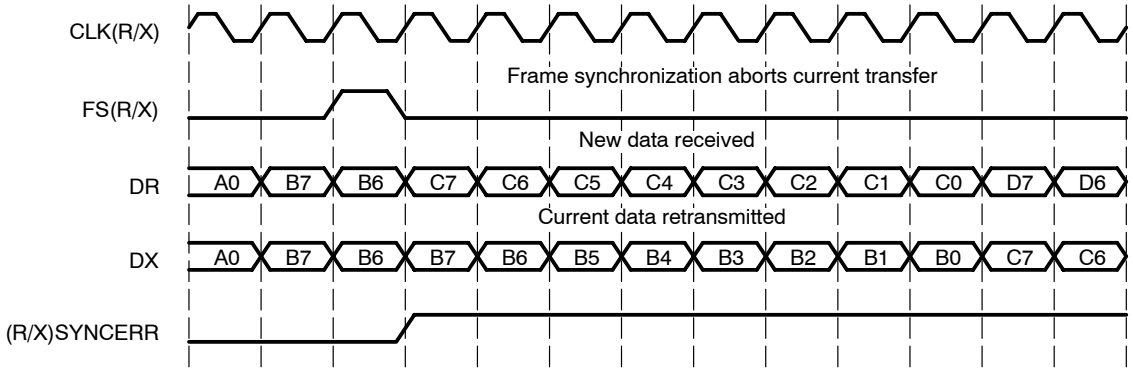
3) Reinitiates transmission of the current word that was aborted.

For more details about the frame-synchronization error condition, see section 4.5, *Unexpected Transmit Frame-Synchronization Pulse*.

8.10.2 Examples Showing the Effects of XFIG

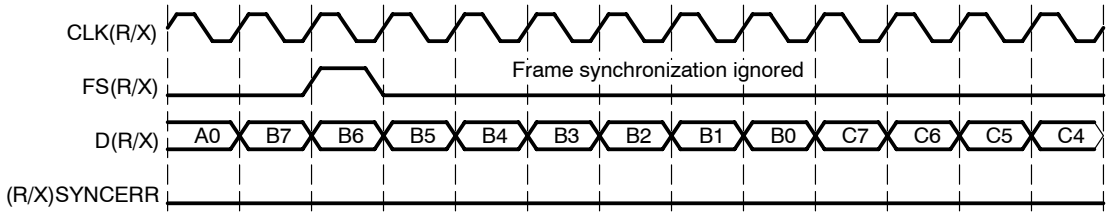
Figure 52 shows an example in which word B is interrupted by an unexpected frame-synchronization pulse when (R/X)FIG = 0. In the case of transmission, the transmission of B is aborted, and B is lost. This condition is a transmit synchronization error, which sets the XSYNCERR bit. No new data has been written to DXR[1,2]; therefore, the McBSP transmits B again.

Figure 52. Unexpected Frame-Synchronization Pulse With (R/X)FIG = 0



In contrast with Figure 52, Figure 53 shows McBSP operation when unexpected frame-synchronization signals are ignored (when (R/X)FIG = 1). Here, the transfer of word B is not affected by an unexpected frame-synchronization pulse.

Figure 53. Unexpected Frame-Synchronization Pulse With (R/X)FIG = 1



8.11 Set the Transmit Companding Mode

Table 57. Register Bits Used to Set the Transmit Companding Mode

Register	Bit	Name	Function		Type	Reset
XCR2	4:3	XCOMPAND	Transmit companding mode		R/W	00
			Modes other than 00b are enabled only when the appropriate XWDLEN is 000b, indicating 8-bit data.			
			XCOMPAND = 00b	No companding, any size data, MSB transmitted first.		
			XCOMPAND = 01b	No companding, 8-bit data, LSB transmitted first (for details, see section 7.11.4, *Option to Receive LSB First*).		
			XCOMPAND = 10b	μ-law companding, 8-bit data, MSB transmitted first.		
			XCOMPAND = 11b	A-law companding, 8-bit data, MSB transmitted first.		

8.11.1 Companding

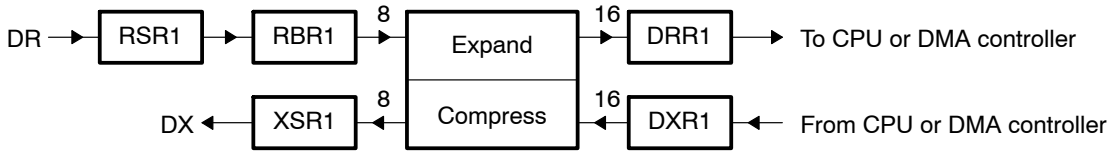
Companding (COMpressing and exPANDing) hardware allows compression and expansion of data in either μ-law or A-law format. The companding standard employed in the United States and Japan is μ-law. The European companding standard is referred to as A-law. The specifications for μ-law and A-law log PCM are part of the CCITT G.711 recommendation.

A-law and μ-law allow 13 bits and 14 bits of dynamic range, respectively. Any 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 via the CPU or DMA controller must be at least 16 bits wide.

The μ-law and A-law formats both encode data into 8-bit code words. Companded data is always 8 bits wide; the appropriate word length bits (RWDLEN1, RWDLEN2, XWDLEN1, XWDLEN2) must therefore be set to 0, indicating an 8-bit wide serial data stream. If companding is enabled and either of the frame phases does not have an 8-bit word length, companding continues as if the word length is 8 bits.

Figure 54 illustrates the companding processes. When companding is chosen for the transmitter, compression occurs during the process of copying data from DXR1 to XSR1. The transmit data is encoded according to the specified companding law (A-law or μ-law). When companding is chosen for the receiver, expansion occurs during the process of copying data from RBR1 to DRR1. The receive data is decoded to twos-complement format.

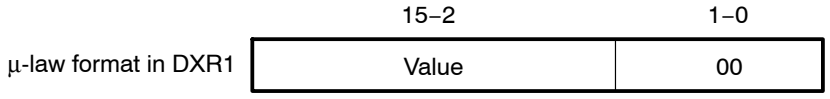
Figure 54. Companding Processes for Reception and for Transmission



8.11.2 Format for Data To Be Compressed

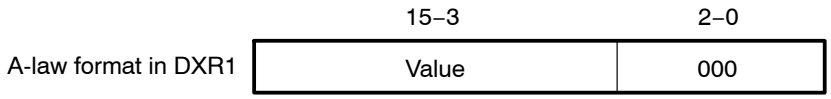
For transmission using μ-law compression, make sure the 14 data bits are left-justified in DXR1, with the remaining two low-order bits filled with 0s as shown in Figure 55.

Figure 55. μ-Law Transmit Data Companding Format



For transmission using A-law compression, make sure the 13 data bits are left-justified in DXR1, with the remaining three low-order bits filled with 0s as shown in Figure 56.

Figure 56. A-Law Transmit Data Companding Format



8.11.3 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. See section 2.2.2, *Capability to Compand Internal Data*.

8.11.4 Option to Transmit LSB First

Normally, the McBSP transmit or receives all data with the most significant bit (MSB) first. However, certain 8-bit data protocols (that do not use companded data) require the least significant bit (LSB) to be transferred first. If you set XCOMPAND = 01b in XCR2, the bit ordering of 8-bit words is reversed (LSB first) before being sent from the serial port. 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.

8.12 Set the Transmit Data Delay

Table 58. Register Bits Used to Set the Transmit Data Delay

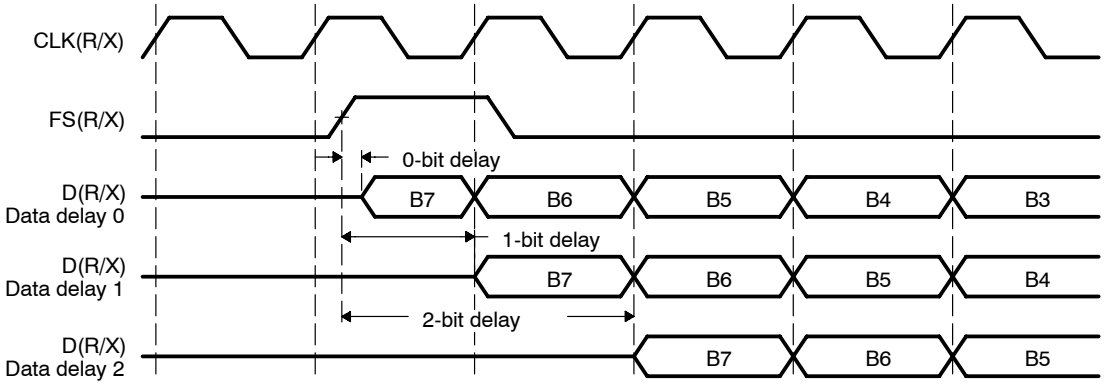
Register	Bit	Name	Function		Type	Reset
XCR2	1:0	XDATDLY	Transmitter data delay		R/W	00
			XDATDLY = 00	0-bit data delay		
			XDATDLY = 01	1-bit data delay		
			XDATDLY = 10	2-bit data delay		
			XDATDLY = 11	Reserved		

8.12.1 Data Delay

The start of a frame is defined by the first clock cycle in which frame synchronization is found to be active. The beginning of actual data reception or transmission with respect to the start of the frame can be delayed if necessary. This delay is called data delay.

XDATDLY specifies the data delay for transmission. The range of programmable data delay is zero to two bit-clocks (XDATDLY = 00b–10b), as described in Table 58 and Figure 57. In this figure, the data transferred is an 8-bit value with bits labeled B7, B6, B5, and so on. Typically a 1-bit delay is selected, because data often follows a 1-cycle active frame-synchronization pulse.

Figure 57. Range of Programmable Data Delay



8.12.2 0-Bit Data Delay

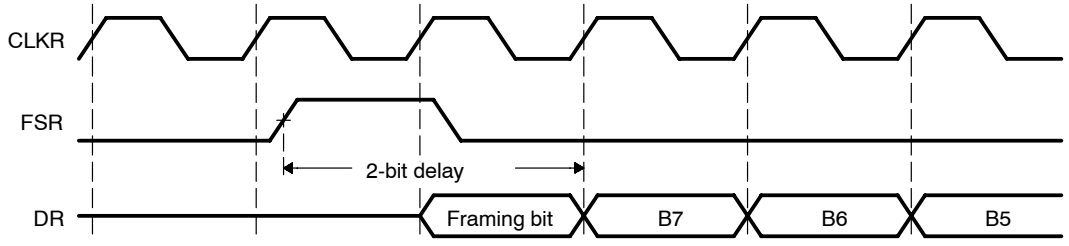
Normally, a frame-synchronization pulse is detected or sampled with respect to an edge of serial clock internal CLK(R/X). Thus, on the following cycle or later (depending on the data delay value), data can be received or transmitted. However, in the case of 0-bit data delay, the data must be ready for reception and/or transmission on the same serial clock cycle.

For reception this problem is solved, because receive data is sampled on the first falling edge of CLKR where an active-high internal FSR is detected. However, data transmission must begin on the rising edge of the internal CLKX clock that generated the frame synchronization. Therefore, the first data bit is assumed to be present in XSR1, and thus DX. The transmitter then asynchronously detects the frame synchronization, FSX, going active high and immediately starts driving the first bit to be transmitted on the DX pin.

8.12.3 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 the following figure. In this figure, the data transferred is an 8-bit value with bits labeled B7, B6, B5, and so on.

Figure 58. 2-Bit Data Delay Used to Skip a Framing Bit



8.13 Set the Transmit DXENA Mode

Table 59. Register Bit Used to Set the Transmit DXENA (DX Delay Enabler) Mode

Register	Bit	Name	Function		Type	Reset
SPCR1	7	DXENA	DX delay enabler mode		R/W	0
			DXENA = 0	DX delay enabler is off.		
			DXENA = 1	DX delay enabler is on.		

8.13.1 DXENA Mode

The DXENA bit controls the delay enabler on the DX pin. Set DXENA to enable an extra delay for turn-on time (for the length of the delay, see the data sheet for your TMS320C55x DSP). This bit does not control the data itself, so only the first bit is delayed.

If you tie together the DX pins of multiple McBSPs, make sure DXENA = 1 to avoid having more than one McBSP transmit on the data line at one time.

8.14 Set the Transmit Interrupt Mode

The transmitter interrupt (XINT) signals the CPU of changes to the serial port status. Four options exist for configuring this interrupt. The options are set by the transmit interrupt mode bits, XINTM, in SPCR2.

Table 60. Register Bits Used to Set the Transmit Interrupt Mode

Register	Bit	Name	Function		Type	Reset
SPCR2	5:4	XINTM	Transmit interrupt mode		R/W	00
			XINTM = 00	XINT generated when XRDY changes from 0 to 1.		
			XINTM = 01	XINT generated by an end-of-block or end-of-frame condition in a transmit multichannel selection mode. In any of the transmit multichannel selection modes, interrupt after every 16-channel block boundary has been crossed within a frame and at the end of the frame. For details, see section 5.8, *Using Interrupts Between Block Transfers*. In any other serial transfer case, this setting is not applicable and, therefore, no interrupts are generated.		
			XINTM = 10	XINT generated by a new transmit frame-synchronization pulse. Interrupt on detection of each transmit frame-synchronization pulse. This generates an interrupt even when the transmitter is in its reset state. This is done by synchronizing the incoming frame-synchronization pulse to the CPU clock and sending it to the CPU via XINT.		
			XINTM = 11	XINT generated when XSYNCERR is set. Interrupt on frame-synchronization error. Regardless of the value of XINTM, XSYNCERR can be read to detect this condition. For more information on using XSYNCERR, see section 4.5, *Unexpected Transmit Frame-Synchronization Pulse*.		

8.15 Set the Transmit Frame-Synchronization Mode

Table 61. Register Bits Used to Set the Transmit Frame-Synchronization Mode

Register	Bit	Name	Function		Type	Reset
PCR	11	FSXM	Transmit frame-synchronization mode		R/W	0
			FSXM = 0	Transmit frame synchronization is supplied by an external source via the FSX pin.		
			FSXM = 1	Transmit frame synchronization is supplied by the McBSP, as determined by the FSGM bit of SRGR2.		
SRGR2	12	FSGM	Sample rate generator transmit frame-synchronization mode		R/W	0
			Used when FSXM = 1 in PCR.			
			FSGM = 0	The McBSP generates a transmit frame-synchronization pulse when the content of DXR[1,2] is copied to XSR[1,2].		
			FSGM = 1	The transmitter uses frame-synchronization pulses generated by the sample rate generator. Program the FWID bits to set the width of each pulse. Program the FPER bits to set the frame-synchronization period.		

8.15.1 Transmit Frame-Synchronization Modes

Table 62 shows how FSXM and FSGM select the source of transmit frame-synchronization pulses. The three choices are:

- ❏ External frame-synchronization input
- ❏ Sample rate generator frame-synchronization signal (FSG)
- ❏ Internal signal that indicates a DXR-to-XSR copy has been made

Table 62 also shows the effect of each bit setting on the FSX pin. The FSXP bit determines the polarity of the signal on the FSX pin.

Table 62. How FSXM and FSGM Select the Source of Transmit Frame-Synchronization Pulses

FSXM	FSGM	Source of Transmit Frame Synchronization	FSX Pin Status
0	0 or 1	An external frame-synchronization signal enters the McBSP through the FSX pin. The signal is then inverted by FSXP before being used as internal FSX.	Input.
1	1	Internal FSX is driven by the sample rate generator frame-synchronization signal (FSG).	Output. FSG is inverted by FSXP before being driven out on FSX pin.
1	0	A DXR-to-XSR copy causes the McBSP to generate a transmit frame-synchronization pulse that is 1 cycle wide.	Output. The generated frame-synchronization pulse is inverted as determined by FSXP before being driven out on FSX pin.

8.15.2 Other Considerations

If the sample rate generator creates a frame-synchronization signal (FSG) that is derived from an external input clock, the GSYNC bit determines whether FSG is kept synchronized with pulses on the FSR pin. For more details, see section 3.3, *Synchronizing Sample Rate Generator 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 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 FSX pin.

8.16 Set the Transmit Frame-Synchronization Polarity

Table 63. Register Bit Used to Set Transmit Frame-Synchronization Polarity

Register	Bit	Name	Function		Type	Reset
PCR	3	FSXP	Transmit frame-synchronization polarity		R/W	0
			FSXP = 0	Frame-synchronization pulse FSX is active high.		
			FSXP = 1	Frame-synchronization pulse FSX is active low.		

8.16.1 Frame Synchronization Pulses, Clock Signals, and Their Polarities

Transmit frame-synchronization pulses can be generated internally by the sample rate generator (see section 3.2) or driven by an external source. The source of frame synchronization is selected by programming the mode bit, FSXM, in PCR. FSX is also affected by the FSGM bit in SRGR2. For information about the effects of FSXM and FSGM, see section 8.15, *Set the Transmit Frame-Synchronization Mode*. Similarly, transmit clocks can be selected to be inputs or outputs by programming the mode bit, CLKXM, in the PCR (see section 8.18, *Set the Transmit Clock Mode*).

When FSR and FSX are inputs (FSXM = FSRM = 0, external frame-synchronization pulses), the McBSP detects them on the internal falling edge of clock, internal CLKR, and internal CLKX, respectively. The receive data arriving at the DR pin is also sampled on the falling edge of internal CLKR. These internal clock signals are either derived from external source via CLK(R/X) pins, or driven by the sample rate generator clock (CLKG) internal to the McBSP.

When FSR and FSX are outputs, implying that they are driven by the sample rate generator, they are generated (transition to their active state) on the rising edge of internal clock, CLK(R/X). Similarly, data on the DX pin is output on the rising edge of internal CLKX.

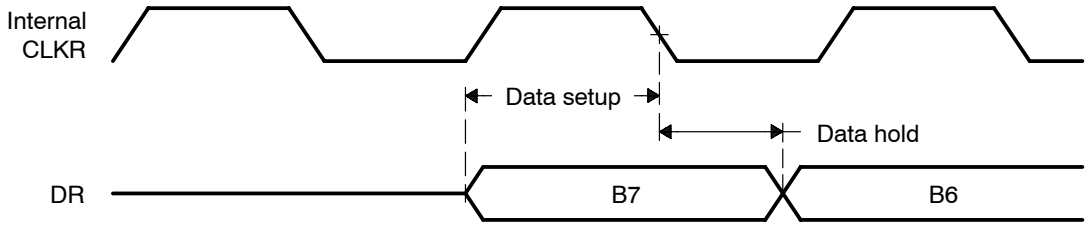
FSRP, FSXP, CLKRP, and CLKXP in the pin control register (PCR) configure the polarities of the FSR, FSX, CLKR, and CLKX signals, respectively. All frame-synchronization signals (internal FSR, internal FSX) that are internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to McBSP) and FSRP = FSXP = 1, the external active-low frame-synchronization signals are inverted before being sent to the receiver (internal FSR) and transmitter (internal FSX). Similarly, if internal synchronization is selected (FSR/FSX are output pins and GSYNC = 0) and the polarity bit FS(R/X)P = 1, the internal active-high frame-synchronization signals 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 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 CLKR pin.

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 59 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 59. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge



8.17 Set the SRG Frame-Synchronization Period and Pulse Width

Table 64. Register Bits Used to Set SRG Frame-Synchronization Period and Pulse Width

Register	Bit	Name	Function	Type	Reset
SRGR2	11:0	FPER	Sample rate generator frame-synchronization period For the frame-synchronization signal FSG, (FPER + 1) determines the period from the start of a frame-synchronization pulse to the start of the next frame-synchronization pulse. Range for (FPER + 1): 1 to 4096 CLKG cycles.	R/W	0000 0000 0000
SRGR1	15:8	FWID	Sample rate generator frame-synchronization pulse width This field plus 1 determines the width of each frame-synchronization pulse on FSG. Range for (FWID + 1): 1 to 256 CLKG cycles.	R/W	0000 0000

8.17.1 Frame-Synchronization Period and Frame-Synchronization Pulse Width

The sample rate generator can produce a clock signal, CLKG, and a frame-synchronization signal, FSG. If the sample rate generator is supplying receive or transmit frame synchronization, you must program the bit fields FPER and FWID.

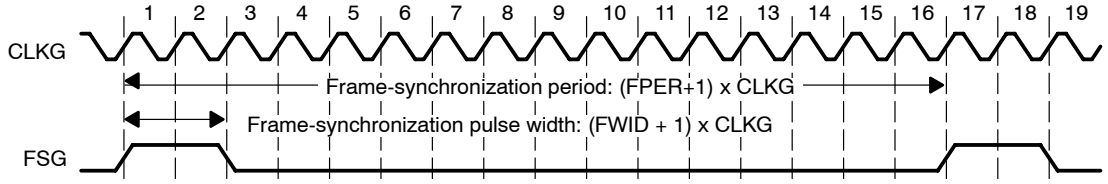
On FSG, the period from the start of a frame-synchronization pulse to the start of the next pulse is (FPER + 1) CLKG cycles. The 12 bits of FPER allow a frame-synchronization period of 1 to 4096 CLKG cycles, which allows up to 4096 data bits per frame. When GSYNC = 1, FPER is a don't care value.

Each pulse on FSG has a width of (FWID + 1) CLKG cycles. The eight bits of FWID allow a pulse width of 1 to 256 CLKG cycles. It is recommended that FWID be programmed to a value less than the programmed word length.

The values in FPER and FWID are loaded into separate down-counters. The 12-bit FPER counter counts down the generated clock cycles from the programmed value (4095 maximum) to 0. The 8-bit FWID counter counts down from the programmed value (255 maximum) to 0.

Figure 60 shows a frame-synchronization period of 16 CLKG periods (FPER = 15 or 00001111b) and a frame-synchronization pulse with an active width of 2 CLKG periods (FWID = 1).

Figure 60. Frame of Period 16 CLKG Periods and Active Width of 2 CLKG Periods



When the sample rate generator comes out of reset, FSG is in its inactive state. Then, when $\overline{\text{GRST}}$ = 1 and FSGM = 1, a frame-synchronization pulse is generated. The frame width value (FWID + 1) is counted down on every CLKG cycle until it reaches 0, at which time FSG goes low. At the same time, the frame period value (FPER + 1) is also counting down. When this value reaches 0, FSG goes high, indicating a new frame.

8.18 Set the Transmit Clock Mode

Table 65. Register Bit Used to Set the Transmit Clock Mode

Register	Bit	Name	Function		Type	Reset
PCR	9	CLKXM	Transmit clock mode		R/W	0
			CLKXM = 0	The transmitter gets its clock signal from an external source via the CLKX pin.		
			CLKXM = 1	The CLKX pin is an output pin driven by the sample rate generator of the McBSP.		

8.18.1 Selecting a Source for the Transmit Clock and a Data Direction for the CLKX Pin

Table 66 shows how the CLKXM bit selects the transmit clock and the corresponding status of the CLKX pin. The polarity of the signal on the CLKX pin is determined by the CLKXP bit.

Table 66. How the CLKXM Bit Selects the Transmit Clock and the Corresponding Status of the CLKX Pin

CLKXM in PCR	Source of Transmit Clock	CLKX Pin Status
0	Internal CLKX is driven by an external clock on the CLKX pin. CLKX is inverted as determined by CLKXP before being used.	Input.
1	Internal CLKX is driven by the sample rate generator clock, CLKG.	Output. CLKG, inverted as determined by CLKXP, is driven out on CLKX.

8.18.2 Other Considerations

If the sample rate generator 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 FSR pin. For more details, see section 3.3, *Synchronizing Sample Rate Generator 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, 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.

8.19 Set the Transmit Clock Polarity

Table 67. Register Bit Used to Set Transmit Clock Polarity

Register	Bit	Name	Function		Type	Reset
PCR	1	CLKXP	Transmit clock polarity		R/W	0
			CLKXP = 0	Transmit data sampled on rising edge of CLKX.		
			CLKXP = 1	Transmit data sampled on falling edge of CLKX.		

8.19.1 Frame Synchronization Pulses, Clock Signals, and Their Polarities

Transmit frame-synchronization pulses can be either generated internally by the sample rate generator (see section 3.2) or driven by an external source. The source of frame synchronization is selected by programming the mode bit, FSXM, in PCR. FSX is also affected by the FSGM bit in SRGR2. For information about the effects of FSXM and FSGM, see section 8.15, *Set the Transmit Frame-Synchronization Mode*). Similarly, transmit clocks can be selected to be inputs or outputs by programming the mode bit, CLKXM, in the PCR (see section 8.18, *Set the Transmit Clock Mode*).

When FSR and FSX are inputs (FSXM = FSRM = 0, external frame-synchronization pulses), the McBSP detects them on the internal falling edge of clock, internal CLKR, and internal CLKX, respectively. The receive data arriving at the DR pin is also sampled on the falling edge of internal CLKR. These internal clock signals are either derived from external source via CLK(R/X) pins or driven by the sample rate generator clock (CLKG) internal to the McBSP.

When FSR and FSX are outputs, implying that they are driven by the sample rate generator, they are generated (transition to their active state) on the rising edge of internal clock, CLK(R/X). Similarly, data on the DX pin is output on the rising edge of internal CLKX.

FSRP, FSXP, CLKRP, and CLKXP in the pin control register (PCR) configure the polarities of the FSR, FSX, CLKR, and CLKX signals, respectively. All frame-synchronization signals (internal FSR, internal FSX) that are internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to McBSP), and FSRP = FSXP = 1, the external active-low frame-synchronization signals are inverted before being sent to the receiver (internal FSR) and transmitter (internal FSX). Similarly, if internal synchronization is selected (FSR/FSX are output pins and GSYNC = 0), the internal active-high frame-synchronization signals are inverted, if the polarity bit FS(R/X)P = 1, before being sent to the FS(R/X) pin.

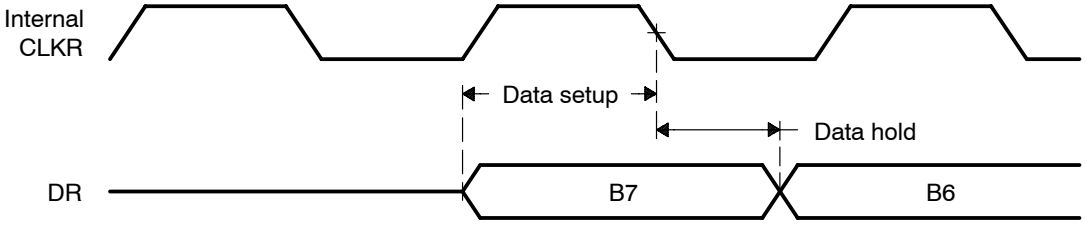
On the transmit side, the transmit clock polarity bit, CLKXP, sets the edge used to shift and clock out transmit data. Data is always transmitted on the rising edge of internal CLKX. If CLKXP = 1 and external clocking is selected (CLKXM = 0 and CLKX is an input), the external falling-edge triggered input 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 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 CLKR pin.

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 (see Figure 59).

Figure 61 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 61. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge



8.20 Set the SRG Clock Divide-Down Value

Table 68. Register Bits Used to Set Sample Rate Generator (SRG) Clock Divide-Down Value

Register	Bit	Name	Function	Type	Reset
SRGR1	7:0	CLKGDV	Sample rate generator clock divide-down value The input clock of the sample rate generator is divided by (CLKGDV + 1) to generate the required sample rate generator clock frequency. The default value of CLKGDV is 1 (divide input clock by 2).	R/W	0000 0001

8.20.1 Sample Rate Generator Clock Divider

The first divider stage generates the serial data bit clock from the input clock. This divider stage utilizes a counter, preloaded by CLKGDV, that contains the divide ratio value.

The output of the first divider stage is the data bit clock, which is output as CLKG and which serves as the input for the second and third stages of the divider.

CLKG has a frequency equal to 1/(CLKGDV + 1) of sample rate generator 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.

8.21 Set the SRG Clock Synchronization Mode

Table 69. Register Bit Used to Set the SRG Clock Synchronization Mode

Register	Bit	Name	Function	Type	Reset
SRGR2	15	GSYNC	Sample rate generator clock synchronization GSYNC is used only when the input clock source for the sample rate generator is external– on the CLKS, CLKR, or CLKX pin. GSYNC = 0: The sample rate generator clock (CLKG) is free running. CLKG oscillates without adjustment, and FSG pulses every (FPER + 1) CLKG cycles. GSYNC = 1: Clock synchronization is performed. When a pulse is detected on the FSR pin: ❑ CLKG is adjusted as necessary so that it is synchronized with the input clock on the CLKS, CLKR, or CLKX pin. ❑ FSG pulses. FSG only pulses in response to a pulse on the FSR pin. The frame-synchronization period defined in FPER is ignored.	R/W	0

For more details on using the clock synchronization feature, see section 3.3, *Synchronizing Sample Rate Generator Outputs to an External Clock*.

8.22 Set the SRG Clock Mode (Choose an Input Clock)

Table 70. Register Bits Used to Set the SRG Clock Mode (Choose an Input Clock)

Register	Bit	Name	Function		Type	Reset
PCR SRGR2	7 13	SCLKME CLKSM	Sample rate generator clock mode		R/W R/W	0 1
			SCLKME = 0 CLKSM = 0	Sample rate generator clock derived from CLKS pin.		
			SCLKME = 0 CLKSM = 1	Sample rate generator clock derived from CPU clock. (This is the condition forced by an OMAP5912 reset.)		
			SCLKME = 1 CLKSM = 0	Sample rate generator clock derived from CLKR pin.		
			SCLKME = 1 CLKSM = 1	Sample rate generator clock derived from CLKX pin.		

8.22.1 SRG Clock Mode

The sample rate generator can produce a clock signal (CLKG) for use by the receiver, the transmitter, or both, but CLKG is derived from an input clock. The preceding table shows the four possible sources of the input clock. For more details on generating CLKG, see section 3.1, *Clock Generation in the Sample Rate Generator*.

8.23 Set the SRG Input Clock Polarity

Table 71. Register Bits Used to Set the SRG Input Clock Polarity

Register	Bit	Name	Function		Type	Reset
SRGR2	14	CLKSP	CLKS pin polarity. CLKSP determines the input clock polarity when the CLKS pin supplies the input clock (SCLKME = 0 and CLKSM = 0).		R/W	0
			CLKSP = 0	Rising edge on CLKS pin generates CLKG and FSG.		
			CLKSP = 1	Falling edge on CLKS pin generates CLKG and FSG.		
PCR	1	CLKXP	CLKX pin polarity. CLKXP determines the input clock polarity when the CLKX pin supplies the input clock (SCLKME = 1 and CLKSM = 1).		R/W	0
			CLKXP = 0	Rising edge on CLKX pin generates transitions on CLKG and FSG.		
			CLKXP = 1	Falling edge on CLKX pin generates transitions on CLKG and FSG.		
PCR	0	CLKRP	CLKR pin polarity. CLKRP determines the input clock polarity when the CLKR pin supplies the input clock (SCLKME = 1 and CLKSM = 0).		R/W	0
			CLKRP = 0	Falling edge on CLKR pin generates transitions on CLKG and FSG.		
			CLKRP = 1	Rising edge on CLKR pin generates transitions on CLKG and FSG.		

8.23.1 Using CLKSP/CLKXP/CLKRP to Choose an Input Clock Polarity

The sample rate generator can produce a clock signal (CLKG) and a frame-synchronization signal (FSG) for use by the receiver, the transmitter, or both. To produce CLKG and FSG, the sample rate generator must be driven by an input clock signal derived from the CPU clock or from an external clock on the CLKS, CLKX, or CLKR pin. If you use a pin, choose a polarity for that pin by using the appropriate polarity bit (CLKSP for the CLKS pin, CLKXP for the CLKX pin, CLKRP for the CLKR pin). The polarity determines whether the rising or falling edge of the input clock generates transitions on CLKG and FSG.

9 General-Purpose I/O on the McBSP Pins

Table 72 summarizes how to use the McBSP pins as general-purpose I/O pins. All of the bits mentioned in the table except $\overline{\text{XRST}}$ and $\overline{\text{RRST}}$ are in the pin control register (PCR). $\overline{\text{XRST}}$ and $\overline{\text{RRST}}$ are in the serial port control registers (SPCRs).

To use receiver pins CLKR, FSR, and DR as general-purpose I/O pins rather than as serial port pins, you must meet two conditions:

- The receiver of the serial port is in reset ($\overline{\text{RRST}}$ = 0 in SPCR1).
- General-purpose I/O is enabled for the serial port receiver (RIOEN = 1 in PCR).

The CLKR and FSR pins can be individually configured as either input or output pins with the CLKRM and FSRM bits, respectively. The DR pin can only be an input pin. Table 72 shows which bits in PCR are used to read from/write to these pins.

For the transmitter pins CLKX, FSX, and DX, you must meet two conditions:

- The transmitter of the serial port is in reset ($\overline{\text{XRST}}$ = 0 in SPCR2).
- General-purpose I/O is enabled for the serial port transmitter (XIOEN = 1 in PCR).

The CLKX and FSX pins can be individually configured as input or output pins with the CLKXM and FSXM bits, respectively. The DX pin can only be an output pin. Table 72 shows which bits in PCR are used to read from/write to these pins.

For the 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 ($\overline{\text{RRST}}$ = 0 and $\overline{\text{XRST}}$ = 0).
- General-purpose I/O is enabled for both the receiver and the transmitter (RIOEN = 1 and XIOEN = 1).

The CLKS pin can only be an input pin. To read the status of the signal on the CLKS pin, read the CLKS_STAT bit in PCR.

Table 72. Using McBSP Pins for General-Purpose Input/Output

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
CLKX	$\overline{\text{XRST}}$ = 0 XIOEN = 1	CLKXM = 1	CLKXP	CLKXM = 0	CLKXP
FSX	$\overline{\text{XRST}}$ = 0 XIOEN = 1	FSXM = 1	FSXP	FSXM = 0	FSXP
DX	$\overline{\text{XRST}}$ = 0 XIOEN = 1	Always	DX_STAT	Never	Does not apply
CLKR	$\overline{\text{RRST}}$ = 0 RIOEN = 1	CLKRM = 1	CLKRP	CLKRM = 0	CLKRP
FSR	$\overline{\text{RRST}}$ = 0 RIOEN = 1	FSRM = 1	FSRP	FSRM = 0	FSRP
DR	$\overline{\text{RRST}}$ = 0 RIOEN = 1	Never	Does not apply	Always	DR_STAT
CLKS	$\overline{\text{RRST}}$ = $\overline{\text{XRST}}$ = 0 RIOEN = XIOEN = 1	Never	Does not apply	Always	CLKS_STAT

10 Emulation, Power, and Reset Considerations

This section covers the following topics:

- How to program McBSP response to a breakpoint in the high-level language debugger (see section 10.1)
- How to conserve power in the OMAP5912 by placing the McBSP into its idle mode (section 10.2)
- How to reset and initialize the various parts of the McBSP (see section 10.3)

10.1 McBSP Emulation Mode

FREE and SOFT are special emulation bits in SPCR2 that determine the state of the McBSP when a breakpoint is encountered in the high-level language debugger. If FREE = 1, the clock continues to run upon a software breakpoint and data is still shifted out. When FREE = 1, the SOFT bit is a *don't care.*

If FREE = 0, the SOFT bit takes effect. If SOFT = 0 when breakpoint occurs, the clock stops immediately, aborting a transmission. If SOFT = 1 and a breakpoint occurs while transmission is in progress, the transmission continues until completion of the transfer and then the clock halts. These options are listed in Table 73.

The McBSP receiver functions in a similar fashion. If a mode other than the immediate stop mode (SOFT = FREE = 0) is chosen, the receiver continues running and an overrun error is possible.

Table 73. McBSP Emulation Modes Selectable with FREE and SOFT Bits of SPCR2

FREE	SOFT	McBSP Emulation Mode
0	0	Immediate stop mode (reset condition) The transmitter or receiver stops immediately in response to a breakpoint.
0	1	Soft stop mode When a breakpoint occurs, the transmitter stops after completion of the current word. The receiver is not affected.
1	0 or 1	Free run mode The transmitter and receiver continue to run when a breakpoint occurs.

10.2 Reducing Power Consumed by McBSPs

The McBSP is placed into an idle mode with reduced power consumption when the PERIPH idle domain is idle (PERIS = 1 in ISTR) and the McBSP idle enable bit is set (IDLE_EN = 1 in PCR).

In the McBSP idle mode:

- If the McBSP is configured to operate with internally generated clocking and frame synchronization, it is completely stopped.
- If the McBSP is configured to operate with externally generated clocking and frame synchronization (either directly or through the sample rate generator), the external interface portion of the McBSP continues to function during external clock activity periods. The McBSP sends a request to activate the PERIPH and DMA idle domains when it needs to be serviced. If the domains were idle, they are made idle again after the McBSP has been serviced.

When IDLE_EN = 0 in PCR, the McBSP keeps running, regardless of whether the PERIPH domain is idle.

10.3 Resetting and Initializing McBSPs

10.3.1 McBSP Pin States: OMAP5912 Reset Versus Receiver/Transmitter Reset

Table 74 shows the state of McBSP pins when the serial port is reset due to direct receiver or transmitter reset on the OMAP5912 device.

Table 74. Reset State of Each McBSP Pin

Pin	Possible State(s)	State Forced by OMAP5912 Reset	State Forced by Receiver/Transmitter Reset
Receiver Reset ($\overline{RRST}$ = 0 and $\overline{GRST}$ = 1)			
DR	I	Input	Input
CLKR	I/O/Z	Input	Known state if input; CLKR running if output
FSR	I/O/Z	Input	Known state if input; FSRP inactive state if output
CLKS	I/O/Z	Input	Input
Transmitter Reset ($\overline{XRST}$ = 0 and $\overline{GRST}$ = 1)			
DX	O/Z	High impedance	High impedance
CLKX	I/O/Z	Input	Known state if input; CLKX running if output
FSX	I/O/Z	Input	Known state if input; FSXP inactive state if output

Note: In Possible State(s) column, I = input, O = output, Z = high impedance

10.3.2 OMAP5912 Reset, McBSP Reset, and Sample Rate Generator Reset

When the McBSP is reset in either of the above two ways, the machine is reset to its initial state, including the reset of all counters and status bits. The receive status bits include RFULL, RRDY, and RSYNCERR. The transmit status bits include XEMPTY_, XRDY, and XSYNCERR.

- OMAP5912 reset. When the whole DSP is reset ($\overline{RESET}$ signal is driven low), the entire serial port, including the transmitter, receiver, and the sample rate generator, is reset. All input-only pins and 3-state pins must be in a known state. The output-only pin DX is in the high-impedance state.

  The OMAP5912 reset forces the sample rate generator clock, CLKG, to have half the frequency of the CPU clock. No pulses are generated on the sample rate generator's frame-synchronization signal, FSG.

  When the device is pulled out of reset, the serial port remains in the reset state. In this state the DR and DX pins can be used as general-purpose I/O pins as described in section 9, *General-Purpose I/O on the McBSP Pins*.

- McBSP reset. When the receiver and transmitter reset bits, $\overline{RRST}$ and $\overline{XRST}$, are loaded with 0s, the respective portions of the McBSP are reset and activity in the corresponding section of the serial port stops. All input-only pins, such as DR and CLKS, and all other pins that are configured as inputs are in a known state. The FSR and FSX pins are driven to their inactive state, if they are not outputs. If the CLKR and CLKX pins are programmed as outputs, they are driven by CLKG, provided that $\overline{GRST}$ = 1. Lastly, the DX pin is in the high-impedance state when the transmitter and/or the device is reset.

During normal operation, the sample rate generator is reset if the $\overline{\text{GRST}}$ bit is cleared. $\overline{\text{GRST}}$ must be 0 only when neither the transmitter nor the receiver is using the sample rate generator. In this case, the internal sample rate generator clock (CLKG) and its frame-synchronization signal (FSG) are driven inactive low.

When the sample rate generator is not in the reset state ($\overline{\text{GRST}}$ = 1), pins FSR and FSX are in an inactive state when $\overline{\text{RRST}}$ = 0 and $\overline{\text{XRST}}$ = 0, respectively, even if they are outputs driven by FSG. This ensures that when only one portion of the McBSP is in reset, the other portion can continue operation when $\overline{\text{GRST}}$ = 1, and its frame synchronization is driven by FSG.

- Sample rate generator reset. The sample rate generator is reset when the DSP is reset or when $\overline{\text{GRST}}$ is loaded with 0. In the case of an OMAP5912 reset, the sample rate generator clock, CLKG, is driven by the CPU clock divided by 2 and the frame-synchronization signal, FSG, is driven inactive low.

  When neither the transmitter nor the receiver is fed by CLKG and FSG, you can reset the sample rate generator by clearing $\overline{\text{GRST}}$. In this case, CLKG and FSG are driven inactive low. If you then set $\overline{\text{GRST}}$, CLKG starts and runs as programmed. Later, if $\overline{\text{GRST}}$ = 1, FSG pulses active high after the programmed number of CLKG cycles has elapsed.

10.3.3 McBSP Initialization Procedure

The serial port initialization procedure is as follows:

1) Make $\overline{\text{XRST}}$ = $\overline{\text{RRST}}$ = $\overline{\text{GRST}}$ = 0 in SPCR[1,2]. If coming out of an OMAP5912 reset, this step is not required.
2) While the serial port is in the reset state, program only the McBSP configuration registers (not the data registers) as required.
3) Wait for two clock cycles. This ensures proper internal synchronization.
4) Set up data acquisition as required (such as writing to DXR[1,2]).
5) Make $\overline{\text{XRST}}$ = $\overline{\text{RRST}}$ = 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 SPCR1 and SPCR2. Otherwise, you change the configuration you selected in step 2.
6) Set $\overline{\text{FRST}}$ = 1, if internally generated frame synchronization is required.
7) Wait two clock cycles for the receiver and transmitter to become active.

Alternatively, on either write (step 1 or 5), the transmitter and receiver can be placed in or taken out of reset individually 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 when the sample rate generator is not used for either operation.

Notes:

1) The necessary duration of the active-low period of $\overline{\text{XRST}}$ or $\overline{\text{RRST}}$ is at least two CLKR/CLKX cycles.

2) The appropriate bits in serial port configuration registers SPCR[1,2], PCR, RCR[1,2], XCR[1,2], and SRGR[1,2] must only be modified when the affected portion of the serial port is in its reset state.

3) In most cases, the data transmit registers (DXR[1,2]) must be loaded by the CPU or by the DMA controller only when the transmitter is enabled ($\overline{\text{XRST}}$ = 1). An exception to this rule is when these registers are used for companding internal data (see section 2.2.2, *Capability to Compand Internal Data*).

4) The bits of the channel control registers (MCR[1,2], RCER[A-H], XCER[A-H]) can be modified at any time as long as they are not being used by the current reception/transmission in a multichannel selection mode.

10.3.4 Resetting the Transmitter While the Receiver is Running

Example 1 shows values in the control registers that reset and configure the transmitter while the receiver is running.

Example 1. Resetting and Configuring McBSP Transmitter While McBSP Receiver Running

```
SPCR1 = 0001h      ; The receiver is running with the receive
SPCR2 = 0030h      ; interrupt (RINT) triggered by the
                   ; receiver ready bit (RRDY). The
                   ; transmitter is in its reset state. The
                   ; transmit interrupt (XINT) will be
                   ; triggered by the transmit frame-sync
                   ; error bit (XSYNCERR).

PCR = 0900h        ; Transmit frame synchronization is
                   ; generated internally according to the
                   ; FSGM bit of SRGR2. The transmit clock
                   ; is driven by an external source. The
                   ; receive clock continues to be driven by
                   ; sample rate generator. The input clock
                   ; of the sample rate generator is supplied
                   ; by the CLKS pin or by the CPU clock
                   ; depending on the CLKSM bit of SRGR2.

SRGR1 = 0001h      ; The CPU clock is the input clock for
SRGR2 = 2000h      ; the sample rate generator. The sample
                   ; rate generator divides the CPU clock by
                   ; 2 to generate its output clock (CLKG).
                   ; Transmit frame synchronization is tied
                   ; to the automatic copying of data from
                   ; the DXR(s) to the XSR(s).

XCR1 = 0740h       ; The transmit frame has two phases.
XCR2 = 8321h       ; Phase 1 has eight 16-bit words. Phase 2
                   ; has four 12-bit words. There is 1-bit
                   ; data delay between the start of a
                   ; frame-sync pulse and the first data bit
                   ; transmitted.

SPCR2 = 0x0031     ; The transmitter is taken out of reset.
```

11 Data Packing Examples

This section shows two ways to implement data packing in the McBSP.

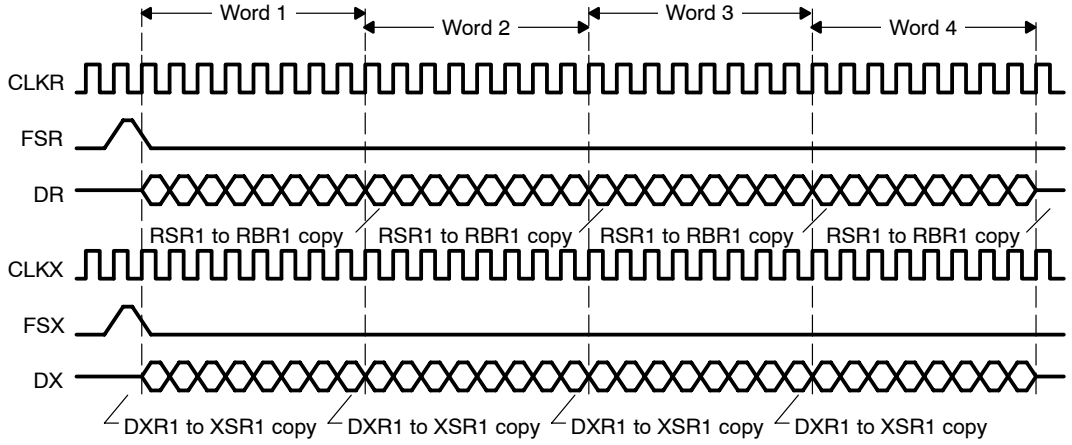
11.1 Data Packing Using Frame Length and Word Length

Frame length and word length can be manipulated to effectively pack data. For example, consider a situation where four 8-bit words are transferred in a single-phase frame as shown in Figure 62. 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 DRR1 and four writes to DXR1 are necessary for each frame.

Figure 62. Four 8-Bit Data Words Transferred To/From the McBSP



This data can also be treated as a single-phase frame consisting of one 32-bit data word, as shown in Figure 63. In this case:

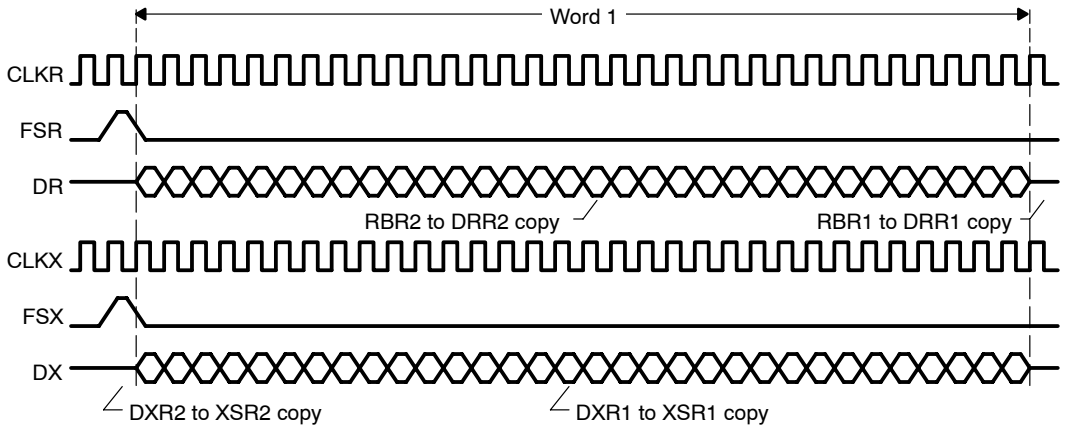
- (R/X)PHASE = 0: Single-phase frame
- (R/X)FRLEN1 = 0000000b: 1-word frame
- (R/X)WDLEN1 = 101b: 32-bit word

Two 16-bit data words are transferred to and from the McBSP by the CPU or DMA controller. Thus, two reads, from DRR2 and DRR1, and two writes, to DXR2 and DXR1, 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 DRR2/DXR2 before you access DRR1/DXR1. McBSP activity is tied to accesses of DRR1/DXR1. During the reception of 24-bit or 32-bit words, read DRR2 and then read DRR1. Otherwise, the next RBR[1,2]-to-DRR[1,2] copy occurs before DRR2 is read. Similarly, during the transmission of 24-bit or 32-bit words, write to DXR2 and then write to DXR1. Otherwise, the next DXR[1,2]-to-XSR[1,2] copy occurs before DXR2 is loaded with new data.

Figure 63. One 32-Bit Data Word Transferred To/From the McBSP



11.2 Data Packing Using Word Length and the Frame-Synchronization 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-synchronization pulses. First, consider Figure 64, which shows the McBSP operating at the maximum packet frequency. Here, each frame only has a single 8-bit word. Notice the frame-synchronization 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 64. 8-Bit Data Words Transferred at Maximum Packet Frequency

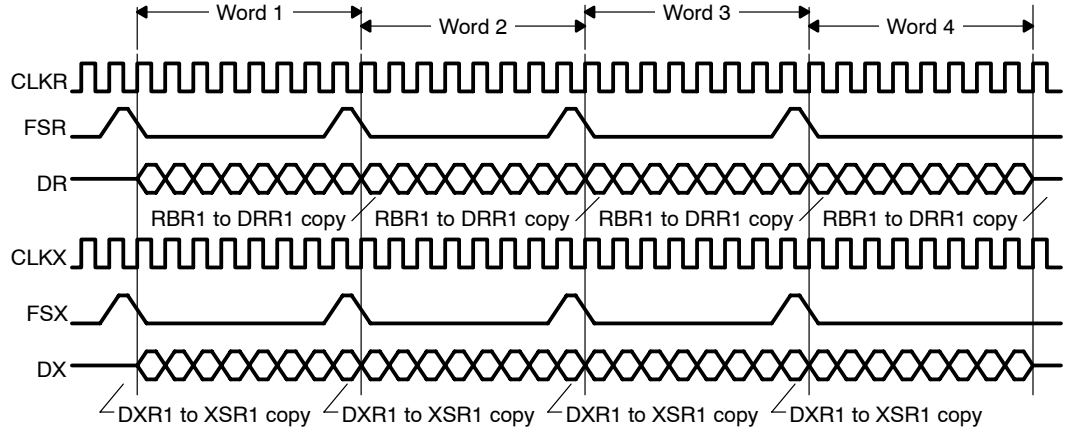
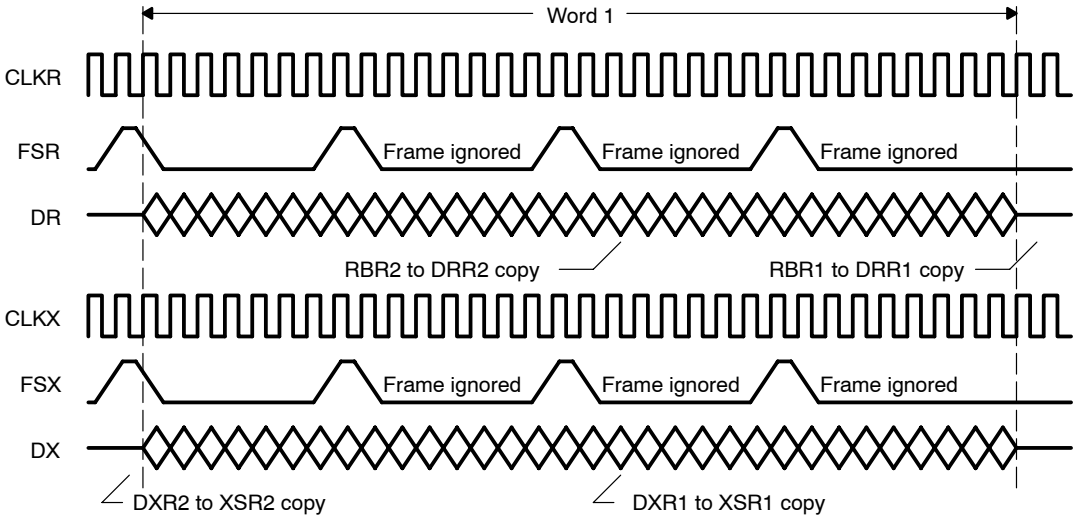


Figure 65 shows the McBSP configured to treat this data stream as a continuous 32-bit word. In this example, the McBSP responds to an initial frame-synchronization pulse. However, (R/X)FIG = 1, so that the McBSP ignores subsequent pulses. Only two read transfers or two write transfers are needed every 32 bits. This configuration effectively reduces the required bus bandwidth to half the bandwidth needed to transfer four 8-bit words.

Figure 65. Configuring the Data Stream of Figure 64 as a Continuous 32-Bit Word



12 McBSP on the Device – Applications

There are three McBSPs on the device.

- McBSP2

  This McBSP is located on the MPU public peripheral bus.

- McBSP1

  This McBSP is located on the DSP public peripheral bus.

- McBSP3

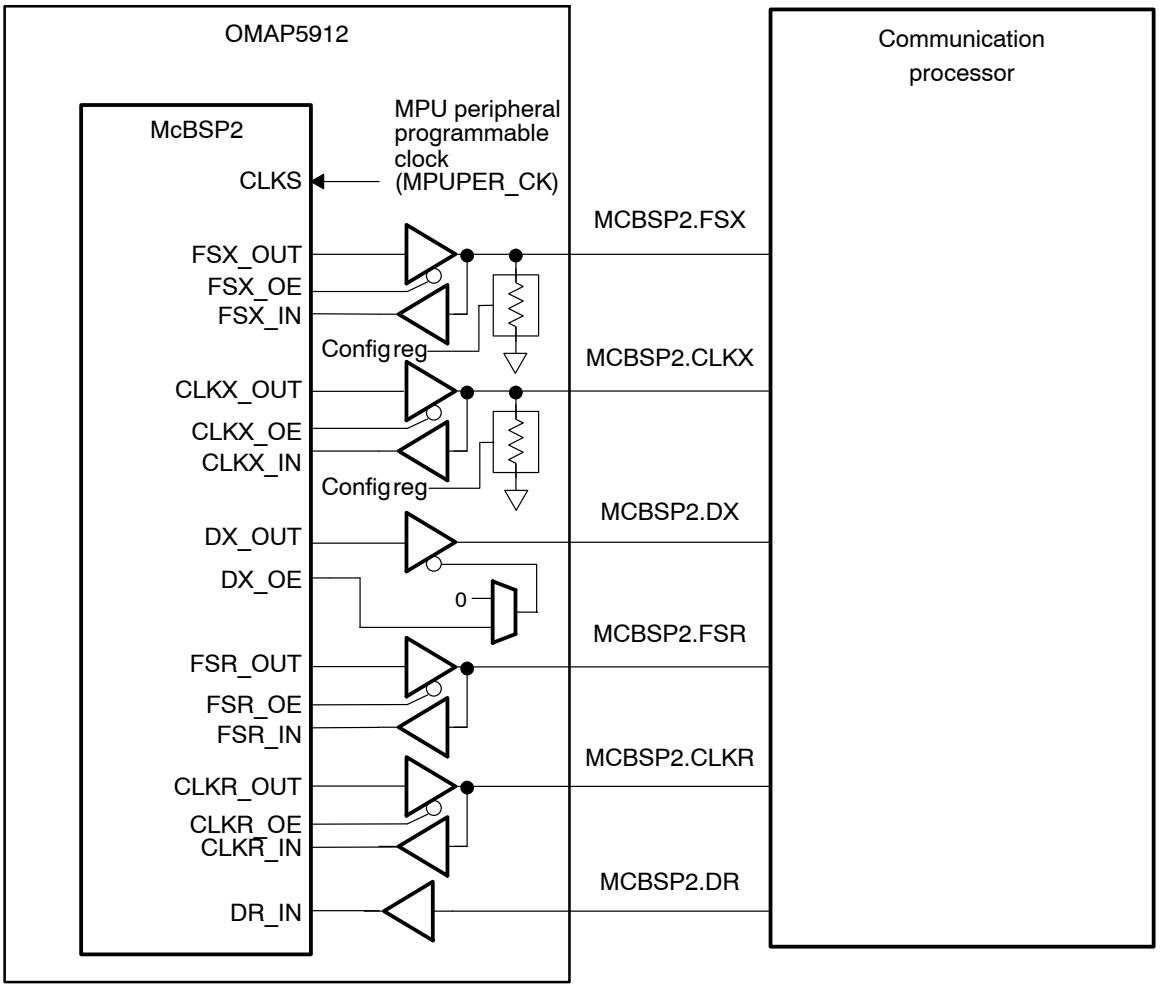
  This McBSP is located on the DSP public peripheral bus.

This section provides configuration examples for three common McBSP applications.

12.1 Communication McBSP Interface

Figure 66 illustrates the use of McBSP2 as a communication processor data interface that is the master of TX and slave for RX communications. The actual implementation is generic: FSX, CLKX, FSR, and CLKR are bidirectional. The direction of these signals is configured by registers in the McBSP module. The CLKS signal is the active input clock for the McBSP modem block. The active input clock can be changed in a McBSP register, but register activity on CLKS is required to perform the setup and write to the McBSP.

Figure 66. Communication Processor Data Interface



Section 12.1.1 through section 12.1.9 explain how to set up the McBSP registers for TX master and RX slave mode with 16-bit transfers using interrupts.

12.1.1 Serial Port Control Register Configuration

1) ARM_Write(0x0000) => SPCR1; set up SPCR1 as initial configuration.

   This setup is not needed after reset.

2) ARM_Write(0x0000) => SPCR2; set up SPCR2 as initial configuration.

   This setup is not needed after reset.

12.1.2 Pin Control Register Configuration

ARM_Write(0x0a00) => PCR; set up PCR per Table 75.

Table 75. Pin Control Register Bit Description

Bit	Configuration Value	Description
15:14	00b	Reserved
13	0b	Set serial port mode for DX, FSX and CLKX pins
12	0b	Set serial port mode for DR, FSR and CLKR pins
11	1b	TX frame-synchronization signal driven by internal generator
10	0b	RX frame-synchronization signal derived by external source
9	1b	CLKX set output pin and driven by internal generator
8	0b	CLKR set input pin and derived by external source
7	0b	Sample rate generator input clock mode bit
6	0b	CLKS pin status (no meaning in the OMAP5912 device)
5	0b	DX pin status
4	0b	DR pin status
3	0b	Set FSX polarity as active high
2	0b	Set FSR polarity as active high
1	0b	Set CLKX polarity as data driven on rising edge
0	0b	Set CLKR polarity as data sampled on falling edge

12.1.3 Receive Control Register Configuration

ARM_Write(0x0040) => RCR1; set up RCR1 per Table 76.

Table 76. Receive Control Register 1 Bit Description (RCR1)

Bit	Configuration Value	Description
15	0b	Reserved
14:8	000 0000b	Set receive frame length as one word per frame
7:5	010b	Set receive word length as 16 bit per frame
4:0	0 0000b	Reserved

ARM_Write(0x0001) = > RCR2; set up RCR2 per Table 77.

Table 77. Receive Control Register 2 Bit Description (RCR2)

Bit	Configuration Value	Description
15	0b	Set single-phase frame
14:8	000 0000b	Don't care for single-phase frame
7:5	000b	Don't care for single-phase frame
4:3	00b	Set no companding data and transfer start with MSB first
2	0b	Set FSR not ignore after the first resets the transfer
1:0	01b	Set data delay as 1 bit

12.1.4 Transmit Control Register Configuration

ARM_Write(0x0040) => XCR1; set up XCR1 per Table 78.

Table 78. Transmit Control Register 1 Bit Description (XCR1)

Bit	Configuration Value	Description
15	0b	Reserved
14:8	000 0000b	Set transmit frame length as one word per frame
7:5	010b	Set transmit word length as 16 bit per frame
4:0	0 0000b	Reserved

ARM_Write(0x0001) => XCR2; set up XCR2 per Table 79.

Table 79. Transmit Control Register 2 Bit Description (XCR2)

Bit	Configuration Value	Description
15	0b	Set single-phase frame
14:8	000 0000b	Don't care for single-phase frame
7:5	000b	Don't care for single-phase frame
4:3	00b	Set no companding data and transfer start with MSB first
2	0b	Set FSX not ignore after the first resets the transfer
1:0	01b	Set data delay as 1 bit

12.1.5 Sample Rate Generator Configuration

To configure the sample rate generator appropriately for CLKX and FSX:

1) Wait two CLKSRG clocks.
2) ARM_Write SPCR2 or (0x0000 0040)=>SPCR2; CLKG enable.
3) Wait two CLKG clocks.

For details, see *TMS320C54x DSP Enhanced Peripherals Reference Set, vol. 5*, SPRU302.

12.1.6 Interrupt Flag Configuration and Clear (ILR, ITR, MIR)

To clear interrupt flag configuration:

1) ARM_Write => ILR; set ILR appropriately for the interrupt handling priority.
2) ARM_Write ITR and (0xFFFF FFCF)=> ITR; clear remaining TX and RX interrupt.

Note:

This setup is not needed after reset.

3) ARM_Write MIR and (0xFFFF FFCF) => MIR; enable SPI TX and RX interrupt.

12.1.7 Take out of Reset for Transmit and Receive Starting (SPCR[1,2])

To enable transmit and receive:

1) ARM_write SPCR1 or (0x0001) => SPCR1; enabled receive port.
2) ARM_write SPCR2 or (0x0001) => SPCR2; enabled transmit port.

12.1.8 Transmit Data Loading (TX_INT Handling in Interrupt Survive Routine)

ARM_Write => DXR

Note:

Clear interrupts flag in ITR, when taking the interrupt handle.

12.1.9 Received Data Loading (RX_INT Handling in Interrupt Survive Routine)

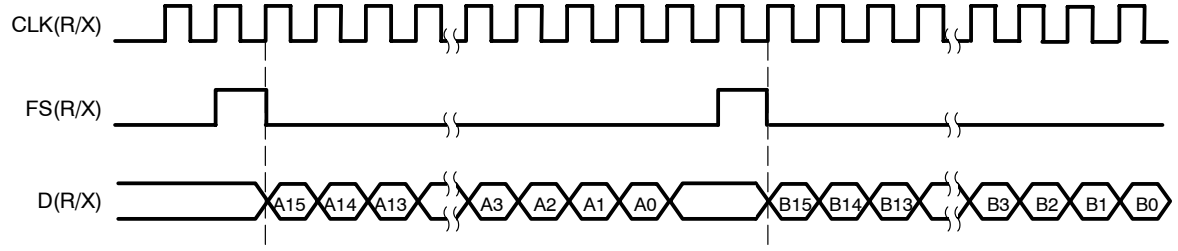
ARM_Read <= DRR

Note:

Clear interrupts flag in ITR, when the interrupt handle is taken.

Waveform Example

Figure 67. Waveform Example



Section 12.1.10 explains how to set up the McBSP registers for TX master and RX slave mode with 16-bit transfers using DMA support.

12.1.10 Serial Port Control Register Configuration

1) ARM_Write(0x0000) => SPCR1; set up SPCR1 as initial configuration.

   This setup is not needed after reset.

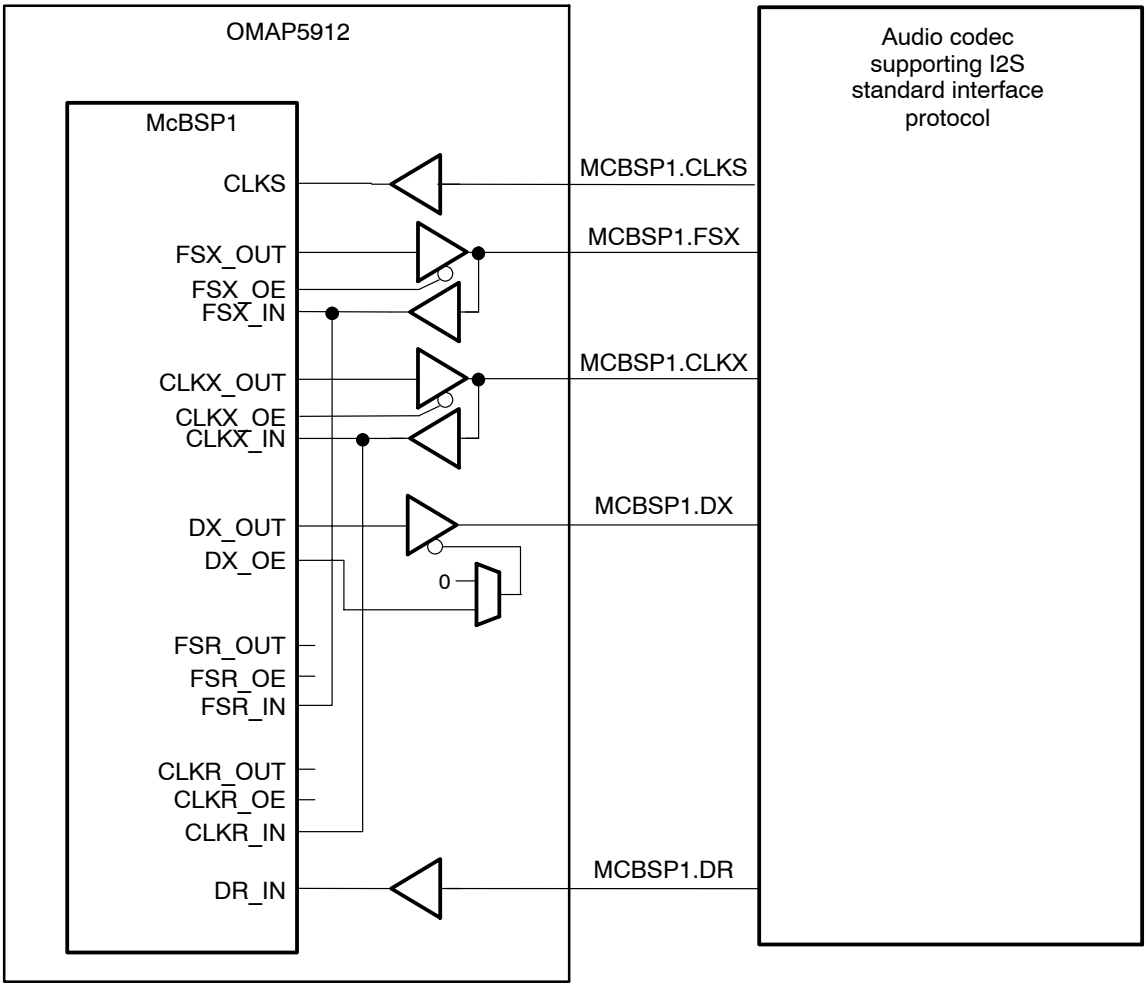
2) ARM_Write(0x0000) => SPCR2; set up SPCR2 as initial configuration.

   This setup is not needed after reset.

12.2 I2S Audio Codec McBSP Interface

This application uses McBSP1 as an I2S audio codec interface (see Figure 68). The OMAP5912 is intended to be either the master or slave device; that is, it either receives or provides the frame synchronization and bit clock.

Figure 68. I2S Audio Codec Interface



Section 12.2.1 through section 12.2.9 explain how to set up the McBSP registers for I2S slave mode with 16-bit transfers using DMA support.

12.2.1 Serial Port Control Register Configuration

1) DSP_Write(0x0000) => SPCR1; set up SPCR1 as initial configuration.
   This setup is not needed after reset.
2) DSP_Write(0x0000) => SPCR2; set up SPCR2 as initial configuration.
   This setup is not needed after reset.

12.2.2 Pin Control Register Configuration

DSP_Write(0x0000) => PCR; set up PCR per Table 80.

Table 80. Pin Control Register Bit Description (PCR)

Bit	Config Value	Description
15:14	00b	Reserved
13	0b	Set serial port mode for DX, FSX and CLKX pins
12	0b	Set serial port mode for DR, FSR and CLKR pins
11	0b	TX frame-synchronization signal derived by external source
10	0b	RX frame-synchronization signal derived by external source
9	0b	CLKX set input pin and derived by external source
8	0b	CLKR set input pin and derived by external source
7	0b	Sample rate generator input clock mode bit
6	0b	CLKS pin status (no meaning in the OMAP5912 device)
5	0b	DX pin status
4	0b	DR pin status
3	0b	Set FSX polarity as active high
2	0b	Set FSR polarity as active high
1	0b	Set CLKX polarity as data driven on rising edge
0	0b	Set CLKR polarity as data sampled on falling edge

12.2.3 Receive Control Register Configuration

DSP_Write(0x00a0) => RCR1; set up RCR1 per Table 81.

Table 81. Receive Control Register 1 Bit Description

Bit	Config Value	Description
15	0b	Reserved
14:8	000 0000b	Set receive frame length as one word per frame
7:5	101b	Set receive word length as 32 bits per frame
4:0	0 0000b	Reserved

DSP_Write(0x80a1) => RCR2; set up RCR2 per Table 82.

Table 82. Receive Control Register 2 Bit Description

Bit	Config Value	Description
15	1b	Set dual-phase frame
14:8	000 0000b	Set receive frame length as one word per frame
7:5	101b	Set receive word length as 32 bits per frame
4:3	00b	Don't care for single-phase frame
2	0b	Set FSR not ignore after the first resets the transfer
1:0	01b	Set data delay as 1 bit

12.2.4 Transmit Control Register Configuration

DSP_Write(0x00a0) => XCR1; set up XCR1 per Table 83.

Table 83. Transmit Control Register 1 Bit Description (XCR1)

Bit	Config Value	Description
15	0b	Reserved
14:8	000 0000b	Set transmit frame length as one word per frame
7:5	101b	Set receive word length as 32 bits per frame
4:0	0 0000b	Reserved

DSP_Write(0x80a1) => XCR2; set up XCR2 per Table 84.

Table 84. Transmit Control Register 2 Bit Description (XCR2)

Bit	Config Value	Description
15	1b	Set dual-phase frame
14:8	000 0000b	Don't care for single-phase frame
7:5	101b	Set receive word length as 32 bits per frame
4:3	00b	Set no companding data and transfer start with MSB first
2	0b	Set FSX not ignore after the first resets the transfer
1:0	01b	Set data delay as 1 bit

12.2.5 Sample Rate Generator Configuration (SRGR[1,2])

It is not necessary to configure the sample rate generator, because external clocks and frame synchronizations are provided appropriately for CLKX and FSX.

12.2.6 DMA Configuration

It is necessary to configure the REVT and XEVT bit for the DMA receive and transmit synchronized invent.

12.2.7 Interrupt Flag Configuration and Clear (ILR, MIR)

1) DSP_Write => ILR; set ILR appropriately for the interrupt handling priority.
2) DSP_Write MIR and (0x0000 0030) => MIR; disabled SPI TX and RX interrupt

Note:

Enable the appropriate DMA channel interrupts.

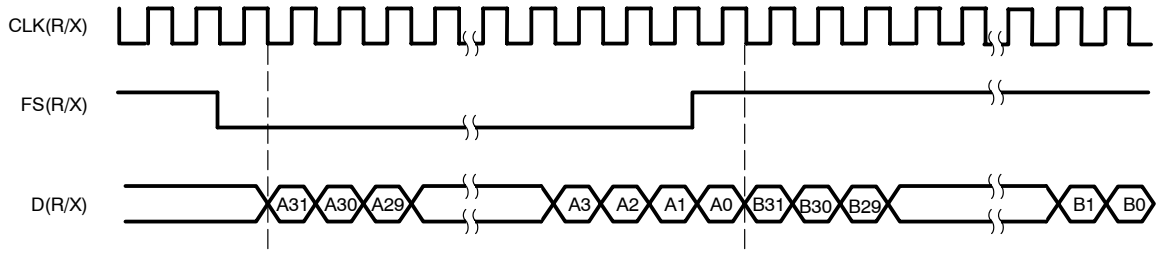
12.2.8 Take out of Reset for Transmit and Receive Starting (SPCR[1,2])

1) DSP_write SPCR1 or (0x0001) => SPCR1; enable receive port
2) DSP_write SPCR2 or (0x0001) => SPCR2; enable transmit port

12.2.9 Data Transfer (DMA Channel)

The DMA channel transfers the received data to the appropriate data buffer and transfers the new transmit data to appropriate TX buffer. Clear the interrupt flag on ITR when the interrupt handle is taken.

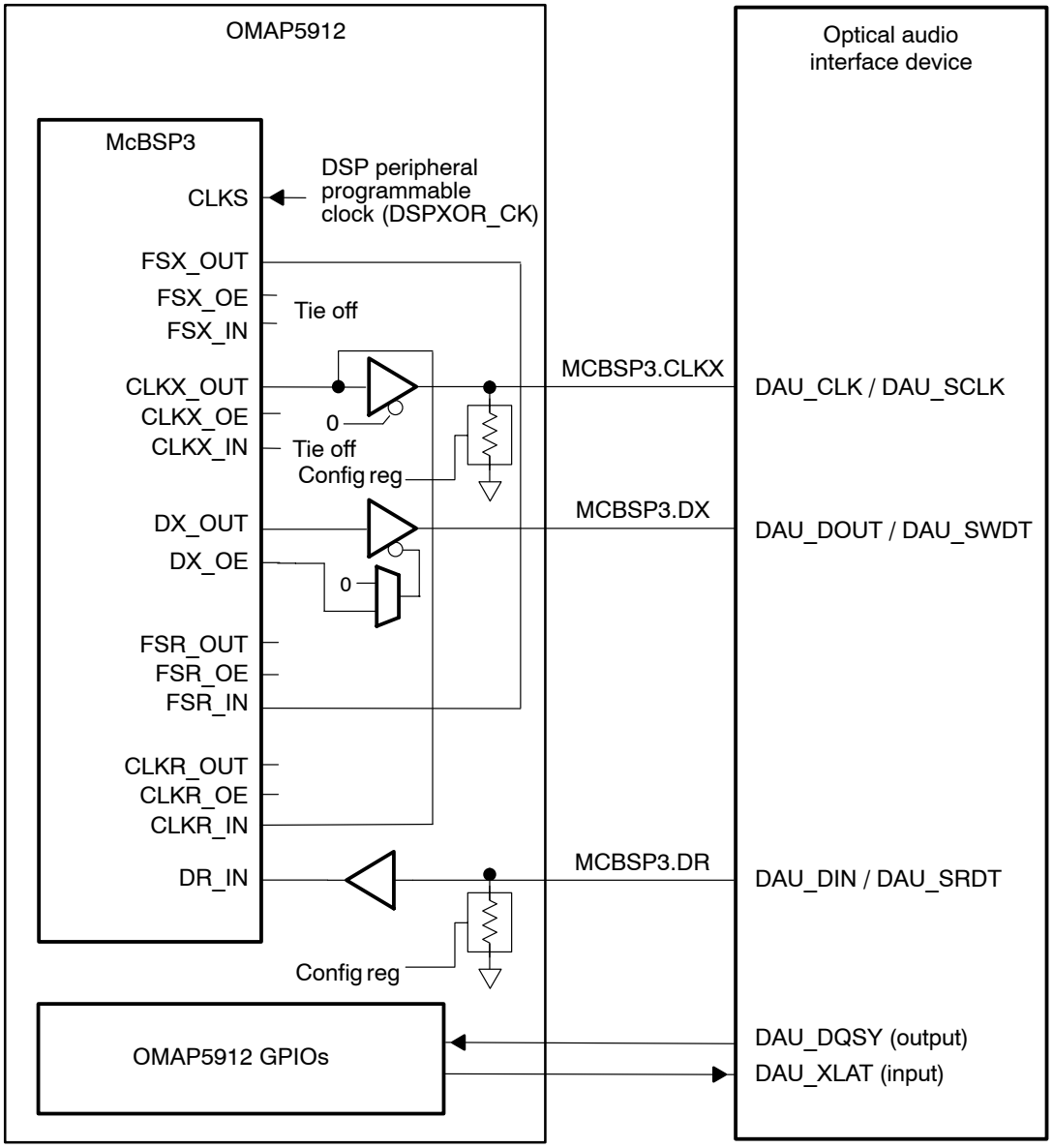
Figure 69. Waveform Example



12.3 Optical Audio McBSP Interface

With the assistance of two GPIOs, McBSP3 is configured to connect to an external optical audio interface device such as the Sanyo LC89051V (see Figure 70). The CLKS signal is the active input clock for the McBSP modem block. The active input clock can be changed in a McBSP register, but activity on CLKS is required to perform the set up and write to the McBSP register.

Figure 70. Optical Audio Interface



Section 12.3.1 through section 12.3.12 explain the McBSP register setup for optical interface with 8-bit transfer per frame in SPI master mode and GPIO mode.

12.3.1 Serial Port Control Register Configuration

DSP_Write(0x1000) => SPCR; set up SPCR1 per Table 85.

Table 85. Serial Port Control Register Bit Description (SPCR1)

Bit	Configuration Value	Description
15	0b	Disable digital loopback mode
14:13	00b	Right-justify and zero-fill MSBs in DRR
12:11	10b	Enabled clock stop mode
10:8	000b	Reserved
7	0b	Turn off the DX enabler
6	0b	Reserved
5:4	00b	Set RINT driven by RRDY mode
3	0b	No synchronization error
2	0b	RBR is not in overrun condition
1	0b	Receiver is not ready
0	0b	Disabled the serial port receiver and in reset state

DSP_Write(0x0000) => SPCR2; set up SPCR2 as initial configuration.

Note:

This setup is not needed after reset.

12.3.2 Pin Control Register Configuration

DSP_Write(0x0a0b) => PCR; set up PCR per Table 86.

Table 86. Pin Control Register Bit Description (PCR)

Bit	Configuration Value	Description
15:4	00b	Reserved
13	0b	Set serial port mode for DX, FSX and CLKX pins
12	0b	Set serial port mode for DR, FSR and CLKR pins
11	1b	TX frame-synchronization signal driven by internal generator
10	0b	RX frame-synchronization signal derived by external source
9	1b	McBSP is set master and generate clock by internal source
8	0b	CLKR set input pin and derived by external source
7	0b	Sample rate generator input clock mode bit
6	0b	CLKS pin status (no meaning in OMAP5912)
5	0b	DX pin status

Table 86. Pin Control Register Bit Description (PCR) (Continued)

Bit	Configuration Value	Description
4	0b	DR pin status
3	1b	Set FSX polarity as active low
2	0b	Set FSR polarity as active high
1	1b	Set CLKX polarity as data driven on falling edge
0	1b	Set CLKR polarity as data sampled on rising edge

12.3.3 Receive Control Register Configuration

DSP_Write(0x0000) => RCR1; set up RCR1 per Table 87.

Table 87. Receive Control Register 1 Bit Description (RCR1)

Bit	Configuration Value	Description
15	0b	Reserved
14:8	000 0000b	Set receive frame length as one word per frame
7:5	000b	Set receive word length as 8 bits per frame
4:0	0 0000b	Reserved

DSP_Write(0x0000) => RCR2; set up RCR2 per Table 88.

Table 88. Receive Control Register 2 Bit Description (RCR2)

Bit	Configuration Value	Description
15	0b	Set single-phase frame
14:8	000 0000b	Don't care for single phase frame
7:5	000b	Don't care for single phase frame
4:3	00b	Set no companding data and transfer start with MSB first
2	0b	Set FSR not ignore after the first resets the transfer
1:0	00b	Set data delay as 0 bit

The values of RWDLEN1, 2 and XWDLEN1, 2 must be set to the same value in SPI mode.

12.3.4 Transmit Control Register Configuration

DSP_Write(0x0000) => XCR1; set up XCR1 per Table 89.

Table 89. Transmit Control Register 1 Bit Description (XCR1)

Bit	Configuration Value	Description
15	0b	Reserved
14:8	000 0000b	Set transmit frame length as one word per frame
7:5	000b	Set transmit word length as 8 bits per frame
4:0	0 0000b	Reserved

DSP_Write(0x0000) => XCR2; set up XCR2 per Table 90.

Table 90. Transmit Control Register 2 Bit Description (XCR2)

Bit	Configuration Value	Description
15	0b	Set single-phase frame
14:8	000 0000b	Don't care for single phase frame
7:5	000b	Don't care for single phase frame
4:3	00b	Set no companding data and transfer start with MSB first
2	0b	Set FSX not ignore after the first resets the transfer
1:0	00b	Set data delay as 0 bit

The values of RWDLEN1, 2 and XWDLEN1, 2 must be set to the same value in SPI mode.

12.3.5 Sample Rate Generator Configuration (SRGR[1,2])

DSP_Write (0x00FF) => SRGR1; set up SRGR1 per Table 91.

Table 91. Sample Rate Generator Register 1 Bit Description (SRGR1)

Bit	Config Value	Description
15:8	0000 0000b	These bits ignored by the FSGM=0 (SRGR2[12:12])
7:0	1111 1111b	Set sample rate generator clock divider

DSP_Write (0x2000) => SRGR2; set up SRGR2 per Table 92.

Table 92. Sample Rate Generator Register 2 Bit Description (SRGR2)

Bit	Config Value	Description
15	0b	Set sample rate generator clock synchronization
14	0b	Set clock polarity
13	1b	Sample rate generator clock derived from DSP clock
12	0b	Set frame-synchronization
11:0	0000 0000 0000b	These bit ignored by the FSGM=0 (SRGR2[12:12])

Wait two CLKSRG clock cycles.

12.3.6 Start Sample Rate Generator (SPCR2)

DSP_Write SPCR2 or (0x0040) => SPCR2; bring sample rate generator out of reset.

Note:

Wait two sample rate clock for McBSP stability.

12.3.7 Interrupt Flag Configuration and Clear (ILR, ITR, MIR) on Level 2 Handler

1) DSP_Write => ILR; set ILR appropriately for the interrupt handling priority.
2) DSP_Write ITR and (0xFFFF F3FF)=> ITR; clear remaining TX and RX interrupts.

Note:

This setup is not needed after reset.

