OMAP5910 Dual-Core Processor

Technical Reference Manual

Literature Number: SPRU602B
January 2003

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 that 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.

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265

Preface

Read This First

About This Manual

This manual describes the Texas Instruments OMAP5910 multimedia processor, hereafter called the OMAP5910 device. The OMAP5910 device supports the development and testing of wireless device applications that use the Microsoft Windows CE™ or the Symbian EPOC operating system. This manual is intended for developers who have knowledge of the Windows CE or the Symbian EPOC operating system and of the wireless application environment.

Notational Conventions

This document uses the following conventions.

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

  Here is a sample program listing:

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

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

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

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

  .asect **"***section name***",** *address*

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

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

  LALK *16–bit constant [, shift]*

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

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

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

  ```
  { * | *+ | *– }
  ```

  This provides three choices: *, *+, or *–.

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

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

  .byte $value_1$ *[, ... ,* $value_n$*]*

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

Information About Cautions and Warnings

This book may contain cautions and warnings.

This is an example of a caution statement.

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

This is an example of a warning statement.

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

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

Related Documentation From Texas Instruments

TMS320C55x DSP CPU Reference Guide (SPRU371D)

TMS320C55x DSP Functional Overview (SPRU312)

TMS320C55x DSP Function Library (DSPLIB) Programmer's Reference (SPRU422D)

TMS320C55x Technical Overview (SPRU393)

TMS320C55x DSP Programmer's Guide (SPRU376A)

TMS320C55x DSP Mnemonic Instruction Set Ref Guide (SPRU374F)

TMS320C55x Assembly Language Tools User's Guide (SPRU280D)

TMS320C55x Optimizing C Compiler User's Guide (SPRU281C)

The MultiMediaCard System Specification Version 3.1 – June 2001. MMCA Technical Committee

SD Memory Card Specifications: Part 1 Physical Layer Specification, Version 1.0 – March 2000 + Supplementary Notes Part 1 June 2000. SD Group

Trademarks

Bluetooth is a trademark owned by the Bluetooth SIG, Inc. and licensed to Texas Instruments.

TMS320C5510, TMS320C55x, C55x, and Code Composer Studio are trademarks of Texas Instruments.

Microsoft, Windows, Windows CE, Windows CE Platform Builder, and Windows NT are trademarks of Microsoft Corporation.

Other trademarks are the property of their respective owners.

Contents

Figures

Tables

Chapter 1

Introduction

This chapter introduces the setup, components, and features of the OMAP5910 processor and provides a high-level view of the device architecture.

Topic **Page**

1.1 Overview

The OMAP5910 is a highly integrated hardware and software platform designed to meet the application processing needs of next-generation embedded devices.

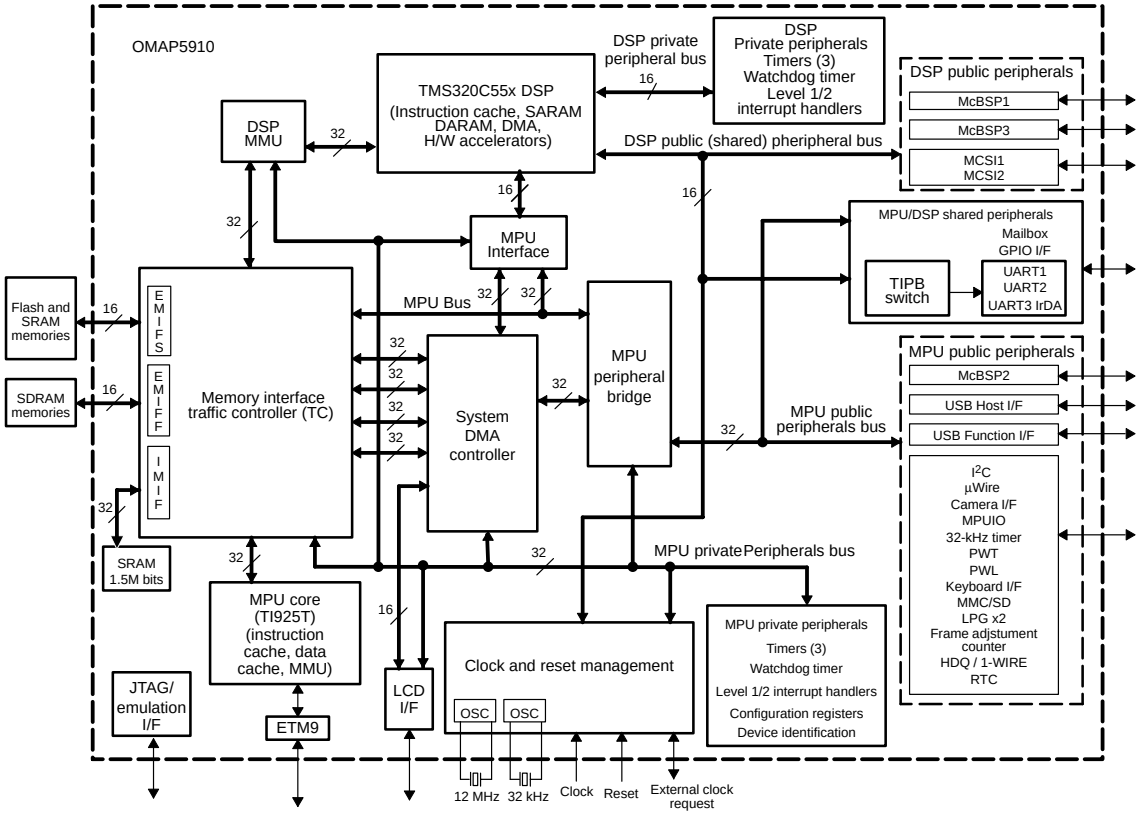
The OMAP5910 processor features a unique dual-core architecture that combines the command and control capabilities of the TI-enhanced ARM™ 925 processor (TI925T) with the high-performance and low-power capabilities of the TMS320C55x™ DSP core. These two key components of the OMAP5910 processor are:

- A TI reduced instruction set computer (RISC) microprocessor unit (MPU) subsystem. The MPU subsystem is based on the TI925T control processor, peripherals, and other components. The TI925T processor is based on the Advanced RISC Machines ARM9TDMI technology.
- A TI digital signal processor (DSP) subsystem. The DSP subsystem incorporates a TI TMS320C55x DSP, peripherals, and other components.

The OMAP5910 processor is available in the small 289-pin MicroStar™ BGA package (12 x 12 mm).

Figure 1–1 is a master block diagram of the 289-pin OMAP5910 processor. Figure 1–2 shows the OMAP5910 in more detail.

Figure 1–1. OMAP5910 Master Block Diagram



1.2 Description

The OMAP5910 processor features the first generation of the Texas Instruments Incorporated OMAP™ architecture.

The OMAP platform enables OEMs and ODMs to quickly bring to market devices featuring rich user interfaces, high processing performance, and long battery life through the maximum flexibility of a fully integrated mixed processor solution.

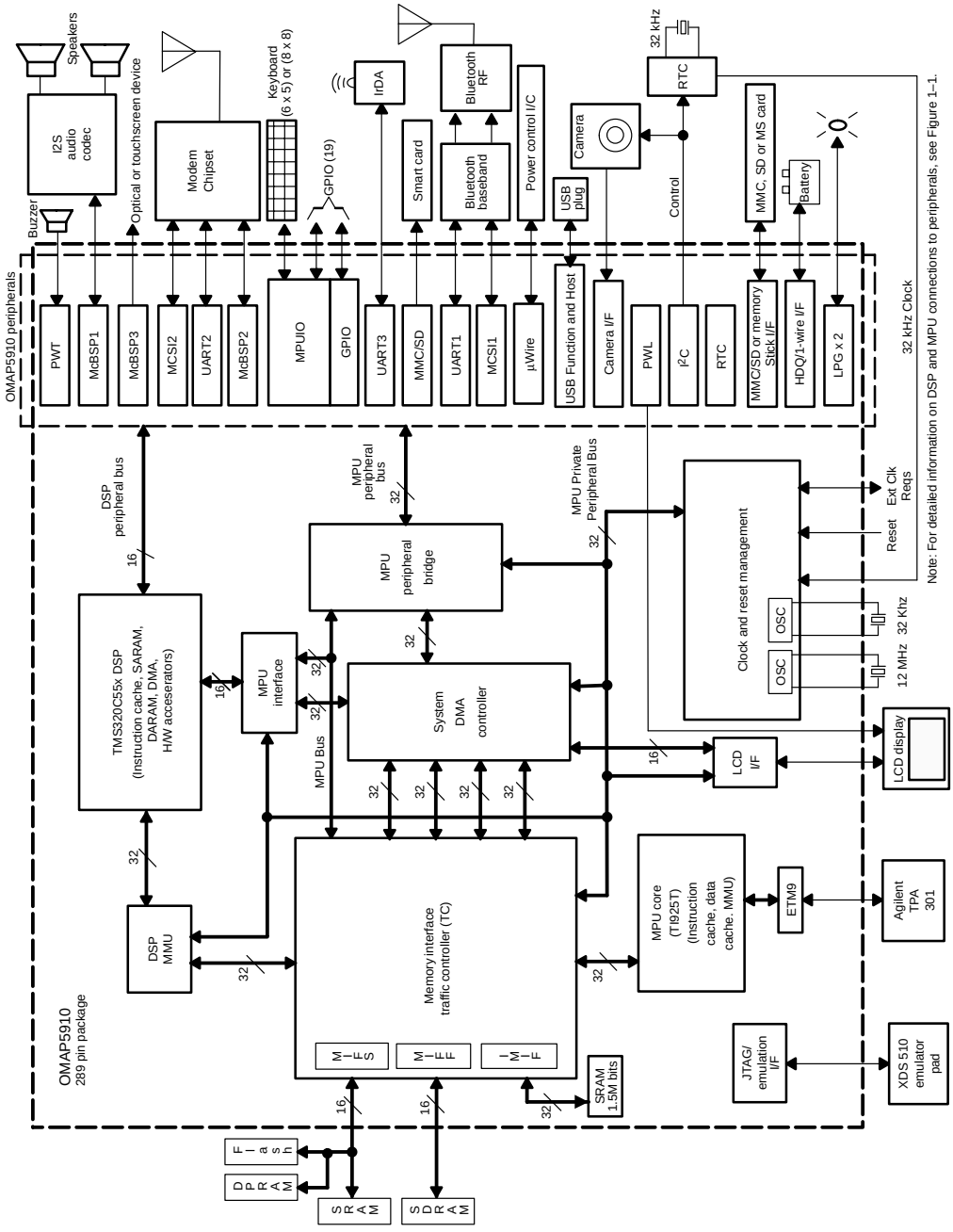
The dual-core architecture provides benefits of both DSP and RISC technologies, incorporating a TMS320C55x DSP core and a high-performance TI925T core.

The OMAP5910 device is designed to run leading open and embedded RISC-based operating systems, as well as the Texas Instruments (TI) DSP/BIOS™ software kernel foundation, and is available in a 289-ball MicroStar BGA package.

The OMAP5910 is targeted for the following applications:

- Applications processing devices
- Mobile communications
  - 802.11
  - Bluetooth™ wireless technology
  - Proprietary government and other
- Video and image processing (MPEG4, JPEG, Windows Media Video, etc.)
- Advanced speech applications (text-to-speech, speech recognition)
- Audio processing (MPEG-1 Audio Layer3 [MP3], AMR, WMA, AAC, and other GSM speech codecs)
- Graphics and video acceleration
- Generalized web access
- Data processing (fax, encryption/decryption, authentication, signature verification and watermarking)

Figure 1–2. OMAP5910 Diagram



Note: As a system real-time time clock, there are two possible solutions:

- ❑ The OMAP5910 internal solution (internal RTC and 32-kHz oscillator pads) is not a low-power solution, because the OMAP5910 RTC power supply is not separate from the OMAP5910 core power supply.
- ❑ The OMAP5910 external solution (external RTC and 32-kHz oscillators) is the recommended solution for low IDDQ.

1.3 Features

The OMAP5910 device has the following features:

- Ability to support reduced instruction set computer (RISC) and DSP operating systems
- TI925T MPU subsystem with:
  - Instruction cache (16K bytes) and data cache (8K bytes)
  - Memory management unit (MMU)
  - A 17-word write buffer (WB)
- DSP subsystem (C55x™ DSP core and subsystems) with:
  - Internal 32K-word dual-access RAM (DARAM), 48K-word single access RAM (SARAM), 16K-word ROM
  - Software-configurable instruction cache (12K words, 128-bit line size, 2-way set-associative + RAM set)
  - Hardware accelerators for video processing, pixel interpolation, and motion estimation
  - Six-channel DMA controller for high-speed data movement without DSP intervention
- DSP MMU for address translation and access permission checks
- System DMA controller with:
  - Six ports and nine independently programmable generic channels
  - An additional dedicated DMA channel tied to the liquid crystal display (LCD) controller
  - Ability to transfer 8-,16-, or 32-bit data between the external memory, the MPU, and peripherals with byte alignment and packing capability
  - Ability to perform simultaneous transfers (single or multiple burst) if no resources conflict
  - Low-power design (no clocking when idle)
- Two external memory interfaces that allow glueless hookup to:
  - A 16-bit bus interface to external memory interface slow (EMIFS), such as flash/SRAM/ROM/page-mode ROM/SB flash/DPRAM), with 128M bytes of memory space
  - A 16-bit bus interface to external memory interface fast (EMIFF), such as memory SDRAM, with 64M bytes of memory space

- 192K bytes of 32-bit-wide internal SRAM memory that allows local storage of operating system (OS) critical routines and that provides a direct path from the SRAM to the LCD controller
- An external memory traffic controller (TC) that allows asynchronous operation among the external memory interface, the MPU, and the DSP
- Mailboxes (two for MPU-to-DSP and two for DSP-to-MPU) for interprocessor communication
- Endianism conversion (default bypass, selectable, and configurable) between the DSP and the traffic controller and between the DSP and the MPU interface (MPUI) port boundaries
- Elastic buffering between the traffic controller and the MPU/DSP controllers to facilitate fully synchronous and synchronous scalable mode clock operations
- JTAG port for test, debug, and emulation
- Clock management:
  - One digital phase-locked loop (DPLL) and three clock management units for MPU, DSP, and traffic controller clock generation and management
  - System power management for idle mode and power-down functions
- Peripherals available for the OS, general-purpose housekeeping, and application-specific functions:
  - For the MPU:
    - Three 32-bit timers
    - A 16-bit watchdog timer
    - An interrupt handler
    - An LCD controller
    - Configuration registers
    - McBSP2 (multichannel buffered serial port)
    - Inter-integrated circuit (I^2C) interface
    - MicroWire interface
    - Keyboard interface
    - Universal serial bus (USB) function and host interface

- Camera interface
- Five MPUIO general-purpose input/output signals in default multiplexing mode; five more available through alternative pin multiplexing modes
- 32-kHz timer
- Pulse-width tone (PWT) module
- Pulse-width light (PWL) module
- Real-time clock (RTC) module
- Multimedia card (MMC) or serial data (SD) card interface
- HDQ and 1-Wire serial interface
- Two light emitting diode (LED) pulse generator modules
- Frame adjustment counter

- For the DSP:
  - Three 32-bit timers
  - A 16-bit watchdog timer
  - An interrupt handler
  - McBSP1: Multichannel buffered serial port
  - McBSP3: Multichannel buffered serial port
  - MCSI1: Multichannel serial voice interface
  - MCSI2: Multichannel serial voice interface
- Shared peripherals:
  - UART1: UART modem with autobaud (16C750 compatible)
  - UART2: UART modem with autobaud (16C750 compatible)
  - UART3: UART modem with IrDA (16C750 compatible)
  - Fourteen general-purpose input/output (GPIO)
  - Mailbox

1.4 Architecture

The OMAP5910 device includes the MPU subsystem, the DSP subsystem, a memory interface traffic controller, general-purpose peripherals, dedicated multimedia application (MMA) peripherals, and multiple interfaces. The MPU is the master of the platform, and it has access to the entire 16M bytes of memory space and to the 128K bytes of I/O space of the DSP subsystem. Additionally, the MPU and DSP share access to the internal SRAM and external memory interfaces.

1.5 Memory Maps

Figure 1–3 shows the MPU memory map. Figure 1–4 shows the DSP memory map.

Figure 1–3. MPU Memory Map

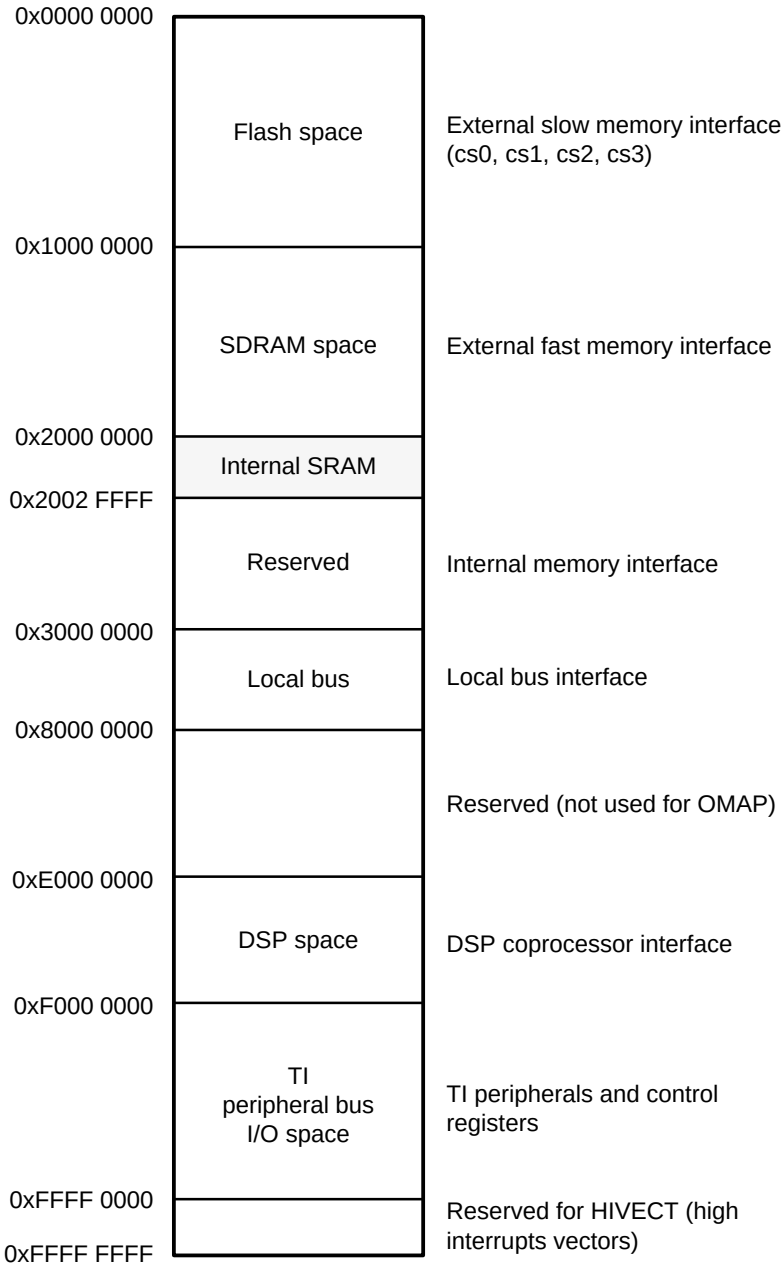
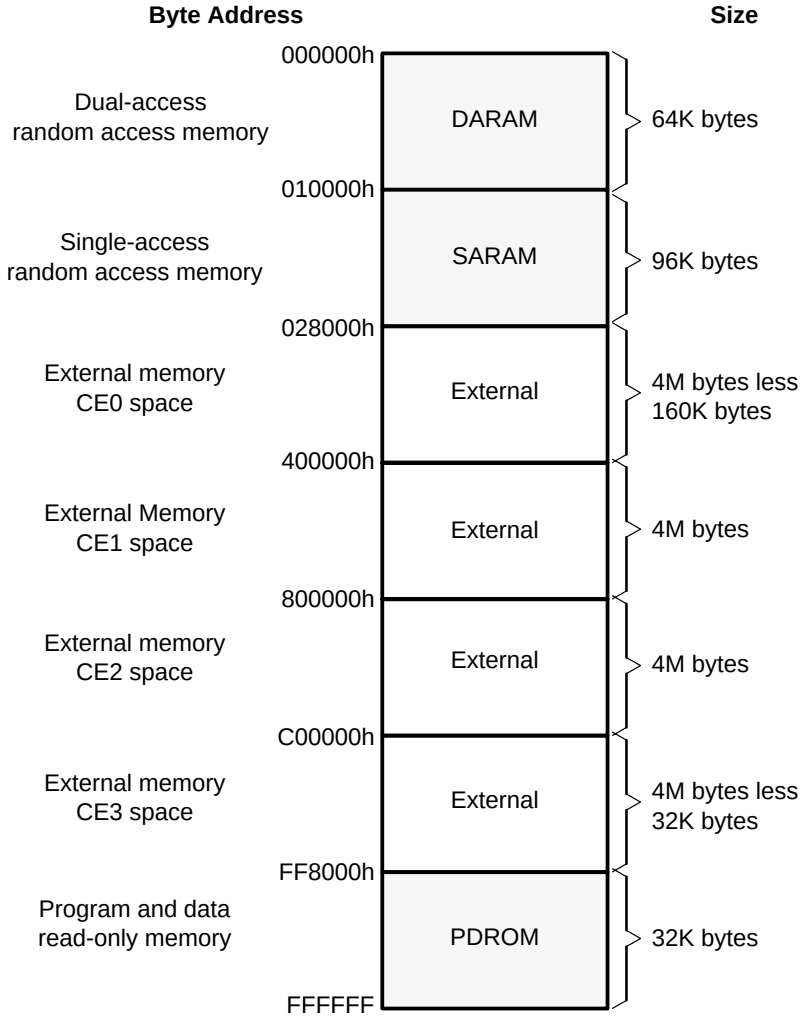


Figure 1–4. DSP Memory Map



The programmable DSP MMU configures how the DSP external address range is physically mapped to the MPU address range. For more information, see Section 2.8, *DSP Memory Management Unit*.

1.6 Software Compatibility

Code compatibility with future OMAP59x devices is only possible if driver writers adhere to the conventions detailed in Section 1.6.1.

1.6.1 OMAP Driver Compatibility Conventions

All locations marked as reserved or unused in the documentation are written as zero, and, in general, values read from reserved locations are not used.

In practice, read-mask-update operations can be applied to registers that include reserved bits, provided the register is initialized by writing 0 to all reserved bits when the register is first used.

These conventions allow use of reserved bits to enable new features in future implementations. Initialization of the complete register, including reserved bits, is required to avoid problems in these future devices when a new driver uses (sets) some of the previously reserved bits. In this case, a following legacy driver must clear the bits.

All software that comes into contact with hardware registers in OMAP devices must follow these conventions.

Chapter 2

MPU Subsystem

This chapter describes the core, caches, memory management units (MMUs), interface, and bridges of the OMAP5910 multimedia processor microprocessor unit (MPU) subsystem.

Topic **Page**

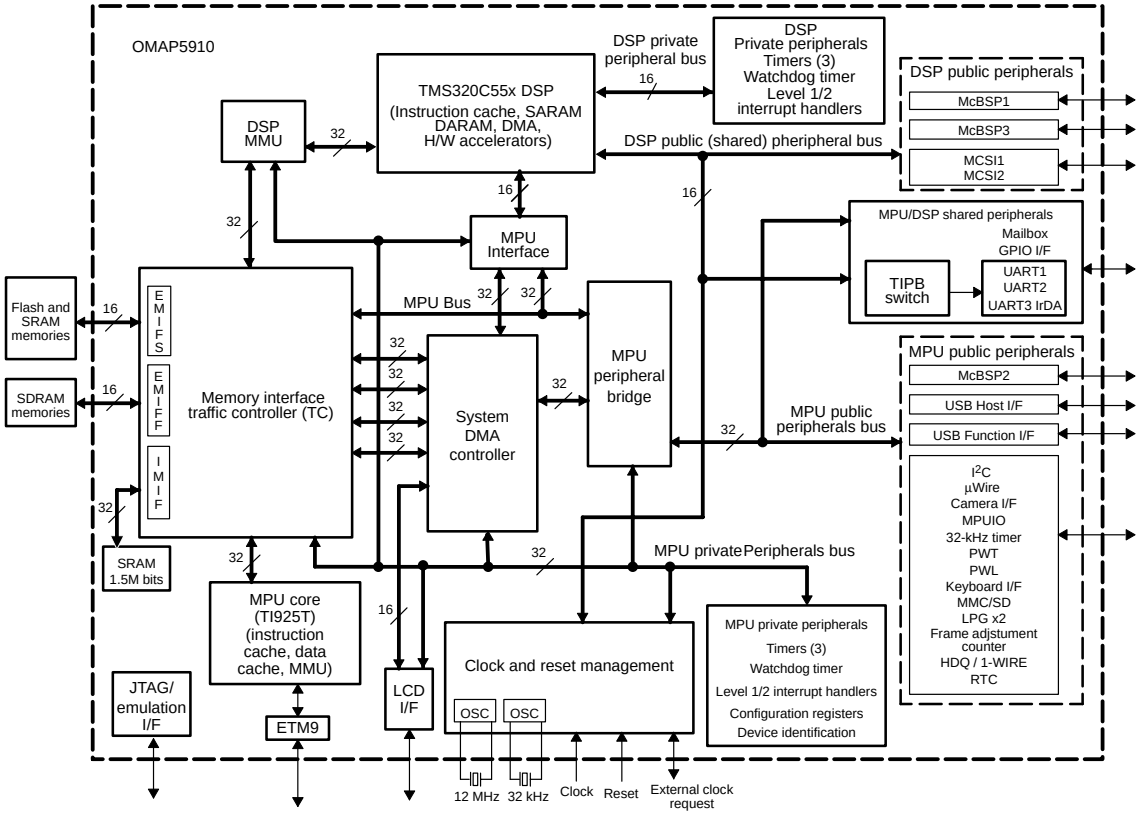
2.1 Introduction

The MPU of the OMAP5910 device controls the memory management units (MMUs), the system direct memory access (DMA) controller, the MPU TI peripheral bus (TIPB) bridge, and peripherals.

Figure 2–1 shows the OMAP5910 device with the MPU subsystem highlighted. The subsystem contains the following components:

- MPU core (see Section 2.2, *MPU Core*)
- Traffic controller (see Chapter 4, *Memory Interface Traffic Controller*)
- MPU MMU (see Section 2.7, *MPU Memory Management Unit*)
- DSP MMU (see Section 2.8, *DSP Memory Management Unit*)
- System DMA controller (see Chapter 5, *System DMA Controller*)
- LCD controller (see Chapter 11, *LCD Controller*)
- MPU TIPB bridge (see Section 2.10, *MPU TI Peripheral Bus Bridges*)
- Clock manager (see Chapter 15, *Clock Generation and System Reset Management*)
- Interrupt handler (see Section 6.4.1, *MPU Level 1 Interrupt Handler*, Section 6.4.2, *MPU Level 2 Interrupt Handler,* and Section 8.4, *[DSP] Interrupt Handlers*)
- Timers (see Section 6.2, *[MPU] Timer Description* and Section 8.2, *[DSP] Timers*)
- Watchdog timer (see Section 6.3, *[MPU] Watchdog Timer* and Section 8.3, *[DSP] Watchdog Timer*)
- Interprocessor communication (see Section 10.2, *Interprocessor Communication*)
- 1.5M-bit SRAM internal memory

Figure 2–1. Highlight of MPU Subsystem



2.2 MPU Core

The MPU core is a TI925T reduced instruction set computer (RISC) processor. The TI925T is a 32-bit processor core that performs 32-bit or 16-bit instructions and processes 32-bit, 16-bit, or 8-bit data. The core uses pipelining so that all parts of the processor and memory system can operate continuously.

The MPU core incorporates:

- A coprocessor 15 (CP15) and protection module
- Data and program memory management units (MMUs) with table look-aside buffers.
- A separate 16K-byte instruction cache and 8K-byte data cache. Both are two-way associative with virtual index virtual tag (VIVT).
- A 17-word write buffer (WB)
- A local bus interface

The OMAP5910 device uses the TI925T core in little endian mode only.

To reduce effective memory access time, the TI925T has an instruction cache, a data cache, and a write buffer. In general, these are transparent to program execution.

2.3 Instruction Cache

The 16K-byte instruction cache (I-cache) has 1024 lines of 16 bytes arranged as a two-way set-associative cache. It uses the virtual addresses generated by the processor core. The I-cache is always reloaded one line at a time. It can be enabled or disabled via the CP15 control register (I_CP15 bit) and is disabled and flushed upon reset.

Disabling the I-cache does not invalidate it.

You can enable the I-cache independently from the MMU.

2.3.1 Operation

When the I-cache is enabled, it is searched whenever the processor requests an instruction. If the cache hits, data is returned to the core whether the MMU is enabled or not. If a cache read misses, a line fetch is performed and data is written to the cache following a least recently used (LRU) replacement algorithm. For best performance, enable the I-cache as soon as possible after reset. If the I-cache is disabled, it is not searched. All instruction fetches generate a single 16-bit or 32-bit external access. An instruction miss generates line load.

2.3.2 Validity

The flush I-cache instruction is fetched at cycle time 0, for example, but not executed until cycle time 4 (the TI925T uses a five-stage opcode pipe). Thus, four additional opcodes potentially are still fetched from the I-cache before the flush I-cache opcode is executed. Once executed, the entire I-cache is invalidated before the next opcode executes. Typically, four non-opcodes following the CP15 instruction flush the cache to avoid confusion.

The I-cache content is not flushed when the I-cache is disabled. Its contents remain valid and are accessible again when the I-cache is reenabled.

2.4 Data Cache

The 8K-byte data cache (D-cache) has 512 lines of 16 bytes arranged as a two-way set-associative cache. It uses the virtual addresses generated by the processor. The D-cache is always reloaded one line at a time, because it always requires the MMU to be enabled. The MMU can operate in write-through (WT) or in copy-back (CB) mode. The translation lookaside buffer (TLB) descriptors that are placed in memory determine which mode is used.

You can enable or disable the D-cache via the CP15 control register: the D-cache is disabled and flushed upon reset. The D-cache supports byte, half-word, and word accesses.

The D-cache is always disabled when the MMU is off.

2.4.1 D-Cache Operation

If the D-cache is enabled (C_CP15 = 1), it is searched whenever the processor performs a data load or store. If the cache hits on a load, data is returned to the core regardless of the C_MMU bit. If a cache read misses, the C_MMU bit is examined. If it is 1, a line fetch is performed and the line is written to the cache following an LRU (least recently used) replacement algorithm. If C_MMU is 0, a single external access is performed and the cache is not updated. Stores that hit the D-cache always update it, regardless of the C_MMU bit, to keep the D-cache contents consistent with the external memory. Stores that miss do not update the D-cache (see Table 2–1).

Table 2–1. Data Cache Configuration

C_CP15	C_MMU	B_MMU	Functional Description
0	X	X	No cache search
1	0	X	Cache search active • Read and write misses are not cached. • Cache serves read hits. • Write hits update the cache. • Read misses and writes generate external accesses.

Note: The load multiple instruction does not perform a burst read.

Table 2–1. Data Cache Configuration (Continued)

C_CP15	C_MMU	B_MMU	Functional Description
1	1	0	Cache search active: write through mode (WT) • Read hits do not generate external accesses. • Write hits update the cache and the external memory. • Read misses cause a line load. • Write misses generate external accesses.
1	1	1	Cache search active: copy-back mode (CB) • Read and write hits do not perform external accesses. • Read misses cause a line load. • Write misses do not update the cache, and they generate an external access.

Note: The load multiple instruction does not perform a burst read.

If C_CP15 = 0, the D-cache is disabled and it is not searched. If a memory region is changed from cacheable to noncacheable and data must come from external memory, the cache must be flushed.

2.4.2 Validity

The D-cache always requires that the MMU be enabled, so virtual addresses are always in use. The TLB descriptors in memory can be cached or not cached. When software is switching virtual address maps, take care to invalidate the data cache so that the wrong data value is not returned (that is, so that a false D-cache hit does not occur). To do this, the CP15 register allows software to invalidate the entire D-cache. As noted before, disabling the D-cache and reenabling it does not invalidate it.

If CB mode is used (see Table 2–1), software must first clean the cache to make it coherent with main memory (this is not necessary in WT mode, because main memory is continuously updated as the data cache is used).

For CB mode, the VIVT algorithm must be used if software is to avoid missing interrupts during the clean operation. Timer interrupts, for example, can be missed.

To avoid this problem, the hardware clean operation can be interrupted, so the software algorithm must check the min/max registers (CP15 registers) to determine if the clean operation has completed. If not, it must repeat the operation until complete.

Note:

Cleaning is not the same as flushing.

The entire D-cache can be invalidated with a single flush D-cache instruction through the CP15 cache operation register. The D-cache is flushed upon reset.

If the D-cache is disabled, its content is maintained valid and is accessible when the cache is reenabled.

2.4.3 Double-Mapped Space

The D-cache works with virtual addresses, and it is assumed that every virtual address maps to a different physical address. If more than one virtual address corresponds to the same physical location, the cache cannot maintain its consistency because each virtual address has a separate entry in the cache and only one entry is updated on a processor write operation. To avoid any cache inconsistency, double-mapped virtual addresses must be marked as uncacheable.

2.5 Write Buffer

The write buffer (WB) increases system performance and can buffer up to seventeen 32-bit words of data. The MMU attributes B (B_MMU) and C (C_MMU) (which are part of the TLB descriptor) and the CP15 control register W bit (W_CP15) control WB behavior.

Clearing W_CP15 and C_CP15 upon reset ensures that all accesses are non-bufferable until the MMU is enabled. To use the write buffer, you must enable the MMU. However, you can enable the two functions simultaneously with a single write to the CP15 control register.

The write buffer is always disabled when the MMU is off.

Clearing bit 3 in the CP15 control register disables the write buffer. Any writes already in the write buffer complete normally.

It is not possible to abort buffered writes externally, because the s_abort external signal is ignored and data is simply discarded. Areas of memory that can generate aborts must be marked as unbufferable in the MMU page tables.

2.5.1 Operation

The WB operation is controlled by four control bits, as shown in Table 2–2.

Table 2–2. Write Buffer Configuration

C_CP15	W_CP15	C_MMU	B_MMU	Functional Description
0	0	X	x	
0	1	X	0	Writes are not buffered.
1	0	X	x	See Note
1	1	0	0	
0	1	X	1	Noncacheable, buffered (NCB)
1	1	0	1	NCB
1	1	1	0	Writes are buffered, write-through mode.
1	1	1	1	Writes are buffered, copy-back mode.

Note: In copy-back mode with the WB disabled (1011 configuration), dirty lines are saved to the external memory via the WB regardless of W_CP15. Write misses go directly to the external memory. If the WB is disabled and the system is configured in copy-back mode, only write misses stall the system.

When writes are not buffered, the processor stalls until the external write access is complete.

2.5.2 SWAP Instruction

When bit L of the CP15 TI925T configuration register is set, the write phase of the SWAP instruction (interlocked read-write) is treated as unbuffered when data belongs to an noncacheable, nonbuffered (NCNB) or NCB region, even if it is marked as buffered. The S_LOCK signal is active through the read and write accesses. If the read of the SWAP instruction hits the cache, S_LOCK is asserted during the read despite the fact that no external access is performed. The write is performed both in the cache and externally with S_LOCK active.

For WT- or CB-mode regions, S_LOCK is not active and accesses are performed like ordinary read or write accesses.

When bit L of the CP15 TI925T configuration register is reset, S_LOCK stays low during the SWAP instruction regardless of the memory region type (NCNB, NCB, WT, or CB). If marked as buffered, data is written to the write buffer and reaches the system bus after an undetermined delay.

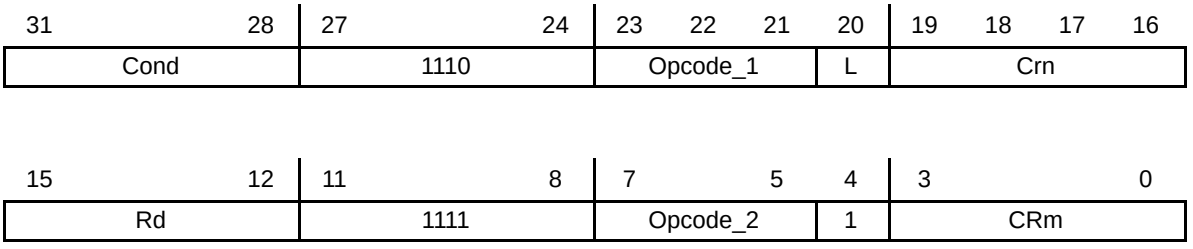
2.6 Coprocessor 15

TI925T operation and configuration are controlled with coprocessor instructions, configuration pins, and the MMU translation tables. The coprocessor instructions manipulate on-chip registers, which control the configuration of the cache memories, write buffer, MMU, and a number of other options described in the following sections.

2.6.1 CP15 Access

The CP15 defines 16 registers. Table 2–3 shows the registers available for reading and for writing. While most registers are used to control various operations, some, such as register 0, only provide information. MRC and MCR instructions can access CP15 registers in privileged mode only. Figure 2–2 contains the instruction bit pattern of the MCR and MRC instructions.

Figure 2–2. MRC, MCR Bit Pattern



The CRn field specifies the coprocessor register to access. The CRm field and opcode_2 fields specify a particular action when addressing some registers or shadow registers. The TI925T takes the undefined instruction trap upon executing CDP, LDC, STC, and unprivileged MCR/MRC instructions on CP15.

2.6.2 Register Descriptions

The following terms and abbreviations are used throughout the register descriptions:

- Unpredictable (UNP): Reading from such a location returns data of unpredictable value. Writing to this location causes unpredictable behavior or an unpredictable change in device configuration.
- Undefined (UND): Any access to such registers makes TI925T take the undefined instruction trap.
- Should be zero (SBZ): All bits written to this field must be 0.

- Ignored: Writing to such a location does not affect the system behavior.
- VA: Virtual address (data or instruction)

In all cases, reading data values from or writing any data values to any CP15 register, including those fields specified as unpredictable or SBZ, causes no permanent damage to the TI925T.

Table 2–3. CP15 Register Summary

Register	Reads	Writes	Access	RD
0	ID register	Ignored	Read-only	31..0
1	Control register	Control register	Read/Modify/Write	14..0
2	Translation table base	Translation table base	Read/Write	31..14
3	Domain access control	Domain access control	Read/Write	31..0
4	Unpredictable	Ignored		-
5	Fault status	Fault status	Read/Write	8..0
6	Fault address	Fault address	Read/Write	31..0
7	Unpredictable	Cache operations	Write-only	31..0
8	Unpredictable	TLB operations	Write-only	31..0
9	Unpredictable	Ignored		-
10	TLB lock-down	TLB lock-down	Read/Write	31..0
11	Unpredictable	Ignored		-
12	Unpredictable	Ignored		-
13	PID	PID	Read/Write	31..25
14	Unpredictable	Ignored		-
15	TI operations	TI operations	Read/Write	31..0

2.6.2.1 ID Register and Cache Information Register

Reading from CP15 register 0 returns either an identification defined by architecture and implementation for the processor or information on the cache, depending on the op-code_2 used. CRm SBZ when reading.

Writing to register 0 is ignored.

Table 2–4. Reading From CP15 Register 0

Function	Opcode_2	CRm	Rd	Instruction
Read ID†	0bXXX	0bXXXX	TI925T ID	MRC p15, 0, Rd, c0, c0, 0
Read CIR	0b001	0b0000	Cache info	MRC p15, 0, Rd, c0, c0, 1

† All opcodes [opcode_2,CRm] except [1,0] return the TI925T ID.

Table 2–5. CP15 ID Register

Bit	Name	Function
31–24	Implementers	Contains the ASCII code of the implementer trademark (0x54 = Texas Instruments)
23–16	Architecture version	Contains the architecture version (0x02 Version v4T)
15–4	Part number	Contains a 3-digit part number in binary-coded decimal format.The OS bit O in the TI925T configuration register sets the value of these fields as follows: 915 in TI925T mode 925 in Windows CE mode
3–0	Reserved	Contains the microprocessor revision number 2

Table 2–6. CP15 Cache Information Register (CIR)

Bit	Name	Value	Function
31–29	Reserved	0	Read as 0.
28–25	Cache type		Cache type: read as 0010. The cache provides clean-cache entry and flush-cache-entry with a cache index in addition of the operations with virtual address (also called clean-cache-step or flush-cache-step). The format of the clean-cache-entry is given in the *Register 7: Cache Operations* section.
24	ID	0	Unified I-/D-cache
		1	Harvard cache
23–21	Reserved	0	Read as 0.
20–18	D-cache information		Base value of D-cache size (same format as for I-cache)
17–15	D-cache information		Base value of D-cache associativity (same format as for I-cache)

Table 2–6. CP15 Cache Information Register (CIR) (Continued)

Bit	Name	Value	Function
14	D-Cache information		Parameter to calculate the real D-cache associativity and size:
		0	D-cache associativity and D-cache size = base value
		1	D-cache associativity and D-cache size = 3/2 of the base value. Exception: If base value of associativity is 1, a 1 indicates that there is no D-cache and 0 indicates that D-cache is really direct-map.
13–12	D-cache information		Indicate line length of D-cache (same format as for I-cache)
11–9	Reserved	0	Read as 0.
8–6	I-cache information		Base value of I-cache size:
		0000	512 bytes
		0001	1K byte
		0010	2K bytes
		0011	4K bytes
		0100	8K bytes
		0101	16K bytes
		0110	32K bytes
		0111	64K bytes
			Note: 2 (bits 8–6 – bits 5–3 – bits 1–0) gives the number of lines.
5–3	I-cache information		Base value of I-cache associativity:
		0000	Direct map
		0001	2-way associative
		0010	4-way associative
		0011	8-way associative
		0100	16-way associative
		0101	32-way associative
		0110	64-way associative
		0111	128-way associative

Table 2–6. CP15 Cache Information Register (CIR) (Continued)

Bit	Name	Value	Function
2	I-cache information		Parameter to calculate the real I-cache associativity and size:
		0	I-cache associativity and I-cache size are equal to the base value.
		1	I-cache associativity and I-cache size are equal to 3/2 of the base value. Exception: If base value of associativity is 1, a 1 indicates here that there is no I-cache; 0 indicates that I-cache is really direct-map.
1–0	I-cache information		Indicates line length of the I-cache:
		00	8 bytes
		01	16 bytes
		10	32 bytes
		11	64 bytes

This register specifies the configuration of the TI925T core. It is recommended that the register be written using a read-modify-write routine.

Reading from CP15 register 1 reads the control bits. The CRm and opcode_2 fields are ignored when reading CP15 register 1, but must be zero.

Writing to CP15 register 1 sets the control bits. The CRm and opcode_2 fields are not used when writing CP15 register 1, but must be zero.

All control bits but V are set to zero upon reset.

Table 2–7. CP15 Control Register

Bit	Name	Value	Function
31–15			Reserved: Do not rely on any particular value in these bit locations during a read (ensure they are masked properly). Write these bits as zero.
14		0	Read as 0. Write is ignored.
13	V		Alternate vector select. Sets the address of the exception vector from address 0x00000000 to 0x0000001F when at zero and from 0xFFFF0000 to 0xFFFF001F when at 1. This bit takes the value of the HIVECS signal port upon reset. After reset, it can be changed by software.

Table 2–7. CP15 Control Register (Continued)

Bit	Name	Value	Function
12	1		Instruction cache enable/disable
		0	Instruction cache disabled
		1	Instruction cache enabled
11–10		0	Read as 0. Write is ignored.
9	R		ROM protection. This bit modifies the MMU protection system (see Table 2–24).
8	S		System protection. This bit modifies the MMU protection system (see Table 2–24).
7	B		Little/big endian configuration. The TI925T on the OMAP5910 device supports only little endian mode due to the system architecture of the device. This bit must always be written as 0.
		0	Little endian
		1	Reserved (do not use)
6–4		1	Read as 1. Write is ignored.
3	W		Write buffer enable/disable
2	C		Data cache enable/disable
		0	Data cache disabled
		1	Data cache enabled
1	A		Alignment fault enable/disable
		0	Address alignment fault checking disabled
		1	Address alignment fault checking enabled
			Note: The alignment is checked only on data; code alignment is always on a 32-bit boundary. If address alignment fault is enabled, words must be word-aligned, and half-words must be half-word-aligned.

Table 2–7. CP15 Control Register (Continued)

Bit	Name	Value	Function
0	M		Memory management unit (MMU) enable/disable
		0	MMU disabled
		1	MMU enabled
			The MMU must be enabled before or at the same time as the data cache (C) and write buffer (W). The instruction cache can be enabled independently. When the MMU is disabled and no address translation occurs, the D-cache and write buffer are forced OFF.

Note:

Care must be taken if the translated address differs from the non-translated address, because the instructions following the enabling of the MMU are fetched using no address translation. Enabling the MMU may be considered as an instruction with delayed execution. A similar situation occurs when the MMU is disabled.

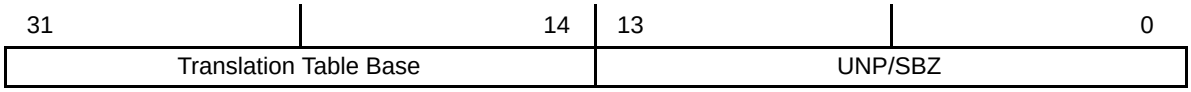
The following code segment example shows correct MMU enabling which takes into account the latency to transition to virtual addressing:

```
ldr r0, =bVirtualStart; Load r0 with virtual jump
location; Enable the MMU.
   mrc p15, 0, r1, c1, c0, 0; Read the control register.
   orr r1, r1, #BIT0; Set the M bit to enable MMU.
   nop
   mcr p15, 0, r1, c1, c0, 0; Write the control register.
   mov pc, r0; Jump to the virtual address.
   nop
bVirtualStart
   nop
   nop
```

The MMU, I-cache, and D-cache can be enabled or disabled independently. If the data cache or write buffer are enabled when the MMU is not enabled, the data cache and the write buffer stay off, preventing invalid combinations.

The functions MMU, D-cache, I-cache, and WB can be enabled simultaneously with a single write to the control register.

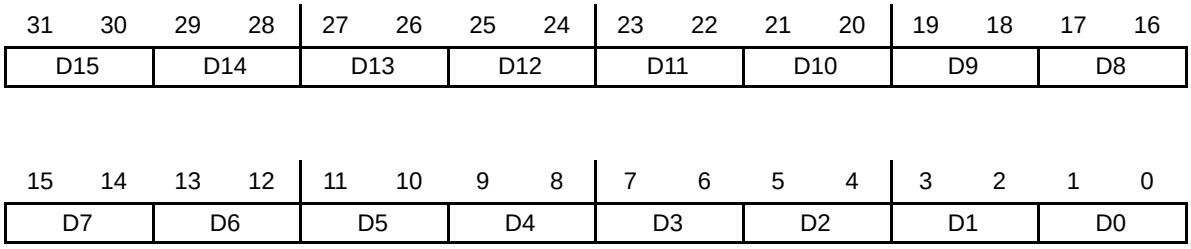
Figure 2–3. Format of the CP15 Translation Table Base Register



Reading from CP15 register 2 returns the pointer to the currently active first-level translation table in bits 31–14 and an unpredictable value in bits 13–0. The CRm and opcode_2 fields are SBZ when reading this register.

Writing to CP15 register 2 updates the pointer to the currently active first-level translation table from the value in bits 31–14 of the written value. Bits 13–0 must be written as zero. The CRm and opcode_2 fields are SBZ when writing to this register.

Figure 2–4. Format of the CP15 Domain Access Control Register



Reading from CP15 register 3 returns the value of the domain access control register. The CRm and opcode_2 fields are SBZ when reading this register.

Writing to CP15 register 3 writes the value of the domain access control register.

The CRm and opcode_2 fields are SBZ when writing to this register.

The domain access control register consists of sixteen two-bit fields, each defining the access permissions for one of the 16 domains (D15-D0). Table 2–8 gives more details on the meaning of each field.

Data, instructions, or both can use each of these domains. Two basic kinds of users are supported: clients and managers.

Table 2–8. Domain Configuration

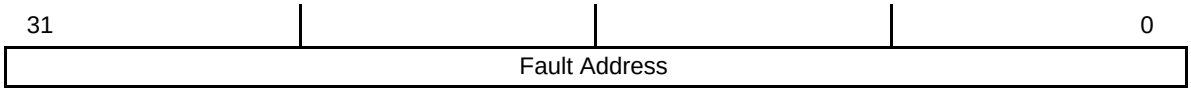
Value	Access Type	Description
0b00	No access	Any access generates a domain fault.
0b01	Client	Access rights are checked against the permission given by the page descriptor.
0b10	Reserved	Behaves like no access
0b11	Manager	The access rights are not checked; permission faults cannot be generated.

Reading CP15 register 5 returns the value of the fault status register (FSR). The FSR contains the source of the last data fault. Only the bottom 9 bits are returned. The top 23 bits are unpredictable. The FSR indicates the domain and type of access being attempted when an abort occurred.

Table 2–9. CP15 Fault Status Register

Bit	Name	Function
31–9	UNP/SB	Reserved: Do not rely on any particular value in these bit locations during a read (ensure they are masked properly). Write these bits as zero.
8	0	Read as 0.
7–4	Domain	Specify which of the 16 domains (D15–D0) was being accessed when the last fault occurred.
3–0	Status	Indicate the type of fault due to the last access being attempted. The encoding of these bits is shown in Table 2–23, *Priority Encoding of the Fault Status Register*. The FSR is only updated for data access faults, not for instruction fetch faults. When a fault occurs during a load or store multiple (LDM or STM instructions), the FSR records the domain corresponding to the first fault caused by LDM or STM. For example, an LDM performing 12 accesses may cross a page boundary with, say, four accesses in one page and eight in the next page. If accessing the second page causes an abort, the FSR and FAR record the information related to the fifth access. The CRm and opcode_2 fields are SBZ when reading this register. Writing CP15 register 5 sets the fault status register to the value of the data written. The upper 24 bits written are SBZ. The CRm and opcode_2 fields are SBZ when writing to this register.

Figure 2–5. Format of the Fault Address Register



Reading CP15 register 6 returns the value of the fault address register (FAR). The FAR holds the virtual address of the access that was attempted when a fault occurred. The FAR is only updated for data access faults, not for instruction fetch faults. When a fault occurs during a load or store multiple (LDM or STM instructions), the FAR records the domain corresponding to the first fault caused by LDM or STM (see example in FSR section above).

The CRm and opcode_2 fields are SBZ when reading this register. Writing CP15 register 6 sets the fault address register to the value of the data written. The CRm and opcode_2 fields are SBZ when writing to this register.

2.6.2.2 Cache Operations

The CP15 register 7 is a write-only pseudoregister managing the instruction and data caches. Several cache operations are defined and are selected by the opcode_2 and CRm fields.

Table 2–10. Cache Operations

Function	Opcode_2	CRm	Rd	Instruction
Flush I- and D-cache	0b000	0b0111	SBZ	MCR p15, 0, Rd, c7. c7, 0
Flush I-cache (1)	0b000	0b0101	SBZ	MCR p15, 0, Rd, c7, c5, 0
Flush I-cache entry	0b001	0b0101	VA	MCR p15, 0, Rd, c7, c5, 1
Flush D-cache (1, 2)	0b000	0b0110	SBZ	MCR p15, 0, Rd, c7, c6, 0
Flush D-cache entry (2)	0b001	0b0110	VA	MCR p15, 0, Rd, c7, c6, 1
Clean D-cache entry	0b001	0b1010	VA	MCR p15, 0, Rd, c7, c10, 1
Clean and flush D-cache entry	0b001	0b1110	VA	MCR p15, 0, Rd, c7, c14, 1
Flush D-cache entry (2)	0b010	0b0110	Set/Index (3)	MCR p15, 0, Rd, c7, c6, 2

Notes:
1) Flush I- and D-cache operations invalidate all entries in the I-cache and D-cache respectively. The flush D-cache also discards any dirty lines present in the D-cache.
2) The flush D-cache and D-cache entry operations do not clean the D-cache entries before they are invalidated. A clean and flush D-cache requires two cache operations; there is a specific operation for cleaning and flushing a D-cache entry at once. First clean then flush the entire cache; this requires two CP15 operations (bear in mind the VIVT clean algorithm). You can clean and flush individual entries in one CP15 operation.
3) Figure 2–6 shows the format of the Rd value for all D-cache operations on a single entry.
4) TI925T supports high performance full cache clean operation with the VIVT algorithm.

Table 2–10. Cache Operations (Continued)

Function	Opcode_2	CRm	Rd	Instruction
Clean D-cache entry	0b010	0b1010	Set/Index (3)	MCR p15, 0, Rd, c7, c10, 2
Clean and flush D-cache entry	0b010	0b1110	Set/Index (3)	MCR p15, 0, Rd, c7, c14, 2
Clean D-cache (4)	0b000	0b1010	SBZ	MCR p15, 0, Rd, c7, c10, 0
Prefetch I-line	0b001	0B1101	VA	MCR p15, 0, Rd, c7, c13, 1
Wait-for-interrupt	0b100	0b0000	SBZ	MCR p15, 0, Rd, c7, c0, 4
Drain write buffer	0b100	0b1010	SBZ	MCR Rd, c7, c10, 4

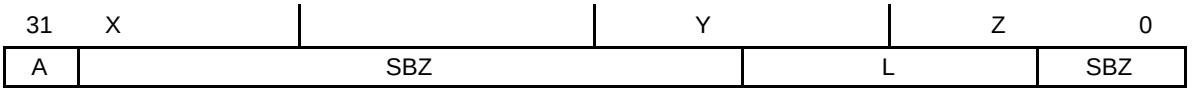
Notes: 1) Flush I- and D-cache operations invalidate all entries in the I-cache and D-cache respectively. The flush D-cache also discards any dirty lines present in the D-cache.

2) The flush D-cache and D-cache entry operations do not clean the D-cache entries before they are invalidated. A clean and flush D-cache requires two cache operations; there is a specific operation for cleaning and flushing a D-cache entry at once. First clean then flush the entire cache; this requires two CP15 operations (bear in mind the VIVT clean algorithm). You can clean and flush individual entries in one CP15 operation.

3) Figure 2–6 shows the format of the Rd value for all D-cache operations on a single entry.

4) TI925T supports high performance full cache clean operation with the VIVT algorithm.

Figure 2–6. D-Cache Clean/Flush Single Entry Operand Format



There are two valid fields. The A-field depends on the level of associativity of the cache. The L-field depends on the number of lines per set.

The CIR register (cache information) provides all the information to calculate x, y, and z following the equations below:

$x = 32 - CIR[17-15] - CIR[14]$

$y = 8 + CIR[20-18] - CIR[17\text{-}15]$

$z = CIR[13-12] + 3$

In addition, one bit (D-cache clean entry mode) of TI925T configuration register allows cleaning of one entry in both sets at a time. D[31] becomes *don't care* when the D-cache clean entry mode is zero (see CP15 register 15 description). In this mode, the software clean operation just cleans one block of virtual memory with an increment corresponding to the line size.

2.6.2.3 TLB Operations

The CP15 register 8 is a write-only pseudoregister that manages the translation look-aside buffers (TLBs). TI925T includes separate instruction and data TLBs. Several TLB functions are defined and are selected by the opcode_2 and CRm fields.

The flush-I and flush-D functions, respectively, flush (invalidate) all unpreserved entries of the instruction and data TLB.

The flush entry functions flush a single entry of the TLB corresponding to the virtual address present in Rd, regardless of its state (preserved/unpreserved).

All unused values of opcode_2 and CRm are ignored. Reading register 8 is ignored.

Table 2–11. TLB Operations

Function	Opcode_2	CRm	Rd	Instruction
Flush I TLB	0b000	0b0101	SBZ	MCR p15, 0, Rd, c8, c5, 0
Flush I TLB entry	0b001	0b0101	VA	MCR p15, 0, Rd, c8, c5, 1
Flush D TLB	0b000	0b0110	SBZ	MCR p15, 0, Rd, c8, c6, 0
Flush D TLB	0b001	0b0110 VA	VA	MCR p15, 0, Rd, c8, c6, 1
Flush I + D TLB	0b000	0b0111	SBZ	MCR p15, 0, Rd, c8, c7, 0

2.6.2.4 TLB Lock-Down Registers

There is a TLB lock down register for both TLBs; the value of opcode_2 determines which TLB register is accessed.

- Opcode_2 = 0 selects the register associated with the D-TLB.
- Opcode_2 = 1 selects the register associated with the I-TLB.

Each TLB has its own victim counter. These registers and counters are set to zero upon reset.

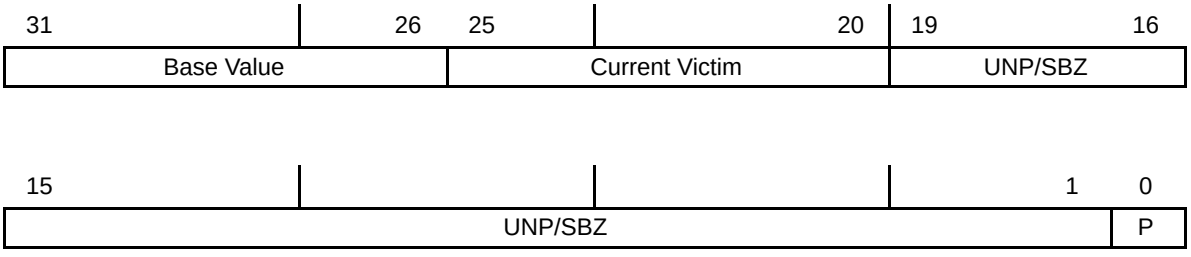
Reading register 10 returns the value of the TLB victim counter base value register, the current value of the victim counter, and the state of the preserved bit. Bits 20–1 are unpredictable when read.

Writing to register 10 updates the base value, the current victim pointer, and the preserved register value. Bits 20–1 are ignored on write but SBZ.

Table 2–12. Lockdown Operations

Function	Opcode_2	CRm	Data	Instruction
Read D-TLB lock	0b000	0b0000	Value	MRC p15, 0, Rd, c10, c0, 0
Write D-TLB lock	0b000	0b0000	Value	MCR p15, 0, Rd, c10, c0, 0
Read I-TLB lock	0b001	0b0000	Value	MRC p15, 0, Rd, c10, c0, 1
Write I-TLB lock	0b001	0b0000	Value	MCR p15, 0, Rd, c10, c0, 1

Figure 2–7. Format of the Lock-Down Registers



Loading of the TLB is managed by a victim counter, which counts from the programmed base value up to 63. Therefore, some pages or sections can be locked inside the TLB if loaded between the entry 0 and the entry pointed to by the base value register.

Flush operations invalidate both locked and non-locked entries. An entry can also be maintained in the TLB during a global flush if the preserved bit was set during the loading of this entry in the TLB. A flush entry operation invalidates a TLB entry regardless of its state (preserved/unpreserved).

The flush operation does not modify the base value register but reinitializes the victim counter to the base value.

The following code sequence locks a page/section in entry 3:

```
{flush page/section from TLB}
MCR p15, 0, Rd, c10, c0, 1
Rd content indicates base value = 4, current victim = 3
MRC p15, 0, Rd, c7, c13, 1
```

Prefetch I-line with VA in Rd generates a miss TLB that loads entry 3 (victim counter is automatically updated to 4).

2.6.2.5 Context Switch (or PID: Process Identifier) Register

The PID register is used in Windows CE mode only. The register is used in conjunction with the fast-context switch hardware support and is only used when the Windows CE mode bit is enabled. More information is available upon request.

2.6.2.6 TI Operations

Register 15 controls specific TI features. Opcode_2 and CRm select the different available registers or operations.

The wait-for-interrupt is write-only. The cache size is hard-wired and read-only. The others are read/write registers.

Table 2–13. TI Operations

Function	Opcode_2	CRm	Rd	Instruction
Set TI925T configuration	0b000	0b0001	Value	MCR p15, 0, Rd, c15, c1, 0
Read TI925T configuration	0b000	0b0001	Value	MRC p15, 0, Rd, c15, c1, 0
Read I_max	0b000	0b0010	Value	MRC p15, 0, Rd, c15, c2, 0
Set I_max	0b000	0b0010	Value	MCR p15, 0, Rd, c15, c2, 0
Read I_min	0b000	0b0011	Value	MRC p15, 0, Rd, c15, c3, 0
Set I_min	0b000	0b0011	Value	MCR p15, 0, Rd, c15, c3, 0
Read thread-ID	0b000	0b0100	Value	MRC p15, 0, Rd, c15, c4, 0
Set thread-ID	0b000	0b0100	Value	MCR p15, 0, Rd, c15, c4, 0
TI925T_status	0b000	0b1000	Value	MRC p15, 0, Rd, c15, c8, 0
Wait-for-interrupt	0b010	0b1000	Ignored	MCR p15, 0, Rd , c15, c8,2

Note: Required for backward code compatibility. Developers must use the wait-for-interrupt described in register 7.

All control bits except L (lock enable) and O (OS type) are set to zero upon reset.

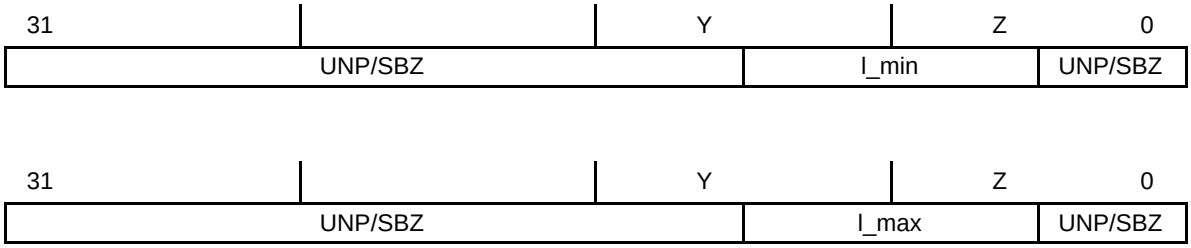
Table 2–14. TI925T Configuration Register

Bit	Name	Value	Function
7	S		Instruction cache streaming disable
		0	I-cache is set in streaming mode. This is the default state after reset.
		1	I-cache is set in nonstreaming mode.

Table 2–14. TI925T Configuration Register (Continued)

Bit	Name	Value	Function
6	R	0	Must be written to as 0.
5	O		OS configuration. This bit takes the value of the OS_TYPE input signal upon reset. It is dependent on the hardware application and may be changed by software. This bit controls the value of the ID register and the enabling of the PID register. The MPU915T_lock input signal forces TI925T into MPU915T mode whatever os_type is.
		0	TI925T (Windows CE mode)
		1	MPU915T (MPU915T mode)
4–3	W		SBZ
2	C		D-cache clean and flush entry mode (See Section 2.6.2.2, *Register 7: Cache Operations*)
		0	The value held in Rd determines the entry of D-cache to clean and the clean operation is done in both sets at a time. This is the default state after reset.
		1	D[31] selects the set targeted by the clean operation.
1	T		Transparent mode
		0	Line loads follow line copy-backs adding some additional latency. This is the default state after reset.
		1	When TI925T is connected to a 16-bit external memory, line loads can hide line copy-backs. There is no extra latency. If the external memory is 32 bits wide, setting this bit to 1 generates an error during copy-back.
0	L		Lock enable
		0	Lock signal stays low during the SWAP instruction (atomic read-write sequence). If marked as buffered, data is sent to the write buffer and reaches the system bus after a slight delay.
		1	The write phase of the SWAP instruction is handled as unbuffered even if it is specified as NCB (non-cacheable and buffered). The S_LOCK signal is active during the SWAP and may be used in a multiprocessor environment. The S_LOCK signal stays low during SWAP instruction accessing regions defined as write-through or copy-back. The L bit is set to one upon reset.

Figure 2–8. Format of the I_min and I_max Registers



I_max indicates the maximum index of the data cache containing a dirty line.

I_min indicates the minimum index of the data cache containing a dirty line.

Upon reset, D-cache flush or end of the full D-cache clean, the value of I_max is cleared and the value of all the I_min bits is set to 1.

The TI debugger uses this register to support multithread debug capability.

Figure 2–9. Format of the Thread-ID Register

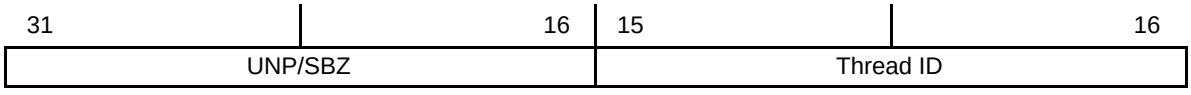


Table 2–15. TI925T_status Register

Bit	Name	Function
31	dcache_dirty	When at 1, indicates the data cache may contain lines marked as dirty.
4	S_abort	When at 1, indicates that external abort occurred. This bit is set to zero upon reset and when read by TI925T.
3	dtlb_mode	When at 1, indicates that DTLB counter is in random mode. Default is set to sequential mode. This bit is set to zero upon reset.
2	Itlb_mode	When at 1, indicates that ITLB counter is in random mode. Default is set to sequential mode. This bit is set to zero upon reset.
1	wb_full	When at 1, indicates that write buffer is full. This bit is set to zero upon reset.
0	buffered_write_aborted	Set to one by the hardware when the system bus controller receives an s_abort following external write from the WB. This is simply an indication for the debug. This bit is set to zero upon reset and when read by TI925T.

2.7 MPU Memory Management Unit

The MPU MMU performs virtual-to-physical address translations and access permission checks for access to the system memory, and it provides the flexibility and security required for the OS to manage physical memory space shared by the DSP subsystem and the MPU subsystem. The MPU MMU provides no protection from DSP shared memory accesses.

The MMU hardware required to perform these functions consists of:

- A 64-entry translation look-aside buffer for instructions (I_TLB)
- A 64-entry translation look-aside buffer for data (D_TLB)
- Access control logic
- Translation table walking logic

The MMU supports memory accesses based on sections or pages:

- Sections represent memory blocks of 1M byte.
- Three different page sizes are supported:
  - Large pages consist of 64K-byte blocks of memory.
  - Small pages consist of 4K-byte blocks of memory.
  - Tiny pages consist of 1K-byte blocks of memory.

Sections and large pages are supported to allow mapping of large regions of memory while using only a single entry in the TLB.

2.7.1 Translation Look-Aside Buffer

The TLB contains entries for virtual-to-physical address translation and access permission checking. If the TLB contains a translated entry for the virtual address, the access control logic determines whether the access is permitted. If access is permitted, the MMU generates the appropriate physical address corresponding to the virtual address. If access is not permitted, the MMU sends an abort signal to TI925T.

Upon a TLB miss (that is, the TLB does not contain an entry corresponding to the virtual address requested), the translation table walking hardware retrieves the translation and access permission information from the translation table in physical memory. Once retrieved, the page or section descriptor is stored into the TLB at a random location.

Note:

Because the load and store multiple instructions can cross a page boundary, the permission access is checked for each sequential address.

Unpredictable behavior occurs if two TLB entries correspond to overlapping areas of memory in the virtual space. This can occur if the TLB is not flushed after the memory is remapped with different-sized pages (leaving an old mapping with different sizes in the TLB and using a new mapping that is loaded into a different TLB location).

2.7.2 Translation Table

The translation table held in main memory has two levels:

- The first-level table can hold both section translation entries and pointers to second-level tables (either fine tables or coarse tables).
- The second-level tables can hold large, small, and tiny page translations entries.

2.7.3 Domains and Access Permissions

The MMU also supports domains. Domains are areas of memory that can be defined to have individual access rights. The CP15 domain access control register can specify access rights for up to 16 separate domains. This register is shared by the instruction access permission logic and data access permission logic.

When the MMU is disabled, there is no address translation and no memory access permission checks are performed.

Small pages are further divided into 1K-byte subpages, and large pages are further divided into 16K-byte subpages with separate access permission rights.

Tiny pages and sections are not divided into subpages (single access permission rights).

2.7.4 MMU Program-Accessible Registers

The system control coprocessor (CP15) registers listed in Table 2–16, in conjunction with the translation tables stored in memory, determine the operation of the MMU or hold the MMU state for access by the processor.

Table 2–16. CP15 Registers or Functions Used by the MMU

Register	Number	Bits
Control register	1	M, A, S, R
Translation table base	2	31..14
Domain access control	3	31..0
Fault status	5 (D)	8..0
Fault address	6 (D)	31..0
TLB operations	8	8 31..0
TLB lock operation	10 (I &D)	31.. 20, 0

All of these registers (except register 8) contain state and can be read from and written to. The MMU also updates registers 5 and 6 upon a data abort to record the cause and address of the abort (see Section 2.6, *Coprocessor 15* for more details on CP15).

2.7.5 Address Translation

Translation information, which consists of both the address translation data and the access permission data, resides in a translation table located in physical memory. The MMU provides the logic needed to traverse this translation table, obtain the translated address, and check the access permission.

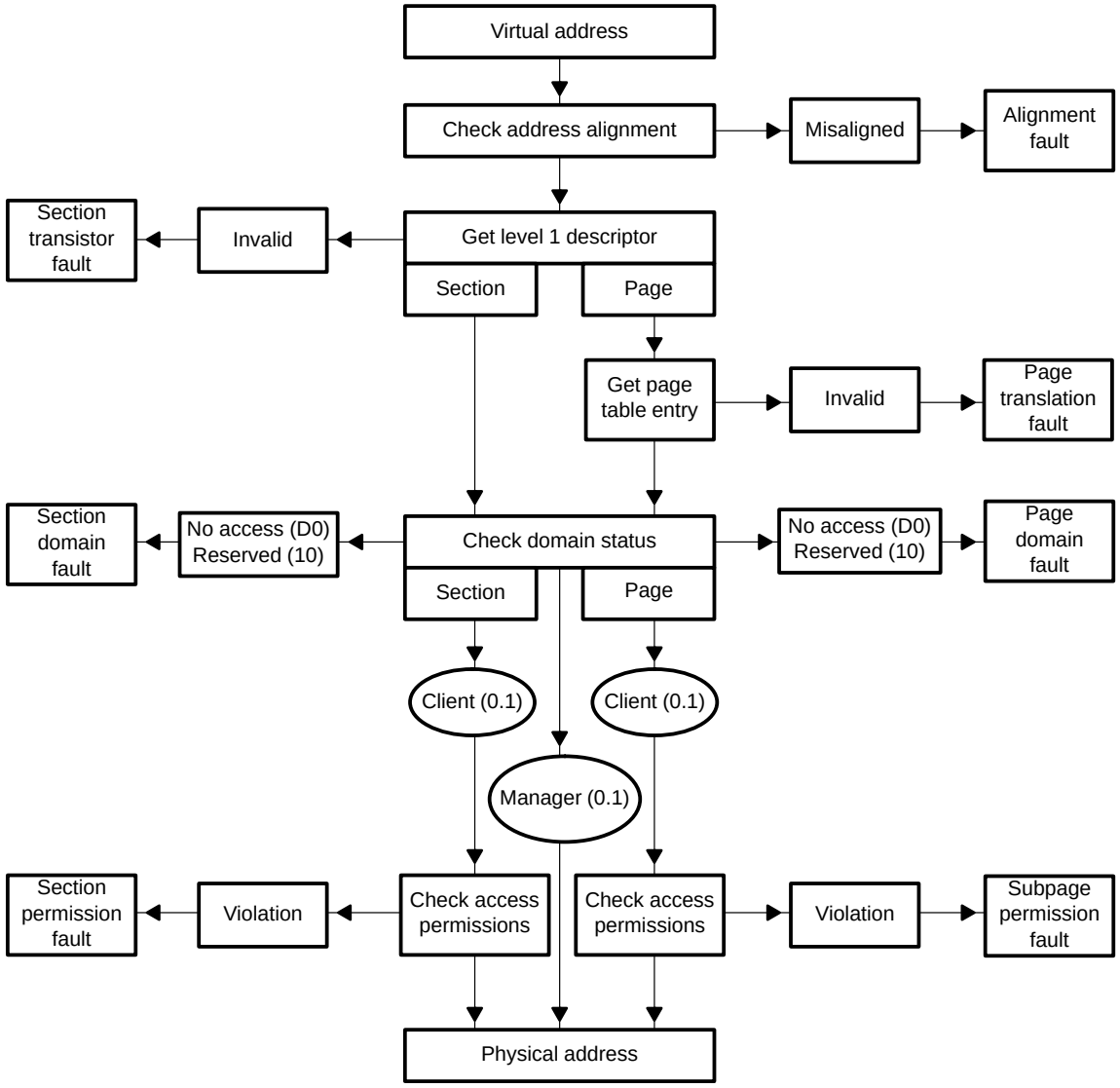
There are four routes by which the address translation (hence access permission) takes place. The route taken depends on whether the address in question has been marked as a section-mapped access or a page-mapped access. There are three sizes of page-mapped access (large, small, and tiny pages). However, the translation process always starts out in the same way, as described below, with a level 1 fetch. A section-mapped access only requires a level 1 fetch, but a page-mapped access also requires a level 2 fetch.

2.7.6 Translation Process

The MMU translates virtual addresses generated by the CPU into physical addresses to access the external memory and checks the access permission using a translation look-aside buffer (TLB) (see Figure 2–10).

The MMU table walking hardware is used to add entries to the TLB.

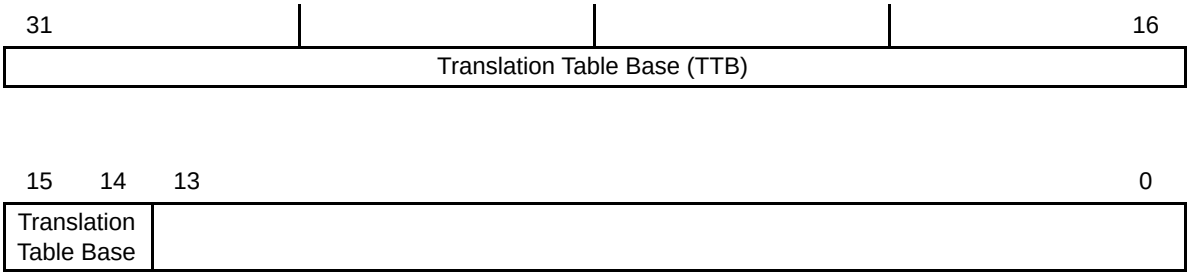
Figure 2–10. Address Translation Process



2.7.6.1 Translation Table Base

The translation process is initiated when the on-chip TLB does not contain an entry corresponding to the requested virtual address (that is, when a TLB-miss occurs). The CP15 translation table base (TTB) register points to the base of a table in physical memory, which contains section and page table descriptors. The 14 LSBs of the TTB register are always set to zero, so the table must start on a 16K-byte boundary.

Figure 2–11. Translation Table Base Register

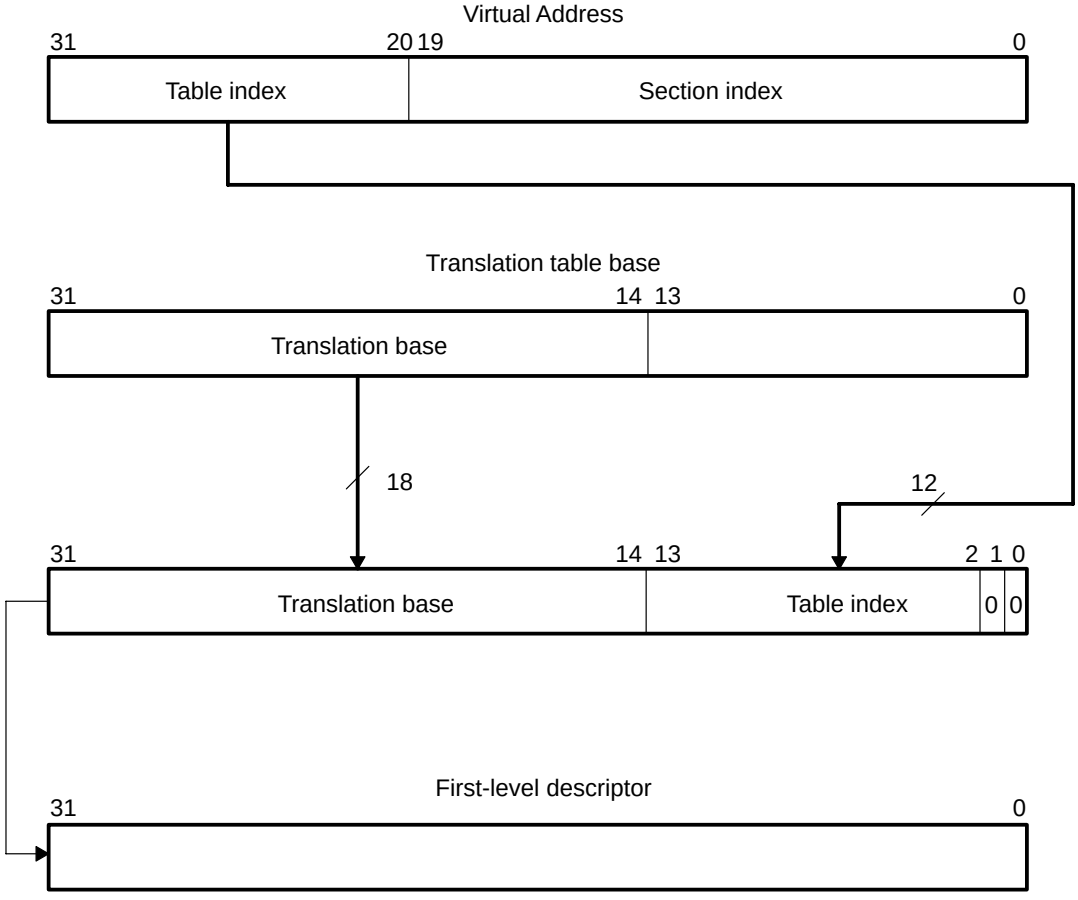


The translation table has up to 4096 32-bit entries, each describing 1M byte of virtual memory. This allows the addressing of up to 4G bytes of virtual memory.

2.7.6.2 Level 1 Fetch

Bits 31–14 of the TTB register are concatenated with bits 31–20 of the virtual address to produce a 30-bit address (see Figure 2–12) by accessing the translation table level 1 descriptors (see Section 2.7.6.3). This address selects a four-byte translation table entry, which is a level 1 descriptor for either a section or a page table.

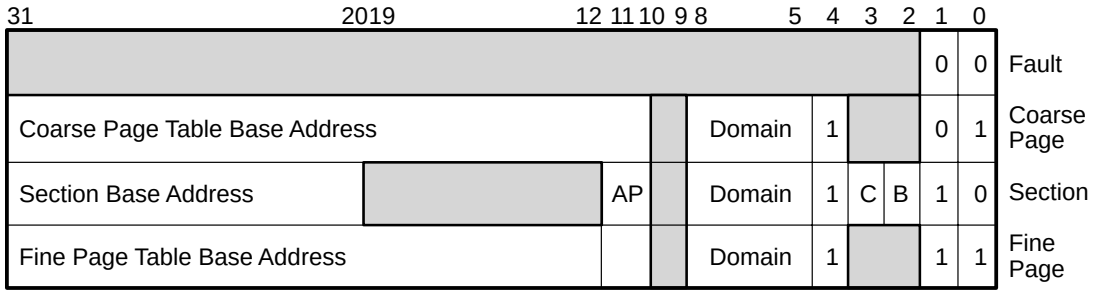
Figure 2–12. Accessing the Translation Table Level 1 Descriptors



2.7.6.3 Level 1 Descriptor

The level 1 descriptor returned is either a coarse or fine page table descriptor or a section descriptor. Its format varies accordingly, as shown in Figure 2–13.

Figure 2–13. Level 1 Descriptors



Note: Bits in gray areas are ignored. They must be written to as 0. The two least significant bits indicate the descriptor type and validity and are interpreted as shown below.

Table 2–17. Level 1 Fine Page Table Descriptor

Bit	Name	Function
31–12	FINE_PG_BASE	Base address used to access the fine page table entry. The fine page table index selecting an entry is derived from the virtual address as illustrated in Figure 2–16, *Tiny Page Translation*.
11–9	RESERVED	Reserved. Must be written as 0.
8–5	DOMAIN	Specify which one of the sixteen domains (held in the domain access control register) contains the primary access controls.
4	RESERVED	Reserved. Must be written to as 1 for backward compatibility.
3–0	RESERVED	Reserved. Must always be written as 0.
1–0	RESERVED	Reserved. Must always be written as 1.

If a page table descriptor is returned from the level 1 fetch (Bit 0 = 1), a level 2 fetch is initiated.

Table 2–18. Interpreting Level 1 Descriptor Bits 1–0

Value	Meaning	Notes
00	Invalid	Generates a section translation fault
01	Coarse	Indicates a coarse page descriptor
10	Section	Indicates a section descriptor
11	Fine	Indicates a fine page descriptor

Table 2–19. Level 1 Coarse Page Table Descriptor

Bit	Name	Function
31–10	COARSE_PG_BASE	Base address used to access the coarse page table entry. The coarse page table index selecting an entry is derived from the virtual address. If a page table descriptor is returned from the level 1 fetch (Bit 0 = 1), a level 2 fetch is initiated.
9	RESERVED	Reserved. Must always be written to as 0.
8–5	DOMAIN	Specify which one of the 16 domains (held in the domain access control register) contains the primary access controls.
4	RESERVED	Reserved. Must be written to as 1 for backward compatibility.
3–2	RESERVED	Reserved. Must be written as 0.
1–0	RESERVED	Reserved. Must be written as 1.

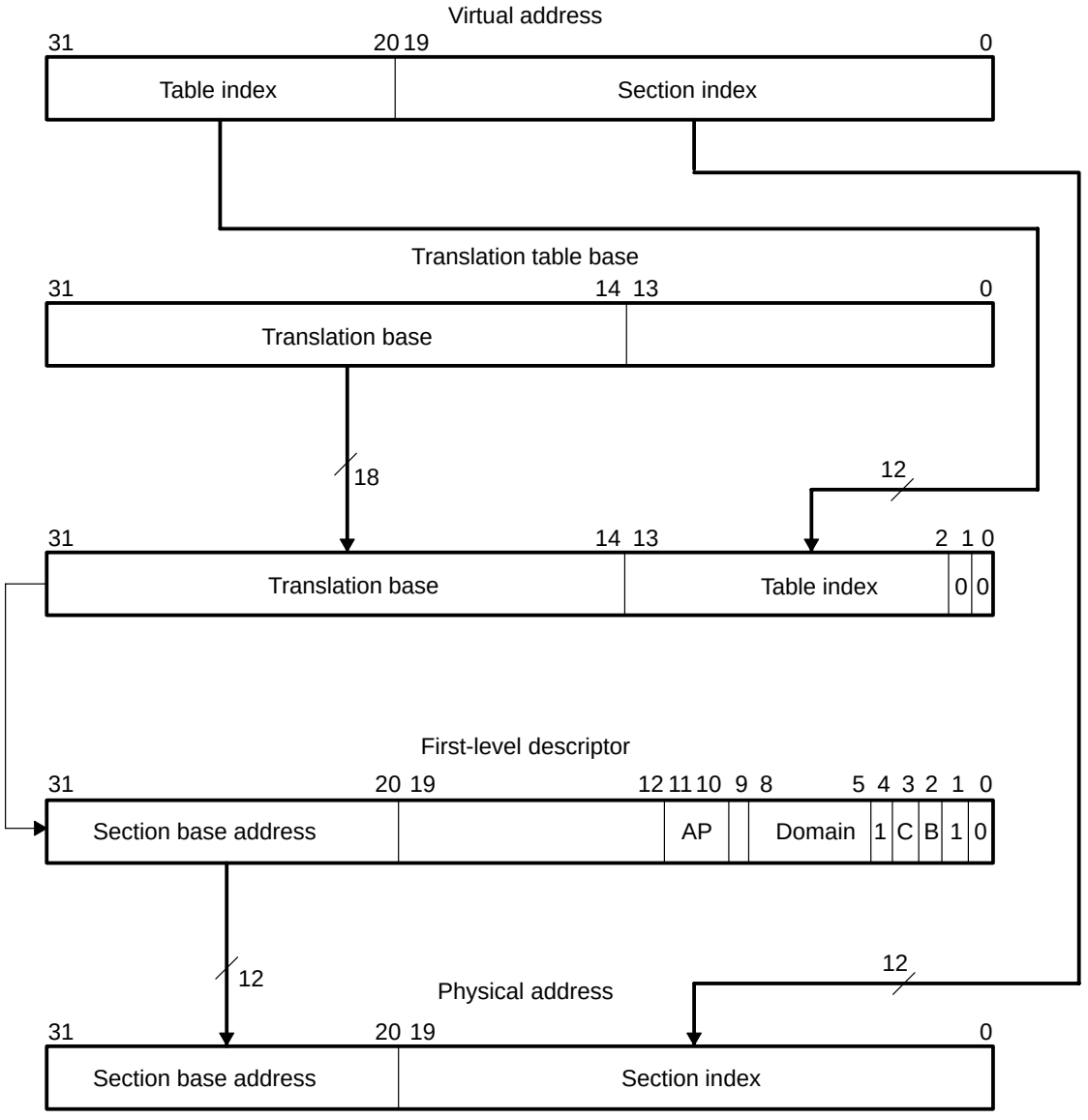
Table 2–20. Level 1 Section Descriptor

Bit	Name	Function
31–20	SECTION_BASE	The 12 MSBs of the address of the section in physical memory (section base address).
19–12	Reserved	Must always be written to as 0.
11–10	AP	Specify the access permissions for this section (see Table 2–24).
9	Reserved	Must always be written to as 0.
8–5	DOMAIN	Specify which one of the 16 domains (held in the domain access control register) contains the primary access controls.
4	Reserved	Must be written to as 1 for backward compatibility.
3	C	Cacheable (C_MMU): indicates that data or instructions at this address are placed in the cache if the cache is enabled.
2	B	Bufferable (B_MMU): indicates that data writes at this address are buffered if the write buffer is enabled.

2.7.6.4 Translating Section References

Figure 2–14 illustrates the complete section translation sequence. The access permissions contained in the level 1 descriptor must be checked before the physical address is put on the address bus.

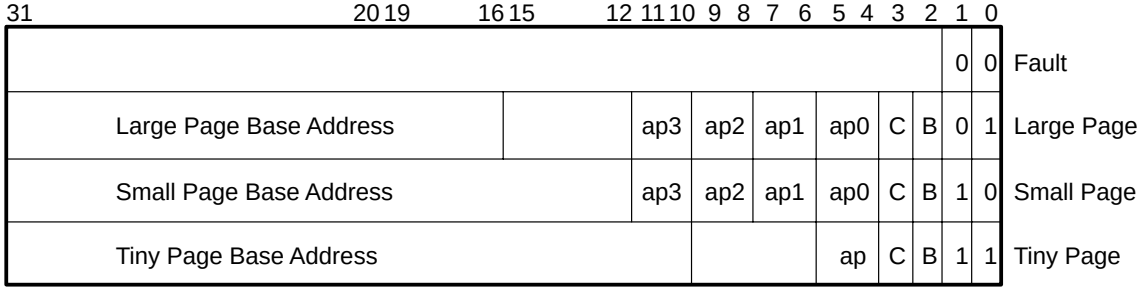
Figure 2–14. Section Translation



2.7.6.5 Level 2 Descriptor

The level 1 fetch, when returning a coarse or fine page table descriptor, provides the base address of the page table to be used. The page table is then accessed, and a level 2 descriptor is returned. This descriptor defines a tiny, small, or large page access. Figure 2–15 shows the format of level 2 descriptors.

Figure 2–15. Page Table Entry (Level 2 Descriptor)



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.

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.

The two least significant bits indicate the page size and validity and are interpreted as follows.

Table 2–21. Level 2 Section Descriptor

Bit	Name	Function
31–10	PG_BASE	Bits 31–10 (tiny pages), bits 31–12 (small pages), or bits 31–16 (large pages) are used to form the corresponding bits of the physical address—the physical page number. The page index is derived from the virtual address.
11–4	AP	Specify the access permissions (ap3-ap0) for the four subpages within large and small pages. Tiny pages do not have subpages and bits 5-4 specify the access permission (see Table 2–24). For large pages, bits 15-12 SBZ.
3	C	Cacheable (C_MMU): indicates that data or instructions at this address are placed in the cache if the cache is enabled.
2	B	Indicates that data writes at this address are buffered if the write buffer is enabled.

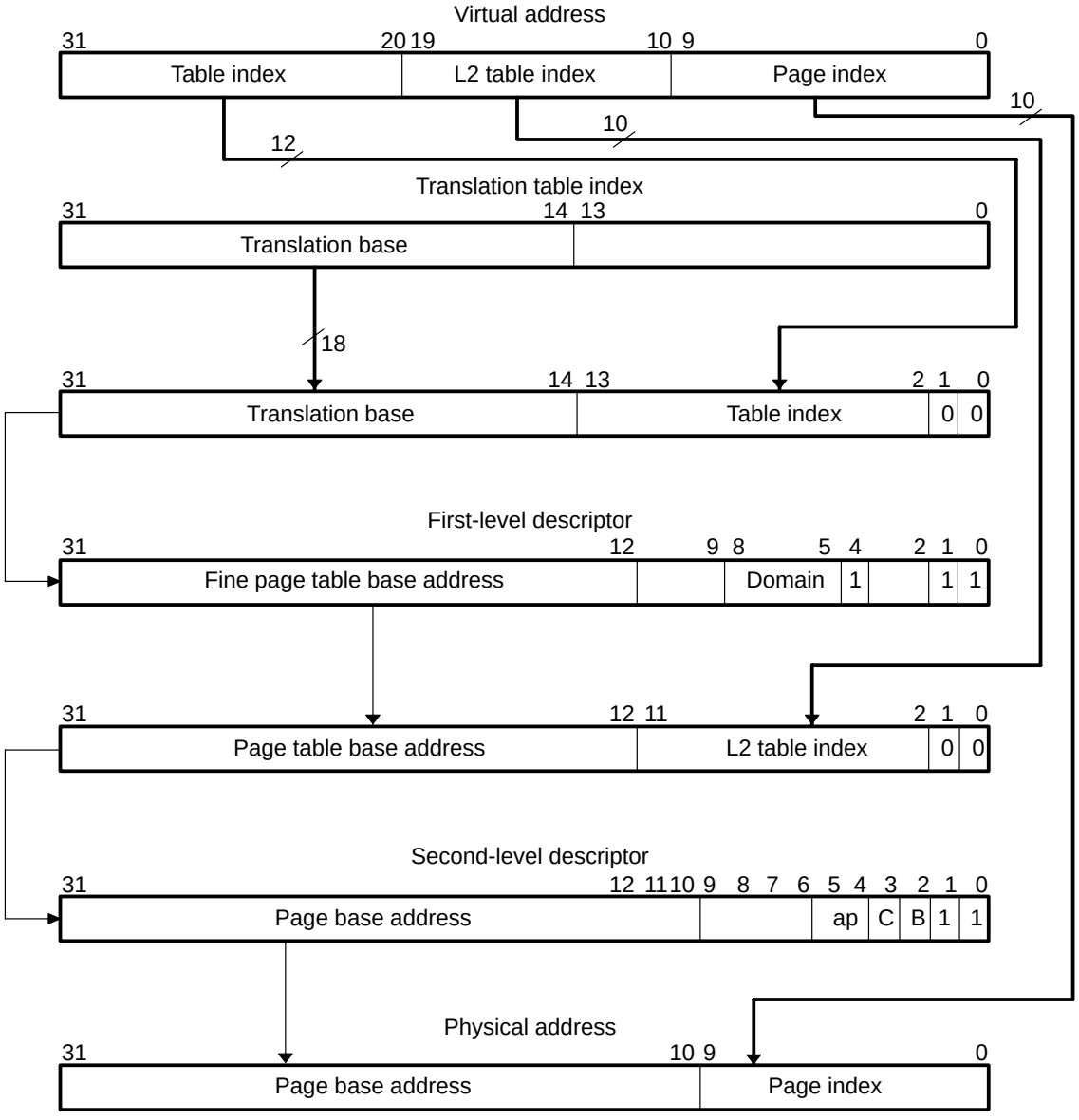
Table 2–22. Interpreting Page Table Entry Bits 1–0

Value	Meaning	Notes
00	Invalid	Generates a page translation fault
01	Large page	Indicates a 64K-byte page
10	Small page	Indicates a 4K-byte page
11	Tiny page	Indicates a 1K-byte page

2.7.6.6 Translating Tiny Pages References

Figure 2–16 illustrates the complete translation sequence for a 1K-byte tiny page. Page translation involves one additional step beyond that of a section translation; the level 1 descriptor is the page table descriptor and is used to point to the level 2 descriptor or page table entry. For pages, the access permissions are contained in the level 2 descriptor and must be checked before the physical address is put on the s_add bus.

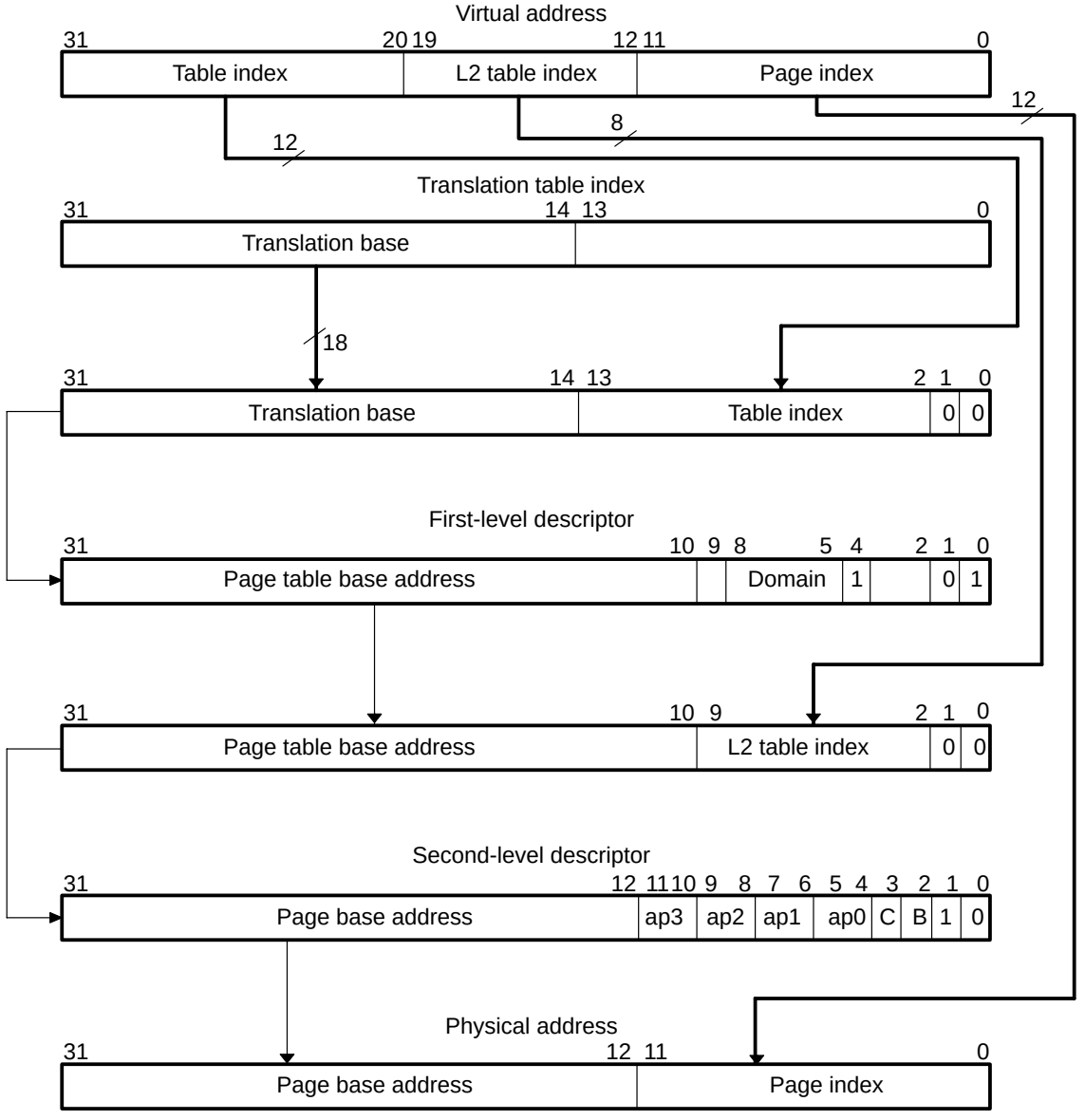
Figure 2–16. Tiny Page Translation



2.7.6.7 Translating Small Page References

Figure 2–17 illustrates the complete translation sequence for a 4K-byte small page. If a small page descriptor is included in a fine page table, the upper two bits of the index of the page overlap the lower two bits of the index of the fine page table. In other words, four consecutive entries must be used for a small page in a fine page table.

Figure 2–17. Small Page Translation



2.7.6.8 Translating Large Page References

Figure 2–18 illustrates the complete translation sequence for a 64K-byte large page. As the upper four bits of the page index and the lower four bits of the coarse page table index overlap, each coarse page table entry for a large page descriptor must be duplicated 16 times (in consecutive memory locations). If the large page is included in a fine page table, the large page descriptor must be duplicated 64 times.

2.7.7 MMU Faults and MPU Aborts

The MMU generates the following types of faults:

- ❑ Alignment fault (on data access only)
- ❑ Translation fault
- ❑ Domain fault
- ❑ Permission fault

In addition, an external abort can be raised on certain types of external data accesses.

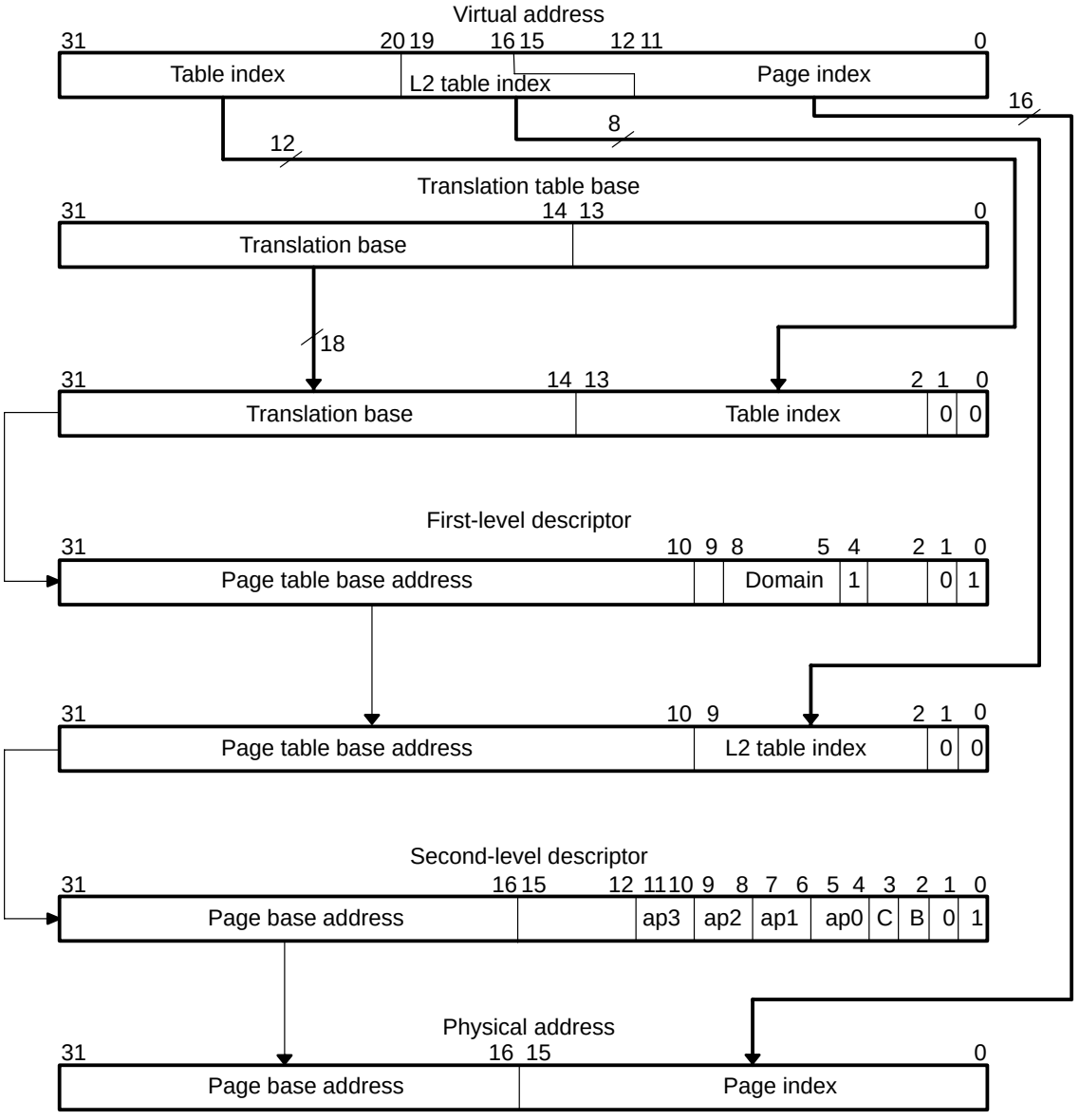
When the MMU is off, the only fault generated is the alignment fault.

The access control mechanism of the MMU detects the conditions that produce these faults. If a fault is detected as the result of a memory access, the MMU aborts the access and signals the fault condition to the MPU. The MMU is also capable of retaining the type and address information of the abort. The MPU recognizes two types of aborts: data and prefetch aborts. The MMU has no FAR or FSR registers.

The MMU detects access violations before starting the external memory access. External aborts do not necessarily inhibit the external access, as described in Section 3.10, *System Operating Details*.

MPU instructions are prefetched, so a prefetch abort simply flags the instruction as it enters the instruction pipeline. An abort does not occur yet, because a previously fetched instruction could render the rest of the pipe moot. For example, if a branch instruction executes first or an interrupt occurs, the instruction that causes the abort never executes. This instruction actually causes the abort to take place only if it is executed. No abort takes place if the instruction is not used (when it is branched around).

Figure 2–18. Large Page Translation



2.7.8 Fault Address and Fault Status Registers (FAR and FSR)

If an illegal data access (data abort) occurs, the MMU places an encoded 4-bit value FS[3–0] and the 4-bit encoded domain number in the fault status register (FSR). In addition, the virtual address associated with the data abort is stored into the fault address register (FAR). If an access violation results from multiple causes, the faults are encoded according to the priorities given in Table 2–23. Faults that occur during an instruction fetch are not stored in FSR and FAR.

The following sections describe the various access permissions and controls supported by the MMU and detail how they are interpreted to generate faults.

Table 2–23. Priority Encoding of the Fault Status Register

Source		Priority	Domain [3-0]	FAR
Highest priority				
Alignment†		0b0001	Invalid‡	VA of access causing abort§
External abort on transaction	First level	0b1100	Invalid	VA of access causing abort
	Second level	0b1110	Valid	
Transaction	Section	0b0101	Invalid	VA of access causing abort
	Page	0b0111	Valid	
Domain	Section	0b1001	Valid	VA of access causing abort
	Page	0b1011	Valid	
Permission	Section	0b1101	Valid	VA of access causing abort
	Page	0b1111	Valid	
External abort on line fetch	Section	0b0100	Valid	VA of start of cache line being loaded
	Page	0b0110	Valid	
External abort on NCNB access	Section	0b1000	Valid	VA of access causing abort
	Page	0b1010	Valid	
Lowest priority				

† Alignment faults write 0b0001 into FS[3-0].
‡ Invalid values in domain[3-0] occur because the fault is raised before a valid domain field has been selected.
§ Fixing the primary abort and restarting the instruction can regenerate any abort masked by the priority encoding.

2.7.9 Domain Access Control

MMU accesses are primarily controlled via domains. There are 16 domains, and each domain has a 2-bit field to define it. Two kinds of users are supported: clients and managers. Clients use a domain; managers control the behavior of the domain. The domains are defined in the domain access control register. The following figure illustrates how the 32 bits of the register are allocated to define the sixteen 2-bit domains.

Figure 2–19. Domain Access Control Register Format

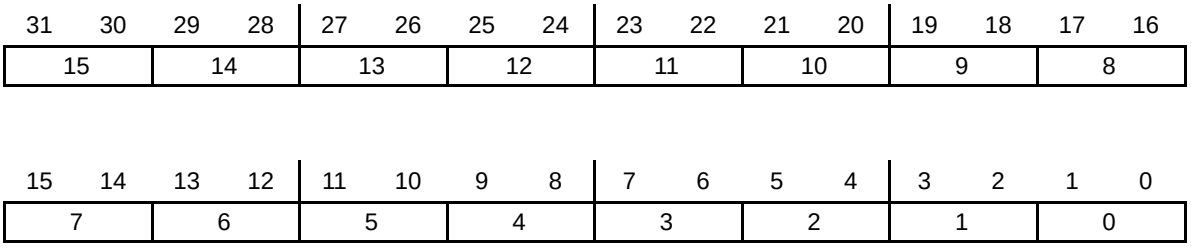


Table 2–24 defines how the bits within each domain are interpreted to specify the access permissions.

Table 2–24. Interpreting Access Bits in Domain Access Control Register

Value	Access Type	Description
0b00	No access	Any access generates a domain fault.
0b01	Client	Access permission is checked against the permission given by the page descriptor.
0b10	Reserved	Behaves like no access
0b11	Manager	The access permission is not checked; permission faults are not generated.

2.7.10 Permission Access

Both instructions and data need access permission checks, but their respective access violations are handled differently. A data access error generates a DABORT and stores the status, domain, and address in FSR and FAR. An instruction fetch generates an IABORT only; it does not update FSR and FAR as it is possible the aborted instruction is not executed (if it is branched around). The IABORT flags the instruction as it enters the TI925T.

When the MMU is turned off, the physical address is output directly and no memory access permission checks are performed.

Table 2–25. Interpreting Access Permission

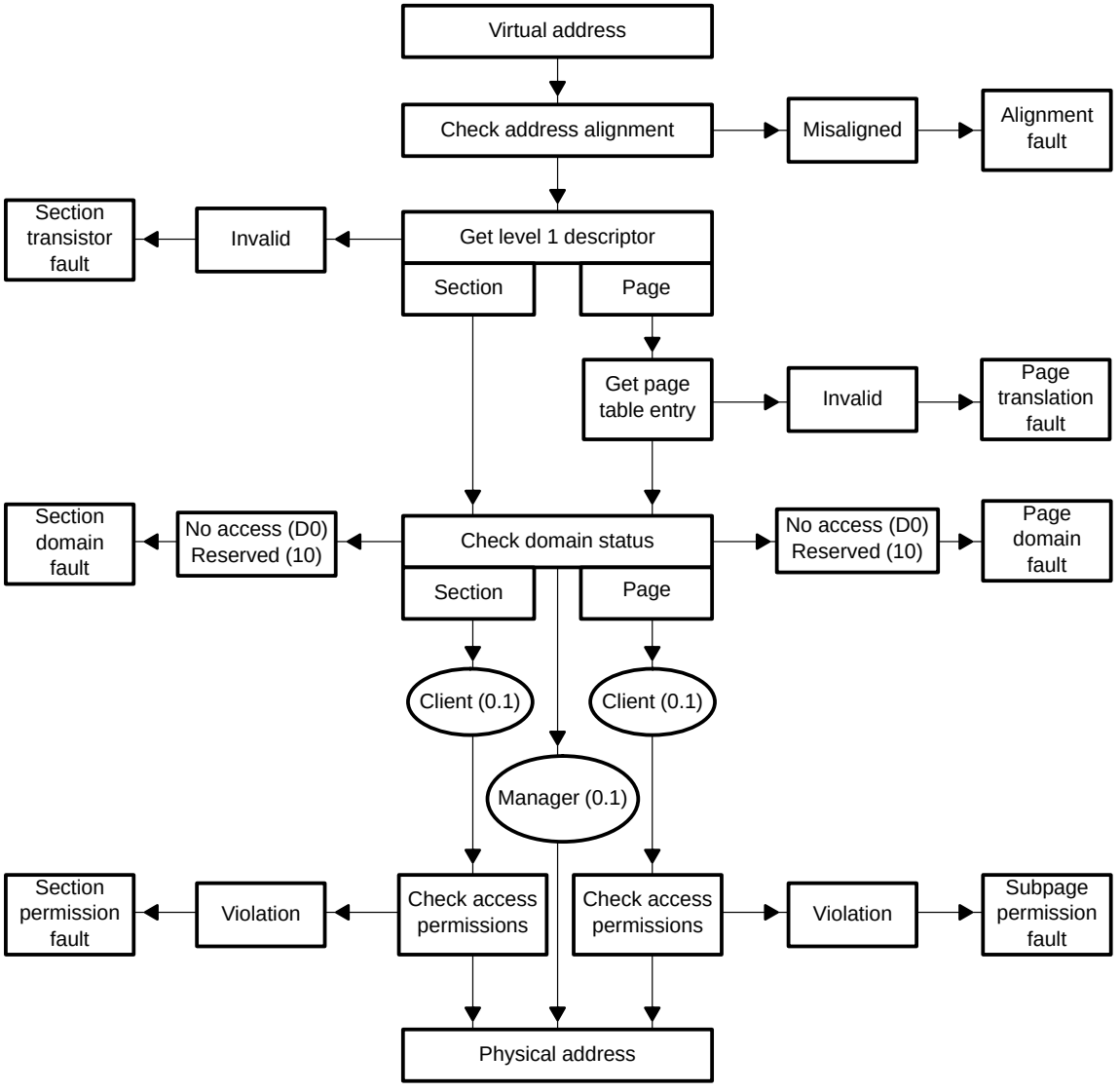
Domain	AP	S	R	Supervisor	User	Description
x0	xx	x	x	No access	No access	Generates a domain fault
01	00	0	0	No access	No access	Generates a permission fault
01	00	1	0	Read only	No access	Supervisor read only permitted
01	00	0	1	Read only	Read only	Any write generates a permission fault.
01	00	1	1	Reserved	Reserved	Generates a permission fault
01	01	x	x	Read/write	No access	Access allowed only in supervisor mode.†
01	10	x	x	Read/write	Read only	User writes cause a permission fault.†
01	11	x	x	Read/write	Read/write	All accesses are allowed.†
01	xx	1	1	Reserved	Reserved	Generates a permission fault
11	xx	x	x	Full access	Full access	No permission fault can be generated.

† In client mode, the combination S/R = 11 is reserved and generates a permission fault. Therefore, on these three lines, S/R can only take the values 00, 01, or 10.

2.7.11 Fault Checking Sequence

The sequence by which the MMU checks for access faults is slightly different for sections and pages. Figure 2–20 illustrates the sequence for both. The following sections describe the conditions that generate each of the faults.

Figure 2–20. Sequence for Checking Faults



2.7.11.1 Alignment Fault

If an alignment fault is enabled (bit 1 in CP15 control register 1), the MMU generates an alignment fault upon 16-bit and 32-bit data accesses that are improperly aligned (not on an address multiple of 2 and 4, respectively). The TI925T checks the alignment even if the MMU is disabled.

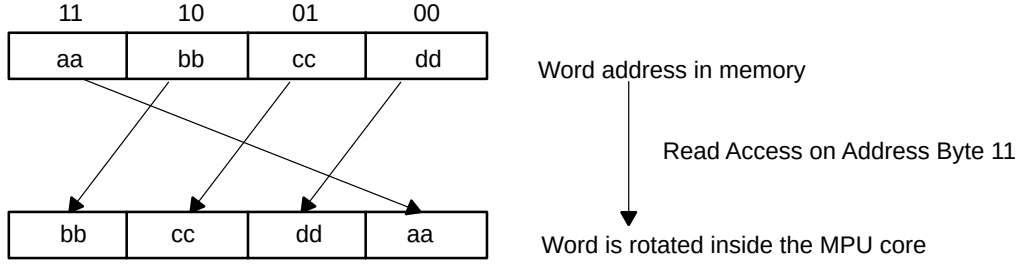
Instruction fetches do not generate alignment faults; they always access memory on 32-bit word boundaries.

If the access generates an alignment fault, the access sequence aborts without checking access rights.

If a nonaligned read access is executed and the alignment fault is disabled, data is accessed at a word address and rotated inside the core as shown below. If a nonaligned half-word or word write is executed while the alignment fault is disabled, the write is done on a half-word or word address boundary.

Figure 2–21 is an example of read word access on byte 11.

Figure 2–21. Nonaligned Read Word Access



2.7.11.2 Translation Fault

There are two types of translation fault: section and page.

- A section translation fault is generated if the level 1 descriptor is marked as invalid. This happens if bits [1–0] of the descriptor are both 0.
- A page translation fault is generated if the page table entry is marked as invalid. This happens if bits [1–0] of the page table entry are both 0.

2.7.11.3 Domain Fault

There are two types of domain faults: section and page. In both cases, the level 1 descriptor holds the 4-bit domain field that selects one of the sixteen 2-bit domains in the domain access control register. The two bits of the specified domain are then checked for access permissions, as detailed in Table 2.15.

- In the case of a section, the domain is checked once the level 1 descriptor is returned.
- In the case of a page, the domain is checked once the page table entry is returned.

A section or page domain fault occurs if the permission access is either no access (00) or reserved (10).

2.7.11.4 Permission Fault

There are two types of permission faults: section and subpage. Permission fault is checked at the same time as the domain fault. If the 2-bit domain field returns client (01), then the permission access check is invoked as follows:

- Section: If the level 1 descriptor defines a section-mapped access, its AP bits define whether or not the the access is allowed (see Table 2–24). Their interpretation is dependent upon the setting of the S bit (CP15 control register bit 8). If the access is not allowed, a section permission fault is generated.
- Subpage: If the level 1 descriptor defines a page-mapped access, the level 2 descriptor specifies four access permission fields (ap3..ap0), each corresponding to one quarter of the page. ap0 corresponds to the subpage located at the lowest addresses. The selected AP bits are then interpreted in the same way as for a section and may generate a subpage permission fault.

2.7.12 External Aborts

In addition to the MMU-generated aborts, the TI925T has an external s_abort port, which can be used to flag errors on external memory accesses. However, not all accesses can be aborted this way, so this signal must be used with great care. This section describes the restrictions.

The accesses listed below can be aborted and restarted safely. In the case of an interlocked read-write (SWAP instruction) in which the read aborts, the write does not happen.

- Reads
- Unbuffered writes
- Level-1 descriptor fetch
- Level-2 descriptor fetch
- Interlocked read-write (SWAP)
- Cacheable reads (line fetches)

A cache line fetch can be safely aborted on any word in the transfer. If an abort occurs during the line fetch, the cache line is invalidated. In addition, if the abort happens upon or before the instruction the TI925T requested, the instruction is aborted. If the abort happens after, the cache line is simply marked as invalid.

2.7.13 Buffered Writes

Buffered writes cannot be aborted externally. Therefore, the system must be configured in such a way that it does not perform buffered writes to areas of memory that can generate an external abort.

There are three instances of MMU: the DSP MMU, the MPU instruction cache MMU, and the MPU data cache MMU. The MPU MMU is that of the TI925T. Because there are multiple MMUs, it is the responsibility of the OS (system software) to ensure data coherence.

2.8 DSP Memory Management Unit

The DSP MMU handles the external memory space mapping of the DSP in the entire shared memory space of the OMAP5910 device. The DSP MMU translates addresses coming from the DSP (virtual address) to addresses mapped by the traffic controller. The MMU is used when the DSP software accesses external memory. This memory can be any mapped on the OMAP5910 address space, on the internal SRAM, or on an external SDRAM.

The DSP MMU sees 16M bytes of virtual program and 16M bytes of virtual data spaces. The 16M bytes of DSP external addresses can be mapped to any of the 4G bytes of addresses on the OMAP5910 device. If a memory protection or memory access violation occurs, the DSP MMU sends an interrupt to the MPU via the second-level interrupt handler on IRQ_28. Information about the violation can be found in the MMU fault address and fault status registers.

The DSP MMU is programmed by the TI925T processor. In general, the MMU is initialized at boot time, but it also can be reprogrammed dynamically. The MMU is programmed through the TI peripheral bus registers. The DSP MMU registers are listed in Table 2–26 and detailed in this section.

Table 2–26. DSP Memory Management Unit Registers

Name	Description	R/W	Size	Address	Reset Value
PREFETCH_REG	Prefetch register	R/W	16 bits	FFFE:D200	0x0000
WALKING_ST_REG	Prefetch status register	R	16 bits	FFFE:D204	0x0000
CNTL_REG	Control register	R/W	16 bits	FFFE:D208	0x0000
FAULT_AD_H_REG	Fault address register MSB	R	16 bits	FFFE:D20C	0x0000
FAULT_AD_L_REG	Fault address register LSB	R	16 bits	FFFE:D210	0x0000
F_ST_REG	Fault status register	R	16 bits	FFFE:D214	0x0000
IT_ACK_REG	Interrupt acknowledge register	W	16 bits	FFFE:D218	0x0000
TTB_H_REG	TTB register MSB	R/W	16 bits	FFFE:D21C	0x0000
TTB_L_REG	TTB register LSB	R/W	16 bits	FFFE:D220	0x0000
LOCK_REG	Lock counter	R/W	16 bits	FFFE:D224	0x0000
LD_TLB_REG	Load entry in TLB	R/W	16 bits	FFFE:D228	0x0000
CAM_H_REG	CAM entry register MSB	R/W	16 bits	FFFE:D22C	0x0000
CAM_L_REG	CAM entry register LSB	R/W	16 bits	FFFE:D230	0x0000
RAM_H_REG	RAM entry register MSB	R/W	16 bits	FFFE:D234	0x0000

Table 2–26. DSP Memory Management Unit Registers (Continued)

Name	Description	R/W	Size	Address	Reset Value
RAM_L_REG	RAM entry register LSB	R/W	16 bits	FFFE:D238	0x0000
GFLUSH_REG	Global flush register	R/W	16 bits	FFFE:D23C	0x0000
FLUSH_ENTRY_REG	Individual flush register	R/W	16 bits	FFFE:D240	0x0000
READ_CAM_H_REG	Read CAM MSB	R/W	16 bits	FFFE:D244	0x0000
READ_CAM_L_REG	Read CAM LSB	R/W	16 bits	FFFE:D248	0x0000
READ_RAM_H_REG	Read RAM MSB	R/W	16 bits	FFFE:D24C	0x0000
READ_RAM_L_REG	Read RAM LSB	R/W	16 bits	FFFE:D250	0x0000

Table 2–27. Prefetch Register (PREFETCH_REG)) – Offset Address (hex): 00

Bit	Function	Size	Access	Value at Hardware Reset
15	Reserved	1		
14	The data to prefetch is data when 1, program when 0.	1	R/W	0
13–0	MSB of virtual address tag of the TLB entry to be prefetched	14	R/W	0

Table 2–28. Prefetch Status Register (WALKING_ST_REG) – Offset Address (hex): 04

Bit	Function	Size	Access	Value at Hardware Reset
15–2	Reserved	14		
1	When 1, table walking is running.	1	R	0
0	Writing in the prefetch data register sets this bit; the acknowledge of the prefetch resets the bit.	1	R	0

Table 2–29. Control Register (CNTL_REG) – Offset Address (hex): 08

Bit	Function	Size	Access	Value at Hardware Reset
15–6	Reserved	10		
5	Enables the 16-bit burst management. Active high.	1	R	0
4	Reserved	1		
3	Reserved	1		
2	When 1, enables the walking table logic. When 0, the walking table is disabled and access to the TLB and lock counter are disabled.	1	R	0
1	Enables MMU. Active high.	1	R	0
0	Resets module. Active low.	1	R	0

Table 2–30. Fault Address Register MSB (FAULT_AD_H_REG) – Offset Address (hex): 0C

Bit	Function	Size	Access	Value at Hardware Reset
15–9	Reserved	7		
8	The access that generated a permission fault is data when 1 or program when 0.	1	R	0
7–0	MSB of virtual address of the access that generated a permission fault	8	R	0

Table 2–31. Fault Address Register LSB (FAULT_AD_L_REG) – Offset Address (hex): 10

Bit	Function	Size	Access	Value at Hardware Reset
15–7	LSB of virtual address of the access that generated a permission fault	9	R	0
6–0	Reserved	7		

Table 2–32. Fault Status Register (F_ST_REG)) – Offset Address (hex): 14

Bit	Function	Size	Access	Value at Hardware Reset
15–4	Reserved	12		
3	Error occurred during a prefetch. Active high.	1	R	0
2	Permission fault. Active high.	1	R	0
1	TLB miss. Active high.	1	R	0
0	Translation fault. Active high.	1	R	0

Table 2–33. IT Acknowledge Register (IT_ACK_REG) – Offset Address (hex): 18

Bit	Function	Size	Access	Value at Hardware Reset
15–1	Reserved	15		
0	Write a 1 to this bit to acknowledge the interrupt. A write of 0 has no effect; a write of 1 clears the bit automatically.	1	W	0

Table 2–34. TTB Register MSB (TTB_H_REG) – Offset Address (hex): 1C

Bit	Function	Size	Access	Value at Hardware Reset
15–0	MSB of TTB	16	R	0

Table 2–35. TTB Register LSB (TTB_L_REG) – Offset Address (hex): 20

Bit	Function	Size	Access	Value at Hardware Reset
15–7	LSB of TTB	9	R	0
6–0	Reserved	7		

Table 2–36. Lock Counter Register (LOCK_REG) – Offset Address (hex): 24

Bit	Function	Size	Access	Value at Hardware Reset
15–10	Locked entries base value	6	R/W	0
9–4	Current entry pointed by the WTL	6	R/W	0
3–0	Reserved	4		

Table 2–37. Load Entry in TLB Register (LD_TLB_REG) – Offset Address (hex): 28

Bit	Function	Size	Access	Value at Hardware Reset
15–2	Reserved	14		
1	Read data in TLB when 1.	1	R/W	0
0	Load data in TLB when 1.	1	R/W	0

Table 2–38. CAM Entry Register MSB (CAM_H_REG) – Offset Address (hex): 2C

Bit	Function	Size	Access	Value at Hardware Reset
15–6	Reserved	10		
5–0	Table index level 1 MSB	6	R/W	0

Table 2–39. CAM Entry Register LSB (CAM_L_REG) – Offset Address (hex): 30

Bit	Value	Function	Size	Access	Value at Hardware Reset
15–10		Table index level 1 LSB	6	R/W	0
9–4		Tiny page bits 9–0 (10 bits long) Small page bits 9–2 (8 bits long) Large page bits 9–6 (4 bits long)	6	R/W	0

Table 2–39. CAM Entry Register LSB (CAM_L_REG) – Offset Address (hex): 30(Continued)

Bit	Value	Function	Size	Access	Value at Hardware Reset
3		Preserved bit	1	R/W	0
	0	CAM entry not preserved			
	1	CAM entry preserved			
2		Valid bit:	1	R	0
	0	CAM entry not valid			
	1	CAM entry valid			
1–0	00	Section (1 MB)	2	R/W	0
	01	Large pages (64 KB)			
	10	Small pages (4 KB)			
	11	Tiny page (1 KB)			

Table 2–40. RAM Entry Register MSB (RAM_H_REG) – Offset Address (hex): 34

Bit	Function	Size	Access	Value at Hardware Reset
15–0	MSB physical address	16	R/W	0

Table 2–41. RAM Entry Register LSB (RAM_L_REG) – Offset Address (hex): 38

Bit	Function	Size	Access	Value at Hardware Reset
15–10	LSB physical address	6	R/W	0
9–8	Access permission bits	2	R/W	0
7–0	Reserved	8		

Table 2–42. Global Flush Register (GFLUSH_REG) – Offset Address (hex): 3C

Bit	Function	Size	Access	Value at Hardware Reset
15–1	Reserved	15		
0	Toggle bit. Flush all nonprotected TLB entries when 1 is written. Always 0 when read. Automatically reset.	1	R/W	0

Table 2–43. Individual Flush Register (FLUSH_ENTRY_REG) – Offset Address (hex):40

Bit	Function	Size	Access	Value at Hardware Reset
15–1	Reserved	15		
0	Toggle bit. Active high. Always 0 when read.	1	R/W	0

Table 2–44. CAM Entry Register MSB (READ_CAM_H_REG) – Offset Address (hex): 44

Bit	Function	Size	Access	Value at Hardware Reset
15–10	Reserved	6		
9–0	Table index level 1 MSB	10	R/W	0

Table 2–45. CAM Entry Register LSB (CAM_CAM_L_REG) – Offset Address (hex): 48

Bit	Value	Function	Size	Access	Value at Hardware Reset
15–10		Table index level 1 LSB	6	R/W	0
9–4		Tiny page bits 9–0 (10 bits long) Small page bits 9–2 (8 bits long) Large page bits 9–6 (4 bits long)	6	R/W	0

Table 2–45. CAM Entry Register LSB (CAM_CAM_L_REG) – Offset Address (hex): 48 (Continued)

Bit	Value	Function	Size	Access	Value at Hardware Reset
3		Preserved bit	1	R/W	0
	0	CAM entry not preserved			
	1	CAM entry preserved			
2		Valid bit:	1	R	0
	0	CAM entry not valid			
	1	CAM entry valid			
1–0	00	Section (1 MB)	2	R/W	0
	01	Large pages (64 KB)			
	10	Small pages (4 KB)			
	11	Tiny page (1 KB)			

Table 2–46. RAM Entry Register MSB (READ_RAM_H_REG) – Offset Address (hex): 4C

Bit	Function	Size	Access	Value at Hardware Reset
15–0	MSB physical address	16	R/W	0

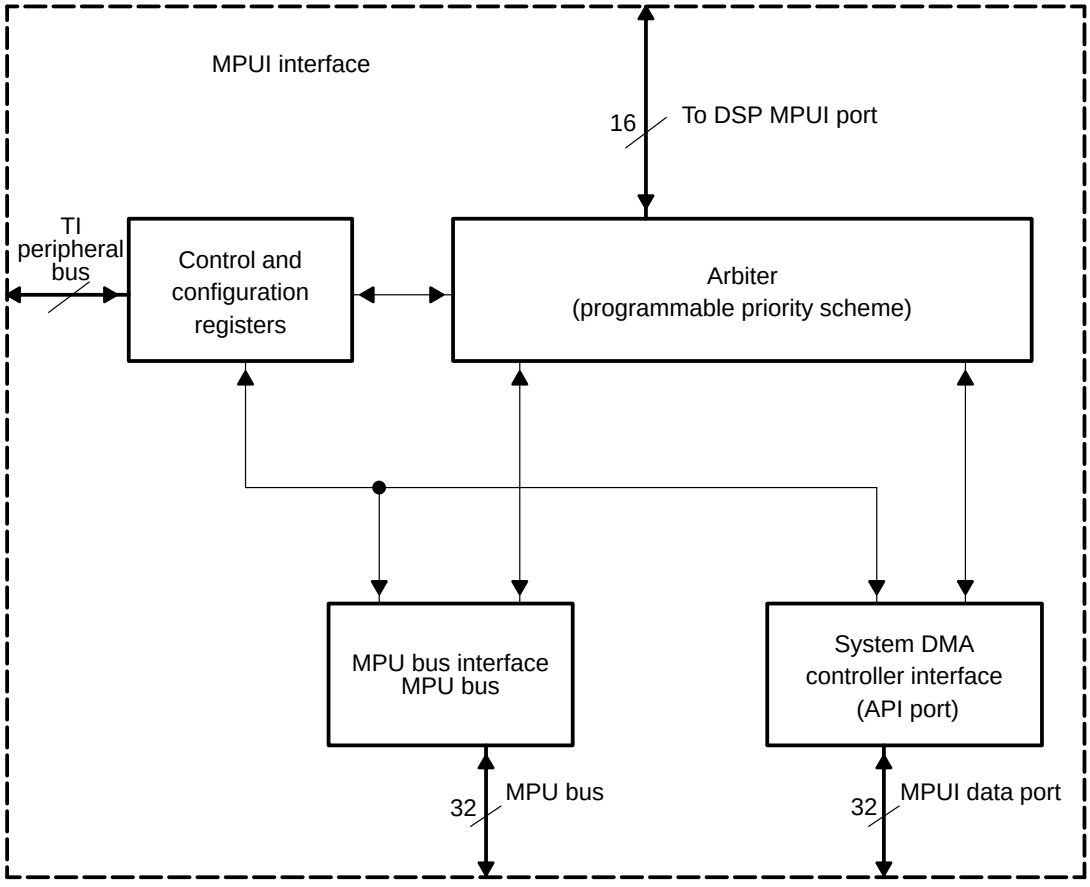
Table 2–47. RAM Entry Register LSB (READ_RAM_L_REG) – Offset Address (hex): 50

Bit	Function	Size	Access	Value at Hardware Reset
15–10	LSB physical address	6	R/W	0
9–8	Access permission bits	2	R/W	0
7–0	Reserved	8		

2.9 MPU Interface

The MPU interface (MPUI) allows the TI925T and the system DMA controller to communicate with the DSP and its peripherals via the DSP MPUI port (part of the DSP); see Figure 2–22. The MPUI provides the capability for the TI925T and the system DMA controller to access the full memory space (16M bytes) of the DSP and the DSP peripheral buses, except the private peripherals. Thus, the TI925T and the system DMA controller have both read and write access to the complete DSP I/O space (128K bytes), including the control registers of the internal DSP peripherals such as the DSP TIPB bridge itself.

Figure 2–22. MPUI Simplified Block Diagram



2.9.1 Functional Features

The MPUI supports the following features:

- Four access modes:
  - Single-access mode (SAM) for SARAM, DARAM, memory interface access
  - Single-access mode (SAM) for peripheral bus access
  - Host-only mode (HOM) for SARAM access
  - Host-only mode (HOM) for peripheral bus access
- An interrupt sent to the TI925T if a time-out occurs
- Programmable priority scheme (TI925T, DMA, etc.) that must be configured during the system boot process
- Packing and unpacking (16-bits to 32-bits, and vice versa)
- 32-bit single access support
- Software control endianism conversion (default is word swap for all access, byte swap for memory access only)
- DMA access to full memory space (16M bytes)
- DMA access to the DSP peripheral bus shared peripherals (up to 128K bytes)

In host-only mode (HOM), the MPUI interface does not have access to the DARAM (0x00 0000 to 0x00 FFFF). All SARAM (0x01 0000 to 0x04 FFFF) is accessible by the MPUI, but the type of access depends on the DSP status (HOM or single-access mode (SAM)) and on the MPUI size register (DSP_API_CONFIG). The following rules apply:

- Before the MPU reset (resetting the DSP MPUI logic) is released, the MPUI cannot access any SARAM.
- After the MPU reset is released and before the DSP reset is released, the DSP is in HOM. The default MPUI size register value (after the MPU_nRESET is released) is 0xFFFF, and the MPUI has exclusive access to all SARAM.
- Shared access: Both the DSP and MPU can access MPUI memory, but the memory space they access is mutually exclusive. The MPUI register controls which memory space (SARAM0, 1, 2...) is accessed by the DSP or the MPU.
- After the DSP reset is released, the DSP goes automatically into SAM; consequently, whatever the value of the MPUI size register, all SARAM is shared between the DSP and the MPUI.

In SAM, all the DSP internal memory is accessible by the MPUI interface. If both the DSP and the MPU controllers (TI925T and system DMA) access the same memory at the same time, priority is given to the DSP controllers. The access is synchronized to the internal DSP CPU clock.

HOM is more efficient than SAM, because there is no synchronization involved. However, HOM depends on the host operating frequency, which is normally slower than the internal DSP CPU 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.

Note: MPUI Port Accesses

MPUI port accesses to the DSP subsystem external address space via the DSP MMU are not supported. MPU and system DMA should access all traffic controller resources (EMIFS, EMIFF, and IMIF) directly through the traffic controller and not via the MPUI port and DSP MMU.

2.9.2 MPUI Registers

Table 2–48 lists the MPUI registers. Table 2–49 through Table 2–56 describe the register bits.

Table 2–48. MPUI Registers

Register Name	Description	R/W	Size	Address (FFFE:x)	Reset Value
CTRL_REG	Control	R/W	32 bits	C900	0x0003 FF1F
DEBUG_ADDR	Debug address—has the address from last operation in case an abort occurs.	R	32 bits	C904	0x00FF FFFF
DEBUG_DATA	Debug data —has the data from last operation in case an abort occurs.	R	32 bits	C908	0xFFFF FFFF
DEBUG_FLAG	Debug flag	R	32 bits	C90C	0x0000 0000
STATUS_REG	MPUIF status	R	32 bits	C910	0x0000 1FFF
DSP_STATUS_REG	Current DSP status	R	32 bits	C914	U
DSP_BOOT_CONFIG	Boot DSP configuration	R/W	32 bits	C918	0x0000 0000
DSP_API_CONFIG	MPUI size information	R/W	32 bits	C91C	0x0000 FFFF

Table 2–49. Control Register (CTRL_REG) – Offset: x00

Bit	Value	Function	Size	Access	Value at Hardware Reset
22–21		Control word swap on the MPUI/DSP interface for a 32-bit access	2	R/W	00
	00	Word swap for all the accesses (OMAP1509 behavior)			
	01	Word swap only for non-APIMEM accesses			
	10	Word swap only for APIMEM accesses			
	11	Turn off word swap for all accesses			
20–18		MPUIF access priority between MPU, reserved port, and DMA requests. The reserved port is not used in the OMAP5910 device and can be disregarde.	3	R/W	000
		Note: the lower the number, the higher the priority.			
	000	MPU-1, DMA-2, reserved port–3			
	001	MPU-1, DMA-3, reserved port–2			
	010	MPU-2, DMA-1, reserved port–3			
	011	MPU-2, DMA-3, reserved port–1			
	1X0	MPU-3, DMA-1, reserved port–2			
	1X1	MPU-3, DMA-2, reserved port–1			
17–16		Control byte swap on the MPUI/DSP interface	2	R/W	11
	00	Turn off byte swap for all accesses			
	01	Byte swap only for non-APIMEM accesses			
	10	Byte swap for all accesses			
	11	Byte swap only for APIMEM accesses			
15–8		MPUI bus access time out	8	R/W	0xFF
7–4		Division factor of APIF_HNSTROBE. For the OMAP5910 device, this field must be set to 2 (10b) or greater. Settings of 00b or 01b should not be used.	4	R/W	0x1

Table 2–49. Control Register (CTRL_REG) – Offset: x00 (Continued)

Bit	Value	Function	Size	Access	Value at Hardware Reset
3	1	Enables sending IRQ_ABORT interrupt to the MPU when an abort condition is indicated by the MPU port from the DSP system.	1	R/W	1
	0	Disables this interrupt source	1	R/W	1
2		Reserved	1	R/W	1
1	1	Enables the time-out feature. An IRQ_ABORT interrupt is sent to the MPU if a time-out occurs.	1	R/W	1
	0	Disables this interrupt source	1	R/W	1
0		Frequency mode	1	R/W	1
	0	Low-frequency MPU clock			
	1	High-frequency MPU clock			

Note: In the MPUI, there are three sources which can generate an IRQ_ABORT:

1) Abort from the DSP: This can be masked by setting CTRL_REG[3] to 0.
2) Time-out event occurred: This can be masked by setting CTRL_REG[1] to 0. But masking the time-out interrupt can cause system to wait forever, if DSP never responds to the MPU request.
3) Burst access detected: This cannot be masked.

These interrupt sources are assigned to the IRQ_ABORT line of the level 1 MPU interrupt handler. The DEBUG_FLAG register contains the information related to which one of these three sources caused the interrupt.

Apart from the MPUI, there are other modules such as the TIPB Bridge which can also generate the IRQ_ABORT interrupt.

Table 2–50. Debug Address Register (DEBUG_ADDR) – Offset: x04

Bit	Function	Size	Access	Value at Hardware Reset
31–24	Reserved	8	R	0x00
23–0	Bits of address bus from MPU/DMA interface. Saved on abort or access mismatch.	24	R	0xFF FFFF

Table 2–51. Debug Data Register (DEBUG_DATA) – Offset: x08

Bit	Function	Size	Access	Value at Hardware Reset
31–0	Value of S_DATA_R is saved when a read access has a size mismatch, and S_DATA_W is saved when a write access is aborted or has a size mismatch.	32	R	0xFFFFFFFF

Table 2–52. Debug Flag Register (DEBUG_FLAG) – Offset: x0C

Bit	Value	Function	Size	Access	Value at Hardware Reset
31–16		Reserved	16	R	0x0000
15–13		Reserved	3		
12–11		Encoded access mode for MPUI	2	R	00
	00	SAM_M and SAM_R			
	01	SAM_M and HOM_R			
	10	HOM_M and SAM_R			
	11	HOM_M and HOM_R			
10–9		Chip-select. Saved on abort. These bits indicate whether memory space or TIPB space was accessed just before the abort was generated.	2	R	00
	01	Memory access			
	10	Peripheral bus or MPUI control register access			
8–7		Burst size saved on abort	3	R	000
6		Read not write on MPUI bus	1	R	0
5		Read not write on MPUI bus. This bit indicates whether a read or write access was active just before the abort was generated.	1	R	0
	1	Read access			
	0	Write access			
4		Flag set to 1 when access size saved on abort is 32 bits	1	R	0

Table 2–52. Debug Flag Register (DEBUG_FLAG) – Offset: x0C (Continued)

Bit	Value	Function	Size	Access	Value at Hardware Reset
3		Flag set to 1 when burst size saved on abort is not equal to 000	1	R	0
2		Flag set to 1 when MPUIF access is aborted by internal time out	1	R	0
1		Flag set to 1 when MPUI aborts access	1	R	0
0		Flag set to 1 when MPUI port on DSP subsystem aborts the access	1	R	0

The STATUS_REG checks the status of the MPU interface during suspend mode (for example, after hitting an emulator breakpoint). The register is for OMAP5910 device chip designers to use for debugging.

Table 2–53. Status Register (STATUS_REG) – Offset: x10

Bit	Value	Function	Size	Access	Value at Hardware Reset
12–11		Current access in progress is:	2	R	11
	00	MPU access			
	01	DMA access			
	10	Reserved port access, should not occur			
	11	No access			
10–3		Current value of time-out counter	8	R	0xFF
2		Enable chip-select bit indicates when MPU wait states are being inserted, which forces chip-selects to inactive:	1	R	1
	0	CSs are forced to inactive state (high).			
	1	CSs are enabled and can be asserted.			
1	0	MPUIF is accessing MPUI.	1	R	1
	1	No access in progress			

Table 2–53. Status Register (STATUS_REG) – Offset: x10 (Continued)

Bit	Value	Function	Size	Access	Value at Hardware Reset
0		Current access mode when ACCESS_DONE = 0 or last access mode when ACCESS_DONE = 1	1	R	1
	0	SAM			
	1	HOM			

Table 2–54. DSP Status Register (DSP_STATUS_REG) – Offset: x14

Bit	Value	Function	Size	Access	Value at Hardware Reset
11		HOM or SAM for accessing DSP peripherals (from DSP)	1	R	1
	0	SAM			
	1	HOM			
10		HOM or SAM for accessing MPUI peripherals (from DSP)	1	R	1
	0	SAM			
	1	HOM			
9		Asynchronous reset controlled by emulation	1	R	1
8		Idle peripherals	1	R	1
	0	Functional mode			
	1	Idle			
		Linked to bit 7 of the ISTR register (from DSP)			
7		Idle peripherals	1	R	1
	0	Functional mode			
	1	Idle			
		Linked to bit 6 of the ISTR register (from DSP)			

Table 2–54. DSP Status Register (DSP_STATUS_REG) – Offset: x14 (Continued)

Bit	Value	Function	Size	Access	Value at Hardware Reset
6		Idle peripherals	1	R	1
	0	Functional mode			
	1	Idle			
		Linked to bit 4 of the ISTR register (from DSP)			
5		Idle peripherals	1	R	1
	0	Functional mode			
	1	Idle			
		LInked to bit 3 of the ISTR register (from DSP)			
4		Interrupt acknowledged by the DSP (from DSP)	1	R	1
3		Output of TMS320C55x CPU ST3 register (from DSP), which is the CPUAVIS bit	1	R	1
2		XF is a signal from the C55x DSP core. On standard DSP devices such as the TMS320C5510, XF is connected to a pin and used as an external flag. The OMAP5910 device does not have an XF pin, so this bit is provided to show tha value of the XF bit in the DSP core status register (ST3)	1	R	1
1		Reset signal from MPU to DSP	1	R	1
0		Master reset (active low)	1	R	1

Table 2–55. DSP Boot Configuration Register (DSP_BOOT_CONFIG) – Offset: x18

Bit	Function	Size	Access	Value at Hardware Reset
15–10	Reserved	6	R/W	0
9–4	Reserved	6	R/W	0
3–0	DSP boot mode inputs (see Section 3.10.4, *Boot Mode for DSP Subsystem*, for more detail.	4	R/W	0

Table 2–56. DSP MPUI Configuration Register (DSP_API_CONFIG) – Offset: x1C

Bit	Function	Size	Access	Value at Hardware Reset
15–0	APISIZE: Specify which blocks of SARAM are accessible by the MPUI in HOM (exclusive access). The amount of SARAM is calculated by this formula: API_SIZE/2) * 8K bytes, starting from SARAM0	16	R/W	0xFFFF

Table 2–57 decodes SARAM 0 through SARAM 11 on 8K boundaries.

Table 2–57. Decoding SARAM 0 Through SARAM 11 on 8K Boundaries

	SARAM			
APISIZE (15..0)	**11**	**7**	**3**	**0**
0X0000 – 0X0001	0000	0000	0000	
0X0002 – 0X0003	0000	0000	0001	
0X0004 – 0X0005	0000	0000	0011	
0X0006 – 0X0007	0000	0000	0111	
0X0008 – 0X0009	0000	0000	1111	
0X000A – 0X000B	0000	0001	1111	
0X000C – 0X000D	0000	0011	1111	
0X000E – 0X000F	0000	0111	1111	
0X0010 – 0X0011	0000	1111	1111	
0X0012 – 0X0013	0001	1111	1111	
0X0014 – 0X0015	0011	1111	1111	
0X0016 – 0X0017	0111	1111	1111	
0X0018 – OTHERS	1111	1111	1111	

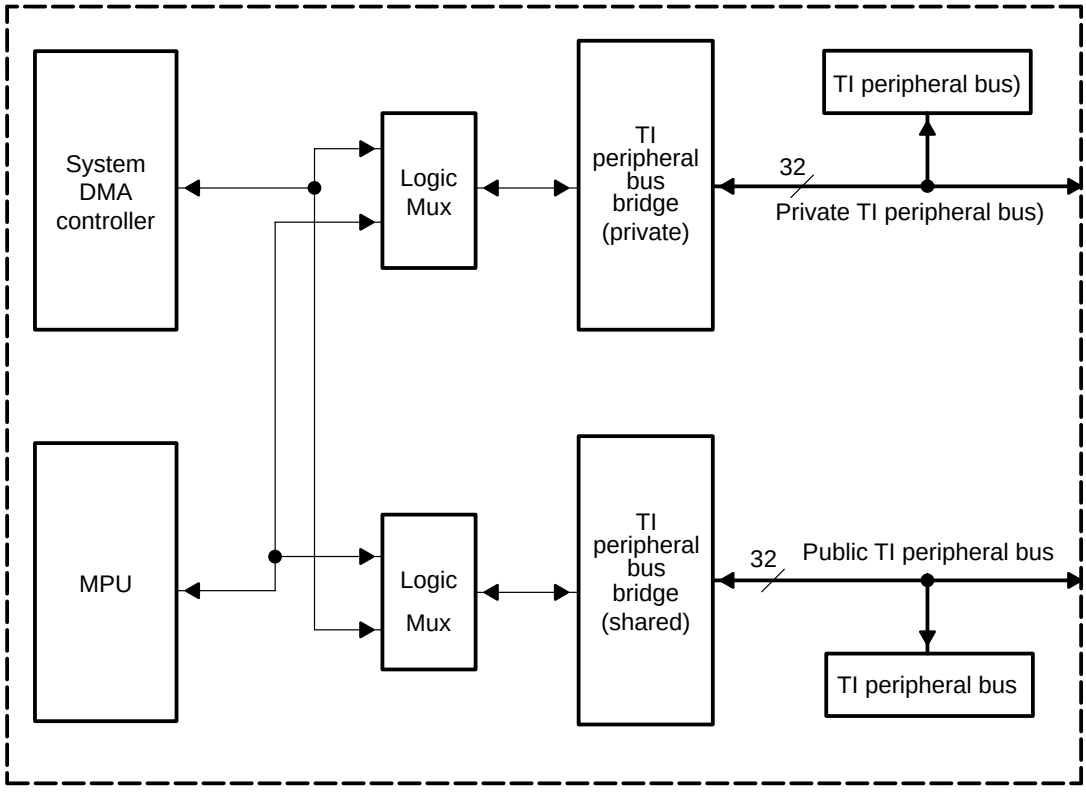
Notes: 1) 0: Shared-access RAM
2) 1: Host-only RAM (no DSP access)

2.10 MPU TI Peripheral Bus Bridges

The MPU TI peripheral bus (TIPB) bridges (see Figure 2–23) connect the TI925T to its peripherals. Two MPU TIPBs, one private and one nonprivate or public, are implemented to reduce access latency and improve system performance. Concurrent transfers are possible if there are no resource conflicts; for example, DMA transfers to the public TIPB and the TI925T both access the private TIPB simultaneously. The timers are connected on the private peripheral bus for low-latency access by an operating system, and the camera is located on the public peripheral bus for access by the DMA.

The private and public peripheral bridges are compatible with the TIPB specification.

Figure 2–23. MPU TI Peripheral Bus Bridge Connections



2.10.1 8-Bit, 16-Bit, and 32-Bit Word Access

The MPU TIPB handles 8-bit, 16-bit, and 32-bit word accesses. Data is loaded and stored in little endian fashion. Data is always right-justified on the TIPB.

2.10.2 TIPB Allocation

The MPU TIPBs are shared between the MPU and the DMA controller. A bus-allocation module is provided to resolve conflicts and prioritize accesses.

The value written in the TIPB_BUS_ALLOC register defines the priority. If the value is 0, the MPU memory interface has priority over the DMA controller. If the value equals *n* (*n* from 1 to 7), the DMA controller has priority over the MPU and it can perform *n* accesses before yielding to the MPU.

2.10.3 Access Factor and Time-Out

The MPU TIPB handles peripherals of varying speeds. To accommodate slow peripherals, the access cycle (strobe period) is programmable.

The frequency of the MPU public and private TIPB strobe 1 and 0 are derived from the traffic controller clock (CLKM3). For both TIPBs, you can use bits 3–0 (strobe 0) and bits 7–4 (strobe 1) of the TIPB control register (TIPB_CNTL) to configure the access factor and consequently the strobe frequencies (as shown in Table 2–58).

Table 2–58. Access Factor

Number of Wait States (Access Factor)	Strobe Frequency
0	TC Clk/1
1	TC Clk/2
2	TC Clk/3
3	TC Clk/4
...	...
15	TC Clk/16

Each bridge in OMAP has two strobe lines, and a different division factor can be programmed on each line.

A TIPB access time-out limits the maximum time a peripheral can stall the processor. When starting a cycle on TIPB, the time-out counter is loaded with this value (see TIPB_CNTL and ENHANCED_TIPB_CNTL registers). If the current cycle is not finished when the counter reaches 0, the cycle is aborted and an abort exception is generated to the MPU. The maximum value is 256 bridge clock cycles.

2.10.4 MPU Posted Write

The MPU can perform a posted write. When posted write is enabled inside the ARM_TIPB_CNTL register, data sent by the MPU is buffered in the MPU TIPB and the MPU can keep going to another access. The bridge takes care of the access towards the TIPB; hence the MPU is not stalled during the access.

2.10.5 Pipeline Mode

When pipeline mode is enabled in the ENHANCED_TIPB_CNTL register, incoming signals from MPU and DMA are buffered. Use pipeline mode when running at a high frequency.