3) DSP_Write MIR and (0xFFFF F3FF) => MIR; enabled SPI TX and RX interrupt.

12.3.8 Interrupt Flag Configuration MASK Release on Level 2 Handler

DSP_Write MIR and (0xFFFF FFFB) => MIR0; enabled INT4 (level 2 interrupt FIR)

12.3.9 Take Out of Reset for Transmit and Receive Starting (SPCR[1,2])

1) DSP_write SPCR1 or (0x0001) => SPCR1; enable receive port.
2) DSP_write SPCR2 or (0x0001) => SPCR2; enable transmit port.

Note:

Wait two sample rate clock cycles for McBSP stability.

12.3.10 Transmit and Received Data Loading (TX_INT Handling in Interrupt Survive Routine)

For data transmit:

1) DSP_Write => DXR; transmit data loading to DXR.

2) DSP_Read <= DRR; wait for data read after the RINT.

For two data received:

1) DSP_Write => DXR; dummy write 0xFFFF for data receive after the TINT.

2) DSP_Read <= DRR; first data read after the RINT.

3) DSP_Write => DXR; dummy write 0xFFFF for data receive after the TINT.

4) DSP_Read <= DRR; second data read after the RINT.

12.3.11 Register Setup GPIO Mode

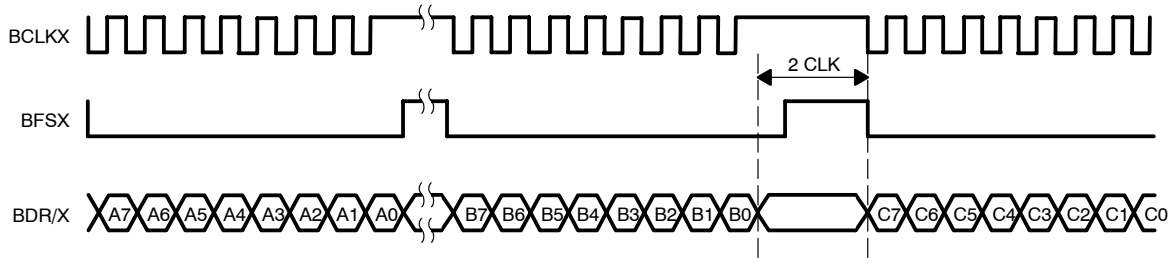
1) DSP_Write SPCR1 and (0xFFFE) => SPCR1; disable receive port.

2) DSP_Write PCR or (0x1000) => PCR; DR pin set as GPI.

12.3.12 Read From GPI

DSP_Read <= PCR; read DR_STAT bit

Figure 71. Waveform Example



Section 12.3.13 through section 12.3.21 explain the McBSP register setup for TX master and RX slave with 8-bit data transfer using DMA support.

12.3.13 Serial Port Control Register Configuration

DSP_Write(0x1000) => SPCR1; set up SPCR1 per Table 93.

Table 93. Serial Port Control Register Bit Description

Bit	Configuration Value	Description
15	0b	Disables digital loopback mode
14:13	00b	Right-justify and zero-fill MSBs in DRR
12:11	10b	Enables clock stop mode
10:8	000b	Reserved
7	0b	Turns off the DX enabler
6	0b	Reserved
5:4	00b	Set RINT driven by RRDY mode
3	0b	No synchronization error
2	0b	RBR is not in overrun condition.
1	0b	Receiver is not ready.
0	0b	Disables the serial port receiver and in reset state

DSP_Write(0x0000) => SPCR2; set up SPCR2 as initial configuration.

Note:

This setup is not needed after reset.

12.3.14 Pin Control Register Configuration

DSP_Write(0x0a0b) => PCR; set up PCR per Table 94.

Table 94. Pin Control Register Bit Description

Bit	Configuration Value	Description
15:14	00b	Reserved
13	0b	Set serial port mode for DX, FSX and CLKX pins
12	0b	Set serial port mode for DR, FSR and CLKR pins
11	1b	TX frame-synchronization signal driven by internal generator
10	0b	RX frame-synchronization signal derived by external source
9	1b	McBSP is set master and generate clock by internal source
8	0b	CLKR set input pin and derived by external source
7	0b	Sample rate generator input clock mode bit
6	0b	CLKS pin status (no meaning in OMAP5912)
5	0b	DX pin status

Table 94. Pin Control Register Bit Description (Continued)

Bit	Configuration Value	Description
4	0b	DR pin status
3	1b	Set FSX polarity as active high
2	0b	Set FSR polarity as active high
1	1b	Set CLKX polarity as data driven on falling edge
0	1b	Set CLKR polarity as data sampled on rising edge

12.3.15 Receive Control Register Configuration

DSP_Write(0x0000) => RCR1; set up RCR1 per Table 95.

Table 95. Receive Control Register 1 Bit Description (RCR1)

Bit	Configuration Value	Description
15	0b	Reserved
14:8	000 0000b	Set receive frame length as one word per frame
7:5	000b	Set receive word length as 8 bits per frame
4:0	0 0000b	Reserved

DSP_Write(0x0000) => RCR2; set up RCR2 per Table 96.

Table 96. Receive Control Register 2 Bit Description (RCR2)

Bit	Configuration Value	Description
15	0b	Set single-phase frame
14:8	000 0000b	Set receive frame length as one word per frame
7:5	000b	Set receive word length as 8 bits per frame
4:3	00b	Set no companding data and transfer start with MSB first
2	0b	Set FSR ignore after the first resets the transfer
1:0	00b	Set data delay as 0 bit

The values of RWDLEN1, 2 and XWDLEN1, 2 must be set to the same value in SPI mode.

12.3.16 Transmit Control Register Configuration

DSP_Write(0x0000) => XCR1; set up XCR1 per Table 97.

Table 97. Transmit Control Register 1 Bit Description (XCR1)

Bit	Config Value	Description
15	0b	Reserved
14:8	000 0000b	Set transmit frame length as one word per frame
7:5	000b	Set transmit word length as 8 bits per frame
4:0	0 0000b	Reserved

DSP_Write(0x0000) => XCR2; set up XCR2 per Table 98.

Table 98. Transmit Control Register 2 Bit Description (XCR2)

Bit	Config Value	Description
15	0b	Set single-phase frame
14:8	000 0000b	Set transmit frame length as one word per frame
7:5	000b	Set transmit word length as 8 bits per frame
4:3	00b	Set no companding data and transfer start with MSB first
2	0b	Set FSX ignore after the first resets the transfer
1:0	00b	Set data delay as 0 bit

The values of RWDLEN1, 2 and XWDLEN1, 2 must be set to the same value in SPI mode.

12.3.17 Sample Rate Generator Configuration (SRGR[1,2])

Configure the sample rate generator appropriately for CLKX and FSX. For details, see *TMS320C54x DSP Enhanced Peripherals Reference Set*, vol. 5, SPRU302.

1) Wait two CLKSRG clocks.
2) ARM_Write SPCR2 or (0x0000 0040)=>SPCR2;CLKG enable.
3) Wait two CLKG clocks.

12.3.18 DMA Configuration

Configure the REVT and XEVT bit for the DMA receive and transmit synchronized invent.

12.3.19 Interrupt Flag Configuration and Clear (ILR, MIR)

1) ARM_Write => ILR; set ILR appropriately for the interrupt handling priority.
2) ARM_Write MIR and (0x0000 0D00) => MIR; disable SPI TX and RX interrupt.

Note:

Enable the appropriate DMA channel interrupts.

12.3.20 Take out of Reset for Transmit and Receive Starting (SPCR[1,2])

1) ARM_write SPCR1 or (0x0001) => SPCR1; enable receive port.

2) ARM_write SPCR2 or (0x0001) => SPCR2; enable transmit port.

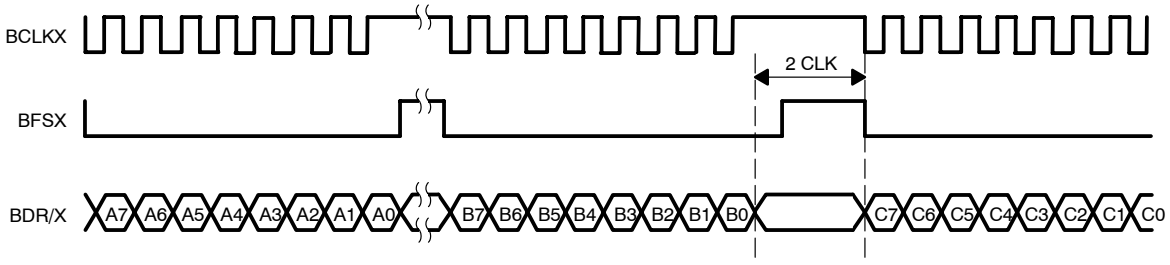
12.3.21 Data Transfer (DMA Channel)

The DMA channel transfers the received data to the appropriate data buffer and transfers the new transmit data to the appropriate TX buffer. Clear interrupts flag on ITR when taking the interrupt handle.

Note:

Clear interrupts flag on ITR, when taking the interrupt handle.

Figure 72. Waveform Example



13 McBSP Registers

The base address for each McBSP register map is as follows:

- McBSP1 (I2S audio):
  - E101:1800 (MPU memory map)
  - x001:1800 (DSP memory map)
- McBSP2 (modem interface): FFFB:1000 (MPU memory map)
- McBSP3 (optical interface):
  - E101:7000 (MPU memory map)
  - x001:7000 (DSP memory map)

Table 99 shows the registers accessible by a user on each McBSP. Table 100 through Table 118 describe register bits.

Table 99. McBSP Registers

Name	Description	Offset
DRR2(15:0)	Data receive register 2	0x00
DRR1(15:0)	Data receive register 1	0x02
DXR2(15:0)	Data transmit register 2	0x04
DXR1(15:0)	Data transmit register 1	0x06
SPCR2(15:0)	Serial port control register 2	0x08
SPCR1(15:0)	Serial port control register 1	0x0A
RCR2(15:0)	Receive control register 2	0x0C
RCR1(15:0)	Receive control register 1	0x0E
XCR2 (15:0)	Transmit control register 2	0x10
XCR1(15:0)	Transmit control register 1	0x12
SRGR2(15:0)	Sample rate generator register 2	0x14
SRGR1(15:0)	Sample rate generator register 1	0x16
MCR2(15:0)	Multichannel register 2	0x18
MCR1(15:0)	Multichannel register 1	0x1A
RCERA(15:0)	Receive channel enable register partition A	0x1C
RCERB(15:0)	Receive channel enable register partition B	0x1E

Table 99. McBSP Registers (Continued)

Name	Description	Offset
XCERA(15:0)	Transmit channel enable register partition A	0x20
XCERB(15:0)	Transmit channel enable register partition B	0x22
PCR0(15:0)	Pin control register	0x24
RCERC(15:0)	Receive channel enable register partition C	0x26
RCERD(15:0)	Receive channel enable register partition D	0x28
XCERC(15:0)	Transmit channel enable register partition C	0x2A
XCERD(15:0)	Transmit channel enable register partition D	0x2C
RCERE(15:0)	Receive channel enable register partition E	0x2E
RCERF(15:0)	Receive channel enable register partition F	0x30
XCERE(15:0)	Transmit channel enable register partition E	0x32
XCERF(15:0)	Transmit channel enable register partition F	0x34
RCERG(15:0)	Receive channel enable register partition G	0x36
RCERH(15:0)	Receive channel enable register partition H	0x38
XCERG(15:0)	Transmit channel enable register partition G	0x3A
XCERH(15:0)	Transmit channel enable register partition H	0x3C

13.1 Data Receive Registers (DRR2 and DRR1)

The CPU or the DMA controller reads received data from one or both of the data receive registers (see Figure 73). If the serial word length is 16 bits or smaller, only DRR1 is used. If the serial length is larger than 16 bits, both DRR1 and DRR2 are used and DRR2 holds the most significant bits. Each frame of receive data in the McBSP can have one phase or two phases, each with its own serial word length.

DRR1 and DRR2 are I/O mapped registers; they are accessible at addresses in I/O space.

Figure 73. Data Receive Registers (DRR2 and DRR1)

DRR2

15	0
High part of receive data (for 20-, 24- or 32-bit data)	
R/W – 0	

DRR1

15	0
Receive data (for 8-, 12-, or 16-bit data) or low part of receive data (for 20-, 24- or 32-bit data)	
R/W – 0	

Legend: R = read; W = write; -*n* = value after reset

13.1.1 Data Travel From Data Receive Pins to the Registers

If the serial word length is 16 bits or smaller, receive data on the DR pin is shifted into receive shift register 1 (RSR1) and then copied into receive buffer register 1 (RBR1). The content of RBR1 is then copied to DRR1, which can be read by the CPU or by the DMA controller.

If the serial word length is larger than 16 bits, receive data on the DR pin is shifted into both of the receive shift registers (RSR2, RSR1) and then copied into both of the receive buffer registers (RBR2, RBR1). The content of the RBRs is then copied into both of the DRRs, which can be read by the CPU or by the DMA controller.

If companding is used during the copy from RBR1 to DRR1 (RCOMPAND = 10b or 11b), the 8-bit compressed data in RBR1 is expanded to a left-justified 16-bit value in DRR1. If companding is disabled, the data copied from RBR[1,2] to DRR[1,2] is justified and bit filled according to the RJUST bits.

The RSRs and RBRs are not accessible. They are not mapped to I/O space like the DRRs.

13.2 Data Transmit Registers (DXR2 and DXR1)

For transmission, the CPU or the DMA controller writes data to one or both of the data transmit registers (see Figure 74). If the serial word length is 16 bits or smaller, only DXR1 is used. If the word length is larger than 16 bits, both DXR1 and DXR2 are used and DXR2 holds the most significant bits. Each frame of transmit data in the McBSP can have one phase or two phases, each with its own serial word length.

DXR1 and DXR2 are I/O mapped registers; they are accessible at addresses in I/O space.

Figure 74. Data Transmit Registers (DXR2 and DXR1)

DXR2

15	0
High part of transmit data (for 20-, 24- or 32-bit data)	
R/W – 0	

DXR1

15	0
Transmit data (for 8-, 12-, or 16-bit data) or low part of receive data (for 20-, 24- or 32-bit data)	
R/W – 0	

Legend: R = read; W = write; *-n* = value after reset

13.2.1 Data Travel From Registers to Data Transmit (DX) Pins

If the serial word length is 16 bits or fewer, data written to DXR1 is copied to transmit shift register 1 (XSR1). From XSR1, the data is shifted onto the DX pin one bit at a time.

If the serial word length is more than 16 bits, data written to DXR1 and DXR2 is copied to both transmit shift registers (XSR2, XSR1). From the XSRs, the data is shifted onto the DX pin one bit at a time.

If companding is used during the transfer from DXR1 to XSR1 (XCOMPAND = 10b or 11b), the McBSP compresses the 16-bit data in DXR1 to 8-bit data in the μ-law or A-law format in XSR1. If companding is disabled, the McBSP passes data from the DXR(s) to the XSR(s) without modification.

The XSRs are not accessible. They are not mapped to I/O space like the DXRs.

13.3 Serial Port Control Registers (SPCR1 and SPCR2)

Each McBSP has two serial port control registers. Table 100 and Table 102 describe the bits in SPCR1 and SPCR2, respectively. These I/O-mapped registers enable you to:

- Control various McBSP modes: digital loopback mode (DLB), sign-extension and justification mode for reception (RJUST), clock stop mode (CLKSTP), interrupt modes (RINTM and XINTM), emulation mode (FREE and SOFT)
- Turn on and off the DX-pin delay enabler (DXENA)
- Check the status of receive and transmit operations (RSYNCERR, XSYNCERR, RFULL, XEMPTY, RRDY, XRDY)
- Reset portions of the McBSP (RRST, XRST, FRST, GRST)

Table 100. Serial Port Control 1 Register (SPCR1)

Bit	Name	Value	Description	Type	Reset
15	DLB		Digital loopback mode bit. DLB disables or enables the digital loopback mode of the McBSP:	R/W	0
		0	Disabled Internal DR is supplied by the DR pin. Internal FSR and internal CLKR can be supplied by their respective pins or by the sample rate generator, depending on the mode bits FSRM and CLKRM.		
		1	Enabled Internal receive signals are supplied by internal transmit signals: DR connected to DX FSR connected to FSX CLKR connected to CLKX Internal DX is supplied by the DX pin. Internal FSX and internal CLKX are supplied by their respective pins or are generated internally, depending on the mode bits FSXM and CLKXM. This mode allows you to test serial port code with a single DSP. The McBSP transmitter directly supplies data, frame synchronization, and clocking to the McBSP receiver.		
14:13	RJUST		Receive sign-extension and justification mode bits. During reception, RJUST determines how data is justified and bit filled before being passed to the data receive registers (DRR1, DRR2). RJUST is ignored if you enable a companding mode with the RCOMPAND bits. In a companding mode, the 8-bit compressed data in RBR1 is expanded to left-justified 16-bit data in DRR1. For more details about the effects of RJUST, see section 7.13, *Set the Receive Sign-Extension and Justification Mode*.	R/W	00
		00	Right justify the data and zero fill the MSBs.		
		01	Right justify the data and sign-extend the data into the MSBs.		
		10	Left justify the data and zero fill the LSBs.		
		11	Reserved (do not use)		

Table 100. Serial Port Control 1 Register (SPCR1) (Continued)

Bit	Name	Value	Description	Type	Reset
12:11	CLKSTP		Clock stop mode bits. CLKSTP allows you to use the clock stop mode to support the SPI master-slave protocol. If you will not be using 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). For more details, see section 7.5, *Enable/Disable the Clock Stop*.	R/W	00
		00/01	Clock stop mode is disabled.		
		10	Clock stop mode, without clock delay.		
		11	Clock stop mode, with half-cycle clock delay.		
10:8	Reserved		Reserved bits (not available for your use). They are read-only bits and return 0s when read.		
7	DXENA		DX delay enabler mode bit. DXENA controls the delay enabler for the DX pin. The enabler creates an extra delay for turn-on time (for the length of the delay, see the data sheet for your TMS320C55x DSP). For more details about the effects of DXENA, see section 8.13, *Set the Transmit DXENA Mode*.	R/W	0
		0	DX delay enabler off.		
		1	DX delay enabler on.		
6	Reserved			R/W	0

Table 100. Serial Port Control 1 Register (SPCR1) (Continued)

Bit	Name	Value	Description	Type	Reset
5:4	RINTM		Receive interrupt mode bits. RINTM determines which event in the McBSP receiver generates a receive interrupt (RINT) request. If RINT is properly enabled inside the CPU, the CPU services the interrupt request; otherwise, the CPU ignores the request.	R/W	00
		00	The McBSP sends a receive interrupt (RINT) request to the CPU when the RRDY bit changes from 0 to 1, indicating that receive data is ready to be read (the content of RBR[1,2] has been copied to DRR[1,2]): Regardless of the value of RINTM, you can check RRDY to determine whether a word transfer is complete. The McBSP sends a RINT request to the CPU when 16 enabled bits have been received on the DR pin.		
		01	In the multichannel selection mode, the McBSP sends a RINT request to the CPU after every 16-channel block is received in a frame. Outside of the multichannel selection mode, no interrupt request is sent.		
		10	The McBSP sends a RINT request to the CPU when each receive frame-synchronization pulse is detected. The interrupt request is sent even if the receiver is in its reset state.		
		11	The McBSP sends a RINT request to the CPU when the RSYNCERR bit is set, indicating a receive frame-synchronization error. Regardless of the value of RINTM, you can check RSYNCERR to determine whether a receive frame-synchronization error occurred.		
3	RSYNCERR		Receive frame-sync error bit. RSYNCERR is set when a receive frame-sync error is detected by the McBSP. If RINTM = 11b, the McBSP sends a receive interrupt (RINT) request to the CPU when RSYNCERR is set. The flag remains set until you write a 0 to it or reset the receiver. Caution**:** If RINTM = 11b, writing a 1 to RSYNCERR triggers a receive interrupt just as if a receive frame-synchronization error occurred.	R/W	0
		0	No error.		
		1	Receive frame-synchronization error. For more details about this error, see section 4.2, *Unexpected Receive Frame-Synchronization Pulse*.		

Table 100. Serial Port Control 1 Register (SPCR1) (Continued)

Bit	Name	Value	Description	Type	Reset
2	RFULL		Receiver full bit. RFULL is set when the receiver is full with new data and the previously received data has not been read (receiver-full condition). For more details about this condition, see section 4.1, *Overrun in the Receiver*.	R	0
		0	No receiver-full condition.		
		1	Receiver-full condition: RSR[1,2] and RBR[1,2] are full with new data, but the previous data in DRR[1,2] has not been read.		
1	RRDY		Receiver ready bit. RRDY is set when data is ready to be read from DRR[1,2]. Specifically, RRDY is set in response to a copy from RBR1 to DRR1. If the receive interrupt mode is RINTM = 00b, the McBSP sends a receive interrupt request to the CPU when RRDY changes from 0 to 1. Also, when RRDY changes from 0 to 1, the McBSP sends a receive synchronization event (REVT) signal to the DMA controller.	R	0
		0	Receiver not ready. When the content of DRR1 is read, RRDY is automatically cleared.		
		1	Receiver ready: New data can be read from DRR[1,2]. Important: If both DRRs are required (word length larger than 16 bits), the CPU or the DMA controller must read from DRR2 first and then from DRR1. As soon as DRR1 is read, the next RBR-to-DRR copy is initiated. If DRR2 is not read first, the data in DRR2 is lost.		

Table 100. Serial Port Control 1 Register (SPCR1) (Continued)

Bit	Name	Value	Description	Type	Reset
0	RRST		Receiver reset bit. You can use RRST to take the McBSP receiver into and out of its reset state. This bit has a negative polarity; RRST = 0 indicates the reset state. To read about the effects of a receiver reset, see section 10.3, *Resetting and Initializing a McBSP*.	R/W	0
		0	If you read a 0, the receiver is in its reset state. If you write a 0, you reset the receiver.		
		1	If you read a 1, the receiver is enabled. If you write a 1, you enable the receiver by taking it out of its reset state.		

Table 101. Serial Port Control 2 Register (SPCR2)

Bit	Name	Value	Description	Type	Reset
15:10	Reserved		Reserved bits (not available for your use). They are read-only bits and return 0s when read.		
9	FREE		Free run bit. When a breakpoint is encountered in the high-level language debugger, FREE determines whether the McBSP transmit and receive clocks continue to run or whether they are affected as determined by the SOFT bit. When one of the clocks stops, the corresponding data transfer (transmission or reception) stops.	R/W	0
		0	The McBSP transmit and receive clocks are affected as determined by the SOFT bit.		
		1	Free run. The McBSP transmit and receive clocks continue to run.		

Table 101. Serial Port Control 2 Register (SPCR2) (Continued)

Bit	Name	Value	Description	Type	Reset
8	SOFT		Soft stop bit. When FREE = 0, SOFT determines the response of the McBSP transmit and receive clocks when a breakpoint is encountered in the high-level language debugger. When one of the clocks stops, the corresponding data transfer (transmission or reception) stops.	R/W	0
		0	Hard stop. The McBSP transmit and receive clocks are stopped immediately.		
		1	Soft stop. The McBSP transmit clock stops after completion of the current serial word transfer. The McBSP receive clock is not affected.		
7	FRST		Frame-synchronization logic reset bit. The sample rate generator of the McBSP includes frame-synchronization logic to generate an internal frame-synchronization signal. You can use FRST to take the frame-synchronization logic into and out of its reset state. This bit has a negative polarity; FRST = 0 indicates the reset state.	R/W	0
		0	If you read a 0, the frame-synchronization logic is in its reset state. If you write a 0, you reset the frame-synchronization logic. In the reset state, the frame-synchronization logic does not generate a frame-synchronization signal (FSG).		
		1	If you read a 1, the frame-synchronization logic is enabled. If you write a 1, you enable the frame-synchronization logic by taking it out of its reset state. When the frame-synchronization logic is enabled (FRST = 1) and the sample rate generator as a whole is enabled (GRST = 1), the frame-synchronization logic generates the frame-synchronization signal FSG as programmed.		

Table 101. Serial Port Control 2 Register (SPCR2) (Continued)

Bit	Name	Value	Description	Type	Reset
6	GRST		Sample rate generator reset bit. You can use GRST to take the McBSP sample rate generator into and out of its reset state. This bit has a negative polarity; GRST = 0 indicates the reset state. To read about the effects of a sample rate generator reset, see section 10.3, *Resetting and Initializing a McBSP*.	R/W	0
		0	If you read a 0, the sample rate generator is in its reset state. If you write a 0, you reset the sample rate generator. If GRST = 0 due to a reset, CLKG is driven by the CPU clock divided by 2, and FSG is driven low (inactive). If GRST = 0 due to program code, CLKG and FSG are both driven low (inactive).		
		1	If you read a 1, the sample rate generator is enabled. If you write a 1, you enable the sample rate generator by taking it out of its reset state. When enabled, the sample rate generator generates the clock signal CLKG as programmed in the sample rate generator registers. If FRST = 1, the generator also generates the frame-synchronization signal FSG as programmed in the sample rate generator registers.		

Table 101. Serial Port Control 2 Register (SPCR2) (Continued)

Bit	Name	Value	Description	Type	Reset
5:4	XINTM		Transmit interrupt mode bits. XINTM determines which event in the McBSP transmitter generates a transmit interrupt (XINT) request. If XINT is properly enabled, the CPU services the interrupt request; otherwise, the CPU ignores the request.	R/W	00
		00b	The McBSP sends a transmit interrupt (XINT) request to the CPU when the XRDY bit changes from 0 to 1, indicating that transmitter is ready to accept new data (the content of DXR[1,2] has been copied to XSR[1,2]). Regardless of the value of XINTM, you can check XRDY to determine whether a word transfer is complete. The McBSP sends an XINT request to the CPU when 16 enabled bits have been transmitted on the DX pin.		
		01b	In the multichannel selection mode, the McBSP sends an XINT request to the CPU after every 16-channel block is transmitted in a frame. Outside of the multichannel selection mode, no interrupt request is sent.		
		10b	The McBSP sends an XINT request to the CPU when each transmit frame-synchronization pulse is detected. The interrupt request is sent even if the transmitter is in its reset state.		
		11b	The McBSP sends an XINT request to the CPU when the XSYNCERR bit is set, indicating a transmit frame-synchronization error. Regardless of the value of XINTM, you can check XSYNCERR to determine whether a transmit frame-synchronization error occurred.		

Table 101. Serial Port Control 2 Register (SPCR2) (Continued)

Bit	Name	Value	Description	Type	Reset
3	XSYNCERR		Transmit frame-synchronization error bit. XSYNCERR is set when a transmit frame-synchronization error is detected by the McBSP. If XINTM = 11b, the McBSP sends a transmit interrupt (XINT) request to the CPU when XSYNCERR is set. The flag remains set until you write a 0 to it or reset the transmitter. If XINTM = 11b, writing a 1 to XSYNCERR triggers a transmit interrupt just as if a transmit frame-synchronization error occurred. For details about this error see section 4.5, *Unexpected Transmit Frame-Synchronization Pulse*.	R/W	0
		0	No error.		
		1	Transmit frame-synchronization error.		
2	XEMPTY		Transmitter empty bit. XEMPTY is cleared when the transmitter is ready to send new data but no new data is available (transmitter-empty condition). This bit has a negative polarity; a transmitter-empty condition is indicated by XEMPTY = 0.	R	0
		0	Transmitter-empty condition. Typically this indicates that all the bits of the current word have been transmitted but there is no new data in DXR1. XEMPTY is also cleared if the transmitter is reset and then restarted. For more details about this error condition, see section 4.4, *Underflow in the Transmitter*.		
		.1	.No transmitter-empty condition.		

Table 101. Serial Port Control 2 Register (SPCR2) (Continued)

Bit	Name	Value	Description	Type	Reset
1	XRDY		Transmitter ready bit. XRDY is set when the transmitter is ready to accept new data in DXR[1,2]. Specifically, XRDY is set in response to a copy from DXR1 to XSR1. If the transmit interrupt mode is XINTM = 00b, the McBSP sends a transmit interrupt (XINT) request to the CPU when XRDY changes from 0 to 1. Also, when XRDY changes from 0 to 1, the McBSP sends a transmit synchronization event (XEVT) signal to the DMA controller.	R	0
		0	Transmitter not ready. When DXR1 is loaded, XRDY is automatically cleared.		
		1	Transmitter ready: DXR[1,2] is ready to accept new data. If both DXRs are needed (word length larger than 16 bits), the CPU or the DMA controller must load DXR2 first and then load DXR1. As soon as DXR1 is loaded, the contents of both DXRs are copied to the transmit shift registers (XSRs), as described in the next step. If DXR2 is not loaded first, the previous content of DXR2 is passed to the XSR2.		
0	XRST		Transmitter reset bit. You can use XRST to take the McBSP transmitter into and out of its reset state. This bit has a negative polarity; XRST = 0 indicates the reset state. To read about the effects of a transmitter reset, see section 10.3, *Resetting and Initializing a McBSP*.	R/W	0
		0	If you read a 0, the transmitter is in its reset state. If you write a 0, you reset the transmitter.		
		1	If you read a 1, the transmitter is enabled. If you write a 1, you enable the transmitter by taking it out of its reset state.		

13.4 Receive Control Registers (RCR1 and RCR2)

Each McBSP has two receive control registers. Table 102 and Table 104 describe the bits of RCR1 and RCR2, respectively. These I/O-mapped registers enable you to:

- Specify one or two phases for each frame of receive data (RPHASE)
- Define two parameters for phase 1 and, if necessary, phase 2: the serial word length (RWDLEN1, RWDLEN2) and the number of words (RFRLEN1, RFRLEN2)
- Choose a receive companding mode, if any (RCOMPAND)
- Enable or disable the receive frame-synchronization ignore function (RFIG)
- Choose a receive data delay (RDATDLY)

Table 102. Receive Control 1 Register (RCR1)

Bit	Name	Value	Description	Type	Reset
15	Reserved	0	Reserved bits (not available for your use). They are read-only bits and return 0s when read.	R	0
14:8	RFRLEN1	0 – 127	Receive frame length 1 (1 to 128 words). Each frame of receive data can have one or two phases, depending on value that you load into the RPHASE bit. If a single-phase frame is selected, RFRLEN1 in RCR1 selects the number of serial words (8, 12, 16, 20, 24, or 32 bits per word) in the frame. If a dual-phase frame is selected, RFRLEN1 determines the number of serial words in phase 1 of the frame, and RFRLEN2 in RCR2 determines the number of words in phase 2 of the frame. The 7-bit RFRLEN fields allow up to 128 words per phase. See Table 103 for a summary of how you determine the frame length. This length corresponds to the number of words or logical time slots or channels per frame-synchronization period. 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.	R/W	0
7:5	RWDLEN1		Receive word length 1. Each frame of receive data 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 in RCR1 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 in RCR2 determines the word length in phase 2 of the frame.	R/W	000
		000b	8 bits		
		001b	12 bits		
		010b	16 bits		
		011b	20 bits		
		100b	24 bits		
		101b	32 bits		
		other	Reserved (do not use)		
4:0	Reserved	0	Reserved bits (not available for your use). They are read-only bits and return 0s when read.	R	0

Table 103. Frame Length Formula for Receive Control 1 Register (RCR1)

RPHASE	RFRLEN1	RFRLEN2	Frame Length
0	0 ≤ RFRLEN1 ≤ 127	Not used	(RFRLEN1 + 1) words
1	0 ≤ RFRLEN1 ≤ 127	0 ≤ RFRLEN2 ≤ 127	(RFRLEN1 + 1) + (RFRLEN2 + 1) words

Table 104. Receive Control 2 Register (RCR2)

Bit	Name	Value	Description	Type	Reset
15	RPHASE		Receive phase number bit. RPHASE determines whether the receive frame has one phase or two phases. For each phase you can define the serial word length and the number of serial words in the phase. To set up phase 1, program RWDLEN1 (word length) and RFRLEN1 (number of words). To set up phase 2 (if there are two phases), program RWDLEN2 and RFRLEN2.	R/W	0
		0	Single-phase frame. The receive frame has only one phase, phase 1.		
		1	Dual-phase frame. The receive frame has two phases, phase 1 and phase 2.		

Table 104. Receive Control 2 Register (RCR2) (Continued)

Bit	Name	Value	Description	Type	Reset
14:8	RFRLEN2	0–127	Receive frame length 2 (1 to 128 words). Each frame of receive data can have one or two phases, depending on value that you load into the RPHASE bit. If a single-phase frame is selected, RFRLEN1 in RCR1 selects the number of serial words (8, 12, 16, 20, 24, or 32 bits per word) in the frame. If a dual-phase frame is selected, RFRLEN1 determines the number of serial words in phase 1 of the frame, and RFRLEN2 in RCR2 determines the number of words in phase 2 of the frame. The 7-bit RFRLEN fields allow up to 128 words per phase. See Table 105 for a summary of how to determine the frame length. This length corresponds to the number of words, or logical time slots, or channels per frame-synchronization period. 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 2, load 127 into RFRLEN2.	R/W	0
7:5	RWDLEN2		Receive word length 2. Each frame of receive data 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 in RCR1 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 in RCR2 determines the word length in phase 2 of the frame.	R/W	000
7:5	RWDLEN2	000b	8 bits		
		001b	12 bits		
		010b	16 bits		
		011b	20 bits		
		100b	24 bits		
		101b	32 bits		
		other	Reserved (do not use)		

Table 104. Receive Control 2 Register (RCR2) (Continued)

Bit	Name	Value	Description	Type	Reset
4:3	RCOMPAND		Receive companding mode bits. Companding (COMpress and exPAND) hardware allows compression and expansion of data in either μ-law or A-law format. For more details about these companding modes, see section 2.2, *Companding (Compressing and Expanding) Data*.	R/W	00
			RCOMPAND allows you to choose one of the following companding modes for the McBSP receiver:		
		00b	No companding, any size data, MSB received first		
		01b	No companding, 8-bit data, LSB received first		
		10b	μ-law companding, 8-bit data, MSB received first		
		11b	A-law companding, 8-bit data, MSB received first		
2	RFIG		Receive frame-synchronization ignore bit. If a frame-synchronization pulse starts the transfer of a new frame before the current frame is fully received, this pulse is treated as an unexpected frame-synchronization pulse. For more details about the frame-synchronization error condition, see section 4.2, *Unexpected Receive Frame-Synchronization Pulse*.	R/W	0
			Setting RFIG causes the serial port to ignore unexpected frame-synchronization signals during reception. For more details on the effects of RFIG, see section 7.10.1, *Enable/Disable the Receive Frame-Synchronization Ignore Function*.		
		0	Frame-synchronization detect. An unexpected FSR pulse causes the receiver to discard the contents of RSR[1,2] in favor of the new incoming data. The receiver: 1) Aborts the current data transfer 2) Sets RSYNCERR in SPCR1 3) Begins the transfer of a new data word		
		1	Frame-synchronization ignore. An unexpected FSR pulse is ignored. Reception continues uninterrupted.		

Table 104. Receive Control 2 Register (RCR2) (Continued)

Bit	Name	Value	Description	Type	Reset
1:0	RDATDLY		Receive data delay bits. RDATDLY specifies a data delay of 0, 1, or 2 receive clock cycles after frame-synchronization and before the reception of the first bit of the frame. For more details, see section 7.12, *Set the Receive Data Delay*.	R/W	00
		00b	0-bit data delay		
		01b	1-bit data delay		
		10b	2-bit data delay		
		11b	Reserved (do not use)		

Table 105. Frame Length Formula for RCR2

RPHASE	RFRLEN1	RFRLEN2	Frame Length
0	0 ≤ RFRLEN1 ≤ 127	Not used	(RFRLEN1 + 1) words
1	0 ≤ RFRLEN1 ≤ 127	0 ≤ RFRLEN2 ≤ 127	(RFRLEN1 + 1) + (RFRLEN2 + 1) words

13.5 Transmit Control Registers (XCR1 and XCR2)

Each McBSP has two transmit control registers. Table 106 and Table 108 describe the bits of XCR1 and XCR2, respectively. These I/O-mapped registers enable you to:

- ❏ Specify one or two phases for each frame of transmit data (XPHASE)
- ❏ Define two parameters for phase 1 and, if necessary, phase 2: the serial word length (XWDLEN1, XWDLEN2) and the number of words (XFRLEN1, XFRLEN2)
- ❏ Choose a transmit companding mode, if any (XCOMPAND)
- ❏ Enable or disable the transmit frame-sync ignore function (XFIG)
- ❏ Choose a transmit data delay (XDATDLY)

Table 106. Transmit Control 1 Register (XCR1)

Bit	Name	Value	Description	Type	Reset
15	Reserved	0	Reserved bit. Read-only; returns 0 when read.	R	0
14:8	XFRLEN1	0–127	Transmit frame length 1 (1 to 128 words). Each frame of transmit data can have one or two phases, depending on value that you load into the XPHASE bit. If a single-phase frame is selected, XFRLEN1 in XCR1 selects the number of serial words (8, 12, 16, 20, 24, or 32 bits per word) in the frame. If a dual-phase frame is selected, XFRLEN1 determines the number of serial words in phase 1 of the frame and XFRLEN2 in XCR2 determines the number of words in phase 2 of the frame. The 7-bit XFRLEN fields allow up to 128 words per phase. See Table 107 for a summary of how you determine the frame length. This length corresponds to the number of words or logical time slots or channels per frame-synchronization period. Program the XFRLEN 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 XFRLEN1.	R/W	0
7:5	XWDLEN1		Transmit word length 1. Each frame of transmit data 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 in XCR1 selects the length for every serial word transmitted 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 in XCR2 determines the word length in phase 2 of the frame.	R/W	000
		000b	8 bits		
		001b	12 bits		
		010b	16 bits		
		011b	20 bits		
		100b	24 bits		
		101b	32 bits		
		other	Reserved (do not use)		
4:0	Reserved	0	Reserved bits. They are read-only bits and return 0s when read.	R	0

Table 107. Frame Length Formula for Transmit Control 1 Register (XCR1)

XPHASE	XFRLEN1	XFRLEN2	Frame Length
0	0 ≤ XFRLEN1 ≤ 127	Not used	(XFRLEN1 + 1) words
1	0 ≤ XFRLEN1 ≤ 127	0 ≤ XFRLEN2 ≤ 127	(XFRLEN1 + 1) + (XFRLEN2 + 1) words

Table 108. Transmit Control 2 Register (XCR2)

Bit	Name	Value	Description	Type	Reset
15	XPHASE		Transmit phase number bit. XPHASE determines whether the transmit frame has one phase or two phases. For each phase you can define the serial word length and the number of serial words in the phase. To set up phase 1, program XWDLEN1 (word length) and XFRLEN1 (number of words). To set up phase 2 (if there are two phases), program XWDLEN2 and XFRLEN2.	R/W	0
		0	Single-phase frame The transmit frame has only one phase, phase 1.		
		1	Dual-phase frame The transmit frame has two phases, phase 1 and phase 2.		
14:8	XFRLEN2	0–127	Transmit frame length 2 (1 to 128 words). Each frame of transmit data can have one or two phases, depending on value that you load into the XPHASE bit. If a single-phase frame is selected, XFRLEN1 in XCR1 selects the number of serial words (8, 12, 16, 20, 24, or 32 bits per word) in the frame. If a dual-phase frame is selected, XFRLEN1 determines the number of serial words in phase 1 of the frame and XFRLEN2 in XCR2 determines the number of words in phase 2 of the frame. The 7-bit XFRLEN fields allow up to 128 words per phase. See Table 109 for a summary of how to determine the frame length. This length corresponds to the number of words or logical time slots or channels per frame-synchronization period. Program the XFRLEN 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 XFRLEN1.	R/W	0

Table 108. Transmit Control 2 Register (XCR2) (Continued)

Bit	Name	Value	Description	Type	Reset
7:5	XWDLEN2		Transmit word length 2. Each frame of transmit data 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 in XCR1 selects the length for every serial word transmitted 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 in XCR2 determines the word length in phase 2 of the frame.	R/W	000
		000b	8 bits		
		001b	12 bits		
		010b	16 bits		
		011b	20 bits		
		100b	24 bits		
		101b	32 bits		
		other	Reserved (do not use)		
4:3	XCOMPAND		Transmit companding mode bits. Companding (COMpress and exPAND) hardware allows compression and expansion of data in either μ-law or A-law format. For more details, see section 2.2, *Companding Data*.	R/W	00
			XCOMPAND allows you to choose one of the following companding modes for the McBSP transmitter. For more details about these companding modes, see section 2.2*, Companding (Compressing and Expanding) Data*.		
		00b	No companding, any size data, MSB transmitted first		
		01b	No companding, 8-bit data, LSB transmitted first		
		10b	μ-law companding, 8-bit data, MSB transmitted first		
		11b	A-law companding, 8-bit data, MSB transmitted first		

Table 108. Transmit Control 2 Register (XCR2) (Continued)

Bit	Name	Value	Description	Type	Reset
2	XFIG		Transmit frame-synchronization ignore bit. If a frame-synchronization pulse starts the transfer of a new frame before the current frame is fully transmitted, this pulse is treated as an unexpected frame-synchronization pulse. For more details about the frame-synchronization error condition, see section 4.5, *Unexpected Transmit Frame-Synchronization Pulse*. Setting XFIG causes the serial port to ignore unexpected frame-synchronization pulses during transmission. For more details on the effects of XFIG, see section 8.10, *Enable/Disable the Transmit Frame-Synchronization Ignore Function*.	R/W	0
		0	Frame-synchronization detect. An unexpected FSX pulse causes the transmitter to discard the content of XSR[1,2]. The transmitter: 1) Aborts the present transmission. 2) Sets XSYNCERR in SPCR2. 3) Begins a new transmission from DXR[1,2]. If new data was written to DXR[1,2] since the last DXR[1,2]-to-XSR[1,2] copy, the current value in XSR[1,2] is lost. Otherwise, the same data is transmitted.		
		1	Frame-synchronization ignore. An unexpected FSX pulse is ignored. Transmission continues uninterrupted.		

Table 108. Transmit Control 2 Register (XCR2) (Continued)

Bit	Name	Value	Description	Type	Reset
1:0	XDATDLY		Transmit data delay bits. XDATDLY specifies a data delay of 0, 1, or 2 transmit clock cycles after frame synchronization and before the transmission of the first bit of the frame. For more details, see section 8.12, *Set the Transmit Data Delay*.	R/W	00
		00b	0-bit data delay		
		01b	1-bit data delay		
		10b	2-bit data delay		
		11b	Reserved (do not use)		

Table 109. Frame Length Formula for Transmit Control 2 Register (XCR2)

XPHASE	XFRLEN1	XFRLEN2	Frame Length
0	0 ≤ XFRLEN1 ≤ 127	Not used	(XFRLEN1 + 1) words
1	0 ≤ XFRLEN1 ≤ 127	0 ≤ XFRLEN2 ≤ 127	(XFRLEN1 + 1) + (XFRLEN2 + 1) words

13.6 Sample Rate Generator Registers (SRGR1 and SRGR2)

Each McBSP has two sample rate generator registers. Table 110 and Table 111 describe the bits of SRGR1 and SRGR2, respectively. The sample rate generator can generate a clock signal (CLKG) and a frame-synchronization signal (FSG). The I/O-mapped registers SRGR1 and SRGR2 enable you to:

- Select the input clock source for the sample rate generator (CLKSM, in conjunction with the SCLKME bit of PCR).
- Divide down the frequency of CLKG (CLKGDV).
- Select whether internally-generated transmit frame-synchronization pulses are driven by FSG or by activity in the transmitter (FSGM).
- Specify the width of frame-synchronization pulses on FSG (FWID) and specify the period between those pulses (FPER).

When an external source (via the CLKS, CLKR, or CLKX pin) provides the input clock source for the sample rate generator:

- If the CLKS pin provides the input clock, the CLKSP bit in SRGR2 allows you to select whether the rising edge or the falling edge of CLKS triggers CLKG and FSG. If the CLKX/CLKR pin is used instead of the CLKS pin, the polarity of the input clock is selected with CLKXP/CLKRP of PCR.
- The GSYNC bit of SRGR2 allows you to make CLKG synchronized to an external frame-synchronization signal on the FSR pin, so that CLKG is kept in phase with the input clock.

Table 110. Sample Rate Generator 1 Register (SRGR1)

Bit	Name	Value	Description	Type	Reset
15:8	FWID	0 – 255	Frame-synchronization pulse width bits for FSG The sample rate generator can produce a clock signal, CLKG, and a frame-synchronization signal, FSG. For frame-synchronization pulses on FSG, (FWID + 1) is the pulse width in CLKG cycles. The eight bits of FWID allow a pulse width of 1 to 256 CLKG cycles: $0 \leq FWID \leq 255$ $1 \leq (FWID + 1) \leq 256$ CLKG cycles The period between the frame-synchronization pulses on FSG is defined by the FPER bits.	R/W	0
7:0	CLKGDV	0 – 255	Divide-down value for CLKG. The sample rate generator can accept an input clock signal and divide it down according to CLKGDV to produce an output clock signal, CLKG. The frequency of CLKG is: CLKG frequency = (Input clock frequency)/ (CLKGDV + 1) The input clock is selected by the SCLKME and CLKSM bits:	R/W	1

SCLKME	CLKSM	Input Clock For Sample Rate Generator
0	0	Signal on CLKS pin
0	1	CPU clock
1	0	Signal on CLKR pin
1	1	Signal on CLKX pin

A reset forces the CLKG frequency to 1/2 the input clock frequency (CLKGDV = 1), and selects the CPU clock as the input clock.

Table 111. Sample Rate Generator 2 Register (SRGR2)

Bit	Name	Value	Description	Type	Reset
15	GSYNC		Clock synchronization mode bit for CLKG. GSYNC is used only when the input clock source for the sample rate generator is external– on the CLKS, CLKR, or CLKX pin. When GSYNC = 1, the clock signal (CLKG) and the frame-synchronization signal (FSG) generated by the sample rate generator are made dependent on pulses on the FSR pin.	R/W	0
		0	No clock synchronization. CLKG oscillates without adjustment, and FSG pulses every (FPER + 1) CLKG cycles.		
		1	Clock synchronization. CLKG is adjusted as necessary so that it is synchronized with the input clock on the CLKS, CLKR, or CLKX pin, and FSG pulses. FSG only pulses in response to a pulse on the FSR pin. The frame-synchronization period defined in FPER is ignored. For more details, see section 3.3, *Synchronizing Sample Rate Generator Outputs to an External Clock*.		
14	CLKSP		CLKS pin polarity bit. CLKSP is used only when the CLKS pin is the input clock source for the sample rate generator. The bit determines which edge of CLKS drives the clock signal (CLKG) and the frame-synchronization signal (FSG) that are generated by the sample rate generator:	R/W	0
		0	A rising edge on the CLKS pin.		
		1	A falling edge on the CLKS pin.		
13	CLKSM		Sample rate generator input clock mode bit. The sample rate generator can accept an input clock signal and divide it down according to CLKGDV to produce an output clock signal, CLKG. The frequency of CLKG is: CLKG frequency = (input clock frequency)/ (CLKGDV + 1) CLKSM is used in conjunction with the SCLKME bit to determine the source for the input clock. A reset selects the CPU clock as the input clock and forces the CLKG frequency to 1/2 the CPU clock frequency.	R/W	1

Table 111. Sample Rate Generator 2 Register (SRGR2) (Continued)

Bit	Name	Value	Description	Type	Reset
		0	The input clock for the sample rate generator is either taken from the CLKS pin or the CLKR pin, depending on the value of the SCLKME bit of PCR: **SCLKME** / **CLKSM** / **Input Clock For Sample Rate Generator** 0 / 0 / Signal on CLKS pin 1 / 0 / Signal on CLKR pin		
		1	The input clock for the sample rate generator is either taken from the CPU clock or the CLKX pin, depending on the value of the SCLKME bit of PCR: **SCLKME** / **CLKSM** / **Input Clock For Sample Rate Generator** 0 / 1 / CPU clock 1 / 1 / Signal on CLKX pin		
12	FSGM		Sample rate generator transmit frame-synchronization mode bit. The transmitter can get frame synchronization from the FSX pin (FSXM = 0) or from inside the McBSP (FSXM = 1). When FSXM = 1, the FSGM bit determines how the McBSP supplies frame-synchronization pulses.	R/W	0
		0	If FSXM = 1, the McBSP generates a transmit frame-synchronization pulse when the content of DXR[1,2] is copied to XSR[1,2].		
		1	If FSXM = 1, the transmitter uses frame-synchronization pulses generated by the sample rate generator, which programs the FWID bits to set the width of each pulse, and programs the FPER bits to set the period between pulses.		
11:0	FPER	0−4095	Frame-synchronization period bits for FSG. The sample rate generator can produce a clock signal, CLKG, and a frame-synchronization signal, FSG. The period between frame-synchronization pulses on FSG is (FPER + 1) CLKG cycles. The 12 bits of FPER allow a frame-synchronization period of 1 to 4096 CLKG cycles: $0 \leq FPER \leq 4095$ $1 \leq (FPER + 1) \leq 4096$ CLKG cycles The width of each frame-synchronization pulse on FSG is defined by the FWID bits.	R/W	0

13.7 Multichannel Control Registers (MCR1 and MCR2)

Each McBSP has two multichannel control registers. MCR1 has control and status bits (with an R prefix) for multichannel selection operation in the receiver. MCR2 contains the same type of bits (bit with an X prefix) for the transmitter. The bits of MCR1 and MCR2 are described in Table 112 and Table 113, respectively. These I/O-mapped registers enable you to:

- Enable all channels or only selected channels for reception (RMCM).
- Choose which channels are enabled/disabled and masked/unmasked for transmission (XMCM).
- Specify whether two partitions (32 channels at a time) or eight partitions (128 channels at a time) can be used (RMCME for reception, XMCME for transmission).
- Assign blocks of 16 channels to partitions A and B when the 2-partition mode is selected (RPABLK and RPBBLK for reception, XPABLK and XPBBLK for transmission).
- Determine which block of 16 channels is currently involved in a data transfer (RCBLK for reception, XCBLK for transmission).

Table 112. Multichannel Control 1 Register (MCR1)s

Bit	Name	Value	Description	Type	Reset
15:10	Reserved	0	Reserved bits (not available for your use). They are read-only bits and return 0s when read.	R	0
9	RMCME		Receive multichannel partition mode bit. 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.	R/W	0
		0	2-partition mode Only partitions A and B are used. You can control up to 32 channels in the receive multichannel selection mode (RMCM = 1). Assign 16 channels to partition A with the 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		
		1	8-partition mode All partitions (A through H) are used. You can control up to 128 channels in the receive multichannel selection mode. You control the channels with the appropriate receive channel enable registers: RCERA: Channels 0 through 15 RCERB: Channels 16 through 31 RCERC: Channels 32 through 47 RCERD: Channels 48 through 63 RCERE: Channels 64 through 79 RCERF: Channels 80 through 95 RCERG: Channels 96 through 111 RCERH: Channels 112 through 127		

Table 112. Multichannel Control 1 Register (MCR1)s (Continued)

Bit	Name	Value	Description	Type	Reset
8:7	RPBBLK		Receive partition B block bits	R/W	00
			RPBBLK is only applicable if channels can be individually enabled or disabled (RMCM = 1) and the 2-partition mode is selected (RMCME = 0). Under these conditions, the McBSP receiver can accept or ignore data in any of the 32 channels that are assigned to partitions A and B of the receiver.		
			The 128 receive channels of the McBSP are divided equally among 8 blocks (0 through 7). When RPBBLK is applicable, use RPBBLK to assign one of the odd-numbered blocks (1, 3, 5, or 7) to partition B. Use the RPABLK bits to assign one of the even-numbered blocks (0, 2, 4, or 6) to partition A.		
			If you want to use more than 32 channels, you can change block assignments dynamically. You can assign a new block to one partition while the receiver is handling activity in the other partition. For example, while the block in partition A is active, you can change which block is assigned to partition B. The RCBLK bits are regularly updated to indicate which block is active.		
			When XMCM = 11b (for symmetric transmission and reception), the transmitter uses the receive block bits (RPABLK and RPBBLK) rather than the transmit block bits (XPABLK and XPBBLK).		
		00b	Block 1: channels 16 through 31		
		01b	Block 3: channels 48 through 63		
		10b	Block 5: channels 80 through 95		
		11b	Block 7: channels 112 through 127		

Table 112. Multichannel Control 1 Register (MCR1)s (Continued)

Bit	Name	Value	Description	Type	Reset
6:5	RPABLK		Receive partition A block bits	R/W	00
			RPABLK is only applicable if channels can be individually enabled or disabled (RMCM = 1) and the 2-partition mode is selected (RMCME = 0). Under these conditions, the McBSP receiver can accept or ignore data in any of the 32 channels that are assigned to partitions A and B of the receiver. See the description for RPBBLK (bits 8–7) for more information about assigning blocks to partitions A and B.		
		00b	Block 0: channels 0 through 15		
		01b	Block 2: channels 32 through 47		
		10b	Block 4: channels 64 through 79		
		11b	Block 6: channels 96 through 111		
4:2	RCBLK		Receive current block indicator. RCBLK indicates which block of 16 channels is involved in the current McBSP reception:	R	000
		000b	Block 0: channels 0 through 15		
		001b	Block 1: channels 16 through 31		
		010b	Block 2: channels 32 through 47		
		011b	Block 3: channels 48 through 63		
		100b	Block 4: channels 64 through 79		
		101b	Block 5: channels 80 through 95		
		110b	Block 6: channels 96 through 111		
		111b	Block 7: channels 112 through 127		

Table 112. Multichannel Control 1 Register (MCR1)s (Continued)

Bit	Name	Value	Description	Type	Reset
1	Reserved	0	Reserved bits (not available for your use). They are read-only bits and return 0s when read.	R	0
0	RMCM		Receive multichannel selection mode bit. RMCM determines whether all channels or only selected channels are enabled for reception:	R/W	0
		0	All 128 channels are enabled.		
		1	Multichannel selection mode. Channels can be individually enabled or disabled. The only channels enabled are those selected in the appropriate receive channel enable registers (RCERs). The way channels are assigned to the RCERs depends on the number of receive channel partitions (2 or 8), as defined by the RMCME bit.		

Table 113. Multichannel Control 2 Register (MCR2)

Bit	Name	Value	Description	Type	Reset
15:10	Reserved	0	Reserved bits (not available for your use). They are read-only bits and return 0s when read.	R	0
9	XMCME		Transmit multichannel partition mode bit. XMCME determines whether only 32 channels or all 128 channels are to be individually selectable. XMCME is only applicable if channels can be individually disabled/enabled or masked/unmasked for transmission (XMCM is nonzero).	R/W	0
		0	2-partition mode. Only partitions A and B are used. You can control up to 32 channels in the transmit multichannel selection mode selected with the XMCM bits. If XMCM = 01b or 10b, assign 16 channels to partition A with the XPABLK bits. Assign 16 channels to partition B with the XPBBLK bits. If XMCM = 11b (for symmetric transmission and reception), assign 16 channels to receive partition A with the RPABLK bits. Assign 16 channels to receive partition B with the RPBBLK bits. You control the channels with the appropriate transmit channel enable registers: XCERA: Channels in partition A XCERB: Channels in partition B		
		1	8-partition mode. All partitions (A through H) are used. You can control up to 128 channels in the transmit multichannel selection mode selected with the XMCM 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		

Table 113. Multichannel Control 2 Register (MCR2) (Continued)

Bit	Name	Value	Description	Type	Reset
8:7	XPBBLK		Transmit partition B block bits	R/W	00
			XPBBLK is only applicable if channels can be individually disabled/enabled and masked/unmasked (XMCM is nonzero) and the 2-partition mode is selected (XMCME = 0). Under these conditions, the McBSP transmitter can transmit or withhold data in any of the 32 channels that are assigned to partitions A and B of the transmitter.		
			The 128 transmit channels of the McBSP are divided equally among 8 blocks (0 through 7). When XPBBLK is applicable, use XPBBLK to assign one of the odd-numbered blocks (1, 3, 5, or 7) to partition B, as shown in the following table. Use the XPABLK bit to assign one of the even-numbered blocks (0, 2, 4, or 6) to partition A.		
			If you want to use more than 32 channels, you can change block assignments dynamically. You can assign a new block to one partition while the transmitter is handling activity in the other partition. For example, while the block in partition A is active, you can change which block is assigned to partition B. The XCBLK bits are regularly updated to indicate which block is active.		
			When XMCM = 11b (for symmetric transmission and reception), the transmitter uses the receive block bits (RPABLK and RPBBLK) rather than the transmit block bits (XPABLK and XPBBLK).		
		00b	Block 1: channels 16 through 31		
		01b	Block 3: channels 48 through 63		
		10b	Block 5: channels 80 through 95		
		11b	Block 7: channels 112 through 127		
6:5	XPABLK		Transmit partition A block bits. XPABLK is only applicable if channels can be individually disabled/enabled and masked/unmasked (XMCM is nonzero) and the 2-partition mode is selected (XMCME = 0). Under these conditions, the McBSP transmitter can transmit or withhold data in any of the 32 channels that are assigned to partitions A and B of the transmitter. See the description for XPBBLK (bits 8–7) for more information about assigning blocks to partitions A and B.	R/W	00

Table 113. Multichannel Control 2 Register (MCR2) (Continued)

Bit	Name	Value	Description	Type	Reset
		00b	Block 0: channels 0 through 15		
		01b	Block 2: channels 32 through 47		
		10b	Block 4: channels 64 through 79		
		11b	Block 6: channels 96 through 111		
4:2	XCBLK		Transmit current block indicator. XCBLK indicates which block of 16 channels is involved in the current McBSP transmission:	R	000
		000b	Block 0: channels 0 through 15		
		001b	Block 1: channels 16 through 31		
		010b	Block 2: channels 32 through 47		
		011b	Block 3: channels 48 through 63		
		100b	Block 4: channels 64 through 79		
		101b	Block 5: channels 80 through 95		
		110b	Block 6: channels 96 through 111		
		111b	Block 7: channels 112 through 127		
1:0	XMCM		Transmit multichannel selection mode bits. XMCM determines whether all channels or only selected channels are enabled and unmasked for transmission. For more details on how the channels are affected, see section 5.7, *Transmit Multichannel Selection Modes*.	R/W	00
		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 (XCERs). If enabled, a channel in this mode is also unmasked. The XMCME bit determines whether 32 channels or 128 channels are selectable in XCERs.		

Table 113. Multichannel Control 2 Register (MCR2) (Continued)

Bit	Name	Value	Description	Type	Reset
		10b	All channels are enabled, but they are masked unless they are selected in the appropriate transmit channel enable registers (XCERs). The XMCME bit determines whether 32 channels or 128 channels are selectable in XCERs.		
		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 (RCERs). Once enabled, they are masked unless they are also selected in the appropriate transmit channel enable registers (XCERs). The XMCME bit determines whether 32 channels or 128 channels are selectable in RCERs and XCERs.		

13.8 Pin Control Register (PCR)

Each McBSP has one pin control register. Table 114 describes the bits of PCR. This I/O-mapped register enables you to:

- ❑ Allow the McBSP to enter a low-power mode when the idle instruction is executed (IDLEEN, in conjunction with the PERI bit of ICR).
- ❑ Specify whether McBSP pins can be used as general-purpose I/O pins when the transmitter and/or receiver is in its reset state (XIOEN and RIOEN).
- ❑ Choose a frame-synchronization mode for the transmitter (FSXM) and for the receiver (FSRM).
- ❑ Choose a clock mode for transmitter (CLKXM) and for the receiver (CLKRM).
- ❑ Select the input clock source for the sample rate generator (SCLKME, in conjunction with the CLKSM bit of SRGR2).
- ❑ Read or write data when the CLKS, DX, and DR pins are configured as general-purpose I/O pins (CLKSSTAT, DXSTAT, and DXSTAT).
- ❑ Choose whether frame-synchronization signals are active low or active high (FSXP for transmission, FSRP for reception).
- ❑ Specify whether data is sampled on the falling edge or the rising edge of the clock signals (CLKXP for transmission, CLKRP for reception).

Table 114. Pin Control Register (PCR)

Bit	Name	Value	Description	Type	Reset
15	Reserved	0	Reserved bit (not available for your use). It is a read-only bit and returns a 0 when read.	R	0
14	IDLEEN		Idle enable bit. If the PERIPH idle domain is configured to be idle and IDLEEN = 1, the McBSP stops and enters a low-power state.	R/W	0
		0	The McBSP is running.		
		1	If the PERIPH domain is idle (PERIS = 1 in the idle status register), the McBSP is stopped in a low-power state.		
13	XIOEN		Transmit I/O enable bit. When the transmitter is in reset (XRST = 0), XIOEN can configure certain McBSP pins as general-purpose I/O (GPIO) pins. For a summary, see the table that follows the RIOEN bit description.	R/W	0
		0	The CLKX, FSX, DX, and CLKS pins are serial port pins.		
		1	If XRST = 0, the CLKX, FSX, and DX pins are GPIO pins. The CLKS is also a GPIO pin if RRST = 0 and RIOEN = 1.		
12	RIOEN		Receive I/O enable bit. When the receiver is in reset (RRST = 0), RIOEN can configure certain McBSP pins as general-purpose I/O (GPIO) pins . For a summary, see the table that follows the RIOEN bit description. XRST and RRST are in the serial port control registers, but all other bits mentioned in this table are in the pin control register.	R/W	0
		0	The CLKR, FSR, DR, and CLKS pins are serial port pins.		
		1	If RRST = 0, the CLKR, FSR, and DR pins are GPIO pins. The CLKS is also a GPIO pin if XRST = 0 and XIOEN = 1.		
11	FSXM		Transmit frame-synchronization mode bit. FSXM determines whether transmit frame-synchronization pulses are supplied externally or internally. The polarity of the signal on the FSX pin is determined by the FSXP bit.	R/W	0
		0	Transmit frame synchronization is supplied by an external source via the FSX pin.		
		1	Transmit frame synchronization is supplied by the McBSP, as determined by the FSGM bit of SRGR2.		

Table 114. Pin Control Register (PCR) (Continued)

Bit	Name	Value	Description	Type	Reset
10	FSRM		Receive frame-synchronization mode bit. FSRM determines whether receive frame-synchronization pulses are supplied externally or internally. The polarity of the signal on the FSR pin is determined by the FSRP bit.	R/W	0
		0	Receive frame synchronization is supplied by an external source via the FSR pin.		
		1	Receive frame synchronization is supplied by the sample rate generator. FSR is an output pin reflecting internal FSR, except when GSYNC = 1 in SRGR2.		
9	CLKXM		Transmit clock mode bit. CLKXM determines whether the source for the transmit clock is external or internal, and whether the CLKX pin is an input or an output. The polarity of the signal on the CLKX pin is determined by the CLKXP bit. 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 CLKX is an output. If the McBSP is a slave, make sure that CLKX is an input. Not in clock stop mode (CLKSTP = 00b or 01b):	R/W	0
		0	The transmitter gets its clock signal from an external source via the CLKX pin.		
		1	Internal CLKX is driven by the sample rate generator of the McBSP. The CLKX pin is an output pin that reflects internal CLKX. In clock stop mode (CLKSTP = 10b or 11b):		
		0	The McBSP is a slave in the SPI protocol. The internal transmit clock (CLKX) is driven by the SPI master via the CLKX pin. The internal receive clock (CLKR) is driven internally by CLKX, so that both the transmitter and the receiver are controlled by the external master clock.		
		1	The McBSP is a master in the SPI protocol. The sample rate generator drives the internal transmit clock (CLKX). Internal CLKX is reflected on the CLKX pin to drive the shift clock of the SPI-compliant slaves in the system. Internal CLKX also drives the internal receive clock (CLKR), so that both the transmitter and the receiver are controlled by the internal master clock.		

Table 114. Pin Control Register (PCR) (Continued)

Bit	Name	Value	Description	Type	Reset
8	CLKRM		Receive clock mode bit. The role of CLKRM and the resulting effect on the CLKR pin depend on whether the McBSP is in the digital loopback mode (DLB = 1). The polarity of the signal on the CLKR pin is determined by the CLKRP bit. Not in digital loopback mode (DLB = 0):	R/W	0
		0	The CLKR pin is an input pin that supplies the internal receive clock (CLKR).		
		1	Internal CLKR is driven by the sample rate generator of the McBSP. The CLKR pin is an output pin that reflects internal CLKR. In digital loopback mode (DLB = 1):		
		0	The CLKR pin is in the high impedance state. The internal receive clock (CLKR) is driven by the internal transmit clock (CLKX). CLKX is derived according to the CLKXM bit.		
		1	Internal CLKR is driven by internal CLKX. The CLKR pin is an output pin that reflects internal CLKR. CLKX is derived according to the CLKXM bit.		
7	SCLKME		Sample rate generator input clock mode bit. The sample rate generator can produce a clock signal, CLKG. The frequency of CLKG is: CLKG freq. = (Input clock frequency) / (CLKGDV + 1) SCLKME is used in conjunction with the CLKSM bit to select the input clock.	R/W	0
		0	The input clock for the sample rate generator is taken from the CLKS pin or from the CPU clock, depending on the value of the CLKSM bit of SRGR2:		

SCLKME	CLKSM	Input Clock For Sample Rate Generator
0	0	Signal on CLKS pin
0	1	CPU clock

Table 114. Pin Control Register (PCR) (Continued)

Bit	Name	Value	Description	Type	Reset
7 (Cont)	SCLKME	1	The input clock for the sample rate generator is taken from the CLKR pin or from the CLKX pin, depending on the value of the CLKSM bit of SRGR2:		
			SCLKME **CLKSM** **Input Clock For Sample Rate Generator**		
			1 0 Signal on CLKR pin		
			1 1 Signal on CLKX pin		
6	CLKSSTAT		CLKS pin status bit. When CLKSSTAT is applicable, it reflects the level on the CLKS pin.	R	0
			CLKSSTAT is only applicable when the transmitter and receiver are both in reset (XRST = RRST = 0) and CLKS is configured for use as a general-purpose input pin (XIOEN = RIOEN = 1).		
		0	The signal on the CLKS pin is low.		
		1	The signal on the CLKS pin is high.		
5	DXSTAT		DX pin status bit. When DXSTAT is applicable, you can toggle the signal on DX by writing to DXSTAT.	R/W	0
			DXSTAT is only applicable when the transmitter is in reset (XRST = 0) and DX is configured for use as a general purpose output pin (XIOEN = 1).		
		0	Drive the signal on the DX pin low.		
		1	Drive the signal on the DX pin high.		
4	DRSTAT		DR pin status bit. When DRSTAT is applicable, it reflects the level on the DR pin.	R	0
			DRSTAT is only applicable when the receiver is in reset (RRST = 0) and DR is configured for use as a general-purpose input pin (RIOEN = 1).		
		0	The signal on DR pin is low.		
		1	The signal on DR pin is high.		
3	FSXP		Transmit frame-synchronization polarity bit. FSXP determines the polarity of FSX as seen on the FSX pin.	R/W	0
		0	Transmit frame-synchronization pulses are active high.		
		1	Transmit frame-synchronization pulses are active low.		

Table 114. Pin Control Register (PCR) (Continued)

Bit	Name	Value	Description	Type	Reset
2	FSRP		Receive frame-synchronization polarity bit. FSRP determines the polarity of FSR as seen on the FSR pin.	R/W	0
		0	Receive frame-synchronization pulses are active high.		
		1	Receive frame-synchronization pulses are active low.		
1	CLKXP		Transmit clock polarity bit. CLKXP determines the polarity of CLKX as seen on the CLKX pin.	R/W	0
		0	Transmit data is sampled on the rising edge of CLKX.		
		1	Transmit data is sampled on the falling edge of CLKX.		
0	CLKRP		Receive clock polarity bit. CLKRP determines the polarity of CLKR as seen on the CLKR pin.	R/W	0
		0	Receive data is sampled on the falling edge of CLKR.		
		1	Receive data is sampled on the rising edge of CLKR.		

Table 115. Bit Configuration for GPIOs

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
CLKX	XRST = 0 XIOEN = 1	CLKXM = 1	CLKXP	CLKXM = 0	CLKXP
FSX	XRST = 0 XIOEN = 1	FSXM = 1	FSXP	FSXM = 0	FSXP
DX	XRST = 0 XIOEN = 1	Always	DXSTAT	Never	Does not apply
CLKR	RRST = 0 RIOEN = 1	CLKRM = 1	CLKRP	CLKRM = 0	CLKRP
FSR	RRST = 0 RIOEN = 1	FSRM = 1	FSRP	FSRM = 0	FSRP
DR	RRST = 0 RIOEN = 1	Never	Does not apply	Always	DRSTAT
CLKS	RRST = XRST = 0 RIOEN = XIOEN = 1	Never	Does not apply	Always	CLKSSTAT

13.9 Receive Channel Enable Registers (RCERA, RCERB, RCERC, RCERD, RCERE, RCERF, RCERG, RCERH)

Each McBSP has eight receive channel enable registers of the format shown in Figure 75. There is one enable register for each of the receive partitions: A, B, C, D, E, F, G, and H. Table 116 provides a summary description that applies to any bit x of a receive channel enable register.

These I/O-mapped registers are only used when the receiver is configured to allow individual enabling and disabling of the channels (RMCM = 1).

Figure 75. Receive Channel Enable Registers (RCERA...RCERH)

15	14	13	12	11	10	9	8
RCE15	RCE14	RCE13	RCE12	RCE11	RCE10	RCE9	RCE8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
RCE7	RCE6	RCE5	RCE4	RCE3	RCE2	RCE1	RCE0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Legend: R = read; W = write; -*n* = value after reset

Table 116. Receive Channel Enable Registers (RCERA...RCERH)

Bit	Name	Value	Description	
x	RCEx		Receive channel enable bit. The role of this bit depends on whether it is used to support the receive multichannel selection mode reception.	
			For receive multichannel selection mode (RMCM = 1):	
		0	Disable the channel that is mapped to RCEx.	
		1	Enable the channel that is mapped to RCEx.	
			RCEx	**Bit x of Incoming Value**
			RCE15	Bit 15: First bit to arrive
			RCE14	Bit 14: Second bit to arrive
			RCE13	Bit 13: Third bit to arrive
			:	:
			RCE0	Bit 0: Last bit to arrive

13.9.1 RCERs Used in the Receive Multichannel Selection Mode

For multichannel selection operation, the assignment of channels to the RCERs depends on whether 32 or 128 channels are individually selectable, as defined by the RMCME bit. For each of these two cases, Table 117 shows which block of channels is assigned to each of the RCERs used. For each RCER, the table shows which channel is assigned to each of the bits.

Table 117. Use of the Receive Channel Enable Registers

Number of Selectable Channels	Block Assignments		Channel Assignments	
	RCERx	Block Assigned	Bit in RCERx	Channel Assigned
32 (RMCME = 0)	RCERA	Channels n to (n + 15) The block of channels is chosen with the RPABLK bits.	RCE0 RCE1 RCE2 : RCE15	Channel n Channel (n + 1) Channel (n + 2) : Channel (n + 15)
	RCERB	Channels m to (m + 15) The block of channels is chosen with the RPBBLK bits.	RCE0 RCE1 RCE2 : RCE15	Channel m Channel (m + 1) Channel (m + 2) : Channel (m + 15)

Table 117. Use of the Receive Channel Enable Registers (Continued)

Number of Selectable Channels	Block Assignments		Channel Assignments	
	RCERx	Block Assigned	Bit in RCERx	Channel Assigned
128 (RMCME = 1)	RCERA	Block 0	RCE0 RCE1 RCE2 : RCE15	Channel 0 Channel 1 Channel 2 : Channel 15
	RCERB	Block 1	RCE0 RCE1 RCE2 : RCE15	Channel 16 Channel 17 Channel 18 : Channel 31
	RCERC	Block 2	RCE0 RCE1 RCE2 : RCE15	Channel 32 Channel 33 Channel 34 : Channel 47
	RCERD	Block 3	RCE0 RCE1 RCE2 : RCE15	Channel 48 Channel 49 Channel 50 : Channel 63
	RCERE	Block 4	RCE0 RCE1 RCE2 : RCE15	Channel 64 Channel 65 Channel 66 : Channel 79
	RCERF	Block 5	RCE0 RCE1 RCE2 : RCE15	Channel 80 Channel 81 Channel 82 : Channel 95
	RCERG	Block 6	RCE0 RCE1 RCE2 : RCE15	Channel 96 Channel 97 Channel 98 : Channel 111
	RCERH	Block 7	RCE0 RCE1 RCE2 : RCE15	Channel 112 Channel 113 Channel 114 : Channel 127

13.10 Transmit Channel Enable Registers (XCERA, XCERB, XCERC, XCERD, XCERE, XCERF, XCERG, XCERH)

Each McBSP has eight transmit channel enable registers of the form shown in Figure 76. There is one for each of the transmit partitions: A, B, C, D, E, F, G, and H. Table 118 provides a summary description that applies to each bit XCEx of a transmit channel enable register.

The I/O-mapped XCERs are only used when the transmitter is configured to allow individual disabling/enabling and masking/unmasking of the channels (XMCM is nonzero).

Figure 76. Transmit Channel Enable Registers (XCERA...XCERH)

15	14	13	12	11	10	9	8
XCE15	XCE14	XCE13	XCE12	XCE11	XCE10	XCE9	XCE8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
XCE7	XCE6	XCE5	XCE4	XCE3	XCE2	XCE1	XCE0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Legend: R = read; W = write; -*n* = value after reset

Table 118. Transmit Channel Enable Registers (XCERA...XCERH)

Bit	Name	Value	Description
x	XCEx		Transmit channel enable bit. The role of this bit depends on which transmit multichannel selection mode is selected with the XMCM bits.
			For multichannel selection when XMCM = 01b (all channels disabled unless selected):
		0	Disable and mask the channel that is mapped to XCEx.
		1	Enable and unmask the channel that is mapped to XCEx.
			For multichannel selection when XMCM = 10b (all channels enabled but masked unless selected):
		0	Mask the channel that is mapped to XCEx.
		1	Unmask the channel that is mapped to XCEx.
			For multichannel selection when XMCM = 11b (all channels masked unless selected):
		0	Mask the channel that is mapped to XCEx. Even if the channel is enabled by the corresponding receive channel enable bit, this channel's data cannot appear on the DX pin.
		1	Unmask the channel that is mapped to XCEx. If the channel is also enabled by the corresponding receive channel enable bit, full transmission can occur.
			XCEx **Bit x of DXR1 Value**
			XCE15 Bit 15 (MSB) in DXR1
			XCE14 Bit 14 in DXR1
			XCE13 Bit 13 in DXR1
			: :
			XCE0 Bit 0 (LSB) in DXR1

13.10.1 XCERs Used in a Transmit Multichannel Selection Mode

For multichannel selection operation, the assignment of channels to the XCERs depends on whether 32 or 128 channels are individually selectable, as defined by the XMCME bit. These two cases are shown in Table 119. The table shows which block of channels is assigned to each XCER that is used. For each XCER, the table shows which channel is assigned to each of the bits.

Note:

When XMCM = 11b (for symmetric transmission and reception), the transmitter uses the receive channel enable registers (RCERs) to enable channels and uses the XCERs to unmask channels for transmission.

Table 119. Use of the Transmit Channel Enable Registers in a Transmit Multichannel Selection Mode

Number of Selectable Channels	Block Assignments		Channel Assignments	
	XCERx	Block Assigned	Bit in XCERx	Channel Assigned
32 (XMCME = 0)	XCERA	Channels n to (n + 15) When XMCM = 01b or 10b, the block of channels is chosen with the XPABLK bits. When XMCM = 11b, the block is chosen with the RPABLK bits.	XCE0 XCE1 XCE2 : XCE15	Channel n Channel (n + 1) Channel (n + 2) : Channel (n + 15)
	XCERB	Channels m to (m + 15) When XMCM = 01b or 10b, the block of channels is chosen with the XPBBLK bits. When XMCM = 11b, the block is chosen with the RPBBLK bits.	XCE0 XCE1 XCE2 : XCE15	Channel m Channel (m + 1) Channel (m + 2) : Channel (m + 15)
128 (XMCME = 1)	XCERA	Block 0	XCE0 XCE1 XCE2 : XCE15	Channel 0 Channel 1 Channel 2 : Channel 15
	XCERB	Block 1	XCE0 XCE1 XCE2 : XCE15	Channel 16 Channel 17 Channel 18 : Channel 31
	XCERC	Block 2	XCE0 XCE1 XCE2 : XCE15	Channel 32 Channel 33 Channel 34 : Channel 47

Table 119. Use of the Transmit Channel Enable Registers in a Transmit Multichannel Selection Mode (Continued)

Number of Selectable Channels	Block Assignments		Channel Assignments	
	XCERx	**Block Assigned**	**Bit in XCERx**	**Channel Assigned**
	XCERD	Block 3	XCE0	Channel 48
			XCE1	Channel 49
			XCE2	Channel 50
			:	:
			XCE15	Channel 63
	XCERE	Block 4	XCE0	Channel 64
			XCE1	Channel 65
			XCE2	Channel 66
			:	:
			XCE15	Channel 79
	XCERF	Block 5	XCE0	Channel 80
			XCE1	Channel 81
			XCE2	Channel 82
			:	:
			XCE15	Channel 95
	XCERG	Block 6	XCE0	Channel 96
			XCE1	Channel 97
			XCE2	Channel 98
			:	:
			XCE15	Channel 111
	XCERH	Block 7	XCE0	Channel 112
			XCE1	Channel 113
			XCE2	Channel 114
			:	:
			XCE15	Channel 127

14 McBSP Register Worksheet

This register worksheet is meant to be printed and used as a guide for configuring the McBSP registers. Each figure on the worksheet provides space in every register field for entering the binary value that must be loaded into that field. For read-only fields, you can use 0s or 1s. When all of the fields have been filled in, you can use the line above the register figure to record the corresponding hexadecimal value to load into the register during initialization.

14.1 General Control Registers

SPCR1 – Initialization Value: _____

15	14–13	12–11	10–8
DLB	RJUST	CLKSTP	Reserved
			Read-only

7	6	5–4	3	2	1	0
DXENA	Reserved	RINTM	RSYNCERR	RFULL	RRDY	$\overline{\text{RRST}}$
				Read-only	Read-only	

SPCR2 – Initialization Value: _____

15–10	9	8
Reserved	FREE	SOFT
Read-only		

7	6	5–4	3	2	1	0
$\overline{\text{GRST}}$	$\overline{\text{GRST}}$	XINTM	XSYNCERR	XEMPTY_	XRDY	$\overline{\text{XRST}}$
				Read-only	Read-only	

PCR – Initialization Value: _____

15	14	13	12	11	10	9	8
Reserved	IDLE_EN	XIOEN	RIOEN	FSXM	FSRM	CLKXM	CLKRM

Read-only

7	6	5	4	3	2	1	0
SCLKME	CLKS_STAT	DX_STAT	DR_STAT	FSXP	FSRP	CLKXP	CLKRP

Read-only Read-only

RCR1 – Initialization Value: _____

15	14–8	7–5	4–0
Reserved	RFRLEN1	RWDLEN1	Reserved

Read-only Read-only

RCR2 – Initialization Value: _____

15	14–8	7–5	4–3	2	1–0
RPHASE	RFRLEN2	RWDLEN2	RCOMPAND	RFIG	RDATDLY

XCR1 – Initialization Value: _____

15	14–8	7–5	4–0
Reserved	XFRLEN1	XWDLEN1	Reserved

Read-only Read-only

XCR2 – Initialization Value: _____

15	14–8	7–5	4–3	2	1–0
XPHASE	XFRLEN2	XWDLEN2	XCOMPAND	XFIG	XDATDLY

SRGR1 – Initialization Value: _____

15–8	7–0
FWID	CLKGDV

SRGR2 – Initialization Value: _____

15	14	13	12	11–0
GSYNC	CLKSP	CLKSM	FSGM	FPER

14.2 Multichannel Selection Control Registers

MCR1 – Initialization Value: _____

15–10	9	8–7	6–5	4–2	1	0
Reserved	RMCME	RPBBLK	RPABLK	RCBLK	Reserved	RMCM
Read-only				Read-only	Read-only	

MCR2 – Initialization Value: _____

15–10	9	8–7	6–5	4–2	1–0
Reserved	XMCME	XPBBLK	XPABLK	XCBLK	XMCM
Read-only				Read-only	

RCERA – Initialization Value: _____

15	14	13	12	11	10	9	8
RCEA15	RCEA14	RCEA13	RCEA12	RCEA11	RCEA10	RCEA9	RCEA8
Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____

7	6	5	4	3	2	1	0
RCEA7	RCEA6	RCEA5	RCEA4	RCEA3	RCEA2	RCEA1	RCEA0
Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____

RCERB – Initialization Value: _____

15	14	13	12	11	10	9	8
RCEB15	RCEB14	RCEB13	RCEB12	RCEB11	RCEB10	RCEB9	RCEB8
Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____

7	6	5	4	3	2	1	0
RCEB7	RCEB6	RCEB5	RCEB4	RCEB3	RCEB2	RCEB1	RCEB0
Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____

RCERC – Initialization Value: _____

15	14	13	12	11	10	9	8
RCEC15	RCEC14	RCEC13	RCEC12	RCEC11	RCEC10	RCEC9	RCEC8
Channel 47	Channel 46	Channel 45	Channel 44	Channel 43	Channel 42	Channel 41	Channel 40

7	6	5	4	3	2	1	0
RCEC7	RCEC6	RCEC5	RCEC4	RCEC3	RCEC2	RCEC1	RCEC0
Channel 39	Channel 38	Channel 37	Channel 36	Channel 35	Channel 34	Channel 33	Channel 32

RCERD – Initialization Value: _____

15	14	13	12	11	10	9	8
RCED15	RCED14	RCED13	RCED12	RCED11	RCED10	RCED9	RCED8
Channel 63	Channel 62	Channel 61	Channel 60	Channel 59	Channel 58	Channel 57	Channel 56

7	6	5	4	3	2	1	0
RCED7	RCED6	RCED5	RCED4	RCED3	RCED2	RCED1	RCED0
Channel 55	Channel 54	Channel 53	Channel 52	Channel 51	Channel 50	Channel 49	Channel 48

RCERE – Initialization Value: _____

15	14	13	12	11	10	9	8
RCEE15	RCEE14	RCEE13	RCEE12	RCEE11	RCEE10	RCEE9	RCEE8
Channel 79	Channel 78	Channel 77	Channel 76	Channel 75	Channel 74	Channel 73	Channel 72

7	6	5	4	3	2	1	0
RCEE7	RCEE6	RCEE5	RCEE4	RCEE3	RCEE2	RCEE1	RCEE0
Channel 71	Channel 70	Channel 69	Channel 68	Channel 67	Channel 66	Channel 65	Channel 64

RCERF – Initialization Value: _____

15	14	13	12	11	10	9	8
RCEF15	RCEF14	RCEF13	RCEF12	RCEF11	RCEF10	RCEF9	RCEF8
Channel 95	Channel 94	Channel 93	Channel 92	Channel 91	Channel 90	Channel 89	Channel 88

7	6	5	4	3	2	1	0
RCEF7	RCEF6	RCEF5	RCEF4	RCEF3	RCEF2	RCEF1	RCEF0
Channel 87	Channel 86	Channel 85	Channel 84	Channel 83	Channel 82	Channel 81	Channel 80

RCERG – Initialization Value: _____

15	14	13	12	11	10	9	8
RCEG15	RCEG14	RCEG13	RCEG12	RCEG11	RCEG10	RCEG9	RCEG8
Channel 111	Channel 110	Channel 109	Channel 108	Channel 107	Channel 106	Channel 105	Channel 104

7	6	5	4	3	2	1	0
RCEG7	RCEG6	RCEG5	RCEG4	RCEG3	RCEG2	RCEG1	RCEG0
Channel 103	Channel 102	Channel 101	Channel 100	Channel 99	Channel 98	Channel 97	Channel 96

RCERH – Initialization Value: _____

15	14	13	12	11	10	9	8
RCEH15	RCEH14	RCEH13	RCEH12	RCEH11	RCEH10	RCEH9	RCEH8
Channel 127	Channel 126	Channel 125	Channel 124	Channel 123	Channel 122	Channel 121	Channel 120

7	6	5	4	3	2	1	0
RCEH7	RCEH6	RCEH5	RCEH4	RCEH3	RCEH2	RCEH1	RCEH0
Channel 119	Channel 118	Channel 117	Channel 116	Channel 115	Channel 114	Channel 113	Channel 112

XCERA – Initialization Value: _____

15	14	13	12	11	10	9	8
XCEA15	XCEA14	XCEA13	XCEA12	XCEA11	XCEA10	XCEA9	XCEA8
Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____

7	6	5	4	3	2	1	0
XCEA7	XCEA6	XCEA5	XCEA4	XCEA3	XCEA2	XCEA1	XCEA0
Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____

XCERB – Initialization Value: _____

15	14	13	12	11	10	9	8
XCEB15	XCEB14	XCEB13	XCEB12	XCEB11	XCEB10	XCEB9	XCEB8
Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____

7	6	5	4	3	2	1	0
XCEB7	XCEB6	XCEB5	XCEB4	XCEB3	XCEB2	XCEB1	XCEB0
Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____

XCERC – Initialization Value: _____

15	14	13	12	11	10	9	8
XCEC15	XCEC14	XCEC13	XCEC12	XCEC11	XCEC10	XCEC9	XCEC8
Channel 47	Channel 46	Channel 45	Channel 44	Channel 43	Channel 42	Channel 41	Channel 40

7	6	5	4	3	2	1	0
XCEC7	XCEC6	XCEC5	XCEC4	XCEC3	XCEC2	XCEC1	XCEC0
Channel 39	Channel 38	Channel 37	Channel 36	Channel 35	Channel 34	Channel 33	Channel 32

XCERD – Initialization Value: _____

15	14	13	12	11	10	9	8
XCED15	XCED14	XCED13	XCED12	XCED11	XCED10	XCED9	XCED8
Channel 63	Channel 62	Channel 61	Channel 60	Channel 59	Channel 58	Channel 57	Channel 56

7	6	5	4	3	2	1	0
XCED7	XCED6	XCED5	XCED4	XCED3	XCED2	XCED1	XCED0
Channel 55	Channel 54	Channel 53	Channel 52	Channel 51	Channel 50	Channel 49	Channel 48

XCERE – Initialization Value: _____

15	14	13	12	11	10	9	8
XCEE15	XCEE14	XCEE13	XCEE12	XCEE11	XCEE10	XCEE9	XCEE8
Channel 79	Channel 78	Channel 77	Channel 76	Channel 75	Channel 74	Channel 73	Channel 72

7	6	5	4	3	2	1	0
XCEE7	XCEE6	XCEE5	XCEE4	XCEE3	XCEE2	XCEE1	XCEE0
Channel 71	Channel 70	Channel 69	Channel 68	Channel 67	Channel 66	Channel 65	Channel 64

XCERF – Initialization Value: _____

15	14	13	12	11	10	9	8
XCEF15	XCEF14	XCEF13	XCEF12	XCEF11	XCEF10	XCEF9	XCEF8
Channel 95	Channel 94	Channel 93	Channel 92	Channel 91	Channel 90	Channel 89	Channel 88

7	6	5	4	3	2	1	0
XCEF7	XCEF6	XCEF5	XCEF4	XCEF3	XCEF2	XCEF1	XCEF0
Channel 87	Channel 86	Channel 85	Channel 84	Channel 83	Channel 82	Channel 81	Channel 80

XCERG – Initialization Value: _____

15	14	13	12	11	10	9	8
XCEG15	XCEG14	XCEG13	XCEG12	XCEG11	XCEG10	XCEG9	XCEG8
Channel 111	Channel 110	Channel 109	Channel 108	Channel 107	Channel 106	Channel 105	Channel 104

7	6	5	4	3	2	1	0
XCEG7	XCEG6	XCEG5	XCEG4	XCEG3	XCEG2	XCEG1	XCEG0
Channel 103	Channel 102	Channel 101	Channel 100	Channel 99	Channel 98	Channel 97	Channel 96

XCERH – Initialization Value: _____

15	14	13	12	11	10	9	8
XCEH15	XCEH14	XCEH13	XCEH12	XCEH11	XCEH10	XCEH9	XCEH8
Channel 127	Channel 126	Channel 125	Channel 124	Channel 123	Channel 122	Channel 121	Channel 120

7	6	5	4	3	2	1	0
XCEH7	XCEH6	XCEH5	XCEH4	XCEH3	XCEH2	XCEH1	XCEH0
Channel 119	Channel 118	Channel 117	Channel 116	Channel 115	Channel 114	Channel 113	Channel 112

Index

E

F

G

I

J

M

N

O

P

R

S

T

U

W

OMAP5912 Multimedia Processor Camera Interface Reference Guide

Literature Number: SPRU763B
October 2004

OMAP™
TEXAS INSTRUMENTS TECHNOLOGY

TEXAS INSTRUMENTS

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.

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
		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 document describes the camera interface implemented in the OMAP5912 multimedia processor.

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.

Related Documentation From Texas Instruments

Documentation that describes the OMAP5912 device, related peripherals, and other technical collateral, is available in the OMAP5912 Product Folder on TI's website: www.ti.com/omap5912.

Trademarks

OMAP and the OMAP symbol are trademarks of Texas Instruments.

Contents

Figures

Tables

This page is intentionally left blank.

Camera Interface

This document describes the camera interface implemented in the OMAP5912 multimedia processor.

1 Camera Parallel Interface

A 32-bit camera interface connects a camera module to the MPU peripheral bus of the device. The interface handles multiple image formats synchronized on vertical and horizontal synchronization signals. Data transfer between the camera and the interface can be done synchronously or asynchronously. The data is stored in a buffer to be sent over the peripheral bus, using the DMA mode or the CPU mode (bypass mode).

The interface supports 8-bit parallel image data ports and horizontal/vertical signal ports separately (stand-alone synchronous method). The camera interface has a DMA port.

1.1 Functional Architecture

The camera architecture consists of four functional blocks:

- Buffer: Stores the data word received from the camera module and transfers it to the MPU peripheral bridge, using the DMA mode or the CPU mode. It contains a 128-word FIFO.

  The 8-bit data received from the camera module is packed into 32 bits. A 128-bit-deep FIFO is implemented to provide local buffering of the data and to control the DMA request when the camera interface is enabled in DMA mode. The main goals of this mode are:

  - To discharge the CPU of the data transfer
  - To reduce the real time constraints of the DMA read (FIFO buffering part)
  - To group the x DMA accesses in only one time slot (FIFO block part)

  The user can, however, forward a direct transfer to the CPU in bypass mode by disabling the DMA request line.

- Clock divider: Manages the clock division and handles the external clock generation for synchronous/asynchronous mode gating.
- Interrupt generator: Generates an interrupt to indicate the start and end of frame, start and end of image, and FIFO overflow.
- Registers: Connect status, control, and data 32-bit registers.

1.1.1 Camera IF Clocks

The camera IF functional clock is either the ARMPER_CK divided down (see the CONTROLCLK.FOCSMOD bits for divide down options) or CAM.LCLK (input pin J15).

The camera IF interface clock is ARMPER_CK.

1.1.2 Camera Data Validation

The incoming byte on CAM_D can be latched on the rising or falling edge of CAM.LCLK generated by the camera itself. The POLCLK bit in the clock control register selects the polarity of CAM.LCLK.

The camera interface must be programmed so that data is always captured opposite the launch edge. For example, if data is latched by the sensor on the rising edge of CAM.LCLK, the interface must be configured to catch the data on the falling edge of CAM.LCLK.

The high level of the vertical synchronous and horizontal synchronous signals indicates that the data is valid on CAM_D. This level is registered in VSTATUS and HSTATUS, which are updated at edge detection of vertical and horizontal synchronous signals.

Figure 1 shows the image data transfer, and Figure 2 shows the timing chart.

Figure 1. Image Data Transfer

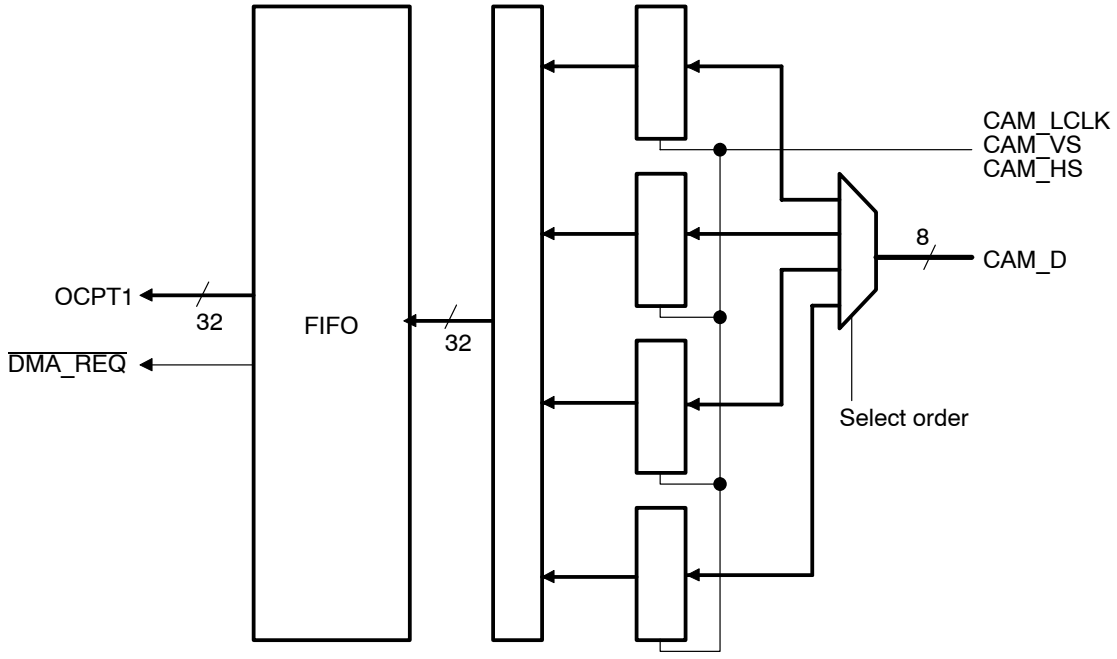
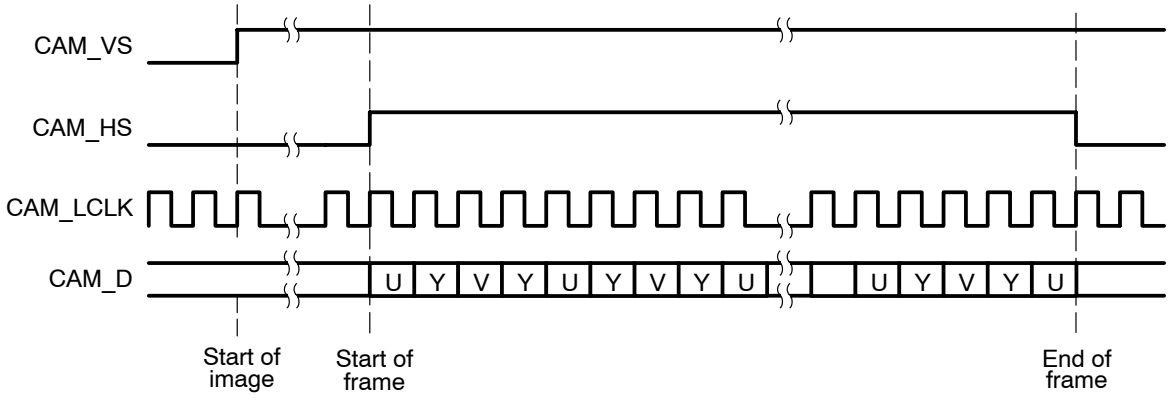


Figure 2. Timing Chart of Image Data Transfer (POLCLK = 1)



The clock can be gated during the VSYNC and/or HSYNC blanking periods, but it must be left to run because a process based on LCLK clears all internal resynchronization registers while VSYNC or HSYNC is low, before starting a new frame or new image. This mechanism prevents the FIFO from retaining any remaining (and likely garbage) data left over from a previous frame, which could corrupt the data of a new frame.

If either CAM_VS or CAM_HS goes inactive before receiving all four bytes, the data in buffers is cleared by the active CAM.LCLK edge and is not written into FIFO.

1.1.3 Autostart

Autostart is a protection function that prevents a start of capture during an image transfer. Autostart is launched after enabling the LCLK and waits for the next inactive level of CAM_VS to enable the data capture so that the transfer starts at the beginning of the image.

Note:

If a reset FIFO occurs (see section 1.1.4) while the interface is latching data, the capture is automatically disabled and the autostart function is enabled.

1.1.4 Reset FIFO

An active-high reset FIFO is implemented at the RAZ_FIFO bit (18) of the camera mode register. This feature clears any remaining data in the FIFO before starting a new transfer. It also resets all status and control signals around the FIFO, such as the read and write pointers, the FIFO full interrupt, the FIFO peak counter, and the 32-bit resynchronization registers.

1.1.5 Set of Order

Each four bytes received from the camera must be packed and can be swapped to follow the order YUV specified in the camera mode register by ORDERCAMD. Figure 3 and Figure 4 show the camera data order not swapped and swapped, respectively.

Figure 3. Order of Camera Data on TIPB (Not Swapped)

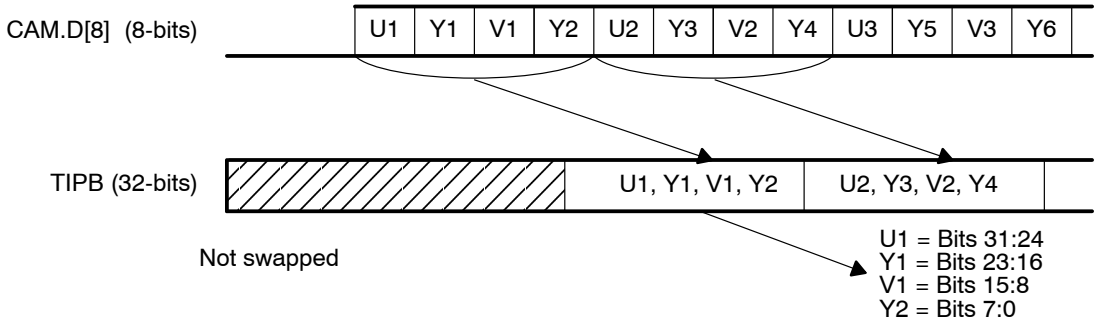
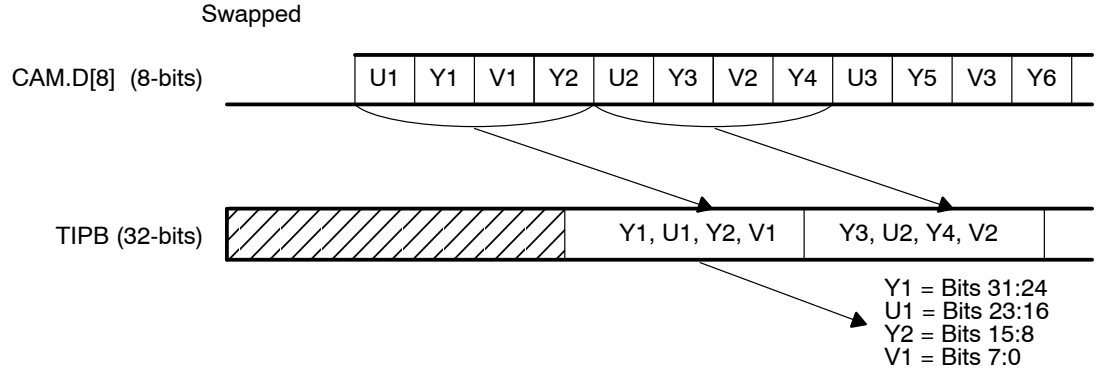


Figure 4. Order of Camera Data on TIPB (Swapped)



1.1.6 FIFO Buffer (128 x 32)

A write access is applied to the FIFO for each 32-bit word received. When the write FIFO counter reaches the trigger level, an interrupt request can be generated. The trigger level is programmable.

In DMA mode, the threshold can be programmed between 1 and 128, but the DMA must be set up to read the threshold amount out of FIFO per the DMA request issued by the camera interface. Otherwise, the locking mechanism is never rearmed, and it prevents DMA requests from being issued after every read.

A pulse on the DMA request (see Figure 5 and Figure 6) occurs when the number of words in the FIFO is above the threshold. The DMA request occurs if the number of remaining words is above the threshold and the system DMA has completed the transfer (number of words read by the DMA = threshold).

The camera FIFO continues to fill (up to its maximum 128 values) when an interrupt or DMA request has been generated but not responded to yet. When a data value is read from the camera FIFO, another IRQ or DMA request is immediately generated, as long as the amount of data present in the FIFO is above the trigger level.

Figure 5. DMA Request

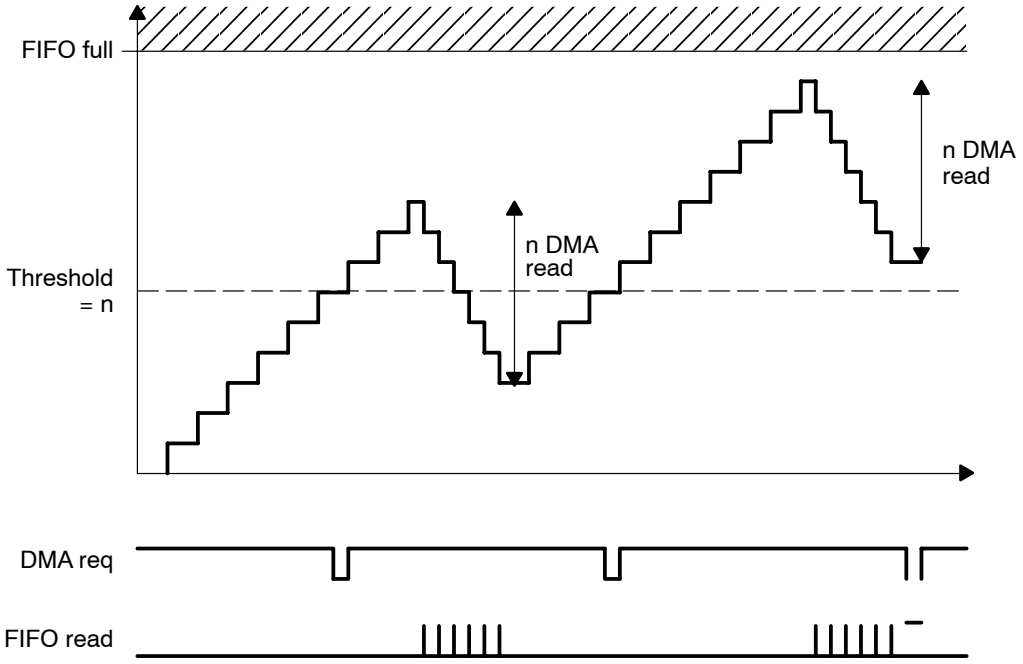
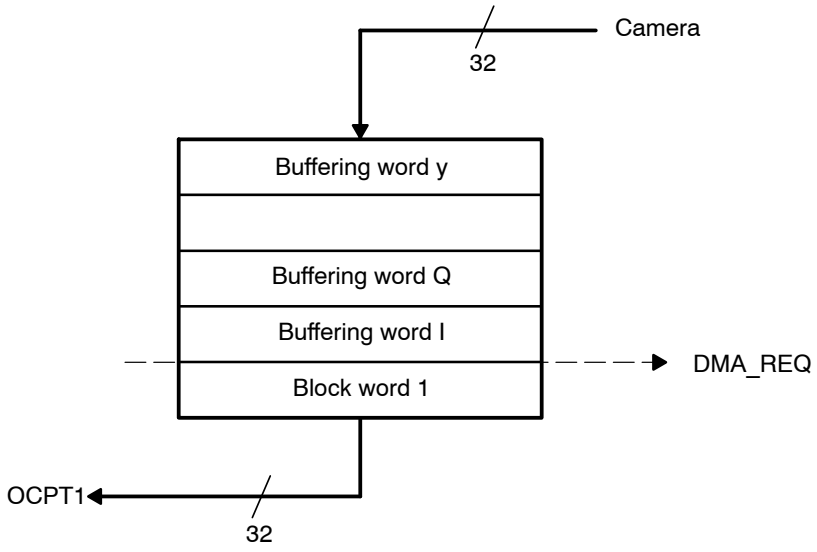


Figure 6. FIFO Buffer Parts



1.1.7 Clock Divider

The clock divider takes the ARMPER_CK clock source to generate the external clock CAM.EXCLK. The division factor is programmable in the clock control register through FOSCMOD (see Table 3).

It is assumed that the switch is made when CAM.EXCLK is disabled (glitch protection).

The clock divider also allows disabling the external clock by setting the CAMEXCLK_EN bit.

1.1.8 Interrupt Generator

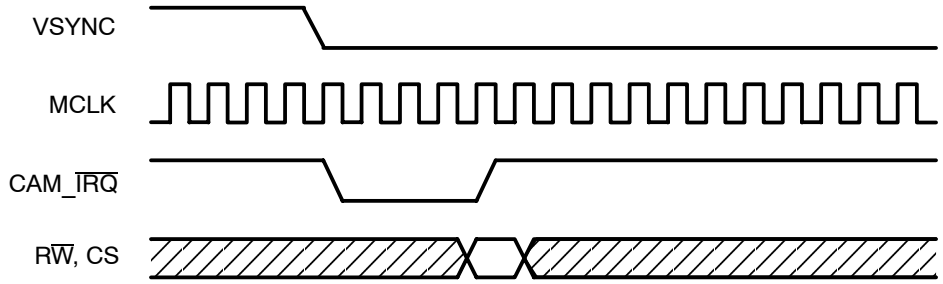
The interrupt generator handles six cases of interrupt:

- ❏ Data transfer interrupt. One IRQ is generated per word received.
- ❏ HSYNC rising edge (start of frame)
- ❏ HSYNC falling edge (end of frame). See Figure 7.
- ❏ VSYNC rising edge (start of image)
- ❏ VSYNC falling edge (end of image)
- ❏ FIFO overflow

Each case is registered by activating (high) one of the six interrupt register bits to indicate the origin of the interrupt. However, the interrupt mask register can disable the source of the interruption.

Only one line of interrupt is used to ask for a read of the interrupt register. When the read occurs, the register is automatically reset and the interrupt signal is released.

Figure 7. IRQ Generated on VSYNC Falling Edge



1.1.9 DMA Procedure

A typical procedure to perform the data transfer by DMA is as follows:

1) Rising edge of VSYNC sends an interrupt to ARM926EJS to alert the system DMA that a start of image has occurred. The system DMA is programmed to move one complete image of data, and give an interrupt when complete.

2) High level of HSYNC and proper clock edge start the first data transfer from the camera to the camera interface. After the first two pixels of data are received (8 bits x 4 transfers = 32 bits), a DMA request is made. The system DMA moves the 32-bit data to a predefined SDRAM location.

3) The camera, the device camera interface, and the system DMA continue the transfer of data. That is, 352/2 * 288 = 50688 transfers for a camera interface image format. After the full image is transferred, the DMA sends an interrupt to the ARM926EJS to signal that the end of frame occurred.

The camera interface and system DMA can be configured in many ways to move the data, but in this sequence the interrupt load on the ARM926EJS is minimal.

1.1.10 Camera IF Registers

The camera interface contains seven registers for communication between the TIPB and camera module. These registers mainly control clock generation, interrupt request, and status register (see section 1.1.11).

Table 1 shows the default configuration at reset.

Table 1. Default Configuration at Reset

Item	Function
ORDERCAMD	Not swapped
MASK	Interrupts on VSYNC and HSYNC disabled
FOSCMOD	Division rate for CAM.EXCLK = 1 (12 MHz)
POLCLK	Data latched on rising edge of CAM.LCLK
CAMEXCLK_EN	CAM.EXCLK disabled
MCLK_EN	Internal clock disabled
APLL_EN	APLL clock source disabled
THRESHOLD	Trigger level = 1 word

1.1.11 Camera Interface Registers

Table 2 lists the camera interface registers. Table 3 through Table 9 describe the bits of the individual registers.

Table 2. Camera Interface Registers

Base Address = 0xFFFB 6800				
Register	**Description**	**R/W**	**Size**	**Offset**
CTRLCLOCK	Clock control	R/W	32 bits	0x00
IT_STATUS	Interrupt source status	R	32 bits	0x04
MODE	Camera interface mode configuration	R/W	32 bits	0x08
STATUS	Status	R	32 bits	0x0C
CAMDATA	Image data	R	32 bits	0x10
GPIO	Camera interface GPIO (general-purpose input/output)	R/W	32 bits	0x14
PEAK_COUNTER	FIFO peak counter	R/W	32 bits	0x18

Table 3. Clock Control Register (CTRLCLOCK)

Base Address = 0xFFFB 6800, Offset = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:8	RESERVED	This field is reserved (unknown value after reset).	R/W	ND
7	LCLK_EN	0: Disables. 1: Enables incoming CAM.LCLK.	R/W	0x0
6	RESERVED	Reserved	R/W	0x0
5	MCLK_EN	0: Disables. 1: Enables internal clock of interface.	R/W	0x0
4	CAMEXCLK_EN	0: Disables. 1: Enables CAM.EXCLK.	R/W	0x0
3	POLCLK	Sets polarity of CAM.LCLK: 0: Data latched on rising edge. 1: Data latched on falling edge.	R/W	0x0
2:0	FOSCMOD	Sets the frequency of the CAM.EXCLK clock 000: ARM_PER_CK / 8 001: ARM_PER_CK / 3 010: ARM_PER_CK / 16 011: ARM_PER_CK / 2 100: ARM_PER_CK / 10 101: ARM_PER_CK / 4 110: ARM_PER_CK / 12 111: Inactive	R/W	0x00

The MCLK_EN bit gates the 12-MHz master clock of the camera interface to either disable the clock when switching between two clock domains or save power consumption when the camera module is not used. To clear PEAK_COUNTER, read all data in FIFO and then write PEAK_COUNTER with 0.

Table 4. Interrupt Source Status Register (IT_STATUS)

Base Address = 0xFFFB 6800, Offset = 0x04				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:6	RESERVED	Reserved bits.	R	0xX
5	DATA_TRANSFER	Data transfer status. Set to 1 when trigger is reached. Reset by reading IT_STATUS if no event in the meantime.	R	0x0
4	FIFO_FULL	Detect rising edge on FIFO full flag. Reset by reading IT_STATUS if no event in the meantime.	R	0x0
3	H_DOWN	Flag for horizontal synchronous falling edge occurred. Reset by reading IT_STATUS if no event in the meantime.	R	0x0
2	H_UP	Flag for horizontal synchronous rising edge occurred. Reset by reading IT_STATUS if no event in the meantime.	R	0x0
1	V_DOWN	Flag for vertical synchronous falling edge occurred. Reset by reading IT_STATUS if no event in the meantime.	R	0x0
0	V_UP	Flag for vertical synchronous rising edge occurred. Reset by reading IT_STATUS if no event in the meantime.	R	0x0

Table 5. Camera Interface Mode Configuration Register (MODE)

Base Address = 0xFFFB 6800, Offset = 0x08				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:19	RESERVED	Reserved bits	R/W	0xX
18	RAZ_FIFO	When 1: Clears data in the FIFO; reinitializes read and write pointers; clears FIFO full interrupt, FIFO peak counter; and resynchronizes.	R/W	0x0
17	EN_FIFO_FULL	0: Disables. 1: Enables interrupt on FIFO_FULL.	R/W	0x0
16	EN_NIRQ	0: Disables. 1: Enables data transfer interrupt (bypass DMA mode).	R/W	0x0

Table 5. Camera Interface Mode Configuration Register (MODE) (Continued)

Base Address = 0xFFFB 6800, Offset = 0x08				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:9	THRESHOLD	Programmable DMA request trigger value. DMA request is made when FIFO counter is equal to the threshold value. Currently, set this field to 1 in DMA mode.	R/W	0x0000001
8	DMA	Enables DMA mode when 1.	R/W	0x0
7	EN_H_DOWN	Enables interrupt on HSYNC falling edge. Active when 1.	R/W	0x0
6	EN_H_UP	Enables interrupt on HSYNC rising edge. Active when 1.	R/W	0x0
5	EN_V_DOWN	Enables interrupt on VSYNC falling edge. Active when 1.	R/W	0x0
4	EN_V_UP	Enables interrupt on VSYNC rising edge. Active when 1.	R/W	0x0
3	ORDERCAMD	Sets order of 2 consecutive bytes received from camera (YUV format). Not swapped when 0, swapped when 1.	R/W	0x0
2:1	IMGSIZE	Sets image size:. - CIF when 00 - QCIF when 01 - VGA when 10 - QVGA when 11 Currently, these bits have no effect on the operation of the camera interface.	R/W	0x00
0	CAMOSC	0: Set synchronous mode. 1: Set asynchronous mode. Currently, this has no effect on the camera interface.	R/W	0x0

Table 6. Status Register (STATUS)

Base Address = 0xFFFB 6800, Offset = 0x0C				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:2	RESERVED	Reserved bits	R	0xX
1	HSTATUS	CAM_HS status (edge detection)	R	0x0
0	VSTATUS	CAM_VS status (edge detection)	R	0x0

Table 7. Camera Interface GPIO Register (GPIO)

Base Address = 0xFFFB 6800, Offset = 0x10				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:1	RESERVED	Reserved bits	R/W	0xX
0	CAM_RST	Reset for camera module	R/W	0x0

Table 8. Image Data Register (CAMDATA)

Base Address = 0xFFFB 6800, Offset = 0x14				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:0	CAMDATA	Image data from FIFO	R	0x0

Table 9. FIFO Peak Counter Register (PEAK_COUNTER)

Base Address = 0xFFFB 6800, Offset = 0x18				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:7	RESERVED	Reserved	R/W	Unknown
6:0	PEAK_COUNTER	Maximum number of words written to FIFO during the transfer since the last clear to zero	R/W	0x0000000

1.2 Clock Switching Procedures

1.2.1 CAM.EXCLK Switch Protocol

The CAM.EXCLK switch protocol is required for any change of the CAM.EXCLK frequency value to first disable both the 12-MHz clock source and the APLL clock source in clock control registers:

1) Disable MCLK and APLL_CLK (MCLK_EN = 0, DPLL_EN = 0, FOSCMOD = FOSCMOD).

2) Change CAM.EXCLK value (FOSCMOD = new FOSCMOD).

3) Enable MCLK and APLL_CLK (MCLK_EN = 1, DPLL_EN = 1, FOSCMOD = FOSCMOD).

1.2.2 CAM.LCLK Switch Protocol

Bit 3 of the clock control register (POLCLK) sets the polarity of CAM.LCLK. CAM.LCLK must be disabled before selecting the rising or falling edge.

1) Disable CAM.LCLK (LCLK_EN = 0).

2) Set the new polarity (POLCLK = 1 or 0).

3) Enable CAM.LCLK (LCLK_EN = 1).

Index

OMAP5912 Multimedia Processor Display Interface Reference Guide

Literature Number: SPRU764B
October 2004

OMAP™
TEXAS INSTRUMENTS TECHNOLOGY

TEXAS INSTRUMENTS

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.

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
		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 document describes the display interface of the OMAP5912 multimedia processor.

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.

Related Documentation From Texas Instruments

Documentation that describes the OMAP5912 device, related peripherals, and other technical collateral, is available in the OMAP5912 Product Folder on TI's website: www.ti.com/omap5912.

Trademarks

OMAP and the OMAP symbol are trademarks of Texas Instruments.

Contents

Figures

Tables

Display Interface

This document describes the display interface of the OMAP5912 multimedia processor.

1 Overview

This document discusses the following components of the display interface:

- ❏ LCD module
- ❏ LCD data conversion module
- ❏ LED pulse generator
- ❏ Display interface

2 LCD Module

The device includes an LCD controller that interfaces with most industry-standard LCD displays through embedded or discrete timing controllers. The LCD controller operates only in single-panel mode (dual-panel mode is not supported). The module is designed to work with a separate RAM block to provide data to the FIFO at the front end of the LCD controller data path at a rate sufficient to support the chosen display mode and resolution.

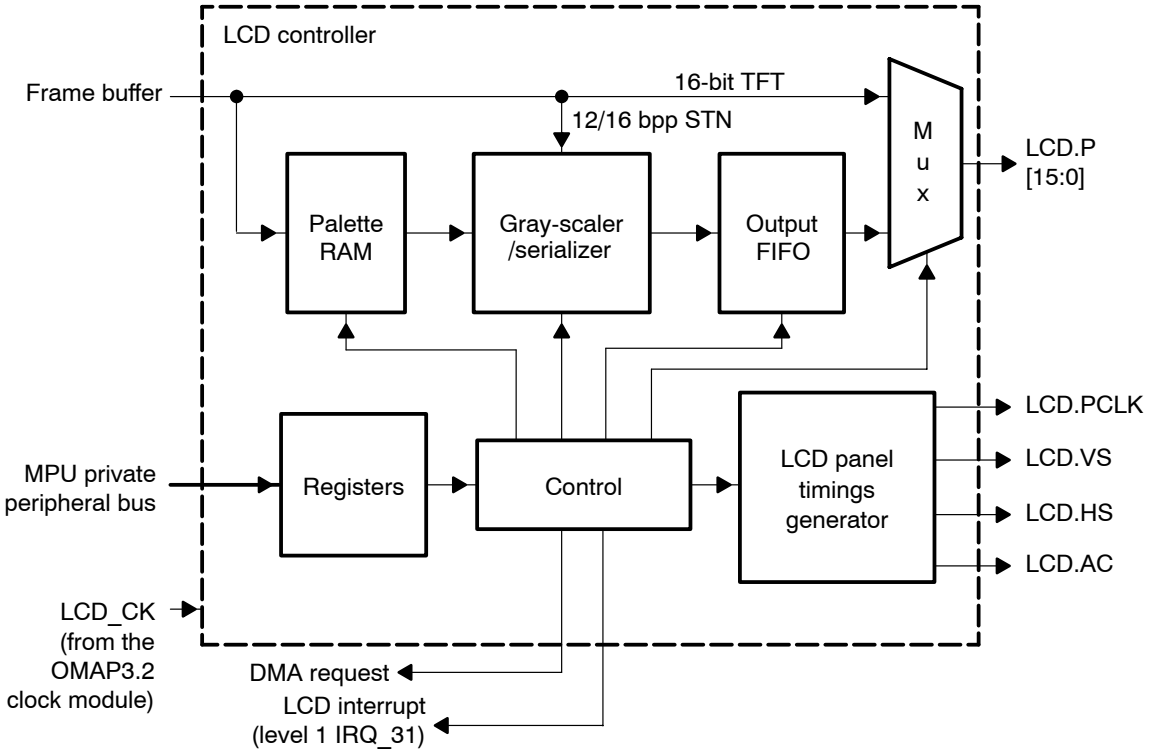
The panel size is programmable and can be any width (line length) from 16 to 1024 pixels, in 16-pixel increments. The number of lines is set by programming the total number of pixels in the LCD. The total frame size is programmable up to 1024×1024.

Frame sizes and frame rates supported in specific applications depend on the available memory bandwidth allowed by the application.

The screen is intended to be mapped to the frame buffer as one contiguous block, where each horizontal line of pixels is mapped to a set of consecutive bytes of words in the frame memory.

Figure 1 shows the LCD controller in detail.

Figure 1. LCD Controller



The LCD controller's principal features are:

- Dedicated 64-entry x 16-bit FIFO
- Dedicated LCD DMA channel for LCD display
- Programmable display including support for 1-, 2-, 4-, 8-, 12-, and 16-bit graphics modes
- Programmable display resolutions up to 1024 pixels by 1024 lines
- Support for passive monochrome (STN) displays
- Support for passive color (STN) displays
- Support for active color (TFT) displays
- Patented dithering algorithm, providing:
  - 15 grayscale levels for monochrome passive displays
  - 3375 colors for color passive displays

- 65536 colors for active color displays
- 256-entry x 12-bit palette
- Programmable pixel rate
- Pixel clock plus horizontal and vertical synchronization signals
- ac-bias drive signal
- Active display-enable signal

Frame buffer data can be formatted for 1-, 2-, 4-, 8-, 12-, or 16-bit pixel sizes. A 16-entry x 12-bit palette supports the 1-, 2-, and 4-bit pixel sizes, whereas a larger 256-entry x 12-bit palette supports the 8-bit pixel size. The 12-bit and 16-bit pixel sizes provide data that bypasses the palettes. The data is then processed according to the desired type of display.

For passive monochrome panels, the 4-bit value indexed from the least significant bits of the palette is passed to the patented dither logic, where the desired brightness is created using temporal dithering. The pixels are passed to the panel via a 4-wire interface, 4 pixels in parallel per pixel clock.

For passive color panels showing 8-bit color or less, an entry from the palette is transferred simultaneously into three parallel dither engines, one for each of the red, green, and blue colors. These values are converted by the three patented temporal dithering logic blocks to provide up to 256 colors out of a possible 3375 colors (15 x 15 x 15). The pixels are passed to the panel via an 8-wire interface, 2 2/3 pixels per clock.

For passive color panels showing 12- or 16-bit color, the data from the frame buffer is passed directly into the dither logic, bypassing the palette. The three parallel dither engines then provide up to 3375 colors. The 16-bit color mode uses only the most significant four bits of each color channel. The pixels are also passed to the panel via an 8-wire interface, 2 2/3 pixels per clock.

For active color panels showing 8-bit color or less, an entry from the palette is expanded from 12 bits to 16 bits and passed to the display, providing up to 256 colors out of a possible 4096 (16 x 16 x 16) colors. The pixels are passed to the panel via a 16-wire interface, 1 pixel per clock.

For active color panels showing 12-bit color, the data is also expanded from 12 bits to 16 bits to provide up to 4096 colors. The pixels are passed to the panel via a 16-wire interface, 1 pixel per clock.

For active color panels showing 16-bit color, the data is passed directly to the display (bypassing palette and dither logic), providing up to 65536 colors. The pixels are passed to the panel via a 16-wire interface, 1 pixel per clock.

The active color modes can also be used with an external DAC to drive a video monitor. The LCD line clock pin functions as a horizontal synchronization (HSYNC) signal, and the frame clock pin functions as a vertical synchronization (VSYNC) signal.

Table 1 shows the details of the LCD controller signals.

Table 1. Interface to LCD Panel Signal Descriptions

Name	Type	Destination	Description
LCD.P[15:0]	Out	LCD panel display	I/O pins used to transfer 4, 8, or 16 data values at a time to the LCD display. For monochrome displays, each signal represents a pixel; for passive color displays, groupings of three signals represent one pixel (red, green, and blue). LCD.P[3:0] is used for monochrome displays of 2, 4, and 8 BPP; LCD.P[7:0] is used for color STN displays and LCD.P[15:0] is used for active (TFT) mode.
LCD.PCLK	Out	LCD panel display	Pixel clock used by the LCD display to clock the pixel data into the line shift register. In passive mode, the pixel clock transitions only when valid data is available on the data lines. In active mode, the pixel clock transitions continuously, and the ac-bias pin is used as an output enable to signal when data is available on the LCD pins.
LCD.HS	Out	LCD panel display	Line clock used by the LCD display to signal the end of a line of pixels that transfers line data from the shift register to the screen and to increment the line pointer(s). Also used by TFT displays as the horizontal synchronization signal.
LCD.VS	Out	LCD panel display	Frame clock used by the LCD display to signal the start of a new frame of pixels. Also used by TFT displays as the vertical synchronization signal.
LCD.AC	Out	LCD panel display	ac-bias used to signal the LCD display to switch the polarity of the power supplies to the row and column axis of the screen to counteract DC offset. Used in TFT mode as the output enable to signal when data is latched from the data pins using the pixel clock.

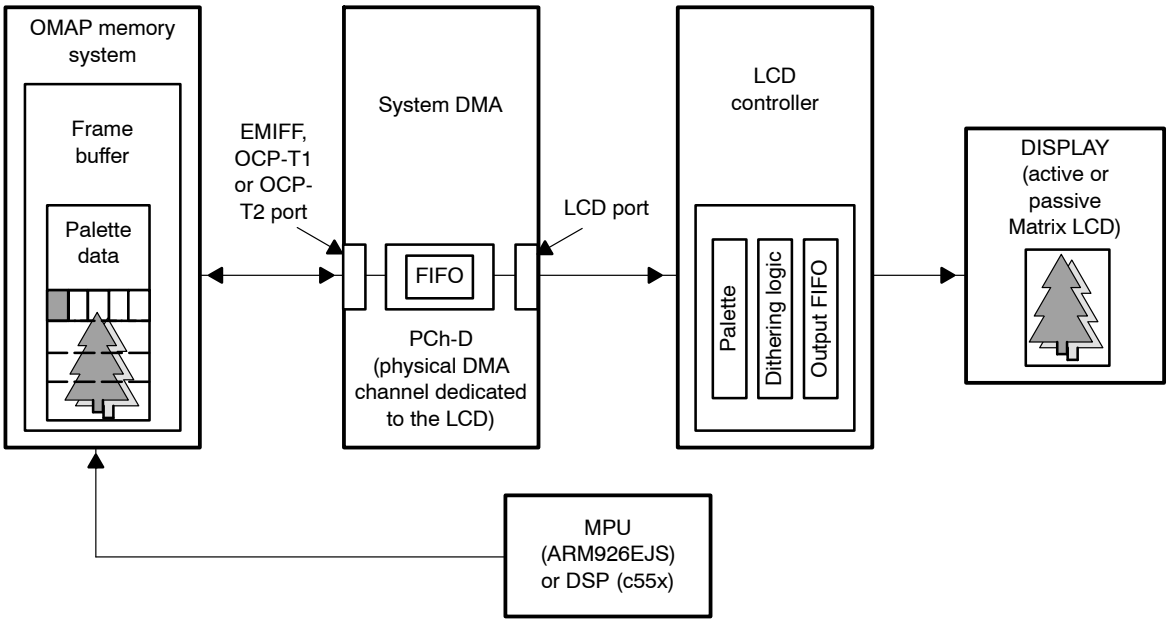
The pixel clock frequency is derived from the clock provided to the LCD controller (LCD_CK) from the clock management logic and is programmable from OMAP3.2 clock management logic (see the *Multimedia Processor OMAP 3.2 Subsystem Reference Guide (SPRU749)*). Each time new data is supplied to the LCD data pins, the pixel clock is toggled to latch the data into the LCD display serial shifter. The line clock toggles after all pixels in a line have been transmitted to the LCD driver and a programmable number of pixel clock wait states have elapsed both at the beginning and end of each line. In passive mode, the frame clock toggles during the first line of the screen, and the beginning and end of each frame is separated by a programmable number of line-clock wait states. Horizontal front porch (HFP) and horizontal back porch (HBP) must be programmed to zero in passive mode.

In active mode, the frame clock is asserted at the end of a frame after a programmable number of line-clock wait states occurs. In passive display mode, the pixel clock does not transition during wait state insertion or when the line clock is asserted. Finally, the ac-bias (LCD.AC) can be configured to transition each time a programmable number of line clocks occurs.

2.1 LCD Controller Environment

The LCD controller provides the necessary control signals to interface the memory directly to the external display through a dedicated DMA channel, as seen in Figure 2.

Figure 2. Data Flow from Microprocessor to Display



The MPU, or the DSP, stores the image to be displayed in a frame buffer. The frame buffer is used to supply enough encoded pixel values to fill the entire screen at least once.

The palette and the picture data are both in system memory. The palette loading mode (PLM) can be switched around so that only the picture data, only the palette data, or both are loaded at a given time (the palette is loaded only when it changes, then the PLM bit-field returns to picture-only mode).

The working copy of the palette resides in the LCD controller itself. See section 2.2, *LCD Controller Operation,* for descriptions of each block.

A specific system DMA channel dedicated to the LCD controller (LCh-D) is in charge of transferring data from the frame buffer to the LCD controller. Data is fetched, then transits into a 64*33-bit FIFO. The 33rd bit is added for frame-synchronization purposes. However, the internal LCD controller receives 16-bit data from the DMA FIFO with each request.

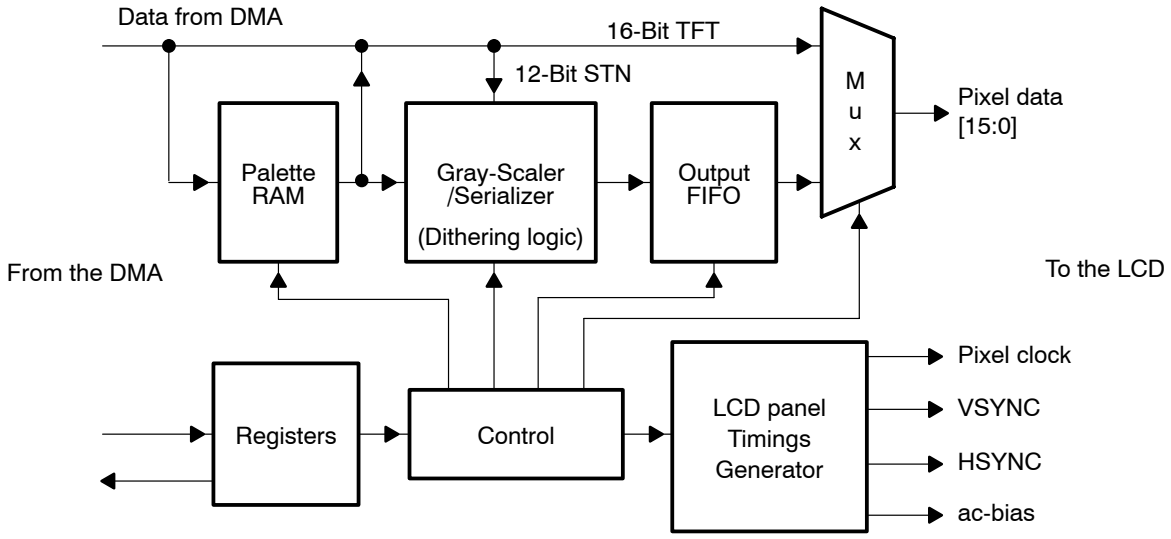
The following section describes the LCD controller operation in detail.

2.2 LCD Controller Operation

The LCD controller essentially consists of three blocks: a palette RAM, dithering logic, and an output FIFO, all associated with a control block operating through registers and a timing generator.

Figure 3 illustrates the LCD controller logic.

Figure 3. LCD Controller Operation Overview (Passive and Active Display Modes)



The LCD controller supports single-panel mode displays with programmable sizes. The display can be any width (line length) from 16 to 1024 pixels in 16-pixel increments (bit pixels per line (PPL) in the LCD timing 0 register). The number of lines is set by programming the lines per panel (LPP) field of the LCD timing 1 register (see section 2.3.3, *LCD Timing 1 Register*). The total video frame size is programmable up to 1024 x1024.

The screen is intended to be mapped to the frame buffer as one continuous block, where each line of pixels is mapped to a set of consecutive bytes or words in the frame buffer.

Two types of display technologies are supported:

- ❏ Passive, also known as super-twisted nematic, or STN
- ❏ Active, also known as thin film transistor, or TFT panels configured with the LCD TFT bit (LcdTFT) in the control register

See section 2.3.1, *LCD Control Register* for more information. Both monochrome and color modes are supported (LCD monochrome bit (LcdBW) in the control register).

In passive STN mode, a total of 3375 possible colors is available, allowing 16, 256, or 3375 colors to be displayed in each frame, depending on the color depth (number of bits per pixel: BPP). Fifteen grayscale levels are available for monochrome screens. See the *Dithering Logic* section, for information regarding the number of colors displayed versus BPP and screen technology.

In active TFT mode, whatever the color depth, 4096 colors can be displayed, except in the 16 BPP mode, where up to 64K colors are supported. See the *Dithering Logic* section for information regarding the number of colors displayed versus BPP and screen technology.

Note:

The active monochrome configuration is intentionally not considered in the remainder of this document.

Frame Buffer

The frame buffer is a memory area used to supply enough encoded pixel values to fill the entire screen one time. It is a part of the memory which is connected either to EMIFF, OCP-T1, or OCP-T2 port. A portion of the LCD controller frame buffer is used as a 32-byte buffer for 1-, 2-, 4- , 12-, and 16-BPP mode operation, or as a 512-byte buffer for 8 BPP mode of operation. The buffer is used to store the current display depth and a look-up palette, which is nonzero-filled for 8BPP and lower modes.

The 32-byte buffer is used to load the 16 entries of the palette for 1-, 2-, 4-, 12-, and 16-BPP encoding, or the 512-byte buffer is used to load the entire 256-entry palette for 8-BPP encoding.

Not all of the 16 entries of the palette are used in 1- and 2-BPP modes. However, all 16 palette entries are loaded regardless. The unused palette entries must be zero-filled.

The palette is not used in 12- or 16-BPP modes, but is still required. The 32 bytes of the palette are zero-filled, except for the first entry, where a 3-bit field provides the information on the number of bits-per-pixel.

Each time a new frame is fetched from the frame buffer, the LCD controller palette is first loaded with data contained within the palette buffer. Normally, this buffer is placed immediately preceding the frame buffer image data to simplify programming of the LCD DMA channel (see *Multimedia Processor Direct Memory Access (DMA) Support Reference Guide (SPRU755)*). However, using the dual-frame feature of the LCD DMA channel, the palette can be placed anywhere within the frame buffer. Separating the palette from the image data is necessary to enable the *pan-and-scan* or *virtual desktop* feature enabled by the LCD DMA channel, and can be useful in other situations. When the palette buffer is in data loading mode only, you do not have to load the palette each time. PLM = 10 in the control register; see subsection *Palette Loading (PLM)*.

Figure 4 and Figure 5 show the palette entry organization.

Figure 4. 16-Entry Palette/Buffer Format (1, 2, 4, 12, 16 BPP)

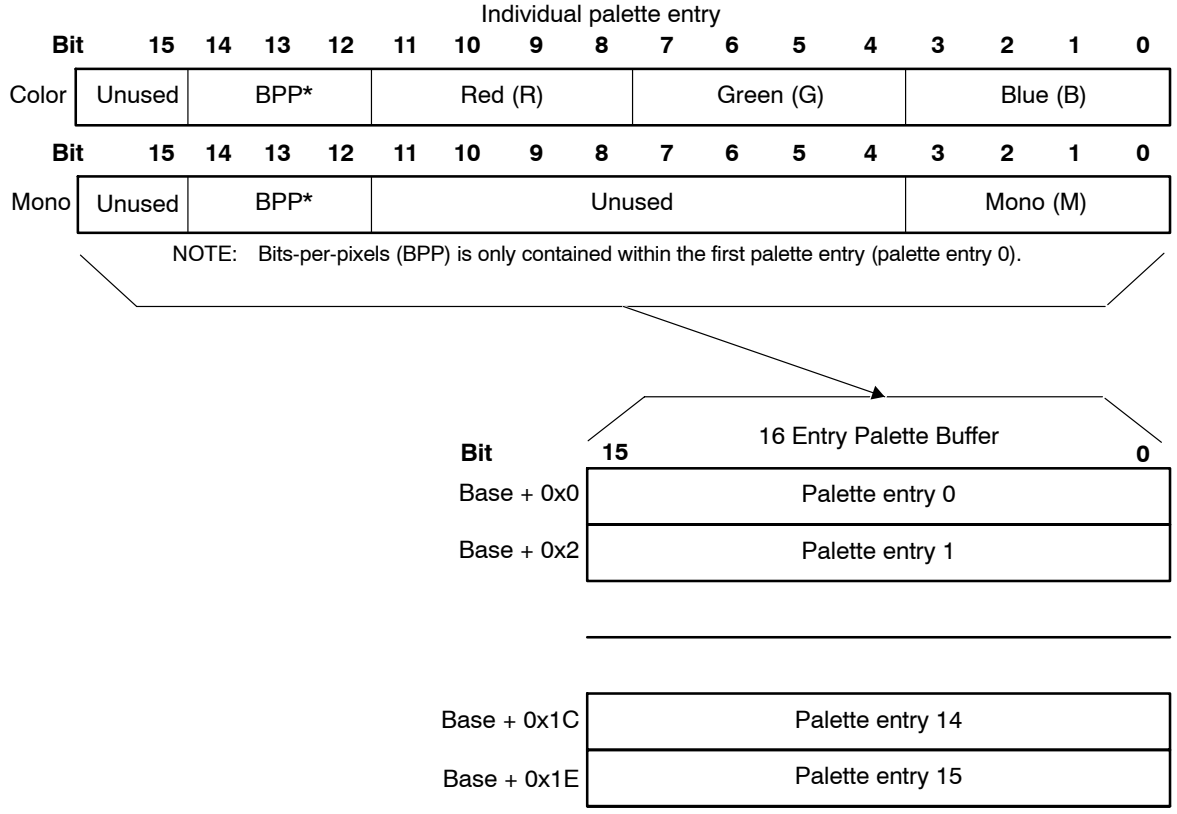
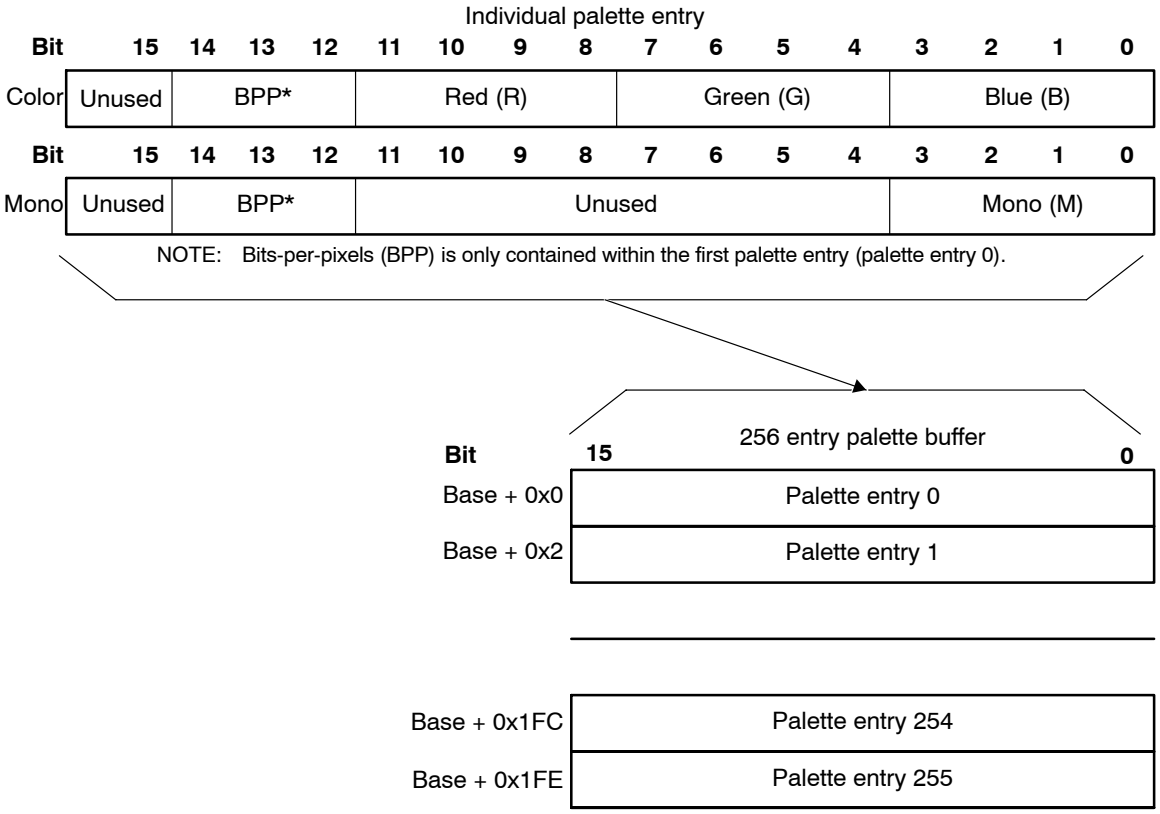


Figure 5. 256-Entry Palette/Buffer Format (8 BPP)



Bits 12, 13, and 14 of the first palette entry select the number of bits-per-pixel to be used in the following frame and thus, the number of palette RAM entries.

The bits-per-pixel (BPP) bit-field is decoded by the LCD to correctly unpack pixel data. It also configures the palette size to 16 or 256 entries.

Table 2 shows the BPP encoding in palette entry 0.

Table 2. Bits-Per-Pixel Encoding for Palette Entry 0 Buffer

Bit	Name	Description
14:12	BPP	Bits-per-pixel
		000: 1 BPP
		001: 2 BPP
		010: 4 BPP
		011: 8 BPP
		1xx: 12 BPP in passive mode (LcdTFT=0 and 565 STN =0),
		16 BPP in passive mode (LcdTFT=0 and 565 STN =1),
		16 BPP in active mode (LcdTFT=1).

Notes: 1) Eight 1-bit pixels, four 2-bit pixels, and two 4-bit pixels are packed into each byte, and 12-bit pixels are right justified on (16-bit) word boundaries (in the same format as palette entry).

2) For 565 STN, see the 16 BPP STN mode bit in the control register section.

The pixel data buffer contains one encoded pixel value for each pixel present on the display. Hence, the number of pixel data values depends on the size of the screen (i.e., 1024 x 768 = 786,432 encoded pixel values). Again, each pixel data value can be 1, 2, 4, 8, 12, or 16 bits wide.

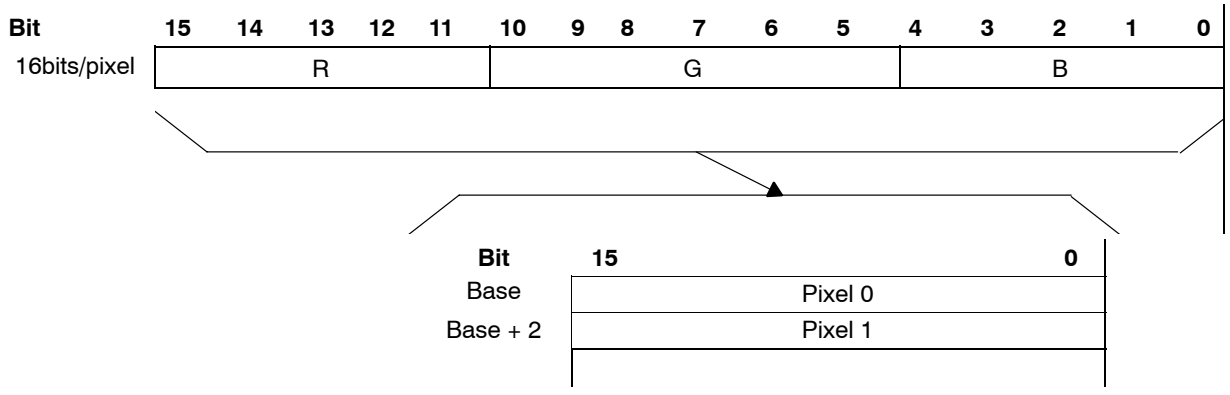
The following figures show the memory organization within the frame buffer for each pixel encoding size.

The LCD controller is fed with 16-bit data from the LCD DMA channel. In 16- and 12-BPP modes, this data is unaffected by LCDCB0 or LCDCB1 configuration bits.

16-BPP Mode (TFT)

Figure 6 shows one RGB representation in 16-BPP (TFT) mode.

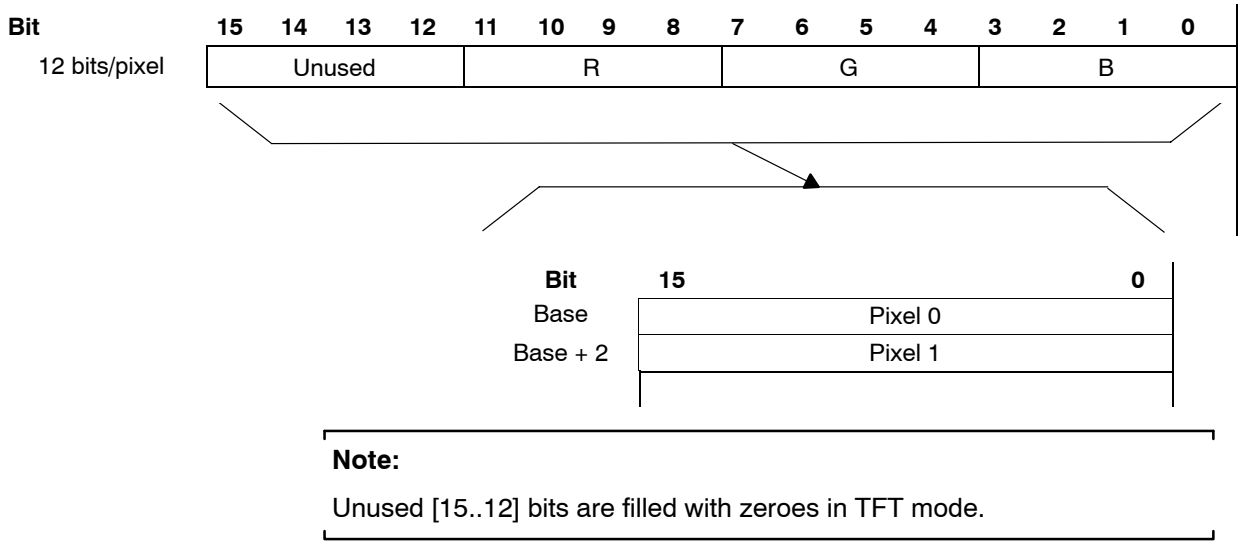
Figure 6. 16-BPP Data Memory Organization (TFT Mode Only)—Little Endian



12-BPP Mode

Figure 7 shows the RGB representation in 12-BPP mode.

Figure 7. 12-BPP Data Memory Organization—Little Endian



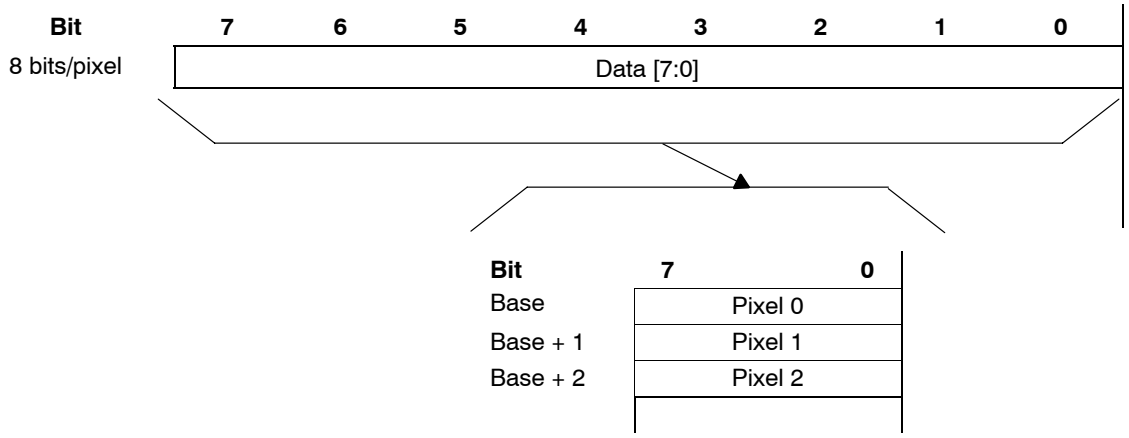
Note:

Unused [15..12] bits are filled with zeroes in TFT mode.

8-BPP Mode

Figure 8 shows the format of the data in 8-BPP mode.

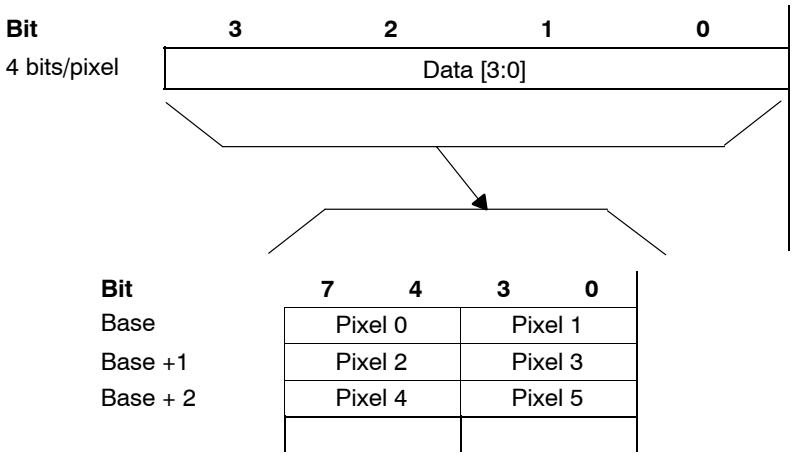
Figure 8. 8-BPP Data Memory Organization



4-BPP Mode

Figure 9 shows the format of the data in 4-BPP mode.

Figure 9. 4-BPP Data Memory Organization



2-BPP Mode

Figure 10. 2-BPP Data Memory Organization

Bit	7 6	5 4	3 2	1 0
Base	Pixel 0	Pixel 1	Pixel 2	Pixel 3
Base + 1	Pixel 4	Pixel 5	Pixel 6	Pixel 7
Base + 2	Pixel 8	Pixel 9	Pixel 10	Pixel 11

1-BPP Mode

Figure 11. 1-BPP Data Memory Organization

Bit	7	6	5	4	3	2	1	0
Base	P0	P1	P2	P3	P4	P5	P6	P7
Base + 1	P8	P9	P10	P11	P12	P13	P14	P15

The top and bottom addresses of the frame buffer (palette entries + pixels data) are programmed in the DMA controller.

The equations shown in Table 3 are used to calculate the total frame buffer size (in bytes) to program in the system DMA, based on varying pixel size encoding and screen sizes.

Table 3. Frame Buffer Size According to BPP

BPP	Frame Buffer Size
1	32 + (Lines * Columns) / 8
2	32 + (Lines * Columns) / 4
4	32 + (Lines * Columns) / 2
8	512 + (Lines * Columns)
12/16	32 + 2 * (Lines * Columns)

It is important to understand that BPP has two different meanings:

- In this section, BPP refers to how pixels are stored in memory. 1, 2, 4, 8, 12, or 16 are the different representations of the pixel within the frame buffer.
- BPP can also refer to how the panel views the pixels. This usage refers to which BPP the panels support (the actual interface, not the memory representation). The supported output panels are:
  - 1 BPP for monochrome panels, packed onto 8 (or 4) data lines
  - 3 BPP (1 bit each for red, green, and blue) for passive matrix technologies (output of dithering logic), packed onto 8 data lines
  - 12 BPP for STN (4, 4, 4) panels
  - 16 BPP for TFT (5, 6, 5) panels

Palette

The encoded pixel values stored in the frame buffer are used as pointers to index the 16-bit-wide palette. Palette entries are configured differently according to the mode used. See Figure 4 and Figure 5 for details.

The number of colors supported is given by $2^{\text{number of BPP}}$. These $2^{\text{number of BPP}}$ colors are to choose within the palette that limits them to $2^4 = 16$ grayscales in monochrome mode, and $2^{12} = 4096$ colors in color mode, where 4 and 12 are the effective palette width in each case (*effective* meaning dedicated to the monochrome/color scales coding).

16 grayscales and 4096 colors are numbers obtained after passing through the palette. A redundancy introduced at the dithering logic step reduces these numbers when displaying. As the dithering logic is bypassed in active mode, there is no redundancy and 4096 different colors are really available. For more detail, see the section *Dithering Logic,* Table 4, *Color/Grayscale Intensities and Modulation Rates,* and Table 5, *Number of Colors/Grayscales Available on Screen.*

❑ ***Passive Matrix Technology***

The palette is bypassed in 12 BPP. In palette plus data or in palette-only modes (PLM = 00 or 01), the first entry is read to acquire the number of BPP. All other entries or useless bits in the first entry are filled with zeroes. But in data-only mode (PLM = 10), the palette is not loaded with every frame.

Note:

Henceforth, the palette is said to be *bypassed* in 12- and 16-BPP modes. The 12-bit values are directly supplied to the dither logic when passive mode is enabled, while the 16-bit values are directly sent to the panel when active mode is enabled. The *bypass* term can be misleading in PLM = 00 or 01 configurations, considering that the palette must be read, at least for the first entry that contains the information of the color depth. The rest is zero-filled and not taken into consideration. In PLM = 10 (data-only mode), the palette is bypassed.

❑ ***Active Matrix Technology***

The palette is bypassed in 16 BPP, allowing 2^{16} = 65536 = 64K colors to be displayed.

Dithering Logic

❑ ***Passive Matrix Technology***

Once a palette entry is selected by the encoded pixel value from the look-up palette, its content is sent to the color/grayscale space/time-based dither generator. The monochrome data, as well as each component, is encoded on 4 bits: red, green, and blue (RGB) in color mode. See individual palette entry in Figure 4 and Figure 5. Each 4-bit value is processed by one dither block. Three separate dither blocks are used in the color mode. These 4-bit values are used to select one of the 16 intensity levels. The gray/color intensity is controlled by turning individual pixels on and off at varying periodic rates. More intense grays/colors are produced by making the average time that the pixel is off longer than the average time that it is on. The dither generator also uses the intensity of adjacent pixels in its calculations to give the screen image a smooth appearance. The proprietary dither algorithm is optimized to provide a range of intensity values that match the eye's visual perception of color/gray gradations.

Table 4 summarizes the duty cycle and resultant intensity level for all 16 color/grayscale levels.

Table 4. Color/Grayscale Intensities and Modulation Rates

Dither Value (4-Bit Value from Palette)	Intensity (0% is White)	Modulation Rate (Ratio of ON to ON+OFF Pixels)
0000	0.0%	0
0001	11.1%	1/9
0010	20.0%	1/5
0011	26.7%	4/15
0100	33.3%	3/9
0101	40.0%	2/5
0110	44.4%	4/9
0111	50.0%	1/2
1000	55.6%	5/9
1001	60.0%	3/5
1010	66.6%	6/9
1011	73.3%	11/15
1100	80.0%	4/5
1101	88.9%	8/9
1110	100.0%	1
1111	100.0%	1

Two of the 16 dither values (shaded in the table) are identical (most intense), which leads to redundancy in the colors.

This redundancy limits the choice to 15 (instead of 16) grayscales and 3375 (instead of 4096) colors. Note that $3375 = 15^3$ which means the equivalent of 15 color scales for each component (R, G, and B).

- ***Active Matrix Technology***

  The dithering logic is always bypassed in active displays. Hence, there is no redundancy introduced at this step, still allowing the choice of the $2^{\text{number of BPP}}$ within the whole 4096 colors.

  Remember that monochrome mode is deliberately not considered in active mode.

Table 5 lists the number of colors/grayscales available on the screen according to both the display technology and the color depth.

Table 5. Number of Colors/Shades of Gray Available on Screen

	Passive Mode (LcdTFT = 0)		Active Mode (LcdTFT = 1)
Number of BPP	**Monochrome (LcdBW = 1)**	**Color (LcdBW = 0)**	**Color Only (LcdBW = 0)**
1	2 palette entries to select within 15 grayscales	2 palette entries to select within 3375 possible colors	2 palette entries to select within 4096 possible colors
2	4 palette entries to select within 15 grayscales	4 palette entries to select within 3375 possible colors	4 palette entries to select within 4096 possible colors
4	16 palette entries to select within 15 grayscales	16 palette entries to select within 3375 possible colors	16 palette entries to select within 4096 possible colors
8	Not relevant since it would consist of 256 palette entries to select within 15 grayscales, but exists anyway	256 palette entries to select 3375 possible colors	256 palette entries to select within 4096 possible colors
12	X	3375 possible colors	4096 possible colors
16	X	3375 possible colors (565 STN = 1)	Up to 65536 possible colors

The Output FIFO

- ***Passive Matrix Technology***

  The LCD 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 size of the shifter is controlled by programming the color/monochrome select bit (LcdBW) in the control register. 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. Once the correct number of pixels has been placed within the shifter, the value is transferred to the top of the output FIFO. The value is then transferred down until it reaches the last empty location within the FIFO. As values reach the bottom of the FIFO, they are driven out one by one onto the LCD data pins on the edge selected by the invert pixel clock (IPC) bit (see subsection *Invert Pixel Clock (IPC)*).

- ***Active Matrix Technology***

  The output FIFO is bypassed in TFT mode.

LCD Inputs

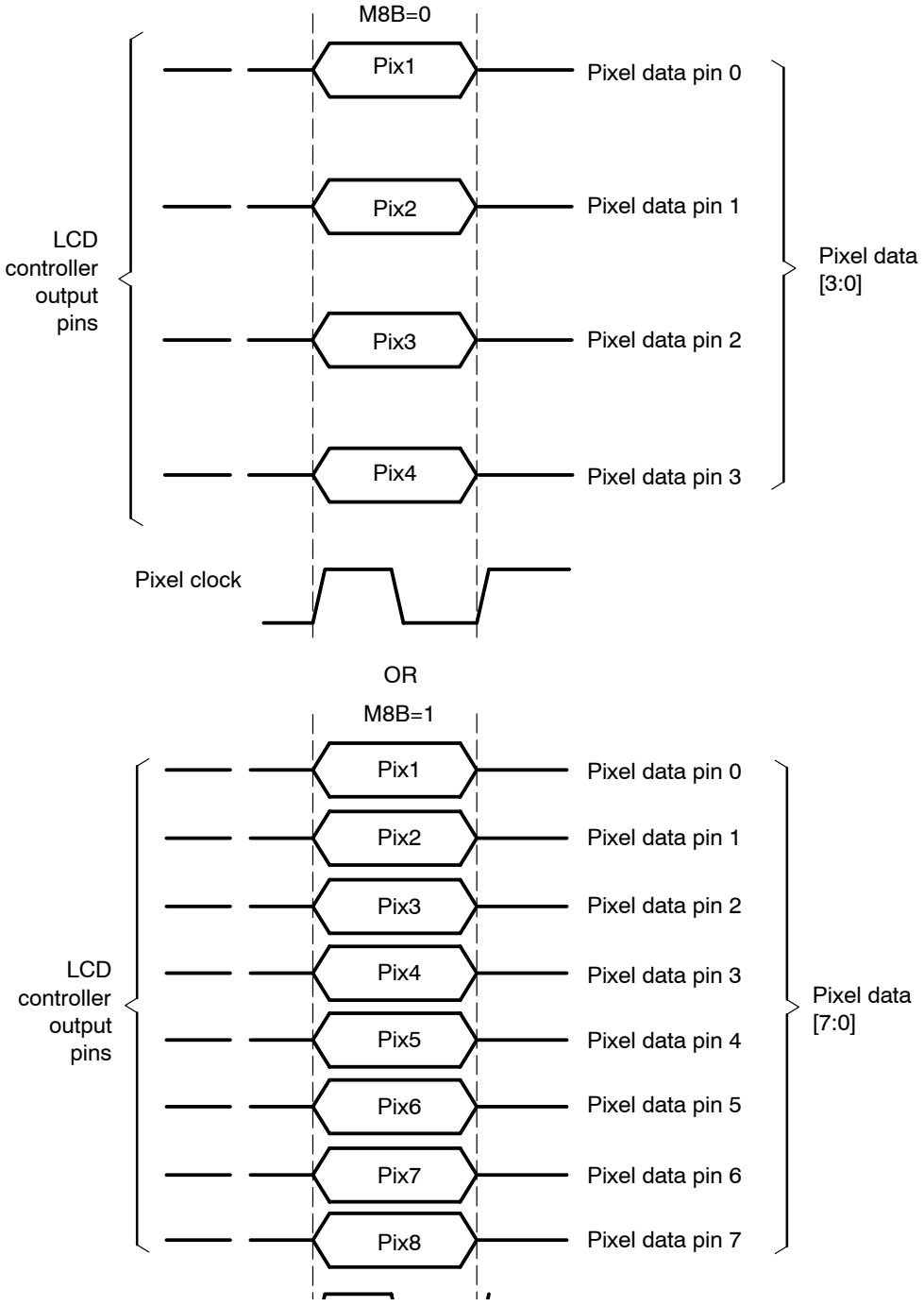
Depending on the type of panel used, the LCD controller is programmed to use either 4-, 8-, 12-, or 16-pixel data output pins. See Table 9, *LCD Controller Data Pin Utilization for Mono/Color Passive/Active Panels.*

- ***Passive Matrix Technology—Monochrome Mode***

  Monochrome displays use 4- or 8-bit data lines, according to the mono 8-bit mode (see subsection *Mono 8 Bit Mode (M8B*). Each line represents one pixel (ON or OFF), which means that at each pixel clock, 4 or 8 pixels, respectively, are sent to the screen.

  Figure 12 shows the passive monochrome mode (IPC = 0, see IPC bit in section 2.3.4, *LCD Timing 2 Register*).

Figure 12. Passive Monochrome Mode



❑ ***Passive Matrix Technology– Color Mode***

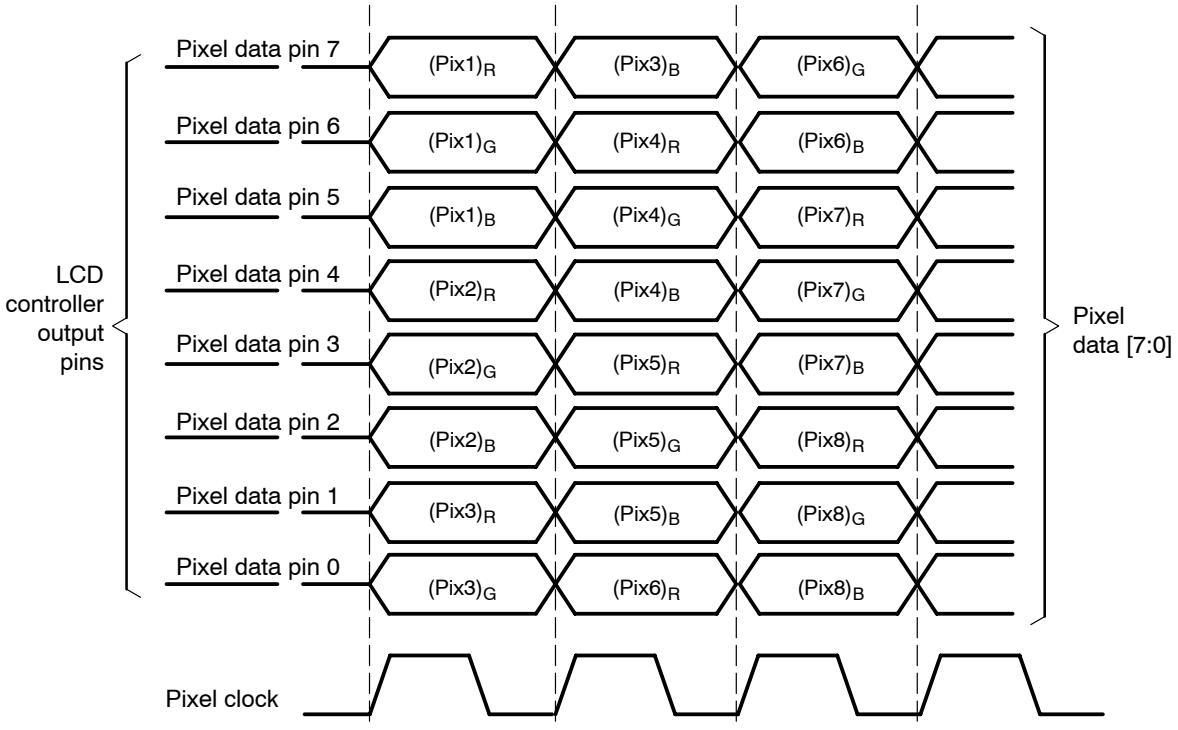
Color passive displays use eight data input lines. Each line represents one color component (red, green, or blue). 2 2/3 pixels are sent to the screen at each pixel clock.

Note:

8 data lines, 1 line per color component, 3 color components per pixel lead to 2 2/3 pixels on 8 data lines.