2.10.6 Abort

When abort interrupt is enabled in the ENHANCED_TIPB_CNTL register, an interrupt is sent to the MPU interrupt handler when a TI peripheral read or write access is aborted or when any TI peripheral access has a size mismatch. In case of abort or size mismatch, the address and data of the corresponding access are saved in the following registers: ADDRESS_DBG, DATA_DEBUG_LOW, DATA_DEBUG_HIGH, DEBUG_CNTR_SIG.

2.10.7 TIPB Bridge Registers

Table 2–59 and Table 2–60 list the TIPB bridge registers. Table 2–61 through Table 2–68 describe the register bits.

Table 2–59. TIPB (Private) Bridge Registers

Register Name	Descriptions	R/W	Size	Address	Reset Value
TIPB_CNTL	TIPB control	R/W	16 bits	FFFE:CA00	0xFF11
TIPB_BUS_ALLOC	TIPB bus allocation	R/W	16 bits	FFFE:CA04	0x0009
MPU_TIPB_CNTL	MPU TIPB control	R/W	16 bits	FFFE:CA08	0x0000
ENHANCED_TIPB_CNTL	Enhanced TIPB control	R/W	16 bits	FFFE:CA0C	0xFFFF
ADDRESS_DBG	Debug address	R	16 bits	FFFE:CA10	0xFFFF
DATA_DEBUG_LOW	Debug data LSB	R	16 bits	FFFE:CA14	0xFFFF
DATA_DEBUG_HIGH	Debug data MSB	R	16 bits	FFFE:CA18	0xFFFF
DEBUG_CNTR_SIG	Debug control signals	R	16 bits	FFFE:CA1C	0x00F8

Table 2–60. TIPB (Public) Bridge Registers

Register Name	Descriptions	R/W	Size	Address	Reset Value
TIPB_CNTL	TIPB control	R/W	16 bits	FFFE:D300	0xFF11
TIPB _BUS_ALLOC	TIPB bus allocation	R/W	16 bits	FFFE:D304	0x0009
MPU_TIPB_CNTL	MPU TIPB control	R/W	16 bits	FFFE:D308	0x0000
ENHANCED_TIPB_CNTL	Enhanced TIPB control	R/W	16 bits	FFFE:D30C	0xFFFF
ADDRESS_DBG	Debug address	R	16 bits	FFFE:D310	0xFFFF
DATA_DEBUG_LOW	Debug data LSB	R	16 bits	FFFE:D314	0xFFFF
DATA_DEBUG_HIGH	Debug data MSB	R	16 bits	FFFE:D318	0xFFFF
DEBUG_CNTR_SIG	Debug control signals	R	8 bits	FFFE:D31C	0xF8

Table 2–61. TIPB Control Register (TIPB_CNTL) – Offset: x00

Bit	Description	Size	Access	Reset Value
15–8	TIPB bus access time out	8	R/W	0xFF
7–4	Division factor of nASTROBE[1]	4	R/W	0x1
3–0	Division factor of nASTROBE[0]	4	R/W	0x1

Table 2–62. TIPB Bus Allocation Register (TIPB_BUS_ALLOC) – Offset: x04

Bit	Value	Description	Size	Access	Reset Value
5–4		Reserved. The reset value of these bits does not have to be changed for this register to operate correctly.	2	R/W	00
3		MPU has higher priority than DMA transfers regarding TIPB allocation when it is in exception mode.	1	R/W	1
2–0		Defines TIPB priority between MPU and DMA	3	R/W	0x1
	0	MPU has priority over DMA.			
	1	DMA has priority over MPU.			

Table 2–63. MPU TIPB Control Register (MPU_TIPB_CNTL_REG) – Offset: x08

Bit	Value	Description	Size	Access	Reset Value
1	1	Write buffer is enabled for strobe domain 1.	1	R/W	0
	0	Write buffer is bypassed.			
0	1	Write buffer is enabled for strobe domain 0.	1	R/W	0
	0	Write buffer is bypassed.			

Table 2–64. Enhanced TIPB Control Register (ENHANCED_TIPB_CNTL) – Offset: x0C

Bit	Description	Size	Access	Reset Value
3	When low, a tc_abort interrupt is sent back to the MPU, when MPU TIPB access is timed out.	1	R/W	1
2	When high, incoming signals from MPU and DMA are clocked. Used when running at high frequency.	1	R/W	1
1	When low, an interrupt is sent to the MPU when a TIPB write access is aborted or when any TIPB access has a size mismatch. When high, the interrupt is masked.	1	R/W	1
0	A value of 1 enables the time-out feature.	1	R/W	1

Table 2–65. Address Debug Register (ADDRESS_DBG) – Offset: x10

Bit	Description	Size	Access	Reset Value
15–0	Address from MPU memory interface saved on abort or access size mismatch	16	R	0xFFFF

Table 2–66. Data Debug Register LSB (DATA_DEBUG_LOW) – Offset: x14

Bit	Description	Size	Access	Reset Value
15–0	Bytes 15–0 of data bus from MPU	16	R	0xFFF

Table 2–67. Data Debug Register MSB (DATA_DEBUG_HIGH) – Offset: x18

Bit	Description	Size	Access	Reset Value
15–0	Bytes 31–16 of data bus from MPU	16	R	0xFFFF

Table 2–68. Debug Control Signals Register (DEBUG_CNTR_SIG) – Offset: x1C

Bit	Description	Size	Access	Reset Value
8	Burst access	1	R	0
7–6	Peripheral memory access size on TIPB	1	R	3
5–4	Memory access size on TIPB	1	R	3
3	Not supervisor mode on TIPB	1	R	1
2	Read not write on TIPB	1	R	0
1	Flag set to 1 when there is a mismatch between memory access size and peripheral memory access size.	1	R	0
0	Flag set to 1 when TIPB access is aborted.	1	R	0

2.11 Endianism Conversion

Because the TI925T is operated in little endian mode and the DSP is operated in big endian mode, shared data must be converted to their respective formats before any processing is done. Table 2–69 and Table 2–70 illustrate the little and big data formats, respectively. In each case, the data is a reference to the little endian format.

There are two ways to share the data between the TI925T and the DSP:

- ❏ Through the DSP MMU
- ❏ Through the MPUI

Endianism conversion is implemented between these two modules and the DSP. The hardware converts both program code and data from big endian to little endian mode when writing to the system memory and from little endian to big endian mode when reading back from the system memory for all the data access sizes when the logic is enabled.

Endianism conversion is performed in hardware, so that the data swapping is transparent to software (reduce software overhead to format the data).

A bypass path is also implemented. If this case is not covered, the swapping logic can be disabled and the conversion handled by software.

Table 2–69. Little Endian Data Format

Little Endian Format (32-Bit Word Access)			
31 24	**23 16**	**15 8**	**7 0**
Word 1		**Word 0**	
Byte 3	Byte 2	Byte 1	Byte 0
AA	BB	CC	DD

Table 2–70. Big Endian Format

Big Endian Format (32-Bit Word Access)			
31 24	**23 16**	**15 8**	**7 0**
Word 0		**Word 1**	
Byte 0	Byte 1	Byte 2	Byte 3
DD	CC	BB	AA

2.11.1 Conversion Through the DSP MMU

Swapping buffers are implemented at the boundary between the DSP and the DSP MMU (see Figure 2–24). Assuming that the OMAP5910 system memory is organized in little endian mode, data from and to the DSP is converted as follows:

- Data is written from the DSP to the system memory.

  In the DSP, data is organized in big endian mode (see Table 2–70), but the bytes are swapped in order to recognize the data in little endian mode (see Table 2–69).

- Data is read from the system memory to the DSP.

  In system memory, data is organized in little endian mode, but the bytes are swapped in order to reorganize the data in big endian mode.

- 32-bit word is written from the DSP to the system memory, but beginning at an odd address (for example, 0x000002) or with the bit *byte_nword* set to 0.

  In the DSP, the data is organized in DSP data format (see Table 2–71), but the 16-bit words are swapped in order to reorganize the 32-bit data in little endian mode.

- 32-bit word is read from the system memory to the DSP, but beginning at an odd address (for example, 0x000002) or with the bit *byte_nword* set to 0.

  In system memory, the data is organized in little endian mode, but the 16-bit words are swapped in order to reorganize the 32-bit data in the DSP data format.

Note:

16-bit word and single byte accesses are always right justified. The swapping logic is power-up disabled.

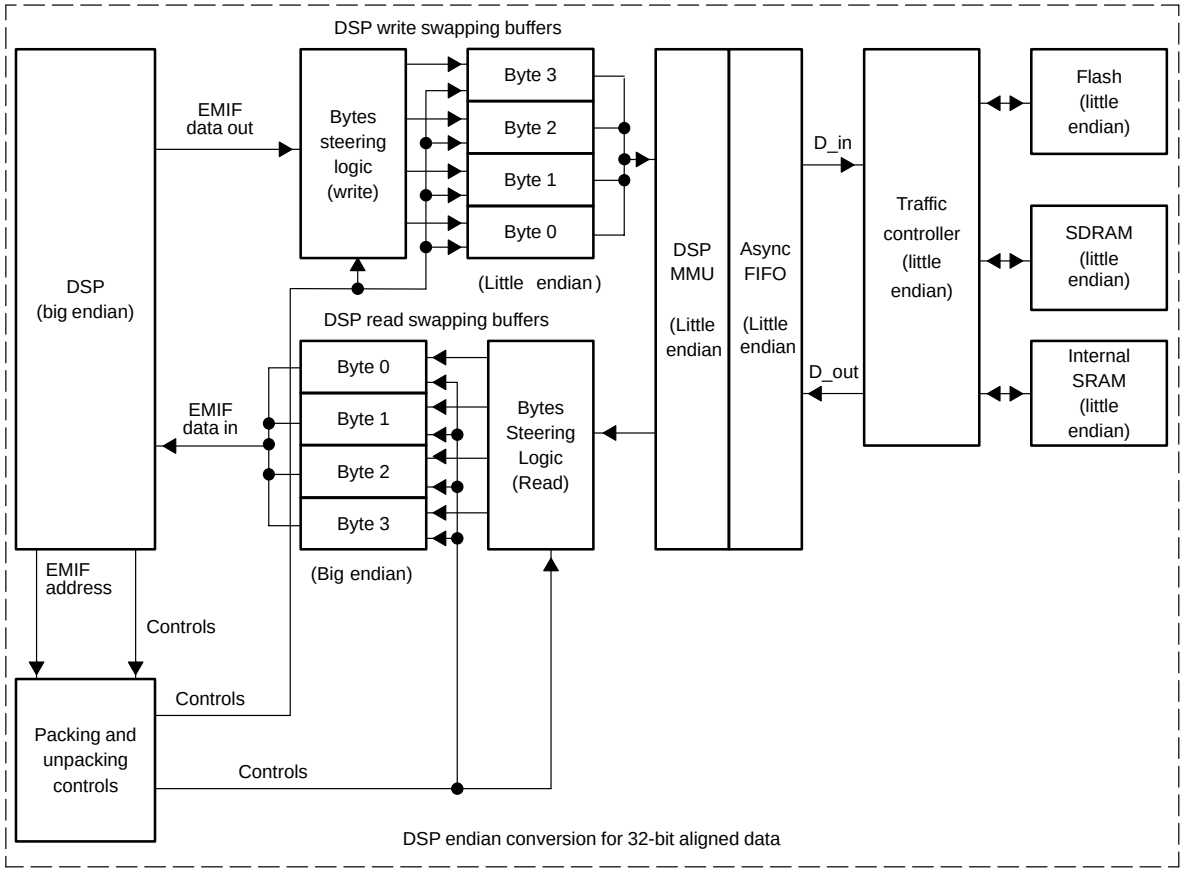
Table 2–71. DSP Data Format

DSP Data Format (32-Bit Word Access— Odd Address or Enabled byte_nword Option)			
31 24	**23 16**	**15 8**	**7 0**
Word		**Word 1**	
Byte 1	Byte 0	Byte 3	Byte 2
CC	DD	AA	BB

Figure 2–24 shows the endian conversion at the DSP MMU interface boundary. The byte and word swapping is done by decoding the data width and data size, then repacking the data into the appropriate formats.

The byte-steering logic provides a mechanism to convert from big to little, little to big, or upper and lower word swap for program code and data accesses.

Figure 2–24. DSP Endian Conversion, 32-Bit Aligned Data



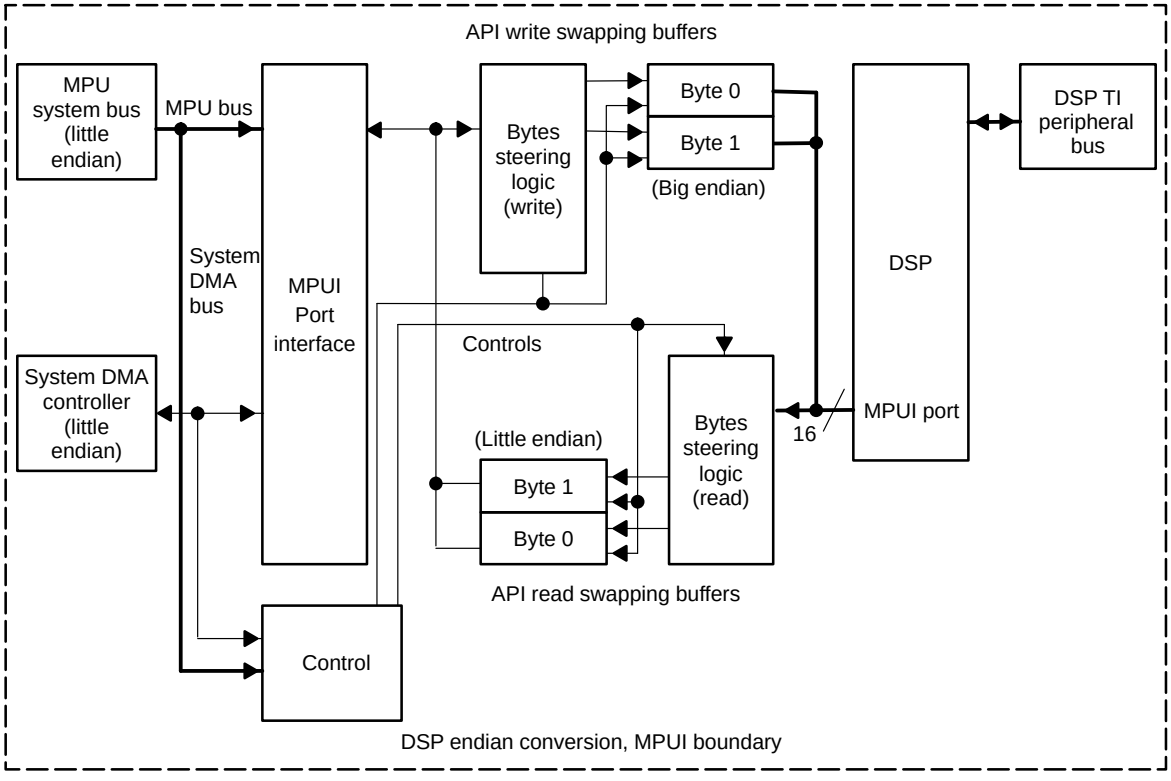
Note: The steering logic puts the byte/word/double-word in appropriate formats.

2.11.2 Conversion Through the MPUI

Swapping buffers are implemented at the boundary between the DSP and the MPUI (See Figure 2–25).

The word and byte swapping can be programmed so swapping is individually controlled for MPU memory access and non-MPU memory (peripheral and MPU register).

Figure 2–25. DSP Endian Conversion, MPUI Port Boundary



Note: The steering logic puts the byte/word/double-word in appropriate formats.

The MPUI port has a 16-bit data bus, thus all 32-bit accesses are divided into two 16-bit accesses. 16-bit word swapping and byte swapping are programmable.

By default:

- ❏ Byte swapping is disabled for all accesses.
- ❏ 16-bit word swapping is enabled for all accesses.

2.12 ETM Environment

The OMAP5910 device has an embedded trace macrocell (ETM) to provide instruction and data trace capabilities of the TI925T processor. ETM9 in large configuration uses an 8-bit data output. The instruction trace shows the instruction flow of the MPU. The data trace shows the data access results after the MPU executes load and store operations.

2.12.1 ETM Features

The ETM has the following features:

- ❏ Instruction/data trace
- ❏ 8-bit trace packet width
- ❏ 45-byte trace packet capture FIFO
- ❏ Eight pairs of address comparators for trace trigger
- ❏ Height comparators for trace trigger
- ❏ Four 16-bit counters
- ❏ 3-state sequencer (state machine)

2.12.2 ETM Interface

The ETM logical signal interface contains 13 trace interface pins and nine JTAG interface pins. The ETM trace interface has the following signals:

- ❏ TRACEPKT[0..7]

  The TRACEPKT signals comprise the 8-bit data trace packets (packaged address and data information).

- ❏ PIPESTAT[0..2]

  The PIPESTAT signals are used to output the MPU pipeline at the MPU execute stage on every TRACECLK and are used by software to reconstruct the compressed trace output.

- ❏ TRACESYNC

  The TRACESYNC signal is used to indicate when the first of multiple packets are to be output on the TRACEPKT bus.

- ❏ TRACECLK

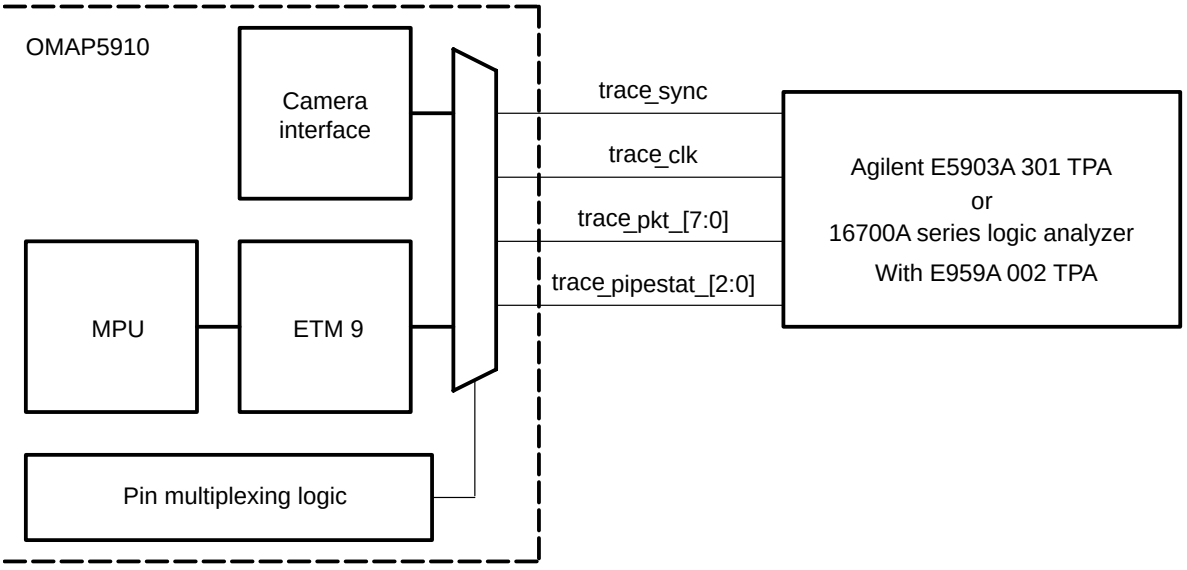
  The TRACECLK operates at one of two frequencies:

  - The same frequency as the MPU
  - The MPU frequency divided by two (half-rate clocking)

    When this rate is selected, the trace port analyzer (TPA) samples the trace data signals on both the rising and the falling edges of the TRACECLK. Bit 13 of the ETM control register enables you to select this rate.

The ETM trace signals are multiplexed with the camera interface pins on the OMAP5910 device. The default value upon reset is the camera interface. Refer to Section 6.8, *Configuration Module*, and Section A.2, *I/O Functional Multiplexing*, for details on pin multiplexing.

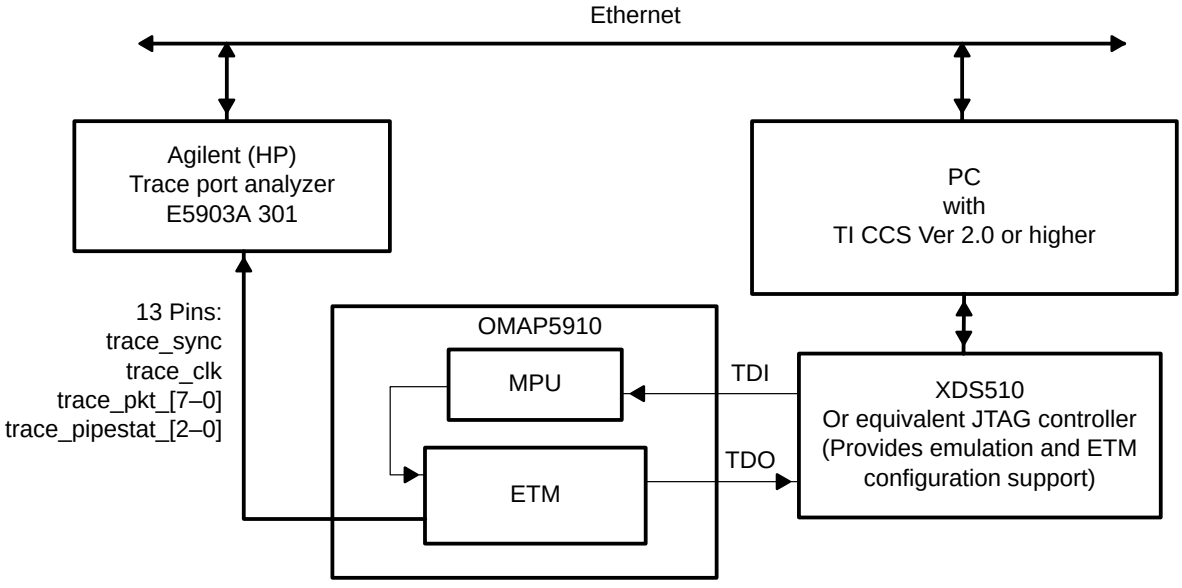
Figure 2–26. Trace Signals Multiplexing



2.12.3 Operation

Figure 2–27 shows how the OMAP5910 ETM is used in a system setup for capture of trace data.

Figure 2–27. Required System for ETM Usage



The TI Code Composer Studio™ Integrated Development environment (IDE) software, trace port analyzer (TPA), and the emulation probe hardware are used for tracing and displaying the MPU operation.

Agilent provides two types of trace port equipment:

- ❏ Dedicated trace port analyzer (TPA) (E5903A #301)
- ❏ A 16700A series logic analyzer used with an analysis probe (E9595A#002)

The Code Composer Studio™ IDE provides support only for the TPA setup.

The Code Composer Studio™ IDE provides a complete interface to the ETM, including the setup of the trace registers, trigger points, and sequencing of trace operations. When a trace trigger occurs, Code Composer Studio™ IDE decompresses and formats the trace information for display. In addition, when the TI software development tools are used, Code Composer Studio™ IDE can also correlate trace data back to the source code, thus providing complete symbolic trace capabilities. All of the ETM functions operate in parallel with the standard debug features provided by Code Composer Studio™ IDE, such as breakpoints, single-stepping, etc.

2.12.4 Additional Reference Materials

Additional MPU (ARM) ETM related publications of interest include:

- *ETM9 (Rev 0/0a) Technical Reference Manual* (ARM DDI 0157B)
- *Trace Port Analysis for ARM ETM Users Guide* (Agilent Publications, publication number E5903-97000)
- *Embedded Trace Macrocell (Rev 1) Specification* (ARM IHI 0014E)

Documentation is also available from Advanced RISC Machines directly via *http:\\www.arm.com*.

Chapter 3

DSP Subsystem

This chapter describes the OMAP5910 multimedia processor DSP subsystem.

Topic **Page**

3.1 Architecture Overview

The digital signal processor (DSP) subsystem is built around a core processor and peripherals that interface with:

- ❏ The TI925T 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 3–1 shows the OMAP5910 device with the DSP subsystem highlighted. Figure 3–2 shows the subsystem and the modules with which it interfaces.

Figure 3–1. Highlight of DSP Subsystem

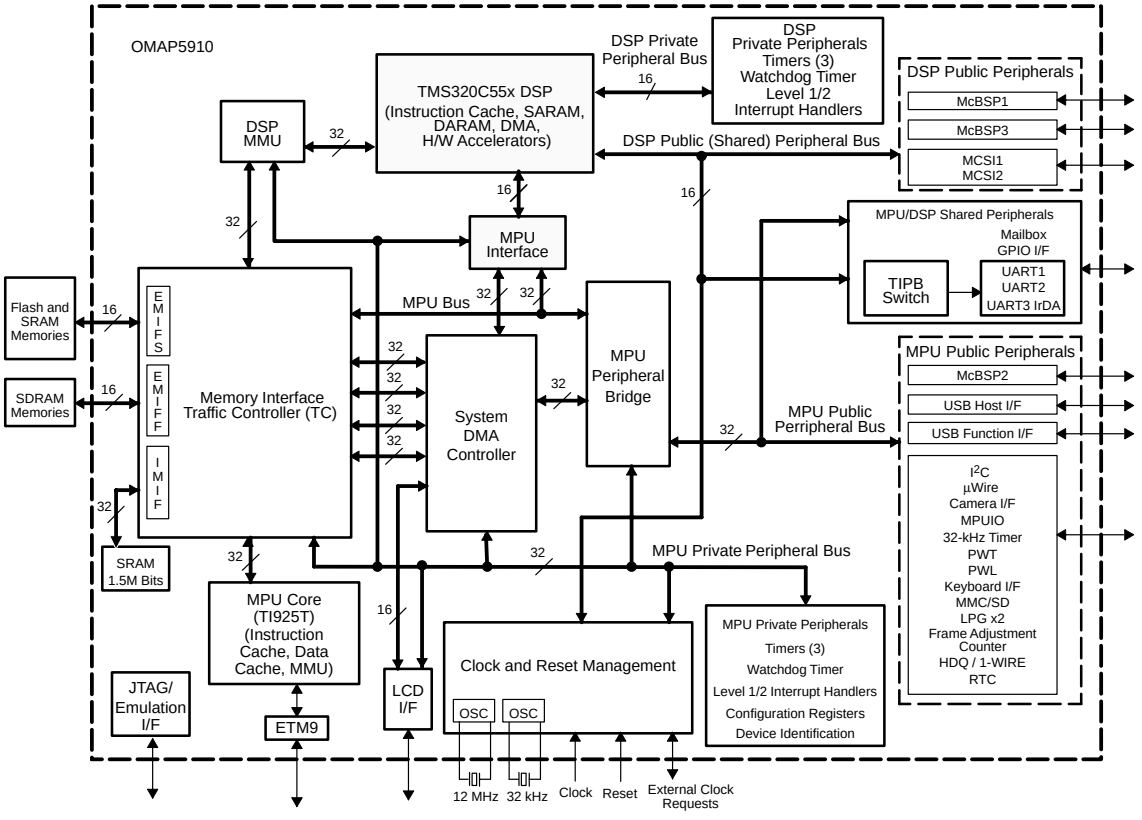
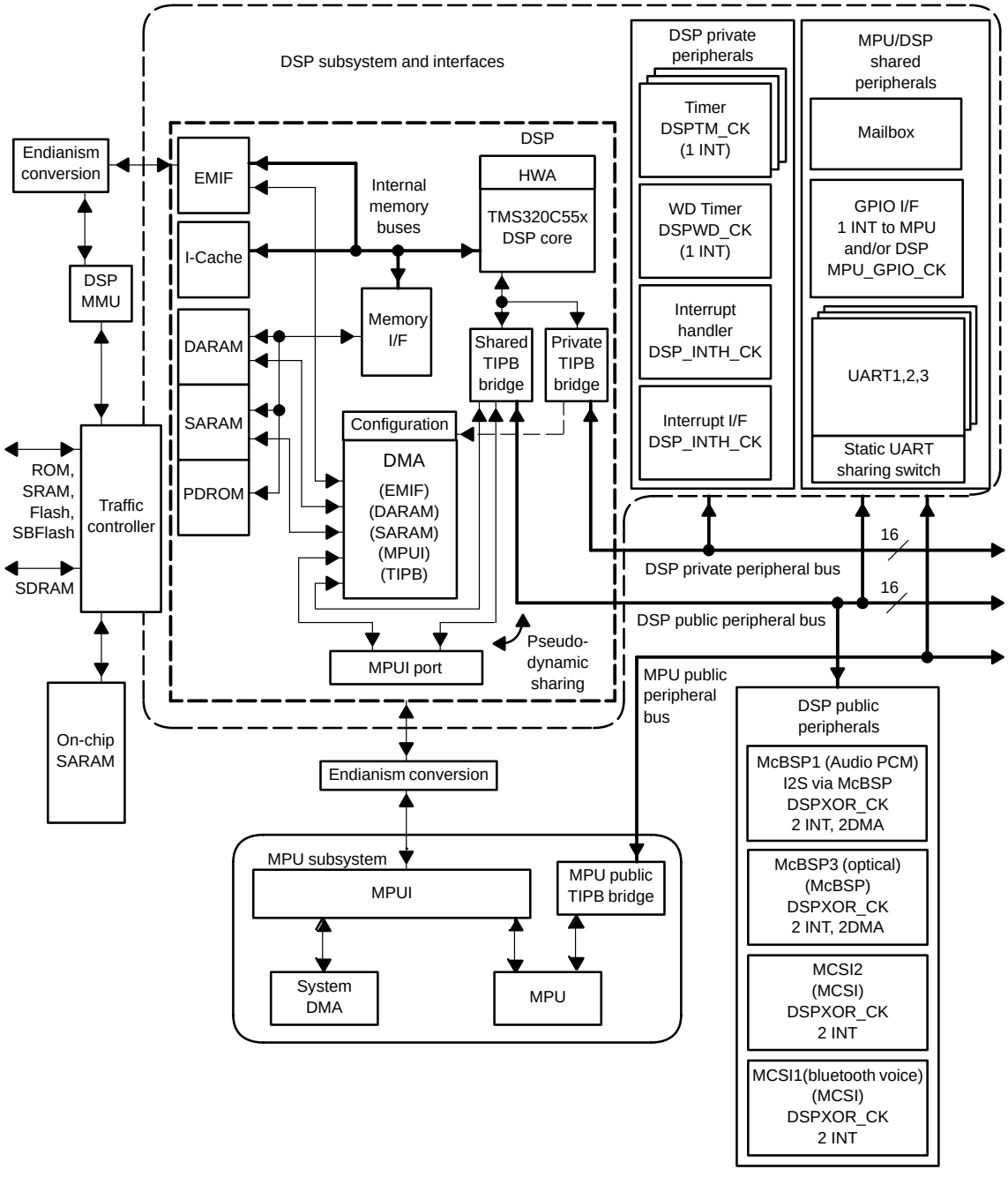


Figure 3–2. DSP Subsystem and Modules



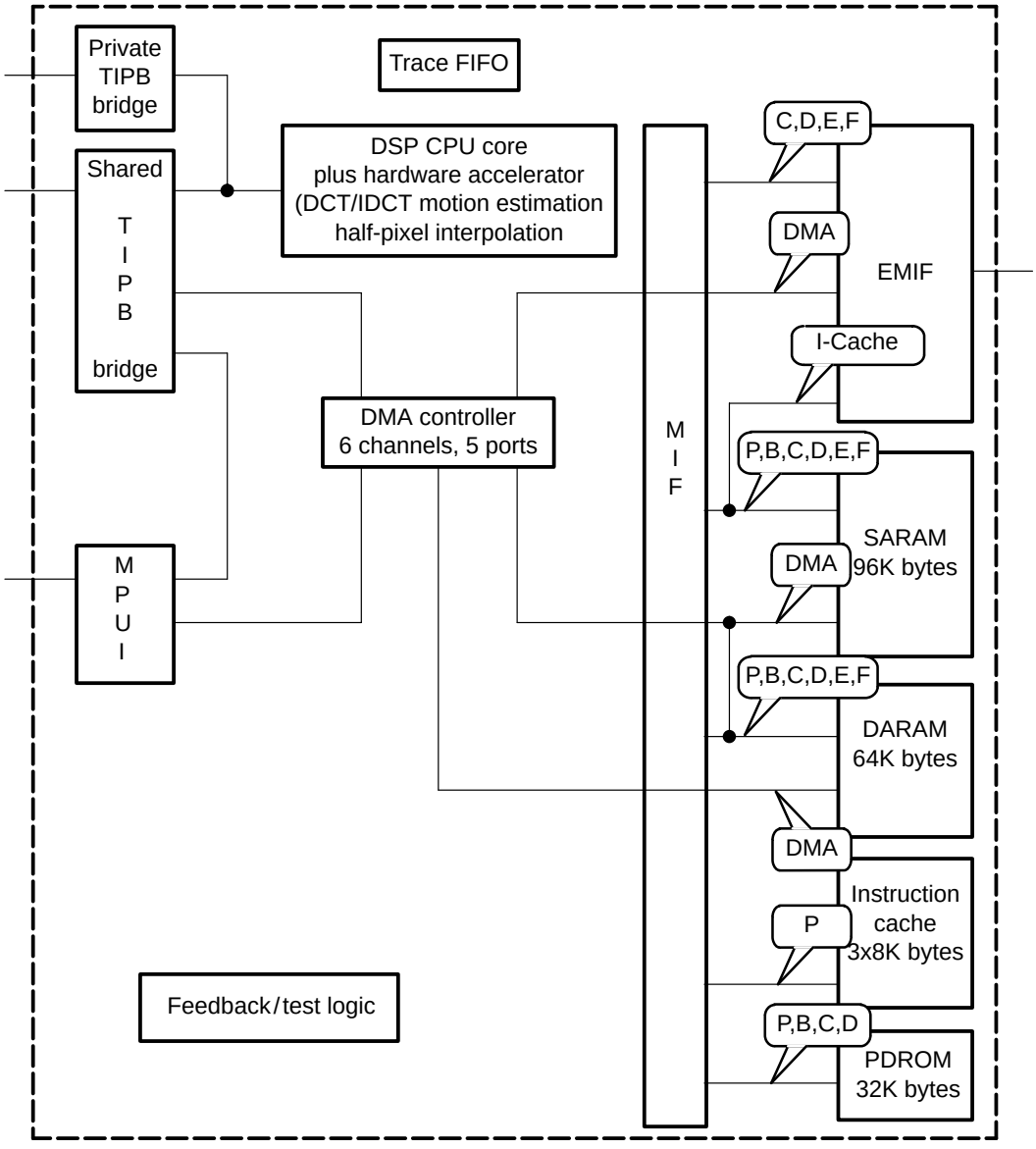
The DSP subsystem has the following components:

- DSP module:
  - TMS320C55x (C55x) DSP CPU core
  - Tightly coupled hardware accelerators—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), instruction cache
  - External memory interface (EMIF) that connects the CPU to external and loosely coupled memories
  - A 6-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

3.1.1 DSP Core

Figure 3–3 shows the DSP core.

Figure 3–3. DSP Core and Internal Bus Designations



3.2 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

3.2.1 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

3.2.1.1 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

3.2.2 Hardware Acceleration Modules

The OMAP5910 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

3.2.3 CPU Overview

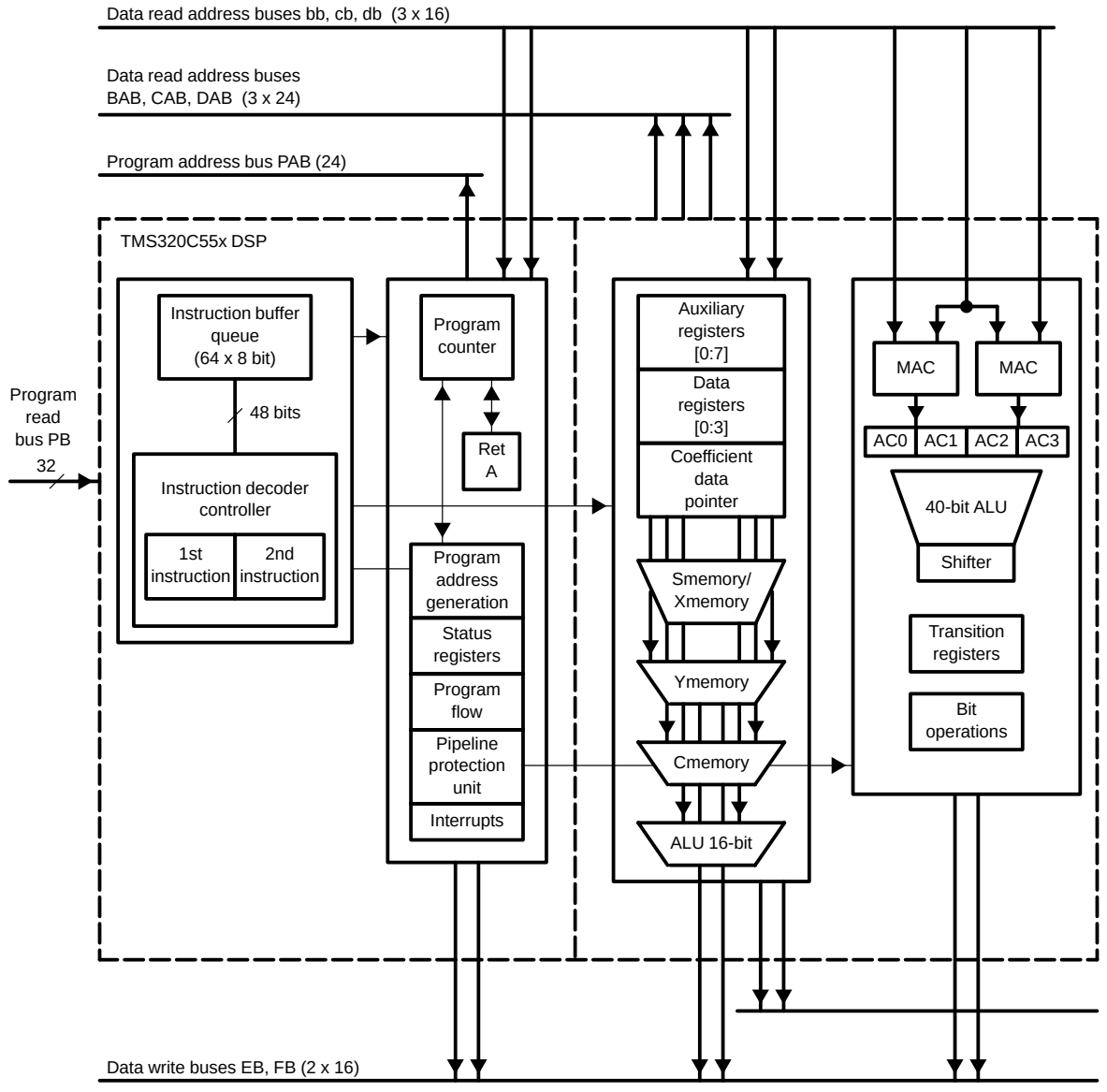
The DSP core has four functional units:

- 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 3–4 shows the C55x DSP architecture.

Figure 3–4. 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 OMAP5910 device)

3.3 DSP Memory

The DSP subsystem contains four types of tightly coupled memory to enable maximum efficiency of the DSP CPU.

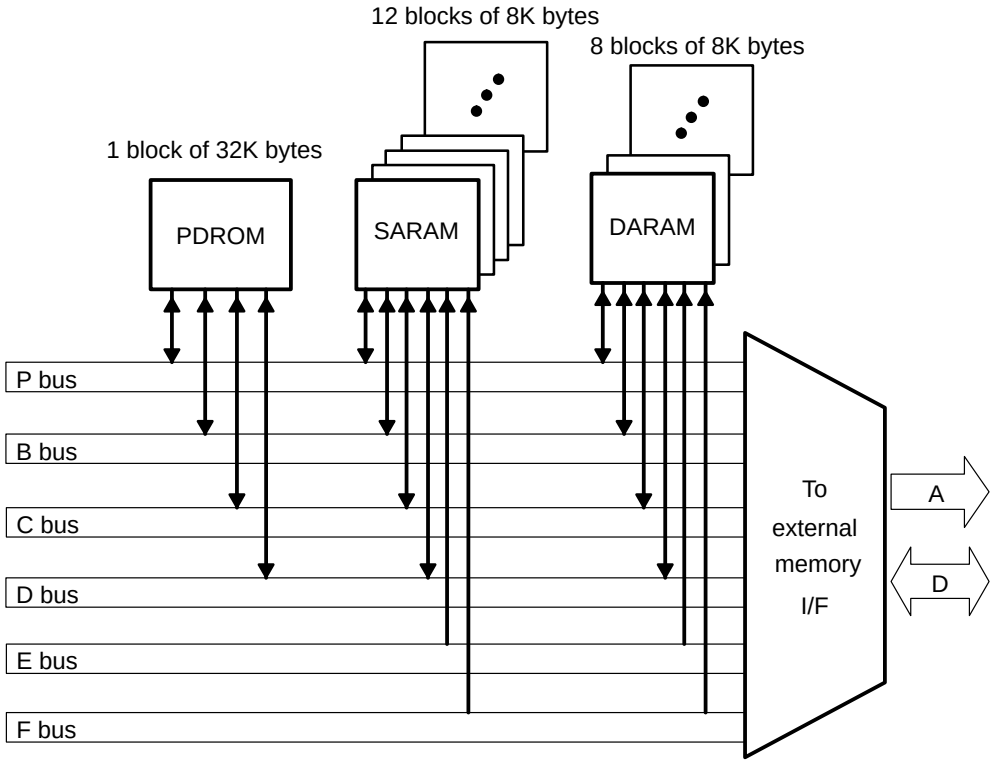
- Dual-access RAM (DARAM)
- Single-access RAM (SARAM)
- Programmable dynamic ROM (PDROM)
- Configurable instruction cache structure

The CPU uses six sets of buses to simultaneously fetch up to 32 bits of program and 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) Chapters 3 and 4.

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 3–5 shows DSP memory connections.

Figure 3–5. DSP Memory Connections



3.3.1 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 8 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.

3.3.2 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 consists of the following:

1) One 2-way cache. The cache uses two-way set associative mapping and holds up to 16K bytes: 512 sets, two lines per set, four 32-bit words per line.
2) Two RAM sets (1 and 2). These two banks of RAM can be used to store blocks of code. Each RAM set holds up to 4K bytes: 256 lines, four 32-bit words per line. Before enabling the I-cache, you configure the I-cache to use zero, one, or both RAM sets

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 (i.e., a branch not taken is always assumed). With this feature, no-branch continuous fetches are no-wait-state operations. The instruction 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 due to 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 instruction cache supports emulation debug read and breakpoint/watchpoint insertion by invalidating cache lines with the corresponding data changed in the external memory space.

The DSP I-cache on OMAP5910 functions as described in the *TMS320C55x DSP Instruction Cache Reference Guide* (literature number SPRU576). See this document (I-Cache Type A section) for additional details on the DSP I-cache operation.

Table 3–1 lists the DSP I-Cache I/O-mapped registers.

Table 3–1. DSP I-Cache Input/Output Memory-Mapped Control Registers

Register	Description	Access	Word Address	Reset Value
ICGR	I-cache global control	R/W	0x1400	C006h
Reserved	Reserved	R/W	0x1401	0000h
Reserved	Reserved	R/W	0x1402	0000h
ICWC	I-cache way control	R/W	0x1403	000Dh
ISR	I-cache status	R	0x1404	0000h
ICRC1	I-cache ½ ramset 1 control	R/W	0x1405	000Dh
ICRTAG1	I-cache ½ ramset 1 TAG	R/W	0x1406	0000h
ICRC2	I-cache ½ ramset 2 control	R/W	0x1407	000Dh
ICRTAG2	I-cache ½ ramset 2 TAG	R/W	0x1408	0000h

3.3.3 System Memory

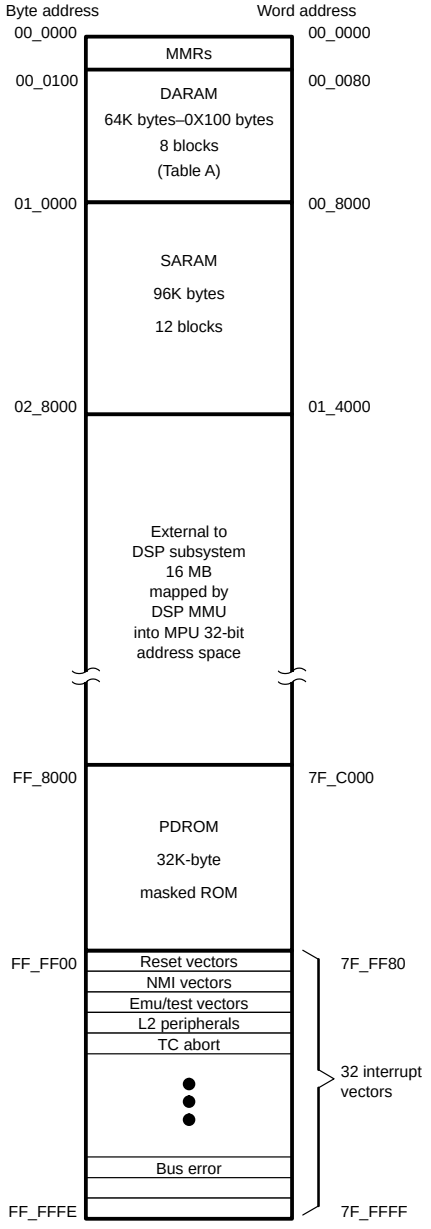
The DSP has access to all system memory managed by the traffic controller. External memory space ranges from 0x50000 to 0xFF8000 if the internal PDROM is enabled, or to 0xFFFFFF if the PDROM is not enabled.

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.

3.3.4 Memory Map

Figure 3–6 shows the DSP memory space.

Figure 3–6. 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 OMAP5910 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).

I/O Space

Byte		Word	Byte		Word
00000	TIPB bridge	00000	10000	UART1	08000
00800		00400	10800	UART2	08400
01000	EMIF	00800	11000		08800
01800	DMA	00C00	11800	McBSP1	08C00
02000		01000	12000	MCSI2	09000
02800	I-Cache	01400	12800	MCSI1	09400
03000		01800	13000		09800
03800		00C00	13800	I-Cache	09C00
04000		02000	14000		0A000
04800		02400	14800		0A400
05000	Timer1	02800	15000		0A800
05800	Timer2	02C00	15800		0AC00
06000	Timer3	03000	16000		0B000
06800	WD_Timer	03400	16800		0B400
07000		03800	17000	McBSP3	0B800
07800		03C00	17800		0BC00
08000	CLKM2	04000	18000		0C000
08800		04400	18800		0C400
09000	L2 Int handler	04800	19000		0C800
09800		04C00	19800	UART3	0CC00
0A000		05000	1A000		0D000
0A800		05400	1A800		0D400
0B000		05800	1B000		0D800
0B800		05C00	1B800		0DC00
0C000		06000	1C000		0E000
0C800		06400	1C800	UART1,2,3 shading SW	0E400
0D000		06800	1D000		0E800
0D800		06C00	1D800		0EC00
0E000		07000	1E000	GPIO	0F000
0E800		07400	1E800		0F400
0F000		07800	1F000	Mailbox	0F800
0F800		07C00	1F800	DSP MPUI register	0FC00

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.

3.3.5 Peripheral Register Addresses

The DSP CPU and the DMA controller can access several classes of peripheral devices:

- DSP private peripherals (see Chapter 8)
  - Three general-purpose timers
  - A watchdog timer
  - An interrupt handler
- MPU/DSP shared peripherals (see Chapter 10)
  - Communications mailbox
  - GPIO control
  - General-purpose UART
- DSP public peripherals (see Chapter 9)
  - 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 3–2 shows the DSP peripheral mapping.

Table 3–2. DSP Peripheral Mapping

Start Byte Address (hex)†	Name	Word Address	Strobe‡
x000000	TIPB bridge	00000	Strobe 1
x001000	EMIF	00800	Fixed strobe period
x001800	DMA	00C00	Fixed strobe period
x002800	I-cache	01400	Fixed strobe period
x005000	TIMER 1	02800	Fixed strobe period
x005800	TIMER 2	02C00	Fixed strobe period
x006000	TIMER 3	03000	Fixed strobe period
x006800	WD_TIMER	03400	Fixed strobe period
x008000	CLKM 2	04000	Strobe 1
x009000	Level 2 interrupt handler	04800	Fixed strobe period
x0010000	UART1	08000	Strobe2
x0010800	UART2	08400	Strobe2
x0011800	McBSP1	08c00	Strobe2
x0012000	MCSI2	09000	Strobe2
x0012800	MCSI1	09400	Strobe2
x0017000	McBSP3	0B800	Strobe2
x019800	UART3	0CC00	Strobe2
x01C800	UART1, 2, 3 sharing switch	0E400	Strobe2
x001E000	GPIO	0F000	Strobe2
x001F000	Mailbox	0F800	Strobe2
x001F800	DSP MPUI register	0FC00	Strobe2

† All other I/O memory addresses are reserved.
‡ Internal wait states for accessing peripherals are set by strobe1 and strobe2 fields in TIPB CM register (see Section 3.5.1, *Control Mode Register*).

3.4 DMA Controller

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 3–7 shows the ports serviced by the DMA controller within the context of the DSP subsystem.

3.4.1 Key Features of the DMA Controller

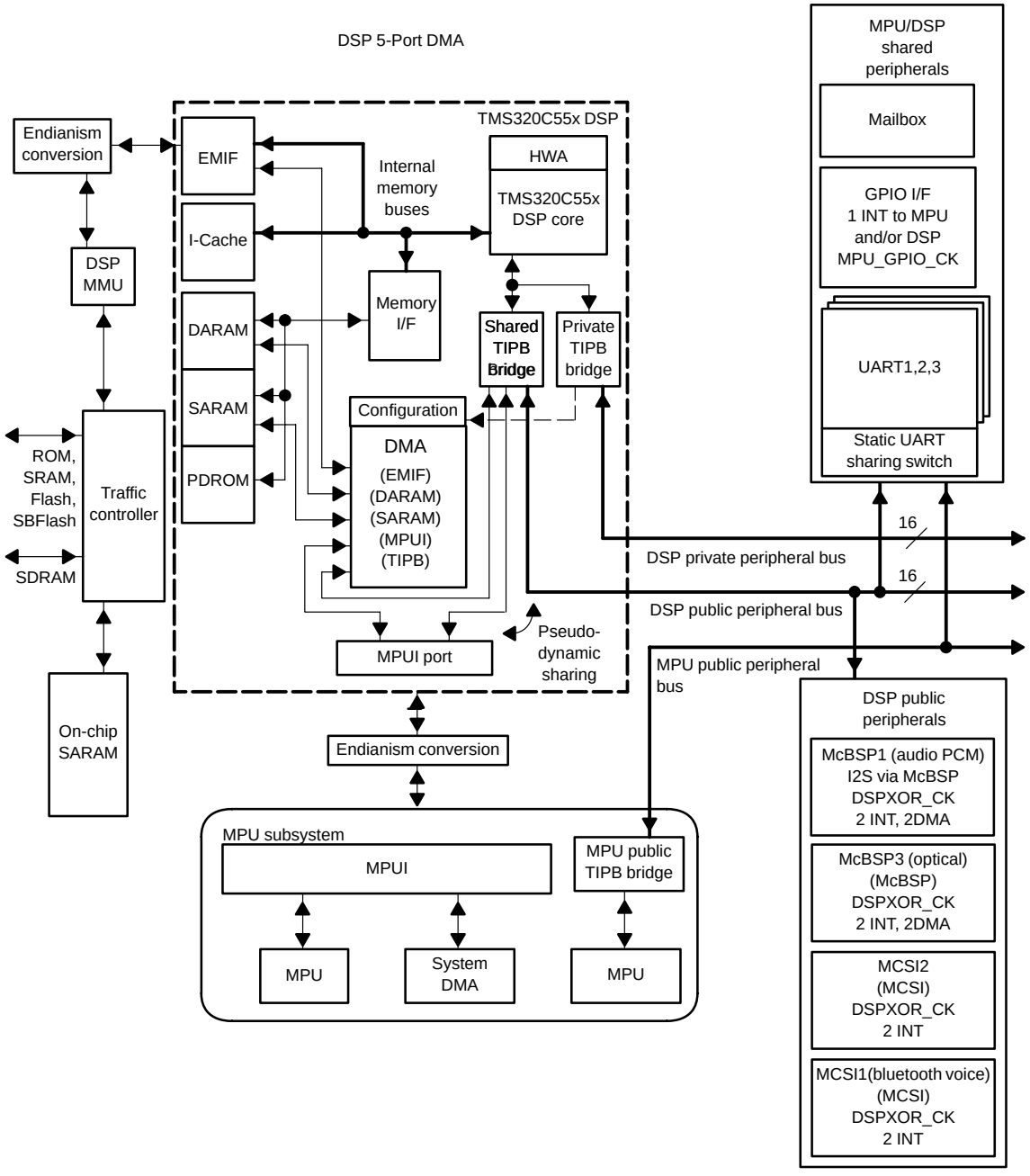
The DMA controller has the following important features:

- Operation independent of the MPU
- Four standard ports, one for each data resource: DARAM, SARAM, external or shared system memory via DSP 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.
- Event synchronization. DMA transfers in each channel can be made dependent on the occurrence of selected events.
- An interrupt for each channel. Each channel can send an interrupt to the DSP CPU on completion of certain operational events.
- Software-selectable options for updating addresses for the sources and destinations of data transfers.

The DMA controller performs data transfers between the following source and destination ports:

- Single-access RAM and SARAM port
- Dual-access RAM and DARAM port
- External memory and EMIF port
- TI peripheral bus and PERIPH port
- MPUI interface and MPUI port

Figure 3–7. DMA and Ports



Data transfers among the SARAM, DARAM, EMIF, and PERIPH ports can occur in six independent DMA channels. 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 (see Section 3.6, *MPU Interface*). 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 3–3 and Figure 3–8). To arbitrate simultaneous requests, the DMA controller has one programmable service chain that is used by each of the standard ports.

The complete operation of the OMAP5910 DSP DMA controller is described in detail in the *Direct Memory Access (DMA) Controller* section of the TMS320C55x Peripherals Reference Guide (literature number SPRU317). The OMAP5910 DSP DMA controller is consistent with SPRU317 with the following exceptions and clarifications:

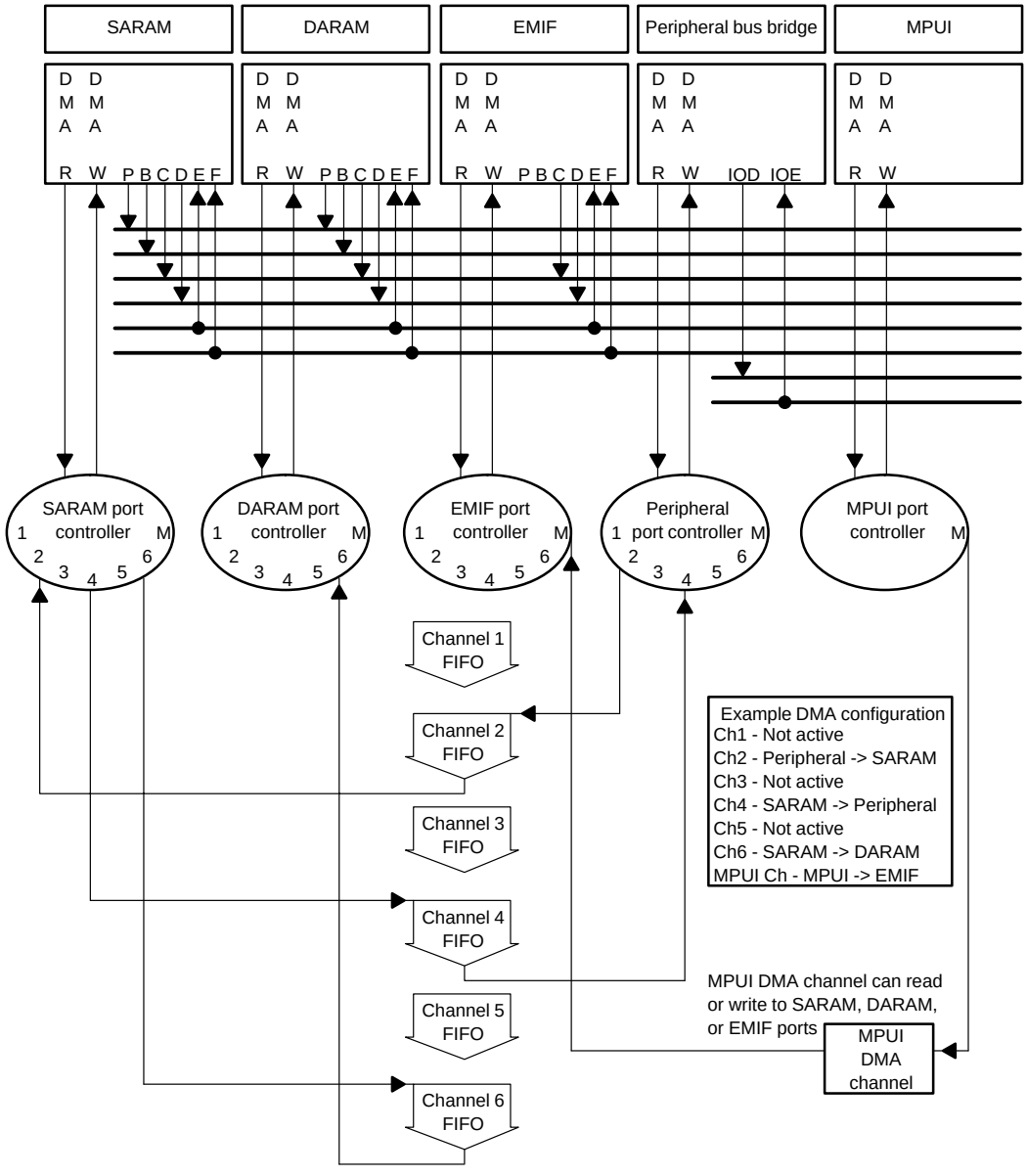
- All references to EHPI (enhanced host port interface) are equivalent to MPUI (MPU interface) on OMAP5910.
- References to the EHPI_PRIO and EHPI_EXCL bits in the DMA_GCR register are equivalent to the MPUI_PRIO and MPUI_EXCL bits on OMAP5910
- Clarification of DMA channel syncronization is described in Section 3.4.2.

Table 3–3. 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 3–8. Example of DMA Configuration



3.4.1.1 DMA Channel 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 3–4):

- 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 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 do not begin until the synchronization event is detected. When the channel is operating in frame-synchronization mode (DMA_CCR_FS = 1), several prereads may 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 3–4. 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

3.4.2 DMA Controller Configuration Registers

Table 3–5 lists the DMA controller configuration registers.

Table 3–5. 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
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

Table 3–5. DMA Controller Configuration Registers (Continued)

Register	Description	Word Address
	Channel 1 (continued)	
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
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

Table 3–5. DMA Controller Configuration Registers (Continued)

Register	Description	Word Address
	Channel 2 (continued)	
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
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

Table 3–5. DMA Controller Configuration Registers (Continued)

Register	Description	Word Address
	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
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

Table 3–5. DMA Controller Configuration Registers (Continued)

Register	Description	Word Address
	Channel 5 (continued)	
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

3.4.3 DSP DMA Event Mapping

Table 3–6 defines the mappings of the DMA channel synchronization settings to the different request sources that can be used to create DSP DMA events on OMAP5910.

Table 3–6. 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
Reserved	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
Reserved	DMA_REQ_016	10000
Reserved	DMA_REQ_017	10001
UART3 TX	DMA_REQ_018	10010
UART3 RX	DMA_REQ_019	10011

3.5 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 3–9):

- Private TIPB: peripherals connected here (timers, interrupt handler) cannot be accessed by the MPU via the MPUI.
- Public TIPB: peripherals connected here (McBSP1, McBSP2, MCSI1, MCSI2, Mailbox, GPIO UART1-3) can be accessed by the MPU via the MPUI port.

See Chapters 8 and 9 for details on DSP private and public peripherals.

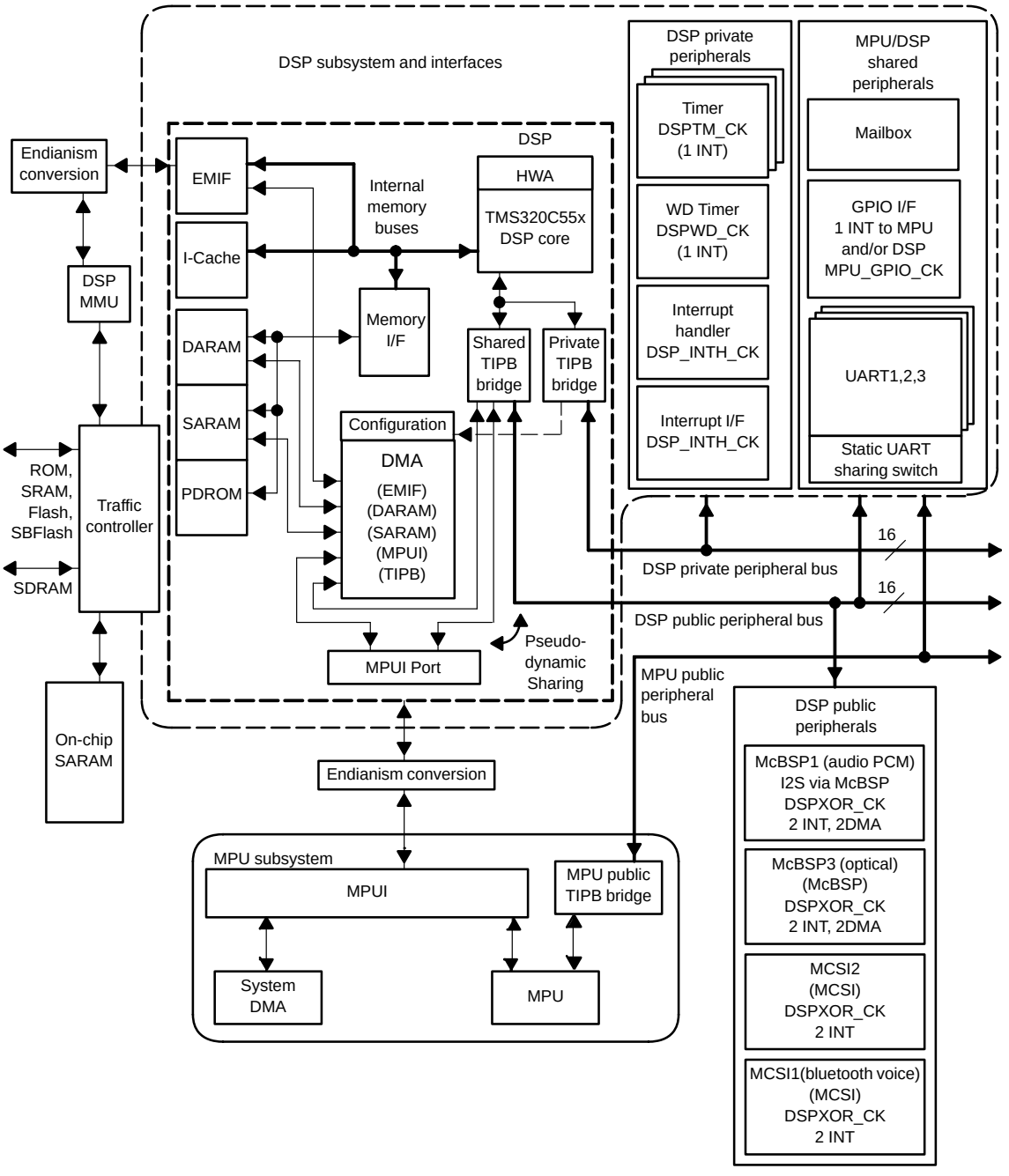
The TIPB bridge consists of two components:

- 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 may 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 3–9. DSP Subsystem Modules



3.5.1 Control Mode Register (CMR)

The CMR indicates 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 3–7. 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 (strobe2)	Strobe1 length (low, medium, high bits)	0	Read/Write	Read
5–3	Wait state (strobe1)	Strobe1 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, *MPU Interface*.

- Bus error

  This bit is set to 1 if the TIPB bridge generates a bus error (due to a time-out condition or SAM/HOM change error), indicating that an error signal has been sent to the DSP CPU, which can read this bit to identify the source of the error condition. 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 strb1 and strb2

  Strb1 field sets the access rate for the following peripherals:

  - TIPB registers
  - CLKM2 registers

  Strb2 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

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 3–8).

Table 3–8. 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).

3.5.2 Idle Control and Idle Status Registers (ICR and ISTR)

To conserve power, the DSP subsystem is capable of idling certain circuits. The DSP CPU and peripherals comprise 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 ICR (see Table 3–9). The current idle domain status is reflected by the state of the ISTR (see Table 3–10).

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 actually idling the DPLL source by writing the appropriate control registers (see Chapter 15, *Clock Generation and System Reset Management*).

WARNING

The DSP must not attempt to read the ISTR while DPLL domain is idled, because this causes a time-out error.

Table 3–9. 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 3–10. 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

3.6 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. 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 may 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 need to be user-configured for MPUI support.

The MPU domain (including TI925T 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 5 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 (TI925T 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 prior to 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.

3.6.1 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 has actually completed the mode switch. Therefore, the DSP polls bits 8 and 9 after requesting a mode change to ensure 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 (see Section 2.9, *MPU Interface*, for details).

3.6.2 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.

3.6.3 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.

See Section 2.9, *MPU Interface*, for complete details on usage, control, and configuration of the MPUI and associated control registers.

3.7 EMIF

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)

3.7.1 EMIF Global Control Register (EMIF_GCR)

The EMIF global control register (GCR) configures general operation of the EMIF module. The EMIF GCR appears at word address 0x0800 in the DSP I/O space.

Table 3–11. 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
1	Reserved		R	0
0	Reserved		RW	0

3.7.2 EMIF Global Reset Register (EMIF GRR)

Any write in the EMIF global reset register (GRR) register brings about 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 , etc.); only the EMIF state machines are reset. The EMIF GRR appears at word address 0x0801 in the DSP I/O space.

3.8 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 OMAP5910 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 performs translation of 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 may 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 TI925T. 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.

The MPU is responsible for configuration of the MMU. See Section 2.8, *DSP Memory Management Unit*, for complete details on MMU configuration and control.

3.9 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 OMAP5910. The clock domains in the OMAP5910 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 OMAP5910 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. See Chapter 15, *Clock Generation and System Reset Management*, for details on the clock generator peripheral. CLKM2 is considered an MPU private peripheral, except for configuration of subdomain clocks for the DSP subsystem discussed in Section 15.2.5.

3.10 System Operating Details

3.10.1 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, which are described in detail in Chapter 8, *DSP Private Peripherals*.

- 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.

3.10.2 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, which are described in detail in Chapter 9, *DSP Public Peripherals*:

- 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 OMAP5910 CLKIN signal.

The access rate to these peripherals is configured by strobe 2 control bits in the DSP TIPB CMR register. See Section 3.5.1, *Control Mode Register*.

3.10.3 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, which are described in detail in Chapter 10, *MPU/DSP Shared Peripherals*.

- ❏ 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 3.5.1, *Control Mode Register*.

3.10.4 Boot Mode for DSP Subsystem

The OMAP5910 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 3–12 shows the DSP boot configuration.

Table 3–12. DSP Boot Configuration

Relative Word Address	Register Name	Bits	Reset Value
0x00000F	DSP_BOOT_CONFIG	BOOT_MODE [3:0]	Depends on external implementation

3.10.4.1 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 (see Chapter 15, *Clock Generation and System Reset Management*).

There are five boot modes for the DSP subsystem. The MPU can select any of these boot modes by the MPU 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 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 3–13.

Table 3–13. 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, 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 upon DPS subsystem release from reset.

3.10.4.2 Boot Table Formats

For 16-bit boot download, the boot table in external memory must be in the format shown in Table 3–14.

Table 3–14. 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 should be zero to indicate 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
	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

Table 3–14. External Memory Boot Table for 16-Bit Boot Download (Continued)

Word Address (16-Bit Word)	Contents
	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 the boot table in external memory must be in the format shown in Table 3–15.

Table 3–15. 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 should be zero to indicate 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
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

Table 3-15. External Memory Boot Table for 32-Bit Boot Download (Continued)

Word Address (16-bit word)	Contents
	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

3.10.4.3 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 checks to see if the DSP subsystem in is SAM. If not, it keeps checking indefinitely. As soon as it is in SAM, it does the following initialization:

- ❏ Setup stack pointer (stack size is set to 0x20, default stack mode is fast dual stack)
- ❏ Disable interrupts globally
- ❏ Mask off interrupts
- ❏ Setup 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, it then boots the DSP subsystem.

Chapter 4

Memory Interface Traffic Controller

This chapter describes the OMAP5910 multimedia processor memory interface traffic controller (TC).

Topic **Page**

4.1 Introduction

The memory interface traffic controller (TC) manages all accesses by the MPU, the TMS320C55x DSP, the system DMA, and the local bus to the OMAP5910 system memory resources (SRAM, SDRAM, flash, ROM, etc.). The TC also manages accesses by the MPU or the USB host. The USB host is an internal OMAP5910 peripheral connected on the local bus, so the TC contributes in managing USB host accesses.

Figure 4–1 shows the OMAP5910 device with the traffic controller highlighted. Figure 4–2 shows the traffic controller in more detail. Table 4–1 lists the access modes and data access width of the controllers (MPU, C55x DSP, system DMA, and local bus).

Figure 4–1. TC Block Diagram

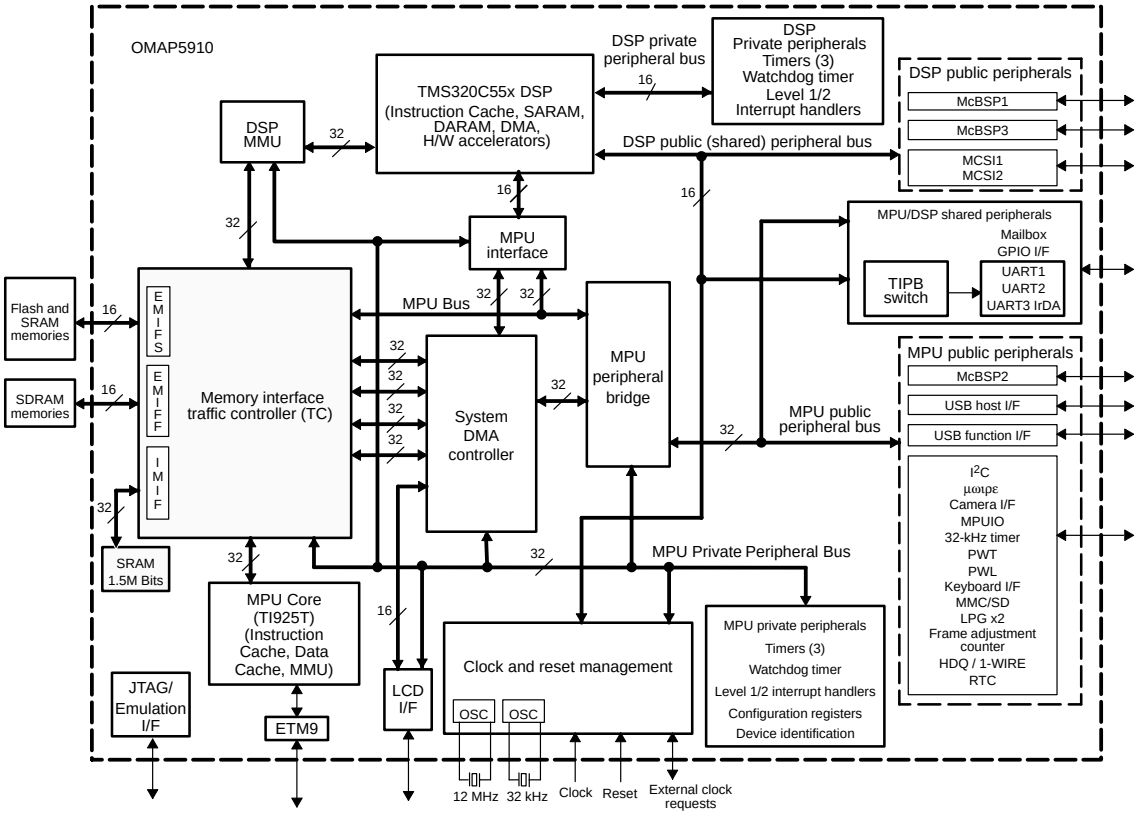


Figure 4–2. Traffic Controller

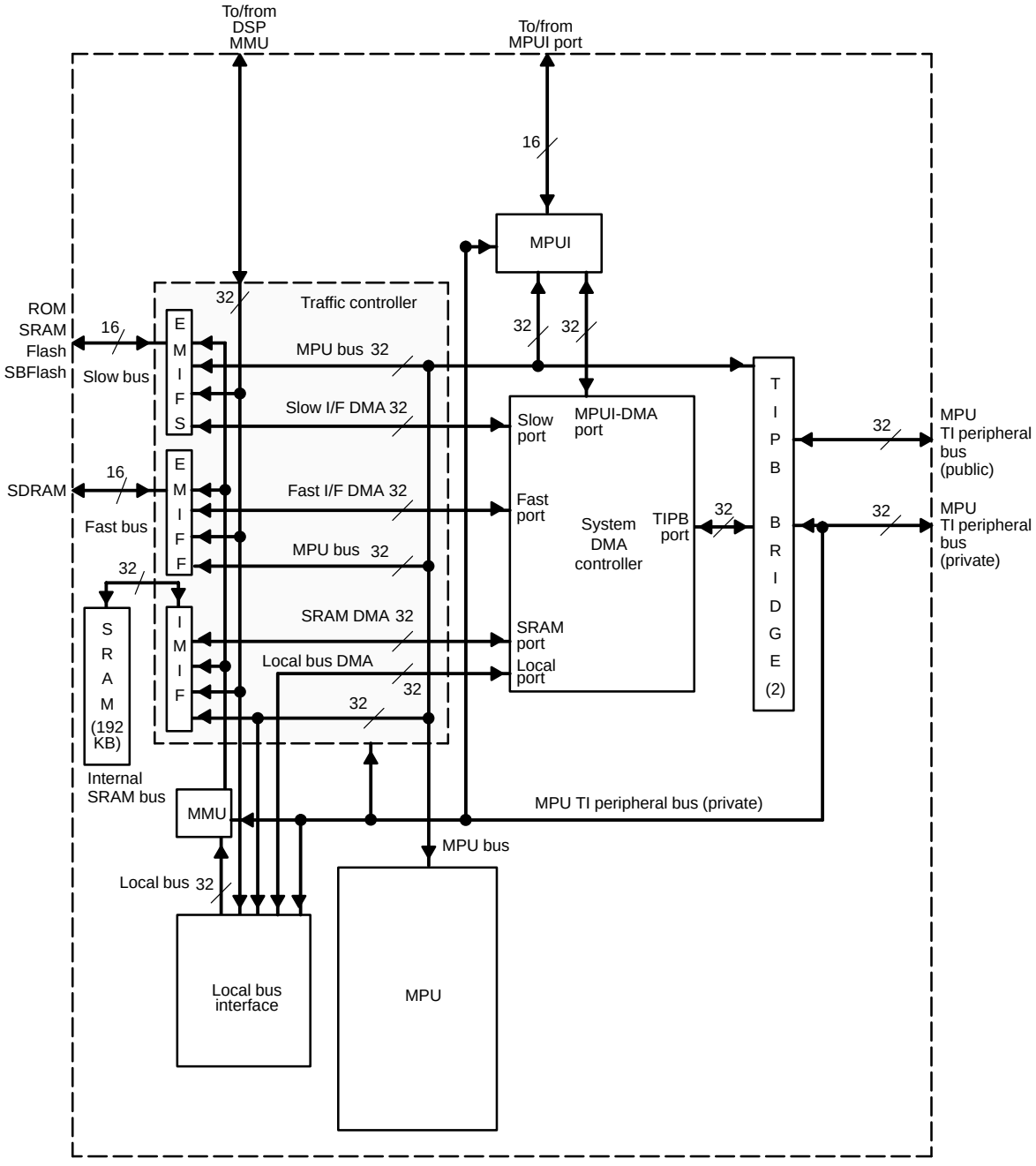


Table 4–1. Controller Access Mode and Data Access Width

Controllers	Single Access Mode Data Access Width (Bits)	Burst Access Mode Data Access Width (Bits)	Remarks
MPU	8, 16, 32	32	Single and burst access
C55x DSP	8, 16, 32	32	Single and burst access
System DMA controller	8, 16, 32	16, 32	Single and burst access; the 16-bit burst access is reserved for the LCD controller channel
Local bus	8, 16, 32	32	Single and burst access

The memories accessed by the TC are separated into two groups:

- External memory is memory that is not part of the OMAP5910 device. It can be SDRAM, flash, ROM, RAM, etc. External memory is accessed using the external memory interface (EMIF). The TC has two separate memory interfaces to access the external memories.
  - External memory interface fast (EMIFF): A fast synchronous interface for SDRAM
  - External memory interface slow (EMIFS): An asynchronous/synchronous interface to handle flash, ROM, RAM, etc.
- Internal memory is memory that is part of the OMAP5910 device and consists of 192K bytes of SRAM. The TC accesses the internal memory using an internal memory interface (IMIF) that is part of the TC.

Four hosts access the system resources using the TC.

- MPU: The MPU is connected to the TC via the MPU bus. The MPU can access memories that are connected to the IMIF, EMIFF, and EMIFS.
- C55x DSP: The C55x DSP is connected to the TC via the DSP MMU bus. The C55x DSP can access memories connected to IMIF, EMIFF, and EMIFS.
- System DMA: The system DMA is connected to the TC using four separate 32-bit buses, providing the system DMA controller with concurrent access to memories connected to the IMIF, EMIFF, and EMIFS.
- Internal local bus interface: An internal local bus interface is connected to the TC to allow access to memories connected to IMIF, EMIFF, and EMIFS. In OMAP5910, the USB host controller interfaces (and is master) to the local bus.

The TC provides each of the four hosts with:

- 32-bit single or burst access to memory (must be aligned with a[1-0] = 00)
- Size adaptation for 8-, 16-, or 32-bit words, with the requirement that address must be aligned on the correct bit boundary. For example, 32-bit access must be aligned on 32-bit boundary, 16-bit access must be aligned on 16-bit boundary, and so forth.
- Access duration management (wait state insertion) to enable the connection of slow memory devices
- Memory control signal generation (chip-select, memory-specific protocol generation)
- Single accesses for 8-bit or 16-bit words, except the TC supports 16-bit word bursts from the EMIFF or IMIF to the LCD controller

4.2 Memory Map

Four external chip-selects and a series of internal address decodes are provided for external and internal memories and for peripherals attached to the TI peripheral bus (see Table 4–2). CS0, CS1, CS2, and CS3 each has an address range of 32M bytes; the external SDRAM space has an address range of 64M bytes; the internal SRAM space has an address range of 512K bytes.

In the boot overlay mode, CS0 and CS3 are swapped so that the MPU can boot from a boot flash. Boot overlay mode is entered if pin MPU_BOOT is high during reset (the state on the MPU_BOOT signal can change after reset). The state of this pin is reflected in the BM bit field of the EMIF slow interface configuration register. For details, see Table 4–12, *EMIF Slow Interface Configuration Register (EMIFS_CONFIG_REG).*

Table 4–2. Device Types Associated With Chip-Select

CS	Device
CS0	External asynchronous RAM External asynchronous ROM or flash External synchronous burst flash
CS1	External asynchronous RAM External asynchronous ROM or flash External synchronous burst flash
CS2	External asynchronous RAM External asynchronous ROM or flash External synchronous burst flash
CS3	External asynchronous RAM External asynchronous ROM or flash External synchronous burst flash
None†	External synchronous dynamic RAM
None†	Internal SRAM

† The interface to these memory devices is activated via internal address decoding. There is no external chip select.

The OMAP5910 peripherals are mapped on the MPU memory space in two different segments: through STROBE0 (public peripherals) and STROBE1 (private peripherals). Each peripheral has a range of 2K bytes.

Table 4–3 shows the MPU memory map.

Table 4–3. MPU Memory Map

Device Name	Start Address	End Address	Size in Bytes	Data Access†
System Memory Address Space				
External Slow Memory Interface (Flash)				
FLASH CS0	0000:0000	01FF:FFFF	32M bytes	8/16/32 R/W
Reserved	0200:0000	03FF:FFFF		
FLASH CS1	0400:0000	05FF:FFFF	32M bytes	8/16/32 R/W
Reserved	0600:0000	07FF:FFFF		
FLASH CS2	0800:0000	09FF:FFFF	32M bytes	8/16/32 R/W
Reserved	0A00:0000	0BFF:FFFF		
FLASH CS3	0C00:0000	0DFF:FFFF	32M bytes	8/16/32 R/W
Reserved	0E00:0000	0FFF:FFFF		
External Fast Memory Interface (SDRAM)				
SDRAM	1000:0000	13FF:FFFF	64M bytes	8/16 R/W
Reserved	1400:0000	1FFF:FFFF		
Internal Memory Interface (SRAM)				
Internal RAM	2000:0000	2002:FFFF	192K bytes	8/16/32 R/W
Reserved	2003:0000	2FFF:FFFF		
DSP Processor Address Space				
DSP MPUI Interface				
MPUI Port RAM	E000:0000	E0FF:FFFF	16M bytes	16/32 R/W
MPUI DSP Peripherals I/O Space	E100:0000	E101:FFFF	128K bytes	16 R/W
DSP Private TIPB Peripherals (Strobe0)				
DSP TI peripheral bus	E100:0000	E100:07FF	2K bytes	16 R/W
Reserved	E100:0800	E100:7FFF	30K bytes	
DSP CLKM (clock control)	E100:8000	E100:87FF	2K bytes	16 R/W
Reserved	E100:8800	E100:8FFF	2K bytes	

† Each register must always be accessed using the appropriate data access width as indicated in this table. Failure to do so may result in unexpected behavior including a TIPB bus error condition with an associated interrupt. Reserved address locations should never be accessed.

Table 4–3. MPU Memory Map (Continued)

Device Name	Start Address	End Address	Size in Bytes	Data Access†
DSP Shared TIPB Peripherals (Strobe1)				
UART1	E101:0000	E101:07FF	2K bytes	8 R/W
UART2	E101:0800	E101:0FFF	2K bytes	8 R/W
Reserved	E101:1000	E101:17FF	2K bytes	
McBSP1	E101:1800	E101:1FFF	2K bytes	16 R/W
MCSI2	E101:2000	E101:27FF	2K bytes	16 R/W
MCSI1	E101:2800	E101:2FFF	2K bytes	16 R/W
Reserved	E101:3000	E101:6FFF	16K bytes	
McBSP3	E101:7000	E101:77FF	2K bytes	16 R/W
Reserved	E101:7800	E101:97FF	8K bytes	
UART3	E101:9800	E101:9FFF	2K bytes	8 R/W
Reserved	E101:A000	E101:DFFF	16K bytes	
GPIOs	E101:E000	E101:E7FF	2K bytes	16 R/W
Reserved	E101:E800	E101:EFFF	6K bytes	
MPU Address Space				
MPUI port interrupt, control and status registers	E102:0000	E102:0003	4 bytes	16 R/W
Reserved	E102:0004	EFFF:FFFF		
Reserved	F000:0000	FFFD:0000		
MPU Public TIPB Peripherals (Strobe 0)				
UART1	FFFB:0000	FFFB:07FF	2K bytes	8 R/W
UART2	FFFB:0800	FFFB:0FFF	2K bytes	8 R/W
McBSP2	FFFB:1000	FFFB:17FF	2K bytes	16 R/W
Reserved	FFFB:1800	FFFB:2FFF	6K bytes	
µWire	FFFB:3000	FFFB:37FF	2K bytes	16 R/W
I^2C	FFFB:3800	FFFB:3FFF	2K bytes	16 R/W

† Each register must always be accessed using the appropriate data access width as indicated in this table. Failure to do so may result in unexpected behavior including a TIPB bus error condition with an associated interrupt. Reserved address locations should never be accessed.

Table 4–3. MPU Memory Map (Continued)

Device Name	Start Address	End Address	Size in Bytes	Data Access†
MPU Public TIPB Peripherals (Strobe 0) (continued)				
USB function	FFFB:4000	FFFB:47FF	2K bytes	16 R/W
RTC	FFFB:4800	FFFB:4FFF	2K bytes	8 R/W
MPUIO	FFFB:5000	FFFB:57FF	2K bytes	16 R/W
PWL	FFFB:5800	FFFB:5FFF	2K bytes	8 R/W
PWT	FFFB:6000	FFFB:67FF	2K bytes	8 R/W
Camera IF	FFFB:6800	FFFB:6FFF	2K bytes	32 R/W
Reserved	FFFB:7000	FFFB:77FF	2K bytes	
MMC	FFFB:7800	FFFB:7FFF	2K bytes	16 R/W
Reserved	FFFB:8000	FFFB:8FFF	4K bytes	
32-kHz timer	FFFB:9000	FFFB:97FF	2K bytes	32 R/W
UART3	FFFB:9800	FFFB:9FFF	2K bytes	8 R/W
USB host	FFFB:A000	FFFB:A7FF	2K bytes	32 R/W
FAC	FFFB:A800	FFFB:AFFF	2K bytes	16 R/W
Reserved	FFFB:B000	FFFB:BFFF	4K bytes	
HDQ/1-Wire	FFFB:C000	FFFB:C7FF	2K bytes	8 R/W
TIPB switches	FFFB:C800	FFFB:CFFF	2K bytes	16 R/W
LED1	FFFB:D000	FFFB:D7FF	2K bytes	8 R/W
LED2	FFFB:D800	FFFB:DFFF	2K bytes	8 R/W
Reserved	FFFB:E000	FFFB:FFFF	8K bytes	
MPU Public TIPB Peripherals (Strobe 1)				
Reserved	FFFC:0000	FFFC:DFFF	56K bytes	
GPIOs	FFFC:E000	FFFC:E7FF	2K bytes	32 R/W
Reserved	FFFC:E800	FFFC:EFFF	2K bytes	
Mailbox	FFFC:F000	FFFC:F7FF	2K bytes	16 R/W

† Each register must always be accessed using the appropriate data access width as indicated in this table. Failure to do so may result in unexpected behavior including a TIPB bus error condition with an associated interrupt. Reserved address locations should never be accessed.

Table 4–3. MPU Memory Map (Continued)

Device Name	Start Address	End Address	Size in Bytes	Data Access†
Reserved	FFFC:F800	FFFC:FFFF	2K bytes	
MPU Private TIPB Peripherals (Strobe 0)				
Reserved	FFFD:0000	FFFD:FFFF	2K bytes	
MPU Private TIPB Peripherals (Strobe 1)				
MPU level 2 interrupt handler	FFFE:0000	FFFE:07FF	2K bytes	32 R/W
ULPD power management	FFFE:0800	FFFE:0FFF	2K bytes	16 R/W
OMAP5910 configuration	FFFE:1000	FFFE:17FF	2K bytes	32 R/W
Die ID	FFFE:1800	FFFE:1FFF	2K bytes	32 R/W
Reserved	FFFE:2000	FFFE:BFFF	40K bytes	
LCD controller	FFFE:C000	FFFE:C0FF	256 bytes	32 R/W
Local bus interface	FFFE:C100	FFFE:C1FF	256 bytes	32 R/W
Local bus MMU	FFFE:C200	FFFE:C2FF	256 bytes	32 R/W
Reserved	FFFE:C300	FFFE:C4FF	512 bytes	
MPU Timer 1	FFFE:C500	FFFE:C5FF	256 bytes	32 R/W
MPU Timer 2	FFFE:C600	FFFE:C6FF	256 bytes	32 R/W
MPU Timer 3	FFFE:C700	FFFE:C7FF	256 bytes	32 R/W
MPU watchdog timer	FFFE:C800	FFFE:C8FF	256 bytes	32 R/W
MPUI	FFFE:C900	FFFE:C9FF	256 bytes	32 R/W
MPU private TIPB bridge	FFFE:CA00	FFFE:CAFF	256 bytes	32 R/W
MPU level 1 interrupt handler	FFFE:CB00	FFFE:CBFF	256 bytes	32 R/W
Traffic controller	FFFE:CC00	FFFE:CCFF	256 bytes	32 R/W
Reserved	FFFE:CD00	FFFE:CDFF	256 bytes	
MPU CLKM (clock control)	FFFE:CE00	FFFE:CEFF	256 bytes	32 R/W
DPLL1	FFFE:CF00	FFFE:CFFF	256 bytes	32 R/W
Reserved	FFFE:D000	FFFE:D0FF	256 bytes	

† Each register must always be accessed using the appropriate data access width as indicated in this table. Failure to do so may result in unexpected behavior including a TIPB bus error condition with an associated interrupt. Reserved address locations should never be accessed.

Table 4–3. MPU Memory Map (Continued)

Device Name	Start Address	End Address	Size in Bytes	Data Access†
MPU Private TIPB Peripherals (Strobe 1) (Continued)				
Reserved	FFFE:D100	FFFE:D1FF	256 bytes	
DSP MMU	FFFE:D200	FFFE:D2FF	256 bytes	32 R/W
MPU public TIPB bridge	FFFE:D300	FFFE:D3FF	256 bytes	16 R/W
JTAG ID code	FFFE:D400	FFFE:D4FF	256 bytes	32 R/W
Reserved	FFFE:D500	FFFE:D7FF		
System DMA controller	FFFE:D800	FFFE:DFFF	2K bytes	16 R/W
Reserved	FFFE:E000	FFFE:FFFF	2K bytes each	

† Each register must always be accessed using the appropriate data access width as indicated in this table. Failure to do so may result in unexpected behavior including a TIPB bus error condition with an associated interrupt. Reserved address locations should never be accessed.

4.3 Memory Interfaces

The TC has three memory interfaces:

- Internal memory interface (IMIF)
- External memory interface slow (EMIFS)
- External memory interface fast (EMIFF)

4.3.1 Internal Memory Interface

The IMIF interfaces to an internal 192K-byte block of SRAM. The interface handles all single and burst requests from the MPU, the C55x DSP, the system DMA engine, and the local bus.

4.3.1.1 IMIF Priority Handler

This memory interface has two software-selectable priority algorithms for resolving simultaneous access requests: least recently used and dynamic priority. The priority scheme is shared with the EMIFS and EMIFF and is set in the OMAP5910 configuration registers (bit 20, LRU_SEL in FUNC_MUX_CTRL_0). See Chapter 6, *MPU Private Peripherals*, for details on configuration registers.

- Least recently used
  - A round-robin arbitration scheme. The highest priority requestor is the one that least recently accessed the memory.
- Dynamic priority
  - Dynamic priority uses high- and low-priority queues.
  - Each requestor, except the MPU, has a time-out register allocated to it (see *Time-Out Registers* in Section 4.4). These registers hold the number of clock cycles that a low-priority queue request must wait before it is moved from the low-priority queue to the high-priority queue.
  - At reset, all requestors are initially in the low-priority queue and the time-out registers are set to minimum value for each requestor. You must program these registers before using dynamic priority.
  - The low-priority queue order is:
    - MPU
    - DSP
    - Local bus
    - DMA (all channels including LCD)

- The high-priority queue order is:
  - DMA transfer involving LCD channel
  - DSP
  - Local bus
  - DMA transfer involving channels other than LCD channel

❑ Fixed priority is a special case of dynamic priority. To create a fixed priority, all time-out registers must have a value of 0. This way any request made goes into the high-priority queue after one clock cycle. Then the high-priority queue provides a fixed priority.

4.3.1.2 IMIF Operation

The 192K bytes of internal SRAM are selected by an internal chip select based on the appropriate address decode. The interface to the SRAM is 32 bits wide and provides support for single and burst accesses. The SRAM operates at the frequency of the traffic controller.

4.3.2 External Memory Interface Slow

The EMIFS interfaces with and handles all transactions to flash memory, ROM, asynchronous memories, and synchronous burst flash. The interface can drive up to four devices by assignment to one of four chip-selects. Each chip-select has a corresponding register to specify the protocol used for the associated external device.

Table 4–4 shows the EMIFS signal list.

Table 4–4. External Memory Interface Slow Signal List

Signal Name	I/O	Bus	Description
FLASH.RDY	I	–	Ready/busy signal from device
$\overline{\text{FLASH.WP}}$	O	–	Write protection
FLASH.CLK	I/O	–	Clock signal for flash device
$\overline{\text{FLASH.RP}}$	O	–	Flash power-down/reset
$\overline{\text{FLASH.CS0}}$	O	–	Active-low chip-select for device
$\overline{\text{FLASH.CS1}}$	O	–	Active-low chip-select for device
$\overline{\text{FLASH.CS2}}$†	O	–	Active-low chip-select for device
$\overline{\text{FLASH.CS3}}$	O	–	Active-low chip-select for device

† $\overline{\text{FLASH.CS2}}$ and $\overline{\text{FLASH.BAA}}$ are multiplexed on the same device pin. Pin function is selected using the OMAP5910 configuration register, FUNC_MUX_CRTL_0. The $\overline{\text{FLASH.CS2}}$ functionality is default.

Table 4–4. External Memory Interface Slow Signal List (Continued)

Signal Name	I/O	Bus	Description
$\overline{\text{FLASH.BAA}}$†	O	–	Active-low burst advance acknowledge for Advanced Micro Devices (AMD) burst flash
$\overline{\text{FLASH.OE}}$	O	–	Active-low output enable
$\overline{\text{FLASH.WE}}$	O	–	Active-low write enable
$\overline{\text{FLASH.ADV}}$	O	–	Active-low address valid
FLASH.D[15:0]	I/O	15–0	Flash data bus from external device
FLASH.A[24:1]	O	24–1	Flash data bus to external device
$\overline{\text{FLASH.BE}}$	O	3–0	External byte enable

† $\overline{\text{FLASH.CS2}}$ and $\overline{\text{FLASH.BAA}}$ are multiplexed on the same device pin. Pin function is selected using the OMAP5910 configuration register, FUNC_MUX_CRTL_0. The $\overline{\text{FLASH.CS2}}$ functionality is default.

Note:

OMAP5910 multiplexes the $\overline{\text{FLASH.CS2}}$ and $\overline{\text{FLASH.BAA}}$ pin functionality to the same device pin. Selecting the $\overline{\text{FLASH.BAA}}$ function to enable burst flash advance acknowledge disables $\overline{\text{FLASH.CS2}}$ functionality. In this case, capability of the EMIFS interface is reduced from a maximum of four external devices to a maximum of three external devices.

4.3.2.1 EMIFS Priority Handler

This memory interface has two software-selectable priority algorithms for resolving simultaneous access requests: least recently used and dynamic priority. The priority scheme is shared with the IMIF and EMIFF and is set in the OMAP5910 configuration registers (bit 20, LRU_SEL in FUNC_MUX_CTRL_0). See Chapter 6, *MPU Private Peripherals*, for details on configuration registers.

- Least recently used
  - A round-robin arbitration scheme. The highest priority requestor is the one that least recently accessed the memory.
- Dynamic priority
  - Dynamic priority uses high- and low-priority queues
  - Each requestor, except the MPU, has a time-out register allocated to it (see *Time-Out Registers* in Section 4.4). These registers hold the number of clock cycles that a low-priority queue request must wait before it is moved from the low priority queue to the high-priority queue.

- At reset, all requestors are initially in the low-priority queue and the time-out registers are set to minimum value for each requestor. You must program these registers before using dynamic priority.
- The low-priority queue order is:
  - MPU
  - DSP
  - Local bus
  - DMA (all channels including LCD)
- The high-priority queue order is:
  - DMA transfer involving LCD channel
  - DSP
  - Local bus
  - DMA transfer involving channels other than LCD channel

❑ Fixed priority is a special case of dynamic priority. To create a fixed priority, all time-out registers must have a value of 0. This way any request made goes into the high-priority queue after one clock cycle. Then the high-priority queue provides a fixed priority.

4.3.2.2 EMIFS Operation

This interface generates the appropriate signal timings to drive the following types of devices or compatible devices:

❑ Intel fast boot block flash (23FxxxF3)
❑ AMD simultaneous read/write boot sector flash (AM29DLxxxG)
❑ AMD burst mode flash (AM29BLxxxC)
❑ Intel StrataFlash memory (28FxxxJ3A)
❑ Intel synchronous StrataFlash memory (28FxxxK3/K18)
❑ Intel wireless flash memory (28FxxxW18)
❑ Asynchronous SRAM

Every macroscopic flash command (read array, program, clear status register) is sent to the flash memory controller by the MPU. The MPU writes in the flash, followed by a read or a write, to set up the flash in the correct mode.

File/boot block flash basic operations supported are:

❑ Asynchronous read, including specific reads like manufacturer ID
❑ Burst read emulation (by multiple asynchronous reads) in 32-bit width
❑ Reset or power down
❑ Asynchronous write with $\overline{WE}$ in 16-bit width

The following operations are also supported for burst flash devices:

- ❏ Synchronous burst read mode (for Intel and AMD flashes)

An additional read mode is provided that supports burst read on page mode ROM devices.

Figure 4–3 through Figure 4–7 show the external timing of the protocols used by the EMIF slow interface.

4.3.2.3 Device Initialization

Depending on the flash memory or RAM device associated with each chip-select, the EMIFS interface must be initialized. If the device used is a flash, the flash may have to be initialized in the correct protocol to achieve maximum performance.

To use the external flash device with the synchronous flash burst protocol, the following configuration must be set in the flash device and in the EMIFS chip-select configuration registers (see Table 4–13, *EMIF Slow Chip-Select Configuration Registers*):

- ❏ Read mode
- ❏ Frequency configuration
- ❏ Data output configuration
- ❏ Burst order. The EMIF only supports linear burst order.
- ❏ Burst length
- ❏ CLK configuration
- ❏ Flash mode operation. Some flash modules use multiple signals for burst operations (see Section 4.3.2.7, *Burst Read Operation*, for more information).

After reset, each of the EMIF slow chip-select configuration registers is configured in asynchronous mode, with 15 wait cycles and a clock divider of 6 (relative to the traffic controller clock). This configuration ensures maximum compatibility with many existing devices.

4.3.2.4 EMIFS Memory Timing Control

In both asynchronous and synchronous modes all EMIFS-to-memory control signals are referenced to an internal EMIFS reference clock. The internal EMIFS reference clock is divided from the TC clock by a programmable value in the FCLKDIV bit field of the EMIFS chip select configuration register (EMIFS_CSx_CONFIG). This allows the EMIFS to accommodate timing constraints of slow devices, even with high system clock rate. Table 4–5 shows valid FCLKDIV settings and resulting EMIFS reference clock values.

Table 4–5. FCLKDIV Settings and Resulting EMIFS Reference Clock

FCLKDIV	EMIFS Reference
00	TC clock/1
01	TC clock/2
10	TC clock/4
11	TC clock/6

Depending on the chip-select mode configuration, the EMIFS reference clock can be output at the FLASH.CLK output pin. In asynchronous read and write modes, EMIFS reference clock is not output and the FLASH.CLK pin remains low. In synchronous modes, EMIFS reference clock is present at the FLASH.CLK device pin.

In synchronous modes a selectable retiming feature enables read data to be latched by a delayed EMIFS reference clock. The retiming feature accounts for delays through the OMAP5910 input/output pins by feeding back FLASH.CLK to offer optimum data and clock alignment. You can select the re-timing mode using the RT bit in the EMIFS chip-select configuration registers.

4.3.2.5 Asynchronous Read Operation

Asynchronous read mode is selected by programming the RDMODE bit field to 000 in the corresponding EMIF slow chip-select configuration register. This is the default mode at reset.

The following characteristics describe asynchronous read mode operation.

- The chip-select pulse width depends on the RDWST bit field of the EMIFS chip-select configuration register. Pulse width equals:

  (RDWST + 2) x EMIFS_Ref (shown as N cycles in Figure 4–3)

  Chip-select minimum pulse width is (2 x EMIFS_Ref).

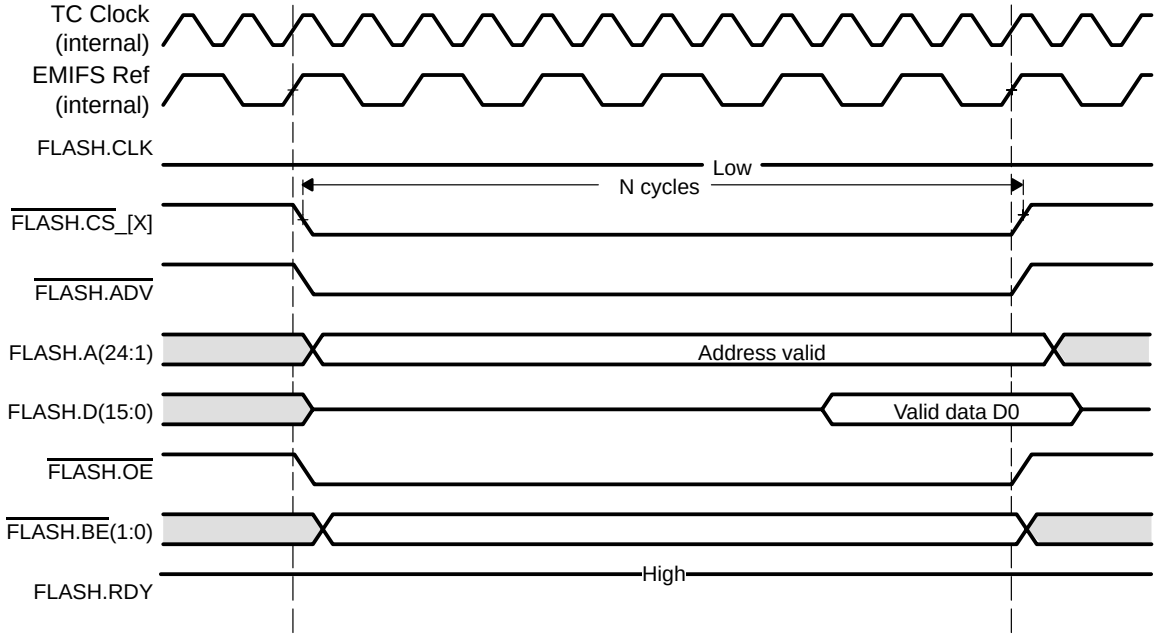
- Address drive time follows $\overline{\text{FLASH.CS}}$_[X] activation (no setup time guaranty). The $\overline{\text{FLASH.ADV}}$ output is asserted with the address for use with Intel and AMD burst flash protocols.

- Read data is latched on the same TC clock rising edge that deactivates the $\overline{\text{FLASH.OE}}$ signal.

- In asynchronous mode, the internal EMIFS reference clock is not provided outside the EMIFS. The FLASH.CLK signal remains low.

- Figure 4–3 shows typical timing for an asynchronous 16-bit read operation on a 16-bit width device with RDWST = 4, FCLKDIV = 01.

Figure 4–3. Asynchronous 16-Bit Read Operation on a 16-Bit Width Device



4.3.2.6 Asynchronous Page Mode Read Operation

The asynchronous read operation (page mode) is similar to the asynchronous read, except that the number of wait states is different between the first access and the subsequent accesses within the page.

This mode of operation is selected by programming the following fields of the EMIF slow chip-select configuration registers (see Table 4–13, *EMIF Slow Chip-Select Configuration Registers*).

- RDMODE selects the memory type and number of words per page for page mode devices; supported values for words per page are 4, 8, or 16.
- RDWST sets the delay to insert prior to latching the first data word read from a page (range 0-15). The resulting delay is equal to (RDWST+2) x EMIFS_ref. This is represented by N cycles in Figure 4–4 and Figure 4–5. When crossing a page boundary, as in Figure 4–5, the RDWST parameter is used again for the first access on the new page.
- PGWST sets the delay between subsequent words in the page (range 0-15). The resulting delay is equal to (PGWST+1) x EMIFS_ref. This is represented by P cycles in Figure 4–4 and Figure 4–5.
- BW defines the word length of the access, which is equal to the memory data bus width.

As in asynchronous mode, device interface signals are referenced to the internal EMIFS reference clock, which is divided from the TC clock using FCLKDIV in the EMIF slow interface configuration register. The FLASH.CLK signal is not externally driven in asynchronous page operating mode.

Figure 4–4 shows typical timing for an asynchronous page mode 8x16-bit read operation on a 16-bit width device with RDWST = 2, PGWST = 0, FCLKDIV = 01, and RDMODE = 2.

Figure 4–5 shows typical timing for an asynchronous page mode 8x16-bit read with page crossing on a 16-bit width device with RDWST = 2, PGWST = 0, FCLKDIV = 01, and RDMODE = 2.

Figure 4–4. Asynchronous Page Mode 8x16-Bit Read Operation on a 16-Bit Width Device (8 Words per Page)

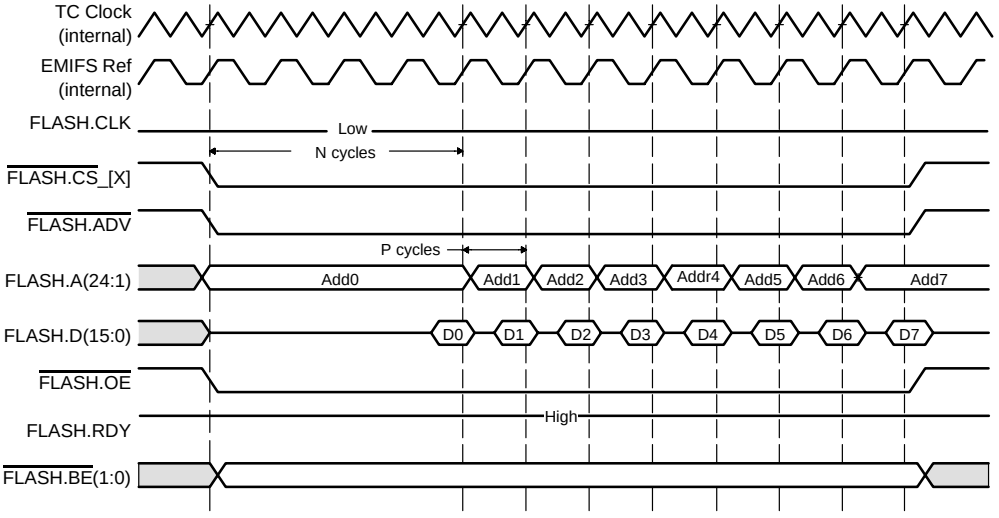
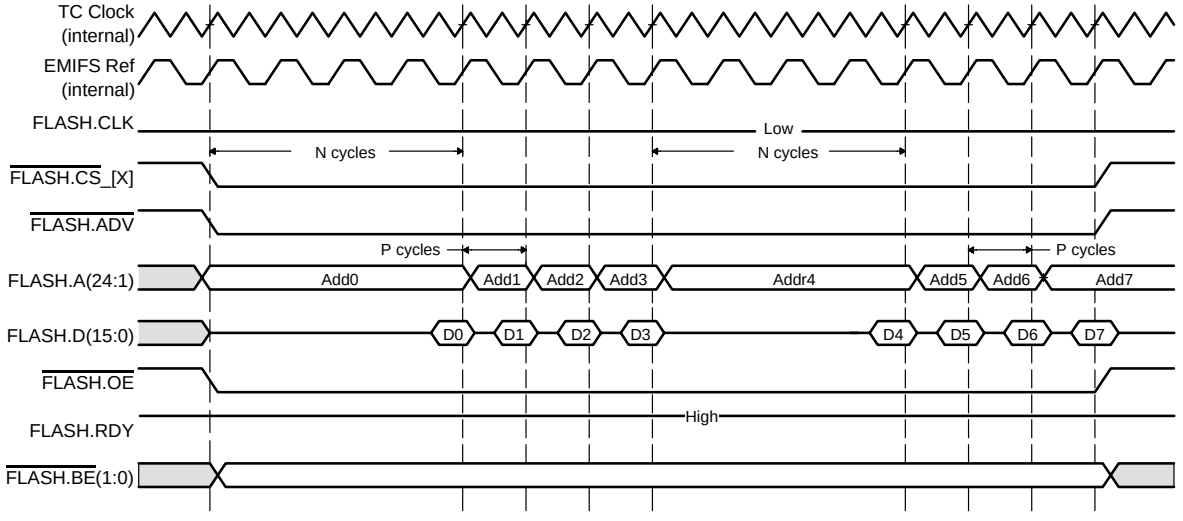


Figure 4–5. Asynchronous Page Mode 8x16-Bit Read With Page Crossing on 16-Bit Width Device (4 Words per Page)



4.3.2.7 Burst Read Operation

The synchronous read mode is selected for each device by setting the RDMODE configuration bit field to 100.

In this mode of operation, FLASH.CLK is driven on the OMAP5910 device pin.

Both AMD burst flash and Inter burst flash have three modes of operation:

- Asynchronous single read mode (device startup mode)
- Synchronous single read or burst read mode
- Asynchronous write

Asynchronous single read mode and asynchronous write modes are compatible with operation described in Section 4.3.2.5, *Asynchronous Read Operation*, and Section 4.3.2.8, *Asynchronous Write With WE Operation*.

Figure 4–6 shows the timing view of synchronous burst read mode operation.

On the AMD device, $\overline{\text{LBA}}$ is directly connected to the $\overline{\text{FLASH.ADV}}$ OMAP5910 pin.

The address is latched on the rising edge of $\overline{\text{FLASH.ADV}}$ with a specified hold time of 3 ns. This is easily met by maintaining the address during two cycles.

Data output of the device is stable on the rising edge of FLASH.CLK (specified with a setup and hold time referenced to this edge).

Two configuration registers are used in this operating mode:

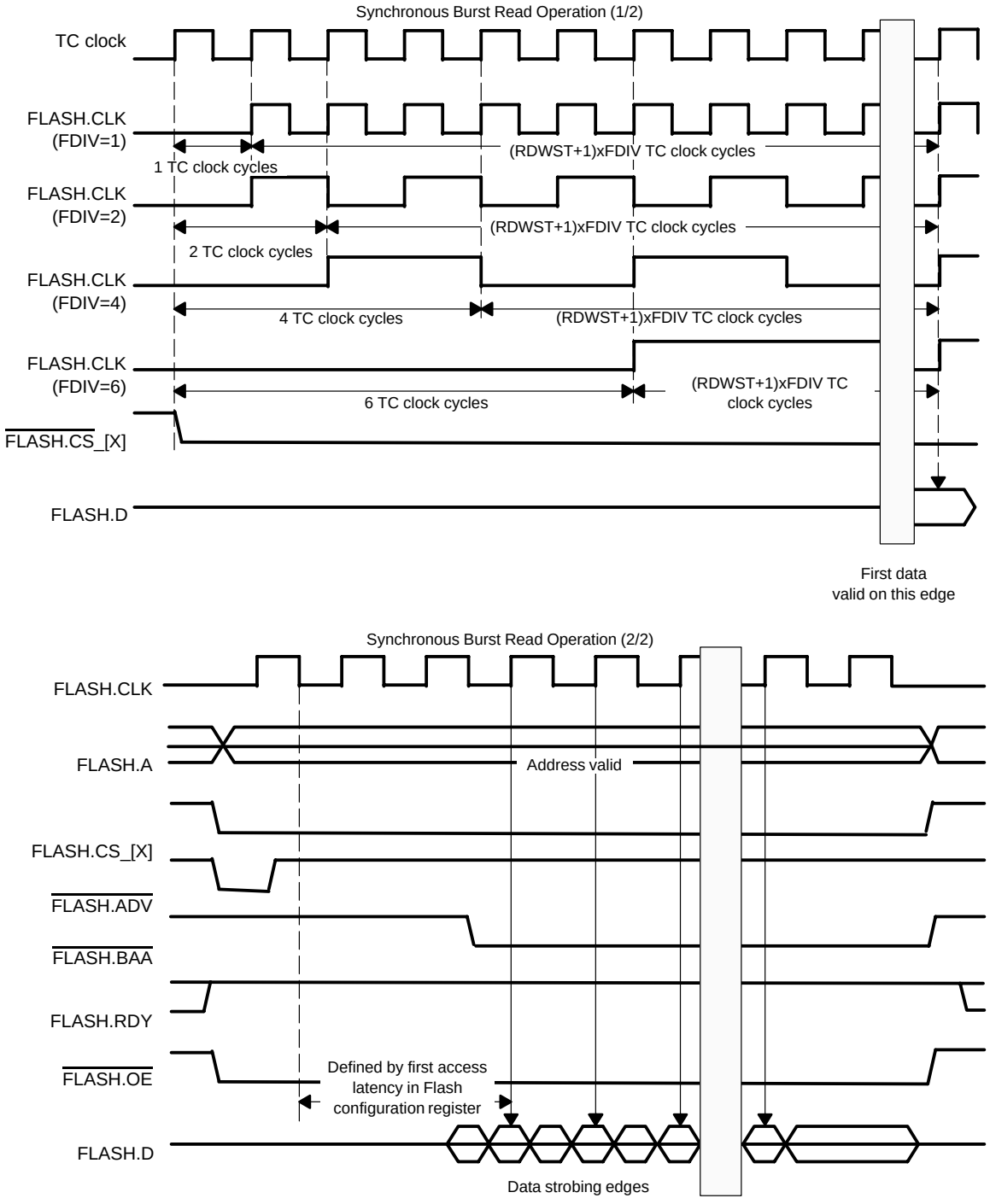
- FCLKDIV. Specifies the frequency ratio between the TC clock and FLASH.CLK (see Table 4–13, *EMIF Slow Chip-Select Configuration Registers*).
- RDWST. Specifies the number of FLASH.CLK cycles between the falling edge of $\overline{\text{FLASH.ADV}}$ and the edge at which first data is valid (see Table 4–13, *EMIF Slow Chip-Select Configuration Registers*).

The FLASH.RDY signal is not used in this mode: however, it is used during flash program and erase operations.

Note: Intel Burst Flash Operation

For proper operation in applications that combine OMAP5910 with Intel burst flash (examples include Intel 28FxxxK3, 28FxxxK18, and 28FxxxW18), the flash WAIT signal must not be connected to the OMAP5910 FLASH.RDY input. Instead, the FLASH.RDY input pin in the OMAP5910 must be tied active high through a pullup resistor. The OMAP5910 traffic controller properly handles all accesses across Intel burst flash boundaries (where WAIT could be asserted) without any input from WAIT, and without performance penalty.

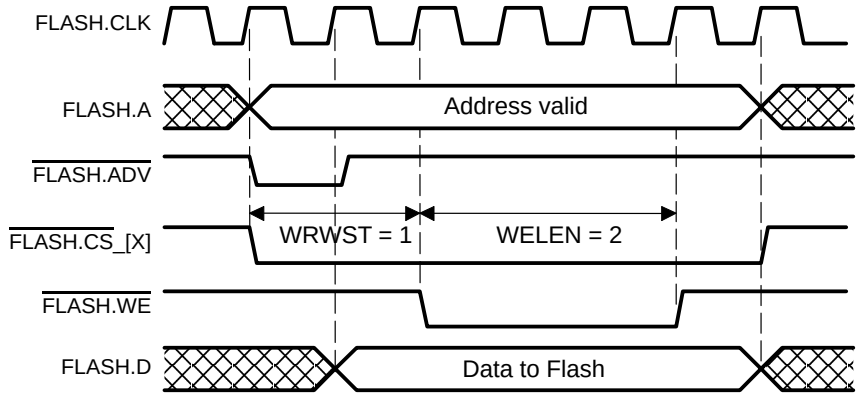
Figure 4–6. Synchronous Burst Read With Page Alignment



4.3.2.8 Asynchronous Write With WE Operation

The asynchronous write is used for both file flash and burst flash devices. Figure 4–7 shows the timing diagram. Burst write operation is not supported.

Figure 4–7. Asynchronous Write With WE Operation



The flash device latches the data on the rising edge of $\overline{\text{FLASH.WE}}$; data hold time specified is 0 ns minimum, which can be ensured by one TC clock cycle. The $\overline{\text{FLASH.WE}}$ low pulse duration is programmable for each device through the WELEN field in the flash configuration register. The number of wait states between write operations is programmable for each device through the WRWST field in the configuration register.

The duration from falling $\overline{\text{FLASH.CS}}$ to falling $\overline{\text{FLASH.WE}}$ (shown in Figure 4–7) is equal to the programmed value of WRWST + 1, and the duration for which $\overline{\text{FLASH.WE}}$ is asserted active low is equal to the programmed value of WELEN + 1.

The FLASH.CLK signal is not driven externally (maintained low) in the asynchronous write mode.

4.3.2.9 EMIFS Not-Ready Functionality

The EMIFS interface includes a feature that allows an external device to assert a not-ready signal via the OMAP5910 FLASH.RDY pin. Two modes of not-ready are available: classic and dynamic. In either mode, if FLASH.RDY is low, the external device is not ready.

Classic not-ready is supported for all EMIFS read modes and all access types (read or write, single or burst). In classic not-ready mode, FLASH.RDY is sampled synchronously at every rising TC clock edge. If FLASH.RDY is active low, then the EMIFS interface is frozen in its current state. If there is an access in progress, then the current state of that access is extended until the FLASH.RDY signal goes inactive high. All information regarding the current access is maintained (wait states, burst count, etc).

Dynamic not-ready is also supported for all accesses through EMIFS. In this mode, FLASH.RDY is sampled synchronously to TC clock, but only after any ongoing access (single or burst) is completed. If FLASH.RDY is sampled active low, the EMIFS interface is placed in a hold state and remains in this state until FLASH.RDY is asserted high by the external device. In dynamic not-ready mode, FLASH.RDY is used by the external device to indicate to OMAP5910 that it will be going inactive after the current access completes.

The programming of FLASH.RDY modes is described in Table 4–27, *EMIF Slow Wait State Configuration.*

4.3.2.10 EMIFS Dual-Port RAM Interface Mode

The OMAP5910 EMIFS includes a programmable mode associated with the $\overline{\text{FLASH.CS2}}$ chip select pin to support external devices that require a valid flash address before chip select is active. An example of such a device is a dual port RAM (DPRAM). When DPRAM mode is enabled, the low transition of $\overline{\text{FLASH.CS2}}$ is delayed to ensure that address is valid. The low to high transition of $\overline{\text{FLASH.CS2}}$ is not changed regardless of the mode setting. EMIFS DPRAM mode is programmed in the OMAP5910 configuration registers using bit 22, CONF_MOD_DPRAM_ENABLE_R, in register MOD_CONF_CTRL_0. See Chapter 6, *MPU Private Peripherals*, for details on configuration registers.

DPRAM interface mode is only applicable to EMIFS chip-select $\overline{\text{FLASH.CS2}}$. Also note that the $\overline{\text{FLASH.CS2}}$ pin multiplexes the $\overline{\text{FLASH.BAA}}$ function (see Table 4–4), and this $\overline{\text{FLASH.CS2}}$/$\overline{\text{FLASH.BAA}}$ multiplexing has highest priority. To activate the DPRAM interface mode, you must first ensure that OMAP5910 pin multiplexing has $\overline{\text{FLASH.CS2}}$ selected, then program for DPRAM interface configuration as described above. The DPRAM interface

mode is recommended only for 16-bit accesses since the OMAP5910 EMIFS keeps chip-select and write enable active between the two accesses generated by one 32-bit access to EMIFS. While nothing prevents the use of 32-bit accesses in DPRAM interface mode, avoid it if address must be known valid while write enable is active.

4.3.3 External Memory Interface Fast

The EMIFF can interface with synchronous DRAM (SDRAM). The interface directs all the transactions to the SDRAM device. The bus width is 16 bits.

Table 4–6 shows the EMIFF signal list.

Table 4–6. External Memory Interface Fast Signal List

Signal Name	I/O	Bus	Description
SDRAM.A[12:0]	O	12-0	SDRAM address bus
SDRAM.D[15:0]	I/O	15-0	Data from SDRAM
SDRAM.CLK	I/O	–	Clock to SDRAM
SDRAM.BA[1:0]	O	1-0	SDRAM bank select
SDRAM.CKE	O	–	SDRAM clock enable
$\overline{\text{SDRAM.RAS}}$	O	–	SDRAM RAS
$\overline{\text{SDRAM.CAS}}$	O	–	SDRAM CAS
$\overline{\text{SDRAM.WE}}$	O	–	SDRAM write enable
$\overline{\text{SDRAM.DQML}}$	O	–	Lower byte 3-state
$\overline{\text{SDRAM.DQMU}}$	O	–	Upper byte 3-state

4.3.3.1 EMIFF Priority Handler

This memory interface has two software-selectable priority algorithms for resolving simultaneous access requests: least recently used and dynamic priority. The priority scheme is shared with the EMIFS and IMIF and is set in the OMAP5910 configuration registers (bit 20, LRU_SEL in FUNC_MUX_CTRL_0). See Chapter 6, *MPU Private Peripherals*, for details on configuration registers.

- Least recently used
  - A round-robin arbitration scheme. The highest priority requestor is the one that least recently accessed the memory.

- Dynamic priority
  - Dynamic priority uses high- and low-priority queues.
  - Each requestor, except the MPU, has a time-out register allocated to it (see *Time-Out Registers* in Section 4.4). These registers hold the number of clock cycles that a low-priority queue request has to wait before it is moved from the low-priority queue to the high-priority queue.
  - At reset, all requestors are initially in the low-priority queue and the time-out registers are set to minimum value for each requestor. You must program these registers before using dynamic priority.
  - The low-priority queue order is:
    - MPU
    - DSP
    - Local bus
    - DMA (all channels including LCD)
  - The high-priority queue order is:
    - DMA transfer involving LCD channel
    - DSP
    - Local bus
    - DMA transfer involving channels other than LCD channel
- Fixed priority is a special case of dynamic priority. To create a fixed priority, all time-out registers must have a value of 0. This way any request made goes into the high-priority queue after one clock cycle. Then the high-priority queue provides a fixed priority.

4.3.3.2 EMIFF Operation

The EMIFF controller can support up to two devices for up to 64M bytes of memory. The following devices are supported:

- 256M-bit, 128M-bit, 64M-bit
- 2 or 4 banks for 64M-byte device
- x8 or x16 data bus configurations

Table 4–7 shows the possible SDRAM configurations.

Table 4–7. Possible SDRAM Configurations

Memory Size (Bytes)	Bus Size	Number of Devices	Type of Device
64M	2 x 8	2	256M bytes organized in 32M x 8
32M	1 x 16	1	256M bytes organized in 16M x 16
	2 x 8	2	128M bytes organized in 16M x 8
16M	1 x 16	1	128M bytes organized in 8M x 16
	2 x 8	2	64M bytes organized in 8M x 8
8M	1 x 16	1	64M bytes organized in 4M x 16
4M	2 x 8	2	16M bytes organized in 2M x 8
2M	1 x 16	1	16M bytes organized in 1M x 16

The burst length on the SDRAM is variable from 1-8 words and can be 32 words in the case of the LCD controller. The burst length is controlled by the SDRAM controller depending on how quickly EMIFF requests are received from the various initiators within the OMAP system (MPU, DSP subsystem, system DMA, local bus). The actual burst length is not controlled with the burst size of the SDRAM MRS configuration register, which must always be set with continuous burst. Depending on the system loading within the device and the specific configurations and interactions between the different initiators, burst transfers may or may not be achieved on the EMIFF by any specific initiator. However, the SDRAM request management logic within the SDRAM controller allows only bursts of 8 words, except for the LCD refresh channel which can achieve a burst of 32 words.

The SDRAM controller supports:

- The self-refresh mode (idle) and autorefresh (normal operation)
- Automatic generation of MRS and EMRS commands to the SDRAM by writing to a mirror configuration register within the OMAP5910 device
- Burst sizes of 1x8, 1x16, 1x32, and 4x32 for all accesses and 8x16 burst access for LCD.
- Burst across page boundary (local address increment coupled with current address register)
- Two pipelined levels of request from the SDRAM request manager to enable page interleave timing and reduce overhead cycles by the burst interruption mechanism

4.3.3.3 SDRAM Mode and Extended Mode Register Initialization

To make SDRAM memory accessible, its internal mode register must first be configured. The MRS register contains the protocol information used to communicate with the OMAP5910 device (burst size, latency, write burst, etc.). The EMRS register enables certain low-power characteristics for the SDRAM.

- Writing to the EMIF fast interface SDRAM MRS register (EMIFF_MRS) automatically forces the generation of an MRS command on the pins of the SDRAM interface. When the command is issued, the content of the OMAP5910 MRS register is placed on the SDRAM address bus and latched by the SDRAM into its internal MRS register.
- OMAP5910 uses the same EMIF fast interface SDRAM MRS register, combined with a control bit setting, to write EMRS commands to the SDRAM. When the CONF_MOD_EMRS_CTRL bit field in the MOD_CONF_CTRL_0 register is set, the OMAP configures SDRAM banks to write out the EMIFF_MRS register as EMRS commands instead of MRS commands.
- Reading from the EMIF fast interface SDRAM MRS register does not generate any external transactions.

Note:

The SDRAM requires 100 µs to stabilize after power up. Software is responsible for performing the initial setup of SDRAM. For more information see Table 4–20, *EMIF Fast Interface SDRAM MRS Register*.

4.3.3.4 SDRAM Autorefresh Initialization

To increase SDRAM bus availability, it is preferable to subdivide the SDRAM into smaller sections and then evenly distribute the refresh of each of these subsections instead of performing a single autorefresh for the entire SDRAM. The OMAP5910 device can support subdividing the autorefresh of the SDRAM into bursts of 1, 4, or 8 rows. It is recommended to set this parameter to 8 rows.

A 16-bit timer is used to track the interval between autorefresh burst requests to the SDRAM. An autorefresh request is issued when the timer reaches a user-defined value based on the following parameters:

- SDRAM frequency
- Refresh rate
- Number of SDRAM rows

The following formula is used to determine the refresh counter value that must be programmed in the EMIF fast interface configuration register 1 (EMIFF_SDRAM_CONFIG):

$$\text{Counter Value} = \frac{\left(\frac{\text{SDRAM refresh rate}}{T_f}\right) - 400}{\text{Number of SDRAM Rows}}$$

where T_f = (1 / traffic controller frequency) and the 400 cycles take into account the worst-case priority scenario where the SDRAM refresh is at the bottom of the priority queue.

Example: 64-ms refresh rate, 100-MHz traffic controller frequency, 4096 rows to be refreshed:

T_f = 10 ns

$$\text{Counter Value} = \frac{\left(\frac{64000000\ \text{ns}}{10\ \text{ns}}\right) - 400}{4096} = 1562\ \text{cycles}$$

This ensures a 64-millisecond refresh period for the full SDRAM. In the event that the downcounter does not equal zero by the time a new autorefresh burst request occurs, the new request is memorized and is done during the current refresh burst.

4.3.3.5 SDRAM Self-Refresh Protection

The traffic controller idle mode is entered after an internal request and acknowledge protocol with the OMAP5910 clock generator and system reset module. In idle mode, the traffic controller clock is stopped. If the clock remains idle for more than 64 milliseconds and the SDRAM was not entered into self refresh mode, SDRAM data corruption results. Setting the RFRSH_STBY bit in the EMIF fast interface SDRAM configuration register 2 (EMIFF_SDRAM_CONFIG_2) avoids SDRAM data corruption by automatically placing the SDRAM in self-refresh mode prior to the traffic controller entering idle mode.

A similar SDRAM data corruption can occur in the event of a warm global system reset from external device pin. Since the reset event is likely to extend beyond 64 milliseconds, and the SDRAM controller does not autorefresh during reset, data is corrupted. Setting the RFRSH_RST bit in the EMIF fast interface SDRAM configuration register 2 (EMIFF_SDRAM_CONFIG_2) avoids SDRAM data corruption for this case by automatically placing the SDRAM in self-refresh mode prior to warm reset being applied to the traffic controller. The SDRAM controller continues in self-refresh mode until the reset is unasserted. Note that RFRSH_RST applies only in the case of warm reset. For cold reset, SDRAM is not set to self-refresh regardless of the state of RFRSH_RST.

Caution: Self-Refresh Mode

When the EMIFF SDRAM is in self-refresh mode, the EMIFF does not respond to TIPB requests including MRS writes. To respond, the SLFR bit must be cleared by firmware. Writes to TC registers which would normally cause EMIFF to perform an action have no effect while EMIFF is in self-refresh mode. If an MRS write is attempted while EMIFF is in self-refresh mode, there is a pending MRS request. This prevents the traffic controller from idling and therefore prevents the device from entering deep sleep mode. The MRS request is not serviced until the SLFR bit is cleared.

4.3.3.6 SDRAM Clock Disable

The EMIF fast SDRAM clock signal (SDRAM.CLK) is disabled using these steps:

1) Set the PDE bit field of the EMIF slow interface configuration register.

2) Set one (or both) of the following bit fields in the EMIF fast SDRAM configuration register 1

   a) Set the SLRF to place the SDRAM into self-refresh mode
   b) Set the PWD to place the SDRAM into power-down mode

3) Set the CLK bit field of the EMIF fast interface SDRAM configuration register 1 to stop the clock

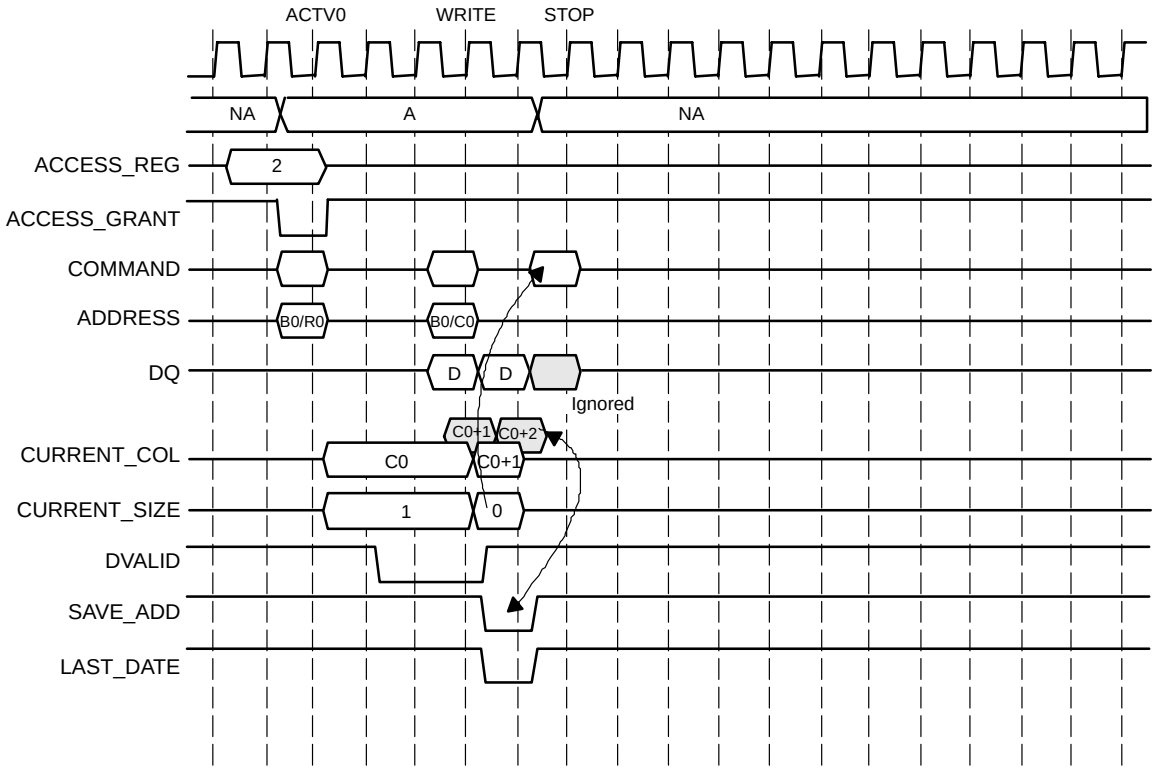
4.3.3.7 Endianism Conversion Control

The traffic controller registers include a register to control endianism conversion in the DSP memory management unit. For details, see Table 4–25, *Endianism Register (ENDIANISM)*.

4.3.3.8 SDRAM Access Timing Diagrams

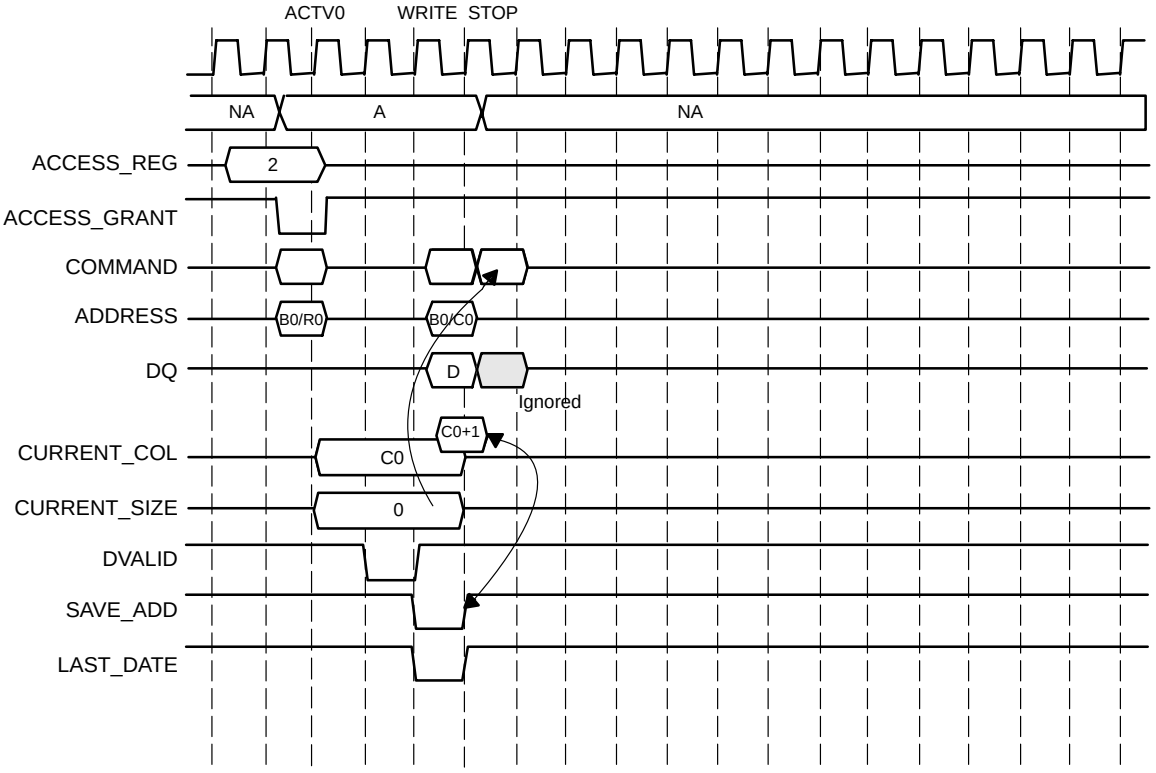
Figure 4–8 through Figure 4–18 show the SDRAM timing diagrams. Burst accesses shown here might not be achievable by all initiators of EMIFF transactions. See Section 4.3.3.2 for more detail on bursting behavior.

Figure 4–8. SDRAM Write Single 32-Bit Word With Burst Stop



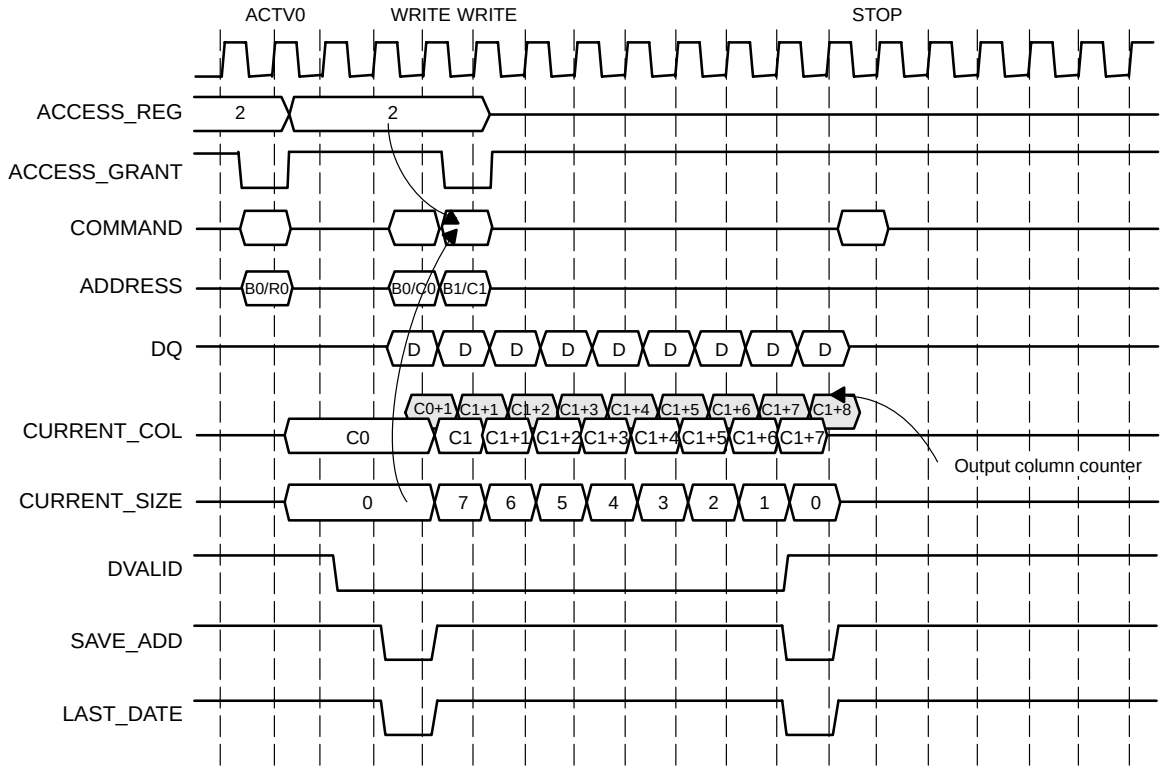
Note: WRITE (burst reduced to 2) is interrupted by a STOP command because no new request is pending.

Figure 4–9. SDRAM Write Single 16-Bit Half-Word With Burst Stop



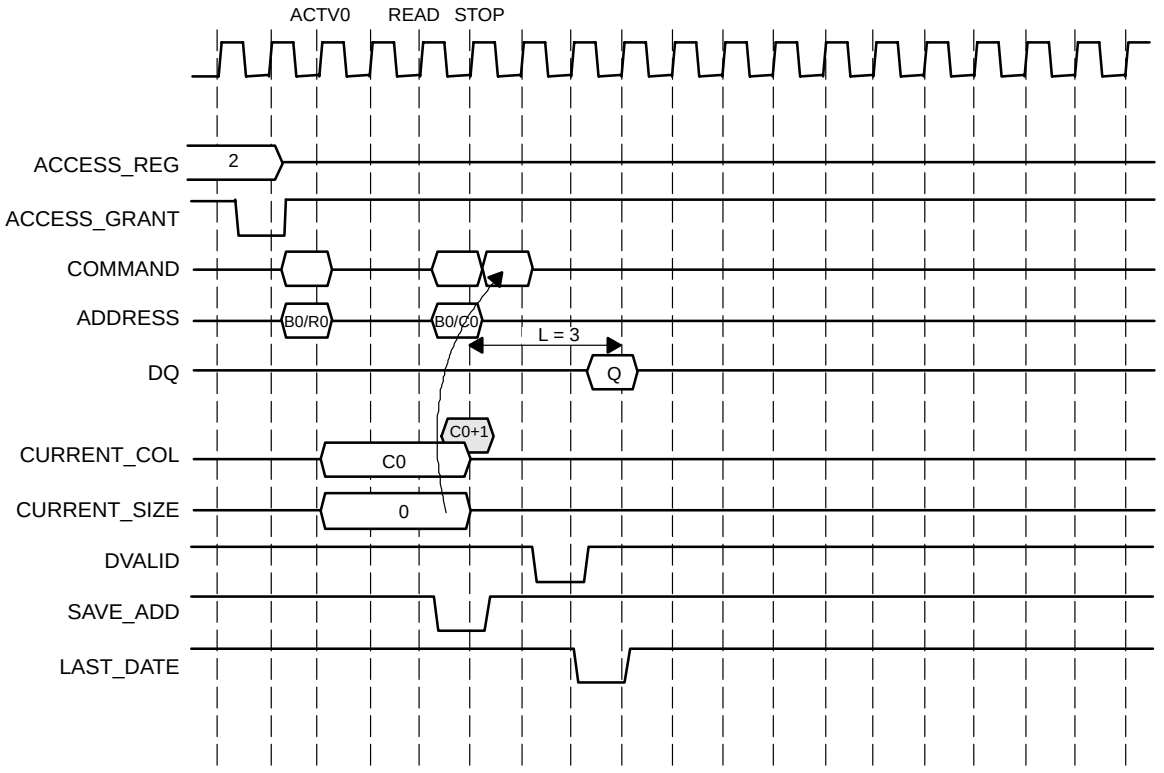
Note: WRITE (burst reduced to 1) is interrupted by a STOP command because no new request is pending.

Figure 4–10. SDRAM Write Single 16-Bit Half-Word Followed by Write Burst 8



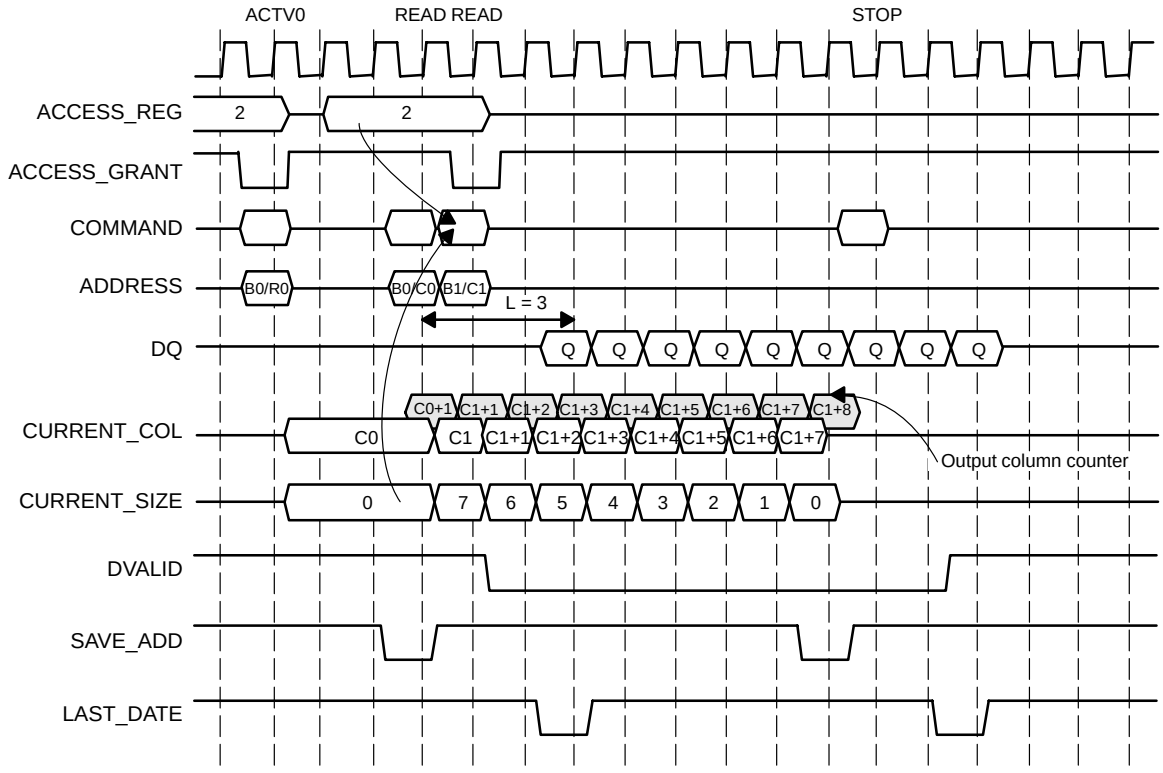
Note: WRITE (burst reduced to 1) is followed by a WRITE (8) in a different bank and in a page already active.

Figure 4–11. SDRAM Read Single 16-Bit Half-Word With Burst Stop



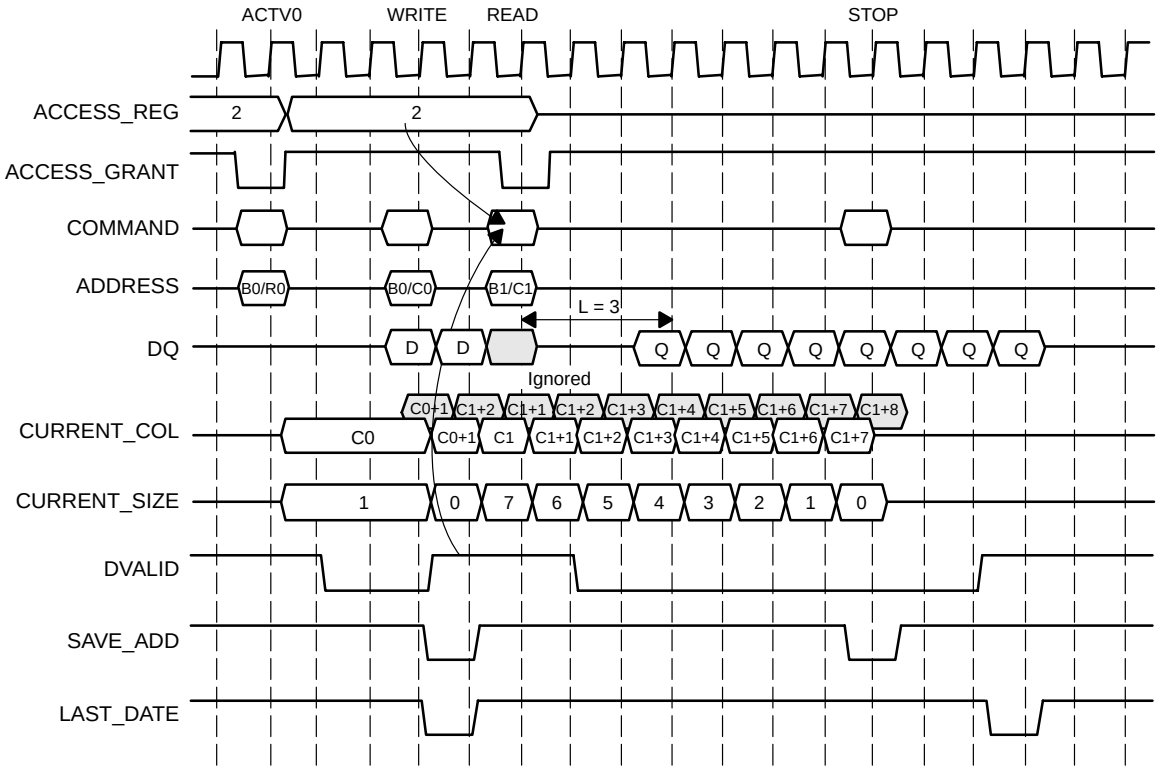
Note: READ (burst reduced to 1) is interrupted by a STOP command because no new request is pending.