Figure 13 shows the passive color mode (IPC = 0, see IPC bit in section 2.3.4, *LCD Timing 2 Register*).

Figure 13. Passive Color Mode



After the pixel clock toggles three times, the situation returns to its initial state. At the fourth clock cycle, the figure becomes identical to itself, i.e., the number of pixels displayed is an integer.

- ***Active Matrix Technology***

  In TFT displays, the dithering logic and the output FIFO are always bypassed. This means that data output from the palette is sent directly to the display. In 16-BPP mode, even the palette is bypassed so that data passes directly from the memory to the panel without being processed.

  Consequently, at each pixel clock, only one pixel is sourced to the display.

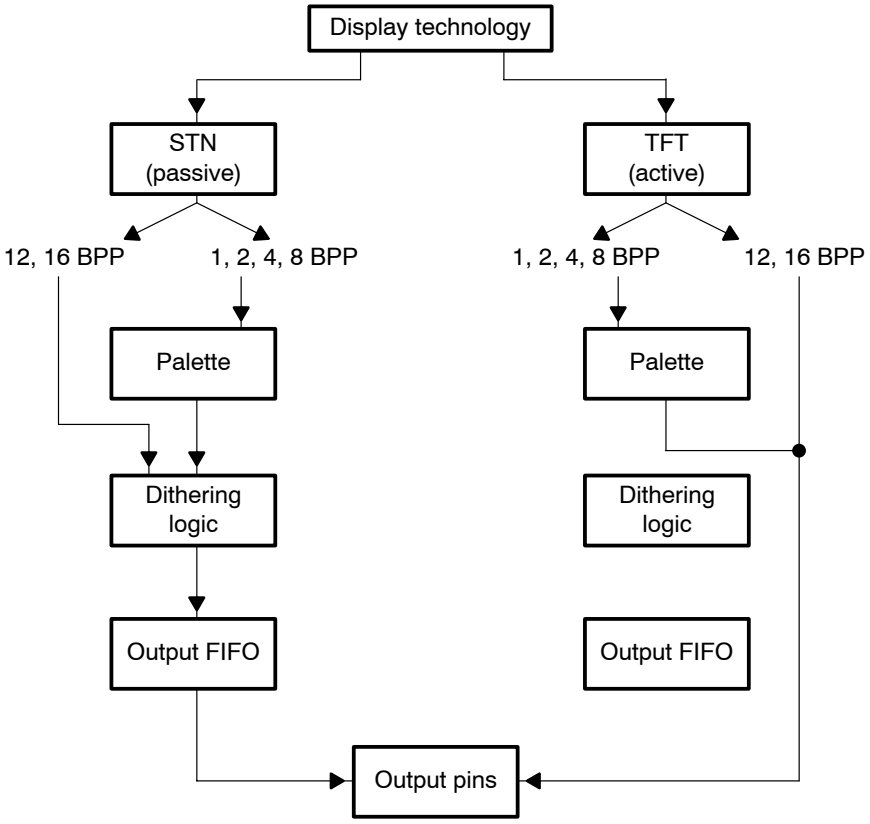
Table 6 is a summary of the number of pixels displayed on the screen at each pixel clock for the different technologies.

Table 6. Number of Pixels Displayed per Pixel Clock

Number of Pixels	Display
1	TFT
2 2/3	STN color
4	Mono 4 bit
8	Mono 8 bit

Figure 14 shows the different data paths depending on the screen technology and the color depth.

Figure 14. LCD Controller Data Paths



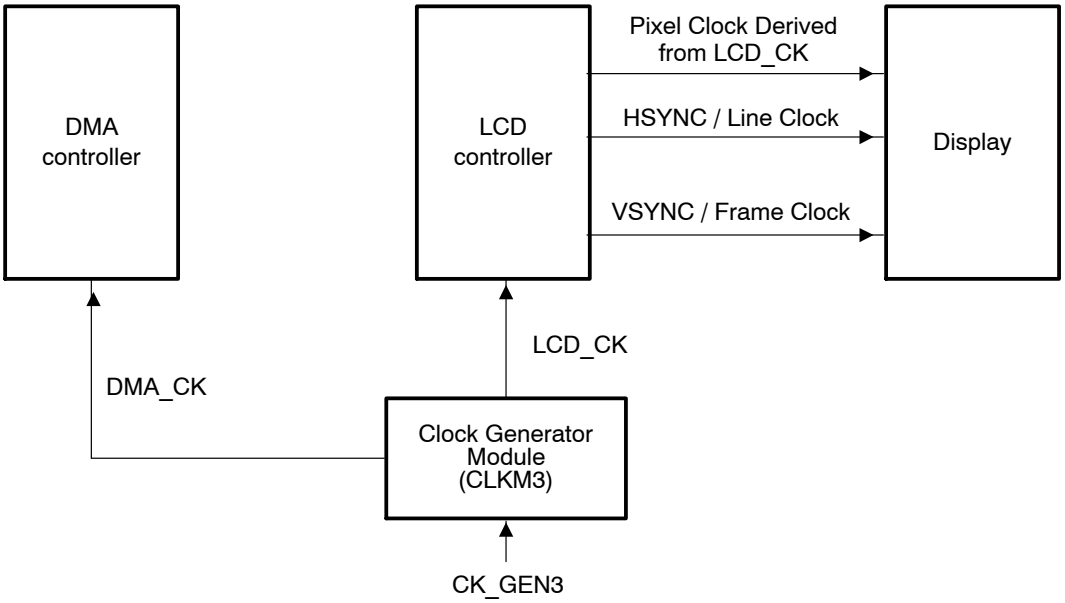
2.2.1 Control Blocks

The previous section explained the data flow from the memory to the LCD panel and the different modules they pass through. The whole process is governed by the registers and timing described in this section.

Timing

This section details the various clocks and signals. Figure 15 shows input and output LCD controller clocks.

Figure 15. Input and Output Clocks



Pixel Clock

The pixel clock frequency is derived from the LCD clock (LCD_CK), which belongs to the traffic controller (TC) domain. It is the output of the on-chip PLL: LCD_CK = (CK_GEN3/1, 2, 4, or 8). See *Multimedia Processor OMAP 3.2 Subsystem Reference Guide (SPRU749)* and *Multimedia Processor Clocks Reference Guide (SPRU751).*

You can program the pixel clock from LCD_CK / 2 to LCD_CK / 255. The divider is called PCD (see subsection *Pixel Clock Divider*).

In any case, the LCD_CK frequency must always be lower than or equal to the TC_CK frequency.

The pixel clock is used by the LCD display to clock the pixel data into the line shift register.

- **Passive Matrix Technology**

  In passive mode, the pixel clock only transitions when valid data is available on the data lines.

  It does not transition during wait state insertion or when the line clock is asserted.

❑ **Active Matrix Technology**

In active mode, the pixel clock transitions continuously as long as the LCD is enabled, depending on the pxl_gated bit in the LCD Control register (see section 2.3.1, *LCD Control Register*). Setting the pxl_gated bit to 1 allows the pixel clock to toggle only when there is valid data to display.

Line Clock (HSYNC)

The line clock toggles after all pixels in a line have been transmitted to the LCD driver and a programmable number of pixel clock wait states has elapsed both at the beginning and end of each line.

For more information, see the section *Horizontal Synchronization Pulse Width*.

❑ **Active Matrix Technology**

The line clock is also used by TFT displays as the horizontal synchronization signal (HSYNC).

The timings of the line clock pins are programmable to support:

❑ Delay insertion both at the beginning and end of each line. See subsection *Horizontal Front Porch (HFP) Bits* and subsection *Horizontal Back Porch (HBP)Bit*.

❑ Line clock polarity. See subsection *Invert HSYNC (IHS) Bit*.

❑ Line clock pulse width, driven on rising or falling edge of pixel clock. See subsection *Horizontal Synchronization Pulse Width (HSW)*, subsection *HSYNC/VSYNC Rise or Fall Programmable (RF)*, and subsection *HSYNC/VSYNC ON or OFF (ON_OFF) Bits*.

Frame Clock (VSYNC)

The frame clock toggles after all lines in a frame have been transmitted to the LCD driver and a programmable number of line clock cycles has elapsed both at the beginning and end of each frame.

For more information, see the section *Vertical Synchronization Pulse Width*.

❑ **Passive Matrix Technology**

In passive mode, the frame clock toggles during the first line of the screen.

- **Active Matrix Technology**

The frame clock occurs between two frames. In active mode, it is asserted at the end of the previous one and after a programmable number of line clock wait states (VFP) has elapsed.

The frame clock is also used by TFT displays as the vertical synchronization signal (VSYNC).

The timings of the frame clock pins are programmable to support:

- Delay insertion both at the beginning and end of each frame. See subsection *Vertical Front Porch (VFP) Bits* and subsection *Vertical Back Porch (VBP) Bits*.
- Frame clock polarity. See subsection *Invert VSYNC (IVS) Bit.*
- Frame clock pulse width, driven on rising or falling edge of pixel clock. See subsection *Vertical Synchronization Pulse Width (VSW),* subsection *Horizontal Synchronization Pulse Width (HSW)*, subsection *HSYNC/VSYNC Rise or Fall Programmable (RF)*, and subsection *HSYNC/VSYNC ON or OFF(ON_OFF) Bits*.

ac-Bias

The ac-bias signal can be configured to transition each time a programmable number of line clocks occurs.

- **Passive Matrix Technology**

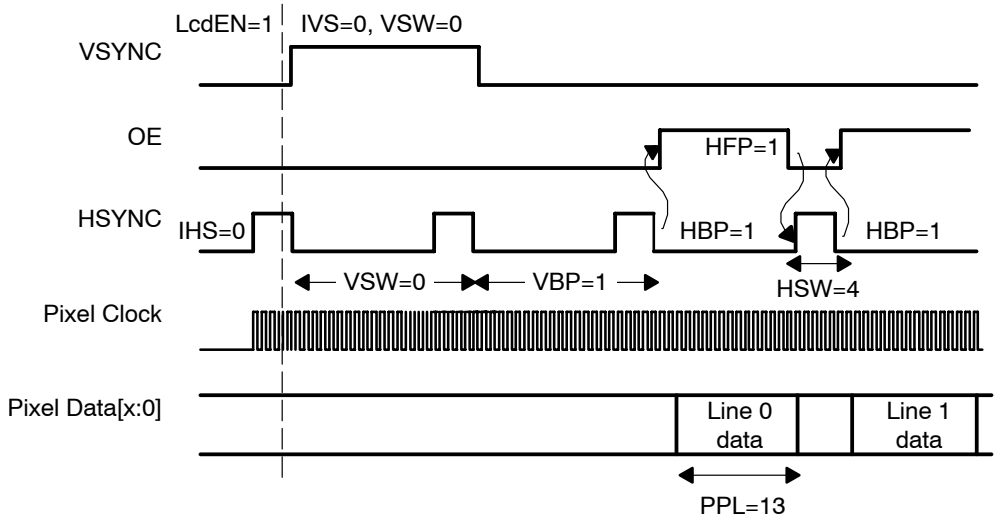
  To prevent a dc charge within the screen pixels, the power and ground supplies of the display are periodically switched. The LCD controller signals the display to switch the polarity by toggling the ac-bias pin.

- **Active Matrix Technology**

  Used in TFT mode, it acts as an output enable to signal when data must be latched from the data pins using the pixel clock.

  Figure 16 shows the different signals toggling in active mode.

Figure 16. Active (TFT) Mode Timing



NOTE: Ensure that HFP, HBP, PPL, and VSW values are programmed to the required value minus 1.

See the register sections, especially sections 2.3.2, *LCD Timing 0 Register*, and 2.3.3*, LCD Timing 1 Register*, for more details on the notations.

2.2.2 Interrupts

Interrupt Sources

Several situations can generate an interrupt:

- ❏ Input and output FIFOs underrun errors
- ❏ Frame synchronization error
- ❏ When the last active frame has completed after the LCD is disabled (maskable)
- ❏ After a programmed number of transitions of the ac-bias pin (passive mode)
- ❏ When the display has reached the user-programmed line number (maskable)
- ❏ VSYNC interrupt after every end of frame (maskable)
- ❏ Palette loading (maskable)

Every hardware-detected event signals an interrupt request to the interrupt controller.

Each interrupt is signaled by a bit in the LCD controller status register. Each of the LCD status bits signals an interrupt request as long as the bit is set. Once the bit is cleared, the interrupt is cleared. Read/write bits are called status bits; read-only bits are called flags. Status bits are referred to as “sticky” (that is, once set by hardware, they must be cleared by software). Read-only flags are set and cleared by hardware; writes have no effect.

Some interrupts are also maskable. For masked bits, see section 2.3.1, *LCD Control Register.*

Note:

A synchronization interrupt occurs if the LCD display information settings that you programmed, such as pixels-per-line, lines per frame, color/monochrome mode, and bits-per-pixel, are not in accordance with the size of the video buffer size programmed in the DMA.

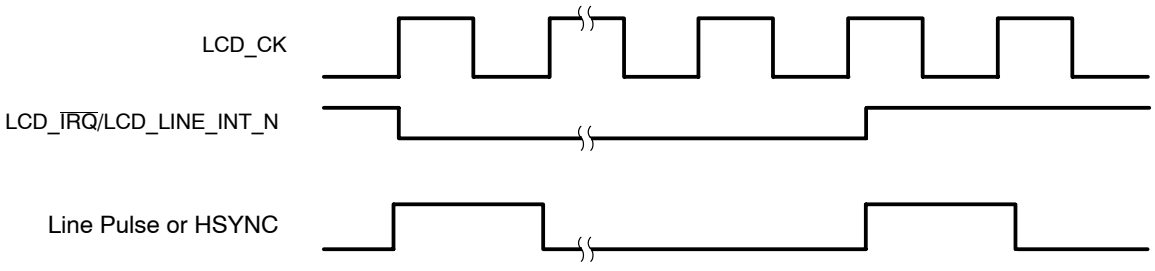
Tearing Effect

A synchronization mismatch between the frame buffer and the display refreshes can lead to images that appear to be torn on the screen.

To avoid this, a synchronization mechanism is needed between the LCD controller and the logical channel (LCh) that updates the frame buffer. For this purpose, a line comparator is implemented in the LCD controller, which delivers an interrupt when the display reaches a predefined line number. This interrupt is a level signal and stays active during the programmed line of the display.

See subsection *Line Interrupt (line_int*). Figure 17 shows the line interrupt output signal behavior for line_int_clr_sel = 1 (see subsection *Line Interrupt Clear Select Bit (line_int_clr_sel)*).

Figure 17. Line Interrupt Output Signal Transitions



Note that lcd_nirq and lcd_line_int_n have the same behavior and are active low signals.

The line interrupt must be connected as a DMA request line to the system DMA. This can then be used by any generic LCh (LCh 0-15) to synchronize a block transfer (BS).

Since all interrupts and DMA request inputs and outputs are brought out to OMAP top level boundary, the DMA request mapping is not predefined. See the chip top-level specification for the exact DMA request mapping.

2.2.3 LCD Subpanel Display Support

Principle

The subpanel display register supports the ability to display only the first or last X lines of the panel and send a fix contents for the others.

This functionality is used for power-saving. To display an image to a small portion of the screen (example: from line 0 to line 15), the data is read from system memory (frame buffer) via the DMA. For line 16 up to line N , where N is the number of lines per panel (LPP), data is read from an LCD register (DPD: default pixel data) instead of through the DMA. By reading into a register, there is no more access to the frame buffer, and power is saved. There is no need to go off-chip and do memory reads. In addition, the DMA can shut off its clocks during this DPD value filling.

For more information on the subpanel functionality, see section 2.3.6, *LCD Subpanel Display Support.*

2.3 Registers

The LCD controller contains eight registers:

- ❏ Four control registers
- ❏ Two status registers (including one display status register)
- ❏ One register related to the subpanel mode
- ❏ One specific register to define the line number where a line interrupt occurs

Table 7 shows the LCD controller registers and their physical addresses.

Table 7. LCD Controller Registers

Name	Offset	Description
LcdControl	0x 00	LCD control register
LcdTiming0	0x 04	LCD timing 0 register
LcdTiming1	0x 08	LCD timing 1 register
LcdTiming2	0x 0C	LCD timing 2 register
LcdStatus	0x 10	LCD status register
LcdSubpanel	0x 14	LCD subpanel display register
LcdLineInt	0x 18	Line interrupt register
LcdDisplayStatus	0x 1C	Display status register

2.3.1 LCD Control Register (LcdControl)

The LCD control register (LcdControl) contains bit-fields to enable/disable the LCD controller to define:

- ❏ The height and width of the screen being controlled
- ❏ Color or monochrome mode
- ❏ Passive or active display
- ❏ Polarity of the control lines
- ❏ Pulse width of the line and frame clocks
- ❏ The pixel clock and ac-bias frequency
- ❏ The number of delays to insert before/after each line and after each frame
- ❏ Interrupt mask bits

An additional control field exists to tune the DMA performance, based on the type of memory system in which the LCD controller is used. This field controls the placement of a minimum delay between each LCD palette request to ensure that enough bus bandwidth is given to other system accesses (see section *FIFO DMA Request Delay*). This field is only used for palette load.

Table 8 describes the LCD control register bits.

Figure 18. LCD Control Register (LcdControl)

Offset: 0h 00 **LCD Control: LCD Control Register** **Read/Write**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Reserved							565 STN	TFT Map	LCD CB1	PLM		FDD			
Reset	x	x	x	x	x	x	x	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FDD				PXL_GATED	LINE_INT_CLR_SEL	M8B	LCD CB0	Lcd TFT	LINE_INT_MASK	LINE_INT_NIRQ_MASk	LOAD MASK	DONE-MASK	VSYNC_MA	LCD BW	LCD EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

NOTE: The reserved bits' reset values are undefined, but reserved bits return 1s when read.

Table 8. LCD Control Register (LcdControl) Bit Descriptions

Bit	Name	Description
24	565 STN	12-BPP (565) mode 0: Off. 1: On (16-bit data in frame buffer, but only 12 bits are dithered and sent out).
23	TFT Map	TFT alternate signal mapping 0: Output pixel data for 1, 2, 4, 8, and 12 BPP modes is right aligned on pixel data [11:0]. 1: Output pixel data for 1, 2, 4, 8, and 12 BPP modes is converted to 5,6,5 format and uses pins [15:0].
22	LCDCB1	LCD control bit 1. Used in conjunction with LCD control bit 0 to control the mapping of pixel data from the frame buffer to the output bus. See Table 10 for the proper settings for this bit.
21:20	PLM	Palette loading mode 00: Palette and data loading 01: Palette loading 10: Data loading 11: Not connected
19:12	FDD	FIFO DMA request delay Encoded value (0-255) used to specify the number of LCD_CK cycles. The input FIFO DMA request must be disabled. The clock count starts after 16 words are read in the input FIFO. Programming FDD=00h disables this function.
11	PXL_GATED	Pixel gated (for TFT mode only) 0: Pixel clock always toggles. 1: Pixel clock only toggles when there is valid data to display.
10	LINE_INT_CLR_SEL	Line interrupt clear select bit 0: TIPB should write 0 to clear line interrupt status register. 1: Line interrupt status register is reset at the end of the line.

Table 8. LCD Control Register (LcdControl) Bit Descriptions (Continued)

Bit	Name	Description
9	M8B	Mono 8-bit mode 0: Pixel data [3:0] is used to output four pixel values to the panel at each pixel clock transition. 1: Pixel data [7:0] is used to output eight pixel values to the panel at each pixel clock transition. This bit is ignored in all other modes.
8	LCDCB0	LCD control bit 0. Used in conjunction with LCD control bit 1 to control the mapping of pixel data from the frame buffer to the output bus. See Table 10 for the proper settings for this bit.
7	LcdTFT	LCD TFT 0: Passive or STN display operation enabled; dither logic is enabled. 1: Active or TFT display operation enabled. External palette and DAC required. Dither logic and output FIFO bypassed. Pin timing changes to support continuous pixel clock, output enable, VSYNC, and HSYNC.
6	line_int_mask	Line interrupt mask (dedicated line) 0: Masks the dedicated line interrupt (line_int), which is connected to the lcd_line_int_n dedicated output line. 1: Mask not active.
5	line_int_nirq_mask	Line interrupt mask 0: The line_int_nirq interrupt is masked. 1: The line_int_nirq interrupt is unmasked.
4	LoadMask	Load mask 0: Masks the loaded palette interrupt, which is connected to the lcd_nirq shared output line. 1: Mask not active.
3	DoneMask	Done mask 0: Masks the frame done (Done) interrupt, which is connected to the lcd_nirq shared output line. 1: Mask not active.
2	VSYNC_mask	LCD VSYNC interrupt mask 0: Interrupt to the lcd_nirq is masked. 1: Interrupt to the lcd_nirq is unmasked.
1	LcdBW	LCD monochrome 0: Color operation enabled. 1: Monochrome operation enabled.
0	LcdEn	LCD controller enable 0: LCD controller disabled. 1: LCD controller enabled.

LCD Enable (LcdEn)

The LCD enable (LcdEn) bit is used to enable and disable all LCD controller operation: When LcdEn=0, the LCD controller is disabled. When LcdEn=1, the LCD controller is enabled.

Note:

You must program all other control bit-fields before setting LcdEn = 1, and must also disable the LCD controller when changing the state of any control bit within the LCD controller.

You can program the LCD control register (LcdControl) last, and configure all twenty-five bit fields at the same time via a word32 write to the register. If you clear LcdEn bit while the LCD controller is enabled, you can complete transmission of the current frame before being disabled. Completion of the current frame is signaled by the LCD controller to the DMA by setting the frame done (Done) bit within the LCD controller status register (see section 2.3.5, *LCD Controller Status Register*), which generates an interrupt request.

If the LCD controller is disabled, the signals on pixel data [15:0] pins are set to 0 and the pixel clock, frame clock, line clock, and ac-bias signals are set to their inactive state. This can be 0 or 1, depending on the inversions programmed in the timing 2 register (see section 2.3.4, *LCD Timing 2 Register*).

LCD Monochrome (LcdBW)

The color/monochrome select (LcdBW) bit is used to determine whether the LCD controller operates in color or monochrome mode. When LcdBW = 0:

- Color mode is selected.
- Palette entries are 12 bits wide, providing up to 4096 colors in active (TFT) mode and up to 3375 colors in passive (STN) color mode (see *Palette*).
- All three dither blocks are used (in passive mode only: LcdTFT = 0), one for each color component (R, G, B).

When LcdBW=1:

- Monochrome mode is selected.
- Palette entries are 4 bits wide *effective* (15 levels of grayscale, see *Palette*).
- 4 or 8 data lines are enabled, according to the mono 8-bit mode (M8B).

Table 9 shows which set of LCD data pins (Pixel Data […]) is used for each mode of operation.

Table 9. LCD Controller Data Pin Utilization for Mono/Color Passive/Active Panels

Color/Mono BPP	Passive/Active Panel	Pins
Mono 1, 2, 4	Passive	Pixel data [3:0]
Mono 8	Passive	Pixel data [7:0]
Color 1,2,4,8,12, 16 (565 STN = 1)	Passive	Pixel data [7:0]
Color 1,2,4,8,12	Active	Pixel data [11:0] or pixel data [15:0] according to TFT map bit in LCD control register
Color 16	Active	Pixel data [15:0]

Note:

Unused pixel data bits always remain low.

LCD Vertical Synchronization Mask (VSYNC_mask)

The LCD VSYNC_mask masks the VSYNC interrupt in the status register going to the lcd_nirq when VSYNC_mask bit is 0. When it is 1, the VSYNC bit affects the lcd_nirq. (See subsection *VSYNC Interrupt*.)

LCD Done Mask (DoneMask)

The LCD done mask (DoneMask) bit masks the path between the frame done interrupt and lcd_nirq, see subsection *Frame Done (Read-Only)*.

When DoneMask = 0, the frame done interrupt is masked.

When DoneMask = 1, the frame done interrupt is not masked.

LCD Loading Mask (LoadMask)

The LCD loading mask (LoadMask) bit masks the path between the palette loading interrupt signal in the LCD status register, and lcd_nirq.

When LoadMask = 0, the loading interrupt is masked.

When LoadMask = 1, the loading interrupt is not masked.

Line Interrupt Mask (line_int_nirq_mask)

This line_int_nirq_mask bit masks the path going to the shared interrupt (lcd_nirq).

When line_int_nirq_mask = 0, the path between the line interrupt and lcd_nirq (shared line) is masked. When line_int_nirq_mask = 1, it is unmasked: any interrupt, among those who share the lcd_nirq output line, can occur, including the line interrupt when the display reaches the programmed line number.

Line Interrupt Mask (Dedicated Line) (line_int_mask)

This line_int_mask bit masks or unmasks the connection to the dedicated top-level entity signal lcd_line_int_n, which is set to 0 when the display reaches the user-programmed line number. (See subsection *Line Interrupt,* and Figure 33, *Line Interrupt Path*).

When line_int_mask = 0, the connection is masked.

When line_int_mask = 1, the connection is unmasked.

Note:

When this bit is disabled, the line_int still can be used as source to generate lcd_nirq.

LCD TFT (LcdTFT)

The LCD TFT (LcdTFT) bit selects whether the LCD controller operates in passive (STN) or active (TFT) display control mode.

When LcdTFT = 0, passive or STN mode is selected. LCD data flows from the frame buffer memory, via the LCD dedicated DMA channel, to the palette (the palette is bypassed for 12 and 16 BPP modes) to the dithering logic and the output FIFO before being output on the LCD data pins.

Figure 19 and Figure 20 describe the clocks and data pin behaviors in monochrome and color passive modes, respectively.

Figure 19. Monochrome Passive Mode Pixel Clock and Data Pin Timing

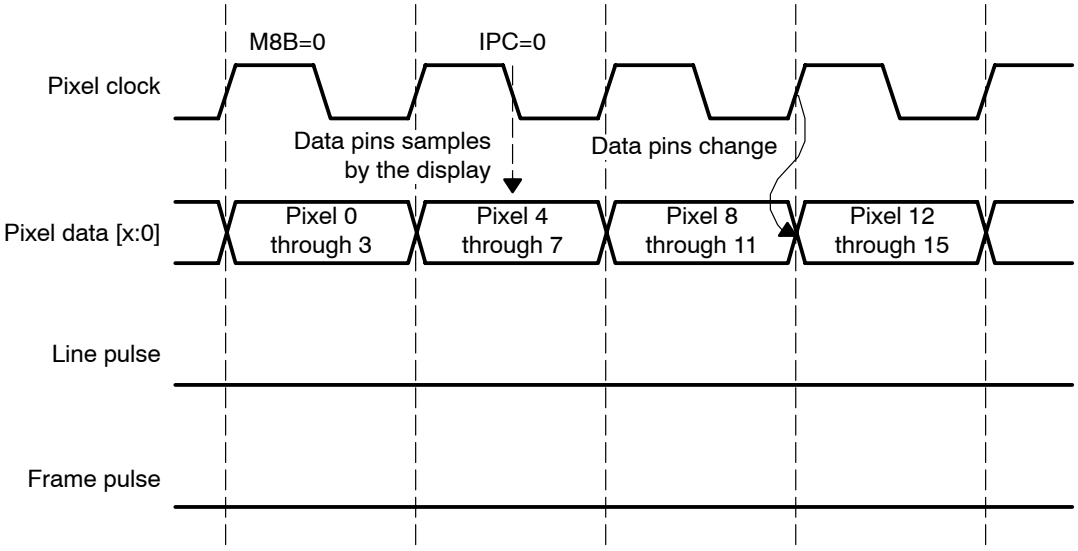
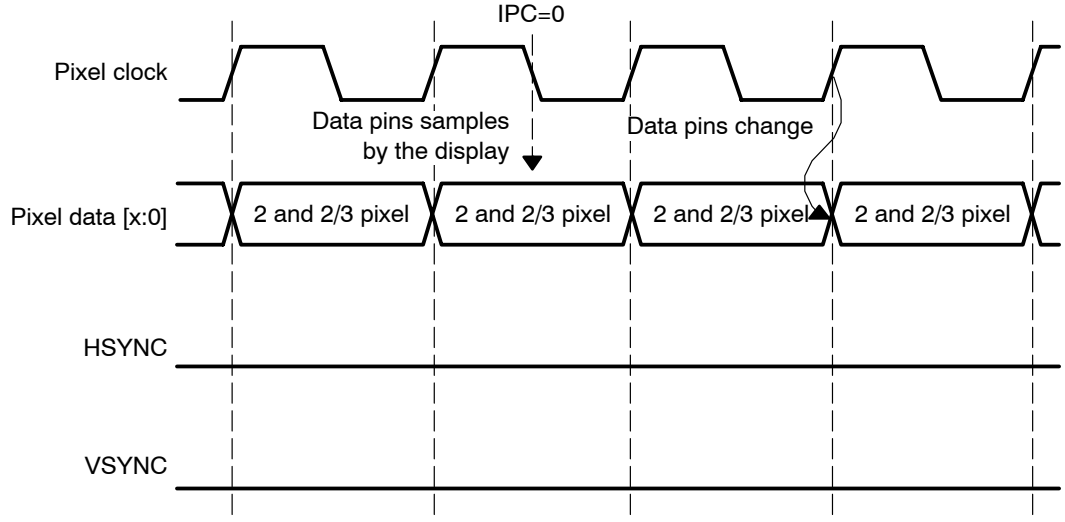


Figure 20. Color Passive Mode Pixel Clock and Data Pin Timing



When LcdTFT = 1, active or TFT mode is selected.

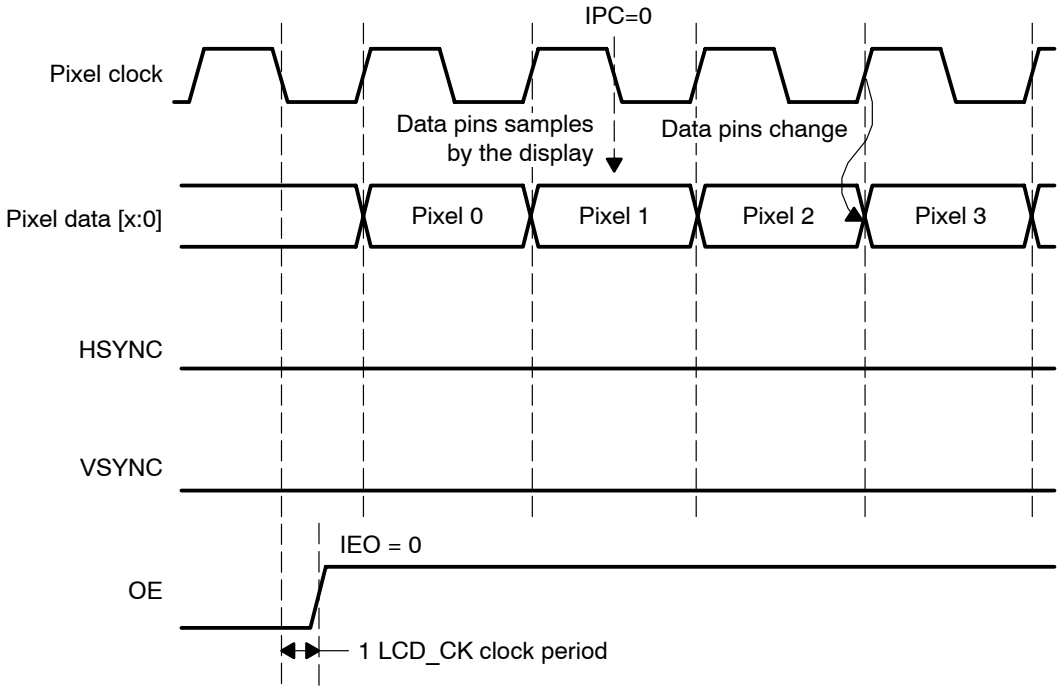
Video data is transferred via the DMA from memory to the input FIFO, then is unpacked and used to select an entry from the palette (for 1, 2, 4, and 8 bits-per pixel modes), just as in passive mode.

The value read from the palette, however, bypasses both the LCD dither logic and the output FIFO to be output on the LCD data pins in TFT mode. The pixel size within the frame buffer is increased to 16 bits when 12- or 16-bit pixel encoding mode is enabled (BPP=1XX).

Remember that the palette is bypassed in 12 BPP for passive mode and 16 BPP for active mode The palette is also bypassed in 12 BPP TFT, because it is derived from the 16 BPP mode. See Figure 14, *LCD Controller Data Paths*.

Figure 21 describes the clocks and data pin behaviors in active mode.

Figure 21. Active Mode Pixel Clock and Data Pin Timing



The size of the pixel encoding is increased in TFT mode because the LCD dither logic is bypassed, i.e., the dither logic only supports 4 bits to encode each color component R, G, B that limits the pixel encoding size in passive mode. Increasing the size of the pixel representation allows a total of 64K colors to be addressed using an off-chip palette in conjunction with the LCD controller.

LCD Control Bit 0 (LCDCB0)

Bit LCDCB0, together with LCDCB1, controls the mapping of graphics data on the output pins. Table 10 shows the settings required for each graphics mode.

Table 10. Control Bit 0 and Control Bit 1 Mapping by Display Types

Display Type	Mode	Control bit 0	Control Bit 1
Passive monochrome	1 BPP	0	1
	2 BPP	0	1
	4 BPP	0	1
	8 BPP	0	0
Passive color	2 BPP	0	1
	4 BPP	0	1
	8 BPP	0	0
	12 BPP	0	0
	16 BPP	0	0
TFT	2 BPP	0	1
	4 BPP	0	1
	8 BPP	0	0
	12 BPP	0	0
	16 BPP	0	0

Mono 8 Bit Mode (M8B)

The mono 8-bit mode (M8B) bit selects whether four or eight data lines are used to output pixel data to the LCD screen. When M8B = 0, pixel data [3:0] is used to output four pixel values to the LCD panel at each pixel clock transition. When M8B = 1, pixel data [7:0] is used to output eight pixel values to the LCD panel at each pixel clock transition.

Note:

M8B does not affect any of the color modes or TFT.

Line Interrupt Clear Select Bit (line_int_clr_sel)

The line_int_clr_sel bit selects between two methods that clear the bit in the status register. You can select either an automatic clear or a TIPB write. When line_int_clr_sel = 0, write 0 in the line_int status bit to clear the interrupt. When line_int_clr_sel = 1, the line interrupt bit in the status register is reset at the end of the programmed line.

Gated Pixel Clock (pxl_gated)

The pxl_gated bit selects between gated or not gated pixel clock when in TFT mode.

When pxl_gated = 0, the pixel clock always toggles.

When pxl_gated = 1, the pixel clock does not toggle when there is not valid data to display. This is a power saving option.

FIFO DMA Request Delay (FDD)

The 8-bit FIFO DMA request delay (FDD) field is used to select the minimum number of LCD_CK cycles to wait between the servicing of each DMA request issued by the LCD controller, sending an address to the input FIFO.

The goal is to ensure enough bandwidth to other system accesses. A delay of FDD cycles is inserted for every 16 words read from the input FIFO. This function is a concern only in 8 BPP mode, where the palette is 256 words.

When FDD = 00h, the FIFO DMA request delay function is disabled. This function is only used for palette loading.

Palette Loading (PLM)

The 2-bit palette loading field describes how the palette loading behaves when each new frame is loaded from memory.

- In 00 mode, the data in the frame buffer represents the palette data and the picture data. Both palette and picture data are loaded.
- 01 is the palette-only mode. The data in the frame buffer just represents a new palette to be loaded. This data is loaded and placed into the palette. Be sure to turn off the LCD after getting the loading interrupt, or the LCD behavior will be unpredictable.
- When in data loading mode (PLM = 10), the data in the frame buffer only represents the picture data (data-only). This data is then used as an index (in the palette) or sent directly out. This mode assumes the palette was previously loaded. There is no need to keep loading the palette if it is not changing. As a matter of fact, in data-only mode, the BPP is fixed and can not change on the fly since the palette is not loaded at every frame.
- 11 mode is reserved.

LCD Control Bit 1 (LCDCB1)

The LCD control bit 1 is used in conjunction with LCD control bit 0 to control the mapping of pixel data from the frame buffer to the output bus.

Refer to Table 10 for the appropriate settings for these bits.

TFT Alternate Signal Mapping (TFT Map)

This bit is relevant only if LcdTFT = 1.

This bit field controls how the TFT pixel data is output. Via this feature, 12-BPP data can be output to all 16-bit LCD pins (this also applies to 1-, 2-, 4-, and 8-BPP). This feature allows you to switch BPP modes on the fly, duplicating the 12-bit output data across the 16 data lines if they are already hardwired to the 16 data lines.

Figure 22 shows how the four red, four green, and four blue bits are mapped to all pixel data [15:0] output pins when this bit is set to 1.

Figure 22. TFT Alternate Signal Mapping Output

Pins	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data	R3	R2	R1	R0	R3	G3	G 2	G1	G0	G 3	G2	B3	B2	B1	B0	B3

When this bit is 0, the four red, four green, and four blue data are right aligned on pixel data [11:0]. The upper pixel data [15:12] are set to 0. There is no duplication.

16 BPP STN Mode (565 STN)

This bit is relevant only if LcdTFT = 0, but has no effect in 1-, 2-, 4- and 8-BPP modes.

If 565 STN = 0, the frame buffer organization is in 12 BPP mode. In this mode, each color component is encoded in 4 bits, as shown in Table 11.

Table 11. 12-Bit STN Data in Frame Buffer

Unused				**Red**				**Green**				**Blue**			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data ignored				R3	R2	R1	R0	G3	G2	G1	G0	B3	B2	B1	B0

If 565 STN = 1, the 16 BPP STN mode is selected. The only difference between this mode and the 12-BPP mode is how the pixel data is organized in the frame buffer and which bits are sent to the dither logic.

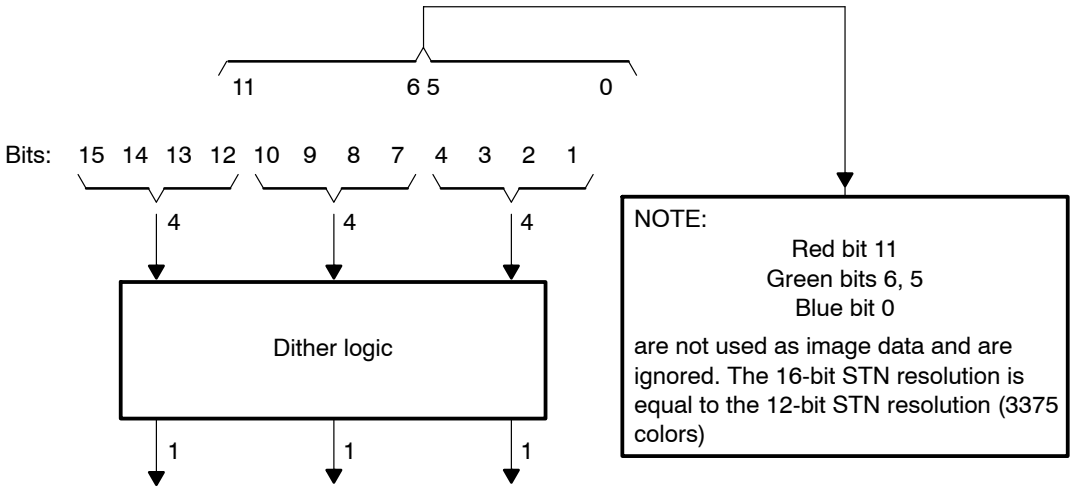
16 bit STN mode appears in the frame buffer memory as shown in Table 12.

Table 12. 16-Bit STN Data in Frame Buffer

Red					**Green**						**Blue**				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0

Nevertheless, 16-bit STN mode only sends 12 bits to the dithering logic as well as the 12-BPP STN mode. The LSB bit of the red component (bit 11), the two LSBs of green (bits 6 and 5), and the LSB of blue (0) are not sent to the dithering logic.

Figure 23. 16 BPP STN Mode



This 12-BPP (5-6-5) mode can be used if the operating system does not support 12 BPP in the frame buffer. Data is arranged in 16 BPP instead, but only 12 bits are dithered and sent to the display.

2.3.2 LCD Timing 0 Register (LcdTiming0)

LCD timing 0 register (LcdTiming0) contains four bit-fields that are used as modulus values for a collection of down counters. This register controls HSYNC-signal generation.

Figure 24. LCD Timing 0 Register (LcdTiming0)

Offset: 0x 04 **LcdTiming0: LCD Timing 0 Register** **Read/Write**

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

Table 13. LCD Timing 0 Register (LcdTiming0) Bit Descriptions

Bit	Name	Description
31:24	HBP	Horizontal back porch The encoded value (from 1−256) used to specify the number of pixel clock periods to add to the beginning of a line transmission before the first set of pixels is output to the display *(program to value required minus 1).* Note that pixel clock is held in its inactive state during the beginning of line wait period in passive display mode and is permitted to transition in active display mode.
23:16	HFP	Horizontal front porch The encoded value (from 1−256) used to specify the number of pixel clock periods to add to the end of a line transmission before line clock is asserted *(program to value required minus 1).* Note that pixel clock is held in its inactive state during the end of line wait period in passive display mode and is permitted to transition in active display mode.
15:10	HSW	Horizontal synchronization pulse width The encoded value (from 1−64) used to specify the number of pixel clock periods to pulse the line clock at the end of each line *(program to value required minus 1).* Note that pixel clock is held in its inactive state during the generation of the line clock in passive display mode and is permitted to transition in active display mode.
9:0	PPL	Pixels per line The encoded value (from 16−1024) used to specify the number of pixels contained within each line on the LCD display *(program to value required minus 1).*

Pixels-Per-Line (PPL)

The pixels-per-line (PPL) bit-field is used to specify the number of pixels in each line on the screen. It represents the screen width. PPL is a 10-bit value. Taking into account that the bottom 4 bits of this register are not used and always read 1, it is possible to support displays in which the number of pixels-per-line ranges between 16 and 1024. PPL is used to count the correct number of pixel clocks that must occur before the line clock can be pulsed.

Notes:

PPL must be programmed to the value required minus 1 (0x27F for a 640-pixel-per-line LCD panel).

PPL must be a multiple of 16 pixels.

Horizontal Synchronization Pulse Width (HSW)

The 6-bit horizontal synchronization pulse width (HSW) field is used to specify the pulse width of the line clock in passive mode, or horizontal synchronization pulse in active mode. The line clock (or HSYNC) is asserted each time a line or row of pixels is output to the display and a programmable number of pixel clock delays have elapsed. When line clock is asserted, the value in HSW is transferred to a 6-bit down counter that uses the programmed pixel clock frequency to decrement. When the counter reaches zero, the line clock is negated. HSW can be programmed to generate a line clock pulse width ranging from 1–64 pixel clock periods (program to value required minus 1).

Note:

The pixel clock does not transition during the line clock pulse in passive display mode, but it transitions in active display mode.

Horizontal Front Porch (HFP)

The 8-bit horizontal front porch (HFP) field is used to specify the number of dummy pixel clocks to insert at the end of each line or row of pixels before pulsing the line clock pin. Once a complete line of pixels is transmitted to the LCD driver, the value in HFP is used to count the number of pixel clocks to wait before pulsing the line clock. HFP generates a wait period ranging from 1–256 pixel clock cycles (program to value required minus 1).

Note:

The pixel clock does not transition during these dummy pixel clock cycles in passive display mode, but it transitions continuously in active display mode.

Horizontal Back Porch (HBP)

The 8-bit horizontal back porch (HBP) field is used to specify the number of dummy pixel clocks to insert at the beginning of each line or row of pixels. After the line clock for the previous line has been negated, the value in HBP is used to count the number of pixel clocks to wait before starting to output the first set of pixels in the next line. HBP generates a wait period ranging from 1–256 pixel clock cycles (program to value required minus 1).

Note:

The pixel clock does not transition during these dummy pixel clock cycles in passive display mode, but it transitions continuously in active display mode.

2.3.3 LCD Timing 1 Register (LcdTiming1)

LCD timing 1 register (LcdTiming1) contains four bit-fields that are used as modulus values for a collection of down counters. This register controls VSYNC-signal generation.

Figure 25. LCD Timing 1 Register (LcdTiming1)

Offset: 0h 08 **LcdTiming1: LCD Timing 1 Register** **Read/Write**

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

Table 14. LCD Timing 1 Register (LcdTiming1) Bit Descriptions

Bit	Name	Description
31:24	VBP	Vertical back porch The value (from 0–255) used to specify the number of line clock periods to add to the beginning of a frame before the first set of pixels is output to the display. Note that line clock transitions during the insertion of the extra line clock periods.
23:16	VFP	Vertical front porch The value (from 0–255) used to specify the number of line clock periods to add to the end of each frame. Note that the line clock transitions during the insertion of the extra line clock periods.
15:10	VSW	Vertical synchronization pulse width In active mode (LcdTFT=1), the encoded value (from 1–64) used to specify the number of line clock periods *(program to value required minus 1)* to pulse the frame clock (VSYNC) pin at the end of each frame *after* the end of frame wait (VFP) period elapses. The frame clock is used as VSYNC signal in active mode. In passive mode (LcdTFT=0), the encoded value (from 1–64) used to specify the number of extra line clock periods to insert *after* the vertical front porch (VFP) period has elapsed. Note that the width of the frame clock (VSYNC) is not affected by VSW in passive mode, and that the line clock transitions during the insertion of the extra line clock periods *(program to value required minus 1).*
9:0	LPP	Lines per panel The encoded value (from 1–1024) used to specify the number of lines per panel. It represents the total number of lines on the LCD.

Lines Per Panel (LPP)

The lines per panel (LPP) bit-field is used to specify the number of lines or rows per LCD panel being controlled. It represents the total number of lines for the entire LCD display (the screen height). LPP is a 10-bit value, which represents between 1−1024 lines per panel. LPP is used to count the correct number of line clocks that must occur before the frame clock can be pulsed.

Note:

LPP must be programmed to the value required minus 1 (0xC7 for 200 lines per panel).

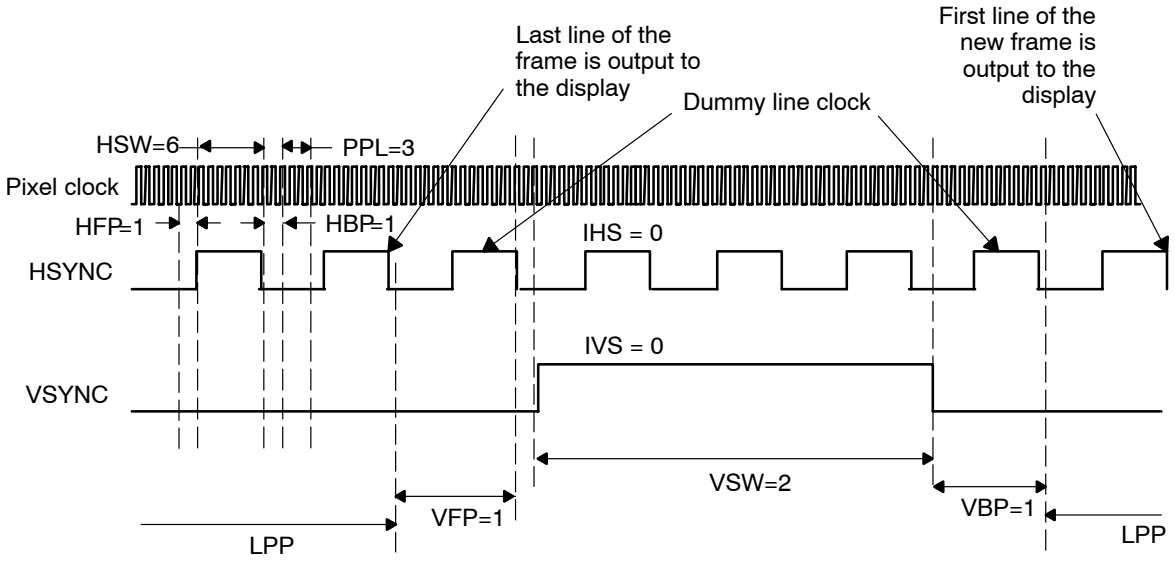
Vertical Synchronization Pulse Width (VSW)

The 6-bit vertical synchronization pulse width (VSW) field is used to specify the pulse width of the vertical synchronization pulse in active mode or is used to add extra dummy line clock cycles between the vertical front porch and vertical back porch in passive mode.

- Active matrix technology

  In active mode (LcdTFT=1), VSYNC is asserted each time the last line or row of pixels from the previous frame is output to the display and a programmable number of line clock delays (VFP) has elapsed. When the frame clock (VSYNC) is asserted, the value in VSW is transferred to a 6-bit down counter that uses the line clock frequency to decrement. When the counter reaches zero, the frame clock (VSYNC) is negated. VSW can be programmed to generate a vertical synchronization pulse width ranging from 1−64 line clock periods (program to value required minus 1, see Figure 26). The following frame starts after VSYNC is deasserted and a programmable number of line clock delays (VBP) has elapsed.

Figure 26. Active Matrix Timing



NOTE: Remember that most of the parameters (HSW, HFP, PPL, HBP) must be programmed to value required minus 1.

- Passive matrix technology

  In passive mode (LcdTFT=0), VSW does not affect the timing of the frame clock, but instead can be used to add extra line clock cycles between the end and beginning of frame line clock cycle counts. The total number of line clock cycles that are inserted between each frame is equal to the sum of the values in VFP, VSW, and VBP. A counter is used to insert dummy line clock cycles between frames by first using the value in VFP, then VSW, then VBP. You must ensure that the sum of the values in the three fields is equal to the total number of line clock cycles that are needed between frames.

  The LCD controller frame clock pin is asserted on the rising-edge of the first pixel clock for each frame. The frame clock remains asserted for the remainder of the first line as pixels are output to the display, also during the assertion of the first line clock for the frame, and then negated on the rising-edge of the first pixel clock of the second line of each frame.

Note:

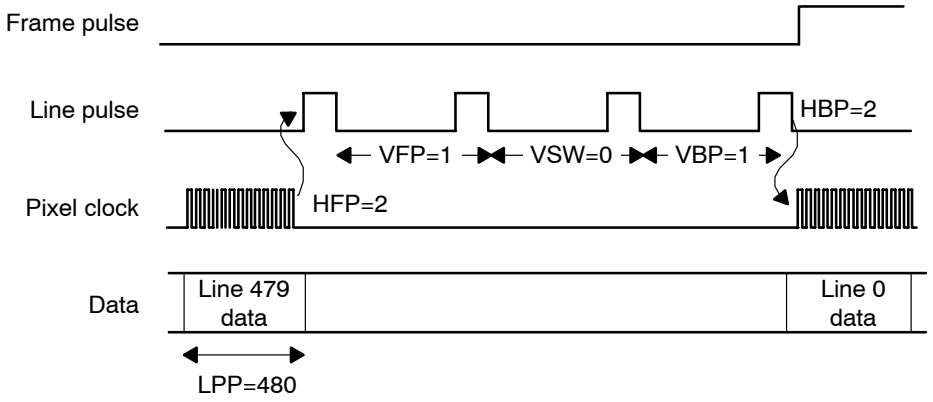
The pixel clock does not transition during the whole dummy line clock periods that are inserted in passive mode before the frame pulse.

The line clock does transition during the insertion of the dummy line clock cycles. VSW must be long enough to load the palette.

Vertical Front Porch (VFP)

The 8-bit vertical front porch (VFP) field is used to specify the number of line clocks to insert at the end of each frame. Once a complete frame of pixels is transmitted to the LCD display, the value in VFP is used to count the number of line clock periods to wait. After the count has elapsed, the VSYNC signal is pulsed in active mode or extra line clocks are inserted as specified by the VSW bit-field in passive mode. VFP generates from 0–255 line clock cycles (see Figure 27).

Figure 27. Passive Mode End of Frame Timing



NOTE: Remember that VSW must be programmed to value required minus 1.

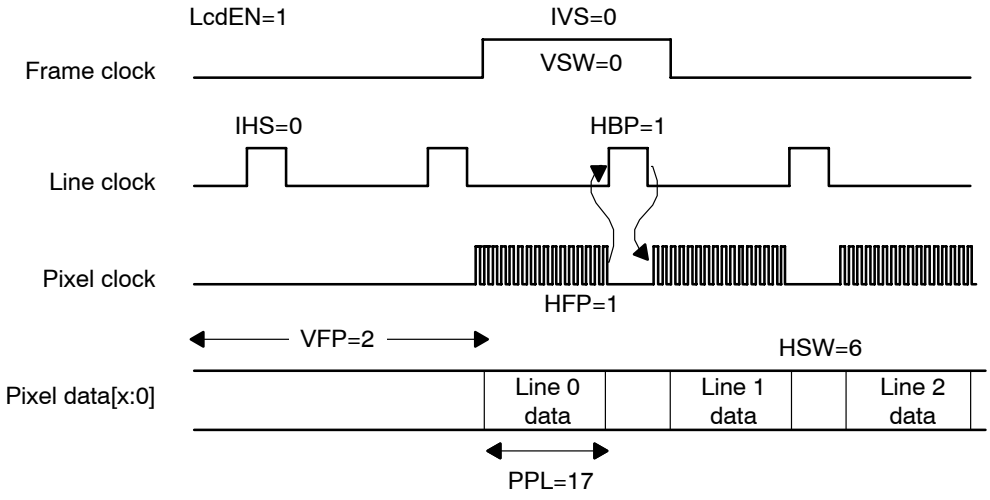
Note:

The line clock transitions during the generation of the VFP line clock periods.

Vertical Back Porch (VBP)

The 8-bit vertical back porch (VBP) field is used to specify the number of line clocks to insert at the beginning of each frame. The VBP count starts just after the VSYNC signal for the previous frame has been negated for active mode, or the extra line clocks have been inserted as specified by the VSW bit-field in passive mode. After this has occurred, the value in VBP is used to count the number of line clock periods to insert before starting to output pixels in the next frame. VBP generates from 0–255 extra line clock cycles (see Figure 28).

Figure 28. Passive Mode Beginning of Frame Timing



Note:

The line clock transitions during the generation of the VBP line clock wait periods. Note also that you must adjust the value of VBP, to allow enough line clock cycles to elapse; this allows the palette to be completely filled via the DMA, and allows a sufficient number of encoded pixel values to be input from the frame buffer, processed by the dither logic, then placed in the output FIFO, ready to be output to the LCD data lines.

2.3.4 LCD Timing 2 Register (LcdTiming2)

The LCD timing 2 register (LcdTiming2) contains nine different bit-fields that are used to control various functions associated with the timing of the LCD controller.

Figure 29. LCD Timing 2 Register (LcdTiming2)

Offset: 0h 0C **LcdTiming2: LCD Timing 2 Register** **Read/Write**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Reserved						ON_OFF	RF	IEO	IPC	IHS	IVS	ACBI			
Reset	x	x	x	x	x	x	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ACB								PCD							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15. LCD Timing 2 Register (LcdTiming2) Bit Descriptions

Bit	Name	Description
31-26	-	Reserved
25	ON_OFF	HSYNC/VSYNC pixel clock control on/off (should be ON only when in TFT mode) 0: Line clock (HSYNC) and frame clock (VSYNC) are driven on opposite edges of pixel clock than the pixel data. 1: Line clock (HSYNC) and frame clock (VSYNC) are driven according to bit 24.
24	RF	Program HSYNC/VSYNC RISE OR FALL 0: Line clock (HSYNC) and frame clock (VSYNC) are driven on falling edge of pixel clock, bit 25 must be set to 1. 1: Line clock (HSYNC) and frame clock (VSYNC) are driven on rising edge of pixel clock, bit 25 must be set to 1.
23	IEO	Invert output enable 0: ac-bias pin is active high in active display mode. 1: ac-bias pin is active low in active display mode. Active display mode: Data is driven out to the LCD data lines on programmed pixel clock edge when ac-bias is active,according to the IPC bit (see subsection *Invert Pixel Clock (IPC)*. Note that IEO is ignored in passive display mode.
22	IPC	Invert pixel clock 0: Data is driven on the LCD data lines on the rising-edge of the pixel clock. 1: Data is driven on the LCD data lines on the falling-edge of the pixel clock.
21	IHS	Invert HSYNC 0: Line clock (HSYNC) pin is active high and inactive low. 1: Line clock (HSYNC) pin is active low and inactive high. Active and passive mode: Horizontal synchronization pulse/line clock is active between lines, after end of line wait period.
20	IVS	Invert VSYNC 0: Frame clock (VSYNC) pin is active high and inactive low. 1: Frame clock (VSYNC) pin is active low and inactive high. Active mode: Vertical synchronization pulse is active between frames and after the end of the frame wait period. Passive mode: Frame clock is active during first line of each frame.
19:16	ACBI	ac-bias pin transitions per interrupt Value (from 0 to 15) used to specify the number of ac-bias pin transitions to count before setting the ac-bias count status bit signaling an interrupt request. The counter is frozen when bit ABC is set and is restarted when bit ABC is cleared by software. This function is disabled when ACBI=0x0000. This bit is only relevant in passive mode because it is used as an output enable in active mode. See subsection *Invert Output Enable (IEO)*. In active mode, this bit is ignored.

Table 15. LCD Timing 2 Register (LcdTiming2) Bit Descriptions (Continued)

Bit	Name	Description
15:8	ACB	ac-bias pin frequency (program to value required minus 1) Value (from 1 to 256) used to specify the number of line clocks to count before transitioning the ac-bias pin. This pin is used to periodically invert the polarity of the power supply to prevent dc charge build-up within the display. ACB = Number of line clocks/toggle of the ac-bias pin. This bit is relevant in passive mode because ac-bias is used as an output enable in active mode. In active mode, this bit is ignored.
7:0	PCD	Pixel clock divisor Value (from 2–255) used to specify the frequency of the pixel clock based on the LCD_CK frequency. Pixel clock frequency can range from LCD_CK /2 to LCD_CK /255. Pixel clock frequency = LCD_CK/PCD.

Pixel Clock Divider (PCD)

The 8-bit pixel clock divider (PCD) field is used to select the frequency of the pixel clock. PCD can generate a range of pixel clock frequencies from LCD_CK/2 to LCD_CK/255, where LCD_CK is derived from CK_GEN3 (see section *Timing)*. The pixel clock frequency must be adjusted to meet the required screen refresh rate. The refresh rate depends on:

- ❏ The number of pixels for the target display
- ❏ Whether monochrome or color mode is selected
- ❏ The number of pixel clock delays programmed at the beginning and end of each line
- ❏ The number of line clocks inserted at the beginning and end of each frame
- ❏ The width of the VSYNC signal in active mode or VSW line clocks inserted in passive mode
- ❏ The width of the line clock or HSYNC signal

All of these factors alter the duration of one frame transmission to the next. Different display manufacturers require different frame refresh rates, depending on the physical characteristics of the display. PCD is used to alter the pixel clock frequency in order to meet these requirements. Pixel clock is used to synchronously signal the device to drive data to the LCD data pins, and to signal the output FIFO to latch the data from the pins. The frequency of the pixel clock for a set PCD value or the required PCD value to yield a target pixel clock frequency can be calculated using the following equation:

PixelClock =LCD_CK / PCD

The pixel clock frequency is programmed taking into account the limitations shown in Table 16.

Table 16. Pixel Clock Frequency Programming Limitations

Type of Screen	Output (in Bits)	Min. Pixel Clock Divider
TFT 1, 2, 4, 8 BPP	12 (1 pixel)	2
TFT 12, 16 BPP	16 (1 pixel)	2
STN monochrome	4 (4 pixels)	4
STN monochrome	8 (8 pixels)	8
STN color	8 (2 2/3 pixels)	3

Note:

If PCD equals 0 or 1, the effect is undefined. Dividing the pixel clock frequency by an odd number distorts the duty cycle.

ac-Bias Pin Frequency (ACB)

The 8-bit ac-bias frequency (ACB) field is used to specify the number of line clock periods to count between each toggle of the ac-bias pin. After the LCD controller is enabled, the value in ACB is loaded to an 8-bit down counter, and the counter begins to decrement using the line clock. When the counter reaches zero it stops, the state of ac-bias pin is reversed, and the whole procedure starts again. The number of line clocks between each ac-bias pin transition ranges from 1–256 (program to value required minus 1). This line is used by the LCD display to periodically reverse the polarity of the power supplied to the screen to eliminate DC offset.

Note:

The ACB bit field has no effect in active mode. This is due to the fact that the pixel clock transitions continuously in active mode (when pixel_gated = 0); the ac-bias line is used as an output enable signal. The ac-bias is asserted by the LCD controller in active mode; this occurs whenever pixel data is driven out to the data pins to signal to the display when it can latch pixels using the pixel clock.

ac-Bias Line Transitions Per Interrupt (ACBI)

The 4-bit ac-bias line transitions per interrupt (ACBI) field is used to specify the number of line transitions to count before setting the ac-bias count status (ABC) bit in the LCD controller status register, which signals an interrupt request. After the LCD controller is enabled, the value in ACBI is loaded to a 4-bit down counter, and the counter decrements each time the ac-bias line state is inverted. When the counter reaches zero it stops, and the ac-bias count (ABC) bit is set in the status register. Once ABC is set, the 4-bit down counter is reloaded with the value in ACBI, and is disabled until ABC is cleared. Once ABC is cleared by the CPU, the down counter is enabled, and again decrements each time the ac-bias line is flipped. The number of ac-bias line transitions between each interrupt request ranges from 0 to 15. Programming ACBI = 0h0000 disables the ac-bias line transitions per interrupt function.

Note:

For the same reasons as in the previous section, this bit-field has no effect in active mode.

Invert VSYNC (IVS)

The invert VSYNC (IVS) bit is used to invert the polarity of the frame clock (VSYNC).

When IVS = 1, the frame clock (VSYNC) is active low.

When IVS = 0, it is active high.

Invert HSYNC (IHS)

The invert HSYNC (IHS) bit is used to invert the polarity of the line clock (HSYNC).

When IHS = 1, the line clock (HSYNC) is active low.

When IHS = 0, it is active high.

Invert Pixel Clock (IPC)

The invert pixel clock (IPC) bit is used to select the edge of the pixel clock that drives pixel data out onto the LCD data lines.

When IPC = 1, data is driven onto the LCD data lines on the falling edge of the pixel clock.

When IPC = 0, data is driven onto the LCD data lines on the rising edge of the pixel clock.

Invert Output Enable (IEO)

The invert output enable (IEO) bit is used to select the active or inactive state of the output enable signal in active display mode. In this mode, the ac-bias pin is used as an enable that signals the device when data is being actively driven out using the pixel clock.

When IEO = 1, the ac-bias pin is active low. In active display mode, data is driven onto the LCD data lines on the programmed edge of the pixel clock when ac-bias pin is in its active state (see subsection *Invert Pixel Clock (IPC).*

When IEO = 0, the ac-bias pin is active high.

Note:

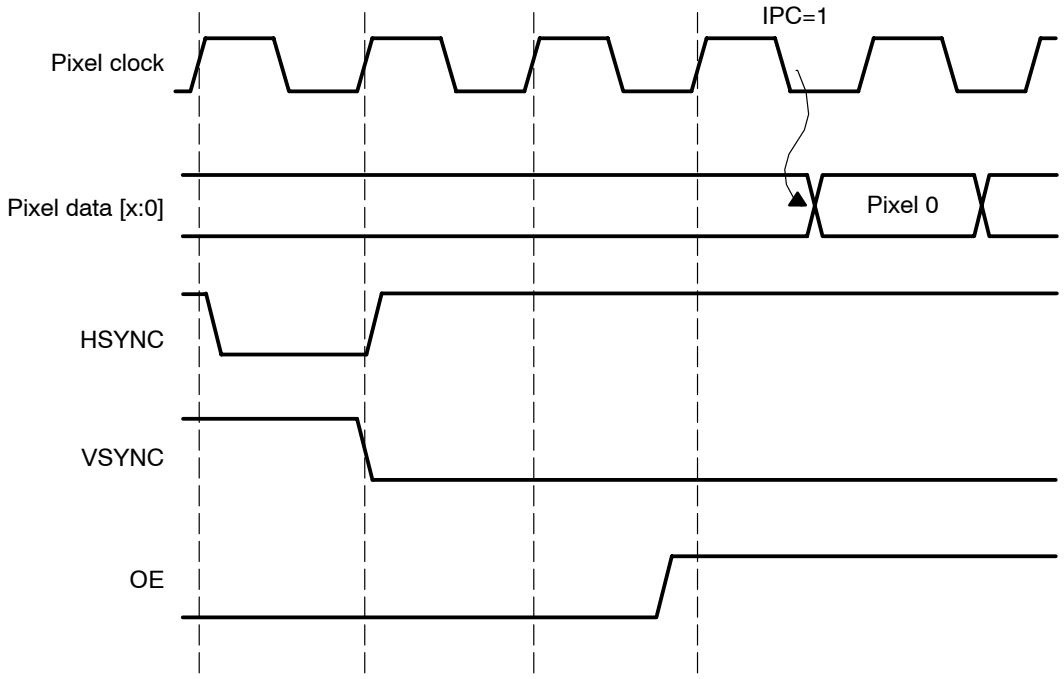
IEO does not affect the ac-bias pin in passive display mode.

HSYNC/VSYNC Rise or Fall Programmable (or not)(RF)

This bit determines whether the HSYNC/VSYNC is driven on the rising or falling edge of the pixel clock (see the HSYNC/VSYNC ON_OFF bit; ON_OFF must be turned on first). By default, the HSYNC and VSYNC signals are driven on the falling edge of the pixel clock, and the pixel data is driven on the rising edge of pixel clock. However, if the invert pixel clock (IPC) bit is set to 1, then the HSYNC and VSYNC signals are driven on rising edge of pixel clock and pixel data is driven on falling edge. By setting the RF bit and enabling it (ON_OFF = 1), you can control on which edge the signals are driven.

Figure 30 shows the timing when ON_OFF = 0 and IPC =1 in TFT mode.

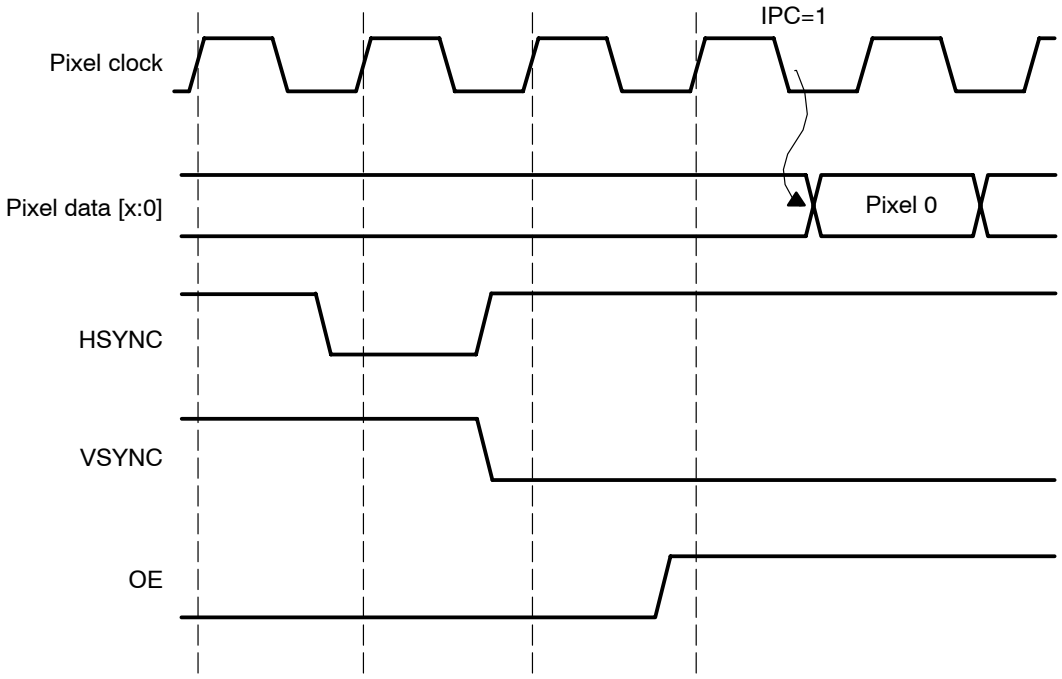
Figure 30. ON_OFF = 0, IPC = 1 in TFT Mode



- IPC = 1 means that pixel data is driven onto the LCD data lines on the falling edge of the pixel clock.
- ON_OFF = 0 means that HSYNC and VSYNC signals are driven on opposite edges of the pixel clock from pixel data (=> rising edge).

Figure 31 shows timing when ON_OFF = 1, RF = 0, and IPC = 1.

Figure 31. ON_OFF = 1, RF = 0, and IPC = 1



- When ON_OFF = 1, HSYNC and VSYNC signals are driven according to the RF bit.
- When RF = 0, HSYNC and VSYNC signals are driven on the falling edge of the pixel clock.
- When IPC = 1, pixel data is driven on the falling edge of the pixel clock.

HSYNC/VSYNC ON or OFF (ON_OFF)

This bit enables/disables the option to make HSYNC and VSYNC programmable.

- When ON_OFF = 1, HSYNC and VSYNC are driven according to the RF bit.
- When ON_OFF = 0, HSYNC and VSYNC are driven on opposite edges of the pixel clock from pixel data.

2.3.5 LCD Controller Status Register (LcdStatus)

The LCD controller status register (LcdStatus) contains seven bits that detect different events. Each of these hardware-detected events signals an interrupt request to the interrupt controller.

Figure 32. LCD Status Register (LcdStatus)

Offset: 0x 10 **LCDStatus: LCD Status Register** **Read/Write and Read-Only**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Reserved															
Reset	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved									LP	FUF	line_ int	ABC	Sync Lost	VS	Done
Reset	x	x	x	x	x	x	x	x	x	0	0	0	0	0	0	0

NOTE: Read-only/clear-only bits are cleared by writing 0 to them.

Table 17. LCD Status Register (LcdStatus) Bit Descriptions

Bit	Name	Description
31:7	-	Reserved
6	LP	Loaded palette (read/clear-only) LP = 0, as long as the palette is not loaded. LP is set to 1 when the palette is loaded.
5	FUF	FIFO underflow status (read-only) FUF = 0, as long as FIFO has not underrun. FUF is set to 1 when the LCD dither logic is not supplying data to FIFO at a sufficient rate, FIFO has completely emptied, and data pin driver logic has attempted to take added data from FIFO.
4	line_int	Line interrupt (read/clear-only) Line_int = 0, as long as the display has not reached the programmed line yet, or if this interrupt has been cleared. Line_int is set to 1 when the display reaches the user-programmed line number and generates the interrupt.
3	ABC	ac-bias count status (read/clear only) ABC = 0, as long as ac-bias transition counter has not decremented to 0. See subsection *ac-Bias Line Transitions Per Interrupt (ACBI)*. ABC is set to 1 when the ac-bias transition counter has decremented to zero, indicating that the ac-bias output line has transitioned the number of times, specified by the ACBI control bit-field. Counter is reloaded with the value in ACBI but is disabled until you clear ABC.
2	Sync_ lost	Synchronization lost (read-only) When Sync_lost = 0, no frame synchronization error occurred. When Sync_lost = 1, frame synchronization lost occurred.

Table 17. LCD Status Register (LcdStatus) Bit Descriptions (Continued)

Bit	Name	Description
1	VS	VSYNC interrupt (read/clear only) VS = 0, as long as VSYNC interrupt is not generated. VS is set to 1 when the VSYNC interrupt occurs at the end of frame.
0	Done	Frame done (read-only) Done is set to 0, as long as the LCD is enabled. Done is set to 1 when the LCD is disabled and the active frame is just completed.

Frame Done (Done) (Read-Only)

When the LCD is disabled by clearing the LCD enable bit (LcdEn=0) in the LcdControl register, the LCD allows the current frame to complete before it is disabled. After the last set of pixels is clocked out onto the LCD data pins by the pixel clock, the LCD is disabled and Done is set.

- ❏ Done = 1 when the frame is complete.
- ❏ Done = 0, as long as the frame is not complete.

The frame done (Done) bit is a read-only bit signaling that the frame is complete. It is cleared when LcdEn bit is set to 1 (turned ON).

VSYNC Interrupt (VS) (Read/Clear-Only)

VSYNC interrupt occurs when the LCD reaches the end of the frame. This interrupt is shared with other LCD interrupts and is output on the lcd_nirq interrupt output line (see Figure 33, *Line Interrupt Path*). You can unmask the VSYNC interrupt by writing 1 to the VSYNC_mask bit in the LCD control register (see section 2.3.1, *LCD Control Register*).

To clear the VSYNC bit in the status register, you must write 0 to it.

- ❏ VSYNC = 1 signals that the end of frame occurred and generated the VSYNC interrupt.
- ❏ VSYNC = 0 if no VSYNC interrupt is generated.

Frame Synchronization Lost (Sync_lost) (Read-Only)

The frame synchronization lost (Sync_Lost) bit is set if the LCD controller detects a frame synchronization error. A frame synchronization error happens when the LCD attempts to read what it believes to be the first word of the video buffer but it cannot be recognized as such. This bit is cleared by disabling the LCD controller (LcdEn bit =0). This also resets the input FIFO in the DMA controller.

- Sync_lost = 1 when a frame synchronization lost occurred.
- Sync_lost = 0, as long as no frame synchronization error occurs.

ac-Bias Count Status (ABC) (Read/Clear-Only)

The ac-bias count status (ABC) bit is set each time the ac-bias line transitions a particular number of times, as specified by the ac-bias line transitions per interrupt (ACBI) field in LcdTiming2. If ACBI is programmed with a non-zero value, a counter is loaded with the value in ACBI and is decremented each time the ac-bias line reverses state. When the counter reaches zero, the ABC bit is set, which signals an interrupt request to the interrupt controller. The counter reloads using the value in ACBI, but does not start to decrement again until you clear ABC by writing 0 to the LCD status register.

- ABC = 1 when the ac-bias transition counter ACBI has decremented to 0.
- ABC = 0, as long as ACBI has not decremented to 0.

Line Interrupt (line_int) (Read/Clear-Only)

You can program the LCD to generate a line interrupt when the LCD reaches a certain line (n) in the display. This interrupt sets the line_int bit in the status register. This dedicated line interrupt can be connected to the hardware synchronized channel of the DMA. It can be used to prevent a tearing effect if double buffering is not implemented. See section *Tearing Effect,* for information on the use of this feature.

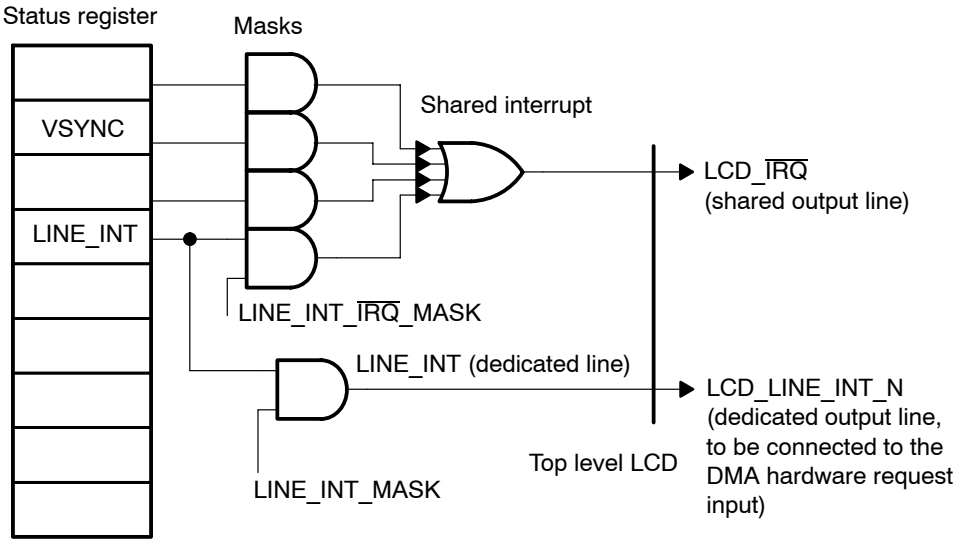
It is possible to connect the line_int interrupt to both the lcd_nirq interrupt output line and to a dedicated lcd_line_int_n output line.

On one hand, the line_int_mask bit masks the dedicated line interrupt. You can unmask it by writing 1 to the control register. Line_int is a status bit which can be cleared by writing 0 to it.

On the other hand, the line_int_nirq_mask bit in the control register masks the shared interrupt.

See Figure 33 for details.

Figure 33. Line Interrupt Path



- Line_int is set to 1 when the display reaches the user-programmed line number and generates the interrupt.
- Line_int = 0, as long as the programmed line is not reached.

The line_int bit is cleared if the TIPB writes 0 to it, or at the end of the programmed line according to the line_int_clr_sel bit in the control register.

FIFO Underflow Status (FUF) (Read-Only)

The FIFO underflow status (FUF) bit is set when the input FIFO is completely empty and the LCD data pins driver logic attempts to fetch data from the FIFO. This bit is cleared by disabling the LCD controller (LcdEn = 0). This also resets the input FIFO in the DMA controller.

- FUF = 1 when the dithering logic is not supplying data to the FIFO at a sufficient rate.
- FUF = 0, as long as FIFO has not underrun.

Loaded Palette (LP) (Read/Clear-Only)

The loaded palette (LP) bit is a read-only bit that is set after the LCD finished loading the palette into memory.

- LP = 1 when the palette is loaded.
- LP = 0, as long as the palette is not loaded.

In data-only (PLM = 10) and palette-plus-data (PLM = 00) modes, write 0 to clear the interrupt. However, in the palette only (PLM = 01) mode, LCD must be turned off in order to reset/clear the interrupt. Make sure not to turn off the LCD before getting the loading interrupt in this mode. See subsection *Palette Loading (PLM).*

2.3.6 LCD Subpanel Display Register (LcdSubpanel)

The LCD subpanel display register (LcdSubpanel) enables displaying only the first or last X lines of the panel and sending fixed content for the other lines.

Figure 34. LCD Subpanel Register (LcdSubpanel)

Offset: 0h 14 **LCDSubpanel: LCD Subpanel Register** **Read and Write**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	SPEN	Res	HOLS	Reserved			LPPT									
Reset	0	x	0	x	x	x	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DPD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18. LCD Subpanel (LcdSubpanel) Bit Descriptions

Bit	Name	Description
31	SPEN	Subpanel enable
		SPEN = 0: Subpanel function is disabled.
		SPEN = 1: Subpanel function is enabled.
30		RESERVED
29	HOLS	High or low signal
		The field indicates the position of the subpanel compared to the LPPT value.
		HOLS = 0: The image from the frame buffer is displayed below the LPPT value.
		HOLS = 1: The image from the frame buffer is displayed above the LPPT value.
28:26		RESERVED

Table 18. LCD Subpanel (LcdSubpanel) Bit Descriptions (Continued)

Bit	Name	Description
25:16	LPPT	Line per panel threshold Value (from 1 to 1023) delimiting the subpanel and the DPD parts of the screen. LPPT is a threshold value delimiting the subpanel and the DPD parts of the screen. It ranges from 1 to 1024 and should be programmed to value required minus one (0-1023).
15:0	DPD	Default pixel data DPD defines the default value of the pixel data sent to the panel for the lines until the LPPT threshold is reached or after passing the LPPT depending on HOLS mode.

DPD, LPPT, HOLS, and SPEN bit fields and bits are not considered if SPEN=0.

Default Pixel Data (DPD)

DPD defines a default value, which is sent to the display in either the top or bottom region of the screen delimited by the LPPT threshold. When displaying the DPD value, there is no DMA activity.

Line-per-Panel Threshold (LPPT)

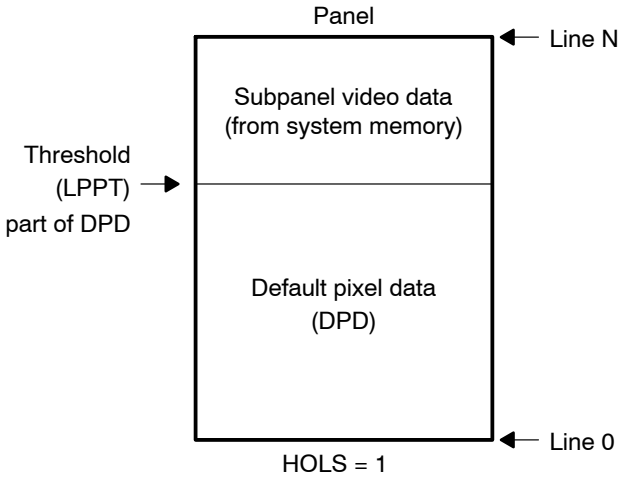
The line-per-panel threshold bit-field delimits the screen portion filled with data fetched from the frame buffer (the subpanel) and the rest of the screen filled with default pixel data (DPD). Note that the LPPT line number points on a line filled with a DPD value when HOLS = 1, but on a line filled with video data when HOLS = 0 (see Figure 35 and Figure 36).

High Or Low Signal (HOLS)

The HOLS bit indicates the position of the subpanel compared to the LPPT value.

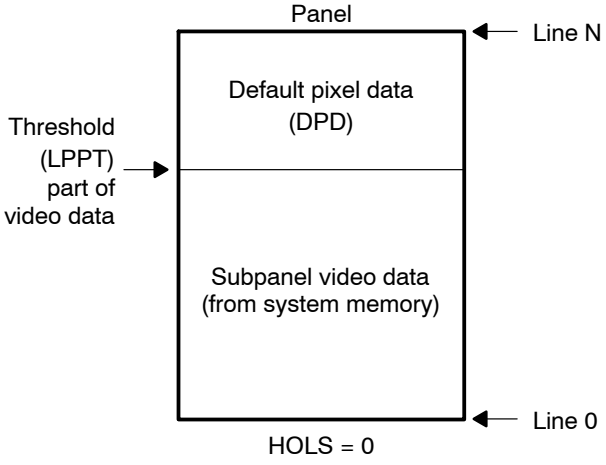
When HOLS = 1, the image from system memory is displayed above the threshold value. The threshold value is the line number where the DPD value begins to be displayed. The rest of the screen is filled with DPD value.