Figure 4–12. SDRAM Read Single 16-Bit Half-Word Followed by Read Burst 8 Half-Word



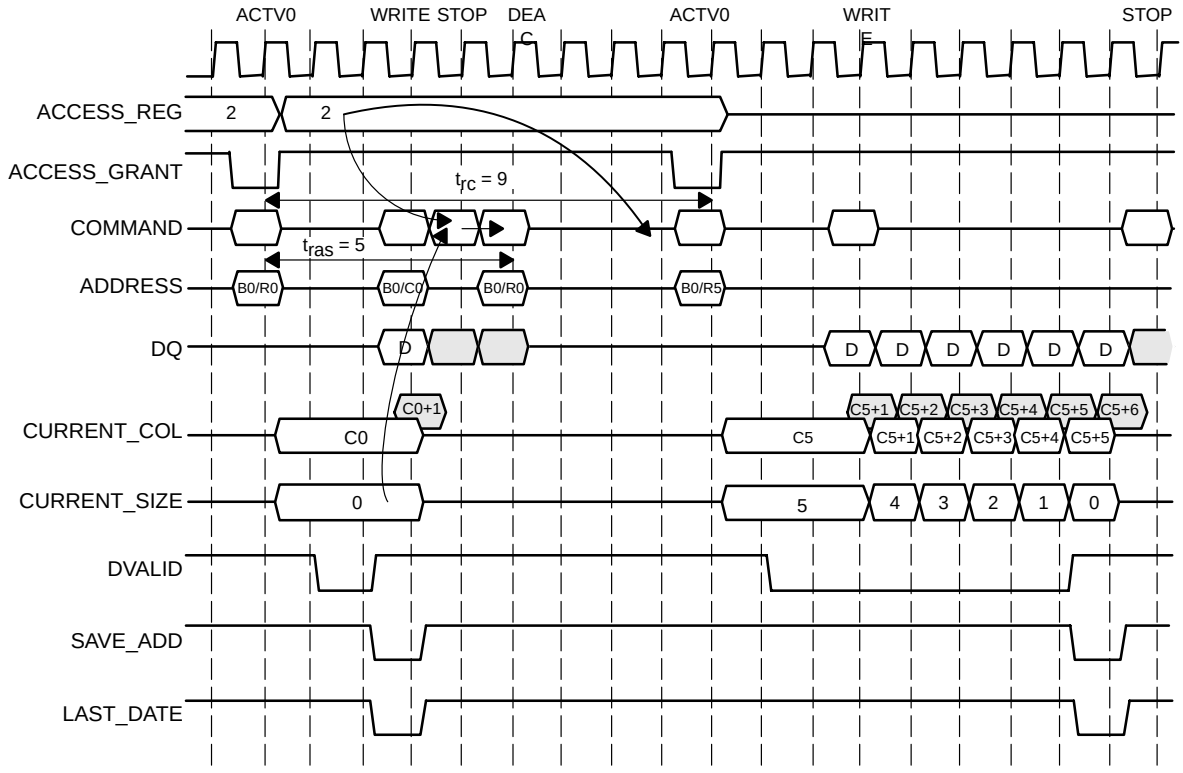
Note: READ (burst reduced to 1) is followed by a READ burst (8) in a different bank and in a page already active.

Figure 4–13. SDRAM Write Burst 32-Bit Word Followed by Read Burst 8 Half-Word



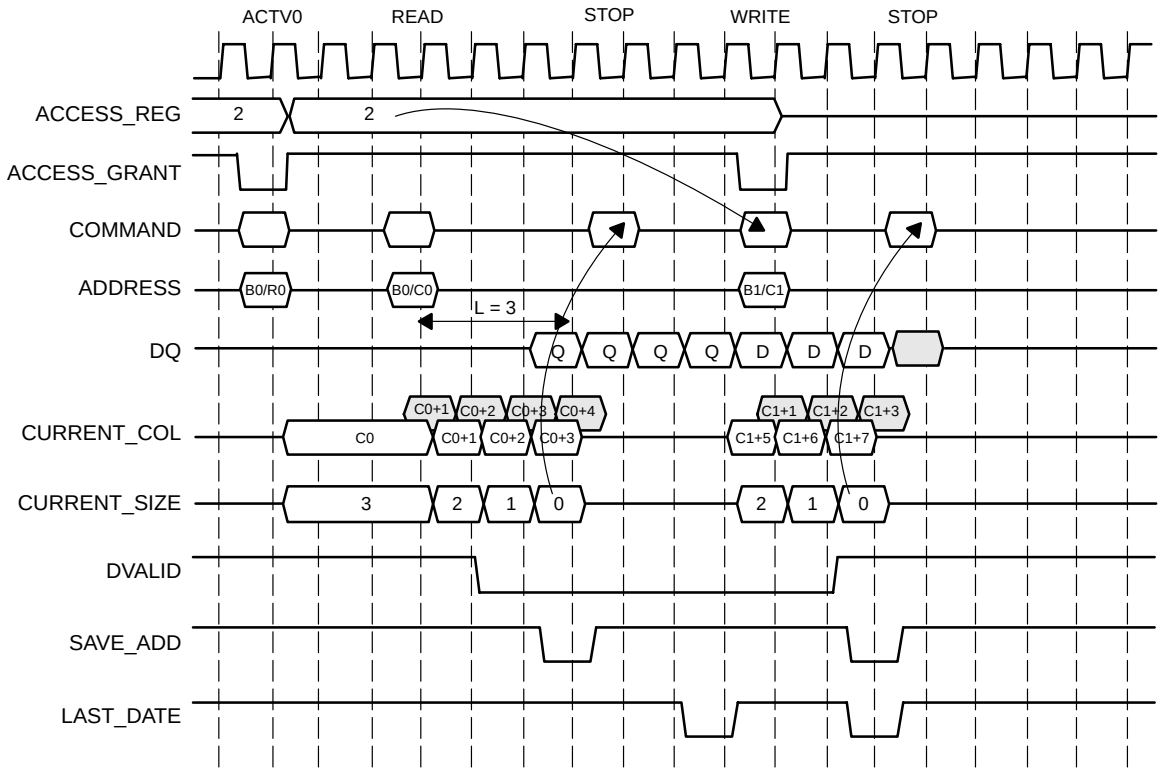
Note: WRITE (burst reduced to 2) is interrupted by a READ request pending on a bank and row already active.

Figure 4–14. SDRAM Single Half-Word Followed by a Read Burst 6 Half-Words



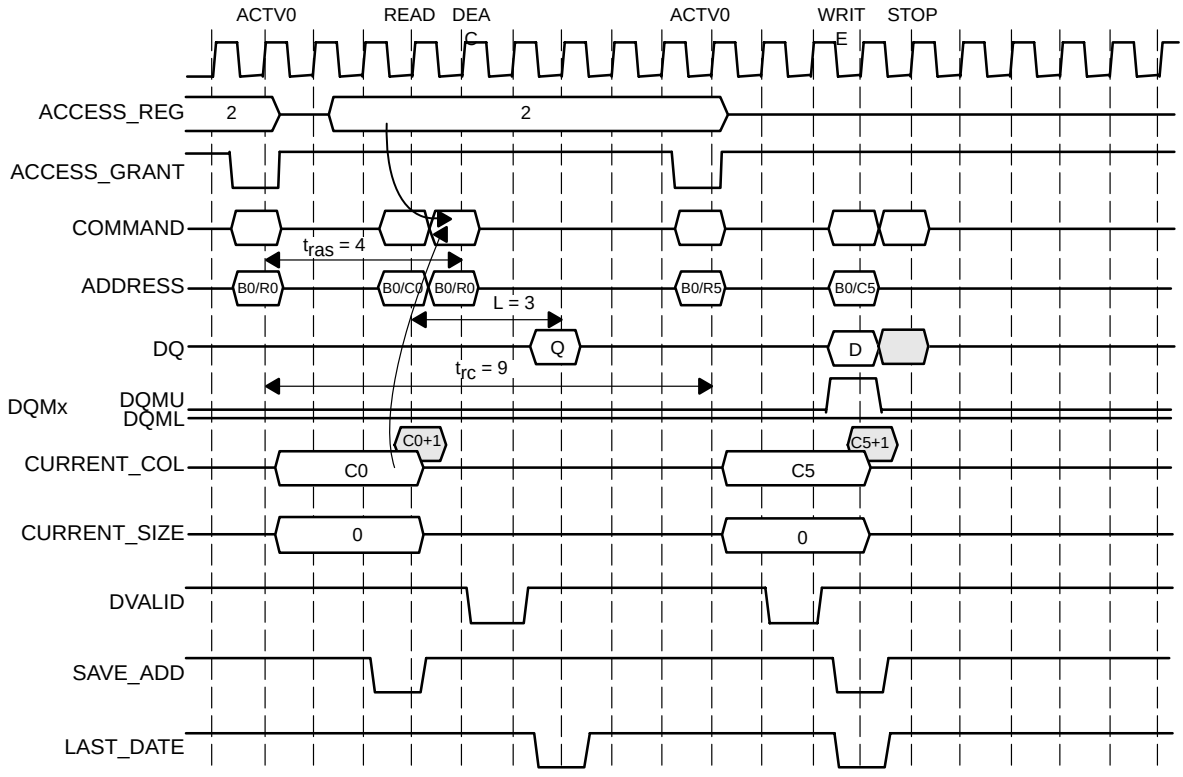
Note: WRITE (burst reduced to 1) is followed by a WRITE (6) in the same bank but on a different page..

Figure 4–15. SDRAM Read Burst 4 Half-Words Followed by a Write Burst 3 Half-Words



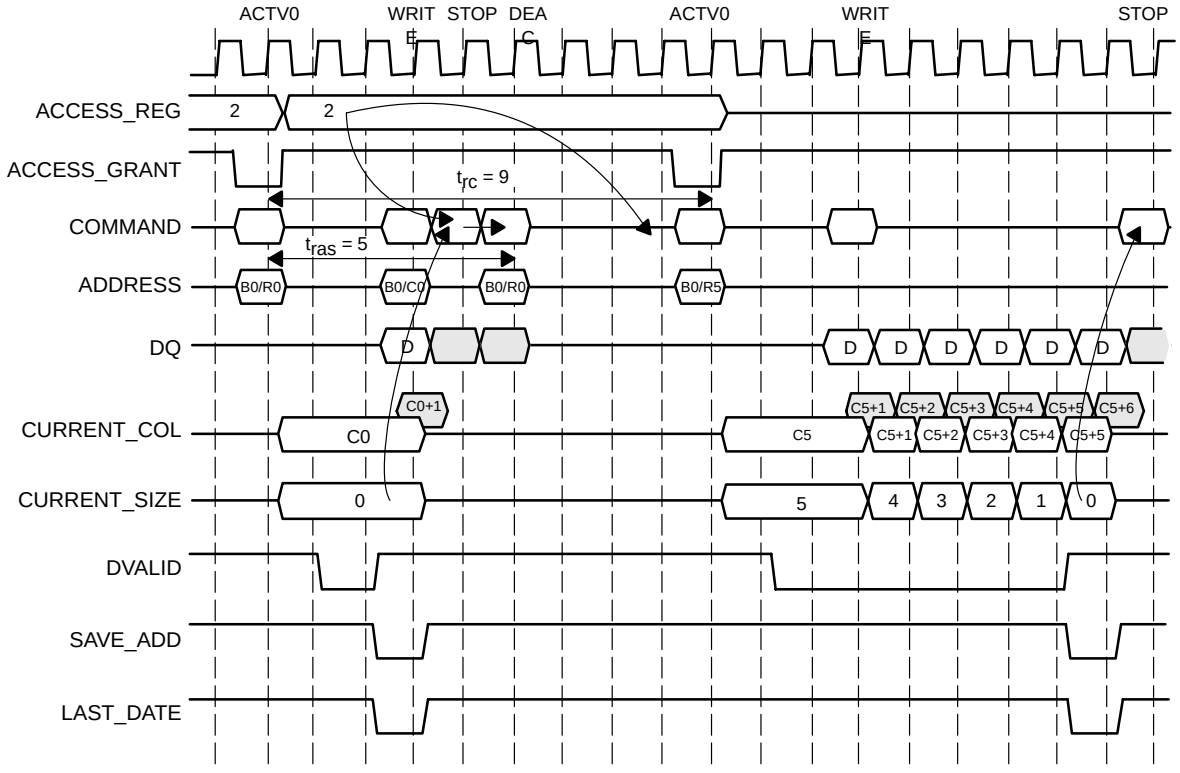
Note: READ (burst reduced to 4) is interrupted by a WRITE request (reduced to 3) pending on a bank and row already active.

Figure 4–16. SDRAM Read Single Half-Word Followed by a Write Byte



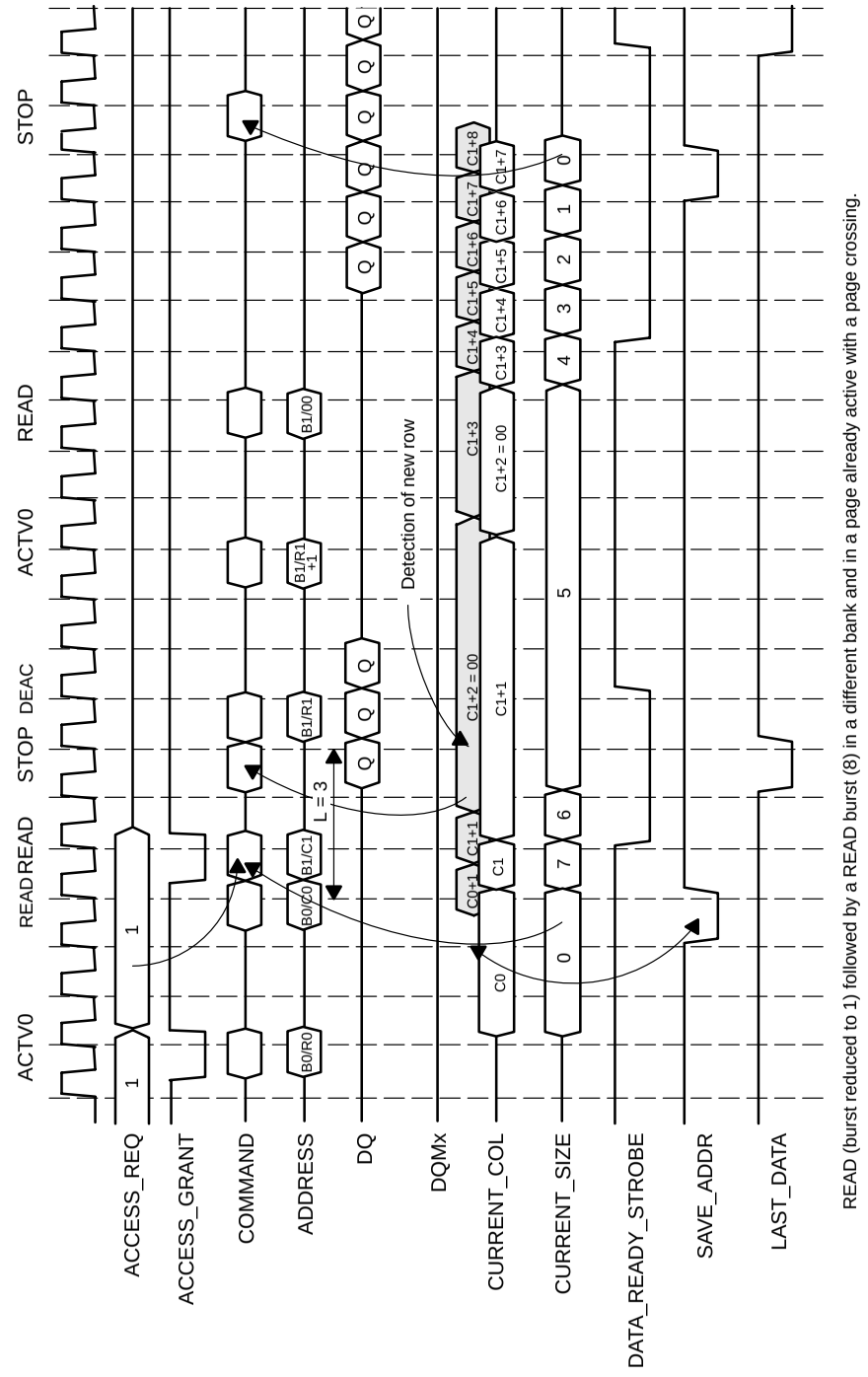
Note: READ (burst reduced to 1) is followed by a single-byte WRITE in the same bank but on a different page.

Figure 4–17. SDRAM Write Single Followed by Write Burst 6 on the Same Bank and Different Page



Note: WRITE (burst reduced to 1) is followed by a WRITE (6) in the same bank but on a different page.

Figure 4–18. SDRAM Read Single Half-Word Followed by a Read Burst 8 With Page Crossing



4.4 Traffic Controller Memory Interface Registers

OMAP5910 traffic controller base address is 0xFFFE:CC00.

Table 4–8 lists the traffic controller registers. Table 4–9 through Table 4–27 describe the register bits.

The EMIF slow interface configuration register provides access to EMIFS boot, operation, and power-down options (see Table 4–12).

Table 4–8. Traffic Controller Registers

Name	Description	R/W	Size	Address	Reset Value
IMIF_PRIO	IMIF priority register	R/W	32 bits	0xFFFE:CC00	0x0000 0000
EMIFS_PRIO	EMIF slow priority register	R/W	32 bits	0xFFFE:CC04	0x0000 0000
EMIFF_PRIO	EMIF fast priority register	R/W	32 bits	0xFFFE:CC08	0x0000 0000
EMIFS_CONFIG_REG	EMIF slow interface configuration register	R/W	32 bits	0xFFFE:CC0C	0x0000 00yy (See Table 4–12 for details on the y values.)
EMIFS_CS0_CONFIG	EMIF slow interface chip-select configuration register nCS0	R/W	32 bits	0xFFFE:CC10	0x0000 FFFB
EMIFS_CS1_CONFIG	EMIF slow interface chip-select configuration register nCS1	R/W	32 bits	0xFFFE:CC14	0x0010 FFFB
EMIFS_CS2_CONFIG	EMIF slow interface chip-select configuration register nCS2	R/W	32 bits	0xFFFE:CC18	0x0010 FFFB
EMIFS_CS3_CONFIG	EMIF slow interface chip-select configuration register nCS3	R/W	32 bits	0xFFFE:CC1C	0x0000 FFFB
EMIFF_SDRAM_CONFIG	EMIF fast interface SDRAM configuration register 1	R/W	32 bits	0xFFFE:CC20	0x0061 8800
EMIFF_MRS	EMIF fast interface SDRAM MRS register	R/W	32 bits	0xFFFE:CC24	0x0000 0037
TIMEOUT1	Timeout1	R/W	32 bits	0xFFFE:CC28	0x0000 0000
TIMEOUT2	Timeout2	R/W	32 bits	0xFFFE:CC2C	0x0000 0000
TIMEOUT3	Timeout3	R/W	32 bits	0xFFFE:CC30	0x0000 0000

Table 4–8. Traffic Controller Registers (Continued)

Name	Description	R/W	Size	Address	Reset Value
ENDIANISM	Endianism	R/W	32 bits	0xFFFE:CC34	0x0000 0000
	Location not used			0xFFFE:CC38	
EMIFF_SDRAM_CONFIG_2	EMIF fast interface SDRAM configuration register 2	R/W	32 bits	0xFFFE:CC3C	0x0000 0003
EMIFS_CFG_DYN_WAIT	EMIF slow wait-state configuration register	R/W	32 bits	0xFFFE:CC40	0x0000 0000

Table 4–9. IMIF Priority Register (IMIF_PRIO)

Bit	Field	Description	Access	Reset Value
31–0	Reserved	Reserved for future expansion. These pins must always be written as 0.	R	All 0s

Table 4–10. EMIF Slow Priority Register (EMIFS_PRIO)

Bit	Field	Description	Access	Reset Value
31–0	Reserved	Reserved for future expansion. These pins must always be written as 0.	R	All 0s

Table 4–11. EMIF Fast Priority Register (EMIFF_PRIO)

Bit	Field	Description	Access	Reset Value
31–0	Reserved	Reserved for future expansion. These pins must always be written as 0.	R	All 0s

Table 4–12. EMIF Slow Interface Configuration Register (EMIFS_CONFIG_REG)

Bit	Field	Value	Description	Access	Reset Value
31–5	Reserved		Read is undefined. Writes must be zero.	R	All 0
4	FR		Ready signal. This bit is a copy of the FLASH.RDY input pin as sampled by TC clock.	R	x
		0	FLASH.RDY pin is low.		
		1	FLASH.RDY pin is high.		
3	PDE		Global power-down enable. This bit is used by EMIFS, EMIFF, and IMIF as an enable for dynamic power down, clock auto-gating. Note, however, that PDE must be set in conjunction with individual power down bits for IMIF and SDRAM before clocks will be cut.	R/W	0
		0	Power down not enabled		
		1	Power down enabled		
2	PWD_EN		IMIF power-down enable. Controls IMIF internal clock enable:	R/W	0
		0	IMIF power down not enabled		
		1	IMIF power down enabled		
			Also note that PWD_EN is one of the prerequisites to meet TC idle. PWD_EN must be set before the memory interface can acknowledge a TC idle request.		
1	BM		MPU boot mode. This bit is sampled at reset from the MPU_BOOT device pin. BM enables CS0 and CS3 address decode swapping.	R/W	x
		0	CS0 [0000:0000 – 01FF:FFFF] CS3 [0C00:0000 – 0DFF:FFFF]		
		1	CS0 [0C00:0000 – 0DFF:FFFF] CS3 [0000:0000 – 01FF:FFFF]		
			Since BM is read/write, care must be exercised not to write the bit since there is potential to inadvertently modify EMIFS memory mapping.		
0	WP		Write protect bit. Enables write protection for all flash devices.	R/W	0
		0	$\overline{\text{FLASH.WP}}$ output pin is set low.		
		1	$\overline{\text{FLASH.WP}}$ output pin is set high.		

The four EMIF slow chip-select configuration registers (see Table 4–13) are used to select the protocols and timings to be used for handshake with devices connected to CS0-CS3 (corresponding to device pins $\overline{\text{FLASH.CS0}}$ - $\overline{\text{FLASH.CS3}}$). Table 4–14 describes the memory types, and Table 4–15 describes the wait cycles insertion.

Table 4–13. EMIF Slow Chip-Select Configuration Registers (EMIFS_CS0_CONFIG...EMIFS_CS3_CONFIG)

Bit	Field	Value	Description	Access	Reset Value
31–22	Reserved		Read is undefined. Writes must be zero.	R	All 0
21	FL		Specifies how EMIFS handles addressing when performing 32-bit writes to the OMAP5910 16-bit data bus.	R/W	0
		0	The address is incremented for the second 16-bit access (default).		
		1	The address is not incremented for the second 16-bit access.		
			This bit is valid only when EMIFS is configured for 16-bit data bus width (BW = 0). This bit has no effect for read operations.		
20	BW		Specifies EMIFS data bus width.	R/W	
		0	16-bit bus. This is the appropriate setting for OMAP5910.		
		1	Reserved. Do not use this setting on OMAP5910.		
			(BW bit reset value depends on the chip-select: For CS0 and CS3, BW = 0; For CS1 and CS2, BW = 1. If CS1 or CS2 is to be used, BW must first be written to 0 since OMAP5910 only supports 16-bit bus.)		
19	Reserved		Read is undefined. Writes must be zero.	R	0
18:16	RDMODE		Read mode select (see Table 4–14)	R/W	000
15:12	PGWST/WELEN		For read accesses, number of wait states for page mode ROM reads within a page. For write accesses, the length of $\overline{\text{WE}}$ pulse duration.	R/W	1111
11:8	WRWST		Numbers of wait states for write operation	R/W	1111

Table 4–13. EMIF Slow Chip-Select Configuration Registers (EMIFS_CS0_CONFIG...EMIFS_CS3_CONFIG) (Continued)

Bit	Field	Value	Description	Access	Reset Value
7:4	RDWST		Number of wait states for asynchronous read operation (see Table 4–15). Number of inserted clock cycles in protocol (value matches the value programmed in Intel flash devices).	R/W	1111
3	Reserved		Read is undefined. Writes must be zero.	R/W	U
2	RT		Retiming control register:	R/W	0
		0	The data is not retimed.		
		1	The data coming from the external bus is retimed with the CLK.		
1:0	FCLKDIV		EMIFS internal reference clock divider:	R/W	11
		00	Reference clock = TC clock divided by 1		
		01	Reference clock = TC clock divided by 2		
		10	Reference clock = TC clock divided by 4		
		11	Reference clock = TC clock divided by 6		

Table 4–14. Memory Type

RDMODE	Memory
000	Asynchronous read
001	Page mode ROM read—4 words per page
010	Page mode ROM read—8 words per page
011	Page mode ROM read—16 words per page
100	Synchronous burst read
Others	Reserved. Do not use.

Table 4–15. Wait Cycles Insertion

RDWST	Number of Cycles Inserted
0	2
1	3
2	4
3	5
4	6
5	7

There is no automatic hardware adjustment of the programmed latencies when the system clock frequency changes.

The following restrictions apply when synchronous burst read Intel protocol is selected:

- ❏ Only continuous burst mode is supported
- ❏ Only sequential data access order is supported
- ❏ Only 1 clock cycle data duration mode is supported (there is no gain to support 2 clock cycle duration since FLASH.CLK may be divided).

Page crossing is supported in page mode ROM burst read.

In asynchronous read mode, $\overline{\text{FLASH.ADV}}$ is activated during one FLASH.CLK cycle in order to ensure compatibility with burst flash.

Table 4–16. EMIF Fast Interface SDRAM Configuration Register 1 (EMIFF_SDRAM_CONFIG)

Bit	Field	Value	Description	Access	Reset Value
31–28	Reserved		Read is undefined. Writes must be zero.	R	All 0
27	CLK		SDRAM clock disable. See section 4.3.3.6, *SDRAM Clock Disable*, for details related to disabling the SDRAM clock.	R/W	0
		0	Clock is not disabled.		
		1	Clock is disabled.		
			CLK is one of the prerequisites to meet TC idle. CLK must be set before the memory interface can acknowledge a TC idle request.		

Table 4–16. EMIF Fast Interface SDRAM Configuration Register 1 (EMIFF_SDRAM_CONFIG) (Continued)

Bit	Field	Value	Description	Access	Reset Value
26	PWD		SDRAM power-down enable. Controls power-down state of SDRAM interface:	R/W	0
		0	SDRAM interface is not powered down.		
		1	SDRAM interface is powered down.		
			PWD is one of the prerequisites to meet TC idle. PWD must be set before the memory interface can acknowledge a TC idle request.		
25–24	SDRAM_ FREQUENCY		SDRAM frequency range. Selects one of four SDRAM timing configurations based on clock latencies. See Table 4–18.	R/W	00
		00	SDF0 (reset value)		
		01	SDF1		
		10	SDF2		
		11	SDF3		
23–8	ARCV		Autorefresh counter register value. Sets the interval between partial refresh requests to the SDRAM. See Section 4.3.3.4, *SDRAM Autorefresh Initialization*, for formula and example.	R/W	0x6188
7–4	SDRAM_TYPE		Set the SDRAM internal organization (see Table 4–17)	R/W	0000
3–2	ARE		Autorefresh enable. When autorefresh enable is set, the EMIF 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.	R/W	00
		00	Autorefresh disable		
		01	Autorefresh enable		
		10	Autorefresh by burst of 4 commands		
		11	Autorefresh by burst of 8 commands		

Table 4–16. EMIF Fast Interface SDRAM Configuration Register 1 (EMIFF_SDRAM_CONFIG) (Continued)

Bit	Field	Value	Description	Access	Reset Value
1	SD_RET		SDRAM retiming:	R/W	0
		0	Data is single buffered with the return clock from SDRAM.		
		1	Data from SDRAM is double buffered. Data is first clocked on return clock from SDRAM, then with the OMAP5910 internal SDRAM clock.		
0	SLRF		When set, places the SDRAM in self-refresh mode. Mode is automatically exited upon the generation of any SDRAM access.	R/W	0

This register is used to configure the SDRAM, interface timing, autorefresh setup, and powerdown modes of the EMIFF interface. Table 4–17 describes the internal organization. Table 4–18 describes the frequency range.

Table 4–17. SDRAM Internal Organization

Register Value	Memory Size (M Bits)	Size Of Data Bus	Number Of Banks
0000	16	8	2
0001		8	4†
0010		16	2
0011		16	4†
0100	64	8	2
0101		8	4
0110		16	2
0111		16	4
1000	128	8	2†
1001		8	4

† Unavailable bank number (not supported). Do not use this setting.

Note: Reset value = 0x2h.

Table 4–17. SDRAM Internal Organization (Continued)

Register Value	Memory Size (M Bits)	Size Of Data Bus	Number Of Banks
1010		16	2†
1011		16	4
1100	256	8	2†
1101		8	4
1110		16	2†
1111		16	4

† Unavailable bank number (not supported). Do not use this setting.

Note: Reset value = 0x0h.

Table 4–18. Frequency Range

ac Parameters	SDF0 (Cycles)	SDF1 (Cycles)	SDF2 (Cycles)	SDF3 (Cycles)
t_{rc}	9	5	3	2
t_{ras}	5	3	2	2
t_{rp}	3	2	2	2
t_{rcd}	3	2	2	2
t_{rrd}†	2	2	2	2
t_{dpl}(trwl)‡	–	–	–	–
t_{dal}	–	–	–	–
t_{rsc}	2	2	2	–

† Write is never interrupted by precharge command directly.

‡ Neither read or write with auto-precharge is supported.

Table 4–19. SDRAM Timing Requirements

ac Parameters	SDRAM Timing Requirements (ns)	Meeting this Timing With SDRAM.CLK = 60 MHz (16.7 ns Period)
t_{rc}	80	5
t_{ras}	48	3
t_{rp}	24	2
t_{rcd}	24	2
t_{rrd}†	16	1
t_{dpl}(trwl)‡	8	–
t_{dal}	27	–
t_{rsc}	2	1

† Write is never interrupted by precharge command directly.
‡ Neither read or write with autoprecharge is supported.

For 60 MHz, timing can be met by using the SDF1 timing configuration.

This register, when written, programs the SDRAM MRS (default) and EMRS configuration registers. In default mode, a write to the register initiates an MRS request to the SDRAM. In EMRS mode, a write to this same register initiates an EMRS request. Reading this register does not issue an external transaction. Table 4–20 describes the bits for the MRS mode. Table 4–21 describes the bits for the EMRS mode.

Table 4–20. EMIF Fast Interface SDRAM MRS Register—Default (EMIFF_MRS)

Bit	Field	Value	Description	Access	Reset Value
31–10	Reserved		Read is undefined. Writes must be zero.	R	All 0
9	WBST		Write burst must be 0 (burst write same as burst read).	R/W	0
8–7	Reserved		Read is undefined. Writes must be zero.	R/W	00
6–4	CASL		CAS latency:	R/W	011
		001	CAS latency = 1		
		010	CAS latency = 2		
		011	CAS latency = 3 (default at reset)		
3	S/I		Serial = 0. This bit must be 0. Interleave = 1. Reserved. Do not use this setting.	R/W	0
2–0	PGBL		Specifies page burst length to be programmed into SDRAM MRS configuration register. The length must always be programmed as full-page burst length (111). (This length is not necessarily the burst length at which the EMIFF operates, but rather a setting for the SDRAM MRS register.)	R/W	111

Note: When the CONF_MOD_EMRS_CTRL bit field (bit 13) of the OMAP5910 control register (MOD_CONF_CTRL_0) is set, the device reconfigures bank settings to write out the EMIFF_MRS register as EMRS commands (see Table 4–21).

Table 4–21. EMIF Fast Interface SDRAM MRS Register—EMRS Mode (EMIFF_MRS)

Bit	Field	Value	Description	Access	Reset Value
31–5	Reserved		Read is undefined. Writes must be zero.	R	See Note 1
4–3	TCSR		SDRAM EMRS register temperature compensated self-refresh setting:	R/W	See Note 1
		00	70 degrees Celsius maximum case temperature		
		01	45 degrees Celsius maximum case temperature		
		10	15 degrees Celsius maximum case temperature		
		11	85 degrees Celsius maximum case temperature		
			Bit descriptions are given with respect to standard SDRAM devices and must be verified with the actual SDRAM chosen for the application.		
2–0	PASR		SDRAM EMRS register partial array self-refresh coverage setting:	R/W	See Note 1
		000	All banks		
		001	Half array		
		010	Quarter array		
		011	Reserved		
		100	Reserved		
		101	Reserved		
		110	Reserved		
		111	Reserved		
			Bit descriptions are given with respect to standard SDRAM devices and must be verified with the actual SDRAM chosen for the application.		

Notes: 1) Reset value is defined by the default mode of the register (see Table 4–20).

The three time-out registers store the number of clock cycles before DSP, DMA, LCD, LB requests are made high-priority in dynamic priority scheme for the TC (see Table 4–22 through Table 4–24).

Table 4–22. Time-Out 1 Register (TIMEOUT1)

Bit	Field	Description	Access	Reset Value
31–24	Reserved	Read is undefined. Writes must be zero.	R	All 0
23–16	Local bus		R/W	0x00
15:8	Reserved	Read is undefined. Writes must be zero.	R/W	All 0
7:0	DMA		R/W	0x00

Table 4–23. Time-Out 2 Register (TIMEOUT2)

Bit	Field	Description	Access	Reset Value
31–24	Reserved	Read is undefined. Writes must be zero.	R	All 0
23–16	DSP		R/W	0x00
15–8	Reserved	Read is undefined. Writes must be zero.	R/W	All 0
7–0	LCD		R/W	0x00

Table 4–24. Time-Out 3 Register (TIMEOUT3)

Bit	Field	Description	Access	Reset Value
31–0	Reserved	Read is undefined. Writes must be zero.	R	All 0

The endianism register (ENDIANISM) is used to control endianism conversion in the DSP memory management unit endianism block.

Table 4–25. Endianism Register (ENDIANISM)

Bit	Field	Value	Description	Access	Reset Value
31–2	Reserved		Read is undefined. Writes must be zero.	R	All 0
1	SWAP	0	Byte swap (8 bits)	R/W	0
		1	Word swap (16 bits)		
0	EN	0	Endianism conversion is disabled (default).	R/W	0
		1	Endianism is enabled.		

Table 4–26. EMIF Fast Interface SDRAM Configuration Register 2 (EMIFF_SDRAM_CONFIG_2)

Bit	Field	Value	Description	Access	Reset Value
31–2	Reserved		Read is undefined. Writes must be zero.	R	All 0
1	RFRSH_ RST		SDRAM self-refresh on warm reset. RFRSH_RST determines what action the TC SDRAM controller takes toward setting SDRAM to self-refresh mode in the event of a warm system reset.	R/W	1
		0	SDRAM is not entered to self-refresh mode.		
		1	SDRAM is entered to self-refresh mode upon warm system reset.		
0	RFRSH_ STBY		SDRAM self-refresh on standby. After the TC receives an idle request from the clock generation module, RFRSH_STBY determines what action the TC SDRAM controller takes toward setting SDRAM to self-refresh mode prior to acknowledging the idle request.	R/W	1
		0	SDRAM enters self-refresh mode.		
		1	SDRAM enters self-refresh mode prior to the TC acknowledging an idle request.		

Table 4–27. EMIF Slow Wait State Configuration (EMIFS_CFG_DYN_WAIT)

Bit	Field	Value	Description	Access	Reset Value
31–4	Reserved		Read is undefined. Writes must be zero.	R	All 0
3	DYNW_CS3		Specifies function of FLASH.RDY for CS3.	R/W	0
		0	Enable classic not-ready for EMIFS CS3.		
		1	Enable dynamic not-ready for EMIFS CS3.		
2	DYNW_CS2		Specifies function of FLASH.RDY for CS2.	R/W	0
		0	Enable classic not-ready for EMIFS CS2.		
		1	Enable dynamic not-ready for EMIFS CS2.		
1	DYNW_CS1		Specifies function of FLASH.RDY for CS1.	R/W	0
		0	Enable classic not-ready for EMIFS CS1.		
		1	Enable dynamic not-ready for EMIFS CS1.		
0	DYNW_CS0		Specifies function of FLASH.RDY for CS0.	R/W	0
		0	Enable classic not-ready for EMIFS CS0.		
		1	Enable dynamic not-ready for EMIFS CS0.		

4.5 Interfacing Memories With the OMAP5910 Device

This section provides two examples of how to connect memories to the OMAP5910 device. Many scenarios can be considered using different kinds of memories. For flash memories, Intel and Hitachi products are used. For SDRAM and SRAM, Hitachi and Toshiba products are used, respectively.

The Intel flash memory has a total capacity of 160M bits (8M x 8 x 2 chips and 2M x 16). Program code uses two Intel 28F64J3A memories, and data uses Intel 28F32J3A. The power supply voltages for these memories range from 2.7 V to 3 V. Hitachi memory has a total capacity of 96M bits in this example. Two flash memories are used for program code, and one flash memory is used for data. The power supply voltage also ranges from 2.7 V to 3 V.

Hitachi SDRAM (HM52Y64165F) has a total capacity of 64M bits (4M x 16). Its power supply voltage ranges from 2.5 V to 2.8 V. Toshiba SRAM (JT5MM6A-AD) has 8M-bit capacity and 2.7-V to 3-V power supply voltage ranges.

Figure 4–19 shows external memory interconnection using Intel flash memory, and Figure 4–20 shows external memory interconnection using Hitachi flash memory.

Figure 4–19. External Memory Interconnection Using Intel Flash Memory

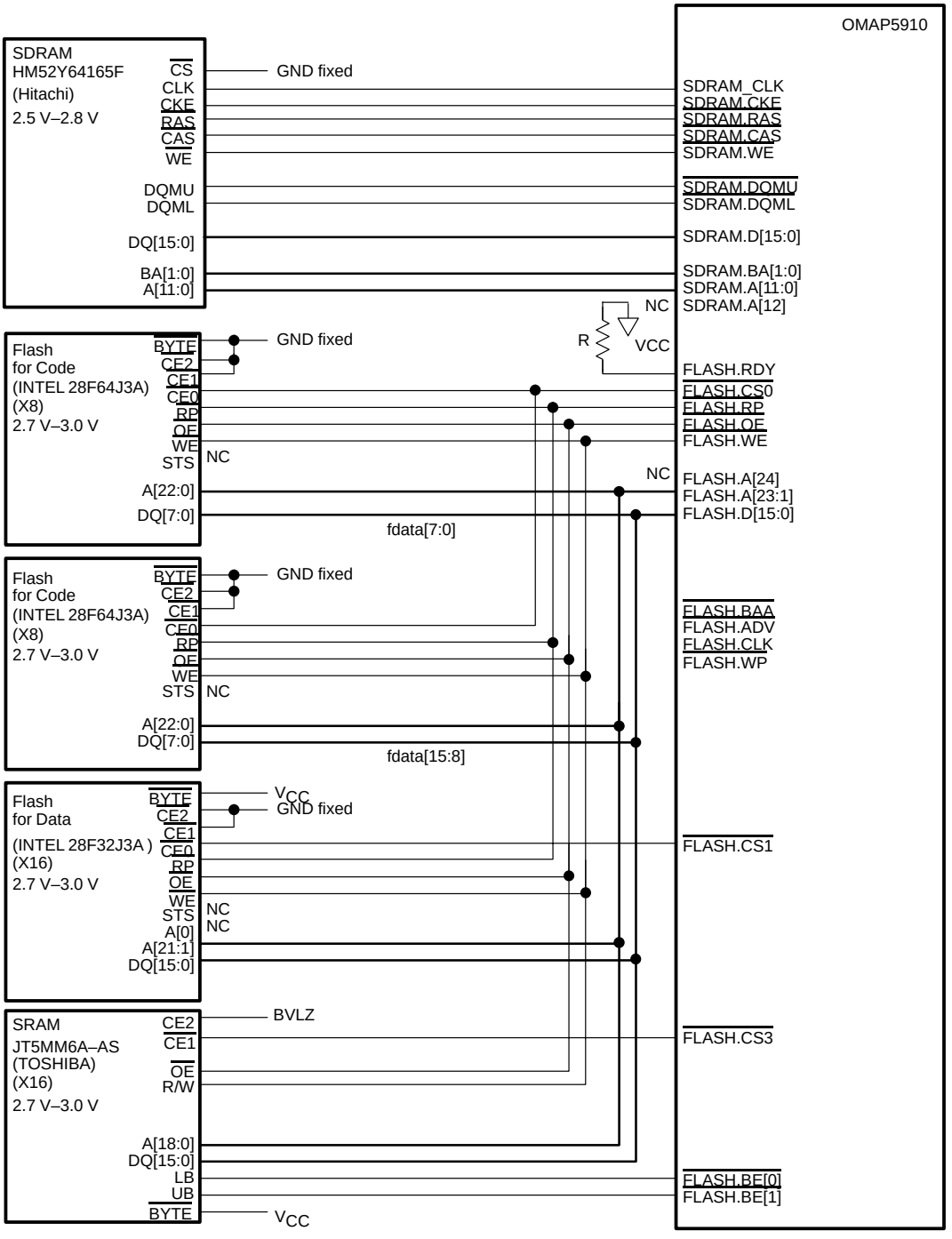
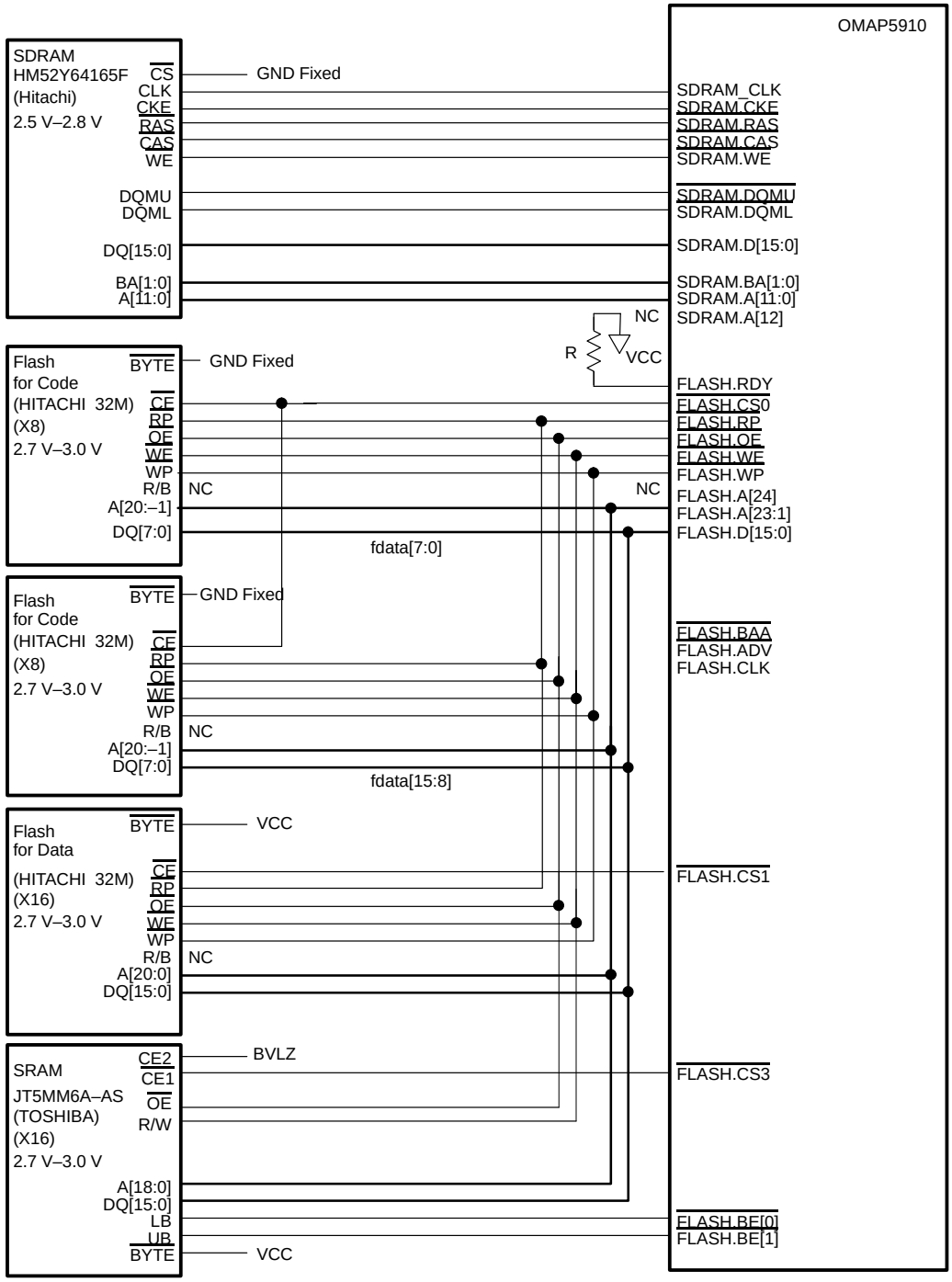


Figure 4–20. External Memory Interconnection Using Hitachi Flash Memory



Chapter 5

System DMA Controller

This chapter describes the system DMA controller for the OMAP5910 multimedia processor.

Topic **Page**

5.1 Introduction

The system direct memory access (DMA) controller transfers data between points in the memory space without intervention by the MPU. The DMA allows movements of data to and from internal memory, external memory, and peripherals to occur in the background of MPU operation. It is designed to off-load the block data transfer function from the MPU processor.

Figure 5–1 shows the OMAP5910 device with the DMA controller highlighted. Figure 5–2 shows the controller in more detail.

Figure 5–1. Highlight of DMA Controller

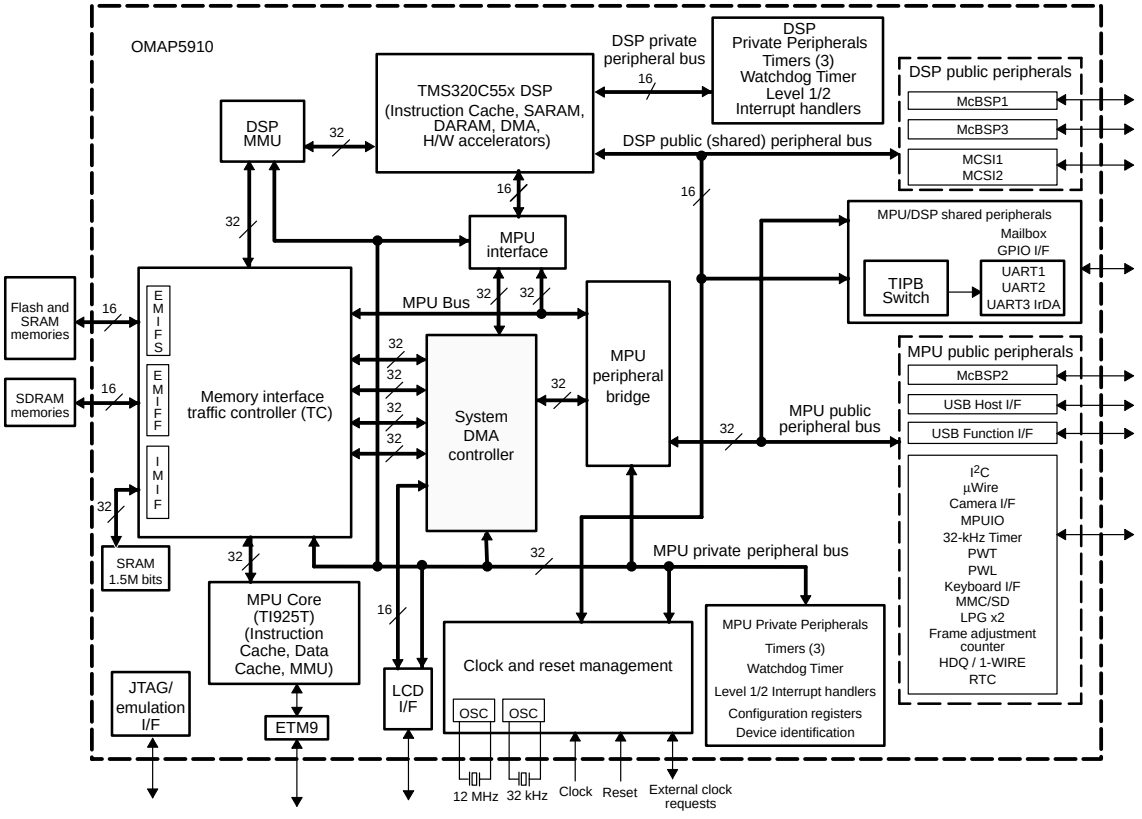
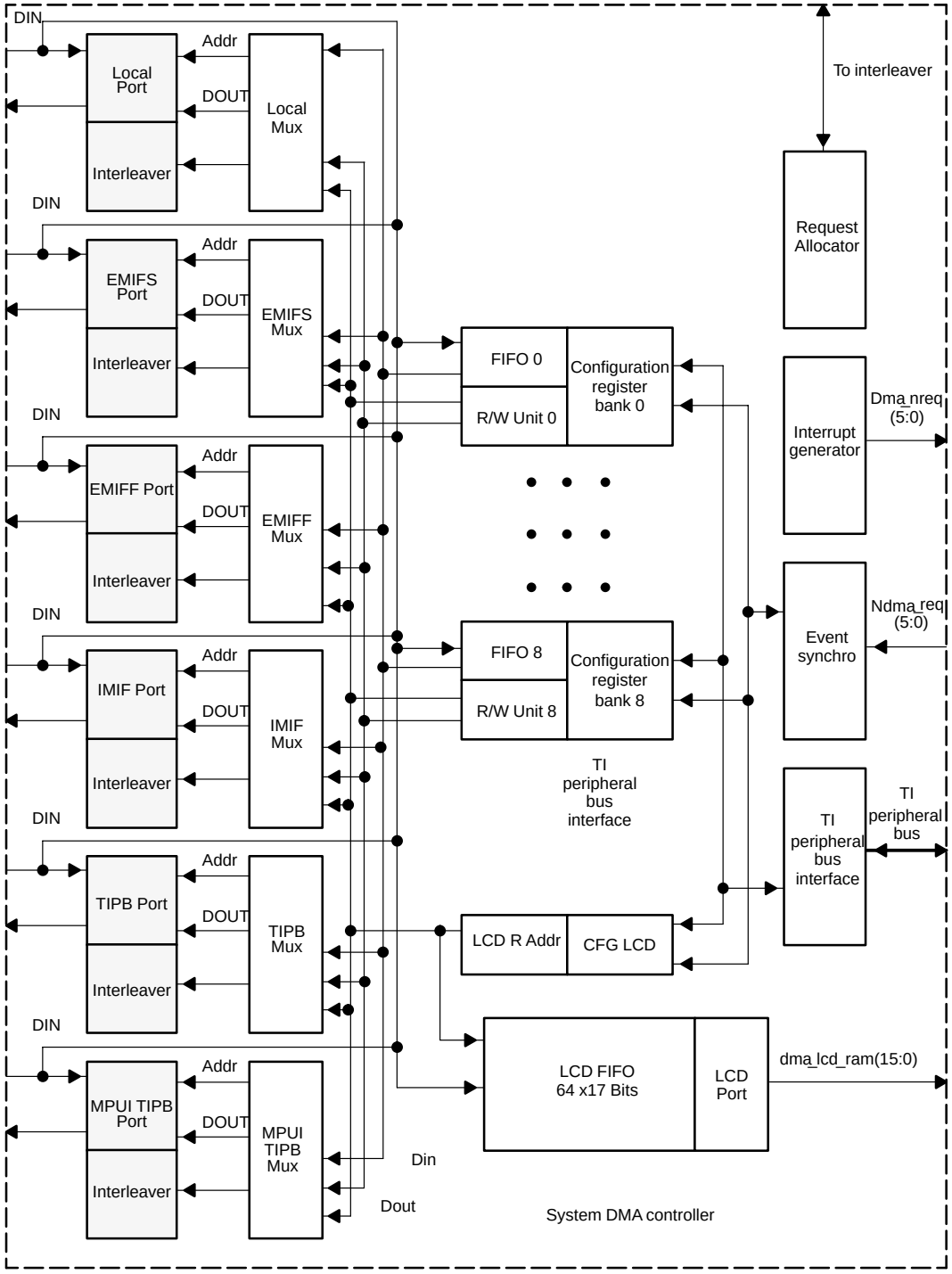


Figure 5–2. DMA Controller Block Diagram



Transfers are made through a physical channel that can be thought of as a pipe that connects a source and a destination for the duration of a transfer. Data flows through this pipe from the source to the destination. After the transfer is completed, the pipe (channel) can be used to perform other data transfers that involve the same or other sources and destinations.

The DMA channels are said to be physical because each of them is hardware-implemented. Each channel is controlled by a set of configuration registers, where the software sets up the transfer parameters such as length, source, and destination addresses. This set of registers is called the transfer descriptor or the transfer contexts. The transfer contexts are set up by the processor through the MPU's private TI peripheral bus (TIPB). Physical channel configuration registers are in the MPU's memory space.

All of the physical channels operate concurrently, which allows several data transfers to run in parallel, one in each channel. If several channels use the same DMA port, they are time-multiplexed by this port and can be priority scheduled through software parameters available to the user. Thus, the user is able to control the sharing of a port between channels by the assignment of priority levels.

The system DMA controller has nine independently programmable generic channels plus one channel that is specifically dedicated only to transfers from either the IMIF or the EMIFF ports to the LCD port. By dynamically assigning one of the nine physical channels to a pair of ports, data can be transferred to/from the designated ports.

The DMA interface with the source and destination of transfers is called a port. Each port can have its own protocol to communicate to the resource (memory or TIPB bridge) to which it is connected. All of the existing DMA ports used by the system DMA are described in detail later in this document.

The DMA has six general-purpose ports:

- ❏ External memory interface fast (EMIFF) port
- ❏ External memory interface slow (EMIFS) port
- ❏ Internal memory interface (IMIF) port
- ❏ Local bus interface (Local) port
- ❏ MPU interface (MPUI) port
- ❏ TIPB interface (TIPB) port

A seventh, special port is provided to interface with an LCD controller. This port, the LCD port, has a dedicated channel that connects either the IMIF or EMIFF port to the LCD controller. Unlike the other DMA ports, the LCD port can only receive data from the IMIF or EMIFF ports.

The system DMA controller is controlled by the MPU (via the TIPB). The DMA controller meets the high-rate-flow requirements of the multichannel applications used by wireless base stations.

The system DMA controller is designed for low-power operation. Its clock can be automatically disabled as required. This function is synchronous to the MPU TIPB and is entirely under hardware control. No specific programming is required.

The functional features of the system DMA controller for general-purpose transfers are:

- ❏ Nine general-purpose and one dedicated (LCD) DMA channels
- ❏ Software programmable DMA access priority-based on resource allocation versus processor (MPU or DSP) access. Through the assignment of priority levels for each channel, the user can determine how the ports are shared between channels.
- ❏ Concurrent DMA transfers capability
- ❏ Start of transfer on peripheral request or host request
- ❏ Byte alignment capability
- ❏ Byte packing/unpacking
- ❏ Byte transfer count
- ❏ Configurable indexes through memory for each channel source and destination address register. The address may remain constant or post-increment.
- ❏ Access available to all of the memory range (physical memory mapping and I/O space)
- ❏ Seven ports available for different kinds of hardware resources. All data exchanges are done with a simple handshake mechanism with request, ready, and abort signals. All of the ports except the TIPB have burst access capability.
  - EMIFS port
  - EMIFF port
  - IMIF port
  - MPUI port
  - TIPB port
  - Local port
  - LCD port

- Memory-to-memory transfer granularity of 8, 16, and 32 bits. Only the number of programmed bytes is transferred; that is, there are no trailing or dirty bytes at the end of transfer.
- TIPB-to-memory transfer:
  - When performing DMA transfers from a TIPB peripheral, is it ideal that the peripheral FIFO size be 16 bytes, which corresponds to the DMA channel FIFO size.
  - If the peripheral FIFO size is not 16 bytes, use a general-purpose timer as a time-out counter to avoid the possibility that some data might remain in the channel FIFO at the end of frame. When the MPU receives the interrupt, it must check the DMA channel FIFO status and read the data located in the FIFO, if needed.

Additional functional features and limitations of the DMA controller include:

- General-purpose DMA channels can perform 4x32 bit bursts; this is the only burst mode supported by the non-LCD DMA channels. Bursts can only be performed on the IMIF, EMIFF, EMIFS, and local bus ports.
- All LCD DMA accesses are performed in bursts of 8 x16 bits. The LCD video buffer data must be a multiple of 16 bytes. The start address and end address must be aligned on a 16-bit boundary.
- General-purpose channel priority on the external EMIFF/IMIF bus is software programmable.
- LCD channel FIFO size is 64x17 bits.
- The interface between the LCD controller and the DMA controller uses a different protocol from the other DMA channels. The LCD controller is the master of this interface, and it generates the addresses according to the FIFO status.
- The LCD controller allows the use of one or two video buffer(s) and is configurable.
- No bursts are supported on the TIPB and MPUI DMA interfaces.
- Burst transfers can be combined with single or double-indexed addressing modes, but the burst request is only issued if contiguous addresses can be ensured for the length of the burst. For example, if the element index contains a value that causes noncontiguous addresses between elements, then the system DMA logic does not generate burst requests.

Table 5–1 lists the possible data transfers, and Table 5–2 lists the possible transfer sizes and types.

Table 5–1. Possible Data Transfers

Destinations vs. Sources	EMIFS Bus	EMIFF Bus	IMIF Bus	TIPB	MPUI	LCD	Local Bus
EMIFS bus	Yes	Yes	Yes	Yes	Yes	No	Yes
EMIFF bus	Yes	Yes	Yes	Yes	Yes	Yes	Yes
IMIF bus	Yes	Yes	Yes	Yes	Yes	Yes	Yes
TIPB	Yes	Yes	Yes	Yes	Yes	No	Yes
MPUI	Yes	Yes	Yes	Yes	Yes	No	Yes
Local bus	Yes	Yes	Yes	Yes	Yes	No	Yes

Table 5–2. Possible Transfer Sizes and Types

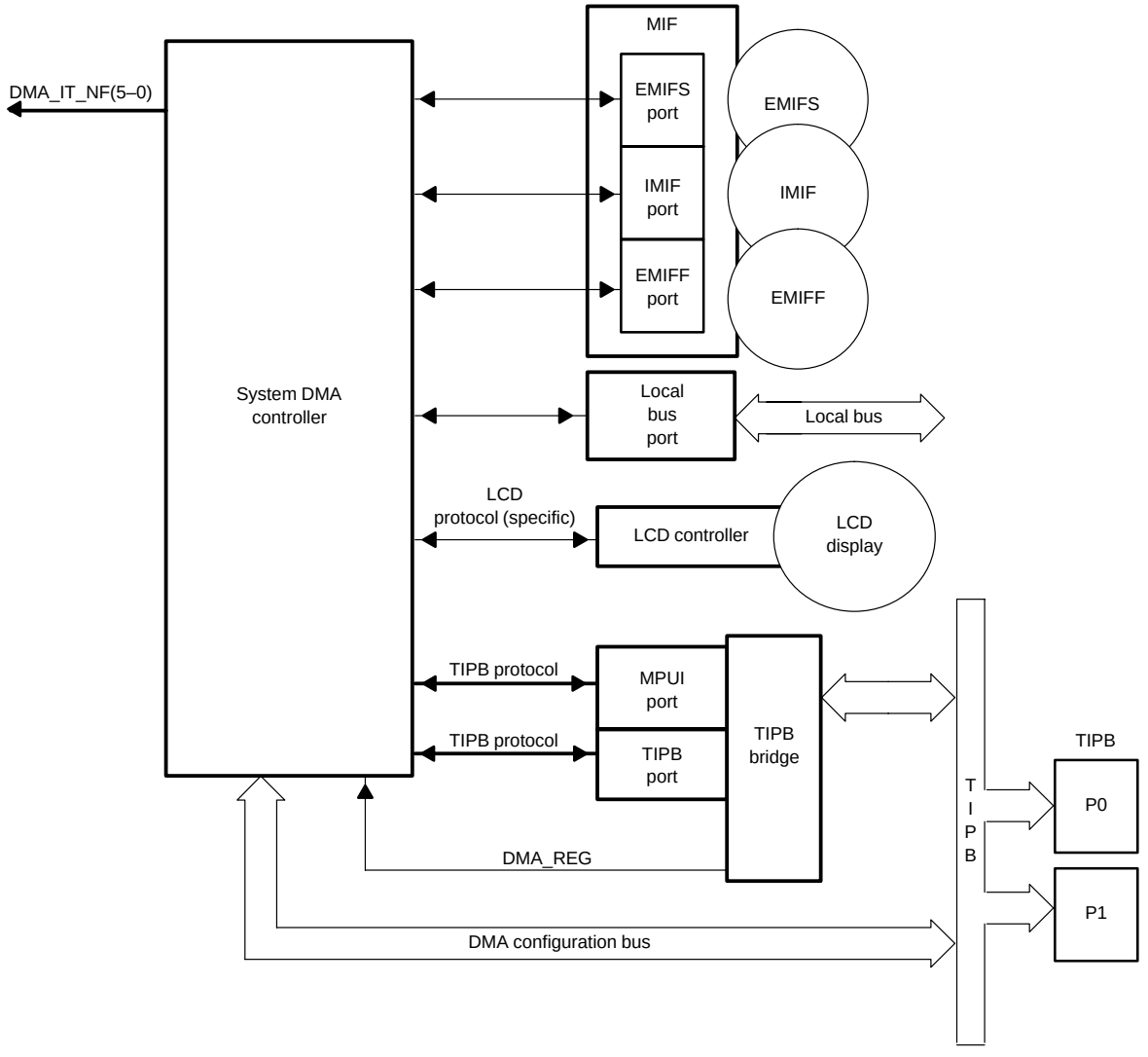
Source	Destination: 8-Bit TIPB	Destination: 8-Bit Non-TIPB	Destination: 16-Bit TIPB	Destination: 16-Bit Non-TIPB	Destination: 32-Bit TIPB	Destination: 32-Bit Non-TIPB
8-bit TIPB	Valid only with no packing s8	Valid s8	Not allowed	Valid s8	Not allowed	Valid s8
8-bit non-TIPB	Valid only with no packing s8	Valid s8	Not allowed	Valid s8,s16	Not allowed	Valid s8,s16,32
16-bit TIPB	Not allowed	Valid s16	Valid only with no packing s16	Valid s16	Not allowed	Valid s16
16-bit non-TIPB	Valid with s8 and no packing	Valid s8	Valid only with no packing s16	Valid s8,s16	Valid only with packing s32	Valid s8,s16,32
32-bit TIPB	Not allowed	Valid s32	Not allowed	Valid s32	Valid s32	Valid s32
32-bit non-TIPB	Valid with s8 and no packing	Valid s8	Valid only with no packing s16	Valid s8,s16	Valid S32	Valid s8,s16,32

Note: s8 = 8-bit scalar data type; s16 = 16-bit scalar data type; s32 = 32-bit scalar data type

5.2 External Connections

The system DMA controller is interconnected with other OMAP5910 components as shown in Figure 5–3.

Figure 5–3. System DMA External Connections



5.3 Generic Channels

This section discusses the following generic channel topics:

- Transfers
- Addressing modes
- Data packing and bursting
- Data alignment
- Constraint on channel configuration parameters
- Endianism
- Interrupt generation
- Memory space protection

5.3.1 Transfers

5.3.1.1 Transfer Sources and Destination

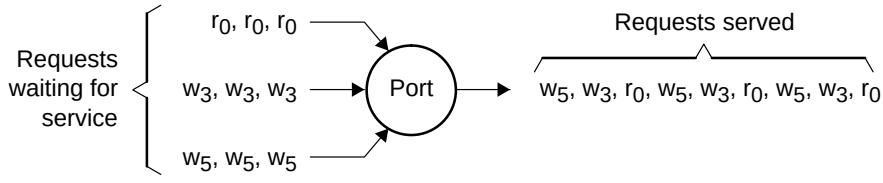
Each DMA channel can be configured independently from other channels. This implies that a port can be shared by several channel requests. Therefore, these requests are time-multiplexed by the port.

For example, in Figure 5–4 a DMA port must service requests from three DMA channels:

- Channel 0 as a source port (read requests, r_0)
- Channel 3 as a destination port (write requests, w_3)
- Channel 5 as a destination port (write requests. w_5)

Figure 5–4 shows how these requests are multiplexed in time by the port.

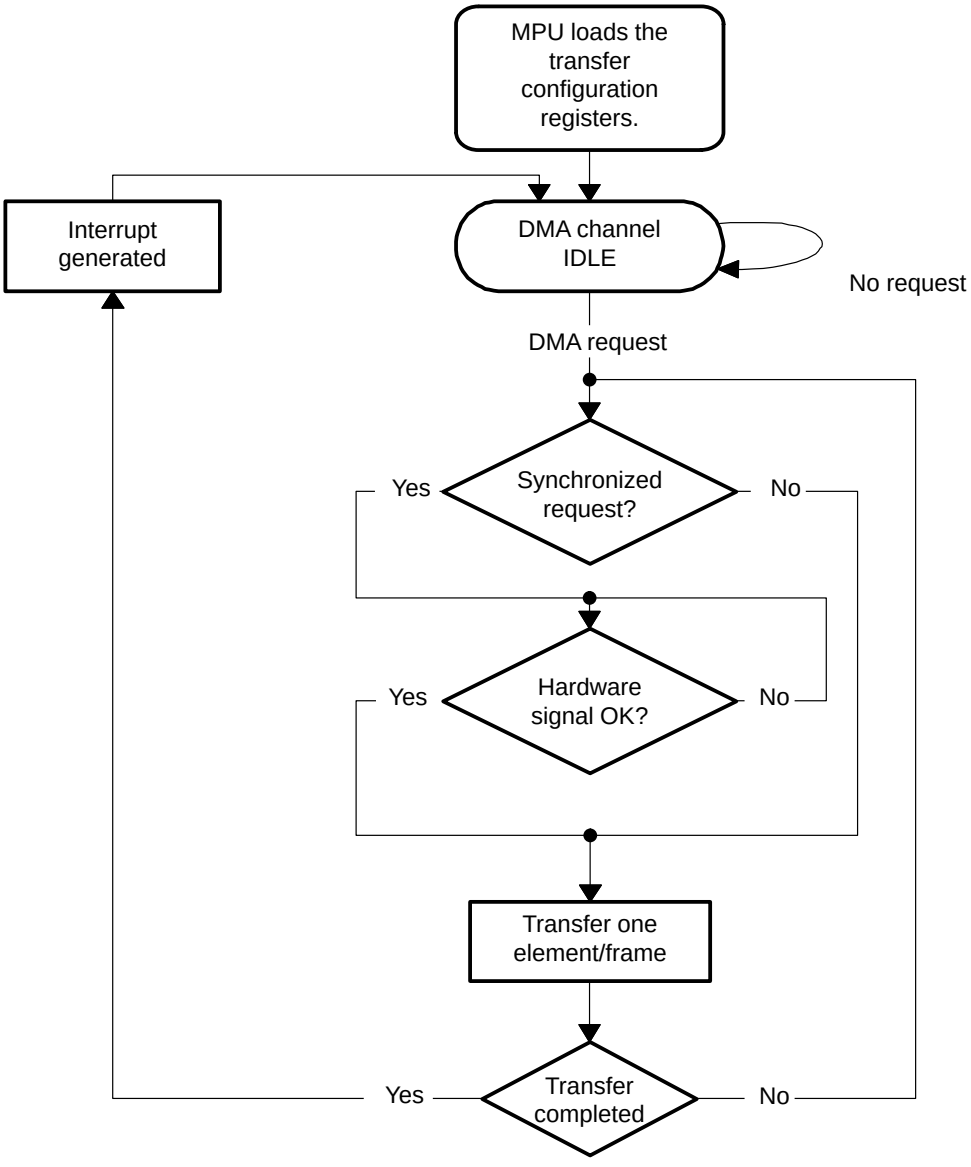
Figure 5–4. Time-Sharing on a DMA Port



A system DMA port can handle 10 read requests and 10 write requests (all the possible requests in the DMA) simultaneously.

5.3.1.2 Transfer Control

Figure 5–5. Basic Flow of DMA Transfer



5.3.1.3 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 starts.
- ❏ Hardware start (hardware request): The processor sets up a DMA channel and sets the transfer in this channel as a synchronized transfer (demand-driven by DMA request signal lines from the outside world). The channel then waits for a DMA request to start transferring data.

Note:

Be sure to enable the channel before sending in the DMA request.

Each time a DMA request is made for this channel, an amount of data is transferred. This amount of data can be:

- An element

  A complete element is transferred in response to a DMA request.
- An entire frame

  A complete frame of several elements is transferred in response to a DMA request.

All the transfers can be synchronized on DMA requests, regardless of their sources and destinations. DMA requests can come from DMA ports. Each channel can be triggered by one DMA request among 31. One DMA request can trigger several channels at the same time. The relevant bits are extracted from the DMA_CCR register.

5.3.1.4 Transfer Suspension

A synchronized channel enters a suspended state when a DMA request is served, and it waits for a new DMA request.

All the DMA channels enter a suspended state if:

- ❏ The DMA idle input is active. The DMA suspends all its transfers and its clock can be externally cut off for power-saving purposes.
- ❏ The DMA suspend input is active, and the FREE bit in the DMA_GCR register is equal to zero. The processor asserts this input, then it is halted by a breakpoint. When this occurs, the DMA suspends or continues its transfers, according to the value of the FREE bit. The DMA clock must be on when the DMA is suspended.

5.3.1.5 Autoinitialization

A DMA channel (synchronized or not) can operate in two modes.

- Single transfer mode

  In this mode, a channel stops when the current transfer finishes.

- Autoinitialization mode

  In this mode, a 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: programming and active. Only the programming set is accessible through the TIPB. 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. You can then program the programming set to configure the next transfer while the current transfer is running.

  This feature can be used in two ways:

  - Continuous operation

    You can change the programming registers while the current configuration is executed. The next transfer is transferred with a new context but without stopping the DMA.

  - Repetitive operation

    You 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. They always have impact on the current transfer.

- DMA_CSDP
- DMA_CCR
- DMA_CICR
- DMA_CSR

To avoid the reload of a configuration when the MPU programs the channel, the MPU can use the autoinitialization bits of the DMA_CCR register, which are described in Table 5–3.

Table 5–3. Autoinitialization Configuration Bits Summary

AUTOINIT	END_PROG	REPEAT	Autoinitialization Behavior
0	x	x	No autoinitialization
1	0	0	At the end of current transfer, channel waits until END_PROG = 1 to load the programming register set in 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	

5.3.1.6 Priorities Between Channels

Each channel can be given a low- or high-priority level. When a DMA port receives requests from several channels, it looks at their priorities:

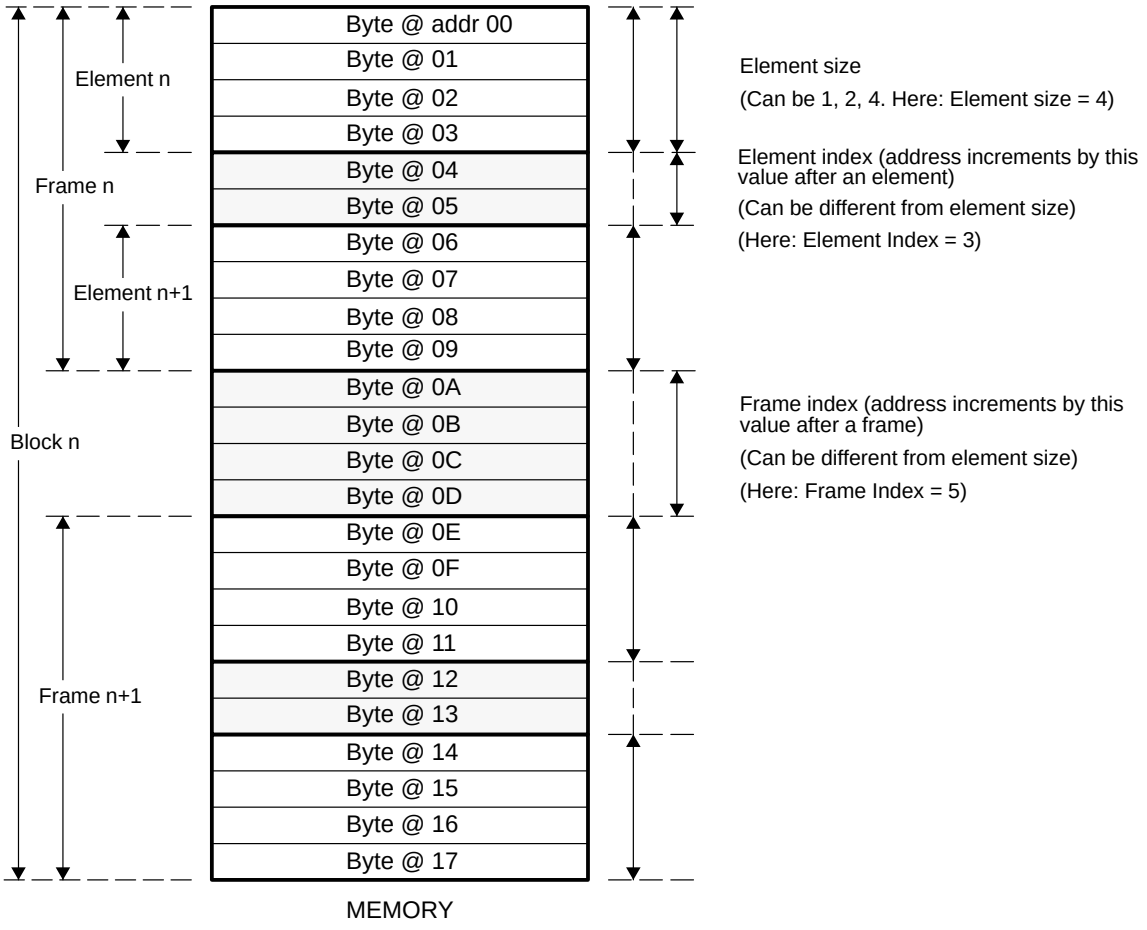
- ❏ Requests from high-priority channels are served first.
- ❏ Requests from low-priority channels are served only if there are no requests from high-priority channels on the port. This can occur if there are no high-priority channels activated, if the high-priority channels are stalled (by a slow source or destination), or if the high-priority channels are waiting for a synchronization event.
- ❏ Requests of the same priority level are served in a round-robin manner (time division multiplex).

5.3.2 Addressing Modes

Figure 5–6 provides an example of address index management. In this example, Element Size = 4, Element Index = 3, Frame Size = 2, and Frame Index = 5.

An addressing mode is an address computation algorithm a DMA channel can use to know where to access data. The system DMA has four addressing modes: constant, post-incremented, indexed, and double-indexed.

Figure 5–6. Memory Representation



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—in other words, non-word aligned (this is true only if bursting is not enabled, because any DMA channel that has bursting enabled must use a start address that is 32-bit word aligned. Furthermore, a DMA channel using bursting must be configured such that all addresses accessed by that channel are also 32-bit word aligned). The data block to transfer is split in 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, $1 \leq FN \leq 65535$.

EN is the number of elements per frame, $1 \leq EN \leq 65535$.

ES is the number of bytes per element, $ES \in \{1, 2, 4\}$.

An element can be:

- 8-bit scalar data, s8
- 16-bit scalar data, s16
- 32-bit scalar data, s32

Types of data transferred in a channel include s8, s16, and s32. 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 mode
- Destination addressing mode

These 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.

5.3.2.1 Constant Addressing Mode

Address remains constant.

$a(i) = SA, 0 \leq i \leq BS - 1$

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.

5.3.2.2 Post-Incremented Addressing Mode

Address is always incremented by 1.

$a(0) = SA$

$a(i) = a(i - 1) + 1,\ 1 \leq i \leq BS - 1$

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.

5.3.2.3 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 $(i \bmod ES) \neq 0,\ 1 \leq i \leq BS - 1$

$a(i) = a(i - 1) + EI$ if $(i \bmod ES) = 0,\ 1 \leq i \leq BS - 1$

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, $1 \leq ES \leq 2$.

EI is the element index in bytes, specified in a configuration register, $-32768 \leq EI \leq 32767$.

5.3.2.4 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) ≠ 0, 1 ≤ i ≤ BS – 1

a(i) = a(i – 1) + EI if (i mod ES) = 0 and (i mod FS) ≠ 0, 1 ≤ i ≤ BS – 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

5.3.3 Data Packing and Bursting

A DMA channel has the capacity to:

- Pack several consecutive byte accesses in a single word16 (16-bit word), word32 (32-bit word), burst4 (burst of four 32-bit words), or burst8 (burst of eight words) access (only the LCD channel can perform burst8 accesses). This increases the transfer rate. For a channel, the decision to pack or burst accesses to its source port is made by the source address unit and depends on source port access capability. The decision to pack and/or burst accesses to its destination port is made by the destination address unit and depends on destination port access capability. Packing and bursting are performed only if the software allows it via proper configuration of the DST_PACK, SRC_PACK, DST_BURST_EN, and SRC_BURST_EN fields in the appropriate DMA_CSDP register.
- Split a single word transfer into several byte accesses. This is done if the DMA port data size is less than the size (or does not match the type) of the data moved.

Table 5–4 summarizes the possible transfer configurations and shows the cases where packing and splitting are performed.

Table 5–4. Packing and Splitting Summary

Data Type	Port Access Capability	Packing/Splitting
s8	8	–
	16	pack 2 x s8 => 16
	32	pack 4 x s8 => 32
s16	8	split s16 => 2 x 8
	16	–
	32	pack 2x s16 => 32
s32	8	split s32 => 4 x 8
	16	split s32 => 2 x 16
	32	–

To compute the type of an access (8-, 16-, 32-bit or burst) and decide whether or not to pack consecutive accesses, an address unit checks:

- Its related DMA port capabilities
  - Can the port perform byte, 16-bit, 32-bit accesses?
  - Can the port perform burst accesses?
- What is allowed by the software in the configuration registers:
  - Is packing allowed (DST_PACK or SRC_PACK set)?
  - Is bursting allowed (DST_BURST_EN or SRC_BURST_EN set)?
- The last bits of the address:
  - Is the address even or odd?
  - Is the address word16, word32, burst4, burst8 aligned?
- The number of elements remaining in the frame.

When the type of access is determined, the current byte address can be incremented by 1, 2, or 4 to reach the next memory space location to access. Then, the DMA port checks the channel FIFO to see if there is enough data (write access) or enough space (read access) in the FIFO before issuing the access.

Not every DMA port has the capability to support every data type (s8, s16, s32) and transfer size. Therefore, software must carefully set up all DMA transfers according to the constrains detailed in Table 5–2.

A transfer to/from a DMA port with 32-bit only access capability must be set up as follows:

- ❏ Element size is a multiple of four.
- ❏ Start address is aligned on a 32-bit word.
- ❏ If frame index is used, it must always produce addresses aligned on a 32-bit word.
- ❏ If element index is used, it must always produce addresses aligned on a 32-bit word.

A transfer to/from a DMA port with 16-bit only access capability must be set up as follows:

- ❏ Element size is a multiple of two.
- ❏ Start address is aligned on a 16-bit word.
- ❏ If frame index is used, it must always produce addresses aligned on a 16-bit word.
- ❏ If element index is used, it must always produce addresses aligned on a 16-bit word.
- ❏ Example 5–1 provides an example of packing and splitting.

Example 5–1. 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.

Memory block to transfer is as identified in Table 5–5 (element i, j is the element number j of frame i).

Table 5–5. 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 as identified in Table 5–6.

Table 5–6. 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		

5.3.4 Data/Address 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).
- ❏ If bursting is enabled, addresses must be aligned on a four-word burst boundary (128 bits) regardless of data type (the four least-significant bits of the address are always 0000).

When using the indexed addressing modes (element index and/or frame index), all the addresses computed must be aligned on the data type.

Failure to adhere to these address alignment requirements could yield unexpected results. In the case of the four-word bursting alignment, failure to properly align the addresses could result in a lockup condition on the DMA channel. To accomplish proper alignment, programming of the start address, block size, frame size, and all indexes must be such that the address of every DMA access is properly aligned (for a burst, this would mean the first access of the burst).

5.3.5 Constraint on Channel Configuration Parameters

Verifying this constraint ensures correct DMA operations when transferring data between ports with different access capabilities as follows:

[SA modulo]/ES = 0,

Where SA is source address and ES is the element size.

5.3.6 Endianism

The endianism of a bus or a memory designates the way data is stored on this bus or in this memory. There are two types of endianism:

- Big endian

  The MSB of the word is stored at the least address in the memory for a bus. The MSB of the word is placed on the MSB of the bus.

- Little endian:

  The MSB of the word is stored at the highest address in the memory for a bus. The MSB of the word is placed on the LSB of the bus.

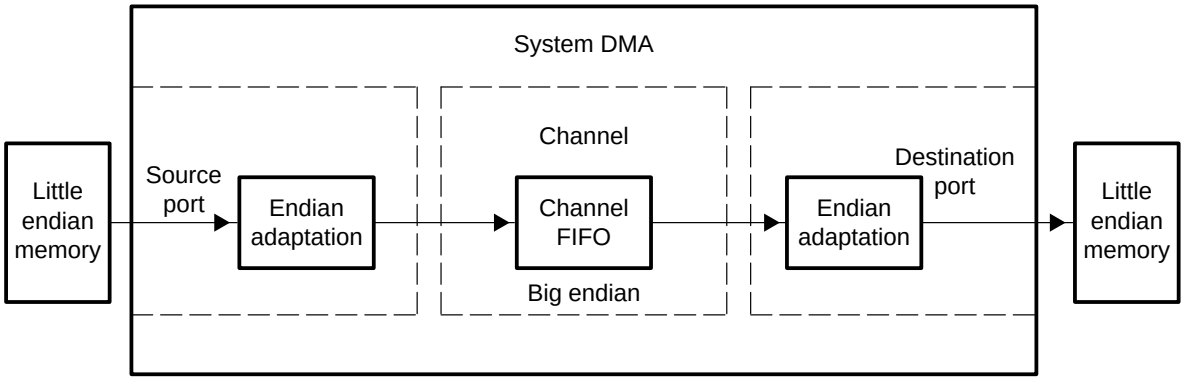
The internal FIFOs of the system DMA are big endian. All system DMA ports are treated as little endian on OMAP5910. Therefore, the system DMA contains adaptation logic to convert incoming data to big endian and outgoing data back to little endian. This adaptation logic is static, can not be configured, and is transparent to system DMA operation.

Additional logic (unrelated to the system DMA) is available to perform endianism conversion between the DSP and the MPU at the DSP MMU and the MPUI interfaces. These are described in section 2.11, *Endianism Conversion*.

The DMA internal FIFOs are big endian.

This adaptation is static and always the same for a given DMA application. Each DMA port contains an endianism adaptation module.

Figure 5–7. Endianism Adaptation on Transferred Data



5.3.7 Interrupt Generation

Each DMA physical channel can generate an interrupt to the processor to reflect the transfer status. Each DMA physical channel has a dedicated interrupt line to the processor. All the DMA interrupts are level interrupts.

For every DMA logical channel, the following interrupt sources can be programmed:

- ❏ End of block: The last byte of the transfer has been written in destination.
- ❏ End of frame: The last byte of the current frame has been written in destination.
- ❏ Half of frame: The middle byte of the current frame has been written in destination.
- ❏ Start of last frame: The first word of the last frame has been written in destination.
- ❏ DMA request collision: A new DMA request occurred before the end of service of the previous one.
- ❏ Time-out: An access has been timed out.

  To prevent a definitive lock by a channel on a memory location or peripheral, all the DMA ports to memory/peripheral requests are monitored by a time-out counter in the following sequence:

  1) When the request is sent by the DMA to transfer data in a channel, a time-out counter is triggered.
  2) The request is acknowledged, and the time-out counter is stopped.
  3) 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.

  The time-out information is generated in the resources accessed by the DMA:

  - System IMIF/local bus port: Time-out is signaled by the IMIF or the local bus.
  - System TI peripheral bus port: Time-out is signaled by the TIPB bridge.
  - System EMIFS/EMIFF port: Time-out is signaled by the EMIFS/ EMIFF (or traffic controller).

The system DMA has nine physical channels; each has the capability to generate interrupts. The DMA has seven interrupt lines, some of which are shared by two physical channels. Each of these seven interrupt lines is routed as an interrupt input on the MPU level2 interrupt handler.

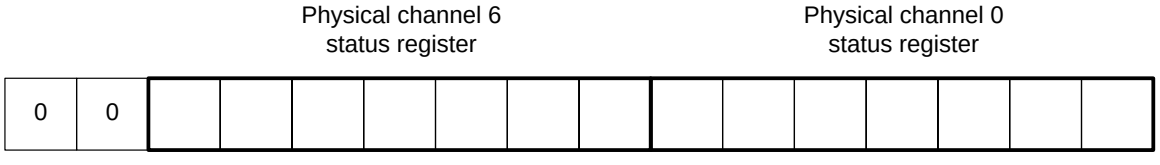
For a physical channel, all the sources are ORed together to generate one interrupt. When an interrupt is issued by a physical channel, its status register (DMA_CSR) is set to record the interrupt cause. The processor interrupt service routine (ISR) can read this channel status register to find the source of the interrupt. The status bits are automatically cleared after they are read. One read in the status register clears all the status bits.

- Interrupt line 0 (MPU level2 IRQ19) is shared by channel 0 and 6.
- Interrupt line 1 (MPU level2 IRQ20) is shared by channel 1 and 7.
- Interrupt line 2 (MPU level2 IRQ21) is shared by channel 2 and 8.
- Interrupt line 3 (MPU level2 IRQ22) is dedicated to channel 3.
- Interrupt line 4 (MPU level2 IRQ23) is dedicated to channel 4.
- Interrupt line 5 (MPU level2 IRQ24) is dedicated to channel 5.
- Interrupt line 6 (MPU level2 IRQ25) is dedicated to the LCD channel.

If simultaneous events occur 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 7-bit status register. When an interrupt is shared by two physical channels, the MPU can read the status from the two channels in one TIPB access. Figure 5–8 shows the data read format for two shared physical channels.

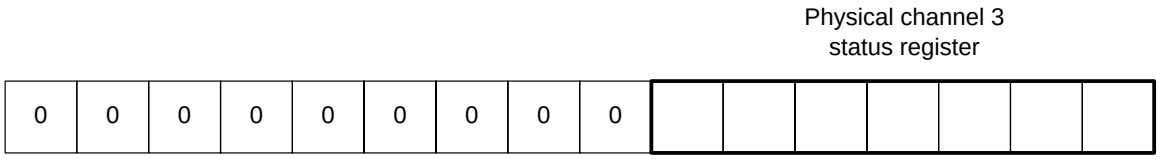
Figure 5–8. Data Read Format—Two Shared Physical Channels



This unique status is accessible either at channel 6 DMA_CSR, or at channel 0 DMA_CSR. Any MPU read (at channel 0 DMA_CSR address or at channel 1 DMA_CSR address) clears all status for the two channels.

When an interrupt is dedicated to 1 physical channel, the MPU can read the status from this channel in one TIPB access. Figure 5–9 shows the data read format for one physical channels.

Figure 5–9. Data Read Format—One Physical Channel



5.3.8 Memory Space Protection

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 = EMIFF with an EMIFS start address specified), the transfer continues and memory can be corrupted. No address space check is performed by the DMA or outside the DMA.

CAUTION

It is the programmer's responsibility to ensure coherency between the source port and source start address, and between the destination port and destination start address.

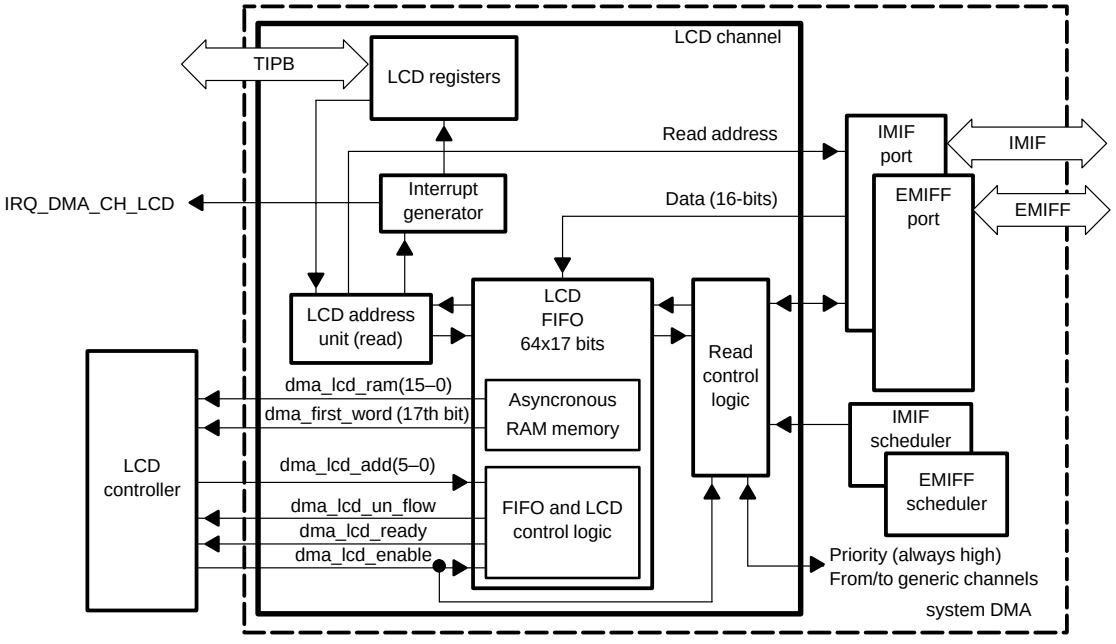
5.4 LCD Dedicated Channel

The LCD channel transfers 16-bit data to the LCD controller from a video frame buffer stored in memory.

5.4.1 Functional Description

The memory source for the LCD dedicated transfer can be either IMIF or EMIFF. These transfers can be arranged to one or two frames but they are always done in frame addressing mode. There is no capability for various addressing modes as in the other channels. The dual-frame mode allows concurrent transfer and image processing (reload of one frame while another is being processed).

Figure 5–10. LCD Channel



Switching from one frame to another is achieved by loading the top address of the second frame buffer after the first frame buffer has been fully transferred. The LCD channel sends the read request on the relevant port according to the LCD_SOURCE bit.

The LCD_SOURCE bit is read from the DMA_LCD_CTRL register. The dedicated DMA channel contains a 64 by 17-bit words FIFO (asynchronous RAM built-in). The dma_lcd_first_word that is used as frame synchronizer by the LCD controller is the 17th FIFO bit output. To ensure correct throughput from

the DMA to the LCD FIFO, the maximum burst length is set to eight. Only near frame boundaries, in case of nonmultiple frames, are single transfers started; otherwise, all requests to the source memory are in 8x16 burst requests. The LCD channel priority bit is fixed high (hard-coded LCD channel constant parameter).

The configuration registers contain top and bottom address registers for the two frame buffers and a control register that manages the operation mode (dual or single), the enable bits for the interrupts, and the source port for the next transfer. It then returns information by setting the status bits in the same control register. An interrupt can be sent at the end of the transfer of each frame. This interrupt line is connected to the global nIRQ line of the DMA.

5.4.2 Addressing Units

The LCD channel does not contain the same (read) address unit as the generic channels. This is because the addressing modes used in the LCD channel are not compliant with the generic modes. The generic channels receive two instances of the address unit. The LCD channel does not have the write address unit instantiated because there is no write address to compute; that is, the read FIFO address is given by the OMAP LCD controller, so there is no write address.

5.4.2.1 LCD Addressing

At the beginning of LCD operation, the LCD channel gives a start address, which is conventionally called the top address of the current frame buffer. This address is sent to the relevant memory port via the scheduler. LCD DMA requests to the memory port are time-multiplexed along with requests from generic channels, as described in section 5.3.1, *Transfers*.

a(0) = SA

a(next) = a(current) + 2

a(next) = TF1 if (a(next) = BF2 and DFM) or (a(next) = BF1 and not (DFM))

a(next) = TF2 if a(next) = BF1 and DFM

where:

a(0) is the first address to be computed.

SA is the start address of the transfer, which is always the top address given for the first frame buffer.

a(current) is the current address of the byte number within the transfer.

a(next) is the next address to be computed.

BF1 is bottom address for frame 1.

BF2 is bottom address for frame 2.

TF1 is top address for frame 1.

TF2 is top address for frame 2.

DFM is the dual-frame mode.

In other words, the next address is always the current address + 2 unless the frame boundaries (inclusive) have been reached (address is a byte address; it is necessary to increment by two, because all LCD transfers are 16 bits). In this case, the next address computed is the top address for the frame 1 if in single frame mode; otherwise, the top address for frame 2 is loaded.

5.4.3 LCD Channel Usage Restrictions

5.4.3.1 Exclusive Frames

The hardware design does not support inclusion of a frame buffer into another frame buffer; that is, the start and stop address of each buffer must represent two different physical parts into the memory. In dual-frame mode the top address for the second frame must be greater (and not equal) than the bottom address of the first frame.

5.4.3.2 Both Frames Must Belong to a Single Source

In case of dual-frame operation it is not possible to have one frame read from one source and one frame read from second source. For changing from one source to another, the LCDEN bit of the LCD control register (see Section 11.8, *LCD Controller Registers*) must be cleared to 0 and all pending LCD interrupts processed. The LCDEN bit level is connected to the dma_lcd_en input of the DMA LCD channel module as pictured in Figure 5–10.

5.4.3.3 LCD Registers Must Remain Steady From One Transfer to Another

It is not possible to change any bit of the LCD channel registers until a transfer has been fully completed. No shadow registers exist in the LCD channel as in generic channels. Changing bits before transfer completion has not been tested; results of doing so are unknown. To update registers, the LCDEN bit should be cleared to 0 and all pending LCD interrupts processed.

5.4.3.4 FIFO Out of Data (Bandwidth Break)

FIFO reads are controlled by a state machine that provides the flow control for the LCD controller. In case of a time-out or bus error while reading the source memory, the state machine detects the error and does not allow read activity

to continue until the LCD enable signal (dma_lcd_en) is disabled one time (LCDEN = 0). This mechanism is provided to avoid having dummy high-priority requests to the ports because the LCD channel's frame data flow has been corrupted. If the LCD controller loses the flow, it should send a software interrupt to cause the MPU to quickly stop the current transfer. The transfer is allowed to run normally after the dma_lcd_en is asserted again (LCDEN = 1).

5.4.4 LCD Transfer Examples

5.4.4.1 EMIFF to LCD, One Frame

Figure 5–11 shows a transfer for a video frame located in EMIFF to the LCD controller. The size for the LCD display is 6x16 pixels with 16 bits per pixel.

So the length of the video frame is:

6 x16 x 2 (in bytes) + 32 bytes for the palette = 224 bytes

If the video frame starts at address 0x0B0000, the bottom address of the video frame is 0x0B00DE.

Registers settings are shown in Table 5–8.

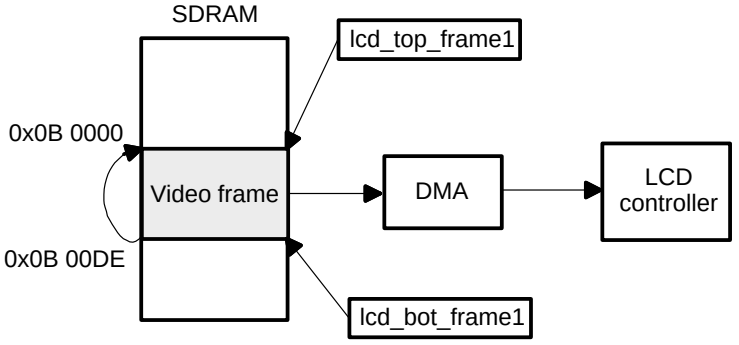
Table 5–7. EMIF to LCD Register Settings—One Frame

DMA_LCD_CTRL	Register Settings
Frame_mode	0 (one frame)
Frame_it_ie	1
Bus_error_ie	1
Lcd_source	0 (SDRAM)
DMA_LCD_TOP_F1_U	0x000B
DMA_LCD_TOP_F1_L	0x0000
DMA_LCD_BOT_F1_U	0x000B
DMA_LCD_BOT_F1_L	0x00DE
DMA_LCD_TOP_F2_U	irrelevant
DMA_LCD_TOP_F2_L	irrelevant
DMA_LCD_BOT_F2_U	irrelevant
DMA_LCD_BOT_F2_L	irrelevant

The transfer starts when the enable (hardware) signal from the LCD controller is asserted high.

The transfer runs, and an interruption is generated at the end of the frame.

Figure 5–11. LCD One Frame Mode Transfer Scheme



When an interrupt occurs, read the DMA_LCD_CTRL register to find the source of the interrupt.

If DMA_LCD_CTRL(3) = 1, end frame 1 interrupt.

If end of frame is reached, the DMA restarts at the top of the frame. Reset DMA_LCD_CTRL(3) and wait for another interrupt.

5.4.4.2 IMIF to LCD, Two Frames

Figure 5–12 shows a transfer from two video frames located in IMIF to the LCD controller. The size for the LCD display is 6 x 16 pixels with 16 bits per pixel. So the length of one video frame is:

6 x 16 x 2 (in bytes) + 32 bytes for the palette = 224 bytes

If the video frame 1 starts at address 0x0B0000, the bottom address of the video frame is 0x0B00DE. If the video frame 2 starts at address 0x0C0000, the bottom address of the video frame is 0x0C00DE.

Registers settings are shown in Table 5–8.

Table 5–8. IMIF LCD Register Settings—Two Frames

DMA_LCD_CTRL	Register Settings
Frame_mode	1 (two frame)
Frame_it_ie	1
Bus_error_ie	1
Lcd_source	1 (IMIF)
DMA_LCD_TOP_F1_U	0x000B

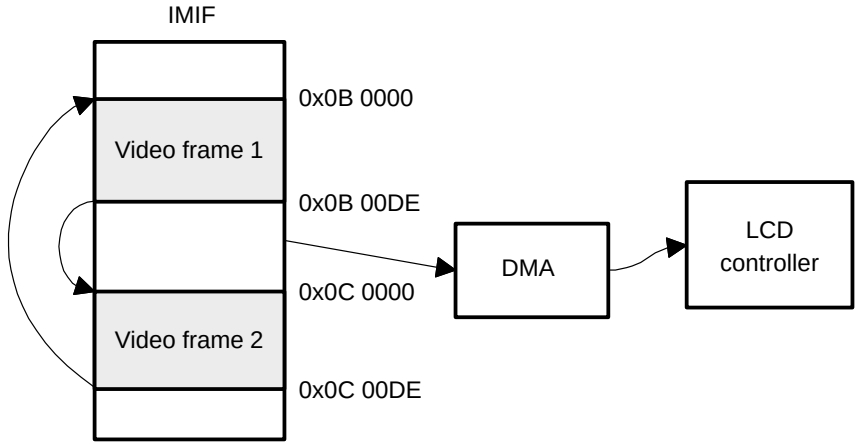
Table 5–8. IMIF LCD Register Settings—Two Frames (Continued)

DMA_LCD_CTRL	Register Settings
DMA_LCD_TOP_F1_L	0x0000
DMA_LCD_BOT_F1_U	0x000B
DMA_LCD_BOT_F1_L	0x00DE
DMA_LCD_TOP_F2_U	0x000C
DMA_LCD_TOP_F2_L	0x0000
DMA_LCD_BOT_F2_U	0x000C
DMA_LCD_BOT_F2_L	0x00DE

The transfer starts when the enable (hardware) signal from the LCD controller is asserted high.

The transfer runs, and the interrupts are generated at the ends of frames 1 and 2.

Figure 5–12. LCD Dual-Frame Mode Transfer Scheme



When an interrupt occurs, read the DMA_LCD_CTRL register to find the source of the interrupt.

If DMA_LCD_CTRL(3) = 1, end frame 1 interrupt.

If DMA_LCD_CTRL(4) = 1, end frame 2 interrupt.

When bottom of frame 1 is reached, the DMA loads the top frame 2 addresses. When bottom of frame 2 is reached, the DMA loads the top frame 1 address.

Reset DMA_LCD_CTRL3 and 4 and wait for another interrupt.

5.5 DMA Request Mapping

Table 5–9 shows the DMA request mapping for the OMAP5910 device.

Table 5–9. DMA Request Mapping

MPU System DMA Requests	MPU System DMA
MCSI1 TX	DMA_REQ_01
MCSI1 RX	DMA_REQ_02
I^2C RX	DMA_REQ_03
I^2C TX	DMA_REQ_04
$\overline{\text{EXT\_DMA\_REQ0}}$ (MPUIO2)	DMA_REQ_05
$\overline{\text{EXT\_DMA\_REQ1}}$ (MPUIO4)	DMA_REQ_06
MicroWire TX	DMA_REQ_07
McBSP1 TX	DMA_REQ_08
McBSP1 RX	DMA_REQ_09
McBSP3 TX	DMA_REQ_10
McBSP3 RX	DMA_REQ_011
UART1 TX	DMA_REQ_012
UART1 RX	DMA_REQ_013
UART2 TX	DMA_REQ_014
UART2 RX	DMA_REQ_015
McBSP2 TX	DMA_REQ_016
McBSP2 RX	DMA_REQ_017
UART3 TX	DMA_REQ_018
UART3 RX	DMA_REQ_019
Camera RX	DMA_REQ_020
MMC TX	DMA_REQ_021
MMC RX	DMA_REQ_022
Reserved	DMA_REQ_023
Reserved	DMA_REQ_024

Table 5–9. DMA Request Mapping (Continued)

MPU System DMA Requests	MPU System DMA
Reserved	DMA_REQ_025
USB function RX0	DMA_REQ_026
USB function RX1	DMA_REQ_027
USB function RX2	DMA_REQ_028
USB function TX0	DMA_REQ_029
USB function TX1	DMA_REQ_030
USB function TX2	DMA_REQ_031

5.6 Registers

Table 5–10 describes the DMA controller registers.

Note:

The DMA control registers are part of a register superset for multiple OMAP-based devices. They are defined for a 16-port, 16-channel DMA controller. Thus as generic as possible a register mapping is provided, so some registers may appear to be almost empty. Only the 16 LSBs are used; in fact, the DMA registers must always be accessed as 16-bit registers.

Base address for system DMA: FFFE–D800

Table 5–10. DMA Controller Registers

Name	Description	R/W	Size (Bits)	Address	Reset Value
DMA_GCR	Global control	R/W	16	0xFFFEDC00	0x0008
DMA_CSDP_CH0	Channel 0 source destination parameters	R/W	16	0xFFFED800	0x0000
DMA_CCR_CH0	Channel 0 control	R/W	16	0xFFFED802	0x0000
DMA_CICR_CH0	Channel 0 interrupt control	R/W	16	0xFFFED804	0x0003
DMA_CSR_CH0	Channel 0 status	R	16	0xFFFED806	0x0000
DMA_CSSA_L_CH0	Channel 0 source start address—lower bits	R/W	16	0xFFFED808	U
DMA_CSSA_U_CH0	Channel 0 source start address—upper bits	R/W	16	0xFFFED80A	U
DMA_CDSA_L_CH0	Channel 0 destination start address—lower bits	R/W	16	0xFFFED80C	U
DMA_CDSA_U_CH0	Channel 0 destination start address—upper bits	R/W	16	0xFFFED80E	U
DMA_CEN_CH0	Channel 0 element number	R/W	16	0xFFFED810	U
DMA_CFN_CH0	Channel 0 frame number	R/W	16	0xFFFED812	U
DMA_CFI_CH0	Channel 0 frame index	R/W	16	0xFFFED814	U
DMA_CEI_CH0	Channel 0 element index	R/W	16	0xFFFED816	U
DMA_CPC_CH0	Channel 0 channel progress counter	R/W	16	0xFFFED818	U
DMA_CSDP_CH1	Channel 1 source destination parameters	R/W	16	0xFFFED840	0x0000 0000

Table 5–10. DMA Controller Registers (Continued)

Name	Description	R/W	Size (Bits)	Address	Reset Value
DMA_CCR_CH1	Channel 1 control	R/W	16	0xFFFED842	0x0000
DMA_CICR_CH1	Channel 1 interrupt control	R/W	16	0xFFFED844	0x0003
DMA_CSR_CH1	Channel 1 status	R	16	0xFFFED846	0x0000
DMA_CSSA_L_CH1	Channel 1 source start address lower bits	R/W	16	0xFFFED848	U
DMA_CSSA_U_CH1	Channel 1 source start address upper bits	R/W	16	0xFFFED84A	U
DMA_CDSA_L_CH1	Channel 1 destination start address lower bits	R/W	16	0xFFFED84C	U
DMA_CDSA_U_CH1	Channel 1 destination start address upper bits	R/W	16	0xFFFED84E	U
DMA_CEN_CH1	Channel 1 element number	R/W	16	0xFFFED850	U
DMA_CFN_CH1	Channel 1 frame number	R/W	16	0xFFFED852	U
DMA_CFI_CH1	Channel 1 frame index	R/W	16	0xFFFED854	U
DMA_CEI_CH1	Channel 1 element index	R/W	16	0xFFFED856	U
DMA_CPC_CH1	Channel 1 channel progress counter	R/W	16	0xFFFED858	U
DMA_CSDP_CH2	Channel 2 source destination parameters	R/W	16	0xFFFED880	0x0000
DMA_CCR_CH2	Channel 2 control	R/W	16	0xFFFED882	0x0000
DMA_CICR_CH2	Channel 2 interrupt control	R/W	16	0xFFFED884	0x0003
DMA_CSR_CH2	Channel 2 status	R	16	0xFFFED886	0x0000
DMA_CSSA_L_CH2	Channel 2 source start address lower bits	R/W	16	0xFFFED888	U
DMA_CSSA_U_CH2	Channel 2 source start address upper bits	R/W	16	0xFFFED88A	U
DMA_CDSA_L_CH2	Channel 2 destination start address lower bits	R/W	16	0xFFFED88C	U
DMA_CDSA_U_CH2	Channel 2 destination start address upper bits	R/W	16	0xFFFED88E	U
DMA_CEN_CH2	Channel 2 element number	R/W	16	0xFFFED890	U

Table 5–10. DMA Controller Registers (Continued)

Name	Description	R/W	Size (Bits)	Address	Reset Value
DMA_CFN_CH2	Channel 2 frame number	R/W	16	0xFFFED892	U
DMA_CFI_CH2	Channel 2 frame index	R/W	16	0xFFFED894	U
DMA_CEI_CH2	Channel 2 element index	R/W	16	0xFFFED896	U
DMA_CPC_CH2	Channel 2 channel progress counter	R/W	16	0xFFFED898	U
DMA_CSDP_CH3	Channel 3 source destination parameters	R/W	16	0xFFFED8C0	0x0000
DMA_CCR_CH3	Channel 3 control	R/W	16	0xFFFED8C2	0x0000
DMA_CICR_CH3	Channel 3 interrupt control	R/W	16	0xFFFED8C4	0x0003
DMA_CSR_CH3	Channel 3 status	R	16	0xFFFED8C6	0x0000
DMA_CSSA_L_CH3	Channel 3 source start address lower bits	R/W	16	0xFFFED8C8	U
DMA_CSSA_U_CH3	Channel 3 source start address upper bits	R/W	16	0xFFFED8CA	U
DMA_CDSA_L_CH3	Channel 3 destination start address lower bits	R/W	16	0xFFFED8CC	U
DMA_CDSA_U_CH3	Channel 3 destination start address upper bits	R/W	16	0xFFFED8CE	U
DMA_CEN_CH3	Channel 3 element number	R/W	16	0xFFFED8D0	U
DMA_CFN_CH3	Channel 3 frame number	R/W	16	0xFFFED8D2	U
DMA_CFI_CH3	Channel 3 frame index	R/W	16	0xFFFED8D4	U
DMA_CEI_CH3	Channel 3 element index	R/W	16	0xFFFED8D6	U
DMA_CPC_CH3	Channel 3 channel progress counter	R/W	16	0xFFFED8D8	U
DMA_CSDP_CH4	Channel 4 source destination parameters	R/W	16	0xFFFED900	0x0000
DMA_CCR_CH4	Channel 4 control	R/W	16	0xFFFED902	0x0000
DMA_CICR_CH4	Channel 4 interrupt control	R/W	16	0xFFFED904	0x0003
DMA_CSR_CH4	Channel 4 status	R	16	0xFFFED906	0x0000
DMA_CSSA_L_CH4	Channel 4 source start address lower bits	R/W	16	0xFFFED908	U

Table 5–10. DMA Controller Registers (Continued)

Name	Description	R/W	Size (Bits)	Address	Reset Value
DMA_CSSA_U_CH4	Channel 4 source start address upper bits	R/W	16	0xFFFED90A	U
DMA_CDSA_L_CH4	Channel 4 destination start address lower bits	R/W	16	0xFFFED90C	U
DMA_CDSA_U_CH4	Channel 4 destination start address upper bit	R/W	16	0xFFFED90E	U
DMA_CEN_CH4	Channel 4 element number	R/W	16	0xFFFED910	U
DMA_CFN_CH4	Channel 4 frame number	R/W	16	0xFFFED912	U
DMA_CFI_CH4	Channel 4 frame index	R/W	16	0xFFFED914	U
DMA_CEI_CH4	Channel 4 element index	R/W	16	0xFFFED916	U
DMA_CPC_CH4	Channel 4 channel progress counter	R/W	16	0xFFFED918	U
DMA_CSDP_CH5	Channel 5 source destination parameters	R/W	16	0xFFFED940	0x0000
DMA_CCR_CH5	Channel 5 control	R/W	16	0xFFFED942	0x0000
DMA_CICR_CH5	Channel 5 interrupt control	R/W	16	0xFFFED944	0x0003
DMA_CSR_CH5	Channel 5 status	R	16	0xFFFED946	0x0000
DMA_CSSA_L_CH5	Channel 5 source start address lower bits	R/W	16	0xFFFED948	U
DMA_CSSA_U_CH5	Channel 5 source start address upper bits	R/W	16	0xFFFED94A	U
DMA_CDSA_L_CH5	Channel 5 destination start address lower bits	R/W	16	0xFFFED94C	U
DMA_CDSA_U_CH5	Channel 5 destination start address upper bits	R/W	16	0xFFFED94E	U
DMA_CEN_CH5	Channel 5 element number	R/W	16	0xFFFED950	U
DMA_CFN_CH5	Channel 5 frame number	R/W	16	0xFFFED952	U
DMA_CFI_CH5	Channel 5 frame index	R/W	16	0xFFFED954	U
DMA_CEI_CH5	Channel 5 element index	R/W	16	0xFFFED956	U
DMA_CPC_CH5	Channel 5 channel progress counter	R/W	16	0xFFFED958	U

Table 5–10. DMA Controller Registers (Continued)

Name	Description	R/W	Size (Bits)	Address	Reset Value
DMA_CSDP_CH6	Channel 6 source destination parameters	R/W	16	0xFFFED980	0x0000
DMA_CCR_CH6	Channel 6 control	R/W	16	0xFFFED982	0x0000
DMA_CICR_CH6	Channel 6 interrupt control	R/W	16	0xFFFED984	0x0003
DMA_CSR_CH6	Channel 6 status	R	16	0xFFFED986	0x0000
DMA_CSSA_L_CH6	Channel 6 source start address lower bit	R/W	16	0xFFFED988	U
DMA_CSSA_U_CH6	Channel 6 source start address upper bits	R/W	16	0xFFFED98A	U
DMA_CDSA_L_CH6	Channel 6 destination start address lower bits	R/W	16	0xFFFED98C	U
DMA_CDSA_U_CH6	Channel 6 destination start address upper bits	R/W	16	0xFFFED98E	U
DMA_CEN_CH6	Channel 6 element number	R/W	16	0xFFFED990	U
DMA_CFN_CH6	Channel 6 frame number	R/W	16	0xFFFED992	U
DMA_CFI_CH6	Channel 6 frame index	R/W	16	0xFFFED994	U
DMA_CEI_CH6	Channel 6 element index	R/W	16	0xFFFED996	U
DMA_CPC_CH6	Channel 6 channel progress counter	R/W	16	0xFFFED998	U
DMA_CSDP_CH7	Channel 7 source destination parameters	R/W	16	0xFFFED9C0	0x0000
DMA_CCR_CH7	Channel 7 control	R/W	16	0xFFFED9C2	0x0000
DMA_CICR_CH7	Channel 7 interrupt control	R/W	16	0xFFFED9C4	0x0003
DMA_CSR_CH7	Channel 7 status	R	16	0xFFFED9C6	0x0000
DMA_CSSA_L_CH7	Channel 7 source start address lower bits	R/W	16	0xFFFED9C8	U
DMA_CSSA_U_CH7	Channel 7 source start address upper bits	R/W	16	0xFFFED9CA	U
DMA_CDSA_L_CH7	Channel 7 destination start address lower bits	R/W	16	0xFFFED9CC	U
DMA_CDSA_U_CH7	Channel 7 destination start address lower bits	R/W	16	0xFFFED9CE	U
DMA_CEN_CH7	Channel 7 element number	R/W	16	0xFFFED9D0	U
DMA_CFN_CH7	Channel 7 frame number	R/W	16	0xFFFED9D2	U

Table 5–10. DMA Controller Registers (Continued)

Name	Description	R/W	Size (Bits)	Address	Reset Value
DMA_CFI_CH7	Channel 7 frame index	R/W	16	0xFFFED9D4	U
DMA_CEI_CH7	Channel 7 element frame	R/W	16	0xFFFED9D6	U
DMA_CPC_CH7	Channel 7 channel progress counter	R/W	16	0xFFFED9D8	U
DMA_CSDP_CH8	Channel 8 source destination parameters	R/W	16	0xFFFEDA00	0x0000
DMA_CCR_CH8	Channel 8 control	R/W	16	0xFFFEDA02	0x0000
DMA_CICR_CH8	Channel 8 interrupt control	R/W	16	0xFFFEDA04	0x0003
DMA_CSR_CH8	Channel 8 status	R	16	0xFFFEDA06	0x0000
DMA_CSSA_L_CH8	Channel 8 source start address lower bits	R/W	16	0xFFFEDA08	U
DMA_CSSA_U_CH8	Channel 8 source start address upper bits	R/W	16	0xFFFEDA0A	U
DMA_CDSA_L_CH8	Channel 8 destination start address lower bits	R/W	16	0xFFFEDA0C	U
DMA_CDSA_U_CH8	Channel 8 destination start address upper bits	R/W	16	0xFFFEDA0E	U
DMA_CEN_CH8	Channel 8 element number	R/W	16	0xFFFEDA10	U
DMA_CFN_CH8	Channel 8 frame number	R/W	16	0xFFFEDA12	U
DMA_CFI_CH8	Channel 8 frame index	R/W	16	0xFFFEDA14	U
DMA_CEI_CH8	Channel 8 element index	R/W	16	0xFFFEDA16	U
DMA_CPC_CH8	Channel 8 channel progress counter	R/W	16	0xFFFEDA18	U
DMA_LCD_CTRL	LCD control	R/W	16	0xFFFEDB00	0x0000
DMA_LCD_TOP_F1_L	LCD top address for frame buffer 1 lower bits	R/W	16	0xFFFEDB02	U
DMA_LCD_TOP_F1_U	LCD top address for frame buffer 1 upper bits	R/W	16	0xFFFEDB04	U
DMA_LCD_BOT_F1_L	LCD bottom address for frame buffer 1 lower bits	R/W	16	0xFFFEDB06	U
DMA_LCD_BOT_F1_U	LCD bottom address for frame buffer 1 upper bits	R/W	16	0xFFFEDB08	U

Table 5–10. DMA Controller Registers (Continued)

Name	Description	R/W	Size (Bits)	Address	Reset Value
DMA_LCD_TOP_F2_L	LCD top address for frame buffer 2 lower bits	R/W	16	0xFFFEDB0A	U
DMA_LCD_TOP_F2_U	LCD top address for frame buffer 2 upper bits	R/W	16	0xFFFEDB0C	U
DMA_LCD_BOT_F2_L	LCD bottom address for frame buffer 2 lower bits	R/W	16	0xFFFEDB0E	U
DMA_LCD_BOT_F2_U	LCD bottom address for frame buffer 2 upper bits	R/W	16	0xFFFEDB10	U

Table 5–11 shows the global control register bit descriptions.

Table 5–11. DMA Global Control register (DMA_GCR)

Bit	Name	Value	Description	Type	Reset Value
15–4	RESERVED				
3	AUTOGATING_ON		DMA clock autogating is as follows:	RW	1
		0	Reserved. Do not use this setting.		
		1	Allows the DMA to dynamically cut off its clocks according to its activity. This bit should always be set to 1.		
2	FREE		DMA reaction to the suspend signal is as follows:	RW	0
		0	The DMA suspends all the current transfers when it receives the suspend signal from the processor. Transfers resume when the processor releases the suspend signal. The DMA clock must not be cut off when the DMA is suspended.		
		1	The DMA continues running when it receives the suspend signal from the processor (when the processor is halted for debug by a breakpoint, for example).		
1–0	RESERVED				

5.6.1 Generic Channel Registers

There is one set of these registers for each generic DMA channel. Although the DMA has a 32-bit TIPB, all registers are in 16-bit format and must be accessed as 16-bit data by the MPU.

Table 5–12. Channel Source Destination Parameters Register (DMA_CSDP)

Bit	Name	Value	Description	Type	Reset Value
15–14	DST_BURST_EN		Destination burst enable	RW	00
			Enable/disable bursting on the destination port. When bursting is enabled, the destination port performs bursts 4 x dst_width. When bursting is disabled, the destination port performs single accesses of dst_width bits.		
		00	Single access (no burst)		
		01	Single access (no burst)		
		10	Burst 4		
		11	Reserved (do not use this setting)		
			If the destination port of the channel has no burst access capability, this field is ignored.		
13	DST_PACK		Destination packing	RW	0
			The DMA ports can have a data bus width different from that of the type of data moved by the DMA channel. For example, s8 data 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 to increase bandwidth.		
		0	The destination port never makes packed accesses.		
		1	The destination port makes packed accesses.		

Table 5–12. Channel Source Destination Parameters Register (DMA_CSDP) (Continued)

Bit	Name	Value	Description	Type	Reset Value
12–9	DST		Transfer destination	RW	0000
			A unique identifier is given to each port. This field indicates which port is the originator of the transfer.		
		0000	EMIFF		
		0001	EMIF		
		0010	IMIF		
		0011	TIPB		
		0100	Local		
		0101	TIPB_MPUI		
			Others: Illegal (do not use this setting)		
8–7	SRC_BURST_EN		Source burst enable	RW	00
			Enable/disable bursting on the source port. When bursting is enabled, the source port performs bursts 4 x src_width. When bursting is disabled, the source port performs single accesses of src_width bits.		
		00	Single access (no burst)		
		01	Single access (no burst)		
		10	Burst 4		
		11	Reserved (do not use this setting)		
			If the source port of the channel has no burst access capability, this field is ignored.		

Table 5–12. Channel Source Destination Parameters Register (DMA_CSDP) (Continued)

Bit	Name	Value	Description	Type	Reset Value
6	SRC_PACK		Source packing	RW	0
			The DMA ports can have a data bus width different from that of the type of data moved by the DMA channel. For example, s8 data 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 to increase bandwidth.		
		0	The source port never makes packed accesses.		
		1	The source port makes packed accesses.		
5–2	SRC		Transfer source	RW	0000
			A unique identifier is given to each port. This field indicates which port is the originator of the transfer.		
		0000	EMIFF		
		0001	EMIF		
		0010	IMIF		
		0011	TIPB		
		0100	Local		
		0101	TIPB_MPUI		
			Others: Illegal (do not use this setting)		

Table 5–12. Channel Source Destination Parameters Register (DMA_CSDP) (Continued)

Bit	Name	Value	Description	Type	Reset Value
1–0	DATA_TYPE		Defines the type of the data moved in the channel	RW	00
		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 type is s32, the source and destination start address must be aligned on a 32-bit word. If type is s8, source and destination start address can have any value. The DMA forces by hardware the start address value on the type of data transferred.		

Table 5–13. DMA Channel Control Register (DMA__CCR)

Bit	Name	Value	Description	Type	Reset Value
15–14	DST_AMODE		Destination addressing mode	RW	00
			This field is used to choose the addressing mode on the destination port of a channel.		
		00	Constant address		
		01	Post-incremented address		
		10	Single index (element index)		
		11	Double index (element index and frame index)		
13–12	SRC_AMODE		Source addressing mode	RW	00
			This field is used to choose the addressing mode on the source port of a channel.		
		00	Constant address		
		01	Post-incremented address		
		10	Single index (element index)		
		11	Double index (element index and frame index)		
11	END_PROG		End of programming	RW-RST	0
		0	Delays the channel autoinitialization if AUTO_INIT = 1 and REPEAT = 0.		
		1	Allows the channel to reinitialize itself if AUTO_INIT = 1		
10	RESERVED			RW-RST	0
9	REPEAT		Repetitive operations	RW	0
		0	When the current transfer is complete, the channel automatically reinitializes itself and starts a new transfer only if END_PROG = 1.		
		1	When the current transfer is complete, the channel automatically reinitializes itself and starts a new transfer disregarding END_PROG.		

Table 5–13. DMA Channel Control Register (DMA__CCR) (Continued)

Bit	Name	Value	Description	Type	Reset Value
8	AUTO_INIT		Autoinitialization at the end of the transfer	RW	0
		0	The channel stops at the end of the current transfer.		
		1	When the current transfer is complete, the channel automatically reinitializes itself and starts a new transfer. There are two ways to stop a channel while it is in autoinitialization mode: ❏ Write a 0 to the DMA_CCR EN bit; the channel immediately stops. ❏ Write a 0 to the DMA_CCR AUTO_INIT bit; the channel completes the current transfer and stops.		
7	EN		Enable This bit is used to enable/disable the transfer in the DMA channel.	RW-RST	0
		0	The transfer stops, and it is reset.		
		1	The transfer starts. This bit is automatically cleared by the DMA once the transfer is accomplished. Clearing of this bit by the DMA has the priority over write by the processor. If both simultaneously occur, the processor write is discarded.		
6	PRIO		Channel priority	RW	0
		0	The channel has low priority level.		
		1	The channel has high priority level.		

Table 5–13. DMA Channel Control Register (DMA__CCR) (Continued)

Bit	Name	Value	Description	Type	Reset Value
5	FS		Frame synchronization	RW	0
			This bit is used to program the way a DMA request is serviced in a synchronized transfer.		
		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.		
		1	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.		
4–0	SYNC		Synchronization control	RW	00000
			This field is used to specify the external DMA request, which can trigger the transfer in this channel. One DMA request among 15 possible can be chosen. The values for this field are defined in Table 5–9.		
		0000	Transfer not synchronized		
		i	Transfer synchronized on DMA request [i], $i \neq 0$ regarding the table described		

Table 5–14. DMA Channel Interrupt Control Register (DMA_CICR)

Bit	Name	Value	Description	Type	Reset Value
15–7	RESERVED				
6	RESERVED			R	0
5	BLOCK_IE		End block interrupt enable	RW	0
		0	The channel does not interrupt the processor when the transfer of the block completes.		
		1	The channel sends an interrupt to the processor when the transfer of the block completes.		
4	LAST_IE		Last frame interrupt enable	RW	0
		0	The channel does not interrupt the processor when the transfer of the last frame starts.		
		1	The channel sends an interrupt to the processor when the transfer of the last frame starts.		
3	FRAME_IE		Frame interrupt enable	RW	0
		0	The channel does not interrupt the processor when the transfer of the current frame completes.		
		1	The channel sends an interrupt to the processor when the transfer of the current frame completes.		
2	HALF_IE		Half frame interrupt enable	RW	0
		0	The channel does not interrupt the processor when the transfer of the first half of the current frame completes.		
		1	The channel sends an interrupt to the processor when the transfer of the first half of the current frame completes.		
1	DROP_IE		Synchronization event drop interrupt enable	RW	1
		0	The channel does not interrupt the processor when a synchronization event drop occurs.		
		1	The channel sends an interrupt to the processor if the channel transfer is synchronized on DMA requests and on successive DMA request drops. This occurs when a new DMA request is made while the service of the previous one is not finished yet.		

Table 5–14. DMA Channel Interrupt Control Register (DMA_CICR) (Continued)

Bit	Name	Value	Description	Type	Reset Value
0	TOUT_IE		Time-out interrupt enable	RW	1
		0	The DMA does not send an interrupt to the processor if a time-out error occurs.		
		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.		

The interrupt enable bits are used to choose the events that cause the DMA channel to send an interrupt to the processor. There are two classes of events:

- ❏ Error events: errors during the transfer (time out, event drop)
- ❏ Status events: 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) or in DMA_TSR (DMA time-out error status register). A status bit is not set if the corresponding interrupt enable bit in DMA_CICR equals 0.

Table 5–15. DMA Channel Status Register (DMA_CSR)

Bit	Name	Value	Description	Type	Reset Value
15–14	RESERVED				
13–7	ALT_STATUS		Alternate status bits for channels with shared interrupts. For DMA channels with shared interrupts, these seven bits have the same function as bits 6–0 of this register, except they correspond to the other channel that shares the interrupt. For example, in register DMA_CSR_CH0, these bits correspond to channel 6 status and mirror the values present in DMA_CSR_CH6[6–0]. DMA_CSR registers for both channel 0 and 6 are cleared when either register is read. For channels without shared interrupts, these bits are reserved.		0000000

Table 5–15. DMA Channel Status Register (DMA_CSR) (Continued)

Bit	Name	Value	Description	Type	Reset Value
6	SYNC		Synchronization status	R	0
			This bit is not set to one when an interrupt is generated, but when a DMA request is made in a synchronized channel. When the DMA request is serviced, the bit returns to zero.		
		0	No DMA request is in service.		
		1	A DMA request was made for this channel when it was in service.		
5	BLOCK		End block	R	0
		0	Current transfer is not finished yet.		
		1	The current transfer in the channel is finished (another one may have start if DMA_CCR2 AUTOINIT = 1).		
4	LAST		Last frame	R	0
		0	Last frame did not start yet.		
		1	The transfer of the last frame has started.		
3	FRAME		Frame	R	0
		0	Transfer of the current frame still in progress		
		1	A complete frame was transferred.		
2	HALF		Half frame	R	0
		0	First half of the current frame not transferred yet		
		1	First half of the current frame was transferred.		
1	DROP		Event drop	R	0
		0	No event drop occurred during the transfer.		
		1	An event drop occurred during the transfer.		
0	TOUT		Time-out in the channel	R	0
		0	No time-out error occurred in channel.		
		1	Time-out occurred in channel.		

This register is written by the DMA to reflect the channel status. It can be read by the processor (by polling or after an interrupt) to see the channel status. After a functional read, all the DMA_CSR bits are automatically cleared. The DMA_CSR bit is not cleared after an emulation read via the debugger. The register bit is only set when its associated DMA_CICR is enabled.

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 ongoing operation of the DMA channel, the ISR must read DMA_CSR after each DMA interrupt.

Table 5–16. DMA Channel Source Start Address–Lower Bits Register (DMA_CSSA_L)

Bit	Name	Description	Type	Reset Value
15–0	Source start address, lower bits	Lower bits of the source start address, expressed in bytes. The source start address output by the DMA is an up-to-32-bit byte address made of the concatenation of DMA_CSSA_U and DMA_CSSA_L.	RW	Undefined

Table 5–17. DMA Channel Source Start Address–Upper Bits Register (DMA_CSSA_U)

Bit	Name	Description	Type	Reset Value
15–0	Source start address, upper bits	Upper bits of the source start address, expressed in bytes. The source start address output by the DMA is a 32-bit byte address made of the concatenation of DMA_CSSA_U and DMA_CSSA_L.	RW	Undefined

Table 5–18. DMA Channel Destination Start Address–Lower Bits Register (DMA_CDSA_L)

Bit	Name	Description	Type	Reset Value
15–0	Destination start address, lower bits	Lower bits for the destination start address, expressed in bytes. The destination start address is up to an up-to-32-bit byte address made of the concatenation of DMA_CDSA_U and DMA_CDSA_L.	RW	Undefined

Table 5–19. DMA Channel Destination Start Address–Upper Bits Register (DMA_CDSA_U)

Bit	Name	Description	Type	Reset Value
15–0	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.	RW	Undefined

Table 5–20. DMA Channel Element Number Register (DMA_CEN)

Bit	Name	Description	Type	Reset Value
15–0	Channel element number	Number of elements within a frame. The maximum element number is 65535.	RW	Undefined

Table 5–21. DMA Channel Frame Number Register (DMA_CFN)

Bit	Name	Description	Type	Reset Value
15–0	Channel frame number	Number of frames within the block to transfer. 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. This size is programmed in bytes to: ❑ Allow transfer of an odd byte number ❑ Accommodate the requirement of different access sizes on source and destination ports	RW	Undefined

Table 5–22. DMA Channel Frame Index Register (DMA_CFI)

Bit	Name	Description	Type	Reset Value
15–0	Frame index	Contains the frame index, expressed in bytes, used to compute the addresses when double-index addressing mode is used.	RW	Undefined

Table 5–23. DMA Channel Element Index Register (DMA_CEI)

Bit	Name	Description	Type	Reset Value
15–0	Element index	Contains the element index, expressed in bytes, used to compute the addresses when single-index addressing mode is used.	RW	Undefined

Table 5–24. DMA Channel Progress Counter Register (DMA_CPC)

Bit	Name	Description	Type	Reset Value
15–0	Last element/ frame address 16 LSB	This register can be used to monitor the progress of a DMA transfer: ❑ If the channel transfer is synchronized on elements (DMA_CCR SYNC ≠ 0 and DMA_CCR FS = 0), the register is updated with the address 16 LSB each time the destination port issues the last request for an element. ❑ If the channel transfer is synchronized on frames (DMA_CCR SYNC ≠ 0 and DMA_CCR FS = 1) or not synchronized (DMA_CCR SYNC = 0), the register is updated with 16 LSB of the address each time the destination port issues the last request for a frame.	R	Undefined

The DMA LCD control register contains seven bits that control the LCD channel operation. There are two cases of interruption: end frame buffer or abort on the bus (bus error). Bit IE (interrupt enable) enables the generation of the interruption.

If the COND bit and the corresponding IE bit are set, an interrupt signal is sent from the DMA channel to the MPU. The MPU reads this register to find the cause of the interruption.

❑ COND = 0: Condition not detected

❑ COND = 1: Condition detected

The COND bit is updated during a transfer, and the host processor must reset this bit after reading.

Table 5–25. DMA LCD Control Register (DMA_LCD_CTRL)

Bit	Name	Value	Description	Type	Reset Value
15–7	RESERVED				
6	LCD_SOURCE		Memory source for the LCD channel	RW	0
			This bit indicates the memory source for the next LCD transfer.		
		0	Memory source is EMIFF.		
		1	Memory source is IMIF.		
5	BUS_ERROR_ IT_COND		Status LCD channel register (must be reset after read)	R–R	0
		0	No bus error interrupt detected		
		1	Bus error interrupt detected		
4	FRAME_2_ IT_COND		Status LCD channel register (must be reset after read)	R–R	0
		0	No end of frame 2 interrupt detected		
		1	End of frame 2 interrupt detected		
3	FRAME_1_ IT_COND		Status LCD channel register (must be reset after read)	R–R	0
		0	No end of frame 1 interrupt detected		
		1	End of frame 1 interrupt detected		
2	BUS_ERROR_ IT_IE		Bus error interrupt enable	RW	0
		0	Interrupt disabled		
		1	Interrupt enabled		
1	FRAME_IT_IE		End frame interrupt enable	RW	0
		0	Interrupt disabled		
		1	Interrupt enabled		

Table 5–25. DMA LCD Control Register (DMA_LCD_CTRL) (Continued)

Bit	Name	Value	Description	Type	Reset Value
0	FRAME_MODE		Kind of frame mode used for LCD transfer	RW	0
		0	One frame buffer; only registers for frame 1 are used.		
		1	Two frame buffers; LCD channel reads alternatively top_frame_1 and top_frame_2		

5.6.1.1 LCD Top Address for Frame Buffer 1 Registers (DMA_LCD_TOP_F1_L and DMA_LCD_TOP_F1_U)

The LCD top address registers are two 16-bit registers that contain the start address for the video RAM buffer 1. The 32-bit address is obtained by the concatenation of the two 16-bit words as described here:
LCD_TOP_F1 = DMA_LCD_TOP_F1_U & DMA_LCD_TOP_F1_L.

Note:

LSB of the 32-bit word is equal to zero. Address of video buffer must always be even.

Table 5–26. LCD Top Address for Frame Buffer 1—Lower Bits Register (DMA_LCD_TOP_F1_L)

Bit	Name	Description	Type	Reset Value
15–1	LCD_TOP_F1_L[15–1]	LCD top address for frame buffer 1 lower bits [15–1]	RW	Unde-fined
0	LCD_TOP_F1_L[0]	Address bit 0. Fixed at 0 since address must be even.	R	0

Table 5–27. LCD Top Address for Frame Buffer 1—Upper Bits Register (DMA_LCD_TOP_F1_U)

Bit	Name	Description	Type	Reset Value
15–0	LCD_TOP_F1_L[31–16]	LCD top address for frame buffer 1 upper bits [31–16]	RW	Unde-fined

5.6.1.2 LCD Bottom Address for Frame Buffer 1 Registers (DMA_LCD_BOT_F1_L and DMA_LCD_BOT_F1_

The LCD bottom address 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 16-bit words as described here:
LCD_BOTTOM_F1 = DMA_LCD_BOT_F1_U and DMA_LCD_BOT_F1_L

Note:

LSB of the 32-bit word is equal to zero. Address of video buffer must always be even.

Table 5–28. LCD Bottom Address for Frame Buffer 1 Register—Lower Bits Register (DMA_LCD_BOT_F1_L)

Bit	Name	Description	Type	Reset Value
15–1	LCD_BOT_F1_L[15–1]	LCD bottom address for frame buffer 1 lower bits [15–1]	RW	Undefined
0	LCD_BOT_F1_L[0]	Address bit 0. Fixed at 0 since address must be even.	R	0

Table 5–29. LCD Bottom Address for Frame Buffer 1 Register—Upper Bits Register (DMA_LCD_BOT_F1_U)

Bit	Name	Description	Type	Reset Value
15–0	LCD_BOT_F1_L[31–16]	LCD bottom address for frame buffer 1 upper bits [31–16]	RW	Undefined

5.6.1.3 LCD Top Address for Frame Buffer 2 Registers (DMA_LCD_TOP_F2_L and DMA_LCD_TOP_F2_U)

The LCD top address 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 16-bit words as described here:
LCD_TOP_F2 = DMA_LCD_TOP_F2_U & DMA_LCD_TOP_F2_L

Note:

LSB of the 32-bit word is equal to zero. Address of video buffer must always be even.

Table 5–30. LCD Top Address for Frame Buffer 2—Lower Bits Register (DMA_LCD_TOP_F2_L)

Bit	Name	Description	Type	Reset Value
15–1	LCD_TOP_F2_L[15–1]	LCD top address for frame buffer 2 lower bits [15–1]	RW	Undefined
0	LCD_TOP_F2_L[0]	Address bit 0. Fixed at 0 since address must be even.	R	0

Table 5–31. LCD Top Address for Frame Buffer 2—Upper Bits Register (DMA_LCD_TOP_F2_U)

Bit	Name	Description	Type	Reset Value
15–0	LCD_TOP_F2_L[31–16]	LCD top address for frame buffer 2 upper bits [31–16]	RW	Undefined

5.6.1.4 LCD Bottom Address for Frame Buffer 2 Registers (DMA_LCD_BOT_F2_L and DMA_LCD_BOT_F2_U)

The LCD bottom 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 16-bit words as described here:
LCD_BOTTOM_F2 = DMA_LCD_BOT_F2_U and DMA_LCD_BOT_F2_L

Note:

LSB of the 32-bit word is equal to zero. Address of video buffer must always be even.

Table 5–32. LCD Bottom Address for Frame Buffer 2—Lower Bits Register (DMA_LCD_BOT_F2_L)

Bit	Name	Description	Type	Reset Value
15–1	LCD_BOT_F2_L[15–1]	LCD bottom address for frame buffer 2 lower bits [15–1]	RW	Undefined
0	LCD_BOT_F2_L[0]	Address bit 0. Fixed at 0 since address must be even.	R	0

Table 5–33. LCD Bottom Address for Frame Buffer 2—Upper Bits Register (DMA_LCD_BOT_F2_U)

Bit	Name	Description	Type	Reset Value
15–0	LCD_BOT_F2_L[31–16]	LCD bottom address for frame buffer 2 upper bits [31–16]	RW	Undefined

Chapter 6

MPU Private Peripherals

This chapter describes the OMAP5910 multimedia processor MPU private peripherals.

Topic **Page**

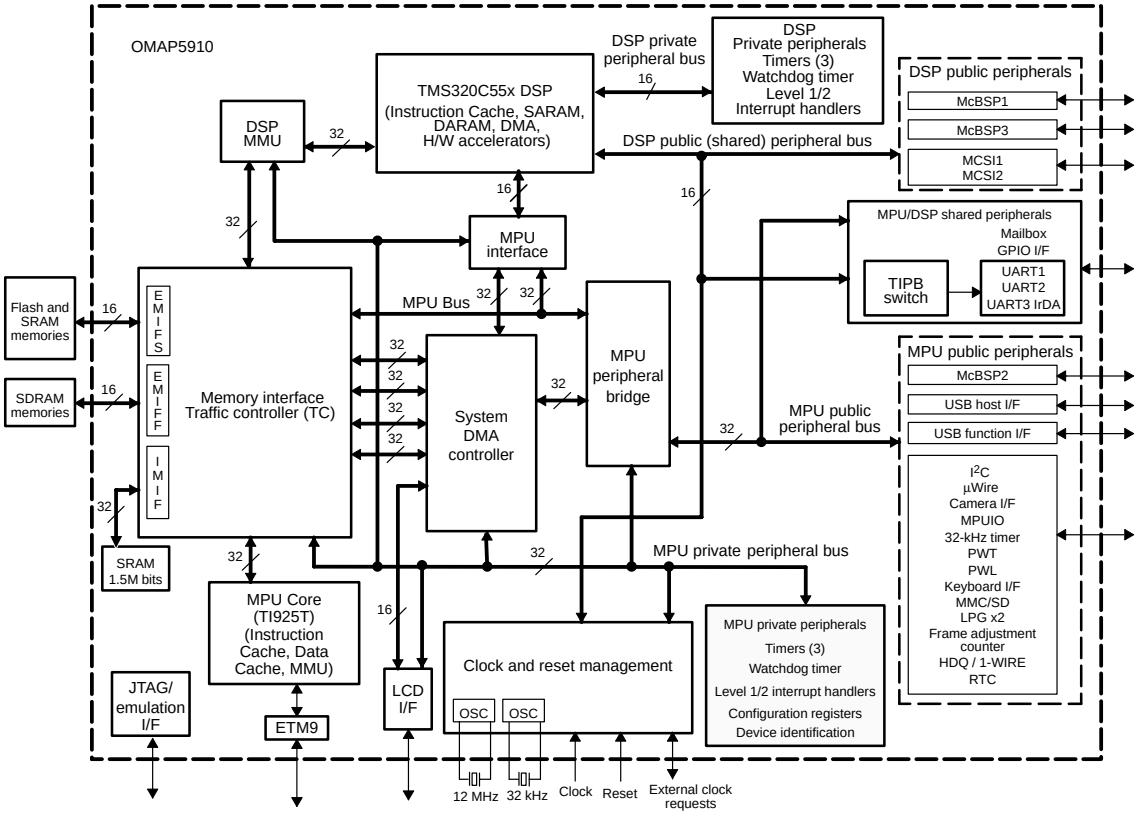
6.1 Overview

Three standard peripherals are attached to and accessible only by the TI925T RISC processor private bus (TIPB) to provide housekeeping functions for the operating system (OS) and applications. These peripherals include timers, a watchdog timer, and interrupt handlers.

The configuration module allows the software to control the different OMAP5910 modes. The device identification registers allow the software to read the different OMAP5910 identification codes.

Figure 6–1 shows the OMAP5910 device with the MPU private peripherals highlighted.

Figure 6–1. MPU Private Peripherals



6.2 Timer Description

Three 32-bit timers for the operating system provide general-purpose housekeeping functions. These timers are configured either in autoreload or one-shot mode with on-the-fly read capability. The timers generate an interrupt to the TI925T RISC processor when equal to zero. Figure 6–2 shows the 32-bit timer.

Figure 6–2. 32-Bit Timer

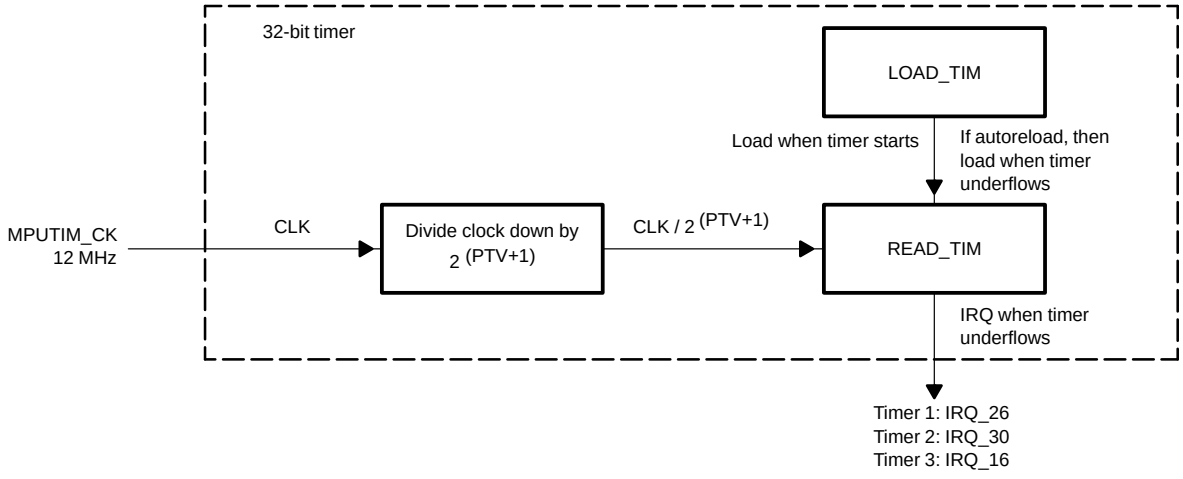


Table 6–1 identifies the level 1 interrupts for the three 32-bit timers.

Table 6–1. Timer Level 1 Interrupt

Timer	Corresponding Level 1 Interrupt
1	IRQ_26
2	IRQ_30
3	IRQ_16

The timers are 32-bit counters that receive a dedicated clock from clock generator module 1 (either CLKIN or DPLL1 output). This clock can then be prescaled, which divides it down further. Prescaling is controlled by the PTV field of the control timer register (CNTL_TIMER) (see Table 6–1).

Table 6–2 provides division values for each PTV field.

Table 6–2. PTV Value and Corresponding Division Value

PTV	Divisor
0	2
1	4
2	8
3	16
4	32
5	64
6	128
7	256

The timer interrupt period is determined in the following manner, where t_{clk} is the clock period of the input clock, LOAD_TIM (see Table 6–1) is the register that holds the value loaded when the timer passes through 0 or when it starts, and PTV is the prescaler field located in the control timer register (CNTL_TIMER):

$$t_{int} = t_{clk} \text{ X } (\text{LOAD\_TIM} + 1) \text{ x } 2^{(\text{PTV}+1)}$$

Table 6–3 shows the timer characteristics for the three timers for different input frequencies.

Table 6–3. Timer Characteristics

Input Clock	t_{clk}, Clock Period	LOAD_TIM	t_{int}, Timer Interrupt Period, for PTV = 0	t_{int}, Timer Interrupt Period, for PTV = 7
100 MHz	10 ns	0000 0001	40 ns	5.12 µs
100 MHz	10 ns	FFFF FFFF (max interrupt period)	85.9 s	10995 s (3 hr 3’ 25”)
12 MHz	83.3 ns	0000 0001	333.4 ns	42.64 µs
12 MHz	83.3 ns	FFFF FFFF (max interrupt period)	715.5 s	91589 s (25 hr, 26’29”)

If LOAD_TIM = 0 and AR (auto-reload mode) = 1, the timer is always 0 and can never decrement. Here the timer interrupt is asserted and stays asserted all the time. Since the timer interrupts are edge-senditive, only one interrupt is recognized because there is one initial edge, and then the interrupt is asserted constantly.

6.2.1 Programming the Timer

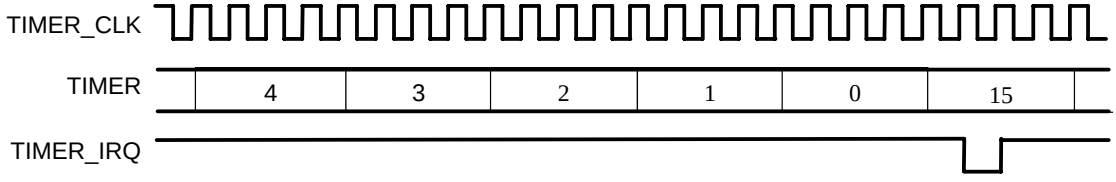
Before a timer can be used, you must enable its internal clock by setting the CLOCK_ENABLE bit of the control timer register (CNTL_TIMER) to 1. A timer is started by setting the ST field of the control timer register (CNTL_TIMER) to 1. It is stopped by resetting this bit to 0. When the timer is stopped, the content of the decrementer is frozen.

If the autoreload bit is disabled (AR field of control timer register (CNTL_TIMER) is 0), the timer decrements from the loaded value down to zero and then stops. If the autoreload bit is enabled (AR=1), the timer continues. A new value (from the load register) is loaded into the timer when it passes though zero or when it starts. An interrupt is produced when the corresponding timer passes through zero.

To avoid undefined results, do not change the settings of the PTV or AR fields of the control timer register (CNTL_TIMER) or the LOAD_TIM register while the timer is running.

The timer value is held in the VALUE_TIM field of the READ_TIM and can be read while the timer is running or stopped.

Figure 6–3. Timer Diagram



6.2.2 Timer Registers

Table 6–4 lists the timer registers. Table 6–5 through Table 6–7 describe the register bits.

Base address for timer 1: FFFE:C500

Base address for timer 2: FFFE:C600

Base address for timer 3: FFFE:C700

Bit width: 32 bits

Table 6–4. Timer Registers

Timer 1, Timer 2, and Timer 3					
Register	**Descriptions**	**R/W**	**Size**	**Offset**	**Reset Value**
CNTL_TIMER	Control timer	R/W	32 bits	x00	0x0000 0000
LOAD_TIM	Load timer	W	32 bits	x04	U
READ_TIM	Read timer	R	32 bits	x08	U

Table 6–5. Control Timer Register (CNTL_TIMER)

Bits	Name	Value	Description	Reset Value
31–7	RESERVED			
6	FREE		FREE bit	0
		0	Timer stops counting in suspend mode.	
		1	Timer continues counting in suspend mode.	
5	CLOCK_ENABLE		External timer clock enable	0
4–2	PTV		Prescale clock timer value (see Table 6–2)	0
1	AR	0	One-shot timer	0
		1	Autoreload timer	
0	ST	0	Stop timer 1:	0
		1	Start timer	

Table 6–6. Load Timer Register (LOAD_TIMER)

Bit	Name	Description	Reset Value
31–0	LOAD_TIM	The value is loaded into the VALUE_TIM when the timer passes through 0 or when it starts.	Undefined

Table 6–7. Read Timer Register (READ_TIMER)

Bit	Name	Description	Reset Value
31–0	VALUE_TIM	Value of timer	Undefined

6.3 Watchdog Timer

The watchdog timer (see Figure 6–4) can be configured as either a watchdog timer or a general-purpose timer.

6.3.1 Introduction

The watchdog timer is power-up enabled and defaults to watchdog timer for the TI925T RISC processor. A watchdog timer requires that the user program or OS periodically write to the count register before the counter underflows. If the counter underflows, the watchdog timer generates a reset to the TI925T RISC processor and to the TMS320C55x DSP. The watchdog timer detects user programs stuck in an infinite loop, loss of program control, or a runaway condition. When used as a general-purpose timer, the watchdog timer is a 16-bit timer configurable either in autoreload or one-shot mode with on-the-fly read capability. The timer generates an interrupt to the TI925T RISC processor when the count passes through zero (see Figure 6–5).

Figure 6–4. Watchdog Timer

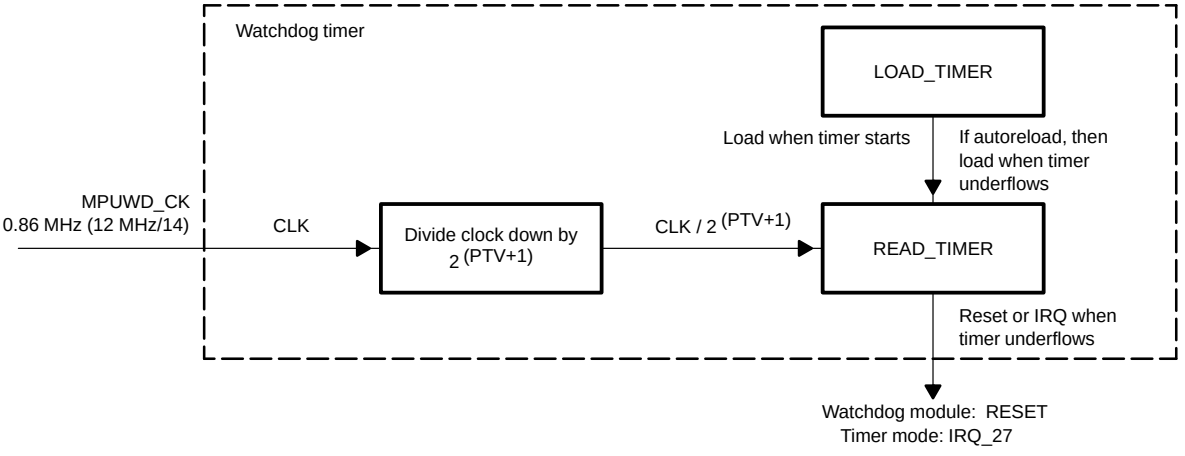


Table 6–8. Watchdog Timer Level 1 Interrupt

Timer	Corresponding Level 1 Interrupt
WD	IRQ_27

By default, this timer is configured as a watchdog timer and generates a reset of the TI925T RISC processor approximately every 19 seconds, unless you disable or update properly. If you do not, you may during system development encounter an unexpected reset every 19 seconds or so.

Be certain to disable the watchdog timer before placing the TI925T RISC processor in deep sleep mode. It must not be left configured as a watchdog timer. The watchdog timer underflow generates a reset to the TI925T RISC processor and the C55x DSP processor. If CLKIN is 12 MHz and the watchdog timer values are left at their power-up state (the value loaded into LOAD_TIM is set to the maximum value of 0xFFFF at power-up), the reset occurs in approximately 19 seconds.

The watchdog timer uses a special clock from the MPU clock frequency generation module (CLKM1). This clock is CLKIN/14. When configured as a watchdog timer, the prescaler field (PTV of CNTL_TIMER (reference)) is fixed at 7. When configured as a general-purpose timer, the prescaler field can range from 0 to 7. The time from writing a new value to counter underflow is between 256*Tclk to 16,777,216*Tclk, where Tclk = CLKIN/14, for a CLKIN clock frequency of 12 MHz, and the reset time is: 298 μs < t > 19s.

Table 6–9. PTV Value and Associated Divisor Value

PTV	Divisor
0	2
1	4
2	8
3	16
4	32
5	64
6	128
7	256

The timer interrupt period is determined in the following manner, where t_{clk} is the clock period of the input clock, LOAD_TIM is the register that holds the value loaded when the timer passes through 0 or when it starts, and PTV is the prescaler field located in the control timer register (CNTL_TIMER). The value of the PTV field is forced to 7 if the timer is in watchdog mode.

$t_{int} = t_{clk}$ X (LOAD_TIM + 1) x $2^{(PTV+1)}$

Table 6–10 shows the characteristics of the watchdog timer for different LOAD_TIM values.

Table 6–10. Watchdog Timer Characteristics

Input clock, CLKIN	t_{clk}, Clock Period†	LOAD_TIM	t_{int}, Timer Interrupt Period, for PTV = 7
12 MHz	1167 ns	0001	597.34 μs
12 MHz	1167 ns	FFFF (max interrupt period)	19.57 s

† The 12-MHz clock is divided by 14.

If LOAD_TIM = 0 and AR (auto-reload mode) = 1, the timer is always 0 and can never decrement. Here the timer interrupt is asserted and stays asserted all the time. Since the timer interrupts are edge-senditive, only one interrupt is recognized because there is one initial edge, and then the interrupt is asserted constantly.

6.3.2 Programming the Watchdog Timer in Watchdog Mode

On power up, the watchdog timer defaults to watchdog mode and the value loaded into the LOAD_TIM register is set to the maximum value (0xFFFF). This gives the user a time of 16,777,216 * t_{clk} (19.57 seconds) to change the timer mode or write a new value (different from 0xFFFF) into the LOAD_TIM register.

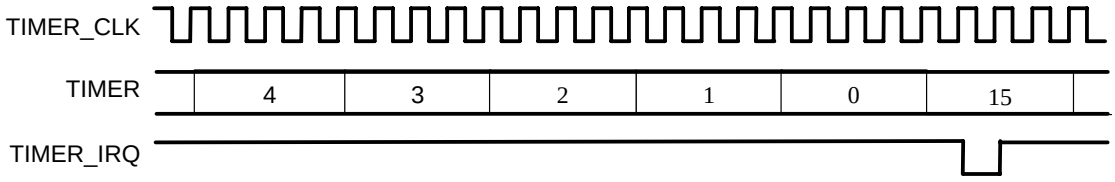
The user program or the OS must write periodically to the count register, LOAD_TIM, before the counter underflows. The new loaded value must be different from the previous value because the write is taken into account only if the newly loaded value is different from the previous value. Due to internal sequencing, the user must wait three timer clock periods before writing a new value into the LOAD_TIM register. If CLKIN is 12 MHz, the duration of three timer clock periods is approximately 3.5 μs.

By writing a predefined sequence (0xF5 followed by 0xA0) to the TIMER_MODE register (see Table 6–15), the timer can be configured as a general-purpose timer. A sequence decode is initialized when 0xF5 is written to the TIMER_MODE register. Once in this state, if the next write is different from 0xA0, the state machine causes a reset as if the watchdog timer has underflowed. You cannot disable the watchdog timer by simply clearing the watchdog bit of the TIMER_MODE register.

When the timer has been configured as a general-purpose timer, it can be switched back to watchdog mode by writing a 1 to the watchdog bit of the TIMER_MODE register. In this case, the value loaded into LOAD_TIM is set to the maximum value (0xFFFF) as on power up.

In watchdog mode, the control timer register (CNTL_TIMER) must not be used. The watchdog timer cannot be stopped by clearing the ST field. The prescale value is 7, regardless of the PTV field value. Autoreload and one-shot do not apply, because, if the counter underflows, the processor is reset and the watchdog registers are reinitialized.

Figure 6–5. Timer Diagram



6.3.3 Programming the Watchdog Timer in Timer Mode

The timer is started by setting the ST field of the control timer register (CNTL_TIMER) to 1. It is stopped by resetting this bit to 0. When the timer is stopped, the content of the decrementer is frozen.

If the autoreload bit is disabled (the AR field of control timer register (CNTL_TIMER) is 0), the timer decrements from the loaded value down to zero and then stops. If the autoreload bit is enabled (AR = 1), the timer continues. A new value (from the load register) is loaded into the timer when it passes though zero or when it starts. An interrupt is produced when the corresponding timer is equal to zero.

To avoid undefined value, do not change the setting of the PTV field or the AR field of the control timer register (CNTL_TIMER) or the LOAD_TIM register while the timer is running. PTV can be set to values other than 7 when the watchdog timer is in timer mode.

The timer value is held in the VALUE_TIM field of the READ_TIM register and can be read while the timer is running or stopped.

6.3.4 Watchdog Timer Registers

Table 6–11 lists the watchdog timer registers. Table 6–12 through Table 6–15 describe the register bits.

Base address for watchdog timer: FFFE:C800

Bit width: 32 bits

Table 6–11. Watchdog Timer Registers

Name	Description	R/W	Size	Offset	Reset Value
CNTL_TIMER	Control timer	R/W	16 bits	x00	0x0002
LOAD_TIM	Load timer	W	16 bits	X04	0xFFFF
READ_TIM	Read timer	R	16 bits	X04	0xFFFF
TIMER_MODE	Timer mode	R/W	16 bits	X08	0x8000

Table 6–12. Control Timer Register (CNTL_TIMER)

Bits	Name	Value	Description	Reset Value
15–12	RESERVED			
11–9	PTV		Prescale clock timer value	0
8	AR	0	One-shot timer	0
		1	Autoreload timer	
7	ST	0	Stop timer	0
		1	Start timer	
6–2	RESERVED			
1	FREE	0	Enables suspend functionality	1
		1	Timer runs free, regardless of suspend value.	
0	RESERVED		Reserved	

Table 6–13. Load Timer Register (LOAD_TIM)

Bit	Name	Description	Reset Value
15–0	LOAD_TIM	General-purpose timer: This value is loaded when timer passes through 0 or when it starts. Watchdog timer: Reload timer with this value.	FFFF

Table 6–14. Read Timer Register (READ_TIM)

Bit	Name	Description	Reset Value
15–0	VALUE_TIM	Read timer value	FFFF

Table 6–15. Timer Mode Register (TIMER_MODE)

Bit	Name	Value	Description	Reset Value
15	WATCHDOG		Write access	1
		1	Switchback timer mode to watchdog. Writing a 0 in this bit has no effect.	
14–8	RESERVED			
7–0	WATCHDOG_DIS		Write access only	NA
			Writing a predefined sequence (0xF5) followed by 0xA0 in this field disables the watchdog.	
			Functionality:	
			After receiving 0xF5, if the second write access is different from 0xA0, the MPU core is reset.	

6.4 MPU Interrupt Handlers

The MPU only supports two interrupt sources: IRQ and FIQ. However, the OMAP5910 has numerous peripherals and DMA channels which provide interrupts. To allow these numerous interrupts to be supported using just two interrupt sources, an interrupt handler is used. The interrupt handlers allow up to 32 individual interrupts to be programmed to assert either IRQ or FIQ and they allow these interrupt sources to be masked as well as prioritized with relationship to one another. If any of these unmasked interrupts occur, then either a FIQ or IRQ interrupt occurs.

The OMAP5910 has two layers of interrupt handlers, as shown in Figure 6–6. If an unmasked interrupt occurs on the level 2 interrupt handler, it asserts IRQ_0 of the level 1 interrupt handler. This allows up to 62 interrupt sources to be supported.

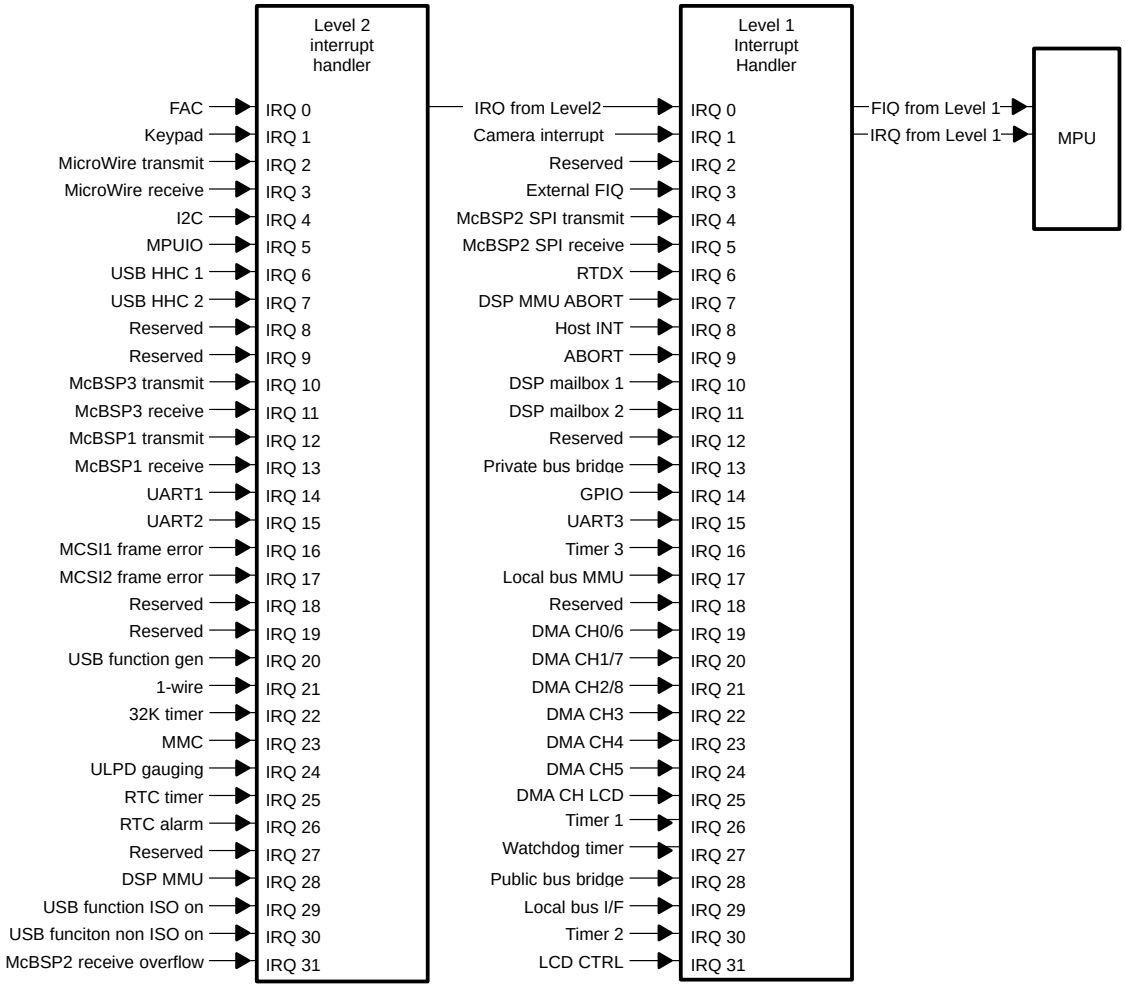
The OMAP5910 device does not support nested interrupts.

6.4.1 MPU Level 1 Interrupt Handler

The MPU level 1 interrupt handler has 32 interrupt request lines (IRQ_[31:0]). These interrupts are generated by peripherals such as the timers, camera, LCD, the system DMA controller, and the DSP. The interrupt handler handles edge-triggered or level-sensitive interrupts (individually programmable via the ILRn registers—see Table 6–23). All interrupts are maskable (individually enabled and disabled via the mask interrupt register (MIR)—see Table 6–19) with an internal register. The interrupt source information can be read back from the ITR register (see Table 6–18, Table 6–19, and Table 6–20). Interrupt priority is also programmable (ILRn registers) to allow flexibility for different applications (see Table 6–1). The output from the interrupt handler is routed to one of the two MPU interrupt (IRQ or FIQ—see Figure 6–6) inputs according to that interrupt ILRn configuration bit.

A clock request mechanism is implemented to wake up and provide a clock to the interrupt handler when the OMAP5910 device is in one of the sleep modes.

Figure 6–6. MPU Interrupt Handlers



6.4.2 MPU Level 2 Interrupt Handler

Because the number of interrupts that the OMAP5910 device must manage is greater than 32, a second interrupt handler is used. The resulting interrupt is connected to the IRQ_0 of the TI925T RISC processor interrupt handler, which must be programmed as a level interrupt. The added (L2) interrupt handler is similar to the level 1 interrupt handler.

The result of connecting the two interrupt handlers in a cascade manner is to increase the total number of input interrupts from 32 to 62.

The simplified sequence for the MPU to receive an input interrupt is as follows:

Step 1: Read the SIR_IRQ_CODE register of the level 1 MPU interrupt handler.

Step 2: If the interrupt is caused by the level 2 interrupt handler (as indicated by an IRQ of 0), read the SIR_IRQ_CODE register of the level 2 interrupt handler.

Step 3: If the interrupt is a level interrupt, the corresponding interrupt routine must first clear the interrupt source (usually by writing to a register in the module generating the interrupt) or at least mask the interrupt. Then it must write 1 into the NEW_IRQ_AGR field of the level 2 interrupt handler CONTROL_REG. Then, the ITR register of the level 1 interrupt handler must be cleared. Finally, 1 must be written into the NEW_IRQ_AGR field of the level 1 interrupt handler.

Step 4: If it is an edge interrupt, read the status register to determine the cause of the interrupt, start interrupt routine, then write 1 into the NEW_IRQ_AGR field of the level 2 interrupt handler CONTROL_REG. Clear the ITR of the level 1 interrupt handler, then write 1 into the NEW_IRQ_AGR field of the level 1 interrupt handler CONTRL_REG.

6.5 Level 1 and Level 2 Interrupt Mapping

Table 6–16 lists the mapping of the incoming interrupts.

IRQ_ABORT (IRQ_9) is the traffic controller abort IRQ. It is also connected to DSP IRQ_12. This interrupt comes from either a TIPB bus or the MPUI and is caused by a time-out abort.

Table 6–16. Level 1 and Level 2 OMAP5910 MPU Interrupt Mapping

Incoming Interrupts	Default Sensitivity Configuration	Interrupt Line on Level 1	Interrupt Line on Level 2
Level 2 interrupt handler IRQ	Level	IRQ_0	—
Camera interrupt	Level	IRQ_1	—
Reserved		IRQ_2	—
External FIQ	Edge	IRQ_3	—
McBSP2 TX interrupt	Edge	IRQ_4	—
McBSP2 RX interrupt	Edge	IRQ_5	—
IRQ_RTDX†	Level	IRQ_6	—
IRQ_DSP_MMU_ABORT	Level	IRQ_7	—
IRQ_HOST_INT	Level	IRQ_8	—
IRQ_ABORT	Level	IRQ_9	—
IRQ_DSP_MAILBOX1	Level	IRQ_10	—
IRQ_DSP_MAILBOX2	Level	IRQ_11	—
Reserved			
IRQ_TIPB_BRIDGE_PRIVATE	Level	IRQ_13	—
IRQ_GPIO	Level	IRQ_14	—
IRQ_UART3	Level	IRQ_15	—
IRQ_TIMER3	Edge	IRQ_16	—
IRQ_LB_MMU	Level	IRQ_17	—
Reserved			
IRQ_DMA_CH0_CH6	Level	IRQ_19	—
IRQ_DMA_CH1_CH7	Level	IRQ_20	—
IRQ_DMA_CH2_CH8	Level	IRQ_21	—

† IRQ_RTDX is used in emulation for the Code Composer Studio RTDX (real time data exchange) interrupt.

Table 6–16. Level 1 and Level 2 OMAP5910 MPU Interrupt Mapping (Continued)

Incoming Interrupts	Default Sensitivity Configuration	Interrupt Line on Level 1	Interrupt Line on Level 2
IRQ_DMA_CH3	Level	IRQ_22	—
IRQ_DMA_CH4	Level	IRQ_23	—
IRQ_DMA_CH5	Level	IRQ_24	—
IRQ_DMA_CH_LCD	Level	IRQ_25	—
IRQ_TIMER1	Edge	IRQ_26	—
IRQ_WD_TIMER	Edge	IRQ_27	—
IRQ_TIPB_BRIDGE_PUBLIC	Level	IRQ_28	—
IRQ_LOCAL_BUS_I/F	Level	IRQ_29	—
IRQ_TIMER2	Edge	IRQ_30	—
IRQ_LCD_CTRL	Level	IRQ_31	—
FAC	Level	IRQ0	IRQ_00
Keyboard	Edge	IRQ0	IRQ_01
MicroWire TX	Edge	IRQ0	IRQ_02
MicroWire RX	Edge	IRQ0	IRQ_03
I2C	Edge	IRQ0	IRQ_04
MPUIO	Level	IRQ0	IRQ_05
USB HHC 1	Level	IRQ0	IRQ_06
Reserved		IRQ0	IRQ_07
Reserved		IRQ0	IRQ_08
Reserved		IRQ0	IRQ_09
McBSP3 TX interrupt	Edge	IRQ0	IRQ_10
McBSP3 RX interrupt	Edge	IRQ0	IRQ_11
McBSP1 TX interrupt	Edge	IRQ0	IRQ_12
McBSP1 RX interrupt	Edge	IRQ0	IRQ_13
UART1 (Bluetooth)	Level	IRQ0	IRQ_14
UART2 (communication)	Level	IRQ0	IRQ_15

† IRQ_RTDX is used in emulation for the Code Composer Studio RTDX (real time data exchange) interrupt.

Table 6–16. Level 1 and Level 2 OMAP5910 MPU Interrupt Mapping (Continued)

Incoming Interrupts	Default Sensitivity Configuration	Interrupt Line on Level 1	Interrupt Line on Level 2
MCSI1 combined TX/RX/frame error interrupt	Level	IRQ0	IRQ_16
MCSI2 combined TX/RX/frame error interrupt	Level	IRQ0	IRQ_17
Reserved		IRQ0	IRQ_18
Reserved		IRQ0	IRQ_19
USB function Geni interrupt	Level	IRQ0	IRQ_20
1-Wire interrupt	Level	IRQ0	IRQ_21
Timer 32K interrupt	Edge	IRQ0	IRQ_22
MMC interrupt	Level	IRQ0	IRQ_23
ULPD interrupt	Level	IRQ0	IRQ_24
RTC periodical timer	Edge	IRQ0	IRQ_25
RTC alarm	Level	IRQ0	IRQ_26
Reserved		IRQ0	IRQ_27
DSPMMU IRQ		IRQ0	IRQ_28
USB function IRQ ISO On	Level	IRQ0	IRQ_29
USB function IRQ Non-ISO On	Level	IRQ0	IRQ_30
McBSP2 RX OVERFLOW It	Edge	IRQ0	IRQ_31

† IRQ_RTDX is used in emulation for the Code Composer Studio RTDX (real time data exchange) interrupt.

Note:

This version of the interrupt controller does not support nested interrupts.

Level-sensitive interrupts remain asserted until acknowledged.

Edge-triggered interrupts do not remain asserted. The interrupt is cleared upon reading the SIR registers or writing a 0 to the ITR registers in the interrupt handler.

6.6 Interrupt Handler Level 1 and Level 2 Registers

There are two sets of interrupt handler registers: one for the level 1 handler, the other for the level 2 handler (see Table 6–17). Table 6–18 through Table 6–24 describe the register bits.

Base address for interrupt handler 1: FFFE:CB00

Base address for interrupt handler 2: FFFE:0000

Bit width: 32 bits

Table 6–17. Interrupt Handler Registers

Name	Description	R/W	Bits	Offset	Reset Value
ITR	Interrupt input	R/W	32 bits	0X00	0x0000 0000
MIR	Mask interrupt	R/W	32 bits	0X04	0xFFFF FFFF
SIR_IRQ_CODE	Interrupt encoded source (IRQ)	R	5 bits	0X10	0x00
SIR_FIQ_CODE	Interrupt encoded source (FIQ)	R	5 bits	0X14	0x00
CONTROL_REG	Interrupt control register	R/W	2 bits	0X18	0x0
ILR0	Interrupt priority level for IRQ 0	R/W	7 bits	0X1C	0x00
ILR1	Interrupt priority level for IRQ 1	R/W	7 bits	0X20	0x00
ILR2	Interrupt priority level for IRQ 2	R/W	7 bits	0X24	0x00
ILR3	Interrupt priority level for IRQ 3	R/W	7 bits	0X28	0x00
ILR4	Interrupt priority level for IRQ 4	R/W	7 bits	0X2C	0x00
ILR5	Interrupt priority level for IRQ 5	R/W	7 bits	0X30	0x00
ILR6	Interrupt priority level for IRQ 6	R/W	7 bits	0X34	0x00
ILR7	Interrupt priority level for IRQ 7	R/W	7 bits	0X38	0x00
ILR8	Interrupt priority level for IRQ 8	R/W	7 bits	0X3C	0x00
ILR9	Interrupt priority level for IRQ 9	R/W	7 bits	0X40	0x00
ILR10	Interrupt priority level for IRQ 10	R/W	7 bits	0X44	0x00
ILR11	Interrupt priority level for IRQ 11	R/W	7 bits	0X48	0x00
ILR12	Interrupt priority level for IRQ 12	R/W	7 bits	0X4C	0x00
ILR13	Interrupt priority level for IRQ 13	R/W	7 bits	0X50	0x00
ILR14	Interrupt priority level for IRQ 14	R/W	7 bits	0X54	0x00

Table 6–17. Interrupt Handler Registers (Continued)

Name	Description	R/W	Bits	Offset	Reset Value
ILR15	Interrupt priority level for IRQ 15	R/W	7 bits	0X58	0x00
ILR16	Interrupt priority level for IRQ 16	R/W	7 bits	0X5C	0x00
ILR17	Interrupt priority level for IRQ 17	R/W	7 bits	0X60	0x00
ILR18	Interrupt priority level for IRQ 18	R/W	7 bits	0X64	0x00
ILR19	Interrupt priority level for IRQ 19	R/W	7 bits	0X68	0x00
ILR20	Interrupt priority level for IRQ 20	R/W	7 bits	0X6C	0x00
ILR21	Interrupt priority level for IRQ 21	R/W	7 bits	0X70	0x00
ILR22	Interrupt priority level for IRQ 22	R/W	7 bits	0X74	0x00
ILR23	Interrupt priority level for IRQ 23	R/W	7 bits	0X78	0x00
ILR24	Interrupt priority level for IRQ 24	R/W	7 bits	0X7C	0x00
ILR25	Interrupt priority level for IRQ 25	R/W	7 bits	0X80	0x00
ILR26	Interrupt priority level for IRQ 26	R/W	7 bits	0X84	0x00
ILR27	Interrupt priority level for IRQ 27	R/W	7 bits	0X88	0x00
ILR28	Interrupt priority level for IRQ 28	R/W	7 bits	0X8C	0x00
ILR29	Interrupt priority level for IRQ 29	R/W	7 bits	0X90	0x00
ILR30	Interrupt priority level for IRQ 30	R/W	7 bits	0X94	0x00
ILR31	Interrupt priority level for IRQ 31	R/W	7 bits	0X98	0x00
ISR	Software interrupt set register	R/W	32 bits	0X9C	0x0000 0000

Table 6–18. Interrupt Input Register(ITR)

Bit	Name	Description	Reset Value
31	IRQ_31	Interrupt request—1 indicates that the peripheral occupying the IRQ_31 address space has requested interrupt service from the MPU. An edge-triggered interrupt is stored in this register as an incoming interrupt. When the MPU reads the SIR_IRQ_CODE or the SIR_FIQ_CODE register, the bit corresponding to the pending interrupt is reset. The MPU can also individually clear each bit by writing a 0 to that bit. (Writing a 1 to the bit does not change the previous state. This can be used just before the MPU unmasks some interrupts to ignore specific interrupts.	0
30–0	IRQ_30–IRQ_0	(Same as bit 31)	0

Table 6–19. Mask Interrupt Register (MIR)

Bit	Name	Description	Reset Value
31	IRQ_31_MSK	Interrupt mask bit—1 prevents IRQ_31 from interrupting MPU program flow. If the peripheral on IRQ_31 has been configured to request an interrupt but masked out in this register, the IRQ_31 bit in the IRQ register is still set on an interrupt event (and can be read by the MPU) but does not interrupt program flow.	1
30–0	IRQ_30_MSK–IRQ_0_MSK	(Same as bit 31)	1

Table 6–20. Binary-Coded Source IRQ Register (SIR_IRQ_CODE)

Bit	Name	Description	Reset Value
4–0	IRQ_NUM	This register indicates the IRQ interrupt that is currently being serviced by the MPU. Reading this register clears the corresponding bit in the ITR register if the interrupt is configured as edge triggered.	0

Table 6–21. Binary-Coded Source FIQ Register (SIR_FIQ_CODE)

Bit	Name	Description	Reset Value
4–0	FIQ_NUM	This register indicates the IRQ interrupt that is currently being serviced by the MPU. Reading this register clears the corresponding bit in the ITR register if the interrupt is configured as edge triggered.	0

This register is only used by the level 1 handler, because the level 2 handler cannot be programmed to generate FIQ interrupts.

Table 6–22. Control Register (CONTROL_REG)

Bit	Name	Description	Reset Value
1	NEW_FIQ_REG	New FIQ agreement. Writing a 1 resets FIQ output, clears source FIQ register, and enables new IRQ generation.	0
0	NEW_IRQ_REG	New IRQ agreement. Writing a 1 resets IRQ output, clears source IRQ register, and enables new IRQ generation.	0

Table 6–23. Interrupt Level Registers (ILR0...ILR31)

Bit	Name	Value	Description	Reset Value
6–2	PRIORITY		Defines the priority level when the corresponding interrupt is routed to IRQ or FIQ (31 down to 0)	0
1	SENS_EDGE	0	Interrupt is falling-edge-triggered.	0
		1	Interrupt is low-level-triggered.	
0	FIQ†	0	Interrupt is routed to IRQ.	0
		1	Interrupt is routed to FIQ.	

† IRQ is the only valid setting fo this bit when used with the level 2 handler—it cannot be used to generate FIQ sources.

Table 6–24. Interrupt Set Register (ISR)

Bit	Name	Description	Reset Value
31–0	SWI[31:0]	Software interrupt set register. Writing a 1 to any bit generates an interrupt to the MPU if the corresponding ILRn is configured as edge-triggered; otherwise no interrupt is generated. A read to this register always returns 0x00000000.	0

6.7 Configuration Module

The OMAP5910 configuration module allows the software of the OMAP5910 device to control the various static modes supported by the device. This module is the primary point of control for the following areas of the OMAP5910 device:

- ❏ Functional I/O multiplexing
- ❏ Debug and observation I/O multiplexing
- ❏ I/O gating and inhibiting for power-down modes
- ❏ Pull-down enable control
- ❏ Interface voltage selection
- ❏ Pseudostatic module configuration

Note:

This configuration must be done only during the boot time while the OMAP5910 peripherals are under reset.

6.7.1 Configuration Register Capabilities

The OMAP5910 configuration module is functionally simple. The module is a bank of 32-bit registers that can be read and written by firmware. This bank of registers can be broken down into eight primary sections. These are:

- ❏ OMAP5910 generic multiplexing registers (0x0010h to 0x0038h address range)
- ❏ OMAP5910 pullup/pulldown control registers (0x0040h to 0x004Ch address range)
- ❏ OMAP5910 gating and inhibiting registers (0x0050h address range)
- ❏ OMAP5910 voltage control registers (0x0060h address range)
- ❏ OMAP5910 test and debug registers (0x0070h address range)
- ❏ OMAP5910 module configuration registers (0x0080h address range)

6.7.2 OMAP5910 Native and Compatibility Modes

The major functionality of this module beyond the register banks is to support compatibility with the previous prototype devices via the implementation of *native* and *compatibility* modes. The OMAP5910 device resets to compatibility mode. This functionality is in place to allow software compatibility of

OMAP5910 with early development devices. The OMAP5910 configuration registers have no effect on the compatibility mode. The firmware must first write 0x0000EAEFh to the COMP_MODE_CTRL_0 register to utilize the pin multiplexing and device configuration features available in native mode. Be careful when enabling the native mode.

All OMAP5910 configuration registers reset to 0x0000h at power-on reset. It is advisable to follow the following procedure before enabling the OMAP5910 mode:

1) Determine the desired values for each OMAP5910 configuration register.

2) Program the desired values by writing to the appropriate register.

3) Program the COMP_MODE_CTRL_0 register to 0x0000EAEFh.

4) The desired modes are now active.

This procedure allows the user to select all OMAP5910 configuration settings with a series of register writes, then to enable all of the modes simultaneously.

6.7.3 OMAP5910 Generic Pin Multiplexing and Pullup/Pulldown Control

The OMAP5910 configuration module was developed with future versions of OMAP5910 in mind. To enable software compatibility between OMAP5910 and future versions, this module allows for up to eight multiplexing options on all device pins and independent pin-by-pin pulldown control except:

- ❑ SDRAM
- ❑ Flash memory
- ❑ LCD
- ❑ Power and ground pins
- ❑ Analog I/O functions
- ❑ Test and emulation pins

The OMAP5910 FUNC_MUX_CTRL (3–D) registers control this generic functional pin multiplexing. The OMAP5910 PULL_DWN_CTRL (0–3) registers control the independent pin-by-pin pulldown enables.

For more information on what functional multiplexing is available on the OMAP5910, see Appendix A, *Input/Output Descriptions*. Once the desired functionality is determined, the OMAP5910 FUNC_MUX_CTRL (3–D) registers can be programmed to correspond to the chosen multiplexing. The value for the three FUNC_MUX_CTRL register bits that correspond to a given pin can be determined in Table 6–25.

Table 6–25. Functional Pin Multiplexing Control Register 3 (FUNC_MUX_CTRL3...FUNC_MUX_CTRLD)

FUNC_MUX_CTRL(2:0) Register Value	Corresponding Functional Modes
000	Default configuration/functional multiplexing 0
001	Functional multiplexing 1
010	Functional multiplexing 2
011	Functional multiplexing 3
100	Functional multiplexing 4 (Reserved)
101	Functional multiplexing 5 (Reserved)
110	Functional multiplexing 6 (Reserved)
111	Functional multiplexing 7 (Reserved)

For a given interface, the value of the FUNC_MUX_CTRL(2:0) register can vary from pin to pin. For example, the USB1_HOST port is split between functional multiplexing 2 and functional multiplexing three modes in Appendix A, *Input/Output Descriptions*. In this case four of the FUNC_MUX_CTRL(2:0) registers has a value of 001 and the other four FUNC_MUX_CTRL(2:0) registers have a value of 010.

6.7.4 OMAP5910 MMC/SD Pin Multiplexing

The enabling of the MMC/SD function on the device's pins is a special case on the OMAP5910 device. The MMC/SD pin interface uses the state of a device pin (STAT_VAL/WKUP) at release of power-on reset to determine if the MMC/SD function is enabled at the device's pins. The power-on reset sampling of a high level on this pin forces the device's I/O into a state that is consistent with MMC/SD. This means that several pullups are enabled when in MMC/SD mode. Users must program the OMAP5910 configuration registers to set up the proper functional multiplexing modes. Users of 4-bit MMC/sd must be particularly aware that the CONF_MOD_MSMMC_VSS_HIZ_OVERRIDE bit in the MOD_CONF_CTRL_0 register must be programmed to a 1 to enable the use of the MMC.DAT2 device pin. For further details on the MMC/SD pin multiplexing on the OMAP5910 device, see Appendix A, *Input/Output Descriptions*, and Section 7.12, *MMC/SD Host Controller*.

6.7.5 OMAP5910 Pullups and Pulldowns

The OMAP5910 device implements both pullups and pulldowns on several I/Os. In this document there are several references to pulldowns and pulldown enables. It is proper to assume that if an OMAP5910 device pin has a pullup, the corresponding pulldown enables (enables = 0/disables = 1) the pullup.

6.8 OMAP5910 Configuration Registers

Table 6–26 lists the 32-bit read/write configuration registers. Table 6–27 through Table 6–49 describe the register bits. The compatibility mode control 0 register (COMP_MODE_CTRL_0) must be programmed to 0xEAEFh for any of these configuration registers to exercise their associated control. The base address for the configuration registers is FFFE:1000.

Table 6–26. Configuration Registers

Register	Description	Offset
FUNC_MUX_CTRL_0	Functional multiplexing control 0	0x00
FUNC_MUX_CTRL_1	Functional multiplexing control 1	0x04
FUNC_MUX_CTRL_2	Functional multiplexing control 2	0x08
COMP_MODE_CTRL_0	Compatibility mode control 0	0x0C
FUNC_MUX_CTRL_3	Functional multiplexing control 3	0x10
FUNC_MUX_CTRL_4	Functional multiplexing control 4	0x14
FUNC_MUX_CTRL_5	Functional multiplexing control 5	0x18
FUNC_MUX_CTRL_6	Functional multiplexing control 6	0x1C
FUNC_MUX_CTRL_7	Functional multiplexing control 7	0x20
FUNC_MUX_CTRL_8	Functional multiplexing control 8	0x24
FUNC_MUX_CTRL_9	Functional multiplexing control 9	0x28
FUNC_MUX_CTRL_A	Functional multiplexing control A	0x2C
FUNC_MUX_CTRL_B	Functional multiplexing control B	0x30
FUNC_MUX_CTRL_C	Functional multiplexing control C	0x34
FUNC_MUX_CTRL_D	Functional multiplexing control D	0x38
PULL_DWN_CTRL_0	Pulldown control 0	0x40
PULL_DWN_CTRL_1	Pulldown control 1	0x44
PULL_DWN_CTRL_2	Pulldown control 2	0x48
PULL_DWN_CTRL_3	Pulldown control 3	0x4C
GATE_INH_CTRL_0	Gate and inhibit control 0	0x50
VOLTAGE_CTRL_0	Voltage control 0	0x60
TEST_DBG_CTRL_0	Test debug control 0	0x70
MOD_CONF_CTRL_0	Module configuration control 0	0x80

Table 6–27. Functional Multiplexing Control 0 Register (FUNC_MUX_CTRL_0)

Bit	Name	Value	Description	R/W	Reset Value
31	CTRL_288_1		This bit configures the control mode 288_1 which enables the control of the OMAP chip_nwakeup signal from the static_valid pad.	R/W	0x0
		0	Functional mode; ULPD controls the OMAP chip_nwakeup signal.		
		1	Debug; the OMAP5910 static_valid pad controls the OMAP chip_nwakeup signal.		
			This bit is valid in compatibility and native modes.		
30–23	RESERVED		Reserved. These bits must always be written as 0.	R/W	0x0
22	LB_RESET_DISABLE		This bit holds the OMAP local bus reset input active. Set this to 1 when using OMAP5910 USB_HHC module.	R/W	0x0
		0	Local bus $\overline{\text{RESET}}$ <= 0		
		1	Local bus $\overline{\text{RESET}}$ <= USB_HHC LB reset		
			This bit is valid in compatibility and native modes.		
21	RESERVED		Reserved. This bit must always be written as 0.	R/W	0x0
20	LRU_SEL		This field configures the OMAP traffic controller arbitration algorithm.	R/W	0x0
		0	LRU priority scheme is used for arbitration.		
		1	Fixed priority scheme is used for arbitration.		
			This bit must only be changed if the DSP is in reset. This bit is valid in compatibility and native modes.		

Table 6–27. Functional Multiplexing Control 0 Register (FUNC_MUX_CTRL_0) (Continued)

Bit	Name	Value	Description	R/W	Reset Value
19	VBUS_CTRL		This bit can be programmed to indicate an external USB insertion/disconnection to the OMAP5910 USB core.	R/W	0x0
		0	Indicates an external USB disconnection		
		1	Indicates an external USB insertion		
			This bit is valid in compatibility and native modes. There are several methods for VBUS detect in native mode.		
18	VBUS_MODE		Selects the USB vbus_ctrl input source, used for USB insertion/disconnection detection.	R/W	0x0
		0	USB input vbus_ctrl <= Hardware detection (see bit (i7) of the MOD_CONF_CTRL_0 register)		
		1	USB input vbus_ctrl <= OMAP5910 configuration VBUS_CTRL bit		
17–15	RESERVED		Reserved. These bits must always be written as 0.	R/W	0x0
14	NRESET_ENABLE		Allows AND gating of OMAP5910 outputs with the OMAP CHIP_NRESET_OUT	R/W	0x0
		0	Disabled		
		1	Allowed		
			This bit is valid in compatibility and native modes.		
13	PWR_MASK_IN	0	Does not allow AND gating of OMAP5910 inputs with COM_PWR_REQ (GPIO9) and COM_STS (MPUIO(3)) OMAP5910 input pins	R/W	0x0
		1	Allows AND gating of OMAP5910 inputs with COM_PWR_REQ (GPIO9) and COM_STS (ARMIO3) OMAP5910 input pins		
			This bit is valid in compatibility and native modes.		

Table 6–27. Functional Multiplexing Control 0 Register (FUNC_MUX_CTRL_0) (Continued)

Bit	Name	Value	Description	R/W	Reset Value
12	PWR_MASK_OUT	0	Does not allow AND gating of OMAP5910 outputs with COM_PWR_REQ (GPIO9) and COM_STS (MPUIO3) OMAP5910 input pins	R/W	0x0
		1	Allows AND gating of OMAP5910 outputs with COM_PWR_REQ (GPIO9) and COM_STS (MPUIO3) OMAP5910 input pins		
			This bit is valid in compatibility and native modes.		
11	BVLZ_MASK_IN	0	Does not allow AND gating of OMAP5910 inputs with BFAIL/EXT_FIQ OMAP5910 input pin	R/W	0x0
		1	Allows AND gating of OMAP5910 inputs with BFAIL/EXT_FIQ OMAP5910 input pin		
			This bit is valid in compatibility and native modes.		
10	BVLZ_MASK_OUT	0	Does not allow AND gating of outputs with BFAIL/EXT_FIQ OMAP5910 input pin	R/W	0x0
		1	Allows AND gating of outputs with BFAIL/EXT_FIQ OMAP5910 input pin		
			This bit is valid in compatibility and native modes.		
9–0	RESERVED		Reserved. These bits must always be written as 0.	R/W	0x0

Table 6–28. Functional Multiplexing Control 1 Register (FUNC_MUX_CTRL_1)

Bits	Name	Description	R/W	Reset Value
31–0	RESERVED	Reserved. These bits must always be written as 0.	R/W	0x00000000

Table 6–29. Functional Multiplexing Control 2 Register (FUNC_MUX_CTRL_2)

Bits	Name	Description	R/W	Reset Value
31–19	RESERVED	Reserved. These bits must always be written as 0.	R/W	0x00000000
18:13	DMAREQ_OBS	This 6-bit field can be used to control the DMA requests observability mux. When a 6-bit value is written in this field, the corresponding interrupt signal is output on the UART3.RX pin for visibility. Legal values are from 0 to 50. 0 is the functional mode, values between 1 and 50 are for observability mode. 0: Default; for i = 1 to 19: observability, pin UART3.RX <= DSP DMA request(i), output; for i = 20 to 50: observability, pin UART3.RX <= system DMA request(i–20), output	R/W	0x0000
12:6	IT_OBS	This 7-bit field can be used to control the interrupt observability mux. When a 7-bit value is written in this field, the corresponding interrupt signal is output on the UART3.TX pin for visibility. Legal values are from 0 to 101. 0 is the functional mode; values between 1 and 101 are for observability mode. 0: Default; for i in 1 to 16: observability, UART3.TX pin <= DSP level2 interrupt(i–1); for i in 17 to 37: observability, UART3.TX pin <= DSP level1 interrupt(i–17); for i in 38 to 69: observability, UART3.TX pin <= MPU level1 interrupt(i–38); for i in 70 to 101: observability, UART3.TX pin <= MPU level2 interrupt(i–70);	R/W	0x0000
5–0	RESERVED	Reserved. These bits must always be written as 0.	R/W	0x00000000

At reset, the OMAP5910 device configuration registers are software compatible with previous prototype devices. Writing an 0x0000EAEFh to the compatibility mode control 0 register (COMP_MODE_CTRL_0) enables the new functional multiplexing registers found at offset 0x10h and above.

Table 6–30. Compatibility Mode Control 0 Register (COMP_MODE_CTRL_0)

Bits	Name	Description	R/W	Reset Value
31–16	RESERVED	Reserved for future expansion. These bits must be written to 0x0000h when enabling the OMAP5910 configuration registers.	R	0x0000
15–0	CONF_COMPATIBILITY_R	These bits must be written to 0x0000EAEFh to enable OMAP5910 configuration bits at offset 0x10h and above. Take care to set the configuration bits at 0x10h and above appropriately before writing 0x0000EAEFh to this register.	R/W	0x0000

Table 6–31. Functional Multiplexing Control 3 Register (FUNC_MUX_CTRL_3)

Bits	Name	Description	R/W	Reset Value
31–0	RESERVED	Reserved for future expansion. These bits must always be written as 0.	R/W	0x0

Table 6–32. Functional Multiplexing Control 4 Register (FUNC_MUX_CTRL_4)

Bits	Name	Description	R/W	Reset Value
31–30	RESERVED	Reserved for future expansion. These bits must always be written as 0.	R/W	0x0
29–27	CONF_CAM_D_7_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to CAM.D[7] at reset. The control for this I/O is forced to 000 at reset and while in compatibility mode.	R/W	0x0
26–24	CONF_CAM_LCLK_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to CAM.LCLK at reset. The control for this I/O is forced to 000 at reset and while in compatibility mode.	R/W	0x0

Table 6–32. Functional Multiplexing Control 4 Register (FUNC_MUX_CTRL_4) (Continued)

Bits	Name	Description	R/W	Reset Value
23–21	CONF_CAM_EXCLK_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to CAM.EXCLK at reset. The control for this I/O is forced to 000 at reset and while in compatibility mode.	R/W	0x0
20–18	RESERVED	Reserved for future expansion. These bits must always be written as 0.	R/W	0x0
17–15	CONF_MCBSP1_DOUT_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to MCBSP1.DX at reset. The control for this I/O is forced to 000 at reset and while in compatibility mode.	R/W	0x0
14–12	CONF_MCBSP1_SYNC_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to MCBSP1.FSX at reset. The control for this I/O is forced to 000 at reset and while in compatibility mode.	R/W	0x0
11–0	RESERVED	Reserved for future expansion. These bits must always be written as 0.	R/W	0x0

Table 6–33. Functional Multiplexing Control 5 Register (FUNC_MUX_CTRL_5)

Bits	Name	Description	R/W	Reset Value
31–30	RESERVED	Reserved for future expansion. These bits must always be written as 0.	R/W	0x0
29–27	CONF_CAM_RSTZ_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to CAM.RSTZ at reset. The control for this I/O is forced to 000 at reset and in compatibility mode.	R/W	0x0

Table 6–33. Functional Multiplexing Control 5 Register (FUNC_MUX_CTRL_5) (Continued)

Bits	Name	Description	R/W	Reset Value
26–24	CONF_CAM_HS_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to CAM.HS at reset. The control for this I/O is forced to 000 at reset and in compatibility mode.	R/W	0x0
23–21	CONF_CAM_VS_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to CAM.VS at reset. The control for this I/O is forced to 000 at reset and in compatibility mode.	R/W	0x0
20–18	CONF_CAM_D_0_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to CAM.D[0] at reset. The control for this I/O is forced to 000 at reset and in compatibility mode.	R/W	0x0
17–15	CONF_CAM_D_1_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to CAM.D[1] at reset. The control for this I/O is forced to 000 at reset and in compatibility mode.	R/W	0x0
14–12	CONF_CAM_D_2_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to CAM.D[2] at reset. The control for this I/O is forced to 000 at reset and in compatibility mode.	R/W	0x0
11–9	CONF_CAM_D_3_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to CAM.D[3] at reset. The control for this I/O is forced to 000 at reset and in compatibility mode.	R/W	0x0
8–6	CONF_CAM_D_4_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to CAM.D[4] at reset. The control for this I/O is forced to 000 at reset and in compatibility mode.	R/W	0x0

Table 6–33. Functional Multiplexing Control 5 Register (FUNC_MUX_CTRL_5) (Continued)

Bits	Name	Description	R/W	Reset Value
5–3	CONF_CAM_D_5_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to CAM.D[5] at reset. The control for this I/O is forced to 000 at reset and in compatibility mode.	R/W	0x0
2–0	CONF_CAM_D_6_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to CAM.D[6] at reset. The control for this I/O is forced to 000 at reset and in compatibility mode.	R/W	0x0

Table 6–34. Functional Multiplexing Control 6 Register (FUNC_MUX_CTRL_6)

Bits	Name	Description	R/W	Reset Value
31–30	RESERVED	Reserved for future expansion. These bits must always be written as 0.	R/W	0x0
29–27	CONF_GPIO_4_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to GPIO4 at reset. The control for this I/O is forced to 000 at reset and while in compatibility mode.	R/W	0x0
26–24	CONF_GPIO_6_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to GPIO6 at reset. The control for this I/O is forced to 000 at reset and while in compatibility mode.	R/W	0x0
23–21	CONF_GPIO_7_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to GPIO7 at reset. The control for this I/O is forced to 000 at reset and while in compatibility mode.	R/W	0x0
20–18	CONF_GPIO_11_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to GPIO11 at reset. The control for this I/O is forced to 000 at reset and while in compatibility mode.	R/W	0x0

Table 6–34. Functional Multiplexing Control 6 Register (FUNC_MUX_CTRL_6) (Continued)

Bits	Name	Description	R/W	Reset Value
17–15	CONF_GPIO_12_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to GPIO12 at reset. The control for this I/O is forced to 000 at reset and while in compatibility mode.	R/W	0x0
14–12	CONF_GPIO_13_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to GPIO13 at reset. The control for this I/O is forced to 000 at reset and while in compatibility mode.	R/W	0x0
11–9	CONF_GPIO_14_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to GPIO14 at reset. The control for this I/O is forced to 000 at reset and while in compatibility mode.	R/W	0x0
8–6	CONF_GPIO_15_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to GPIO15 at reset. The control for this I/O is forced to 000 at reset and while in compatibility mode.	R/W	0x0
5–3	CONF_RX3_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to UART3.RX at reset. The control for this I/O is forced to 000 at reset and in compatibility mode.	R/W	0x0
2–0	CONF_TX3_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to UART3.TX at reset. The control for this I/O is forced to 000 at reset and in compatibility mode.	R/W	0x0

Table 6–35. Functional Multiplexing Control 7 Register (FUNC_MUX_CTRL_7)

Bits	Name	Description	R/W	Reset Value
31–21	RESERVED	Reserved for future expansion. These bits must always be written as 0.	R/W	0x0
20–18	CONF_ARMIO_2_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to MPUIO2 at reset The control for this I/O is forced to 000 at reset and while in compatibility mode.	R/W	0x0
17–15	CONF_ARMIO_4_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to MPUIO4 at reset The control for this I/O is forced to 000 at reset and while in compatibility mode.	R/W	0x0
14–12	CONF_ARMIO_5_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to MPUIO5 at reset. The control for this I/O is forced to 000 at reset and while in compatibility mode.	R/W	0x0
11–9	CONF_GPIO_0_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to GPIO0 at reset. The control for this I/O is forced to 000 at reset and while in compatibility mode.	R/W	0x0
8–6	CONF_GPIO_1_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to GPIO1 at reset. The control for this I/O is forced to 000 at reset and while in compatibility mode.	R/W	0x0
5–3	CONF_GPIO_2_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to GPIO2 at reset. The control for this I/O is forced to 000 at reset and while in compatibility mode.	R/W	0x0
2–0	CONF_GPIO_3_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to GPIO3 at reset. The control for this I/O is forced to 000 at reset and while in compatibility mode.	R/W	0x0

Table 6–36. Functional Multiplexing Control 8 Register (FUNC_MUX_CTRL_8)

Bits	Name	Description	R/W	Reset Value
31–30	RESERVED	Reserved for future expansion. These bits must always be written as 0.	R/W	0x0
29–27	CONF_ARM_BOOT_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to MPU_BOOT at reset. The control for this I/O is forced to 000 at reset and while in compatibility mode.	R/W	0x0
26–15	RESERVED	Reserved for future expansion. These bits must always be written as 0.	R/W	0x0
14–12	CONF_WIRE_NSCS3_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to UWIRE.CS3 at reset. The control for this I/O is forced to 000 at reset and in compatibility mode.	R/W	0x0
11–9	CONF_WIRE_NSCS0_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to UWIRE.CS0 at reset. The control for this I/O is forced to 000 at reset and in compatibility mode.	R/W	0x0
8–6	CONF_WIRE_SCLK_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to UWIRE.SCLK at reset. The control for this I/O is forced to 000 at reset and in compatibility mode.	R/W	0x0
5–3	CONF_WIRE_SDO_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to UWIRE.SDO at reset. The control for this I/O is forced to 000 at reset and in compatibility mode.	R/W	0x0
2–0	CONF_WIRE_SDI_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to UWIRE.SDI at reset. The control for this I/O is forced to 000 at reset and in compatibility mode.	R/W	0x0

Table 6–37. Functional Multiplexing Control 9 Register (FUNC_MUX_CTRL_9)

Bits	Name	Description	R/W	Reset Value
31–30	RESERVED	Reserved for future expansion. These bits must always be written as 0.	R/W	0x0
29–27	CONF_UARTS_CLKREQ_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to UART3.CLKREQ at reset. The control for this I/O is forced to 000 at reset and in compatibility mode.	R/W	0x0
26–24	CONF_MCSI1_DOUT_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to MCSI1.DOUT at reset. The control for this I/O is forced to 000 at reset and in compatibility mode.	R/W	0x0
23–21	CONF_TX1_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to UART1.TX at reset. The control for this I/O is forced to 000 at reset and in compatibility mode.	R/W	0x0
20–15	RESERVED	Reserved for future expansion. These bits must always be written as 0.	R/W	0x0
14–12	CONF_RTS1_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to UART1.RTS at reset. The control for this I/O is forced to 000 at reset and in compatibility mode.	R/W	0x0
11–6	RESERVED	Reserved for future expansion. These bits must always be written as 0.	R/W	0x0
5–3	CONF_MCBSP3_CLK_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to MCBSP3.CLKX at reset. The control for this I/O is forced to 000 at reset and in compatibility mode.	R/W	0x0
2–0	CONF_COM_ SHUTDOWN_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to RST_HOST_OUT at reset. The control for this I/O is forced to 000 at reset and in compatibility mode.	R/W	0x0

Table 6–38. Functional Multiplexing Control A Register (FUNC_MUX_CTRL_A)

Bits	Name	Description	R/W	Reset Value
31–27	RESERVED	Reserved for future expansion. These bits must always be written as 0.	R/W	0x0
26–24	CONF_MMC_DAT1_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to MMC.DAT1 at reset. As long as the STATIC_VALID pin is sampled high upon reset, the control for this I/O is force to 000 at reset and while in compatibility mode. STATIC_VALID must sample high at reset for the associated OMAP5910 pin to function properly.	R/W	0x0
23–21	RESERVED	Reserved. These bits must always be written as 0.	R/W	0x0
20–18	CONF_MMC_DAT2_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to MMC.DAT2 at reset. As long as the STATIC_VALID pin is sampled high upon reset, the control for this I/O is force to 000 at reset and while in compatibility mode. STATIC_VALID must sample high at reset for the associated OMAP5910 pin to function properly.	R/W	0x0
17–15	RESERVED	Reserved for future expansion. These bits must always be written as 0.	R/W	0x0
14–12	CONF_CLK32K_OUT_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to CLK32K_OUT at reset. The control for this I/O is forced to 000’at reset and in compatibility mode.	R/W	0x0
11–9	CONF_MCSI1_DIN_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to MCSI1.DIN at reset. The control for this I/O is forced to 000 at reset and in compatibility mode.	R/W	0x0
8–6	CONF_MCSI1_BCLK_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to MCSI1.CLK at reset. The control for this I/O is forced to 000’at reset and in compatibility mode.	R/W	0x0

Table 6–38. Functional Multiplexing Control A Register (FUNC_MUX_CTRL_A) (Continued)

Bits	Name	Description	R/W	Reset Value
5–3	CONF_MCSI1_SYNC_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to MCSI1.SYNC at reset. The control for this I/O is forced to 000 at reset and in compatibility mode.	R/W	0x0
2–0	CONF_UARTS_CLKIO_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to BCLK at reset. The control for this I/O is forced to 000 at reset and in compatibility mode.	R/W	0x0

Table 6–39. Functional Multiplexing Control B Register (FUNC_MUX_CTRL_B)

Bits	Name	Description	R/W	Reset Value
31–21	RESERVED	Reserved for future expansion. These bits must always be written as 0.	R/W	0x0
20–18	CONF_COM_MCLK_REQ_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to UART2.CLKREQ at reset. The control for this I/O is forced to 000 at reset and while in compatibility mode.	R/W	0x0
17–15	RESERVED	Reserved for future expansion. These bits must always be written as 0.	R/W	0x0
14–12	CONF_MCSI2_SYNC_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to MCSI2.SYNC at reset. The control for this I/O is forced to 000 at reset and in compatibility mode.	R/W	0x0
11–9	CONF_MCSI2_DOUT_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to MCSI2.DOUT at reset. The control for this I/O is forced to 000 at reset and in compatibility mode.	R/W	0x0

Table 6–39. Functional Multiplexing Control B Register (FUNC_MUX_CTRL_B) (Continued)

Bits	Name	Description	R/W	Reset Value
8–6	CONF_MCSI2_DIN_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to MCSI2.DIN at reset. The control for this I/O is forced to 000 at reset and in compatibility mode.	R/W	0x0
5–3	CONF_MCSI2_CLK_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to MCSI2.CLK at reset. The control for this I/O is forced to 000 at reset and in compatibility mode.	R/W	0x0
2–0	RESERVED	Reserved for future expansion. These bits must always be written as 0.	R/W	0x0

Table 6–40. Functional Multiplexing Control C Register (FUNC_MUX_CTRL_C)

Bits	Name	Description	R/W	Reset Value
31–30	RESERVED	Reserved for future expansion. These bits must always be written as 0.	R/W	0x0
29–27	CONF_TX2_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to UART2.TX at reset. The control for this I/O is forced to 000 at reset and in compatibility mode.	R/W	0x0
26–24	CONF_RTS2_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to UART2.RTS at reset. The control for this I/O is forced to 000’at reset and in compatibility mode.	R/W	0x0
23–21	CONF_CTS2_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to UART2.CTS at reset. The control for this I/O is forced to 000 at reset and while in compatibility mode.	R/W	0x0

Table 6–40. Functional Multiplexing Control C Register (FUNC_MUX_CTRL_C) (Continued)

Bits	Name	Description	R/W	Reset Value
20–18	CONF_RX2_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to UART2.RX at reset. The control for this I/O is forced to 000 at reset and while in compatibility mode.	R/W	0x0
17–15	CONF_MCBSP2_DOUT_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to MCBSP2.DOUT at reset. The control for this I/O is forced to 000 at reset and while in compatibility mode.	R/W	0x0
14–12	CONF_MCBSP2_RSYNC_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to MCBSP2.FSR at reset. The control for this I/O is forced to 000 at reset and while in compatibility mode.	R/W	0x0
11–9	RESERVED	Reserved for future expansion. These bits must always be written as 0.	R/W	0x0
8–6	CONF_MCBSP2_CLKR_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to MCBSP2.CLKR at reset. The control for this I/O is forced to 000 at reset and while in compatibility mode.	R/W	0x0
5–3	RESERVED	Reserved for future expansion. These bits must always be written as 0.	R/W	0x0
2–0	CONF_MCBSP2_DIN_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to MCBSP2.DR at reset. The control for this I/O is forced to 000 at reset and while in compatibility mode.	R/W	0x0

Table 6–41. Functional Multiplexing Control D Register (FUNC_MUX_CTRL_D)

Bits	Name	Description	R/W	Reset Value
31–15	RESERVED	Reserved for future expansion. These bits must always be written as 0.	R/W	0x00000
14–12	CONF_MMC_DAT3_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to MMC.DAT3 at reset The control for this I/O is forced to 000 at reset.	R/W	0x0
11–9	RESERVED	Reserved for future expansion. These bits must always be written as 0.	R/W	0x0
8–6	CONF_NFCS2_R	These bits control the multiplexing on the OMAP5910 I/O, which defaults to FLASH.CS2 at reset. The control for this I/O is forced to 000 at reset.	R/W	0x0
5–0	RESERVED	Reserved for future expansion. These bits must always be written as 0.	R/W	0x0

Table 6–42. Pulldown Control 0 Register (PULL_DWN_CTRL_0)

Bits	Name	Value	Description (see Note)	R/W	Reset Value
31–29	RESERVED		Reserved for future expansion. These bits must always be written as 0.	R/W	0x0
28	CONF_PDEN_CAM_HS_R		These bits control the pulldown enable on the OMAP5910 I/O, which defaults to CAM.HS at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
27–25	RESERVED		Reserved for future expansion. These bits must always be written as 0.	R/W	0x0
24	CONF_PDEN_CAM_D_2_R		These bits control the pulldown enable on the OMAP5910 I/O, which defaults to CAM.D[2] at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
23	CONF_PDEN_CAM_D_3_R		These bits control the pulldown enable on the OMAP5910 I/O, which defaults to CAM.D[3] at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
22	RESERVED		Reserved for future expansion. These bits must always be written as 0.	R/W	0x0
21	CONF_PDEN_CAM_D_5_R		These bits control the pulldown enable on the OMAP5910 I/O, which defaults to CAM.D[5] at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		

Note: Unless otherwise indicated, pulldown control for each I/O is forced off at reset while in compatibility mode. The pulldown control register bits only control the pulldowns while in native mode. Depending upon the pin multiplexing configuration of any particular I/O, a pulldown may not be available. Consult Appendix A of this document or the OMAP5910 data manual (literature number SPRS197) to determine whether a pulldown exists for each I/O.

Table 6–42. Pulldown Control 0 Register (PULL_DWN_CTRL_0) (Continued)

Bits	Name	Value	Description (see Note)	R/W	Reset Value
20–17	RESERVED		Reserved for future expansion. These bits must always be written as 0.	R/W	0x0
16	CONF_PDEN_MCBSP1_DIN_R		These bits control the pulldown enable on the OMAP5910 I/O, which defaults to MCBSP1.DR at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
15–0	RESERVED		Reserved for future expansion. These bits must always be written as 0.	R/W	0x0

Note: Unless otherwise indicated, pulldown control for each I/O is forced off at reset while in compatibility mode. The pulldown control register bits only control the pulldowns while in native mode. Depending upon the pin multiplexing configuration of any particular I/O, a pulldown may not be available. Consult Appendix A of this document or the OMAP5910 data manual (literature number SPRS197) to determine whether a pulldown exists for each I/O.

Table 6–43. Pulldown Control 1 Register (PULL_DWN_CTRL_1)

Bit	Name	Value	Description (See Note)	R/W	Reset Value
31–30	RESERVED		Reserved for future expansion. These bits must always be written as 0.	R/W	0x0
29	CONF_PDEN_MCBSP3_CLK_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to MCBSP3.CLKX at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
28	RESERVED		Reserved for future expansion. These bits must always be written as 0.	R/W	0x0

Note: Unless otherwise indicated, pulldown control for each I/O is forced off at reset while in compatibility mode. The pulldown control register bits only control the pulldowns while in native mode. Depending upon the pin multiplexing configuration of any particular I/O, a pulldown may not be available. Consult Appendix A of this document or the OMAP5910 data manual (literature number SPRS197) to determine whether a pulldown exists for each I/O.

Table 6–43. Pulldown Control 1 Register (PULL_DWN_CTRL_1) (Continued)

Bit	Name	Value	Description (See Note)	R/W	Reset Value
27	CONF_PDEN_ARM_BOOT_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to MPU_BOOT at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
			Control for this pulldown is forced on at reset and while in compatibility mode.		
26	CONF_PDEN_NEMU1_R		This bit controls the pullup enable on the OMAP5910 I/O, which defaults to EMU1 at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
25	CONF_PDEN_NEMU0_R		This bit controls the pullup enable on the OMAP5910 I/O, which defaults to EMU0 at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
24–19	RESERVED		Reserved for future expansion. These bits must always be written as 0.	R/W	0x0
18	CONF_PDEN_WIRE_SDI_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to UWIRE.SDI at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
17–15	PULLDOWN		Reserved for future expansion. These bits must always be written as 0.	R/W	0x0

Note: Unless otherwise indicated, pulldown control for each I/O is forced off at reset while in compatibility mode. The pulldown control register bits only control the pulldowns while in native mode. Depending upon the pin multiplexing configuration of any particular I/O, a pulldown may not be available. Consult Appendix A of this document or the OMAP5910 data manual (literature number SPRS197) to determine whether a pulldown exists for each I/O.

Table 6–43. Pulldown Control 1 Register (PULL_DWN_CTRL_1) (Continued)

Bit	Name	Value	Description (See Note)	R/W	Reset Value
14	CONF_PDEN_ARMIO_2_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to MPUIO2 at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
			The control for this pulldown is forced on at reset and while in compatibility mode.		
13	CONF_PDEN_ARMIO_4_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to MPUIO4 at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
			The control for this pulldown is forced on at reset and while in compatibility mode.		
12	CONF_PDEN_ARMIO_5_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to MPUIO5 at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
			The control for this pulldown is forced on at reset and while in compatibility mode.		

Note: Unless otherwise indicated, pulldown control for each I/O is forced off at reset while in compatibility mode. The pulldown control register bits only control the pulldowns while in native mode. Depending upon the pin multiplexing configuration of any particular I/O, a pulldown may not be available. Consult Appendix A of this document or the OMAP5910 data manual (literature number SPRS197) to determine whether a pulldown exists for each I/O.

Table 6–43. Pulldown Control 1 Register (PULL_DWN_CTRL_1) (Continued)

Bit	Name	Value	Description (See Note)	R/W	Reset Value
11	CONF_PDEN_GPIO_0_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to GPIO0 at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
			The control for this pulldown is forced on at reset and while in compatibility mode.		
10	CONF_PDEN_GPIO_1_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to GPIO1 at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
			The control for this pulldown is forced on at reset and while in compatibility mode.		
9	CONF_PDEN_GPIO_2_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to GPIO2 at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
			The control for this pulldown is forced on at reset and while in compatibility mode.		

Note: Unless otherwise indicated, pulldown control for each I/O is forced off at reset while in compatibility mode. The pulldown control register bits only control the pulldowns while in native mode. Depending upon the pin multiplexing configuration of any particular I/O, a pulldown may not be available. Consult Appendix A of this document or the OMAP5910 data manual (literature number SPRS197) to determine whether a pulldown exists for each I/O.

Table 6–43. Pulldown Control 1 Register (PULL_DWN_CTRL_1) (Continued)

Bit	Name	Value	Description (See Note)	R/W	Reset Value
8	CONF_PDEN_GPIO_3_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to GPIO3 at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
			The control for this pulldown is forced on at reset and while in compatibility mode.		
7	CONF_PDEN_GPIO_4_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to GPIO4 at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
			The control for this pulldown is forced on at reset and while in compatibility mode.		
6	CONF_PDEN_GPIO_6_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to GPIO6 at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
			The control for this pulldown is forced on at reset and while in compatibility mode.		

Note: Unless otherwise indicated, pulldown control for each I/O is forced off at reset while in compatibility mode. The pulldown control register bits only control the pulldowns while in native mode. Depending upon the pin multiplexing configuration of any particular I/O, a pulldown may not be available. Consult Appendix A of this document or the OMAP5910 data manual (literature number SPRS197) to determine whether a pulldown exists for each I/O.

Table 6–43. Pulldown Control 1 Register (PULL_DWN_CTRL_1) (Continued)

Bit	Name	Value	Description (See Note)	R/W	Reset Value
5	CONF_PDEN_GPIO_7_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to GPIO7 at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
			The control for this pulldown is forced on at reset and while in compatibility mode.		
4	CONF_PDEN_GPIO_11_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to GPIO11 at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
			The control for this pulldown is forced on at reset and while in compatibility mode.		
3	CONF_PDEN_GPIO_12_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to GPIO12 at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
			The control for this pulldown is forced on at reset and while in compatibility mode.		

Note: Unless otherwise indicated, pulldown control for each I/O is forced off at reset while in compatibility mode. The pulldown control register bits only control the pulldowns while in native mode. Depending upon the pin multiplexing configuration of any particular I/O, a pulldown may not be available. Consult Appendix A of this document or the OMAP5910 data manual (literature number SPRS197) to determine whether a pulldown exists for each I/O.

Table 6–43. Pulldown Control 1 Register (PULL_DWN_CTRL_1) (Continued)

Bit	Name	Value	Description (See Note)	R/W	Reset Value
2	CONF_PDEN_GPIO_13_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to GPIO13 at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
			The control for this pulldown is forced on at reset and while in compatibility mode.		
1	CONF_PDEN_GPIO_14_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to GPIO14 at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
			The control for this pulldown is forced on at reset and while in compatibility mode.		
0	CONF_PDEN_GPIO_15_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to GPIO15 at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
			The control for this pulldown is forced on at reset and while in compatibility mode.		

Note: Unless otherwise indicated, pulldown control for each I/O is forced off at reset while in compatibility mode. The pulldown control register bits only control the pulldowns while in native mode. Depending upon the pin multiplexing configuration of any particular I/O, a pulldown may not be available. Consult Appendix A of this document or the OMAP5910 data manual (literature number SPRS197) to determine whether a pulldown exists for each I/O.

Table 6–44. Pulldown Control 2 Register (PULL_DWN_CTRL_2)

Bit	Name	Value	Description (See Note)	R/W	Reset Value
31	CONF_PDEN_MCBSP2_ DOUT_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to MCBSP2.DX at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
30	CONF_PDEN_MCBSP2_ RSYNC_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to MCBSP2.FSR at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
29	CONF_PDEN_MCBSP2_ CLKX_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to MCBSP2.CLKX at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
			The control for this pulldown is forced on at reset and while in compatibility mode.		
28	CONF_PDEN_MCBSP2_ CLKR_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to MCBSP2.CLKR at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		

Note: Unless otherwise indicated, pulldown control for each I/O is forced off at reset while in compatibility mode. The pulldown control register bits only control the pulldowns while in native mode. Depending upon the pin multiplexing configuration of any particular I/O, a pulldown may not be available. Consult Appendix A of this document or the OMAP5910 data manual (literature number SPRS197) to determine whether a pulldown exists for each I/O.

Table 6–44. Pulldown Control 2 Register (PULL_DWN_CTRL_2) (Continued)

Bit	Name	Value	Description (See Note)	R/W	Reset Value
27	CONF_PDEN_MCBSP2_ XSYNC_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to MCBSP2.FSX at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
			The control for this pulldown is forced on at reset and while in compatibility mode.		
26	CONF_PDEN_MCBSP2_ DIN_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to MCBSP2.DR at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
25	CONF_PDEN_ ARMIO_3_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to MPUIO3 at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
			The control for this pulldown is forced on at reset and while in compatibility mode.		
24	CONF_PDEN_GPIO_8_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to GPIO.8 at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
			The control for this pulldown is forced on at reset and while in compatibility mode.		

Note: Unless otherwise indicated, pulldown control for each I/O is forced off at reset while in compatibility mode. The pulldown control register bits only control the pulldowns while in native mode. Depending upon the pin multiplexing configuration of any particular I/O, a pulldown may not be available. Consult Appendix A of this document or the OMAP5910 data manual (literature number SPRS197) to determine whether a pulldown exists for each I/O.

Table 6–44. Pulldown Control 2 Register (PULL_DWN_CTRL_2) (Continued)

Bit	Name	Value	Description (See Note)	R/W	Reset Value
23	CONF_PDEN_GPIO_9_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to GPIO.9 at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
			The control for this pulldown is forced on at reset and while in compatibility mode.		
22	CONF_PDEN_COM_ MCLK_REQ_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to UART2.CLKREQ at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
			The control for this pulldown is forced on at reset and while in compatibility mode.		
21	RESERVED		Reserved for future expansion. These bits must always be written as 0.	R/W	0x0
20	CONF_PDEN_MCSI2_ SYNC_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to MCSI2.SYNC at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
19	RESERVED		Reserved for future expansion. These bits must always be written as 0.	R/W	0x0
18	CONF_PDEN_MCSI2_ DIN_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to MCSI2.DIN at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		

Note: Unless otherwise indicated, pulldown control for each I/O is forced off at reset while in compatibility mode. The pulldown control register bits only control the pulldowns while in native mode. Depending upon the pin multiplexing configuration of any particular I/O, a pulldown may not be available. Consult Appendix A of this document or the OMAP5910 data manual (literature number SPRS197) to determine whether a pulldown exists for each I/O.

Table 6–44. Pulldown Control 2 Register (PULL_DWN_CTRL_2) (Continued)

Bit	Name	Value	Description (See Note)	R/W	Reset Value
17	CONF_PDEN_MCSI2_CLK_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to MCSI2.CLK at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
16	CONF_PDEN_MMC_DAT0_R		This bit controls the pullup enable on the OMAP5910 I/O, which defaults to MMC.DAT0 at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
15	CONF_PDEN_MMC_CMD_R		This bit controls the pullup enable on the OMAP5910 I/O, which defaults to MMC.CMD_SPI.DO at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
14	CONF_PDEN_MMC_DAT1_R		This bit controls the pullup enable on the OMAP5910 I/O, which defaults to MMC.DAT1 at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
13	RESERVED		Reserved for future expansion. These bits must always be written as 0.	R/W	0x0
12	CONF_PDEN_MMC_DAT2_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to MMC.DAT2 at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		

Note: Unless otherwise indicated, pulldown control for each I/O is forced off at reset while in compatibility mode. The pulldown control register bits only control the pulldowns while in native mode. Depending upon the pin multiplexing configuration of any particular I/O, a pulldown may not be available. Consult Appendix A of this document or the OMAP5910 data manual (literature number SPRS197) to determine whether a pulldown exists for each I/O.

Table 6–44. Pulldown Control 2 Register (PULL_DWN_CTRL_2) (Continued)

Bit	Name	Value	Description (See Note)	R/W	Reset Value
11–10	RESERVED		Reserved for future expansion. These bits must always be written as 0.	R/W	0x0
9	CONF_PDEN_MCSI1_ DIN_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to MCSI1.DIN at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
8	CONF_PDEN_MCSI1_ BCLK_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to MCSI1.CLK at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
			The control for this pulldown is forced on at reset and while in compatibility mode.		
7	CONF_PDEN_MCSI1_ SYNC_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to MCSI1.SYNC at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
			The control for this pulldown is forced on at reset and while in compatibility mode.		
6	CONF_PDEN_UARTS_ CLKIO_R		Reserved for future expansion. These bits must always be written as 0.	R/W	0x0

Note: Unless otherwise indicated, pulldown control for each I/O is forced off at reset while in compatibility mode. The pulldown control register bits only control the pulldowns while in native mode. Depending upon the pin multiplexing configuration of any particular I/O, a pulldown may not be available. Consult Appendix A of this document or the OMAP5910 data manual (literature number SPRS197) to determine whether a pulldown exists for each I/O.

Table 6–44. Pulldown Control 2 Register (PULL_DWN_CTRL_2) (Continued)

Bit	Name	Value	Description (See Note)	R/W	Reset Value
5	CONF_PDEN_UARTS_ CLKREQ_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to UART3.CLKREQ at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
			The control for this pulldown is forced on at reset and while in compatibility mode.		
4:3	RESERVED		Reserved for future expansion. These bits must always be written as 0.	R/W	0x0
2	CONF_PDEN_RX1_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to UART1.RX at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
1	CONF_PDEN_R_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to UART1.CTS at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
0	RESERVED		Reserved for future expansion. These bits must always be written as 0.	R/W	0x0

Note: Unless otherwise indicated, pulldown control for each I/O is forced off at reset while in compatibility mode. The pulldown control register bits only control the pulldowns while in native mode. Depending upon the pin multiplexing configuration of any particular I/O, a pulldown may not be available. Consult Appendix A of this document or the OMAP5910 data manual (literature number SPRS197) to determine whether a pulldown exists for each I/O.

Table 6–45. Pulldown Control 3 Register (PULL_DWN_CTRL_3)

Bit	Name	Value	Description	R/W	Reset Value
31–14	RESERVED		Reserved for future expansion. These bits must always be written as 0.	R/W	0x00000
13	CONF_PDEN_NTRST_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to $\overline{\text{TRST}}$ at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
12	CONF_PDEN_TCK_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to TCK at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
11	CONF_PDEN_TMS_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to TMS at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
10	CONF_PDEN_TDI_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to TDI at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
9	CONF_PDEN_CONF_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to CONF at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		

Table 6–45. Pulldown Control 3 Register (PULL_DWN_CTRL_3) (Continued)

Bit	Name	Value	Description	R/W	Reset Value
8	CONF_PDEN_MMC_DAT3_R		This bit controls the pullup enable on the OMAP5910 I/O, which defaults to MMC.DAT3 at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
7–2	RESERVED		Reserved for future expansion. These bits must always be written as 0.	R/W	0x0
1	CONF_PDEN_CTS2_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to UART2.CTS at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		
0	CONF_PDEN_RX2_R		This bit controls the pulldown enable on the OMAP5910 I/O, which defaults to UART2.RX at reset.	R/W	0x0
		0	Pulldown enabled		
		1	Pulldown disabled		

Table 6–46. Gate and Inhibit Control 0 Register (GATE_INH_CTRL_0)

Bit	Name	Value	Description	R/W	Reset Value
31–4	RESERVED		Reserved for future expansion. These bits must always be written as 0.	R/W	0x0000000
3	CONF_HIGH_IMP3		This bit is for control of high-impedance on MCSI1.DOUT.	R/W	0x0
		0	Normal function		
		1	Hi-impedance		

Table 6–46. Gate and Inhibit Control 0 Register (GATE_INH_CTRL_0) (Continued)

Bit	Name	Value	Description	R/W	Reset Value
2	CONF_ SOFTWARE_PWR_R		This bit controls software gating and inhibiting of the OMAP5910 I/O, which are gated or inhibited by COM_PWR status. If the gating and inhibiting logic are 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. This bit has no effect in compatibility mode.	R/W	0x0
1	CONF_ SOFTWARE_BVLZ_R		This bit controls software gating and inhibiting of the OMAP5910 I/O, which are gated or inhibited by BFAIL/EXT_FIQ. If the gating and inhibiting logic are enabled by FUNC_MUX_CTRL_0 (10–13) bits and conf_software_gate_ena_r is set to 1, this bit controls the BFAIL/EXT_FIQ gating and inhibiting instead of device pins. This bit has no effect in compatibility mode.	R/W	0x0
0	CONF_ SOFTWARE_ GATE_ENA_R		This bit controls software gating of the OMAP5910 I/O, which are gated or inhibited. If the gating and inhibiting logic are enabled by FUNC_MUX_CTRL_0 (10–13) bits, this enables software to control the gating and inhibiting instead of device pins. This bit has no effect in compatibility mode.	R/W	0x0

Table 6–47. Voltage Control 0 Register (VOLTAGE_CTRL_0)

Bit	Name	Value	Description	R/W	Reset Value
31–3	RESERVED		Reserved for future expansion. These bits must always be written as 0.	R/W	0x0000000
2	CONF_VOLTAGE_COMIF_R		This bit controls the drive strength of the OMAP5910 communication processor interface I/O. This allows the interface to be run at 1.8 V nom or 2.75 V nom.	R/W	0x0
		0	Drive strength is 1.80 V		
		1	Drive strength is 2.75 V		
			At reset and in compatibility mode, the interface is set for 2.75-V operation. This register only controls the interface in OMAP5910 mode.		
1	CONF_VOLTAGE_SDRAM_R		This bit controls the drive strength of the OMAP5910 SDRAM interface I/O. This allows the interface to be run at 1.8 V nom or 2.75 V nom.	R/W	0x0
		0	Drive strength is 1.80 V		
		1	Drive strength is 2.75 V		
			At reset and in compatibility mode, the interface is set for 2.75-V operation. This register only controls the interface in OMAP5910 mode.		

Table 6–47. Voltage Control 0 Register (VOLTAGE_CTRL_0) (Continued)

Bit	Name	Value	Description	R/W	Reset Value
0	CONF_VOLTAGE_FLASH_R		This bit controls the drive strength of the OMAP5910 flash interface I/O. This allows the interface to be run at 1.8 V nom or 2.75 V nom.	R/W	0x0
		0	Drive strength is 1.80 V		
		1	Drive strength is 2.75 V		
			At reset and in compatibility mode, the interface is set for 2.75-V operation. This register only controls the interface in OMAP5910 mode.		

Table 6–48. Test Debug Control 0 Register (TEST_DBG_CTRL_0)

Bit	Name	Description	R/W	Reset Value
31–0	RESERVED	These register is reserved for factory testing purposes. All bits must be 0 at all times to avoid errant behavior.	R/W	0x00000000

Table 6–49. Module Configuration Control 0 Register (MOD_CONF_CTRL_0)

Bit	Name	Value	Description	R/W	Reset Value
31	CONF_MOD_UART3_ CLK_MODE_R		This bit determines the clock source of UART3 on the OMAP5910 device.	R/W	0x0
		0	12 MHz		
		1	48 MHz		
30	CONF_MOD_UART2_ CLK_MODE_R		This bit determines the clock source of UART2 on the OMAP5910 device.	R/W	0x0
		0	32 kHz/12 MHz (see Chapter 12, *UART Devices*)		
		1	48 MHz		
29	CONF_MOD_UART1_ CLK_MODE_R		This bit determines the clock source of UART1 on the OMAP5910 device.	R/W	0x0
		0	12 MHz		
		1	48 MHz		
28	MOD_MCBSP3_MODE_R		This bit determines the method of frame synchronization wrap-around used on MCBSP3.	R/W	0x0
		0	Wrap-around done in hardware external to the McBSP.		
		1	Wrap-around disabled. Wrap around can be performed within the McBSP module.		
			Modes documented in Chapter 9, *DSP Public Peripherals*.		
27–24	MOD_32KOSC_SW_R		These bits determine the configuration of the the 32-kHz oscillator. The reset condition corresponds to a fast start-up time.	R/W	0x0
		1011	Fast start-up time		
		1000	Lowest-power mode		
			These bits are forced to 1011 during reset and in compatibility mode. The user must take care to program these bits appropriately before entering native mode.		

Table 6–49. Module Configuration Control 0 Register (MOD_CONF_CTRL_0) (Continued)

Bit	Name	Value	Description	R/W	Reset Value
23	CONF_MOD_MMC_SD_CLK_REQ_R		This is the functional 48-MHz clock request for the OMAP5910 device MMC/SD interface.	R/W	0x0
			This bit resets to 0 at reset. This corresponds to the MMC/SD clock not being requested. Set the bit to 1 to request the clock for the MMC/SD interface.		
22	CONF_MOD_DPRAM_ENABLE_R		This bit controls the DPRAM I/F of the OMAP5910 device.	R/W	0x0
		0	Normal flash interface operation		
		1	$\overline{\text{FLASH.CS2}}$ assertion low is delayed to allow for a DPRAM to be interfaced to the flash interface of OMAP5910.		
21	CONF_MOD_MSMMC_VSS_HIZ_OVERRIDE		This bit disables the forced HI-Z on the the MMC.DAT2 pin of the device. In order to use this pin in a functional mode, the user must set this bit to a 1.	R/W	0x0
20	CONF_MOD_MCBSP3_AUXON		This bit enables the McBSP3 AUXON functionality, which gates the functional clock to the corresponding McBSP module.	R/W	0x0
		0	The internal functional clock to McBSP3 is active and depends upon the McBSP configuration.		
		1	The internal functional clock to McBSP3 is disabled or gated.		
19	CONF_MOD_MCBSP2_AUXON		This bit enables the McBSP2 AUXON functionality, which gates the functional clock to the corresponding McBSP module.	R/W	0x0
		0	The internal functional clock to McBSP2 is active and depends upon the McBSP configuration.		
		1	The internal functional clock to McBSP2 is disabled or gated.		

Table 6–49. Module Configuration Control 0 Register (MOD_CONF_CTRL_0) (Continued)

Bit	Name	Value	Description	R/W	Reset Value
18	CONF_MOD_MCBSP1_ AUXON		This bit enables the McBSP1 AUXON functionality, which gates the functional clock to the corresponding McBSP module.	R/W	0x0
		0	The internal functional clock to McBSP1 is active and depends upon the McBSP configuration.		
		1	The internal functional clock to McBSP1 is disabled or gated.		
17	CONF_MOD_USB_W2FC_ VBUS_MODE_R		This bit determines what hardware method is used for USB.VBUS detection.	R/W	0x0
		0	The VBUS detection is under control of the GPIO0 input.		
		1	The VBUS detection is under control of the VBUS detection I/O cell.		
			This bit resets to 0 during reset and compatibility mode.		
16	CONF_MOD_I2C_ SELECT_R		This bit selects the I^2C module compatibility mode. This bit resets to standard mode.	R/W	0x0
		0	The I^2C module is in standard mode.		
		1	The I^2C module is in compatibility mode.		
15–14	RESERVED		Reserved for future expansion. These bits must always be written as 0.	R/W	0x0
13	CONF_MOD_SDRAM_ EMRS_BA1_CTRL		This bit allows the user to force the SDRAM SDRAM.BA[1] pin to a high. With proper disabling of SDRAM accesses from OMAP5910, users can use this to program the EMRS register of the SDRAM with an MRS write instruction.	R/W	0x0
			There are no hardware hooks to only assert this when performing an MRS write. Firmware must determine how to properly control this.		

Table 6–49. Module Configuration Control 0 Register (MOD_CONF_CTRL_0) (Continued)

Bit	Name	Value	Description	R/W	Reset Value
12	CONF_MOD_COM_ MCLK_12_48_SEL_R		This bit determines if the UART2.CLKREQ output of the OMAP5910 device is 12 MHz or 48 MHz.	R/W	0x0
			This bit resets to 0, which causes a 12-MHz clock to be seen on MCLK when UART2.CLKREQ is low. When written to a 1, this bit causes 48-MHz clock to be seen on MCLK when UART2.CLKREQ is low. When 1, UART2.CLKREQ also starts the 12-MHz to 48-MHz DPLL.		
11	CONF_MOD_USB_HOST_ UART_SELECT_R		This bit enables the multiplexing of UART1.CTS, UART1.RX, and UART1.TX signals to the USB_HMC host mux module.	R/W	0x0
		0	UART1 uses the standard source location as defined by the OMAP5910 functional multiplexing.		
		1	UART1.TX, UART1.RX, and UART1.CTS1 are sourced from the USB_HMC module.		
			For details on this multiplexing please see the USB_HMC spec.		
10	RESERVED		Reserved for future expansion. This bit must always be written as 0.	R/W	0x0
9	CONF_MOD_USB_HOST_ HHC_UHOST_EN_R		Enable input for functional-mode clocking of USB_HHC	R/W	0x0
		0	Internal functional mode 48-MHz and 12-MHz clocks are disabled; USB_HHC can not function as a USB host.		
		1	Internal functional mode 48-MHz and 12-MHz clocks are enabled.		

Table 6–49. Module Configuration Control 0 Register (MOD_CONF_CTRL_0) (Continued)

Bit	Name	Value	Description	R/W	Reset Value
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, determines whether the modelling of the device pullup resistor is on the internal D+ or internal D– signal. The pullup is only modeled when HMC_TLL_ATTACH_I is active. This signal is ignored when either device drives USB data and whenever HMC_MODE or HMC_JTAG_EN_I specify that the TLL is not being used.	R/W	0x0
		0	When HMC_TLL_ATTACH_I is high, 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, 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.		

Table 6–49. Module Configuration Control 0 Register (MOD_CONF_CTRL_0) (Continued)

Bit	Name	Value	Description	R/W	Reset Value
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, 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.	R/W	0x0
		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 attached device.		
		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 attached device with the bus in an IDLE condition.		
6–1	CONF_MOD_USB_HOST_HMC_MODE_R		USB_HHC port multiplexing control. See section 15.5, *USB Pin Multiplexing*, for details. This resets to the following configuration:	R/W	0x00
		000000b	USB port 0 is controlled by the USB function, and USB ports 1 and 2 are held in benign states.		
			All others: See section 15.5, *USB Pin Multiplexing*.		
0	RESERVED		Reserved for future expansion. This bit should always be written as 0.	R/W	0x0

6.9 Device Identification

The device identification can be done by software via two registers:

- The identification code (IDCODE) register identifies the OMAP5910 device.
- The identification die (ID) register identifies the die.

6.9.1 Identification Code Register

The identification code register (IDCODE), shown in Table 6–50, can be split into four fields:

- VERSION number (4 bits) (MSB) 31 to 28
- PART number (16 bits) 27 to 12
- Manufacturer Identity (11 bits) 11 to 1
- Fixed LSB (1 bit) (LSB) 0

Table 6–50. ID Code Register (IDCODE)

Register Name	Size	Access	Capture Value	Address
IDCODE	32	R	See below	FFFE:D404

The TI manufacturer identity is IEEE WW defined as 000 0001 0111.

For OMAP5910 design:
ID code = xxxx 1011 0100 0111 0000 0000 0010 1111 = xb47002f

The IDCODE register bits are described in Table 6–51.

Table 6–51. ID Code Register (IDCODE) Bits

Field	Binary Value	Decimal Value	Hex Value
Version number	xxxx [31–28]	x	x
Part number	1011 0100 0111 0000 [27–12]	46192	0xB470
Manufacturer identity	000 0001 0111 [11–1]	23	0x17
Fixed LSB	1	1	0x1

6.9.2 Die Identification (ID)

An electrically readable die ID permits tracing of individual dies back to manufacturing data and aids in rapid ramp. Access to the die ID by both the tester and the application is necessary.

The die ID is a 64-bit code. Of these, 56 bits (bits 0–55) are data bits that contain x and y coordinates of the die, wafer number, lot number, and manufacturing number. Eight bits (bits 56–63) are check bits computed using a Hamming code.

The die ID can be read by software via the private TIPB (see Table 6–52).

Table 6–52. Die ID Address Space—Private TIPB Bridge

Device Name	Start Address	Size in Bytes	Data Access	
OMAP5910 Die ID	FFFE:1800	4 bytes	32	LSB
OMAP5910 Die ID	FFFE:1804	4 bytes	32	MSB

Chapter 7

MPU Public Peripherals

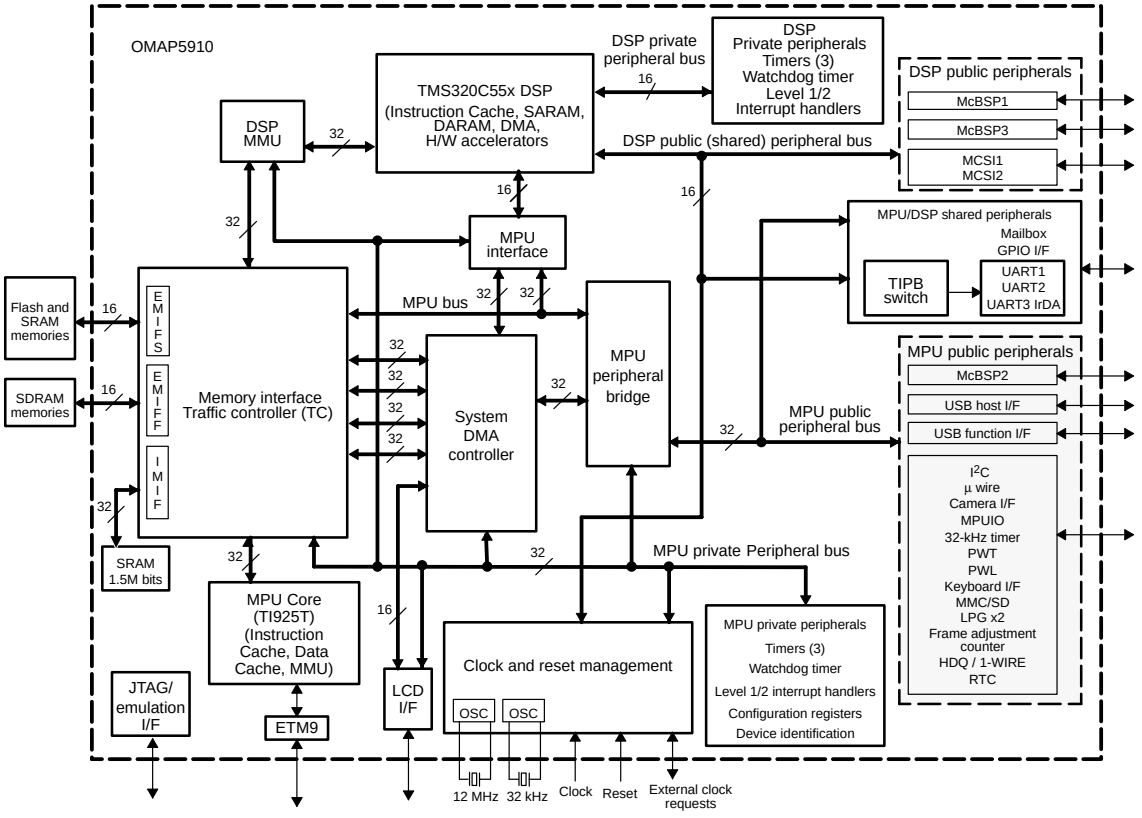
This chapter describes the MPU public peripherals.

Topic **Page**

7.1 MPU Public Peripherals

Figure 7-1 shows the OMAP5910 device with the MPU public peripherals highlighted.

Figure 7-1. MPU Public Peripherals Area



7.2 Camera Interface

An 8-bit camera interface (32-bit internal bus on the TIPB side) connects a camera module to the MPU peripheral bus of the OMAP5910 device. The interface handles multiple image formats synchronized on vertical and horizontal synchronization signals. Data transfer between camera and interface can be done synchronously or asynchronously. The data is stored in a buffer to be sent over the peripheral bus using DMA mode or 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.

7.2.1 Functional Architecture

The architecture consists of four functional blocks:

- Buffer:

  A buffer is used to store the data word received from the camera module and transfer 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 latched and mixed to be compliant with the 32-bit data format of the MPU TIPB. A 128-bit-deep FIFO is implemented to provide local buffering of 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 CPU of the data transfer
  - To reduce real time constraints of DMA read (FIFO's buffering part)
  - To group x DMA accesses in only one time slot (FIFO's block part)

  It is, however, possible to forward a direct transfer to the CPU in bypass mode by disabling the DMA request line.

- Clock divider:

  This function is mainly used to manage clock division and to handle external clock generation for synchronous/asynchronous mode gating.

- Interrupt generator:

  An interrupt is generated to indicate start and end of frame, start and end of image, and FIFO overflow.

- TIPB registers:

  Status, control, and data 32-bit registers connect via the TIPB.

Figure 7-2. Camera Interface Block Diagram

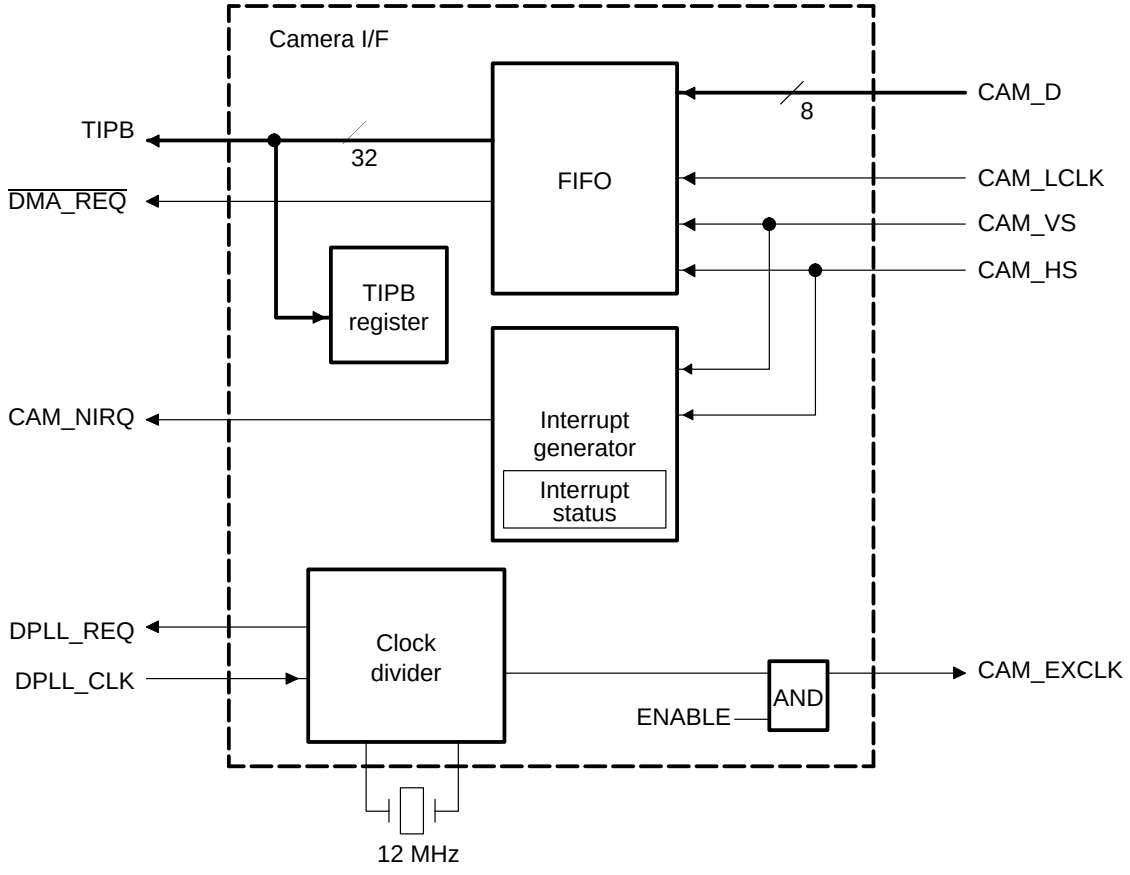
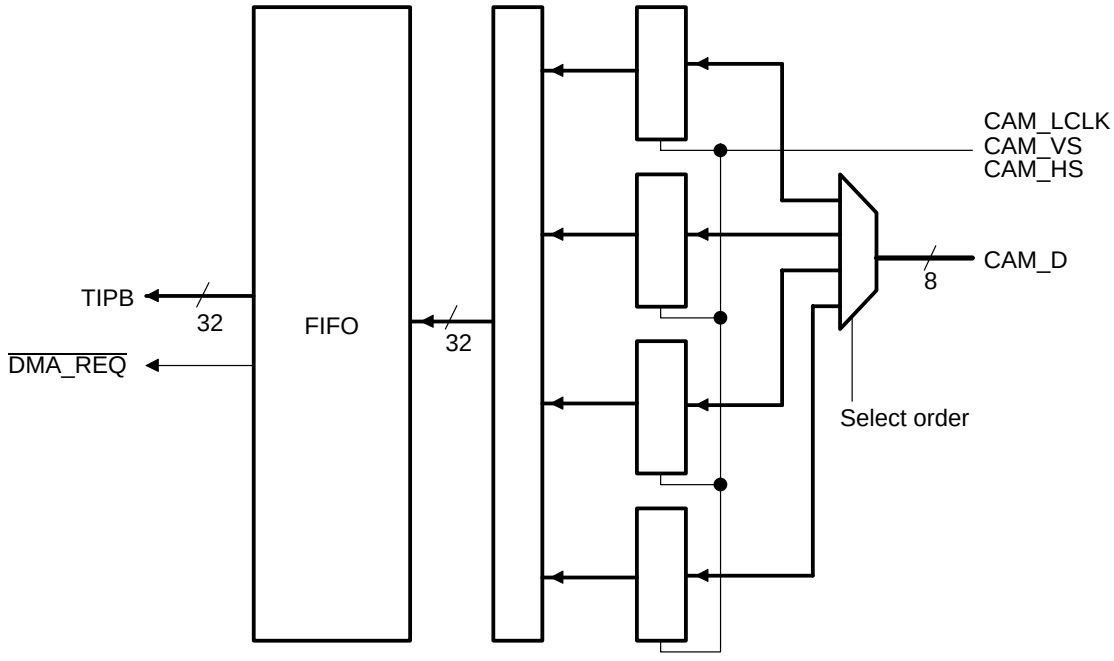


Figure 7-3. Image Data Transfer



7.2.1.1 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 can select the polarity of CAM.LCLK in the clock control register.

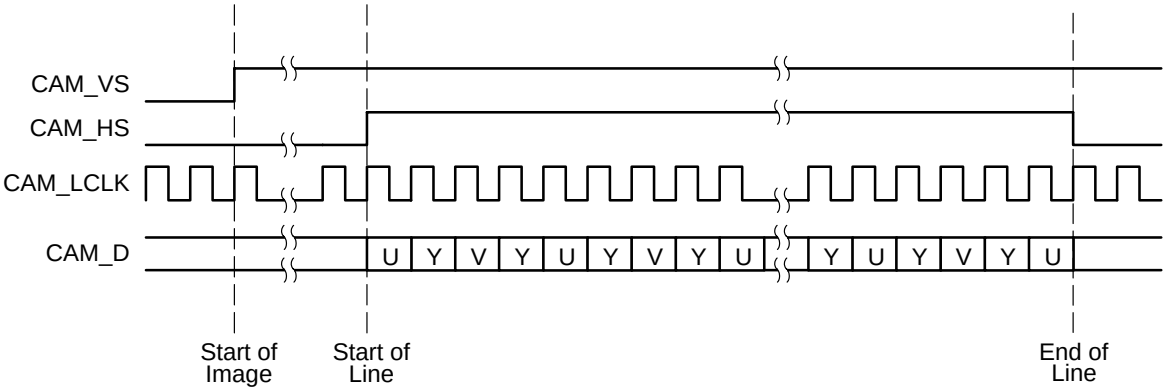
Program the camera interface so that data is always captured opposite the launch edge. For example, if data is latched by the sensor on the rising edge of CAM.LCLK, configure the interface to catch the data on the falling edge of CAM.LCLK.

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 on edge detection of vertical and horizontal synchronous signals.

It is possible to gate the clock during the VSYNC and/or HSYNC blanking periods. However it is recommended to let the clock run, because there is a process based on LCLK that clears all internal resynchronization registers while VSYNC or HSYNC is low before starting a new line or a new image. This mechanism prevents the FIFO from remaining word, which could corrupt the data of a new line.

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.

Figure 7-4. Timing Chart of Image Data Transfer (POLCLK = 1)



7.2.1.2 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 7.2.1.3) while the interface is latching data, the capture is automatically disabled and the autostart function is enabled.

7.2.1.3 Reset FIFO

An active-high reset FIFO is implemented at bit 18, RAZ FIFO, 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.

Before the FIFO is reset via the RAZ_FIFO bit, CAM.LCLK needs to be disabled by setting CTRLCLOCK[7] = 0. Then RAZ_FIFO may be set (MODE[7] = 1) to reset the FIFO. Then RAZ_FIFO must be set back to inactive (MODE[7] = 0) before the camera interface is functional.

You should use the RAZ_FIFO bit to clear any remaining data in FIFO before starting a new transfer. This bit also resets all status and control signals related to the FIFO, and it disables interrupt generation from the camera interface, so the RAZ_FIFO bit must be inactive before any camera interface transfers are started.

7.2.1.4 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 7-5. Order of Camera Data on TIPB (Not Swapped)

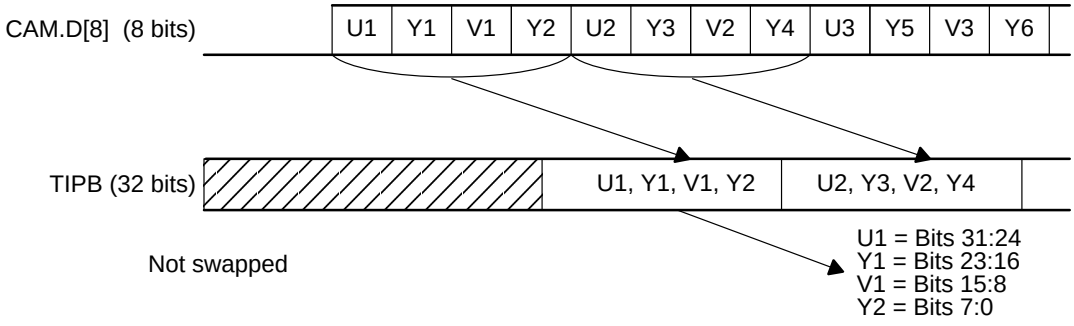
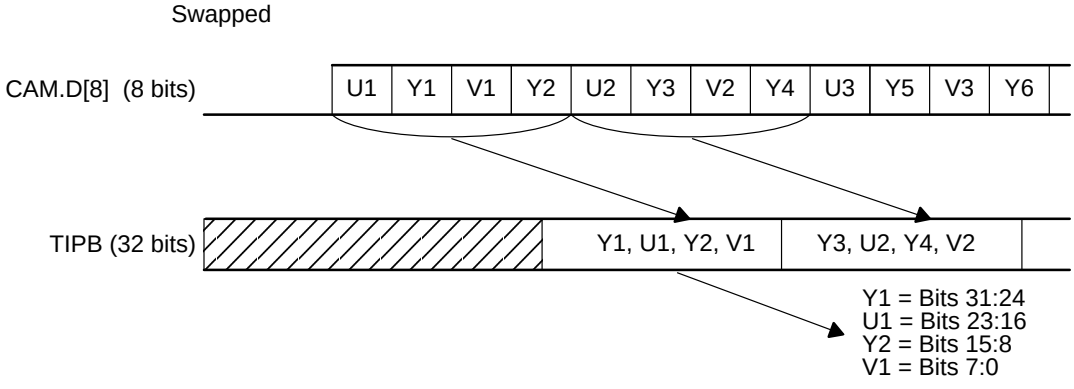


Figure 7-6. Order of Camera Data on TIPB (Swapped)



7.2.1.5 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, you can program the threshold 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, thus preventing DMA requests from being issued after every read.

A pulse on the DMA request (see Figure 7-7 and Figure 7-8) 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 yet responded to. 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 7-7. DMA Request

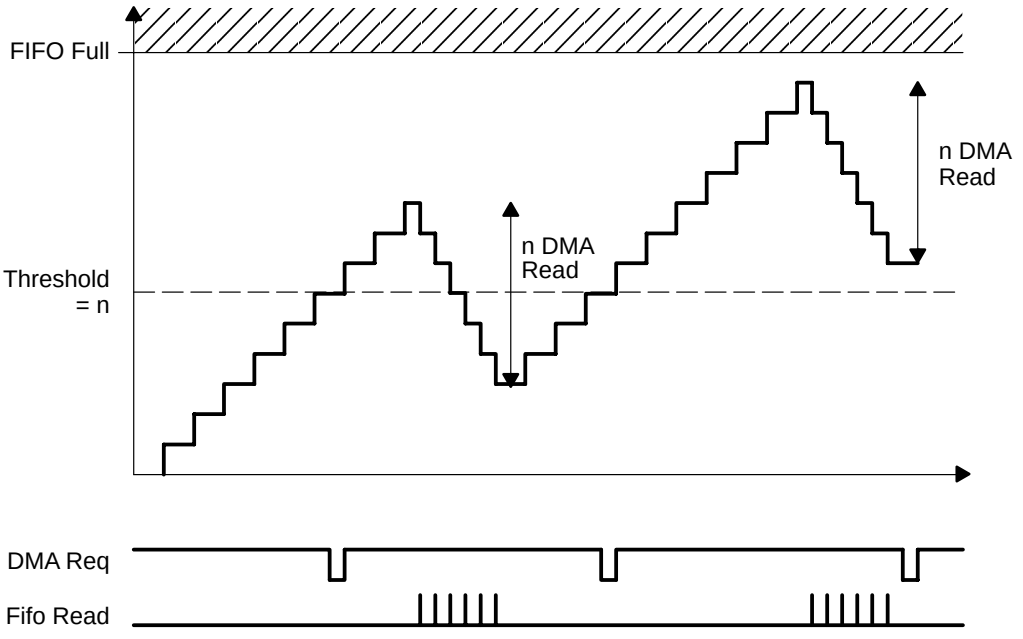
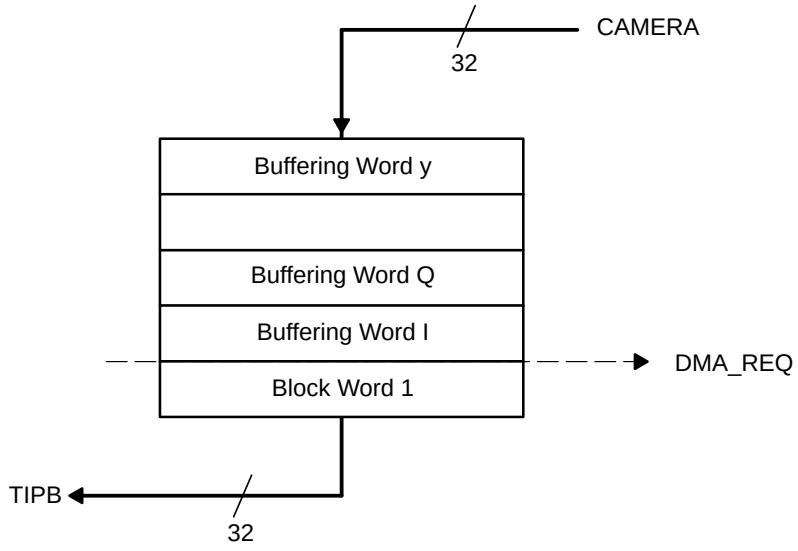


Figure 7-8. FIFO Buffer Parts



When the threshold value is set to 0, the interrupt is generated immediately. This is the equivalent of the threshold always being exceeded regardless of whether any data is present in the FIFO.

7.2.1.6 Clock Divider

The clock divider takes the internal 12-MHz clock source or the 48-MHz source from DPLL to generate the external clock CAM.EXCLK. The division factor is programmable in the clock control register through FOSCMOD (see Table 7-1).

Table 7-1. Clock Ratios

Ratio	CAM.EXCLK	
	From 12 MHz	From 48 MHz
1	12 MHz	-
1/2	6 MHz	24 MHz
1/5	-	9.6 MHz
1/6	-	8 MHz

A request is automatically generated to wake up the DPLL when 48 MHz is needed. The switch is performed when the 48-MHz signal is stable.

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.

7.2.1.7 Interrupt Generator

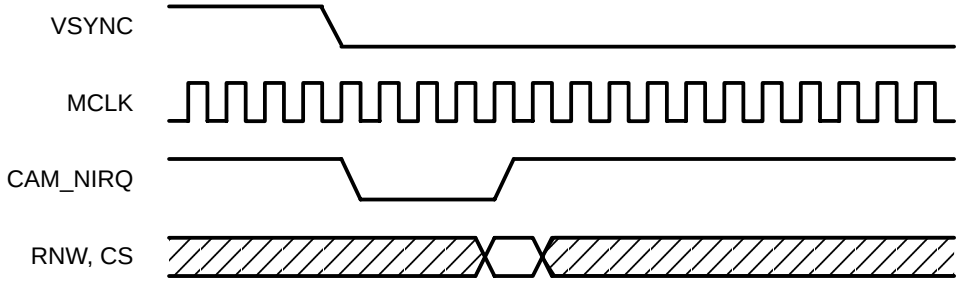
The interrupt generator handles six cases of interrupt:

- ❏ Data transfer interrupt. One IRQ is generated per word received.
- ❏ HSYNC rising edge (start of frame)
- ❏ HSYNC falling edge (end of frame)
- ❏ 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 7-9. IRQ Generated on VSYNC Falling Edge



7.2.1.8 DMA Procedure

A typical procedure to perform the data transfer by DMA is as follows:

1) Rising edge of VSYNC sends an interrupt to TI925T to alert the system DMA that a start of image has occurred. The system DMA is programmed to move one complete image of data then give an interrupt when complete.

2) High level of HSYNC and proper clock edge start the first data transfer from the camera to the OMAP5910 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 OMAP5910 device camera interface, and the system DMA continue 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 TI925T to signal that the end of frame occurred.

There are many ways that the camera interface and system DMA can be configured to move the data, but in the above sequence the interrupt load on the TI925T is minimal.

7.2.1.9 TIPB Registers

The camera interface contains seven registers for communication between the TIPB and camera module. They mainly control clock generation, interrupt request, and status register (see Section 7.2.1.10).

The address of each register is the start address (FFFB:6800) plus the offset indicated in Table 7-3.

Table 7-2 shows the default configuration of several critical register fields at reset. See Table 7-4 through Table 7-10 for full descriptions of these register fields.

Table 7-2. 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
DPLL_EN	DPLL clock source disabled
THRESHOLD	Trigger level = 1 word

7.2.1.10 Camera Interface Registers (FFFB:6800)

Because the TIPB register read accesses are resynchronized to the camera interface clock, the MCLK_EN bit must first be set before any camera interface register reads are performed. Table 7-3 lists the camera interface registers. Table 7-4 through Table 7-10 describe the individual registers.

Table 7-3. Camera Interface Registers

Register	Description	R/W	Size	Offset Address
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

The MCLK_EN bit gates the 12-MHz master clock of the camera interface to disable the clock when switching between two clock domains or to save power consumption when the camera module is not used. To clear PEAK_COUNTER, read all data in FIFO then write PEAK_COUNTER with 0.

Table 7-4. Clock Control Register (CTRLCLOCK)

Bit	Name	Value	Function	R/W	Reset Value
31-8	RESERVED		This field is reserved (unknown value after reset).	R/W	0xX
7	LCLK_EN	0	Disables	R/W	0x0
		1	Enables incoming CAM.LCLK		
6	DPLL_EN	0	Disables	R/W	0x0
		1	Enables DPLL source (48 MHz)		
5	MCLK_EN	0	Disables	R/W	0x0
		1	Enables internal clock of interface		

Table 7-4. Clock Control Register (CTRLCLOCK) (Continued)

Bit	Name	Value	Function	R/W	Reset Value
4	CAMEXCLK_EN	0	Disables	R/W	0x0
		1	Enables CAM.EXCLK		
3	POLCLK		Sets polarity of CAM.LCLK	R/W	0x0
		0	Data latched on rising edge		
		1	Data latched on falling edge		
2-0	FOSCMOD		Sets the frequency of the CAM.EXCLK clock	R/W	0x00
		000	12 MHz		
		010	6 MHz		
		100	9.6 MHz (48 MHz/5)		
		101	24 MHz (48 MHz/2)		
		110	8 MHz (48 MHz/6)		

Table 7-5. Interrupt Source Status Register (IT_STATUS)

Bit	Name	Function	R/W	Reset Value
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 7-6. Camera Interface Mode Configuration Register (MODE)

Bit	Name	Value	Function	R/W	Reset Value
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	R/W	0x0
		1	Enables interrupt on FIFO_FULL		
16	EN_NIRQ	0	Disables	R/W	0x0
		1	Enables data transfer interrupt (bypass DMA mode)		
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).	R/W	0x0
		0	Not swapped		
		1	Swapped		

Table 7-6. Camera Interface Mode Configuration Register (MODE) (Continued)

Bit	Name	Value	Function	R/W	Reset Value
2-1	IMGSIZE		Sets image size	R/W	0x00
		00	CIF		
		01	QCIF		
		10	VGA		
		11	QVGA		
			Currently, these bits have no effect on the operation of the camera interface.		
0	CAMOSC	0	Set synchronous mode	R/W	0x0
		1	Set asynchronous mode		
			Currently this has no effect on the camera interface.		

Table 7-7. Status Register (STATUS)

Bit	Name	Function	R/W	Reset Value
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 7-8. Camera Interface GPIO Register (GPIO)

Bit	Name	Function	R/W	Reset Value
31-1	RESERVED	Reserved bits	R/W	0xX
0	CAM_RST	Reset for camera module	R/W	0x0

Table 7-9. Image Data Register (CAMDATA)

Bit	Name	Function	R/W	Reset Value
31-0	CAMDATA	Image data from FIFO	R	0x0

Table 7-10. FIFO Peak Counter Register (PEAK_COUNTER)

Bit	Name	Function	R/W	Reset Value
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

7.2.2 Clock Switching Procedures

7.2.2.1 CAM.EXCLK Switch Protocol

The CAM.EXCLK switch protocol is required for any change of the CAM.EXCLK frequency value to first disable both 12-MHz clock source and the DPLL clock source in clock control registers:

1) Disable MCLK and DPLL_CLK (MCLK_EN = 0, DPLL_EN = 0, FOSCMOD = FOSCMOD).

2) Change CAM.EXCLK value (FOSCMOD = new FOSCMOD).

3) Enable MCLK and DPLL_CLK (MCLK_EN = 1, DPLL_EN = 1, FOSCMOD = FOSCMOD).

7.2.2.2 CAM.LCLK Switch Protocol

Bit 3 of the clock control register (POLCLK) sets the polarity of CAM.LCLK. You must disable CAM.LCLK before selecting the rising or the 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).

7.3 MPU I/O

The MPU I/O module enables direct I/O communications between the MPU (through the public TIPB) and external devices (see Figure 7-10).

Two types of I/Os can be used:

- Specific I/Os dedicated for 8x8 keyboard connection:
  - Eight inputs (KB R[7:0]) for row lines
  - Eight outputs (KB.C[7:0]) for column lines
- General-purpose I/Os:
  - Five MPU I/O signals (5, 4, 3, 2, and 1) are available in the default OMAP5910 multiplexing.
  - Five additional MPU I/O signals (12, 11, 7, 6, and 0) can be used by configuring the OMAP5910 multiplexing. For more detail, see Section 6.8 and Section A.2, *I/O Functional Multiplexing*.

7.3.1 MPU I/O Interrupts

The MPU I/O module generates two interrupts:

- The keyboard interrupt (KEYBOARD_INT), used to detect a key press, connected to the MPU interrupt handler level2, line1 (edge-sensitive)
- The GPIO interrupt (GPIOS_INT), used to detect an edge on one MPUIO input, connected to the MPU interrupt handler level2, line5 (level-sensitive).

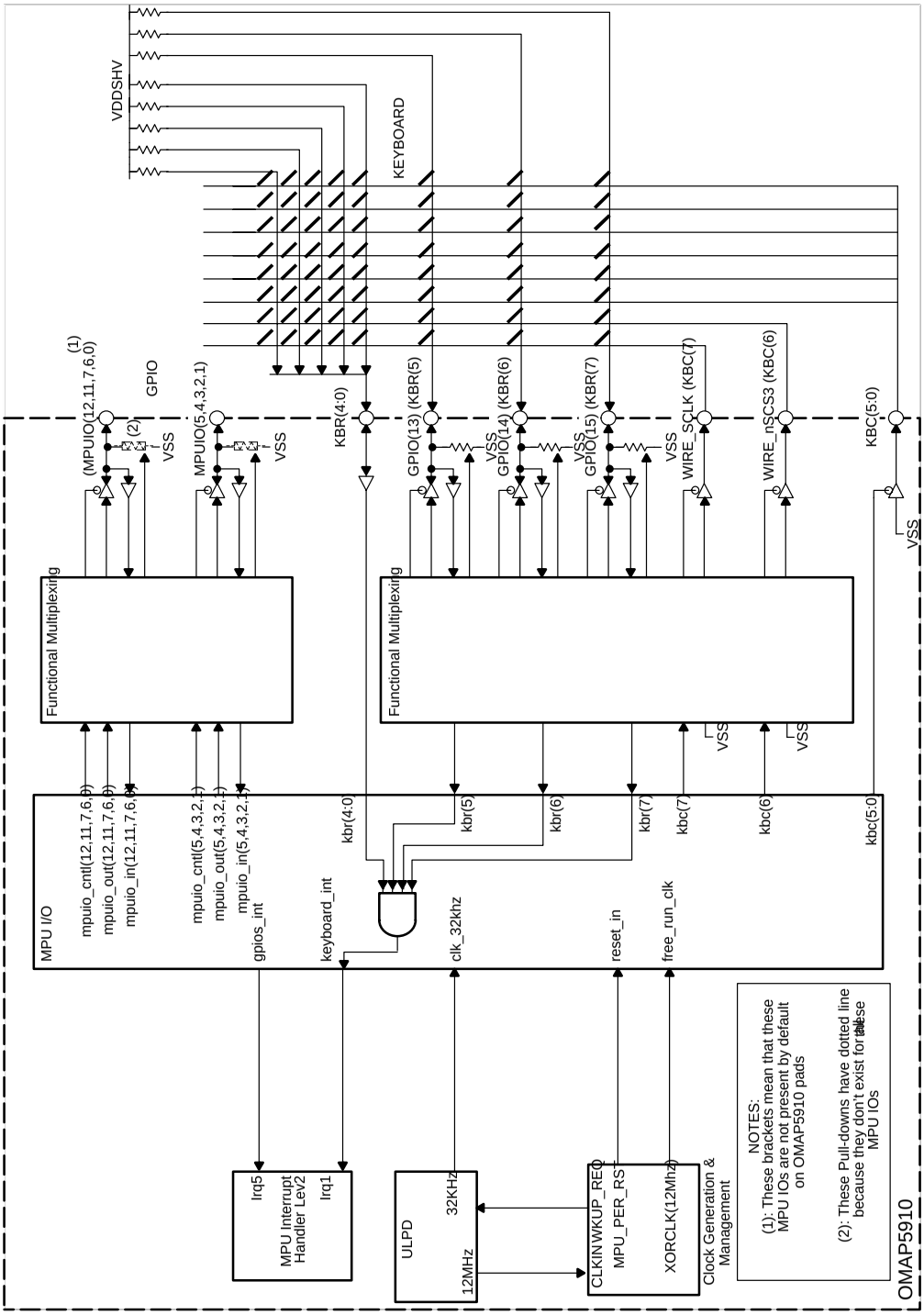
7.3.2 MPU I/O Clocks and Reset

The MPU I/O module has two clocks:

- The 32-kHz system clock (CLK_32KHZ), which comes, through the ULPD, from either the OMAP5910 32-kHz oscillator or the OMAP5910 CLK32K_IN CMOS input. For more detail, see Chapter 15, *Clock Generation and System Reset Management*).
- The 12-MHz clock (FREE_RUN_CLK), used to resynchronize the GPIO_INT register read. Comes from the MPU peripheral fixed clock (XORCLK). This clock is free running when OMAP5910 is awake.

The MPU TIPB reset (MPU_PER_RESET) resets the MPU I/O module.

Figure 7-10. MPU I/O Environment



7.3.3 MPUIO Keyboard Interface

To allow button press detection:

- ❏ All the row lines (KB.R) must have an external pullup.
- ❏ All the column lines (KB.C) drive a low level (idle state of Table 7-11).

  The output drivers of the KBC output pins act as open-drain outputs in that they only drive low or are 3-state. So external pullup resistors are required to achieve a high state when these outputs are 3-stated.

The keyboard interrupt (keyboard_int) to the MPU is an AND of the eight row lines filtering during one 32-kHz clock period (CLK_32KHZ).

As soon as any key of the keyboard matrix is pressed, the corresponding row and column lines are shorted together and a low level is driven on the corresponding row line, generating a keyboard interrupt (see Figure 7-11).

Once the keyboard interrupt is received, the MPU scans the column lines in the sequence described in the Table 7-11 in order to detect the key that has been pressed.

Table 7-11. Keyboard Scanning Sequence

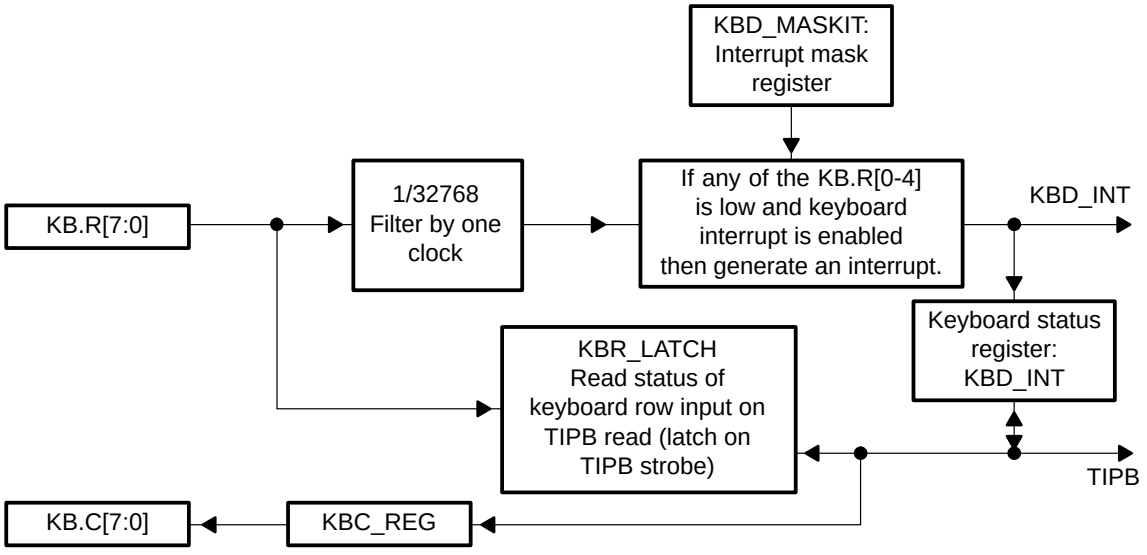
	Idle	Keyboard Scanning								Idle
KB.C[0]	0	1	0	1	1	1	1	1	1	0
KB.C[2]	0	1	1	1	0	1	1	1	1	0
KB.C[3]	0	1	1	1	1	0	1	1	1	0
KB.C[4]	0	1	1	1	1	1	0	1	1	0
KB.C[5]	0	1	1	1	1	1	1	0	1	0
KB.C[6]	0	1	1	1	1	1	1	1	0	0
KB.C[7]	0	1	1	1	1	1	1	1	1	0
KB.C[1]	0	1	1	0	1	1	1	1	1	0

For each step of the sequence, the MPU:

- ❏ Writes the specific value (with a low level on one column line) in the KBC_REG register
- ❏ Reads the value of the KBR_LATCH register and thus detects if one key of the concerned column line (this one which drives a low level) is pressed

At the end of the scanning sequence, the MPU is able to establish which keys have been pressed.

Figure 7-11. Keyboard Process Block Diagram

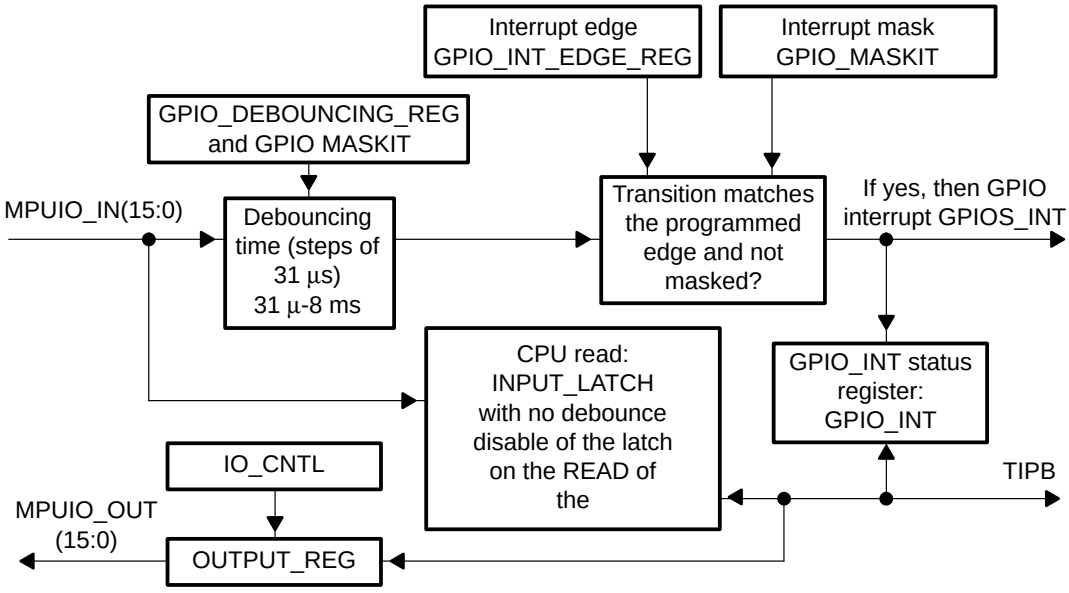


7.3.4 MPUIO General-Purpose I/O Interface

This interface has the following characteristics (see Figure 7-12):

- ❏ Every MPUIO can be configured individually either in input or in output mode.
- ❏ Interrupt generation (GPIOS_INT) on edge detection (rising or falling) after debouncing preprocessing.
- ❏ Edge detection can be used to latch all the GPIOs (event capture mode).
- ❏ GPIO interface works with the 32-kHz-system clock and consequently can be used to wake up the OMAP5910 device by generating the GPIO interrupt.

Figure 7-12. GPIO Process



7.3.5 GPIO Interrupt Reset

The GPIO interrupt (gpios_int) is generated when one event occurs on one MPU I/O input (see Figure 7-13).

The edge detection and the interrupt generation are done synchronously with the 32-kHz system clock (clk_32khz).

These events (and consequently the GPIO_INT interrupt) are reset on one GPIO interrupt register (GPIO_INT) read.

Only the bits that are active after masking are reset.

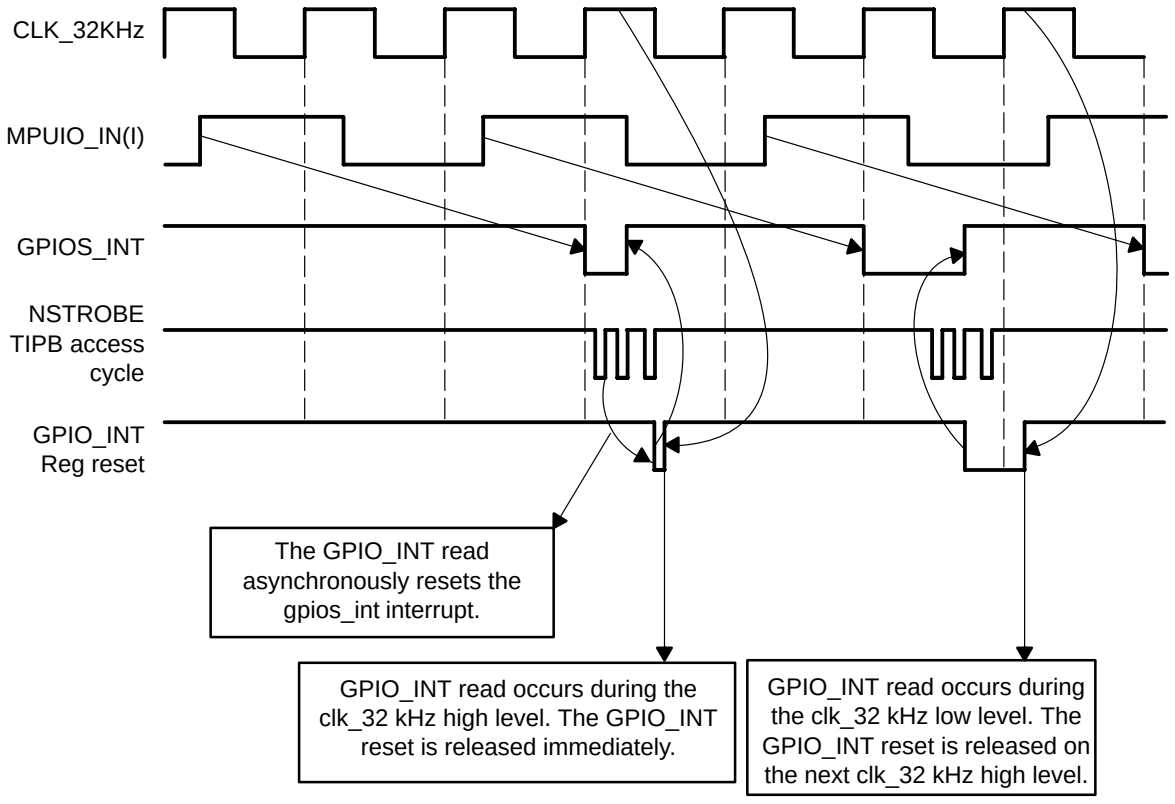
The GPIO_INT reset is synchronously asserted and synchronously released with the 32-kHz system clock. The GPIO_INT register read and the 32-kHz-system clock are resynchronized with the MPU TIPB fixed peripheral clock (12-MHz clock) free_run_clock.

When the GPIO_INT read occurs:

- ❏ During a high level of the system clock, the release of the reset is done immediately.
- ❏ During a low level of the system clock, the reset is done on the next high level of the system clock.

Even the worst case (reset release on the next 32-kHz cycle) supports the maximum speed of the MPU I/O module (one edge can be detected every two 32-kHz cycles with a debouncing 0).

Figure 7–13. GPIO_INT Register Read Timing



7.3.6 GPIO Interrupt Masking

The GPIO interrupt mask register (GPIO_MASKIT) can mask the edge detection on the MPU I/O inputs.

This mask is applied asynchronously on each detected edge after debouncing. If all the edges detected are masked, then the gpios_int interrupt is masked.

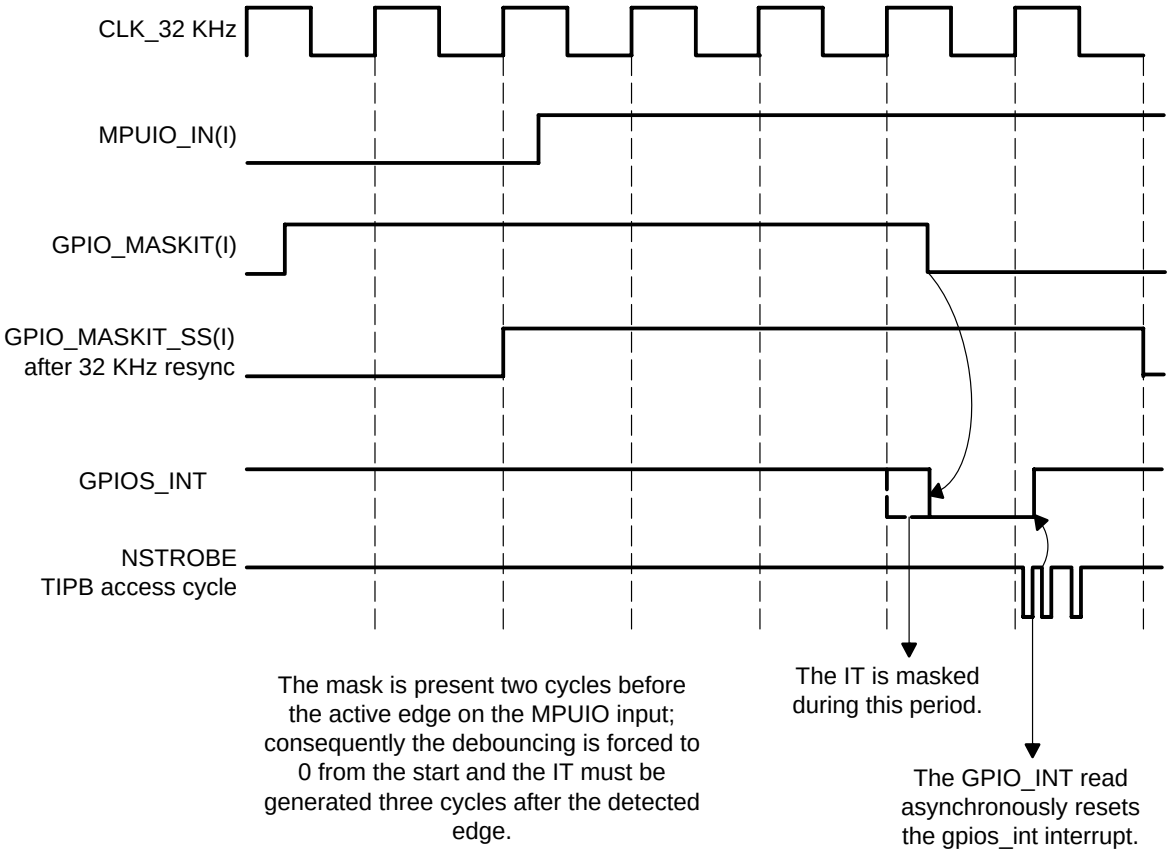
Masking one MPU I/O input forces its corresponding debouncing value to 0, which ensures that gpios_int is generated three cycles after the corresponding detected edge.

To ensure that this force is active from the start of the edge detection, the mask must be present two cycles before the detected edge. In this case, the interrupt is generated three cycles after the detected edge (see the corresponding timing in Figure 7–14). Otherwise, the mask can be activated during the

debouncing period; the debouncing is then forced dynamically to 0 and the interrupt is generated five cycles after the mask presence. You must decide whether or not to mask these interrupts by maintaining or releasing the mask activation.

When one detected edge is masked, the event is not reset when a GPIO_INT register read occurs. Thus, when the mask becomes inactive, the corresponding detected edge generates the gpios_int interrupt, and this interrupt is reset on the next GPIO interrupt register (GPIO_INT) read, as shown in Figure 7–14.

Figure 7–14. MPU I/O Input Masking Timing

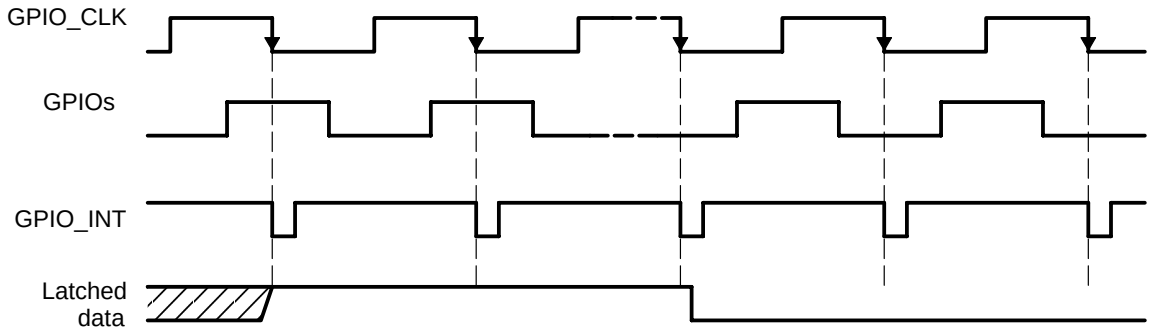


When a GPIO interrupt is masked, the GPIO_INT register does not indicate any active interrupt status if an edge occurs on the masked interrupt. However, if an edge occurs while the GPIO interrupt is masked, the active interrupt status is stored and an interrupt is sent to the interrupt handler as soon as that GPIO interrupt is unmasked (enabled).

7.3.7 Event Capture Module

The GPIO event capture mode allows latching the input value present on the GPIO ports each time a rising or a falling edge occurs on a selected GPIO port, here called GPIO_CLK. If not masked, the GPIO_CLK-selected edge generates an interrupt to the processor, as shown in Figure 7–15.

Figure 7–15. GPIO_CLK Timing



GPIO_CLK can be generated from an external physical module. Consequently, it may be necessary to insert a debouncing delay on this signal. The debouncing time is programmable in the GPIO debouncing register (GPIO_DEBOUNCING_REG) in steps of 31 μs.

The GPIO event mode register (GPIO_EVENT_MODE_REG) enables or disables the GPIO event mode. It also selects the external pin used as the GPIO_CLK. The GPIO interrupt edge register (GPIO_INT_EDGE) selects the GPIO_CLK falling or rising edge to generate the GPIO_INT interrupt.

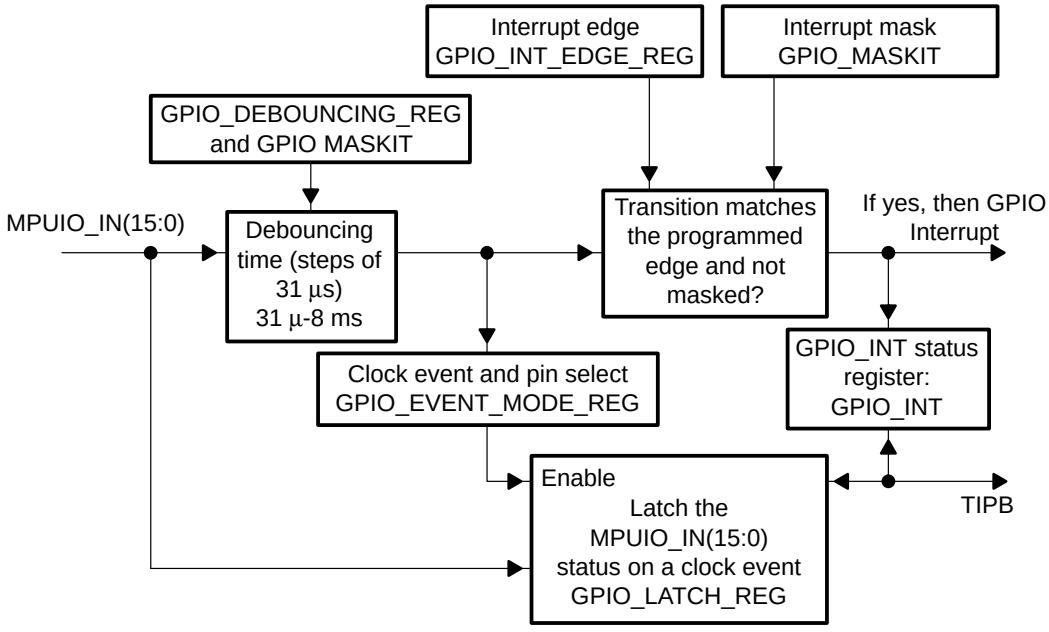
When the gpios_int interrupt (active low) is generated, the GPIO_INT register must be read by the MPU to define any active GPIO signal interrupts.

The GPIO interrupt mask register (GPIO_MASKIT) masks GPIO interrupts individually.

On the GPIO_CLK programmed edge, after the debouncing delay, the internal MPUIO_IN bus is latched in the GPIO latch register (GPIO_LATCH_REG). Its value can be read after the detection of the interrupt, even if the external value has changed.

The event capture process is shown in Figure 7–16.

Figure 7–16. Event Capture Process



7.3.8 MPU I/O Registers

Start address in the MPU I/O range (hex): FFFB:5000

Table 7–12 lists the MPU I/O registers. Table 7–13 through Table 7–25 describe the individual registers.

Table 7–12. MPU Input/Output Registers

Register	Description	R/W	Size	Address	Offset
INPUT_LATCH	General-purpose input	R	16 bits	FFFB:5000	0x00
OUTPUT_REG	Output	R/W	16 bits	FFFB:5000	0x04
IO_CNTL	Input/Output control	R/W	16 bits	FFFB:5000	0x08
KBR_LATCH	Keyboard row inputs	R	16 bits	FFFB:5000	0x10
KBC_REG	Keyboard column outputs	R/W	16 bits	FFFB:5000	0x14
GPIO_EVENT_MODE_REG	GPIO event mode	R/W	16 bits	FFFB:5000	0x18
GPIO_INT_EDGE_REG	GPIO interrupt edge	R/W	16 bits	FFFB:5000	0x1C
KBD_INT	Keyboard interrupt	R	16 bits	FFFB:5000	0x20

Table 7–12. MPU Input/Output Registers (Continued)

Register	Description	R/W	Size	Address	Offset
GPIO_INT	GPIO interrupt	R	16 bits	FFFB:5000	0x24
KBD_MASKIT	Keyboard mask interrupt	R/W	16 bits	FFFB:5000	0x28
GPIO_MASKIT	GPIO mask interrupt	R/W	16 bits	FFFB:5000	0x2C
GPIO_DEBOUNCING_REG	GPIO debouncing	R/W	16 bits	FFFB:5000	0x30
GPIO_LATCH_REG	GPIO latch	R	16 bits	FFFB:5000	0x34

Table 7–13. General-Purpose Input Register (INPUT_LATCH)

Bit	Name	Function	Reset Value
15–0	INPUT_LATCH	General-purpose inputs	Reflects input pins

Table 7–14. Output Register (OUTPUT_REG)

Bit	Name	Function	Reset Value
15–0	OUTPUT_REG	General-purpose outputs	Undefined

Table 7–15. Input/Output Control Register (IO_CNTL)

Bit	Name	Value	Function	Reset Value
15–0	IO_CNTL		In/out control for general-purpose I/O	All bits at 1
		0	I/O is configured as output	
		1	I/O is configured as input	

Table 7–16. Keyboard Row Inputs Register (KBR_LATCH)

Bit	Name	Function	Reset Value
15–7	Reserved		
4–0	KBR_LATCH	Keyboard row inputs	Reflects input pins

Table 7–17. Keyboard Column Outputs Register (KBC_REG)

Bit	Name	Function	Reset Value
15–8	Reserved		
7–0	KBC_REG	Keyboard columns outputs	0

Table 7–18. GPIO Event Mode Register (GPIO_EVENT_MODE_REG)

Bit	Name	Value	Function	Reset Value
15–5	Reserved			
4–1	PIN_SELECT		Select MPUI/O_IN[15:0] pin to be the GPIO_CLK event	0000
		0000	Pin 0	
		1111	Pin 15	
0	SET_GPIO_EVENT_MODE	0	GPIO event mode disable	0
		1	GPIO event mode enable	

Table 7–19. GPIO Interrupt Edge Register (GPIO_INT_EDGE_REG)

Bit	Name	Value	Function	Reset Value
15–0	EDGE_SELECT[15:0]		Set interrupt on falling/rising edge	0
		0	Falling edge	
		1	Rising edge	

Table 7–20. Keyboard Interrupt Register (KBD _INT)

Bit	Name	Function	Reset Value
15–1	Reserved		
0	KBD_INT	Keyboard interrupt (active low)	1

Note: KBD_INT is a status bit only (duplication of the level of the corresponding interrupt signal).

Table 7–21. GPIO Interrupt Register (GPIO_INT)

Bit	Name	Function	Reset Value
15–0	GPIO_INT	GPIO interrupts (active high)	0

Note: GPIO_INT is reset on read access to the GPIO_INT register. The value read is the value after mask application.

Even in emulation mode, the GPIO interrupts are reset by a read in the GPIO interrupt register (GPIO_INT).

Table 7–22. Keyboard Mask Interrupt Register (KBD_ MASKIT)

Bit	Name	Function	Reset Value
15–1	Reserved		
0	KBD_MASKIT	Mask is active at level 1, inactive at level 0	00

Table 7–23. GPIO Mask Interrupt Register (GPIO_MASKIT)

Bit	Name	Function	Reset Value
15–0	GPIO_MASKIT[15:0]	Mask is active at level 1, inactive at level 0	00

Table 7–24. GPIO Debouncing Register (GPIO_DEBOUNCING_REG)

Bit	Name	Function	Reset Value
15–9	Reserved		
8–0	GPIO_ DEBOUNCING_REG	000000000: 0 μσ to 31 μs debouncing time 100000010: 7,97 ms to 8 ms debouncing time Programming step is 31 μs.	0000

Note: Because GPIO_CLK is an asynchronous signal, loading GPIO_DEBOUNCING_REG with 01 hex minimum value is recommended to ensure that you have a 31-μs minimum debouncing time. If the value is 00 hex, the interrupt may be generated immediately when an edge is met.

Table 7–25. GPIO Latch Register (GPIO_LATCH_REG)

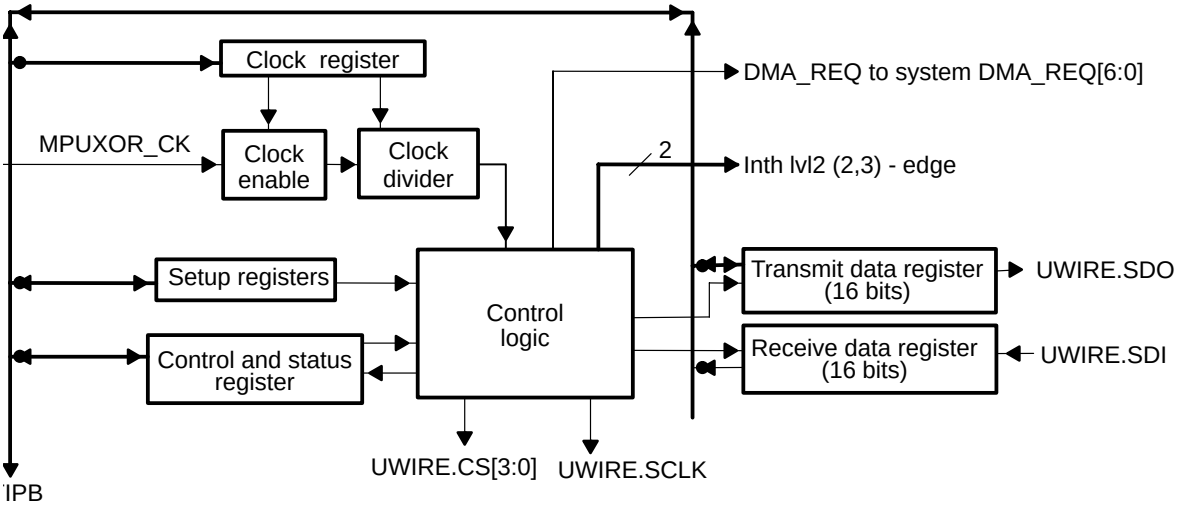
Bit	Name	Function	Reset Value
15–0	GPIO_LATCH_REG	After debouncing time, the ARMI/O_IN bus is latched in this.	00

7.4 MicroWire Interface

This serial synchronous interface can drive two serial external components. For the external devices, this interface is compatible with the µWire standard and is seen as the master (see Figure 7–17).

A transmit DMA mode is available.

Figure 7–17. Block Diagram



7.4.1 MicroWire Registers

Start address in the peripheral range (hex): FFFB:3000

Table 7–26 lists the MicroWire registers. Table 7–27 through Table 7–34 describe the individual registers.

Table 7–26. 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

Table 7–26. MicroWire Registers (Continued)

Register	Description	R/W	Size	Address	Offset
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 7–27. Transmit Data Register (TDR)

Bit	Name	Function	Reset Value
15–0	TD	Data to transmit	Undefined

Note: MSB (bit 15) is the first transmitted bit.