Figure 35. Subpanel Display: SPEN = 1, HOLS = 1



When HOLS = 0, the beginning of the screen is filled with DPD value until the LPPT excluded. From the LPPT line number, the rest of the screen (below LPPT) displays the image from system memory.

Figure 36. Subpanel Display: SPEN = 1, HOLS = 0



Note:

The bottom of the panel is line 0, and top line of the panel is line N (where N is the number of lines-per-panel).

For example, if you want to display four lines of video data at the bottom of the panel, the correct settings are HOLS = 0 and LPPT = 3. Here, the amount of video data to be transferred from the DMA_LCD channel is only four lines.

Note:

If the LPPT is above the number of LPP, then:

- When HOLS = 1: panel with default data (whole panel is filled with DPD value).
- When HOLS = 0: normal panel (whole panel is filled with video data from the frame buffer).

Subpanel Enable (SPEN)

This bit enables or disables the subpanel mode.

When SPEN = 0, subpanel mode is disabled.

When SPEN = 1, subpanel mode is enabled.

2.3.7 Line Interrupt Register (LcdLineInt)

The line interrupt register (LcdLineInt) enables you to program the line number in bits (0-9) where you want the line interrupt to be generated.

Figure 37. Line Interrupt Register (LcdLineInt)

Offset: 0h 18 **LCDLineInt: Line Interrupt Register** **Read and Write**

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

Table 19. Line Interrupt Register (LcdLine Int) Bit Descriptions

Bit	Name	Description
31:10	-	Reserved
9:0	Line_int_number	Line number at which line interrupt occurs. Programmable from line 0 up to line 1023.

2.3.8 Display Status Register (LcdDisplayStatus)

The display status register (LcdDisplayStatus) contains the line number currently being displayed.

Figure 38. Display Status Register (LcdDisplayStatus)

Offset: 0h 1C	LCDDisplayStatus: Display Status Register											Read-Only				
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Reserved															
Reset	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved						CURRENT_LINE_NUMBER									
Reset	x	x	x	x	x	x	1	1	1	1	1	1	1	1	1	1

Table 20. Line Interrupt Register (LcdDisplayStatus) Bit Descriptions

Bit	Name	Description
31:10	-	Reserved
9:0	Current_line_number	Line number being displayed. As the number of lines can be programmed from 1 to 1024, the current line number varies between 0 and 1023.

3 LCD Data Conversion Module

The LCD data conversion module (LCDCONV) supports two operation modes: the 16-bit LCD mode and the 18-bit LCD mode. The mode switching is done by setting the MODE_SET bit in the control register (LCDCONV_CONTROL_REG). The mode change is synchronized with the LCD pixel frame synchronization signal. At the active edge of the pixel frame synchronization, the MODE_SET bit in the control register is sampled and the right mode is set up accordingly. Software reads the MODE_STATUS bit in the control register to know whether the LCD mode has changed.

In 16-bit LCD mode, the 16-bit LCD pixel signal from the LCD controller bypasses the RGB look-up table and goes directly to the external LCD display. The LCD RGB look-up table is accessible only in this mode. The host microcontroller programs and reads back the contents of the RGB look-up table.

In 18-bit LCD mode, the 16-bit LCD pixel signal is converted to an 18-bit LCD pixel signal through an RGB look-up table. The input 16-bit LCD pixel signal is used as an index to a programmable RGB look-up table to generate an 18-bit pixel signal for the external LCD display. The OCP bus cannot access the RGB look-up table in this mode. An OCP bus read in this mode reads all zeros. For an OCP bus write, nothing is written to the RGB look-up table. The size of the look-up table is defined in Table 21.

Table 21. RGB Lookup Table Size

RGB Lookup Table	Lookup Table Size (Bits)	Index (LCD Input Pixel Bit)
R	32 x 6	0-4
G	64 x 6	5-10
B	32 x 6	11-15

3.1 Data Conversion

In the 18-bit mode, the 16-bit LCD signals (5 bits for R, 6 bits for G, and 5 bits for B) are used as the address for the LUT. The LUT is divided into three sections: R LUT 32x6 bits, G LUT 64x6 bits, and B LUT 32 x 6 bits. The word width of the three LUTs is 6 bits. The length of the LUT is 32 lines red, 64 lines green, and 32 lines blue. These correspond to the 16-bit RGB signal (5 red, 6 green, and 5 blue) address decoding. The user can program the content of the LUT so that the application determines which conversion algorithm is necessary. The converted 18-bit LCD signals are mapped to the LCDCONV LCD_PIXEL_OUT output ports. The LCDCONV output LCD signal consists of two parts: LCD_PIXEL_OUT[15:0], RED_LSB and BLUE_LSB. The mapping is as follows:

In 18-bit mode:

- LCD_PIXEL_OUT[15:11] <= R_LUT[5:1]
- LCD_PIXEL_OUT[10:5] <= G_LUT[5:0]
- LCD_PIXEL_OUT[4:0] <= B_LUT[5:1]
- RED_LSB <= R_LUT[0]
- BLUE_LSB <= B_LUT[0]

In 16-bit mode:

- LCD_PIXEL_OUT[15:0] <= LCD_PIXEL_IN[15:0]
- RED_LSB <= LCD_PIXEL_IN[15]
- BLUE_LSB <= LCD_PIXEL_IN[4]

Table 22 summarizes the mapping for the16-bit and 18-bit modes.

Table 22. LCD 16-Bit to 18-Bit Conversion

	RED						GREEN						BLUE					
LCD pixel input to LDCONV	15	14	13	12	11		10	9	8	7	6	5	4	3	2	1	0	
LCD pixel output (16-bit mode)	R4	R3	R2	R1	R0	R4 (R_LSB)	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0	B4 (B_LSB)
LCD output 1(18-bit mode)	R5	R4	R3	R2	R1	R0 (R_LSB)	G5	G4	G3	G2	G1	G0	B5	B4	B3	B2	B1	B0 (B_LSB)

Figure 39 shows the 16-bit to 18-bit LCD data block.

Figure 39. 16-Bit to 18-Bit LCD Data Block

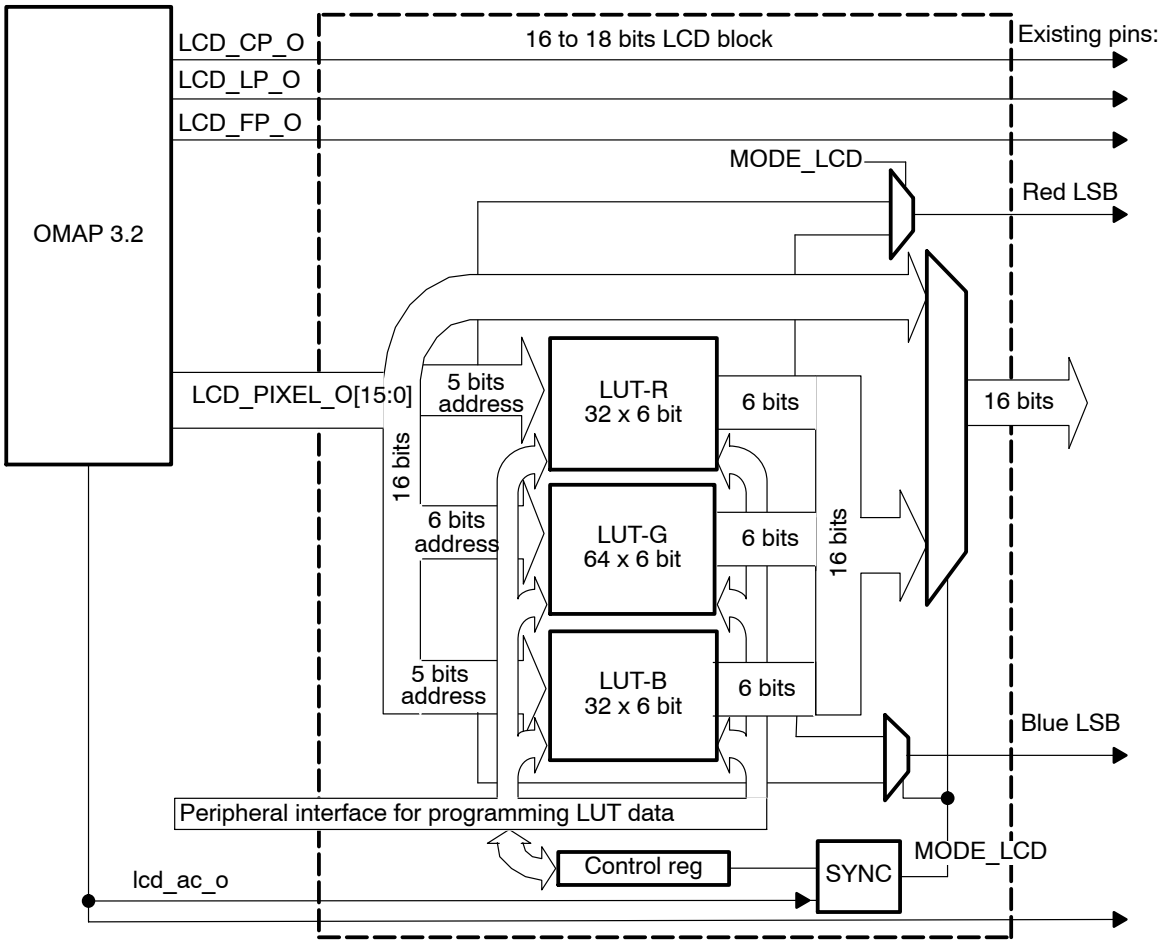


Table 23. Top Level I/O Signals

Signal Name	I/O	Description	Reset
		Clock and Reset:	
CLK	Input	Free-running OCP bus clock	N/A
RST_N	Input	Asynchronous global reset, active low	N/A
		OCP Bus Interface Signals	
MADDR[7:0]	Input	OCP address bus from the master	N/A
MCMD[2:0]	Input	Input transfer command (idle, read, write mode only)	N/A
MDATA[7:0]	Input	OCP data bus from the master	N/A

Table 23. Top Level I/O Signals (Continued)

Signal Name	I/O	Description	Reset
SDATA_OUT[7:0]	Output	OCP data bus to the master	0
SCMDACCEPT	Output	OCP command accepts transfer	0
SRESP	Output	OCP response field from the slave to transfer request from the master	0
		LCD signals	
LCD_PIXEL_IN[15:0]	Input	Pixel signals from the LCD controller	N/A
LCD_AC	Input	LCD frame sync from the LCD controller	N/A
LCD_PIXEL_OUT[15:0]	Output	Pixel signals to the external LCD display	LCD_PIXEL_IN [15:0]
RED_LSB	Output	Red LSB signal for the 18-bit LCD mode	LCD_PIXEL_IN [15]
BLUE_LSB	Output	Blue LSB signal for the18-bit LCD mode	LCD_PXIEL_IN [4]

3.2 Software Interface

Table 24. Register Summary

Base Address = FFFE 3000

Address	Type	Bit	Name	Description
0x0000 -0x001F	MG register file	5:0	R look-up table	32 x 6 look-up table for R signal
		6:7	Reserved	Read as 0, no impact on write
0x0020 –0x003F	MG register file	5:0	B look-up table	32 x 6 look-up table for B signal
		6:7	Reserved	Read as 0, no effect on write
0x0040– 0x007F	MG register file	5:0	G look-up table	64 x 6 look-up table for G signal
		6:7	Reserved	Read as 0, no impact on write
0x0080	Register	3:0	LCDCONV_CONTROL_REG	Control register
		4:7	Reserved	Read as 0, no effect on write
0x0084	Register	7:0	DEV_REV_REG	Device revision register, read only

The RGB look-up table icon consists of 128 x 6-bit, two-port MG RAM as shown in Table 24. The host microcontroller can program and read the RAM only in 16-bit mode. In 18-bit mode, the content of the look-up table is sent directly to the LCD display.

Table 25. Control Register (LCDCONV_CONTROL_REG)

Bit	Name	Function	R/W	Reset
7:4	Reserved	Reserved.	R	0
3	LCD_AC_EDGE	This bit represents the active edge of the LCD frame sync signal. 0: Active edge is positive. 1: Active edge is negative.	R/W	0
2	CLOCK_EN	Enables the write clock to RGB look-up table RAM	R/W	0
1	MODE_STATUS	Actual mode of LCD. 0: 16-bit mode. 1: 18-bit mode.	R	0
0	MODE_SET	This bit represents the software setup for the LCD mode. The actual mode switching occurs at the active edge of the pixel frame sync signal. 0: 16-bit mode. 1: 18-bit mode.	R/W	0

The control register has four control bits.

The CLOCK_EN bit gates the clock to the module. When this bit is on, the OCP bus clock clocks the RGB look-up table. When this bit is set to 0, the RGB look-up table clock is turned off. The OCP bus clock always clocks the control register.

The two mode-control bits are MODE_SET and MODE_STATUS. The MODE_STATUS bit is a read-only bit that represents the actual state of the LCD display mode. The MODE_SET bit is used for LCD mode programming. Software programs this bit to choose either the 18-bit mode or the 16-bit mode for the LCD. The actual mode switching occurs only at the next active edge of the pixel frame synchronization signal. When the mode is switched, the MODE_STATUS bit is updated to the MODE_SET bit. Using these two mode bits, the software can monitor the control register to determine whether the mode switching has occurred.

The active edge of the pixel frame synchronization signal is programmable. The LCD_AC_EDGE bit represents the polarity of the pixel frame synchronization and is used for the mode switching.

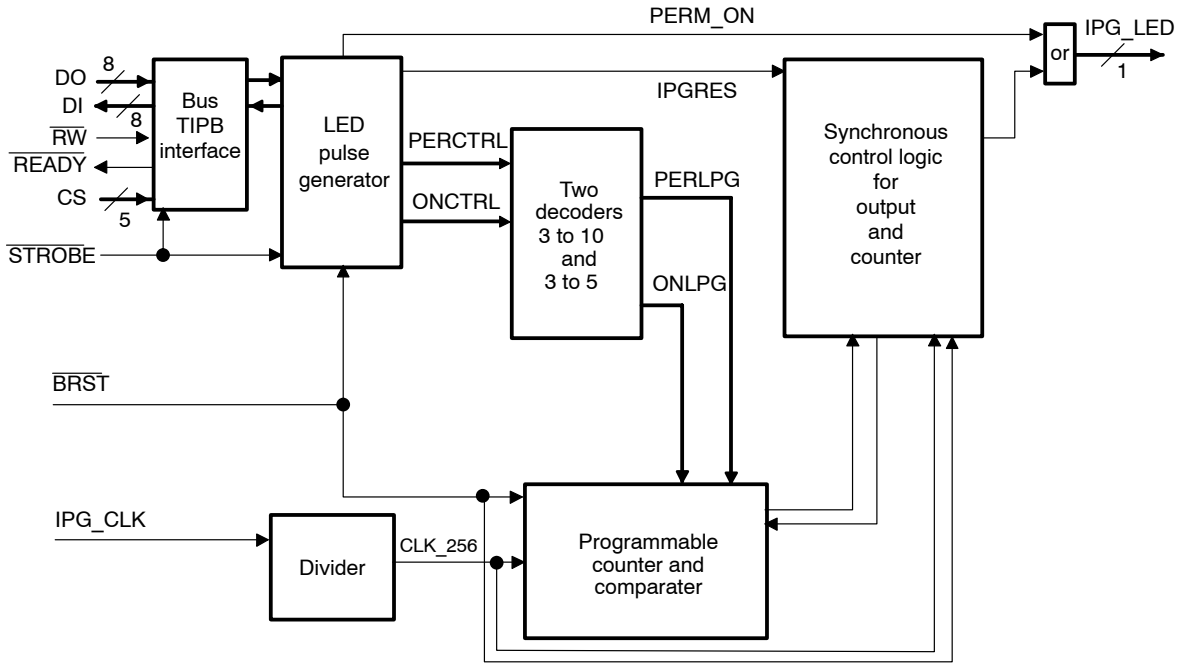
3.3 Bus Interface

The host microcontroller accesses the RGB look-up table and the control register through the OCP bus. The LDCONV module supports only 8-bit OCP bus access. No burst mode is supported. The RGB look-up table is accessible only by the OCP bus in the 16-bit mode. In the 18-bit mode, an OCP read reads in all zeros, and an OCP write does nothing to the RGB look-up table.

4 LED Pulse Generator

The LED pulse generator (LPG) module controls an indication LED (see Figure 40). The blinking period is programmable between 152 ms and 4 s, and the LED can be switched on permanently.

Figure 40. LED Pulse Generator Block Diagram



4.1 Features

The LPG has the following features:

- Divider generating a 256-Hz frequency clock
- TIPB control interface
- Two 8-bit registers to control the whole LPG block
- Decoder for three blink-frequency control bits (LPG2-0)
- Decoder for three pulse-width control bits (LPG5-3)
- Programmable counter with integrated comparison for the PWM
- Synchronous control logic for the output and the counter
- Multiplexer to generate a faster clock for testing

Table 26 lists the LPG functional I/O signals.

Table 26. LPG Functional I/O Signals

Name	Description	Type	Size	Active Level	Reset Level
nRESET	Asynchronous general reset	IN	1	0	--
LPG_CLK	LPG functional clock, 32-kHz frequency	IN	1	--	--
LPG_LED	Control LED signal	OUT	1	1	0

4.2 LPG Design

LCR bit 6 = 0 resets the whole PWM circuit (but not the control register) and switches off the LED. It is possible to switch on the LED independently from the PWM circuit with bit 7 of the LCR (1 = permanent light). The reset PWRON is active-low and resets the whole LPG (with the control register) and the output LPG_LED to zero asynchronously.

4.3 LPG Power Management

The LPG input clock comes from the 32-kHz ULPD clock, because it must work even when the system is in deep sleep mode. The internal clock of the LPG runs with 256 Hz, so the power consumption of this block can be neglected. Nevertheless, the LPG_CLK must be switched off if LPG is not used.

4.4 LPG Registers

LPG registers are mapped in the MPU address space.

Two instances of LPG are mapped in the device:

- First LPG: LPG_1 address is FFFB:D000
- Second LPG: LPG_2 address is FFFB:D800

Table 27 lists the LPG receive and transmit registers. Table 28 and Table 31 describe the register bits.

Table 27. LED Pulse Generator Receive and Transmit Registers

Register	Description	Access	Field Size	Offset (hex)
LCR	LPG control	R/W	8 bits	0x00
PMR	Power management	R/W	8 bits	0x04

Table 28. LPG Control Register (LCR)

Bit	Name	Function	R/W	Reset
7	PERM_ON	Set high to force permanent light on. Asynchronous writing and reading.	R/W	0
6	LPGRES	LPG counter reset active low. Asynchronous writing and reading.	R/W	0
5:3	ONCTRL	Time LED is on parameter. Asynchronous writing and reading.	R/W	000
2:0	PERCTRL	LED blink frequency. Asynchronous writing and reading.	R/W	000

The blinking period of the LED is determined with the LCR bits 2-0.

Table 29. LED Blinking Period

LCR Bit 2	LCR Bit 1	LCR Bit 0	Period of LED	No. of Clock Cycles
0	0	0	125 ms	32
0	0	1	250 ms	64
0	1	0	500 ms	128
0	1	1	1 s	256
1	0	0	1.5 s	384
1	0	1	2 s	512
1	1	0	2.5 s	640
1	1	1	3 s	768

The on-time of the LED is determined with the LCR bits 5-3.

Table 30. LED On Time

LCR Bit 5	LCR Bit 4	LCR Bit 3	Time LED On	No. of Clock Cycles
0	0	0	3.889 ms	1
0	0	1	7.789 ms	2
0	1	0	15.59 ms	4
0	1	1	31.39 ms	8
1	0	0	46.59 ms	12
1	0	1	62.59 ms	16
1	1	0	78.39 ms	20
1	1	1	93.59 ms	24

Table 31. Power Management Register (PMR)

Bit	Name	Function	R/W	Reset
0	CLK_EN	Functional clock enable: 1: Clock enabled. 0: Clock disabled. Asynchronous writing and reading.	R/W	0

This page is intentionally left blank.

Index

OMAP5912 Multimedia Processor Multimedia Card (MMC/SD/SDIO) Interface Reference Guide

Literature Number: SPRU765A
March 2004

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.

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
		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 document describes the multimedia card (MMC) interface of the OMAP5912 multimedia processor.

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.

Related Documentation From Texas Instruments

The following documents describe the OMAP5910 device and related peripherals. Copies of these documents are available on the Internet at www.ti.com. *Tip:* Enter the literature number in the search box provided at www.ti.com.

OMAP5912 Multimedia Processor Device Overview and Architecture Reference Guide (literature number SPRU748) introduces the setup, components, and features of the OMAP5912 multimedia processor and provides a high-level view of the device architecture.

OMAP5912 Multimedia Processor OMAP 3.2 Subsystem Reference Guide (literature number SPRU749) introduces and briefly defines the main features of the OMAP3.2 subsystem of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor DSP Sybsystem Reference Guide (literature number SPRU750) describes the OMAP5912 multimedia processor DSP subsystem. The digital signal processor (DSP) subsystem is built around a core processor and peripherals that interface with: 1) The ARM926EJS via the microprocessor unit interface (MPUI); 2) Various standard memories via the external memory interface (EMIF); 3) Various system peripherals via the TI peripheral bus (TIPB) bridge.

OMAP5912 Multimedia Processor Clocks Reference Guide (literature number SPRU751) describes the clocking mechanisms of the OMAP5912 multimedia processor. In OMAP5912, various clocks are created from special components such as the digital phase locked loop (DPLL) and the analog phase-locked loop (APLL).

OMAP5912 Multimedia Processor Initialization Reference Guide (literature number SPRU752) describes the reset architecture, the configuration, the initialization, and the boot ROM of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Power Management Reference Guide (literature number SPRU753) describes power management in the OMAP5912 multimedia processor. The ultralow-power device (ULPD) generates and manages clocks and reset signals to OMAP3.2 and to some peripherals. It controls chip-level power-down modes and handles chip-level wake-up events. In deep sleep mode, this module is still active to monitor wake-up events.This book describes the ULPD module and outline architecture.

OMAP5912 Multimedia Processor Security Features Reference Guide (literature number SPRU754) describes the security features of teh OMAP5912 multimedia processor. The OMAP5912 security scheme relies on the OMAP3.2 secure mode. The distributed security on the OMAP3.2 platform is a Texas Instruments solution to address m-commerce and security issues within a mobile phone environment. The OMAP3.2 secure mode was developed to bring hardware robustness to the overall OMAP5912 security scheme.

OMAP5912 Multimedia Processor Direct Memory Access (DMA) Support Reference Guide (literature number SPRU755) describes the direct memory access support of the OMAP5912 multimedia processor. The OMAP5912 processor has three DMAs:

- The system DMA is embedded in OMAP3.2. It handles DMA transfers associated with MPU and shared peripherals.
- The DSP DMA is embedded in OMAP3.2. It handles DMA transfers associated with DSP peripherals.
- The generic distributed DMA (GDD) is an OMAP5912 resource attached to the SSI peripheral. It handles only DMA transfers associated with the SSI peripheral.

OMAP5912 Multimedia Processor Memory Interfaces Reference Guide (literature number SPRU756) describes the memory interfaces of the OMAP5912 multimedia processor.

- SDRAM (external memory interface fast, or EMIFF)
- Asynchronous and synchronous burst memory (external memory interface slow, or EMIFS)
- NAND flash (hardware controller or software controller)
- CompactFlash on EMIFS interface
- Internal static RAM

OMAP5912 Multimedia Processor Interrupts Reference Guide (literature number SPRU757) describes the interrupts of the OMAP5912 multimedia processor. Three level 2 interrupt controllers are used in OMAP5912:

- One MPU level 2 interrupt handler (also referred to as MPU interrupt level 2) is implemented outside of OMAP3.2 and can handle 128 interrupts.
- One DSP level 2 interrupt handler (also referred to as DSP interrupt level 2.1) is instantiated outside of OMAP3.2 and can handle 64 interrupts.
- One OMAP3.2 DSP level 2 interrupt handler (referenced as DSP interrupt level 2.0) can handle 16 interrupts.

OMAP5912 Multimedia Processor Peripheral Interconnects Reference Guide (literature number SPRU758) describes various periperal interconnects of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Timers Reference Guide (literature number SPRU759) describes various timers of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Serial Interfaces Reference Guide (literature number SPRU760) describes the serial interfaces of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Universal Serial Bus (USB) Reference Guide (literature number SPRU761) describes the universal serial bus (USB) host on the OMAP5912 multimedia processor. The OMAP5912 processor provides several varieties of USB functionality. Flexible multiplexing of signals from the OMAP5912 USB host controller, the OMAP5912 USB function controller, and other OMAP5912 peripherals allow a wide variety of system-level USB capabilities. Many of the OMAP5912 pins can be used for USB-related signals or for signals from other OMAP5912 peripherals. The OMAP5912 top-level pin multiplexing

controls each pin individually to select one of several possible internal pin signal interconnections. When these shared pins are programmed for use as USB signals, the OMAP5912 USB signal multiplexing selects how the signals associated with the three OMAP5912 USB host ports and the OMAP5912 USB function controller can be brought out to OMAP5912 pins.

OMAP5912 Multimedia Processor Multi-channel Buffered Serial Ports (McBSPs) Reference Guide (literature number SPRU762) describes the three multi-channel buffered serial ports (McBSPs) available on the OMAP5912 device. The OMAP5912 device provides multiple high-speed multichannel buffered serial ports (McBSPs) that allow direct interface to codecs and other devices in a system.

OMAP5912 Multimedia Processor Camera Interface Reference Guide (literature number SPRU763) describes two camera inerfaces implemented in the OMAP5912 multimedia processor: compact serial camera port and camera parallel interface.

OMAP5912 Multimedia Processor Display Interface Reference Guide (literature number SPRU764) describes the display interface of the OMAP5912 multimedia processor.

- LCD module
- LCD data conversion module
- LED pulse generator
- Display interface

OMAP5912 Multimedia Processor Multimedia Card (MMC/SD/SDIO) (literature number SPRU765) describes the multimedia card (MMC) interface of the OMAP5912 multimedia processor. The multimedia card/secure data/secure digital IO (MMC/SD/SDIO) host controller provides an interface between a local host, such as a microprocessor unit (MPU) or digital signal processor (DSP), and either an MMC or SD memory card, plus up to four serial flash cards. The host controller handles MMC/SD/SDIO or serial port interface (SPI) transactions with minimal local host intervention.

OMAP5912 Multimedia Processor Keyboard Interface Reference Guide (literature number SPRU766) describes the keyboard interface of the OMAP5912 multimedia processor. The MPUIO module enables direct I/O communication between the MPU (through the public TIPB) and external devices. Two types of I/O can be used: specific I/Os dedicated for 8 x 8 keyboard connection, and general-purpose I/Os.

OMAP5912 Multimedia Processor General-Purpose Interface Reference Guide (literature number SPRU767) describes the general-purpose in-

terface of the OMAP5912 multimedia processor. There are four GPIO modules in the OMAP5912. Each GPIO peripheral controls 16 dedicated pins configurable either as input or output for general purposes. Each pin has an independent control direction set by a programmable register. The two−edge control registers configure events (rising edge, falling edge, or both edges) on an input pin to trigger interrupts or wake−up requests (depending on the system mode). In addition, an interrupt mask register masks out specified pins. Finally, the GPIO peripherals provide the set and clear capabilities on the data output registers and the interrupt mask registers. After detection, all event sources are merged and a single synchronous interrupt (per module) is generated in active mode, whereas a unique wake−up line is issued in idle mode. Eight data output lines of the GPIO3 are ORed together to generate a global output line at the OMAP5912 boundary. This global output line can be used in conjunction with the SSI to provide a CMT−APE interface to the OMAP5912.

OMAP5912 Multimedia Processor VLYNQ Serial Communications Interface Reference Guide (literature number SPRU768) describes the VLYNQ of the OMAP5912 multimedia processor.

VLYNQ is a serial communications interface that enables the extension of an internal bus segment to one or more external physical devices. The external devices are mapped into local, physical address space and appear as if they are on the internal bus of the OMAP 5912. The external devices must also have a VLYNQ interface. The VLYNQ module serializes bus transactions in one device, transfers the serialized data between devices via a VLYNQ port, and de-serializes the transaction in the external device.

OMAP5912 includes one VLYNQ module connected on OCPT2 target port and OCPI initiator port. These connections are configured via a static switch, which selects either SSI or VLYNQ module. This switch, forbids the simultaneous use of GDD/SSI and VLYNQ. The switch is controlled by the VLYNQ_EN bit in the OMAP5912 configuration control register (CONF_5912_CTRL).

OMAP5912 Multimedia Processor Pinout Reference Guide (literature number SPRU769) provides the pinout of the OMAP5912 multimedia processor. After power-up reset, the user can change the configuration of the default interfaces. If another interface is available on top of the default, it is possible to enable a new interface for each ball by setting the corresponding 3-bit field of the associated FUNC_MUX_CTRL register. It is also possible to configure on-chip pullup/pulldown. This document

also describes the various power domains so that the user can apply the different interfaces seamlessly with external components.

OMAP5912 Multimedia Processor Window Tracer (WT) Reference Guide (literature number SPRU770) describes the window tracer module used to capture the memory transactions from four interfaces: EMIFF, EMIFS, OCP-T1, and OCP-T2. This module is located in the OMAP3.2 traffic controller (TC).

OMAP5912 Multimedia Processor Real-Time Clock Reference Guide (literature number SPRUxxx) describes the real-time clock of the OMAP5912 multimedia processor. The real-time clock (RTC) block is an embedded real-time clock module directly accessible from the TIPB bus interface.

Trademarks

Contents

Figures

Tables

Notes

Multimedia Card (MMC/SD/SDIO) Interface

This document describes the multimedia card (MMC) interface of the OMAP5912 multimedia processor.

1 MMC Overview

The multimedia card/secure data/secure digital IO (MMC/SD/SDIO) host controller provides an interface between a local host, such as a microprocessor unit (MPU) or digital signal processor (DSP), and either an MMC or SD memory card, plus up to four serial flash cards. The host controller handles MMC/SD/SDIO or serial port interface (SPI) transactions with minimal local host intervention. Figure 2 provides an overview of the system.

The host controller supports the following combination of external devices:

- One or more MMC memory cards sharing the same bus, plus up to four devices with 8-bit SPI protocol interface (serial flash memories)
- One SD memory card or SDIO card, plus up to four devices with 8-bit SPI protocol interface

Other combinations, such as two SD cards or one MMC card plus one SD card, are not supported through a single controller.

The application interface manages transaction semantics. The MMC/SD/SDIO host controller handles the MMC/SD protocol at the transmission level, including data packing, adding the cyclic redundancy check (CRC) and start/end bits, and checking for syntactical correctness. It also supports SD mode wide-bus width.

The application interface can send every MMC/SD/SDIO command and either poll for the status of the adapter, wait for an interrupt request, which is sent back in case of exceptions, or warn of the end of the operation. The application interface reads card responses and flag registers, and masks interrupt sources individually. These operations are performed by reading and writing control registers. The MMC/SD/SDIO module also supports two direct memory access (DMA) channels.

Figure 1 shows the OMAP5912 processor with the MMC area highlighted.

Figure 1. OMAP5912 MMC Area

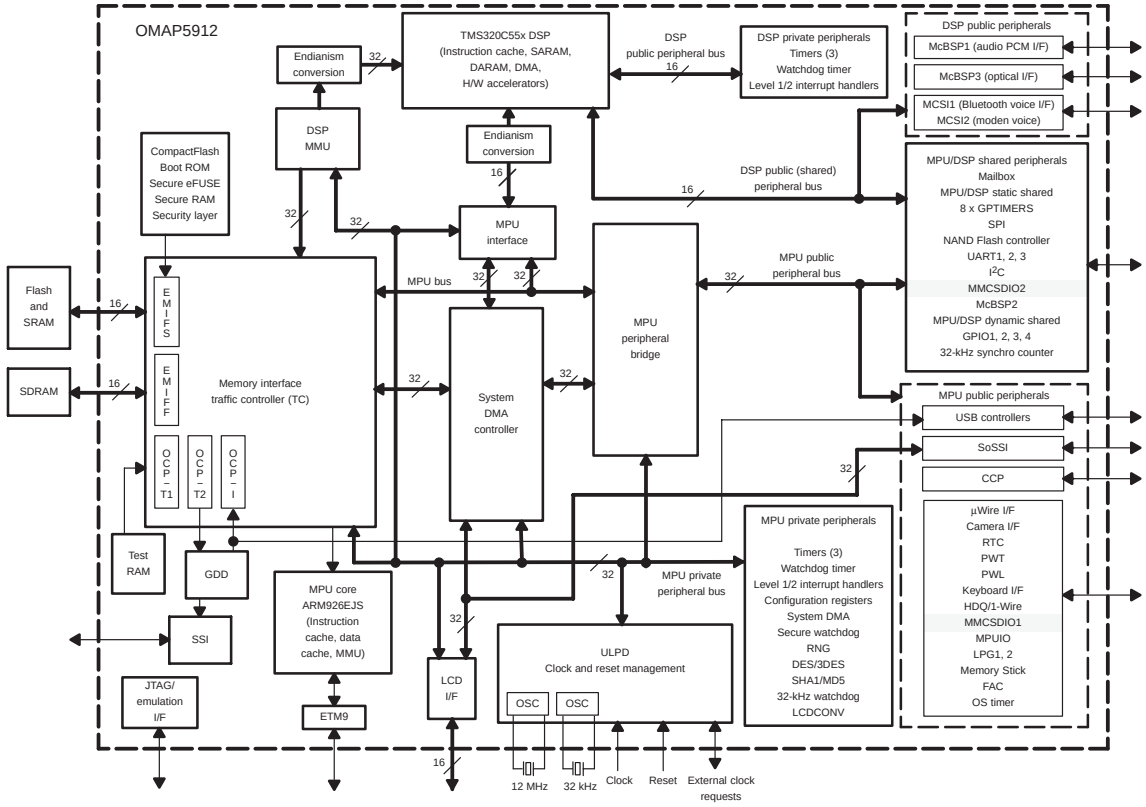
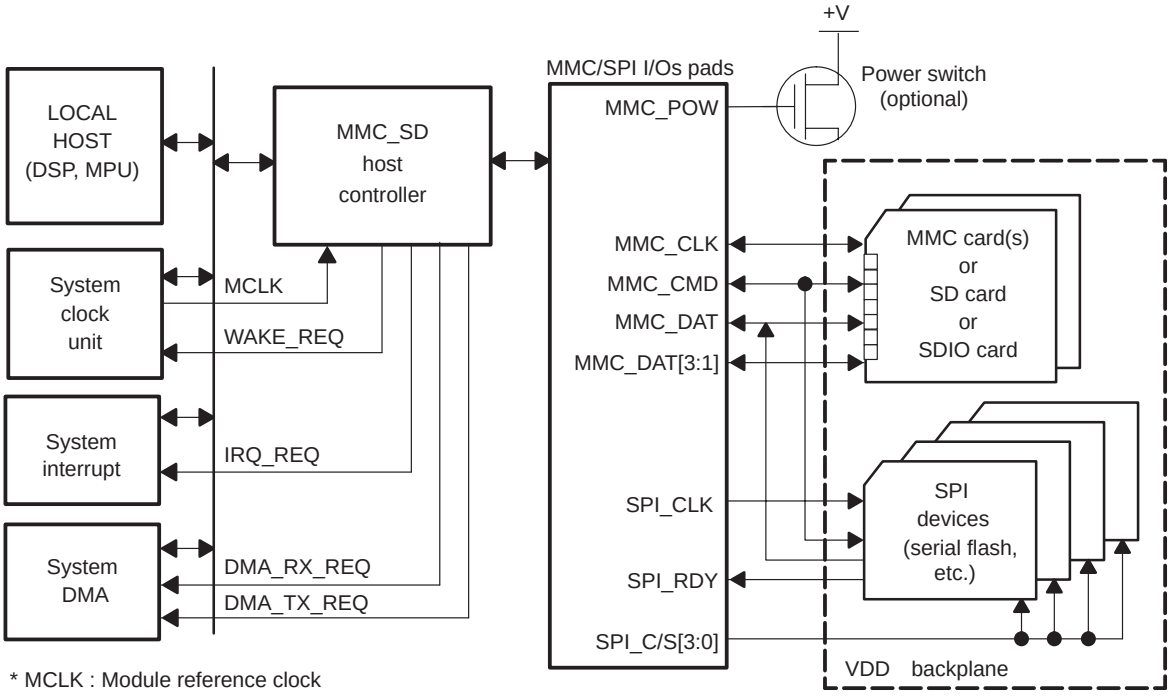


Figure 2. MMC/SD/SDIO System Overview



1.1 MMC/SD/SDIO Host Controller Features

The main features of the controller are:

- ❏ Full compliance with MMC command/response sets as defined in *The Multimedia Card-System Specification*, MMCA Technical Committee, Version 3.1, June 2001
- ❏ Full compliance with SD command/response sets as defined in *SD Memory Card Specification−Part 1, Physical Layer Specification*, SD Group, Version 1.0, March 2000, and *Supplementary Notes−Part 1, Physical Layer Specification*, SD Group, June 2000
- ❏ Full compliance with SDIO command/response sets and interrupt/read-wait mode as defined in *SDIO Card Specification Part E1*, SDIO Working Group, Version 1.0, October 2001
- ❏ Flexible architecture that allows support for new command structure
- ❏ Separate SPI interface with four CS. Provides support for up to four serial flash devices
- ❏ Built-in 64-byte FIFO for buffered read or write

- 16-bit-wide access bus to maximize bus throughput
- Low-power design
- Wide interrupt capability
- Programmable clock generation
- Two DMA channels
- Big- /little-endian mode for data

Known limitations:

- No built-in hardware support for error correction codes (ECC)

1.2 MMC/SD Host Controller Signal Pads

The signal pads listed in Table 1 describe the physical interface between the driving IC (the transceiver) and the target MMC/SD memory cards, SDIO device, or serial flash memories.

The transceiver provides a dc-level adaptation function between the controller core and the target devices. It can be integrated either on-chip with the controller or implemented off-chip (system-dependent issue).

Table 1. Signal Pads

Name	Type	Pull-Up	Reset Value	Description
MMC.CLK	Out	–	0	MMC/SD/SDIO card CLK signal. Only active during active command to MMC/SD/SDIO card using MMC or SPI protocols.
MMC.CMD/SPI.DO (SPI_SO)	In/out	Yes	Input	MMC/SD card CMD signal in MMC/SD mode. SPI serial out signal in SPI modes (output—goes to serial in of target device(s)).
MMC.DAT0 (SPI_SI)	In/out	Yes	Input	MMC card DAT signal or SD/SDIO card DAT[0] signal in MMC/SD mode. SPI serial in signal in SPI modes (input—comes from serial out of target device(s)).
MMC.DAT1 (SDIO_IRQ)†	In/out	Yes	Input	SD/SDIO card DAT[1] signal. Interrupt (IRQ) for SDIO card (SD and SPI protocol).
MMC.DAT2 (SDIO_RW)†	In/out	Yes	Input	SD/SDIO card DAT[2] signal. Read wait (RW) for SDIO card.
MMC.DAT3 (MMC_CS, SD_CD)¶	In/out	Yes	Input	SD/SDIO card DAT[3] signal. Chip-select (CS) for MMC/SD/SDIO cards using SPI protocol. Chip detect (CD) for SD/SDIO cards.
SPI.CLK‡	Out	–	0	SPI CLK signal. Only active in SPI mode during active SPI transfers, except when MMC_CLK is selected.
SPI_C/Sn[3:0]‡	Out	–	b1111	Four SPI chip-select signals. Active low. Only active in SPI mode during active SPI transfers
SPI.RDY‡	In	Yes	Input	SPI ready/busy signal. When low, denotes a busy condition. Only active in SPI mode during active SPI transfers.
MMC_POW§	Out	–	0	MMC/SD cards on/off power supply control. When high, denotes power-on condition.

† Optional signals. Only needed for SD/SDIO cards.
‡ Optional signals. Only needed for devices with SPI interfaces (serial flash, additional MMC, SD, or SDIO cards).
§ Optional signal. Only needed if power supply (VDD) of cards or other SPI devices are to be switched on and off in the application.
¶ Optional signals. Only needed for SD/SDIO cards or for MMC card operated in SPI mode.

1.3 MMC.CLK, SPI.CLK Signals ac Characteristics

The core internally gates the MMC or SPI clock signals to be active only during a valid transaction to the selected target device (memory cards or serial flash). The duty cycle of the clock depends on the clock division factor and the polarity setting.

Figure 3. MMC.CLK/SPI.CLK Signals ac Characteristics

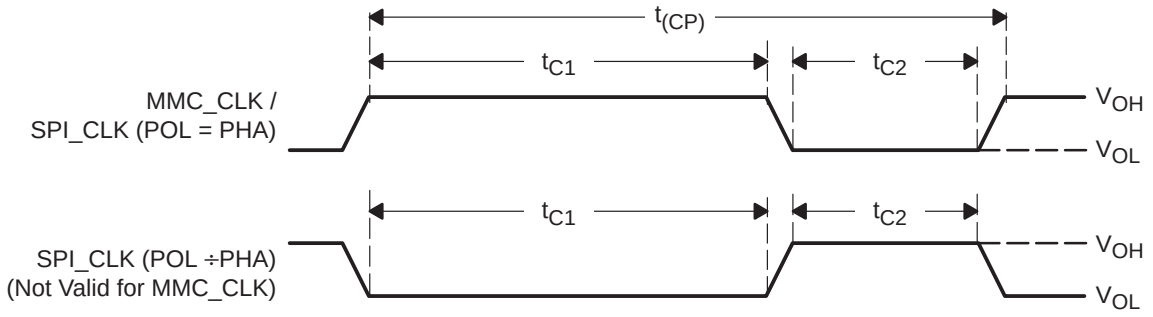


Table 2. MMC.CLK/SPI.CLK Signals ac Parameters

Parameter	Description	Min	Max	Unit
t_{CP}	Clock period	40	–	ns
t_{C1}	MMC mode: Clock high time	–	–	ns
	SPI mode: Clock high time (POL = PHA), clock low time (POL ≠ PHA)			
t_{C2}	MMC mode: Clock low time	–	–	ns
	SPI mode: Clock low time (POL = PHA), clock high time (POL ≠ PHA)			

The clock period and the high and low times specified in Table 2, as well as all timings in subsequent pages, are controller capabilities.

The real clock period must be computed as a function of the system reference clock and adjusted to the target device used in the application. All derived timings must be checked against selected target device specifications.

For example:

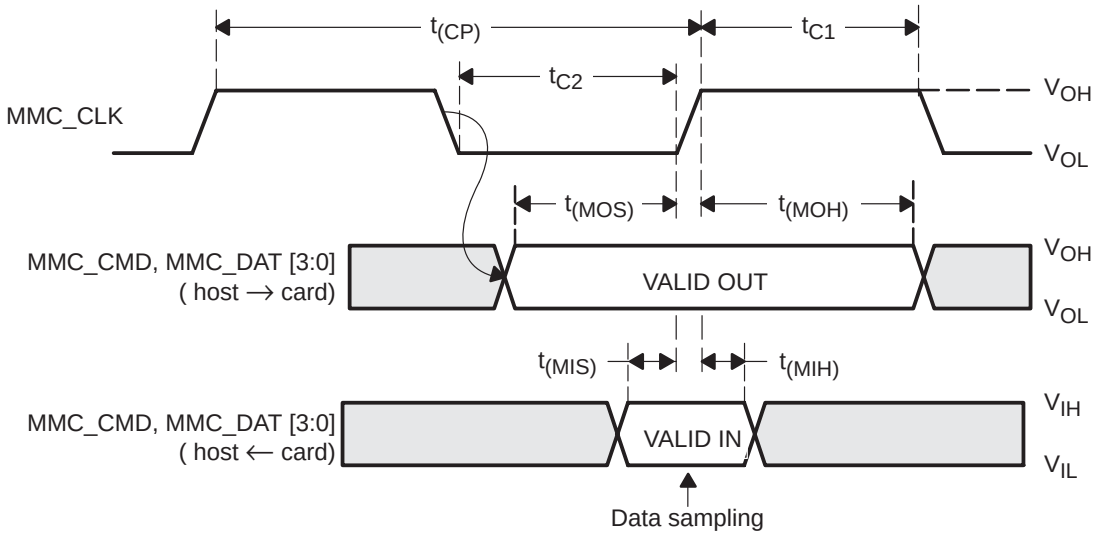
- ❏ MMC card: max 20 MHz (min 50 ns)
- ❏ SD card: max 25 MHz (min 40 ns)
- ❏ SPI serial flash: max 13 MHz (min 77 ns)

1.4 MMC/SD/SDIO Modes—Interface Signal ac Characteristics

Figure 4 depicts the ac characteristics of the interface signals when the interface is configured for MMC/SD/SDIO operation.

SPI-specific output signals (SPI_C/Sn[3:0], SPI.CLK) are held to their inactive state, and SPI-specific input signals are don't care (SPI.RDY).

Figure 4. MMC/SD/SDIO ac Characteristics



Data is sampled on the rising edge of the clock.

Table 3. MMC/SD/SDIO ac Parameters

Parameter	Description	Min	Max	Unit
t_{MOS}	Data output setup to rising edge of clock	t_{C2}–5	–	ns
t_{MOH}	Data output hold to rising edge of clock	t_{C1}–5	–	ns
t_{MIS}	Data input setup to rising edge of clock	4	–	ns
t_{MIH}	Data input hold to rising edge of clock	4	–	ns

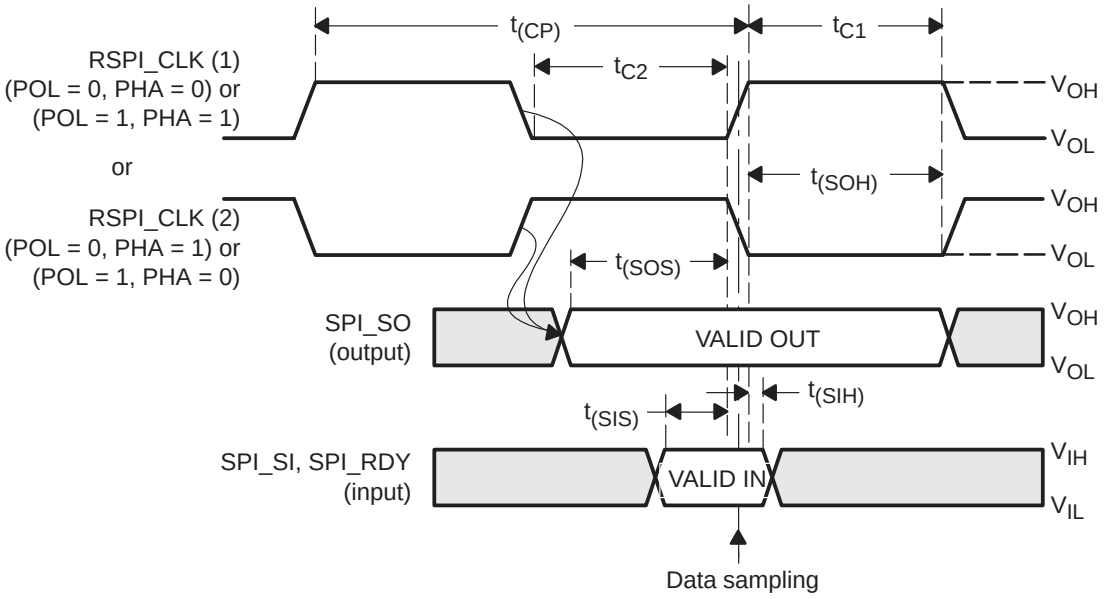
1.5 SPI Mode—Interface Signal ac Characteristics

Figure 5 depicts the ac characteristics of the interface signals when the interface is configured for SPI operation.

MMC-specific input/output signals (MMC.DAT3) are held to their inactive state.

The SPI interface is master only.

Figure 5. SPI ac Characteristics



Data is sampled on the rising or falling edge of the clock, depending on the polarity/phase setting. See Table 4.

Table 4. SPI ac Parameters

Parameter	Description	Min	Max	Unit
t_{SOS}	Data output set up to rising edge of clock (1) or falling edge of clock (2)	t_{C2}−5	−	ns
t_{SOH}	Data output hold to rising edge of clock (1) or falling edge of clock (2)	t_{C1}−5	−	ns
t_{SIS}	Data input set up to rising edge of clock (1) or falling edge of clock (2)	16	−	ns
t_{SIH}	Data input hold to rising edge of clock (1) or falling edge of clock (2)	0	−	ns

2 MMC Registers

Table 5 lists the MMC registers. Table 6 through Table 36 describe the registers bits.

Table 5. MMC Registers

Base Address = FFFB 7800 and 0xFFFB 7C00			
Register	**Description**	**R/W**	**Offset**
MMC.CMD/SPI.DO	MMC command		0x00
MMC_ARGL	MMC argument low		0x04
MMC_ARGH	MMC argument high		0x08
MMC_CON	MMC module configuration		0x0C
MMC_STAT	MMC module status		0x10
MMC_IE	MMC system interrupt enable		0x14
MMC_CTO	MMC command time-out		0x18
MMC_DTO	MMC data read time-out		0x1C
MMC_DATA	MMC data access		0x20
MMC_BLEN	MMC block length		0x24
MMC_NBLK	MMC number of blocks		0x28
MMC_BUF	MMC buffer configuration		0x2C
MMC_SPI	MMC SPI configuration		0x30
MMC_SDIO	MMC SDIO mode configuration		0x34
MMC_SYST	MMC system test		0x38
MMC_REV	MMC module revision		0x3C
MMC_RSP0	MMC/SD command response 0		0x40
MMC_RSP1	MMC/SD command response 1		0x44
MMC_RSP2	MMC/SD command response 2		0x48
MMC_RSP3	MMC/SD command response 3		0x4C
MMC_RSP4	MMC/SD command response 4		0x50
MMC_RSP5	MMC/SD command response 5		0x54
MMC_RSP6	MMC/SD command response 6		0x58
MMC_RSP7	MMC/SD command response 7		0x5C
MMC_IOSR	MMC SDIO suspend/resume control		0x60
MMC_SYSC	MMC system control		0x64
MMC_SYSS	MMC system status		0x68
Reserved	Reserved		0x6C−7C

Table 6. MMC Command Register (MMC_CMD)

Base Address = 0xFFFB 7800 and 0xFFFB 7C00, Offset Address = 0x00		
Bit	**Name**	**Description**
15	DDIR	Data direction [write, read]
14	SHR	Stream command or broadcast host response [normal, stream/host]
13:12	TYPE	Command type [bc, bcr, ac, adtc]
11	BUSY	Command with busy response
10:8	RSP	Command responses [no response, R1/R1b, R2, R3, R4, R5, R6, reserved]
7	INAB	Send initialization stream/data abort command
6	ODTO	Card open drain mode/extended command time-out mode
5:0	INDX	Command index [CMD0, …, CMD63]

A write to this register sends a command to the card.

If the local host accesses this register byte-wise, the card receives the command only after a write access to the least significant (LS) byte (bits 7:0). Therefore, the most significant (MS) byte must always be written first in a byte-accessed situation.

A read has no effect except to return the last command that was sent.

Note: Writes

A write into this register with MMC_CMD[TYPE] = adtc resets the FIFO. Writes with other type of values (bc, bcr, ac) do not affect the FIFO contents. Hence, data must be written into the FIFO after sending a single or multiple block-write command.

MMC.CMD/SPI.DO[15] Data Direction (DDIR)

This bit specifies whether the data transfer is a read or a write, and is valid only if the command type is adtc.

This bit has the same polarity as the RD/WR argument bit 0 for a GEN_CMD command (CMD56):

- 0: Data write
- 1: Data read

Value after reset is low.

MMC.CMD/SPI.DO[14] Stream Command or Broadcast Host Response (SHR)

MMC card only.

The SD card does not support stream operation or host-generated response. This bit must be set to 1 in two cases:

- Associated with adtc command type, if the command is a stream data transfer (read or write). Stream read is a class 1 command (CMD11: READ_DAT_UNTIL_STOP). Stream write is a class 3 command (CMD20: WRITE_DAT_UNTIL_STOP).
- Associated with a bc command type, the host generates a 48-bit response instead of a command. It can be used to terminate the interrupt mode by generating a CMD40 response by the core (see Section 4.3, *Interrupt Mode* in *The Multimedia Card-System Specification*). In order for the host response to be generated in open drain mode, MMC_CMD[ODTO] must be set to 1.

This bit is valid only if the command type is adtc or bc:

- 0: Normal mode
- 1: Stream mode (MMC_CMD[TYPE] = adtc), host response (MMC_CMD[TYPE] = bc)

Value after reset is low.

MMC.CMD/SPI.DO[13:12] Command Type (TYPE)

Encoded bits that define the type of command passed by the core to the MMC/SD memory card (see Section 4.7.1, *Command Types*, *The Multimedia Card−System Specification*, or *SD Memory Card Specification−Part 1 Physical*
Layer Specification and *Supplementary Notes−Part 1 Physical Layer Specification).*

- 00: Broadcast commands (bc), no response
- 01: Broadcast commands with response (bcr)
- 10: Addressed commands (ac), no data transfer
- 11: Addressed data transfer commands (adtc) and reset of the FIFO

Value after reset is low (both bits).

MMC.CMD/SPI.DO[11] Command With Busy Response (BUSY)

This bit must be set to 1 if the response to the command sent is type R1b (R1 + busy):

- 0: Response without busy (R1, R2, R3, R4, R5, R6)
- 1: Response with busy (R1b)

Value after reset is low.

MMC.CMD/SPI.DO[10:8] Command Response (RSP)

Encoded bits that define the response for the command that the core passes to the MMC/SD memory card (see Section 4.9, *Responses* in *The Multimedia Card-System Specification*, or *SD Memory Card Specifications−Part 1 Physical Layer Specification* and *Supplementary Notes−Part 1 Physical Layer Specification*).

- 000: No response
- 001: R1/R1b (normal response command)
- 010: R2 (CID, CSD registers)
- 011: R3 (OCR register)
- 100: R4 (fast I/O—MMC card only)
- 101: R5 (interrupt request—MMC card only/IO_RW_DIRECT—SDIO card only)
- 110: R6 (published RCA response—SD card only)
- 111: Reserved

Value after reset is low (3 bits).

MMC.CMD/SPI.DO[7] Send Initialization Stream/Data Abort Command (INAB)

This bit must only be set in two particular cases:

- When the card is idle, to send an initialization sequence. This option simplifies the acquisition of new cards. An initialization sequence consists of setting the CMD line to 1 during 80 clock cycles (see Section 6.3, *Power-Up* in *The Multimedia Card-System Specification*, or Section 6.4, *SD Memory Card Specifications−Part 1 Physical Layer Specification* and

Supplementary Notes−Part 1 Physical Layer Specification SD Memory Card Specifications−Part 1 Physical Layer Specification and *Supplementary Notes−Part 1 Physical Layer Specification*). In this mode, no command is sent to the card and no response is expected.

- When the card is in the data transfer stage, to stop or abort an ongoing data transfer. The card is said to be in such state when the previous command was of type adtc and has not yet completed (MMC_STAT[BRS] = 0). A stop or aborted data command:
  - Freezes the MMC_BLEN[BLEN] and MMC_NBLK[NBLK] values according to the last valid byte written to or read from the card
  - Sets the MMC_STAT[BRS] status bit as follows:
    - 0: No action
    - 1: Initialization (80 clock cycles)/data abort command

Value after reset is low.

MMC.CMD/SPI.DO[6] Card Open Drain Mode/Extended Command Time-Out (ODTO)

This bit has a dual function, depending upon the value set in the MMC_SDIO[XDTS] bit.

- Open drain control function (MMC_SDIO[XDTS] = 0—MMC card only

This bit must be set to 1 if the MMC card bus is operating in open-drain mode during the response phase to the command sent. Typically, it occurs during card identification mode when the card is in idle, ready, or ident state. It is also necessary to set this bit to 1 for a broadcast host response (see MMC.CMD[SHR]). This bit must be set for MMC card commands 1, 2, 3, and 40.

For the SD card, this bit must always be kept low because SD cards do not have open drain capability.

- 0: Push-pull
- 1: Open drain

- Extended command time-out function (MMC_SDIO[XDTS] = 1)—SDIO card only.

This bit must be set to 1 if the SDIO command response requires a long time-out (typically an IO_RW_DIRECT CMD52). When set, the command time-out is set to the data time-out value (see MMC_DTO[DTO]). When clear, the normal command time-out applies (see MMC_CTO[CTO]).

- 0: Command time-out equals CTO
- 1: Command time-out equals DTO

Value after reset is low.

MMC.CMD/SPI.DO[5:0] Command Index (INDX)

Binary encoded value from 0 to 63 specifying the command number sent to the card.

- 000000: CMD0
- 000001: CMD1

  …
- 111111: CMD63

Value after reset is low (all 6 bits).

Table 7. System Argument Low Register (MMC_ARGL)

Base Address = 0xFFFB 7800 and 0xFFFB 7C00, Offset Address = 0x04		
Bit	**Name**	**Description**
15:0	ARGL	Command argument bits [15:0]

Value after reset is low (all 16 bits).

Table 8. System Argument High Register (MMC_ARGH)

Base Address = 0xFFFB 7800 and 0xFFFB 7C00, Offset Address = 0x08		
Bit	**Name**	**Description**
15:0	ARGH	Command argument bits [31:16]

Value after reset is low (all 16 bits).

Table 9 lists the module configuration characteristics.

These two 16-bit registers (Table 7 and Table 8) specify the 32-bit argument value that is passed with the command. These registers must be initialized before sending the command to the card (write action into the MMC_CMD register). The only exception making a write unnecessary is a command index specifying stuff bits in arguments.

This section describes the module configuration register.

Table 9. Module Configuration Register (MMC_CON)

Base Address = 0xFFFB 7800 and 0xFFFB 7C00, Offset Address = 0x0C		
Bit	**Name**	**Description**
15	DW	Data bus width
14	–	Reserved
13:12	MODE	Operating mode select [MMC/SD, SPI, SYSTEST]
11	POWER_UP	Power-up control
10	BE	Big-endian mode [little, big]
9:0	CLKD	Clock divider [disabled, 1:1023]

Note: Active Transfer Phase

This register must never be written during an active transfer phase. Changing it may result in unpredictable behavior.

MMC_CON[15] Bus Width During Data Phase (DW)

SD mode only.

This bit must be set following a valid SET_BUS_WIDTH command (ACMD6) with the value written in bit 1 of the argument. Prior to this command, the SD card configuration register (SCR) must be verified for the bus width supported by the SD card.

- 0: 1-bit data width (DAT[0] used)
- 1: 4-bit data width (DAT[3:0] used, SD card only)

Value after reset is low.

This bit must always be set to 0 for MMC cards or during SPI transfer. Failing to set this bit correctly results in unpredictable behavior.

MMC_CON[13:12] Mode Select (MODE)

These two bits select among MMC/SD, SPI, and SYSTEST modes.

- In MMC/SD mode, transfers to the MMC/SD/SDIO card follow the MMC protocol. The MMC clock is enabled and the SPI clock is disabled. MMC/SD transfers are operated under the control of the MMC.CMD register.

- In SPI mode, transfers to as many as four SPI controlled devices (serial flash, MMC/SD/SDIO cards) are supported. SPI transfers are operated under the control of the MMC_SPI register.
- In SYSTEST mode, the signal pins are configured as general-purpose input/output, and the 64-byte FIFO is configured as a stack memory accessible only by the local host or system DMA. The pins retain their default type (input, output, or in-out). The YSTEST mode is operated under the control of the MMC_SYST register.
  - 00: MMC/SD mode (MMC/SD/SDIO cards using MMC/SD protocol)
  - 01: SPI mode (for serial flash or others SPI slave devices)
  - 10: SYSTEST mode
  - 11: Reserved

Value after reset is low (both bits).

MMC_CON[11] Power-Up Control (POW)

This bit must be set to 1 before any valid transaction to either MMC/SD or SPI memory cards.

When 1, the card is considered powered-up and the controller core is enabled.

When 0, the card is considered powered-down (system dependent), and the controller core logic is in pseudo-reset state. This is, the MMC_STAT flags and the FIFO pointers are reset, any access to MMC_DATA[DATA] has no effect, a write into the MMC.CMD register is ignored, and a setting of MMC_SPI[STR] to 1 is ignored.

This bit directly controls the MMC_POW signal (if implemented as device pin).

- 0: Powered-down/pseudo-reset state.
- 1: Powered-up/normal operation mode.

Value after reset is low.

MMC_CON[10] Big Endian (BE)

When this bit is 0 (default), the FIFO is accessed in little endian format. In transmit mode, the LS byte (MMC_DATA[7:0]) is transmitted first, and the MS byte (MMC_DATA[15:8]) is transmitted to the card in second position. Conversely, in receive mode, the first or odd byte received (1, 3, 5, …) is stored in the LS byte position, and the second or even byte received is stored in the MS byte position.

When the LH sets this bit to a 1, the FIFO is accessed in big endian format. In transmit mode, the MS byte (MMC_DATA[15:8]) is transmitted first and the LS byte (MMC_DATA[7:0]) is transmitted to the card in second position. Conversely, in receive mode, the first or odd byte received (1, 3, 5, …) is stored in the MS byte position, and the second or even byte received is stored in the LS byte position.

- 0: Little-endian mode
- 1: Big-endian mode

Value after reset is low.

MMC_CON[9:0] Clock Divider (CLKD)

These bits define the ratio between a reference clock frequency (system dependent) and the output clock frequency on the CLK pin of either the memory card (MMC or SD) or other 8-bit mode SPI-controlled device.

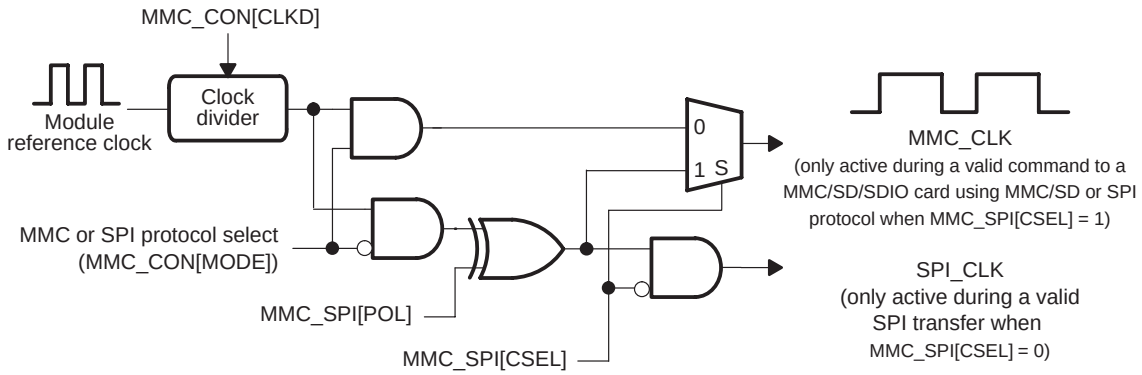
The division factor is equal to the binary encoded decimal value for values between 1 and 1023. A value of 0 disables the clock.

- 0x00: Clock disabled
- 0x01: Ref clock/1

  ....
- 0x3FF: Ref clock/1023

Value after reset is low (all 10 bits).

Figure 6. Clock Control



Notes: 1) During the identification phase, the maximum frequency on the MMC CLK line is 400 kHz (see Section 6.7, *Bus Timing* in *The Multimedia Card-System Specification*, or Section 6.8, *SD Memory Card Specifications−Part 1, Physical Layer Specification* or *Supplementary Notes−Part 1, Physical Layer Specification*). A value of 50 must be set into the frequency ratio register if the reference clock frequency to the MMC/SD host controller is 20 MHz.

2) During the data transfer phase, the maximum frequency is 20 MHz for MMC card and 25 MHz for SD cards.

3) The duty cycle of the generated MMC.CLK and SPI.CLK signals depends on the clock divider value (MMC_CON[CLKD]) and on the polarity setting (MMC_SPI[POL]) in SPI mode only. The low and high times approximate values can be computed using rules set in Table 10.

4) In MMC/SD mode, the idle value of MMC.CLK signal is low. In SPI mode, the idle value of either the MMC.CLK (MMC_SPI[CSEL] = 1) or the SPI.CLK (MMC_SPI[CSEL] = 0) is a function of the polarity setting (low if MMC_SPI[POL] = 0, high if MMC_SPI[POL] = 1).

Table 10. MMC.CLK/SPI.CLK High/Low Time Computation

MMC_CON[CLKD]	MMC.CLK/SPI.CLK High Time	MMC.CLK/SPI.CLK Low Time
1	REF_CLK_HIGH_TIME	REF_CLK_LOW_TIME
Even ≥ 2	REF_CLK_PER (CLKD/2)	REF_CLK_PER (CLKD/2)
Odd ≥ 3 (POL = PHA)	REF_CLK_PER (TRUNC[CLKD/2] + 1)	REF_CLK_PER (TRUNC[CLKD/2])
Odd ≥ 3 (POL ≠ PHA)	REF_CLK_PER (TRUNC[CLKD/2])	REF_CLK_PER (TRUNC[CLKD/2] + 1)

Notes:

1) REF_CLK_PER is the reference clock period (in ns) to the module (end-system dependent).

2) TRUNC is the truncate function to an integer number (round down).

Example 1:

Module reference clock = 48 MHz (20.83 ns); target is the MMC card.

1) a) (MMC_CON[CLKD] = 3 (because the MMC card is 20 MHz max)

2) b) MMC_CLK period = 62.5 ns (> 50 ns OK)

3) c) Ideal MMC_CLK high time = 41.66 ns (>>10 ns OK)

4) d) Ideal MMC_CLK low time = 20.83 ns (>>10 ns OK)

Example 2:

Module reference clock = 60 MHz (16.67 ns); target is the 13-MHz serial flash requiring polarity 1 programming.

1) a) (MMC_CON[CLKD] = 5 (because 13-MHz serial flash)
2) b) SPI_CLK period = 83.33 ns (> 77 ns OK)
3) c) Ideal SPI_CLK high time = 33.3 ns (<35 ns FAIL, use (MMC_CON[CLKD] = 6)
4) d) Ideal SPI_CLK low time = 50 ns (>>35 ns OK)
5) e) (MMC_CON[CLKD] = 6 (because clock high time min 35 ns)
6) f) SPI_CLK period = 100 ns (> 77 ns OK)
7) g) Ideal SPI_CLK high time = 50 ns (>>35 ns OK)
8) h) Ideal SPI_CLK low time = 50 ns (>>35 ns OK)

Table 11. Module Status Register (MMC_STAT)

Base Address = 0xFFFB 7800 and 0xFFFB 7C00, Offset Address = 0x10		
Bit	**Name**	**Description**
15	−	Reserved
14	CERR	Card status error in response
13	CIRQ	MMC card IRQ received (following CMD40) or SDIO card interrupt
12	OCRB	Operation condition register (OCR) busy (following CMD1 or ACMD41)
11	AE	Buffer almost empty
10	AF	Buffer almost full
9	CRW	Card read wait
8	CCRC	Command CRC error
7	CTO	Command response time-out (no response)
6	DCRC	Data CRC error
5	DTO	Data response time-out (no response)
4	EOFB	Card exit busy state
3	BRS	Block received/sent
2	CB	Card enter busy state
1	CD	Card detect on DAT3
0	EOC	End of command phase

The following is common to all bits:

- ❑ The local host can clear a set bit location only by writing a 1 into that bit location. Writing 0 has no effect.
- ❑ When the core sets a bit location to 1, the local host receives an interrupt signal if the interrupt was enabled.

MMC_STAT[14] Card Status Error (CERR)

Table 12. Card Status Error (CERR)

Response Type	Card Status Bits With Error	Response Register Significant Bits	Comments
R1 (MMC, SD, SDIO)	31−26, 24−16,	MMC_RSP7[15−10, 8−0]†	Bit 4 if MMC_SDIO[C14E] = 1 (SDIO)
	4[2] 3[2], 2[2] (opt)	MMC_RSP6[4, 3, 2]‡	Bit 3 if MMC_SDIO[C13E] = 1 (SD/app spec)
			Bit 2 if MMC_SDIO[C12E] = 1 (app spec)
R6 (SD, SDIO)	15−13, 3	MMC_RSP6[15−13, 3]	These correspond to 23, 22, 19, 3 card status errors (SDIO card does not generate error on bit 3 force 0—superset)
R5 (SDIO)	7, 6, 3, 1, 0[3]	MMC_RSP6[15,14, 11, 9, 8]	[3] Only if MMC_SDIO[15] = 1

† These 15 bits can all generate errors (SDIO spec specifies 31, 23−22, 19 while others are 0—superset).
‡ These 3 bits can also generate errors if enabled.

MMC/SD mode only.

The core automatically sets this bit when there is at least one error in a response of type R1, R1b, R6, or R5 if enabled. Only bits referenced as type E (error) set a card status error.

The error handler must parse the response registers to understand the source of the error.

Other responses (type R2, R3, or R4) do not trigger a card status error.

This bit has no meaning and always reads 0 in SPI or SYSTEST modes.

- 0: No action or no error
- 1: Error occurred

Value after reset is low.

MMC_STAT[13] Card IRQ (CIRQ)

MMC card only:

The core automatically sets this bit when a card is in interrupt mode and exits Wait_IRQ state (irq) by asserting a low level on the CMD line (cards are in open-drain mode). Only Class 9 MMC cards can be put into interrupt mode when in standby state using a GO_IRQ_STATE (CMD40) command.

SDIO card only:

The core automatically sets this bit when a SDIO card has signaled an interrupt on DAT1 line and if the MMC_SDIO[IRQE] bit was set to 1. The interrupt condition is detected in either 1-bit or 4-bit transfer mode and for either MMC/SD or SPI operation mode. SD memory cards do not support interrupt mode.

This bit has no meaning and always reads 0 in SPI or SYSTEST mode.

- ❏ 0: No action or idle
- ❏ 1: Card exits IRQ state (MMC card), card interrupt detected (SDIO card).

Value after reset is low.

MMC_STAT[12] OCR Busy (OCRB)

MMC/SD mode only:

The core automatically sets this bit after a SEND_OP_COND (CMD1) or a SD_APP_OP_COND (ACMD1) command when one or more cards have not yet completed power-up. When this bit is set, the CMD1/ACMD1 command must be repeated until the card stops responding with a busy condition (a low value on bit 31 of OCR register indicates a busy condition) (see Section 6.3, *Power-Up*, in *The MultiMediaCard−System Specification* or Section 6.4, *SD Memory Card Specifications−Part 1, Physical Layer Specification* or *Supplementary Notes−Part 1, Physical Layer Specification*).

This bit has no meaning and always reads 0 in SPI or SYSTEST mode.

- ❏ 0: No action or card powered up
- ❏ 1: OCR busy

Value after reset is low.

MMC_STAT[11] Buffer Almost Empty (AE)

The core automatically sets this bit during a write operation to the card (see class 4 block-oriented write command in Section 4.7.3, *Command Classes,* in *The MultiMediaCard−System Specification−Part 1, Physical Layer Specification, SD Group June 2000*) when the level equals or is below the threshold value (in 16-bit words) set in MMC_BUF[AEL]. It indicates that the memory card has emptied the buffer to the specified level and that the local host is able to write more data into the buffer.

If the DMA transmit mode is enabled, this bit is never set. Instead, a DMA TX request is generated to the system's main DMA controller.

Note: DMA TX Request

The almost-empty status bit and DMA TX request are generated under the same conditions. This bit is set initially when a new block write command is sent to the card. Once the bit is set, the core internally masks a new set condition until the local host has performed MMC_BUF[AEL] 16-bit word write access(es) to the FIFO.

- 0: No action or buffer is equal or above almost-empty level
- 1: Buffer almost empty

Value after reset is low.

MMC_STAT[10] Buffer Almost Full (AF)

The core automatically sets this bit during a read operation to the card (see class 2 block-oriented read commands in Section 4.7.3, *Command Classes,* in *The MultiMediaCard−System Specification−Part 1, Physical Layer Specification, SD Group June 2000*) when the level equals or is above the threshold value (in 16-bit words) set in MMC_BUF[AFL]. It indicates that the memory card has filled out the buffer to the specified level and that the local host needs to empty the buffer by reading it.

If the DMA-receive mode is enabled, this bit is never set. Instead, a DMA RX request is generated to the system's main DMA controller.

Note: DMA RX Request

The almost full status bit and DMA RX request are generated under the same conditions. Once set, the core internally masks a new set condition till the local host has performed MMC_BUF[AFL] 16-bit word read access(es) from the FIFO.

- 0: No action or buffer is below or equal almost full level
- 1: Buffer almost full

Value after reset is low.

MMC_STAT[9] Card Read Wait (CRW)

SDIO card only.

The core automatically sets this bit when an SDIO card has entered read wait. It indicates that the previous read multiple transfer (CMD53) has been temporarily stalled and that a new command without data stage (such as CMD52) can be sent to the SDIO card.

This bit is set on the condition that the core has requested a wait to the card (MMC_SDIO[RW] = 1) and the read wait mode is enabled (MMC_SDIO[RWE] = 1). The read wait condition is detected in either 1- or 4-bit transfer mode.

- ❏ 0: No action
- ❏ 1: SDIO card in read wait

Value after reset is low.

MMC_STAT[8] Command CRC Error (CCRC)

MMC/SD mode only.

The core automatically sets this bit if there is a CRC7 error in the command response (bits 7:1 of response). CRC7 is checked for all command response types (R1 through R6) with the exception of type R3, and conditionally for type R4 if MMC_SDIO[DCR4] = 1.

In SPI or SYSTEST modes, this bit has no meaning and always reads 0.

- ❏ 0: No action or no CRC7 error
- ❏ 1: CRC7 error

Value after reset is low.

MMC_STAT[7] Command Time-Out Error (CTO)

MMC/SD mode only.

The core automatically sets this bit if the card does not respond to any command requiring a response within the specified number of command time-out clock cycles set in MMC_CTO[CTO].

Note: Command Time-Out

If this bit is set after a command time-out, clearing this bit automatically stops the MMC clock and forces the controller FSM to its default state.

This bit has no meaning and always reads 0 in SPI or SYSTEST mode.

- ❏ 0: No action or no command time-out
- ❏ 1: Command time-out

Value after reset is low.

MMC_STAT[6] Data CRC Error (DCRC)

MMC/SD mode only.

The core automatically sets this bit if there is a CRC16 error in the data phase response following a block read command (single or multiple), or if a 3-bit CRC status differs from a positive 010 token during a block-write command (single or multiple). A token error can be either a data transmission error 101, or a no CRC response 111 in the case of a programming error (SD card only). Every block of the CRC is checked in a multiple-block transfer.

This bit has no meaning and always reads 0 in SPI or SYSTEST mode.

- ❏ 0: No action or no CRC error
- ❏ 1: CRC16 error (read), 3-bit CRC token error (write)

Value after reset is low.

MMC_STAT[5] Data Time-Out Error (DTO)

The core automatically sets this bit if the card does not respond within the specified number of data time-out clock cycles (DTO) set in MMC_DTO[DTO].

This bit is also set in SPI mode if the RDY/BUSY signal remains asserted in busy condition for MMC_DTO[DTO] consecutive clock cycles.

Note: Data Time-Out

If this bit is set after a data time-out, clearing this bit automatically stops the MMC or SPI clock and forces the controller FSM to its default state.

This bit has no meaning and always reads 0 in SYSTEST mode.

- ❏ 0: No action or no data time-out
- ❏ 1: Data time-out

Value after reset is low.

MMC_STAT[4] Card Exit Busy State (EOFB)

MMC/SD mode only.

The core automatically sets this bit when the addressed card releases the DAT line from its busy state (low level = busy). This bit can only be set during a programming phase (write operation) to an MMC or SD memory card.

This bit has no meaning and always reads 0 in SPI or SYSTEST mode.

- 0: No action
- 1: Data line released/exit busy state

Value after reset is low.

MMC_STAT[3] Block Received/Sent (BRS)

The core automatically sets this bit at the end of a block transfer (read or write).

In MMC or SD mode, this bit is set when the block transfer completes without error. If a CRC error occurs, this bit is not set. Instead, a data CRC error is set to 1. For either multiple block or stream transfer, this bit is set only once after the last successful block transfer (when MMC_NBLK[NBLK] decrements down to 0) or until interrupted by a stop command.

In SPI mode, this bit is set when either the read or write command completes *(MMC_BLEN[BLEN] decrements down to 0).*

The difference between a DMA and a non-DMA receive operation are:

- In non-DMA RX mode, this bit is set after the last byte has been received in the FIFO. At this stage, the FIFO is not empty and must be read by the LH until it is emptied before sending a new command.
- In DMA RX mode, this bit is set after both the last byte has been received and the FIFO is empty.

This bit has no meaning and always reads 0 in SYSTEST mode.

- 0: No action
- 1: Block received/sent

Value after reset is low.

MMC_STAT[2] Card Enter Busy State (CB)

MMC/SD mode only.

The core automatically sets this bit when the addressed card asserts the DAT line to a low level during a programming phase (write operation) to an MMC or SD memory card. For an MMC card only, users can optionally use this interrupt to deselect the card, which continues to program, and select another card.

This bit has no meaning and always reads 0 in SPI or SYSTEST mode.

- ❏ 0: No action
- ❏ 1: Data line asserted low/card busy

Value after reset is low.

MMC_STAT[1] Card Detected on DAT3 (CD)

MMC/SD mode only.

The core automatically sets this bit after it has detected a card-detect condition on the DAT3 line and if MMC_SDIO[CDE] = 1.

This bit has no meaning and always reads 0 in SPI or SYSTEST mode.

- ❏ 0: No action
- ❏ 1: Card-detect event

Value after reset is low.

MMC_STAT[0] End of Command (EOC)

MMC/SD mode only.

The core automatically sets this bit at the end of a successful command/response sequence or at the end of a command without response. This bit is not set in case of a card status error (MMC_STAT[CERR] = 1), command CRC error (MMC_STAT[CCRC] = 1), or command time-out (MMC_STAT[CTO] = 1).

This bit has no meaning and always reads 0 in SPI or SYSTEST mode.

- ❏ 0: No action
- ❏ 1: End of command/response sequence

Value after reset is low.

Table 13 lists the characteristics of the system interrupt enable register.

Table 13. System Interrupt Enable Register (MMC_IE)

Base Address = 0xFFFB 7800 and 0xFFFB 7C00, Offset Address = 0x14		
Bit	**Name**	**Description**
15	-	Reserved
14	CERR	Card status error interrupt enable
13	CIRQ	Card IRQ interrupt enable
12	OCRB	OCR busy interrupt enable
11	AE	Buffer almost empty interrupt enable
10	AF	Buffer almost full interrupt enable
9	CRW	Card read wait enable
8	CCRC	Command CRC error interrupt enable
7	CTO	Command response time-out interrupt enable
6	DCRC	Data CRC error interrupt enable
5	DTO	Data response time-out interrupt enable
4	EOFB	Card exit busy state interrupt enable
3	BRS	Block received/sent interrupt enable
2	CB	Card enter busy state interrupt enable
1	CD	Card-detect interrupt enable
0	EOC	End of command interrupt enable

Common to all bits:

When the local host sets a bit location to 1, it is notified of an interrupt if the core asserted the corresponding bit location in MMC_STAT register to 1.

If set to 0, the interrupt is masked and the local host is not signaled.

- ❏ 0: Interrupt disabled
- ❏ 1: Interrupt enabled

Value after reset is low (all bits).

Table 14. Command Time-Out Register(MMC_CTO)

Base Address = 0xFFFB 7800 and 0xFFFB 7C00, Offset Address = 0x18		
Bit	**Name**	**Description**
15:8	–	Reserved
7:0	CTO	MMC command time-out value

This 16-bit register specifies the maximum number of clock cycles before a command time-out condition occurs.

MMC_CTO[7:0] Command Time-Out Value (CTO)

MMC/SD mode only.

The local host sets this field based on N_{CR} clock cycles. The MMC and SD cards specify N_{CR} to be between 2 and 64 clock cycles.

If the card does not respond within the specified number of cycles, command time-out is set to 1 in the MMC_STAT[CTO] register bit.

For MMC card-interrupt mode support, this time-out is disabled when the command passes with an R5 response (CMD40) if register bit MMC_SDIO[C5E] = 0.

For the SDIO card, this time-out value is not valid for command passes with MMC_CMD[ODTO] = 1 if MMC_SDIO[XDTS] = 1; MMC_DTO[DTO] applies instead.

- 0x00: Command time-out disabled
- 0x01: 1 clock cycle

  …
- 0xFF: 255 clock cycles (2^8–1)

Value after reset is low (all 8 bits).

Table 15 lists the characteristics of the data read time-out register.

Table 15. Data Read Time-Out Register (MMC_DTO)

Base Address = 0xFFFB 7800 and 0xFFFB 7C00, Offset Address = 0x1C		
Bit	**Name**	**Description**
15:0	DTO	Data read time-out

This 16-bit register specifies the maximum number of clock cycles before a data time-out condition occurs.

MMC_DTO[15:0] Data Time-Out Value (DTO)

In MMC/SD mode, the local host must set this field based on N_{AC} clock cycles. N_{AC} is computed from the parameters TAAC and NSAC and the operating clock frequency. TAAC and NSAC are CSD card parameters and are obtained by reading the response register after successful execution of a SEND_CSD command (CMD9).

If the card does not respond within the specified number of cycles, data time-out is set to 1 in the MMC_STAT[DTO] register bit.

The effective number of clock cycles for time-out value are to be multiplied by 1024 if MMC_SDIO[DPE] = 1 and by 1 if MMC_SDIO[DPE] = 0.

For the SDIO card, this time-out value is also valid for command without data stage passes with MMC_CMD[ODTO] = 1 and MMC_SDIO[XDTS] = 1.

In SPI mode, a data time-out condition is also generated if the RDY/BUSY signal is asserted low (BUSY) for DTO consecutive clock cycles.