Whatever its size, the word must be aligned on the most significant bit (MSB) side.

Table 7–28. Receive Data Register (RDR) – Offset address (hex): 0x00

Bit	Name	Function	Reset Value
15–0	TD	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 7–29. Control and Status Register (CSR)

Bit	Name	Value	Function	Reset Value
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
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

Table 7–29. Control and Status Register (CSR) (Continued)

Bit	Name	Value	Function	Reset Value
11–10	INDEX		Index of the external device	Undefined
		00	CS0	
		01	Reserved	
		10	Reserved	
		11	CS3	
9–5	NB_BITS_WR		Number of bits to transmit	Undefined
4–0	NB_BITS_RD		Number of bits to receive	Undefined

This register sets up the serial interface for the first and the second external components.

Table 7–30. Setup Register 1 (SR1)

Bit	Name	Value	Function	Reset Value
11–6	Reserved			
5	CS0_CHK		Before activating a write process, checks if external device is ready.	Undefined
		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 that the first external component is ready and starts the write process.	
			Used when CS0 is selected.	

Table 7–30. Setup Register 1 (SR1) (Continued)

Bit	Name	Value	Function	Reset Value
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 to the microwire control logic as defined in register SR3).	Undefined
		00	F_INT/2	
		01	F_INT/4	
		10	F_INT/8	
		11	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)	Undefined
		0	Falling (serial clock not inverted)	
		0	Rising (when serial clock inverted)	
		1	Rising (serial clock not inverted)	
		1	Falling (when serial clock inverted)	
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)	Undefined
		0	Falling (serial clock not inverted)	
		0	Rising (when serial clock inverted)	
		1	Rising (serial clock not inverted)	
		1	Falling (when serial clock inverted)	

Note: Content of this register must not be changed when a read or write process is running.

This register sets up the serial interface for the first and the second external components.

Table 7–31. Setup Register 2 (SR2)

Bit	Name	Value	Function	Reset Value
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	Undefined
		00	F_INT/2	
		01	F_INT/4	
		10	F_INT/8	
		11	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
5–0	Reserved			

Notes: 1) Content of this register must not be changed when a read or write process is running.

This register sets up the serial interface for the internal clock.

Table 7–32. Setup Register 3 (SR3)

Bit	Name	Value	Function	Reset Value
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
		00	ARMOXR_CK/2	
		01	ARMOXR_CK/4	
		10	ARMOXR_CK/7	
		11	ARMOXR_CK/10	
0	CLK_EN	0	Switch off the clock if 0.	0
		1	Switch on the clock if 1.	

Note: Content of this register must not be changed when a read or write process is running.

This register sets up the serial clock polarity.

Table 7–33. Setup Register 4 (SR4) (Read/Write)

Bit	Name	Function	Reset Value
0	CLK_IN	Serial clock is not inverted if 0.	0
		Serial clock is inverted if 1.	

Note: Content of this register must not be changed when a read or write process is running.

Table 7–34. Setup Register 5 (SR5) (Read/Write)

Bit	Name	Value	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.	0
		0	CS_toggle transmit mode is disabled if 0.	
		1	CS_toggle transmit mode is enabled if 1.	
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, then starts the transmission. The CS-CMD and the START bits in the control and status register (CSR) are not updated during autotransmit.	0
		0	Autotransmit mode is disabled if 0.	
		1	Autotransmit mode is enabled if 1.	
1	IT_EN		In IT mode, an interrupt is generated each time a word has been transferred or a received. This interrupt is a negative edge-triggered interrupt. A status register (IST) allows the CPU to know which interrupt (receive or/and transmit) occurred.	0
		0	IT mode is disabled if 0.	
		1	IT mode is enabled if 1.	
0	DMA_TX_EN	0	DMA transmit mode is disabled if 0.	0
		1	DMA transmit mode is enabled if 1.	

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.

Notes:

You cannot use another CS in normal or DMA mode when a DMA mode is active on one specific CS.

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.

7.4.2 Protocol Description

The serial port must be configured in order to use the setup registers.

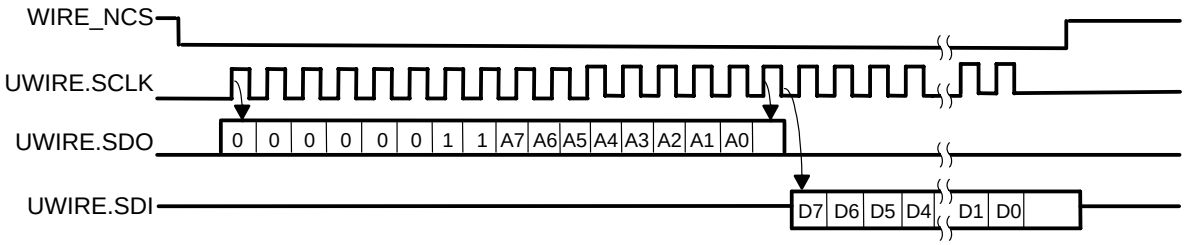
This interface can only drive one device at a given time. Therefore, the chip-select of the selected device must be set to its active level before starting any read or write process.

After the loading of the transmit data register (TDR), a 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 CSx_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 CSx_EDGE_RD bits of the setup registers). When CSx_EDGE_WR and CSx_EDGE_RD bits have the same value, it is assumed that the device behavior is the one shown in Figure 7–18. Otherwise, the required behavior of the external device is shown in Figure 7–19.

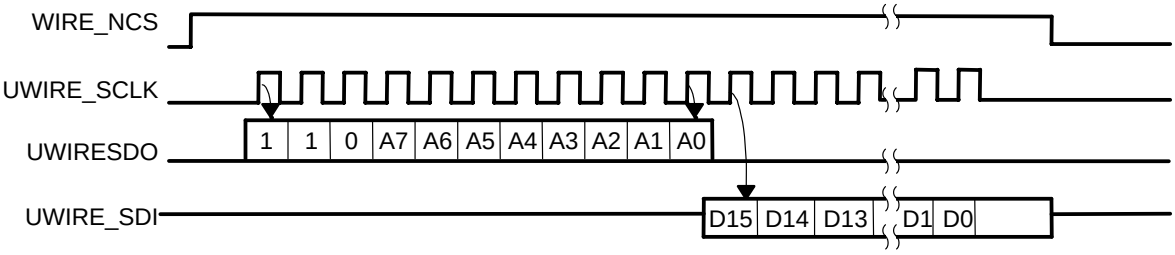
Figure 7–18. Behavior of a X25C02 EEPROM Read Cycle



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 7–19. Behavior of a 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.

7.4.3 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.

7.4.3.1 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 8. Else go to 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 5.
9) Set the following fields of the control and status register (CSR):
   - INDEX: 00
   - CS_CMD: 0
   - START: 0

7.4.3.2 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: *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

7.4.4 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

In this example, a loading sequence to drive a four-digit display is described.

7.4.4.1 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

7.4.5 Example of Protocol Using Autotransmit Mode

The autotransmit mode is controlled by the setup 5 register (SR5). 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 falling edge of the serial clock. Data in is sampled on rising edge.

3) SR3 = CLK_EN: 1

   CK_FREQ: 00 (must wait for 1 external clock + 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).

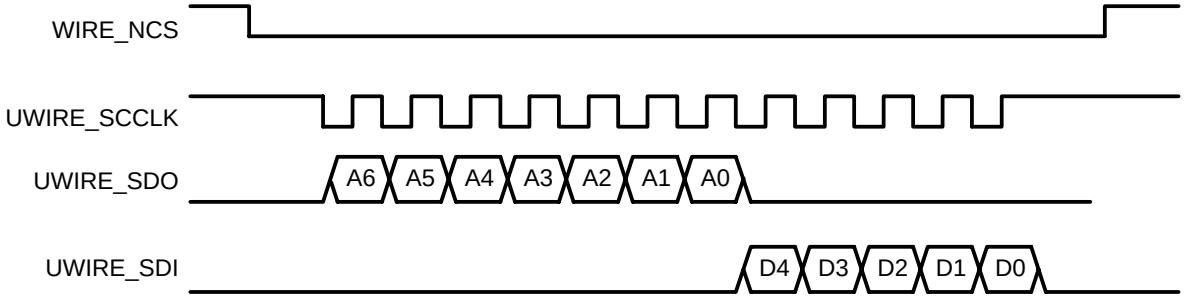
10) To continue reading data external component, go to 5 else go to 11.

11) Release auto transmit mode: SR5 = AUTO_TX_EN: 0.

12) END

The corresponding behavior of the serial interface is described in Figure 7–20.

Figure 7–20. Read Cycle in Autotransmit Mode



7.4.6 Example of Autotransmit Mode With DMA Support

The autotransmit mode and DMA mode are controlled by the setup 5 register (SR5). The following example configures µWire for a 16-bit write access on CS1 with serial clock out not inverted, CS auto toggle 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 (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 find 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 2 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.

7.5 32-kHz Timer

The MPU subsystem operating system (OS) requires interrupts at regular time intervals for OS scheduling purpose. OS time intervals can be from 1 ms to 30 ms. These time intervals can be generated using the three 32-bit OS/general-purpose TI925T timers, which use CLKIN or DPLL1; however, they can not be used when the system clock is not operating. Therefore, a 32-kHz clock-based timer is needed to provide the required OS timing interval. The clock period of 32 kHz is 30.60 µs. 32 kHz refers to 32678, not 32000.

7.5.1 Operating System Scalable Clock-Tick Interrupt Function

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

For example, Microsoft Windows CE OS scheduling requires the following:

- ❏ The periodic interrupt occurs every 1-25 ms.
- ❏ The timer is expected to run in all modes except when suspended.

32-kHz timer is a 24-bit down-counter that generates CPU interrupts for the TI925T processor. The following capabilities are available:

- ❏ Timer reset
- ❏ Timer current value reading
- ❏ Timer start and stop
- ❏ Interrupt generation as timer down-counts to zero
- ❏ Timer autorestart after it counts to zero
- ❏ On-the-fly register read and write
- ❏ Interrupt disabling the by writing a 1 to the interrupt bit in the control register

Timer	Corresponding Level 2 Interrupt
32-kHz timer	IRQ_22

The tick value register (TVR) contains the desired value for the timer to count down. The tick counter register (TCR) is loaded with this value, then starts to count down to zero and generates a negative edge sensitive interrupt (low-level pulse duration = 15 µs) to the interrupt handler. Once the interrupt is back to the high level, the counter is reloaded from its register and then starts to count down again.

7.5.1.1 Overriding Normal Counting

Normal operation can be overridden by using two bits in the timer control register (TCR):

- The timer reload bit (TRB) causes the counter to be reloaded on the next clk32-kHz cycle (whether or not the timer is counting).
- The timer start stop bit (TSS) causes the counter to be stopped on the next clk32-kHz cycle. When the timer is stopped, the content of the counter is frozen.

7.5.1.2 Loading/Autorestart of the Timer

Loading the counter in the timer can be done in two fashions:

- Write a 1 to the TRB bit in the timer control register (TCR).
- Wait until the counter reaches zero and is reloaded from its register if the autorestart bit (ARL) in the timer control register (TCR) is set to 1. If not, then the timer is stopped.

7.5.1.3 Timer Interrupt Period

The timer interrupt period is defined by the value loaded into the tick value register (TVR).

The timer interrupt rate is as follows:

IRQ rate =(TVR+1) / 32768

Table 7–35. Timer Interrupt Period

TVR Value	Interrupt Period
0x000000	30.5 µs
0x00028F	19.9 ms
0xFFFFFF (Value at reset)	512 s (8 min 32 s)

7.5.2 32-kHz Timer Registers

Base address for 32-kHz timer: FFFB:9000

Table 7–36 lists the 32-kHz timer registers. Table 7–38 through Table 7–40 describe the individual registers.

Table 7–36. 32-kHz Timer Registers

Name	Description	R/W	Size	Address	Offset
CR	Timer control	R/W	32 bits	FFFB:9000	0x08
TVR	Tick value	R/W	32 bits	FFFB:9000	0x00
TCR	Tick counter	R	32 bits	FFFB:9000	0x04

7.5.2.1 Synchronization Issues

Synchronization of reads and writes to the 32-kHz clock is done in different ways for each register. This leads to slight restrictions concerning register access (see Table 7–37).

Table 7–37. Read/Write Synchronization

Register Name	Read	Write
CR	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 µs). If this is not the case, the value written is not guaranteed
TCR	Reads are resynchronized on MPUXOR_CK clock to prevent peripheral bus from timing out. Can be read anytime, providing MPUXOR_CK is running. If not, the value is not guaranteed. Software must wait one 32-kHz period after turning on the MPUXOR_CK clock before the TCR register can be read.	Writing to this has no effect.
TVR	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 µs). If this is not the case the value written is not guaranteed

Table 7–38. Timer Control Register (CR)

Bit	Name	Value	Function	Reset Value
31–4	Reserved			
3	ARL		Autoreload/start	1
		0	One-shot mode. When the counter reaches zero, an interrupt is generated and the timer is stopped.	
		1	Sets the timer to autorestart mode	
2	IT_ENA		Interrupt enable	0
		0	Interrupt disabled	
		1	Interrupt enabled	
1	TRB		Timer reload bit	0
			TRB = 1 reloads the counter. Once the counter is reloaded, TRB is set to 0.	
0	TSS		Timer start/stop	0
		0	Stop timer	
		1	Start timer	
			If one-shot mode is selected (ARL = 0), this bit is automatically reset by internal logic when timer is equal to 0.	

Table 7–39. Tick Value Register (TVR)

Bit	Name	Function	Reset Value
31–24	Reserved		
23–0	TICK_VALUE_REG	This value is loaded when timer passes through 0 or when it starts.	0xFFFFFF

Table 7–40. Tick Counter Register (TCR)

Bit	Name	Function	Reset Value
31–24	Reserved		
23–0	TICK_COUNTER_REG	Value of timer	0xFFFFFF

7.6 Pseudonoise Pulse-Width Light Modulator

This pulse-width light (PWL) module provides control of 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.

7.6.1 PWL Functional Description

The PWL module is composed of a pseudorandom 8-bit data generator and a programmable threshold comparator (see Figure 7–21).

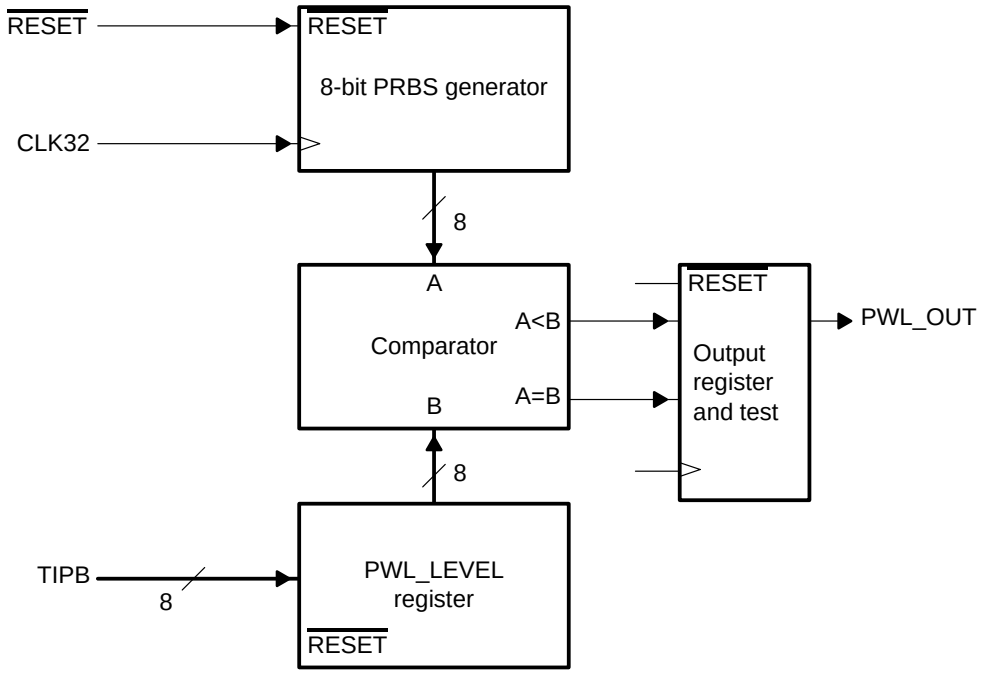
The pseudorandom 8-bit data generator is built using an LFSR. It generates a white normal-law random value between 1 and 255. The LFSR polynomial generator is P(x) = x[7] + x[3] + x[2] + x[1].

The comparator generates:

- 0 if the random value is greater or equal than the programmable threshold
- 1 if the random value is less than the programmable threshold

Assuming the random sequence is normal, it generates a sequence whose mean value is proportional to the comparator threshold.

Figure 7–21. PWL Block Diagram



7.6.2 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 there is no TIPB wait-state insertion.

Table 7–41 lists the PWL registers. Table 7–42 and Table 7–43 describe the individual registers.

Start address (hex): FFFB:5800

Table 7–41. PWL Registers

Name	Description	R/W	Size	Address	Offset
PWL_LEVEL	PWL-level	R/W	8 bits	FFFB:5800	0x00
PWL_CTRL	PWL control	R/W	8 bits	FFFB:5800	0x04

Table 7–42. PWL Level Register (PWL_LEVEL)

Offset address (hex): 0x00

Bit	Name	Function	Reset Value
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

Table 7–43. PWL Control Register (PWL_CTRL)

Offset address (hex): 0x04

Bit	Name	Function	Reset Value
7–1	–	Reserved	
0	CLK_ENABLE	Internal clock is enabled when 1.	0

7.7 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.

7.7.1 Overview

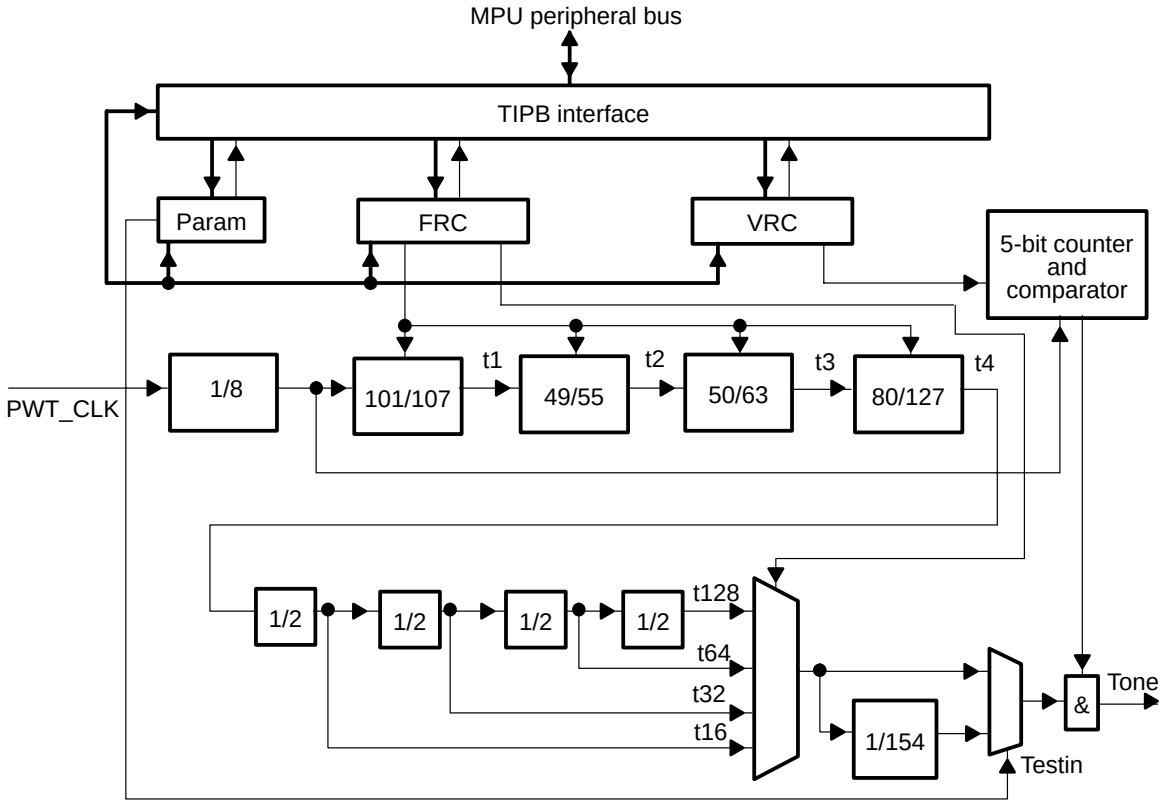
The PWT module creates the output tone signal for a buzzer. The frequency and the volume of this signal are programmable.

7.7.2 PWT Features

The PWT module has the following features (see Figure 7–22):

- ❏ 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 7–22. PWT Block Diagram



7.7.3 PWT Registers

Start address (hex): FFFB6000

Table 7–44 lists the PWT registers. Table 7–45 through Table 7–47 describe the individual registers.

Table 7–44. PWT Registers

Register	Description	R/W	Size	Address	Offset
FRC	PWT frequency control	R/W	8 bits	FFFB:6000	0x00
VRC	PWT volume control	R/W	8 bits	FFFB:6000	0x04
GCR	PWT general control	R/W	8 bits	FFFB:6000	0x08

Table 7–45. PWT Frequency Control Register (FRC) – Offset address (hex): 0x00

Bit	Name	Function	R/W	Reset Value
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

Table 7–46. PWT Volume Control Register (VRC) – Offset address (hex): 0x04

Bit	Name	Function	R/W	Reset Value
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

Table 7–47. PWT General Control Register (GCR) – Offset address (hex): 0x08

Bit	Name	Function	R/W	Reset Value
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

7.7.4 PWT Programming

7.7.4.1 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 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). Due to 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 resulting in short periods producing low jitter (see Table 7–48).

Table 7–48. Buzzer Frequencies

FRC Bits 5-2 1-0	Buzzer Frequency	FRC Bits 5-2 1-0	Buzzer Frequency
0000 00	4868 Hz	0000 10	1217 Hz
0001 00	4595 Hz	0001 10	1149 Hz
0010 00	4337 Hz	0010 10	1084 Hz
0011 00	4093 Hz	0011 10	1023 Hz
0100 00	3864 Hz	0100 10	966 Hz
0101 00	3647 Hz	0101 10	912 Hz
0110 00	3442 Hz	0110 10	860 Hz
0111 00	3249 Hz	0111 10	812 Hz
1000 00	3066 Hz	1000 10	767 Hz
1001 00	2894 Hz	1001 10	723 Hz
1010 00	2732 Hz	1010 10	683 Hz
1011 00	2579 Hz	1011 10	644 Hz
0000 01	2434 Hz	0000 11	608 Hz
0001 01	2297 Hz	0001 11	574 Hz
0010 01	2168 Hz	0010 11	541 Hz
0011 01	2046 Hz	0011 11	511 Hz
0100 01	1932 Hz	0100 11	483 Hz
0101 01	1824 Hz	0101 11	456 Hz

Table 7–48. Buzzer Frequencies (Continued)

FRC Bits 5-2 1-0	Buzzer Frequency	FRC Bits 5-2 1-0	Buzzer Frequency
0110 01	1721 Hz	0110 11	430 Hz
0111 01	1624 Hz	0111 11	406 Hz
1000 01	1533 Hz	1000 11	383 Hz
1001 01	1447 Hz	1001 11	361 Hz
1010 01	1366 Hz	1010 11	341 Hz
1011 01	1289 Hz	1011 11	322 Hz
		11XX XX	Not allowed

Note: The PWT was originally designed for a 13-MHz input clock, but the OMAP5910 device implements PWT with a 12-MHz clock. Consequently, frequencies shown are not exact tones.

7.7.4.2 Buzzer Volume

The buzzer volume can be programmed (see Table 7–49) with bits 6 to 1 in the volume control register VRC. The higher the 6 bit value is, the louder is 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 7–49. Buzzer Volume

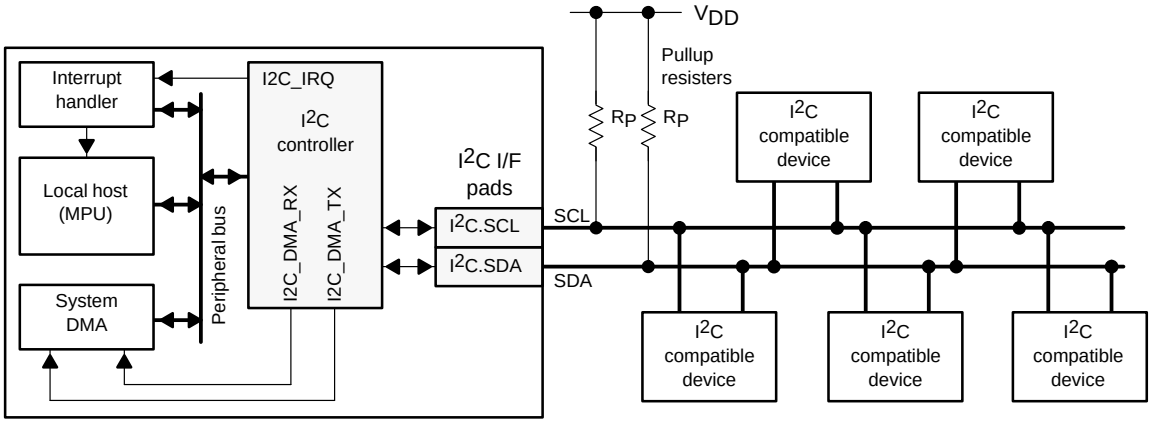
VRC Bits 6-1 ; 0	Buzzer/Loudspeaker
111111 1	Loud
000000 1	Quiet
xxxxx 0	Off

7.8 Inter-Integrated Circuit Controller

7.8.1 I2C Protocol Description

This section describes the I2C protocol. Figure 7–23 shows the I2C system overview. References to a local host in this section refer to the MPU processor.

Figure 7–23. I2C System Overview



7.8.1.1 Functional Overview

The I2C bus is a multimaster bus. The I2C controller function does support the multimaster mode, to which more than one device capable of controlling the bus can be connected. Including the OMAP5910, each I2C device is recognized by a unique address and can operate as either transmitter or receiver depending on 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 the device which 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.

7.8.1.2 I2C Controller Signals Pads

Data is communicated to devices interfacing the I2C via the serial data pin (SDA) and the serial clock pin (SCL). These two wires carry information between the OMAP5910 device and others connected to the I2C bus. Both 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 7–50 lists the signal pads. Table 7–51 lists the reset state of the I2C signals.

Table 7–50. Signal Pads

Name	Type	Description	Reset Value
I2C.SCL	In/ Out(OD)	I2C serial CLK line. Open-drain output buffer—requires external pullup resistor (Rp)	Input
I2C.SDA	In/ Out(OD)	I2C serial data line. Open-drain output buffer—requires external pullup resistor (Rp)	Input

Table 7–51. Reset State of I2C Signals

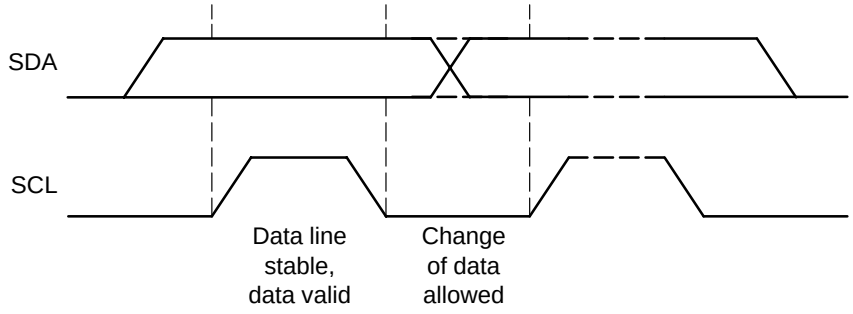
Pin	Pads	System Reset	I2C Reset (I2C_EN =0)
SDA	I/O	High impedance	High impedance
SCL	I/O	High impedance	High impedance

The master device generates one clock pulse for each data bit transferred. Due to variety of different technology devices (CMOS, NMOS, bipolar) that can be connected to the I2C bus, the levels of logical 0 (low) and 1 (high) are not fixed and depend on the associated level of VDD.

7.8.1.3 I^2C Bus Base Principal

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 only change when the clock signal on the SCL line is low (see Figure 7–24).

Figure 7–24. Data Validity on the I^2C Bus

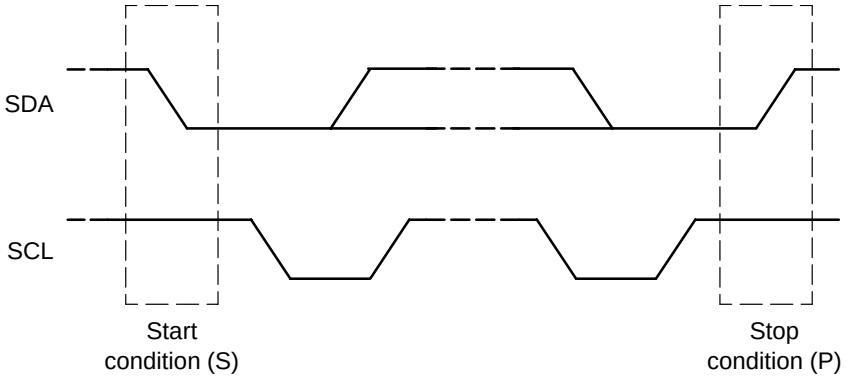


The I^2C module generates start and stop conditions when it is configured as a master (see Figure 7–25):

- ❏ 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 busy after the start condition (BB = 1) and free after the stop condition (BB = 0).

Figure 7–25. Start and Stop Conditions

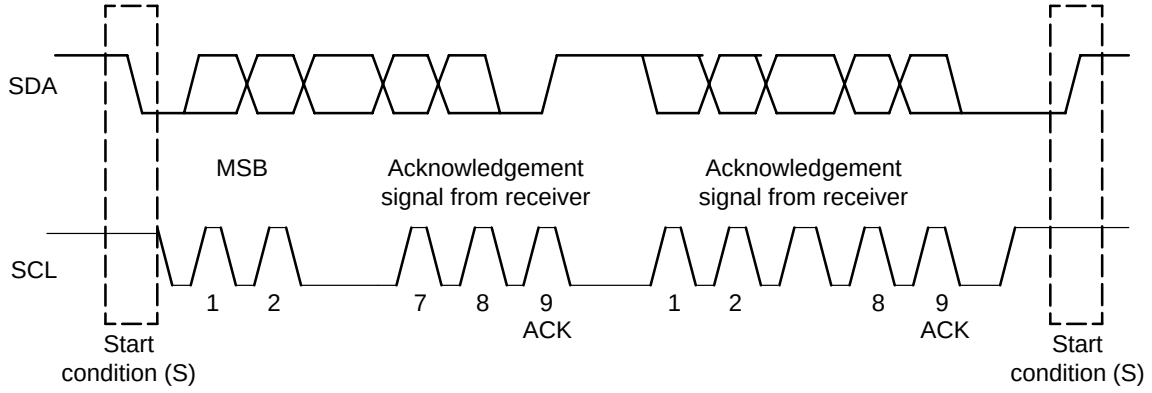


7.8.1.4 I²C Operation

Serial Data Formats

Each byte put on the SDA line is 8 bits long. The number of bytes that can be transmitted or received is unrestricted. The data is transferred with the most significant bit (MSB) first. Each byte is followed by an acknowledge bit from the module I²C if it is in receiver mode (see Figure 7–26).

Figure 7–26. I²C Data Transfer



The I²C protocol supports two data formats that are shown in Figure 7–27.

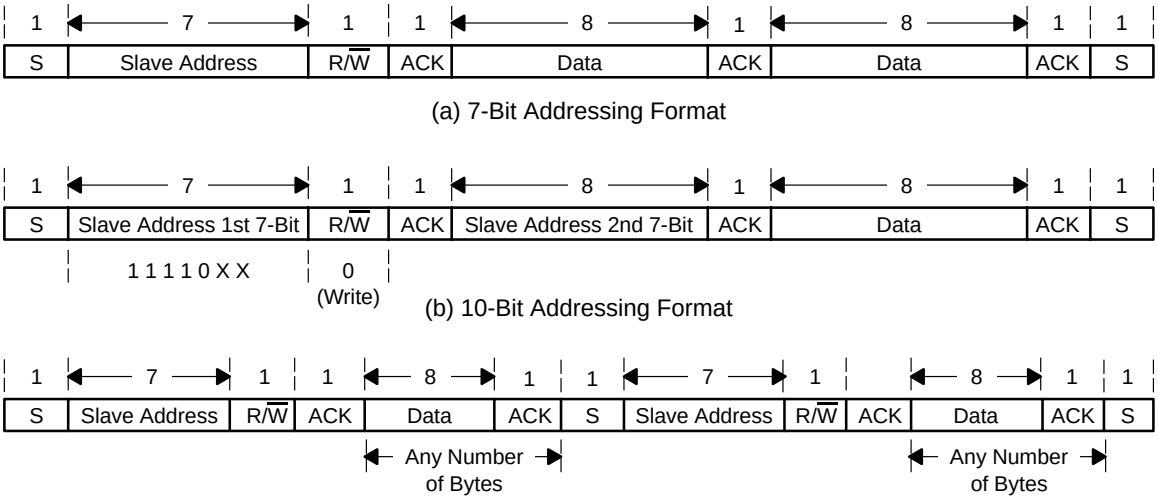
- 7-bit/10-bit addressing format
- 7-bit/10-bit addressing format with repeated start condition

The first byte after a start condition (S) always consists of 8 bits. In the acknowledge mode, an extra bit dedicated for acknowledgement bit is inserted after each byte.

In the addressing formats with 7-bit addresses, the first byte is composed by seven MSB slave address bits and one LSB R/$\overline{W}$ bit. While in the addressing formats with 10-bit addresses, the first byte is composed by seven MSB slave address, such as 11110XX, where XX is the two MSB of the 10-bit addresses, and one LSB R/$\overline{W}$ bit, which is 0 in this case.

The least significant R/$\overline{W}$ of the address byte indicates the direction of transmission of the following data bytes. If R/$\overline{W}$ is 0, the master writes (transmit) data into the selected slave; if it is 1, the master reads (receive) data out of the slave.

Figure 7–27. I^2C 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 synchronism with the self-generated clock pulses on the serial clock line SCL. The clock pulses are inhibited and SCL held low when the intervention of the processor is required after a byte has been transmitted.

Master Receiver

This mode can only be entered from the master transmitter mode. With either of the address formats (Figure 7–27 (a), (b), and (c)), the master receiver is entered after the slave address byte and bit R/$\overline{W}$ has been transmitted if R/$\overline{W}$ is high. Serial data bits received on bus line SDA are shifted in synchronism 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 after a byte has been transmitted. At the end of a transfer, it generates the stop condition.

Slave Transmitter

This mode can only be entered from the slave receiver mode. With either of the address formats (Figure 7–27 (a), (b), and (c)), the slave transmitter is entered if the slave address byte is the same as its own address and bit R/$\overline{W}$ has been transmitted if R/$\overline{W}$ is high. The slave transmitter shifts the serial data

out on the data line SDA in synchronism with the clock pulses that are generated by the master device. It does not generate the clock, but it can hold the clock line SCL low while intervention of the local host is required.

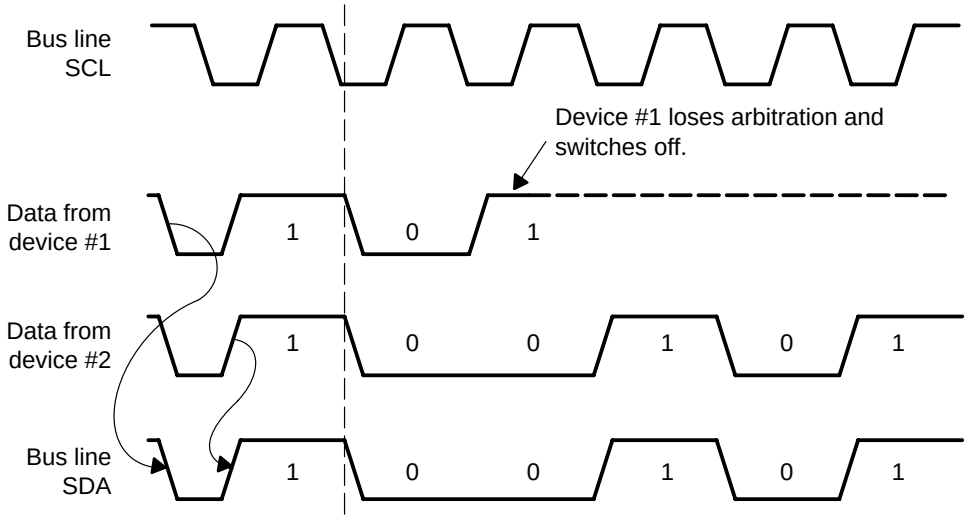
Slave Receiver

In this mode serial data bits received on the bus line SDA are shifted-in synchronously with the clock pulses on SCL, which are generated by the master device. It does not generate the clock, but it can hold the clock line SCL low while intervention of the local host is required following the reception of a byte.

Arbitration

If two or more master transmitters start a transmission on the same bus almost simultaneously, 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. Figure 7–28 shows the arbitration procedure between two devices. The arbitration procedure gives priority to the device that transmits the serial data stream with the lowest binary value. If two or more devices send identical first bytes, arbitration continues on the subsequent bytes.

Figure 7–28. Arbitration Procedure Between Two Master Transmitters

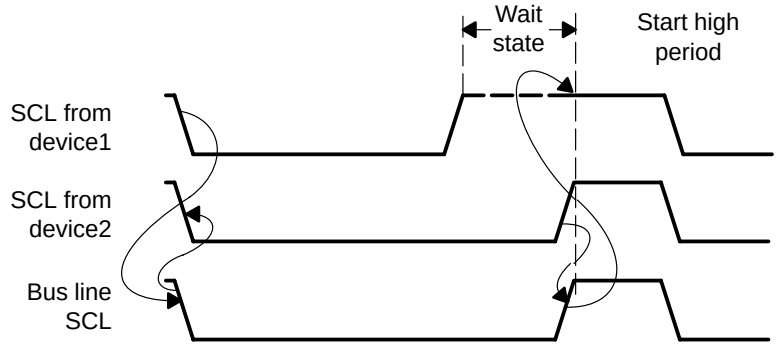


I²C Clock Generation and I²C Clock Synchronization

Under normal conditions, only one master device generates the clock signal, SCL. During the arbitration procedure, however, there are two or more master devices and the clock must be synchronized so that the data output can be compared. The wired-AND property of the clock line means that a device that first generates a low period of the clock line overrules the other devices. At this high/low transition, the clock generators of the other devices are forced to start generation of their own low period. The clock line then is held low by the device with the longest low period, while the other devices that finish their low periods must wait for the clock line to be released before starting their high periods. A synchronized signal on the clock line is thus obtained, where the slowest device determines the length of the low period and the fastest 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 to prepare a byte to be transmitted. Figure 7–29 illustrates the clock synchronization.

Figure 7–29. Synchronization of Two I²C Clock Generators



7.8.2 OMAP5910 I2C (Master/Slave I2C Controller)

The multimaster I2C peripheral provides an interface between TIPB bus and any I2C-bus compatible devices that connect via the I2C serial bus. External components attached to the I2C bus can serially transmit/receive up to 8-bit data to/from the local host device through the two-wire I2C interface. All references to a local host in this section refer to the MPU processor.

This I2C peripheral supports any slave or master I2C-compatible device. Figure 7–23 shows an example of a system with multiple I2C-compatible devices in which the I2C serial ports are all connected together for a two-way transfer from one device to other devices.

7.8.2.1 I2C Controller Features

The main features of the I2C controller are as follows:

- ❏ Compliant with Philips I2C specification version 2.1 [1]
- ❏ Support standard mode (up to 100 kbit/s) and Fast mode (up to 400 kbit/s)
- ❏ 7-bit and 10-bit device addressing modes
- ❏ General call
- ❏ Start/Restart/Stop
- ❏ Multimaster transmitter/slave receiver mode
- ❏ Multimaster receiver/slave transmitter mode
- ❏ Combined master transmit/receive and receive/transmit mode
- ❏ Built-in FIFO for buffered read or write
- ❏ Module enable/disable capability
- ❏ Programmable clock generation
- ❏ 16-bit wide access to maximize bus throughput
- ❏ Designed for low power
- ❏ Two DMA channels
- ❏ Wide interrupt capability

The present I2C does *not* support:

- ❏ High-speed (HS) mode for transfer up to 3.4M bits
- ❏ C-bus compatibility mode.

7.8.2.2 Data Format

The I2C controller operates in 16-bit word data format (byte write access supported for the last access), and it supports endianism.

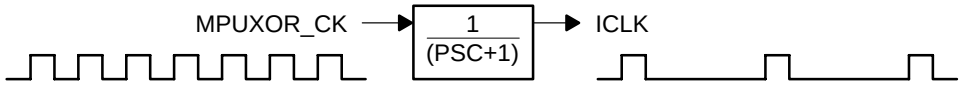
7.8.2.3 I2C Reset

The I2C_EN bit in the I2C configuration register (I2C_CON) can also reset the I2C module. When the system bus reset is removed (RESET_ = 1), I2C_EN = 0 keeps the I2C module in reset state.

7.8.2.4 Prescaler (ICLK)

The I2C module is operated with an internal ~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 peripheral clock (MPUXOR_CK) to obtain a ~12-MHz clock for the I2C module (see Figure 7–30).

Figure 7–30. Prescale Sampling Clock Divider Value



0x0: Divide by 1
0x1: Divide by 2
↓ ↓
0xFF: Divide by 256

Values after reset are low (All 8 bits).

Noise Filter

The noise filter suppresses any noise that is 50 ns or less. It is designed to suppress noise with one ICLK assuming the lower and upper limits of ICLK are 8 MHz and 16 MHz respectively.

7.8.2.5 I2C Interrupts

The I2C module generates five types of interrupt: arbitration-lost, no-acknowledge, registers-ready-for-access, receive, and transmit. These five interrupts are accompanied with five interrupt masks and flags defined in the I2C_IE and I2C_STAT registers respectively.

An arbitration-lost interrupt (AL) is generated when the I2C arbitration procedure is lost.

A no-acknowledge interrupt (NACK) is generated when the master I2C does not receive an acknowledge from the receiver.

A registers-ready-for-access interrupt (ARDY) is generated by the I2C when the previously programmed address, data and command have been performed and the status bits has been updated. This interrupt is used to let the local host knows that the I2C registers are ready to be accessed.

Receive interrupt/status (RRDY) is generated when there was received data ready to be read by the local host from the I2C_DATA register. This bit can also be polled by the local host to read the received data from the I2C_DATA register.

Transmit interrupt/status (XRDY) is generated when the local host needs to put another data in the I2C_DATA register after the transmitted data has been shifted out on the SDA pin. This bit can also polled by the local host to write the next transmitted data into the I2C_DATA register.

The interrupt vector register, I2C_IVR, contains one of the binary-coded-interrupt-vector to indicate which interrupt has occurred. Reading the I2C_IVR clears the interrupt flag; if other interrupts are pending, a new interrupt is generated. If there are more than one interrupt flag, reading the I2C_IVR clears the highest priority interrupt flag.

The I2C interrupt signal (I2C_IRQ) is one local host pulse clock wide active high signal. It must be considered an edge-sensitive input by the interrupt handler.

7.8.2.6 DMA Events

The I2C module can generate two DMA requests events, read (I2C_DMA_RX) and write (I2C_DMA_TX), that can be used by the DMA controller 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 one local host-pulse-clock-wide, active high signals for every new 16-bit word to be read or write in the FIFOs. They must be considered as edge sensitive inputs by the DMA.

7.8.2.7 I²C Registers

Table 7–52 lists the I²C registers. Table 7–53 through Table 7–71 describe the register bits.

Table 7–52. I²C Registers

Register	Description	Access	Offset Address
I2C_REV	I²C module version	R	0x00
I2C_IE	I²C interrupt enable	R/W	0x04
I2C_STAT	I²C status	R	0x08
I2C_IV	I²C interrupt vector	R	0x0C
Reserved			0x10
I2C_BUF	I²C buffer configuration	R/W	0x14
I2C_CNT	I²C data counter	R/W	0x18
I2C_DATA	I²C data access	R/W	0x1C
Reserved			0x20
I2C_CON	I²C configuration	R/W	0x24
I2C_OA	I²C own address	R/W	0x28
I2C_SA	I²C slave address	R/W	0x2C
I2C_PSC	I²C clock prescaler	R/W	Ox30
I2C_SCLL	I²C SCL low time control	R/W	0x34
I2C_SCLH	I²C SCL high time control	R/W	0x38
I2C_SYSTEST	I²C system test	R/W	0x3C

The read-only I²C module version register (I2C_REV) contains the hard coded revision number of the module. A write to this register has no effect.

This 8-bit field (7:0) indicates the revision number of the current I²C controller module. Its value is fixed by hardware and corresponds to the RTL revision of this module.

The four LSBs indicate a minor revision.

The four MSBs indicate a major revision.

- Ex: 0x10: version 1.0
- 0x11: version 1.1

A reset has no effect on the value returned.

Table 7–53. I2C Module Version Register (I2C_REV)

Bit	Name	Description
15–8	–	Reserved
7-0	REV	Module version number

The read/write I2C interrupt enable register (I2C_IE) controls interrupts mask/unmask function.

Table 7–54. I2C Interrupt Enable Register (I2C_IE)

Bit	Name	Description
15–5	–	Reserved
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

Common to all bits:

When a bit location is set to 1 by the local host, an interrupt is signaled to the local host if the corresponding bit location in the I2C status register (I2C_STAT) is asserted to 1 by the core.

If set to 0 the interrupt is masked and not signaled to the local host.

- 0: Interrupt disabled
- 1: Interrupt enabled

Values after reset are low (all bits)

The read-only I2C status register (I2C_STAT) provides core status information for interrupt handling and other I2C control management. This register is always read before reading the I2C_IV register itself to retain an accurate status (some bits are cleared following a read into I2C_IV).

Table 7–55. I2C 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	AD0	Address zero
7:5	–	Reserved
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

Single Byte Data (SBD)

This read-only bit (15) is set to 1 in slave receive or master receive modes when the last byte that was read from I2C_DATA register contains a single valid byte.

This bit is cleared to 0 by the core when the local host reads the I2C_IV register if INTCODE is register access ready.

- When SBD = 1, in little-endian data format (BE = 0) the MS byte reads as 0x00 and in big-endian format (BE = 1) the LS byte reads as 0x00.
- Whenever the number of bytes to be received is unknown (ex: slave receiver), the local host must poll this bit prior to attempting to read I2C_IV.
  - 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 (12) indicates the state of the serial bus.

- In the slave mode, on reception of a start condition, the device sets BB to 1. BB is clear to 0 after reception of a stop condition.
- In the master mode, the software controls BB. To start a transmission with a start condition, MST, TRX, and STT must be set to 1. To end a transmission with a stop condition, STP must be set to 1. 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 (11) indicates whether the receiver has experienced overrun. Overrun occurs when the receive shift register (ICRSR) 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) and prevent others bytes from being received.

- ROVR is set to 1 when the I^2C has recognized an overrun.
- ROVR is clear when reading the I2C_DATA register or resetting the I^2C (I2C_EN=0).
  - 0: Normal operation
  - 1: Receiver overrun

Value after reset is low.

Transmit Underflow (XUDF)

This read-only bit (10) indicates whether the transmitter has experienced underflow.

- In the master transmit mode, underflow occurs when the transmit shift register (ICXSR) is empty, the transmit FIFO is empty, and there are still bytes to transmit (DCOUNT $\neq$ 0).
- In the slave transmit mode, underflow occurs when the transmit shift register (ICXSR) is empty, the transmit FIFO is empty, and there are still bytes to transmit (read request from external I^2C master).

- XUDF is set to 1 when the I^2C has recognized an underflow. The core holds the line till the underflow cause has disappeared.
- XUDF is clear when writing I2C_DATA register or resetting the I^2C (I2C_EN=0).
  - 0: Normal operation
  - 1: Transmit underflow

Value after reset is low.

Address As Slave (AAS)

This read-only bit (9) is set to 1 by the device when it has recognized its own slave address or an address of all (8) zeros. The AAS bit is reset to 0 by restart or stop.

- 0: No action
- 1: Address as slave

Value after reset is low.

Address Zero Status (AD0)/General Call

This read-only bit (8) is set to 1 by the device if it detects the address of all eight zeros (that is, general call). The AD0 bit is reset to 0 (default value) when a start or stop condition is detected.

This bit must be checked following a shared NACK/general call Interrupt to determine the source of the interrupt.

When this bit is set to 1, AAS also reads as set to 1.

- 0: No action
- 1: General call

Value after reset is low.

Transmit Data Ready (XRDY)

Transmit mode only.

XRDY (bit 4) is set to 1 when I^2C peripheral is a master or slave transmitter, the local host is able to write a new data into the I2C_DATA register, and the transmitter still requires a new data. A master transmitter requests new data if DCOUNT $\neq$ 0, and a slave transmitter requests new data if a read request from external master.

Note:

The transmitter requests 2 bytes to be written even if only a single byte is needed. In this case, the other byte must be filled with a dummy 0x00 value that is not transmitted over the I^2C line.

XRDY is automatically cleared to 0 by the core when I2C_DATA is written and the transmit FIFO buffer is full. The local host can also poll this bit to write newly transmitted data into I2C_DATA register.

- 0: Transmit buffer full (or receiver mode)
- 1: Transmit data ready (for write) and byte is needed.

Value after reset is low.

Receive Data Ready (RRDY)

RRDY (bit 3) is set to 1 when the local host is able to read new data from the I2C_DATA register. RRDY is automatically cleared to 0 by the core when the I2C_DATA is read and the receive FIFO buffer is empty. The local host can also poll this bit to read the received data in the I2C_DATA register.

Interrupt mode, the local host 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 must be cleared to 0 to receive a new RRDY interrupt.

- 0: Receive buffer empty
- 1: Receive data ready (for read)

Value after reset is low.

Register Access Ready (ARDY)

This bit (2) when set to 1 indicates that the previously programmed data and command (receive or transmit, master or slave) have been performed and the status bit has been updated. This flag is used by the local host to let it know that the I^2C registers are ready to be accessed again.

Table 7–56. Register Access Ready (ARDY) Set Conditions

Mode	Others	ARDY Set Conditions
Master transmit	STP = 1, RM = 0	DCOUNT=0
Master receive	STP = 1, RM = 0	DCOUNT = 0 and receiver FIFO empty
Master transmit or receive	STP = 0, RM = 0	DCOUNT passed 0
Master transmit or receive	RM=1	Never
Slave transmit	–	Stop condition received from master
Slave receive	–	Stop condition and receiver FIFO empty

This bit is cleared to 0 by the core with a read of the matching interrupt vector in I2C_IV register.

- ❏ 0: No action
- ❏ 1: Access ready

Value after reset is low.

No Acknowledgment (NACK)

The no acknowledge flag bit (1) is set when the hardware detects no acknowledge has been received.

This bit is cleared to 0 by the core with a read of the matching interrupt vector in I2C_IV register.

- ❏ 0: Normal/no action required
- ❏ 1: NACK

Value after reset is low.

When a NACK occurs, the system has to perform the following actions to recover:

1) Read the INTCODE in the I2C_IV register to release NACK in I2C_STAT.

2) Write to the STP bit in the I2C_CON register to realse I^2C data line.

Do not poll the NACK and AL bits in the I2C_CON register because an update could be missed. These bits require an interrupt process to be handled correctly (also, the INTCODE field in the I2C_IV register should be read before any action is taken in the subroutine).

Arbitration Lost (AL)

The arbitration lost flag bit is set to 1 when the device in the master transmitter mode senses it has lost an arbitration 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 MST/STP bits are automatically cleared by the core and the I^2C becomes a slave receiver.

This bit is cleared to 0 by the core with a read of the matching interrupt vector in I2C_IV register.

- ❏ 0: Normal/no action required
- ❏ 1: Arbitration lost

Value after reset is low.

Table 7–57. I^2C Interrupt Vector Register (I2C_IV)

Bit	Name	Description
15–3	–	Reserved
2–0	INTCODE	Interrupt code

Interrupt Code (INTCODE)

The binary-coded-interrupt vector (bit 2–> 0) indicates which interrupt has occurred. Reading the I2C_IV clears the interrupt flag; if other interrupts are pending, a new interrupt is generated. If there is more than one interrupt flag, reading the I2C_IV clears the highest priority interrupt flag.

Values after reset are low (all 3 bits).

Table 7–58. Interrupt Code (INTCODE) Conditions

Interrupt Code	Interrupt Occurred	Priority
000	None	–
001	Arbitration lost interrupt	Highest
010	No acknowledgement interrupt/general call	↓
011	Register access ready interrupt	
100	Receive data ready interrupt	
101	Transmit data ready interrupt	Lowest
Others	Reserved	–

The read/write I2C buffer configuration register (I2C_BUF) enables DMA transfers.

Table 7–59. I2C 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

Receive DMA Channel Enable (RDMA_EN)

When this bit (15) is set to 1, the receive DMA channel is enabled and the receive data ready interrupt is automatically disabled (RRDY_IE bit cleared).

- ❏ 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 transmit data ready interrupt is automatically disabled (XRDY_IE bit cleared).

- ❏ 0: Transmit DMA channel disabled
- ❏ 1: Transmit DMA channel enabled

Value after reset is low.

The read/write I2C data counter register (I2C_CNT) controls the numbers of bytes in the I2C data payload.

Table 7–60. I²C Data Counter Register (I2C_CNT)

Bit	Name	Description
15–0	DCOUNT	Data count

Data Count (DCOUNT)

Master mode 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 (STP = 1) and the ARDY status flag is set to 1.

If STP = 0, then the I²C asserts SCL = 0 when DCOUNT reaches 0. The local host 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 STP is set to 1 by the LH.

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: Reserved value. Do not use this setting.
- 0x1: Data counter = 1 bytes.
- ↓ ↓
- 0xFFFF: Data counter = 65535 bytes (2^{16} -1)

Note that DCOUNT is a *don't care* when RM is set to 1.

Values after reset are low (all 16 bits).

The I2C data access register (I2C_DATA) is the entry point for the local host to read data from, or write data into, the FIFO buffer. The FIFO size is 2x16bits (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).

Table 7–61. I2C Data Access Register (I2C_DATA)

Bit	Name	Description
15–0	DATA	Transmit/Receive FIFO data

When read, this register contains the received I2C data packet (1 or 2 bytes). This register must be accessed in 16-bit mode by the LH. In case of an odd number of bytes received to read, the upper byte of the last access always reads as 0x00. The local host must check the SBD status bit in I2C_STAT register to flush this null byte.

When written, this register contains the byte(s) value(s) to transmit over the I2C data line (1or 2 bytes). This register must be accessed in 16-bit mode except for the last byte in case of an odd number of bytes to transmit. The last byte of the data packet may be written using a byte write access or a 16-bit write access.

When transmit FIFO, the last data transfer must be a 16-bit transfer when it is written by the DMA, and it can either be an 8-bit or 16-bit transfer when it is written by the MPU. When an odd number of bytes is to be transferred, the DMA uses all 16-bit transfers and fills the unused byte (upper or lower byte according to the selected endianism) of the last 16-bit transfers with all 0s.

In SYSTEST loop back 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 7–62. 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
7–3	Reserved	
2	RM	Repeat mode (master mode only)
1	STP	Stop condition (master mode only)
0	STT	Start condition (master mode only)

I²C Module Enable (I2C_EN)

When this bit (15) 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.

The local host must set this bit to 1 for normal operation.

- ❏ 0: I²C controller in reset
- ❏ 1: I²C module enabled

Value after reset is low.

I²C Big Endian (BE)

When this bit (14) is 0 (default), the FIFO is accessed in little endian format. In transmit mode, the LS byte (I2C_DATA[7:0]) is transmitted first and the MS byte (I2C_DATA[15:8]) is transmitted in 2nd position over the I²C line. Conversely, in receive mode, the 1st or odd byte received (1, 3, 5…) is stored in the LS byte position and the 2nd or even byte received in the MS byte position.

When the local host 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 LS byte (I2C_DATA[7:0]) is transmitted in 2nd position over the I²C line. Conversely, in receive mode, the 1st or odd byte received (1,3, 5…) is stored in the MS byte position and the 2nd or even byte received in the LS byte position.

- 0: Little endian mode
- 1: Big endian mode

Value after reset is low.

Start Byte (STB)

Master mode only.

The start byte mode bit (11) is set to 1 by the local host to configure the I^2C in start byte mode (I2C_SA=00000001). See the Philips I^2C specification for more details.

- 0: Normal mode
- 1: Start byte mode

Value after reset is low.

Master/Slave Mode (MST)

When this bit (10) 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.

Once set, this bit is automatically cleared by a stop condition.

- 0: Slave mode
- 1: Master mode

Value after reset is low.

Transmitter/Receiver Mode (TRX)

Master mode only.

When this bit (9) 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 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.

Table 7–63 defines the operating modes.

Table 7–63. 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 (8) expands the address to 10-bit.

- ❏ 0: 7-bit address mode
- ❏ 1: 10-bit address mode

Value after reset is low.

Repeat Mode (RM)

Mater mode only.

This bit (2) is set to a 1 by the local host to put the I^2C in the repeat mode. In this mode, data is continuously transmitted out of the I2C_DATA transmit register until the STP bit is set to 1 regardless of DCOUNT value. This bit is *don't care* if the I^2C is configured in slave mode.

- ❏ 0: Normal mode
- ❏ 1: Repeat mode

Value after reset is low.

Table 7–64. Repeat Mode Conditions

RM	STT	STP	Conditions	Bus Activities	Mode
0	0	0	Idle	None	NA
0	0	1	Stop	P	NA
0	1	0	(Re)Start	S-A-D..(n)..D	Repeat n
0	1	1	(Re)Start-Stop	S-A-D..(n)..D-P	Repeat n
1	0	0	Idle	none	NA
1	0	1	Stop	P	NA
1	1	0	(Re)Start	S-A-D-D-D…..	Continuous
1	1	1	Reserved	None	NA

Stop Condition (STP)

Master mode only.

This bit (1) can be set to a 1 by the local host to generate a stop condition. It is reset to 0 by the hardware after the stop condition has been generated. The stop condition is generated when DCOUNT passes 0.

- 0: No action or stop condition detected
- 1: Stop condition queried

Value after reset is low.

Start Condition (STT)

Master mode only.

This bit (0) can be set to a 1 by the local host to generate a start condition. It is reset to 0 by the hardware after the start condition has been generated. The start/stop bits can be configured to generate different transfer formats. The STT and STP can be used to terminate the repeat mode.

- 0: No action or start condition generated
- 1: Start

Value after reset is low.

Table 7–65. STT Settings

STT	STP	Conditions	Bus Activities
1	0	Start	S-A-D
0	1	Stop	P
1	1	Start/stop (COUNT= n)	S-A-D..(n)..D-P
1	0	Start (DCOUNT= n)	S-A-D..(n)..D

DCOUNT is data count value.

The I2C own address register (I2C_OA) specifies the module I2C 7-bit or 10-bit address (own address).

Table 7–66. I2C Own Address Register (I2C_OA)

Bit	Name	Description
15–10	Reserved	
9–0	OA	Own address

This field (bits 9-0) specifies either:

- A 10-bit address coded on OA[9:0] when XA (expand address, I2C_MCR[8]) is set to 1.
- A 7-bit address coded on OA[6:0] when XA (expand address, I2C_MCR[8]) is set to 0. In this case, OA[9:7] bits must be set to 000 by application software.

Values after reset are low (all 10 bits).

The I2C slave address register (I2C_SA) specifies the addressed I2C module 7-bit or 10-bit address (slave address).

Table 7–67. I2C Slave Address Register (I2C_SA)

Bit	Name	Description
15–10	Reserved	
9–0	SA	Slave address This field (bits 9:0) specifies either: ❑ A 10-bit address coded on SA[9:0] when XA (expand address, I2C_MCR[8]) is set to 1. ❑ A 7-bit address coded on SA[6:0] when XA (expand address, I2C_MCR[8]) is set to 0. In this case, SA[9:7] bits must be set to 000 by application software. Values after reset are high (all 10 bits).

This register is used to specify the internal clocking of the I2C peripheral core.

Table 7–68. I2C Clock Prescaler Register (I2C_PSC)

Bit	Name	Description
15–8	Reserved	
7–0	PSC	Prescale sampling clock divider value The core (bits 7-0) uses this 8-bit value to divide the peripheral clock (MPUXOR_CK) to 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 ❑ All other settings are Reserved. Values after reset are low (all 8 bits).

This I2C SCL low-time control register (I2C_SCLL) is used to determine the SCL low-time value when master.

Table 7–69. I2C SCL Low-Time Control Register (I2C_SCLL)

Bit	Name	Description
15–8	Reserved	
7–0	SCLL	SCL low0x0: 6 * ICLK time period time Master mode only. This 8-bit value (bits 7:0) is used to generate the SCL low-time value (t_{LOW}) when the peripheral is operated in master mode. The SCL low-time equals (SCLL+6) * ICLK time period (internal sampling clock rate). ❑ 0x0: 6 * ICLK time period ❑ 0x1: 7 * ICLK time period ❑ ↓↓ ❑ 0xFF: 261 * ICLK time period Values after reset are low (all 10 bits).

The I2C SCL high-time control register (I2C_SCLL) determines the SCL high-time value when master.

Table 7–70. I2C SCL High Time Control Register (I2C_SCLH)

Bit	Name	Description
15–8	Reserved	
7–0	SCLH	SCL high time Master mode only. This 8-bit value (bits 7-0) is used to generate the SCL high time value (t_{HIGH}) when the peripheral is operated in master mode. The SCL high time equals (SCLH+6) * ICLK time period (internal sampling clock rate). ❑ 0x0: 6 * ICLK time period ❑ 0x1: 7 * ICLK time period ❑ ↓↓ ❑ 0xFF: 261 * ICLK time period Values after reset are low (all 10 bits).

The I2C system test register (I2C_SYSTEST) is used to facilitate system level test 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 loop-back for self-test when the module is configured in system test (SYSTEST) mode. It also provides stop/no-stop function in debug mode. Never set for normal I2C operation.

Table 7–71. I2C 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
11–4	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 Enable (ST_EN)

This bit (15) 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 (14) is used to determine 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: Stops immediately if SCL is low and keeps driving SCL low whether I^2C is master transmitter/receiver. If SCL is high, I^2C waits until SCL becomes low and then stops. If the I^2C is a slave, it stops when the transmission/receiving completes.

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 (13-12) are *don't care*. They read always 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.

Table 7–72. TMODE Settings

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 are low (2 bits).

In SCL counter test mode, the SCL pin is driven with a permanent clock as if master with the parameters set in 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 is normally generated if enabled.

In SDA/SCL I/O mode, the SCL IO and SDA IO are controlled via the I2C_SYSTEST[3:0] register bits.

SCL Line Sense Input Value (SCL_I)

In normal functional mode (ST_EN = 0), this read-only bit (3) always reads as 0.

In system test mode (ST_EN = 1 and 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 (2) is *don't care*, and always reads as 0. Writes are ignored.

In system test mode (ST_EN = 1 and TMODE = 11), a 0 forces a low level on the SCL line and a 1 puts the I^2C output driver in a high-impedance state.

- ❏ 0: Force 0 on the SCL data line
- ❏ 1: SCL output driver in HI-Z state

Value after reset is low.

SDA Line Sense Input Value (SDA_I)

In normal functional mode (ST_EN = 0), this read-only-bit (1) always reads as 0.

In system test mode (ST_EN = 1 and 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 (0) is *don't care*, and always reads as 0. Writes are ignored.

In system test mode (ST_EN = 1 and TMODE = 11), a 0 forces a low level on the SDA line and a 1 puts the I^2C output driver in a high-impedance state.

- ❏ 0: Forces 0 on the SDA data line
- ❏ 1: SDA output driver in HIZ state

Value after reset is low.

7.8.3 Programming

7.8.3.1 Main Program

State after reset:

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 CPU frequency).
   - ■ If using interrupt for transmit/receive data, enable interrupt masks.
   - ■ If using DMA for transmit/receive data, enable the DMA and program the DMA controller.
2) Take the I^2C module out of reset (I2C_EN = 1).

Initialization procedure: Configure the I^2C mode register (I2C_CON) bits.

Program clock control registers (I2C_SCLL and I2C_SCLH): 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 CPU frequency).

- ❏ Configure address registers:
  - ■ Configure its own address (I2C_OA = x).
  - ■ Configure the slave address (I2C_SA = x).
- ❏ Program transmit data register (I2C_DATA): If in master transmitter mode, program the data transmit register (I2C_DATA = x).
- ❏ Configure status and mode register (I2C_STAT): 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 condition to initiate a transfer.

- Poll receive data: Poll the receive data ready interrupt flag bit (RRDY) in the I2C status register (I2C_STAT), use the RRDY interrupt, or use 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 I2C status register (I2C_STAT), use the XRDY interrupt, or use the DMA to write data into the data transmit register (I2C_DATA).

Interrupt subroutines:

1) Test for arbitration lost and resolve accordingly.
2) Test for no-acknowledge and resolve accordingly.
3) Test for register access ready and resolve accordingly.
4) Test for receive data and resolve accordingly.
5) Test for transmit data and resolve accordingly.

7.8.4 Flowcharts

Figure 7–31 through Figure 7–42 show the master/slave I2C flowcharts.

Figure 7–31. Setup Procedure

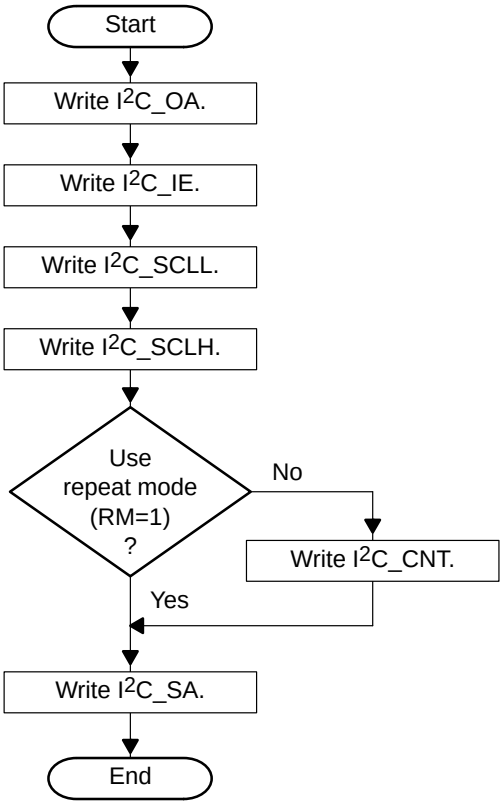
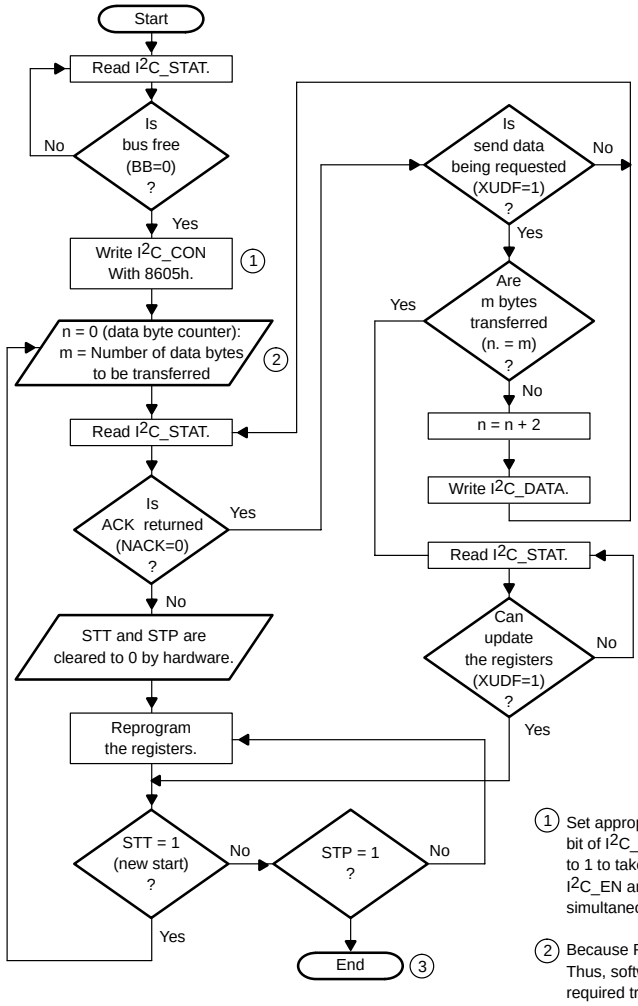


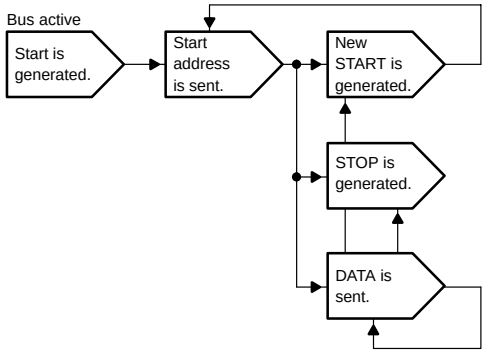
Figure 7–32. Master Transmitter Mode, RM = 1



1 Set appropriate values to every bit of I2C_CON. I2C_EN bit must be set to 1 to take I2C out of reset condition. Setting I2C_EN and setting other mode bits can be done simultaneously.

2 Because RM=1. hardware counter does not run. Thus, software counter counts the number of the required transfer.

3 The I2C goes into slave receiver mode.



[EXPECTED COMMAND]
At the beginning,
(STT,STP) = (1.0)
in the middle,
(STT, STP) = (0.0)
At the end,
(STT, STP) = (0.1)

[EXPECTED I2C_IE]
I2C_IE = 00000b

Figure 7–33. Master Receiver Mode, RM = 1, Polling 1 (Software Counter, Number of the Receive Data Fixed)

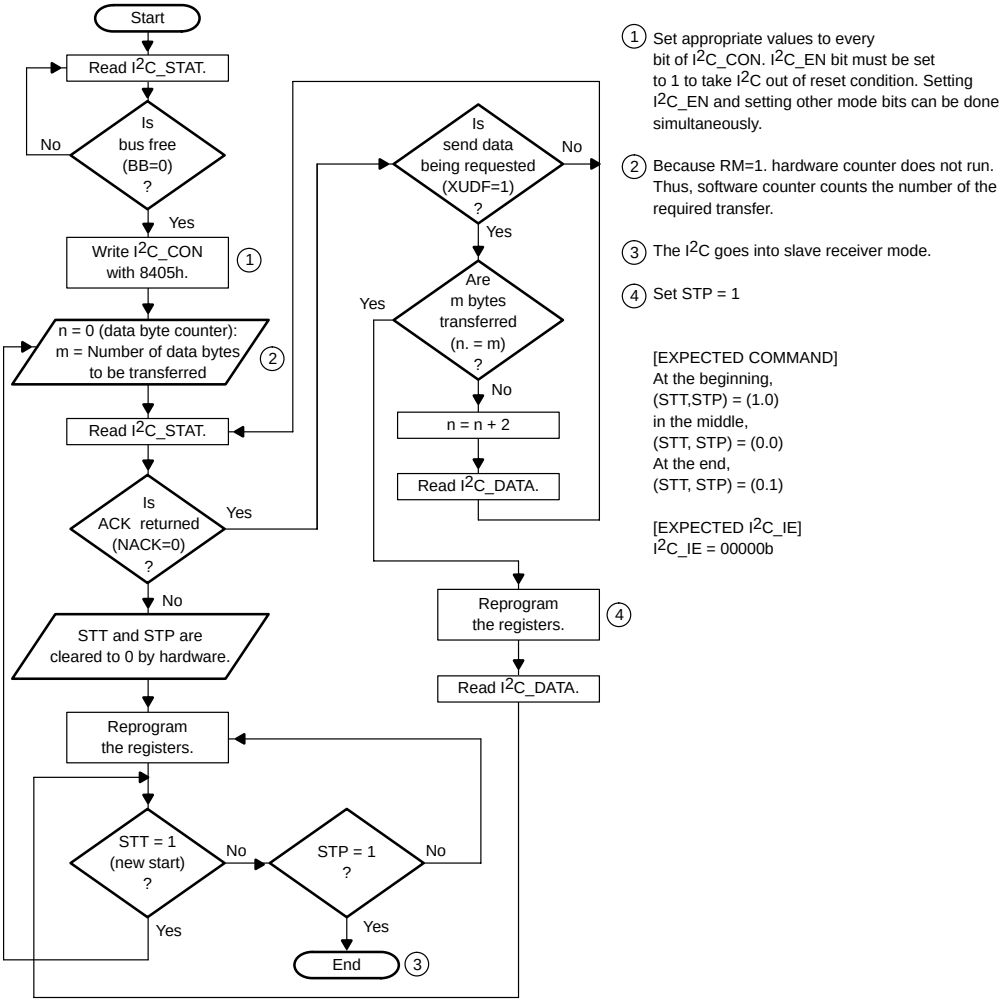
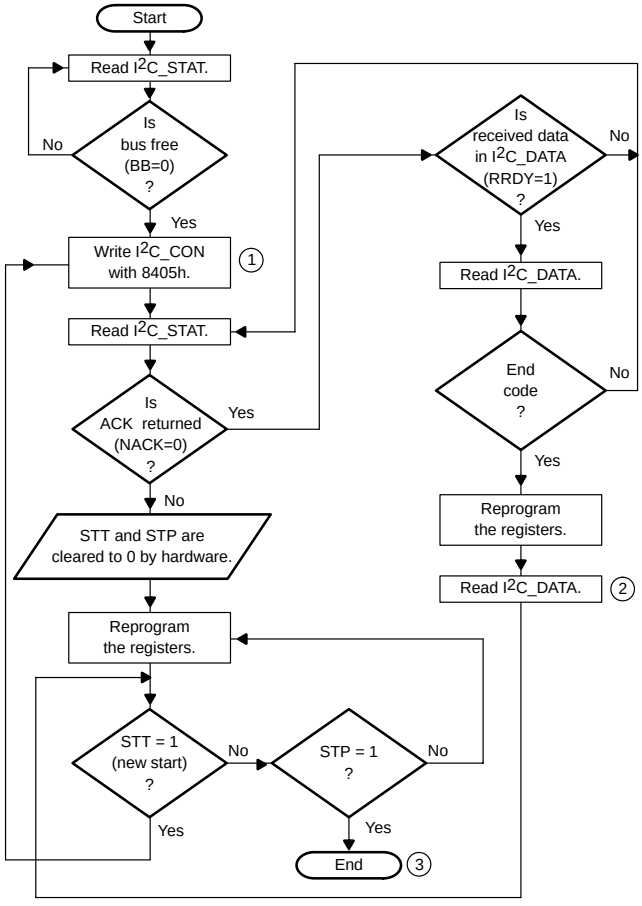


Figure 7–34. Master Receiver Mode, RM =1 , Polling 2 (Number of the Receive Data is Variable, Data Contents Dependent)



1 Set appropriate values to every bit of I2C_CON. I2C_EN bit must be set to 1 to take I2C out of reset condition. Setting I2C_EN and setting other mode bits can be done simultaneously.

2 Dummy read. The contents of this read data has no meaning.

3 The I2C goes into slave receiver mode.

[EXPECTED COMMAND]
At the beginning,
(STT,STP) = (1.0)
in the middle,
(STT, STP) = (0.0)
At the end,
(STT, STP) = (0.1)

[EXPECTED I2C_IE]
I2C_IE = 00000b

Figure 7–35. Master Transmitter Mode, RM = 0, Polling

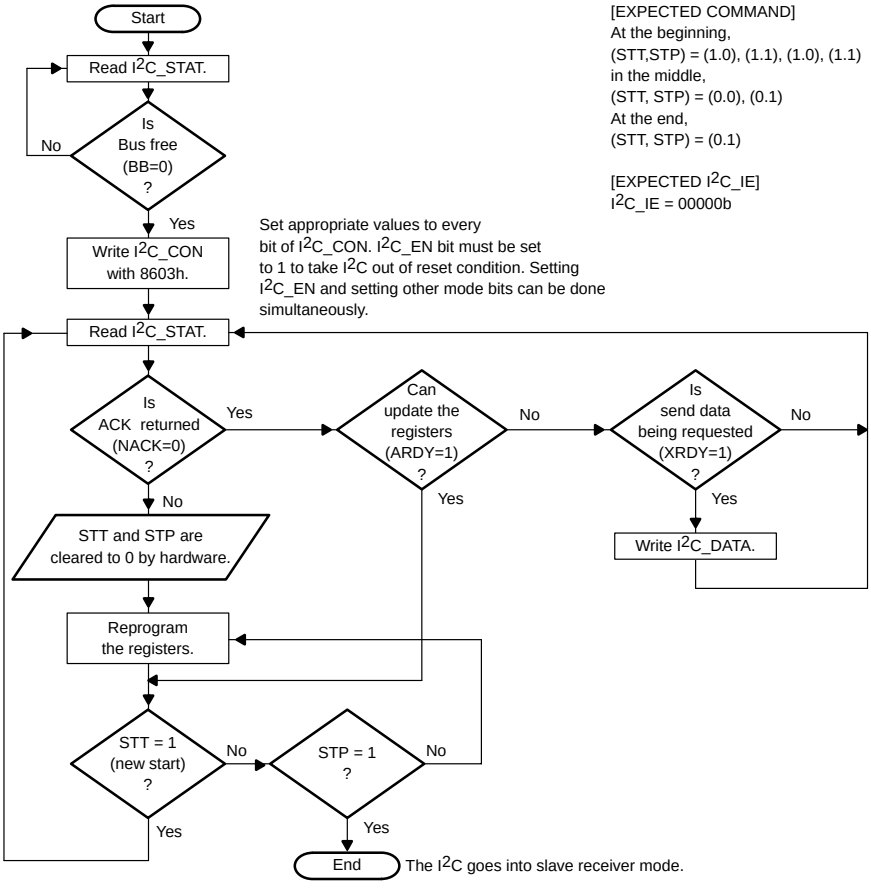


Figure 7–36. Master Receiver Mode, RM = 0, Polling

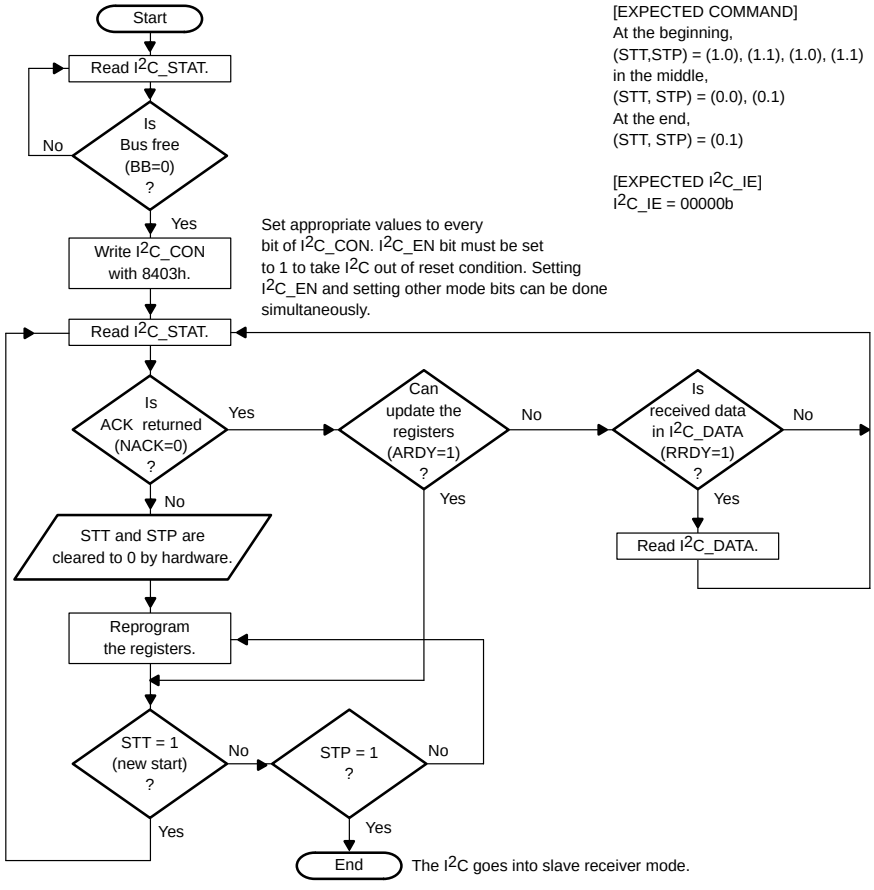


Figure 7–37. Master Transmitter Mode, RM = 0, Interrupt

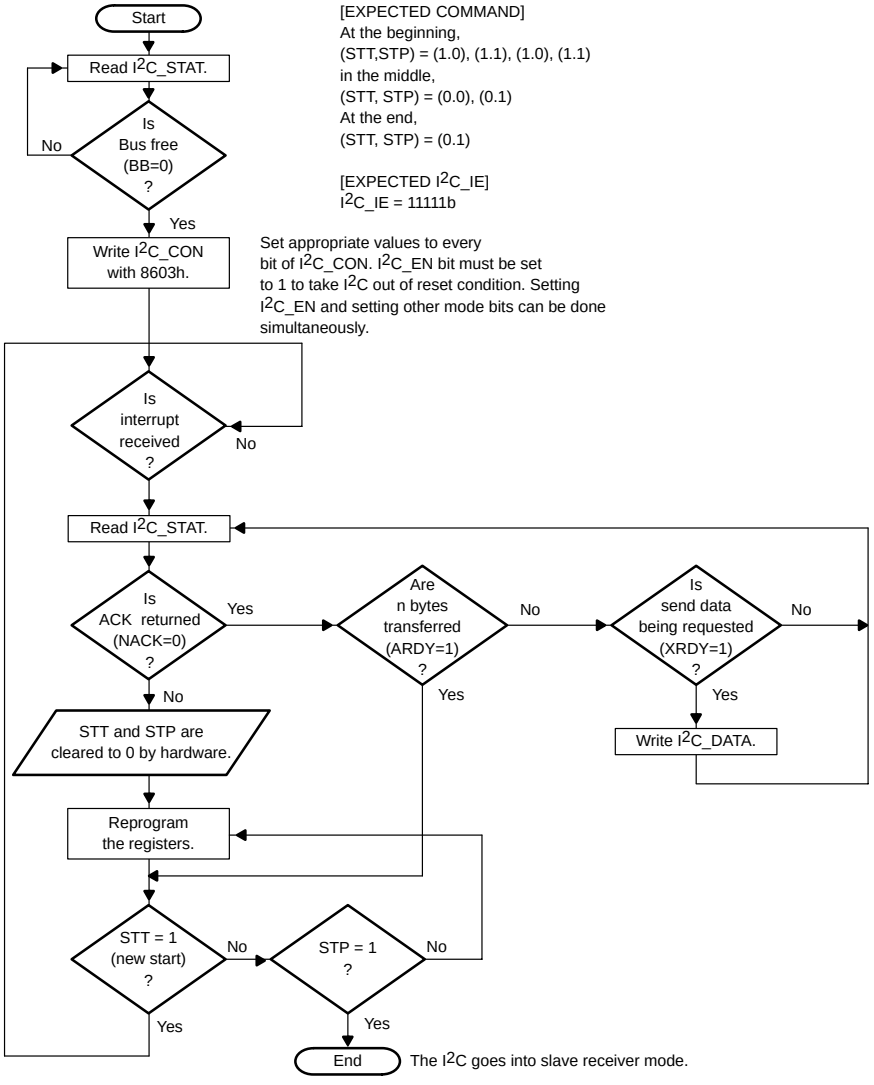


Figure 7–38. Master Receiver Mode, RM = 0, Interrupt

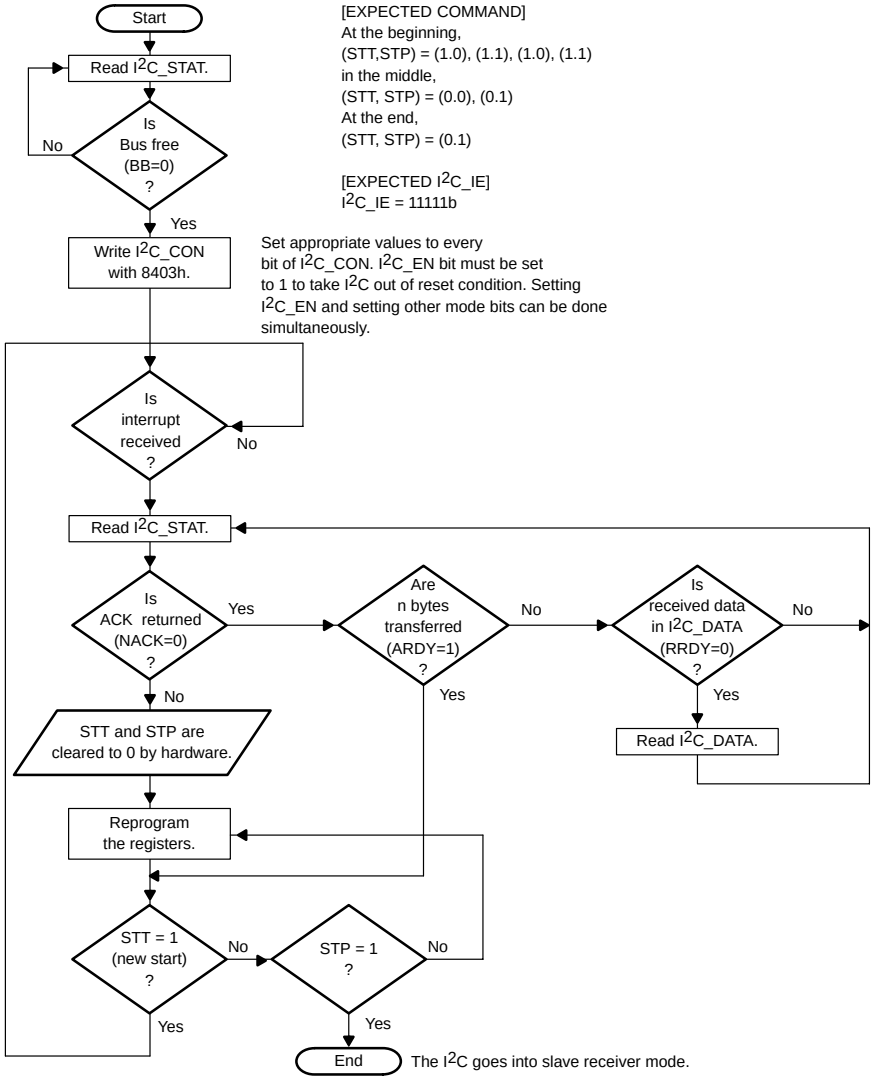


Figure 7–39. Master Transmitter Mode, RM = 0, DMA

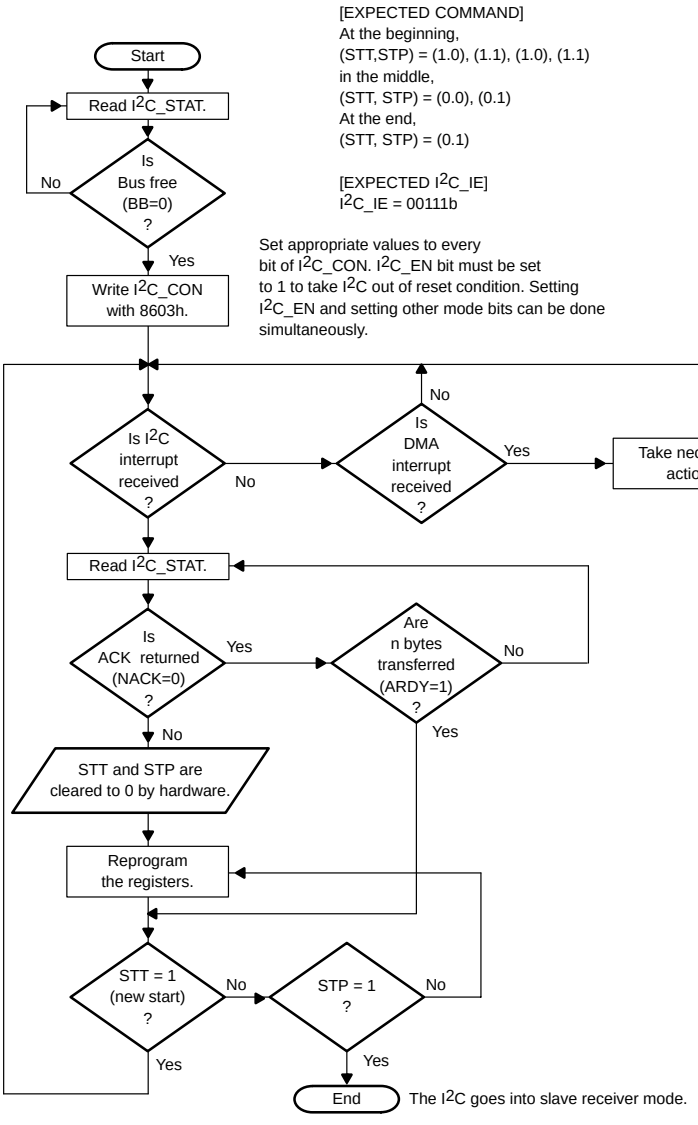


Figure 7–40. Master Receiver Mode, RM = 0, DMA

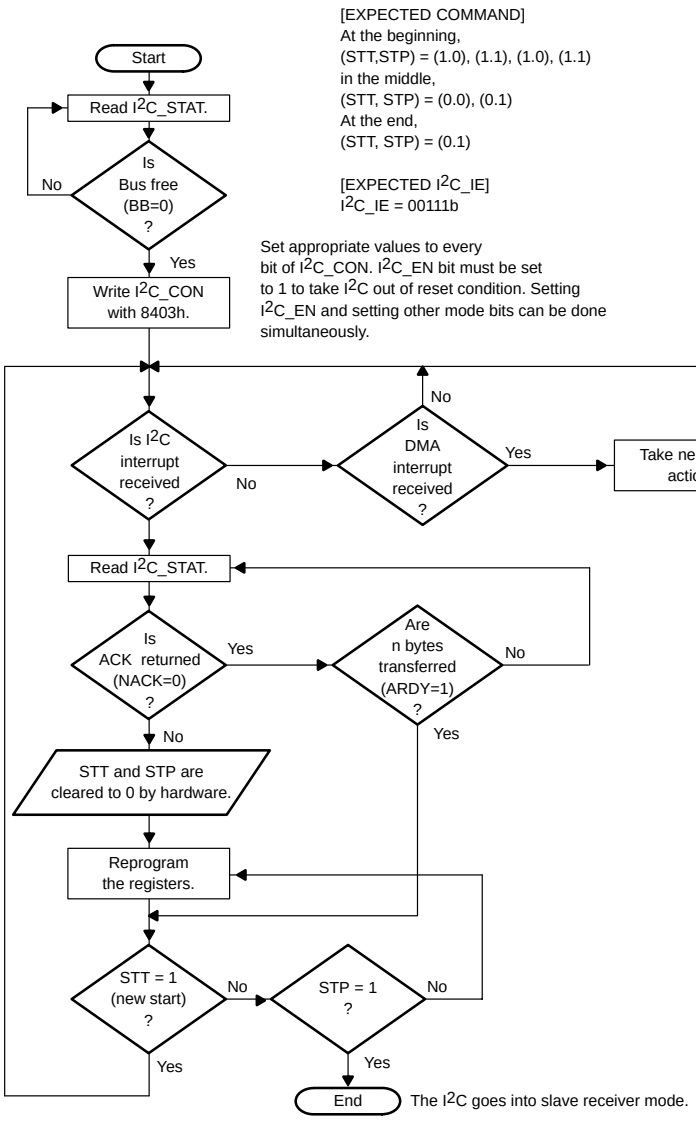


Figure 7–41. Slave Transmitter/Receiver Mode, Polling

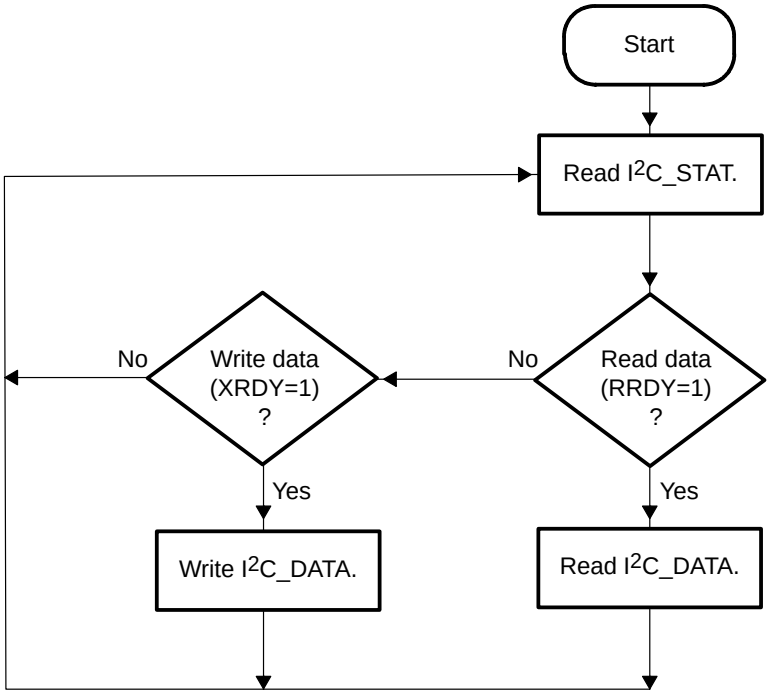
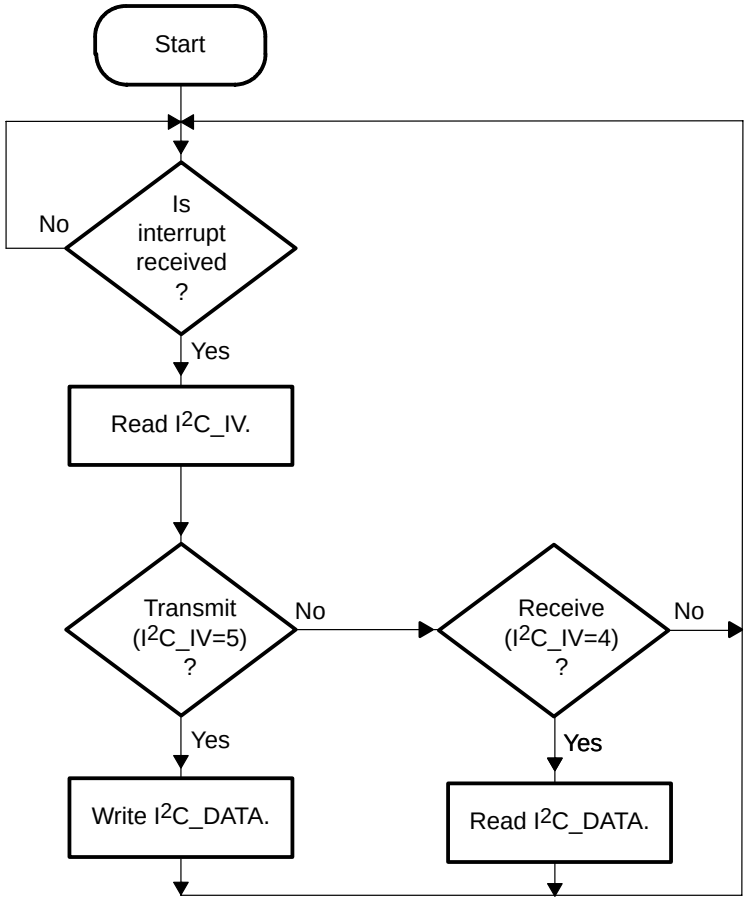


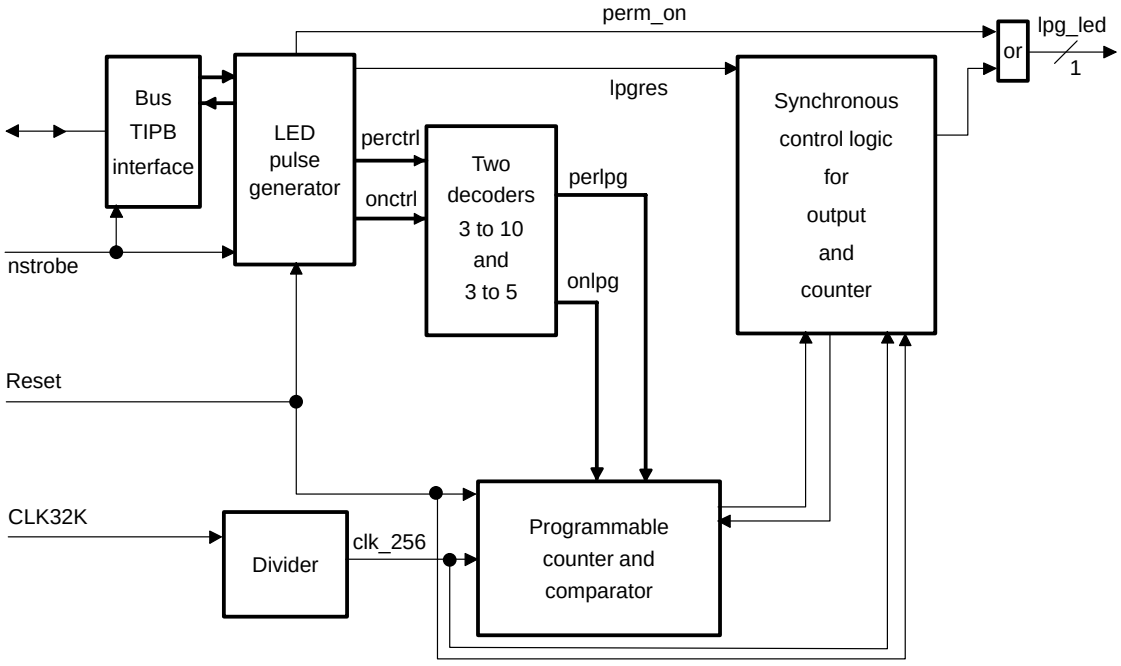
Figure 7–42. Slave Transmitter/Receiver Mode, Interrupt



7.9 LED Pulse Generator

The LED pulse generator (LPG) module controls an indication LED (see Figure 7–43). The blinking period is programmable between 152 ms and 4s, and the LED can be switched on permanently. The OMAP5910 device has two LPG modules. Each LPG module drives a single output pin on the OMAP5910 device which can be used to switch an LED driver.

Figure 7–43. LED Pulse Generator Block Diagram



7.9.1 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

7.9.2 LPG Design

LCR bit 6 = 0 resets the whole pulse generator circuit (but not the control register) and switches off the LED. It is possible to switch on the LED independently from the pulse generator circuit with bit 7 of the LCR (1 = permanent light). A device reset causes a reset to the whole LPG (with the control register) and the output LPG_LED to zero asynchronously.

Because the TIPB write to the LPG control register is asynchronous, the value written to the control register may be unstable for one blink period. Consequently, the LED output could, in the worst case, be switched on at maximum intensity during one additional blink period.

7.9.3 LPG Power Management

The LPG input clock comes from the 32-kHz ULPD clock, because it must work even when the OMAP5910 system is in deep sleep mode. The internal clock of the LPG runs with 256 Hz. For this reason the power consumption of this block can be neglected. Nevertheless, switch the LPG_CLK off if LPG is not used.

7.9.4 LPG Registers

Both receive and transmit registers are mapped in the MPU address space.

Two instances of LPG are mapped in the OMAP5910 device:

- First LPG: LPG_1 address is FFFB:D000
- Second LPG: LPG_2 address is FFFB:D800

Table 7–73 lists the LPG receive and transmit registers. Table 7–74 and Table 7–77 describe the register bits.

Table 7–73. 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 7–74. LPG Control Register (LCR)

Bit	Name	Function	R/W	Reset Value
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

With the LCR bits 2-0, the blinking period of the LED is determined.

Table 7–75. LED Blinking Period

LCR Bit 2	LCR Bit 1	LCR Bit 0	Period of LED	Number 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

With the LCR bits 5-3, the on time of the LED is determined.

Table 7–76. LED On Time

LCR Bit 5	LCR Bit 4	LCR Bit 3	Time LED On	Number 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 7–77. Power Management Register (PMR)

Bit	Name	Value	Function	R/W	Reset Value
0	CLK_EN		Functional clock enable:	R/W	0
		0	Clock disabled		
		1	Clock enabled		
			Asynchronous writing and reading		

7.10 McBSP2

Multichannel buffered serial ports (McBSPs) are configurable, high-speed, full-duplex serial ports that allow direct interfacing to external communication devices. There are three McBSPs on OMAP5910. McBSP2 is on the MPU public peripheral bus and is covered briefly in this section. McBSP1 and McBSP3 are on the DSP public peripheral bus and are covered briefly in Chapter 9, *DSP Public Peripherals*. For more detail on the functions of all three McBSPs, see the *TMS320C55x DSP Peripherals Reference Guide* (literature number SPRU317).

Key features of McBSP2 include:

- Full-duplex communication
- DMA support for both RX and TX transfers
- Double-buffered data registers, which allow a continuous data stream
- Independent framing and clocking for receives and transmits
- External shift clock generation or an internal programmable frequency shift clock
- Multichannel transmits and receives of up to 128 channels.
- A wide selection of data sizes, including 8-, 12-, 16-, 20-, 24-, or 32-bits
- μ-Law and A-Law companding
- Data transfers with LSB or MSB first
- Programmable polarity for both frame synchronization and data clocks
- Highly programmable internal clock and frame generation
- RX and TX interrupts as well as RX data overrun interrupt

The operation of the three OMAP5910 McBSPs is consistent with SPRU317 with the following exceptions and clarifications:

- Only DXENA = 0 setting is supported
- The transmit output (DX) pins don not go to high-impedance state when the transmitter is not actively sending data. In other words, the OMAP5910 always actively drives the DX pins.
- The CLKS input is only available on McBSP1.
- On McBSP1 and McBSP3, the receiver can only operate in slave mode.
- BIS is not supported.

Table 7–78 describes the McBSP2 pins. Table 7–79 lists the McBSP2 registers. Figure 7–44 shows the McBSP2 interface.

Table 7–78. McBSP2 Pin Descriptions

Pin	I/O Direction	Description
MCBSP2.CLKR	In/out	Receive clock
MCBSP2.CLKX	In/out	Transmit clock
MCBSP2.DR	In	Data input
MCBSP2.DX	Out	Data output
MCBSP2.FSR	In/out	Receive frame synchronization
MCBSP2.FSX	In/out	Transmit frame synchronization

The McBSP2 base address is FFFB:1000 (MPU memory map).

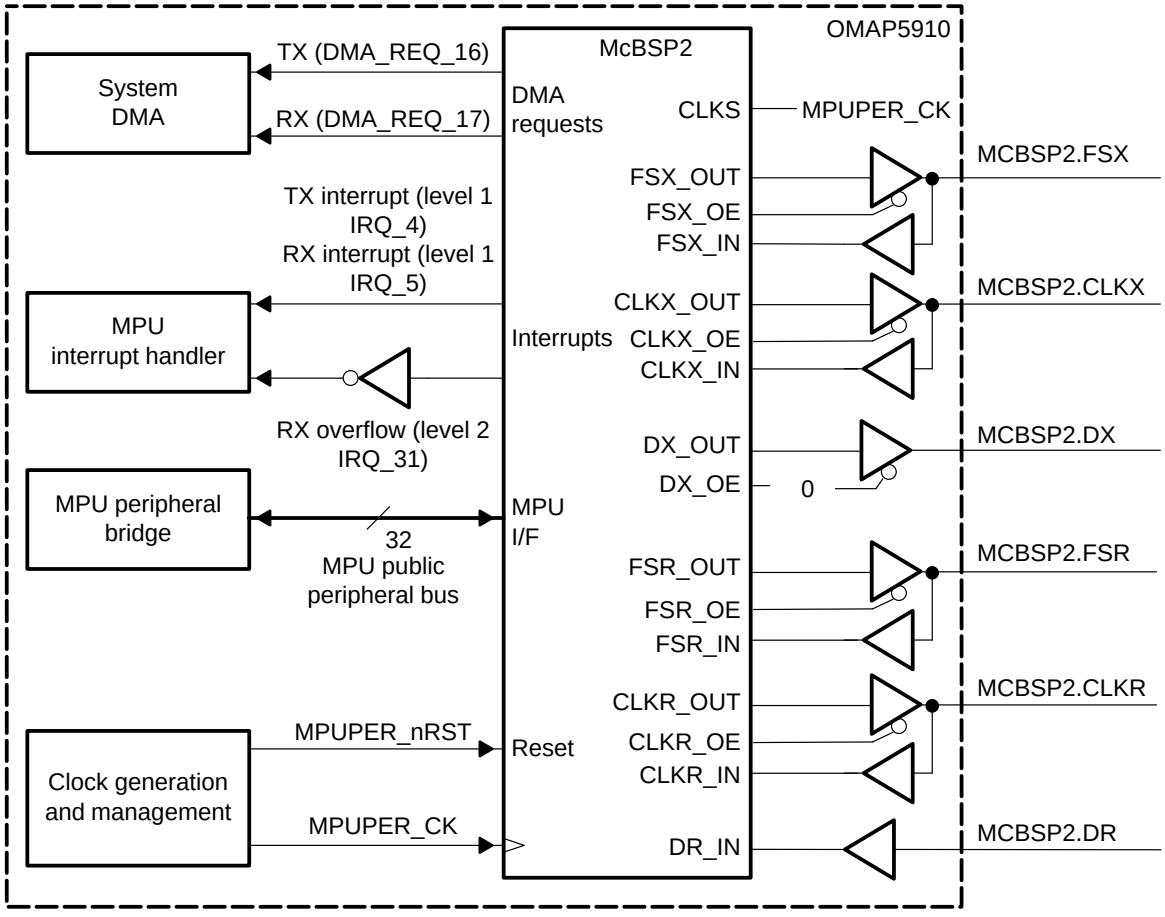
Table 7–79. McBSP2 Registers

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

Table 7–79. McBSP2 Registers (Continued)

Name	Description	Offset
RCERA (15:0)	Receive channel enable register partition A	0x1C
RCERB (15:0)	Receive channel enable register partition B	0x1E
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

Figure 7–44. McBSP2 Interface Diagram

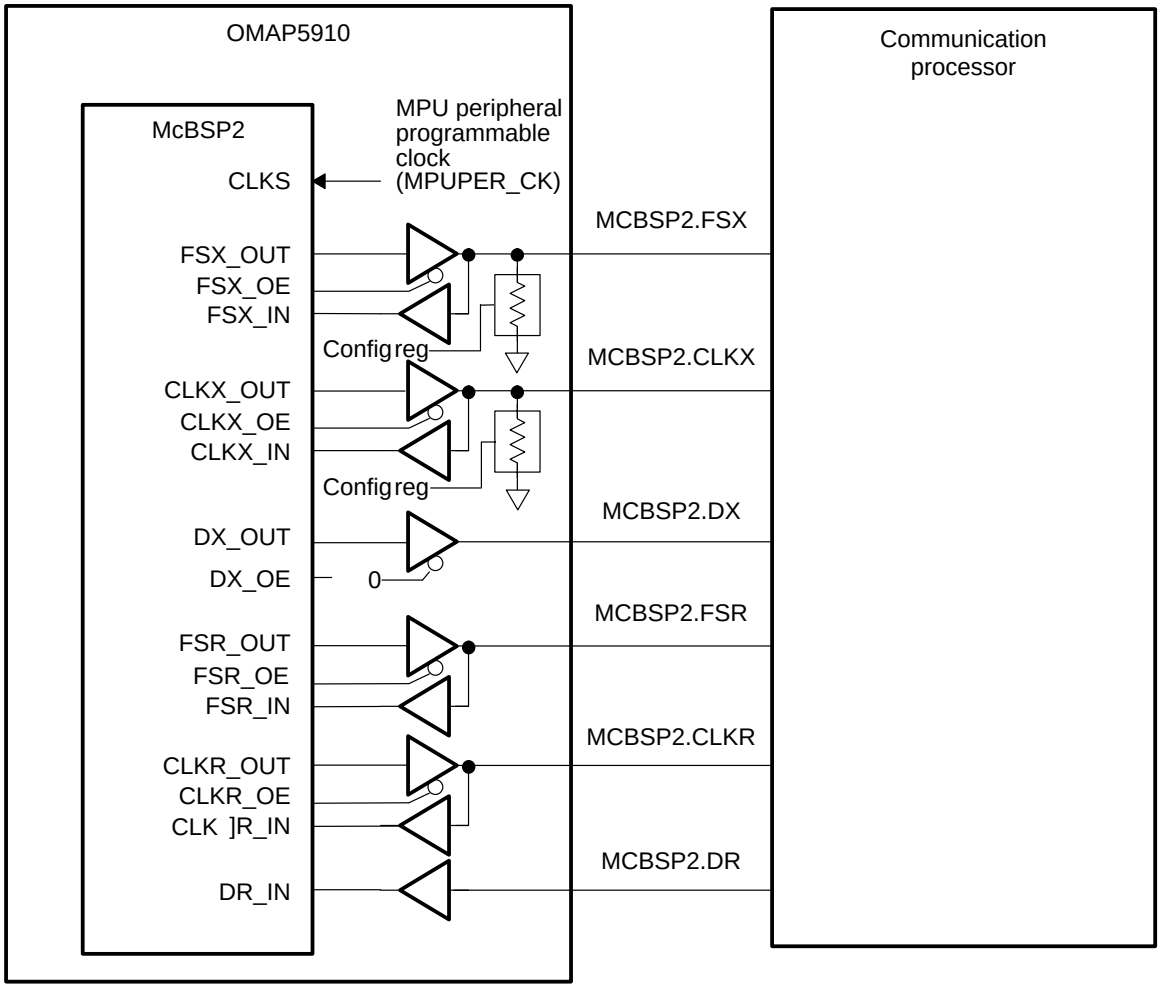


Note: You can use the AUXON feature to gate the functional clock to the McBSP2 module by setting MOD_CONF_CTRL_0[19] to 1.

7.10.1 McBSP2 Application Example: Communication Interface

Figure 7–45 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 set up and write to the McBSP.

Figure 7–45. Communication Processor Data Interface



Section 7.10.1.1 through Section 7.10.1.9 explain how to set up the McBSP registers for TX master and RX slave mode with 16-bit transfers using interrupts.

7.10.1.1 Serial Port Control Register Configuration

ARM_Write(0x0000) => SPCR1; set up SPCR1 as initial configuration.

This setup is not needed after reset.

ARM_Write(0x0000) => SPCR2; set up SPCR2 as initial configuration.

This setup is not needed after reset.

7.10.1.2 Pin Control Register Configuration

ARM_Write(0x0a00) => PCR; set up PCR per the following configuration.

Table 7–80. Pin Control Register Configuration

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 OMAP5910 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

7.10.1.3 Receive Control Register Configuration

ARM_Write(0x0040) => RCR1; set up RCR1 per below configuration.

Table 7–81. Receive Control Register 1 Configuration

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

Table 7–82. Receive Control Register 2 Configuration (ARM_Write(0x0001) = > 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

7.10.1.4 Transmit Control Register Configuration

ARM_Write(0x0040) => XCR1; set up XCR1 per below configuration.

Table 7–83. Transmit Control Register 1 Configuration

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 below configuration.

Table 7–84. Transmit Control Register 2 Configuration (ARM_Write(0x0001) => 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

7.10.1.5 Sample Rate Generator Configuration (SRGR[1,2])

1) Configure the sample rate generator appropriately for CLKX and FSX. For details, see *TMS320C54x DSP Enhanced Peripherals Reference Set*, vol. 5, SPRA302.
2) Wait for two CLKSRG clocks.
3) ARM_Write SPCR2 or (0x0000 0040) → SPCR2;CLKG enable
4) Wait two CLKG clocks.

7.10.1.6 Interrupt Flag Configuration and Clear (ILR, ITR, MIR)

1) ARM_Write → ILR; set ILR appropriately for the interrupt handling priority.
2) ARM_Write ITR and (0xFFFF FFCF) → ITR; clear remained TX and RX interrupt.

Note:

This setup is not needed after reset.

3) ARM_Write MIR and (0xFFFF FFCF) → MIR; enabled SPI TX and RX interrupt

7.10.1.7 Take out of Reset for Transmit and Receive Starting (SPCR[1,2])

1) ARM_write SPCR1 or (0x0001) → SPCR1; enabled receive port
2) ARM_write SPCR2 or (0x0001) → SPCR2; enabled transmit port

7.10.1.8 Transmit Data Loading (TX_INT Handling in Interrupt Survive Routine)

ARM_Write → DXR

Note:

Clear interrupts flag on ITR, when taken the interrupt handle.

7.10.1.9 Received Data Loading (RX_INT Handling in Interrupt Survive Routine)

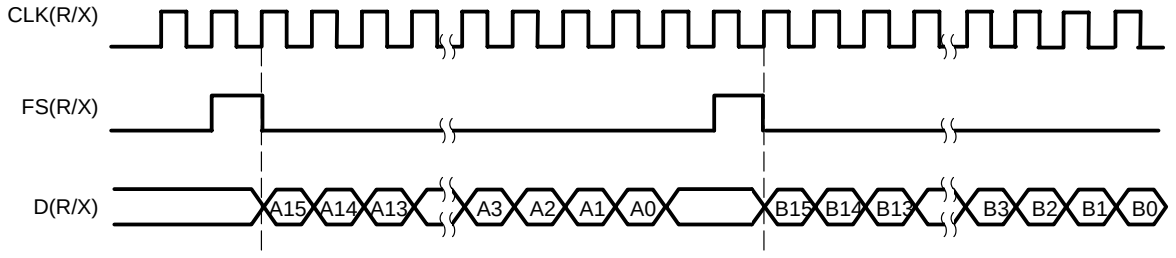
ARM_Read ← DRR

Note:

Clear interrupts flag on ITR, when taken the interrupt handle.

Waveform Example

Figure 7–46. Waveform Example



Section 7.10.1.10 through Section 7.10.1.18 explain how to set up the McBSP registers for TX master and RX slave mode with 16-bit transfers using DMA support.

7.10.1.10 Serial Port Control Register Configuration

ARM_Write(0x0000) => SPCR1; set up SPCR1 as initial configuration.

This setup is not needed after reset.

ARM_Write(0x0000) => SPCR2; set up SPCR2 as initial configuration.

This setup is not needed after reset.

7.10.1.11 Pin Control Register Configuration

ARM_Write(0x0a00) => PCR; set up PCR per below configuration.

Table 7–85. Pin Control Register Configuration

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 OMAP5910)
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

7.10.1.12 Receive Control Register Configuration

ARM_Write(0x0040) => RCR1; set up RCR1 per below configuration.

Table 7–86. Receive Control Register 1 Configuration

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 bits per frame
4–0	0 0000b	Reserved

ARM_Write(0x0001) => RCR2; set up RCR2 per below configuration.

Table 7–87. Receive Control Register 2 Configuration

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	Don’t care for single-phase 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

7.10.1.13 Transmit Control Register Configuration

ARM_Write(0x0040) => XCR1; set up XCR1 per below configuration.

Table 7–88. Transmit Control Register 1 Configuration

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 bits per frame
4–0	0 0000b	Reserved

ARM_Write(0x0001) => XCR2; set up XCR2 per below configuration.

Table 7–89. Transmit Control Register 2 Configuration

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

7.10.1.14 Sample Rate Generator Configuration (SRGR[1,2])

1) Configure the sample rate generator appropriately for CLKX and FSX. For details, see *TMS320C54x DSP Enhanced Peripherals Reference Set*, vol. 5, SPRA302.
2) Wait two CLKSRG clocks.
3) ARM_Write SPCR2 or (0x0000 0040)=>SPCR2;CLKG enable.
4) Wait two CLKG clocks.

7.10.1.15 DMA Configuration

Configure the REVT and XEVT bit for the DMA receive and transmit synchronized invent.

7.10.1.16 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 0030) => MIR ; disabled SPI TX and RX interrupt

Note:

Enable the appropriate DMA channel interrupts.

7.10.1.17 Take out of Reset for Transmit and Receive Starting (SPCR[1,2])

1) ARM_write SPCR1 or (0x0001) => SPCR1; enabled receive port.
2) ARM_write SPCR2 or (0x0001) => SPCR2; enabled transmit port.

7.10.1.18 Data Transfer (DMA Channel)

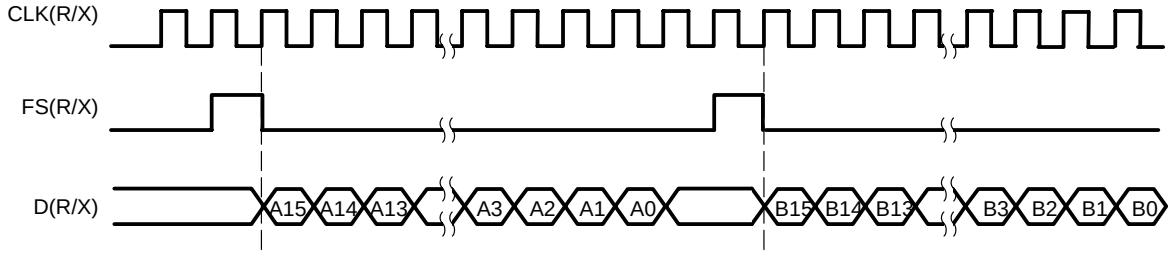
The DMA channel transfers the received data to appropriate data buffer and transfer the new transmit data to appropriate TX buffer. Clear interrupts flag on ITR, when taking the interrupt handle.

Note:

Clear interrupts flag on ITR, when taken the interrupt handle.

Waveform Example

Figure 7–47. Waveform Example



7.11 USB Function Overview

The universal serial bus (USB) function module supports the implementation of a full-speed device fully compliant with the USB 1.1 standard (see Figure 13–2). It provides an interface between the MPU core (TI925T) and the USB wire and handles USB transactions with minimal TI925T intervention.

The 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 module also supports three DMA channels for IN endpoints and three DMA channels for OUT endpoints for either bulk/interrupt or ISO transactions. For more detail, see Chapter 13, *USB Function Module*.

The MPU base address is FFFB:4000.

Table 7–90. USB Function Registers

Name	Description	Offset Address
REV	Revision number read	0x00
	Endpoint	
EP_NUM	Selects and enables the endpoint that can be accessed by the TI925T	0x04
DATA	The entry point to write into a selected TX endpoint, to read data from a selected RX endpoint, or to read data from setup FIFO	0x08
CTRL	Controls the FIFO and status of the selected endpoint	0x0C
STAT_FLG	Provides a status of the FIFO and the results of the transactions handshakes for the selected endpoint	0x10
RXFSTAT	The number of bytes which are in the receive FIFO for the selected endpoint	0x14
SYSCON1	Control functions for power management and miscellaneous control for the core	0x18
SYSCON2	Miscellaneous controls for the function	0x1C
DEVSTAT	Status reflecting the visible device states as defined in Section 13.6.6, *Device States Changed Handler*	0x20
SOF	Provides a frame timer status for use in ISO communications	0x24
IRQ_EN	Enables all non-DMA interrupts	0x28

Table 7–90. USB Function Registers (Continued)

Name	Description	Offset Address
DMA_IRQ_EN	Enables all DMA interrupts	0x2C
IRQ_SRC	Identify and clear the source of the interrupt signaled by a set flag	0x30
EPN_STAT	Identify the non-ISO endpoint causing an EPn interrupt	0x34
DMAN_STAT	Identify the endpoint causing a $\overline{\text{DMA}}$ interrupt	0x38
RESERVED		0x3C
	DMA Configuration	
RXDMA_CFG	Enables the three possible DMA receive channels and selects the endpoint number that is assigned to each of these DMA channels	0x40
TXDMA_CFG	Enables the three possible DMA transmit channels and selects the endpoint number that is assigned to each of these DMA channels	0x44
	DMA	
DATA_DMA	Entry point to write or to read data into/from an endpoint used in a DMA transfer through DMA channel 0, 1, or 2	0x48
Reserved		0x4C
TXDMA0	Controls the operation of the transmit DMA channel 0	0x50
TXDMA1	Controls the operation of the transmit DMA channel 1	0x54
TXDMA2	Controls the operation of the transmit DMA channel 2	0x58
Reserved		0x5C
RXDMA0	Permits monitoring of incoming OUT transactions during DMA transfer on channel 0	0x60
RXDMA1	Permits monitoring of incoming OUT transactions during DMA transfer on channel 1	0x64
RXDMA2	Permits monitoring of incoming OUT transactions during DMA transfer on channel 2	0x68
Reserved		0x7C

Table 7–90. USB Function Registers (Continued)

Name	Description	Offset Address
	Endpoint Configuration	
EP0	Gives the device configuration for control endpoint 0	
EP1_RX	Gives the device configuration for non-control receive endpoint 1	0x84
EP2_RX	Gives the device configuration for non-control receive endpoint 2	0x88
…	…	…
EP15_RX	Gives the device configuration for non-control receive endpoint 15	0xBC
Reserved		0xC0
EP1_TX	Gives the device configuration for non-control transmit endpoint 1	0xC4
EP2_TX	Gives the device configuration for non-control transmit endpoint 2	0xC8
…	…	…
EP15_TX	Gives the device configuration for non-control transmit endpoint 15	0xFC

7.12 MMC/SD Host Controller

The MMC/SD host controller provides an interface between the TI925T and either MMC or SD memory card plus up to three serial flash cards and handles MMC/SD or SPI transactions with minimal TI925T intervention. All references to a local host in this section refer to the TI925T MPU processor.

The following combination of external devices is supported:

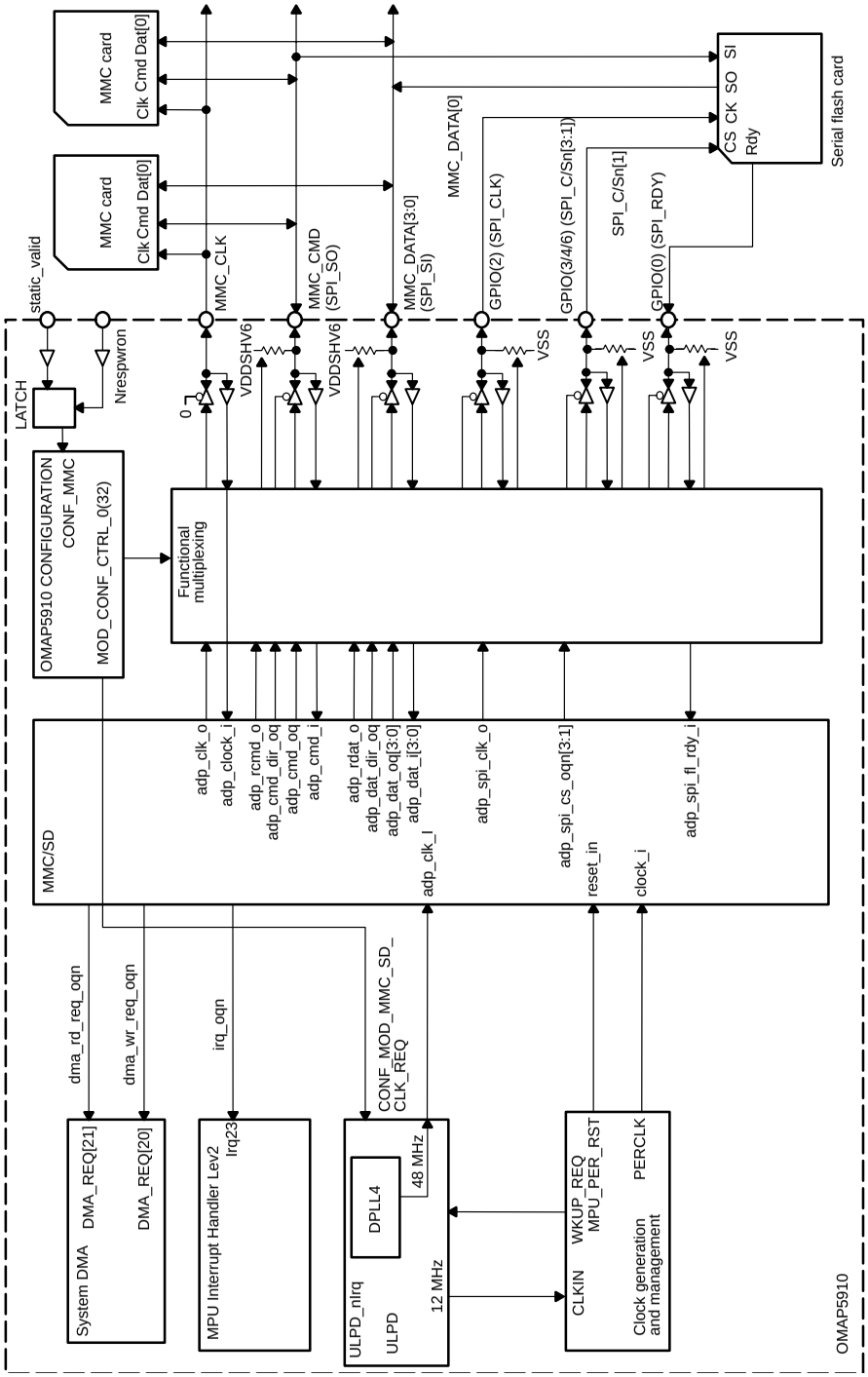
- One or more MMC memory cards sharing the same bus plus up to three devices with 8-bit SPI protocol interface (flash serial memory).
- One single-SD memory card plus up to three devices with 8-bit SPI protocol interface.

Other combinations like two SD cards, one MMC card + one SD card are not supported.

The application interface is responsible for managing transaction semantics. The MMC/SD host controller handles MMC/SD protocol at transmission level, packing data, adding cyclic redundancy check (CRC), start/end bit, and checking for syntactical correctness. SD mode wide bus width is also supported.

The application interface can send every MMC/SD command and either poll for the status of the adapter or wait for an interrupt request, which is sent back in case of exceptions or to warn for end of operations. The application interface can read card responses or flag registers. It can also mask individually interrupt sources. All these operations can be performed by reading and writing control registers.

Figure 7–48. MMC/SD Host Controller Environment



7.12.1 MMC/SD Host Controller Features

Main features of the controller are:

- Full compliance with MMC command/response sets as defined in the MMC standard specification [1]
- Full compliance with SD command/response sets as defined in the SD Physical Layer specification [2]
- Flexible architecture allowing support for new command structure
- Separate SPI interface with 3 C/S. Provides supports for up to three serial devices such as serial flash
- Built-in 64-byte FIFO for buffered read or write
- 16-bit-wide accesses bus to maximize bus throughput
- Designed for low power
- Wide interrupt capability
- Programmable clock generation
- Two DMA channels

Known limitations:

- No built-in hardware support for error correction codes (ECC)
- No built-in support for card detection
- No full compliance to SDIO specification.

7.12.2 MMC/SD Host Controller Signals Pads

The signal pads, listed in Table 7–91, describe the physical interface between OMAP5910—the transceiver—and the target MMC/SD memory card(s) or serial flash memories.

The transceiver provides dc-level adaptation functions between OMAP5910 and the target devices.

The state of the OMAP5910 static_valid input during power on, determines the functional multiplexing on the OMAP5910 MMC/SD pads.

The OMAP5910 static_valid pad must be held to 1 during power on so that the MMC/SD host controller signals are usable from the power-on reset on the OMAP5910 MMC/SD pads (described in Table 7–91).

This functional multiplexing, which is configured in static, does not concern the SPI signals, which are multiplexed on the GPIO pads. For these pads, the functional multiplexing is done classically by programming an OMAP5910 configuration register.

Table 7–91. MMC/SD Signal Pads

Pad Name	Type	Pullup/ Pulldown	Reset Value	Description
MMC.CLK	Out	-	0	MMC/SD card CLK signal. Only active during active command to a MMC/SD card using MMC or SPI protocols.
MMC.CMD_SPI. DO/SPI_SO	In-Out	Pullup (*3)	Input	MMC/SD card CMD signal in MMC/SD mode. SPI serial out signal in SPI mode (output—goes to serial In of target device(s)).
MMC.DAT[0] /SPI_SI	In-Out	Pullup (*3)	Input	MMC card DAT or SD card DAT[0] signal in MMC/SD mode. SPI serial in signal in SPI mode (input—comes from serial out of target device(s)).
MMC.DAT[3-1] (*1)	In-Out	Pullup (*3)	Input	SD card DAT[3-1] signals in MMC/SD mode. Reserved signals for MMC card or in SPI mode.
GPIO2 (SPI.CLK) (*2)	In- Out (Out)	Pulldown (disabled)	Input (0)	By default, pad used by the GPIO2. The SPI_CLK output signal can be multiplexed. SPI_CLK only active during SPI transfers to serial flash or other SPI devices (except MMC/SD cards).
GPIO3 (SPI.C/Sn[3]) (*2)	In-Out (Out)	Pulldown (disabled)	Input (1)	By default, pad used by the GPIO3. The SPI CSn(3) output signal can be multiplexed. SPI CSn(3) is active low, only active in SPI mode during SPI transfers. Reserved in MMC/SD mode.
GPIO4 (SPI.C/Sn[2]) (*2)	In-Out (Out)	Pulldown (disabled)	Input (1)	By default, pad used by the GPIO4. The SPI CSn(2) output signal can be multiplexed. SPI CSn(2) is active low, only active in SPI mode during SPI transfers. Reserved in MMC/SD mode.
GPIO6 (SPI.C/Sn[1]) (*2)	In-Out (Out)	Pulldown (disabled)	Input (1)	By default, pad used by the GPIO6. The SPI CSn(1) output signal can be multiplexed. SPI CSn(1) is active low, only active in SPI mode during SPI transfers. Reserved in MMC/SD mode.
GPIO0 (SPI.RDY) (*2)	In-Out (In)	Pulldown (disabled)	Input (Input)	By default, pad used by the GPIO0. The SPI ready/busy input can be multiplexed. When SPI_RDY is low, it denotes busy condition. Only active in SPI mode during SPI transfers. Reserved signal in MMC/SD mode

Notes: 1) Optional signals. Only needed for SD cards.

2) Optional signals. Only needed for devices with SPI interfaces (serial flash, etc.).

3) This pullup is enabled/disabled dynamically by the MMC/SD host controller.

7.12.3 MMC/SD Host Controller Clocks and Reset

The MMC/SD host controller has two clocks:

- An interface clock (clock_i) used between the MPU TIPB and the MMC/SD host controller and connected to the MPU peripheral programmable clock (PERCLK), is determined dividing CK_GEN1 (the output of DPLL1) by the value associated with the PERDIV field of the ARM_CKCTL register (0xFFFECE00).

  This clock is a free-running clock when the system is awake.

- A 48-MHz functional clock (ADP_CLK_I), which is generated by the ULPD DPLL.

This clock is requested by setting to 1 the CONF_MOD_MMC_SD_CLK_REQ bit(23) of the MOD_CONF_CTRL_0 register.

The MPU TIPB reset (MPU_PER_RST) resets the MMC/SD host controller.

7.12.4 MMC/SD Host Controller DMA Request

The MMC/SD host controller can use:

- Receive DMA channel (DMA_RD_REQ_OQN), which is connected to the SYSTEM DMA request [21].

- Transmit DMA channel (DMA_WR_REQ_OQN), which is connected to the SYSTEM DMA request [20].

See Section 13.1.4 DMA for more details.

7.12.5 MMC/SD Host Controller Interrupt

The MMC/SD controller can generate one interrupt (IRQ_OQN), which is connected to the MPU level 2 interrupt handler, line 23 (level-sensitive).

7.12.6 MMC/SD Internal Pullups

There are internal pullups on the following pins:

- MMC.CMD I/O pin
- MMC.DAT[3:0] I/O pins

MMC cards work in open drain mode on the MMC.CMD line during the identification phase, and more generally for broadcast MMC commands; consequently, a pullup on the MMC.CMD line is needed.

When MMC.CMD and MMC.DAT[3:0] line work in push/pull mode, it is important to prevent bus floating conditions. Consequently, pullups are needed.

These pullups are directly controlled by the MMC/SD host controller (adp_rcmd_o and adp_rdat_o) and are only active when required, which saves power.

Table 7–92 and Table 7–93 show activation conditions for the MMC.CMD and MMC.DAT pullups.

Table 7–92. MMC_CMD Pullups

MMC_SD Host Controller Status	MMC CARD Status	MMC_CMD Pullup (Open Drain Mode)	MMC_CMD Pullup (Push/Pull Mode)
Input	Input	Active	Active
Input	Output	Active	Disabled
Output	Input	Disabled	Disabled

Table 7–93. MMC_DAT Pullups

MMC_SD Host Controller Status	MMC CARD Status	MMC_DAT Pullup (Both Modes)
Input	Input	Active
Input	Output	Disabled
Output	Input	Disabled

In flash-SPI mode, when no data is on the MMC.CMD and MMC.DAT lines (input and output of the flash), the pullups are disabled.

7.12.7 MMC/SD Registers

Table 7–94 lists the MMC/SD controller registers. Table 7–95 through Table 7–122 describe the register bits.

Table 7–94. MMC/SD Registers

Register	Description	Access	Address
MMC_CMD	MMC command	R/W	FFFB:7800
MMC_ARGL	MMC argument low	R/W	FFFB:7804
MMC_ARGH	MMC argument high	R/W	FFFB:7808
MMC_CON	MMC system configuration	R/W	FFFB:780C
MMC_STAT	MMC status	R/W	FFFB:7810
MMC_IE	MMC system interrupt enable	R/W	FFFB:7814
MMC_CTO	MMC command time-out	R/W	FFFB:7818
MMC_DTO	MMC data time-out	R/W	FFFB:781C
MMC_DATA	MMC TX/RX FIFO data	R/W	FFFB:7820
MMC_BLEN	MMC block length	R/W	FFFB:7824
MMC_NBLK	MMC number of blocks	R/W	FFFB:7828
MMC_BUF	MMC buffer configuration	R/W	FFFB:782C
MMC_SPI	MMC serial port interface	R/W	FFFB:7830
MMC_SDIO	MMC SDIO mode configuration	R/W	FFFB:7834
MMC_SYST	MMC system test	R/W	FFFB:7838
MMC_REV	MMC module version	R	FFFB:783C
MMC_RSP0	MMC command response 0	R	FFFB:7840
MMC_RSP1	MMC command response 1	R	FFFB:7844
MMC_RSP2	MMC command response 2	R	FFFB:7848
MMC_RSP3	MMC command response 3	R	FFFB:784C
MMC_RSP4	MMC command response 4	R	FFFB:7850
MMC_RSP5	MMC command response 5	R	FFFB:7854
MMC_RSP6	MMC command response 6	R	FFFB:7858
MMC_RSP7	MMC command response 7	R	FFFB:785C
Reserved			FFFB:7860- FFFB:787C

Table 7–95. MMC Command Register (MMC_CMD)

Bit	Name	Description
15	DDir	Data direction [read/write]
14	SHR	Stream command or broadcast host response
13–12	Type	Command types [bc,bcr,ac,adtc]
11	Busy	Command with busy response [R1b]
10–8	Response	Command responses [no response, R1/R1b, R2, R3, R4, R5,R6]
7	Init	Send initialization stream
6	OD	Card open drain mode
5–0	Cmd_Index	Command index [63:0]

A write to the MMC command register (MMC_CMD) sends a command to the card. If the local host accesses this register byte-wise, the command is sent to the card only after a write access to the least significant LSB (bits 7:0). Hence, the MSB must always be written first in a byte-accessed situation.

A read has no effect except to return the last command that was previously sent.

Note:

A write into this register with Type = adtc resets the FIFO pointers and pre-fetch register. Writes with other type values (bc, bcr, ac) do not affect the FIFO contents. Hence, data must be written inside the FIFO after sending a single or multiple block write command.

A write into this register also clears the MMC_RSP[07] registers.

Data Direction (DDir)

This bit (15) specifies if the data transfer is a read or a write. This bit is only valid if the command type is adtc.

This bit has the same polarity as RD/WR argument bit 0 for a GEN_CMD command (CMD56).

- ❏ 0: Data write
- ❏ 1: Data read

Value after reset is low.

Stream Command or Broadcast Host Response (SHR)

MMC card only. SD card does not support stream operation or host generated response.

This bit (14) must be set to 1 in two cases:

- Associated with adtc 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 bc 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 Interrupt Mode section 4.3 in MMC [1] specification).

This bit is only valid if the command type is adtc or bc.

- 0: Normal mode
- 1: Stream mode (type = adtc), host response (type = bc)

Value after reset is low.

Command Type (Type)

Encoded bits (13-12) that define the type of the command that is passed by the core to the MMC/SD memory card (see command types Section 4.7.1 in MMC [1] or SD [2] specifications).

- 00: bc (broadcast—no response)
- 01: bcr (broadcast with response)
- 10: ac (addressed—no data transfer)
- 11: adtc (addressed with data transfer)

Note:

Also resets the FIFO.

Values after reset are low (two bits).

Command With Busy Response (Busy)

This bit (11) must be set to 1 if the response to the command sent is of type R1b (R1 + busy).

- ❏ 0: Response without busy (R1, R2, R3, R4, R5, R6)
- ❏ 1: Response with busy (R1b)

Value after reset is low.

Command Response (Response)

Encoded bits (10-8) that define the response for the command passed by the core to the MMC/SD memory card (see Responses section 4.9 in MMC [1] or SD [2] specifications).

- ❏ 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)
- ❏ 110: R6 (Published RCA response—SD card only)
- ❏ 111: Reserved

Values after reset are low (three bits).

Send Initialization Stream (Init)

When this bit (7) is set, an initialization sequence is sent prior to the command. This option can simplify acquisition of new cards. An initialization sequence consists of setting CMD line to 1 during 80 CLK cycles (see *Power-Up Description* Section 6.3—MMC spec [1], or Section 6.4—SD spec [2]).

- ❏ 0: No initialization sequence (normal procedure)
- ❏ 1: Initialization sequence send prior to command

Value after reset is low.

Card Open Drain Mode (OD)

This bit (6) 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, during card identification mode, the card is either in idle, ready or identification state. This bit must be set for MMC card commands 1, 2, 3, and 40.

For SD card, this bit must always be kept low, because SD cards do not have open drain capability.

- 0: Push/pull
- 1: Open drain

Value after reset is low.

Command Index (Cmd_index)

Binary encoded value (bits 5-0) from 0 to 63 specifying the command number sent to the card.

- 000000: CMD0
- 000001: CMD1
- …
- 111111: CMD63

Values after reset are low (all 6 bits).

The MMC argument low and high registers specify the 32-bit argument value that is passed with the command. These registers must be initialized prior to sending the command itself to the card (write action into the MMC_CMD register). The only exception is for a command index specifying stuff bits in arguments, which makes a write unnecessary.

Table 7–96. MMC Argument Low Register (MMC_ARGL)

Bit	Name	Description
15–0	ARG_low	Command argument bits [15:0]

Values after reset are low (all 16 bits).

Table 7–97. MMC Argument High Register (MMC_ARGH)

Bit	Name	Description
15–0	ARG_high	Command argument bits [31:16]

Values after reset are low (all 16 bits).

Table 7–98. MMC System Configuration Register (MMC_CON)

Bit	Name	Description
15	DW	Data bus width
14	Reserved	
13–12	Mode	Operating mode select (MMC/SD, SPI, SYSTEST, or MMC SPI protocol).
11	Power-up	Power-up control
10–8	Reserved	
7–0	Clk_div	Clock divider [No clock, 1:255]

Bus Width During Data Phase (DW)

SD card only.

This bit (15) must be set following a valid SET_BUS_WIDTH command (ACMD6) with the value written in bit [1] of the argument. Prior to this command, the SD card configuration register (SCR) must be verified for the supported bus width by the SD card.

- 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. Not setting this bit correctly can result in an unpredictable behavior.

Mode Select (Mode)

These bits (13-12) select between MMC/SD mode, SPI mode 1, SYSTEST mode and SPI mode 2.

In MMC/SD mode, transfers to the MMC/SD card follow the MMC protocol. MMC clock is enabled and the SPI clock is disabled.

In SPI mode1, transfers to up to three SPI controlled devices (serial flash, etc.) are supported. In this mode, SPI clock is enabled and MMC clock is disabled.

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. The pins retain their default type (input, output or in/out).

In SPI mode 2, transfers to the MMC/SD card follow the SPI protocol. MMC clock is enabled and the SPI clock is disabled. MMC protocol must be implemented in software when using this mode since the MMC interface acts as a generic SPI port and does not utilize the MMC-specific features available in MMC/SD mode.

- 00: MMC/SD mode (MMC/SD cards using MMC protocol)
- 01: SPI mode 1 (for serial flash or others SPI slave devices)
- 10: SYSTEST mode
- 11: SPI mode 2 (MMC/SD cards using SPI protocol)

Values after reset are low (2 bits).

Power Up-Control (Power_Up)

This bit (11) must be set to 1 prior to 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 dependant) and the controller core logic in pseudoreset state. That is, the MMC_STAT register flags are reset, the FIFO pointers are reset, any access to DATA register has no effect, a write into MMC_CMD register is ignored, and setting of MMC_SPI:start to 1 is ignored.
  - 0: Powered-down/pseudoreset state
  - 1: Powered-up/normal operation mode

Value after reset is low.

Clock Divider (Clk_div)

These bits (7-0) define the ratio between a reference clock frequency (48 MHz) 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.

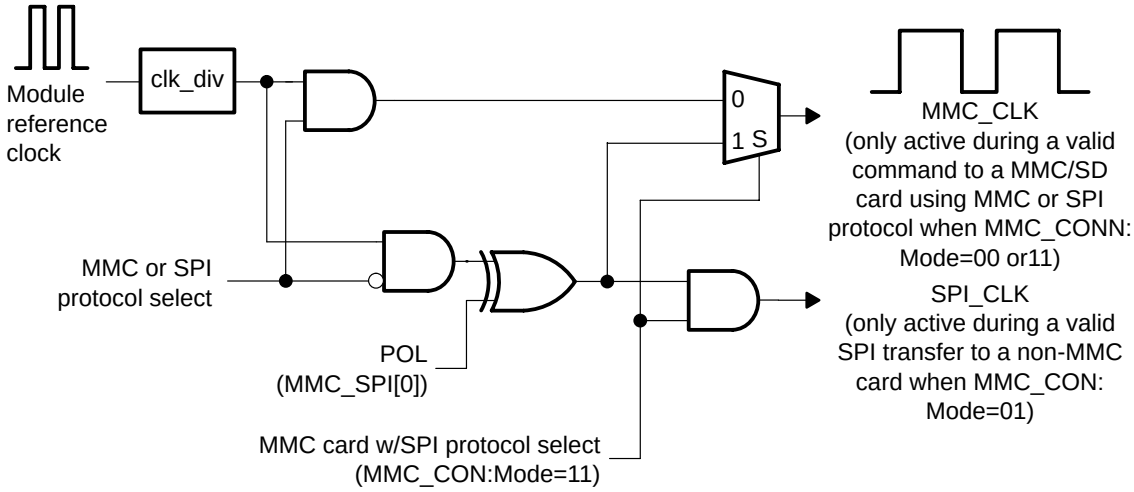
The division factor is exactly the binary encoded decimal value for values between 1 and 255.

A value of 0 disables the clock.

- ❏ 0x00: Clock disabled
- ❏ 0x01: Ref clk/1
- ❏ ....
- ❏ 0xFF: Ref clk/255

Values after reset are low (all 8 bits).

Figure 7–49. Clock Control



Notes: 1) During the identification phase, the maximal frequency on the MMC CLK line is 400 kHz (reference: bus timing specifications Chapter 6 of the *MultiMediaCard System Specification Version 3.1 – June, 2001. MMCA Technical Committee* or the *SD Memory Card Specifications – Part 1 Physical Layer Specification, Version 1.0 – March 2000 + Supplementary Notes Part 1 June 2000. SD Group*). That is, you must set a value of 120 into the frequency ratio register because the reference clock frequency is 48 MHz.

Notes: 2) During data transfer phase the maximum frequency is 16 MHz for MMC cards, 24 MHz for SD cards, and 12 MHz for SPI serial flash cards.

Notes: 3) The duty cycles of the generated MMC_CLK and SPI_CLK clock signals depend on the Clk_div value and on the polarity setting (MMC_SPI:POL) for SPI_CLK signal only. The low- and high-time approximate values can be computed using set-in rules.

Table 7–99. MMC_CLK/SPI_CLK High-/Low-Time Computation

Clk_Div	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 (Clk_div/2)	ref_clk_per (Clk_div/2)
Odd ≥ 3 (POL=PHA)	ref_clk_per (TRUNC[Clk_div/2] + 1)	ref_clk_per (TRUNC[Clk_div/2])
Odd ≥ 3 (POL≠PHA)	ref_clk_per (TRUNC[Clk_div/2])	ref_clk_per (TRUNC[Clk_div/2] + 1)

ref_clk_per is reference clock period (in ns) to the module (end-system dependant).

TRUNC is the truncate to an integer number function (round down).

Example 1: Module reference clock = 48 MHz (20.83 ns), target is MMC card.

- clk_div = 3 (MMC card is 20 MHz max).
- MMC_CLK period = 62.5 ns (> 50 ns OK)
- Ideal MMC_CLK high time = 41.66 ns (>>10 ns)
- Ideal MMC_CLK low time = 20.83 ns (>>10 ns)

Table 7–100. MMC System Status Register (MMC_STAT)

Bit	Name	Description
15	Reserved	
14	Card_Err	Card status error in response
13	Card_IRQ	Card IRQ received (following CMD40)
12	OCR_busy	OCR busy (following CMD1 or ACMD41)
11	A_Empty	Buffer almost empty
10	A_Full	Buffer almost full
9	Reserved	
8	Cmd_CRC	Command CRC error
7	Cmd_timeout	Command response time-out (no response)
6	Data_CRC	Data CRC error
5	Data_timeout	Data response time-out (no response)
4	EOF_Busy	Card exit busy state
3	Block_RS	Block received/sent
2	Card_Busy	Card enter busy state
1	Reserved	
0	End_of_Cmd	End of command phase

Common to all bits:

- The local host can only clear a set bit location by writing a 1 into the bit location. A write 0 has no effect.
- When a bit location is set to 1 by the core, an interrupt is signaled to the local host if the interrupt was enabled.

Card Status Error (Card_Err)

MMC/SD mode only.

The core automatically sets this bit (14) when there is at least one error in a response of type R1, R1b or R6. Only bits referenced as type E (error) can set a card status error (see Table 7–101).

Table 7–101. Response Types

Response Type	Card Status Bits With Error	Response Register Significant Bits	Comments
R1 (MMC, SD)	31-26, 24-16,	MMC_RSP7[15:10,8-0]	These 15 bits can all generates errors.
	3* (opt)	MMC_RSP6[3]	This bit can also generate an error if enabled (bit 3 if MMC_SDIO[13]=1) per the SD application specification
R6 (SD)	15:13, 3	MMC_RSP6[15:13,3]	Correspond to 23, 22, 19, 3 card status errors

The error handler must parse the response registers to understand the source of the error.

Others responses (type R2/R3/R4/R5) do not trigger a card status error.

In SPI or SYSTEST modes, this bit has no meaning and always reads as 0.

- ❏ 0: No action or no error
- ❏ 1: Error occurred

Value after reset is low.

Card IRQ (Card_IRQ)

MMC mode only.

The core automatically sets this bit (13) when a card is in interrupt mode and exits Wait_IRQ state (irq) by asserting a 0 on the CMD line (cards are in open-drain mode). Only Class 9 MMC cards can be put into interrupt mode when in stand-by state using a GO_IRQ_STATE (CMD40) command (see *Interrupt Mode Description,* Section 4.3 of the *MultiMediaCard System Specification Version 3.1 – June, 2001. MMCA Technical Committee.* [1]). SD cards do not support interrupt mode.

In SPI or SYSTEST modes, this bit has no meaning and always reads as 0.

- ❏ 0: No action or idle
- ❏ 1: Card exits IRQ state

Value after reset is low.

OCR Busy (OCR_busy)

MMC/SD mode only.

The core automatically sets this bit (12) after a SEND_OP_COND (CMD1) or a SD_APP_OP_COND (ACMD41) command when one or more cards have not yet completed power up. When this bit is set, the CMD1/ACMD41 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 *Power-Up Description*—Section 6.3 of *MultiMediaCard System Specification Version 3.1 – June, 2001. MMCA Technical Committee* or section 6.4 of the *SD Memory Card Specifications – Part 1 Physical Layer Specification, Version 1.0 – March 2000 + Supplementary Notes Part 1 June 2000. SD Group* [2]).

In SPI or SYSTEST modes, this bit has no meaning and always reads as 0.

- ❏ 0: No action or card powered-up
- ❏ 1: OCR busy

Value after reset is low.

Buffer Almost Empty (A_Empty)

The core automatically sets this bit (11) during a write operation to the card when the level is below the threshold value set in MMC_BUFF:AE_Level register bits. 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 to the main DMA controller of the system is generated.

The A_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 send to the card. Also, once set, the core internally masks a new set condition till the local host has performed [AE_Level+ 1] write access(es) to the FIFO.

AE_Level is the decimal equivalent set binary value (0–31).

- ❏ 0: No action or buffer is equal or above almost empty level.
- ❏ 1: Buffer almost empty

Value after reset is low.

Buffer Almost Full (A_Full)

The core automatically sets this bit (10) during a read operation to the card when the level is above the threshold value set in MMC_BUFF:AF_Level register bits. This bit 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 to the main DMA controller of the system is generated.

The A_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 [AF_Level +1] read access(es) from the FIFO.

AF_Level is the decimal equivalent set binary value (0–31).

- ❏ 0: No action or buffer is below or equal almost full level.
- ❏ 1: Buffer almost full

Value after reset is low.

Command CRC Error (Cmd_CRC)

MMC/SD mode only.

The core automatically sets this bit (8) if there is a CRC7 error in the command response (bits 7:1 of all response types except type R3). A CMD1 (MMC) or ACDM41 (SD) cannot trigger a CRC 7 error.

In SPI or SYSTEST modes, this bit has no meaning and always reads as 0

- ❏ 0: No action or no CRC7 error
- ❏ 1: CRC7 error

Value after reset is low.

Command Time-out Error (Cmd_timeout)

MMC/SD mode only.

The core automatically sets this bit (7) if the card does not respond within the specified number of command time-out clock cycles (CTO) that is set in MMC_CTO register (see N_{CR} timing requirements) to any command requiring a response.

If this bit is set after a command time-out, clearing this bit automatically stops the MMC clock and force the controller state to idle.

In SPI or SYSTEST modes, this bit has no meaning and always reads as 0.

- 0: No action or no command time-out
- 1: Command time-out

Value after reset is low.

Data CRC Error (Dat_CRC)

MMC/SD mode only.

The core automatically sets this bit (6) if there is CRC16 error in the data phase response following a block read command (single or multiple) or if there is a 3-bit CRC status token error 101 to signal for data transmission error during a block write command (single or multiple). For a multiple block transfer, the CRC is checked for every block.

In SPI or SYSTEST modes, this bit has no meaning and always reads as 0.

- 0: No action or no CRC error
- 1: CRC16 error (read), 3-bit CRC token error (write)

Value after reset is low.

Data Time-out Error (Dat_timeout)

The core automatically sets this bit (5) if the card does not respond within the specified number of data time-out clock cycles (DTO) that is set in MMC_DTO register.

In SPI mode, this bit also is set if the RDY/BUSY signal remains asserted in busy condition for DTO consecutive clock cycles.

If this bit is set after a data time-out, a clear of this bit automatically stops the MMC or SPI clock and forces the controller state to idle.

In SYSTEST mode, this bit has no meaning and always reads as 0.

- 0: No action or no data time-out.
- 1: Data time-out

Value after reset is low.

Card Exit Busy State (EOF_Busy)

MMC/SD mode only.

The core automatically sets this bit (4) when the addressed card releases the DAT line from its busy state (low level = busy). This bit can only get set during a programming phase (write operation) to a MMC or SD memory card.

In SPI or SYSTEST modes, this bit has no meaning and always reads as 0.

- ❏ 0: No action
- ❏ 1: Data line released/exit busy state

Value after reset is low.

Block Received/Sent (Block_RS)

The core automatically sets this bit (3) at the end of a block transfer (read or write).

In MMC or SD mode, this bit is set when the block transfer completes with no 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 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).*

There is a distinction to be made between DMA and non-DMA receive operation.

In non-DMA RX mode, this bit is set after the very last byte has been received in the FIFO. At this stage, the FIFO is not empty and must be read by the local host till it gets empty 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.

In SYSTEST mode, this bit has no meaning and always reads as 0.

- ❏ 0: No action
- ❏ 1: Block received/sent

Value after reset is low.

Card Enter Busy State (Card_Busy)

MMC/SD mode only.

The core automatically sets this bit (2) when the addressed card asserts the DAT line to a low level during a programming phase (write operation) to a MMC or SD memory card. For the MMC card only, the user can optionally use this interrupt to deselect the card (which continues to program) and select another card.

In SPI or SYSTEST modes, this bit has no meaning and always reads as 0.

- ❏ 0: No action
- ❏ 1: Data line asserted low/card busy

Value after reset is low.

End of Command (End_of_Cmd)

MMC/SD mode only.

The core automatically sets this bit (0) 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.

In SPI or SYSTEST modes, this bit has no meaning and always reads as 0.

- ❏ 0: No action
- ❏ 1: End of command/response sequence

Value after reset is low.

When a CMD12 command is transferred after a multiple block read, the End_of_Cmd bit [0] is not set. Alternatively, the Card_Busy bit [2] is set. After this, the EOF_Busy bit [4] cannot be set. Avoid this condition with software by using the following sequence:

1) Mask the busy interrupt before the CMD12 command is sent, because it is possible that the unexpected busy interrupt to the MPU is generated.
2) Clear the Card_Busy [2] after the response for CMD12 is returned, because it is possible that this flag is set.
3) Enable the busy interrupt if it is to be used (write 0xFFFF to the MMC_IE register).

Table 7–102. MMC System Interrupt Register (MMC_IE)

Bit	Name	Description
15	Reserved	
14	Card_Err_IE	Card status error interrupt enable
13	Card_IRQ_IE	Card IRQ interrupt enable
12	OCR_busy_IE	OCR busy interrupt enable
11	A_Empty_IE	Buffer almost empty interrupt enable
10	A_Full_IE	Buffer almost full interrupt enable
9	Reserved	
8	Cmd_CRC_IE	Command CRC error interrupt enable
7	Cmd_timeout_IE	Command response time-out Interrupt enable
6	Data_CRC_IE	Data CRC error interrupt enable
5	Data_timeout_IE	Data response time-out interrupt enable
4	EOF_Busy_IE	Card exit busy state interrupt enable
3	Block_RS_IE	Block received/sent interrupt enable
2	Card_Busy_IE	Card enter busy state interrupt enable
1	Reserved	
0	End_of_Cmd_IE	End of command interrupt enable

Common to all bits:

- When a bit location is set to 1 by the local host, an interrupt is signaled to the local host if the corresponding bit location in MMC_STAT register is asserted to 1 by the core.
- If set to 0, the interrupt is masked and not signaled to the local host.
  - 0: Interrupt disabled
  - 1: Interrupt enabled
- Values after reset are low (all bits).

The 16-bit MMC command time-out register (MMC_CTO) specifies the maximum number of clock cycles before a command time-out condition occurs.

Table 7–103. MMC Command Time-out Register (MMC_CTO)

Bit	Name	Description
15–8	Reserved	
7–0	CTO	MMC command time-out value.

Command Time-out Value (CTO)

MMC/SD mode only.

The local host sets this field (bits 7:0) based on N_{CR} clock cycles. MMC and SD card specifies 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 gets set to 1 in MMC_STAT[7] register bit.

For MMC card interrupt mode support, this time-out is disabled when the command passes with an R5 response (CMD40).

- 0x00: Command time-out disabled
- 0x01: One clock cycle
- 0xFD: 253 clocks cycles ($2^8 – 3$)

  The 0xFF and 0xFE cannot be used.

Values after reset are low (all 8 bits).

This 16-bit register specifies the maximum number of clock cycles before a data time-out condition occurs.

Table 7–104. MMC Data Time-out Register (MMC_DTO)

Bit	Name	Description
15–0	DTO	Data read time-out

Data Time-out Value (DTO)

In MMC/SD mode, the local host sets this field (bits 15-0) 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 can be obtained by reading the response register after a successful execution of a SEND_CSD command (CMD9).

If the card does not respond within the specified number of cycles, data time-out gets set to 1 in MMC_STAT[5] register bit.

The effective number of clock cycles for time-out value are to be multiplied by 1024 if MMC_SDIO:DTO_PS_En=1 and by 1 if DTO_PS_En=0.

In SPI mode, a data time-out condition is also generated if the RDY/BUSY signal is asserted low (BUSY) for DTO consecutive clocks cycles (see Table 7–105).

Table 7–105. Data Time-out Conditions

DTO	DTO_PS_En=0	DTO_PS_En=1	
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})	

Values after reset are low (all 16 bits).

The MMC data access register (MMC_DATA) is the entry point for the local host to read data from, or write data into, the FIFO buffer. The FIFO size is 32x16bits (64 bytes). Bytes within a word are stored and read in little endian format.

If the local host accesses this register byte-wise, the MSB (bits [15:8]) must be always written/read first. A byte access to the LSB without a prior write into the MSB results in having the MSB filled with 0x00.

Table 7–106. MMC Data Access Register (MMC_DATA)

Bit	Name	Description
15–0	DATA	Transmit/receive FIFO data

Transmit/Receive FIFO Data Value (DATA)

In MMC/SD mode, this register field (bits 15-0) 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.

Since the block length is passed as an argument, it is legal for the local host to perform only 16-bit accesses (read or write) to 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 (op-code 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 purpose.

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.

This register configures the core for the number of bytes to read or write. It must be initialized at least once prior to starting an MMC, SD, or SPI block data transfer (read or write).

Table 7–107. MMC Block Length Register (MMC_BLEN)

Bit	Name	Description
15–11	Reserved	
10–0	BLEN	Block length value

Block Length (BLEN)

General operation: A write into this register (bits 10-0) initializes an 11-bit counter that decrements by 1 after each byte transferred. A read into this register returns the number of bytes remaining to be transferred. When the counter reaches 0 and after the last byte transfer completes, BLEN is automatically reloaded to its programmed value by the core.

In MMC/SD mode, this 11-bit value specifies the data block length. This value must be set respectively with max $2^{READ\_BL\_LEN}$-1 for a block read or max $2^{WRITE\_BL\_LEN}$-1 for a block write.

READ_BL_LEN and WRITE_BL_LEN are CSD register settings of the card returned in a response, R2, following a SEND_CSD command (CMD9).

In SPI modes and for a read transaction, BLEN must be initialized with the exact byte count to read negative 1, excluding the op-code and address arguments.

Op-code and address arguments that are passed to the SPI device must be written into the FIFO buffer prior to 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 modes and for a write transaction, BLEN must be initialized with the exact byte count to write negative 1, including the number of bytes needed to pass the op-code and address arguments.

It is recommended to have op-code and addresses that are passed to the SPI module written into the FIFO buffer prior to starting the SPI transfer. This allows in DMA write operation to access only the data portion. BLEN starts to decrement for every byte shifted out to the SPI device.

- 0x000: 1 byte
- 0x7FF: 2048 bytes

Values after reset are low (all 11 bits).

This register 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.

Table 7–108. MMC Number of Blocks Register (MMC_NBLK)

Bit	Name	Description
15–11	Reserved	
10–0	NBLK	Number of blocks value

Number of Blocks (NBLK)

MMC/SD mode only.

In MMC/SD mode, this 11-bit value (bits 10-0) specifies the number of blocks for a multiple block data transfer (read or write). Each block is of size MMC_BLEN:BLEN (block length value). This value must be set with the number of blocks – 1.

This register must be programmed prior to any multiple block data transfer. A write into this register initializes an 11-bit counter that decrements by one after each block transfer. A read into 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 transfer, a Block_RS interrupt is generated only once after the last successful transfer when NBLK reaches 0.

In stream mode, the minimum allowable number of blocks is two blocks.

Note:

This value must be 0x000 for a single block transfer. In stream mode, the minimum allowable number of blocks is two blocks. If the transfer is interrupted by a STOP_TRANSMISSION command (CMD12) before the counter reached 0, you must reprogram this register prior to starting any new single or multiple block data transfers.

- 0x000: 1 block
- 0x7FF: 2048 blocks

Values after reset are low (all 11 bits).

This register configures the buffer threshold level of the thirty two 16-bit-word FIFO and enables DMA transfers.

Table 7–109. MMC Buffer Configuration Register (MMC_BUF)

Bit	Name	Description
15	RX_DMA_En	Receive DMA channel enable
14–13	Reserved	
12–8	AF_Level	Buffer almost full level
7	TX_DMA_En	Transmit DMA channel enable
6–5	Reserved	
4–0	AE_Level	Buffer almost empty level

Receive DMA Channel Enable (RX_DMA_En)

When this bit (15) is set to 1, the receive DMA channel is enabled and the A_Full status bit is forced to 0 by the core irrespectively of AF_level setting (see Table 7–109).

- ❏ 0: Receive DMA channel disabled
- ❏ 1: Receive DMA channel enabled

Value after reset is low.

Buffer Almost Full Level (AF_Level)

This register (bits 12-8) holds the programmable almost full level value used to determine almost full buffer condition. If you want an interrupt or a DMA read request to be issued during a read operation when the data buffer holds n words of 16 bits, then AF_Level must be set with n-1.

- ❏ 0x00 1 16-bit word (2 bytes)
- ❏ 0x1E: 31 16-bit word (62 bytes)
- ❏ 0x1F: 32 16-bit word (64 bytes)

Values after reset are 0x1F.

Transmit DMA Channel Enable (TX_DMA_En)

When this bit (7) is set to 1, the transmit DMA channel is enabled and the A_Empty status bit is forced to 0 by the core irrespectively of AE_level setting (see Table 7–109).

More information regarding DMA operation can be found in Chapter 5, *System DMA Controller*.

- ❏ 0: Receive DMA channel disabled
- ❏ 1: Receive DMA channel enabled

Value after reset is low.

Buffer Almost Empty Level (AE_Level)

This register (bits 4-0) holds the programmable almost empty level value used to determine almost empty buffer condition. If you want an interrupt or a DMA write request to be issued during a write operation when the data buffer holds n words of 16 bits, then AE_Level must be set with n.

- ❏ 0x00: 0 16-bit word (0 bytes)
- ❏ 0x1E: 30 16-bit word (60 bytes)
- ❏ 0x1F: 31 16-bit word (62 bytes)

Value after reset is low.

This register is used to configure the SPI interface and start an SPI transfer if SPI mode has been enabled.

Table 7–110. MMC SPI Configuration Register (MMC_SPI)

Bit	Name	Description
15	Start	Start SPI transfer
14	WnR	Write/not read
13–12	Reserved	
11–10	TCSH	Chip-select hold time control
9–8	TCSS	Chip-select setup time control
7–6	Reserved	
5–4	CS	Chip-select control
3	CSM	Chip-select mode
2	CSD	Chip-select disable
1	PHA	Phase control
0	POL	Polarity control

Start SPI Transfer (Start)

This set-only bit (15) always reads as 0. A write to 0 has no effect.

When set to 1 by the local host, a SPI transfer is automatically started.

Note:

The user must take care to initialize MMC_BLEN:BLEN before starting an SPI transfer.

SPI transfer automatically stops when the size programmed in MMC_BLEN:BLEN decrements down to 0 (in read and in write).

- ❏ 0: No action
- ❏ 1: SPI transfer started

Value after reset is low.

Write/Not Read (WnR)

This bit (14) instructs the 11-bit block length counter in MMC_BLEN:BLEN to decrement either on byte read when WnR = 0 or on byte write when WnR = 1.

- 0: Decrement on byte received
- 1: Decrement on byte sent

Value after reset is low.

Chip-Select Hold Time Control (TCSH)

This field (bits 11-10) defines the number of interface clock cycles that the core waits after last SPI_CLK clock edge before asserting the chip-select signals to their inactive-high level.

- 00: Minimum 0.5 clock cycle
- 01: Minimum 1.5 clock cycles
- 10: Minimum 2.5 clock cycles
- 11: Minimum 3.5 clock cycles

Values after reset are low (2 bits).

Chip-Select Setup Time Control (TCSS)

This field (bits 9-8) defines the number of interface clock cycles that the core waits after asserting the chip-select signals to their active-low level before asserting first SPI_CLK clock edge.

- 00: Minimum 1 clock cycle
- 01: Minimum 2 clock cycles
- 10: Minimum 3 clock cycles
- 11: Minimum 4 clock cycles

Values after reset are low (2 bits).

Figure 7–50. SPI Mode C/S Timings Controls (POL = 0)

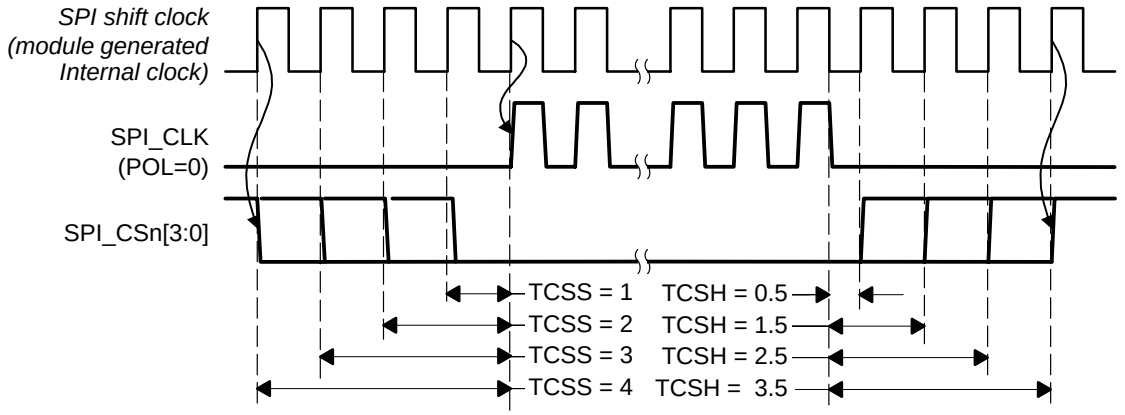
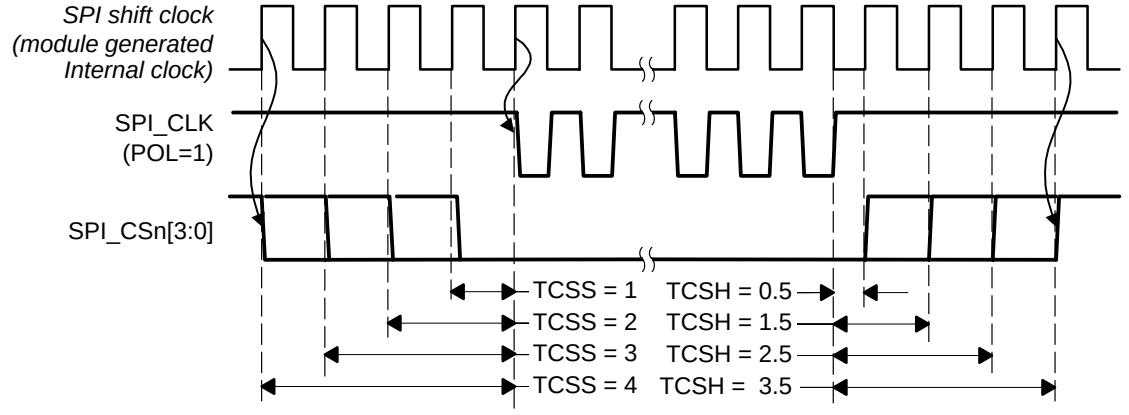


Figure 7–51. SPI Mode C/S Timings Controls (POL = 1)



Chip-Select Control (CS)

Encoded value (bits 5-4) that selects the device being targeted for SPI transfer.

- ❏ 00: Reserved (no device is selected)
- ❏ 01: C/S 1
- ❏ 10: C/S 2
- ❏ 11:C/S 3

Values after reset are low (2 bits).

Chip-Select Mode (CSM)

When this bit (3) is set to 0 and enabled (CSD=0), the selected CS signal pin goes active (low) only when SPI transfer is started and brought back automatically to its inactive state (high), when the SPI transfer completes.

When set to 1, the automatic control of the CS signal is disabled. Instead, the selected CS signal pin is manually controlled by the chip-select disable register bit (CSD). This mode provides support for complex SPI transfer scheme that requires CS to be kept active during the entire transfer (ex: MMC card write with busy condition).

- ❏ 0: Automatic mode
- ❏ 1: Manual mode (controlled by CSD)

Value after reset is low.

Chip-Select Disable (CSD)

When this bit (2) is set to 0, the selected CS signal is asserted to its active (low) state either automatically when CSM = 0, or manually when CSM = 1.

When set to 1, the selected CS signal is forced to its inactive (high) state. It can be used to send dummy clocks with CS inactive to a MMC or SD card.

- ❏ 0: Selected CS is conditionally asserted (low).
- ❏ 1: Selected CS is deasserted (high).

Value after reset is low.

Table 7–111. 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)

Clock Phase (PHA)

The clock polarity and clock phase bits select four different clocking schemes for the SPICLK pin.

The clock phase bit (1) selects a half cycle delay for clock.

When clock phase = 0:

- ❏ MSB data is ready one half cycle of SPICLK before the SPI clock starts.
- ❏ Data is shifted-in in reception on the first edge transition of SPICLK.
- ❏ Data is shifted-out in transmission on the second edge transition of SPICLK.

When clock phase = 1:

- ❏ Data is shifted-out in transmission on the first edge transition of SPICLK.

Data is shifted-in in reception on the second edge transition of SPICLK:

- ■ 0: Phase 0
- ■ 1: Phase 1

Value after reset is low.

Clock Polarity (POL)

The clock polarity bit (0) selects the active edge of the clock, either rising or falling.

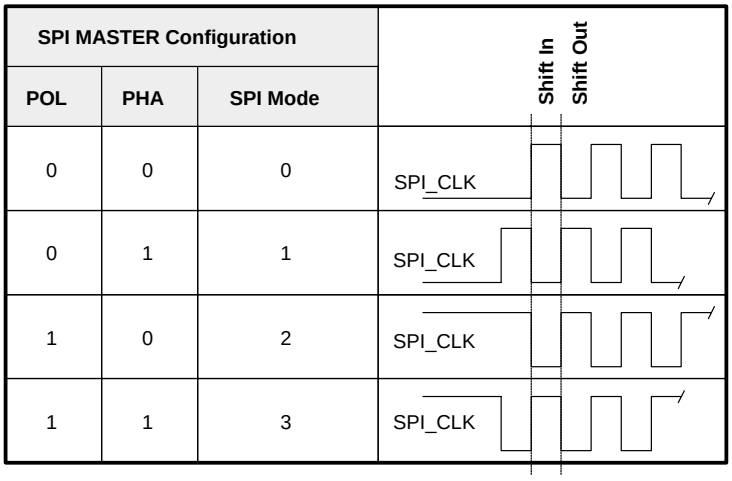
When 0, the idle value of the SPI clock signal is low and the rising edge is active.

When 1, the idle value of the SPI clock signal is high and the falling edge is active.

- ❏ 0: Rising edge active
- ❏ 1: Falling edge active

Value after reset is low.

Figure 7–52. SPI Master Configuration Bits



This register provides additional controls for the MMC/SD interface. It is also reserved for future SDIO operation (not supported in present version).

Table 7–112. MMC SDIO Mode Configuration Register (MMC_SDIO)

Bit	Name	Description
15–14	Reserved	
13	CER1_3_En	Card status error on bit 3 of response 1 enable
12–6	Reserved	
5	DTO_PS_En	Data time-out prescaler enable
4–0	Reserved	

Card Status Error on Bit 3 of Response R1 Enable (CER1_3_En)

This bit (13) must be set to 1 for SD cards only or application-specific commands that generates an error.

If set to 1, a card status error is generated if bit 3 of the status is 1 for a R1 or R1b response.

- ❏ 0: Error on bit 3 masked
- ❏ 1: Card status errors on bit 3 of response 1 enabled (SD card or application specific only)

Value after reset is low.

Data Time-out Prescaler Enable (DTO_PS_En)

When this bit (5) is set to 1 by the LH, the data time-out set in MMC_DTO register is x1024 the number of MMC_CLK cycles.

- ❏ 0: x1 (Prescaler off)
- ❏ 1: x1024 (Prescaler on)

Value after reset is low.

The MMC system test register (MMC_SYST) is used to control the signals that connect to I/O pins when the module is configured in system test (SYSTEST) mode (see Table 7–113).

Table 7–113. MMC System Test Register (MMC_SYST)

Bit	Name	Description
15–14	Reserved	
13	RDY_dat	Ready/busy input signal data value
12	DAT-dir	DAT[3–0] signals direction
11	DAT3_dat	DAT3 input/output signal data value
10	DAT2_dat	DAT2 input/output signal data value
9	DAT1_dat	DAT1 input/output signal data value
8	DAT0_dat	DAT0/SI input/output signal data value
7	CMD-dir	CMD/SO signal direction
6	CMD_dat	CMD/SO input/output signal data value
5	MMC_CK_dat	MMC clock output signal data value
4	SPI_CK_dat	SPI clock output signal data value
3	CS3_dat	C/S3 output signal data value
2	CS2_dat	C/S2 output signal data value
1	CS1_dat	C/S1 output signal data value
0	Reserved	

Ready/Busy Data (RDY_dat)

This read-only bit (13) 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.

DAT[3:0] Direction (DAT_dir)

When set, this bit (12) places all the in/out DAT[3:0] pins in output mode.

- ❏ 0: Input
- ❏ 1: Output

Value after reset is low.

DAT[3:0] Data (DATn_dat)

If DAT_dir = 0 (input mode direction), these bits (11-8) return the value on the corresponding DAT pins (high or low). A write into these bits has no effect.

If DAT_dir =1 (output mode direction), the DAT pins is driven high or low according to the value written into these register bits.

Values after reset are low (all 4 bits).

CMD Direction (CMD_dir)

When set, this bit (7) places the in/out CMD pin in output mode.

- ❏ 0: Input
- ❏ 1: Output

Value after reset is low.

CMD Data (CMD_dat)

If CMD_dir=0 (input mode direction), this bit (6) returns the value on the CMD pin (high or low). A write into this bit has no effect.

If CMD_dir =1 (output mode direction), the CMD pin is driven high or low according to the value written into this register bit.

Value after reset is low.

MMC_CLK Data (MMC_CK_dat)

The MMC_CK pin is driven high or low according to the value written into this register bit (5).

Value after reset is low.

SPI_CLK Data (SPI_CK_dat)

The SPI_CK pin is driven high or low according to the value written into this register bit (4).

Value after reset is low.

CS[3:1] Data (CSn_dat)

The CS[3:1] pins are driven high or low according to the values written into these register bits (3-1).

Values after reset are low (all 3 bits).

The read-only MMC module version register (MMC_REV) contains the revision number of the module. A write to this register has no effect.

Table 7–114. MMC Module Version Register (MMC_REV)

Bit	Name	Description
15–8	–	Reserved
7–0	REV	Module version number

Module Version Number (REV)

This 8-bit field (bits 7-0) indicates the revision number of the RTL for this module. This value is fixed by hardware.

The four LSBs indicate a minor revision.

The four MSBs indicate a major revision.

- 0x14: Version 1.4
- 0x20: Version 2.0

A reset has no effect on the value returned.

Table 7–115 through Table 7–122 describe 16-bit registers that hold specified bits positions for a 128-bit response of type R2.

Table 7–115. MMC/SD Command Response Register 0 (MMC_RSP0)

Bit	Name	Description
15–0	RESP0	CMD response (R2[15:0])

Table 7–116. MMC/SD Command Response Register 1 (MMC_RSP1)

Bit	Name	Description
15–0	RESP0	CMD response (R2[31:16])

Table 7–117. MMC/SD Command Response Register 2 (MMC_RSP2)

Bit	Name	Description
15–0	RESP0	CMD response (R2[47:32])

Table 7–118. MMC/SD Command Response Register 3 (MMC_RSP3)

Bit	Name	Description
15–0	RESP0	CMD response (R2[63:48])

Table 7–119. MMC/SD Command Response Register 4 (MMC_RSP4)

Bit	Name	Description
15–0	RESP0	CMD response (R2[79:64])

Table 7–120. MMC/SD Command Response Register 5 (MMC_RSP5)

Bit	Name	Description
15–0	RESP0	CMD response (R2[95:80])

Table 7–121 and Table 7–122 describe registers that also hold specified bit positions for a 32-bit response of type R1/R1b/R3/R4/R5/R6.

Table 7–121. MMC/SD Command Response Register 6 (MMC_RSP6)

Bit	Name	Description
15–0	RESP6	CMD response (R2[111:96], R1/R1b/R3/R4/R5/R6[23:8])

Table 7–122. MMC/SD Command Response Register 7 (MMC_RSP7)

Bit	Name	Description
15–0	RESP6	CMD response (R2[111:96], R1/R1b/R3/R4/R5/R6[39:24])

7.12.8 Command Flow

To correctly drive the MMC/SD adapter for a command execution, the host must follow the process shown in Figure 7–53 and Figure 7–54.

Figure 7–53. Command Flow

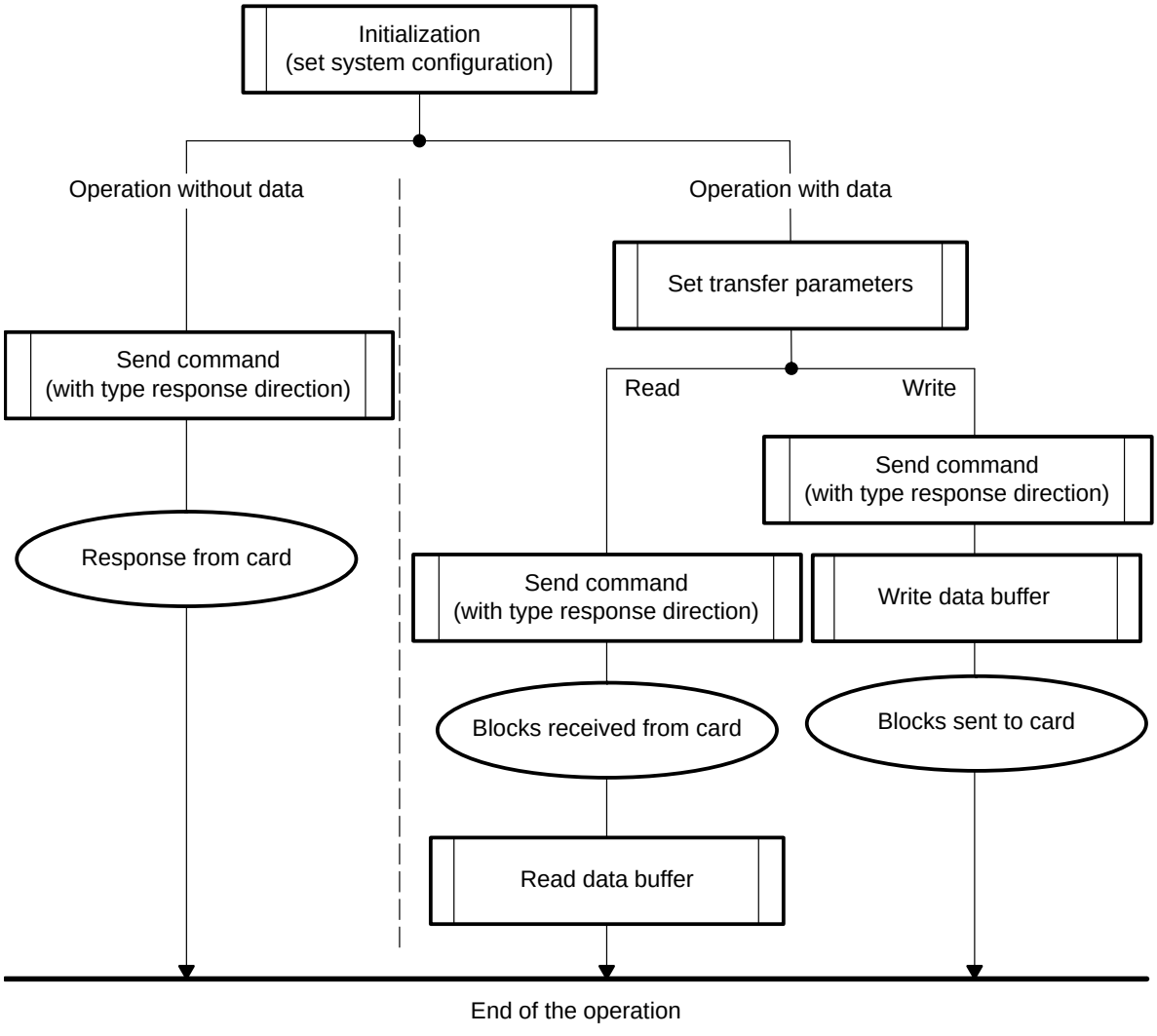
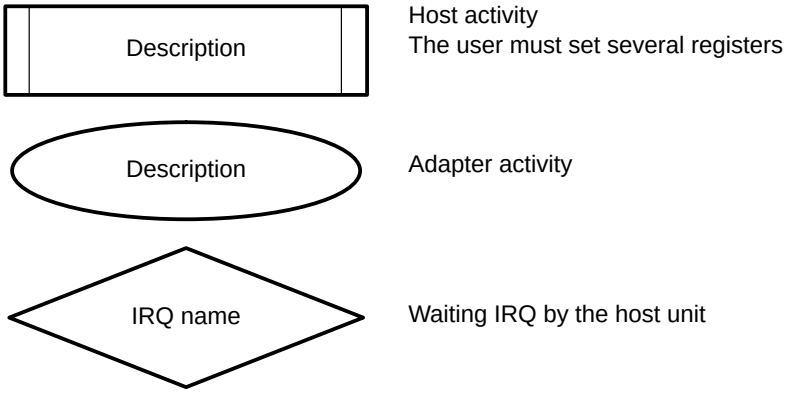


Figure 7–54. Initialization Phase



In all modes (MMC, SD, SPI) a initialization phase is necessary at the beginning. After the MMCSD adapter woks differently depending if the host sends a command without data, with data, and, depending the type, the index of the response and the direction.

Figure 7–55. Detail of Basic Operation

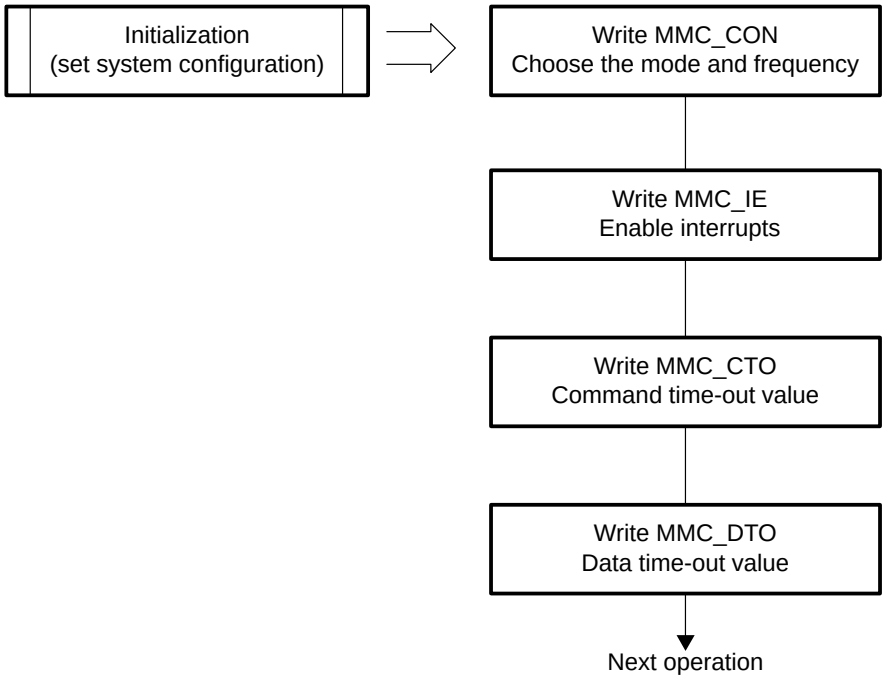
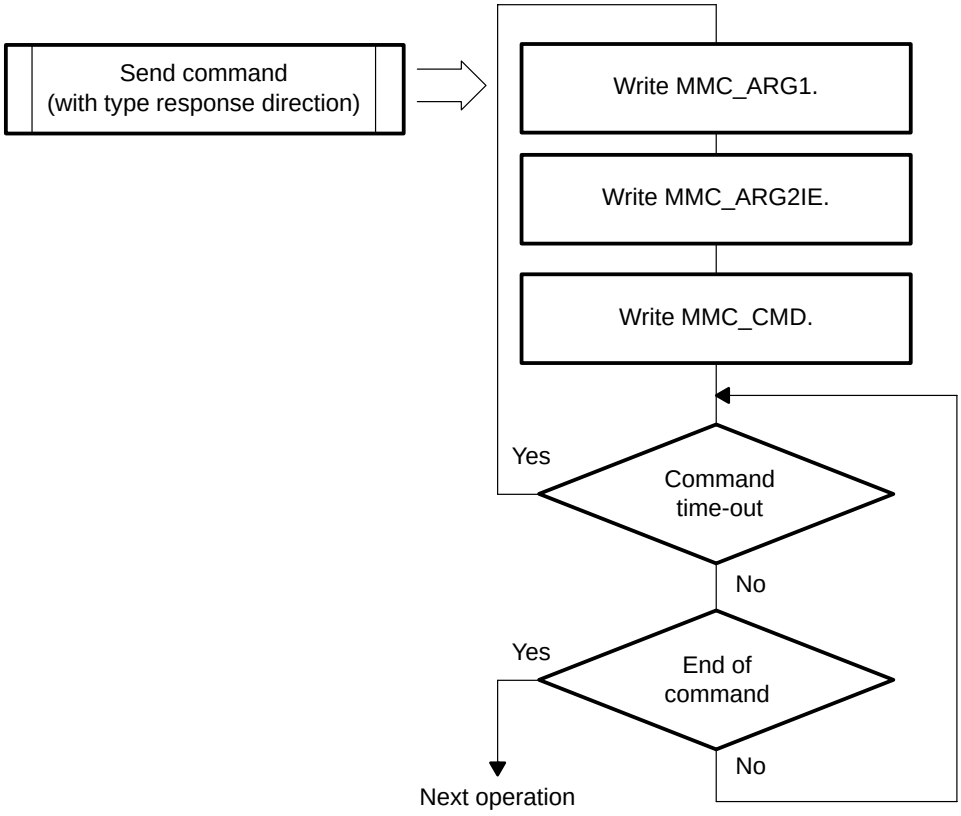
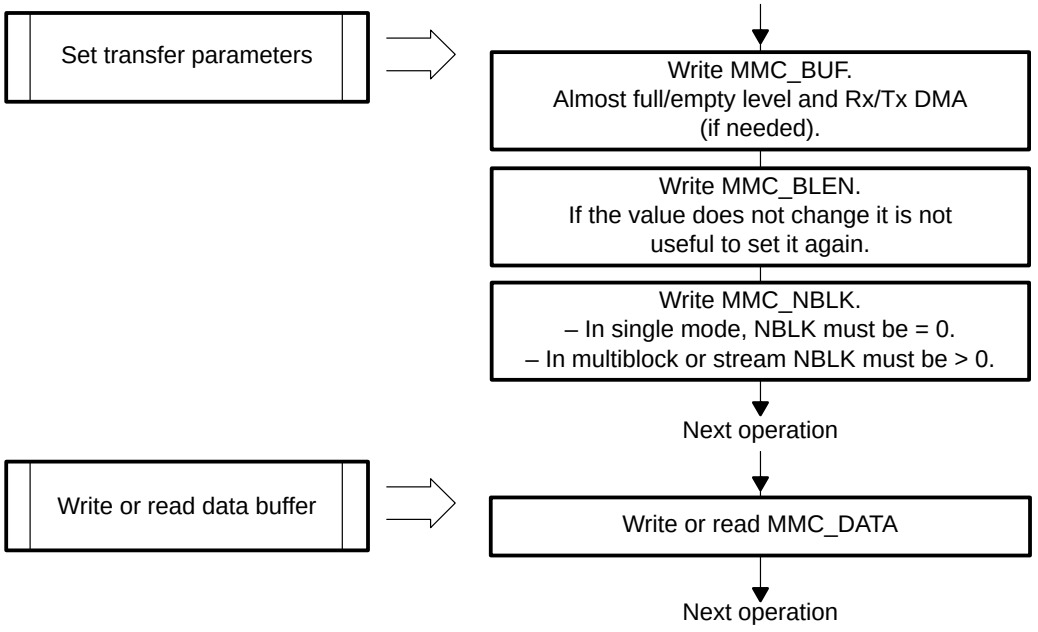


Figure 7–56. Command Transfer



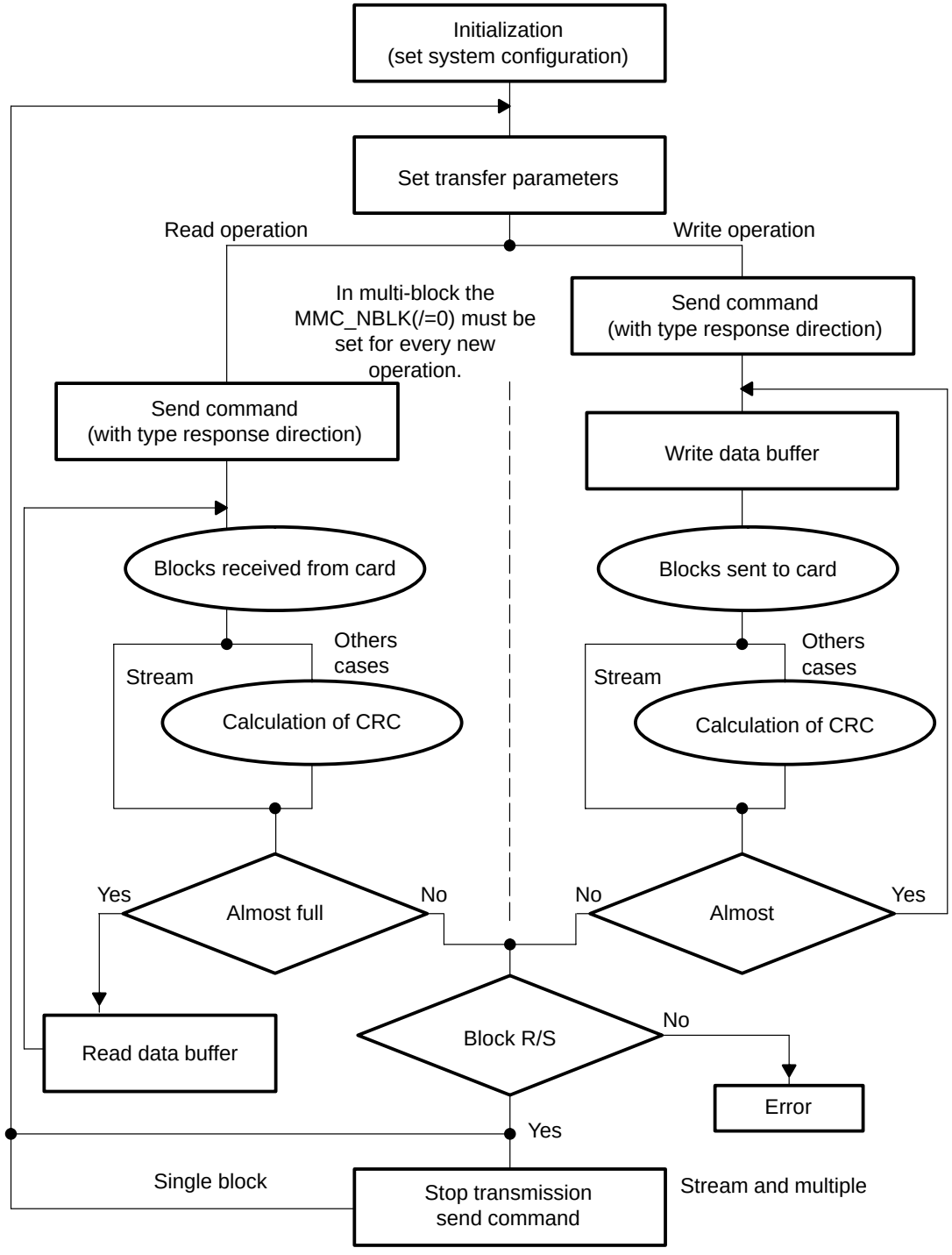
When a command that has no response is used (that is, CMD0, CMD4, CMD15), the command timeout condition can never occur, since there is no response expected. In this case, the NO path from the command timeout clock is taken, and the timeout interrupt is not valid.

Figure 7–57. Data Transfer



For the example shown in Figure 7–58, the mode selected is MMC/SD (MMC_CON[13:12] = 00).

Figure 7–58. Data Transfer in MMC/SD Mode Example



7.12.9 DMA Operation

7.12.9.1 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 equal or greater than the threshold set in AF_level.
- The DMA RX request is deasserted to its inactive level when the system DMA has read one word from the FIFO.

Because the request lasts one 16-bit word read cycle, it is recommended that the threshold level (AF_level) equal the DMA burst size (n) minus 1. For instance, if the system DMA is programmed to support one word read access, AF_level must be set to 0.

The MMC/SD host controller does not generate a new DMA request until the system DMA has read the N words corresponding to the previous DMA request, even if the FIFO level is equal to or greater than the programmed threshold.

Since each DMA transfer has equal size, it is necessary to have the total data size of the transfer be a multiple of the DMA read access size (max 32 words).

Summary:

DMA transfer size = n ≤ FIFO size
(max 32 16-bit words)

AF_level = n -1 (FIFO threshold level)

n = submultiple of total transfer size

Example: Multiple block read of 10 blocks of 512 bytes each.

The DMA transfer size n can be set to 20 words (40 bytes) and AF_level= 0x13 (0x14-1). Then the read transfer operation completes after 128 system DMA read requests.

The receive FIFO does not overflow. If the FIFO gets full, the MMC_clk clock signal is momentarily stopped until the system DMA or the TI925T performs a read access, which starts emptying the FIFO.

When using the MMC DMA receive mode, the MPU software must enforce that the MPU never access the MMC_DAT register at the same time the DMA is accessing the register. Failure to enforce this restriction may cause unexpected results.

7.12.9.2 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 the threshold set in AE_level after the block write command has been set (write action into MMC_CMD).
- 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 level (AE_level) equal DMA burst size (n) minus 1. For instance, if the system DMA is programmed to transfer one word write access, AE_level must be set to 0.

In DMA mode, because a new DMA TX request can be generated after the first read from the FIFO by the core, it is possible for the FIFO to hold a maximum of two DMA transfers of n words minus one. Hence the maximum permitted DMA transfer size is half the FIFO size.

The MMC/SD host controller does not generate a new DMA request until the system DMA has written the N words corresponding to the previous DMA request, even if the FIFO level is equal to or greater than the programmed threshold.

Because each DMA transfer has equal size, it is necessary to have the total data size of the transfer be a multiple of the DMA write access size (max 16 words).

Summary:

DMA transfer size = n ≤ FIFO size/2
(max 16 16-bit words)

AE_level = n -1 (FIFO threshold level)

n = submultiple of total transfer size

Example: Multiple block write of 10 blocks of 512 bytes each. The DMA transfer size n can be set to 10 words (20 bytes) and AE_level= 0x9. Then the write transfer operation completes after 256 system DMA write requests.

The transmit FIFO does not underflow. If the FIFO gets empty, the MMC_clk clock signal is momentarily stopped till the system DMA or the local host performs a write access, which starts filling the FIFO.

7.12.10 Local Host (IRQ/Polling) Mode

7.12.10.1 MMC Local Host (IRQ/Polling) Receive Mode

During a local host block read operation (single or multiple) using inerrupt/polling mode, the A_Full status bit is set active (high level) when the FIFO level becomes equal or greater than the threshold set in AF_level. The local host can only clear the A_Full set bit by writing a '1' into the A_Full status bit location.

The threshold level (AF_level) equals LH (local host) transfer size (n) minus 1. If the threshold is set to 0 (AF_level=0x00), the local host has to read one word by one word. If the threshold is set to 31 (AF_level=0x1F), the local host has to read 32 words by 32 words.

New A_Full status bit assertion (high level) is internally masked until the LH has not performed exactly n reads.

Unlike DMA mode, it is not needed to have the total data size of the transfer be a multiple of the LH read access size (maximum 32 words). The last LH read access can be down to the LH transfer size when ending any total data size transfer, as follows:

- ❏ LH transfer size = n =FIFO size (max 32 16–bit words)
- ❏ AF_level = n –1 (FIFO threshold level)

An example is a multiple block read of 10 blocks of 512 bytes each.

The LH transfer size n can be set to 20 words (40 bytes) and AF_level= 0x13 (0x14–1). Then the LH read transfer operation completes after 128 A_Full status bit assertions.

The LH transfer size n can also be set to 15 words (30 bytes) and AF_level = 0x0E (0x0F–1). Then the LH read transfer operation completes after 170 system A_Full status bit assertion/deassertions, plus a last assertion for the last transfer of 10 words (20bytes).

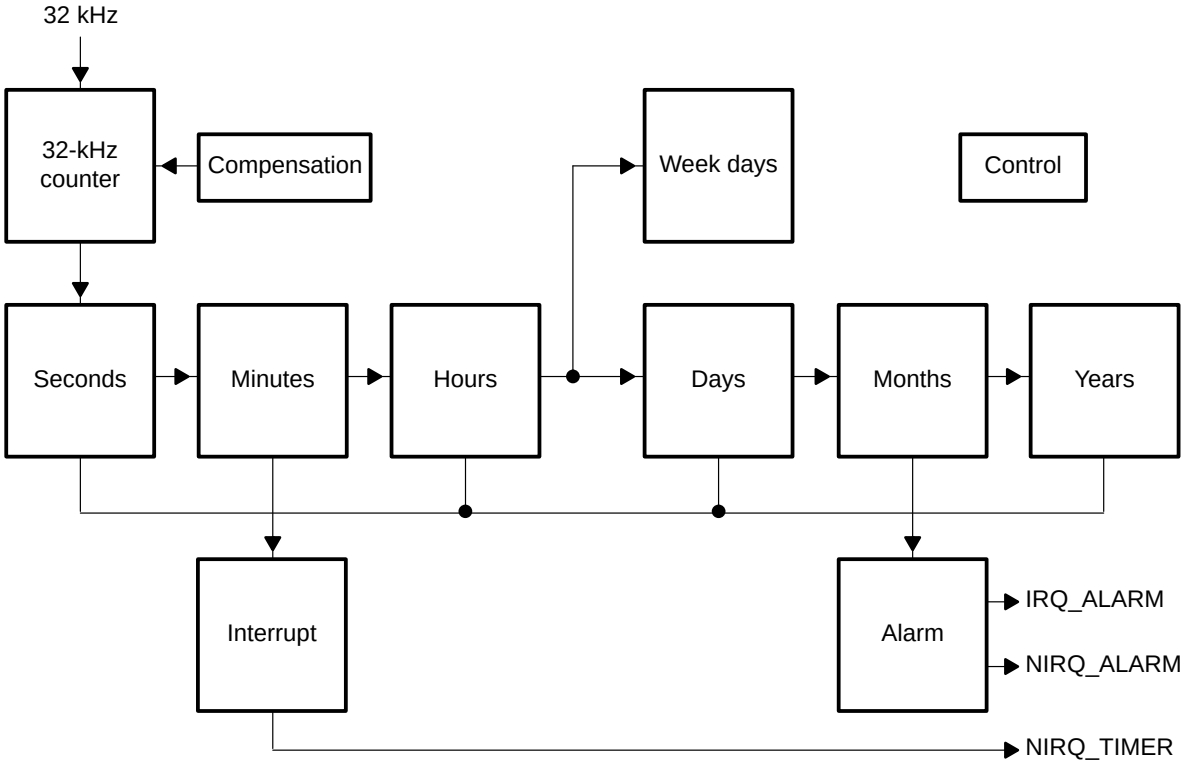
The receive FIFO never overflows. If the FIFO gets full, the MMC_clk clock signal is momentarily stopped untll the system DMA or the LH performs a read access, which starts emptying the FIFO.

7.13 Real-Time Clock

The real time clock (RTC) is an embedded module (see Figure 7–59). Its basic features 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 drift compensation

Figure 7–59. RTC Clock Diagram



7.13.1 Register Descriptions

All the time and calendar information is available in dedicated registers, which are called time and calendar registers. Time and calendar register values are written in binary coded decimal (BCD) code (see Table 7–123).

Table 7–123. Time and Calendar Register Values

Time Unit	Range	Remarks
Year	00 to 99	Leap year: year divisible by 4
		Common year: other year
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-hour mode
		01 to 12 in AM/PM mode
Minutes	00 to 59	
Seconds	00 to 59	

7.13.2 Register Access

There are three types of registers:

- ❑ Time and calendar registers/alarm
- ❑ General
- ❑ Compensation

These three types have their own access constraints.

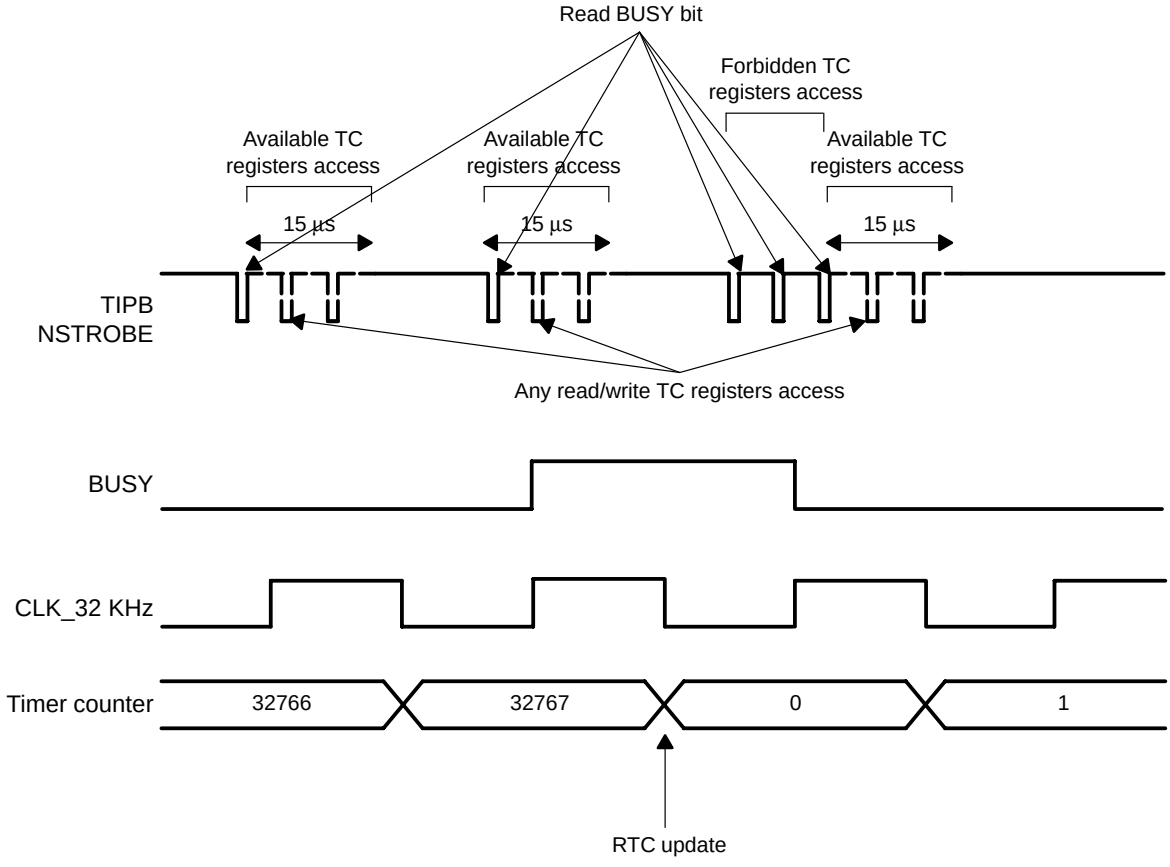
Access Period Violation

Disable all incoming interrupts during the register read processs to prevent process interruption and possible violation of the authorized 15-μs access period.

7.13.2.1 Time and Calendar Registers/Alarm Registers

To read or write correct data to and from the time and calendar registers/alarm registers, the MPU must first poll 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 safely access the time and calendar registers/ alarm registers. At the end of the access period, the MPU must restart the previous sequence. If the MPU accesses the time and calendar registers outside of the access period, the access is not ensured (see Figure 7–60).

Figure 7–60. Time and Calendar Registers and Alarm Register Access



7.13.2.2 General Registers

The MPU can access the STATUS_REG and the CTRL_REG at any time (except 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 only be used 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. To save power, this bit must only be used if the RTC function is not wanted in the application.

7.13.2.3 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 it is acceptable to update them during the second preceding a compensation event (see Figure 7–61).

For example, the MPU can load the compensation value into these registers after each hour event during an available access period.

7.13.2.4 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 and 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 ensure that the RTC updating takes place in more than 15 µs (bit BUSY should be 0). Then the MPU must perform all changes in less than 15 µs to prevent partial updating between the beginning and the end of the writing sequence into time and calendar registers.

Also, the MPU can stop the RTC by clearing 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), update time and calendar values, and restart the RTC by resetting the STOP_RTC bit.

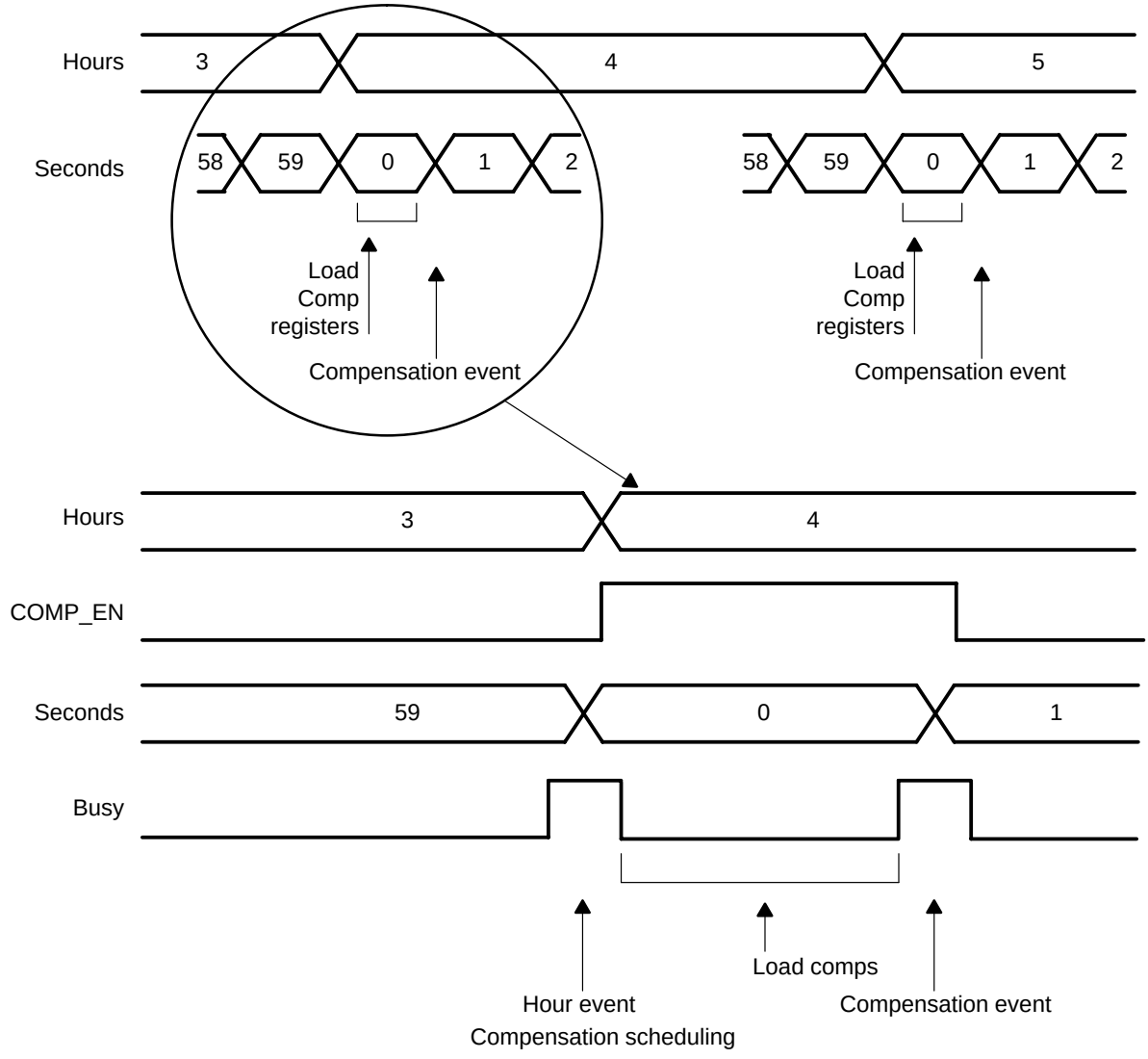
7.13.2.5 Rounding Seconds

Time can be rounded to the closest minute, by setting 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. 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.

Figure 7–61. Compensation Scheduling



7.13.2.6 Interrupts Management

RTC can generate two interrupts:

- ❏ Timer interrupt (IRQ_TIMER)
- ❏ Alarm interrupt (IRQ_ALARM_CHIP)

7.13.2.7 Timer Interrupt

IRQ_TIMER interrupt can be generated periodically every second, every minute, every hour, or every day (RTC_INTERRUPTS_REG[1:0]).

The IT_TIMER bit of the interrupt register enables this interrupt.

The timer interrupt is a negative-edge-sensitive interrupt (low-level pulse duration = 15 µs).

RTC_STATUS_REG [5:2] are only updated at each new interrupt and show what events have happened, as shown in Table 7–124.

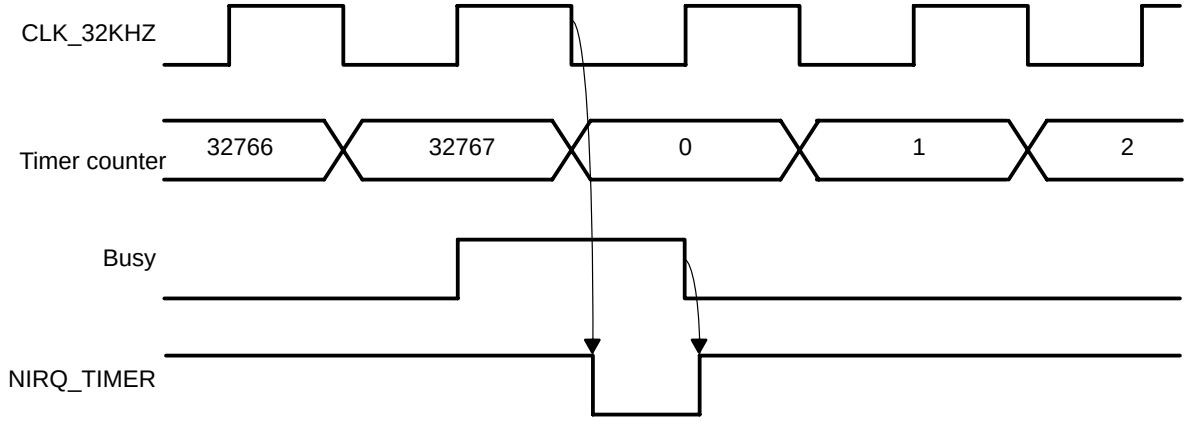
Table 7–124. Timer Interrupts

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 to programmed periodical period

Figure 7–62 shows IRQ generation waveform.

Figure 7–62. IRQ Generation Waveform



7.13.2.8 Alarm Interrupt

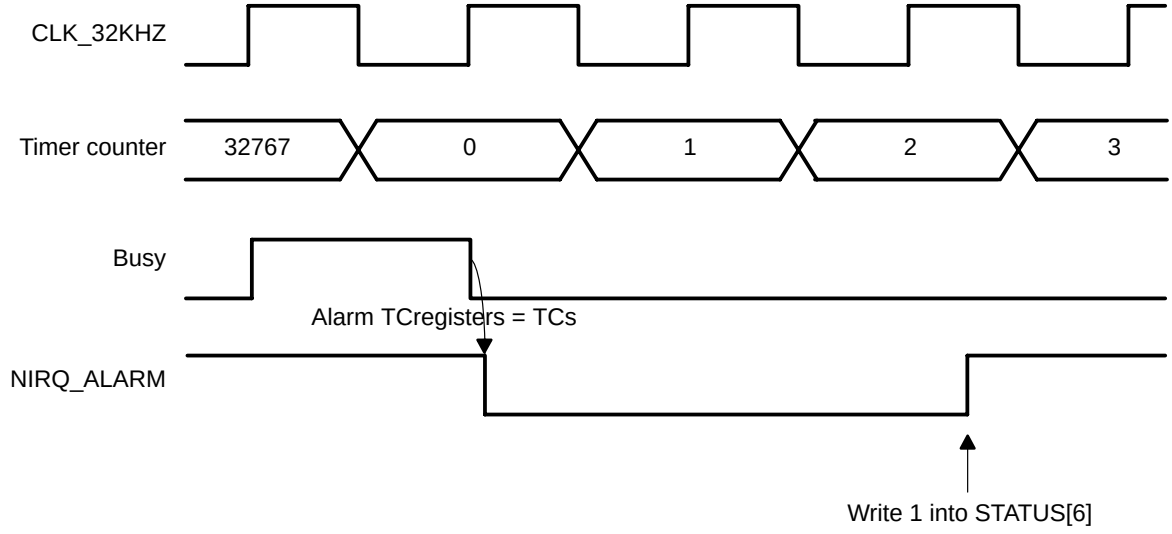
IRQ_ALARM_CHIP interrupt can be generated when the time set into time and calendar alarm registers is exactly the same as in the time and calendar registers (see Figure 7–63).

This interrupt is then generated if the IT_ALARM bit of the interrupts 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 7–63. IRQ Alarm Interrupt Waveform



7.13.2.9 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 (see Figure 7–64).

Autocompensation is enabled by the AUTO_COMP_EN bit in the RTC_CTRL register.

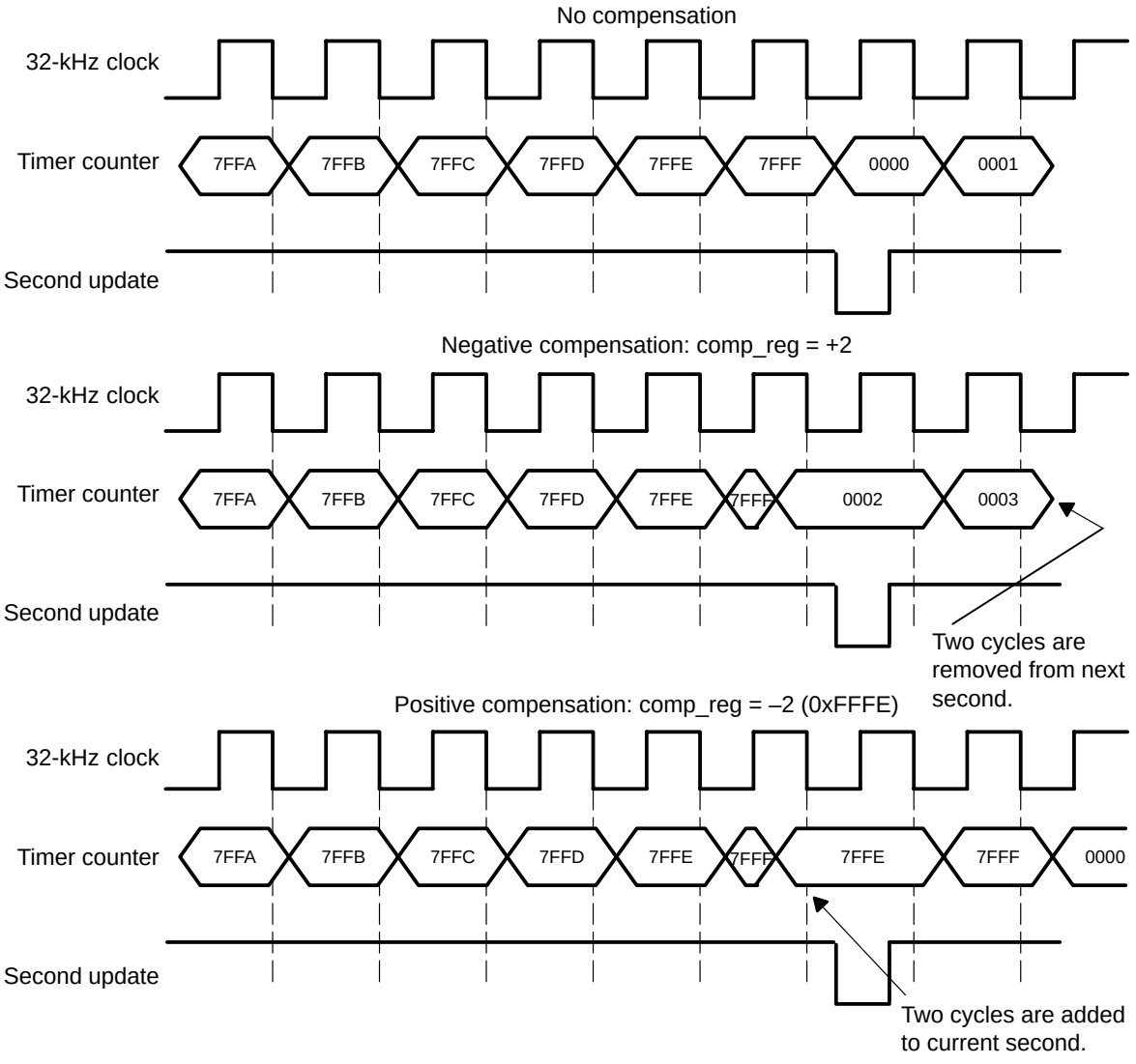
If the COMP_REG value is positive, compensation occurs after the second change event. COMP_REG cycles are removed from the next second.

If the COMP_REG value is negative, compensation occurs before the second change event. COMP_REG cycles are added to the current second.

This enables compensation with one 32-kHz period accuracy each hour.

Figure 7–64 summarizes positive and negative compensation effect.

Figure 7–64. Positive and Negative Compensation Effect



7.13.3 Register Descriptions and Mapping

Table 7–125 lists the RTC registers. Table 7–126 through Table 7–143 describe the register bits.

Table 7–125. RTC Registers

Register	Description	Size	Access	Base Address	Offset Address
SECONDS_REG	Seconds	8 bits	R/W	FFFB:4800	0x00
MINUTES_REG	Minutes	8 bits	R/W	FFFB:4800	0x04
HOURS_REG	Hours	8 bits	R/W	FFFB:4800	0x08
DAYS_REG	Days	8 bits	R/W	FFFB:4800	0x0C
MONTHS_REG	Months	8 bits	R/W	FFFB:4800	0x10
YEARS_REG	Years	8 bits	R/W	FFFB:4800	0x14
WEEK_REG	Weeks	8 bits	R/W	FFFB:4800	0x18
Reserved		8 bits		FFFB:4800	0x1C
ALARM_SECOND_REG	Alarm seconds	8 bits	R/W	FFFB:4800	0x20
ALARM_MINUTES_REG	Alarm minutes	8 bits	R/W	FFFB:4800	0x24
ALARM_HOURS_REG	Alarm hours	8 bits	R/W	FFFB:4800	0x28
ALARM_DAYS_REG	Alarm days	8 bits	R/W	FFFB:4800	0x2C
ALARM_MONTHS_REG	Alarm months	8 bits	R/W	FFFB:4800	0x30
ALARM_YEARS_REG	Alarm years	8 bits	R/W	FFFB:4800	0x34
Reserved		8 bits	R/W	FFFB:4800	0x38
Reserved		8 bits	R/W	FFFB:4800	0x3C
RTC_CTRL_REG	RTC control	8 bits	R/W	FFFB:4800	0x40
RTC_STATUS_REG	RTC status	8 bits	R/W	FFFB:4800	0x44
RTC_INTERRUPTS_REG	RTC interrupts	8 bits	R/W	FFFB:4800	0x48
RTC_COMP_LSB_REG	RTC compensation LSB	8 bits	R/W	FFFB:4800	0x4C
RTC_COMP_MSB_REG	RTC compensation MSB	8 bits	R/W	FFFB:4800	0x50

Table 7–126. Seconds Register (SECONDS_REG)

Bit	Name	Function	R/W	Reset Value
7	Reserved		R	0
6–4	SEC1	2nd digit of seconds Range is 0 to 5	R/W	000
3–0	SEC0	1st digit of seconds Range is 0 to 9	R/W	0000

Table 7–127. Minutes Register (MINUTES_REG)

Bit	Name	Function	R/W	Reset Value
7	Reserved		R	0
6–4	MIN1	2nd digit of minutes Range is 0 to 5	R/W	000
3–0	MIN0	1st digit of minutes Range is 0 to 9	R/W	0000

Table 7–128. Hours Register (HOURS_REG)

Bit	Name	Value	Function	R/W	Reset Value
7	PM_nAM		Only used in PM_AM mode (otherwise 0)	R	0
		0	AM		
		1	PM		
6	Reserved			R	0
5–4	HOUR1		2nd digit of hours Range is 0 to 2	R/W	00
3–0	HOUR0		1st digit of hours Range is 0 to 9	R/W	0000

Table 7–129. Days Register (DAYS_REG)

Bit	Name	Function	R/W	Reset Value
7–6	Reserved		R	0
5–4	DAY1	2nd digit of days Range from 0 to 3	R/W	00
3–0	DAY0	1st digit of days Range from 0 to 9	R/W	0001

Table 7–130. Months Register (MONTHS_REG)

Bit	Name	Function	R/W	Reset Value
7–5	Reserved		R	000
4	MONTH1	2nd digit of months Range from 0 to 1	R/W	0
3–0	MONTH0	1st digit of months Range from 0 to 9	R/W	0001

Note: Usual notation for month value: 01: January, 02: February, ... 12: December

Table 7–131. Years Register (YEARS_REG)

Bit	Name	Function	R/W	Reset Value
7–4	YEAR1	2nd digit of years Range from 0 to 9	R/W	0000
3–0	YEAR0	1st digit of years Range from 0 to 9	R/W	0000

Table 7–132. Weeks Register (WEEKS_REG)

Bit	Name	Function	R/W	Reset Value
7–3	Reserved		R	00000
2–0	WEEK	1st digit of days in a week Range from 0 to 6	R/W	000

Table 7–133. Alarm Seconds Register (ALARM_SECONDS_REG)

Bit	Name	Function	R/W	Reset Value
7	Reserved		R	0
6–4	ALARM_SEC1	2nd digit of seconds Range from 0 to 5	R/W	000
3–0	ALARM_SEC0	1st digit of seconds Range from 0 to 9	R/W	0000

Table 7–134. Alarm Minutes Register (ALARM_MINUTES_REG)

Bit	Name	Function	R/W	Reset Value
7	Reserved		R	0
6–4	ALARM_MIN1	2nd digit of minutes Range from 0 to 5	R/W	000
3–0	ALARM_MIN0	1st digit of minutes Range from 0 to 9	R/W	0000

Table 7–135. Alarm Hours Register (ALARM_HOURS_REG)

Bit	Name	Value	Function	R/W	Reset Value
7	ALARM_PM_nAM		Only used in PM_AM mode (otherwise 0)	R	0
		0	AM		
		1	PM		
6	Reserved			R	0
5–4	ALARM_HOUR1		2nd digit of hours Range from 0 to 2	R/W	00
3–0	ALARM_HOUR0		1st digit of hours Range from 0 to 9	R/W	0000

Table 7–136. Alarm Days Register (ALARM_DAYS_REG)

Bit	Name	Function	R/W	Reset Value
7–6	Reserved		R	00
5–4	ALARM_DAY1	2nd digit for days Range from 0 to 3	R/W	00
3–0	ALARM_DAY0	1st digit for days Range from 0 to 9	R/W	0001

Table 7–137. Alarm Months Register (ALARM_MONTHS_REG)

Bit	Name	Function	R/W	Reset Value
7–5	Reserved		R	000
4	ALARM_MONTH1	2nd digit of months Range from 0 to 1	R/W	0
3–0	ALARM_MONTH0	1st digit of months Range from 0 to 9	R/W	0001

Table 7–138. Alarm Years Register (ALARM_YEARS_REG)

Bit	Name	Function	R/W	Reset Value
7–4	ALARM_YEAR1	2nd digit of years Range from 0 to 9	R/W	0000
3–0	ALARM_YEAR0	1st digit of years Range from 0 to 9	R/W	0000

Table 7–139. RTC Control Register (RTC_CTRL_REG)

Bit	Name	Reset Value	Function	R/W
7	Reserved			R
6	RTC_DISABLE†	0	RTC enabled	R/W
		1	RTC disabled (no 32-kHz clock)	
5	SET_32_COUNTER‡	0	No action	R/W
		1	Sets the 32-kHz counter with COMP_REG (14:0) value	
4	TEST_MODE	0	Functional mode	R/W
		1	Test mode (autocompensation is enabled when 32-kHz counter reaches its end)	
3	MODE_12_24§	0	24-hour mode	R/W
		1	12-hour mode (PM/AM mode)	
2	AUTO_COMP	0	Autocompensation disabled	R/W
		1	Autocompensation enabled	
1	ROUND_30S¶	0	No update	R/W
		1	Rounding enabled	
0	STOP_RTC	0	RTC is frozen	R/W
		1	RTC is running	

† RTC_DISABLE can be written to 1 to disable RTC clock. Behavior is unpredictable if this bit is reset to 0 after having been set to 1.

‡ SET_32_COUNTER must only be used when the RTC is frozen. The set operation is asynchronous, which means the RTC counter is frozen to the compensation value as long as this bit is set. The correct sequence: reset STOP_RTC (freezes RTC), set SET_32_COUNTER bit, set STOP_RTC (launches RTC), then reset SET_32_COUNTER bit (3 register writes).

§ You can switch between the two modes at any time without disturbing the RTC. Read or write is always performed using the current mode.

¶ ROUND_30S is a toggle bit: MPU can only write 1, RTC clears it. If the MPU sets this bit and then reads it, the MPU reads 1 until the rounding to the closet minute is performed at the next second.

Table 7–140. RTC Status Register (RTC_STATUS_REG)

Bit	Name	Value	Function	R/W	Reset Value
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	R	0
		1	RTC is running		
0	BUSY	0	Updating event in more than 15 µs	R	0
		1	Updating event		

† POWER_UP is set by a reset and cleared by writing 1 to this bit.
‡ The alarm interrupt keeps its low level until the MPU writes 1 in the ALARM bit of this register. The timer interrupt is a low-level pulse (15 µs duration).
§ The STOP_RTC signal is synchronized on the 32-kHz clock, so only 1 clock period can elapse between the write to STOP_RTC and the RTC actually being stopped. The RUN bit shows the actual state of the RTC.

Table 7–141. RTC Interrupts Register (RTC_INTERRUPTS_REG)

Bit	Name	Value	Function	R/W	Reset Value
7–4	Reserved			R	0000
3	IT_ALARM		Enable one interrupt when the alarm value is reached (time and calendar alarms) by the time and calendars.	R/W	0
2	IT_TIMER		Enable periodic interrupt	R/W	0
		0	Interrupt disabled		
		1	Interrupt enabled		

Note: The MPU must respect the busy period to prevent spurious interrupt.

Table 7–141. RTC Interrupts Register (RTC_INTERRUPTS_REG) (Continued)

Bit	Name	Value	Function	R/W	Reset Value
1–0	EVERY		Interrupt period	R/W	00
		0	Every second		
		1	Every minute		
		2	Every hour		
		3	Every day		

Note: The MPU must respect the busy period to prevent spurious interrupt.

Table 7–142. RTC Compensation LSB Register (RTC_COMP_LSB_REG)

Bit	Name	Function	R/W	Reset Value
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. That means that to add one 32-kHz oscillator period every hour, MPU must write FFFF into RTC_COMP_MSB_REG and RTC_COMP_LSB_REG. To remove one 32-kHz oscillator period every hour, MPU must write 0001 into RTC_COMP_MSB_REG and RTC_COMP_LSB_REG. The 7FFF value is forbidden.

Table 7–143. RTC Compensation MSB Register (RTC_COMP_MSB_REG)

Bit	Name	Function	R/W	Reset Value
7–0	RTC_COMP_MSB	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. That means that to add one 32-kHz oscillator period every hour, MPU must write FFFF into RTC_COMP_MSB_REG and RTC_COMP_LSB_REG. To remove one 32-kHz oscillator period every hour, MPU must write 0001 into RTC_COMP_MSB_REG and RTC_COMP_LSB_REG. The 7FFF value is forbidden.

7.14 USB Host Controller Overview

The OMAP5910 device implements a three-port USB host controller that is compatible with the USB Revision 1.1 specification and the *Open Host Controller Interface Specification for USB (OHCI) Revision 1.0a*. It provides USB host connectivity for USB low-speed (1.5M bit/sec maximum) and full-speed (12M bit/sec maximum) devices.

For details, see Chapter 14, *Universal Serial Bus Host*.

7.15 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 this must be done before any transmit and receive from the module is performed. The mode is assumed static during operation of the device. From a timing perspective, both the 1-Wire and the HDQ protocols use HDQ timing.

7.15.1 Functional Description

The module is intended to work with both the 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 necessitates an external pullup.

An open-drain configuration is used on the wire. Therefore, the HDQ pin only actively drives low and goes to the high-impedance state otherwise, allowing an external pullup resistor to pull the wire high.

A control bit selects whether the HDQ or the 1-Wire protocol is to be used. This bit should not be modified during active data cycles on the interface. Therefore, it is recommended that the bit be only modified as part of boot up configuration. For the design, the bit is assumed static. By default, the configuration complies with the HDQ spec.

7.15.1.1 Receive and Transmit Operation

The receive and the transmit operations are performed with respect to the timing that is 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.

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, two 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, one command/address byte is sent to the slave, and the slave returns a byte of data.

The master implementation is a byte engine. The sending of the ID, command/address, and data is the responsibility of the firmware. The master engine provides for 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 OMAP5910 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 happens 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 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. Each of the interrupt conditions cannot be individually masked.

The following sequence must be performed by the programmer 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.
    a) The hardware sends the byte from the TX data register.
    b) The time-out bit always is cleared in a write, because the hardware has no acknowledge mechanism from the slave.
    c) 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.
    d) 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.
    a) 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 go bit is written by the software.
    b) If a time-out occurs, a time-out bit is set in the control and status register.
    c) 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.
    d) 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 if RX was complete or whether there was a time-out.

6) Software does a read of the RX buffer register to retrieve the read data from slave.

7) Repeat for each successive byte.

In HDQ 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.

1-Wire Mode

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 INIT bit and the GO bit in the control and status register. A presence detect is indicated in the appropriate bit of the register. If no presence is received, then 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 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. Each of the interrupt conditions cannot be individually masked.

The following sequence must be performed by the programmer 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.
    a) The hardware sends the one byte of the TX write data register.
    b) The time-out bit is always cleared in a write.
    c) 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.
    d) 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.
    a) 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.

b) The time-out bit is always cleared in a read.

c) 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.

d) 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 slave.

9) Repeat for each successive byte.

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.

7.15.1.2 Timing Diagrams

Figure 7–65 through Figure 7–67 show the timing diagram for the read, reset, and write. In HDQ, the reset pulse only contains the initialization and not the presence pulse. The timing required for the various signals are specified in the *BQ2023*.

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 7–65. Read Timing Diagram

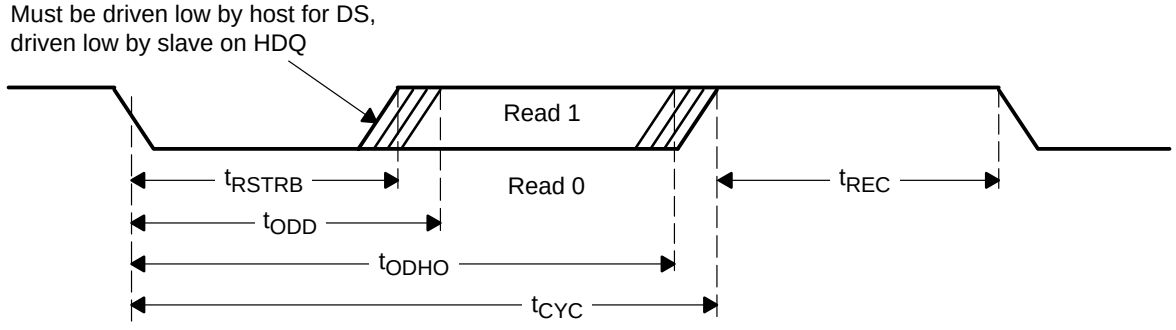


Figure 7–66. Reset Timing Diagram

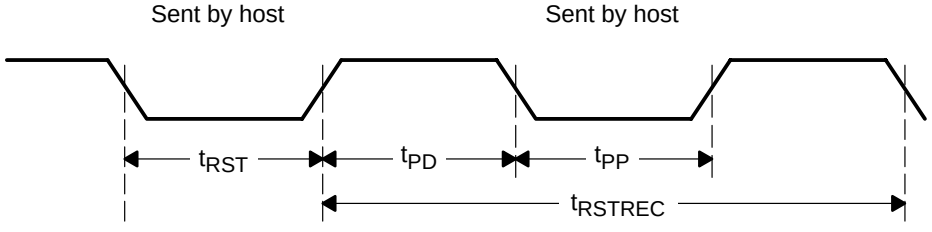
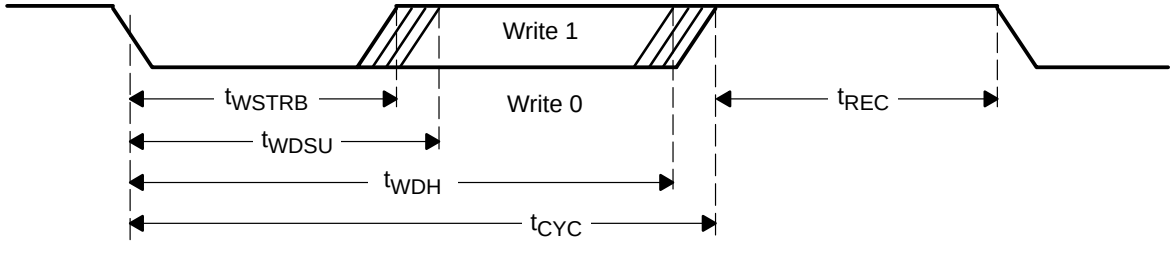
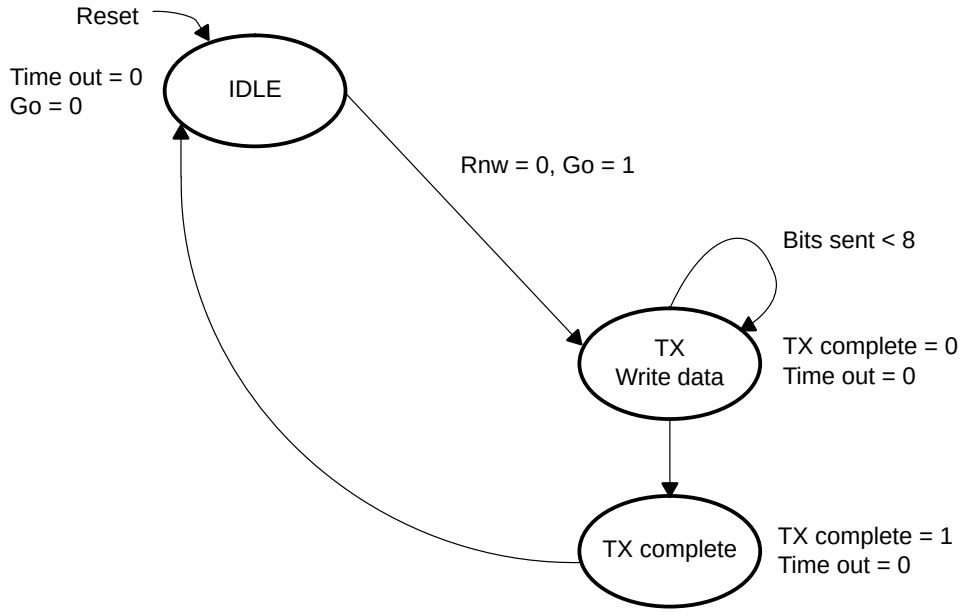


Figure 7–67. Write Timing Diagram



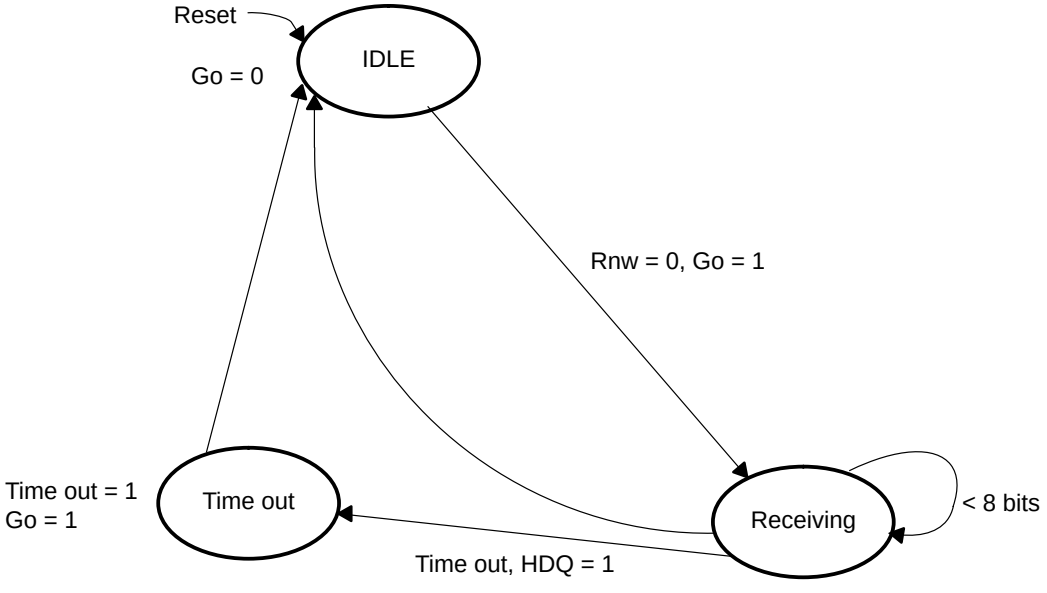
7.15.1.3 Write State Diagram

Figure 7–68. Write State Machine #1



7.15.1.4 Read State Diagram

Figure 7–69. Read State Machine #1



7.15.1.5 Status Flags

The status flags are provided in the status register, which 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.

7.15.1.6 Interrupts

The following interrupt status is provided by the module:

- 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 specified time. In HDQ mode, bit is cleared at beginning of read command.

Only one interrupt is generated to the MPU, based on any of the above interrupt status conditions. A read to the interrupt status register clears all 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).

7.15.2 Power-Down Mode

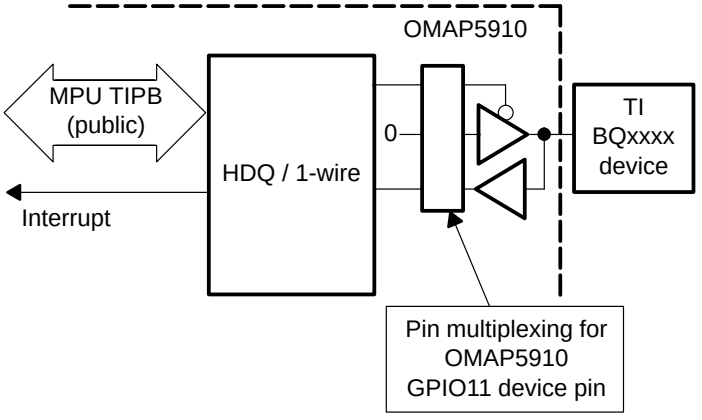
Writing to the appropriate bit in the control and status register shuts the clock to the state machine. The state machines are reset when the clock is disabled, and if any transaction is being performed it is aborted into the reset state. The register values are not affected by disabling the clock. No 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.

7.15.3 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 TI/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 doing the correct sequencing in the command registers. The module only implements 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 7–70. HDQ and 1-Wire Overview



7.15.4 Software Interface

The mapping of registers to the TI peripheral bus (TIPB) address signals is shown in Table 7–144 and Table 7–145. The base address for the HDQ registers is FFFB:C000.

No synchronization is provided by the hardware 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 and 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.
- Polling of the interrupt status register is not allowed by software to determine if 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 7–144. Memory Map Summary

Address	Name	Type
8h00	TX write data	R/W
8h04	RX receive buffer	R
8h08	Control and status	R/W
8h0C	Interrupt status, read to clear	R/W
Other	Writes ignored; reads return 0	Reserved

Table 7–145. Registers Accessible From TIPB

Bit	Name	Value	Description	Access Type at Address	Reset Value
31–24	TX write data		Reserved—read aliased to bits 7:0, writes ignored	Read/Write at 8h00	0000h
23–16			Reserved—read aliased to bits 7:0, writes ignored		
15–8			Reserved—read aliased to bits 7:0, writes ignored		
7–0			Write data (Used in both HDQ and 1-Wire modes)		
31–24	RX buffer register		Reserved—read aliased to bits 7:0, writes ignored	Read only at 8h04	Unknown (read only when data is ready)
23–16			Reserved—read aliased to bits 7:0, writes ignored		
15–8			Reserved—read aliased to bits 7:0, writes ignored		
7–0			Next received character.		
31–24	Interrupt status register		Bit is set to 1 if cause of interrupt. Read of the clears all interrupts that have been set. Reserved—read aliased to bits 7:0, writes ignored	Read only/ read to clear at 8h0C	
23–16			Reserved—read aliased to bits 7:0, writes ignored		
15–8			Reserved—read aliased to bits 7:0, writes ignored		
7–3			Reserved—read 0s, writes ignored		0
2			TX completed		0
1			Read complete		0
0			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.		0

Table 7–145. Registers Accessible From TIPB (Continued)

Bit	Name	Value	Description	Access Type at Address	Reset Value
31–24	Control and status register		Reserved—read aliased to bits 7:0, writes ignored	at 8h08	
23–16			Reserved—read aliased to bits 7:0, writes ignored		
15–8			Reserved—read aliased to bits 7:0, writes ignored		
7			Single-bit mode for 1-Wire	R/W	0
6			Interrupt mask	R/W	0
		0	Disable interrupts		
		1	Enable interrupts		
5			Power-down mode	R/W	0
		0	Disable clocks		
		1	Enable clocks		
4			Go bit	R/W	0
			Write 1 to send the appropriate commands.		
			Bit returns to 0 after the command is complete.		
3			Presence detect received, 1-Wire mode only.	R	0
		0	Not detected		
		1	Detected		
2			Write 1 to this bit, and set the GO bit, to send Initialization pulse.	R/W	0
			Bit returns to 0 after pulse is sent.		
1			R/W bit (determines if next command is read or write)	R/W	0
		0	Write		
		1	Read		
0			Mode	R/W	0
		0	HDQ		
		1	1-Wire		

7.16 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). This count value can then be used by system-level software 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.

7.16.1 Features

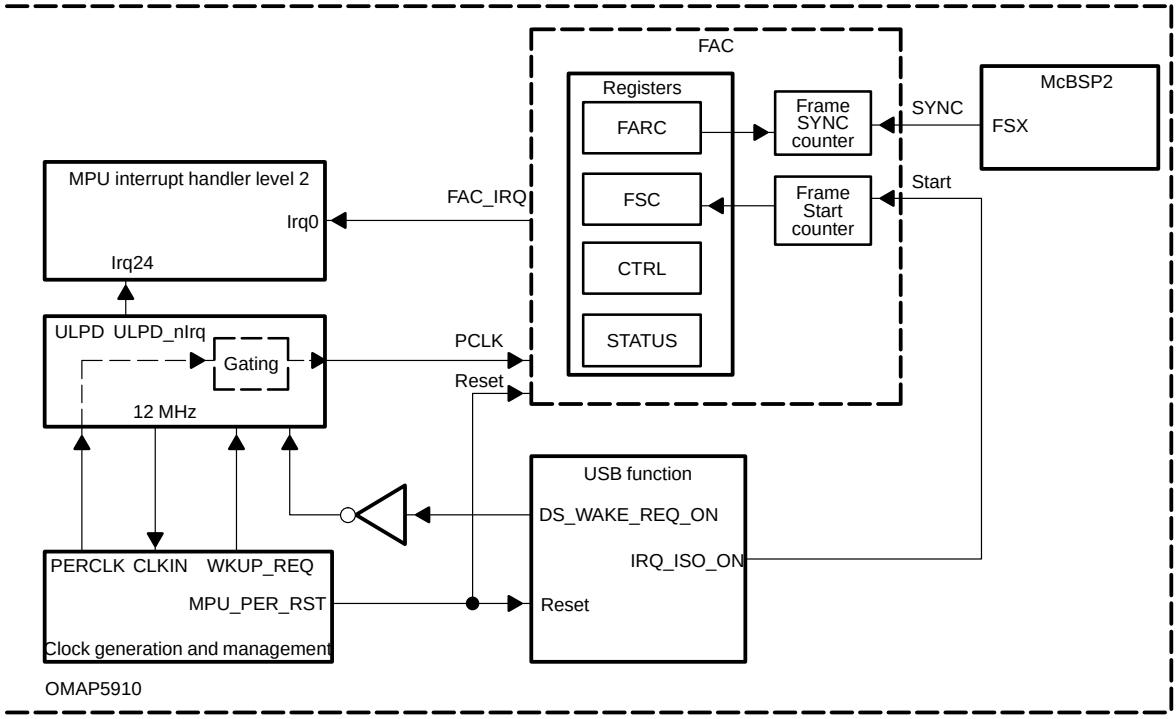
The frame adjustment counter (FAC) is a module that consists of a frame-synchronization capture pin and a frame-start capture pin. The respective count values can then be used by system software to adjust the duration of the two time domains with respect to each other to reduce the overflow and underflow.

A frame-adjustment reference count register (FARC) is programmed with the number of frame-synchronization pulses over which the frame-start pulses are to be counted. A frame-start count register (FSC) is updated with the number of frame-start rising edges that occur during the programmable FARC period. A control and configuration register (CTRL) allows you to put the module 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 zero, the counters are reset immediately even though the count is not finished. The software can use this bit as a software reset. Additionally, there is a status register (STATUS) containing a FSC_FULL bit that indicates to system software if FSC has been read subsequent to the last FSC update.

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)

Figure 7–71. FAC Top-Level Diagram



7.16.2 Synchronization and Counter Control

Because frame-start and frame-synchronization signals are from different time domains, the FAC module synchronizes these two signals 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.

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 7–72 and Figure 7–73 show the synchronization logic and the counter hookup.

Figure 7–72. FAC Module Counters and Clock Synchronization

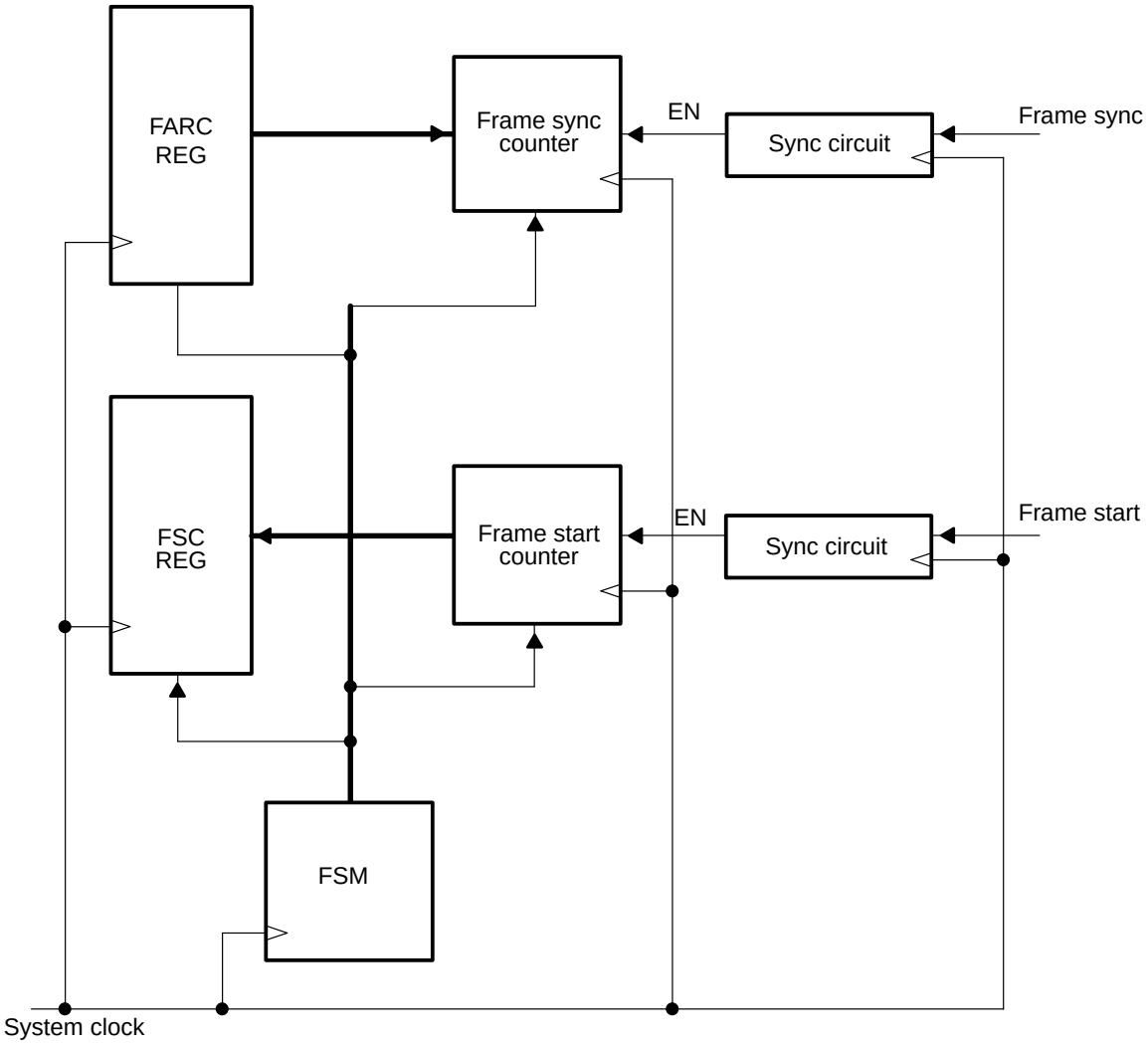


Figure 7–73. Synchronization Circuit for Frame Synchronization and Frame Start Signals

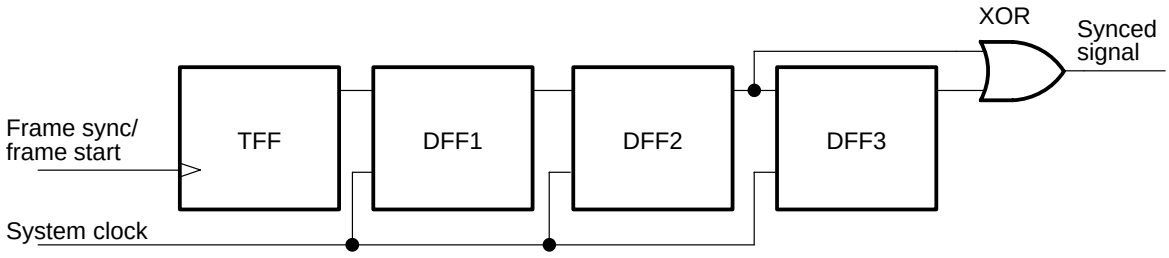
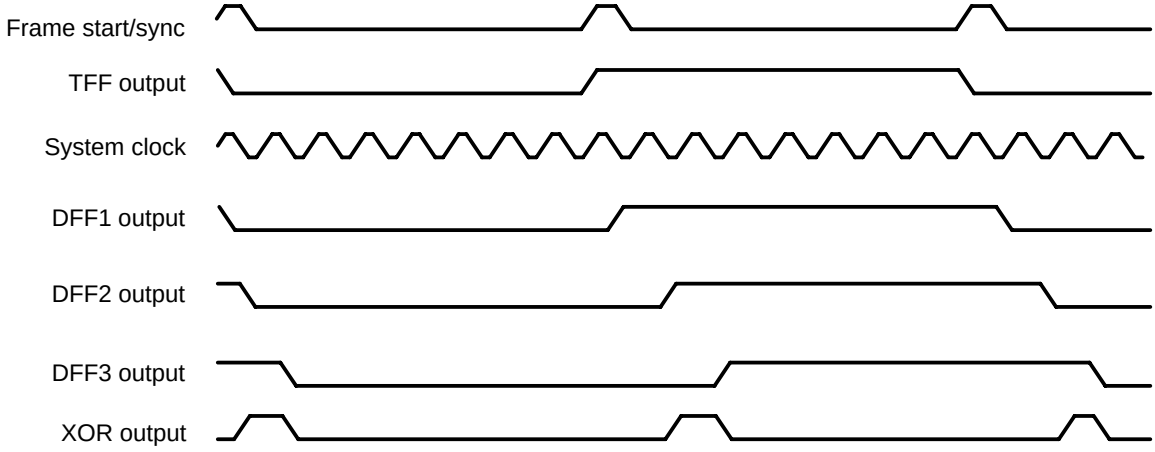


Figure 7–74 shows the actual waveforms of at the output of each flip-flop and the XOR output.

Figure 7–74. Synchronization Circuit Waveforms



7.16.3 FAC Interrupt

The FAC generates 1 interrupt, FAC_IRQ (in halt mode when the FARC value is met), connected to the MPU level 2 interrupt handler, line 0 (level-sensitive)

7.16.4 FAC Clocks and Reset

The FAC works with a clock (PCLK), which is 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 (for more detail, see Chapter 15, *Clock Generation and System Reset Management*).

The MPU TIPB reset (MPU_PER_RST) resets the FAC.

7.16.5 Software Interface

Table 7–146 lists the FAC registers. Table 7–147 through Table 7–150 describe the register bits.

Table 7–146. 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 zero disables the count operation.

Table 7–147. Frame Adjustment Reference Count Register (FARC)

Bit	Name	Function	Reset Value
15–0	FARC	16-bit value in the range 0-65536: 0 = disable counting	0

The frame-start count register (FSC) 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 two modes. When the CNT bit in the control and configuration register (CTRL) is set to one, the counting is in continuous mode and this register is periodically updated (every time the frame adjustment reference count is met) with the new count value. If CNT is zero, 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 register (CTRL). If this bit is set to one, an interrupt is generated when the FSC register is updated. Since 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 zero. When the FSC is read or RUN bit in control register is set to zero, the interrupt signal is reset to one. When in IN_ENABLE bit is set to zero, no interrupt is generated. The interrupt can be enabled or disabled for both continuous mode and halt mode.

Table 7–148. Frame Start Count Register (FSC)

Bit	Name	Function	Reset Value
15–0	FS	16-bit value	0

The control and configuration register (CTRL) is a read/write register used to configure the module. The RUN bit is used to enable 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 zero. When the RUN bit is set to zero, the frame-start counter, the frame-synchronization counter, and the FSC register are reset to zero. The software reset also clears the status register FSC_FULL bit to zero. 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 reset the counter starts counting again, regardless of whether the FSC register has been read or not.

Table 7–149. FAC Control and Configuration Register (CTRL)

Bit	Name	Function	Reset Value
15–3	Reserved		0
2	INT_ENABLE	When this bit is set to a 1, an interrupt is generated when FSC is updated. If this bit is set to a 0, no interrupt is generated. 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
1	RUN	Enables operation of the counter. When this bit is set to zero, the frame start counter, the frame-synchronization counter, and the FSC are reset to zero. Any pending interrupt also is cleared when RUN is set to zero.	0
0	CNT	1: Continuous mode: Periodically updates FSC value each time the frame-adjustment reference count is met. 0: Halt mode: Updates FSC value when the frame-adjustment reference count is met and halts operation until FSC is read.	0

The status register (STATUS) is a read/write register that contains an interrupt status bit.

Table 7–150. FAC Status Register (STATUS)

Bit	Name	Function	Reset Value
15–1	Reserved		0
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 zero.	0

Chapter 8

DSP Private Peripherals

This chapter describes the following DSP private peripherals and their associated memory and mapping:

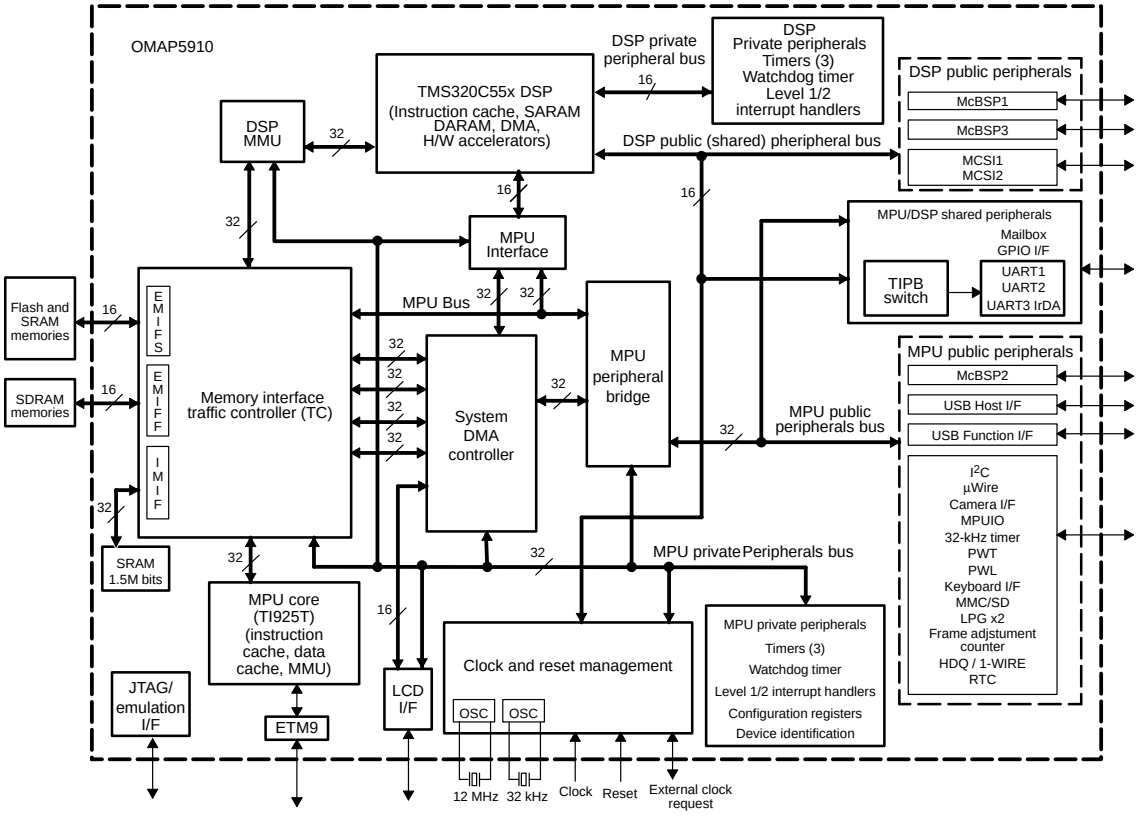
- Timers
- Watchdog timer
- Interrupt handlers

Topic **Page**

8.1 DSP Private Peripherals

Figure 8–1 shows the OMAP5910 device with the DSP private peripherals highlighted.

Figure 8–1. Highlight of DSP Peripherals

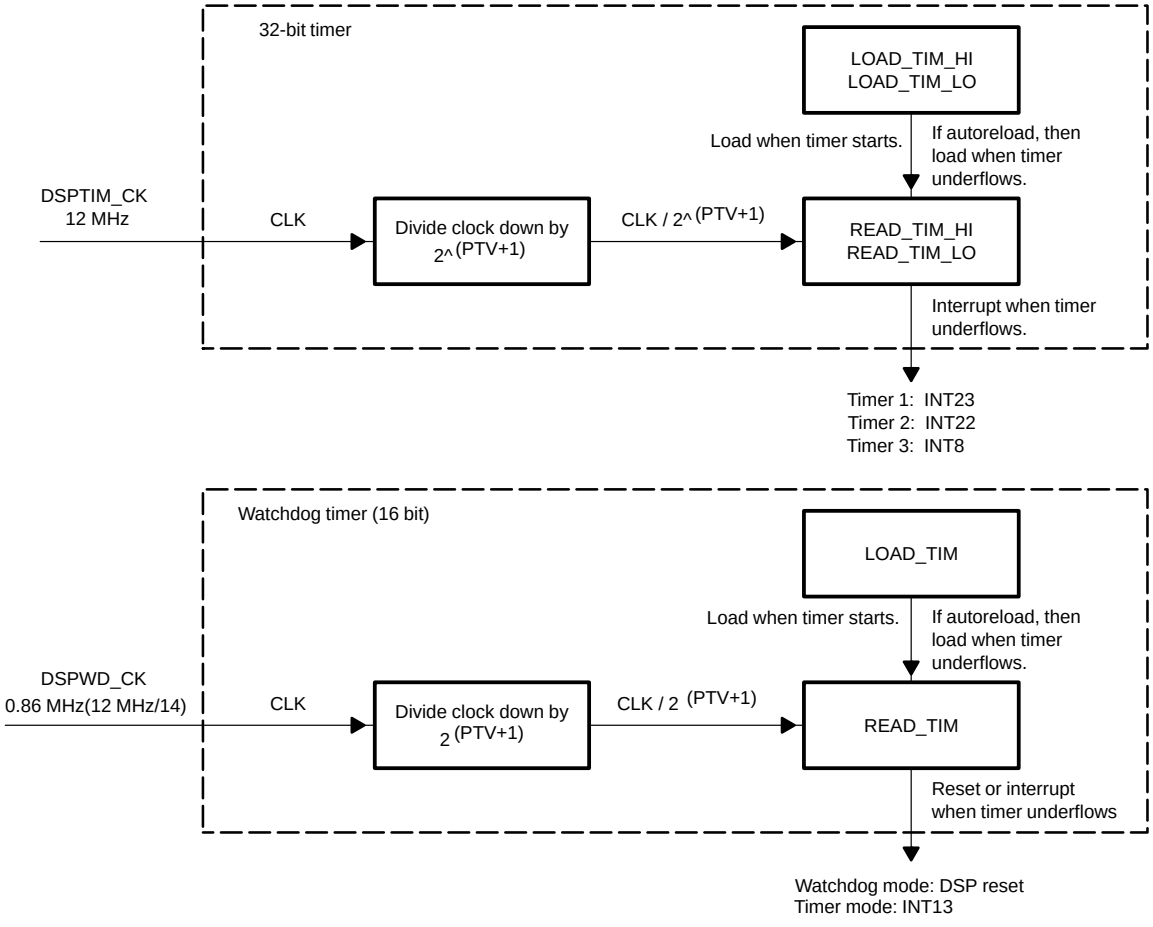


8.2 Timers

Figure 8–2 shows the DSP timers in detail.

Three 32-bit timers are available for general-purpose housekeeping functions. The counters/timers are configurable either in autoreload or in one-shot mode with on-the-fly read capability. Each timer generates a corresponding level 1 interrupt to the DSP when equal to zero, as shown in Table 8–1, *Timer Interrupt Levels*.

Figure 8–2. DSP Timers



8.2.1 Timer Interrupt Levels

Table 8–1. Timer Interrupts Levels

Timer	Corresponding Level 1 Interrupt	Required Sensitivity Setup
1	INT23	Edge
2	INT22	Edge
3	INT8	Edge

The timers are counters that receive a dedicated clock from clock generator module #2 (either CK_REF or CK_GEN2 output divided by 2). This clock can be prescaled (divided down) as controlled by the prescale clock timer value (PTV) field of the control timer register (shown in Table 8–2, *PTV Divisors: 32-Bit Timers*).

Table 8–2. PTV Divisors: 32-Bit Timers

PTV	Divisor
0	2
1	4
2	8
3	16
4	32
5	64
6	128
7	256

The timer interrupt period is calculated as follows:

$$t_{int} = t_{clk} \times (LOAD\_TIM + 1) \times 2^{(PTV+1)}$$

where:

t_{clk} is the clock period of the input clock.

The load timer register (LOAD_TIM) holds the value loaded when the timer passes through 0 or when it starts.

PTV is the prescaler field located in the control timer register.

Table 8–3 shows the characteristics for all three timers for different input frequencies.

8.2.2 Timer Characteristics

Table 8–3. Timer Characteristics

Input Clock	t_{clk}, Clock Period	LOAD_TIM	t_{int}, Timer Interrupt Period for PTV = 0	t_{int}, Timer Interrupt Period for PTV = 7
12 MHz	83.3 ns	0001	333 ns	42.67 µs
12 MHz	83.3 ns	FFFF (max interrupt period)	10.92 ms	1398.1 ms

If LOAD_TIM = 0 and AR (auto-reload mode) = 1, the timer is always 0 and can never decrement. Here the timer interrupt is asserted and stays asserted all the time. Since the timer interrupts are edge-senditive, only one interrupt is recognized because there is one initial edge, and then the interrupt is asserted constantly.

8.2.3 Programming the Timer

To start a timer, set the start timer bit (ST) of the control timer to 1. Reset the bit to 0 to stop the timer. When the timer stops, the decrementer content is frozen.

Set the autoreload bit (AR) of the control timer to 0 to have the timer decrement from the loaded value down to zero and then stop. Set the AR to 1 to have the timer continue.

- A new value from the load register is loaded into the timer when it passes though zero or when it starts.
- An interrupt is produced when the corresponding timer is equal to zero.

To avoid undefined results, do not program the PTV, AR, or load register while the timer is running.

The timer value is held in the value timer register (VALUE_TIM) and can be read while the timer is running or stopped.

The load timer and read timer registers are actually split into two 16-bit portions; therefore, two 16-bit accesses from the DSP are required. To properly read the read timer register, access the upper 16 bits first, then the lower 16 bits. The DSP can access them through a single 32-bit access.

8.2.4 Timer Registers

Table 8–4 lists the timer registers. Table 8–5 through Table 8–12 describe the register bits.

Table 8–4. Timer Registers

Register Name	Description	R/W	Size (Bits)	Offset	Reset Value
CNTL_TIMER	Control timer	R/W	16	0x00	0x0002
LOAD_TIM_HI	Load timer—high	W	16	0x02	0xFFFF
LOAD_TIM_LO	Load timer—low	W	16	0x03	0xFFFF
READ_TIM	Read timer	R	16	0x02	0xFFFF
TIMER_MODE	Timer mode	R/W	16	0x04	0x8000

Table 8–5. Control Timer Register (CNTL_TIMER)

Bit	Name	Value	Descriptions	Reset Value
15–8	Unused			
7	SOFT		This bit is used with the FREE bit to determine peripheral state when a breakpoint is encountered. Used in emulation mode.	0
		0	Peripheral halts immediately, either retaining or discarding current state.	
		1	Peripheral stops after completion of current task.	
6	FREE		This bit is used with the SOFT bit to determine peripheral state when a breakpoint is encountered. Used in emulation mode.	0
		0	SOFT bit selects emulation mode.	
		1	Peripheral clock runs free regardless of the SOFT bit.	
5	CLOCK_ENABLE		External timer clock enable	0
4–2	PTV		Prescale clock timer value	0
1	AR	0	0: One-shot timer	0
		1	Autoreload timer	

Table 8–5. Control Timer Register (CNTL_TIMER) (Continued)

Bit	Name	Value	Descriptions	Reset Value
0	ST	0	0: Stop timer	0
		1	Start timer With one-shot mode selected (AR = 0), bit is automatically reset by internal logic when timer equals 0.	

The load timer register (LOAD_TIM) is a 32-bit register (see Table 8–6 and Table 8–7). The data width of the TIPB connected to this peripheral is only 16 bits. Therefore, two 16-bit TIPB write transactions are needed to load the load timer register (LOAD_TIM).

If the DSP is ready to load the load timer register (LOAD_TIM), it can send a 32-bit write request (with offset address of 04) to the DSPI. The DSPI has the capability of converting this 32-bit request into two 16-bit TIPB writes on the DSP TIPB.

Table 8–6. Load Timer High Register (LOAD_TIM_HI)

Bit	Name	Description	Reset Value
15–0	LOAD_TIM_HI	This value is loaded when the timer passes through 0 or when it starts. LOAD_TIM_HI is the same as LOAD_TIM[31:16].	Undefined

Table 8–7. Load Timer Low Register (LOAD_TIM_LO)

Bit	Name	Description	Reset Value
15–0	LOAD_TIM_LO	This value is loaded when the timer passes through 0 or when it starts. LOAD_TIM_LO is the same as LOAD_TIM[15:0].	Undefined

The read timer register (READ_TIM) is a 32-bit register (see Table 8–8 and Table 8–9). The data width of the TIPB connected to this peripheral is only 16 bits. So two 16-bit TIPB read transactions are required to read the value of the read timer register (READ_TIM). Also, note that since the TIPB strobe is completely asynchronous with the timer_clk, synchronization is done to make sure that the read timer register (READ_TIM) value is not read while it is being incremented.

Table 8–8. Read Timer High Register (VALUE_TIM_HI)

Bit	Name	Description	Reset Value
15–0	VALUE_TIM_HI	Value of timer. This is the same as READ_TIM[31:16],	Undefined

Table 8–9. Read Timer Low Register (VALUE_TIM_LO)

Bit	Name	Description	Reset Value
15–0	VALUE_TIM_LO	Value of timer. This is the same as READ_TIM[15:0] at the time of the last TIPB read to READ_TIM_HI.	Undefined

The following sequence must be followed to read the READ_TIM register properly:

1) Perform a TIPB read transaction to read the upper 16 bits of the read timer register (READ_TIM) (offset = 8). When the read timer register (READ_TIM) is read and synchronized, the upper 16 bits are driven onto the data bus of the TIPB and the lower 16 bits of the read timer register (READ_TIM) are stored in a temporary register.

2) Perform a TIPB read transaction to read the lower 16 bits of the read timer register (READ_TIM) (offset = 10). During this read, the value of the temporary register is forwarded onto the TIPB bus instead of reading the read timer register (READ_TIM) again. This is done because the TIMER can change value between the two TIPB read transactions.

Therefore, to read the value of the read timer register (READ_TIM) correctly, the first TIPB read access must be to the upper 16 bits (that is, offset = 8), followed by TIPB read access to the lower 16 bits (that is, offset = 10 (decimal)).

Note:

If the DSP is ready to read the read timer register (READ_TIM), it can send a 32-bit read request (with offset address of 08) to the DSPI. The DSPI can convert this 32-bit request into two 16-bit TIPB writes on the DSP TIPB, thus resolving all the above sequencing issues.

Table 8–10. DSP Timer 1 Registers

Register Name	Description	R/W	Size (Bits)	Word Address	Reset Value
CNTL_TIMER1	Timer control register	R/W	16	0x2800	0x0000
LOAD_TIM1	Value that must be loaded into timer when timer passes through 0	W	32	0x2802	U
READ_TIM1	Timer counter	R	32	0x2804	U

Table 8–11. DSP Timer 2 Registers

Register Name	Description	R/W	Size (Bits)	Word Address	Reset Value
CNTL_TIMER2	Timer control register	R/W	16	0x02C00	0x0000
LOAD_TIM2	Value that must be loaded into timer when it passes through 0	W	32	0x02C02	U
READ_TIM2	Timer counter	R	32	0x02C04	U

Table 8–12. DSP Timer 3 Registers

Register Name	Description	R/W	Size (Bits)	Word Address	Reset Value
CNTL_TIMER3	Timer control register	R/W	16	0x03000	0x0000
LOAD_TIM3	Value that must be loaded into timer when it passes through 0	W	32	0x03002	U
READ_TIM3	Timer counter	R	32	0x03004	U

8.3 Watchdog Timer

When powered up, the timer defaults to the watchdog timer for the DSP. This configuration requires that the user program or the OS periodically write to the count register before the counter underflows to prevent the timer from generating a reset to the DSP. This function detects user programs stuck in infinite loops, which can result in program control loss or runaway programs.

Table 8–13. Watchdog Timer Interrupt

Timer	Corresponding Level 1 Interrupt
WD	INT13

Note:

By default, this timer is configured as a watchdog timer and, unless disabled or updated properly, generates a reset of the DSP approximately every 19 seconds. If, during system development, you encounter an unexpected reset every 19 seconds or so, this is probably the reason.

CAUTION

Be certain to disable the watchdog timer before placing the DSP processor in deep sleep mode. It must not be left configured as a watchdog timer.

The watchdog timer underflow resets the DSP. If the input clock is 12 MHz and the watchdog timer values are left at their power-up state (the value loaded into the load timer register (LOAD_TIM) is set to the maximum value of 0xFFFF), reset occurs in approximately 19 seconds.

The watchdog timer uses the clock derived from the clock frequency generation module for synchronization. This clock is [input clock]/14. When configured as a watchdog timer, the prescaler field (PTV) of the control timer register (CNTL_TIMER) is fixed to 7. When configured as a general-purpose timer, the prescaler field can range from 0 to 7. The time from writing a new value to counter underflow is:

between $256*t_{clk}$ to $16{,}777{,}216*t_{clk}$

where t_{clk} = [input clock]/14 for a clock frequency of 12 MHz

and the reset time is: 298 $\mu s < t > 19$ s).

Table 8–14. PTV Divisors: Watchdog Timer

PTV	Divisor
0	2
1	4
2	8
3	16
4	32
5	64
6	128
7	256

The timer period is defined by:

- The value of the PTV, which is forced to 7 if the timer is in watchdog mode
- The value of the load register

The timer interrupts period is:

$t_{int} = t_{clk} \times (LOAD\_TIM + 1) \times 2^{(PTV+1)}$

where t_{clk} is the clock period of the input clock.

Table 8–15 shows the characteristics of the watchdog timer for different input frequencies:

Table 8–15. Watchdog Timer Characteristics

Input Clock	t_{clk}, Clock Period†	LOAD_TIM	t_{int}, Timer Interrupt Period, PTV = 7
12 MHz	1167 ns	0001	597.34 µs
12 MHz	1167 ns	FFFF (max interrupt period)	19.57 s

† The 12-MHz clock is divided by 14.

If LOAD_TIM = 0 and AR (auto-reload mode) = 1, the timer is always 0 and can never decrement. Here the timer interrupt is asserted and stays asserted all the time. Since the timer interrupts are edge-senditive, only one interrupt is recognized because there is one initial edge, and then the interrupt is asserted constantly.

8.3.1 Programming the Watchdog Timer in Watchdog Mode

On power up, the watchdog timer is enabled in the watchdog mode and the value loaded into the load timer register is set to the maximum value (0xFFFF). This gives the user a duration of 16,777,216 * t_{clk} to change the timer mode or write a new value (different from 0xFFFF) into the load timer register.

The user program or the OS must write periodically to the load timer register (LOAD_TIM) before the counter underflows. The new loaded value must be different from the previous because the write is taken into account only if the newly loaded value is different from the previous one. Due to internal sequencing, the user must wait for three timer clock periods before writing a new value into the load timer register. If the input clock is 12 MHz, three timer clock periods are approximately 3.5 µs.

You can not disable the watchdog timer by only clearing bit 15 (WATCHDOG) of the timer mode register (TIMER_MODE). Once the timer has been configured as a general-purpose timer, it can be switched back to watchdog mode by writing a 1 to bit 15 (WATCHDOG) of the timer mode register (TIMER_MODE). In this case, the value loaded into the load timer register (LOAD_TIM) is set to the maximum value (0xFFFF) as on power-up.

In watchdog mode, the control timer register (CNTL_TIMER) must not be used. The watchdog timer can not be stopped by clearing bit 7 (ST), and the prescale value is 7 regardless of the PTV field. Autoreload and one-shot do not apply, because if the counter underflows the processor is reset and the watchdog registers are reinitialized.

8.3.2 Programming the Watchdog Timer in Timer Mode

To start a timer, set the start timer (ST) bit of the control timer register to 1. Reset the bit to 0 to stop the timer. When the timer stops, the decrementer content is frozen.

Set the autoreload (AR) bit of the control timer register to 0 to decrement from the loaded value down to zero and then stop. Set the AR bit to 1 to continue. A new value from the load register is loaded into the timer when it passes though zero or when it starts. An interrupt is produced when the corresponding timer is equal to zero.

To avoid undefined results, do not program the PTV and AR bits or the load register while the timer is running. You can set the PTV bits to values other than 7 when the watchdog timer is in timer mode.

The timer value is held in the read timer register and can be read while the timer is running or stopped.

The base word address for watchdog timer is 0x003400.

8.3.3 Watchdog Timer Registers

Table 8–16 shows the DSP watchdog timer registers. Table 8–17 through Table 8–20 describe the register bits.

Table 8–16. DSP Watchdog Timer Registers

Register Name	Description	R/W	Size (Bits)	Address	Reset Value
CNTL_TIMER	Control timer	R/W	16	x003400	0x0002
LOAD_TIM	Load timer	W	16	x003402	0xFFFF
READ_TIM	Read timer	R	16	x003402	0xFFFF
TIMER_MODE	Timer mode	R/W	16	x003404	0x8000

Table 8–17. Control Timer Register (CNTL_TIMER)

Bit	Name	Value	Description	Reset Value
15–12	Reserved			
11–9	PTV		Prescale clock timer value	0
8	AR	0	One-shot timer	0
		1	Autoreload timer	
			If one-shot mode is selected (AR = 0), this bit is automatically reset by internal logic when timer is equal to 0.	
7	ST	0	Stop timer	0
		1	Start timer	
6–2			Reserved	
1	FREE	0	Enables emulation suspend function; timer can be frozen during emulation halt on the DSP.	1
		1	Timer runs free, regardless of emulation halt condition.	
0			Reserved	

Table 8–18. Load Timer Register (LOAD_TIM)

Bit	Name	Description	Reset Value
15–0	LOAD_TIM	General-purpose timer. This value is loaded when timer passes through 0 or when it starts. Watchdog timer. Reload timer with this value.	FFFF

Table 8–19. Read Timer Register (READ_TIM)

Bit	Name	Description	Reset Value
15–0	VALUE_TIM	Value of timer	FFFF

Table 8–20. Timer Mode (TIMER_MODE)

Bit	Name	Value	Description	Reset Value
15	WATCHDOG		Write access: 1: Switch back from timer mode to watchdog Writing a 0 in this bit has no effect. Read access: Status of timer mode:	1
		0	Timer is used as a general-purpose counter.	
		1	Timer is used as a watchdog timer.	
14–8			Reserved	
7–0	WATCHDOG_DIS		Write access only: Writing a predefined sequence (0xF5 followed by 0xA0) in this field disables watchdog functionality. After having received 0xF5, if the second write access is different from 0xA0, the DSP core is reset.	1

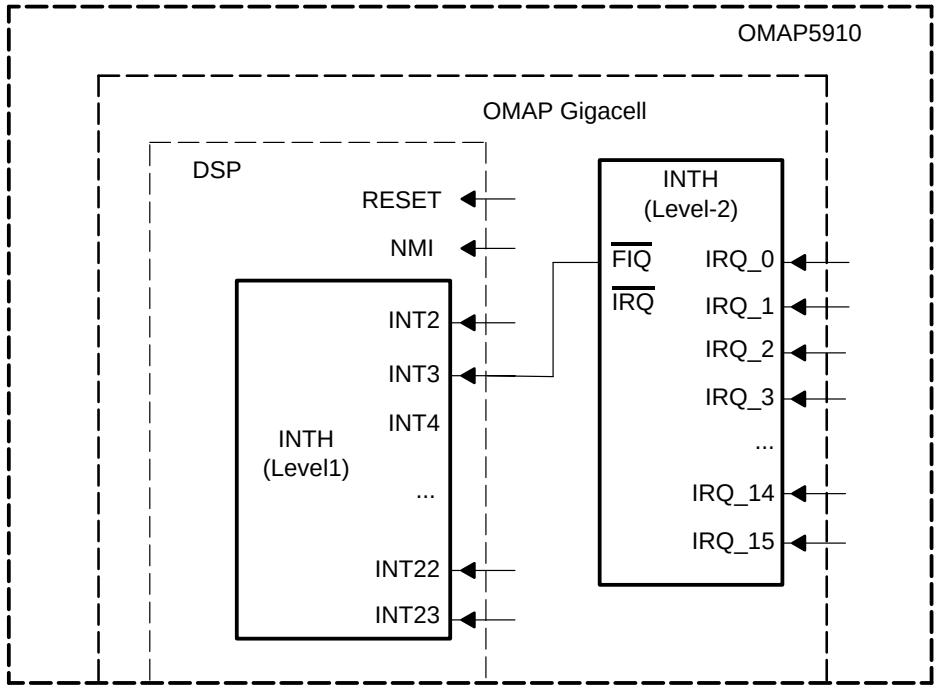
8.4 Interrupt Handlers

The interrupt handler handles interrupts generated by modules and peripherals (DMA controller, MMU, watchdog timer, timers, software interrupt, etc.). The interrupt handler processes, on a programmable basis, edge-triggered or level-sensitive interrupts. All interrupts are maskable (that is, individually enabled and disabled) with internal registers except for reset and NMI. The interrupt sources information can be read back. Interrupt priority is programmable to provide flexibility for different applications. All of these interrupts are routed to the DSP core interrupt inputs.

Interrupts are handled through two cascaded interrupt controllers. One is the level 1 handler and is inside the DSP core. The second is the level 2 handler and is external to the DSP and functions similarly to the MPU interrupt handler.

- The 22 level 1 interrupts are handled by the DSP internal interrupt controller provided by the DSP core (see Table 8–21, *DSP Level 1 Interrupt Mapping*).
- The 16 level 2 interrupts are handled by the external interrupt controller, cascaded into INT3 of the DSP internal interrupt controller.

Figure 8–3. DSP Interrupt Handler Cascade



8.4.1 Level 1 Interrupts

The DSP level 1 interrupt controller receives interrupts from peripherals and sends them to the DSP core (see Table 8–21). The TI peripheral bus is responsible for prioritizing, capturing, and synchronizing interrupts, before sending them to the DSP. The level 1 interrupt controller has a nonmaskable interrupt (NMI) and 22 maskable interrupts. Of the 22 maskable interrupts, 21 are peripheral interrupts and the remaining one is an MPU interrupt.

Level 1 DSP interrupts must be at least two DSP_CLK cycles long in order for the DSP to recognize it. To ensure that this requirement is met, the DSP is provided with and internal hardware module called the DSP interrupt interface (described in Section 8.5).

Table 8–21. Level 1 Interrupt Mapping

Level 1 Interrupt	Priority	DSP Interrupt	Vector Location	DSP IFR_bit/IMT_bit (26:0)
RESET	0		FFFF00	
NMI	1		FFFF08	
Emulator/Test	3	INT2	FFFF10	2
Level-2 INTH FIQ	5	INT3	FFFF18	3
TC_ABORT	6	INT4	FFFF20	4
MAILBOX 1	7	INT5	FFFF28	5
Reserved	9	INT6	FFFF30	6
GPIO	10	INT7	FFFF38	7
TIMER3	11	INT8	FFFF40	8
DMA_channel_1	13	INT9	FFFF48	9
MPU	14	INT10	FFFF50	10
Reserved	15	INT11	FFFF58	11
UART	17	INT12	FFFF60	12
WDGTIMER	18	INT13	FFFF68	13
DMA_channel_4	21	INT14	FFFF70	14
DMA_channel_5	22	INT15	FFFF78	15
EMIF	4	INT16	FFFF80	16
Local Bus	8	INT17	FFFF88	17

Table 8–21. Level 1 Interrupt Mapping (Continued)

Level 1 Interrupt	Priority	DSP Interrupt	Vector Location	DSP IFR_bit/IMT_bit (26:0)
DMA_channel_0	12	INT18	FFFF90	18
Mailbox 2	16	INT19	FFFF98	19
DMA_channel_2	19	INT20	FFFFA0	20
DMA_channel_3	20	INT21	FFFFA8	21
TIMER2	23	INT22	FFFFB0	22
TIMER1	24	INT23	FFFFB8	23

8.4.2 Level 2 Interrupts

The level 2 interrupt controller provides up to 16 prioritized and maskable interrupts to the DSP core.

The level 2 interrupt controller resides on the 16-bit TI peripheral bus. This module is clocked by the DSP_INTH_CK clock, which is fixed at half the CK_GEN2 frequency (see Chapter 15 for details). Configuration registers configure incoming interrupts as level-sensitive or edge-sensitive interrupts.

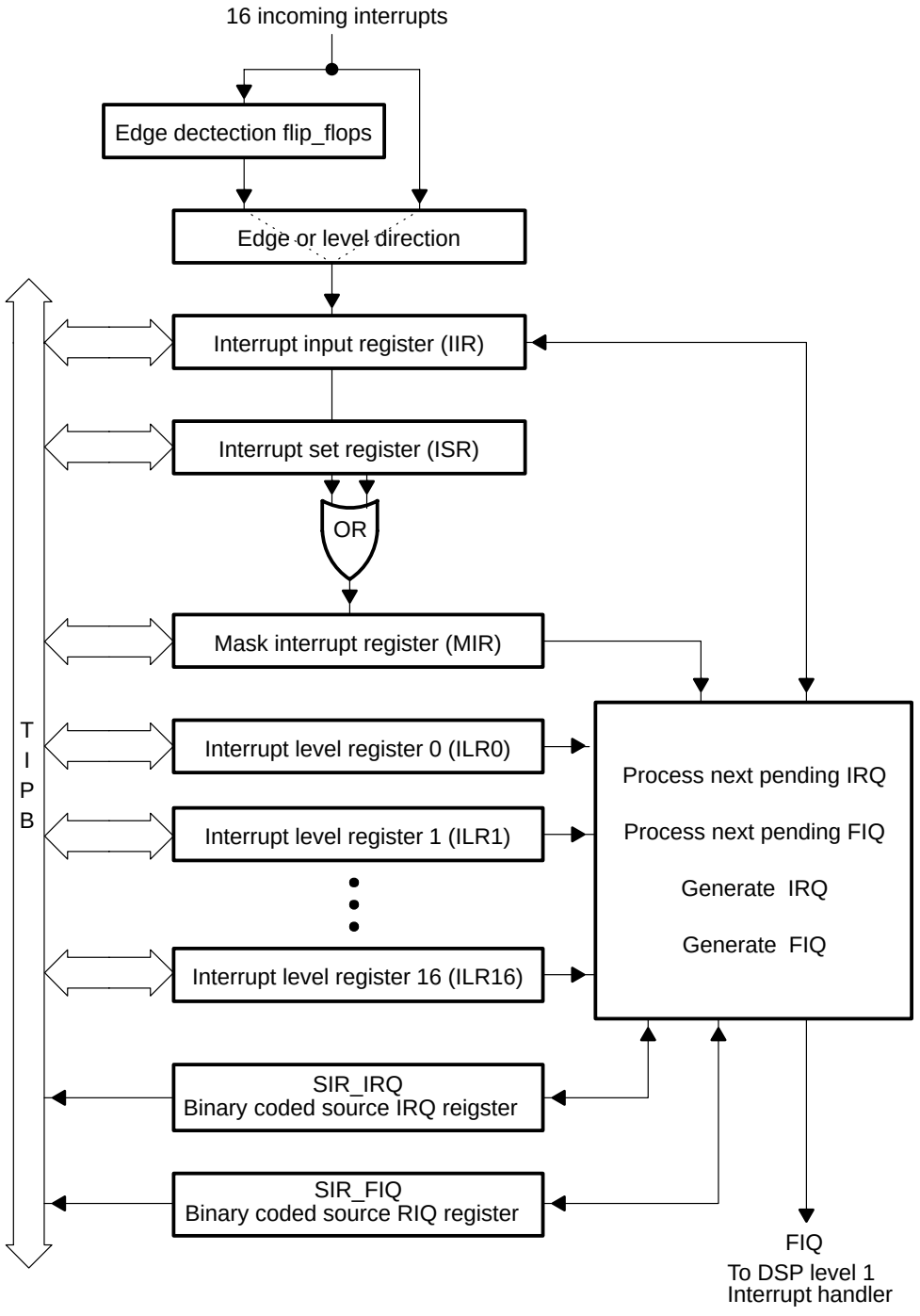
One interrupt-level register (ILR) is associated with each incoming interrupt. It assigns a priority to the corresponding interrupt,determines whether it is to be level- or edge-sensitive, and determines to which DSP interrupt (fast interrupt request (FIQ) or low priority interrupt request (IRQ)) the incoming interrupt goes. If several interrupts have the same priority level assigned, they are serviced in a predefined order.

All level 2 interrupts are routed to FIQ. IRQ output is unconnected.

The interrupt controller also provides a 16-bit software interrupt register. This 16-bit register corresponds to the same 16-bit external interrupt lines. By writing a 1 to the targeted bit, an interrupt is generated if the corresponding ILR is set to edge sensitive; otherwise, no interrupt is generated. External interrupt request and internal software request are ORed together first before being sent to the interrupt controller to be serviced. The software interrupt register is always read back with a zero. This allows simulation of external interrupts to test the corresponding interrupt driver by using the software interrupt mechanism at any time.

The FIQ outputs from the interrupt controller are reset by writing a 1 to the corresponding bit of the control register.

Figure 8–4. Level 2 Interrupt Control Flow



8.4.2.1 Interrupt Sequence

1) One or several incoming interrupts go down, setting the corresponding ITR bits.

2) At this time, two possibilities exist:

   - If there is only one active incoming interrupt and FIQ is not already active, the interrupt controller sends an FIQ.
   - When several incoming interrupts are active, the interrupt controller must determine which is the new interrupt to be serviced. It does so by comparing the priority level of an interrupt with the one held in a dedicated register N_FIQ and stores the interrupt with the highest priority in the dedicated register N_FIQ. It performs this comparison until all the active interrupts have been processed. If FIQ is not already active, the interrupt controller sends an FIQ.

3) When an FIQ is sent, the source interrupt encoded register (SIR_FIQ) is updated (indicating the interrupt contained in N_FIQ) and the priority resolver is reset (and restarted if necessary).

4) To determine which incoming interrupt has requested a DSP action, the source interrupt encoded register (SIR_FIQ) must be read. The register contains an encoded number that tells which interrupt lines are being serviced. After that, it runs the corresponding subroutine.

5) To finish the interrupt sequence, DSP software must first clear the interrupt bit in the interrupt input register (ITR) that is being serviced. This is done by writing a 0 to the corresponding bit in the interrupt input register (ITR) directly or by reading the source interrupt encoded register (SIR_FIQ). For a level-sensitive interrupt, the level must also be removed for the next interrupt to occur. Then set a dedicated bit (NEW_FIQ_AGR bit of the control register) in order to reset the FIQ output and the source encoded register (SIR_FIQ), thus allowing a new FIQ generation.

8.4.2.2 DSP Accessible Registers

DSP start word address (hex): 0x004800

Bus width: 16 bits

DSP address of a register = Start address + offset TIPB address

Table 8–22 lists the interrupt handler level 2 registers. Table 8–23 through Table 8–29 describe the register bits.

Table 8–22. Interrupt Handler Level 2 Registers

Register Name	Description	Default Value	Read/ Write	Size	DSP Address
ITR	Interrupt	00000000	R/W	16 bits	0x004800
MIR	Interrupt mask	FFFFFFFF	R/W	16 bits	0x004802
SIR_IRQ	Interrupt encoded source (IRQ)	00	R	4 bits	0x004804
SIR_FIQ	Interrupt encoded source (FIQ)	00	R	4 bits	0x004806
CONTROL_REG	Interrupt control	0	R/W	2 bits	0x004808
ISR	Software interrupt set	00000000	R/W	16 bits	0x00480A
ILR0	Interrupt priority level bit 0	00	R/W	6 bits	0x00480C
ILR1	Interrupt priority level bit 1	00	R/W	6 bits	0x00480E
ILR2	Interrupt priority level bit 2	00	R/W	6 bits	0x004810
ILR3	Interrupt priority level bit 3	00	R/W	6 bits	0x004812
ILR4	Interrupt priority level bit 4	00	R/W	6 bits	0x004814
ILR5	Interrupt priority level bit 5	00	R/W	6 bits	0x004816
ILR6	Interrupt priority level bit 6	00	R/W	6 bits	0x004818
ILR7	Interrupt priority level bit 7	00	R/W	6 bits	0x00481A
ILR8	Interrupt priority level bit 8	00	R/W	6 bits	0x00481C
ILR9	Interrupt priority level bit 9	00	R/W	6 bits	0x00481E
ILR10	Interrupt priority level bit 10	00	R/W	6 bits	0x004820
ILR11	Interrupt priority level bit 11	00	R/W	6 bits	0x004822
ILR12	Interrupt priority level bit 12	00	R/W	6 bits	0x004824
ILR13	Interrupt priority level bit 13	00	R/W	6 bits	0x004826
ILR14	Interrupt priority level bit 14	00	R/W	6 bits	0x004828
ILR15	Interrupt priority level bit 15	00	R/W	6 bits	0x00482A

Table 8–23. Interrupt Input Register (ITR)

Bit	Name	Type	Reset Value
15	IRQ_15	R/W	0
:::	:::	:::	:::
0	IRQ_0	R/W	0

In the event of an edge-sensitive interrupt, ITR stores an incoming interrupt. When the DSP accesses the SIR_FIQ register, the bit corresponding to the interrupt that has requested the DSP action is reset.

The DSP can also clear each bit individually by writing a 0 to the corresponding bits at the ITR address. A 1 bit keeps its previous value. If the individual bit is cleared just before the DSP unmasks the interrupts, the interrupt is not processed.

The DSP reads this register. If an incoming interrupt is edge sensitive, the read value corresponds to the value held in the storage element.

> IRQ (FIQ) output and SIR_IRQ (SIR_FIQ) registers are reset only if the bit of ITR register corresponding to the interrupt that requested DSP action is already cleared or masked.
>
> The time when this ITR bit is reset depends on the sensitivity of the incoming interrupt. In case of an edge-sensitive interrupt, the IT register bit is cleared when reading SIR_IRQ (SIR_FIQ) register. Otherwise, it is reset when the corresponding interrupt line becomes inactive (low).
>
> For a level-sensitive interrupt, the level must be removed before the write to the control register. Otherwise, the interrupt controller is not reset for a new interrupt.

Table 8–24. Mask Interrupt Register (MIR)

Bit	Name	Description	Type	Reset Value
15	IRQ_15_MSK	Disable IRQ_15 interrupt	R/W	1
:::	:::	:::	:::	:::
0	IRQ_0_MSK	Disable IRQ_0 interrupt	R/W	1

Each incoming interrupt can be masked individually by this register by setting the corresponding bit to 1.

The mask interrupt register (MIR) operates after interrupt input register (ITR); this means that occurrences of incoming interrupts are always stored in interrupt input register (ITR).

Table 8–25. IRQ Binary-Coded Source Register (SIR_IRQ)

Bit	Name	Type	Reset Value
3–0	IRQ_NUM	R	0

This register saves software processing time by recognizing the interrupt number as being either an IRQ or FIQ request. Reading this register clears the corresponding bit in the interrupt input register (ITR) if the interrupt is set as edge-sensitive. This register will not normally be used since all level 2 DSP interrupts must be configured as FIQ to generate DSP interrupts because IRQ is not connected.

Table 8–26. FIQ Binary-Coded Source Register (SIR_FIQ)

Bit	Name	Type	Reset Value
3–0	FIQ_NUM	R	0

In order to save software processing time, this register indicates the interrupt number that has an IRQ or FIQ request. Reading this register clears the corresponding bit in the interrupt input register (ITR) if the interrupt is set as edge-sensitive.