Table 16. Clock Cycles for Time-out Value

MMC_DTO[DTO]	MMC_SDIO[DPE] = 0	MMC_SDIO[DPE] = 1	Units
0x0000	No time-out	No time-out	
0x0001	1	1024	MMC clock cycles
0x0002	2	2048	“
…	…	…	“
0xFFFF	65535 (2^{16}–1)	67107840 (2^{26}–2^{10})	“

Value after reset is low (all 16 bits).

Table 17. Data Access Register (MMC_DATA)

Base Address = 0xFFFB 7800 and 0xFFFB 7C00, Offset Address = 0x20		
Bit	**Name**	**Description**
15:0	DATA	Transmit/receive FIFO data

This register is the entry point for the local host to read data from, or write data into, the FIFO buffer. The FIFO size is 32x16 bits (64 bytes). Bytes within a word are stored and read in little endian format (MMC_CON[BE] = 0) or big endian format (MMC_CON[BE] = 1).

If the local host accesses this register byte-wise, the MS byte (bits [15:8]) must always be written/read first. A byte access to the LS byte without a prior write into the MS byte results in the MS byte being filled with 0x00.

MMC_DATA[15:0] Transmit/Receive FIFO Data Value (DATA)

In MMC/SD mode, this register contains either the data packet associated with block transfer (read or write), the CID contents for a PROGRAM_CID (CMD26) command, or the CSD contents for a PROGRAM_CSD (CMD27) command.

Because the block length is passed as an argument, the local host can perform only 16-bit access (read or write) into the buffer even if the block length is not an even number. In case of an odd number of bytes to read, the upper byte of the last access always reads as 0x00. Conversely, for an odd number of bytes to write, the upper byte must be filled with 0x00 for the last data value.

In SPI mode, the register contains both the command (opcode and address for a serial flash) and the data.

In SYSTEST mode, the FIFO behaves as a stack accessible only by the local host (push and pop operations). In this mode, the set FIFO threshold values are active, as are the associated interrupts and DMA if enabled. This special mode can be used for system test purposes.

Value after reset is low (all 16 bits).

Note: Read/Write Access

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 when the local host performs a 16-bit access if the buffer contains a single byte. A debugger read access (SUSPEND operation) returns the current read data value but does not update the FIFO pointers or generate an access error.

Figure 7 shows the behavior of the data access to/from the FIFO as a function of the MMC_CON[BE] bit value.

Figure 7. Little/Big Endian Mode FIFO Access

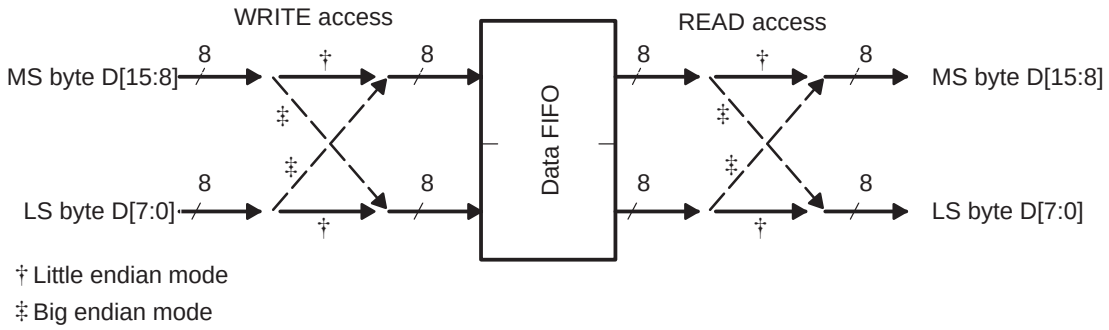


Table 18. Block Length Register (MMC_BLEN)

Base Address = 0xFFFB 7800 and 0xFFFB 7C00, Offset Address = 0x24		
Bit	**Name**	**Description**
15:11	−	Reserved
10:0	BLEN	Block length value

This register (Table 18) configures the core for the number of bytes to read or write. It must be initialized at least once before starting an MMC, SD, or SPI block data transfer (read or write).

MMC_BLEN[10:0] Block Length (BLEN)

A write into this register initializes an 11-bit counter that decrements by 1 after each byte is transferred. A read of register returns the number of bytes remaining to be transferred. When the counter reaches 0 and the last byte transfer completes, the core automatically reloads the block length to its programmed value, except in the case of an aborted/stopped data transfer.

In MMC/SD mode, this 11-bit value specifies the data block length. This value must be set with max $2^{READ\_BL\_LEN}-1$ for a block read or max $2^{WRITE\_BL\_LEN}-1$ for a block write, respectively.

READ_BL_LEN and WRITE_BL_LEN are CSD register settings of the card returned in an R2 response following a SEND_CSD command (CMD9).

In SPI mode and for a read transaction, the block length must be initialized with the exact byte count to read −1, excluding the opcode and address arguments.

Opcode and address arguments passed to the SPI device must be written into the FIFO buffer before starting the SPI transfer. BLEN starts to decrement as soon as the buffer contents have been shifted out to the SPI device. The buffer then starts to be filled with the received data from the SPI device.

In SPI mode and for a write transaction, the block length must be initialized with the exact byte count to write −1, including the number of bytes needed to pass the opcode and address arguments.

It is recommended to have opcodes and address that are passed to the SPI module written into the FIFO buffer before starting the SPI transfer. This allows the DMA write operation to access only the data portion. The block length starts to decrement for every byte shifted out to the SPI device.

- 0x000: 1 byte

  …

- 0x7FF: 2048 bytes

Value after reset is low (all 11 bits).

Table 19. Number of Blocks Register (MMC_NBLK)

Base Address = 0xFFFB 7800 and 0xFFFB 7C00, Offset Address = 0x28		
Bit	**Name**	**Description**
15:11	−	Reserved
10:0	NBLK	Number of blocks value

This register (Table 19) configures the number of blocks for a multiple block data transfer (read or write) operation for MMC/SD cards. This register is not used for SPI transfers.

MMC_NBLK[10:0] Number of Blocks (NBLK)

MMC/SD mode only.

In MMC/SD mode, this 11-bit value specifies the number of blocks for a multiple block data transfer (read or write). This value must be set with the number of blocks −1. Note that each block is of size MMC_BLEN[BLEN].

This register must be programmed before any multiple block data transfer. A write into this register initializes an 11-bit counter that decrements by 1 after each block transfer. A read of this register returns the number of blocks remaining to be transferred to the card. When the counter reaches 0, the transfer stops after the last transfer completes.

For stream or multiple block transfers, a block received/sent interrupt is generated only once after the last successful transfer when MMC_NBLK[NBLK] reaches 0. In stream mode, once the block receive/sent interrupt is received, MMC_NBLK[NBLK] can be reprogrammed to continue the transfer from the point it was interrupted.

Note: Value Requirement

This value must be 0x000 for a single block transfer. In stream mode, the minimum allowable number of blocks is 2. Finally, if the transfer is interrupted by a STOP_TRANSMISSION command (CMD12) before the counter reaches 0, reprogram this register before starting any new single- or multiple-block data transfer.

- 0x000: 1 block

  …

- 0x7FF: 2048 blocks

Value after reset is low (all 11 bits).

Table 20. Buffer Configuration Register (MMC_BUF)

Base Address = 0xFFFB 7800 and 0xFFFB 7C00, Offset Address = 0x2C		
Bit	**Name**	**Description**
15	RXDE	Receive DMA channel enable
14:13	–	Reserved
12:8	AFL	Buffer almost full level
7	TXDE	Transmit DMA channel enable
6:5	–	Reserved
4:0	AEL	Buffer almost-empty level

This register configures the buffer threshold level of the 32 16-bit-word FIFO entries and enables DMA transfers.

MMC_BUF[15] Receive DMA Channel Enable (RXDE)

When this bit is set to 1, the receive DMA channel is enabled and the core forces the MMC_STAT[AF] status bit to 0 regardless of the buffer almost-full level setting (MMC_BUF[AFL]). See *Buffer Almost Full (AFL)*; for more detail on DMA operations, see Section 4, *DMA Operations*.

- 0: Receive DMA channel disabled
- 1: Receive DMA channel enabled

Value after reset is low.

MMC_BUF[12:8] Buffer Almost Full Level (AFL)

This register holds the programmable almost-full level value used to determine the almost-full buffer condition. If users want an interrupt or a DMA read request to be issued during a read operation when the data buffer holds *at least* n 16-bit words, the buffer MMC_BUF[AFL] must be set with n−1.

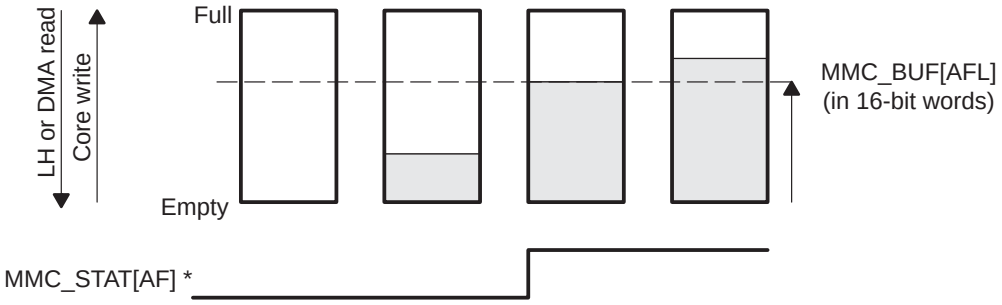
- 0x00: 1 16-bit word (2 bytes)

  …

- 0x1E: 31 16-bit words (62 bytes)
- 0x1F: 32 16-bit words (64 bytes)

Value after reset is low (all 5 bits).

Figure 8. Buffer Almost Full Level (AFL)



* Non-DMA mode only. In DMA mode, the DMA TX request is asserted to its active level under identical conditions.

MMC_BUF[7] Transmit DMA Channel Enable (TXDE)

When this bit is set to 1, the transmit DMA channel is enabled and the core forces the MMC_STAT[AE] status bit to 0 regardless of the buffer almost-empty level (MMC_BUF[AEL]). See *Buffer Almost Empty (AEL)*; for more detail on DMA operations, see Section 4, *DMA Operations*.

- 0: Receive DMA channel disabled
- 1: Receive DMA channel enabled

Value after reset is low.

MMC_BUF[4:0] Buffer Almost Empty Level (AEL)

This register holds the programmable almost-empty level value used to determine an almost-empty buffer condition. If users want an interrupt or a DMA write request to be issued during a write operation when the data buffer is able to receive n words of 16 bits, then MMC_BUF[AEL] must be set with n−1.

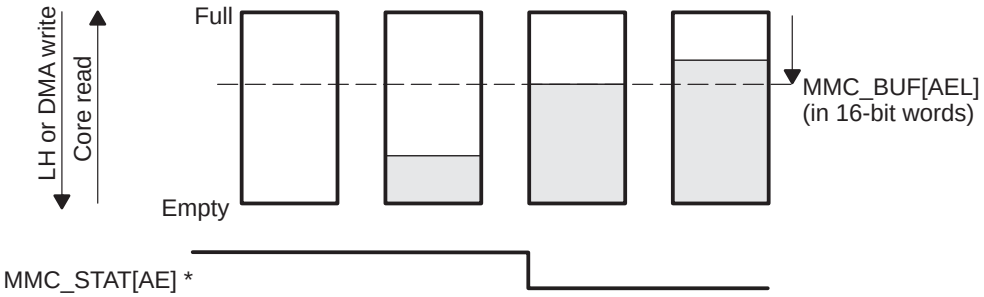
- 0x00: 1 16-bit word (2 bytes)

  …

- 0x1E: 31 16-bit words (62 bytes)
- 0x1F: 32 16-bit words (64 bytes)

Value after reset is low (all 5 bits).

Figure 9. Figure 1: Buffer Almost-Empty Level (AEL)



* Non−DMA mode only. In DMA mode, the DMA RX request is asserted to its active level under identical conditions.

Table 21. SPI Configuration Register (MMC_SPI)

Base Address = 0xFFFB 7800 and 0xFFFB 7C00, Offset Address = 0x30		
Bit	**Name**	**Description**
15	STR	Start SPI transfer
14	WNR	Write/not read
13	SODV	SPI_SO serial out pin default value
12	CSTR	SPI transfer controlled start
11:10	TCSH	Chip-select hold time control
9:8	TCSS	Chip-select setup time control
7	CSEL	Card socket connector select
6	−	Reserved
5:4	CS	Chip-select control
3	CSM	Chip-select mode
2	CSD	Chip-select disable

Table 21. SPI Configuration Register (MMC_SPI) (Continued)

Base Address = 0xFFFB 7800 and 0xFFFB 7C00, Offset Address = 0x30		
Bit	**Name**	**Description**
1	PHA	Phase control
0	POL	Polarity control

This register is used to configure the SPI interface and start an SPI transfer if the SPI mode has been enabled.

MMC_SPI[15] Start SPI Transfer (STR)

Set only bit.

This bit can be set only if MMC_SPI[CSTR] = 0. This bit always reads as 0. A write to 0 has no effect.

When the local host sets the bit to 1, an SPI transfer automatically starts. Users must initialize MMC_BLEN[BLEN] before starting an SPI transfer.

The SPI transfer automatically stops when the size programmed in MMC_BLEN[BLEN] decrements to 0 (in read and in write).

- 0: No action
- 1: SPI transfer starts (condition: MMC_SPI[CSTR] = 0)

Value after reset is low.

MMC_SPI[14] Write /Not Read (WNR)

This bit instructs the 11-bit block length counter (MMC_BLEN[BLEN]) to decrement either on byte read when MMC_SPI[WNR] = 0 or on byte write when MMC_SPI[WNR] = 1.

- 0: Decrement on byte received
- 1: Decrement on byte sent

Value after reset is low.

MMC_SPI[13] Serial-Out Default Value (SODV)

This bit determines the default value for the serial-out signal before SPI start and during the data phase of a SPI read transfer. This bit must be set to 1 for MMC/SD/SDIO cards using SPI protocol.

- 0: SPI_SO default value = 0
- 1: SPI_SO default value = 1

Value after reset is low.

MMC_SPI[12] SPI Transfer Controlled Start (CSTR)

When this bit is set, the FIFO is disabled and a write in MMC_DATA register automatically triggers an SPI transfer of one byte or one 16-bit word, depending on the local host write access. Because the FIFO is disabled, the local host must wait for the SPI transfer completion signaled by MMC_STAT[AF] = 1 before attempting to write again in the MMC_DATA register.

- 0: No action
- 1: A write in MMC_DATA starts an SPI transfer.

Value after reset is low.

MMC_SPI[11:10] Chip-Select Hold Time Control (TCSH)

This 2-bit field defines the number of interface clock cycles that the core waits after the last serial clock edge before asserting the chip-select signal to its inactive high level.

- 00: 0.5 clock cycle
- 01: 1.5 clock cycles
- 10: 2.5 clock cycles
- 11: 3.5 clock cycles

Value after reset is low (2 bits).

MMC_SPI[9:8] Chip-Select Setup Time Control (TCSS)

This 2-bit field defines the number of interface clock cycles that the core waits after asserting the chip-select signal to its active-low level before asserting the first serial clock edge.

- 00: 1 clock cycle
- 01: 2 clock cycles
- 10: 3 clock cycles
- 11: 4 clock cycles

Value after reset is low (2 bits).

Figure 10. SPI Mode C/S Timing Controls (POL = 0)

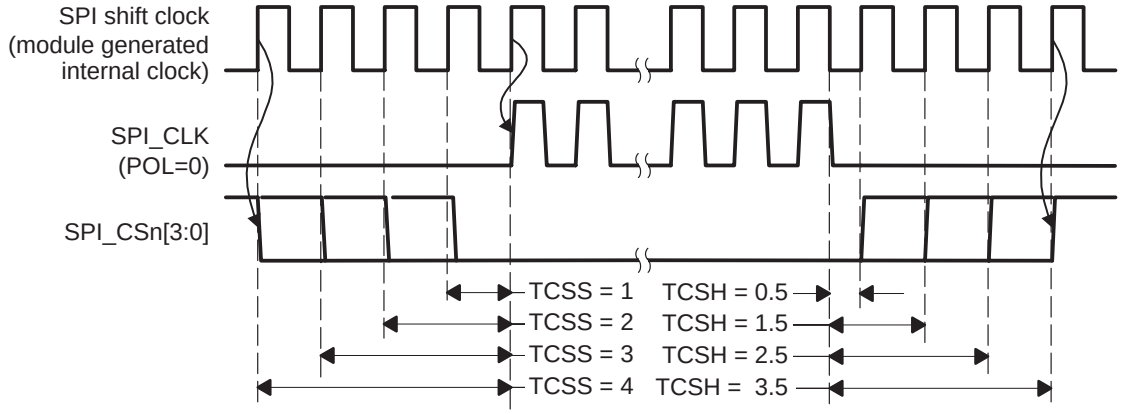
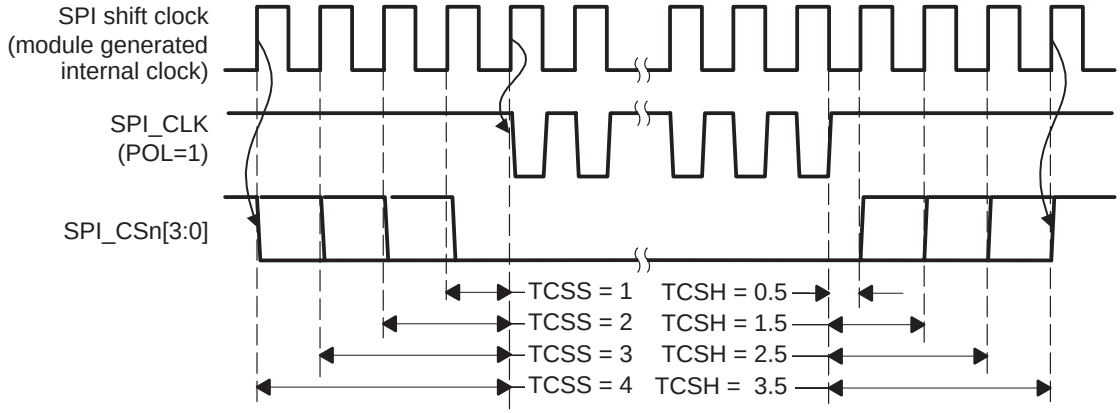


Figure 11. SPI Mode C/S Timing Controls (POL = 1)



MMC_SPI[7] Card Socket Connector Select (CSEL)

When this bit is set, the core outputs the serial clock on the MMC.CLK signal instead of the SPI.CLK signal. It also outputs the chip-select 0 on the MMC.DAT3 signal instead of the SPI_C/Sn[0] signal. This can be used to communicate with MMC/SD/SDIO cards using the SPI protocol on the default MMC/SD card connector.

- ❏ 0: Use the SPI.CLK and MMC_C/Sn[0] signals during SPI transfer (MMC.CLK inactive low and MMC.DAT3 inactive high though pull-up).
- ❏ 1: Use the MMC.CLK and MMC.DAT3 signals during SPI transfer (SPI.CLK inactive low and SPI_C/Sn[0] inactive high).

Value after reset is low.

MMC_SPI[5:4] Chip-Select Control (CS)

Encoded value that selects the device being targeted for SPI transfer.

- 00: C/S 0
- 01: C/S 1
- 10: C/S 2
- 11: C/S 3

Value after reset is low (2 bits).

MMC_SPI[3] Chip-Select Mode (CSM)

When this bit is set to 0 and enabled (MMC_SPI[CSD] = 0), the selected chip-select signal pin is brought active (low) only when the SPI transfer starts, and returns automatically to its inactive state (high) when the SPI transfer completes.

When this bit is set to 1, automatic control of the chip-select signal is disabled, and the signal pin is then manually controlled by the chip-select disable register bit (MMC_SPI[CSD]). This mode provides support for a complex SPI transfer scheme that requires chip-select to be kept active during the entire transfer (for example, MMC card write with busy condition).

- 0: Automatic mode
- 1: Manual mode (controlled by CSD)

Value after reset is low.

MMC_SPI[2] Chip-Select Disable (CSD)

When this bit is set to 0, the selected chip-select signal is asserted to its active (low) state either automatically when MMC_SPI[CSM] = 0, or manually when MMC_SPI[CSM] = 1.

When this bit is set to 1, the selected chip-select signal is forced to its inactive (high) state. This bit can be used to send dummy clocks with chip-select inactive to an MMC or SD card.

- 0: Selected chip-select conditionally asserted (low). See Table 22.
- 1: Selected chip-select deasserted (high)

Value after reset is low.

Table 22. Chip-Select Control (SPI Mode)

CSM	CSD	Selected CS	Comment
0	0	High → Low → High	Automatic mode: CS asserted active (low) during SPI transfer
0	1	High	Automatic mode: CS forced inactive (high)
1	0	Low	Manual mode: CS asserted active (low)
1	1	High	Manual mode: CS asserted inactive (high)

MMC_SPI[1] Clock Phase (PHA)

The clock polarity and clock phase bits select four different clocking schemes for the serial clock pin (SPI.CLK or MMC.CLK). The clock phase bit (MMC_SPI[PHA]) selects a half-cycle delay for the clock.

When the clock phase = 0:

- MSB data is ready one-half cycle of the serial clock before the SPI clock starts.
- Data is shifted in during reception on the first edge transition of the serial clock.
- Data is shifted out in transmission on the second edge transition of the serial clock.

When the clock phase = 1:

- Data is shifted out in transmission on the first edge transition of the serial clock.
- Data is shifted in during reception on the second edge transition of the serial clock.
  - 0: Phase 0
  - 1: Phase 1

Value after reset is low.

MMC_SPI[0] Clock Polarity (POL)

The clock polarity bit (MMC_SPI[POL]) selects the active edge of the clock, either rising or falling.

When this bit is 0, the idle value of the serial clock signal (SPI.CLK or MMC.CLK) is low, and the rising edge of the clock is active.

When this bit is 1, the idle value of the serial clock signal is high, and and the falling edge of the clock is active.

- 0: Rising edge active
- 1: Falling edge active

Value after reset is low.

Table 23. SDIO Mode Configuration Register (MMC_SDIO)

Base Address = 0xFFFB 7800 and 0xFFFB 7C00, Offset Address = 0x34		
Bit	**Name**	**Description**
15	C5E	Card status error on bit 5 of response 1 enable
14	C14E	Card status error on bit 4 of response 1 enable
13	C13E	Card status error on bit 3 of response 1 enable
12	C12E	Card status error on bit 2 of response 1 enable
11	D3PS	DAT3 polarity control
10	D3ES	DAT3 edge/level detection mode
9	CDWE	Card-detect wake request enable
8	IWE	Interrupt wake request enable
7	DCR4	Disable CRC7 check in R4 response
6	XDTS	Extended data time-out mode select (default OD)
5	DPE	Data time-out prescaler enable
4	RW	Assert read wait condition to the card
3	−	Reserved
2	CDE	Card-detect mode enable
1	RWE	SDIO read wait mode enable
0	IRQE	SDIO interrupt mode enable

This register configures the MMC/SD interface for SDIO operation and the card-detection mechanism on DAT3. If the card-detection mechanism is based on a mechanical switch, its control/sense is the responsibility of another system module.

MMC_SDIO[15] Card Status Error on R5 Enable (C5E)

This bit must be set to 1 for SDIO cards only.

If this bit is set to 1, the R5 response generates card status errors and enables the command time-out. See SDIO CMD52. See Section 5.1, IO_RW_DIRECT command (CMD52), *SDIO Card Specification*.

By default, when this bit is set to 0, the R5 response does not generate errors or disable the command time-out for MMC interrupt mode support.

- ❏ 0: MMC default mode
- ❏ 1: Card status errors on response R5 enabled (SDIO card only)

Value after reset is low.

MMC_SDIO[14] Card Status Error on Bit 4 of Response R1 Enable (C14E)

This bit must be set to 1 for SDIO cards only.

When this bit is set to 1, a card status error is generated if bit 4 of the status is 1 for an R1 or R1b response.

- ❏ 0: Error on bit 4 masked
- ❏ 1: Card status errors on bit 4 of response R1 enabled (SDIO card only)

Value after reset is low.

MMC_SDIO[13] Card Status Error on Bit 3 of Response R1 Enable (C13E)

This bit must be set to 1 for SD cards only or for an application-specific command that generates an error.

When this bit is set to 1, a card status error is generated if bit 3 of the status is 1 for an R1 or R1b response.

- ❏ 0: Error on bit 3 masked
- ❏ 1: Card status errors on bit 3 of response R1 enabled (SD card or application specific only)

Value after reset is low.

MMC_SDIO[12] Card Status Error on Bit 2 of Response R1 Enable (C12E)

This bit must be set to 1 for an application-specific command that generates an error.

When this bit is set to 1, a card status error is generated if bit 2 of the status is 1 for an R1 or R1b response.

- ❏ 0: Error on bit 2 masked
- ❏ 1: Card status errors on bit 2 of response R1 enabled (application specific only)

Value after reset is low.

MMC_SDIO[11] DAT3 Polarity Select (D3PS)

This bit determines the active edge/level condition for DAT3 signal.

- ❏ 0: Not inverted
- ❏ 1: Inverted

Value after reset is low.

MMC_SDIO[10] DAT3 Polarity Select (D3ES)

This bit determines whether DAT3 is an edge- or level-sensitive signal.

- ❏ 0: Level sensitive
- ❏ 1: Edge sensitive

Value after reset is low.

MMC_SDIO[9] Card-Detect Wake-Request Enable (CDWE)

SD/SDIO card only.

This bit must be set to allow a valid card-detect condition to assert the WAKE_REQ signal active.

- ❏ 0: Masked
- ❏ 1: Unmasked

Value after reset is low.

MMC_SDIO[8] Interrupt Wake Request Enable (IWE)

SDIO card only.

This bit must be set to allow a valid SDIO interrupt condition to assert the WAKE_REQ signal active.

- ❏ 0: Masked
- ❏ 1: Unmasked

Value after reset is low.

MMC_SDIO[7] Disable CRC7 Check on R4 Response (DCR4)

SDIO card only.

This bit must be set to disable the CRC7 check on R4 response. This prevents the MMC_STAT[CCRC] from being set following an IO_SEND_OP_COND command (see *SDIO Card Specification Part E1*).

- ❏ 0: CRC7 check enabled on R4 response
- ❏ 1: CRC7 check disabled on R4 response

Value after reset is low.

MMC_SDIO[6] Extended Data Time-Out Mode Select (XDTS)

This bit determines the default action of the MMC_CMD[ODTO] register bit when passing the command to the card (see section).

The local host must set this bit to 1 after the card has been identified as an SDIO card in order to support the IO_RW_DIRECT command with long time-out values.

This bit can remain clear in all other cases.

- ❏ 0: Open-drain control mode
- ❏ 1: Time-out control mode

Value after reset is low.

MMC_SDIO[5] Data Time-Out Prescaler Enable (DPE)

When the local host sets this bit to 1, the data time-out value (MMC_DTO[DTO]) is x1024 the number of MMC.CLK cycles.

- ❏ 0: x1 (prescaler off)
- ❏ 1: x1024 (prescaler on)

Value after reset is low.

MMC_SDIO[4] Assert Read Wait Condition (RW)

SDIO card only.

The local host can set this bit to stall a read multiple data transfer (CMD53) temporarily. After setting this bit, the host must wait until the wait becomes effective (MMC_STAT[CRW] = 1) before sending a new command without data phase (such as CMD52) to the card.

After completion of CMD52, the local host can resume the read multiple data transfer by clearing this bit.

- 0: No action or read wait released
- 1: Read wait requested (if MMC_SDIO[RWE] = 1)

Value after reset is low.

MMC_SDIO[2] Card-Detect Enable (CDE)

SD or SDIO card only.

The local host must set this bit to 1 to permit the card detection operation.

When this bit is set, the core sets the MMC_STAT[CD] register bit to 1 if a valid card-detection condition occurs on the DAT3 line. This happens when a selectable level/edge sensitive event occurs on DAT3 and the card is in idle state. The MMC_SDIO[D3ES] register bit controls edge/level, and the MMC_SDIO[D3PS] register bit controls polarity. This event also asynchronously asserts the WAKE_REQ signal event to warn the system to wake up its clocks if needed. The MMC_SDIO[CDWE] register bit masks the WAKE_REQ signal for this event.

- 0: Card-detect mode disabled
- 1: Card-detect mode enabled

Value after reset is low.

MMC_SDIO[1] SDIO Read Wait Mode Enable (RWE)

SDIO card only.

The local host must set this bit to 1 to permit the read wait operation.

When this bit is set to 0, the read wait operation is disabled, and setting a 1 in the MMC_SDIO[RW] register bit results in no action.

- 0: Read wait mode disabled
- 1: Read wait mode enabled

Value after reset is low.

MMC_SDIO[0] SDIO Interrupt Mode Enable (IRQE)

SDIO card only.

The local host must set this bit to 1 to permit the interrupt operation.

When this bit is set, the core sets the MMC_STAT[CIRQ] register bit to 1 if an interrupt condition is detected on the DAT1 line. A detected interrupt condition when the card is in idle state also asynchronously asserts the WAKE_REQ signal event to warn the system to wake up its clocks if needed. The MMC_SDIO[IWE] register bit masks the WAKE_REQ signal for this event.

- ❏ 0: Interrupt mode disabled
- ❏ 1: Interrupt mode enabled

Value after reset is low.

Table 24. System Test Register (MMC_SYST)

Base Address = 0xFFFB 7800 and 0xFFFB 7C00, Offset Address = 0x38		
Bit	**Name**	**Description**
15	WAKD	WAKE_REQ output signal data value (internal signal to system)
14	SSB	Set status bits
13	RDYD	Ready/busy input signal data value
12	DDIR	DAT[3:0] signals direction
11	D3D	DAT3 input/output signal data value
10	D2D	DAT2 input/output signal data value
9	D1D	DAT1 input/output signal data value
8	D0D	DAT0/SI input/output signal data value
7	CDIR	CMD/SO signal direction
6	CDAT	CMD/SO input/output signal data value
5	MCKD	MMC clock output signal data value
4	SCKD	SPI clock output signal data value
3	CS3D	C/S3 output signal data value
2	CS2D	C/S2 output signal data value
1	CS1D	C/S1 output signal data value
0	CS0D	C/S0 output signal data value

This register controls the signals that connect to I/O pins when the module is configured in system test (SYSTEST) mode.

MMC_SYST[15] WAKE_REQ Data (WAKD)

The WAKE_REQ signal is driven high or low according to the value written into this register bit.

Value after reset is low.

MMC_SYST[14] Set Status Bits (SSB)

Writing 1 into this bit sets to 1 all status bits contained in the MMC_STAT register.

Writing 0 into this bit does not clear already set status bits; only writing 1 into a set status bit can clear it (see MMC_STAT operation). This bit must be cleared before attempting to clear a status bit.

- ❏ 0: No action
- ❏ 1: Set all status bits

Value after reset is low.

MMC_SYST[13] Ready/Busy Data (RDYD)

This read-only bit returns the value of the signal on the input pad (high or low).

- ❏ 0: Ready/busy low
- ❏ 1: Ready/busy high

Value after reset is high.

MMC_SYST[12] Direction (DDIR)

When set, this bit places all of the in/out MMC_DAT[3:0] pins in output mode.

- ❏ 0: Input
- ❏ 1: Output

Value after reset is low.

MMC_SYST[11:8] Data (DnD)

If MMC_SYST[DDIR] = 0 (input mode direction), these bits return the value on the corresponding MMC.DAT pins (high or low). A write into these bits has no effect.

If MMC_SYST[DDIR] = 1 (output mode direction), the MMC.DAT pins are driven high or low according to the value written into these register bits.

Value after reset is low (all 4 bits).

MMC_SYST[7] CMD Direction (CDIR)

When set, this bit places the in/out MMC.CMD/SPI.DO pin in output mode.

- 0: Input
- 1: Output

Value after reset is low.

MMC_SYST[6] CMD Data (CDAT)

If MMC_SYST[CDIR] = 0 (input mode direction), these bits return the value on the MMC.CMD/SPI.DO pin (high or low). A write into this bit has no effect.

If MMC_SYST[CDIR] = 1 (output mode direction), the MMC_CMD pin is driven high or low according to the value written into this register bit.

Value after reset is low.

MMC_SYST[5] MMC.CLK Data (MCKD)

The MMC.CLK pin is driven high or low according to the value written into this register bit.

Value after reset is low.

MMC_SYST[4] SPI.CLK Data (SCKD)

The SPI.CLK pin is driven high or low according to the value written into this register bit.

Value after reset is low.

MMC_SYST[3:0] CSData (CSnD)

The CS[3:0] pins are driven high or low according to the values written into these register bits.

Value after reset is low (all 4 bits).

Table 25. Module Revision Register (MMC_REV)

Base Address = 0xFFFB 7800 and 0xFFFB 7C00, Offset Address = 0x3C		
Bit	**Name**	**Description**
15:8	–	Reserved
7:0	REV	Module revision number

This read-only register contains the revision number of the module. A write to this register has no effect.

MMC_REV[7:0] Module Revision Number (REV)

This 8-bit field indicates the revision number of the RTL for this module. This value is fixed by hardware.

- ❏ The 4 LSBs indicate a minor revision.
- ❏ The 4 MSBs indicate a major revision.
- ❏ 0x30: Revision 3.0
- ❏ 0x31: Revision 3.1

A reset has no effect on the value returned.

Notes: 1) MMC/SD HOST controller without SDIO support is revision 1.x or 2.x (WMU_020_1 spec)
2) MMC/SD/SDIO HOST controller is revision 3.x.

Table 26. MMC/SD Command Response Register 0 (MMC_RSP0)

Base Address = 0xFFFB 7800 and 0xFFFB 7C00, Offset Address = 0x40		
Bit	**Name**	**Description**
15:0	RSP0	CMD response (R2[15:0])

This 16-bit register holds bit positions [15:0] for a 128-bit response of type R2.

MMC_RSP0 is also reset after a write into the MMC.CMD/SPI.DO register (command sent).

Table 27. MMC/SD Command Response Register 1 (MMC_RSP1)

Base Address = 0xFFFB 7800 and 0xFFFB 7C00, Offset Address = 0x44		
Bit	**Name**	**Description**
15:0	RSP1	CMD response (R2[31:16])

This 16-bit register holds bit positions [31:16] for a 128-bit response of type R2.

MMC_RSP1 is also reset after a write into the MMC.CMD/SPI.DO register (command sent).

Table 28. MMC/SD Command Response Register 2 (MMC_RSP2)

Base Address = 0xFFFB 7800 and 0xFFFB 7C00, Offset Address = 0x48		
Bit	**Name**	**Description**
15:0	RSP2	CMD response (R2[47:32])

This 16-bit register holds bit positions [47:32] for a 128-bit response of type R2.

MMC_RSP2 is also reset after a write into the MMC.CMD/SPI.DO register (command sent).

Table 29. MMC/SD Command Response Register 3 (MMC_RSP3)

Base Address = 0xFFFB 7800 and 0xFFFB 7C00, Offset Address = 0x4C		
Bit	**Name**	**Description**
15:0	RSP3	CMD response (R2[63:48])

This 16-bit register holds bit positions [63:48] for a 128-bit response of type R2.

MMC_RSP3 is also reset after a write into the MMC.CMD/SPI.DO register (command sent).

Table 30. MMC/SD Command Response Register 4 (MMC_RSP4)

Base Address = 0xFFFB 7800 and 0xFFFB 7C00, Offset Address = 0x50		
Bit	**Name**	**Description**
15:0	RSP4	CMD response (R2[79:64])

This 16-bit register holds bit positions [79:64] for a 128-bit response of type R2.

MMC_RSP4 is also reset after a write into the MMC.CMD/SPI.DO register (command sent).

Table 31. MMC/SD Command Response Register 5 (MMC_RSP5)

Base Address = 0xFFFB 7800 and 0xFFFB 7C00, Offset Address = 0x54		
Bit	**Name**	**Description**
15:0	RSP5	CMD response (R2[95:80])

This 16-bit register holds bit positions [95:80] for a response of type R2.

MMC_RSP5 is also reset after a write into the MMC.CMD/SPI.DO register (command sent).

Table 32. MMC/SD Command Response Register 6 (MMC_RSP6)

Base Address = 0xFFFB 7800 and 0xFFFB 7C00, Offset Address = 0x58		
Bit	**Name**	**Description**
15:0	RSP6	CMD response (R2[111:96], R1/R1b/R3/R4/R5/R6[23:8])

This 16-bit register holds bit positions [111:96] for a 128-bit response of type R2 and bit positions [23:8] for a 32-bit response of types R1/R1b/R3/R4/R5/R6.

MMC_RSP6 is also reset after a write into the MMC_CMD register (command sent).

Table 33. MMC/SD Command Response Register 7 (MMC_RSP7)

Base Address = 0xFFFB 7800 and 0xFFFB 7C00, Offset Address = 0x5C		
Bit	**Name**	**Description**
15:0	RSP7	CMD response (R2[127:112], R1/R1b/R3/R4/R5/R6[39:24])

This 16-bit register holds bit positions [127:112] for a 128-bit response of type R2 and bit positions [39:24] for a 32-bit response of types R1/R1b/R3/R4/R5/R6.

MMC_RSP7 is also reset after a write into the MMC.CMD/SPI.DO register (command sent).

Table 34. SDIO Suspend/Resume Control Register (MMC_IOSR)

Base Address = 0xFFFB 7800 and 0xFFFB 7C00, Offset Address = 0x60		
Bit	**Name**	**Description**
15:4	–	Reserved
3	STOP	Stop core data operation request (after card grants suspend)
2	SAVE	Save FIFO contents of suspended function
1	RESU	Next SD command is a resume request
0	SUSP	Next SD command is a suspend request

This register implements the necessary controls for SDIO suspend/resume operations.

MMC_IOSR[3] Stop Core Data Operation Request (STOP)

SDIO cards only

The local host can set this bit to 1 only when the SDIO card function has been suspended. When this bit is set, the core immediately stops signaling the data transfer request (either DMA request or interrupt) to the system if no request is pending, or immediately after it has been serviced if a request is pending. Once the bit is set, the core automatically clears it to signal that FIFO

operations are effectively suspended. This happens when all data requests have been disabled and serviced. The local host must poll this bit until cleared to 0 before attempting to save the FIFO contents.

A write to 0 has no action.

- 0: No action, or FIFO operations suspended
- 1: Stop core data operation request

Value after reset is low.

MMC_IOSR[2] Save FIFO Contents of Suspended Function (SAVE)

SDIO cards only.

This bit must be set to 1 when the core acknowledges the stop data operation request MMC_IOSR[STOP] = 0. When this bit is set, the FIFO is placed in read mode, and the local host downloads the FIFO contents of the suspended function until totally empty (signaled by MMC_STAT[AF] = 0).

In this mode, the actual threshold value set in MMC_BUFF[AFL] is a don't care. MMC_STAT[AF] equals 0 only when the FIFO is totally empty.

- 0: No action
- 1: Save action (SDIO card only)

Value after reset is low.

MMC_IOSR[1] Next SD Command Is a RESUME Request (RESU)

SDIO cards only.

This bit must be set to 1 when the next command passed to the card is a resume request for a suspended function.

- 0: No action
- 1: Next command is resume (SDIO card only).

MMC_IOSR[0] Next SD Command Is a SUSPEND Request (SUSP)

SDIO cards only.

This bit must be set to 1 when the next command passed to the card is a suspend request of the current active function.

When this bit is set for a multiple block read or write operation, the core automatically stops its data transfer operation at the end of the current active block. It also stops requesting new data from the local host or system DMA for a new block write.

- 0: No action
- 1: Next command is suspend (SDIO card only).

Value after reset is low.

Table 35 lists the characteristics of the system control register.

Table 35. System Control Register(1.1MMC_SYSC)

Base Address = 0xFFFB 7800 and 0xFFFB 7C00, Offset Address = 0x64		
Bit	**Name**	**Description**
15:2	–	Reserved
1	SRST	Software reset
0	–	Reserved

MMC_SYSC[1] Software Reset (SRST)

When this bit is set to 1, the entire module is reset as for the hardware reset. The core automatically sets this bit to 0, and it is only reset by the hardware reset. It always returns 0 during reads.

The module can be also partially reset using the MMC_CON[POW] bit.

- 0: No action
- 1: Module is reset.

Value after reset is low.

Table 36. System Status Register(MMC_SYSS)

Base Address = 0xFFFB 7800 and 0xFFFB 7C00, Offset Address = 0x68		
Bit	**Name**	**Description**
15:1	–	Reserved
0	RSTD	Reset done

MMC_SISS[0] Reset Done Status (RSTD)

This read-only bit indicates the state of the reset in case of hardware reset, global software reset MMC_SYSC[SRST], or partial software reset MMC_CON[POW]).

The module must receive all of its clocks before it can grant a reset completed status.

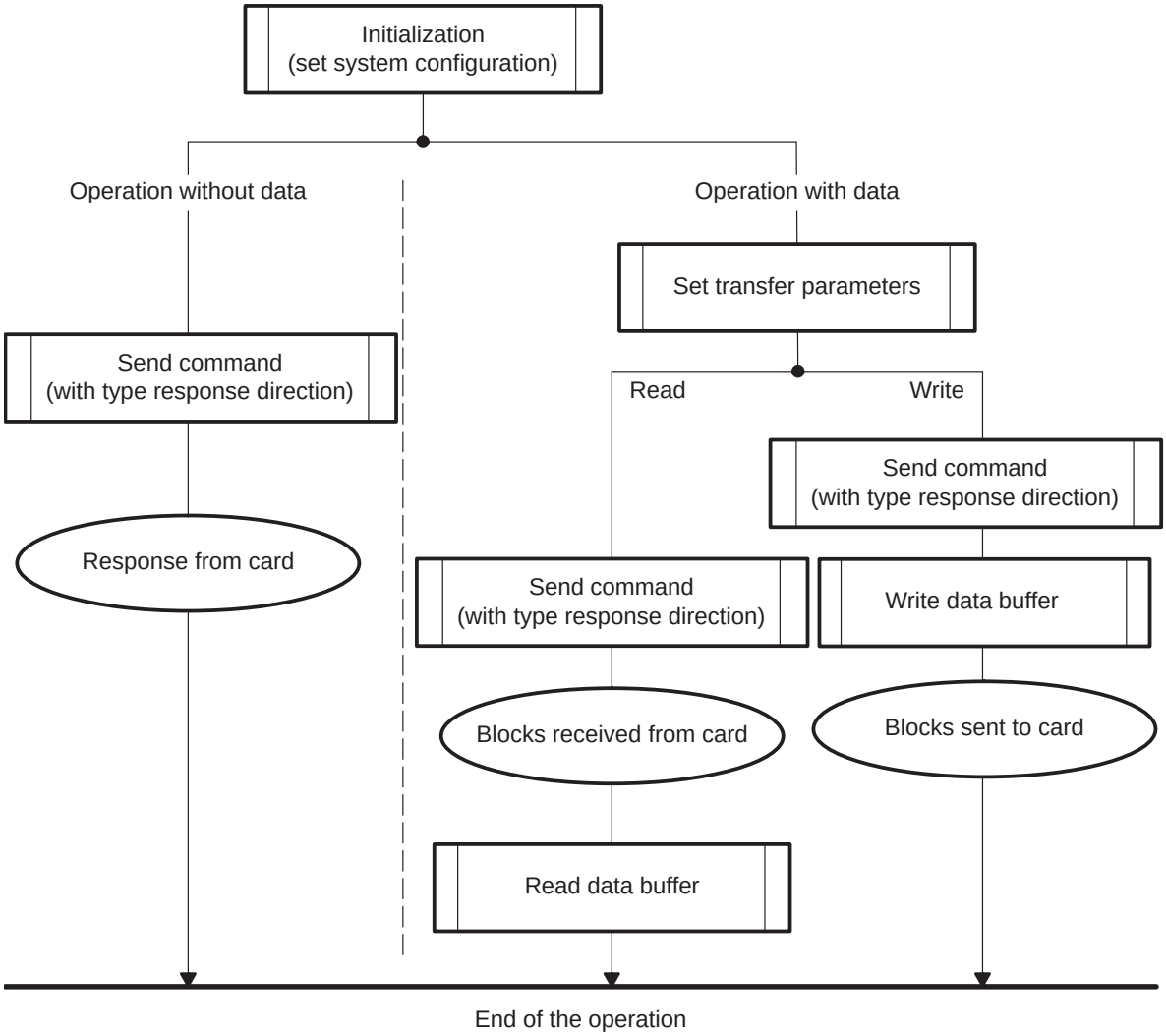
- 0: Internal module reset in ongoing or partially held in reset
- 1: Reset completed

Value after reset is low.

3 MMC Command Flow

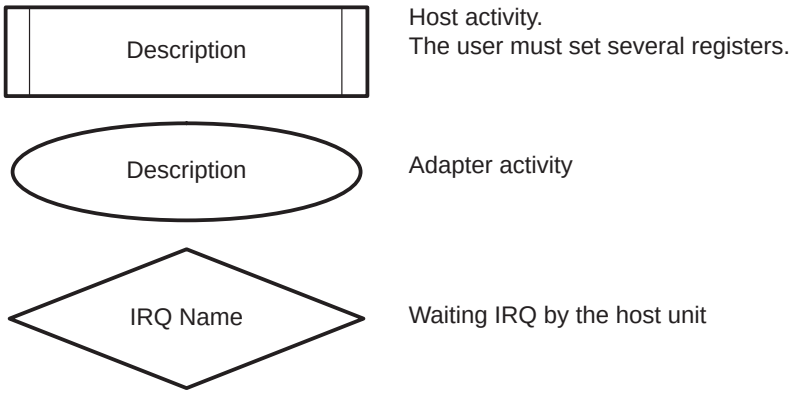
To properly drive the MMC/SD/SDIO correctly for the execution of a command, the host must process as shown in Figure 12 and Figure 13.

Figure 12. General Command Flow



In all modes (MMC, SD, SPI), an initialization phase is necessary at the beginning. After the initialization, the MMC/SD adapter works differently depending on whether the host sends a command without data or with data, and also according to the type, the index of the response, and the direction.

Figure 13. Flow Conventions



3.1 Basic Operations

Figure 14 depicts initialization.

Figure 14. Initialization

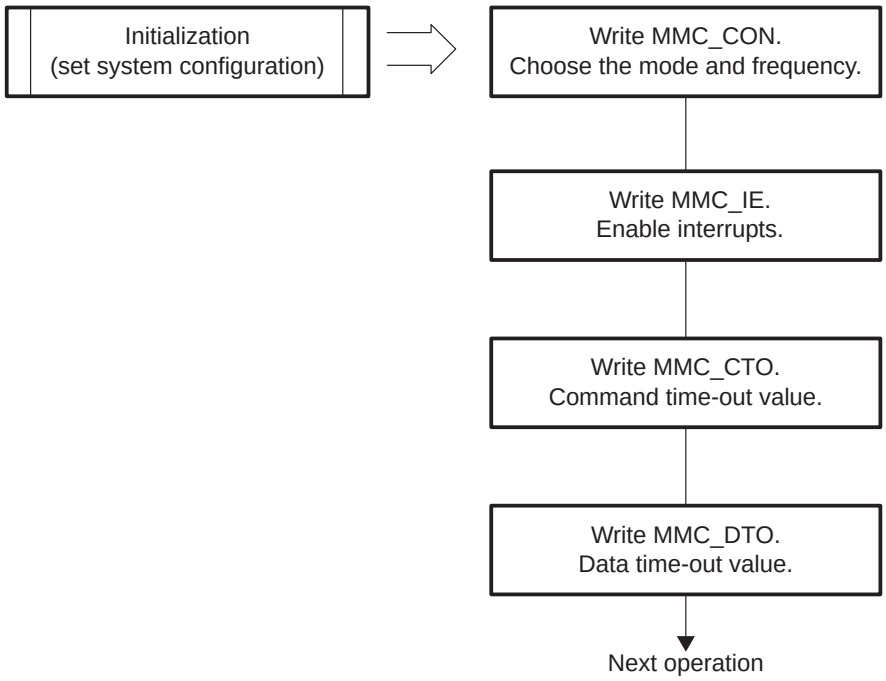


Figure 15 depicts the command transfer.

Figure 15. Command Transfer

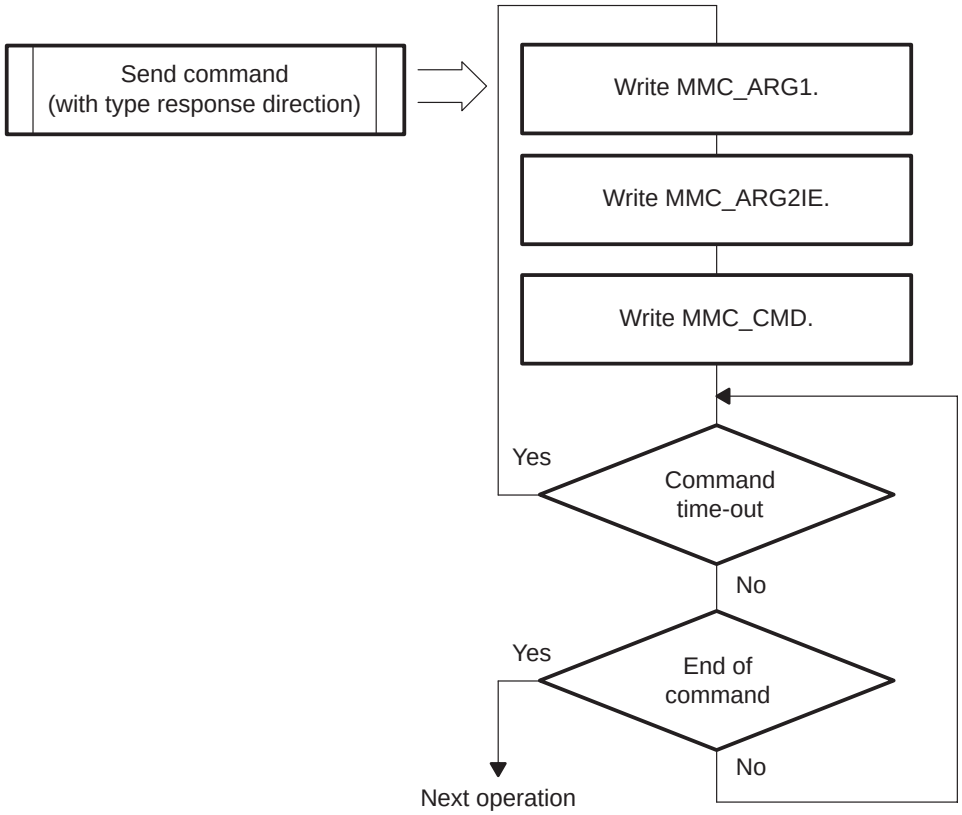
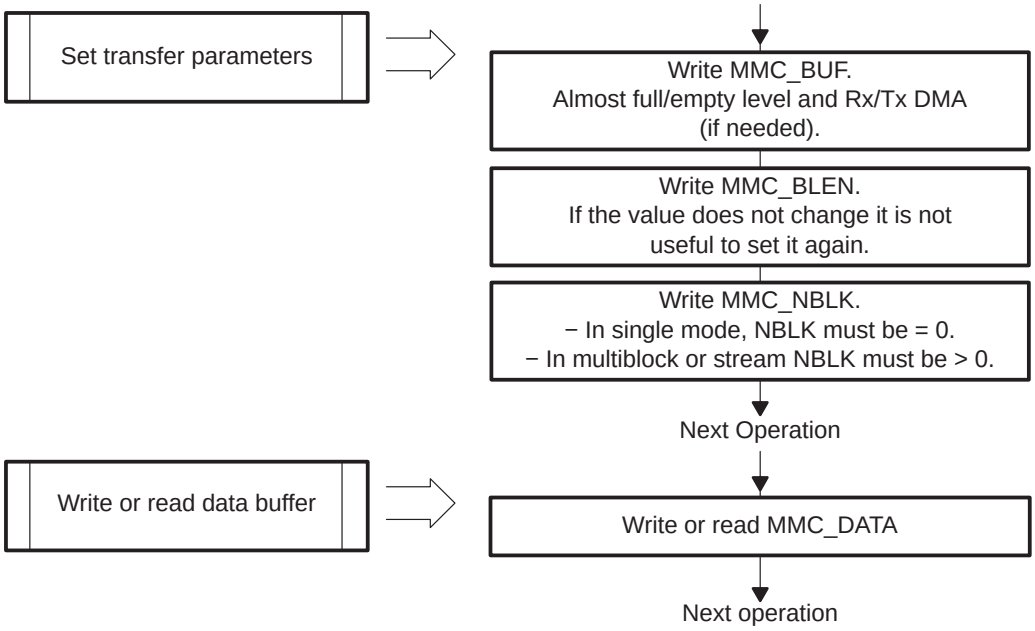


Figure 16 illustrates the transfer of data.

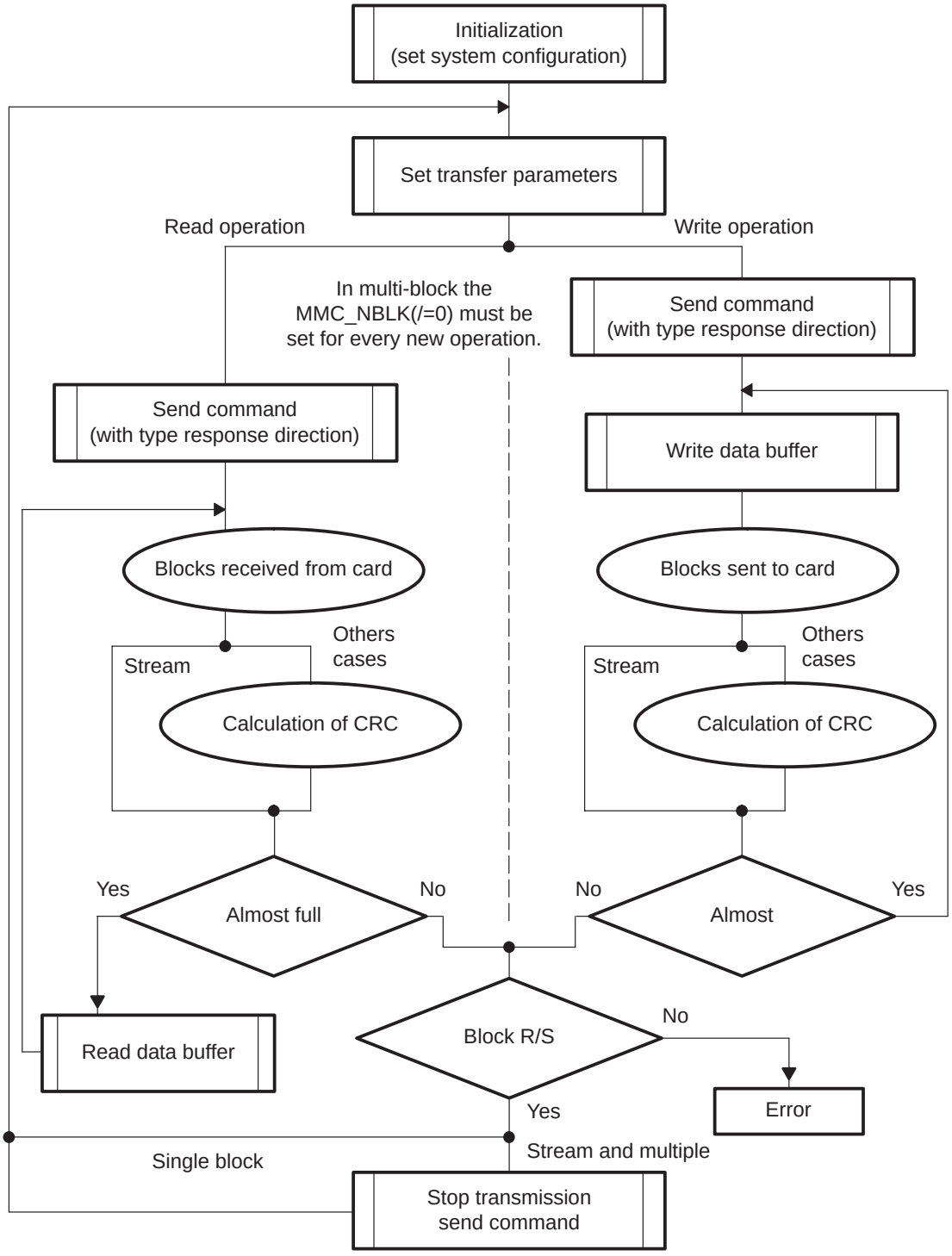
Figure 16. Data Transfer



Mode MMC/SD (MMC_CON[MODE] = 00) is selected for this example.

Figure 17 illustrates the transfer of data in MMC/SD mode.

Figure 17. Data Transfer in MMC/SD Mode


Initialization
(set system configuration)
Set transfer parameters
Read operation
Write operation
In multi-block the
MMC_NBLK(/=0) must be
set for every new operation.
Send command
(with type response direction)
Send command
(with type response direction)
Write data buffer
Blocks received from card
Blocks sent to card
Others
cases
Stream
Calculation of CRC
Others
cases
Stream
Calculation of CRC
Yes
Almost full
No
No
Almost
Yes
Read data buffer
Block R/S
No
Error
Yes
Single block
Stream and multiple
Stop transmission
send command

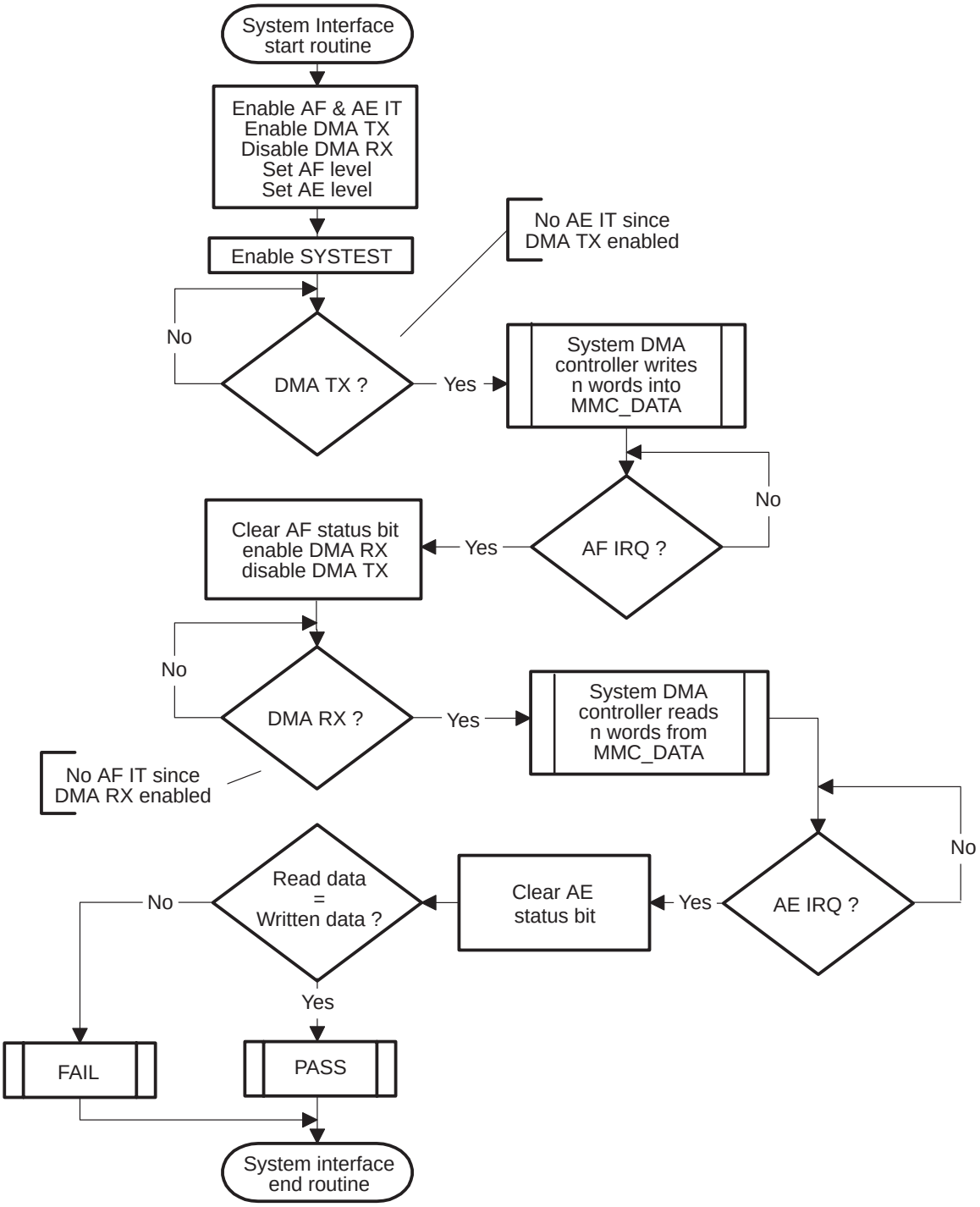
3.2 System Test Mode

The system test (SYSTEST) mode implemented in the MMC/SD host controller easily checks the correctness of the system interconnect either internally to the peripheral bus, central DMA, and interrupt handler, or externally to device I/O pads.

I/O verification is not depicted here but can be performed in SYSTEST mode by toggling the outputs and capturing the logic state of the inputs.

Figure 18 depicts a scenario to validate data DMA transfers and interrupt assertions in a one-pass flow. All data transfers are performed under DMA control.

Figure 18. System Interface Test Flow



3.3 SPI Mode

In write operations, the block length register (MMC_BLEN[BLEN]) is loaded with the data transfer dimension, opcode, and address.

In read operations, the block length register is only loaded with the data transfer dimension.

4 DMA Operations

4.1 MMC DMA Receive Mode

In a DMA block read operation (single or multiple), the DMA RX request signal is asserted to its active level when the FIFO level becomes greater than or equal to the threshold value (in 16-bit words) set in MMC_BUF[AFL]. The DMA RX request is deasserted to its inactive level when the system DMA has read one single word from the FIFO.

Because the request lasts one 16-bit word read cycle, it is recommended that the threshold value in MMC_BUF[AFL] (expressed in 16-bit words) be equal to the DMA burst access size (n). If the system DMA does not support more than one 16-bit word read access, MMC_BUF[AFL] must be set to 0.

New DMA requests are internally masked until the system DMA has performed exactly n reads.

Note: Block Size

Because each DMA transfer is of equal size, the block size of the transfer must be a multiple of the DMA read access size.

Summary:

- DMA transfer size = n ≤ FIFO size (max 32 16-bit words)
- MMC_BUF[AFL] = n − 1 (FIFO threshold level)
- n = Submultiple of block size

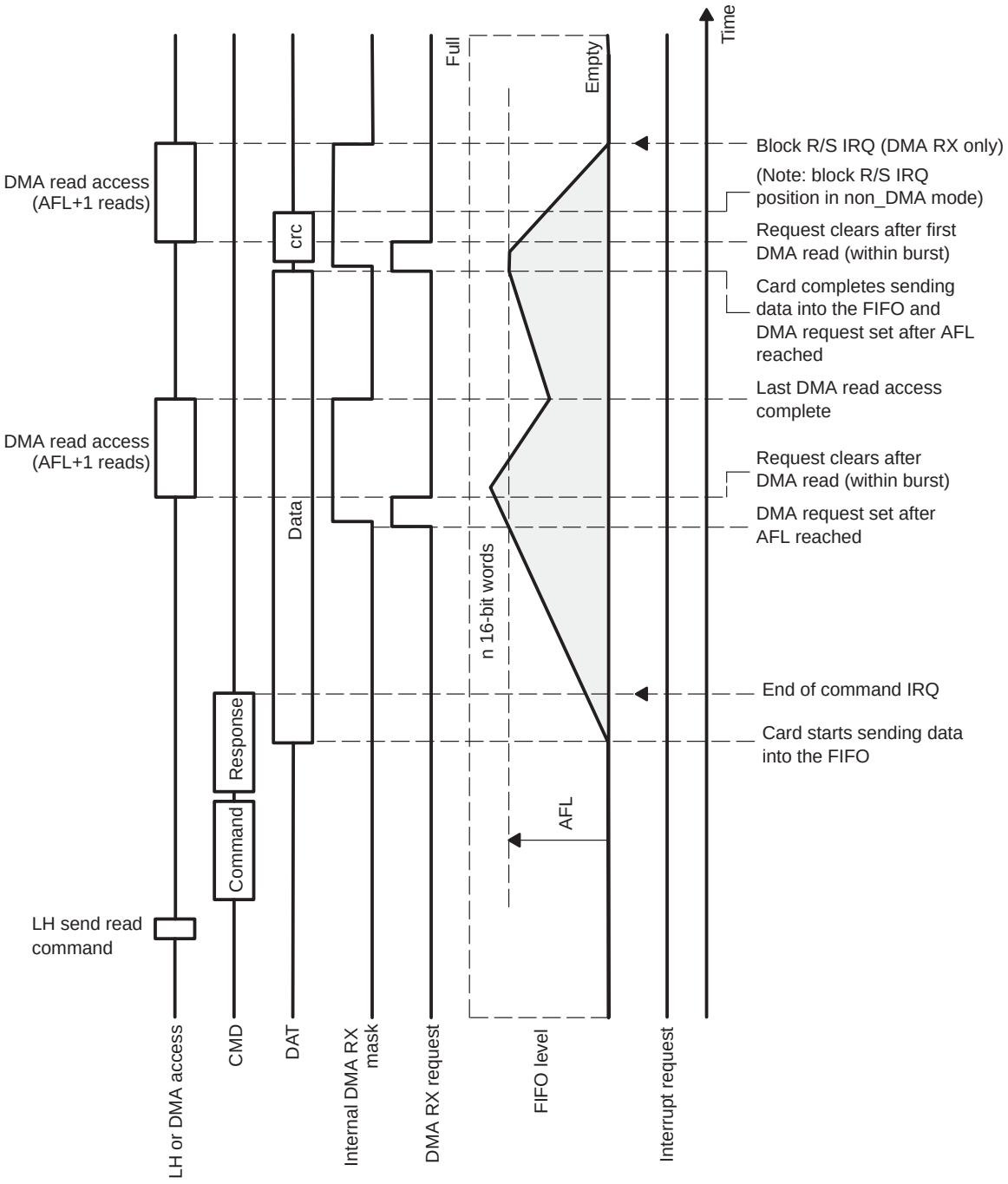
Example: Multiple block read of 7 blocks of 256 bytes each.

The DMA transfer size n can be set to 4 16-bit words (8 bytes) and MMC_BUF[AFL] = 0x3. The read transfer operation completes after 224 system DMA RX requests.

The receive FIFO never overflows. If the FIFO becomes full, the MMC.CLK signal is momentarily stopped until the system DMA or the local host performs a read access, which starts emptying the FIFO.

Figure 19 shows a typical DMA read operation to an MMC card.

Figure 19. MMC Mode DMA RX Transfer



Signals are represented in a symbolic form (a high level for the DMA mask or the request signals denotes an ON condition).

4.2 MMC DMA Transmit Mode

In a DMA block write operation (single or multiple), the DMA TX request signal is asserted to its active level when the FIFO level becomes less than or equal to the threshold value (in 16-bit words) set in MMC_BUF[AEL] after the block write command has been set (write action into MMC.CMD/SPI.DO). The DMA TX request is deasserted to its inactive level when the system DMA has written one single word into the FIFO.

Because the request lasts one 16-bit word write cycle, it is recommended that the threshold value in MMC_BUF[AEL] (expressed in 16-bit words) be equal to the DMA burst size (n). If the system DMA does not support more than one 16-bit word write access, MMC_BUF[AEL] must be set to 0.

New DMA requests are internally masked until the system DMA has performed exactly n writes.

Note: Block Size

Because each DMA transfer is of equal size, it is necessary to have the block size of the transfer be a multiple of the DMA write access size.

Summary:

- DMA transfer size = n ≤ FIFO size (max 32 16-bit words)
- MMC_BUF[AEL] = n − 1 (FIFO threshold level)
- n = Submultiple of block size

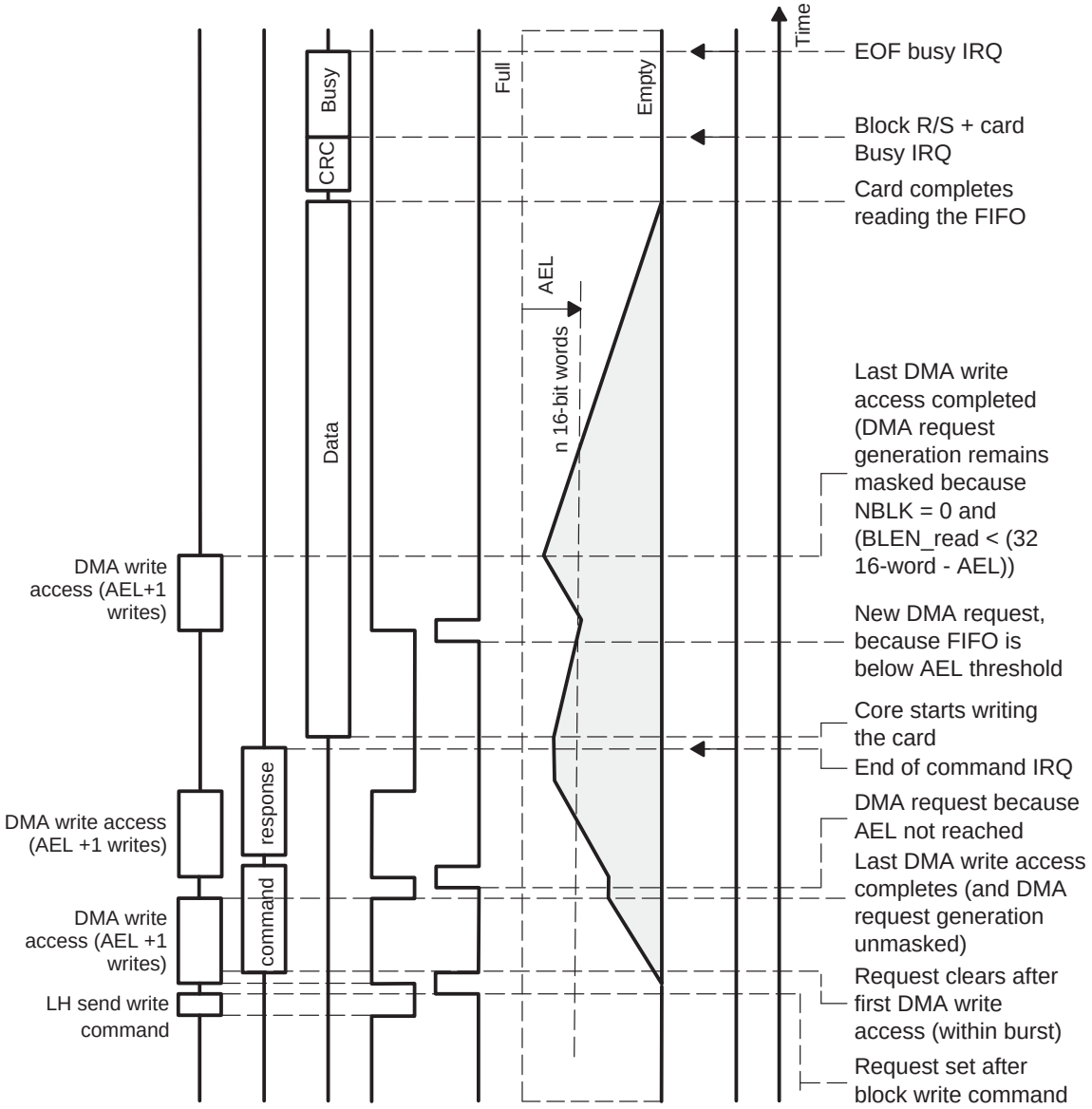
Example: Multiple block write of 10 blocks of 512 bytes each

The DMA transfer size n can be set to 16 16-bit words (32 bytes) and MMC_BUF[AEL] = 0xF. The write transfer operation completes after 160 system DMA TX requests.

The transmit FIFO never underflows. If the FIFO is emptied, the MMC.CLK clock signal is momentarily stopped until the system DMA or the local host performs a write access, which starts filling the FIFO.

Figure 20 shows a typical DMA write operation to an MMC card.

Figure 20. MMC Mode DMA TX Transfer



Signals are represented in a symbolic form (a high level for the DMA mask or the request signals denotes an ON condition).

4.3 SDIO Suspend/Resume

Suspend/resume only apply to multiple-block read or write operations.

To suspend a function, the local host must:

1) Save MMC_CMD[DDIR] value.
2) Set MMC_IOSR[SUSP] to 1.
3) Send the suspend command (CMD52) and check the SDIO card status for acknowledgment. Repeat the command as long as necessary.
4) Set MMC_IOSR[STOP] to 1 and read MMC_IOSR[STOP] until 0.
5) Disable DMA channel if enabled.
6) Save MMC_NBLK register.
7) Set MMC_IOSR[SAVE] bit to 1 and empty the FIFO by reading MMC_DATA register until empty (signaled by MMC_STAT[AF] = 0.
8) Clear MMC_IOSR[SUSP] and MMC_IOSR[SAVE] bits to 0.

A new command can be sent to another function at this point.

To resume the suspended function, the local host must:

1) Restore the FIFO contents by writing into MMC_DATA register.
2) Restore MMC_NBLK register with the saved value.
3) Reenable DMA channel if in DMA mode.
4) Set MMC_IOSR[RESU] bit to 1.
5) Send the resume command (CMD52) with MMC_CMD[DDIR] set according to the suspended function (needed to resume the function as a multiple-block read or as a multiple-block write)
6) Clear MMC_IOSR[RESU] bit to 0.

Table 37 contains information for programming the CMD register.

Table 37. Programming Aid for CMD Register (MMC_CMD)

	DDIR	SHR	TYPE	BUSY	Resp	Hex. Value (MSB of MMC.CMD/ SPI.DO)
bc: no resp	0	0	00	0	000	0x00
bcr: R3	0	0	01	0	011	0x13
bcr – R6	0	0	01	0	110	0x16
bcr: R2	0	0	01	0	010	0x12
bcr: R5	0	0	01	0	101	0x15
ac: R1	0	0	10	0	001	0x21
ac: R1b	0	0	10	1	001	0x29
ac: R2	0	0	10	0	010	0x22
ac: no resp	0	0	10	0	000	0x20
ac: R4	0	0	10	0	100	0x24
adtc: R1	1	0	11	0	001	0xB1
(no stream)	0					0x31
adtc: R1b	1	0	11	1	001	0xB9
(no stream)	0					0x39
adtc: R1	1	1	11	0	001	0xF1
(stream)	0					0x71
adtc: R1b	1	1	11	1	001	0xF9
(stream)	0					0x79
Host response	0	1	00	0	000	0x40

4.4 Programming Model Incompatibility

Software developed for the previous WMU_020_1 MMC/SD host controller version is compatible with this version except for the listed changes:

- MMC_CMD[INAB]: The previous core, send 80-clock initialization sequence followed by a command. This core only sends 80 clocks without any commands.

- MMC_CON[MODE]: The previous core defines an SPI mode 2 operation for SPI operated MMC/SD/SDIO cards on top of the SPI mode 1 operation for serial flash card. This is now replaced by a single SPI operation with a new SPI control bit MMC_SPI[CSEL].
- MMC_CMD[SHR]: The previous core did not send the host response in open-drain mode. The new core does, so it must be performed at the proper device identification speed.
- MMC_BUF[AFL]: The previous core defines a reset value of 0x1F. The reset value is 0x00 for this core.

Index

OMAP5912 Multimedia Processor Keyboard Interface Reference Guide

Literature Number: SPRU766A
March 2004

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.

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
		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 document describes the keyboard interface of the OMAP5912 multimedia processor.

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.

Related Documentation From Texas Instruments

The following documents describe the OMAP5910 device and related peripherals. Copies of these documents are available on the Internet at www.ti.com. *Tip:* Enter the literature number in the search box provided at www.ti.com.

OMAP5912 Multimedia Processor Device Overview and Architecture Reference Guide (literature number SPRU748) introduces the setup, components, and features of the OMAP5912 multimedia processor and provides a high-level view of the device architecture.

OMAP5912 Multimedia Processor OMAP 3.2 Subsystem Reference Guide (literature number SPRU749) introduces and briefly defines the main features of the OMAP3.2 subsystem of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor DSP Sybsystem Reference Guide (literature number SPRU750) describes the OMAP5912 multimedia processor DSP subsystem. The digital signal processor (DSP) subsystem is built around a core processor and peripherals that interface with: 1) The ARM926EJS via the microprocessor unit interface (MPUI); 2) Various standard memories via the external memory interface (EMIF); 3) Various system peripherals via the TI peripheral bus (TIPB) bridge.

OMAP5912 Multimedia Processor Clocks Reference Guide (literature number SPRU751) describes the clocking mechanisms of the OMAP5912 multimedia processor. In OMAP5912, various clocks are created from special components such as the digital phase locked loop (DPLL) and the analog phase-locked loop (APLL).

OMAP5912 Multimedia Processor Initialization Reference Guide (literature number SPRU752) describes the reset architecture, the configuration, the initialization, and the boot ROM of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Power Management Reference Guide (literature number SPRU753) describes power management in the OMAP5912 multimedia processor. The ultralow-power device (ULPD) generates and manages clocks and reset signals to OMAP3.2 and to some peripherals. It controls chip-level power-down modes and handles chip-level wake-up events. In deep sleep mode, this module is still active to monitor wake-up events.This book describes the ULPD module and outline architecture.

OMAP5912 Multimedia Processor Security Features Reference Guide (literature number SPRU754) describes the security features of teh OMAP5912 multimedia processor. The OMAP5912 security scheme relies on the OMAP3.2 secure mode. The distributed security on the OMAP3.2 platform is a Texas Instruments solution to address m-commerce and security issues within a mobile phone environment. The OMAP3.2 secure mode was developed to bring hardware robustness to the overall OMAP5912 security scheme.

OMAP5912 Multimedia Processor Direct Memory Access (DMA) Support Reference Guide (literature number SPRU755) describes the direct memory access support of the OMAP5912 multimedia processor. The OMAP5912 processor has three DMAs:

- The system DMA is embedded in OMAP3.2. It handles DMA transfers associated with MPU and shared peripherals.
- The DSP DMA is embedded in OMAP3.2. It handles DMA transfers associated with DSP peripherals.
- The generic distributed DMA (GDD) is an OMAP5912 resource attached to the SSI peripheral. It handles only DMA transfers associated with the SSI peripheral.

OMAP5912 Multimedia Processor Memory Interfaces Reference Guide (literature number SPRU756) describes the memory interfaces of the OMAP5912 multimedia processor.

- SDRAM (external memory interface fast, or EMIFF)
- Asynchronous and synchronous burst memory (external memory interface slow, or EMIFS)
- NAND flash (hardware controller or software controller)
- CompactFlash on EMIFS interface
- Internal static RAM

OMAP5912 Multimedia Processor Interrupts Reference Guide (literature number SPRU757) describes the interrupts of the OMAP5912 multimedia processor. Three level 2 interrupt controllers are used in OMAP5912:

- One MPU level 2 interrupt handler (also referred to as MPU interrupt level 2) is implemented outside of OMAP3.2 and can handle 128 interrupts.
- One DSP level 2 interrupt handler (also referred to as DSP interrupt level 2.1) is instantiated outside of OMAP3.2 and can handle 64 interrupts.
- One OMAP3.2 DSP level 2 interrupt handler (referenced as DSP interrupt level 2.0) can handle 16 interrupts.

OMAP5912 Multimedia Processor Peripheral Interconnects Reference Guide (literature number SPRU758) describes various periperal interconnects of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Timers Reference Guide (literature number SPRU759) describes various timers of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Serial Interfaces Reference Guide (literature number SPRU760) describes the serial interfaces of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Universal Serial Bus (USB) Reference Guide (literature number SPRU761) describes the universal serial bus (USB) host on the OMAP5912 multimedia processor. The OMAP5912 processor provides several varieties of USB functionality. Flexible multiplexing of signals from the OMAP5912 USB host controller, the OMAP5912 USB function controller, and other OMAP5912 peripherals allow a wide variety of system-level USB capabilities. Many of the OMAP5912 pins can be used for USB-related signals or for signals from other OMAP5912 peripherals. The OMAP5912 top-level pin multiplexing

controls each pin individually to select one of several possible internal pin signal interconnections. When these shared pins are programmed for use as USB signals, the OMAP5912 USB signal multiplexing selects how the signals associated with the three OMAP5912 USB host ports and the OMAP5912 USB function controller can be brought out to OMAP5912 pins.

OMAP5912 Multimedia Processor Multi-channel Buffered Serial Ports (McBSPs) Reference Guide (literature number SPRU762) describes the three multi-channel buffered serial ports (McBSPs) available on the OMAP5912 device. The OMAP5912 device provides multiple high-speed multichannel buffered serial ports (McBSPs) that allow direct interface to codecs and other devices in a system.

OMAP5912 Multimedia Processor Camera Interface Reference Guide (literature number SPRU763) describes two camera inerfaces implemented in the OMAP5912 multimedia processor: compact serial camera port and camera parallel interface.

OMAP5912 Multimedia Processor Display Interface Reference Guide (literature number SPRU764) describes the display interface of the OMAP5912 multimedia processor.

- LCD module
- LCD data conversion module
- LED pulse generator
- Display interface

OMAP5912 Multimedia Processor Multimedia Card (MMC/SD/SDIO) (literature number SPRU765) describes the multimedia card (MMC) interface of the OMAP5912 multimedia processor. The multimedia card/secure data/secure digital IO (MMC/SD/SDIO) host controller provides an interface between a local host, such as a microprocessor unit (MPU) or digital signal processor (DSP), and either an MMC or SD memory card, plus up to four serial flash cards. The host controller handles MMC/SD/SDIO or serial port interface (SPI) transactions with minimal local host intervention.