Table 8–27. Interrupt Control Register (CONTROL_REG)

Bit	Name	Description	Type	Reset Value
1	NEW_FIQ_AGR	New FIQ agreement Writing a 1 resets FIQ output and clears source FIQ register. Enables a new FIQ generation, reset by internal logic. Corresponding bit of ITR must be cleared first.	R/W	0
0	NEW_IRQ_AGR	New IRQ agreement Writing a 1 resets IRQ output and clears source IRQ register. Enables a new IRQ generation, reset by internal logic. Corresponding bit of ITR must be cleared first. Note: All level 2 DSP interrupts must be configured as FIQ to generate DSP interrupts because IRQ is not connected.	R/W	0

The software interrupt set register is a 16-bit, read/write register. Writing a 1 to any bit generates an interrupt to the DSP if the corresponding ILR register is set as edge-triggered; otherwise, no interrupt is generated. A 0 is always returned from a read to this register. External interrupts are ORed with the software interrupts before they are sent to the mask interrupt register for interrupt masking.

Table 8–28. Interrupt Level Registers (ILR0...ILR15)

DSP Word Offset Address (hex)	Name	Corresponding Interrupt
0x0C	ILR_IRQ_0	IRQ_0
0x0E	ILR_IRQ_1	IRQ_1
:::	:::	:::
0x0C + (N–1)*2	ILR_IRQ_N–1	IRQ_N–1
:::	:::	:::
0x2A	ILR_IRQ_15	IRQ_15

Table 8–29. Interrupt Level Registers (ILR0...ILR15)

Bit	Name	Value	Description	Type	Reset Value
5–2	PRIORITY		Define 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	0
1	SENS_EDGE	0	The corresponding interrupt is falling-edge-sensitive.	R/W	0
		1	The corresponding interrupt is low-level-sensitive.		
0	FIQ	0	0: The corresponding interrupt is routed to IRQ.	R/W	0
		1	The corresponding interrupt is routed to FIQ. Note: Since IRQ is not connected, only the FIQ setting is useful. This bit must be set to 1 for the corresponding level 2 interrupt to cause a DSP interrupt.		

Note:

Assuming that all interrupts have the same priority level and they are active at the same at the same moment, the order of servicing is as follows: IRQ_15, IRQ_N–1, IRQ_N–2, …, IRQ_0.

8.4.2.3 Level 2 Interrupt Mapping

Table 8–30 shows the DSP level 2 interrupt mapping.

Table 8–30. DSP Level 2 Interrupt Mapping

Incoming Interrupts	Required Sensitivity Setup	Level 2 Interrupt
McBSP3 TX	Edge	IRQ_00
McBSP3 RX	Edge	IRQ_01
McBSP1 TX	Edge	IRQ_02
McBSP1 RX	Edge	IRQ_03
UART2	Level	IRQ_04
UART1	Level	IRQ_05
MCSI1 TX	Level	IRQ_06
MCSI1 RX	Level	IRQ_07
MSCI2 TX	Level	IRQ_08
MCSI2 RX	Level	IRQ_09
MCSI1 frame error	Level	IRQ_10
MCSI2 frame error	Level	IRQ_11
Reserved		IRQ_12
Reserved		IRQ_13
Reserved		IRQ_14
Reserved		IRQ_15

Level-sensitive interrupts are level-active; the interrupt line must remain asserted until it has been acknowledged.

Edge-triggered interrupts are edge-triggered; just an edge is required for generating the interrupt. The interrupt to the DSP is cleared upon reading of the interrupt registers or writing a 0 to the interrupt mask registers in the interrupt handler.

8.5 DSP Interrupt Interface

The DSP interrupt interface (DSP_INT_IF) augments the capability of the DSP interrupt processing by providing user-definable edge-triggered and level-sensitive implementations for each of the interrupt lines. This is necessary to allow edge-triggered interrupts, since the DSP level 1 interrupts must be active for greater than two DSP_CLK cycles to be recognized as being active. The DSP_INT_IF module is clocked by the DSP_INTH_CK clock, which which is fixed at half the CK_GEN2 frequency (see Chapter 15).

8.5.1 Functional Description

The implementation of each of the interrupt channels to the DSP interrupt handler is shown in Figure 8–5.

Each interrupt channel processes the incoming interrupt as both an edge-triggered interrupt and a level-sensitive interrupt. The decision of which process to use is made by the interrupt (N) edge-triggered enable input, which is bit 2^N of the edge-enable control register. If this bit is 1, the edge-triggered process path is chosen. If 0, the level-sensitive process path is chosen.

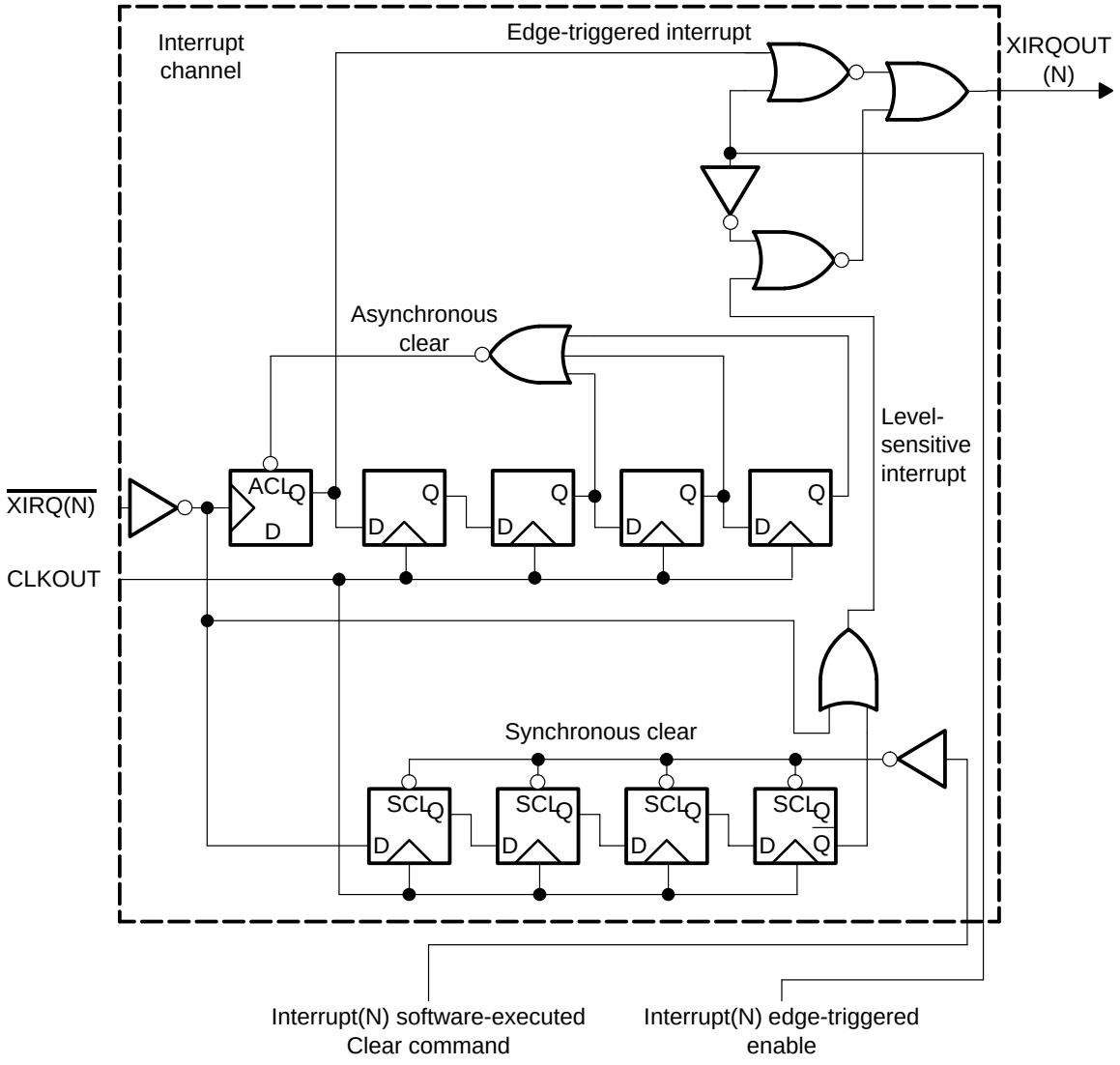
8.5.2 Edge-Triggered Interrupts

The edge-triggered interrupt process consists of an edge-registration flip-flop and a chain of four positive-edge triggered timing flip-flops. A negative transition (falling edge) on the incoming nXIRQ(N) interrupt line sets the edge-registration flip-flop to 1, and the output of this flip-flop is the edge-triggered interrupt. In addition to activating the output interrupt line nIRQ(N), this output also propagates through the four timing flip-flops. When the 1 output of the edge-registration flip-flop has propagated to the fourth flip-flop, an asynchronous reset is generated, clearing the edge-registration flip-flop and deactivating nIRQ(N).

nIRQ(N) then lasts between three and four DSP_INTH_CK clock periods, depending on when the asynchronous falling edge of nXIRQ(N) occurs with respect to the rising edge of DSP_INTH_CK clock. In OMAP, the frequency of DSP_INTH_CK clock is set to half the DSP_CLK frequency. The DSP requires the nIRQ(N) to transition from high to low and to be low for at least two DSP_CLK cycles, so that nIRQ(N) can be recognized. Also the DSP requires between two back-to-back interrupts on nIRQ(N), nIRQ(N) be high for at least one DSP_CLK cycle. nIRQ(N) is generated by the rising edge of DSP_INTH_CK clock and lasts for a minimum of three DSP_INTH_CK clock periods (which is actually six DSP_CLK cycles). The requirements on nIRQ(N) are clearly met.

When the edge-registration flip-flop is cleared by the asynchronous reset, two DSP_INTH_CK clock periods must expire before another negative edge transition can be registered. Thus successive negative transitions must be a minimum of six DSP_INTH_CK clock periods apart in time to be ensured of being recognized as two separate incidents. This minimal time does not take into account the processing time of the interrupts once recognized by the DSP processor, and this time must be taken into account to derive the minimum time between interrupts from a system perspective.

Figure 8–5. Interrupt Channel Implementation



8.5.3 Level-Sensitive Interrupts

The level-sensitive interrupt process is, in many ways, identical to the edge-triggered interrupt process. This process also uses a chain of four positive-edge triggered timing flip-flops, but this chain is driven by the inverted representation of the incoming interrupt nXIRQ(N). A negative transition (falling edge) on the incoming nXIRQ(N) line activates the output interrupt nIRQ(N) and must be held low for one DSP_INTH_CK cycle (which is equivalent to two DSP_CLK cycles) to be recognized by the DSP. Even when nXIRQ(N) remains at 0 for a time period exceeding three to four DSP_INTH_CK periods (depending on when the asynchronous falling edge of nXIRQ(N) occurs with respect to the rising edge of DSP_INTH_CK), the interrupt nIRQ(N) remains activated until the nXIRQ(N) is deactivated. However, the DSP recognizes this as only one interrupt, because there was only one falling edge of nIRQ(N).

8.5.4 Internal Registers

DSP word start address: 0x003800

Bit width: 16 bits

DSP word address of a register = start word address + offset address

The DSP_INT_IF has two control registers (one 16-bit and one 7-bit) and two clear command registers (one 16-bit and one 7-bit). The control registers are used exclusively for assigning edge-triggered/level-sensitive status to each of the 23 interrupt channels. The clear command registers are not actual physical registers, but rather a block of decoding logic that issues clear commands to the level-sensitive logic in each interrupt channel upon detecting a TIPB write transaction to an address that falls within the required address range.

The bit-alignment of interrupt channel assignments within the control register, the definition of the assignments, the default values at power turn-on, the address used to write to the control register, and the address used to read the content of the register are all presented in Table 8–31 through Table 8–34.

Table 8–31. Edge-Triggered/Level-Sensitive Control Register Low

Bit	Name	Value	Description	Type	Reset Value
15–0	CHx Trig/Level		This bit defines whether channel CHx is edge- or level-sensitive where CHx corresponds to interrupt channels nXIRQ[15:0]. Channels nXIRQ[15:0] correspond to the DSP level 1 interrupts IRQ17:2, respectively.	R/W	0
		0	CHx is level-sensitive.		
		1	CHx is edge-sensitive.		

Table 8–32. Edge-Triggered/Level-Sensitive Control Register High

Bit	Name	Value	Description	Type	Reset Value
15–8	Reserved				0
7	Host Interrupt Trig/Level		This bit defines whether the host interrupt is edge or level-sensitive.	R/W	
		0	NHOSTINT is level-sensitive.		
		1	NHOSTINT is edge-sensitive.		
6	NMI Trig/Level		This bit defines whether the nonmaskable interrupt is edge or level-sensitive. The NMI channel corresponds to the DSP NMI interrupt.	R/W	0
		0	NMI is level-sensitive.		
		1	NMI is edge-sensitive.		
5–0	CHx Trig/Level		This bit defines whether channel CHx is edge or level-sensitive, where CHx corresponds to interrupt channels nXIRQ[21:16]. Channels nXIRQ[21:16] correspond to the DSP level 1 interrupts IRQ23:18, respectively.	R/W	0
		0	CHx is level-sensitive.		
		1	CHx is edge-sensitive.		

8.5.4.1 Level-Sensitive Clear Commands (Write Only)

A write transaction issues a clear to those interrupt channels whose assigned bit in the 16-bit word being written is 1. Commands to clear interrupt channels are necessary for those channels assigned as level-sensitive interrupt channels. Figure 8–6 illustrates the alignment of the channel clear assignments within the 16-bit word written to the XIO interrupt processor and gives the permissible range of addresses over which the write can take place.

A write to the level-sensitive clear low register (RST_LVL_LO), whose offset address is 02, clears interrupts corresponding to nXIRQ[15:0].

Table 8–33. Level-Sensitive Clear Low Register (RST_LVL_LO)

Bit	Name	Value	Description	Type	Reset Value
15–0	Reset_CHx		Reset CHx if a 1 is written into RST_LVL_LO[x] and CHx is configured as level-sensitive interrupt, where CHx corresponds to interrupt channels nXIRQ[15:0]	0	0
		0	Do not reset CHx.		
		1	Reset interrupt channel CHx if level is configured as level sensitive.		

A write to the level-sensitive clear high register (RST_LVL_HI), whose offset address is 03, clears interrupts from interrupt channels [20:16], NMI, and HOSTINT.

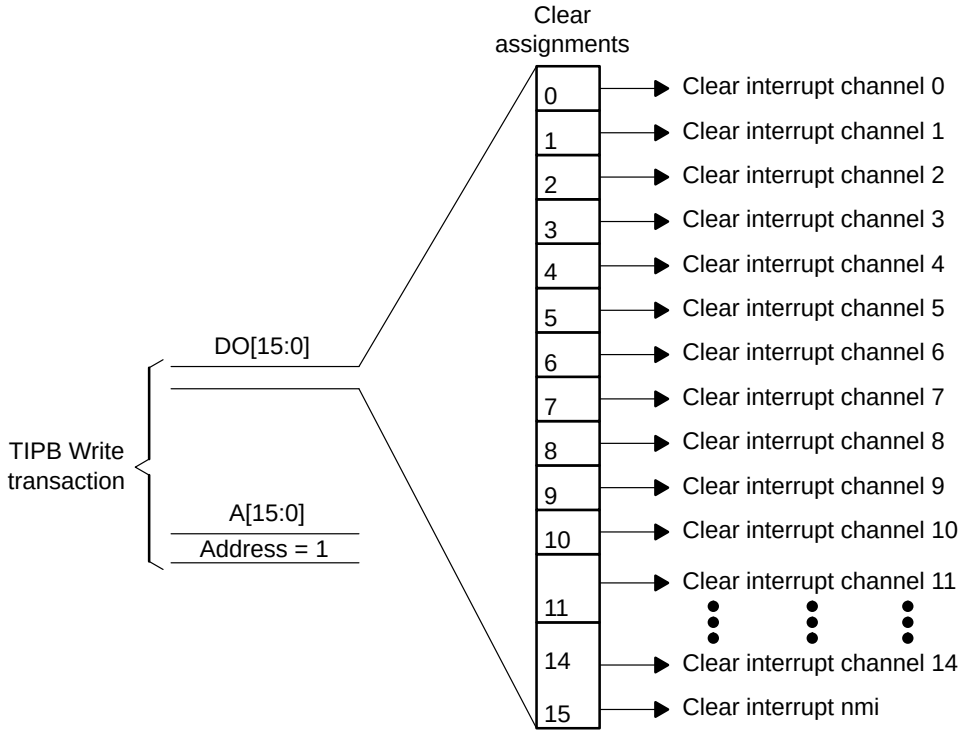
Table 8–34. Level-Sensitive Clear High Register (RST_LVL_HI)

Bit	Name	Value	Description	Type	Reset Value
7	Reset_NHOSTINT		Reset NHOSTINT channel if a 1 is written into this bit and NHOSTINT is configured as level-sensitive interrupt.		
		0	Do not reset NHOSTINT.		
		1	Reset NHOSTINT interrupt channel, if configured as level-sensitive interrupt.		
6	Reset_NMI		Reset NMI channel if a 1 is written into this bit and NMI is configured as level-sensitive interrupt.		
		0	Do not reset CHx.		
		1	Reset NMI interrupt channel if configured as level-sensitive interrupt.		

Table 8–34. Level-Sensitive Clear High Register (RST_LVL_HI) (Continued)

Bit	Name	Value	Description	Type	Reset Value
5–0	Reset_CHx		Reset CHx if a 1 is written into RST_LVL_LO[x] and CHx is configured as level-sensitive interrupt, where CHx corresponds to interrupt channels nXIRQ[20:16].		0
		0	Do not reset CHx.		
		1	Reset interrupt channel CHx if level is configured as level-sensitive.		

Figure 8–6. Level-Sensitive Interrupt Clear Commands



Chapter 9

DSP Public Peripherals

This chapter describes the DSP public peripherals for the OMAP5910 multimedia processor.

Topic **Page**

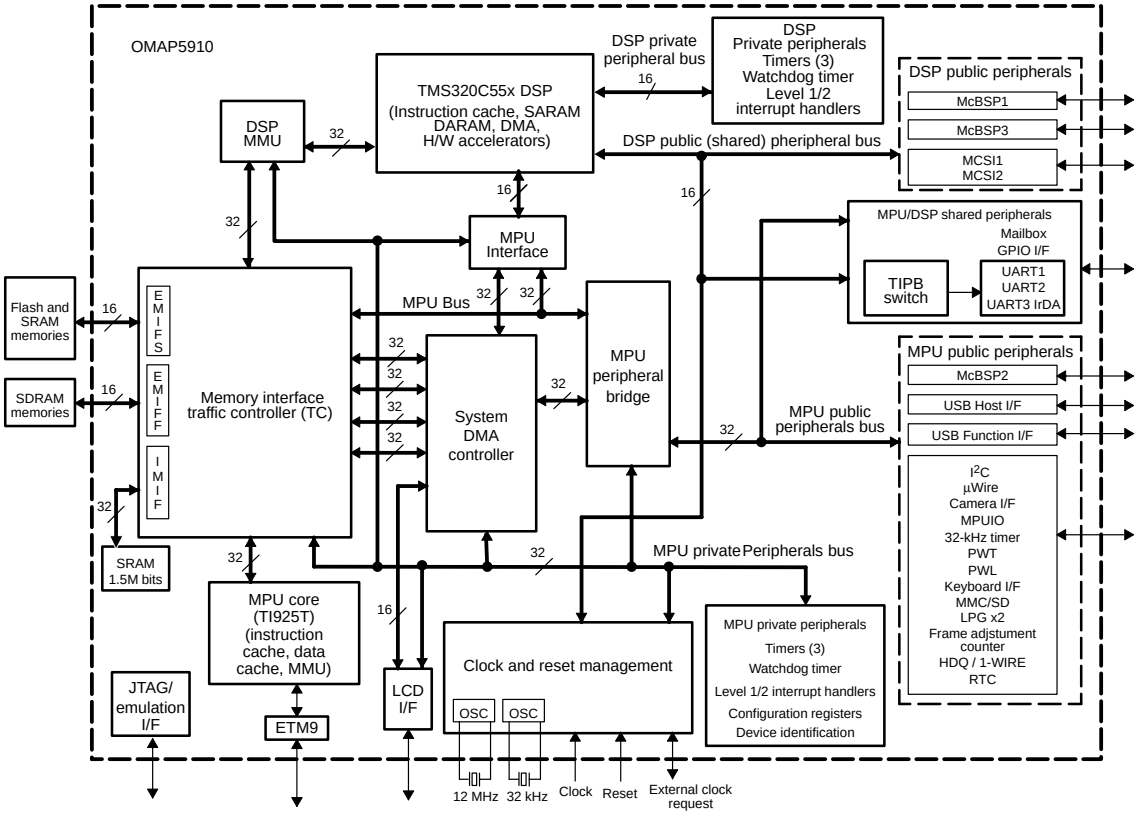
9.1 Introduction

The four DSP public peripherals for the OMAP5910 processor include two multichannel buffered serial ports (McBSPs) and two multichannel serial interfaces (MCSIs):

- McBSP1
- McBSP3
- MCSI1
- MSCI2

Figure 9–1 shows the OMAP5910 device with the DSP public peripherals highlighted.

Figure 9–1. Highlight of Public Peripherals Area



9.2 McBSPs

Multichannel buffered serial ports (McBSPs) are configurable, high-speed, full-duplex serial ports that allow direct interface to external communications devices. There are three McBSPs on the OMAP5910 device.

- McBSP1 and McBSP3 are on the DSP public peripheral bus and are covered briefly in this chapter.
- McBSP2 is on the MPU public peripheral bus and is covered briefly in Chapter 7, *MPU Public Peripherals*.

For a detailed description of the functionality of all three McBSPs, see the TMS320C55x DSP Peripherals Reference Guide (literature number SPRU317).

Key features of the McBSPs include:

- Full-duplex communication
- DMA support for both RX and TX transfers
- Double-buffered data registers, which allow a continuous data stream
- Independent framing and clocking for receives and transmits
- External shift clock generation or an internal programmable frequency shift clock
- Multichannel transmits and receives of up to 128 channels
- A wide selection of data sizes, including 8-, 12-, 16-, 20-, 24-, or 32-bits
- μ-Law and A-Law companding
- Data transfers with LSB or MSB first
- Programmable polarity for both frame synchronization and data clocks
- Highly programmable internal clock and frame generation
- Supports bit rates up to 25M bits/second
- RX and TX interrupts as well as RX data overrun interrupt

The operation of the OMAP5910 McBSPs is consistent with the SPRU317, with the following exceptions and clarifications:

- Only DXENA = 0 setting is supported.
- The transmit output (DX) pins don not go to high impedance when the transmitter in not actively sending data. In other words, the OMAP5910 always actively drives the DX pins.
- The CLKS input is only available on McBSP1.
- The receiver can only operate in slave mode on McBSP1 and McBSP3.

9.3 McBSP1

This section provides information specific to McBSP1 of the OMAP5910 device. For a full description of McBSP functionality and register descriptions, see the TMS320C55x DSP Peripherals Reference Guide (literature number SPRU317).

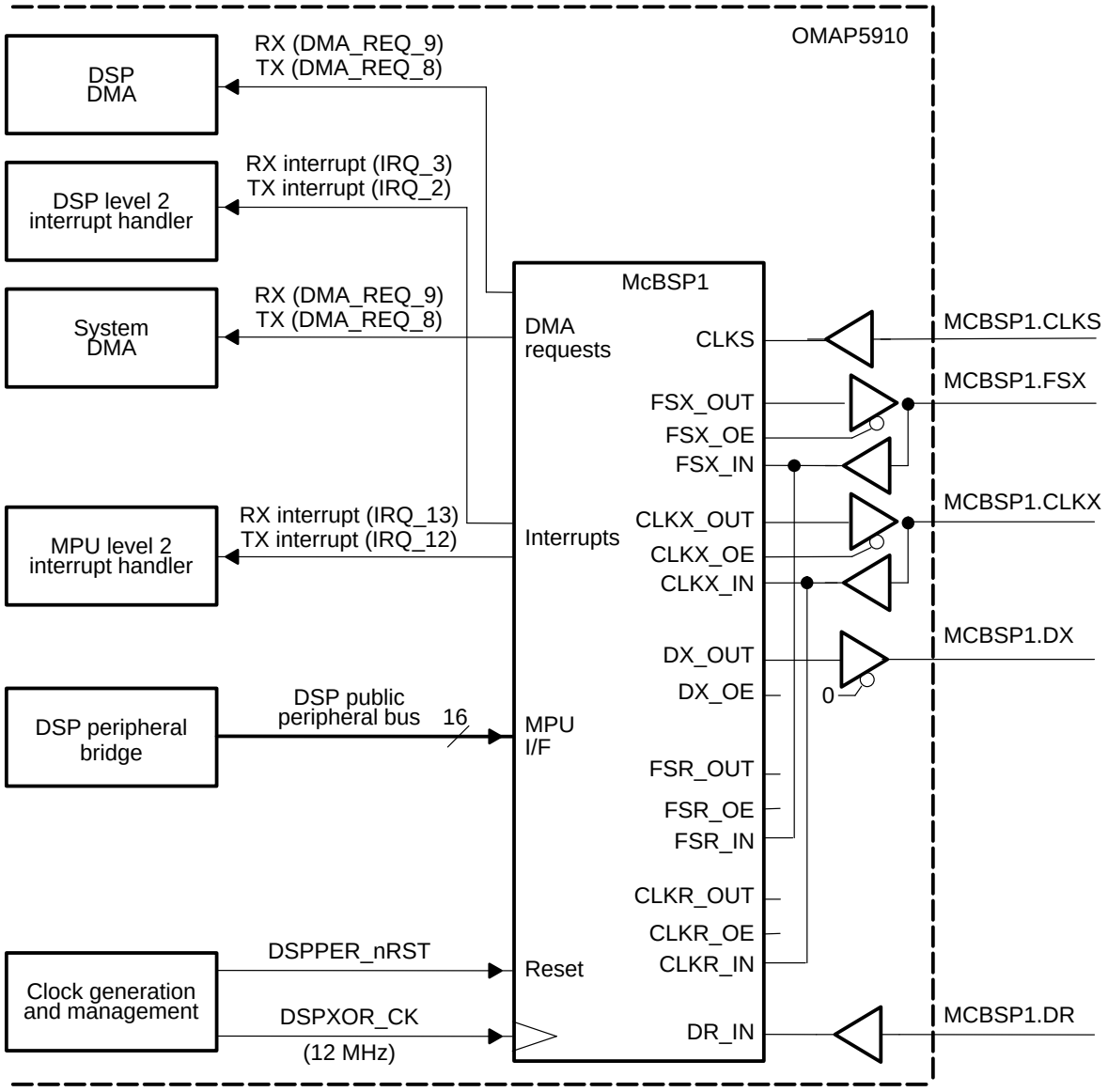
9.3.1 McBSP1 Pin Descriptions

Table 9–1 identifies the McBSP1 I/O pins.

Table 9–1. McBSP1 Pin Descriptions

Pin	I/O Direction	Description
MCBSP1.CLKS	In	Clock input
MCBSP1.DR	In	Data input
MCBSP1.DX	Out	Data output
MCBSP1.CLKX	In/out	Bit clock
MCBSP1.FSX	In/out	Frame synchronization

Figure 9–2. McBSP1 Interface Diagram



Note: You can use the AUXON feature to gate the functional clock to the McBSP1 module by setting MOD_CONF_CTRL_0[18] to 1.

The McBSP1 is half duplex, master/slave for transmission, slave for reception. Table 9–2 lists the McBSP1 signals are available at the OMAP5910 level

Table 9–2. Available McBSP1 Signals

Generic McBSP Signal Name	Description	McBSP1 Signal Name
FSX2	Transmission frame (bidirectional)	McBSP1.FSX
CLKX	Transmission clock (bidirectional)	McBSP1.CLKX
DX	Transmit data (output only)	McBSP1.DX
FSR	Receive frame (input only)	Not available (internal feedback from CLXX)
DR	Receive data	McBSP1.DR
CLKS	Clock input	McBSP1.CLKS

9.3.2 McBSP1 Interrupt Mapping

Table 9–3 identifies the McBSP1 interrupts. McBSP1 generates level 2 interrupts for both the DSP and the MPU.

Table 9–3. McBSP1 Interrupt Mapping

Incoming Interrupts	Level 2 DSP Interrupt	Level 2 MPU Interrupt
McBSP1 TX interrupt	IRQ_02	IRQ_12
McBSP1 RX interrupt	IRQ_03	IRQ_13

9.3.3 McBSP1 DMA Request Mapping

Table 9–4 identifies McBSP1 DMA request lines.

Table 9–4. DMA Request Mapping—McBSP1

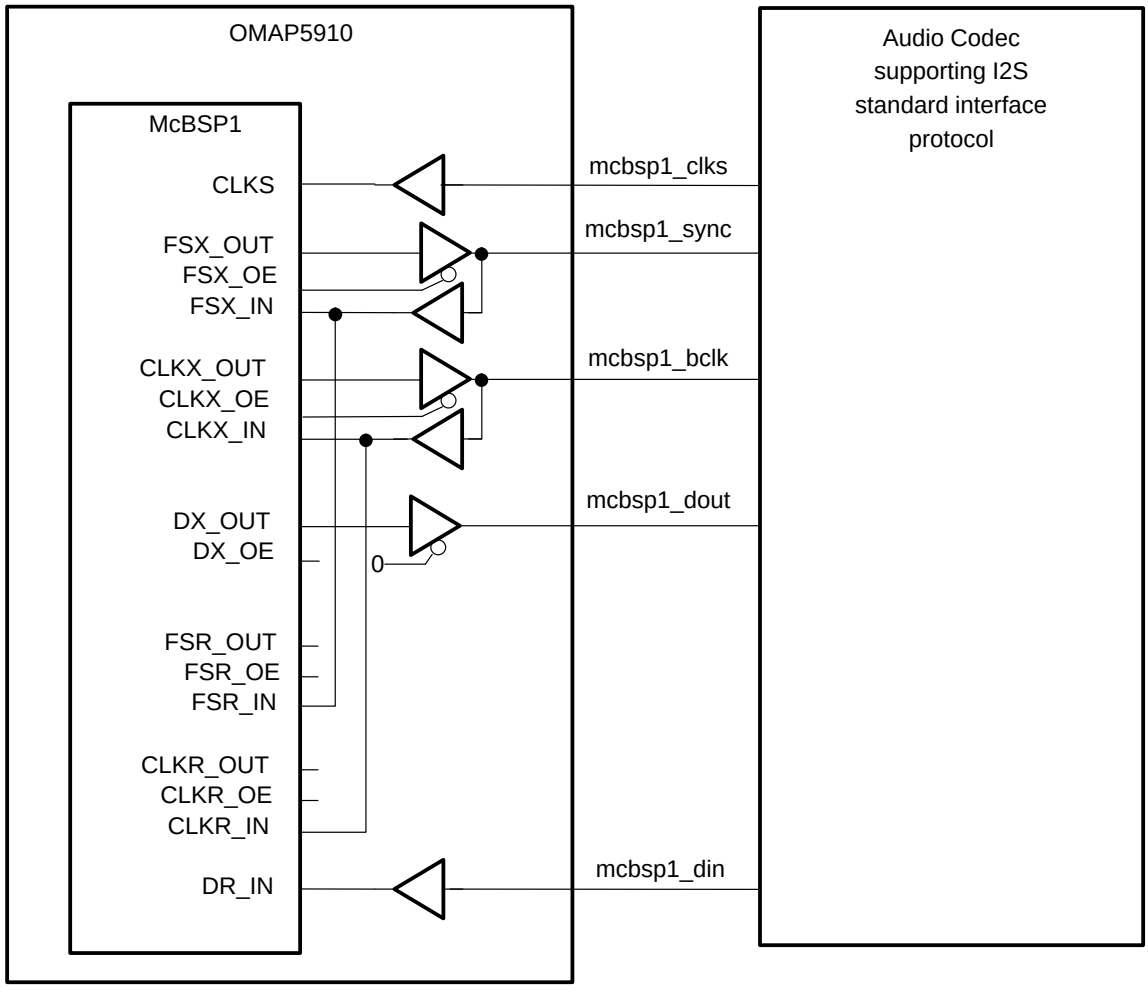
DMA Request Source	DMA Request Line—DSP	DMA Request Line—MPU
McBSP1 TX	DMA_REQ_08	DMA_REQ_08
McBSP1 RX	DMA_REQ_09	DMA_REQ_09

9.3.4 McBSP1 Application Example: I2S Interface

This application uses McBSP1 as an I2S audio codec interface (see Figure 9–3). The OMAP5910 is intended to be either the master or slave device; that is, it either receives or provides the frame synchronization and bit clock.

Section 9.3.4.1 through Section 9.3.4.9 explain how to set up the McBSP registers for I2S slave mode with 16-bit transfers using DMA support.

Figure 9–3. I2S Audio Codec Interface



9.3.4.1 Serial Port Control Register Configuration

DSP_Write(0x0000) => SPCR1; set up SPCR1 as initial configuration.

This setup is not needed after reset.

DSP_Write(0x0000) => SPCR2; set up SPCR2 as initial configuration.

This set up is not needed after reset.

9.3.4.2 Pin Control Register Configuration

DSP_Write(0x0000) => PCR; set up PCR as shown in Table 9–5.

Table 9–5. Pin Control Register Configuration (DSP_Write(0x0000) => 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 OMAP5910 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

9.3.4.3 Receive Control Register Configuration

DSP_Write(0x00a0) => RCR1; set up RCR1 as shown in Table 9–6.

Table 9–6. Receive Control Register 1 Configuration (DSP_Write(0x00a0) => RCR1)

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 as shown in Table 9–7.

Table 9–7. Receive Control Register 2 Configuration (DSP_Write(0x80a1) => RCR2)

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

9.3.4.4 Transmit Control Register Configuration

DSP_Write(0x00a0) => XCR1; set up XCR1 as shown in Table 9–8.

Table 9–8. Transmit Control Register 1 Configuration (DSP_Write(0x00a0) => 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 as shown in Table 9–9.

Table 9–9. Transmit Control Register 2 Configuration (DSP_Write(0x80a1) => 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

9.3.4.5 Sample Rate Generator Configuration (SRGR[1,2])

It is not necessary to configure the sample rate generator, because external clocks and frames are provided appropriately for CLKX and FSX.

9.3.4.6 DMA Configuration

It is necessary to configure the REVT and XEVT bit for the DMA receive and transmit synchronized invent.

9.3.4.7 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.

9.3.4.8 Take out of Reset for Transmit and Receive Starting (SPCR[1,2])

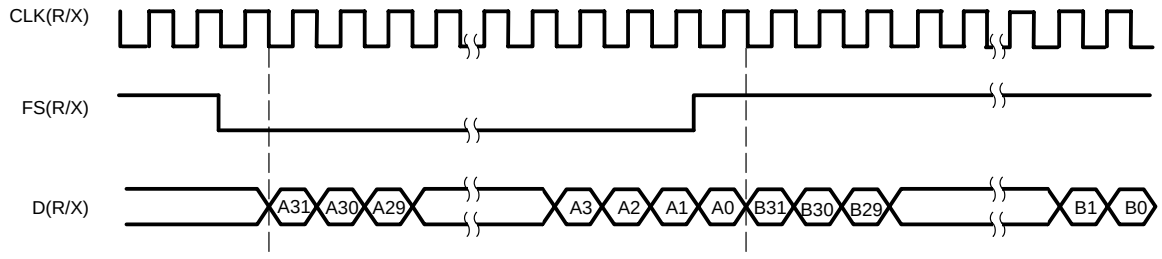
1) DSP_write SPCR1 or (0x0001) => SPCR1; enabled receive port

2) DSP_write SPCR2 or (0x0001) => SPCR2; enabled transmit port

9.3.4.9 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 9–4. Waveform Example



9.4 McBSP3

This section provides information specific to McBSP3 on the OMAP5910 device. For a full description of McBSP functionality and register definitions, see the TMS320C55x DSP Peripherals Reference Guide (literature number SPRU317).

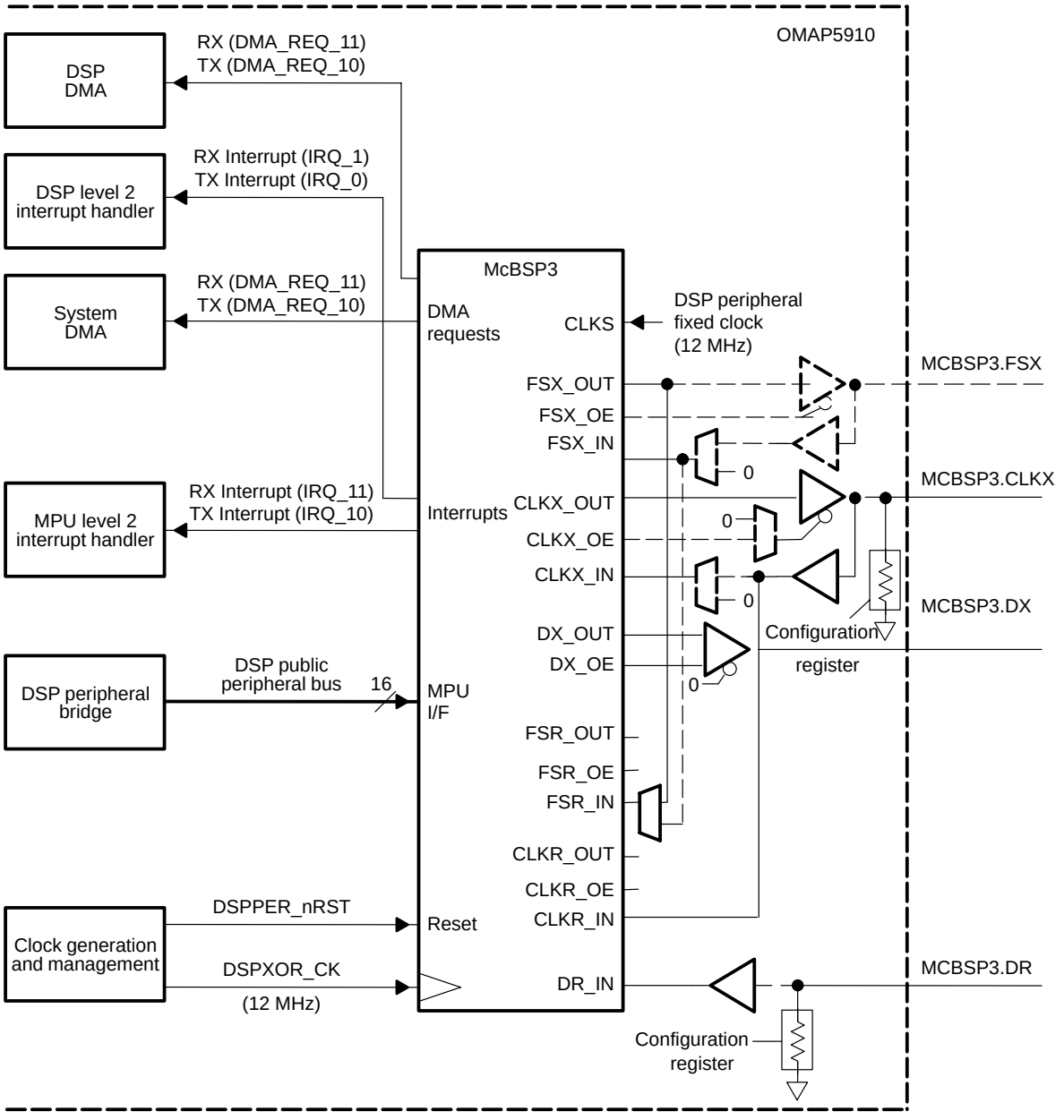
9.4.1 McBSP3 Pin Descriptions

Table 9–10 identifies the McBSP3 I/O pins.

Table 9–10. McBSP3 Pin Descriptions

Pin	I/O Direction	Description
MCBSP3.DR	In	Data input
MCBSP3.DX	Out	Data output
MCBSP3.CLKX	In/out	Bit clock
MCBSP3.FSX	In/out	Frame synchronization

Figure 9–5. McBSP3 Interface Diagram



Note: You can use the AUXON feature to gate the functional clock to the McBSP3 module by setting MOD_CONF_CTRL_0[20] to 1.

There are two modes for the McBSP3, which are selected by the bit 28, MOD_MCBSP3_MODE_R, of the MOD_CONF_CTRL_0 register:

- MOD_MCBSP3_MODE_R = 0 (default). In this case, the McBSP3 is half duplex, master for transmission, slave for reception. The paths shown as dashed lines in Figure 9–5 are not available in this mode. Table 9–11 lists the McBSP3 signals available in this mode at the OMAP5910 level.

Table 9–11. Available McBSP3 Signals in R = 0 Mode

Generic McBSP Signal Name	Description	McBSP3 Signal Name
FSX	Transmission frame (output only)	Not available
CLKX	Transmission clock (output only)	McBSP3.CLKX
DX	Transmit data (output only)	McBSP3.DX
FSR	Receive frame (input only)	Not available (internal feedback from FSX)
CLKR	Receive clock (input only)	Not available (internal feedback from FSX)
DR	Receive data	McBSP3.DR

- MOD_MCBSP3_MODE_R = 1. In this case, the McBSP3 is half duplex, master/slave for transmission, slave for reception. This mode utilizes the paths shown as dashed lines in Figure 9–5. Table 9–12 lists the McBSP3 signals available in this mode at the OMAP5910 level.

Table 9–12. Available McBSP3 Signals in R = 1 Mode

Generic McBSP Signal Name	Description	McBSP3 Signal Name
FSX	Transmission frame (bidirectional)	McBSP3.FSX (multiplexed on another pad)
CLKX	Transmission clock (bidirectional)	McBSP3.CLKX
DX	Transmit data (output only)	McBSP3.DX
FSR	Receive frame (input only)	Not available (internal feedback from FSX)
CLKR	Receive clock (input only)	Not available (internal feedback from CLXX)
DR	Receive data	McBSP3.DR

9.4.2 McBSP3 Interrupt Mapping

Table 9–13 identifies the McBSP3 interrupts. McBSP3 generates level 2 interrupts for both the DSP and the MPU.

Table 9–13. McBSP3 Interrupt Mapping

Incoming Interrupts	Level 2 DSP Interrupt	Level 2 MPU Interrupt
McBSP3 TX interrupt	IRQ_00	IRQ_10
McBSP3 RX interrupt	IRQ_01	IRQ_11

9.4.3 McBSP3 DMA Request Mapping

Table 9–14 identifies McBSP3 DMA request lines.

Table 9–14. DMA Request Mapping—McBSP3

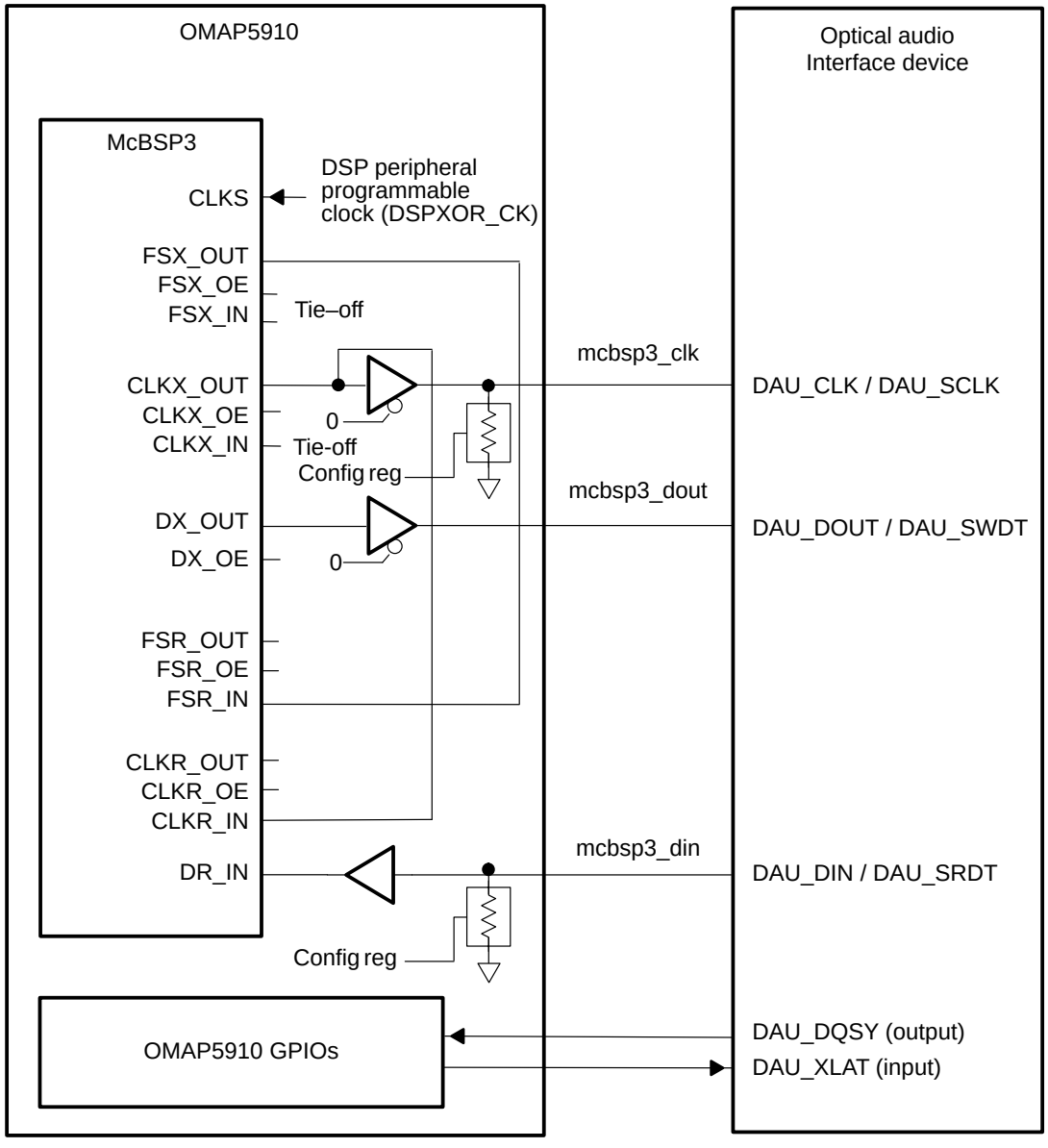
DMA Request Source	DMA Request Line—DSP	DMA Request Line—MPU
McBSP3 TX	DMA_REQ_10	DMA_REQ_10
McBSP3 RX	DMA_REQ_11	DMA_REQ_11

9.4.4 McBSP3 Application Example: Optical Interface

With the assistance of two GPIOs, McBSP3 is configured to connect to an external optical audio interface (see Figure 9–6) device such as the Sanyo LC89051V. 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.

Section 9.4.4.1 through Section 9.4.4.12 explain the McBSP register setup for optical interface with 8-bit transfer per frame in SPI master mode and GPIO mode.

Figure 9–6. Optical Audio Interface



9.4.4.1 Serial Port Control Register Configuration

DSP_Write(0x1000) => SPCR; set up SPCR1 per below configuration.

Table 9–15. Serial Port Control Register Configuration (DSP_Write(0x1000) => SPCR)

Bit	Config 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 set up is not needed after reset.

9.4.4.2 Pin Control Register Configuration

DSP_Write(0x0a0b) => PCR; set up PCR per below configuration.

Table 9–16. Pin Control Register Configuration (DSP_Write(0x0a0b) => PCR)

Bit	Config 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 OMAP5910)
5	0b	DX pin status
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

9.4.4.3 Receive Control Register Configuration

The values of RWDLEN1, 2 and XWDLEN1, 2 must be set to the same value in SPI mode.

DSP_Write(0x0000) => RCR1; set up RCR1 per below configuration.

Table 9–17. Receive Control Register 1 Configuration (DSP_Write(0x0000) => RCR1)

Bit	Config 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 below configuration.

Table 9–18. Receive Control Register 2 Configuration (DSP_Write(0x0000) => RCR2)

Bit	Config 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

9.4.4.4 Transmit Control Register Configuration

The values of RWDLEN1, 2 and XWDLEN1, 2 must be set to the same value in SPI mode.

DSP_Write(0x0000) => XCR1; set up XCR1 per below configuration.

Table 9–19. Transmit Control Register 1 Configuration (DSP_Write(0x0000) => 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 below configuration.

Table 9–20. Transmit Control Register 2 Configuration (DSP_Write(0x0000) => XCR2)

Bit	Config 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

9.4.4.5 Sample Rate Generator Configuration (SRGR[1,2])

DSP_Write (0x00FF) => SRGR1; set up SRGR1 per below configuration.

Table 9–21. Sample Rate Generator 1 Configuration (SRGR[1,2]) (DSP_Write (0x00FF) => 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 below configuration.

Table 9–22. Sample Rate Generator 2 Configuration (SRGR[1,2]) (DSP_Write (0x2000) => 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.

9.4.4.6 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.

9.4.4.7 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 set up is not needed after reset.

3) DSP_Write MIR and (0xFFFF F3FF) => MIR; enabled SPI TX and RX interrupt

9.4.4.8 Interrupt Flag Configuration MASK Release on Level 2 Handler

DSP_Write MIR and (0xFFFF FFFB) => MIR0; enabled INT4 (level 2 interrupt FIR)

9.4.4.9 Take Out of Reset for Transmit and Receive Starting (SPCR[1,2])

1) DSP_write SPCR1 or (0x0001) => SPCR1; enabled receive port
2) DSP_write SPCR2 or (0x0001) => SPCR2; enabled transmit port

Note:

Wait two sample rate clock cycles for McBSP stability.

9.4.4.10 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

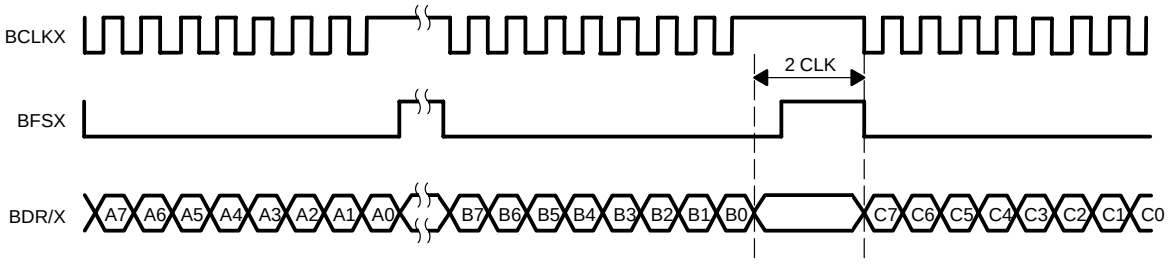
9.4.4.11 Register Setup GPIO Mode

1) DSP_Write SPCR1 and (0xFFFE) => SPCR1; disabled receive port
2) DSP_Write PCR or (0x1000) => PCR; DR pin set as GPI

9.4.4.12 Read From GPI

DSP_Read <= PCR; read DR_STAT bit

Figure 9–7. Waveform Example



Section 9.4.4.13 through Section 9.4.4.21 explain the McBSP register setup for TX master and RX slave with 8-bit data transfer using DMA support.

9.4.4.13 Serial Port Control Register Configuration

DSP_Write(0x1000) => SPCR1; set up SPCR1 per below configuration.

Table 9–23. Serial Port Control Register Configuration (DSP_Write(0x1000) => SPCR1)

Bit	Config 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.

9.4.4.14 Pin Control Register Configuration

DSP_Write(0x0a0b) => PCR; set up PCR per below configuration.

Table 9–24. Pin Control Register Configuration (DSP_Write(0x0a0b) => 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	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 OMAP5910)
5	0b	DX pin status
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

9.4.4.15 Receive Control Register Configuration

The values of RWDLEN1, 2 and XWDLEN1, 2 must be set to same value in SPI mode.

DSP_Write(0x0000) => RCR1; set up RCR1 per below configuration.

Table 9–25. Receive Control Register 1 Configuration (DSP_Write(0x0000) => RCR1)

Bit	Config 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 below configuration.

Table 9–26. Receive Control Register 2 Configuration (DSP_Write(0x0000) => RCR2)

Bit	Config 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

9.4.4.16 Transmit Control Register Configuration

The values of RWDLEN1, 2 and XWDLEN1, 2 must be set to the same value in SPI mode.

DSP_Write(0x0000) => XCR1; set up XCR1 per below configuration.

Table 9–27. Transmit Control Register 1 Configuration (DSP_Write(0x0000) => 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 below configuration.

Table 9–28. Transmit Control Register 2 Configuration (DSP_Write(0x0000) => 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

9.4.4.17 Sample Rate Generator Configuration (SRGR[1,2])

1) Configure the sample rate generator appropriately for CLKX and FSX. For details, see *TMS320C54x DSP Enhanced Peripherals Reference Set*, vol. 5, SPRA302.

2) Wait two CLKSRG clocks.

3) ARM_Write SPCR2 or (0x0000 0040)=>SPCR2;CLKG enable

4) Wait two CLKG clocks.

9.4.4.18 DMA Configuration

Configure the REVT and XEVT bit for the DMA receive and transmit synchronized invent.

9.4.4.19 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; disabled SPI TX and RX interrupt

Note:

Enable the appropriate DMA channel interrupts.

9.4.4.20 Take out of Reset for Transmit and Receive Starting (SPCR[1,2])

1) ARM_write SPCR1 or (0x0001) => SPCR1; enabled receive port
2) ARM_write SPCR2 or (0x0001) => SPCR2; enabled transmit port

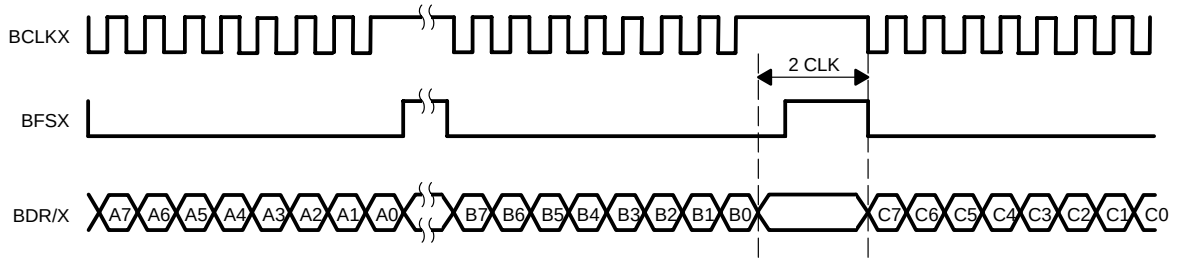
9.4.4.21 Data Transfer (DMA Channel)

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 taken the interrupt handle.

Figure 9–8. Waveform Example



9.5 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 OMAP5910 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

9.5.1 Communication Protocol

9.5.1.1 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 either is 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 either is 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 positive edge clock and reading on negative edge clock or inverted with writing on negative edge clock and reading on 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
($t_{CLK} = t_{12MHz} * CLK\_FREQ$)

9.5.1.2 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

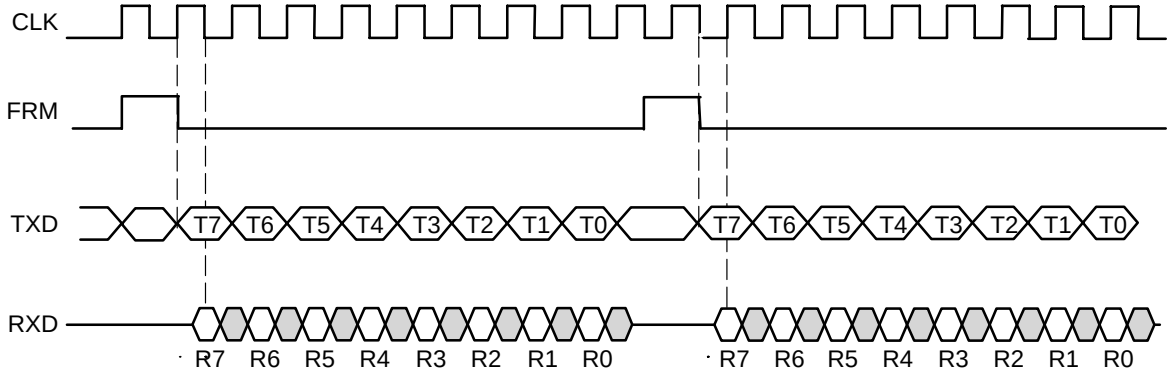
Received Data Loading (RX_INT ISR)

- DSP_Read = RX_REG

Stop MCSI

- DSP_Write(0x0000) = CONTROL_REG (disable MCSI clock)
- DSP_Write(0x0002) = CONTROL_REG (reset MCSI registers)

Figure 9–9. Communication μ-Law Interface Interrupts Waveform Example



9.5.1.3 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

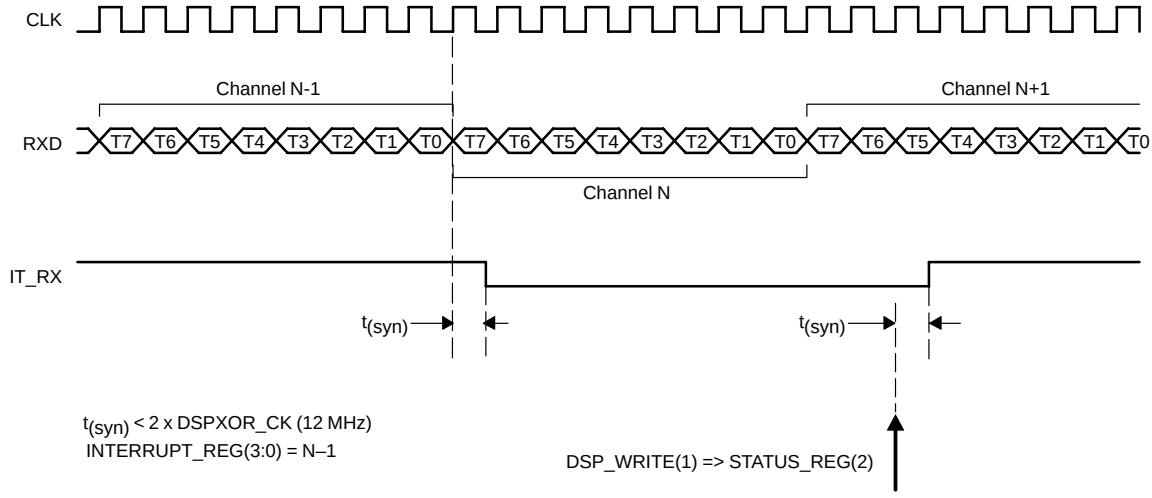
The receive interrupt is generated every frame after the completion of the reception of a data word:

- 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 9–10. Receive Interrupt Timing Diagram

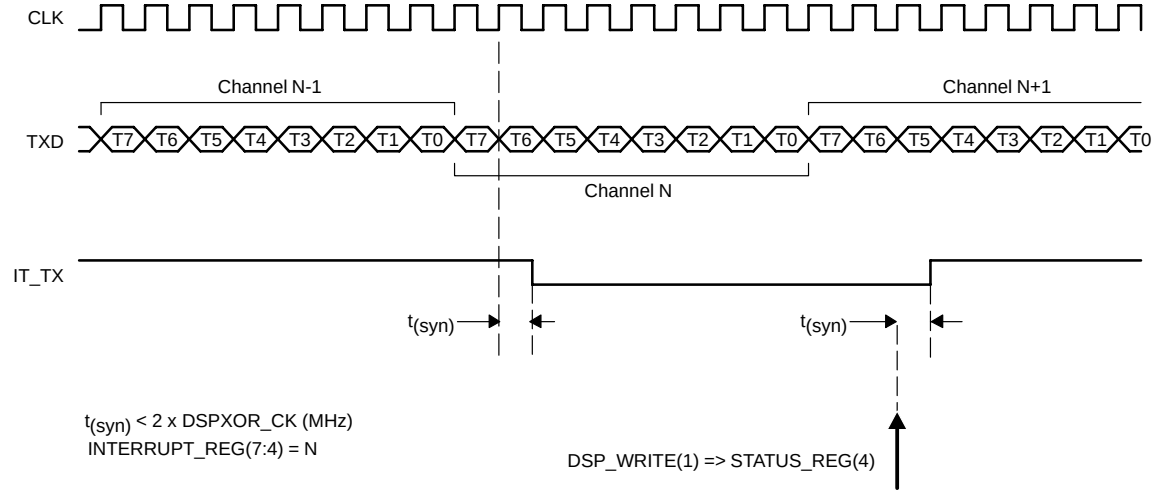


Transmit Interrupt

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 9–11. 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 9–12. Frame Duration Error—Too Many (Long)

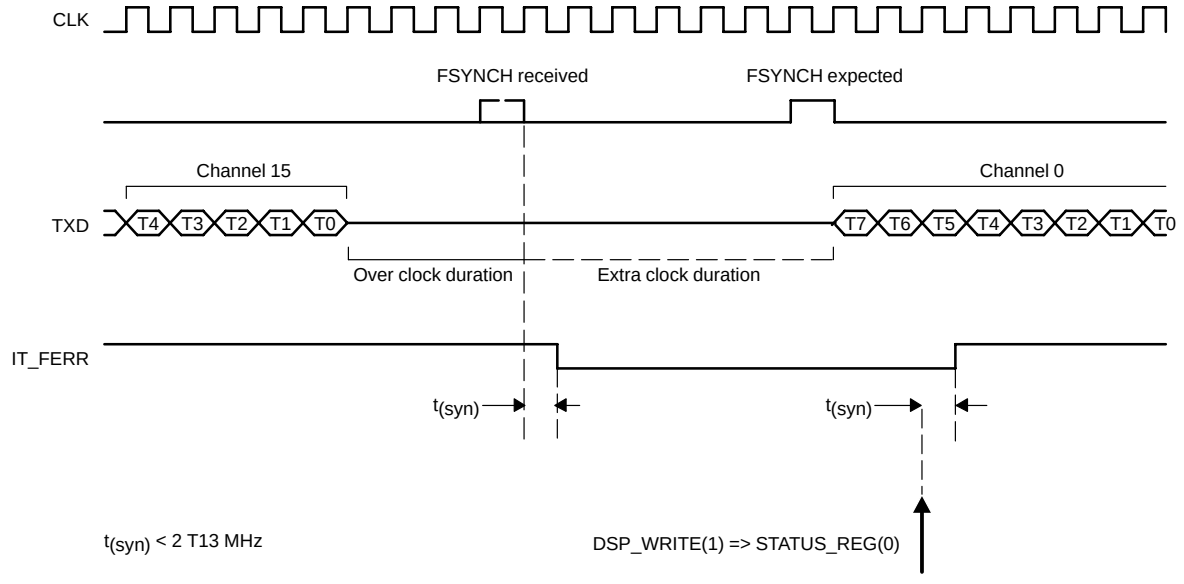
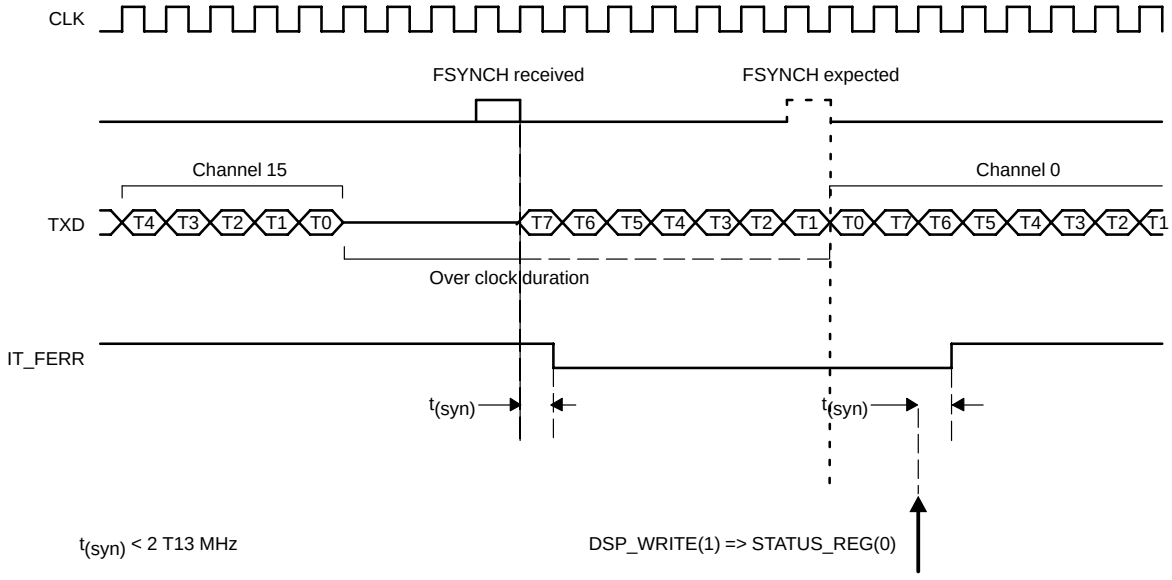


Figure 9–13. Frame Duration Error—Too Few (Short)



9.5.1.4 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(1) 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(1) for FERR_INT release
  - STATUS_REG(2) for RX_INT release
  - STATUS_REG(4) for TX_INT release

9.5.1.5 DMA Channel Operation

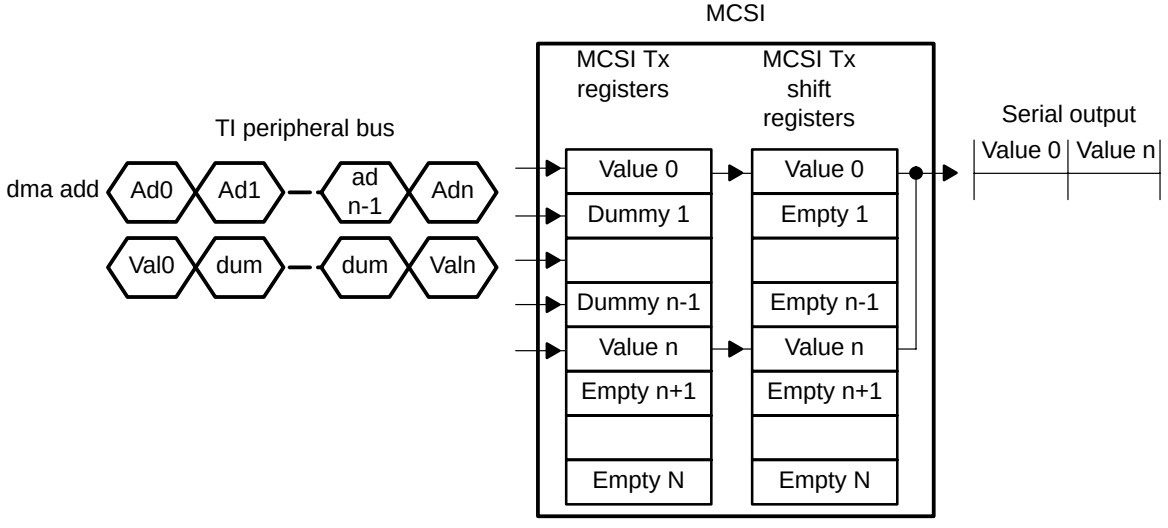
Both transmit and receive operations can be supported by DMA. DMA support is enabled by control bits in the MAIN_PARAMETERS_REG:

- 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 9–14).

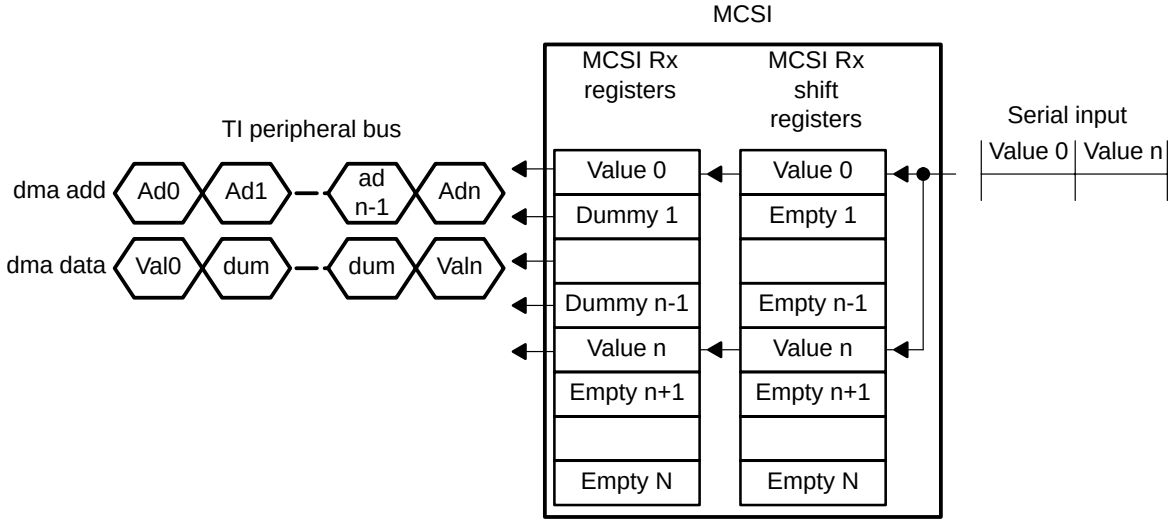
Figure 9–14. Transmit DMA Transfers



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 9–15).

Figure 9–15. 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.

9.5.1.6 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 9–34, *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.

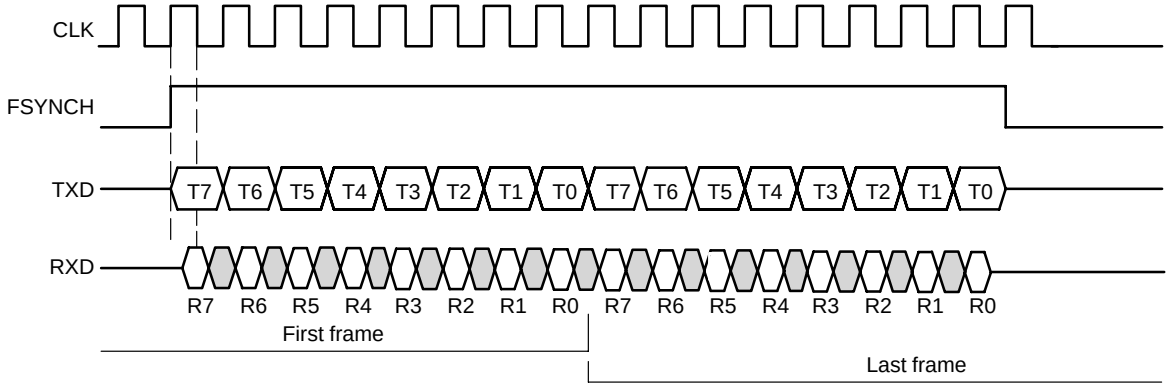
9.5.1.7 Functional Mode Timing Diagrams

The following timing diagrams are based on a positive clock polarity with parameter CLOCK_POL = 0.

(Transmit on rising edge/receive on falling edge)

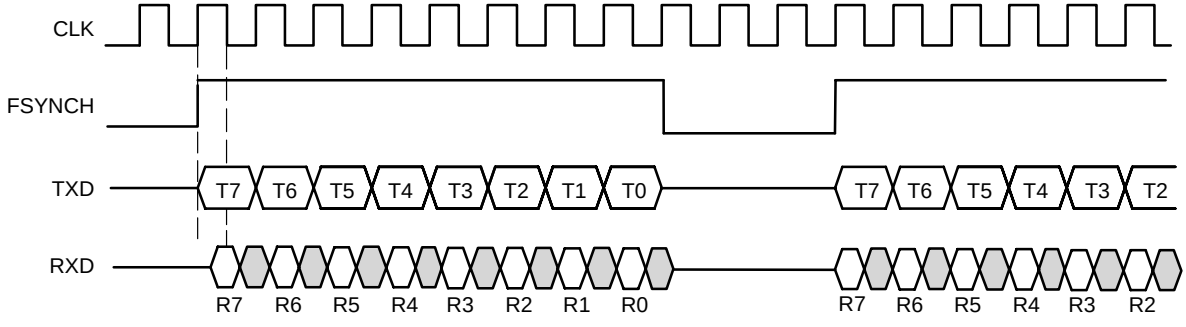
Single-Channel/Alternate Long Framing

Figure 9–16. Single-Channel/Alternate Long Framing



Single-Channel/Alternate Long Framing/Burst

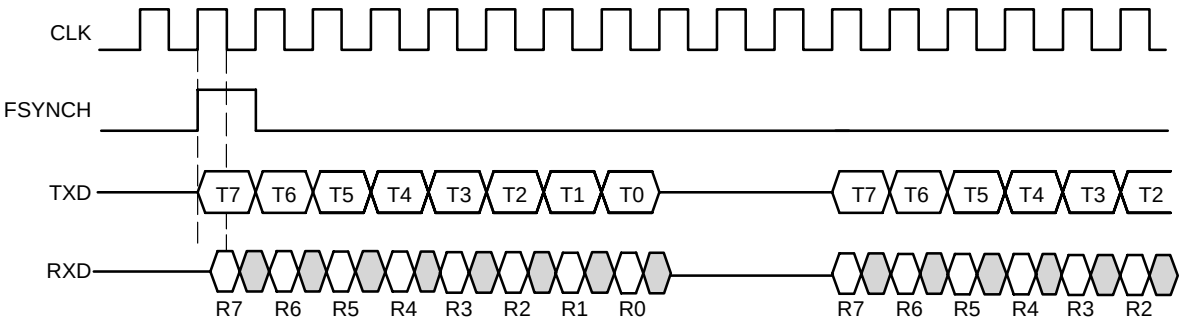
Figure 9–17. Single-Channel/Alternate Long Framing/Burst



OVER_CLOCK_REG = 0x0003

Single-Channel/Alternate Short Framing/Continuous/Burst

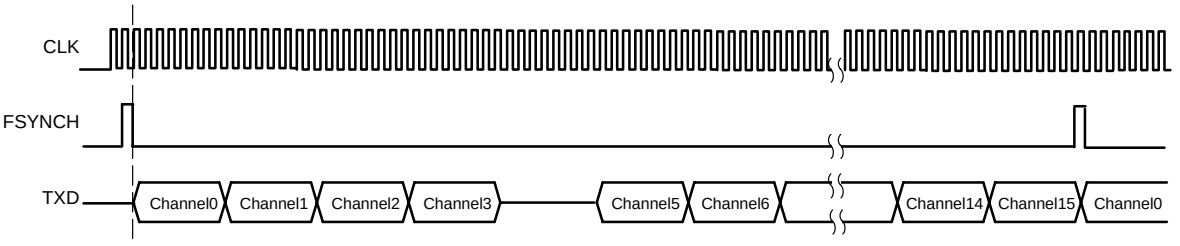
Figure 9–18. Single-Channel/Alternate Short Framing/Continuous/Burst



OVER_CLOCK_REG = 0x0003

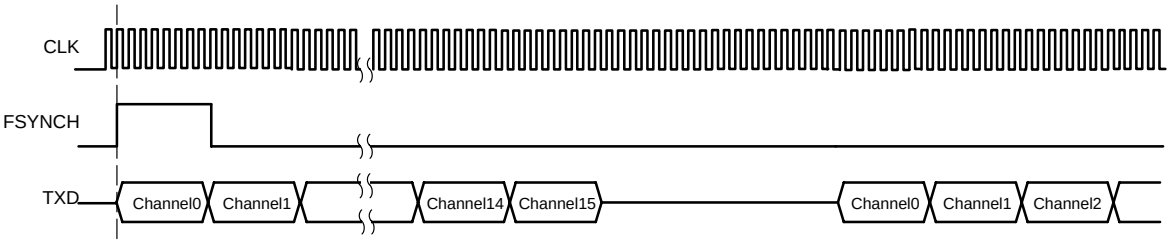
Multichannel/Normal Short Framing/Channel4 Disable

Figure 9–19. Multichannel/Normal Short Framing/Channel4 Disable



Multichannel/Alternate Long Framing/Continuous/Burst

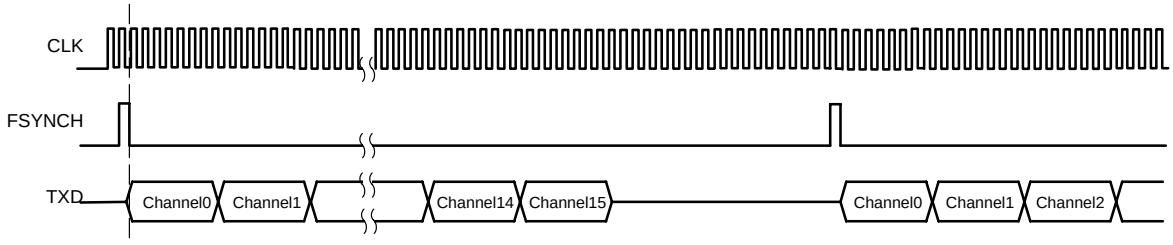
Figure 9–20. Multichannel/Alternate Long Framing/Continuous/Burst



OVER_CLOCK_REG = 0x0013

Multichannel/Normal Short Framing/Burst

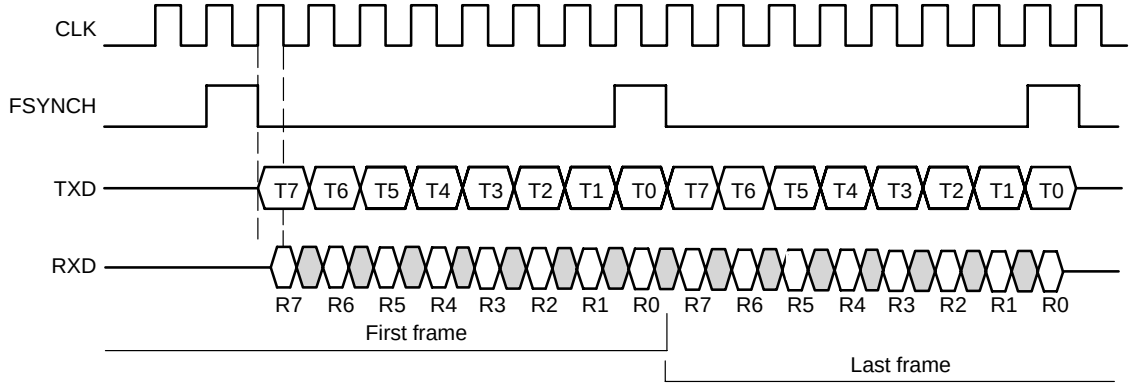
Figure 9–21. Multichannel/Normal Short Framing/Burst



OVER_CLOCK_REG = 0x0013

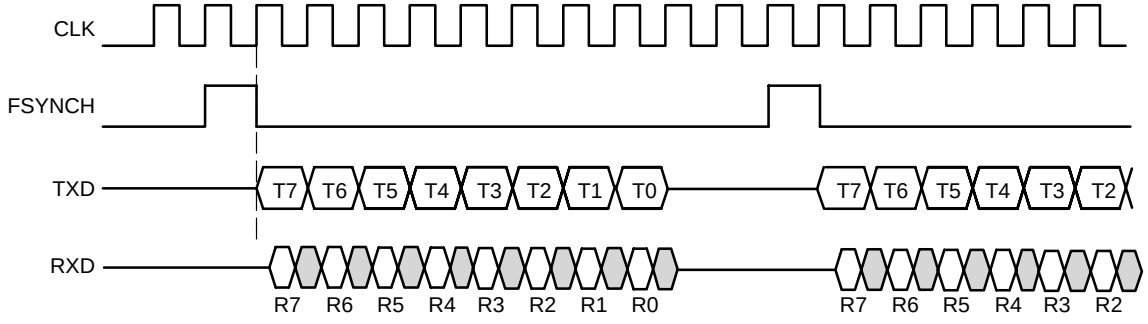
Single-Channel/Normal Short Framing

Figure 9–22. Single-Channel/Normal Short Framing



Single-Channel/Normal Short Framing/Burst

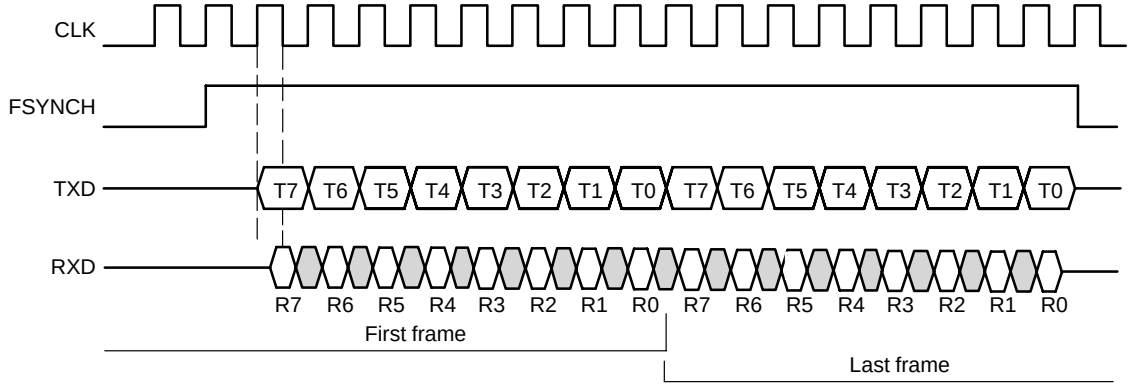
Figure 9–23. Single-Channel/Normal Short Framing/Burst



OVER_CLOCK_REG = 0x0003

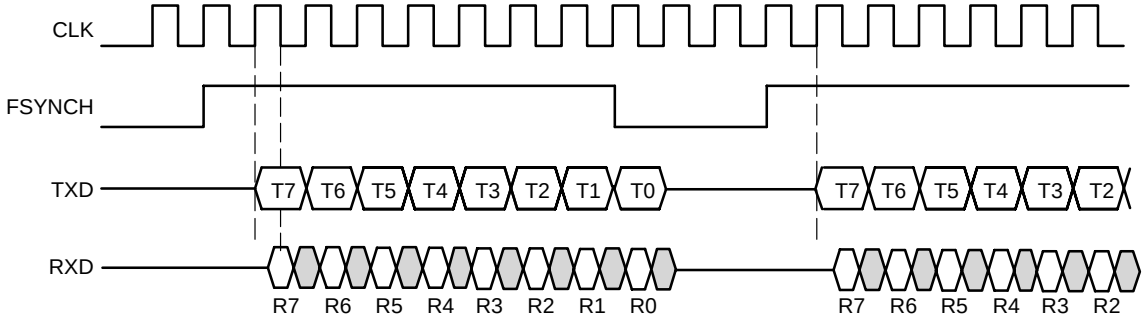
Single-Channel/Normal Long Framing

Figure 9–24. Single-Channel/Normal Long Framing



Single-Channel/Normal Long Framing/Burst

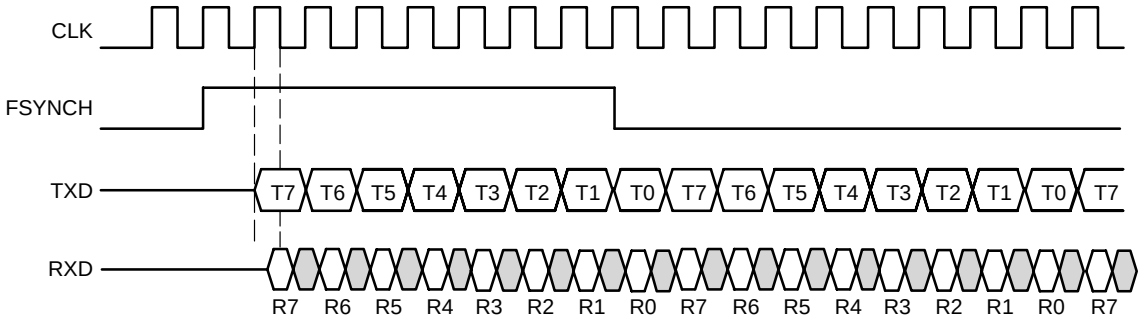
Figure 9–25. Single-Channel/Normal Long Framing/Burst



OVER_CLOCK_REG = 0x0003

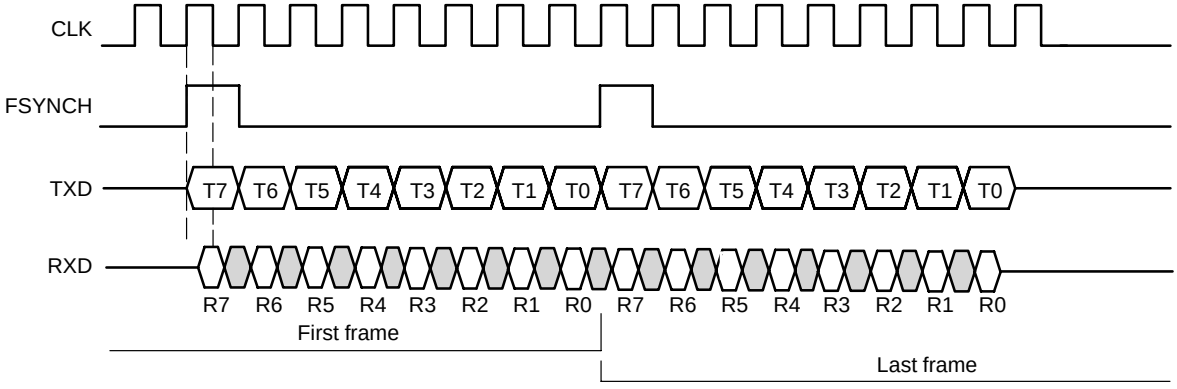
Single-Channel/Normal Long Framing/Continuous

Figure 9–26. Single-Channel/Normal Long/Continuous



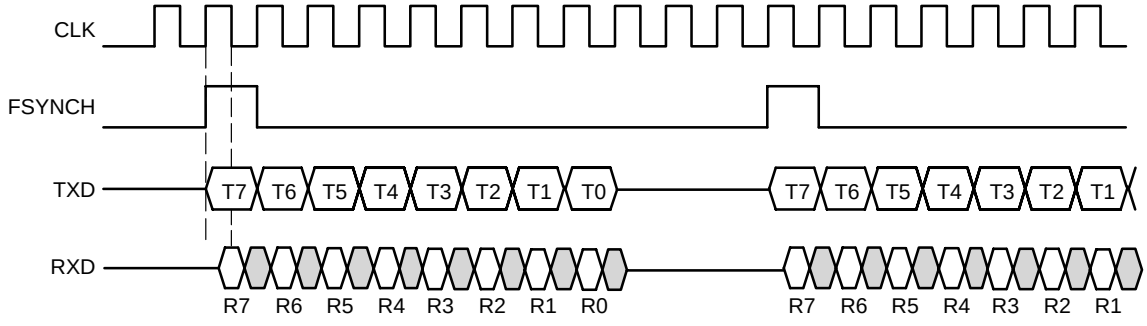
Single-Channel/Alternate Short Framing

Figure 9–27. Single-Channel/Alternate Short Framing



Single-Channel/Alternate Short Framing/Burst

Figure 9–28. Single-Channel/Alternate Short Framing/Burst



OVER_CLOCK_REG = 0x0003

9.5.2 MCSI Register Descriptions

Table 9–29 through Table 9–37 describe the MCSI registers. The CHANNEL_USED_REG, CLOCK_FREQUENCY_REG, OVER_CLOCK_REG, INTERRUPTS_REG, and MAIN_PARAMETERS_REG are write protected if the MCSI is enabled (control_reg[0] = 1).

The channel selection register is only used in multichannel mode (see Table 9–29).

Table 9–29. 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).

The clock frequency register is used only in master mode when the interface generates the serial clock (see Table 9–30).

Table 9–30. Clock Frequency Register (CLOCK_FREQUENCY_REG)

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 9–31. 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 9–32. 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
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 9–33. 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 should 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		

Table 9–33. Main Parameters Register (MAIN_PARAMETERS__REG) (Continued)

Bit	Name	Value	Description	Access	Hardware Reset
8	fsynch_size		Frame-synchronization pulse shape	R/W	0
		0	Short		
		1	Long		
7	Multi/single		Frame structure	R/W	0
		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 9–34. Activity Control Register (CONTROL_REG)

Bit	Name	Value	Description	Access	Hardware Reset	Software Reset
15–3	Reserved		Reserved bits. These bits should always be written as 0.	R	0000 0000 0000 0	0000 0000 0000 0
2	Reserved		Reserved bits. These bits should 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			
0	MCSI clk 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. In order to prevent clearing the interrupt handler before the interrupt line is cleared, the interrupt routine must be at least two cycles long.

Table 9–35. Interface Status Register (STATUS_REG)

Bit	Name	Value	Description	Access	Hardware Reset	Software Reset
15–7	Reserved		Reserved bits. These bits should always be written as 0.	R	0000 0000 0	0000 0000 0
6	Reserved		Reserved bits. These bits should 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 int			
		1	Int			
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 int			
		1	Int			
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 9–36. 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
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 zeroes.

Table 9–37. 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
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.

9.6 MCSI1

This section provides information specific to MCSI1 (Figure 9–29) on the OMAP5910 device.

9.6.1 MCSI1 Pin Description

Table 9–38 identifies the MCSI1 I/O pins.

Table 9–38. 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

9.6.2 MCSI1 Interrupt Mapping

Table 9–39 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 9–39. 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

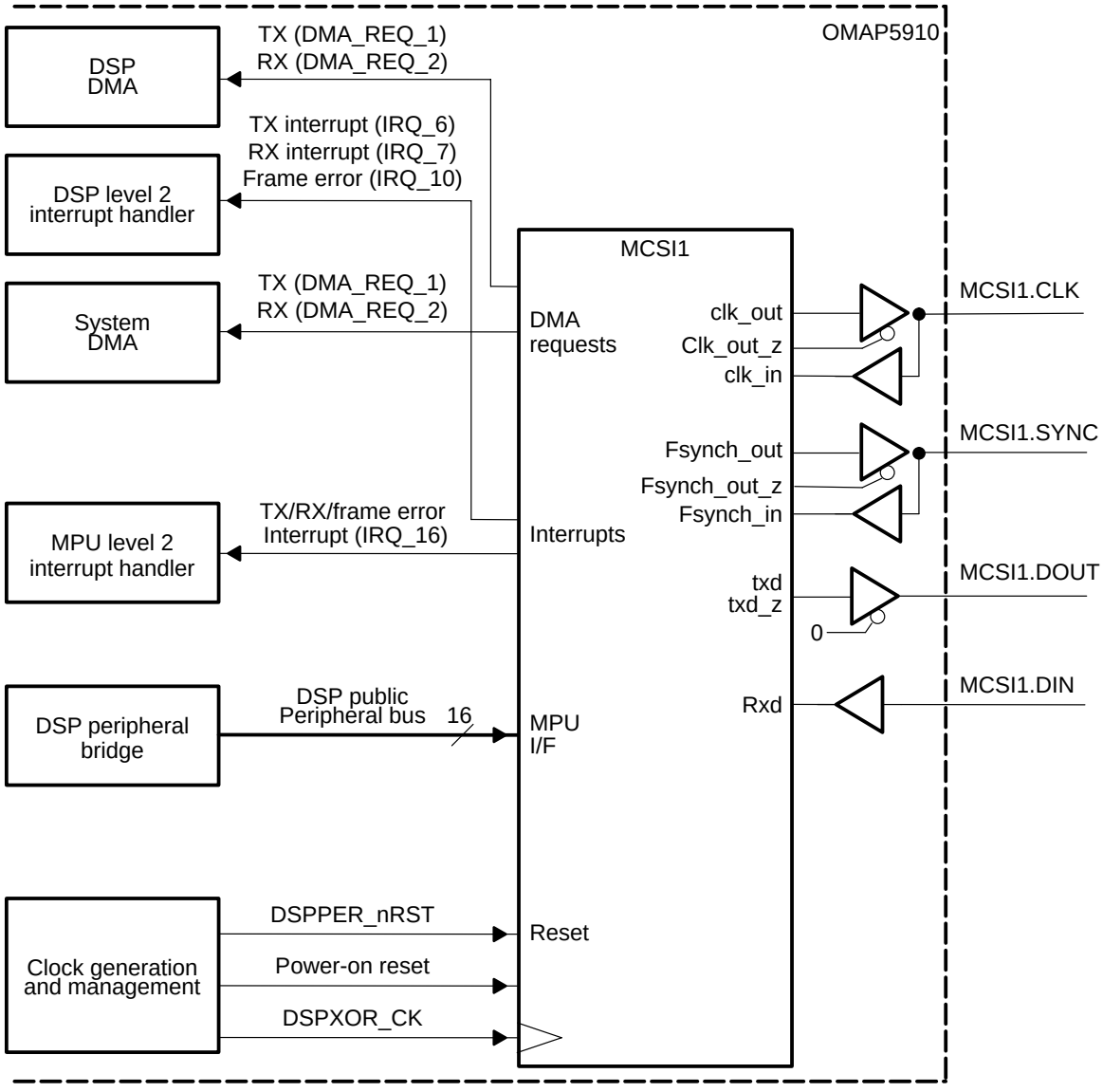
9.6.3 MCSI1 DMA Request Mapping

Table 9–40 identifies MCSI1 DMA request lines.

Table 9–40. 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

Figure 9–29. MCSI1 Interface Diagram



9.7 MCSI2

This section provides information specific to MCSI2 (Figure 9–30) on the OMAP5910 device.

9.7.1 MCSI2 Pin Description

Table 9–41 identifies the MCSI2 I/O pins.

Table 9–41. 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

9.7.2 MCSI2 Interrupt Mapping

Table 9–42 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 9–42. 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

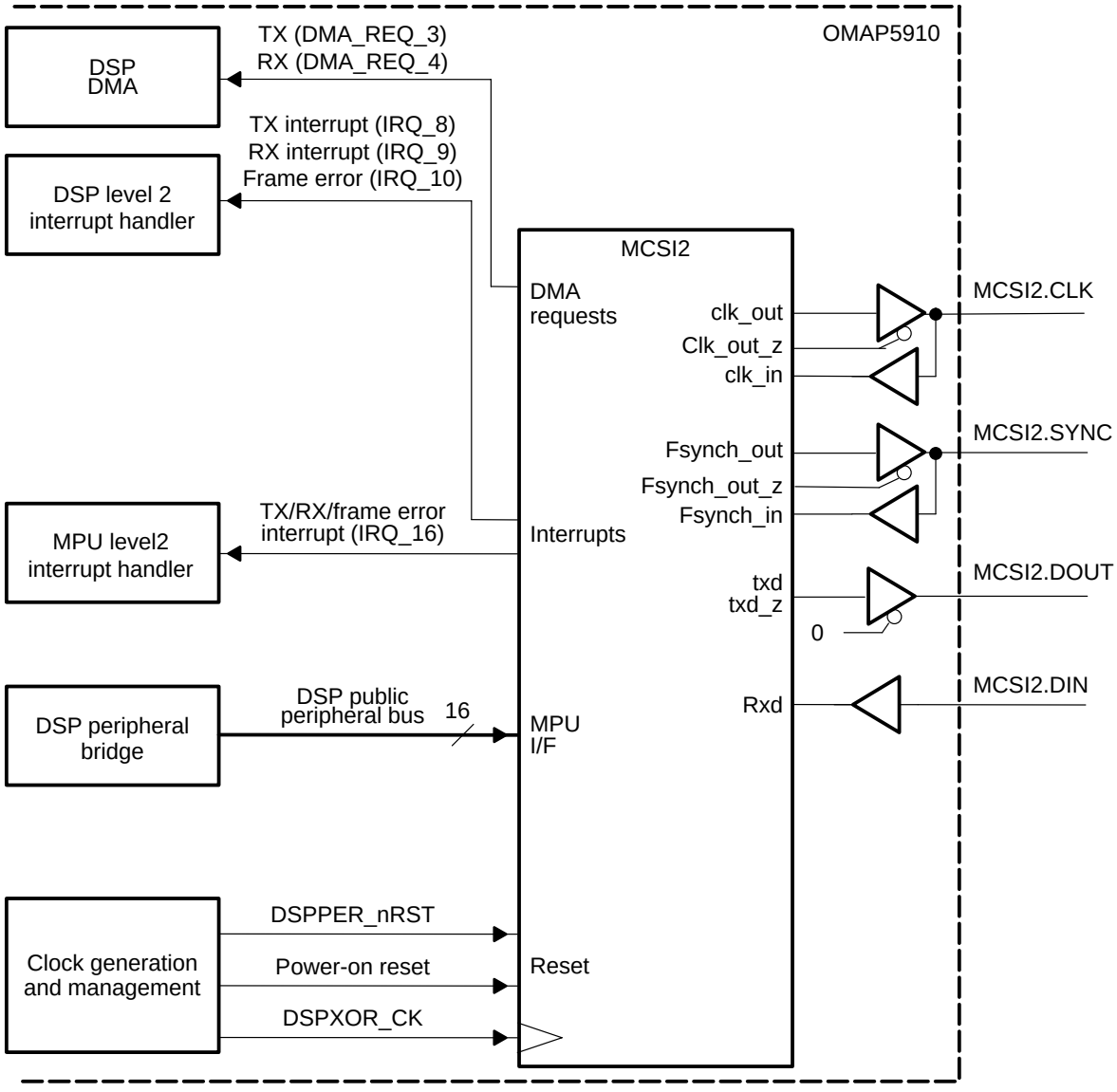
9.7.3 MCSI2 DMA Request Mapping

Table 9–43 identifies MCSI2 DMA request lines. Only the DSP DMA controller can transfer MCSI2 data; there is no MPU DMA capability.

Table 9–43. 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	–

Figure 9–30. MCSI2 Interface Diagram



9.8 McBSP and MCSI Memory and Peripheral Mapping

The base address for each McBSP register map is as follows:

- McBSP1 (I2S audio):
  - 0x08C00 (DSP memory map)
  - E101:1800 (MPU memory map)
- McBSP2 (modem interface): FFFB:1000 (MPU memory map)
- McBSP3 (optical interface):
  - 0x0B800 (DSP memory map)
  - E101:7000 (MPU memory map)

Table 9–44 shows the 19 registers accessible on each McBSP. Table 9–44 through Table 9–45 describe register bits.

Table 9–44. McBSP Registers

Name	Description	Offset In Bytes
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 9–44. McBSP Registers (Continued)

Name	Description	Offset In Bytes
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

9.8.1 MCSI Addresses and Mapping

The base address for each MCSI register map is:

- MCSI1 (Bluetooth MCSI):
  - 0x09400 (DSP memory map word address)
  - E101:2800 (MPU memory map byte address)
- MCSI2 (Modem MCSI):
  - 0x09000 (DSP memory map word address)
  - E101:2000 (MPU memory map byte address)

Table 9–45 shows the MCSI registers and their offset addresses.

Table 9–45. MCSI Register Mapping

Register Name	Offset In Bytes	Register Name	Offset In Bytes
RX15	0x7E	TX10	0x54
RX14	0x7C	TX9	0x52
RX13	0x7A	TX8	0x50
RX12	0x78	TX7	0x4E
RX11	0x76	TX6	0x4C
RX10	0x74	TX5	0x4A
RX9	0x72	TX4	0x48
RX8	0x70	TX3	0x46
RX7	0x6E	TX2	0x44
RX6	0x6C	TX1	0x42
RX5	0x6A	TX0	0x40
RX4	0x68	Unused	0x3F
RX3	0x66	/	/
RX2	0x64	Unused	0x0E
RX1	0x62	Status	0x0C
RX0	0x60	Clock frequency	0x0A
TX15	0x5E	Over-clock	0x08
TX14	0x5C	Channel used	0x06
TX13	0x5A	Interrupts	0x04
TX12	0x58	Main parameters	0x02
TX11	0x56	Control	0x00

Chapter 10

MPU/DSP Shared Peripherals

This chapter describes the MPU/DSP shared peripherals for the OMAP5910 multimedia processor.

Topic **Page**

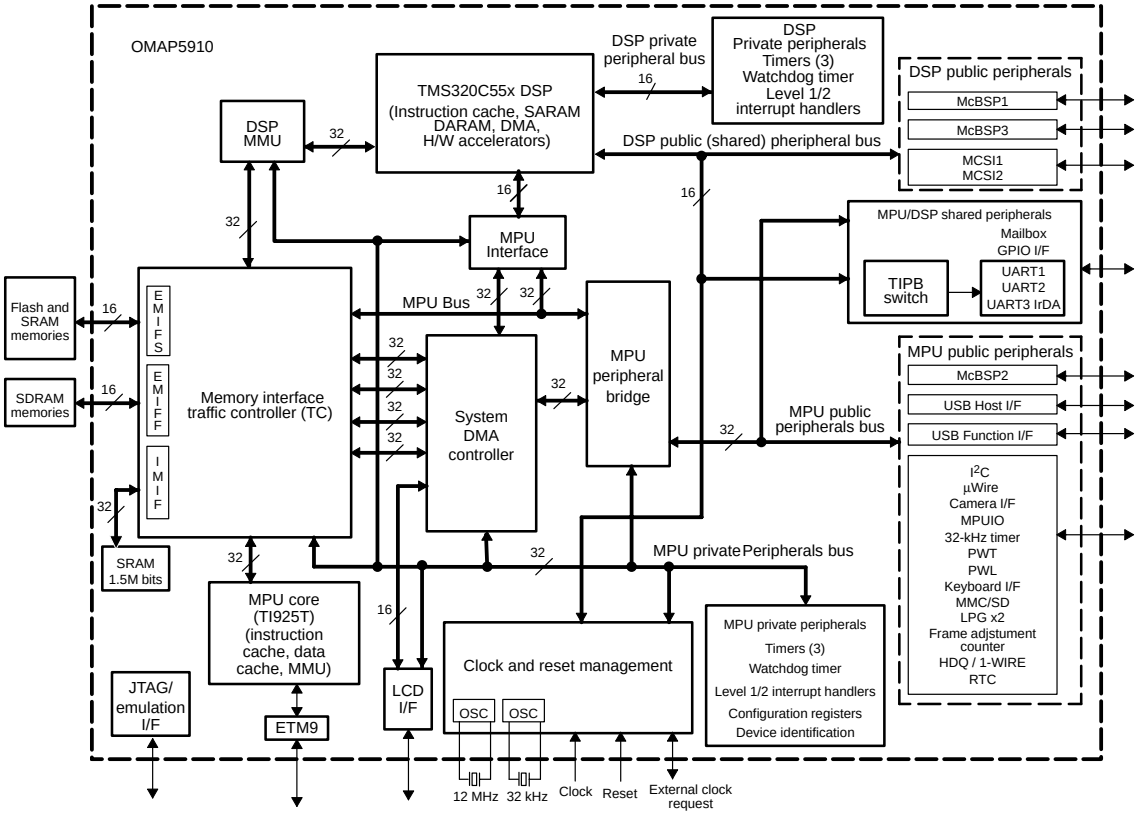
10.1 Introduction

The OMAP5910 device has five peripherals that appear on both MPU and DSP public peripheral buses:

- Mailbox registers for interprocessor communication
- General-purpose I/O (GPIO)
- UART1
- UART2
- UART/IrDA

Figure 10–1 shows the OMAP5910 device with the MPU/DSP peripherals highlighted.

Figure 10–1. Highlight of MPU/DSP Peripherals



10.2 Interprocessor Communication

The MPU and DSP processors communicate with each other via a mailbox-interrupt mechanism. This mechanism provides a very flexible software protocol between the processors. The mailboxes are located in the shared memory space (byte address 0xFFFC:E000 for MPU; word address 0x0F800 for DSP).

10.2.1 Mailbox Register Data Structure

There are four sets of mailbox registers: two for the MPU to send messages and issue an interrupt to the DSP, the other two for the DSP to send messages and issue an interrupt to the MPU. Each set of mailbox registers consists of two 16-bit registers and a 1-bit flag register. The interrupting processor can use one 16-bit register to pass a data word to the interrupted processor and the other 16-bit register to pass a command word.

Table 10–1 shows the mailbox registers, and Figure 10–2 shows the interrupt generating mechanism for the DSP-to-MPU scenario. The mechanism for MPU-to-DSP interrupt generation uses identical hardware.

The data word from the interrupting processor is user-defined but can be an address pointer or status information.

Upon writing to the command word processor, an interrupt is generated to the other processor and the 1-bit flag register is set. Use of the data word is optional and at the discretion of the software, but the data word must always be written before the command field. The ARM2DSP1 and ARMDSP2 interrupts are registered as INT5 and INT19, respectively, in the DSP. The DSP2ARM1 and DSP2ARM2 interrupts are mapped to the MPU level 1 interrupt handler as IRQ10 and IRQ11, respectively.

The interrupted processor must acknowledge the interrupt by reading the data word (if necessary) and the command word for the associated interrupt. The interrupt is reset and the 1-bit flag register is cleared when the command word is read by the interrupted processor. If software uses the data word, it must always read the data field prior to the command field.

The interrupts that are generated are level sensitive, and writing to the command register of any mailbox generates an interrupt. If the interrupt is masked in the interrupt handler when the command register is written, no interrupt is generated to the processor. However, the command flag for the particular mailbox is set. If the interrupt is unmasked at a later time, an interrupt is generated to the processor. Only the interrupting processor can read the corresponding flag bit (that is, only MPU can read ARM2DSP1_FLAG. So if polling is used

instead of interrupts, the command or data registers must be polled, not the flag register. The flag registers are only useful for the interrupting processor to see if the interrupted processor has responded to the interrupt by reading the command register.

By default, these interrupts are masked by the respective processor interrupt handler and must be unmasked for the mailbox mechanism to be used.

The following software setup procedures are provided as an example.

1) System software initializes all four of the mailboxes (during powerup or when the system must put the mailboxes in a known state).

2) System software enables the interrupt mask in the respective interrupt handler associated with each processor.

3) The interrupting processor writes to the mailbox data location with the data word information when it must alert to the word for the other processor (at this point, the associated word command for the other processor should not have been set yet).

4) The interrupting processor writes to the mailbox command word a predefined command (predefined and understood by both processors). This write issues the interrupt to the other processor.

5) In response to the interrupt, the interrupted processor acknowledges the interrupt by reading the mailbox registers. Reading the two locations is performed by the software protocol; the system software must read the data first and then read the command register (the associated interrupt and 1-bit flag register are cleared upon read).

6) System software examines the data and command words to determine what to do.

7) System software calls an interrupt service routine (ISR), to do whatever processing is necessary. System software returns to normal processing.

Note:

For the mailbox interrupt procedure, the use of the data word is optional and can be omitted. This eliminates step 3 and the first portion of step 5.

Base Address: 0xFFFC:E000 (byte) for MPU; 0x0F800 (word) for DSP

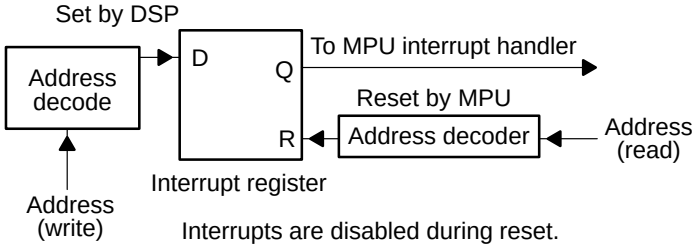
Table 10–1. Mailbox Registers

Bits	Name	Function	Byte Offset	Reset Value
15–0	ARM2DSP1	Writing to this location stores a software-defined data value to be used in conjunction with the ARM2DSP1 interrupt. Can be written only by the MPU.	0x00	0000
15–0	ARM2DSP1b	Writing to this location stores a software-defined command value in this register, issues the ARM2DSP1 interrupt, and sets the ARM2DSP1_Flag. When the DSP reads this register, the interrupt and the ARM2DSP1_Flag are cleared. This register must always be written, or read, after ARM2DSP1. Can be written only by the MPU.	0x04	0000
15–0	ARM2DSP2	Writing to this location stores a software-defined data value to be used in conjunction with the ARM2DSP2 interrupt. Can be written only by the MPU.	0x24	0000
15–0	ARM2DSP2b	Writing to this location stores a software-defined data value, issues the ARM2DSP1 interrupt, and sets the ARM2DSP2_Flag. When the DSP reads this register, the interrupt and the ARM2DSP2_Flag are cleared. This register must always be written, or read, after ARM2DSP2. Can be written only by the MPU.	0x28	0000
15–0	DSP2ARM1	Writing to this location stores a software-defined data value to be used in conjunction with the DSP2ARM1 interrupt. Can be written only by the DSP.	0x08	0000
15–0	DSP2ARM1b	Writing to this location stores a software-defined command value in this register, issues the DSP2ARM1 interrupt, and sets the DSP2ARM1_Flag. When the MPU reads this register, the interrupt and the DSP2ARM1_Flag are cleared. This register must always be written, or read, after DSP2ARM1. Can be written only by the DSP.	0x0C	0000
15–0	DSP2ARM2	Writing to this location stores a software-defined data value to be used in conjunction with the DSP2ARM2 interrupt. Can be written only by the DSP.	0x10	0000
15–0	DSP2ARM2b	Writing to this location stores a software-defined command value in this register, issues the DSP2ARM2 interrupt, and sets the DSP2ARM2_Flag. When the MPU reads this register, the interrupt and the DSP2ARM2_Flag are cleared. This register must always be written, or read, after DSP2ARM2. Can be written only by the DSP.	0x14	0000

Table 10–1. Mailbox Registers (Continued)

Bits	Name	Function	Byte Offset	Reset Value
15–1	ARM2DSP1_Flag	Reserved	0x18	xxxx
0		Flag indicating that the ARM2DSP1 interrupt has been generated. Set by MPU write to ARM2DSP1b; cleared by DSP read of ARM2DSP2b. This bit can only be read by the MPU.		
15–1	ARM2DSP2_Flag	Reserved	0x2c	xxxx
0		Flag indicating that the ARM2DSP2 interrupt has been generated. Set by MPU write to ARM2DSP2b; cleared by DSP read of ARM2DSP1b. This bit can only be read by the MPU.		
15–1	DSP2MPU1_Flag	Reserved	0x1C	xxxx
0		Flag indicating that the DSP2ARM1 interrupt has been generated. Set by DSP write to DSP2ARM1b; cleared by MPU read of DSP2ARM1b. This bit can only be read by the DSP.		
15–1	DSP2MPU2_Flag	Reserved	0x20	xxxx
0		Flag indicating that the DSP2ARM2 interrupt has been generated. Set by DSP write to DSP2ARM2b; cleared by MPU read of DSP2ARM2b. This bit can only be read by the DSP.		

Figure 10–2. Interrupt Generating Mechanism



10.3 General-Purpose I/O

The GPIOs (see Figure 10–3) are programmable inputs or outputs. They generate a level interrupt, and the sources of this interrupt can be masked from within the GPIO module. Under software control, the GPIOs can be individually dedicated to either the DSP or the MPU.

GPIOs are general-purpose input and output external pins available to the user for system-level control and general-purpose functions. The signals are user-defined as either input or output. The output state can be controlled. And inputs can be configured to provide an interrupt.

The MPU and the DSP have separate instances of the GPIO registers, but share the same device pins. The determination of whether MPU or DSP has control of the device pin is controlled by a single shared register, the pin control register (at offset 0x18). In Figure 10–3 this register is shown as the configuration and control register. This register is read/write from the MPU, but read-only from the DSP. The MPU is responsible for writing to this register to assign any necessary GPIO signals to the DSP. By default, all GPIO are assigned to the MPU.

GPIO interrupts are routed to both the MPU and DSP interrupt handlers, but a GPIO can only signal an interrupt to the processor to which it is assigned in the pin control register. By default, GPIO interrupts are disabled at both of the interrupt handlers.

There are no I/O signals associated with GPIO.5 and GPIO.10 interrupts.

10.3.1 Input/Outputs of the GPIO Module

Some GPIO signals are multiplexed with other peripheral functions. See Appendix A, *Input/Output Descriptions,* for pin multiplexing information.

10.3.2 GPIO Port Registers

Table 10–2 lists the GPIO port registers. Table 10–3 through Table 10–10 describe the individual registers.

Each register exists both in the DSP GPIO and the MPU GPIO, except for the pin control/pin status register.

Base Address: 0xFFFC:E000 (byte) for MPU; 0x0F000 (word) for DSP

Figure 10–3. GPIO Module Architecture

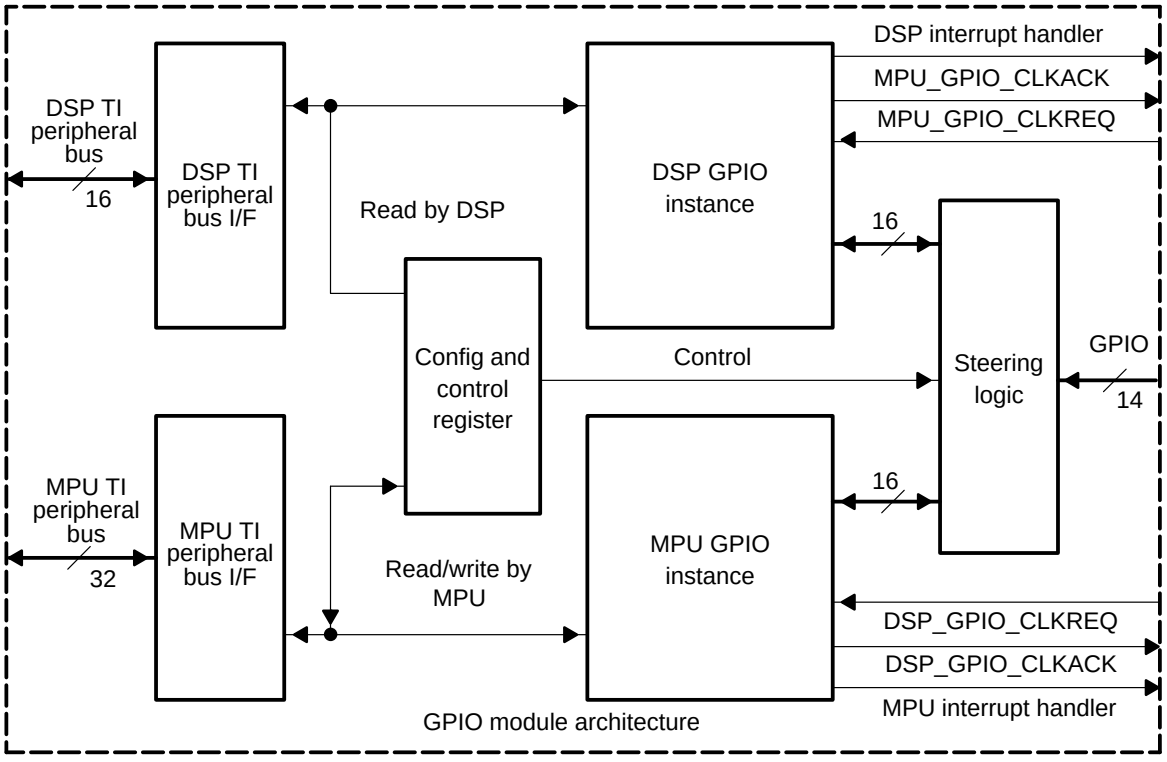


Table 10–2. GPIO Port Registers

Name	R/W	Size	Offset	Description
DATA_INPUT_REG	R	16 bits	0x00	Data input register
DATA_OUTPUT_REG	R/W	16 bits	0x04	Data output register
DIRECTION_CONTROL_REG	R/W	16 bits	0x08	Direction control register
INTERRUPT_CONTROL_REG	R/W	16 bits	0x0C	Interrupt control register
INTERRUPT_MASK_REG	R/W	16 bits	0x10	Interrupt mask register
INTERRUPT_STATUS_REG	R/W	16 bits	0x14	Interrupt status register
PIN_CONTROL_REG	R/W	16 bits	0x18	Pin control register (MPU only)
PIN_CONTROL_STATUS_REG	R	16 bits	0x18	Pin control status register (DSP only)

The data input register is used to register the data that is read from the GPIO input pins. The input data is captured synchronously and clocked by an internal peripheral clock. The data input register is a read-only register. The GPIO input data is captured into this register three clock cycles after the GPIO input pin(s) change for synchronization and debouncing used to remove any input glitches. Bits not configured as input are undefined during read back. Both the MPU and DSP have read access to this register, but each processor only has read access to the individual bits controlled by that processor.

In the registers described in Table 10–4 through Table 10–8, both the MPU and the DSP have read/write access to only the bits they control within a register (bits associated with GPIO they control). The MPU (DSP) can write values to the bits not controlled by the MPU (DSP), but the written value is not valid and does not affect the configuration of the associated GPIO.

Table 10–3. Data Input Register (DATA_INPUT_REG)

Bit	Function	Access (R/W)	Reset Value
15–0	Receive data	R	0x0000

The data output register is used for setting the state on the GPIO output pins.

Table 10–4. Data Output Register (DATA_OUTPUT_REG)

Bit	Function	Access (R/W)	Reset Value
15–0	Data to transmit	R/W	0xFFFF

The direction control register is used to configure the GPIO pins for either input or output. At reset, all of the GPIO pins are configured as inputs.

Table 10–5. Direction Control Register (DIRECTION_CONTROL_REG)

Bit	Value	Function	Access (R/W)	Reset Value
15–0	0	Output	R/W	0xFFFF
	1	Input		

The interrupt control register allows the user to define when an interrupt request occurs. The interrupt can either be generated from a high-to-low transition (function 0) or a low-to-high transition (function 1).

Table 10–6. Interrupt Control Register (INTERRUPT_CONTROL_REG)

Bit	Value	Function	Access (R/W)	Reset Value
15–0	0	Selects high to low transition	R/W	0xFFFF
	1	Selects low to high transition		

The interrupt mask register allows the user to mask(disable) certain input pins from generating an interrupt request.

Table 10–7. Interrupt Mask Register (INTERRUPT_MASK_REG)

Bit	Value	Function	Access (R/W)	Reset Value
15–0	0	Enables interrupt	R/W	0xFFFF
	1	Disables interrupt		

The interrupt status register is used to determine which of the input pins requested an interrupt. Bit 0 corresponds to GPIO0 and so forth. If the value is a 1, then that pin is requesting the interrupt. The processor services the interrupt and resets the appropriate bit in the status register. If the user wants to reset the status bit, then a 1 must be written to the appropriate bit. However, the user can not generate an interrupt by writing a 1 to the *interrupt status register*. If the user writes a 0 to a bit in the status register, the value remains unchanged.

Table 10–8. Interrupt Status Register (INTERRUPT_STATUS_REG)

Bit	Value	Function	Access (R/W)	Reset Value
15–0	0	0: No interrupt request	R/W	0x0000
	1	An interrupt has been requested		

The pin control register is only in the MPU. MPU is the master and is responsible for assigning the top-level GPIO I/O pins to either the MPU GPIO or the DSP GPIO. At reset, all pins are configured for MPU GPIO.

Table 10–9. MPU GPIO Pin Control Register (PIN_CONTROL_REG)

Bit	Value	Function	Access (R/W)	Reset Value
15–0	0	DSP GPIO pin	R/W	0xFFFF
	1	MPU GPIO pin		

The pin control status register is only in the DSP. This is a read-only register. The status register allows the DSP to find out how the MPU has configured the top-level GPIO pins.

Table 10–10. DSP GPIO Pin Control Status Register (PIN_CONTROL_STATUS_REG)

Bit	Value	Function	Access (R/W)	Reset Value
15–0	0	DSP GPIO pin	R	0xFFFF
	1	MPU GPIO pin		

10.4 UART1, UART2, and UART3/IrDA

The MPU and DSP share UART port 1, UART port 2, and the IrDA-capable UART3. For more details on the UARTs, see Chapter 12, *UART Devices*.

Chapter 11

LCD Controller

This chapter describes the LCD controller module of the OMAP5910 device.

Topic **Page**

11.1 Module Overview

The OMAP5910 device includes an LCD controller that interfaces with most industry-standard LCD displays. 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.

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.

Frame sizes and frame rates supported in specific applications depend upon the available memory bandwidth allowed by the application.

Figure 11–1 shows the OMAP5910 device with the LCD controller highlighted. Figure 11–2 shows the LCD controller in more detail.

The principal features of the LCD controller are:

- ❏ Dedicated 64-entry x 16-bit FIFO
- ❏ Dedicated LCD DMA channel for LCD display
- ❏ Programmable display including support for 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

Figure 11–1. LCD Controller on Board the OMAP5910 Device

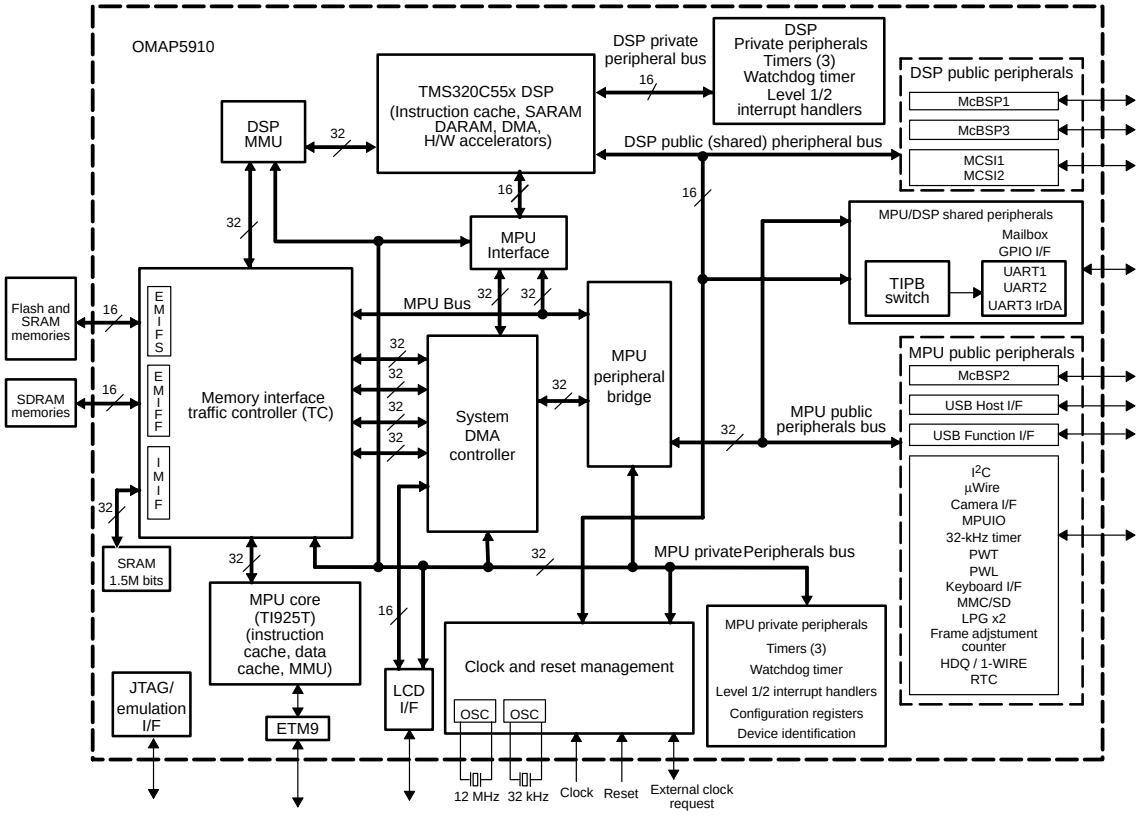
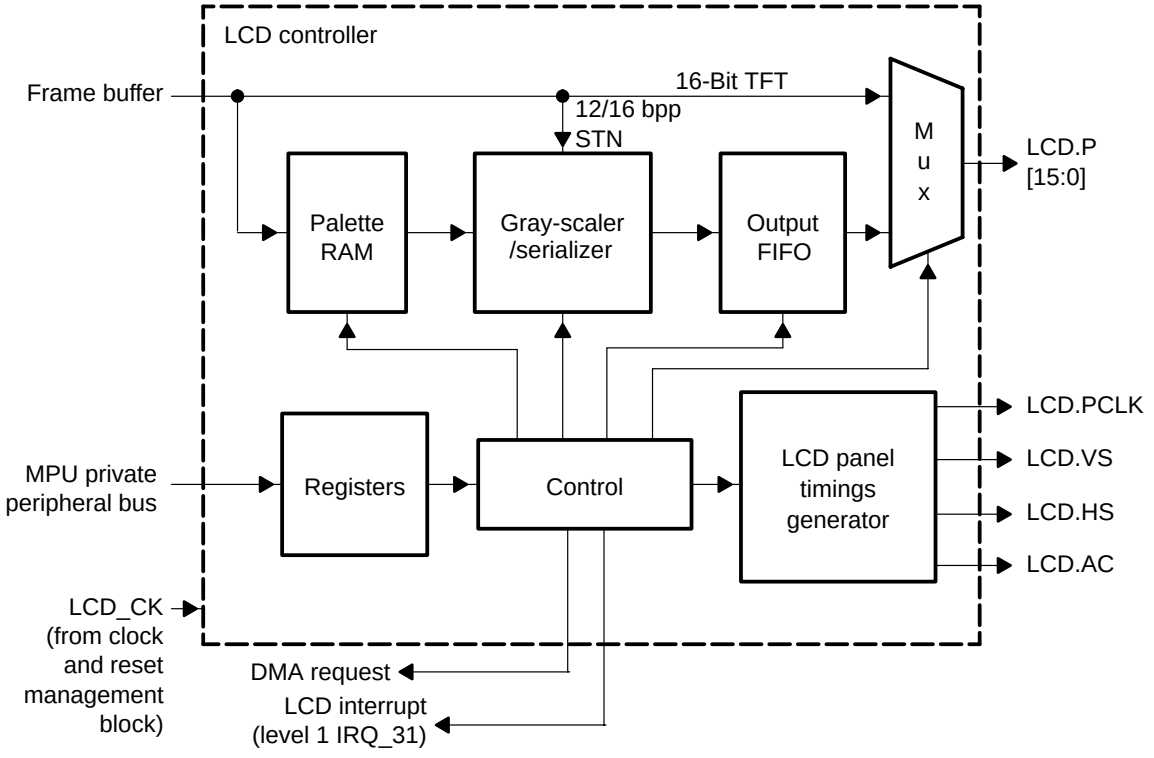


Figure 11–2. LCD Controller Block Diagram



Frame buffer data can be formatted for 2, 4, 8, 12, or 16-bit pixel sizes. A 16-entry x 12-bit palette supports the 2 and 4-bit pixel sizes, while a larger 256-entry x 12-bit palette supports the 8-bit pixel size. 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 palette is passed to the patented dither logic, where the desired brightness is created using spatial and 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 lower, 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 spatial and 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 utilizes 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 lower, 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 also 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 again 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.

The pixel clock frequency is derived from the clock provided to the LCD controller (LCD_CK) from the OMAP5910 clock management logic and is programmable from LCD_CK/2 to LCD_CK/255 (see Chapter 15, *Clock Generation and System Reset Management*). 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 are separated by a programmable number of line clock wait states. Program horizontal front porch (HFP) and horizontal back porch (HBP) 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 occur. 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.

Table 11–1 shows details relating to the LCD controller signals.

Table 11–1. Interface to LCD Panel Signal Descriptions

Name	Type	Destination	Description
LCD.P[15:0]	Out	LCD panel display	I/O pins used to transfer either four, eight, or sixteen 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] are 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, pixel clock only transitions 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 displays 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.

11.2 Display Specifications

The following information shows the number of palette entries and thus the number of possible screen colors per frame that can be displayed in each mode with the corresponding number of bits-per-pixel (BPP).

Mono passive: 1 BPP, 2 BPP, 4 BPP, and 8 BPP

- 1 BPP: Two palette entries selecting one of 15 grayscale
- 2 BPP: Four palette entries selecting one of 15 grayscale
- 4 BPP: 16 palette entries selecting one of 15 grayscale
- 8 BPP: 256 palette entries selecting one of 15 grayscale

Color passive: 2 BPP, 4 BPP, 8 BPP, 12, and 16 BPP

- 2 BPP: Four palette entries from 3375 possible colors
- 4 BPP: 16 palette entries from 3375 possible colors
- 8 BPP: 256 palette entries from 3375 possible colors
- 12 BPP: 3375 possible on-screen colors
- 16 BPP: 3375 possible on-screen colors

Active: 2 BPP, 4BPP, 8BPP, 12 BPP, and 16BPP

- 2 BPP: Four palette entries selecting from 4096 colors
- 4 BPP: 16 palette entries selecting from 4096 colors
- 8 BPP: 256 palette entries selecting from 4096 colors
- 12 BPP: Maximum 64K colors
- 16 BPP: Maximum 64K colors, depending on LCD panel

Palette entries are 16 bits wide (2 bytes) and therefore 2 and 4 BPP require 32 bytes of storage. 8 BPP modes require 512 bytes. 12 or 16 BPP modes do not use palette data but need the bits-per-pixel information to be loaded, so these modes use 32 bytes similar to that of the 2 and 4 BPP modes.

Mono passive mode supports two different interfaces: 4-bit panel and 8-bit panel. All modes (color/mono, 2, 4, 8, 12, or 16 bits-per-pixel) operate independently of each other.

The vertical synchronization signal (VSYNC) width must be programmed to be as small as possible on passive screen modes, but long enough to load the palette without stealing all the memory bandwidth from the MPU. To satisfy the system requirement, the following equation must be met:

$$\{256 + (15 * FDD)\} < \left\{\left(HBP + HFP + \frac{(PPL + 1)}{d} + HSW + 3\right) * VSW * PCD\right\}$$

d	Display
1	TFT
2 2/3	STN color
4	Mono 4 bits

Note: If the condition is not true, the LCD controller displays a black screen every other frame.

Pixels-per-line (PPL) must be in multiples of 16. Most LCD panels ignore data at the end of the line that is not needed—that is, they ignore data at the right hand side of the screen.

11.3 LCD Controller Operation

The LCD controller supports a variety of user-programmable options, including display type and size, frame buffer pixel size, and output data width. Although all programmable combinations are possible, the selection of displays available within the market dictate which combinations of these programmable options are practical. In addition, the type of external memory system implemented by the user limits the bandwidth of the LCD DMA controller, which in turn limits the size and type of screen that can be controlled.

The following sections describe individual functional blocks within the LCD controller, the frame buffer and palette memory organization, and the LCD DMA controller. The sections are arranged in order of data flow, starting with the off-chip frame buffer and ending with the pins that interface to the LCD display.

11.3.1 Frame Buffer

The frame buffer is an area within on-chip SRAM or off-chip memory that is used to supply enough encoded pixel values to fill the entire screen one time. The first 32 bytes of the buffer (for 2-, 4-, 12-, and 16-bit mode operation, 512 bytes for 8 BPP mode of operation) are used to store the look-up palette data for each frame. Not all of the 16 entries of the palette are used in 2 BPP mode. However, all 16 palette entries must be present. The palette is not used for 12 or 16 bits-per-pixel encoding. The 32 bytes at the top of the frame buffer, however, must be zero-filled even though the data is not used. This is to provide the bits-per-pixel to the LCD controller.

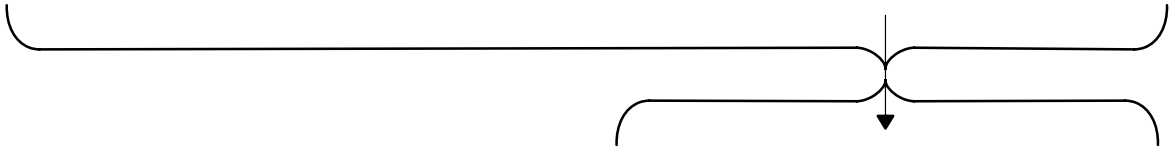
Each time a new frame is fetched from the frame buffer, the LCD controller palette is first loaded with data contained within the palette buffer (this is the default setting). Figure 11–3 and Figure 11–4 show the palette entry organization. You can configure the LCD palette loading by setting the LCD control register bits 21-20.

Figure 11–3.256 Palette Entry/Buffer Format (8 BPP)

Individual Palette Entry

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Color	n/u	BPP†			Red (R)				Green (G)				Blue (B)			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mono	n/u	BPP†			Unused								Mono (M)			

† Bits-per-pixel (BPP) is only contained within the first palette entry (palette entry0).



256 Entry Palette Buffer

Bit	15 … 0
Base + 0x0	Palette entry 0
Base + 0x2	Palette entry 1
Base + 0x1FC	Palette entry 254
Base + 0x1FE	Palette entry 255
Base + 0x200	Start of pixel data

Figure 11–4.16 Palette Entry/Buffer Format (1, 2, 4, 12, 16 BPP)

16 Entry Palette Buffer

Bit	15 … 0
Base + 0x0	Palette entry 0
Base + 0x2	Palette entry 1
Base + 0x1C	Palette entry 14
Base + 0x1E	Palette entry 15
Base + 0x20	Start of pixel data

The first palette entry (palette entry 0) also contains an extra field that is used to configure the LCD controller synchronously at the beginning of each frame. Bits 12, 13, and 14 of the first palette entry contain a field that is used to select the number of bits-per-pixel that is to be used in the following frame and the number of entries that are used in the palette RAM. The bits-per-pixel (BPP) bit-field is decoded by the LCD to correctly unpack pixel data into 1-bit, 2-bit, nibbles, bytes, 12-bit values, or words, and it is decoded by the palette to tell it how many address bits are contained in the pixel data it is supplied, configuring the palette size to 16 or 256 entries. The 12- and 16-bit pixel modes bypass the LCD palette and supply 12-bit values directly to the dither logic when passive mode is enabled or else supply 16-bit values directly to the output FIFOs when active mode is enabled. Table 11–2 shows the encoding of the BPP bit field.

Table 11–2. Bits Per Pixel Encoding for Palette Entry 0 Buffer

Bit	Name	Value	Description
14–12	BPP		Bits-per-pixel
		001	2 bits-per-pixel
		010	4 bits-per-pixel
		011	8 bits-per-pixel
		1xx	12 bits-per-pixel and 16 bits-per-pixel

Note: Four 2-bit pixels and two 4-bit pixels are packed into each byte, and 12-bit pixels are right-justified on half-word boundaries (in the same format as palette entry).

Following the palette buffer is the pixel data buffer that contains one encoded pixel value for each of the pixels present on the display. The number of pixel data values depends on the size of the screen (that is, 1024 x 768 = 786,432 encoded pixel values). Again, each pixel data value can be 2, 4, 8, 12, or 16 bits wide. Figure 11–5 through Figure 11–9 show the memory organization within the frame buffer for each size pixel encoding. For 4-bit encoding, four pixels are placed into each half-word; for 12-bit encoding, the value is right-justified within a half-word.

Figure 11–5.2 BPP Frame Buffer Memory Organization

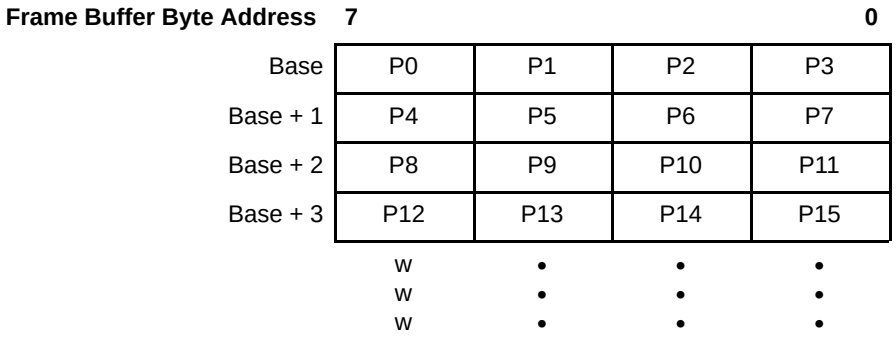


Figure 11–6.4 BPP Frame Buffer Memory Organization

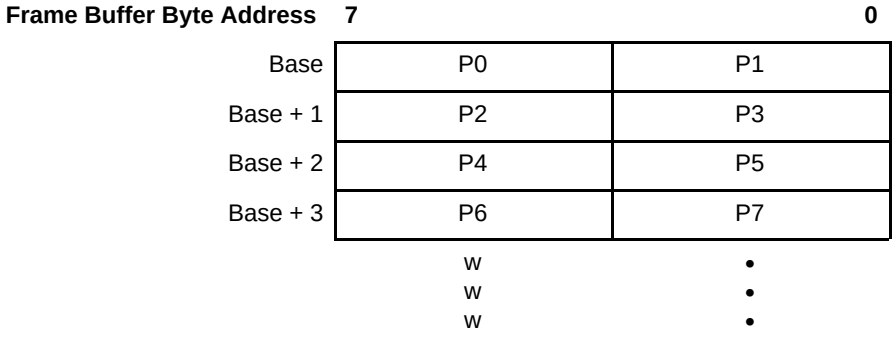


Figure 11–7.8 BPP Frame Buffer Memory Organization

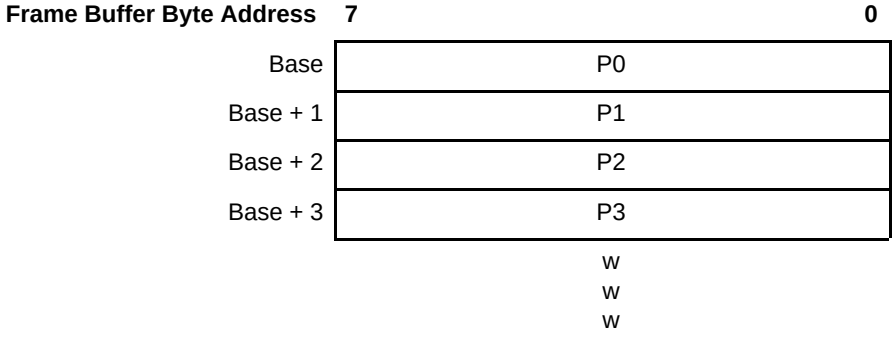


Figure 11–8.12 BPP Frame Buffer Memory Organization

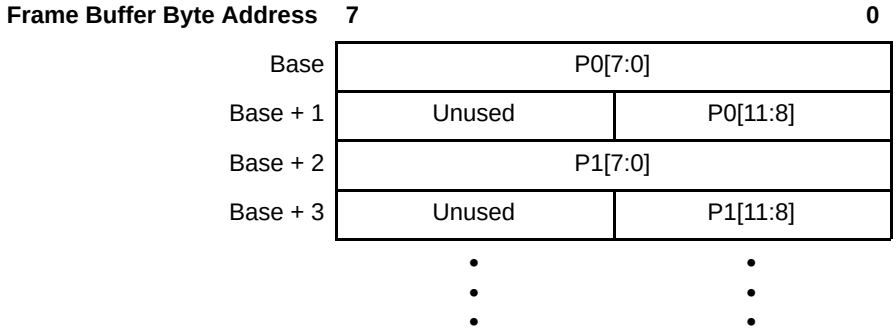
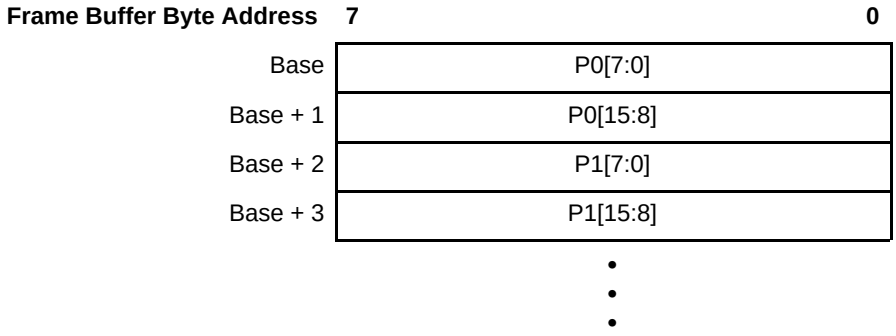


Figure 11–9.16 BPP Frame Buffer Memory Organization



The OMAP5910 MPU operates in little endian mode and the number and position of pixels in an access depend on access type (byte, half-word, or word). For example, if the LCD controller is in 2 BPP mode and the MPU performs a read at the beginning of the frame buffer, the result of the read is:

Byte access (8-bit read):

P0 P1 P2 P3

Half-word access (16-bit read):

P4 P5 P6 P7 P0 P1 P2 P3

Word access (32-bit read):

P12 P13 P14 P15 P8 P9 P10 P11 P4 P5 P6 P7 P0 P1 P2 P3

The pixel data is stored in the frame buffer the same way in all three cases (as shown in Figure 11–5); only the little endian accesses of the MPU result in the different pixel positions of each access.

The top and bottom addresses of the frame buffer (palette entries + pixels data) are programmed in the DMA controller. A synchronization interrupt occurs if the LCD display information settings such as pixels-per-line, lines per frame, color/monochrome mode, and bits-per-pixel that are programmed by the user are not in accordance with the size of the frame buffer as programmed in the DMA.

The following equations are used to calculate the total frame buffer size (in bytes) to be programmed in the system DMA, based on varying pixel size encoding and screen sizes.

$$\textit{For } 2\ \textit{bits/pixel}: \quad \textit{FrameBufferSize} = 32 + \frac{(\textit{Lines} * \textit{Columns})}{4}$$

$$\textit{For } 4\ \textit{bits/pixel}: \quad \textit{FrameBufferSize} = 32 + \frac{(\textit{Lines} * \textit{Columns})}{2}$$

$$\textit{For } 8\ \textit{bits/pixel}: \quad \textit{FrameBufferSize} = 512 + (\textit{Lines} * \textit{Columns})$$

$$\textit{For } 12/16\ \textit{bits/pixel}: \quad \textit{FrameBufferSize} = 32 + 2(\textit{Lines} * \textit{Columns})$$

11.4 Lookup Palette

The encoded pixel data from the input FIFO is used as an address to index and select individual palette locations: 2-bit pixels address four locations, 4-bit pixels address sixteen locations, and 8-bit pixels select any of the 256 palette entries.

When a palette entry is selected by the encoded pixel value, the contents of the entry are sent to the color/grayscale space/time base dither circuit. In color mode, the value within the palette is made up of three 4-bit fields, one for each color component: red, green, and blue. In monochrome mode, only one 4-bit value is present. For both modes, the 4-bit values represent 1 of 15 intensity levels. For color operation, an individual frame is limited to a selection of 256 colors (the number of palette entries). The LCD controller, however, can generate a total of 3375 colors (15 levels per color x 3 colors). When 12 or 16 bit-per-pixel mode is enabled, the palette is bypassed. For passive displays, 12-bit pixels and 16-bit pixels are sent directly to the dither logic.

11.5 Color/Grayscale Dithering

Entries selected from the lookup palette are sent to the color/grayscale space/timebase dither generator. Each 4-bit value is used to select one of 15 intensity levels. Two of the 16 dither values are identical (most intense). 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 visual perception of color/gray gradations. In color mode, three separate dither blocks are used to process the three color components: red, green, and blue.

The duty cycle and resultant intensity level for all 15 color/grayscale levels is summarized in Table 11–3.

Table 11–3. 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

11.6 Output FIFO

The LCD controller contains a 2-entry by 8-bit wide output FIFO that is used to store pixel pin data before it is driven out to the 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 in the LCD control registers. 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 (4-, 8-, or 2 2/3-pixel values), 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.

Note:

The output FIFO is bypassed in TFT mode.

11.7 LCD Controller Pins

When the shifter is filled, the value is driven to the LCD controller data bus pins in one of several configurations: LCD.P[3:0] for passive monochrome panels, LCD.P[7:0] for passive color panels, and LCD.P[15:0] for active displays. In addition, the pixel clock pin (LCD.PCLK) is toggled. The remaining unused LCD pixel bits always remain low.

When an entire line of pixels has been output to the LCD screen, the line clock pin (LCD.HS) is toggled. In the same manner, if the controller is in passive mode and the start of the first line of a new frame of pixels has been output to the LCD controller screen, the frame clock pin (LCD.VS) is toggled. To prevent a dc charge from building within the screen pixels, the display power and ground supplies are periodically switched. The LCD controller signals the display to switch the polarity by toggling the ac-bias pin (LCD.AC). The user can control the frequency of the bias pin by programming the number of line clock transitions between each toggle.

When active display mode is enabled, the timing of the pixel, line, and frame clocks and the ac-bias pin change. The pixel clock transitions continuously in this mode for as long as the LCD is enabled. The ac-bias pin functions as an output enable. When it is asserted, the display can use it to latch data from the LCD pins using the pixel clock.

The timing of the line and frame clock pins is programmable to support both passive and active mode. Programming options include:

- Delay insertion both at the beginning and end of each line and frame (front and back porch)
- Pixel clock, line clock, frame clock, and ac-bias signal polarity
- Line and frame clock pulse width

If the LCD is disabled, the signals LCD.P[15:0] are set to 0 and LCD.PCLK, LCD.VS, LCD.HS, and LCD.AC are set to their inactive state. This can be 0 or 1 depending on the inversions programmed in the timing 2 register. See Table 11–24.

The OMAP5910 LCD controller provides outputs compatible with passive monochrome, passive color (STN), and active color (TFT) displays. Recommended connections to each type of display are outlined in the sections below.

11.7.1 Passive Monochrome Panels

Passive monochrome displays can be supported for graphics depths of 8 BPP (256 entry palette), 4 BPP, 2 BPP, or 1 BPP. For passive monochrome displays, four signals are supplied. Each signal represents one pixel that is dithered over successive frames to achieve a maximum of 15 gray levels (see Table 11–4).

Table 11–4. Passive Monochrome Panel Inputs

OMAP5910 LCD Controller Output	Passive Monochrome Panel Input
LCD.P[0] (leftmost pixel)	D[3]
LCD.P[1]	D[2]
LCD.P[2]	D[1]
LCD.P[3] (rightmost pixel)	D[0]

11.7.2 Passive Color (STN) Panels

Passive color displays can be supported for palletized graphics depths of 2 BPP, 4 BPP, and 8 BPP (256 color palette), as well as direct graphics depths of 12 BPP and 16 BPP. For passive color displays, eight signals are supplied. Each signal represents one color channel of one pixel that is dithered over successive frames to achieve a maximum of 15 shade levels per color (for a total of 15x15x15=3375 colors). This means that each set of eight signals represents 2 2/3 pixels (8 signals/3 colors per pixel). Table 11–5 shows the relationship of these signals to the color channel and pixel position on screen.

Table 11–5. 8-Bit Panel

OMAP5910 LCD Controller Output	Passive Color Panel Input
LCD.P[7] (0 red, 2 blue, 5 green…)	D[7]
LCD.P[6] (0 green, 3 red, 5 blue…)	D[6]
LCD.P[5] (0 blue, 3 green, 6 red…)	D[5]
LCD.P[4] (1 red, 3 blue, 6 green…)	D[4]
LCD.P[3] (1 green, 4 red, 6 blue…)	D[3]
LCD.P[2] (1 blue, 4 green, 7 red…)	D[2]
LCD.P[1] (2 red, 4 blue, 7 green…)	D[1]
LCD.P[0] (2 green, 5 red, 7 blue…)	D[0]

11.7.3 Active Color (TFT) Panels

Active color displays can be supported for palletized graphics depths of 2 BPP, 4 BPP, and 8 BPP (256 color palette), as well as direct graphics depths of 12 BPP and 16 BPP. When displaying 16 BPP, the 16 output signals are mapped directly to the 16 bits in the frame buffer memory. When displaying 12 BPP or less, the 12-bit pixel values (direct or from the palette) are mapped to the full 16 signal lines to provide a full-scale-corrected display. In this case, five bits of red and blue data are provided with six bits of green data. If this same orientation is used for the 16 BPP mode, the signal configuration is constant for all modes. Connecting these signals to the appropriate input signals of the panel allows support of a color TFT panel of any color depth. Table 11–6 to Table 11–9 illustrate the relationship of these signals to the most common panel types.

Note:

The actual number of colors displayed is limited to the smaller of $2^{\text{output depth}}$ and $2^{\text{panel input pins}}$.

Connecting a 12-bit panel for 16 BPP operation involves truncating the 16 bits of data to the 12 bits required by the panel (see Table 11–6).