OMAP5912 Multimedia Processor Keyboard Interface Reference Guide (literature number SPRU766) describes the keyboard interface of the OMAP5912 multimedia processor. The MPUIO module enables direct I/O communication between the MPU (through the public TIPB) and external devices. Two types of I/O can be used: specific I/Os dedicated for 8 x 8 keyboard connection, and general-purpose I/Os.

OMAP5912 Multimedia Processor General-Purpose Interface Reference Guide (literature number SPRU767) describes the general-purpose in-

terface of the OMAP5912 multimedia processor. There are four GPIO modules in the OMAP5912. Each GPIO peripheral controls 16 dedicated pins configurable either as input or output for general purposes. Each pin has an independent control direction set by a programmable register. The two−edge control registers configure events (rising edge, falling edge, or both edges) on an input pin to trigger interrupts or wake−up requests (depending on the system mode). In addition, an interrupt mask register masks out specified pins. Finally, the GPIO peripherals provide the set and clear capabilities on the data output registers and the interrupt mask registers. After detection, all event sources are merged and a single synchronous interrupt (per module) is generated in active mode, whereas a unique wake−up line is issued in idle mode. Eight data output lines of the GPIO3 are ORed together to generate a global output line at the OMAP5912 boundary. This global output line can be used in conjunction with the SSI to provide a CMT−APE interface to the OMAP5912.

OMAP5912 Multimedia Processor VLYNQ Serial Communications Interface Reference Guide (literature number SPRU768) describes the VLYNQ of the OMAP5912 multimedia processor.

VLYNQ is a serial communications interface that enables the extension of an internal bus segment to one or more external physical devices. The external devices are mapped into local, physical address space and appear as if they are on the internal bus of the OMAP 5912. The external devices must also have a VLYNQ interface. The VLYNQ module serializes bus transactions in one device, transfers the serialized data between devices via a VLYNQ port, and de-serializes the transaction in the external device.

OMAP5912 includes one VLYNQ module connected on OCPT2 target port and OCPI initiator port. These connections are configured via a static switch, which selects either SSI or VLYNQ module. This switch, forbids the simultaneous use of GDD/SSI and VLYNQ. The switch is controlled by the VLYNQ_EN bit in the OMAP5912 configuration control register (CONF_5912_CTRL).

OMAP5912 Multimedia Processor Pinout Reference Guide (literature number SPRU769) provides the pinout of the OMAP5912 multimedia processor. After power-up reset, the user can change the configuration of the default interfaces. If another interface is available on top of the default, it is possible to enable a new interface for each ball by setting the corresponding 3-bit field of the associated FUNC_MUX_CTRL register. It is also possible to configure on-chip pullup/pulldown. This document

also describes the various power domains so that the user can apply the different interfaces seamlessly with external components.

OMAP5912 Multimedia Processor Window Tracer (WT) Reference Guide (literature number SPRU770) describes the window tracer module used to capture the memory transactions from four interfaces: EMIFF, EMIFS, OCP-T1, and OCP-T2. This module is located in the OMAP3.2 traffic controller (TC).

OMAP5912 Multimedia Processor Real-Time Clock Reference Guide (literature number SPRUxxx) describes the real-time clock of the OMAP5912 multimedia processor. The real-time clock (RTC) block is an embedded real-time clock module directly accessible from the TIPB bus interface.

Trademarks

Contents

Figures

Tables

Keyboard Interface

This document describes the keyboard interface of the OMAP5912 multimedia processor.

1 MPUIO

The MPUIO module enables direct I/O communication between the MPU (through the public TIPB) and external devices (see Figure 1).

Two types of I/O can be used:

- Specific I/Os dedicated for 8 x 8 keyboard connection:
  - Eight inputs (KB.R[7:0]) for row lines
  - Eight outputs (KB.C[7:0]) for column lines

  The keyboard feature allows communication with a keyboard. The MPUIO supports keyboards with up to eight rows and eight columns and has the capability to detect multiple key presses. A keyboard event is signaled to the host by an interrupt.

- General-purpose I/Os:
  - Five MPUIO signals (5, 4, 3, 2, and 1) are available in the default OMAP multiplexing.
  - Eleven additional MPUIO signals (15, 14, 13, 12, 11, 10, 9, 8, 7, 6, and 0) can be used by configuring the OMAP multiplexing.

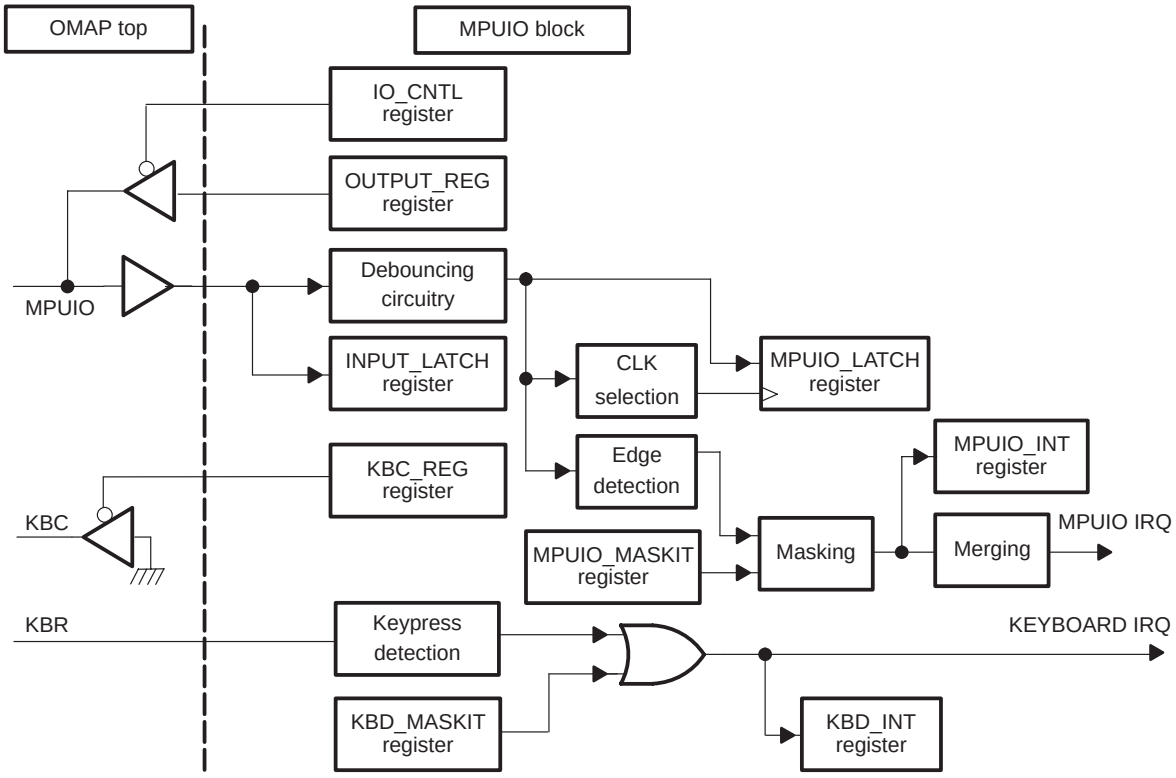
  The MPUIO feature allows communication with an external device through as many as 16 MPUIOs. These MPUIOs can be configured on a pin-by-pin basis as inputs or outputs. When configured as input, each MPUIO can be individually selected to generate interrupts on a level change (rising or falling edge). All the MPUIO inputs can be latched on this event.

  A simple filtering circuitry is implemented on the MPUIO input to allow de-bouncing.

The MPUIO module functional clock domain is clocked by the OMAP 32-kHz clock. This clock is always fed into the block, regardless of the state of the chip (awake, asleep, or idle). This allows external event latching and interrupt generation even when the system is in idle mode, to wake up the system via interrupt.

The MPUIO module interfaces with the host through a TIPB bus. The MPU peripheral clock resynchronizes register access to the module and avoids time-out on the TIPB bus caused by the functional clock being too slow.

Figure 1. MPUIO Module



1.1 MPUIO Function

The MPUIO module function allows:

- Individual configuration of 16 pins as input or output
- Capture of the 16-pin MPUIO bus on an edge of any one of the MPUIOs (rising- or falling-edge sensitivity can be chosen). The MPUIO selected to trigger the capture is the MPUIO_CLK.
- Interruption generation on the selected edge of any MPUIO. The interrupt generated is low-level sensitive, that is, it stays asserted low until cleared or masked.

1.1.1 MPUIO Debouncing

When MPUIO pins are configured in input, they can be connected to an external mechanical module. Debouncing circuitry is added on each MPUIO input. This circuitry allows an MPUIO pulse shorter than a given time to be ignored. Thus, bounce oscillations can be filtered out.

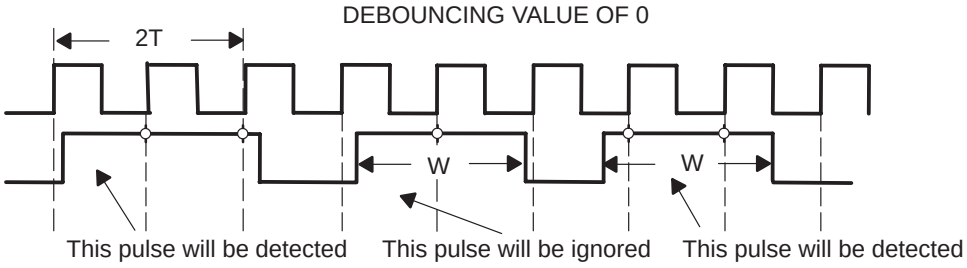
The debouncing time is programmed through the MPUIO_DEBOUNCING_REG register. It is the same for all MPUIO pins. When reference is made in all subsequent chapters to an event on an MPUIO input, it must be understood as an event after debouncing.

Because of internal resynchronization, the incoming MPUIO input must be sampled (DEBOUNCING_VALUE+2) times on the 32-kHz clock to be taken into account. This means that:

- Any pulse shorter than (DEBOUNCING_VALUE+1) 32-kHz clock periods is ignored.
- Any pulse longer than (DEBOUNCING_VALUE+2) 32-kHz clock periods is taken into account.
- Any pulse between these two durations is detected or not according to whether it can be sampled (DEBOUNCING_VALUE+2) times (see Figure 2).

This implies that any incoming MPUIO pulse shorter than two 32-kHz clock periods can be ignored even if debouncing is turned off (debouncing value equals 0).

Figure 2. Debouncing Value



1.2 Interrupt Handling

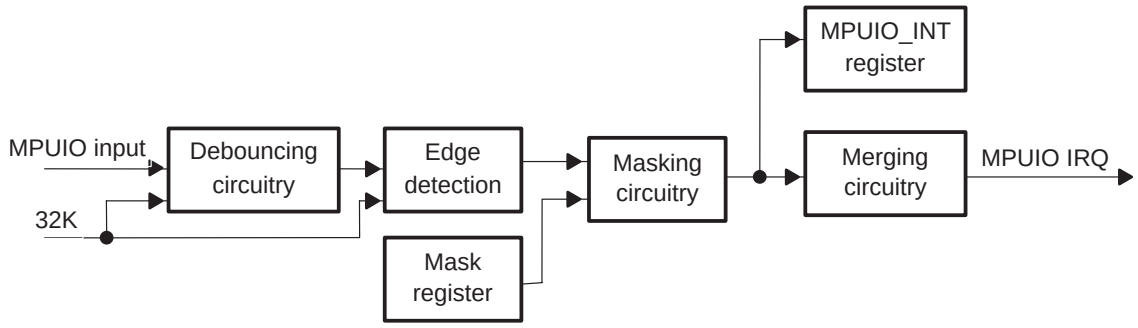
1.2.1 Interrupt Generation

Each MPUIO can be configured to generate an interrupt, either on the rising or falling edge, by unmasking interrupt generation in the MPUIO_MASKIT register. Edge sensitivity is programmed in MPUIO_INT_EDGE_REG.

On the specified edge of an incoming MPUlO, the event is registered in the MPUIO_INT register. If the interrupt generation for this MPUIO is enabled (unmasked), the MPUIO generates a low-level-sensitive interrupt to the host. Reading the MPUIO_INT register immediately resets all the pending interrupt bits and deasserts MPUIO interrupt.

Interrupt masking occurs after storage into the MPUIO interrupt register, so unmasking a pending interrupt immediatly asserts the MPUIOS_INT signal (see Figure 3).

Figure 3. MPUIOS_INT Generation



The debouncing filtering induces a delay between GPIO incoming pulse to the interrupt generation. The induced delay is more than (GPIO_DEBOUNCING_REG + 3) and less than (GPIO_DEBOUNCING_REG + 4) 32-kHz clock cycles.

For example, assuming a debouncing value of 0x000, the propagation time of an interrupt (from GPIO input edge to GPIOS_INT assertion) is between three and four 32-kHz clock cycles (between 93.75 and 125 μs).

For high debouncing values, this delay can be quite large. Therefore, when a GPIO interrupt is masked out, the debouncing value is set to 0 (GPIO_DEBOUNCING_REG is assumed to be 0) for this GPIO only. This feature ensures low-interrupt latency when an interrupt is masked and then unmasked, for slowly bouncing external devices.

1.2.2 Interrupt Acknowledgement

Acknowledging an MPUIO interrupt is done by reading the MPUIO_INT register. This immediately resets the active bits in this register and deasserts the MPUIO interrupt.

However, owing to internal reset resynchronization, the MPUIO lines being reset are not able to generate interrupts before the next 32-kHz clock rising edge. Reset of interrupts is asynchronously asserted but released synchronously with 32-kHz clock.

1.3 Interrupt Masking

Each incoming MPUIO line can be individually masked through the MPUIO_MASKIT register. When an MPUIO is masked, it does not generate an interrupt to the host.

Masking is done asynchronously, so masking all incoming MPUIOs immediately deasserts the MPUIOS_INT interrupt, if it was asserted.

When an MPUIO is masked, its debouncing value is forced to zero. This means the correct edge on this MPUIO generates an update of the MPUIO_INT register after a maximum of six 32-kHz clock cycles. As a side effect, no filtering is done on this MPUIO.

If an MPUIO can generate interrupts at a higher frequency than $\frac{CLK_{32kHz}}{6}$, masking and then unmasking it dynamically can result in interrupt loss.

1.4 MPUIO Event Capture

The MPUIO event capture allows latching the input value present on the MPUIO ports each time a rising or a falling edge occurs on a selected MPUIO port, here called MPUIO_CLK (see Figure 4).

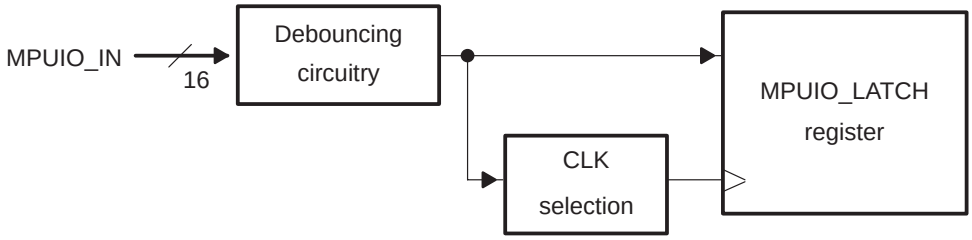
In all respects, MPUIO_CLK behaves as a regular MPUIO. It is debounced as all the others, and, if not masked, the MPUIO_CLK selected edge normally generates an interrupt to the host.

The MPUIO_EVENT_MODE_REG register enables or disables the MPUIO event capture mode. It also selects the external pin used as the MPUIO_CLK.

The MPUIO_INT_EDGE_REG bit that corresponds to the MPUIO_CLK determines which edge of MPUIO_CLK is used for capture.

On the MPUIO_CLK programmed edge, after the debouncing delay, the internal ARMIO_IN bus is latched in the MPUIO_LATCH_REG register. Its value can be read after the detection of the interrupt even if the external value has changed.

Figure 4. MPUIO Event Capture Mode



1.5 Keyboard Function

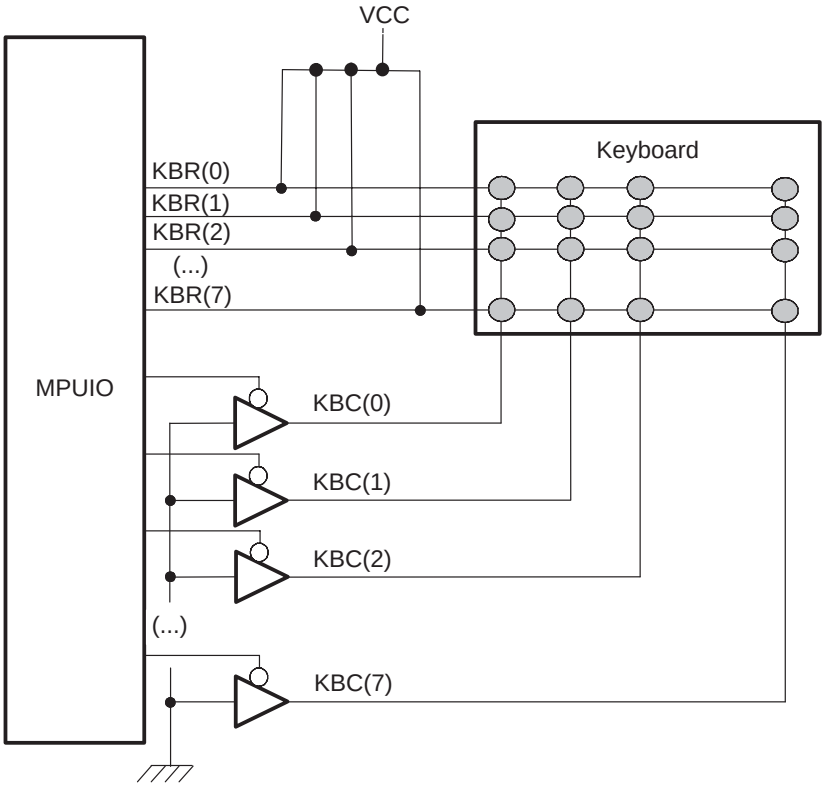
The keyboard is connected to the chip (see Figure 5) using:

- KBR(7:0) input pins for row lines
- KBC(7:0) output pins for column lines

All input pins (KBR) are externally pulled up to VCC.

By default, all output pins (KBC) drive a low level.

Figure 5. Keyboard Connection to MPUIO Module



1.6 Keyboard Scanning

If a key of the keyboard matrix is pressed, the corresponding row and column lines are shorted together, which causes a low level to be input on one of the KBR pins. This generates an interrupt to the host. On receiving the interrupt, the host must scan the column lines according to the sequence shown in Table 1.

Table 1. Keyboard Scanning Sequence for Four Pressed Keys

Pin	Idle	Keyboard Scanning†									Idle
KBC(0)	0	Z	0	Z	Z	Z	Z	Z	Z	Z	0
KBC(1)	0	Z	Z	0	Z	Z	Z	Z	Z	Z	0
KBC(2)	0	Z	Z	Z	0	Z	Z	Z	Z	Z	0
KBC(3)	0	Z	Z	Z	Z	0	Z	Z	Z	Z	0
KBC(4)	0	Z	Z	Z	Z	Z	0	Z	Z	Z	0
KBC(5)	0	Z	Z	Z	Z	Z	Z	0	Z	Z	0
KBC(6)	0	Z	Z	Z	Z	Z	Z	Z	0	Z	0
KBC(7)	0	Z	Z	Z	Z	Z	Z	Z	Z	0	0

† Z means the output is in high impedance state (1 is written in the KBC_REG corresponding bit).

This sequence is written to allow detection of simultaneous press actions on several buttons. It ensures that any combination of pressed keys can be read, if proper hardware protection has been implemented to prevent ghostkey detection.

Writing a pattern on KBC is done by writing into the KBC_REG register. The KBR inputs are directly accessed by reading the KBR_LATCH register.

See Table 2 for values read on the KBR lines during the scanning sequence if keys at (row, column) (3,3), (3,5), (5,3), and (7,7) are simultaneously pressed.

Table 2. Keyboard Scanning Result for Four Pressed Keys

Pin	Idle	Keyboard Scanning									Idle
KBR(0)	1	1	1	1	1	1	1	1	1	1	1
KBR(1)	1	1	1	1	1	1	1	1	1	1	1
KBR(2)	1	1	1	1	1	1	1	1	1	1	1
KBR(3)	0	1	1	1	1	0†	1	0‡	1	1	0

Table 2. Keyboard Scanning Result for Four Pressed Keys (Continued)

Pin	Idle	Keyboard Scanning									Idle
KBR(4)	1	1	1	1	1	1	1	1	1	1	1
KBR(5)	0	1	1	1	1	0†	1	1	1	1	0
KBR(6)	1	1	1	1	1	1	1	1	1	1	1
KBR(7)	0	1	1	1	1	1	1	1	1	0§	0

† KBC(3) drives 0. As keys (3,3) and (5,3) are pressed, KBR(3) and KBR(5) read as zero.
‡ KBC(5) drives 0. As key (3,5) is pressed, KBR(3) reads as zero.
§ KBC(7) drives 0. As key (7,7) is pressed, KBR(7) reads as zero.

The keyboard interrupt is falling-edge sensitive (the interrupt controller must detect a falling edge on this line).

2 MPUIO Registers

Table 3 lists the 16-bit MPUIO registers. Table 4 through Table 16 describe the individual registers.

Table 3. MPU Input/Output Registers

Start Address in the MPUIO range (hex): FFFB:5000			
Register	**Description**	**R/W**	**Offset**
INPUT_LATCH	General-purpose input	R	0x00
OUTPUT_REG	General-purpose output	R/W	0x04
IO_CNTL	Input/output control	R/W	0x08
KBR_LATCH	Keyboard row inputs	R	0x10
KBC_REG	Keyboard column outputs	R/W	0x14
MPUIO_EVENT_MODE_REG	MPUIO event mode	R/W	0x18
MPUIO_INT_EDGE_REG	MPUIO interrupt edge	R/W	0x1C
KBD_INT	Keyboard interrupt	R	0x20
MPUIO_INT	MPUIO interrupt	R	0x24
KBD_MASKIT	Keyboard mask interrupt	R/W	0x28
MPUIO_MASKIT	MPUIO mask interrupt	R/W	0x2C
MPUIO_DEBOUNCING_REG	MPUIO debouncing	R/W	0x30
MPUIO_LATCH_REG	MPUIO latch	R	0x34

Table 4. General-Purpose Input Register (INPUT_LATCH)

Base Address = 0xFFFB 5000, Offset Address = 0x00			
Bit	**Name**	**Function**	**Reset**
15:0	INPUT_LATCH	General-purpose inputs	0x0000

Table 5. Output Register (OUTPUT_REG)

Base Address = 0xFFFB 5000, Offset Address = 0x04			
Bit	**Name**	**Function**	**Reset**
15:0	OUTPUT_REG	General-purpose outputs	0x0000

Table 6. Input/Output Control Register (IO_CNTL)

Base Address = 0xFFFB 5000, Offset Address = 0x08				
Bit	**Name**	**Value**	**Function**	**Reset**
15:0	IO_CNTL		In/out control for general-purpose I/O	0xFFFF
		0	I/O is configured as output.	
		1	I/O is configured as input.	

Table 7. Keyboard Row Inputs Register (KBR_LATCH)

Base Address = 0xFFFB 5000, Offset Address = 0x10			
Bit	**Name**	**Function**	**Reset**
15:8	Reserved		0xFF
7:0	KBR_LATCH	Keyboard row inputs	0xFF

Table 8. Keyboard Column Outputs Register (KBC_REG)

Base Address = 0xFFFB 5000, Offset Address = 0x14				
Bit	**Name**	**Value**	**Function**	**Reset**
15:8	Reserved			0xFF
7:0	KBC_REG		Keyboard column outputs	0x00
		0	Output drives a low level.	
		1	Output is in high-impedance state.	

Table 9. MPUIO Event Mode Register (MPUIO_EVENT_MODE_REG)

Base Address = 0xFFFB 5000, Offset Address = 0x18				
Bit	**Name**	**Value**	**Function**	**Reset**
15:5	Reserved			0x7FF
4:1	PIN_SELECT		Select MPUIO_IN[15:0] pin to be the MPUIO_CLK event	0x0
		0000	Pin 0	
		0001	Pin 1	
		. . .		
		1111	Pin 15	
0	SET_MPUIO_EVENT_MODE	0	MPUIO event mode disable	0x0
		1	MPUIO event mode enable	

Table 10. MPUIO Interrupt Edge Register (MPUIO_INT_EDGE_REG)

Base Address = 0xFFFB 5000, Offset Address = 0x1C				
Bit	**Name**	**Value**	**Function**	**Reset**
15:0	EDGE_SELECT		Set interrupt on falling/rising edge	0x0
		0	Falling edge	
		1	Rising edge	

Table 11. Keyboard Interrupt Register (KBD _INT)

Base Address = 0xFFFB 5000, Offset Address = 0x20				
Bit	**Name**	**Value**	**Function**	**Reset**
15:1	Reserved			0x7FFF
0	KBD_INT		Keyboard interrupt (active low)	0x1
		0	Interrupt occurred	
		1	No interrupt pending	

Note: KBD_INT is a status bit only (duplication of the level of the corresponding interrupt signal).

Table 12. MPUIO Interrupt Register (MPUIO_INT)

Base Address = 0xFFFB 5000, Offset Address = 0x24			
Bit	**Name**	**Function**	**Reset**
15:0	MPUIO_INT	MPUIO interrupts (active high)	0x0

Note: MPUIO_INT is reset on read access to the MPUIO_INT register. The value read is the value after mask application.

Even in emulation mode, the MPUIO interrupts are reset by a read in the MPUIO interrupt register (MPUIO_INT).

Table 13. Keyboard Mask Interrupt Register (KBD_ MASKIT)

Base Address = 0xFFFB 5000, Offset Address = 0x28			
Bit	**Name**	**Function**	**Reset**
15:1	Reserved		0x7FFF
0	KBD_MASKIT	Mask is active at level 1, inactive at level 0.	0x0

Table 14. MPUIO Mask Interrupt Register (MPUIO_MASKIT)

Base Address = 0xFFFB 5000, Offset Address = 0x2C			
Bit	**Name**	**Function**	**Reset**
15:0	MPUIO_MASKIT[15:0]	Mask is active at level 1, inactive at level 0.	0x0

Table 15. MPUIO Debouncing Register (MPUIO_DEBOUNCING_REG)

Base Address = 0xFFFB 5000, Offset Address = 0x30				
Bit	**Name**	**Value**	**Function**	**Reset**
15:9	Reserved			0x7F
8:0	MPUIO_ DEBOUNCING_REG	0: 0−511 cycles of T32k debouncing time. Programming step is T32k = 32.5 µs.		0x0

Note: MPUIO_DEBOUNCING_REG ignores pulse widths less than 32 kHz. For example, when T_{32k}=31.25 µs, a value of 0x000 filters nothing, whereas a value of 0x100 ignores pulses smaller than 8 ms. *The MPUIO accommodates values between 0x000 and 0x100. If a greater value is set into the MPUIO_DEBOUNCING_REG register, the module behavior is not assured.*

Table 16. MPUIO Latch Register (MPUIO_LATCH_REG)

Base Address = 0xFFFB 5000, Offset Address = 0x34			
Bit	**Name**	**Function**	**Reset**
15:0	MPUIO_LATCH_REG	After debouncing time, the MPUIO_IN bus is latched in this register.	0x0

Index

OMAP5912 Multimedia Processor General-Purpose Interface Reference Guide

Literature Number: SPRU767A
March 2004

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.

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
		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 document describes the general-purpose interface of the OMAP5912 multimedia processor.

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.

Related Documentation From Texas Instruments

The following documents describe the OMAP5910 device and related peripherals. Copies of these documents are available on the Internet at www.ti.com. *Tip:* Enter the literature number in the search box provided at www.ti.com.

OMAP5912 Multimedia Processor Device Overview and Architecture Reference Guide (literature number SPRU748) introduces the setup, components, and features of the OMAP5912 multimedia processor and provides a high-level view of the device architecture.

OMAP5912 Multimedia Processor OMAP 3.2 Subsystem Reference Guide (literature number SPRU749) introduces and briefly defines the main features of the OMAP3.2 subsystem of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor DSP Sybsystem Reference Guide (literature number SPRU750) describes the OMAP5912 multimedia processor DSP subsystem. The digital signal processor (DSP) subsystem is built around a core processor and peripherals that interface with: 1) The ARM926EJS via the microprocessor unit interface (MPUI); 2) Various standard memories via the external memory interface (EMIF); 3) Various system peripherals via the TI peripheral bus (TIPB) bridge.

OMAP5912 Multimedia Processor Clocks Reference Guide (literature number SPRU751) describes the clocking mechanisms of the OMAP5912 multimedia processor. In OMAP5912, various clocks are created from special components such as the digital phase locked loop (DPLL) and the analog phase-locked loop (APLL).

OMAP5912 Multimedia Processor Initialization Reference Guide (literature number SPRU752) describes the reset architecture, the configuration, the initialization, and the boot ROM of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Power Management Reference Guide (literature number SPRU753) describes power management in the OMAP5912 multimedia processor. The ultralow-power device (ULPD) generates and manages clocks and reset signals to OMAP3.2 and to some peripherals. It controls chip-level power-down modes and handles chip-level wake-up events. In deep sleep mode, this module is still active to monitor wake-up events.This book describes the ULPD module and outline architecture.

OMAP5912 Multimedia Processor Security Features Reference Guide (literature number SPRU754) describes the security features of teh OMAP5912 multimedia processor. The OMAP5912 security scheme relies on the OMAP3.2 secure mode. The distributed security on the OMAP3.2 platform is a Texas Instruments solution to address m-commerce and security issues within a mobile phone environment. The OMAP3.2 secure mode was developed to bring hardware robustness to the overall OMAP5912 security scheme.

OMAP5912 Multimedia Processor Direct Memory Access (DMA) Support Reference Guide (literature number SPRU755) describes the direct memory access support of the OMAP5912 multimedia processor. The OMAP5912 processor has three DMAs:

- The system DMA is embedded in OMAP3.2. It handles DMA transfers associated with MPU and shared peripherals.
- The DSP DMA is embedded in OMAP3.2. It handles DMA transfers associated with DSP peripherals.
- The generic distributed DMA (GDD) is an OMAP5912 resource attached to the SSI peripheral. It handles only DMA transfers associated with the SSI peripheral.

OMAP5912 Multimedia Processor Memory Interfaces Reference Guide (literature number SPRU756) describes the memory interfaces of the OMAP5912 multimedia processor.

- SDRAM (external memory interface fast, or EMIFF)
- Asynchronous and synchronous burst memory (external memory interface slow, or EMIFS)
- NAND flash (hardware controller or software controller)
- CompactFlash on EMIFS interface
- Internal static RAM

OMAP5912 Multimedia Processor Interrupts Reference Guide (literature number SPRU757) describes the interrupts of the OMAP5912 multimedia processor. Three level 2 interrupt controllers are used in OMAP5912:

- One MPU level 2 interrupt handler (also referred to as MPU interrupt level 2) is implemented outside of OMAP3.2 and can handle 128 interrupts.
- One DSP level 2 interrupt handler (also referred to as DSP interrupt level 2.1) is instantiated outside of OMAP3.2 and can handle 64 interrupts.
- One OMAP3.2 DSP level 2 interrupt handler (referenced as DSP interrupt level 2.0) can handle 16 interrupts.

OMAP5912 Multimedia Processor Peripheral Interconnects Reference Guide (literature number SPRU758) describes various periperal interconnects of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Timers Reference Guide (literature number SPRU759) describes various timers of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Serial Interfaces Reference Guide (literature number SPRU760) describes the serial interfaces of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Universal Serial Bus (USB) Reference Guide (literature number SPRU761) describes the universal serial bus (USB) host on the OMAP5912 multimedia processor. The OMAP5912 processor provides several varieties of USB functionality. Flexible multiplexing of signals from the OMAP5912 USB host controller, the OMAP5912 USB function controller, and other OMAP5912 peripherals allow a wide variety of system-level USB capabilities. Many of the OMAP5912 pins can be used for USB-related signals or for signals from other OMAP5912 peripherals. The OMAP5912 top-level pin multiplexing

controls each pin individually to select one of several possible internal pin signal interconnections. When these shared pins are programmed for use as USB signals, the OMAP5912 USB signal multiplexing selects how the signals associated with the three OMAP5912 USB host ports and the OMAP5912 USB function controller can be brought out to OMAP5912 pins.

OMAP5912 Multimedia Processor Multi-channel Buffered Serial Ports (McBSPs) Reference Guide (literature number SPRU762) describes the three multi-channel buffered serial ports (McBSPs) available on the OMAP5912 device. The OMAP5912 device provides multiple high-speed multichannel buffered serial ports (McBSPs) that allow direct interface to codecs and other devices in a system.

OMAP5912 Multimedia Processor Camera Interface Reference Guide (literature number SPRU763) describes two camera inerfaces implemented in the OMAP5912 multimedia processor: compact serial camera port and camera parallel interface.

OMAP5912 Multimedia Processor Display Interface Reference Guide (literature number SPRU764) describes the display interface of the OMAP5912 multimedia processor.

- LCD module
- LCD data conversion module
- LED pulse generator
- Display interface

OMAP5912 Multimedia Processor Multimedia Card (MMC/SD/SDIO) (literature number SPRU765) describes the multimedia card (MMC) interface of the OMAP5912 multimedia processor. The multimedia card/secure data/secure digital IO (MMC/SD/SDIO) host controller provides an interface between a local host, such as a microprocessor unit (MPU) or digital signal processor (DSP), and either an MMC or SD memory card, plus up to four serial flash cards. The host controller handles MMC/SD/SDIO or serial port interface (SPI) transactions with minimal local host intervention.

OMAP5912 Multimedia Processor Keyboard Interface Reference Guide (literature number SPRU766) describes the keyboard interface of the OMAP5912 multimedia processor. The MPUIO module enables direct I/O communication between the MPU (through the public TIPB) and external devices. Two types of I/O can be used: specific I/Os dedicated for 8 x 8 keyboard connection, and general-purpose I/Os.

OMAP5912 Multimedia Processor General-Purpose Interface Reference Guide (literature number SPRU767) describes the general-purpose in-

terface of the OMAP5912 multimedia processor. There are four GPIO modules in the OMAP5912. Each GPIO peripheral controls 16 dedicated pins configurable either as input or output for general purposes. Each pin has an independent control direction set by a programmable register. The two−edge control registers configure events (rising edge, falling edge, or both edges) on an input pin to trigger interrupts or wake−up requests (depending on the system mode). In addition, an interrupt mask register masks out specified pins. Finally, the GPIO peripherals provide the set and clear capabilities on the data output registers and the interrupt mask registers. After detection, all event sources are merged and a single synchronous interrupt (per module) is generated in active mode, whereas a unique wake−up line is issued in idle mode. Eight data output lines of the GPIO3 are ORed together to generate a global output line at the OMAP5912 boundary. This global output line can be used in conjunction with the SSI to provide a CMT−APE interface to the OMAP5912.

OMAP5912 Multimedia Processor VLYNQ Serial Communications Interface Reference Guide (literature number SPRU768) describes the VLYNQ of the OMAP5912 multimedia processor.

VLYNQ is a serial communications interface that enables the extension of an internal bus segment to one or more external physical devices. The external devices are mapped into local, physical address space and appear as if they are on the internal bus of the OMAP 5912. The external devices must also have a VLYNQ interface. The VLYNQ module serializes bus transactions in one device, transfers the serialized data between devices via a VLYNQ port, and de-serializes the transaction in the external device.

OMAP5912 includes one VLYNQ module connected on OCPT2 target port and OCPI initiator port. These connections are configured via a static switch, which selects either SSI or VLYNQ module. This switch, forbids the simultaneous use of GDD/SSI and VLYNQ. The switch is controlled by the VLYNQ_EN bit in the OMAP5912 configuration control register (CONF_5912_CTRL).

OMAP5912 Multimedia Processor Pinout Reference Guide (literature number SPRU769) provides the pinout of the OMAP5912 multimedia processor. After power-up reset, the user can change the configuration of the default interfaces. If another interface is available on top of the default, it is possible to enable a new interface for each ball by setting the corresponding 3-bit field of the associated FUNC_MUX_CTRL register. It is also possible to configure on-chip pullup/pulldown. This document

also describes the various power domains so that the user can apply the different interfaces seamlessly with external components.

OMAP5912 Multimedia Processor Window Tracer (WT) Reference Guide (literature number SPRU770) describes the window tracer module used to capture the memory transactions from four interfaces: EMIFF, EMIFS, OCP-T1, and OCP-T2. This module is located in the OMAP3.2 traffic controller (TC).

OMAP5912 Multimedia Processor Real-Time Clock Reference Guide (literature number SPRUxxx) describes the real-time clock of the OMAP5912 multimedia processor. The real-time clock (RTC) block is an embedded real-time clock module directly accessible from the TIPB bus interface.

Trademarks

Contents

Figures

Tables

General-Purpose Interface

This document describes the general-purpose interface of the OMAP5912 multimedia processor.

1 GPIO Peripheral

The general-purpose input/output (GPIO) peripheral can be used for the following types of applications:

- Input/output data
- Generation of an interrupt in active mode upon the detection of external events
- Generation of a wake-up request in idle mode upon the detection of external events

There are four GPIO modules in the OMAP5912 (see Figure 1).

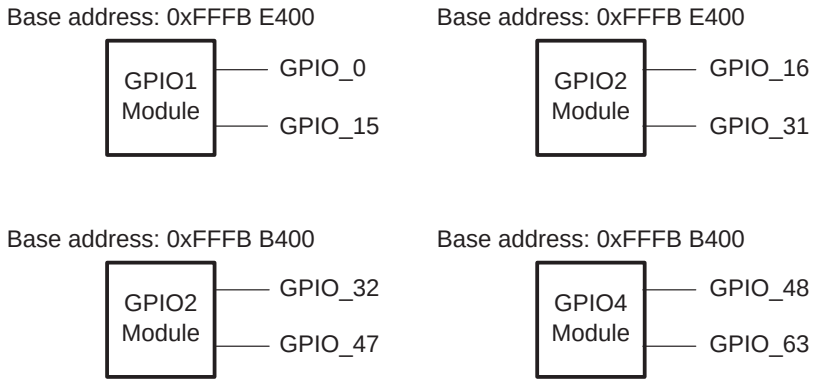
Each GPIO peripheral controls 16 dedicated pins configurable either as input or output for general purposes. Each pin has an independent control direction set by a programmable register. The two−edge control registers configure events (rising edge, falling edge, or both edges) on an input pin to trigger interrupts or wake−up requests (depending on the system mode). In addition, an interrupt mask register masks out specified pins. Finally, the GPIO peripherals provide the set and clear capabilities on the data output registers and the interrupt mask registers. After detection, all event sources are merged and a single synchronous interrupt (per module) is generated in active mode, whereas a unique wake−up line is issued in idle mode. Eight data output lines of the GPIO3 are ORed together to generate a global output line at the OMAP5912 boundary. This global output line can be used in conjunction with the SSI to provide a CMT−APE interface to the OMAP5912.

All wake−up events of each GPIO module are merged and connected to the MPU interrupt handler to enable the system wake−up. Then, the MPU is in charge.

From one event detection, the GPIO is able to generate two distinct interrupts with independent status and mask registers. One of these interrupts is mapped on the DSP interrupt handler and the other on the MPU interrupt handler.

Some GPIO lines are instanciated more than once at the OMAP5912 level to offer greater flexibility. You must ensure that not more than one GPIO line is enabled at any given time.

Figure 1. GPIO Modules



1.1 Functional Description

Each GPIO pin has an independent control direction set by a programmable register that enables the 16 GPIO pins to be configured individually either as input or output.

The GPIO has two registers to control the expected transition (rising and/or falling edge) on the input line which activates the interrupt line(s). Through these registers, the user can also disable the edge detection individually for each input GPIO.

This peripheral also provides an interrupt enable register for each synchronous interrupt line to enable or disable the source of interrupt (input GPIO). All peripheral registers are 16- or 32-bit accessible with an OCP interface compatible.

To avoid the atomic test and set usage for a register update, the peripheral offers the set and clear protocol register update.

In active mode, the edge detection is performed synchronously with the GPIO functional clock. The precision of the detection is then set by the frequency of this functional clock: the minimum pulse width on the input GPIO to trigger a synchronous interrupt request is two times the functional clock period. This minimum pulse width is required before and after any expected level transition detection.

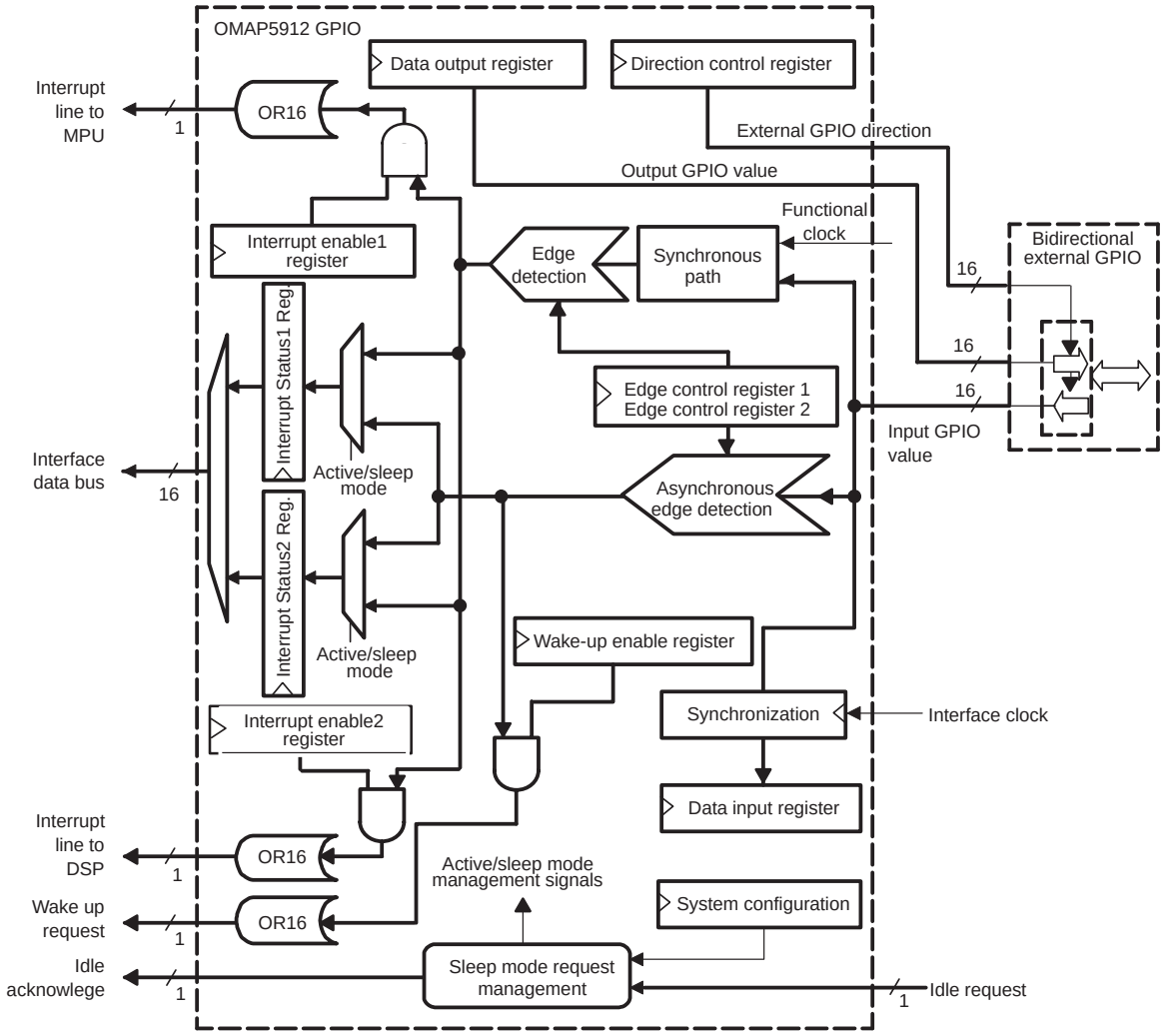
After the detection, all interrupt lines are merged together in two symmetrical paths to issue two synchronous interrupt request lines. Each interrupt line owns its dedicated interrupt status register to determine which source has activated the interrupt request. For each interrupt line, an interrupt enable register can mask the interrupt request activation without affecting the interrupt status register update in case of expected transition on one input GPIO pin.

In idle mode (the system does not provide any other functional or interface clocks), an asynchronous path is able to detect the expected event (a transition occurring on an enabled I/O configured as input) to generate an asynchronous wake-up request. A wake-up enable register enables or disables the source of the wake up. As in active mode, all wake-up sources (input GPIO) are merged together to issue one wake-up signal to the system.

Figure 2 details the GPIO block diagram with its configuration registers and main functional paths:

- The synchronous path to generate a synchronous interrupt request upon expected edge 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 (GPIO_IRQENABLE1, GPIO_IRQENABLE2).
- The asynchronous path to generate an asynchronous wake-up request upon expected edge detection on any input GPIO; the asynchronous wake-up request line is active according to the wake-up enable register (GPIO_WAKEUPENABLE).
- The logic block managing the sleep mode request/acknowledge protocol: this part enables the synchronous path in active mode and the asynchronous path in sleep mode.

Figure 2. GPIO Block Diagram



1.2 Set and Clear Instructions

To avoid using the atomic test-and-set to update the data output register (GPIO_DATAOUT) and the interrupt enable (GPIO_IRQENABLEx) or wake-up enable (GPIO_WAKEUPENABLE) registers, the GPIO peripheral offers a set-and-clear protocol. This protocol consists of writing an operation at dedicated addresses (one address for clearing bits and one address for setting bits). A written 1 clears (or sets) bits; unaffected bits are 0. See Section 1.3 for the clear instruction example and Section 1.4 for the set instruction example.

1.3 Clear Instruction Example

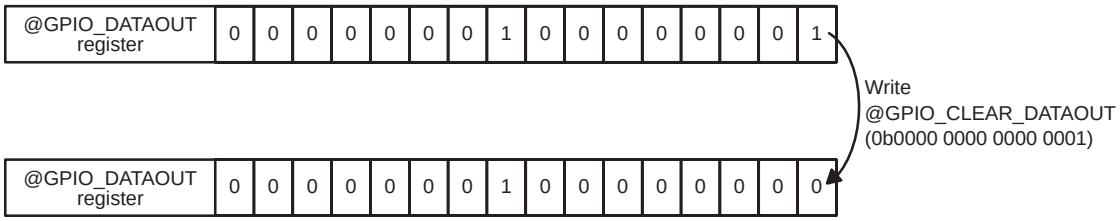
In this example, the data output register(GPIO_DATAOUT) contains the binary value 0b0000 0001 0000 0001, and the user wants to clear the bit [0].

With the clear instruction feature, the user writes 0b0000 0000 0000 0001 at the clear data output register address (GPIO_CLEAR_DATAOUT).

After this write operation, a reading of the data output register returns 0b0000 0001 0000 0000: the bit [0] has been cleared.

Similarly, the interrupt enable registers and the wakeup enable register can be cleared with GPIO_CLEAR_INTERRUPTx and GPIO_WAKEUPENABLE, respectively.

Figure 3. Write @GPIO_CLEAR_DATAOUT Register Example

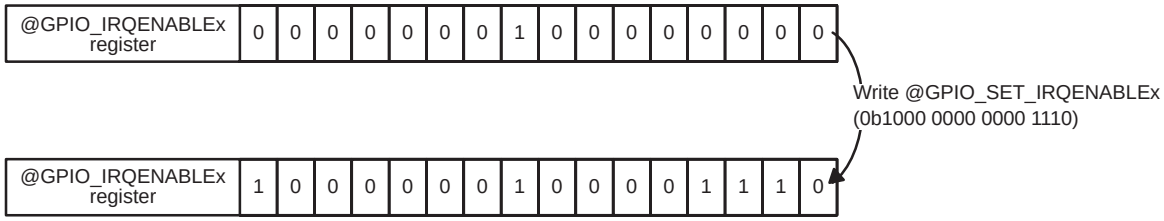


1.4 Set Instruction Example

In this example, the interrupt enable1 or 2 register contains the binary value 0b0000 0001 0000 0000, and the user wants to set the bits 15, 3, 2, and 1.

With the set instruction feature, the user writes 0b1000 0000 0000 1110 at the address of the set interrupt enable1 or 2 register (see Table 18). After this write operation, a reading of the interrupt enable1 or 2 register returns 0b1000 0001 0000 1110: the bits 15, 3, 2 , and 1 have been set.

Figure 4. Write @GPIO_SET_IRQENABLEx Register Example



Similarly, the wakeup enable register and the data output register can be set using GPIO_SET_WAKEUP and GPIO_SET_DATAOUT, respectively.

1.5 GPIO Registers

Base address (in hexadecimal):

- ❏ Bit width: 32 bits
- ❏ Supported Accesses: 32-bit and 16-bit accesses
- ❏ Address of one register is: Base address + Offset address

Table 1 lists the 32-bit GPIO registers. Table 2 through Table 20 describe the register bits.

Table 1. GPIO Registers—General Description (1 and 2)

Base Address = 0xFFFB E400, 0xFFFB EC00, 0xFFFB B400, 0xFFFB BC00		
Register	**Description**	**Offset**
GPIO_REVISION	Revision	0x0000
GPIO_SYSCONFIG	System configuration	0x0010
GPIO_SYSSTATUS	System status	0x0014
GPIO_IRQSTATUS1	Interrupt status1	0x0018
GPIO_IRQENABLE1	Interrupt enable1	0x001C
GPIO_IRQSTATUS2	Interrupt status2	0x0020
GPIO_IRQENABLE2	Interrupt enable2	0x0024
GPIO_WAKEUPENABLE	wake-up enable	0x0028
GPIO_DATAIN	Data input	0x002C
GPIO_DATAOUT	Data output	0x0030
GPIO_DIRECTION	Direction control	0x0034
GPIO_EDGE_CTRL1	Edge control 1	0x0038
GPIO_EDGE_CTRL2	Edge control 2	0x003C
GPIO_CLEAR_IRQENABLE1	Clear interrupt enable1	0x009C
GPIO_CLEAR_IRQENABLE2	Clear interrupt enable2	0x00A4
GPIO_CLEAR_WAKEUPENA	Clear wake-up enable	0x00A8
GPIO_CLEAR_DATAOUT	Clear data output	0x00B0
GPIO_SET_IRQENABLE1	Set interrupt enable1	0x00DC
GPIO_SET_IRQENABLE2	Set interrupt enable2	0x00E4

Table 1. GPIO Registers—General Description (1 and 2) (Continued)

Base Address = 0xFFFB E400, 0xFFFB EC00, 0xFFFB B400, 0xFFFB BC00		
Register	**Description**	**Offset**
GPIO_SET_WAKEUPENA	Set wake-up enable	0x00E8
GPIO_SET_DATAOUT	Set data output	0x00F0

The write latency for the R/W registers is immediate, except for the direction control and the edge control 1 and 2 registers. Any write access in these three registers is resynchronized in the functional clock domain.

Table 2. Revision Register (GPIO_REVISION)

Base Address = 0xFFFB E400, 0xFFFB EC00, 0xFFFB B400, 0xFFFB BC00, Offset Address = 0x00					
Bit	**Field Name**	**Function**		**Access (R/W)**	**Value at Reset**
31:8	Reserved	Reserved		R	0x000000
7:0	GPIO revision	Bits	RTL revision number	R	tbd
		[3:0]	Minor revision		
		[7:4]	Major revision		

This is a read-only register containing the revision number of the GPIO module. A write to this register has no affect, as the reset.

The GPIO revision 8-bit field indicates the revision number of the current module. This value is fixed by hardware.

The 4 LSBs indicate a minor revision.

The 4 MSBs indicate a major revision.

Example:

0x10 => version 1.0

This register sets various parameters that control the idle mode of the GPIO module.

The autoidle field (bit 0) sets the internal interface clock-gating strategy in active mode. The soft reset field (bit 1) resets the whole GPIO module (the same effect as the OCP hardware reset). The ENAWAKEUP field (bit 2) enables or disables the wake-up request generation upon the expected transition happening on the GPIO input pins. The idle mode field (bits [4:3]) controls the power-saving management.

Table 3. System Configuration Register (GPIO_SYSCONFIG)

Base Address = 0xFFFB E400, 0xFFFB EC00, 0xFFFB B400, 0xFFFB BC00, Offset Address = 0x10					
Bit	**Field Name**	**Function**		**R/W**	**Reset**
31:5	Reserved	Reserved		R/W	0x000...0
4:3	IDLEMODE	Value	Power management, REQ/ACK control	R/W	00
		00	Force idle. An idle request is acknowledged unconditionally.		
		01	No idle. An idle request is never acknowledged.		
		10	Smart idle. The acknowledgement to an idle request is given based on the internal activity (see Section 1.8.2).		
		11	Reserved		
2	ENAWAKEUP	0: Wake-up generation is disabled.		R/W	0
		1: Wake-up capability is enabled upon expected transition on input GPIO pin.			
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.		R/W	0
		0 : Normal mode			
		1 : The module is reset.			
0	AUTOIDLE	0: Internal interface OCP clock is free−running		R/W	0
		1: Automatic internal OCP clock gating, based on the OCP interface activity			

Table 4. System Status Register (GPIO_SYSSTATUS)

Base Address = 0xFFFB E400, 0xFFFB EC00, 0xFFFB B400, 0xFFFB BC00, Offset Address = 0x14					
Bit	**Field Name**	**Function**		**R/W**	**Reset**
31:1	Reserved	Reserved		R	0x000…0
0	RESETDONE	Value	Internal reset monitoring	R	−
		0	Internal reset is ongoing.		
		1	Reset completed		

This register provides the reset status information about the GPIO module. It is a read-only register; a write to this register has no effect.

Table 5. Interrupt Status Registers(1 and 2) (GPIO_IRQSTATUSx)

Base Address=0xFFFB E400, 0xFFFB EC00, 0xFFFB B400, 0xFFFB BC00, Offset Address=0x18, 0x20			
Bit	**Function**	**R/W**	**Reset**
31:16	Reserved	R/W	0x0000
15:0	0: No interrupt set 1: Interrupt set	R/W	0x0000

This register determines which of the input GPIO pins triggered the interrupt line 1 or 2 request, or the wake-up. As illustrated in Table 12, bit 0 corresponds to PIGPIOPINSI[1], and bit 1 to PIGPIOPINSI[2], and so on.

When a bit in this register is set to 1, it indicates that the corresponding GPIO pin is requesting the interrupt or wake up. If the user wants to reset a bit in this register, a 1 must be written to the appropriate bit. However, the user cannot generate an interrupt by writing a 1 to the interrupt status1 or 2 register. If the user writes a 0 to a bit in this register, the value remains unchanged. The interrupt status 1 or 2 register is synchronous with the interface clock. In idle mode, the event is detected via an asynchronous path, and the corresponding bit in the interrupt status 1 and 2 registers is set when the GPIO peripheral is awakened.

Table 6. Interrupt Enable Registers (1 and 2)(GPIO_IRQENABLEx)

Base Address=0xFFFB E400, 0xFFFB EC00, 0xFFFB B400, 0xFFFB BC00, Offset Address=0x1C, 0x24			
Bit	**Function**	**R/W**	**Reset**
31:16	Reserved	R/W	0x0000
15:0	0: Disabled 1: Enabled	R/W	0x0000

This register allows the user 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.

A feature enables the user to set or clear a bit of these registers with a single write access to the corresponding set interrupt enable 1 or 2 register, or to the clear interrupt enable 1 or 2 register address (see Table 15, and Table 18.)

Table 7. Wake-up Enable Register (GPIO_WAKEUPENABLE)

Base Address = 0xFFFB E400, 0xFFFB EC00, 0xFFFB B400, 0xFFFB BC00, Offset Address = 0x28			
Bit	**Function**	**R/W**	**Reset**
31:16	Reserved	R/W	0x0000
15:0	0: Wake-up generation is disabled. 1: Wake-up generation is enabled.	R/W	0x0000

Note: In force idle mode, the module wake-up feature is inhibited.

This register allows the user to mask the expected transition on input GPIO from generating a wake-up request. It is programmed synchronously with the interface clock before any idle mode request comes from the host processor.

A feature enables the user to set or clear a bit 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 1.3 and Table 16 and Table 19)

Table 8. Data Input Register (GPIO_DATAIN)

Base Address = 0xFFFB E400, 0xFFFB EC00, 0xFFFB B400, 0xFFFB BC00, Offset Address = 0x2C			
Bit	**Function**	**R/W**	**Reset**
31:16	Reserved	R	0x0000
15:0	Received data	R	0x0000

This register registers the data that is read from the GPIO pins. It is a read-only register. The input data is sampled synchronously with the interface clock and then captured in the data input register synchronously with the interface clock. Any change on the GPIO input pin is captured into this register after two interface clock cycles (seeing the required cycles to synchronize and write the data). See Figure 7. The data input register copies the PIGPIOPINSI[15:0], regardless of the GPIO_DIRECTION register.

Table 9. Data Output Register (GPIO_DATAOUT)

Base Address = 0xFFFB E400, 0xFFFB EC00, 0xFFFB B400, 0xFFFB BC00, Offset Address = 0x30			
Bit	**Function**	**R/W**	**Reset**
31:16	Reserved	R/W	0x0000
15:0	Data to transmit	R/W	0x0000

This register sets the value to the GPIO output pins. Data is written to the register synchronously with the interface clock. It is possible to set or clear a bit of this register with a single write access to the set output data register, or to the clear output data register address (see Table 17 and Table 20.)

Table 10. Direction Control Register (GPIO_DIRECTION)

Base Address = 0xFFFB E400, 0xFFFB EC00, 0xFFFB B400, 0xFFFB BC00, Offset Address = 0x34			
Bit	**Function**	**R/W**	**Reset**
31:16	Reserved	R/W	0x0000
15:0	0: Output 1: Input	R/W	0xFFFF

This register configures the GPIO pins for either input or output. At reset, all of the GPIO pins are configured as inputs. Data is written to this register synchronously with the interface clock.

Table 11. Edge Control Register 1 (GPIO_EDGE_CTRL1)

Base Address = 0xFFFB E400, 0xFFFB EC00, 0xFFFB B400, 0xFFFB BC00, Offset Address = 0x38					
Bit	**Function**			**R/W**	**Reset**
15:0	bit[2n+1]	bit[2n]	Input GPIO[n+1] pins	R/W	0x0000
	0	0	No edge detection		
	0	1	Falling edge detection		
	1	0	Rising edge detection		
	1	1	Both edges detection		

Note: $n \in [7:0]$; GPIO pins 1 to 8.

This register allows the user to define, for each external GPIO[8:1] pin configured as input, the expected edge to trigger an interrupt or a wake-up request. The request can be generated either from a high-to-low transition (bits are 01), a low-to-high transition (bits are 10), or both transitions (bits are 11) occurring on an external GPIO pin configured as input. To disable the edge detection capability, the relevant bits corresponding to the GPIO must be reset (00).

As described in Table 12 , each edge control 1 register bit corresponds to a PIGPIOPINSI[8:1] pin.

Table 12. Edge Ctrl1 Register Bits Correspondences With PIGPIOPINSI[8:1]

EDGE CTRL 1 Register Bits and Corresponding PIGPIOPINSI[n]							
PIGPIOPINSI[4]		**PIGPIOPINSI[3]**		**PIGPIOPINSI[2]**		**PIGPIOPINSI[1]**	
bit[7]	bit[6]	bit[5]	bit[4]	bit[3]	bit[2]	bit[1]	bit[0]
PIGPIOPINSI[8]		**PIGPIOPINSI[7]**		**PIGPIOPINSI[6]**		**PIGPIOPINSI[5]**	
bit[15]	bit[14]	bit[13]	bit[12]	bit[11]	bit[10]	bit[9]	bit[8]

Table 13. Edge Control Register 2 (GPIO_EDGE_CTRL2)

Base Address = 0xFFFB E400, 0xFFFB EC00, 0xFFFB B400, 0xFFFB BC00, Offset Address = 0x3C					
Bit	**Function**			**R/W**	**Reset**
15:0	bit[2n+1]	bit[2n]	Input GPIO[n+9] pins	R/W	0x0000
	0	0	No edge detection		
	0	1	Falling edge detection		
	1	0	Rising edge detection		
	1	1	Both edges detection		

Note: n ∈ [7:0]; GPIO pins 9 to 16.

This register allows the user to define, for each external GPIO[16:9] pin configured as input, the expected edge to trigger an interrupt or a wake-up request. The request can be generated either from a high-to-low transition (bits are 01), a low-to-high transition (bits are 10), or both transitions (bits are 11) occurring on an external GPIO pin configured as input. To disable the edge detection
capability, the relevant bits corresponding to the GPIO must be reset (00).

As described in Table 14, each edge control 2 register bit corresponds to a PIGPIOPINSI[16:9] pin.

Table 14. Edge Ctrl2 Register Bits Correspondences With PIGPIOPINSI[16:9]

EDGE CTRL 2 Register Bits and Corresponding PIGPIOPINSI[n]							
PIGPIOPINSI[12]		**PIGPIOPINSI[11]**		**PIGPIOPINSI[10]**		**PIGPIOPINSI[9]**	
bit[7]	bit[6]	bit[5]	bit[4]	bit[3]	bit[2]	bit[1]	bit[0]
PIGPIOPINSI[16]		**PIGPIOPINSI[15]**		**PIGPIOPINSI[14]**		**PIGPIOPINSI[13]**	
bit[15]	bit[14]	bit[13]	bit[12]	bit[11]	bit[10]	bit[9]	bit[8]

To avoid any inconsistency between the edge detection and the bit(s) in the interrupt status registers when the edge control registers are updated to change the expected transition, you must:

- ❏ Modify the edge detection register and then clear the interrupt status registers 1 and 2.

  Or

- ❏ Stop the edge detection by setting the relevant GPIO line as output, modify the edge detection register, and clear the interrupt status registers 1 and 2.

Table 15. Clear Interrupt Enable Registers (1 and 2) (GPIO_CLEAR_IRQENABLEx)

Base Address = 0xFFFB E400,0xFFFB EC00,0xFFFB B400,0xFFFB BC00, Offset Address = 0x9C,0xA4			
Bit	**Function**	**R/W**	**Reset**
31:16	Reserved	R/W	0x0000
15:0	0: No effect 1: Clear the corresponding bit in the relevant interrupt enable register	R/W	0x0000

A write operation in this register clears the corresponding bit in the interrupt enable1, or 2, register when the written bit is 1; a written bit 0 has no effect (see Section 1.4).

A read of the clear interrupt enable1 or 2 register returns the value of the interrupt enable 1 or 2 register.

Table 16. Clear Wake-up Enable Register (GPIO_CLEAR_WAKEUPENA)

Base Address = 0xFFFB E400, 0xFFFB EC00, 0xFFFB B400, 0xFFFB BC00, Offset Address = 0xA8			
Bit	**Function**	**R/W**	**Reset**
31:16	Reserved	R/W	0x0000
15:0	0: No effect 1: Clear the corresponding bit in the wake-up enable register	R/W	0x0000

A write operation in this register clears the corresponding bit in the wake-up enable register when the written bit is 1; a written bit at 0 has no effect (see Section 1.4).

A read of the clear wake-up enable register returns the value of the wake-up enable register.

Table 17. Clear Data Output Register (GPIO_CLEAR_DATAOUT)

Base Address = 0xFFFB E400, 0xFFFB EC00, 0xFFFB B400, 0xFFFB BC00, Offset Address = 0xB0			
Bit	**Function**	**R/W**	**Reset**
31:16	Reserved	R/W	0x0000
15:0	0: No effect 1: Clear the corresponding bit in the data output register	R/W	0x0000

A write operation in this register clears the corresponding bit in the data output register when the written bit is 1; a written bit at 0 has no effect (see Section 1.4).

A read of the clear data output register returns the value of the data output register.

Table 18. Set Interrupt Enable Registers (1 and 2)(GPIO_SET_IRQENABLEx)

Base Address =0xFFFB E400,0xFFFB EC00,0xFFFB B400,0xFFFB BC00, Offset Address = 0xDC,0xE4			
Bit	**Function**	**R/W**	**Reset**
31:16	Reserved	R/W	0x0000
15:0	0: No effect 1: Set the corresponding bit in the relevant interrupt enable register	R/W	0x0000

A write operation in this register sets the corresponding bit in the interrupt enable1 or 2 register when the written bit is 1; a written bit at 0 has no effect (see Section 1.5).

A read of the set interrupt enable1 or 2 register returns the value of the interrrupt enable1 or 2 register.

Table 19. Set Wake-up Enable Register (GPIO_SET _WAKEUPENA)

Base Address = 0xFFFB E400, 0xFFFB EC00, 0xFFFB B400, 0xFFFB BC00, Offset Address = 0xE8			
Bit	**Function**	**R/W**	**Reset**
31:16	Reserved	R/W	0x0000
15:0	0: No effect	R/W	0x0000
	1: Set the corresponding bit in the wake-up enable register		

A write operation in this register sets the corresponding bit in the wake-up enable register when the written bit is 1; a written bit at 0 has no effect (see Section 1.5).

A read of the set wake-up enable register returns the value of the wake-up enable register.

Table 20. Set Data Output Register (GPIO_SET_DATAOUT)

Base Address = 0xFFFB E400, 0xFFFB EC00, 0xFFFB B400, 0xFFFB BC00, Offset Address = 0xF0			
Bit	**Function**	**R/W**	**Reset**
31:16	Reserved	R/W	0x0000
15:0	0: No effect	R/W	0x0000
	1: Set the corresponding bit in the data output register		

A write operation in this register sets the corresponding bit in the data output register when the written bit is 1; a written bit at 0 has no effect (see Section 1.5).

A read of the set data output register returns the value of the data output register.

1.6 Interrupt and Wake-up Features

In order to generate an interrupt or a wake-up request to a host processor upon a defined logic transition occurring on a GPIO pin, the GPIO configuration registers must be programmed as follows:

- The GPIO pin must be configured as an input in the direction register (see Table 10).
- The GPIO pin must be enabled in the interrupt enable 1 and/or 2 register, or in the wake-up enable register (see Table 6 and Table 7).
- The expected transition on input GPIO triggering the interrupt or the wake-up request must be set in the edge control 1 or 2 registers (see Table 11 and Table 13).

Because of the sample operation with the functional clock, the minimum pulse width on the input GPIO to trigger a synchronous interrupt request is two times the functional clock period. This minimum pulse width must be met before and after any expected level transition detection.

All interrupt or wake-up sources (the 16 input GPIO pins) are merged (see Figure 5 and Figure 6) to issue two synchronous interrupt requests 1 and 2, and a single asynchronous wake-up request.

1.6.1 Synchronous Path: Interrupt Request Generation

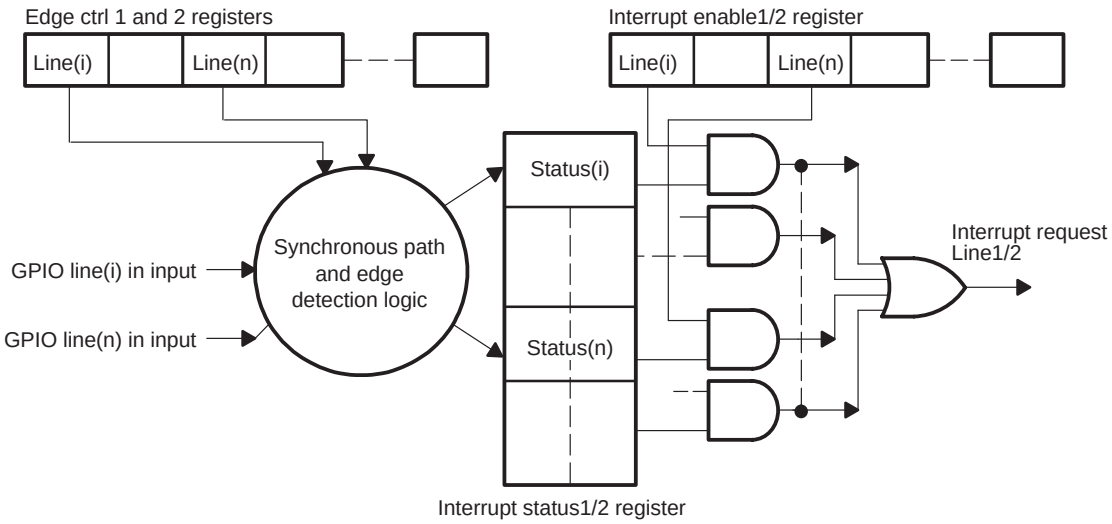
In active mode (functional and interface clocks are running), once the GPIO configuration registers have been set to enable the interrupt generation, a synchronous path samples the transitions on the input GPIO with the functional clock. Because of the sample operation, the minimum pulse width on the input GPIO is two times the functional clock period. When a transition matches with the edge control 1 and 2 registers programming (see Table 11 and Table 13), the corresponding bit in the interrupt status1 and 2 registers is set to 1, synchronously with the interface clock. On the following interface clock cycle, the interrupt lines 1 and/or 2 are active (low) (see Figure 7), depending on the interrupt enable1 and 2 registers.

The timing latency between the expected edge occurring at the input GPIO and the activation of the interrupt line(s) varies between four functional + five interface OCP clock cycles and nine functional + 11 interface OCP clock cycles. This timing latency depends on whether the ongoing edge detection process occurred during the processing (functional domain to interface OCP domain synchronization) of previous edge detection or not.

For any isolated expected edge detection, the interrupt line is active (low level) four functional + five interface OCP clock cycles after the transition occurred (see Figure 7).

In case of multiple and consecutive expected edge detections on different input GPIOs, the maximum timing latency to activate the interrupt line(s) is nine functional + 11 interface OCP clock cycles after the transition occurred (see Figure 8).

Figure 5. Interrupt Request Generation

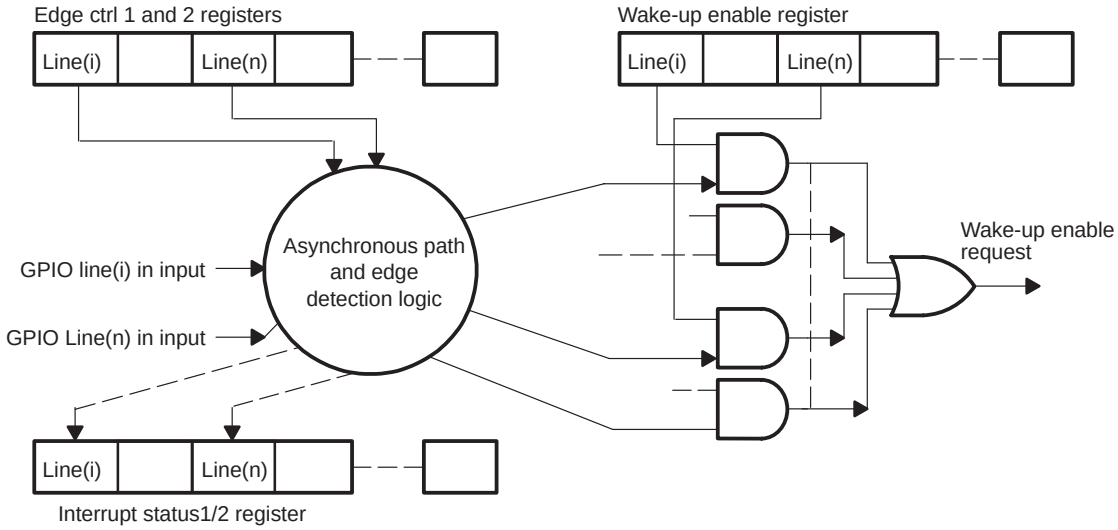


1.6.2 Asynchronous Path: Wake-up Request Generation

In idle mode (all clocks are shut down, the GPIO configuration registers have been previously programmed), an asynchronous path detects the expected transition on the input GPIO, according to edge control 1 and 2 register programming (see Table 11 and Table 13) and sends an asynchronous wake-up request if the wake-up enable register is set (the wake-up line is active high). As shown in Figure 6, there is only one external wake-up line, because the sources of all the wake-ups are merged together. Once the system is awakened, the interface clock is restarted and, according to the input GPIO pin that triggered the wake-up request, the corresponding bits in the interrupt status1 and 2 registers are synchronously set to 1. On the following interface clock cycle, the interrupt lines 1 and/or 2 are active (low) when the corresponding interrupt enable1 and 2 registers are set.

The EnaWakeUp bit of the GPIO_SYSCONFIG register (see Table 3) allows the GPIO wake-up feature to be enabled or disabled globally. If this bit is 0, the wake-up enable register has no effect.

Figure 6. Wake-up Request Generation



1.6.3 Interrupt (or Wake-up) Line Release

When the host processor receives an interrupt request issued by the GPIO peripheral, it can read the corresponding interrupt status register to find out which input GPIO triggered the interrupt or wake-up requeues. After servicing the interrupt or acknowledging the wake-up request, the processor resets the status bit and releases the interrupt line by writing a 1 in the corresponding bit of the interrupt status register (see Figure 7). If an interrupt request is stillpending (the interrupt status register bits are not cleared), the interrupt line is inactive (high) for two cycles before being reasserted (see Figure 8).

1.7 Timing Diagrams

In Figure 7 and Figure 8, the enable register is set (the I/O events are unmasked from triggering an interrupt request).

Figure 7. Timing Diagram of a Single Interrupt

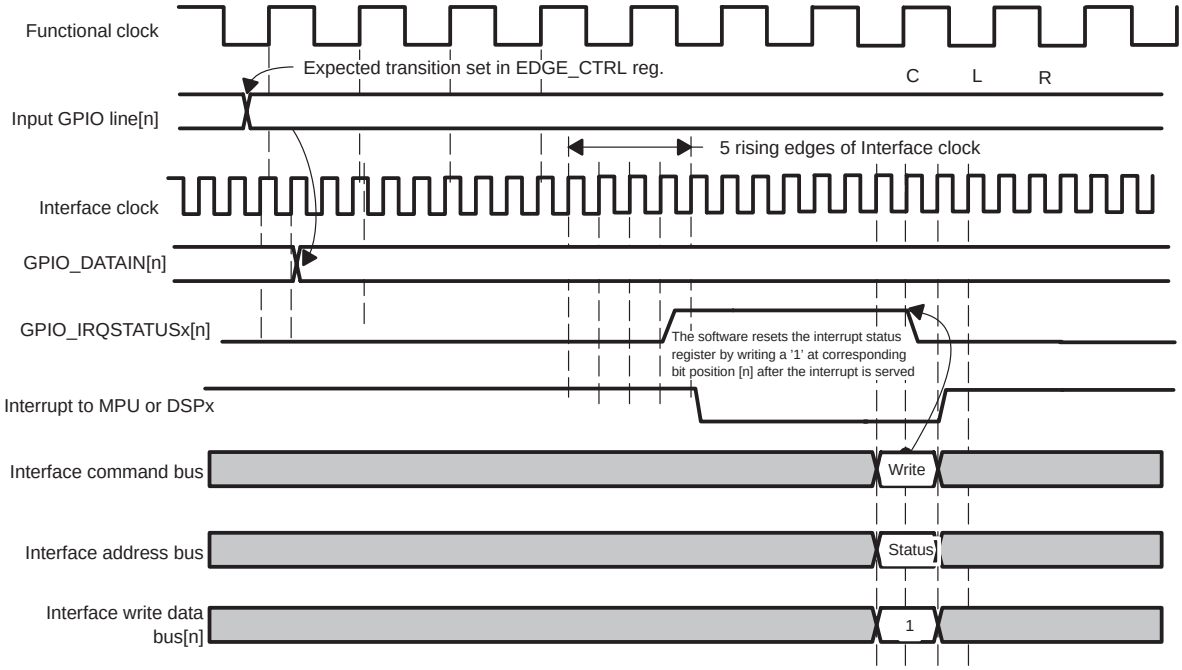
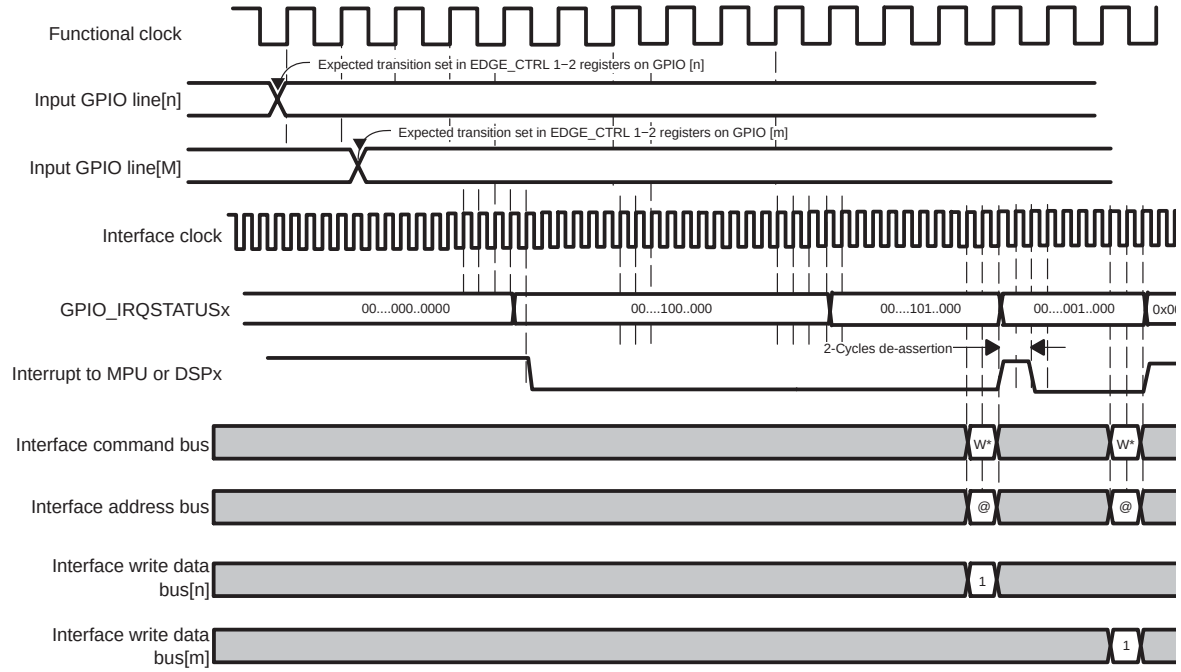


Figure 8. Timing Diagram of Two Consecutive Interrupts



* The software resets the interrupt status register by writing a 1 at the corresponding bit position [n])or[m]) after the interrupt is served.

1.8 Clocking and Reset Strategy

1.8.1 Clocks and Active Edge Definitions

The clock used to sample the GPIO input pins before writing in the data input register (GPIO_DATAIN) is the interface clock (using the rising edges).

On the synchronous path (active mode), the GPIO input transitions sample with the rising edges of the functional clock. Therefore, the minimum pulse width on the input GPIO to trigger a synchronous interrupt request is two times the functional clock period.

On the asynchronous path (in sleep mode, it issues a wake-up request), there is no minimum input pulse width because there is no sampling operation.

The data load in the data output register is set at the output GPIO pins synchronously with the rising edge of the interface clock.

All GPIO registers are accessible synchronously with the interface clock.

1.8.2 Sleep Mode Request and Acknowledge

When the host processor issues a sleep mode request (the idle request signal is active), the GPIO module goes into sleep mode according to the idle mode field of the system configuration register.

If the idle mode field sets the no idle mode, the GPIO does not go into sleep mode and the idle acknowledge signal is not asserted.

If the idle mode field sets the force idle mode, the GPIO goes into sleep mode independently of the internal module state, and the idle acknowledge signal is unconditionally asserted. In the force idle mode, the module is inactive and its wake-up feature is inhibited.

If the idle mode field sets the smart idle mode, the GPIO module evaluates its internal capability to have the functional and interface clocks switched off. Once there is no more internal activity (the data input register completed to capture the input GPIO pins; there is no pending interrupt), the Idle acknowledge signal is asserted and the GPIO enters into sleep mode, ready to issue a wake-up request when the expected transition occurs on an enabled GPIO input pin. This wake-up request is sent only if the field ENAWAKEUP of the system configuration register enables the GPIO wake-up capability (see Table 3). When the system is awakened, the idle request signal goes inactive and the wake-up request signal is also deasserted (if it is the GPIO that triggered the system wake-up).

Once the GPIO acknowledges the sleep mode request (the idle acknowledge signal is active), the functional and interface clocks can be stopped at any time.

1.8.3 Reset

The OCP hardware reset signal has a global reset action on the GPIO. All configuration registers, all DFFs clocked with the functional clock and all internal state machine, are reset when the OCP hardware reset is active (low level). This hardware reset signal is synchronous to the OCP interface clock and internally resynchronized with the functional clock domain to ensure a correct reset of the DFFs using the functional clock.

The reset done field of the system status register (see Table 4) monitors the internal reset status. It is set when the reset is complete.

The software reset (soft reset field of the system configuration register—see Table 3), has the same effect as the OCP hardware reset signal, and the reset done field of the system status register is updated in the same condition.

2 Pulse-Width Tone

This pulse-width tone (PWT) module generates a modulated frequency signal for the external buzzer. The frequency is programmable between 322 Hz and 4868 Hz, with 12 half-tone frequencies per octave. The volume level is also programmable. All frequencies are generated from the PWT_CLK, which is a 12-MHz clock.

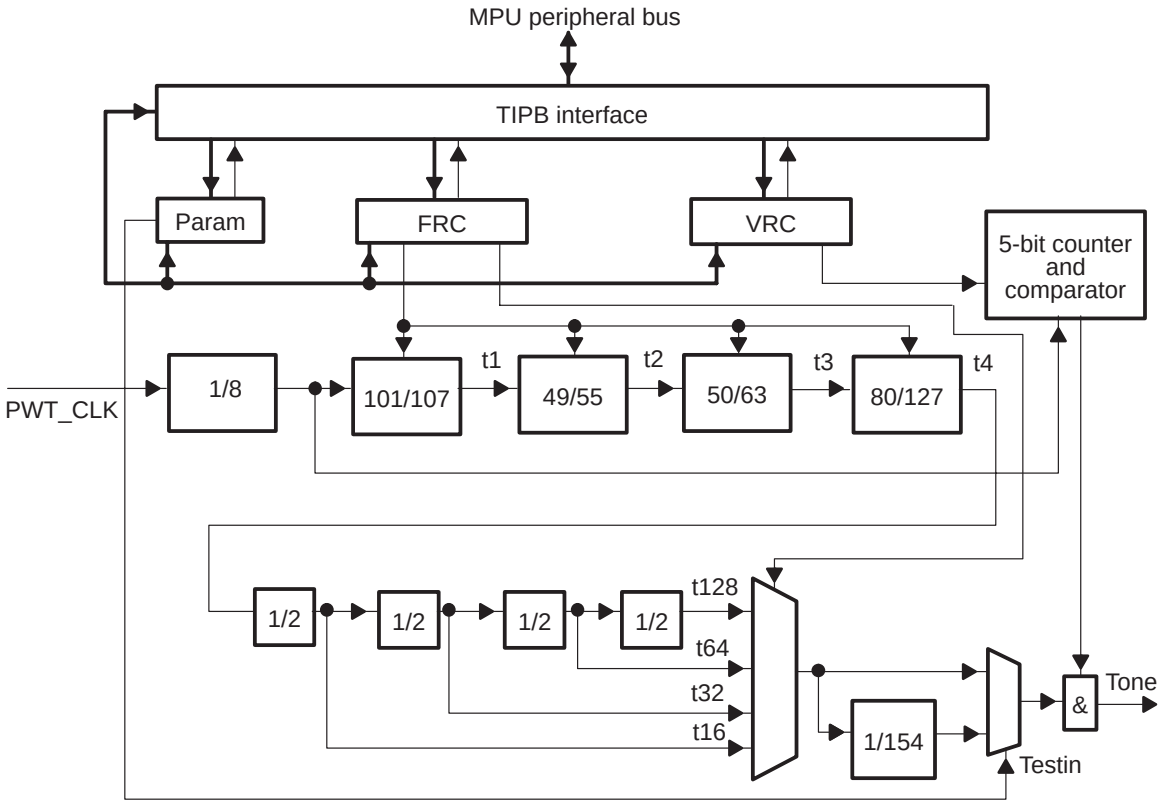
The PWT module creates the output tone signal for a buzzer. The frequency and the volume of this signal are programmable.

2.1 PWT Features

The PWT module has the following features (see Figure 9):

- Divider generating a 1500-kHz frequency clock
- TIPB control interface
- Four dividers with $^{101}/_{107}$, $^{49}/_{55}$, $^{50}/_{63,}$ and $^{80}/_{127}$ to generate each note particularity
- Four dividers 1/2 and a mux to select the octave
- 6-bit register to control tone frequency
- 6-bit register to control tone volume
- 2-bit register for testing and CLK_EN
- 5-bit counter and comparator for creating volume pulse
- Divide by 1/154 to obtain the final right frequency

Figure 9. PWT Block Diagram



2.2 PWT Registers

Start address (hex): FFFB6000

Table 21 lists the 8-bit PWT registers. Table 22 through Table 24 describe the individual registers.

Table 21. PWT Registers

Name	Description	R/W	Offset
FRC	PWT frequency control	R/W	0x00
VRC	PWT volume control	R/W	0x04
GCR	PWT general control	R/W	0x08

Offset address (hex): 0x00

Table 22. PWT Frequency Control Register (FRC)

Bit	Name	Function	R/W	Reset
5:2	FRQ	Frequency selection (12 frequencies) Resynchronized writing, asynchronous reading	R/W	0000
1:0	OCT	Octave selection Resynchronized writing, asynchronous reading	R/W	00

Offset address (hex): 0x04

Table 23. PWT Volume Control Register (VRC)

Bit	Name	Function	R/W	Reset
6:1	VOL	Volume selection Resynchronized writing, asynchronous reading	R/W	000000
0	ONOFF	Switch ON/OFF tone (on: 1, off: 0). Resynchronized writing, asynchronous reading	R/W	0

Offset address (hex): 0x08

Table 24. PWT General Control Register (GCR)

Bit	Name	Function	R/W	Reset
1	TESTIN	Divider 1/154 switched ON/OFF (on: 0, off: 1). Asynchronous writing and reading	R/W	0
0	CLK_EN	PWT clock enable (clock disabled: 0, clock enabled: 1). Asynchronous writing and reading	R/W	0

2.3 PWT Programming

2.3.1 Buzzer Frequency

To obtain the required frequencies, the PWT clock is divided in a special way. Four frequency dividers with the coefficients $^{101}/_{107}$, $^{49}/_{55}$, $^{50}/_{63}$, and $^{80}/_{127}$ are connected in series and can be enabled with the four frequency selection bits (FRQ) in the frequency register. If a divider is not enabled, the clock passes through the divider without any change so that different frequencies can be produced. After this, a multiplexer can choose between this clock, divided by 2, 4, 8, or 16. The frequency is always halved (this unit is called an octave). Because of this, the PWT has a range of four octaves.

The clock behind the multiplexer is divided by 154 to get the required frequencies on the tone output. The 12 frequencies in an octave can be programmed with bits 5 to 2 of the frequency control register (FRC), and the octave can be programmed with bits 1 to 0 of the FRC. Forty-eight different frequencies can be programmed, subdivided into four octaves with twelve frequencies. The four frequency dividers with the complex coefficients $^{101}/_{107}$, $^{49}/_{55}$, $^{50}/_{63}$, and $^{80}/_{127}$ work with the fade-out principle. To get the divider $^{101}/_{107}$ from a periodic pulse, 6 pulses every 107 pulses are fade out. Over a long time, the resulting frequency is $^{101}/_{107}$. The resulting signal has two different periods, which differ by one period of the original signal. Because of this difference, the resulting signal has jitter. To minimize this jitter, the divider works with high frequencies that result in short periods producing low jitter (see Table 25).

Table 25. Buzzer Frequencies

FRC Bits 5-2 1-0	Buzzer Frequency	FRC Bits 5-2 1-0	Buzzer Frequency
0000 00	4868 Hz e5	0000 10	1217 Hz e3
0001 00	4595 Hz dis5	0001 10	1149 Hz dis3
0010 00	4337 Hz d5	0010 10	1084 Hz d3
0011 00	4093 Hz cis5	0011 10	1023 Hz cis3
0100 00	3864 Hz c5	0100 10	966 Hz c3
0101 00	3647 Hz h4	0101 10	912 Hz h2
0110 00	3442 Hz ais4	0110 10	860 Hz ais2
0111 00	3249 Hz a4	0111 10	812 Hz a2
1000 00	3066 Hz gis4	1000 10	767 Hz gis2
1001 00	2894 Hz g4	1001 10	723 Hz g2
1010 00	2732 Hz fis4	1010 10	683 Hz fis2
1011 00	2579 Hz f4	1011 10	644 Hz f2
0000 01	2434 Hz e4	0000 11	608 Hz e2
0001 01	2297 Hz dis4	0001 11	574 Hz dis2
0010 01	2168 Hz d4	0010 11	541 Hz d2
0011 01	2046 Hz cis4	0011 11	511 Hz cis2
0100 01	1932 Hz c4	0100 11	483 Hz c2
0101 01	1824 Hz h3	0101 11	456 Hz h1
0110 01	1721 Hz ais3	0110 11	430 Hz ais1

Note: The PWT was originally designed for a 13-MHz input clock, but most wireless OMAP solutions implement PWT with a 12-MHz clock. Consequently, frequencies shown are not exact tones.

Table 25. Buzzer Frequencies (Continued)

FRC Bits 5-2 1-0	Buzzer Frequency	FRC Bits 5-2 1-0	Buzzer Frequency
0111 01	1624 Hz a3	0111 11	406 Hz a1
1000 01	1533 Hz gis3	1000 11	383 Hz gis1
1001 01	1447 Hz g3	1001 11	361 Hz g1
1010 01	1366 Hz fis3	1010 11	341 Hz fis1
1011 01	1289 Hz f3	1011 11	322 Hz f1
		11XX XX	Not allowed

Note: The PWT was originally designed for a 13-MHz input clock, but most wireless OMAP solutions implement PWT with a 12-MHz clock. Consequently, frequencies shown are not exact tones.

2.3.2 Buzzer Volume

The buzzer volume can be programmed (see Table 26) with bits 6 to 1 in the volume control register VRC. The higher the 6-bit value is, the louder the buzzer/loudspeaker. To perform this programming, a 6-bit binary counter is clocked with the PWT clock and rolls over to 0h after reaching its terminal value (3 Fh). The counter value is compared with the 6-bit value programmed in VRC. If the count value is less than or equal to VRC, the output has the value H, else L:

- Y = VOL value: 0 =<y<64
- PWT_CLK = 12 MHz
- Output signal H period = (y+1).PWT_CLK
- Output signal L period = (63-y).PWT_CLK

Table 26. Buzzer Volume

VRC Bits 6-1 ; 0	Buzzer/Loudspeaker
111111 1	Loud
000000 1	Quiet
xxxxx 0	Off

3 Pulse-Width Light

The pulse-width light (PWL) module provides control of the LCD backlighting and keypad by employing a 4096-bit random sequence generator. This voltage-level control technique decreases the spectral power at the modulator harmonic frequencies. The module uses a 32-kHz clock from ULPD.

3.1 Functional Description

The PWL module is composed of a pseudorandom 8-bit data generator and a programmable threshold comparator (see Figure 10).

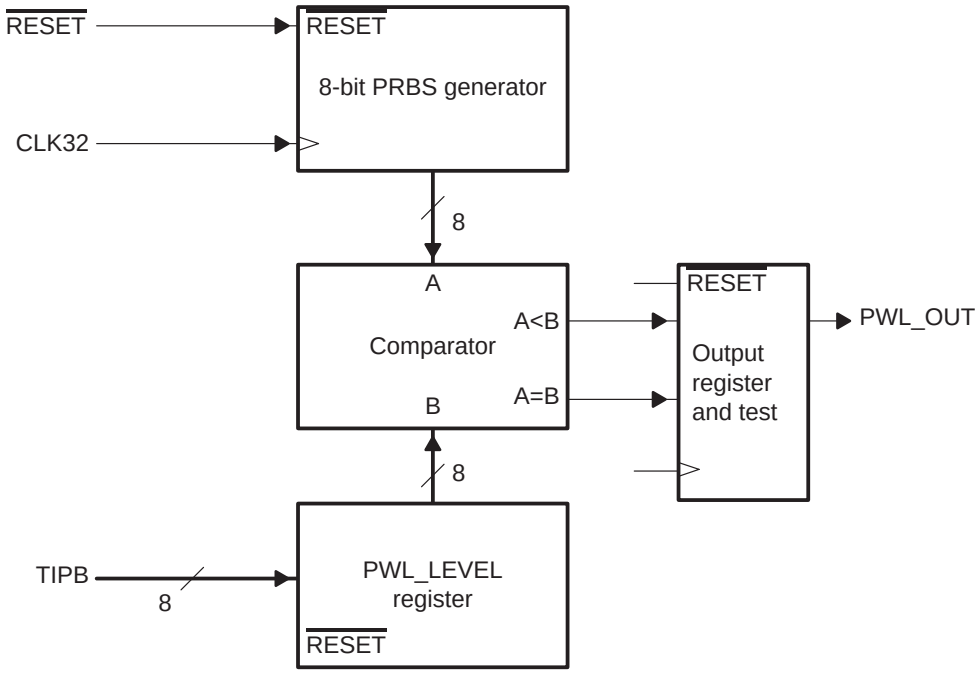
The pseudorandom 8-bit data generator is built using an LFSR. It generates a white normal-law random value between 1 and 255. The LFSR polynomial generator is P(x) = x[7] + x[3] + x[2] + x + 1.

The comparator generates:

- 0 if the random value is greater than or equal to the programmable threshold
- 1 if the random value is less than the programmable threshold

Assuming that the random sequence is normal, it generates a sequence whose mean value is proportional to the comparator threshold.

Figure 10. PWL Block Diagram



3.2 PWL Registers

The PWL is connected to the host with a TIPB. The PWL control is done with two 8-bit registers. All TIPB accesses are done asynchronously with the 32-kHz clock, meaning that there is no TIPB wait-state insertion.

Table 27 lists the 8-bit PWL registers. Table 28 through Table 29 describe the individual registers.

Start address (hex): FFFB:5800

Table 27. PWL Registers

Name	Description	R/W	Offset
PWL_LEVEL	PWL-level	R/W	0x00
CLK_ENABLE	Clock enable	R/W	0x04

Offset address: 0x00

Table 28. PWL Level Register (PWL_LEVEL)

Bit	Name	Function	Reset
7:0	PWL_LEVEL	Defines the mean value of the PWL output signal. 0 leads to a continuous 0 output. 255 to an almost continuous 1 output: 255/256 cycles in high level.	0

Offset address: 0x04

Table 29. PWL Control Register (PWL_CTRL)

Bit	Name	Function	Reset
7:1	–	Reserved	
0	CLK_ENABLE	Internal clock is enabled when 1.	0

Index

OMAP5912 Multimedia Processor Real-Time Clock and Split Power Reference Guide

Literature Number: SPRU782A
March 2004

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.

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
		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 document describes the real-time clock (RTC) block. The RTC is an embedded real-time clock module directly accessible from the TIPB bus interface.

This document also describes split power.

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.

Related Documentation From Texas Instruments

The following documents describe the OMAP5910 device and related peripherals. Copies of these documents are available on the Internet at www.ti.com. *Tip:* Enter the literature number in the search box provided at www.ti.com.

OMAP5912 Multimedia Processor Device Overview and Architecture Reference Guide (literature number SPRU748) introduces the setup, components, and features of the OMAP5912 multimedia processor and provides a high-level view of the device architecture.

OMAP5912 Multimedia Processor OMAP 3.2 Subsystem Reference Guide (literature number SPRU749) introduces and briefly defines the main features of the OMAP3.2 subsystem of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor DSP Sybsystem Reference Guide (literature number SPRU750) describes the OMAP5912 multimedia processor DSP subsystem. The digital signal processor (DSP) subsystem is built around a core processor and peripherals that interface with: 1) The

ARM926EJS via the microprocessor unit interface (MPUI); 2) Various standard memories via the external memory interface (EMIF); 3) Various system peripherals via the TI peripheral bus (TIPB) bridge.

OMAP5912 Multimedia Processor Clocks Reference Guide (literature number SPRU751) describes the clocking mechanisms of the OMAP5912 multimedia processor. In OMAP5912, various clocks are created from special components such as the digital phase locked loop (DPLL) and the analog phase-locked loop (APLL).

OMAP5912 Multimedia Processor Initialization Reference Guide (literature number SPRU752) describes the reset architecture, the configuration, the initialization, and the boot ROM of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Power Management Reference Guide (literature number SPRU753) describes power management in the OMAP5912 multimedia processor. The ultralow-power device (ULPD) generates and manages clocks and reset signals to OMAP3.2 and to some peripherals. It controls chip-level power-down modes and handles chip-level wake-up events. In deep sleep mode, this module is still active to monitor wake-up events.This book describes the ULPD module and outline architecture.

OMAP5912 Multimedia Processor Security Features Reference Guide (literature number SPRU754) describes the security features of teh OMAP5912 multimedia processor. The OMAP5912 security scheme relies on the OMAP3.2 secure mode. The distributed security on the OMAP3.2 platform is a Texas Instruments solution to address m-commerce and security issues within a mobile phone environment. The OMAP3.2 secure mode was developed to bring hardware robustness to the overall OMAP5912 security scheme.

OMAP5912 Multimedia Processor Direct Memory Access (DMA) Support Reference Guide (literature number SPRU755) describes the direct memory access support of the OMAP5912 multimedia processor. The OMAP5912 processor has three DMAs:

- The system DMA is embedded in OMAP3.2. It handles DMA transfers associated with MPU and shared peripherals.
- The DSP DMA is embedded in OMAP3.2. It handles DMA transfers associated with DSP peripherals.
- The generic distributed DMA (GDD) is an OMAP5912 resource attached to the SSI peripheral. It handles only DMA transfers associated with the SSI peripheral.

OMAP5912 Multimedia Processor Memory Interfaces Reference Guide (literature number SPRU756) describes the memory interfaces of the OMAP5912 multimedia processor.

- SDRAM (external memory interface fast, or EMIFF)
- Asynchronous and synchronous burst memory (external memory interface slow, or EMIFS)
- NAND flash (hardware controller or software controller)
- CompactFlash on EMIFS interface
- Internal static RAM

OMAP5912 Multimedia Processor Interrupts Reference Guide (literature number SPRU757) describes the interrupts of the OMAP5912 multimedia processor. Three level 2 interrupt controllers are used in OMAP5912:

- One MPU level 2 interrupt handler (also referred to as MPU interrupt level 2) is implemented outside of OMAP3.2 and can handle 128 interrupts.
- One DSP level 2 interrupt handler (also referred to as DSP interrupt level 2.1) is instantiated outside of OMAP3.2 and can handle 64 interrupts.
- One OMAP3.2 DSP level 2 interrupt handler (referenced as DSP interrupt level 2.0) can handle 16 interrupts.

OMAP5912 Multimedia Processor Peripheral Interconnects Reference Guide (literature number SPRU758) describes various periperal interconnects of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Timers Reference Guide (literature number SPRU759) describes various timers of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Serial Interfaces Reference Guide (literature number SPRU760) describes the serial interfaces of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Universal Serial Bus (USB) Reference Guide (literature number SPRU761) describes the universal serial bus (USB) host on the OMAP5912 multimedia processor. The OMAP5912 processor provides several varieties of USB functionality. Flexible multiplexing of signals from the OMAP5912 USB host controller, the OMAP5912 USB function controller, and other OMAP5912 peripherals allow a wide variety of system-level USB capabilities. Many of the OMAP5912 pins can be used for USB-related signals or for signals from other OMAP5912 peripherals. The OMAP5912 top-level pin multiplexing

controls each pin individually to select one of several possible internal pin signal interconnections. When these shared pins are programmed for use as USB signals, the OMAP5912 USB signal multiplexing selects how the signals associated with the three OMAP5912 USB host ports and the OMAP5912 USB function controller can be brought out to OMAP5912 pins.

OMAP5912 Multimedia Processor Multi-channel Buffered Serial Ports (McBSPs) Reference Guide (literature number SPRU762) describes the three multi-channel buffered serial ports (McBSPs) available on the OMAP5912 device. The OMAP5912 device provides multiple high-speed multichannel buffered serial ports (McBSPs) that allow direct interface to codecs and other devices in a system.

OMAP5912 Multimedia Processor Camera Interface Reference Guide (literature number SPRU763) describes two camera inerfaces implemented in the OMAP5912 multimedia processor: compact serial camera port and camera parallel interface.

OMAP5912 Multimedia Processor Display Interface Reference Guide (literature number SPRU764) describes the display interface of the OMAP5912 multimedia processor.

- LCD module
- LCD data conversion module
- LED pulse generator
- Display interface

OMAP5912 Multimedia Processor Multimedia Card (MMC/SD/SDIO) (literature number SPRU765) describes the multimedia card (MMC) interface of the OMAP5912 multimedia processor. The multimedia card/secure data/secure digital IO (MMC/SD/SDIO) host controller provides an interface between a local host, such as a microprocessor unit (MPU) or digital signal processor (DSP), and either an MMC or SD memory card, plus up to four serial flash cards. The host controller handles MMC/SD/SDIO or serial port interface (SPI) transactions with minimal local host intervention.

OMAP5912 Multimedia Processor Keyboard Interface Reference Guide (literature number SPRU766) describes the keyboard interface of the OMAP5912 multimedia processor. The MPUIO module enables direct I/O communication between the MPU (through the public TIPB) and external devices. Two types of I/O can be used: specific I/Os dedicated for 8 x 8 keyboard connection, and general-purpose I/Os.

OMAP5912 Multimedia Processor General-Purpose Interface Reference Guide (literature number SPRU767) describes the general-purpose in-

terface of the OMAP5912 multimedia processor. There are four GPIO modules in the OMAP5912. Each GPIO peripheral controls 16 dedicated pins configurable either as input or output for general purposes. Each pin has an independent control direction set by a programmable register. The two−edge control registers configure events (rising edge, falling edge, or both edges) on an input pin to trigger interrupts or wake−up requests (depending on the system mode). In addition, an interrupt mask register masks out specified pins. Finally, the GPIO peripherals provide the set and clear capabilities on the data output registers and the interrupt mask registers. After detection, all event sources are merged and a single synchronous interrupt (per module) is generated in active mode, whereas a unique wake−up line is issued in idle mode. Eight data output lines of the GPIO3 are ORed together to generate a global output line at the OMAP5912 boundary. This global output line can be used in conjunction with the SSI to provide a CMT−APE interface to the OMAP5912.

OMAP5912 Multimedia Processor VLYNQ Serial Communications Interface Reference Guide (literature number SPRU768) describes the VLYNQ of the OMAP5912 multimedia processor.

VLYNQ is a serial communications interface that enables the extension of an internal bus segment to one or more external physical devices. The external devices are mapped into local, physical address space and appear as if they are on the internal bus of the OMAP 5912. The external devices must also have a VLYNQ interface. The VLYNQ module serializes bus transactions in one device, transfers the serialized data between devices via a VLYNQ port, and de-serializes the transaction in the external device.

OMAP5912 includes one VLYNQ module connected on OCPT2 target port and OCPI initiator port. These connections are configured via a static switch, which selects either SSI or VLYNQ module. This switch, forbids the simultaneous use of GDD/SSI and VLYNQ. The switch is controlled by the VLYNQ_EN bit in the OMAP5912 configuration control register (CONF_5912_CTRL).

OMAP5912 Multimedia Processor Pinout Reference Guide (literature number SPRU769) provides the pinout of the OMAP5912 multimedia processor. After power-up reset, the user can change the configuration of the default interfaces. If another interface is available on top of the default, it is possible to enable a new interface for each ball by setting the corresponding 3-bit field of the associated FUNC_MUX_CTRL register. It is also possible to configure on-chip pullup/pulldown. This document

also describes the various power domains so that the user can apply the different interfaces seamlessly with external components.

OMAP5912 Multimedia Processor Window Tracer (WT) Reference Guide (literature number SPRU770) describes the window tracer module used to capture the memory transactions from four interfaces: EMIFF, EMIFS, OCP-T1, and OCP-T2. This module is located in the OMAP3.2 traffic controller (TC).

OMAP5912 Multimedia Processor Real-Time Clock Reference Guide (literature number SPRUxxx) describes the real-time clock of the OMAP5912 multimedia processor. The real-time clock (RTC) block is an embedded real-time clock module directly accessible from the TIPB bus interface.

Trademarks

Contents

Figures

Tables

Real-Time Clock (RTC)

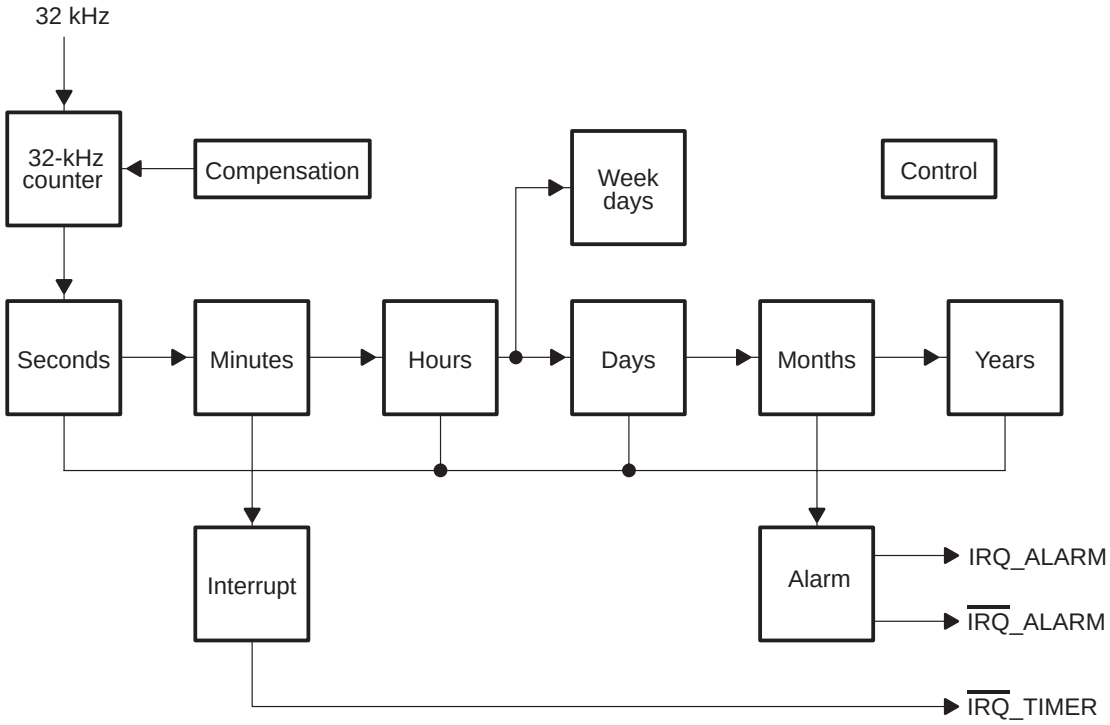
1 Overview

The real-time clock (RTC) block is an embedded real-time clock module directly accessible from the TIPB bus interface. The basic functions of RTC block are:

- Time information (seconds/minutes/hours) directly in binary coded decimal (BCD) code
- Calendar information (day/month/year/day of the week) directly in BCD code up to the year 2099
- Interrupt generation, periodically (1s/1m/1h/1d period) or at a precise time of the day (alarm function)
- 30-s time correction
- Oscillator frequency calibration

Figure 1 shows the real-time clock block.

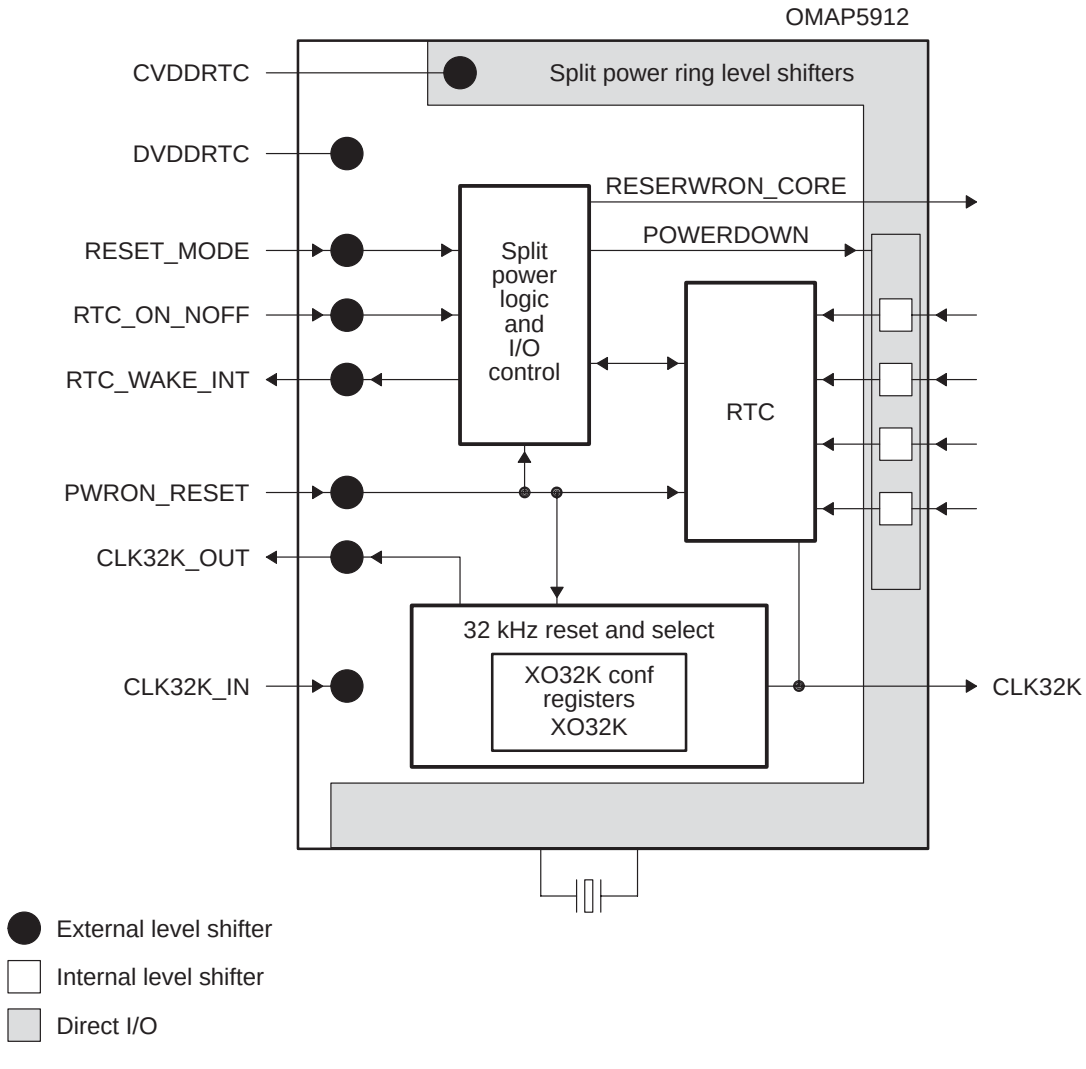
Figure 1. Real-Time Clock



2 Split Power Overview

To achieve minimum consumption in the OFF state in device equipment, some active logic elements are supplied. Those elements are the real-time clock (RTC) and the 32-kHz oscillator (OSC32K) in the digital baseband (DBB), and the power-on reset (POR) and the dedicated regulator in the analog baseband (ABB). This approach is possible when using split power, which splits the core power domain into two subdomains powered with different voltage supplies. Internal level shifters handle the separation between the core and the active domain.

Figure 2. Split Power System in OMAP5912



3 Internal Level Shifters

Internal level shifters are library-standard macrocells that interface two core domains powered with two different voltage supplies.

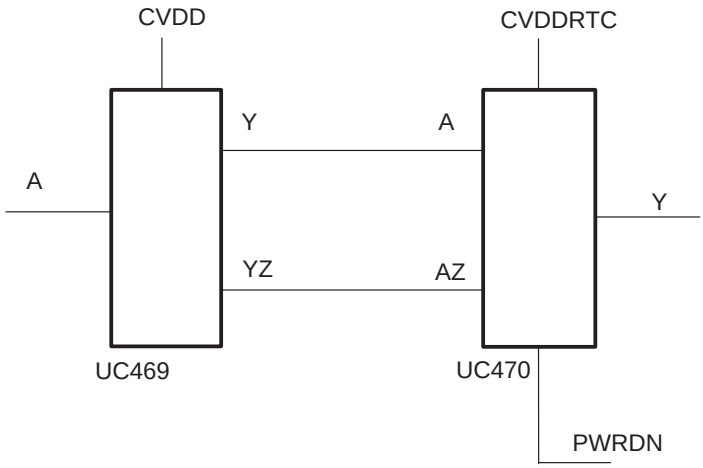
The cell can be seen as a means to isolate one power domain from an other. In the case where one domain is shut down. the level shifters ensure a known state at the boundary of the two domains.

The level shifters are divided into two parts:

- UC469: Powered by the primary power supply. Because of the input signal A, it creates Y and YZ, which are A buffered and A inverted, respectively.

- UC470: Powered by the secondary power supply. Because of the differential signals, A and AZ (given by UC469), it creates the signal Y that is equivalent to the input of UC469 but in the second core supply domain. This cell has a PWRDN signal so that when power supply of UC469 does not exist (input signals A and AZ are ambiguous), this cell does not have any through-current and the output is set to 0.

Figure 3. Internal Level Shifter



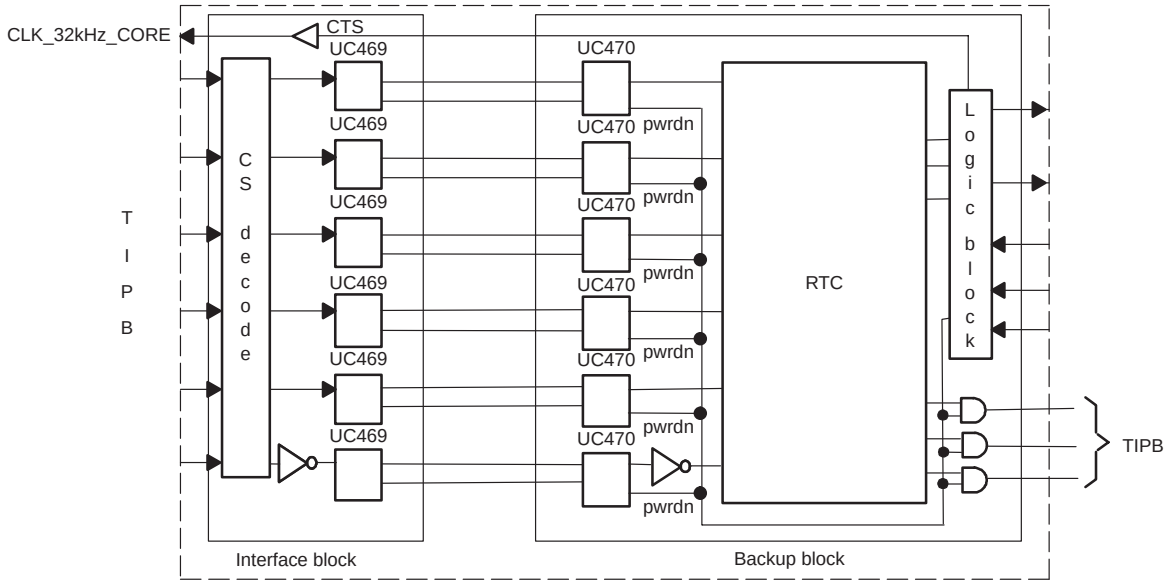
PWRDN = 0 => Y=A

PWRDN =1 => Y= 0

4 Split Power Block

The split power module contains two blocks: the interface block and the backup block. The interface block, between the core and the backup elements, is supplied by the core power supply CVDD. The backup block contains the RTC and some logic.

Figure 4. Split Power Block Diagram



4.1 Interface Block

This block separates the backup elements and the ASIC core. It also contains the UC469 and some logic. The logic allows masking of the TIPB bus signals to avoid consumption of internal level shifters, except in RTC accesses.

4.2 Backup Block

This block contains the elements kept in OFF mode in the DBB, the RTC, the 32-kHz clock, the UC470, and the logic to force the DBB to OFF mode. The module input/outputs are RTC_ON_NOFF, $\overline{\text{PWRON\_RESET}}$, RTC_WAKE_INT, RESET_MODE, and POWERDOWN.

5 Output Control

To keep them from supplying the ASIC core, the split power block outputs are forced to logical zero.

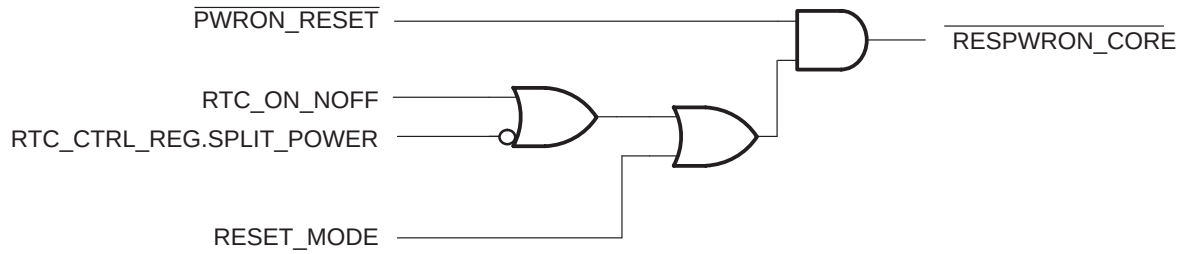
5.1 Backup Signal Management

In OFF mode, only the split power module is supplied. The ON_OFF, RTC_WAKE_INT, and PWRON_RESET signals, which control the activity management of the DBB, are also active.

6 On-Chip Reset Generation

- If RTC_CTRL_REG.SPLIT_POWER = 0 or RESET_MODE = 1 then the OMAP5912 is reset only by PWRON_RESET.:
- If RESET_MODE = 0 and RTC_CTRL_REG.SPLIT_POWER= 1 then PWRON_RESET AND RTC_ON_NOFF will reset OMAP5912.:

Figure 5. ASIC Reset Scheme



6.1 Resets for OMAP5912 Device With Split Power Feature Enabled

The RTC module and its internal real time counter are reset by PWRON_RESET during power up. OMAP5912 ASIC gates are reset by PWRON_RESET_CORE.

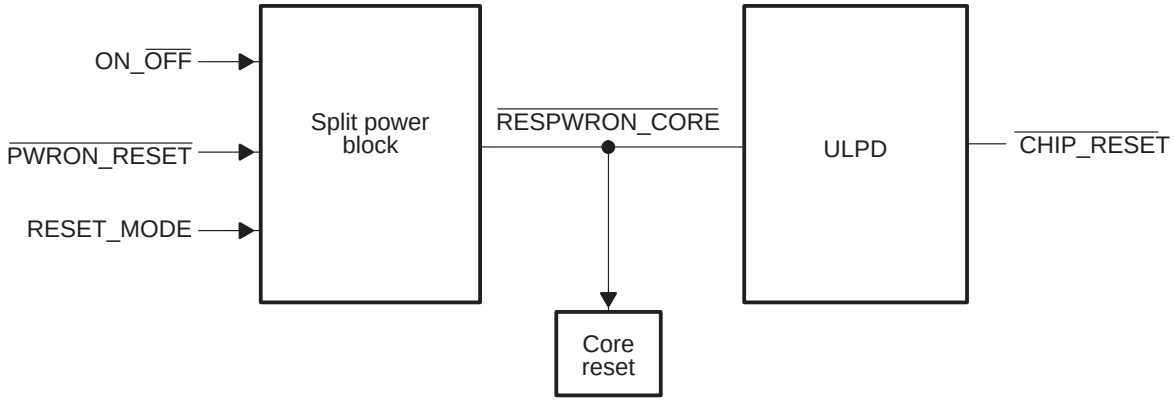
Subsequent resets are asserted with RTC_ON_NOFF. The RTC module is not reset by the assertion of RTC_ON_NOFF. While RTC_ON_NOFF is asserted low, the system powers down OMAP5912 ASIC gates.

6.2 Resets for OMAP5912 Device With Split Power Feature Not Used

For OMAP5912 devices that are forced during reset in reset mode 1, PWRON_RESET_CORE is logically equal to PWRON_RESET and only to PWRON_RESET.

For OMAP5912 devices that are forced during reset in reset mode 0 (OMAP1510 legacy) and no split power, PWRON_RESET_CORE is logically equal to PWRON_RESET and only to PWRON_RESET.

Figure 6. $\overline{\text{PWRON\_RESET}}$ Connection



RTC_WAKE_INT

The RTC_WAKE_INT pin now collects the interrupt sources used to awaken the ULPD from deep sleep. It is gated with RTC_CTRL_REG.SPLIT_POWER and the RTC alarm. In OFF mode the IRQ_SET is set to 0, and only the IRQ_ALARM_EXT can generate an interrupt on the pin RTC_WAKE_INT.

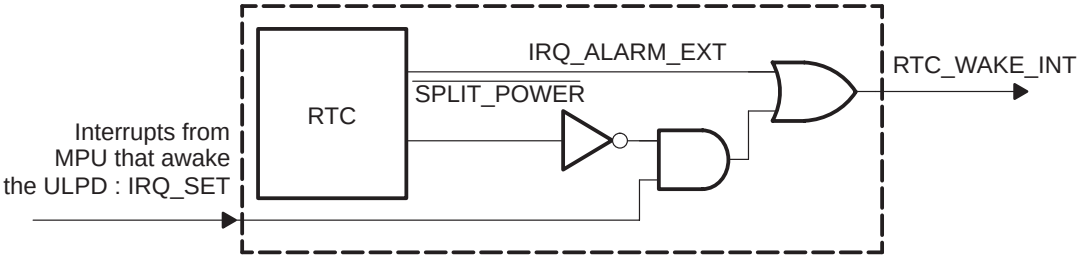
With a device in ON state, both IRQ_ALARM_EXT and IRQ_SET can generate an RTC_WAKE_INT.

The RTC_CTRL_REG.SPLIT_POWER signal generates RTC_WAKE_INT to avoid an RTC_WAKE_INT in ON mode for the ABB.

Figure 7 shows the generation of the RTC_WAKE_INT.

In the OMAP5912 core, IRQ_SET is not used and is set to 0. RTC_WAKE_INT is used only for the ABB.

Figure 7. RTC_WAKE_INT Generation

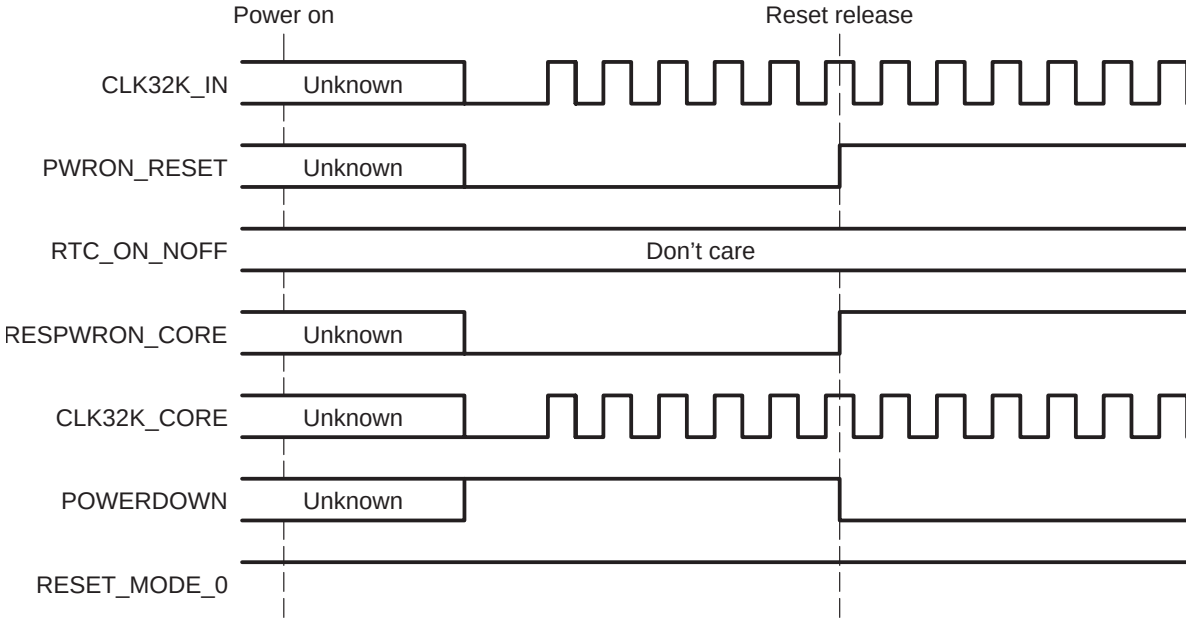


7 Using Split Power

7.1 ABB Functions Incompatible With Split Power

Do not use split power. The power cannot be cut in the core.

Figure 8. OMAP5912 in RESET_MODE = 1 or RTC_CTRL_REG.SPLIT_POWER = 0



7.2 ABB Functions Compatible With Split Power

Figure 9. Startup With RTC_CTRL_REG.SPLIT_POWER = 1 for OMAP5912 RESET_MODE = 0

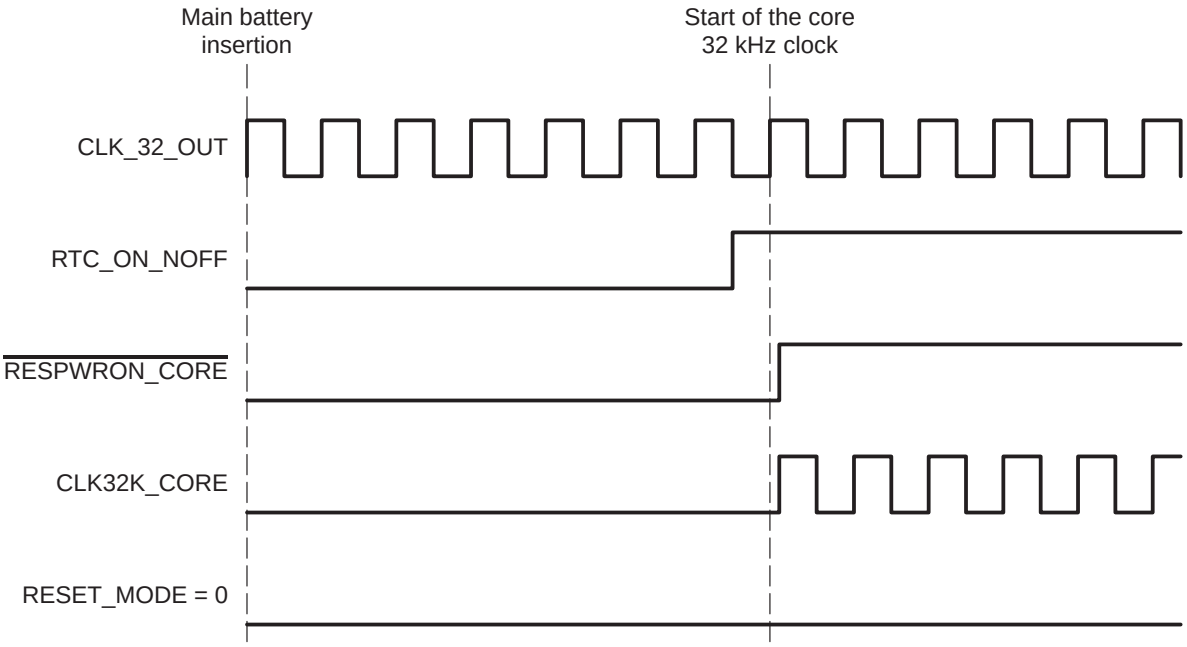
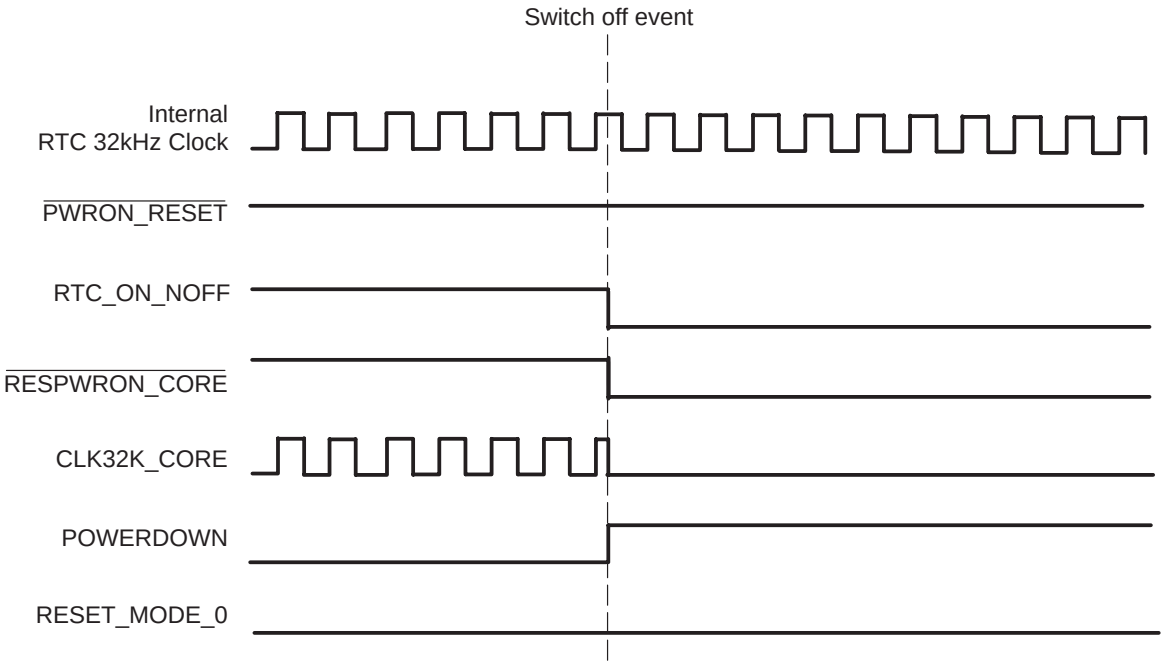


Figure 9 assumes that backup battery is already inserted, only the RTC power domain is powered and that the RTC is isolated from the core (RTC_ON_NOFF ball is held low, and SPLIT_POWER bit in RTC is set to 1). Once ‘the main battery is plugged in, the core domain is reset by $\overline{\text{PWRON_RESET}}$_CORE until RTC_ON_NOFF becomes high; the isolation mode also becomes inactive. The ULPD state machine can start.

7.3 ON to OFF Description

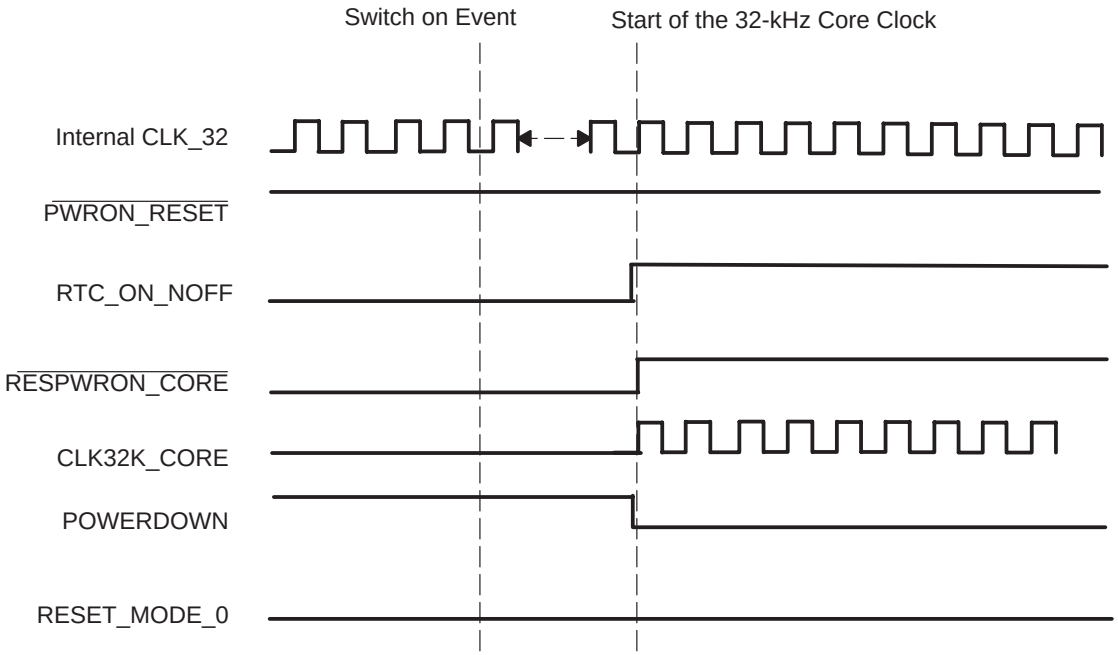
Figure 10. ON to OFF With SPLIT POWER = 1 for OMAP5912 RESET_MODE = 0



On a switch-off event, RTC_ON_NOFF is set to 0. The 32-kHz clock is not fed to OMAP5912 core anymore, and PWRON_RESET_CORE is set to 0 to indicate that the device goes into OFF state. PWRON_RESET_CORE becomes active, and the DBBcore is reset. POWERDOWN becomes active and the backup module is isolated. The ABB circuit disables the CORE LDO regulators, and thus the DBB core is not powered.

7.4 OFF to ON Description

Figure 11. OFF to ON With RTC_CTRL_REG.SPLIT_POWER = 1 for OMAP5912 RESET_MODE = 0



On a switch-on event, the ASIC LDO regulators are enabled. The DBB core is also supplied and reset by PWRON_RESET_CORE until RTC_ON_NOFF is set to 1. Then, the isolation mode is inactive, the ULPD state machine receives PWRON_RESET_CORE, and the clock starts its state machine.

8 Interrupt Management

RTC can generate three interrupts:

- A timer interrupt (IRQ_TIMER),
- An alarm interrupt (IRQ_ALARM_CHIP)
- An alarm interrupt external (IRQ_ALARM_EXT or IRQ_ALARM_CHIP inverted)

8.1 Timer Interrupt

IRQ_TIMER interrupt can be generated periodically, every second, every minute, every hour, or everyday (RTC_INTERRUPTS_REG[1:0]).

The IT_TIMER bit of the interrupt register enables this interrupt.

It is a negative edge-sensitive interrupt (low-level pulse duration = 15 µs).

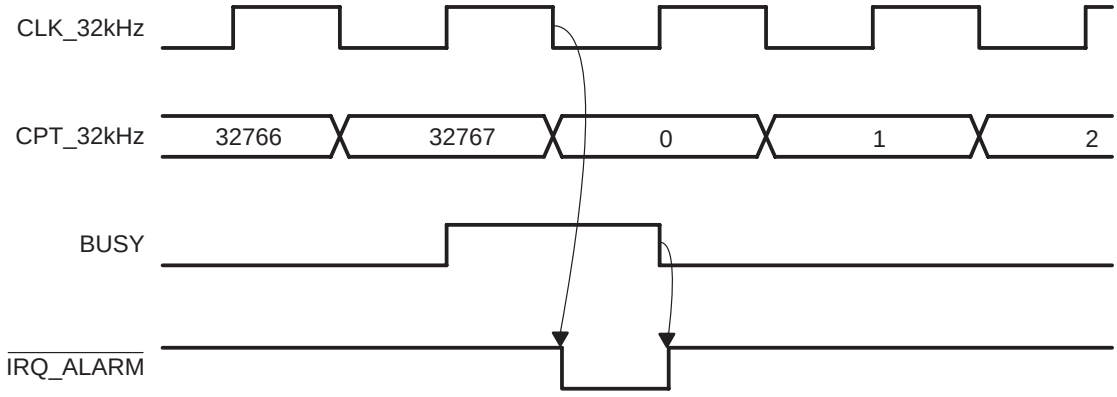
The RTC_STATUS_REG[5:2] are only updated at each new interrupt and show the events that have occurred, according to Table 1.

Table 1. Timer Interrupt Events

RTC_INTERRUPTS_REG[1:0]	11	10	01	00
RTC_STATUS_REG[5] (DAY)	1	0/1†	0/1†	0/1†
RTC_STATUS_REG[4] (HOUR)	1	1	0/1†	0/1†
RTC_STATUS_REG[3] (MIN)	1	1	1	0/1†
RTC_STATUS_REG[2] (SEC)	1	1	1	1

† 1 when this event is concurrent with programmed periodical period.

Figure 12. Periodic Interrupt



8.2 Alarm Interrupt

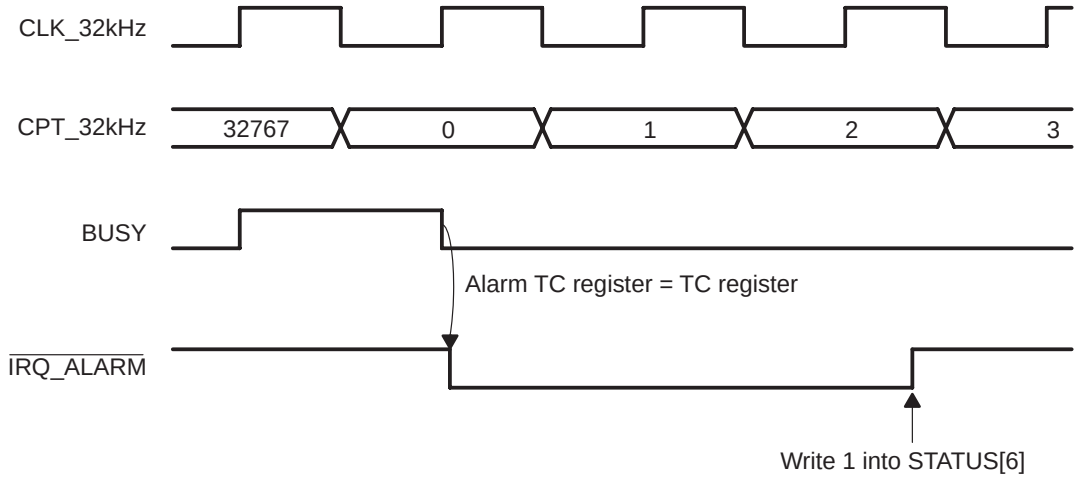
IRQ_ALARM_CHIP interrupt can be generated when the time set into the time and calendar ALARM registers is identical in the time and calendar registers.

This interrupt is then generated if the IT_ALARM bit of the interrupt register is set.

This interrupt is low-level sensitive. RTC_STATUS_REG[6] indicates that IRQ_ALARM_CHIP occurred.

This interrupt is disabled by writing 1 into the RTC_STATUS_REG[6].

Figure 13. Alarm Interrupt



9 Oscillator Drift Compensation

To compensate for any inaccuracy of the 32-kHz oscillator, the MPU can perform a calibration of the oscillator frequency, calculate the drift compensation versus one-hour period, and load the compensation registers with the drift compensation value.

Autocompensation is enabled by the AUTO_COMP_EN bit in the RTC_CTRL register.

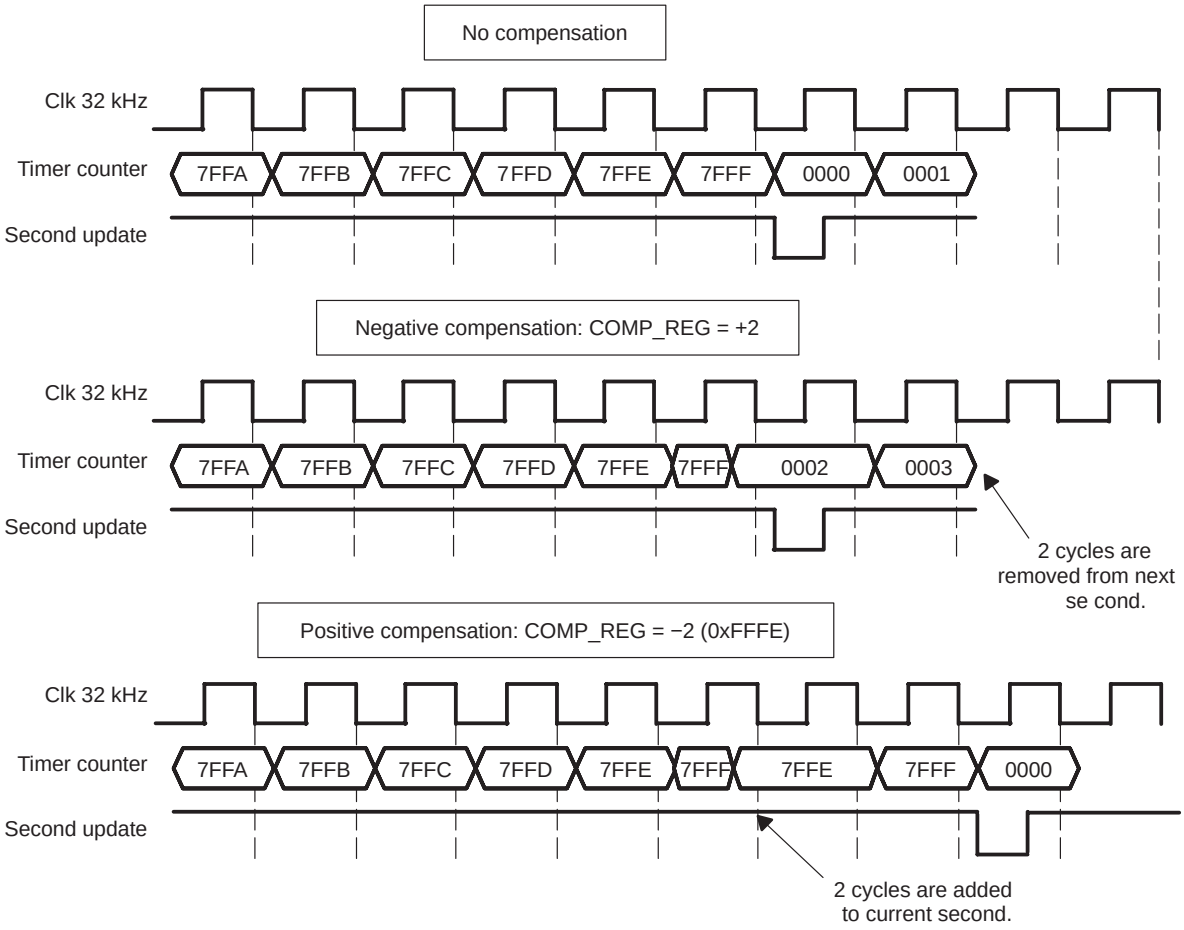
If the COMP_REG value is positive, compensation occurs *after* the second change event. COMP_REG cycles are removed from the next second.

If the COMP_REG value is negative, compensation occurs *before* the second change event. The COMP_REG cycles are added to the current second.

This compensation enables a 32-kHz period accuracy each hour.

The waveform in Figure 14 summarizes the positive and negative compensation effect.

Figure 14. Oscillator Drift Compensation



10 Split Power Compatibility

The RTC and the 32-kHz oscillator are the only elements that must be active in the device in OFF state. Therefore, the RTC has been modified to use split power. One register has been modified (RTC_CTRL_REG), and one register has been added (RTC_RES_PROG_REG).

In the RTC_CTRL_REG, the $\overline{\text{RTC\_CTRL\_REG.SPLIT\_POWER}}$ bit has been added so the user can choose whether to use the power split mode.

The RTC_RES_PROG_REG has been added to back up the resistance value of the oscillator in OFF state.

A power-down signal has been added to set the outputs to 0 in OFF.

11 RTC Registers

There are three types of registers:

- Time and calendar, and time and calendar alarm
- General
- Compensation

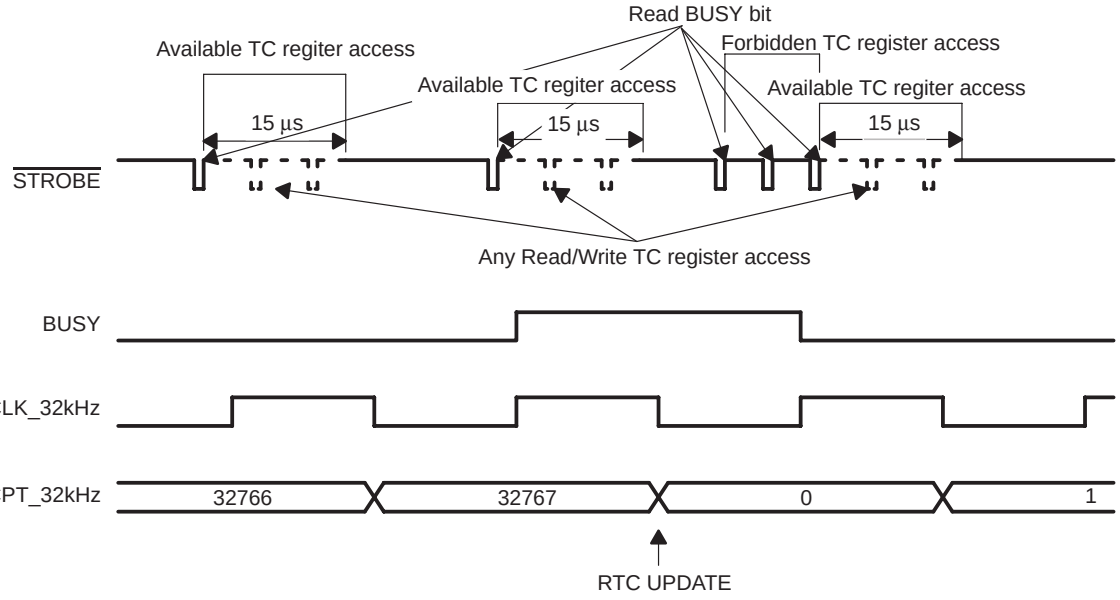
These three types have their own access constraints.

11.1 Time and Calendar Registers and Time and Calendar Alarm Registers

To read or write correct data from/to the time and calendar registers and time and calendar alarm registers, the MPU must first read the BUSY bit of the STATUS register until BUSY is equal to zero. From this time, and for a time of 15 µs (the available access period), the MPU can perform several accesses into the time and calendar registers and time and calendar alarm registers with guaranteed read/write data. At the end of one available access period, the MPU must restart the previous sequence. If the MPU accesses the time and calendar registers during an unavailable access period, access is not ensured.

To remove any possibility of interrupting the registers read process, thus introducing a potential risk of violating the authorized 15−µs access period, it is strongly recommended that the user disable all incoming interrupts during the register read process.

Figure 15. Time and Calendar Register and Time and Calendar Alarm Register Access



11.2 General Registers

The MPU can access the STATUS_REG and the CTRL_REG at any time (with the exception of the CTRL_REG[5] bit, which must be changed only when the RTC is stopped).

For the INTERRUPTS_REG, the MPU must respect the available access period to prevent spurious interrupt.

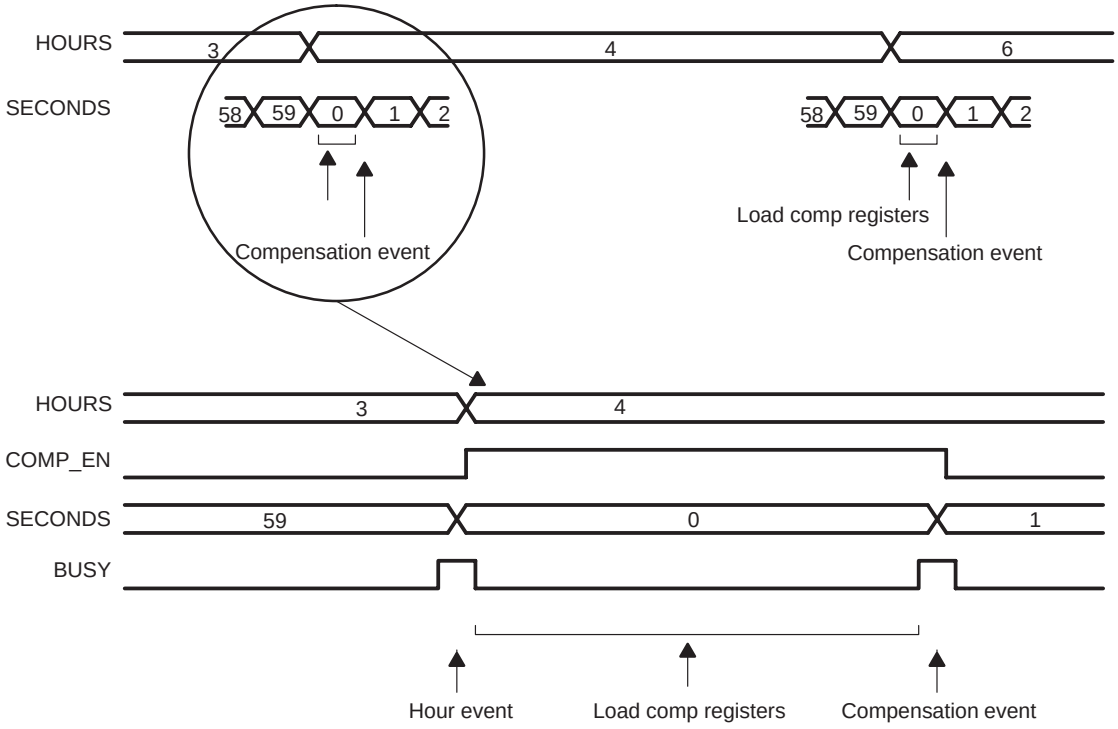
The RTC_DISABLE bit of the CTRL register must be used only to completely disable the RTC function. When this bit is set, the 32-kHz clock is gated, and the RTC is frozen. From this point, resetting this bit to zero can lead to unexpected behavior. This bit must only be used if the RTC function is unwanted in the application, to save power.

11.3 Compensation Registers

Access to the COMP_MSB_REG and COMP_LSB_REG registers must respect the available access period. These registers must not be updated during compensation (first second of each hour), but they can be updated during the second access period preceding a compensation event.

For example, the MPU can load the compensation value into these registers after each hour event during an available access period.

Figure 16. Compensation Scheduling



12 Setting Time and Calendar Information

12.1 Modify Time and Calendar Registers

To modify the current time, the MPU writes the new time into time and calendar registers to fix the time/calendar information. The MPU can write into time and calendar registers without stopping the RTC, but in this case the MPU must read the status register to be sure that the RTC updating takes place in more than 15 µs (bit BUSY must be at 0). The MPU must perform all changes in less than 15 µs, to prevent partial updating between the beginning and the end of the writingsequence into time and calendar registers.

Also, the MPU can stop the RTC by clearing the STOP_RTC bit of the control register (owing to internal resynchronization, the RUN bit of the status must be checked to ensure that the RTC is frozen), updating the time and calendar values, and restarting the RTC by resetting STOP_RTC bit.

12.2 Rounding Seconds

Time can be rounded to the closest minute by setting the ROUND_30S bit of the control register. When this bit is set, time and calendar values are set to the closest minute value at the next second. The ROUND_30S bit is automatically cleared when rounding time is performed.

Example:

If current time is 10H59M45S, round operation changes time to 11H00M00S.

If current time is 10H59M29S, round operation changes time to 10H59M00S.

Table 2 lists the RTC registers. The tables below describe the register bits. The RTC register types are grouped as follows:

- General registers
- Compensation registers
- Time and calendar alarm registers
- Time and calendar registers

Table 2. RTC Registers

Base Address = 0xFFFB 4800		
Name	**Description**	**Address Offset**
SECONDS_REG	Seconds	0x00
MINUTES_REG	Minutes	0x04
HOURS_REG	Hours	0x08
DAYS_REG	Days	0x0C
MONTHS_REG	Months	0x10
YEARS_REG	Years	0x14
WEEKS_REG	Weeks	0x18
RESERVED	Reserved	0x1C
ALARM_SECONDS_REG	Alarm seconds	0x20
ALARM_MINUTES_REG	Alarm minutes	0x24
ALARM_HOURS_REG	Alarm hours	0x28
ALARM_DAYS_REG	Alarm days	0x2C
ALARM_MONTHS_REG	Alarm months	0x30
ALARM_YEARS_REG	Alarm years	0x34
RESERVED	Reserved	0x38

Table 2. RTC Registers (Continued)

Base Address = 0xFFFB 4800		
Name	**Description**	**Address Offset**
RESERVED	Reserved	0x3C
RTC_CTRL_REG	RTC control	0x40
RTC_STATUS_REG	RTC status	0x44
RTC_INTERRUPTS_REG	RTC interrupt	0x48
RTC_COMP_LSB_REG	RTC compensation LSB	0x4C
RTC_COMP_MSB_REG	RTC compensation MSB	0x50
RTC_OSC_REG	RTC oscillator	0x54

12.2.1 Time and Calendar Registers

The time and calendar information is available in dedicated registers, called time and calendar registers. These register values are written in binary coded decimal (BCD) code.

Table 3. Time and Calendar Register Time Units

Time Unit	Range	Remarks
Year	00 to 99	Leap year: year divisible by four
		Common year: other years
Month	01 to 12	
Day	01 to 31	01 to 31 for months 1, 3, 5, 7, 8, 10, 12
		01 to 30 for months 4, 6, 9 ,11
		01 to 29 for month 2 (leap year)
		01 to 28 for month 2 (common year)
Week	00 to 06	Weekday
Hour	00 to 23	00 to 23 in 24 hours mode
		01 to 12 in AM/PM mode

Table 3. Time and Calendar Register Time Units (Continued)

Minutes	00 to 59
Seconds	00 to 59

Table 4. Seconds Register (SECONDS_REG)

Base Address = 0xFFFB 4800, Offset = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
7:4	SEC1	2nd digit of seconds Range is 0 to 5.	R/W	0000
3:0	SEC0	1st digit of seconds Range is 0 to 9.	R/W	0000

Table 5. Minutes Register (MINUTES_REG)

Base Address = 0xFFFB 4800, Offset = 0x04				
Bit	**Name**	**Function**	**R/W**	**Reset**
7:4	MIN1	2nd digit of minutes Range is 0 to 5.	R/W	0000
3:0	MIN0	1st digit of minutes Range is 0 to 9.	R/W	0000

Table 6. Hours Register (HOURS_REG)

Base Address = 0xFFFB 4800, Offset = 0x08				
Bit	**Name**	**Function**	**R/W**	**Reset**
7	PM_$\overline{\text{AM}}$	Only used in PM_AM mode (otherwise 0) 0: AM 1: PM	R/W	0
6:4	HOUR1	2nd digit of hours Range is 0 to 2.	R/W	000
3:0	HOUR0	1st digit of hours Range is 0 to 9.	R/W	0000

Table 7. Days Register (DAYS_REG)

Table 7. Days Register (DAYS_REG) (Continued)

Base Address = 0xFFFB 4800, Offset = 0x0C				
Bit	**Name**	**Function**	**R/W**	**Reset**
7:4	DAY1	2nd digit of days Range is 0 to 3.	R/W	0000
3:0	DAY0	1st digit of days Range is 0 to 9.	R/W	0001

Table 8. Months Register (MONTHS_REG)

Base Address = 0xFFFB 4800, Offset = 0x10				
Bit	**Name**	**Function**	**R/W**	**Reset**
7:4	MONTH1	2nd digit of months Range is 0 to 1.	R/W	0000
3:0	MONTH0	1st digit of months Range is 0 to 9.	R/W	0001

Usual notation taken for the month value:

01: January
02: February
....
12: December

Table 9. Years Register (YEARS_REG)

Base Address = 0xFFFB 4800, Offset = 0x14				
Bit	**Name**	**Function**	**R/W**	**Reset**
7:4	YEAR1	2nd digit of years Range is 0 to 9.	R/W	0000
3:0	YEAR0	1st digit of years Range is 0 to 9.	R/W	0000

Table 10. Weeks Register (WEEKS_REG)

Base Address = 0xFFFB 4800, Offset = 0x18				
Bit	**Name**	**Function**	**R/W**	**Reset**
3:0	WEEK	1st digit of days in a week Range is 0 to 6.	R/W	0000

Table 11. Reserved

Base Address = 0xFFFB 4800, Offset = 0x1C				
Bit	**Name**	**Function**	**R/W**	**Reset**
???:0	RESERVED	Reserved		

Table 12. Alarm Seconds Register (ALARM_SECONDS_REG)

Base Address = 0xFFFB 4800, Offset = 0x20				
Bit	**Name**	**Function**	**R/W**	**Reset**
7:4	ALARM_SEC1	2nd digit of seconds Range is 0 to 5.	R/W	0000
3:0	ALARM_SEC0	1st digit of seconds Range is 0 to 9.	R/W	0000

Table 13. Alarm Minutes Register (ALARM_MINUTES_REG)

Base Address = 0xFFFB 4800, Offset = 0x24				
Bit	**Name**	**Function**	**R/W**	**Reset**
7:4	ALARM_MIN1	2nd digit of minutes Range is 0 to 5.	R/W	0000
3:0	ALARM_MIN0	1st digit of minutes Range is 0 to 9.	R/W	0000

Table 14. Alarm Hours Register (ALARM_HOURS_REG)

Base Address = 0xFFFB 4800, Offset = 0x28				
Bit	**Name**	**Function**	**R/W**	**Reset**
7	ALARM_PM_$\overline{AM}$	Only used in PM_AM mode (otherwise 0) 0: AM 1: PM	R/W	0
6:4	ALARM_HOUR1	2nd digit of hours Range is 0 to 2.	R/W	000
3:0	ALARM_HOUR0	1st digit of hours Range is 0 to 9.	R/W	0000

Table 15. Alarm Days Register (ALARM_DAYS_REG)

Base Address = 0xFFFB 4800, Offset = 0x2C				
Bit	**Name**	**Function**	**R/W**	**Reset**
7:4	ALARM_DAY1	2nd digit for days Range is 0 to 3.	R/W	0000
3:0	ALARM_DAY0	1st digit for days Range is 0 to 9.	R/W	0001

Table 16. Alarm Months Register (ALARM_MONTHS_REG)

Base Address = 0xFFFB 4800, Offset = 0x30				
Bit	**Name**	**Function**	**R/W**	**Reset**
7:4	ALARM_MONTH1	2nd digit of months Range is 0 to 1.	R/W	0000
3:0	ALARM_MONTH0	1st digit of months Range is 0 to 9.	R/W	0001

Table 17. Alarm Years Register (ALARM_YEARS_REG)

Base Address = 0xFFFB 4800, Offset = 0x34				
Bit	**Name**	**Function**	**R/W**	**Reset**
7:4	ALARM_YEAR1	2nd digit of years Range is 0 to 9.	R/W	0000

Table 17. Alarm Years Register (ALARM_YEARS_REG) (Continued)

Base Address = 0xFFFB 4800, Offset = 0x34				
Bit	**Name**	**Function**	**R/W**	**Reset**
3:0	ALARM_YEAR0	1st digit of years Range is 0 to 9.	R/W	0000

Table 18. RTC Control Register (RTC_CTRL_REG)

Base Address = 0xFFFB 4800, Offset = 0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
7	$\overline{\text{SPLIT\_POWER}}$	0: Cannot use split power 1: Can use split power	R/W	0
6	RTC_disable	0: RTC enabled 1: RTC disabled (no 32-kHz clock)	R/W	0
5	SET_32_COUNTER	0: No action 1: Set the 32-kHz counter with COMP_REG value.	R/W	0
4	TEST_MODE	0: Functional mode 1: Test mode (autocompensation is enabled when the 32-kHz counter reaches its end)	R/W	0
3	MODE_12_24	0: 24-hour mode 1: 12-hour mode (PM/AM mode)	R/W	0
2	AUTO_COMP	0: No autocompensation 1: Autocompensation enabled	R/W	0
1	ROUND_30S	0: No update 1: When a 1 is written, the time is rounded to the closest minute.	R/W	0
0	STOP_RTC	0: RTC is frozen. 1: RTC is running.	R/W	0

The SET_32_COUNTER must only be used when the RTC is frozen.

ROUND_30S is a toggle bit. MPU can only write 1, and RTC clears it. If the MPU sets the ROUND_30S bit and then reads it, the MPU read 1 until the round-to-the-closest-minute is performed at the next second.

MODE_12_24: It is possible to switch between the two modes at any time without disturbing the RTC. Read or write is always performed with the current mode.

Table 19. RTC Status Register (RTC_STATUS_REG)

Base Address = 0xFFFB 4800, Offset = 0x44				
Bit	**Name**	**Function**	**R/W**	**Reset**
7	POWER_UP	Indicates that a reset occurred	R/W	1
6	ALARM	Indicates that an alarm interrupt has been generated	R/W	0
5	1D_EVENT	One day has occurred.	R	0
4	1H_EVENT	One hour has occurred.	R	0
3	1M_EVENT	One minute has occurred.	R	0
2	1S_EVENT	One second has occurred.	R	0
1	RUN	0: RTC is frozen. 1: RTC is running.	R	0
0	BUSY	0: Updating event in more than 15 µs 1: Updating event	R	0

The alarm interrupt keeps its low level until the MPU writes 1 in the ALARM bit of the RTC_STATUS_REG register.

The timer interrupt is a low-level pulse (15-µs duration).

The RUN bit shows the real state of the RTC. Because the STOP_RTC signal is resynchronized on the 32-kHz clock, the action of this bit is delayed.

POWER_UP is set by a reset and is cleared by writing 1 in this bit.

Table 20. RTC Interrupts Register (RTC_INTERRUPTS_REG)

Base Address = 0xFFFB 4800, Offset = 0x48				
Bit	**Name**	**Function**	**R/W**	**Reset**
3	IT_ALARM	Enables one interrupt when the alarm value is reached (time and calendar alarm registers) by the time and calendar registers	R/W	0
2	IT_TIMER	Enable periodic interrupt 0: Interrupt disabled 1: Interrupt enabled	R/W	0
1:0	EVERY	Interrupt period 0: Every second 1: Every minute 2: Every hour 3: Every day	R/W	00

Note:

The MPU must respect the BUSY period to prevent spurious interrupt.

Table 21. RTC Compensation LSB Register (RTC_COMP_LSB_REG)

Base Address = 0xFFFB 4800, Offset = 0x4C				
Bit	**Name**	**Function**	**R/W**	**Reset**
7:0	RTC_COMP_LSB	Indicates number of 32-kHz periods to be added into the 32-kHz counter every hour	R/W	0x00

Note:

This register must be written in twos complement. This means that to add one 32-kHz oscillator period every hour, the MPU must write FFFF into RTC_COMP_MSB_REG and RTC_COMP_LSB_REG. To remove one 32-kHz oscillator period every hour, the MPU must write 0001 into RTC_COMP_MSB_REG and RTC_COMP_LSB_REG. The 7FFF value is not allowed.

Table 22. RTC Compensation MSB Register (RTC_COMP_MSB_REG)

Base Address = 0xFFFB 4800, Offset = 0x50				
Bit	**Name**	**Function**	**R/W**	**Reset**
7:0	RTC_COMP_MSB	Indicates number of 32-kHz periods to be added into the 32-kHz counter every hour	R/W	0x00

Table 23. RTC Oscillator Register (RTC_OSC_REG)

Base Address = 0xFFFB 4800, Offset = 0x54				
Bit	**Name**	**Function**	**R/W**	**Reset**
4	OSC32K_PWRDN_R	Control of 32-kHz oscillator power down (function mode)	R/W	0
3:0	SW_RES_PROG	Value of the oscillator resistance	R/W	

The oscillator receives the register value when TST_OSC32K_MUX_CTRL = 0 (functional mode); otherwise, the value resistance and the power down are from the JTAG register.

Index