Table 11–6. 16-Bit Per Pixel and 12-Bit Panel

OMAP5910 LCD Controller Output	12-Bit TFT Panel Input
LCD.P[15] (red[4])	red[3]
LCD.P[14] (red[3])	red[2]
LCD.P[13] (red[2])	red[1]
LCD.P[12] (red[1])	red[0]
LCD.P[11] (red[0])	n/c
LCD.P[10] (green[5])	green[3]
LCD.P[9] (green[4])	green[2]
LCD.P[8] (green[3])	green[1]
LCD.P[7] (green[2])	green[0]
LCD.P[6] (green[1])	n/c
LCD.P[5] (green[0])	n/c
LCD.P[4] (blue[4])	blue[3]

Table 11–6. 16-Bit Per Pixel and 12-Bit Panel (Continued)

OMAP5910 LCD Controller Output	12-Bit TFT Panel Input
LCD.P[3] (blue[3])	blue[2]
LCD.P[2] (blue[2])	blue[1]
LCD.P[1] (blue[1])	blue[0]
LCD.P[0] (blue[0])	n/c

Connecting a 15-bit panel for 16 BPP operation involves truncating the 16 bits of data to the 15 bits required by the panel (see Table 11–7).

Table 11–7. 16-Bit or Per Pixel and 15-Bit Panel

OMAP5910 LCD Controller Output	12-Bit TFT Panel Input
LCD.P[15] (red[4])	red[4]
LCD.P[14] (red[3])	red[3]
LCD.P[13] (red[2])	red[2]
LCD.P[12] (red[1])	red[1]
LCD.P[11] (red[0])	red[0]
LCD.P[10] (green[5])	green[4]
LCD.P[9] (green[4])	green[3]
LCD.P[8] (green[3])	green[2]
LCD.P[7] (green[2])	green[1]
LCD.P[6] (green[1])	green[0]
LCD.P[5] (green[0])	n/c
LCD.P[4] (blue[4])	blue[4]
LCD.P[3] (blue[3])	blue[3]
LCD.P[2] (blue[2])	blue[2]
LCD.P[1] (blue[1])	blue[1]
LCD.P[0] (blue[0])	blue[0]

Connecting an 18-bit panel for 16 BPP operation involves replicating (note *'s below) the 16 bits of data to fill in the entire 18 bits required by the panel. This is preferable to hardwiring the extra bits to a constant value, which reduces the dynamic range of the display and causes a color error (see Table 11–8).

Table 11–8. 16-Bit Per Pixel and 18-Bit Panel

OMAP5910 LCD Controller Output	18-Bit TFT Panel Input
LCD.P[15] (red[4])	red[5]
LCD.P[14] (red[3])	red[4]
LCD.P[13] (red[2])	red[3]
LCD.P[12] (red[1])	red[2]
LCD.P[11] (red[0])	red[1]
*LCD.P[15] (red[4])	red[0]
LCD.P[10] (green[5])	green[5]
LCD.P[9] (green[4])	green[4]
LCD.P[8] (green[3])	green[3]
LCD.P[7] (green[2])	green[2]
LCD.P[6] (green[1])	green[1]
LCD.P[5] (green[0])	green[0]
LCD.P[4] (blue[4])	blue[5]
LCD.P[3] (blue[3])	blue[4]
LCD.P[2] (blue[2])	blue[3]
LCD.P[1] (blue[1])	blue[2]
LCD.P[0] (blue[0])	blue[1]
*LCD.P[4] (blue[4])	blue[0]

Connecting a 24-bit panel for 16 BPP operation involves replicating (note *'s below) the 16 bits of data to fill in the entire 24 bits required by the panel. This is preferable to hard-wiring the extra bits to a constant value, which reduces the dynamic range of the display and causes a color error (see Table 11–9).

Table 11–9. 16-Bit-Per-Pixel and 24-Bit Panel

OMAP5910 LCD Controller Output	24-Bit TFT Panel Input
LCD.P[15] (red[4])	red[7]
LCD.P[14] (red[3])	red[6]
LCD.P[13] (red[2])	red[5]
LCD.P[12] (red[1])	red[4]
LCD.P[11] (red[0])	red[3]
*LCD.P[15] (red[4])	red[2]
*LCD.P[14] (red[3])	red[1]
*LCD.P[13] (red[2])	red[0]
LCD.P[10] (green[5])	green[7]
LCD.P[9] (green[4])	green[6]
LCD.P[8] (green[3])	green[5]
LCD.P[7] (green[2])	green[4]
LCD.P[6] (green[1])	green[3]
LCD.P[5] (green[0])	green[2]
*LCD.P[10] (green[5])	green[1]
*LCD.P[9] (green[4])	green[0]
LCD.P[4] (blue[4])	blue[7]
LCD.P[3] (blue[3])	blue[6]
LCD.P[2] (blue[2])	blue[5]
LCD.P[1] (blue[1])	blue[4]
LCD.P[0] (blue[0])	blue[3]
*LCD.P[4] (blue[4])	blue[2]
*LCD.P[3] (blue[3])	blue[1]
*LCD.P[2] (blue[2])	blue[0]

11.8 LCD Controller Registers

The LCD controller contains four control registers and one status register.

The control registers contain bit fields to enable and 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

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 enough bus bandwidth is given to other systems access. This field is only used for palette load.

The status register contains bits that signal:

- FIFO underrun error
- Frame synchronization error
- When the last active frame has completed after the LCD is disabled (maskable)
- ac counter, if programmed

Each of these hardware-detected events signals an interrupt request to the interrupt controller.

Table 11–10 lists the LCD controller registers. Table 11–11 through Table 11–23 describe the register bits.

Table 11–10. LCD Controller Registers

Register	Description	R/W	Size	Address
LcdControl	LCD control	R/W	32 bits	FFFE:C000
LcdTiming0	LCD timing 0	R/W	32 bits	FFFE:C004
LcdTiming1	LCD timing 1	R/W	32 bits	FFFE:C008
LcdTiming2	LCD timing 2	R/W	32 bits	FFFE:C00C
LcdStatus	LCD status	R/W	32 bits	FFFE:C010
LcdSubpanel	LCD subpanel display	R/W	32 bits	FFFE:C014

11.8.1 LCD Control Register 1 (LCDControl)

Table 11–11. LCD Control Register (LCDControl)

Bit	Name	Value	Description	Reset Value
31–25	–		Reserved	0
24	5-6-5 STN		12 BPP (5-6-5) mode	0
		0	On	
		1	Off	
			16 bits of data are in the frame buffer, but only 12 bits are dithered and sent out.	
23	TFT Map		TFT alternate signal mapping:	0
		0	Output pixel data for 1, 2, 4, and 8 BPP modes are right aligned on LCD pins (11:0)	
		1	Output pixel data for 1, 2, 4, and 8 BPP are converted to 5-6-5 format using pins (15:0) R3 R2 R1 R0 R3 G3 G2 G1 G0 G3 G2 B3 B2 B1 B0 B3	
22	LCDCB1		LCD control bit 1	0
			See Table 11–16 for proper settings for this field.	
21–20	PLM		Palette loading mode. Must precede data-loading-only mode.	0
		00	Palette and data loading, reset value	
		01	Palette loading	
		10	Data loading	
19–12	FDD		FIFO DMA request delay	0
			Encoded value (0–255) used to specify the number of memory controller clocks. The input FIFO DMA request must be disabled. The clock count starts after 16 words read in the input FIFO. Programming FDD = 00h disables this function.	
11–10	–		Reserved	0

Table 11–11. LCD Control Register (LCDControl) (Continued)

Bit	Name	Value	Description	Reset Value
9	M8B		Mono 8-bit mode. Selects 4 or 8 data lines to output pixel data to the screen.	0
		0	LCD_PIXEL[3:0] is used to output four pixel values to the panel each pixel clock transition.	
		1	LCD_PIXEL[7:0] is used to output eight pixel values to the panel each pixel clock transition.	
			This bit is ignored in al other modes.	
8	LCDCB0		LCD control bit 0. Used with LCD control bit 1 to control mapping of pixel data from the frame buffer to the output bus LCD.P[16:0].	0
			See Table 11–16 for proper settings for this field.	
7	LCDTFT		LCD TFT	0
		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 bypassed, pin timing changes to support continuous pixel clock, output enable, VSYNC, HSYNC signals	
5–6	–		Reserved	0
4	LoadMask		Load mask	0
		0	Mask out the loaded palette interrupt	
		1	Mask not active	
3	Done-Mask		Done mask	0
		0	Mask out the frame done (done) interrupt	
		1	Mask not active	
2	-		Reserved	0
1	LCDBW		LCD Monochrome	0
		0	Color operation enable	
		1	Monochrome operation enabled	
0	LCDEN		LCD controller enable	0
		0	LCD controller disabled	
		1	LCD controller enabled	

Table 11–12 lists suggested LCD register settings for various operating modes.

Table 11–12. LCD Control Register Settings

Panel Type	Graphics Mode	Register Setting	First Palette Entry
Monochrome	2 BPP	0x00400002	0x1XXX
Monochrome	4 BPP	0x00400002	0x2XXX
Monochrome	8 BPP	0x00010002	0x3XXX
Passive color	2 BPP	0x00400000	0x1XXX
Passive color	4 BPP	0x00400000	0x2XXX
Passive color	8 BPP	0x00010000	0x3XXX
Passive color	12 BPP	0x00000000	0x4XXX
Passive color	16 BPP	0x01000000	0x4XXX
Active color	2 BPP	0x00C00080	0x1XXX
Active color	4 BPP	0x00C00080	0x2XXX
Active color	8 BPP	0x00800080	0x3XXX
Active color	12 BPP	0x00800080	0x4XXX
Active color	16 BPP	0x00000080	0x4XXX

Bits Per Pixel STN Mode (5-6-5 STN)

The 16 BPP STN mode is handled similarly to the 12 BPP mode. The differences are in how the pixel data is organized in the frame buffer and in which bits are sent to the dither logic.

The 12-bit STN mode remains the same in the frame buffer memory as: 4-4-4.

Table 11–13. 12-Bit STN Data in Frame Buffer

	Unused				Red				Green				Blue			
Pins	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data	Data ignored				R3	R2	R1	R0	G3	G2	G1	G0	G3	G2	G1	G0

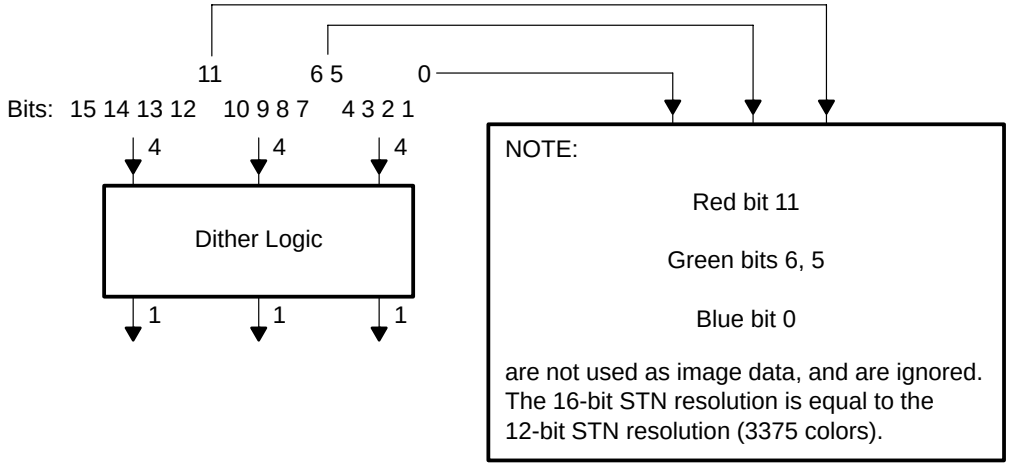
The 16-bit STN mode appears in the frame buffer memory as follows: 5-6-5.

Table 11–14. 16-Bit STN Data in Frame Buffer

	Red					Green						Blue				
Pins	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0

The 16-bit STN mode sends only 12 bits to the dither logic (bits 11, 6, 5, and 0 are not sent to dither logic). These bits are the 4 MSBs of each color. Figure 11–10 shows the dither logic.

Figure 11–10. Dither Logic



The 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.

16 Bits Per Pixel STN Mode

The 16-bit per pixel (BPP) STN mode is used to enable display of 16 BPP data on a passive color display. When this bit is enabled, data stored in the frame buffer as 16 BPP (5 bits red, 6 bits green, 5 bits blue) is converted internally to 12 bits (4 bits red, 4 bits green, 4 bits blue) for input into the STN dither logic. This bit does not affect 12 BPP mode.

TFT Alternate Signal Mapping (TFT Map)

This bit controls how the TFT pixel data are output.

When this bit is set to 1, the four red bits, the four green bits, and the four blue bits are mapped to all LCD.P[15:0] output pins, as shown in Table 11–15.

Table 11–15. 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	G2	G1	G0	G3	G2	B3	B2	B1	B0	B3

When this bit is set to 0 (default), the four red bits, the four green bits, and the four blue bits are right aligned on LCD.P[11:0] pins. The upper LCD.P[15:12] are set to 0.

LCD Control Bit 1

The LCD control bit 1 is used along with LCD control bit 0 to control the mapping of pixel data from the frame buffer to the output bus LCD.P[15:0]. Table 11–16 shows the appropriate settings for this bit.

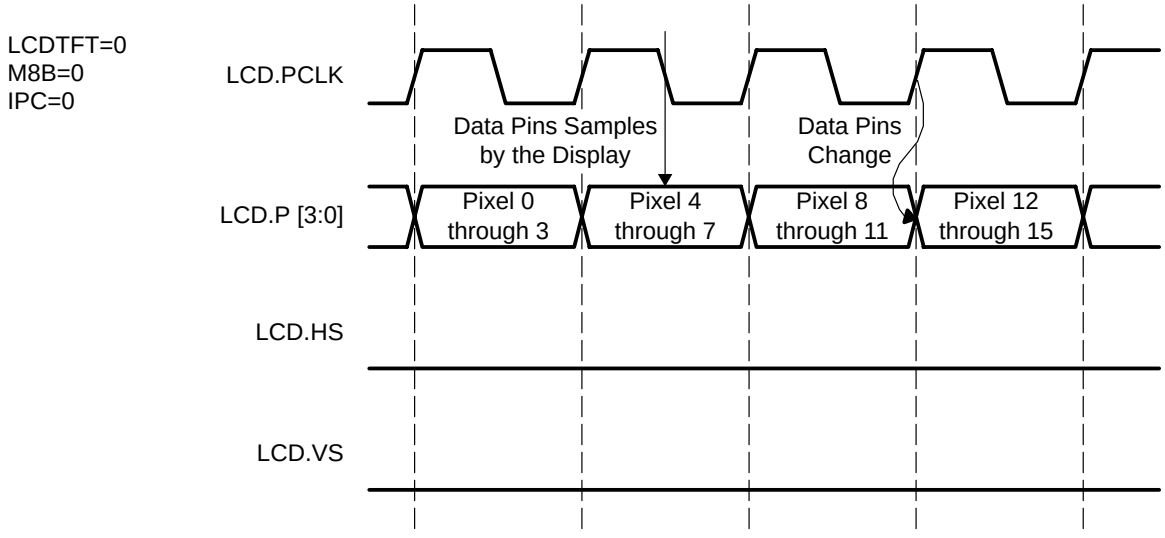
Table 11–16. Control Bit 0 And Control Bit 1 Mapping by Display Types

Display Type	Mode	Control Bit 0	Control Bit 1
Passive monochrome	2 BPP	0	1
	4 BPP	0	1
	8 BPP	0	0
Passive color	8 BPP	0	0
	12 BPP	0	0
TFT	16 BPP	0	0

LCD TFT (LCDTFT)

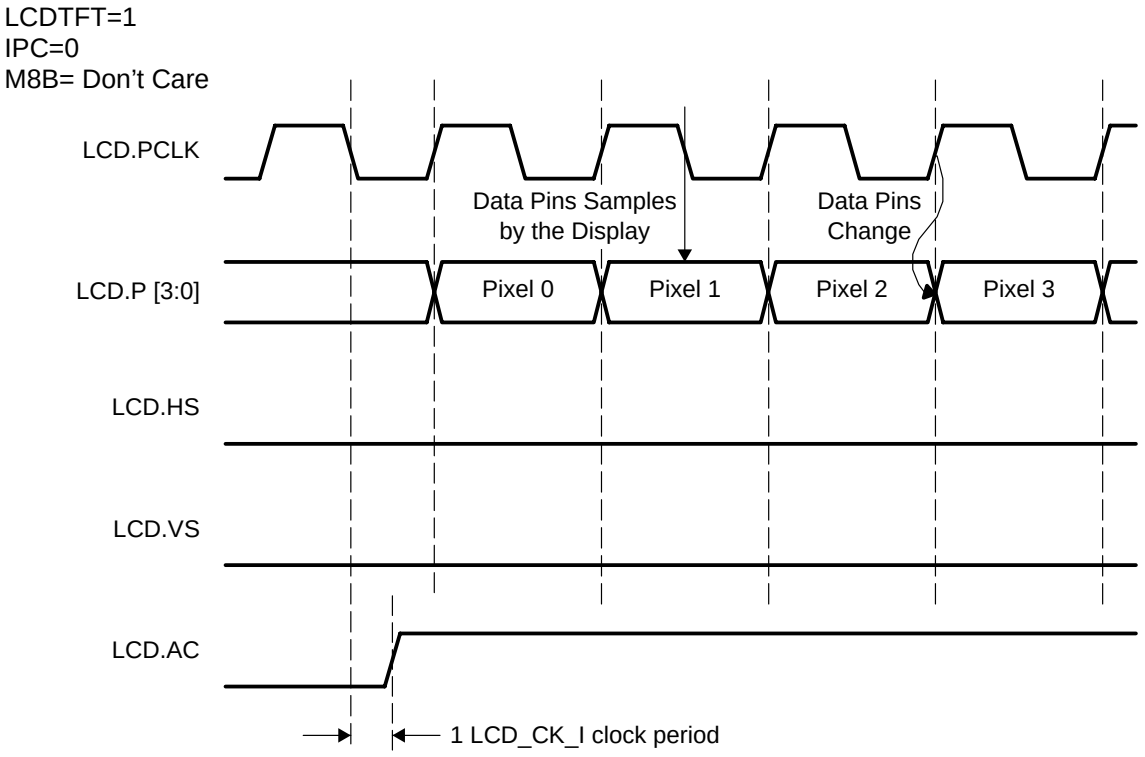
The LCD TFT (LCDTFT) bit selects whether the LCD controller operates in passive (STN) or active (TFT) display control mode. When LCDTFT = 0: passive or STN mode is selected; all LCD data flow operates normally (including the use of the LCD dither logic); and all LCD controller pin timing operates as described in Section 11.7, *LCD Controller Pins*. When LCDTFT = 1, active or TFT mode is selected. Frame data is transferred via the DMA from off-chip memory to the input FIFO, is unpacked, and is used to select an entry from the palette (for 1, 2, 4, and 8 bits-per-pixel modes), just as for passive mode (see Figure 11–11).

Figure 11–11. Passive Mode Pixel Clock and Data Pin Timing



The value read from the palette, however, bypasses the LCD dither logic and is sent directly to 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). Thus, two 16-bit values are packed into each word in the frame buffer. See Figure 11–12.

Figure 11–12. 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 (which only supports 3-bit RGB dithering). Increasing the size of the pixel representation allows a total of 64K colors to be addressed using an off-chip palette that is used in conjunction with the LCD controller.

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 (4 bits per color).
- ❏ All three dither blocks are used: one each for the red, green, and blue pixel components.
- ❏ Palette entries are 4 bits wide (15 levels of grayscale).
- ❏ Four or eight data lines are enabled.

Table 11–17 shows which set of LCD data pins (and LCD.P pins) is used for each mode of operation.

Table 11–17. LCD Controller Data Pin Utilization for Mono/Color, Passive/Active Panels

Color/Mono	Passive/Active Panel	Screen Portion	Pins
Mono 2, 4, 8	Passive	Whole	LCD_PIXEL[3:0]
Color 2, 4, 8, 12, 16	Passive	Whole	LCD_PIXEL[7:0]
Color 2, 4, 8, 16	Active	Whole	LCD_PIXEL[15:0]

LCD Enable (LCDEN)

The LCD enable (LCDEN) bit is used to enable and disable LCD controller operation. When LCDEN = 0, the LCD controller is disabled. When LCDEN = 1, the LCD controller is enabled.

Note:

All other control registers must be initialized before setting LCDEN.

You program LCDControl last, and you can configure all eight bit fields at the same time via a word write to the register. If the user clears LCDEN while the LCD controller is enabled, it is permitted to complete transmission of the current frame before being disabled. Completion of the current frame is signaled by the DMA when it sets the frame done bit (Done) within the LCD status register, which generates an interrupt request.

Table 11–18 shows the location of all seven bit fields located in the LCD control register (LCDControl). LCDEN is the only control bit that is reset to a known state, ensuring that the LCD is disabled after a reset of the LCD controller. The user must program all other control bit fields before setting LCDEN = 1 (a half-word or word write can be used to configure the whole register while setting LCDEN) and must also disable the LCD controller when changing the state of a control bit within the LCD controller.

Note:

Writes to reserved bits are ignored, and reads return 1s.

The LCD timing 0 register contains four bit fields that are used as modulus values for a collection of down counters, each of which performs a different function to control the timing of several of the LCD pins.

The LCD controller must be disabled (LCDEN = 0) when changing the state of any field within this register. The reset state of all bit fields is unknown and must be initialized before enabling the LCD.

11.8.2 LCD Timing 0 Register (LcdTiming0)

Table 11–18 describes the LCD timing 0 register (LcdTiming0) bits.

Table 11–18. LCD Timing 0 Register (LcdTiming0)

Bit	Name	Description	Reset Value
31–24	HBP	Horizontal back porch	x
		Encoded value (from 1–256) used to specify number of pixel clock periods to add to the beginning of a line transmission before the first set of pixels is output to the display (program to value minus one).	
		The 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	x
		Encoded value (from 1–256) used to specify number of pixel clock periods to add to the end of a line transmission before line clock is asserted (program to value minus one).	
		The 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	x
		Encoded value (from 1–64) used to specify number of pixel clock periods to pulse the line clock at the end of each line (program to value minus one).	
		The 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	x
		Encoded value (from 1–1024) used to specify number of pixels contained within each line on the LCD display (program to value minus one).	

Note: X = Unknown

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 one).

Note:

The pixel clock pin LCD.PCLK, does not transition during these dummy pixel clock cycles in passive display mode (pixel clock transitions continuously in active display mode).

Figure 11–13 and Figure 11–14 show the use of LCD timing register 0 control fields for active and passive displays, respectively. Timing is shown for the middle of a frame, not at the beginning or end where VSYNC also occurs. See Section 11.8.3, *LCD Timing 1 Register*, for information on VSYNC timing. In Figure 11–14, the dashed lines on LCD.PCLK indicate that the signal is not actively toggling: LCD.PCLK is inactive at end-of-line mode. Virtual clocks are shown to demonstrate the behavior of the HFP, HSW, and HBP bit fields in the timing 0 register.

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 one).

Note:

The pixel clock pin LCD.PCLK, does not transition during these dummy pixel clock cycles in passive display mode (pixel clock transitions continuously in active display mode).

Figure 11–13. Active Mode End of Line Timing

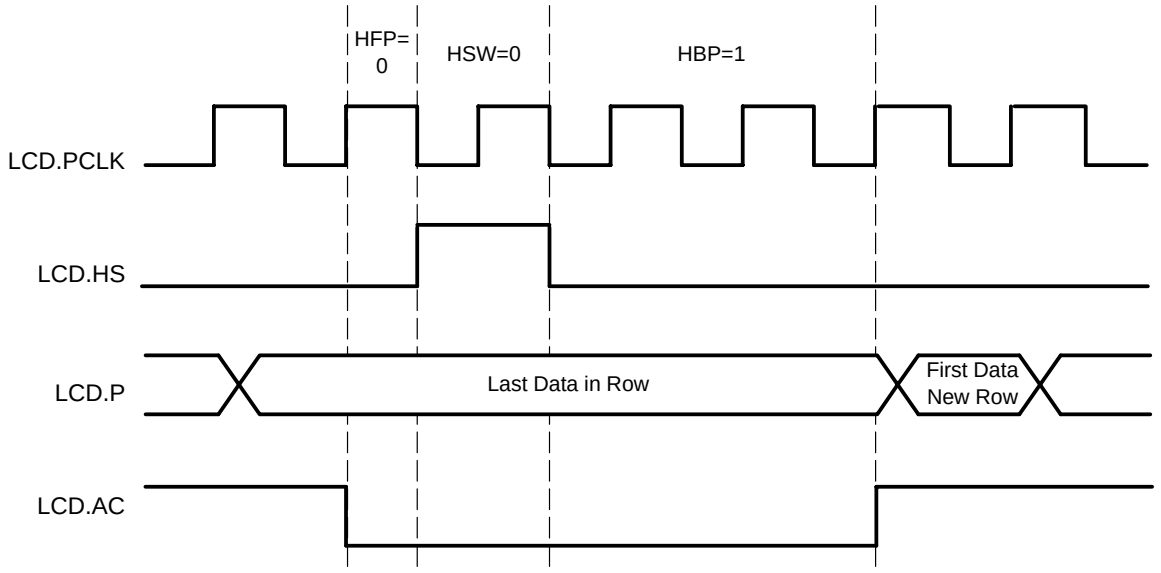
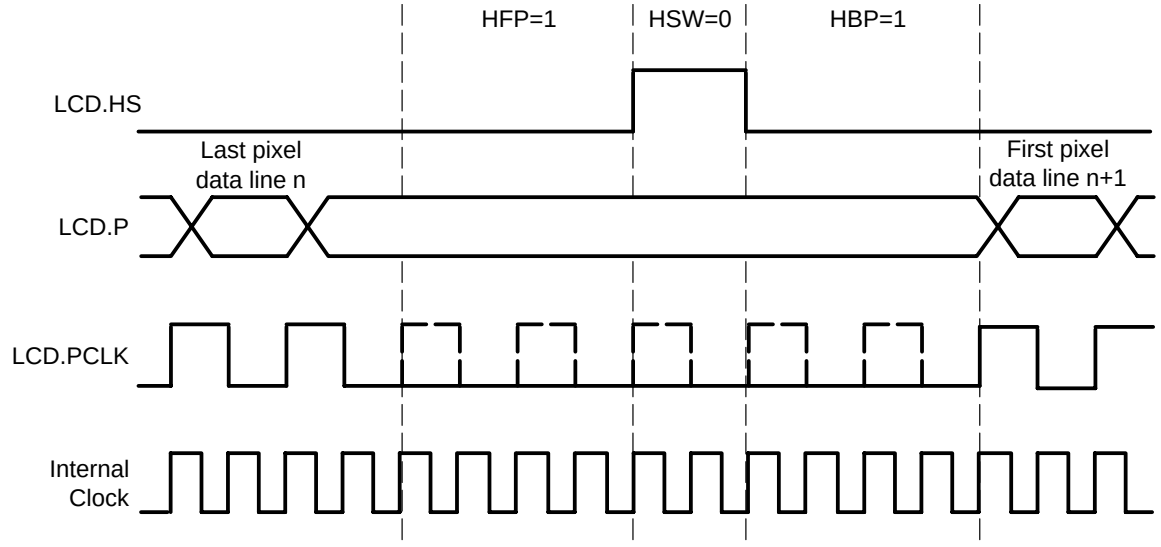


Figure 11–14. Passive Mode End of Line Timing



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. LCD.HS 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 one).

Note:

The pixel clock does not transition during the line clock pulse in passive display mode, but transitions in active display mode. Also, the polarity (active and inactive state) of the line clock is programmed using the invert HSYNC (IHS) bit in LCDTiming2.

Pixels-Per-Line (PPL)

The pixels-per-line (PPL) bit-field is used to specify the number of pixels in each line or row on the screen. PPL is a 10-bit value that represents 16–1024 pixels-per-line. PPL is used to count the correct number of pixel clocks that must occur before the line clock can be pulsed. (The bottom four bits of this register are not used and always read 1).

Note:

PPL must be programmed to the value required minus one (that is, 0x27F for a 640 pixels per line LCD panel).

11.8.3 LCD Timing 1 Register (LcdTiming1)

The LCD timing 1 register contains four bit fields that are used as modulus values for a collection of down counters, each of which performs a different function to control the timing of several of the LCD lines.

Table 11–19 shows the location of the bit fields located in LCD timing 1 register (LCDTiming1) and provides bit descriptions. The LCD controller must be disabled (LCDEN = 0) when changing the state of any field within this register. The reset state of all bit fields is unknown and must be initialized before enabling the LCD.

Table 11–19. LCD Timing 1 Register (LcdTiming1)

Bit	Name	Description	Reset Value
31–24	VBP	Vertical back porch	0
		Value (0−255) used to specify number of line clock periods to add to the beginning of a frame before the first set of pixels is output to the display. The line clock transitions during the insertion of the extra line clock periods.	
23–16	VFP	Vertical front porch	0
		Value (0−255) used to specify number of line clock periods to add to the end of each frame. The line clock transitions during the insertion of the extra line clock periods.	
15–10	VSW	Vertical synchronization pulse width	0
		In active mode (LCDTFT = 1), encoded value (1−64) used to specify number of line clock periods to pulse the LCD.VS pin at the end of each frame after the end of frame wait (VFP) period elapses. Frame clock used as VSYNC signal in active mode (program to value minus one).	
		In passive mode (LCDTFT = 0), encoded value (1−64) used to specify number of extra line clock periods to insert after the vertical front porch (VFP) period has elapsed. The width of LCD.VS is not effected by VSW in passive mode and that line clock transitions during the insertion of the extra line clock periods (program to value minus one).	
9–0	LPP	Lines per panel	0
		Encoded value (1−1024) used to specify number of lines per panel. It represents the total number of lines on the LCD (program to value minus one).	

Vertical Back Porch (VBP)

The 8-bit vertical back porch (VBP) field is used to specify the number of horizontal synchronizations (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 horizontal synchronizations 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 horizontal synchronization periods to insert before starting to output pixels in the next frame. VBP generates 0–255 extra line clock cycles.

Figure 11–15 and Figure 11–16 show the use of LCD timing register 1 control fields for active and passive displays, respectively.

Vertical Front Porch (VFP)

The 8-bit vertical front porch (VFP) field is used to specify the number of horizontal synchronizations (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 horizontal synchronization periods to wait. After the count has elapsed, the VSYNC (LCD.VS) signal is pulsed in active mode, or extra horizontal synchronizations are inserted as specified by the VSW bit field in passive mode. VFP generates 0–255 line clock cycles.

Note:

The line clock pin LCD.HS transitions during the generation of the VFP line clock periods.

Figure 11–15. Active Mode End of Frame Timing

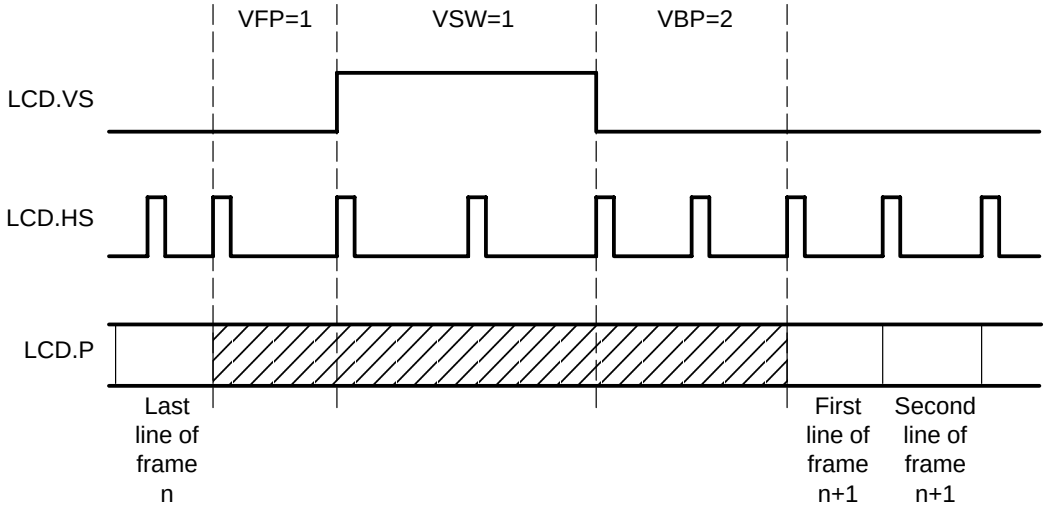
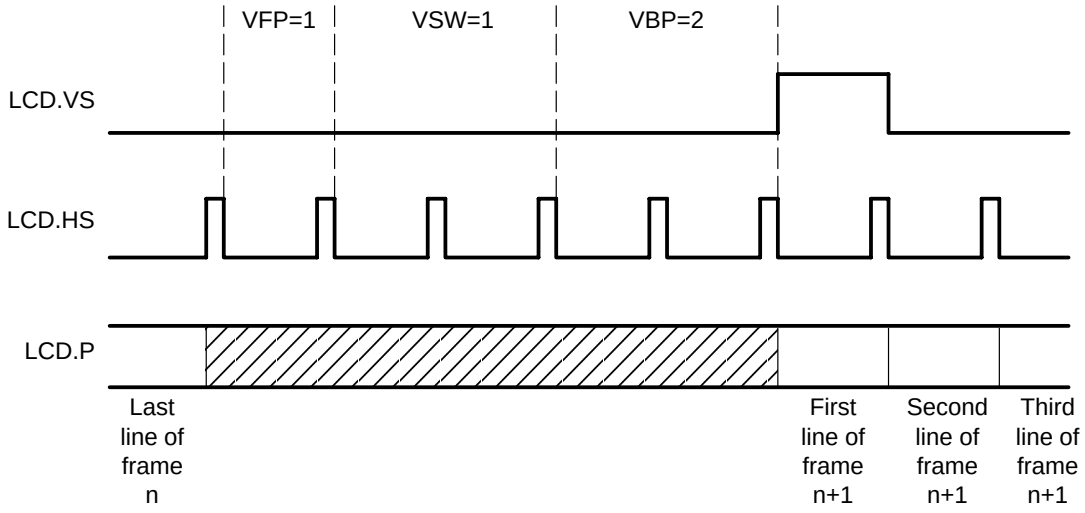


Figure 11–16. Passive Mode End of Frame Timing



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 to add extra dummy horizontal synchronization delays (i.e., dummy lines or rows) between the vertical front porch and vertical back porch in passive mode.

In active mode (LCDTFT = 1), LCD.VS is used to generate the vertical synchronization signal. It is asserted each time the last line or row of pixels for a frame is output to the display and a programmable number of line clock delays have elapsed. When LCD.VS 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, LCD.VS 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 one).

In passive mode (LCDTFT = 0), VSW does not affect the timing of the LCD.VS pin, but instead can be used to add extra horizontal synchronization delays (that is, dummy lines or rows) between the end and beginning of frame line clock delay counts. The total number of horizontal synchronization delays 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 horizontal synchronization delays between frames by first using the value in VFP, then VSW, then VBP. In passive mode, it is irrelevant if one or all three of the fields are used to insert delays; the user need only ensure that the sum of the values in the three fields is equal to the total number of line clock delays that are needed between frames.

Note:

The line clock transitions during the insertion of the dummy horizontal synchronization delay periods. VSW must be long enough to load the palette.

As mentioned, VSW does not affect generation of the frame clock (i.e., vertical synchronization) signal in passive mode. Passive LCD displays require that the frame clock is active on the rising-edge of the first line clock (i.e., horizontal synchronization) pulse of each frame, with adequate set-up and hold time. To meet this requirement, 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 and during the assertion of the first line clock for the frame and are then negated on the rising-edge of the first pixel clock of the *second* line of each frame.

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. LPP is a 10-bit value that represents 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 one (that is, 0xc7 for a 200 lines per panel).

11.8.4 LCD Timing 2 Register (LcdTiming2)

The LCD timing 2 register (LcdTiming2) contains seven different bit fields that are used to control various functions associated with the timing of the LCD controller (see Table 11–20).

The LCD controller must be disabled (LCDEN = 0) when changing the state of any field within this register. The reset state of all bit fields is unknown and must be initialized before enabling the LCD. Write functions to reserved bits are ignored and read functions return ones.

Table 11–20. LCD Timing 2 Register (LcdTiming2)

Bit	Name	Value	Description	Reset Value
31–26	-		Reserved	1
25	PHSVS On_Off		HSYNC/VSYNC pixel clock control on/off (on only when in TFT mode); off by default	0
		0	LCD.HS and LCD.VS are driven on the opposite edges of the pixel clock than the lcd_data.	
		1	LCD.HS and LCD.VS are driven according to bit 24.	
24	PHSVS RF		Program HSYNC/VSYNC rise and fall	0
		0	LCD.HS and LCD.VS are driven on the falling edge of the pixel clock (bit 25 is set to 1).	
		1	LCD.HS and LCD.VS are driven on the rising edge of the pixel clock (bit 25 is set to 1).	
23	IEO		Invert output enable	0
		0	LCD.AC pin is active high in active display mode.	
		1	LCD.AC pin is active low in active display mode.	
			Active display mode: data driven out to the LCD data lines on programmed pixel clock edge when ac-bias is active. IEO is ignored in passive display mode.	
22	IPC		Invert pixel clock	0
		0	Data is driven on the LCD data lines on the rising edge of LCD.PCLK.	
		1	Data is driven on the LCD data lines on the falling edge of LCD.PCLK.	

Table 11–20. LCD Timing 2 Register (LcdTiming2) (Continued)

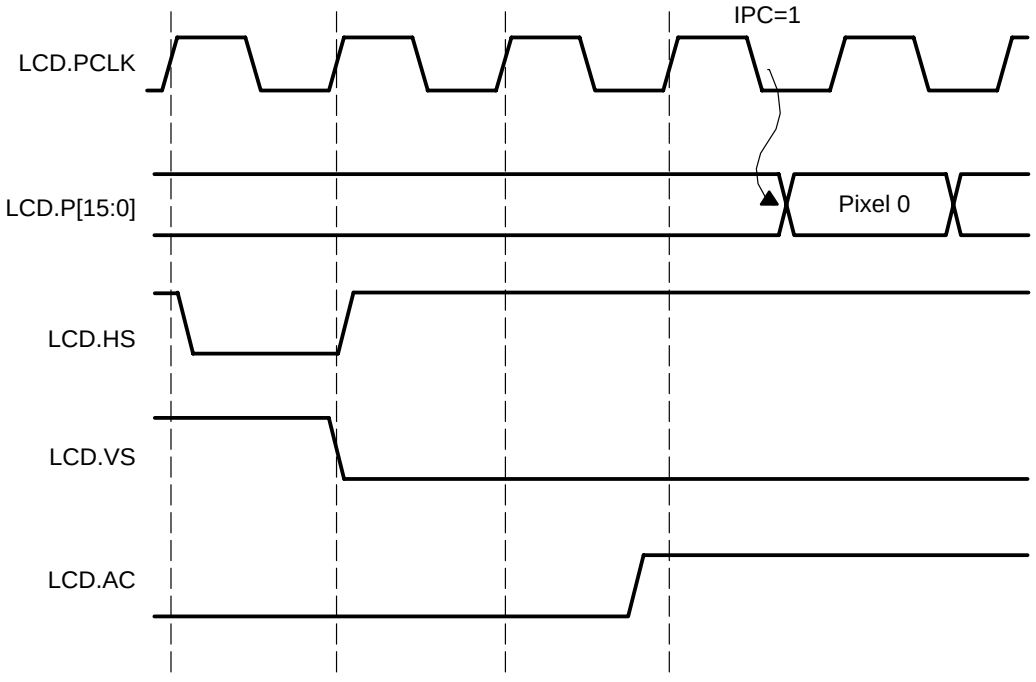
Bit	Name	Value	Description	Reset Value
21	IHS		Invert HSYNC	0
		0	LCD.HS pin is active high and inactive low.	
		1	LCD.HS pin is active low and inactive high.	
			Active and passive mode: horizontal synchronization pulse/line clock active between lines and after end of line wait period	
20	IVS		Invert VSYNC	0
		0	LCD.VS pin is active high and inactive low.	
		1	LCD.VS pin is active low and inactive high.	
			Active mode: vertical synchronization pulse active between frames and after end of frame wait period.	
			Passive mode: frame clock active during first line of each frame	
19–16	ACBI		ac-bias line transitions per interrupt	0
			Value (0-255) used to specify the number of ac-bias pin transitions to count before setting the line count status (LCS) bit, signaling an interrupt request. Counter is frozen when LCS is set and is restarted when LCS is cleared by software. This function is disabled when ACBI = 0x0000.	
15–8	ACB		ac bias pin frequency	0
			Value (0−255) used to specify 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 LCD.AC pin	
7–0	PCD		Pixel clock divider	0
			Value (2−255) used to specify pixel clock frequency based on CPU clock (LCD_CK) frequency. Pixel clock frequency can range from LCD_CK/2 to LCD_CK/255.	
			Pixel clock frequency = LCD_CK/2(PCD)	

HSYNC/VSYNC Rise or Fall Programmability

This bit determines whether the HSYNC/VSYNC signals are driven on the rising or falling edge of the pixel clock (PHSVS_ON_OFF must be turned on first). By default, the HSYNC/VSYNC signals are driven on the falling edge of the pixel clock, and the LCD 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 the rising edge of the pixel clock and pixel data is driven on the falling edge. By setting the PHSVS_RISE_FALL bit and enabling it (PHSVS_ON_OFF = 1), you can control on which edge the signals are driven.

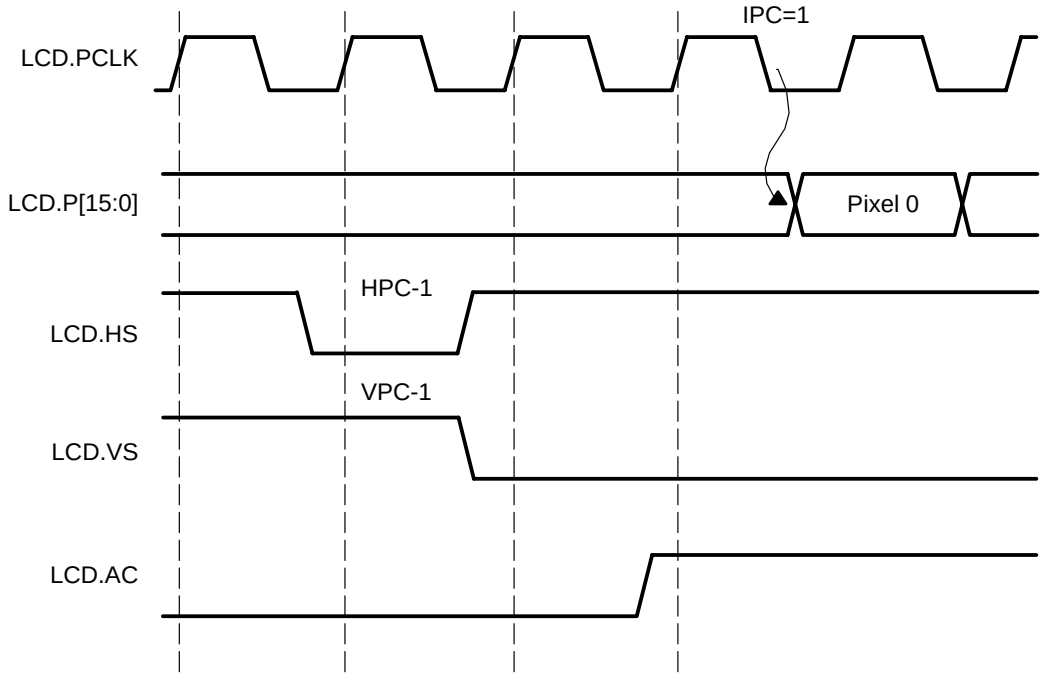
The waveforms in Figure 11–17 show PHSVS_ON_OFF = 0 and IPC = 1 in TFT mode.

Figure 11–17. Signal Timing When PHSVS_ON_OFF = 0



The waveforms in Figure 11–18 show PHSVS_ON_OFF = 1, PHSVS_RISE_FALL = 0, and IPC = 1.

Figure 11–18. Signal Timing When PHSVS_ON_OFF = 1



ac-Bias Line Transactions Per Interrupt (ACBI)

The 4-bit ac-bias line transitions per interrupt (ACBI) field is used to specify the number of LCD.AC 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. When ABC is set, the 4-bit down counter is reloaded with the value in ACBI and is disabled until ABC is cleared. When ABC is cleared by the CPU, the down counter is enabled and it 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 the interrupt function.

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 (LCD, AC). 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 LCD, AC is reversed, and the whole procedure starts again. The number of line clocks between ac-bias pin transition ranges from 0–255 (program to value required minus one). 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 on LCD.AC in active mode. This is because the pixel clock transitions continuously in active mode; 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.

Pixel Clock Divider (PCD)

The 8-bit pixel clock divider (PCD) field is used to select the frequency of the pixel clock (see Table 11–21). PCD can generate a range of pixel clock frequencies from LCD_CK/2 to LCD_CK/255, where LCD_CK is the LCD controller clock from the OMAP5910 clock management logic (see Chapter 15, *Clock Generation and System Reset Management*). 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 frame clock or HSYNC signal

All of these factors alter the time duration from one frame transmission to the next. Different display manufacturers require different frame refresh rates, depending on the physical characteristics of the display. The PCD is used to alter the pixel clock frequency in order to meet these requirements. The PCD is also used in parallel data input mode to select the frequency of pixel clock. Pixel clock is used to synchronously signal the off-chip 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:

Pixel Clock = LCD_CK/PCD

The pixel clock frequency can be programmed with the following limitations.

Table 11–21. Minimum Pixel Clock Divider (PCD)

Type of Display	Output (Number of Signals)	Minimum Pixel Clock Divider
Active	16 (1 pixel/clock)	2
Monochrome	4 (4 pixels/clock)	4
Passive color	8 (2 $^{2/3}$ pixels/clock)	3

11.8.5 LCD Status Register (LcdStatus)

The LCD controller status register (LCSR) contains bits that signal overrun and underrun errors for the input and output FIFOs and the ac-bias pin transition count, LCD disabled, DMA base update ready, and DMA transfer bus error conditions. Each of these hardware-detected events signals an interrupt request to the interrupt controller.

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). Writing 1 to a sticky status bit clears it; writing zero has no effect. Read-only flags are set and cleared by hardware; writes have no effect.

Table 11–22 describes the LCD status register (LcdStatus) bits.

See Table 11–23 for the location of the bit fields located in LCD subpanel register and provides bit descriptions.

Table 11–22. LCD Status Register (LcdStatus)

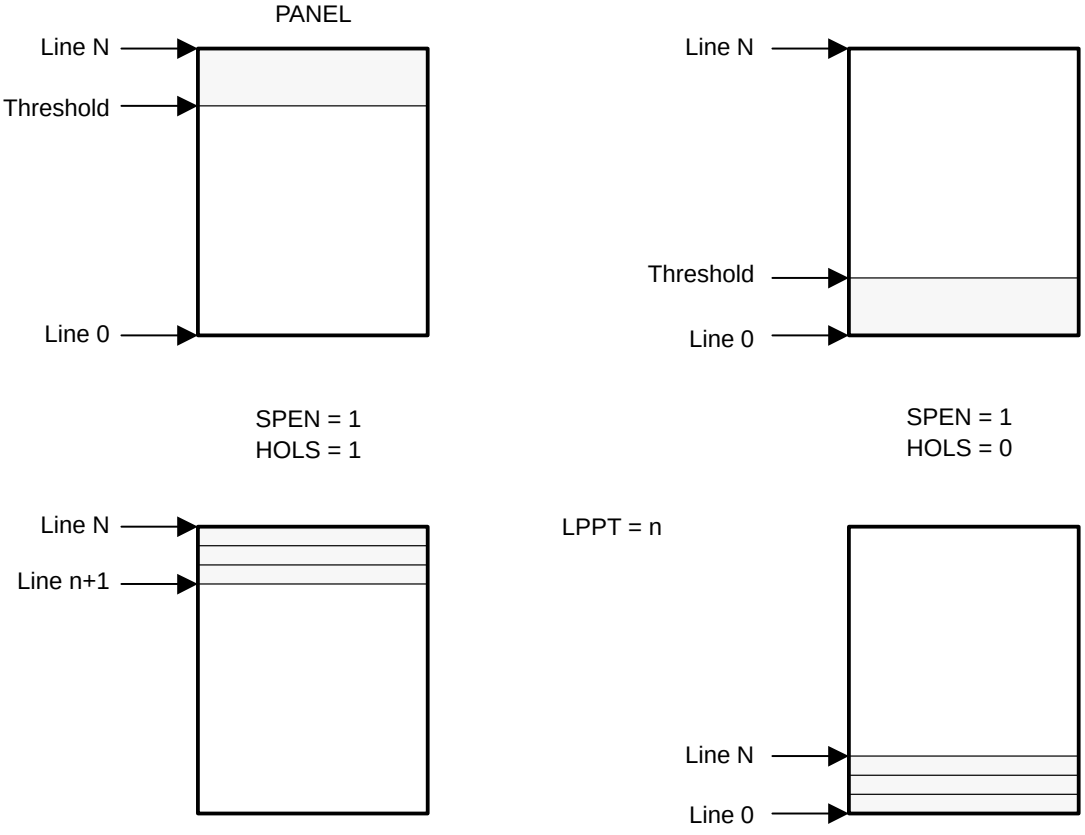
Bit	Name	Value	Description	Reset Value
31–7	–		Reserved	0
6	PL		Loaded palette (read-only)	0
		0	The palette is not loaded.	
		1	The palette is loaded.	
5	FUF		FIFO underflow status (read-only). Cleared by setting LCDEN to 0, which also resets the input FIFO in the DMA controller.	0
		0	FIFO has not underrun.	
		1	LCD dither logic 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	–		Reserved	0
3	ABC		ac-bias count status (read/clear only)	0
		0	ac-bias transition counter has not decremented to zero.	
		1	ac-bias transition counter has decremented to zero, indicating that the LCD.AC line has transitioned the number of times specified by the ACBI control bit-field. Counter is reloaded with value in ACBI but is disabled until the user clears ABC.	
2	Sync		Synchronization lost (read-only). Cleared by setting LCDEN to 0, which also resets the input FIFO in the DMA controller.	0
		0	Normal	
		1	Frame synchronization lost has occurred.	
1	–		Reserved	0
0	Done		Frame done (read-only). Cleared by writing base address and enabling the LCD for single-panel mode. When the LCD is disabled by clearing the LCD enable bit (LCDEN = 0) in LCDControl, 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.	0
		0	LCD is enabled.	
		1	LCD disabled and the active frame has just completed.	

Table 11–23. LCD Subpanel Register (LcdSubpanel)

Bit	Name	Value	Description	Reset Value
31	SPEN		Subpanel enable	0
		0	Function disabled	
		1	Subpanel function mode enabled	
30			Reserved	0
29	HOLS		High or low signal	0
			The field indicates the position of subpanel compared to the LPPT value.	
28–26			Reserved	0
25–16	LPPT		Line per panel threshold	0
			This field defines the number of lines to be refreshed (1–1024).	
			(Program to value minus 1.)	
15–0	DPD		Default pixel data	0
			DPD defines the default value of the pixel data sent to the panel for the lines until LPPT is reached or after passing the LPPT.	

The ability to display only the first or last n lines of the panel and send a fixed contents for the other lines is supported with the subpanel display register, shown in Figure 11–19. For the other lines, there is no access to the frame buffer because the value stored in default pixel data is used.

Figure 11–19. LCD Subpanel Display Register (LcdSubpanel)



11.9 Interface to LCD Panel Signal Reset Values

The LCD panel signal outputs can accept two distinct reset values (see Table 11–24):

- After a hardware reset by setting the LCD_RESET_I signal to low
- By disabling the LCD (setting LCDEN bit to low)

The default value depends solely upon the signal polarity control, as defined in the LCD timing 2 register, except for LCD.P[15:0] when driven low and LCD.AC, which does not change status when in STN mode.

Table 11–24. LCD Panel Signals Reset Values

	LCD.P[0][15:0]	LCD.PCLK	LCD.HS	LCD.VS	LCD.AC
Reset (LCD_RESET_I = 0)	0	0	0	0	0
Disable (LCDEN = 0)	0	0 (IPC = 0) 1 (IPC = 1)	0 (HIS = 0) 1 (HIS = 1)	0 (IVS = 0) 1 (IVS = 1)	TFT: 0 (IEO = 0) 1 (IEO = 1) STN: No change

Chapter 12

UART Devices

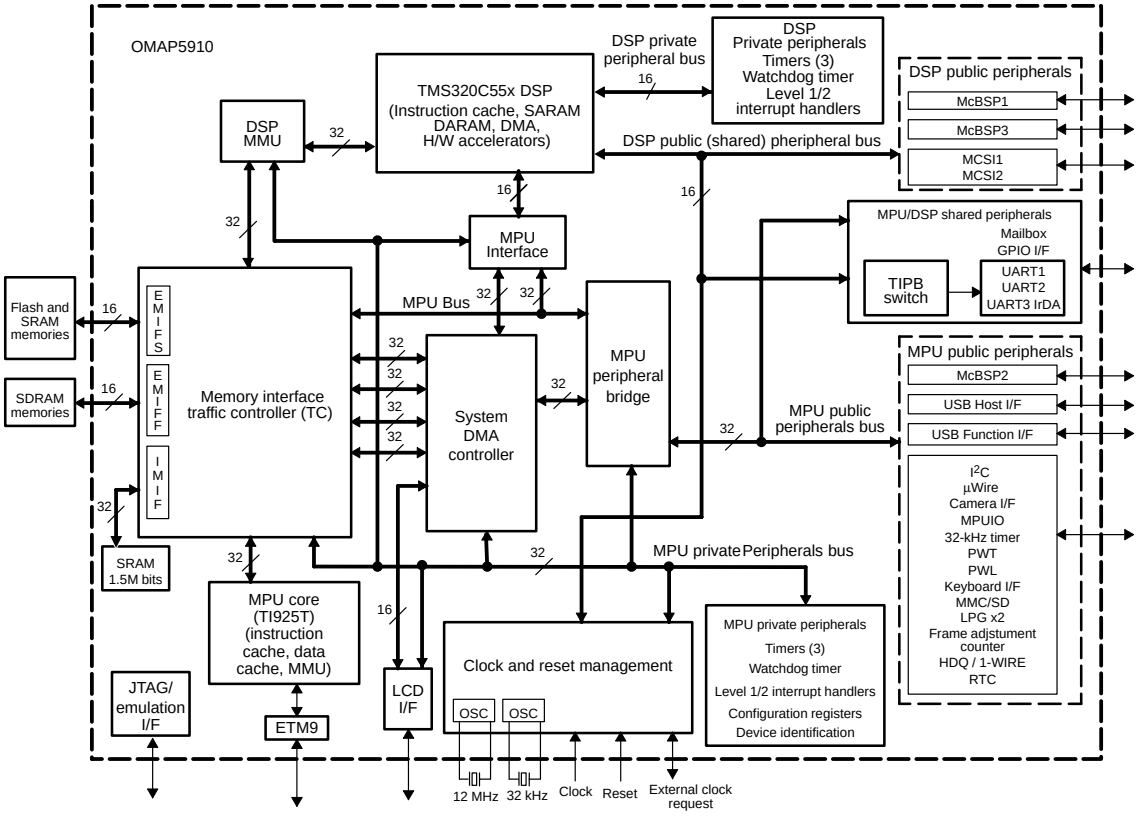
This chapter describes the three universal asynchronous receiver/transmitter (UART) devices in the OMAP5910 multimedia processor.

Topic **Page**

12.1 UART Introduction

Either the MPU (default) or the DSP controls the three UARTs in the OMAP5910 processor via three TIPB switches (one for each UART). Figure 12–1 shows the OMAP5910 device with the UART modem module highlighted. UART1 and 2 are UART modems with autobaud capability. UART3 is a modem with IrDA.

Figure 12–1. UART Modem Module



12.1.1 Main UART Features (UART1/2/3)

The main features are as follows:

- ❏ Selectable UART/autobaud modes (UART1 and 2 only)
- ❏ Dual 64-entry FIFOs for received and transmitted data payload
- ❏ 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 mode
- ❏ Frequency prescaler values from 0 to 65535 to generate the appropriate baud rates
- ❏ An interrupt request to the system if there are multiple DMA requests

12.1.1.1 UART/Modem Functions (UART1/2/3)

- ❏ Baud rate from 300 bits/s up to 1.5M bits/s
- ❏ Autobaud between 1200 bits/s and 115.2K bits/s
- ❏ Software/hardware flow control
  - ■ Programmable XON/XOFF characters
  - ■ Programmable AUTO_RTS and AUTO_CTS
- ❏ Programmable serial interface characteristics
  - ■ 5-, 6-, 7-, or 8-bit characters
  - ■ Even-, odd-, or no-parity 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 ($\overline{\text{CTS}}$, $\overline{\text{RTS}}$, $\overline{\text{DSR}}$, and $\overline{\text{DTR}}$)

12.1.1.2 IrDA Functions (UART3 Only)

- Slow infrared (SIR) operations
- Framing error, cyclic redundancy check (CRC) error, abort pattern (SIR) detection
- 8-entry status FIFO (with selectable trigger levels) available to monitor frame length and frame errors

Table 12–1 describes the I/O module at the module level.

12.1.1.3 UART Signals

The signals available on the UART modules are illustrated in Figure 12–2. These signals are described in Table 12–1.

Figure 12–2. UART Signals

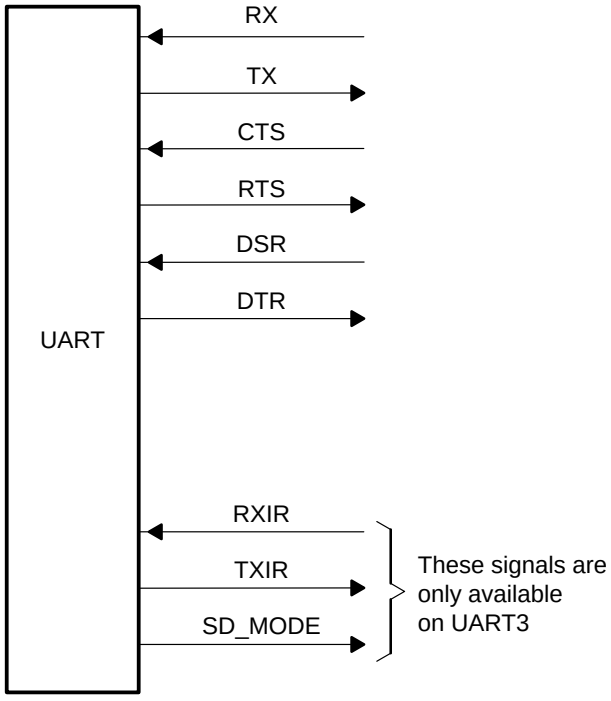


Table 12–1. I/O Description

Signal	I/O	Description	Reset Value
UART/MODEM Signals			
RX	I	Serial data input	–
TX	O	Serial data output	1
$\overline{\text{CTS}}$	I	Clear to send. Active-low modem status signal. Reading bit 4 of the modem status register checks the condition of $\overline{\text{CTS}}$. Reading bit 0 of that register checks a change of state of $\overline{\text{CTS}}$ since the last read of the modem status register. $\overline{\text{CTS}}$ is used in automatic CTS mode to control the transmitter.	–
$\overline{\text{RTS}}$	O	Request to send. When active (low), the module is ready to receive data. Setting modem control register bit 1 activates $\overline{\text{RTS}}$. It becomes inactive as a result of a module reset, loop back mode or by clearing the MCR1. In automatic RTS mode, it becomes inactive as a result of the receiver threshold logic.	1
$\overline{\text{DSR}}$	I	Data set ready Active-low modem status signal. Reading bit 5 of the modem status register checks the condition of $\overline{\text{DSR}}$. Reading bit 1 of that register checks a change of state of $\overline{\text{DSR}}$ since the last read of the modem status register.	–
$\overline{\text{DTR}}$	O	Data transmit ready Active-low modem control signal. Reading bit 0 of the modem control register checks the condition of $\overline{\text{DTR}}$.	1
IrDA Signals (UART3 Only)			
RXIR	I	Serial data input	–
TXIR	O	Serial data output	0
SD_MODE	O	Signal used to configure transceivers	1

12.2 UART Environments

Each UART is controllable through a TIPB switch, either by the MPU (default) or the DSP.

12.2.1 UART1 Environment

UART1 is a UART modem with autobaud capability. Table 12–2 lists the UART1 modem signals accessible at the OMAP5910 level.

Table 12–2. Available UART1 Signals

Generic UART Signal Name	Description	UART1 Signal Name
RX	Serial data input	UART1.RX
TX	Serial data output	UART1.TX
CTS	Clear to send input	UART1.CTS
RTS	Request to send input	UART1.RTS
DTR	Data transmit ready output	UART1.DTR
DSR	Data set ready input	UART1.DSR

The functional clock is either a 12-MHz or a 48-MHz clock. You can select the clock with the CONF_MOD_UART1_CLK_MODE_R bit (29) of the MOD_CONF_CTRL_0 register (see Section 6.8, *OMAP5910 Configuration Registers*) as follows:

- CONF_MOD_UART1_CLK_MODE_R = 0: 12 MHz (default)
- CONF_MOD_UART1_CLK_MODE_R = 1: 48 MHz

NDMA_REQ [1:0] are connected to the DMA request [13:12] of both the MPU system DMA controller and the DSP DMA controller.

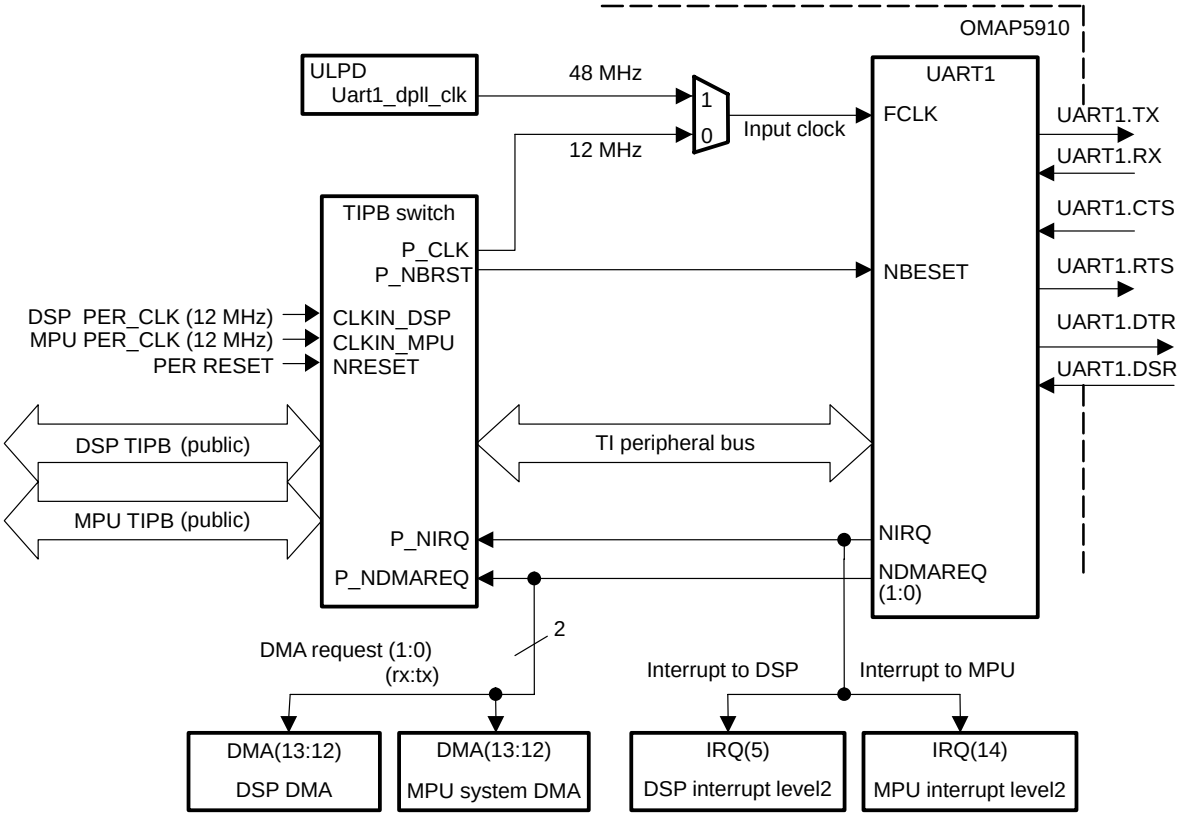
NDMA_REQ[1] is a RX request, and NDMA_REQ[0] is a TX request.

NIRQ from UART1 is connected to:

- The interrupt line IRQ[14] of the MPU level 2 interrupt handler
- The interrupt line IRQ[5] of the DSP level 2 interrupt handler

Figure 12–3 shows the UART1 environment.

Figure 12–3. UART1 Environment



12.2.2 UART2 Environment

UART2 is an UART modem with autobaud capability.

Table 12–3 lists the UART2 modem signals accessible at the OMAP5910 level.

Table 12–3. Available UART2 Signals

Generic UART Signal Name	Description	UART1 Signal Name
RX	Serial data input	RX2
TX	Serial data output	TX2
CTS	Clear to send input	CTS2
RTS	Request to send input	RTS2
FSR	Receive frame (input only)	Not available (internal feedback from FSX)
BDCLK	16x baud clock input	BDCLK2

The functional clock is either a 32-kHz/12-MHz or a 48-MHz clock. You can select the clock with the CONF_MOD_UART2_CLK_MODE_R bit (30) of the MOD_CONF_CTRL_0 register (see Section 6.8, *OMAP5910 Configuration Registers*) as follows:

- CONF_MOD_UART1_CLK_MODE_R = 0: 32 kHz/12 MHz (default)
- CONF_MOD_UART1_CLK_MODE_R = 1: 48 MHz

The frequency of the 32-kHz/12-MHz clock depends on the OMAP5910 system state:

- 32 kHz in deep sleep modes
- 12 MHz in big sleep and awake mode

Note that the UPLD clock control register (CLOCK_CTRL_REG) bit 0 MODEM_32K_EN must be controlled as follows:

- MODEM_32K_EN = 0: Disables 32-kHz on UART clock in deep sleep
- MODEM_32K_EN = 1: Enables 32-kHz on UART clock when in deep sleep

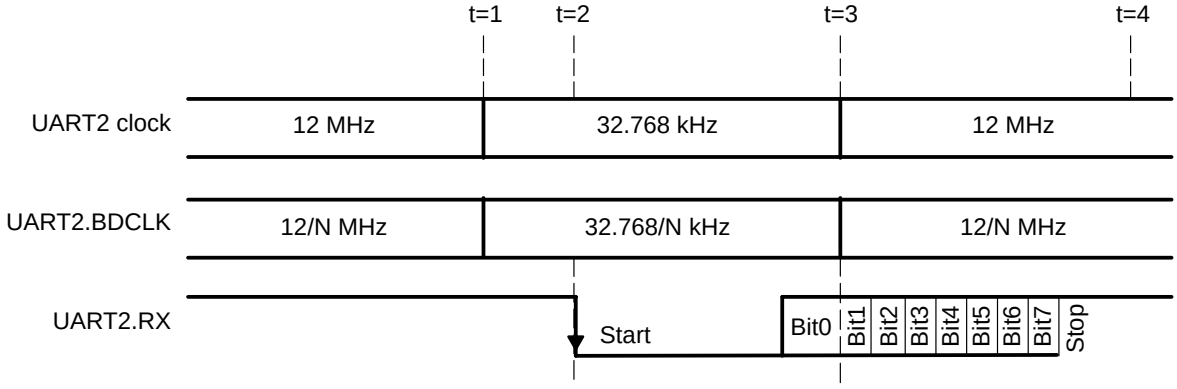
The reset condition is 0.

When the CONF_MOD_UART1_CLK_MODE_R = 0 and MODEM_32K_ EN = 1, the UART2 operates at 32kHz input clock while the device is in deep sleep.

UART2.BDCLK (UART2 baud clock) automatically switches to a lower frequency in deep sleep based on the 12MHz to 32kHz clock switch in the ULPD. Putting the baud clock outside of the device enables an external UART to use this baud clock to remain synchronized even while the OMAP5910 device is in deep sleep. If the external UART sends a byte when the OMAP5910 device uis in deep sleep, it isreceived by UART2 correctly without loss of data, although at a slow rate. The activity detection circuit monitors UART2.RX activity using the UART2.BDCLK clock and requests the ULPD to wakeup by the periph_clk_nreq signal. The activity detection logic uses a two out of three voting logic. The following sampled combinations of UART2.RX will produce a periph_clk_nreq: 001, 010, and 100.

Figure 12–4 shows the sequence of the wakeup by UART2.RX.

Figure 12–4. UART2.RX Wakeup Sequence



- T = #1: The MPU goes to standby and enters “Deep Sleep” state. The 12MHz osc is turned off. The Baud clock automatically switches to 32,768 / N. (N = Uart Div ratio)
- T = #2: A falling edge on UART2.RX is sensed and causes wake up of the 12MHz OSC, the wake up time depends primarily on the analog wait timer (12MHz osc start up delay).
- T = #3: The ULPD transitions to “Awake” mode after 12MHz is stable. The UART2 Baud clock will switch back to 12/N MHz. Note the MPU is still sleeping now waiting for interrupt to wakeup.
- T = #4: There are two ways the MPU wakes up:
  - RHR interrupt
  - RX Timeout interrupt (shown in Figure 12–4): When UART2.RX has been high for a time equivalent to (4 * Programmed word length + 12bits) / baud rate, the RX Timeout interrupt occurs and wakes up the MPU.

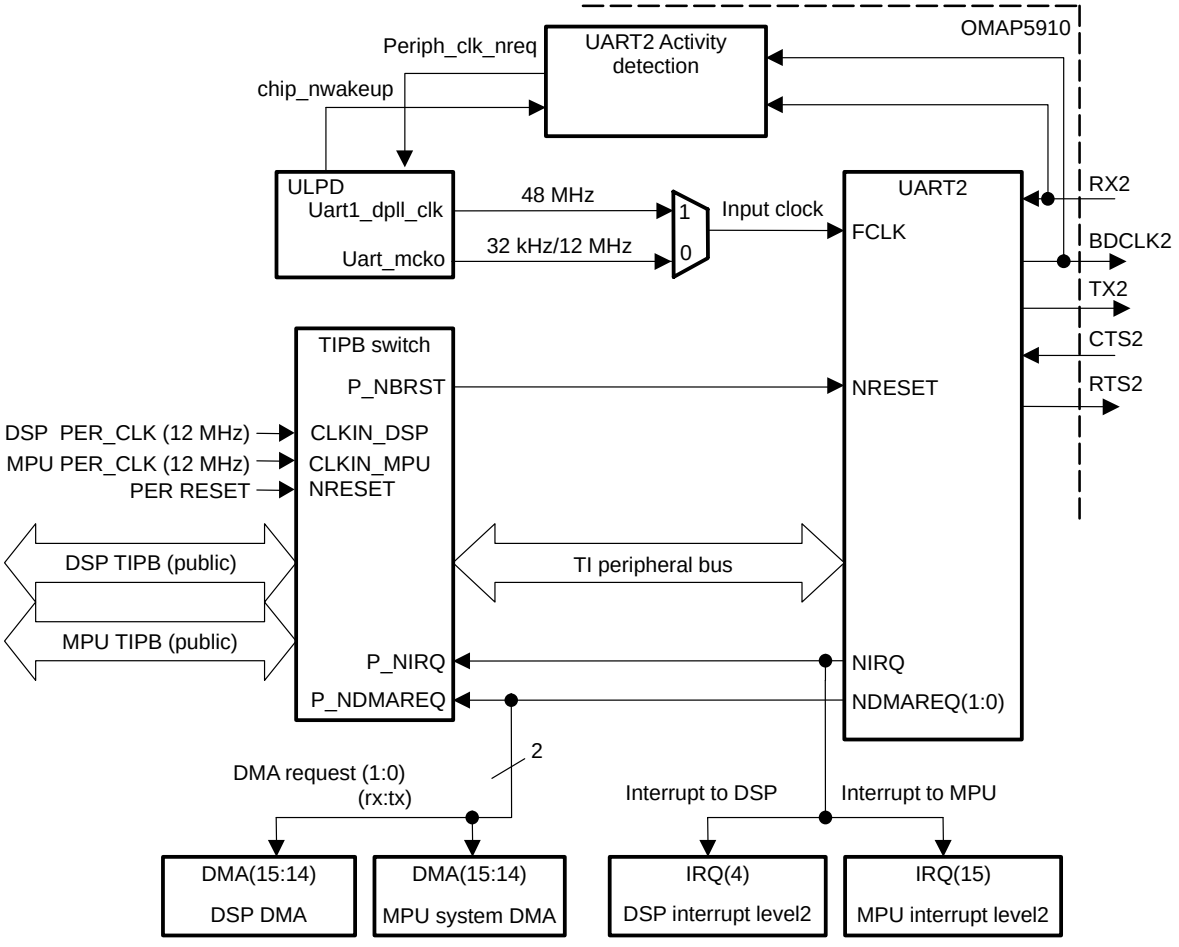
There is an alternate wakeup from deep sleep: By programming bit 4 of the UART2 SCR register, it is possible to create an interrupt by a low going edge on RX or CTS. The interrupt would be generated by RX at time #2 in Figure 12–4.

NDMA_REQ [1:0] are connected to the DMA request [15:14] of both MPU system DMA controller and the DSP DMA controller. NDMA_REQ[1] is a RX request, and the NDMA_REQ[0] is a TX request. NIRQ from UART2 is connected to the following:

- Interrupt line IRQ[15] of the MPU level 2 interrupt handler
- Interrupt line IRQ[4] of the DSP level 2 interrupt handler

Figure 12–5 shows the UART2 environment.

Figure 12–5. UART2 Environment



12.2.3 UART3 Environment

The UART3 is a UART modem with IrDA capability.

The IrDA mode (SIR) is selectable by setting the MODE_SELECT bits (2:0) of the UART3 MDRI register to 001. Set the IRDA_SELECT signal to 1.

You can use the IRDA_SELECT signal to control the multiplexing on the UART3 I/Os between the UART3 modem signals and the UART3 IrDA signals.

Table 12–4 lists the UART3 IrDA signals accessible at the OMAP5910 level when IRDA_SELECT = 1.

Table 12–4. Available UART3 Signals in IrDA = 1 Mode

Generic UART Signal Name	Description	UART1 Signal Name
TXIR	IrDA serial data input	TX3
RXIR	IrDA serial data output	UART3.RX
RX	Serial data input	High
SD_MODE	Signal used to configure transceivers	RTS3

Table 12–5 lists the UART3 IrDA signals accessible at the OMAP5910 level when IRDA_SELECT = 0.

Table 12–5. Available UART3 Signals in IrDA = 0 Mode

Generic UART Signal Name	Description	UART3 Signal Name
TX	Serial data output	TX3
RX	Serial data input	UART3.RX
RXIR	IrDA serial data input	High
RTS	Request to send output	RTS3
CTS	Clear to send input	CTS3
DTR	Data transmit ready output	DTR3
DSR	Data set ready input	DSR3

The functional clock is either a 12-MHz or a 48-MHz clock. You can select the clock with the CONF_MOD_UART3_CLK_MODE_R bit (30) of the MOD_CONF_CTRL_0 register (see Section 6.8, *OMAP5910 Configuration Registers*) as follows:

- CONF_MOD_UART3_CLK_MODE_R = 0: 12 MHz (default)
- CONF_MOD_UART3_CLK_MODE_R = 1: 48 MHz

The NDMA_REQ [1:0] are connected to DMA request [19:18] of both the MPU system DMA controller and the DSP DMA controller.

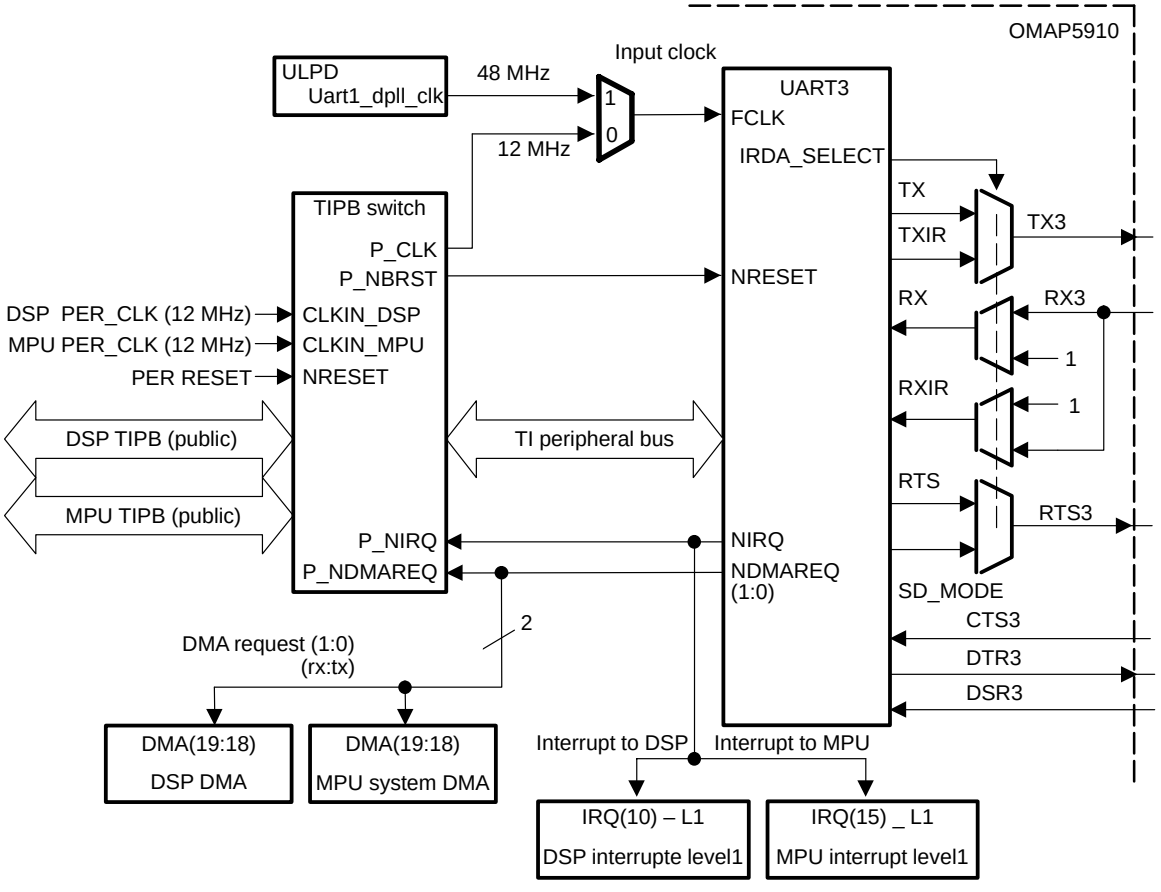
NDMA_REQ[1] is a RX request and the NDMA_REQ[0] is a TX request.

NIRQ from UART2 is connected to:

- The interrupt line IRQ[15] of the MPU level 1 interrupt handler
- The interrupt line IRQ[10] of the DSP level 1 interrupt handler

Figure 12–6 shows the UART3 environment.

Figure 12–6. UART3 Environment



12.2.4 TIPB Switch

By default, the three UARTs are controllable from the MPU public TIPB.

The three TIPB switch modules allow you to change the default configuration individually and thus to control the UARTs from the DSP public TIPB.

This change can only be done during the boot time. Dynamic switches are not supported.

This switch is software programmable, so each TIPB switch has two sets of registers:

- The MPU-accessible registers are listed in Table 12–6. Table 12–7 and Table 12–8 describe the register bits.
- The DSP-accessible registers are listed in Table 12–9. Table 12–10 and Table 12–11 describe the register bits.

Table 12–6. MPU Registers

UART	Register	Description	R/W	Bits	Address
UART1	RHSW_ARM_CNF	TIPB switch configuration	R/W	16	FFFB:C800
UART1	RHSW_ARM_STA	TIPB switch status	R	16	FFFB:C804
UART2	RHSW_ARM_CNF	TIPB switch configuration	R/W	16	FFFB:C840
UART2	RHSW_ARM_STA	TIPB switch status	R	16	FFFB:C844
UART3	RHSW_ARM_CNF	TIPB switch configuration	R/W	16	FFFB:C880
UART3	RHSW_ARM_STA	TIPB switch status	R	16	FFFB:C884

Table 12–7. TIPB Switch Configuration MPU Register (RHSW_ARM_CNF)

Bit	Name	Value	Function	R/W	Reset Value
15–2	Reserved		–	–	–
1	DSP_PERIPH_LOCK	0	No lock	R	0
		1	DSP bus is allocated.		
0	ARM_PERIPH_LOCK	0	No lock	R/W	1
		1	MPU bus is allocated.		

Table 12–8. TIPB Switch Status MPU Register (RHSW_ARM_STA)

Bit	Name	Value	Function	R/W	Reset Value
15–4	Reserved		–	–	–
3	RHSW_BOTH_LCK_ERR	0	Normal operation	R	0
		1	Lock error		
2	RHSW_ITPEND_ERR	0	Normal operation	R	0
		1	DMA request error		
1	RHSW_DMAREQ_ERR	0	Normal operation	R	0
		1	IT pending error		
0	RHSW_ERR_NIRQ	0	Clears IRQ line and all others status bits of register	R/W	1
		1	Normal operation		

Table 12–9. DSP Registers

UART	Register	Description	R/W	Bits	Address
UART1	RHSW_DSP_CNF	TIPB switch control	R/W	16	001:C800
UART1	RHSW_DSP_STA	TIPB switch status	R	16	001:C802
UART2	RHSW_DSP_CNF	TIPB switch control	R/W	16	001:C820
UART2	RHSW_DSP_STA	TIPB switch status	R	16	001:C822
UART3	RHSW_DSP_CNF	TIPB switch control	R/W	16	001:C840
UART3	RHSW_DSP_STA	TIPB switch status	R	16	001:C842

Table 12–10. TIPB Switch Configuration DSP Register (RHSW_DSP_CNF)

Bit	Name	Value	Function	R/W	Reset Value
15–2	Reserved		–	–	–
1	DSP_PERIPH_LOCK	0	No lock	R/W	0
		1	DSP bus is allocated.		
0	ARM_PERIPH_LOCK	0	No lock	R	1
		1	MPU bus is allocated.		

Table 12–11. TIPB Switch Status DSP Register (RHSW_DSP_STA)

Bit	Name	Value	Function	R/W	Reset Value
15–4	Reserved		–	–	–
3	RHSW_BOTH_LCK_ERR	0	Normal operation	R	0
		1	Lock error		
2	RHSW_ITPEND_ERR	0	Normal operation	R	0
		1	DMA request error		
1	RHSW_DMAREQ_ERR	0	Normal operation	R	0
		1	IT pending error		
0	RHSW_ERR_NIRQ	0	Clears IRQ line and all others status bits of register	R/W	1
		1	Normal operation		

12.2.5 Switching Procedures

The following procedures enable you to switch from MPU to DSP.

For switching UART1 to DSP:

1) MPU: Write 0 into the UART1 TIPB switch configuration MPU register (RHSW_ARM_CNF) to unlock UART1.

2) DSP: Write 2 into the UART1 TIPB switch status DSP register (RHSW_DSP_CNF) to lock UART1.

For switching UART2 to DSP:

1) MPU: Write 0 into the UART2 TIPB switch configuration MPU register (RHSW_ARM_CNF) to unlock UART2.

2) DSP: Write 2 into UART2 TIPB switch configuration DSP register (RHSW_DSP_CNF) to lock UART2.

For switching UART3 to DSP:

1) MPU: Write 0 into the UART3 TIPB switch configuration MPU register (RHSW_ARM_CNF) to unlock UART3.

2) DSP: Write 2 into UART3 TIPB switch configuration DSP register (RHSW_DSP_CNF) to lock UART3.

Note: PERIF_LOCK Bits

If either the DSP_PERIF_LOCK (in the RHSW_DSP_CNF register) or the ARM_PERIF_LOCK bit (in the RHSW_ARM_CNF register) is already set to 1, then a write to the other PERIF_LOCK bit has no effect on the TIPB switches, even though such a write may be performed. Before attempting to write 1 to a PERIF_LOCK bit, the DSP and MPU software must always read the corresponding read-only PERIF_LOCK bits to confirm that the other processor PERIF_LOCK bit is not already set.

12.3 UART/Autobaud Control and Status Registers

The programming combinations for register selection are shown in Table 12–12.

12.3.1 UART/Autobaud Modem Register Mapping

UART1 and UART2 are accessible as follows:

- MPU (32-bit-byte aligned address) from the following base addresses:
  - UART1: 0xFFFB 0000
  - UART2: 0xFFFB 0800
- DSP (16-bit-aligned word address) from the following base addresses:
  - UART1: 0x008000
  - UART2: 0x008400

Table 12–12. UART Modem Register Program

MPU Byte Offset	DSP Byte Offset	Registers					
		LCR[7] = 0		LCR[7] = 1 LCR[7:0] ≠ 0xBF		LCR[7:0] = 0xBF	
		READ	WRITE	READ	WRITE	READ	WRITE
0x00	0x00	RHR	THR	DLL	DLL	DLL	DLL
0x04	0x02	IER†	IER†	DLH	DLH	DLH	DLH
0x08	0x04	IIR	FCR†	IIR	FCR†	EFR	EFR
0x0C	0x06	LCR	LCR	LCR	LCR	LCR	LCR
0x10	0x08	MCR†	MCR†	MCR†	MCR†	XON1	XON1
0x14	0x0A	LSR	-	LSR	-	XON2	XON2
0x18	0x0C	MSR/TCR‡	TCR‡	MSR/TCR‡	TCR‡	XOFF1/TCR‡	XOFF1/TCR‡
0x1C	0x0E	SPR/TLR‡	SPR/TLR‡	SPR/TLR‡	SPR/TLR‡	XOFF2/TLR‡	XOFF2/TLR‡
0x20	0x10	MDR1	MDR1	MDR1	MDR1	MDR1	MDR1
0x24	0x12	-	-	-	-	-	-
0x28	0x14	-	-	-	-	-	-
0x2C	0x16	-	-	-	-	-	-

† MCR[7:5], FCR[5:4], and IER[7:4] 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 12–12. UART Modem Register Program (Continued)

MPU Byte Off-set	DSP Byte Off-set	Registers					
		LCR[7] = 0		LCR[7] = 1 LCR[7:0] ≠ 0xBF		LCR[7:0] = 0xBF	
		READ	WRITE	READ	WRITE	READ	WRITE
0x30	0x18	-	-	-	-	-	-
0x34	0x1A	-	-	-	-	-	-
0x38	0x1C	-	-	UASR	-	UASR	-
0x3C	0x1E	-	-	-	-	-	-
0x40	0x20	SCR	SCR	SCR	SCR	SCR	SCR
0x44	0x22	SSR	-	SSR	-	SSR	-
0x48	0x24	-	-	-	-	-	-
0x4C	0x26	-	OSC_12M_ SEL	-	-	-	-
0x50	0x28	MVR	-	MVR	-	MVR	-

† MCR[7:5], FCR[5:4], and IER[7:4] 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 12–13 lists the UART/autobaud registers. Table 12–14 through Table 12–41 describe specific register bits.

Table 12–13. UART/Autobaud Registers

Register	Description	Size	Access
RHR	Receive holding	8-bit	R
THR	Transmit holding	8-bit	W
FCR	FIFO control	8-bit	W
SCR	Supplementary control	8-bit	R/W
LCR	Line control	8-bit	R/W
LSR	UART mode (LSR)	8-bit	R
SSR	Supplementary status	8-bit	R
MCR	Modem control	8-bit	R/W

Table 12–13. UART/Autobaud Registers (Continued)

Register	Description	Size	Access
MSR	Modem status	8-bit	R
IER	Interrupt enable (IER)	8-bit	R/W
IIR	Interrupt identification (IIR)	8-bit	R
EFR	Enhanced feature	8-bit	R/W
XON1	XON1	8-bit	R/W
XON2	XON2	8-bit	R/W
XOFF1	XOFF1	8-bit	R/W
XOFF2	XOFF2	8-bit	R/W
SPR	Scratchpad	8-bit	R/W
DLL	Divisor latch low	8-bit	R/W
DLH	Divisor latch high	8-bit	R/W
TCR	Transmission control	8-bit	R/W
TLR	Trigger level	8-bit	R/W
MDR1	Mode definition 1	8-bit	R/W
UASR	UART autobauding status	8-bit	R
OSC_12M_SEL	12-MHz oscillator select	8-bit	R
MVR	Module version	8-bit	R

The receiver section consists of the receiver holding register (RHR) and the receiver shift register. The RHR is actually a 64-byte FIFO. The receiver shift register receives serial data from RX input. The data is converted to parallel data and moved to the RHR. If the FIFO is disabled, location zero of the FIFO is used to store the single data character.

Note:

If overflow occurs, data in the RHR is not overwritten.

Table 12–14. Receive Holding Register (RHR)

Bit	Name	Function	R/W	Reset Value
7–0	RHR	Receive holding register	R	Undefined

The transmitter section consists of the transmit holding register (THR) and the transmit shift register. The THR is actually a 64-byte FIFO. The host (MPU or DSP) 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 12–15. Transmit Holding Register (THR)

Bit	Name	Function	R/W	Reset Value
7–0	THR	Transmit holding register	W	Undefined

Table 12–16. FIFO Control Register (FCR)

Bit	Name	Value	Function	R/W	Reset Value
7–6	RX_FIFO_TRIG		Sets the trigger level for the RX FIFO:	W	00
			If SCR7 = 0 and TLR7:4 = 0000:		
		00	8 characters		
		01	16 characters		
		10	56 characters		
		11	60 characters		
			If SCR7 = 0 and TLR7:4 ≠ 0000, RX_FIFO_TRIG is not considered.		
			If SCR7 = 1, RX_FIFO_TRIG is two LSBs of the trigger level (1 - 63 on 6 bits) with granularity of 1.		

Notes: 1) Bits 4 and 5 can only be written when EFR[4] = 1.

2) Bits 0 to 3 can be changed only when baud clock is not running (DLL and DLH set to 0).

3) See Table 12–36 for FCR[5:4] setting restriction when SCR[6] = 1.

4) See Table 12–37 for FCR[7:6] setting restriction when SCR[7] = 1.

Table 12–16. FIFO Control Register (FCR) (Continued)

Bit	Name	Value	Function	R/W	Reset Value
5–4	TX_FIFO_TRIG		Sets the trigger level for the TX FIFO:	W	00
			If SCR6 = 0 and TLR3:0 = 0000:		
		00	8 characters		
		01	16 characters		
		10	56 characters		
		11	60 characters		
			If SCR6 = 0 and TLR3:0 ≠ 0000, TX_FIFO_TRIG is not considered.		
			If SCR6 = 1, TX_FIFO_TRIG is two LSBs of the trigger level (1 - 63 on 6 bits) with granularity of 1.		
3	DMA_MODE	0	DMA_MODE 0 (No DMA)	W	0
		1	DMA_MODE 1 (UART_nDMA_REQ0 in TX, UART_nDMA_REQ1 in RX)		
			This register only has effect if SCR0 = 0.		
2	TX_FIFO_CLEAR	0	No change	W	0
		1	Clears the transmit FIFO and resets its counter logic to zero. Returns to zero after clearing FIFO.		
1	RX_FIFO_CLEAR	0	No change	W	0
		1	Clears the receive FIFO and resets its counter logic to zero. Returns to zero after clearing FIFO.		
0	FIFO_EN	0	Disables the transmit and receive FIFOs	W	0
		1	Enables the transmit and receive FIFOs		

Notes: 1) Bits 4 and 5 can only be written when EFR[4] = 1.
2) Bits 0 to 3 can be changed only when baud clock is not running (DLL and DLH set to 0).
3) See Table 12–36 for FCR[5:4] setting restriction when SCR[6] = 1.
4) See Table 12–37 for FCR[7:6] setting restriction when SCR[7] = 1.

Table 12–17. Supplementary Control Register (SCR)

Bit	Name	Value	Function	R/W	Reset Value
7	RX_TRIG_GRANU1	0	Disables the granularity of 1 for trigger RX level	R/W	0
		1	Enables the granularity of 1 for trigger RX level		
6	TX_TRIG_GRANU1	0	Disables the granularity of 1 for trigger TX level	R/W	0
		1	Enables the granularity of 1 for trigger TX level		
5	DSR_IT	0	Disables $\overline{\text{DSR}}$ interrupt	R/W	0
		1	Enables $\overline{\text{DSR}}$ interrupt		
4	RX_CTS_DSR_ WAKE_UP_ENABLE	0	Disables the wake up interrupt and clears SSR1	R/W	0
		1	Waits for a falling edge of pins RX, $\overline{\text{CTS}}$, or $\overline{\text{DSR}}$ to generate an interrupt		
3	TX_EMPTY_CTL_IT	0	Normal mode for THR interrupt (see Table 12–23)	R/W	0
		1	The THR interrupt is generated when TX FIFO and TX shift register are empty.		
2–1	DMA_MODE_2		Used to specify the DMA mode valid if SCR0 = 1	R/W	00
		00	DMA mode 0 (no DMA)		
		01	DMA mode 1 (UART_nDMA_REQ0 in TX, UART_nDMA_REQ1 in RX)		
		10	DMA mode 2 (UART_nDMA_REQ0 in RX)		
		11	DMA mode 3 (UART_nDMA_REQ0 in TX)		
0	DMA_MODE_CTL	0	The DMA_MODE is set with FCR3.	R/W	0
		1	The DMA_MODE is set with SCR2:1.		

Note: Bit 4 enables the wake-up interrupt, but this interrupt is not mapped on the IIR register. Therefore, when an interrupt occurs and if there is no interrupt pending in IIR, SSR[1] must be checked. To clear the wake-up interrupt, SCR[4] must be reset to 0.

Table 12–18. Line Control Register (LCR)

Bit	Name	Value	Function	R/W	Reset Value
7	DIV_EN	0	Normal operating condition	R/W	0
		1	Divisor latch enable. Allows access to DLL, DLH, and other registers (see the register mapping).		
6	BREAK_EN		Break control bit.	R/W	0
		0	Normal operating condition		
		1	Forces the transmitter output to go low to alert the communication terminal		
5	PARITY_TYPE2		Selects the forced parity format (if LCR3 = 1)	R/W	0
			If LCR5 = 1 and LCR4 = 0, the parity bit is forced to 1 in the transmitted and received data.		
			If LCR5 = 1 and LCR4 = 1, the parity bit is forced to 0 in the transmitted and received data.		
4	PARITY_TYPE1	0	Odd parity is generated (if bit 3 = 1).	R/W	0
		1	Even parity is generated (if bit 3 = 1).		
3	PARITY_EN	0	No parity	R/W	0
		1	A parity bit is generated during transmission and the receiver checks for received parity.		
2	NB_STOP		Specifies the number of stop bits:	R/W	0
		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)		

Note: As soon as LCR[6] is set to 1, the RX line is forced to 0 and remains in this state as long as LCR[6] = 1.

Table 12–18. Line Control Register (LCR) (Continued)

Bit	Name	Value	Function	R/W	Reset Value
1–0	CHAR_LENGTH		Specifies the word length to be transmitted or received.	R/W	00
		00	5 bits		
		01	6 bits		
		10	7 bits		
		11	8 bits		

Note: As soon as LCR[6] is set to 1, the RX line is forced to 0 and remains in this state as long as LCR[6] = 1.

Table 12–19. UART Mode Line Status Register (LSR)

Bit	Name	Value	Function	R/W	Reset Value
7	RX_FIFO_STS	0	Normal operation	R	0
		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.		
6	TX_SR_E	0	Transmitter hold and shift registers are not empty.	R	1
		1	Transmitter hold and shift registers are empty.		
5	TX_FIFO_E	0	Transmit hold register is not empty.	R	1
		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.		
4	RX_BI	0	No break condition	R	0
		1	A break was detected while the data being read from the RX FIFO was being received (i.e., RX input was low for one character time frame).		

Table 12–19. UART Mode Line Status Register (LSR) (Continued)

Bit	Name	Value	Function	R/W	Reset Value
3	RX_FE	0	No framing error in data being read from RX FIFO	R	0
		1	Framing error occurred in data being read from RX FIFO (i.e., received data did not have a valid stop bit).		
2	RX_PE	0	No parity error in data being read from RX FIFO	R	0
		1	Parity error in data being read from RX FIFO		
1	RX_OE	0	No overrun error	R	0
		1	Overrun error has occurred. Set when the character held in receive shift register is not transferred to the RX FIFO. This case can occurs only when receive FIFO is full.		
0	RX_FIFO_E	0	No data in the receive FIFO	R	0
		1	At least one data character in the RX_FIFO		

When the LSR is read, LSR[4:2] reflect 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, and PE (see Table 12–14).

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.

Note:

Reading the LSR does not cause an increment of the RX FIFO read pointer. The RX FIFO read pointer is incremented by reading the RHR.

Reading LSR clears OE, if OE is set (see Table 12–19).

Table 12–20. Supplementary Status Register (SSR)

Bit	Name	Value	Function	R/W	Reset Value
7–2	–		Reserved	R	000000
1	RX_CTS_DSR_WAKE_UP_STS	0	No falling edge event on RX, $\overline{\text{CTS}}$ and $\overline{\text{DSR}}$	R	0
		1	A falling edge occurred on RX, $\overline{\text{CTS}}$ or $\overline{\text{DSR}}$.		
0	TX_FIFO_FULL	0	TX FIFO not full	R	0
		1	TX FIFO full		

Note: Bit 1 is reset only when SCR[4] is reset to 0.

The modem control register (MCR)[3:0] controls the interface with the modem, data set, or peripheral device that is emulating the modem.

Table 12–21. Modem Control Register (MCR)

Bit	Name	Value	Function	R/W	Reset Value
7	CLKSEL	0	No action	R/W	0
		1	Divide clock input by 4		
6	TCR_TLR	0	No action	R/W	0
		1	Enables access to the TCR and TLR registers		
5	XON_EN	0	Disable XON any function	R/W	0
		1	Enable XON any function		
4	LOOPBACK_EN	0	Normal operating mode	R/W	0
		1	Enable local loop back mode (internal). In this mode the MCR3:0 signals are looped back into MSR7:4. The transmit output is looped back to the receive input internally.		
3	CD_STS_CH	0	In loopback mode, forces IRQ outputs to inactive state	R/W	0
		1	In loopback mode, forces IRQ outputs to inactive state		

Note: Bits 5, 6, and 7 can be written only when EFR[4] = 1.

Table 12–21. Modem Control Register (MCR) (Continued)

Bit	Name	Value	Function	R/W	Reset Value
2	RESERVED		Reserved. This bit must always be written as 0.	R/W	0
1	RTS	0	0: Forces $\overline{\text{RTS}}$ output to inactive (high)	R/W	0
		1	Forces $\overline{\text{RTS}}$ output to active (low) In loopback mode controls MSR4. If automatic RTS is enabled, the $\overline{\text{RTS}}$ output is controlled by hardware flow control.		
0	DTR	0	Forces $\overline{\text{DTR}}$ output to inactive (high)	R/W	0
		1	Forces $\overline{\text{DTR}}$ output to active (low)		

Note: Bits 5, 6, and 7 can be written only when EFR[4] = 1.

The modem status register (MSR) provides information about the current state of the control lines from the modem, data set, or peripheral device to the host (MPU or DSP). It also indicates when a control input from the modem changes state.

Table 12–22. Modem Status Register (MSR)

Bit	Name	Function	R/W	Reset Value
7–6	RESERVED	Reserved	R	Input signal
5	NDSR_STS	This bit is the complement of the $\overline{\text{DSR}}$ input. In loopback mode it is equivalent to MCR0	R	Input signal
4	NCTS_STS	This bit is the complement of the $\overline{\text{CTS}}$ input. In loopback mode it is equivalent to MCR1	R	Input signal
3–2	RESERVED	Reserved	R	0
1	DSR_STS	1: Indicates that $\overline{\text{DSR}}$ input (or MCR0 in loopback) has changed state. Cleared on a read	R	0
0	CTS_STS	1: Indicates that $\overline{\text{CTS}}$ input (or MCR1 in loopback) has changed state. Cleared on a read.	R	0

The interrupt enable register (IER) can be programmed to enable/disable any of the following interrupts:

- Receiver error
- RHR
- THR
- XOFF received
- $\overline{\text{CTS}}$/$\overline{\text{RTS}}$ change of state from low to high

These interrupts can be enabled/disabled individually. There is also a sleep mode enable bit.

Table 12–23. UART Mode Interrupt Enable Register (IER)

Bit	Name	Value	Function	R/W	Reset Value
7	CTS_IT	0	Disables the $\overline{\text{CTS}}$ interrupt	R/W	0
		1	Enables the $\overline{\text{CTS}}$ interrupt		
6	RTS_IT	0	Disables the $\overline{\text{RTS}}$ interrupt	R/W	0
		1	Enables the $\overline{\text{RTS}}$ interrupt		
5	XOFF_IT	0	Disables the XOFF interrupt	R/W	0
		1	Enables the XOFF interrupt		
4	SLEEP_MODE	0	Disables sleep mode	R/W	0
		1	Enables sleep mode (stops baud rate clock when the module is inactive)		
3	MODEM_STS_IT	0	Disables the modem status register interrupt	R/W	0
		1	Enables the modem status register interrupt		
2	LINE_STS_IT	0	Disables the receiver line status interrupt	R/W	0
		1	Enables the receiver line status interrupt		
1	THR_IT	0	Disables the THR interrupt	R/W	0
		1	Enables the THR interrupt		
0	RHR_IT	0	Disables the RHR interrupt and time out interrupt.	R/W	0
		1	Enables the RHR interrupt and time out interrupt.		

Note: Bits 4, 5, 6, and 7 can only be written when EFR[4] = 1.

The IIR is a read-only register that provides the source of the interrupt in a prioritized manner.

Table 12–24. UART Mode Interrupt Identification Register (IIR)

Bit	Name	Value	Function							R/W	Reset Value
7–6	FCR_MIRROR		Mirror the contents of FCR(0) on both bits							R	00
5–1	IT_TYPE		**Priority**	**5**	**4**	**3**	**2**	**1**	**Source**	R	00000
			1	0	0	0	1	1	Receiver line status error		
			2	0	0	1	1	0	RX time-out		
			2	0	0	0	1	0	RHR interrupt		
			3	0	0	0	0	1	THR interrupt		
			4	0	0	0	0	0	Modem interrupt		
			5	0	1	0	0	0	Xoff/special character		
			6	1	0	0	0	0	CTS, RTS, DSR change state from active (low) to inactive (high)		
0	IT_PENDING	0	An interrupt is pending (nIRQ active).							R	1
		1	No interrupt is pending (nIRQ inactive).								

The enhanced feature register (EFR) enables or disables enhanced features.

Table 12–25. Enhanced Feature Register (EFR)

Bit	Name	Value	Function	R/W	Reset Value
7	AUTO_CTS_EN		Automatic CTS enable bit	R/W	0
		0	Normal operation		
		1	Automatic CTS flow control is enabled; that is, transmission is halted when the $\overline{\text{CTS}}$ pin is high (inactive).		
6	AUTO_RTS_EN		Automatic RTS enable bit	R/W	0
		0	Normal operation		
		1	Automatic RTS flow control is enabled; that is, $\overline{\text{RTS}}$ 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.		

Table 12–25. Enhanced Feature Register (EFR) (Continued)

Bit	Name	Value	Function	R/W	Reset Value
5	SPECIAL_CHAR_ DETECT	0	Normal operation	R/W	0
		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 a special character has been detected.		
4	ENHANCED_EN		Enhanced functions write enable bit	R/W	0
		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.		
3–0	SW_FLOW_CON-TROL		Combinations of software flow control can be selected by programming bit 3-bit 0. See Section 12.5.10, *Software Flow Control.*	R/W	0

Table 12–26. EFR[0-3]: Software Flow Control Options

Bit 3	Bit 2	Bit 1	Bit 0	TX,RX Software Flow Controls
0	0	X	X	No transmit flow control
1	0	X	X	Transmit XON1, XOFF1
0	1	X	X	Transmit XON2, XOFF2
1	1	X	X	Transmit XON1, XON2: XOFF1, XOFF2
X	X	0	0	No receive flow control
X	X	1	0	Receiver compares XON1, XOFF1
X	X	0	1	Receiver compares XON2, XOFF2
X	X	1	1	Receiver compares XON1, XON2: XOFF1, XOFF2

Note: XON1 and XON2 must be set to different values if the software flow control is enabled.

Table 12–27. XON1 Register (XON1)

Bit	Name	Function	R/W	Reset Value
7–0	XON_WORD1	Used to store the 8-bit XON1 character	R/W	0x00

Table 12–28. XON2 Register (XON2)

Bit	Name	Function	R/W	Reset Value
7–0	XON_WORD2	Used to store the 8-bit XON2 character	R/W	0x00

Table 12–29. XOFF1 Register (XOFF1)

Bit	Name	Function	R/W	Reset Value
7–0	XOFF_WORD1	Used to store the 8-bit XOFF1 character	R/W	0x00

Table 12–30. XOFF2 Register (XOFF2)

Bit	Name	Function	R/W	Reset Value
7–0	XOFF_WORD2	Used to store the 8-bit XOFF2 character	R/W	0x00

The scratchpad register (SPR) does not control the module in any way; rather, it is used by the programmer to hold temporary data.

Table 12–31. Scratchpad Register (SPR)

Bit	Name	Function	R/W	Reset Value
7–0	SPR_WORD	Scratchpad register	R/W	0x00

The divisor latch low register (DLL) and divisor latch high register (DLH) store the 16-bit divisor for generation of the baud clock in the baud rate generator. DLH stores the most significant part of the divisor; DLL stores the least significant part of the divisor.

Note:

The DLL and DLH can only be written to before sleep mode is enabled (that is, before IER[4] is set).

Table 12–32. Divisor Latch Low Register (DLL)

Bit	Name	Function	R/W	Reset Value
7–0	CLOCK_LSB	Used to store the 8-bit LSB divisor value	R/W	0x00

Table 12–33. Divisor Latch High Register (DLH)

Bit	Name	Function	R/W	Reset Value
7–0	CLOCK_MSB	Used to store the 8-bit MSB divisor value	R/W	0x00

To achieve the required baud rate, you must program DLL/DLH with the integer part of the divisor value.

Choosing the appropriate divisor value:

UART: Divisor value = Operating Frequency / (16 x Baud Rate)

Just as in autobaud mode, the input frequency of the UART modem must be fixed to the operating frequency (here 12 MHz; no CLKSEL bit setting) and the the OSC_12M_SEL bit must be set to be able to reach desired baud rate. Setting OSC_12M_SEL to 1 enables turning on the 6.5 division factor. For instance, 12 MHz/16/6.5 = 115200 bps; in case OSC_12M_SEL is not set, reached baud rate is either 12 MHz/16/6 or 12 MHz/16/7, which are out of permitted tolerance.

The transmission control register (TCR) stores the receive FIFO threshold levels to start/stop transmission during hardware/software flow control.

Table 12–34. Transmission Control Register (TCR)

Bit	Name	Function	R/W	Reset Value
7–4	RX_FIFO_TRIG_START	RCV FIFO trigger level to RESTORE transmission (0–60)	R/W	0000
3–0	RX_FIFO_TRIG_HALT	RCV FIFO trigger level to HALT transmission (0–60)	R/W	1111

Notes: 1) Trigger levels from 0 - 60 bytes are available with a granularity of four (Trigger level = 4 x [4-bit register value]).

2) The programmer must ensure that TCR[3:0] > TCR[7:4] whenever automatic RTS or software flow control is enabled to avoid a faulty operation of the device.

3) In FIFO interrupt mode with flow control, programmer must also ensure that trigger level to HALT transmission is greater or equal to receive FIFO trigger level (either TLR[7:4] or FCR[7:6]): otherwise, FIFO operation stalls. In FIFO DMA mode with flow control, this issue does not exist because a DMA request is sent each time a byte is received.

The trigger level register (TLR) stores the programmable transmit and receive FIFO trigger levels used for DMA and IRQ generation.

Table 12–35. Trigger Level Register (TLR)

Bit	Name	Function	R/W	Reset Value
7–4	RX_FIFO_TRIG_DMA	Receive FIFO trigger level	R/W	0000
3–0	TX_FIFO_TRIG_DMA	Transmit FIFO trigger level	R/W	0000

Table 12–36 and Table 12–37 summarize the different ways to set the trigger levels for the transmit FIFO and the receive FIFO.

Table 12–36. TX FIFO Trigger Level Setting Summary

SCR[6]	TLR[3:0]	TX FIFO Trigger Level
0	0000	Defined by FCR5:4 (either 8, 16, 32, 56 spaces)
0	≠0000	Defined by TLR3:0 (from 4 to 60 spaces with a granularity of 4 spaces)
1	Any value	Defined by the concatenated value of TLR3:0 and FCR 5:4 (from 1 to 63 spaces with a granularity of 1 space). The combination of TLR3:0 = 0000 and FCR 5:4 = 00 (all zeros) is not supported (minimum 1 space required). All zeros result in unpredictable behavior.

Note: The protocol to set the concatenation of TLR and FCR is:
- Set SCR[6] = 0
- Set the value of threshold into FCR and TLR
- Set SCR[6] = 1

Table 12–37. RX FIFO Trigger Level Setting Summary

SCR[7]	TLR[7:4]	RX FIFO Trigger Level
0	0000	Defined by FCR7:6 (either 8, 16, 56, 60 characters)
0	≠0000	Defined by TLR7:4 (from 4 to 60 characters with a granularity of 4 characters)
1	Any value	Defined by the concatenated value of TLR7:4 and FCR 7:6 (from 1 to 63 characters with a granularity of 1 character). The combination of TLR7:4 = 0000 and FCR 7:6 = 00 (all zeros) is not supported (minimum 1 character required). All zeros result in unpredictable behavior.

Note: The protocol to set the concatenation of TLR and FCR is:
- Set SCR[7] = 0
- Set the value of threshold into FCR and TLR
- Set SCR[7] = 1

The mode of operation can be programmed by writing to MDR1[2:0]; therefore the MDR1 must be programmed on start-up after configuration of the configuration registers (DLL, DLH, LCR). The value of MDR1[2:0] must not be changed again during normal operation.

Table 12–38. Mode Definition Register 1 (MDR1)

Bit	Name	Value	Function	R/W	Reset Value
7–3	–		Reserved	R/W	00000
2–0	MODE_SELECT†		000: UART	R/W	111
		010	UART with autobauding		
		111	Disables UART/default state		
			All the other values are reserved.		

† The MODE_SELECT = 0x7 setting disables the UART module by disabling the FIFO and the state machine. It does not gate the functional clock to the module. The lowest power state is not achieved by setting MODE_SELECT = 0x7, but by putting the UART into sleep mode. The lowest power state is achieved when in sleep mode witht DLL = 0xFFFF and DLH = 0xFFFF.

The UART autobauding status register (UASR) returns the speed, the number of bits by characters, and the type of the parity in UART autobaud mode.

Table 12–39. Autobauding Status Register (UASR)

Bit	Name	Value	Function	R/W	Reset Value
7–6	PARITY_TYPE	00	00: No parity identified	R	00
		01	Parity space		
		10	Even parity		
		11	Odd parity		
5	BIT_BY_CHAR	0	7-bit character identified	R	0
		1	8-bit character identified		

Note: This register is used to set up transmission according to characteristics of previous reception instead of LCR, DLL, and DLH registers when UART is in autobaud 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 then to the UART in autobaud mode 010 or UART in standard mode 000.

Table 12–39. Autobauding Status Register (UASR) (Continued)

Bit	Name	Value	Function	R/W	Reset Value
4–0	SPEED		Used to report the speed identified	R	0000
		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		

Note: This register is used to set up transmission according to characteristics of previous reception instead of LCR, DLL, and DLH registers when UART is in autobaud 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 then to the UART in autobaud mode 010 or UART in standard mode 000.

Table 12–40. OSC_12_MHz Register Select (OSC_12M_SEL)

Bit	Name	Function	R/W	Reset Value
7–1	–	Reserved	R	0000000
0	OSC_12M_SEL†	When 1, selects 6.5 division factor with a 12-MHz system clock.	W	0

† This register is write-only and cannot be read.

Table 12–41. Module Version Register (MVR)

Bit	Name	Function	R/W	Reset Value
7-4	MAJOR_REV	Major revision number of the module	R	---†
3-0	MINOR_REV	Minor revision number of the module	R	---

† For example: MVR = 0x11 => Version 1.1

12.4 UART/Autobaud Modes of Operation

The UART/autobaud module can operate in two different modes: UART mode and UART with autobaud mode.

The modules perform serial-to-parallel conversion on data characters received and parallel-to-serial conversion on data characters transmitted by the processor. The complete status of each channel of the modules and each received character/frame can be read at any time during functional operation via the line status register (LSR).

You can place the modules in an alternate mode (FIFO mode) to relieve the processor of excessive software overhead by buffering received/transmitted characters. Both the receiver and transmitter FIFOs can store up to 64 bytes of data (plus three 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 host (MPU or DSP) and the module.

12.4.1 UART Mode

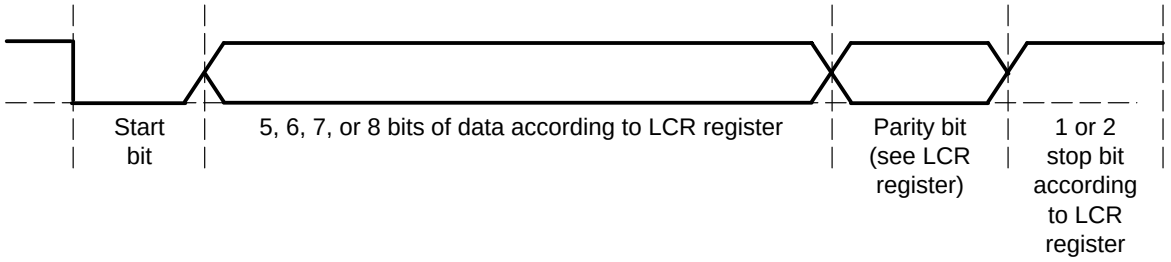
The UART modem uses a wired interface for serial communication with a remote device.

The UART modem module is functionally compatible with the TL16C750 UART and is also functionally compatible with earlier designs such as the TL16C550. The UART modem module can use hardware or software flow controls to manage transmission/reception. Hardware flow control significantly reduces software overhead and increases system efficiency by automatically controlling serial data flow using the $\overline{\text{RTS}}$ output and $\overline{\text{CTS}}$ input signals. Software flow control automatically controls data flow by using programmable XON/XOFF characters.

12.4.2 UART Mode With Autobauding

The UART modem module is enhanced with an autobauding functionality, which in control mode allows automatically setting the speed, the number of bits per character, and the parity selected (see Figure 12–7).

Figure 12–7. UART Data Format

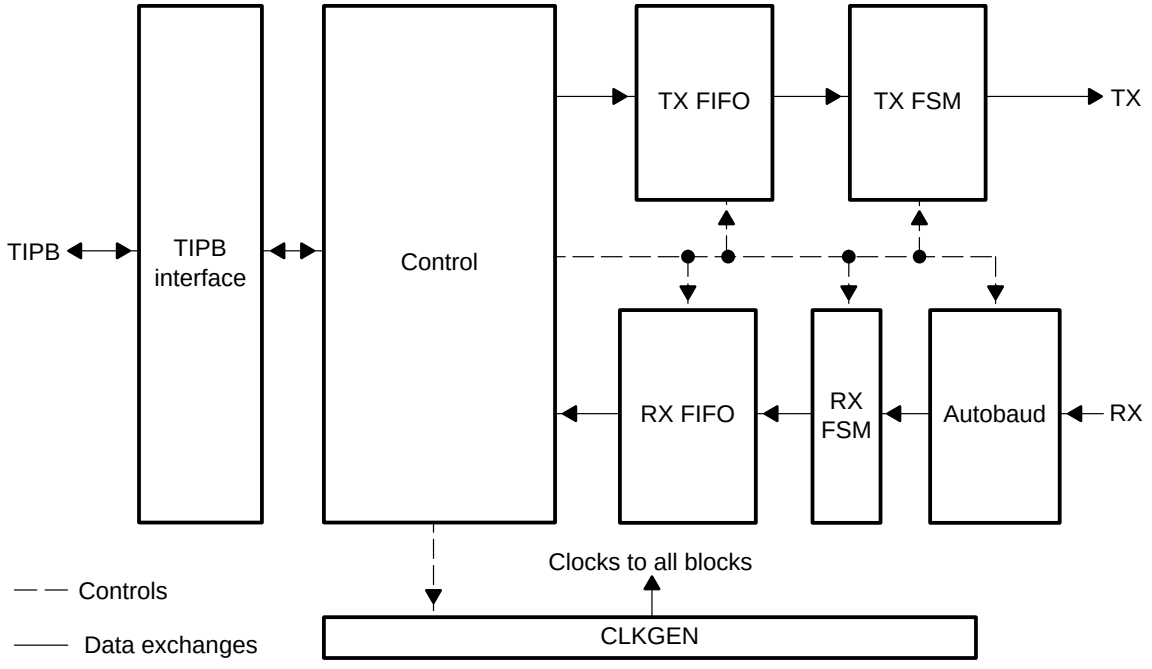


12.5 UART/Autobaud Functional Description

12.5.1 UART/Autobaud Functional Block Diagram

Figure 12–8 shows the UART/autobaud (FSM stands for finite state machine).

Figure 12–8. Functional Block Diagram



12.5.2 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 (in effect, the trigger level is the default value of one byte). Programmable trigger level is an enhanced feature available via the trigger level register (TLR).

12.5.3 Interrupts

The UART generates interrupts on the UART_nIRQ output pin. All interrupts can be 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).

12.5.3.1 Generic Interrupts Description

There are seven possible interrupts, prioritized to six different levels.

When an interrupt is generated, the interrupt identification register (IIR) indicates a pending interrupt by bringing IIR[0] to logic 0, and it specifies the type of interrupt through IIR[5-1]. Table 12–42 summarizes the interrupt control functions.

Table 12–42. Generic Interrupt Descriptions in Modem Mode

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: All erroneous characters are read form the RX FIFO. 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	MSR1:0/ = 0	Read MSR

Table 12–42. Generic Interrupt Descriptions in Modem Mode (Continued)

IIR[5-0]	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Method
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	$\overline{\text{CTS}}$, $\overline{\text{RTS}}$, $\overline{\text{DSR}}$	$\overline{\text{RTS}}$ pin, $\overline{\text{CTS}}$ pin or $\overline{\text{DSR}}$ pin change state from active (low) to inactive (high).	Read IIR

Note: Once LSR[7] (RX_FIFO_STS) is set to FIFO disable (FCR[0]=0), this bit bit cannot be cleared by reading LSR. First, FCR[1] (RX_FIFO_CLERA) must be set to 1, then LSR[7] can be cleared.

LSR[7] generates the receiver line status interrupt.

For the XOFF interrupt, if an XOFF flow character detection caused the interrupt, the interrupt is cleared by an XON flow character detection. If special character detection caused the interrupt, the interrupt is cleared by a read of the interrupt identification register (IIR).

12.5.3.2 Wake-Up Interrupt

The wake-up interrupt is uniquely designed and is enabled when SCR[4] is set to 1. The interrupt identification register (IIR) is not modified when this interrupt 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.

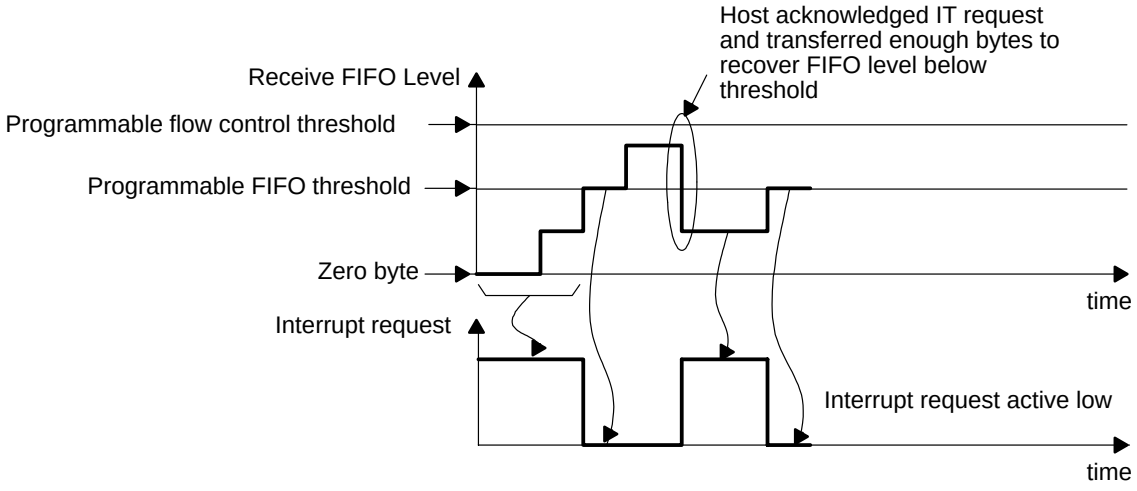
12.5.3.3 FIFO Interrupt Mode

In FIFO interrupt mode, FCR[0] = 1 and relevant interrupts are enabled via the interrupt enable register (IER). The processor is informed of the status of the receiver and transmitter by an interrupt signal, nIRQ. These interrupts are raised when receive/transmit FIFO threshold (respectively TLR[7:4] and TLR[3:0] or FCR[7:6] and FCR[5:4]) are reached; they instruct the host (MPU or DSP) to transfer data to the destination (from UART module in receive mode and from any source to UART FIFO in transmit mode).

When UART flow control is enabled along with interrupt capabilities, you must ensure that the UART flow control FIFO threshold (TCR[3:0]) is greater than or equal to the receive FIFO threshold.

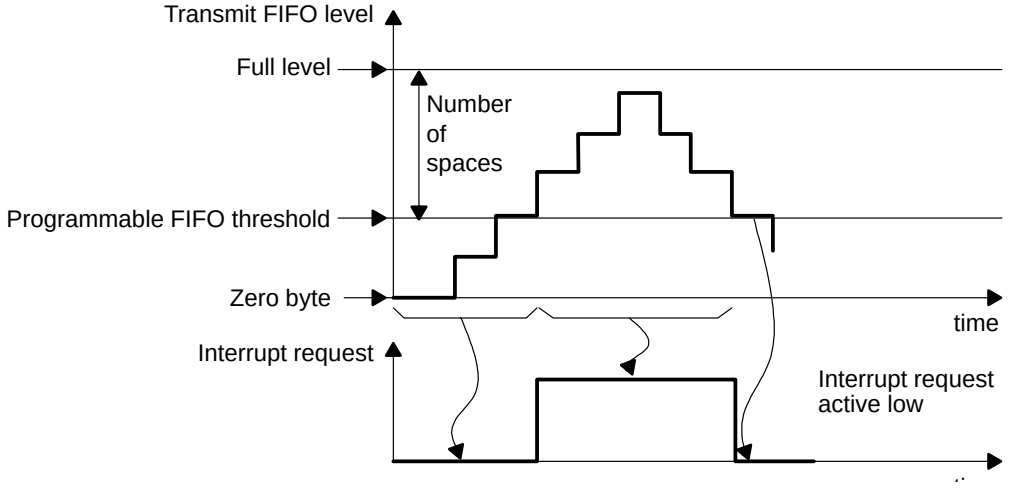
Figure 12–9 shows receive IT operations; Figure 12–10 shows transmit IT operations.

Figure 12–9. Receive FIFO IT Request Generation



In receive, no interrupt is generated until receive FIFO reaches its threshold. A low interrupt can only be deasserted when the host (MPU or DSP) has handled enough bytes to make the FIFO level below threshold. Notice that flow control threshold is set at a higher value than FIFO threshold.

Figure 12–10. Transmit FIFO IT Request Generation



In transmit mode, an interrupt request is automatically asserted when the 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 have been transmitted to go below FIFO threshold.

12.5.4 FIFO Polled Mode

In FIFO polled mode (FCR [0] = 0, relevant interrupts disabled via IER) the status of the receiver and transmitter can be checked by polling the line status register (LSR). This mode is an alternative to the FIFO interrupt mode of operation where the status of the receiver and transmitter is automatically known by means of interrupts sent to the host (MPU or DSP).

12.5.5 FIFO DMA Mode

12.5.5.1 DMA Signalling

There are four modes of DMA operation: DMA mode 0, DMA mode 1, DMA mode 2, and DMA mode 3. They can be selected as follows.

- When SCR[0] = 0:
  - Setting FCR[3] to 0 enables DMA mode 0.
  - Setting FCR[3] to 1 enables DMA mode 1.
- When SCR[0] = 1, SCR[2:1] determine DMA mode 0 to 3 according to SCR register description.

So, for instance:

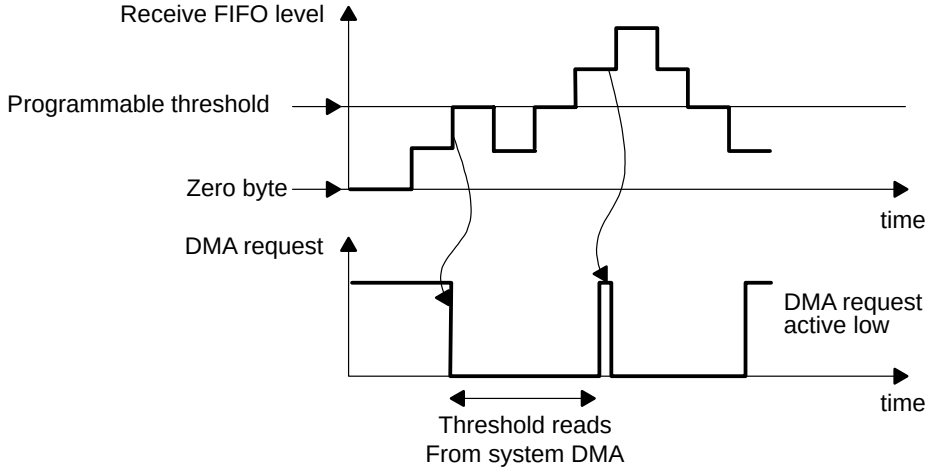
- If no DMA operation is desired, set SCR[0] to 1 and SCR[2:1] to 00 (FCR[3] is disregarded).
- If DMA mode 1 is desired, either set SCR[0] to 0 and FCR[3] to 1 or set SCR[0] to 1 SCR[2:1] to 01 (FCR[3] is disregarded).

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.

12.5.5.2 DMA Transfers

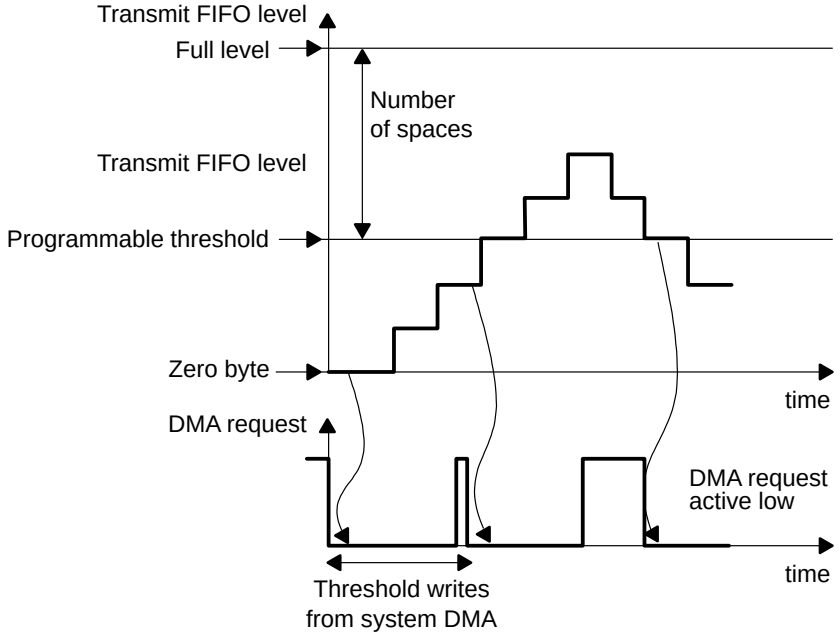
Figure 12–11 shows DMA operation at receive; Figure 12–12 shows DMA operation at transmit.

Figure 12–11. Receive FIFO DMA Request Generation



In receive mode, a DMA request is generated as soon as receive FIFO reaches its threshold. This request is deasserted when the number of bytes defined by the threshold level has been read by the system DMA.

Figure 12–12. Transmit FIFO DMA Request Generation



In transmit mode, a DMA request is automatically asserted when FIFO is empty. This request is deasserted when the number of bytes defined by the threshold level has been written by the system DMA. The DMA request is again asserted if the FIFO is able to receive the number of bytes defined by the threshold.

12.5.6 Sleep Mode

Sleep mode is a low-power, enhanced feature of the UART that can be enabled by writing a 1 to IER[4] (when EFR[4] = 1).

Sleep mode is entered when:

- Serial RX data input line is idle.
- TX FIFO and TX shift register are empty.
- No interrupts are pending except transmit holding register (THR) interrupts.

Sleep mode is a good way to lower UART power consumption, but this state can be achieved only when the UART is set in modem mode. Therefore, even if the UART does not have a 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. Since 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, when data is written to the TX FIFO, or when there is any change in the state of the modem input pins. An interrupt can be generated on a wake-up event by setting SCR[4] to 1.

Note:

Writing to the divisor latches, DLL and DLH, to set the baud clock, BCLK, must not be done during sleep mode. Disable sleep mode using IER[4] before writing to DLL or DLH.

12.5.7 Break and Time-out Conditions

- Time-out counter

  The RX idle condition is detected when the RX line has been high for a time equivalent to (4X programmed word length) plus12 bits. The receiver line is sampled midway through each bit.

  For sleep mode, the counter is reset when there is activity on the RX line.

  For the time-out interrupt, the counter only counts when there is data in the RX FIFO and the count is reset when there is activity on the RX line or when the RHR is read.

- Break condition

  When a break condition occurs, the TX line is pulled low. A break condition is activated by setting LCR[6]. The break condition is not aligned on the word stream; that is, a break condition can occur in the middle of a character. The only way to send a break condition on a full character is as follows:

  - 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.

  The break condition is asserted as long as LCR[6] is set to 1.

12.5.8 Programmable Baud Rate Generator

The programmable baud generator takes any clock input and divides it by a divisor between 1 and (2^{16-1}). The CLKSEL register bit MCR[7] can be used to select the 1X or 1X/4 clock for the internal baud rate generator. The output frequency of the baud rate generator is 16x the baud rate.

You must write to the DLL register (least significant bytes) and DLH register (most significant bytes) of the baud rate divisor to program the baud rate.

Writing to these registers can result in wait states being inserted during the write access while the baud rate generator is loaded with the new value. If both registers are 0, the module is effectively disabled and no baud clock is generated.

Note:

The programmable baud rate generator selects both the transmit and receive clock rates.

12.5.9 Hardware Flow Control

Hardware flow control is composed of automatic RTS and automatic CTS. Both can be enabled/disabled independently by programming EFR[7:6]. With automatic CTS, $\overline{\text{CTS}}$ must be active before the module can transmit data.

Automatic RTS only activates the $\overline{\text{RTS}}$ output when there is enough room in the FIFO to receive data, and it 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 automatic CTS and automatic 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 not enabled, overrun errors occur if the transmit data rate exceeds the receive FIFO latency.

- Automatic RTS

  Automatic RTS data flow control originates in the receiver block (see Figure 12–8). The receiver FIFO trigger levels used in automatic 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.

- Automatic 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 automatic CTS function reduces interrupts to the host system. When automatic 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 automatic CTS, the transmitter sends any data present in the transmit FIFO and a receiver overrun error can result.

12.5.10 Software Flow Control

Software flow control is enabled through the enhanced feature register (EFR) and the modem control register (MCR). Different combinations of software flow control can be enabled by setting different combinations of EFR[3-0].

There are two other enhanced features related to software flow control:

- ❏ XON any function [MCR(5)]: Operation resumes 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. The XOFF interrupt is cleared by a read of the interrupt identification register (IIR). The special character is transferred to the RX FIFO.

12.5.10.1 RX

When software flow control operation is enabled, the UART compares incoming data with XOFF1/2 programmed characters (in certain cases XOFF1 and XOFF2 must be received sequentially). When the correct XOFF characters are received, transmission is halted after completing transmission of the current character. XOFF detection also sets IIR(4) (if enabled via IER(5)) and causes 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, this character is treated as normal data and is written to the RX FIFO.

When XON-any and special character detect are disabled and software flow control is enabled, no valid XON or XOFF characters are written to the RX FIFO. For example, if EFR[1:0] = 10, then received XON1 and XOFF1 characters are not written to the RX FIFO.

When pairs of software flow characters are programmed to be received sequentially (EFR[1:0] = 11), they 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.

12.5.10.2 TX

With XOFF1, two characters are transmitted when the RX FIFO has passed the programmed trigger level TCR(3:0).

With XON1, two characters are transmitted when the RX FIFO reaches the trigger level programmed via TCR(7:4).

After an XOFF character has been sent, if software flow control is disabled, the module transmits XON characters automatically to enable normal transmission to proceed.

The transmission of XOFF/XON follows the exact 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, the 5, 6, or 7 least significant bits of XOFF1,2/XON1,2 are transmitted. (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 are never enabled simultaneously.

12.5.11 Autobauding Mode

In autobaud mode, UART can extract 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/

A line break during the acquisition of the sequence AT is not recognized and echo functionality is not implemented in hardware.

A/ and a/ are not used to extract characteristics, but they must be recognized because of their special meaning. They 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/.

When a valid AT is received, it and all subsequent data are saved into the FIFO, including final CR (0x0D). Then the autobaud state machine waits for the next valid AT command. If an a/ (A/) is received, the a/ (A/) is saved into the FIFO and the state machine waits for next valid AT command.

The following settings are allowed in autobaud mode:

- Speed:

  115.2K baud, 57.6K baud, 38.4K baud, 28.8K baud, 19.2K baud, 14.4K baud, 9.6K baud, 4.8K baud, 2.4K baud, or 1.2K baud.

- Length: 7 or 8 bits

- Parity: Odd, even, or space

Combination 7-bit space parity is forbidden.

The method 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 three 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 three sampling periods.

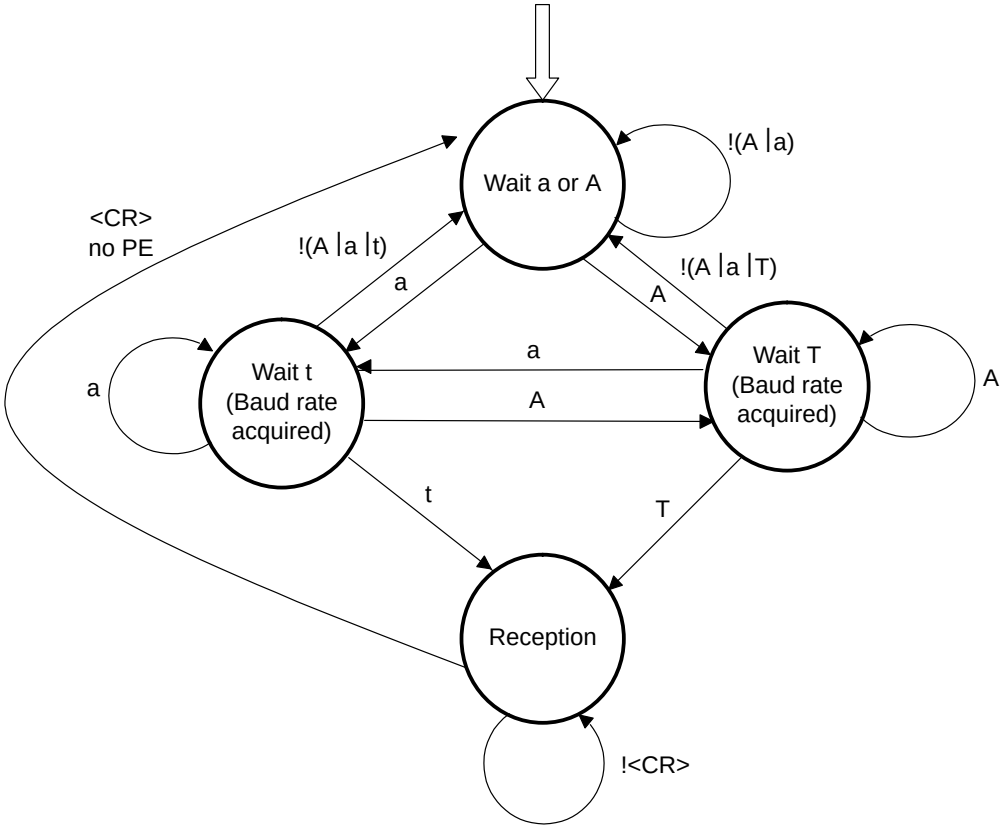
3) Compare the sampled value with a table. If the sampled value is outside a valid range, an error is reported (no speed identified), and the hardware goes back to the first state (1).

4) Otherwise, the first data bit in the received register (for serial to parallel conversion) is stored and goes to frame format identification.

5) The next received bits are sampled according to the programmed baud rate. After reception of seven bits, the speed identification must be restarted, because you may receive several a or A characters before a valid t or T character.

Autobaud mode is selected when MDR1[2:0] = 010. In UART autobaud mode, DLL, DLH, and LCR[5:0] settings are not used. Instead, UASR is updated with the configuration detected by the autobauding logic (see Figure 12–13).

Figure 12–13. Autobaud State Machine



12.6 UART/Autobaud Configuration Example

This section specifies the programming stages to operate one UART module with FIFO, interrupt, and no DMA capabilities. It is a three-step procedure that ensures quick start of these modules (it does not cover all UART module features and performance):

1) Software reset of the module (interrupts, status and controls
2) FIFO configuration and enable
3) Baud rate data and stop configuration

This procedure is programming-language-agnostic.

12.6.1 UART SW Reset

The goal is to clear IER and MCR registers, remove UART breaks (LCR[6] = 0), and put the module in reset (MDR1[2:0] = 0x3).

1) To write into both the IER and MCR registers, set EFR[4] to 1.
2) To enable access to the EFR register, write 0xBF to the LCR register:
   - LCR = 0xBF
   - EFR[4] = 1
   - LCR = 0x80 (access to IER and MCR allowed)
   - IER = 0x00
   - MCR = 0x00
   - LCR[6] = 0 (UART breaks removed)
   - MDR1 = 0x03 (UART in reset)

12.6.2 UART FIFO Configuration

The goal is to set trigger level for halt/restore (TCR register), set trigger level for transmit/receive (TLR register), and configure the FIFO (FCR register).

Procedure:

1) To write into both the TLR and TCR registers, set EFR[4] and MCR[6] to 1. To write into FCR, set EFR[4] to 1. Notice that EFR[4] = 1 has already been done in the software reset, so a simple write to MCR[6] is necessary.
2) Set TCR TLR and FCR to the desired value.
3) Disable accesses to TCR, TLR, and FCR to avoid any further undesired write to these registers:
   - LCR = 0xBF (provides EFR access)
   - EFR[4] = 0
   - LCR[7] = 0
   - MCR[6] = 0

12.6.3 Baud Rate Data and Stop Configurations

The goal is to configure UART data, stop (LCR register) baud rate (DLH and DLL registers), and enable UART operation. If needed, you can add interrupt capability configuration right before UART enable.

1) Input clock is 12 MHz, so set OSC_12M_SEL to 1.
2) Set LCR to desired value.
3) Set LCR[7] to 1 (access to DLH and DLL registers).
4) Set DLH and DLL LCR[7] = 0 (remove access to DLH and DLL registers).
5) Set IER to desired value (set interrupts).
6) MDR1[2:0] = 0
7) Enable UART without autobauding.

12.7 UART/IrDA Control and Status Registers

Each register is selected using a combination of address and some LCR register bit settings, as shown in Table 12–43.

UART3 is accessible as follows:

- MPU (32-bit aligned byte address) from the following base address:
  - UART3: 0xFFFB 9800
- DSP (16-bit aligned word address) from the following base address:
  - UART3: 0x00CC00

Table 12–43. UART IrDA Register Program

MPU Byte Offset	DSP Byte Offset	Registers					
		LCR[7] = 0		LCR[7] = 1 LCR[7:0] ≠ 0xBF		LCR[7:0] = 0xBF	
		READ	WRITE	READ	WRITE	READ	WRITE
0x00	0x00	RHR	THR	DLL	DLL	DLL	DLL
0x04	0x02	IER†	IER†	DLH	DLH	DLH	DLH
0x08	0x04	IIR	FCR‡	IIR	FCR‡	EFR	EFR
0x0C	0x06	LCR	LCR	LCR	LCR	LCR	LCR
0x10	0x08	MCR‡	MCR‡	MCR‡	MCR‡	XON1/ADDR1	XON1/ADDR1
0x14	0x0A	LSR	-	LSR	-	XON2/ADDR2	XON2/ADDR2
0x18	0x0C	MSR/TCR§	TCR§	MSR/TCR§	TCR§	XOFF1/TCR§	XOFF1/TCR§
0x1C	0x0E	SPR/TLR§	SPR/TLR§	SPR/TLR§	SPR/TLR§	XOFF2/TLR§	XOFF2/TLR§
0x20	0x10	MDR1	MDR1	MDR1	MDR1	MDR1	MDR1
0x24	0x12	MDR2	MDR2	MDR2	MDR2	MDR2	MDR2
0x28	0x14	SFLSR	TXFLL	SFLSR	TXFLL	SFLSR	TXFLL
0x2C	0x16	RESUME	TXFLH	RESUME	TXFLH	RESUME	TXFLH
0x30	0x18	SFREGL	RXFLL	SFREGL	RXFLL	SFREGL	RXFLL
0x34	0x1A	SFREGH	RXFLH	SFREGH	RXFLH	SFREGH	RXFLH

† In UART mode, IER[7:4] can only be written when EFR[4] = 1. In SIR mode, EFR[4] has no impact on the 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 12–43. UART IrDA Register Program (Continued)

MPU Byte Off-set	DSP Byte Off-set	Registers					
		LCR[7] = 0		LCR[7] = 1 LCR[7:0] ≠ 0xBF		LCR[7:0] = 0xBF	
		READ	WRITE	READ	WRITE	READ	WRITE
0x38	0x1C	BLR	BLR	-	-	-	-
0x3C	0x1E	ACREG	ACREG	DIV1.6	DIV1.6	DIV1.6	DIV1.6
0x40	0x20	SCR	SCR	SCR	SCR	SCR	SCR
0x44	0x22	SSR	-	SSR	-	SSR	-
0x48	0x24	EBLR	EBLR	-	-	-	-
0x4C	0x26		OSC_12M_ SEL	-	-	-	-
0x50	0x28	MVR	-	MVR	-	MVR	-

† In UART mode, IER[7:4] can only be written when EFR[4] = 1. In SIR mode, EFR[4] has no impact on the 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 12–44 lists the UART/IrDA registers. Table 12–45 through Table 12–87 describe the register bits.

Table 12–44. UART/IrDA Registers

Register	Description	Access
RHR	Receive holding	8 bits R
THR	Transmit holding	8 bits W
FCR	FIFO control	8 bits W
SCR	Supplementary control	8 bits R/W
LCR	Line control	8 bits R/W
LSR	UART mode (LSR)	8 bits R
SSR	Supplementary status	8 bits R
MCR	Modem control	8 bits R/W
MSR	Modem status	8 bits R
IER	Interrupt enable (IER)	8 bits R/W

Table 12–44. UART/IrDA Registers (Continued)

Register	Description	Access
IIR	Interrupt identification (IIR)	8 bits R
EFR	Enhanced feature	8 bits R/W
XON1/ADDR1	XON1/Address 1	8 bits R/W
XON2/ADDR2	XON2/Address 2	8 bits R/W
XOFF1	XOFF1	8 bits R/W
XOFF2	XOFF2	8 bits R/W
SPR	Scratchpad	8 bits R/W
DLL	Divisor latch low	8 bits R/W
DLH	Divisor latch high	8 bits R/W
TCR	Transmission control	8 bits R/W
TLR	Trigger level	8 bits R/W
MDR1	Mode definition 1	8 bits R/W
MDR2	Mode definition 2	8 bits R/W
TXFLL	Transmit frame length low	8 bits W
TXFLH	Transmit frame length high	8 bits W
RXFLL	Received frame length low	8 bits W
RXFLH	Received frame length high	8 bits W
SFLSR	Status FIFO line status	8 bits R
RESUME	Resume	8 bits R
SFREGL	Status FIFO low	8 bits R
SFREGH	Status FIFO high	8 bits R
BLR	BOF control	8 bits R/W
EBLR	BOF length	8 bits R/W
DIV16	DIV1.6	8 bits R/W
ACREG	Auxiliary control	8 bits R/W
OSC_12M_SEL	OSC 12-MHz select	8 bits W
MVR	Module version	8 bits R

The receiver section consists of the receiver holding register (RHR) and the receiver shift register. The RHR is actually a 64-byte FIFO. The receiver shift register receives serial data from RX input. The data is converted to parallel data and moved to the RHR. If the FIFO is disabled, location zero of the FIFO is used to store the single data character. If overflow occurs, data in the RHR is not overwritten.

Table 12–45. Receive Holding Register (RHR)

Bit	Name	Function	R/W	Reset Value
7–0	RHR	Receive holding register	R	Undefined

The transmitter section consists of the transmit holding register (THR) and the transmit shift register. The THR is actually a 64-byte FIFO. The host (MPU or DSP) 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 12–46. Transmit Holding Register (THR)

Bit	Name	Function	R/W	Reset Value
7–0	THR	Transmit holding register	W	Undefined

Table 12–47. FIFO Control (FCR) Register

Bit	Name	Value	Function	R/W	Reset Value
7–6	RX_FIFO_TRIG		Sets the trigger level for the RX FIFO:	W	00
			If SCR7 = 0 and TLR7:4 = 0000:		
		00	8 characters		
		01	16 characters		
		10	56 characters		
		11	60 characters		
			If SCR7 = 0 and TLR7:4 ≠ 0000, RX_FIFO_TRIG is not considered.		
		1	RX_FIFO_TRIG is two LSBs of the trigger level (1-63 on 6 bits) with granularity of 1.		
5–4	TX_FIFO_TRIG		Sets the trigger level for the TX FIFO:	W	00
			If SCR6 = 0 and TLR3:0 = 0000:		
			00: 8 spaces 01: 16 spaces 10: 32 spaces 11: 56 spaces		
		00	8 characters		
		01	16 characters		
		10	56 characters		
		11	60 characters		
			If SCR6=1, TX_FIFO_TRIG is two LSBs of the trigger level (1-63 on 6 bits) with granularity of 1.		

Notes:
1) Bits 4 and 5 can only be written when EFR[4] = 1.
2) Bits 0 to 3 can be changed only when baud clock is not running (DLL and DLH set to 0).
3) See Table 12–36 for FCR[5:4] setting restriction when SCR[6] = 1.
4) See Table 12–37 for FCR[7:6] setting restriction when SCR[7] = 1.

Table 12–47. FIFO Control (FCR) Register (Continued)

Bit	Name	Value	Function	R/W	Reset Value
3	DMA_MODE	0	DMA_MODE 0 (no DMA)	W	0
		1	DMA_MODE 1 (UART_nDMA_REQ0 in TX, UART_nDMA_REQ1 in RX)		
			This register is considered if SCR0 = 0.		
2	TX_FIFO_CLEAR	0	No change	W	0
		1	Clears the transmit FIFO and resets its counter logic to zero. Returns to zero after clearing FIFO.		
1	RX_FIFO_CLEAR	0	No change	W	0
		1	Clears the receive FIFO and resets its counter logic to zero. Returns to zero after clearing FIFO.		
0	FIFO_EN	0	Disables the transmit and receive FIFOs	W	0
		1	Enables the transmit and receive FIFOs		

Notes: 1) Bits 4 and 5 can only be written when EFR[4] = 1.
2) Bits 0 to 3 can be changed only when baud clock is not running (DLL and DLH set to 0).
3) See Table 12–36 for FCR[5:4] setting restriction when SCR[6] = 1.
4) See Table 12–37 for FCR[7:6] setting restriction when SCR[7] = 1.

Table 12–48. Supplementary Control Register (SCR)

Bit	Name	Value	Function	R/W	Reset Value
7	RX_TRIG_GRANU1	0	Disables the granularity of 1 for trigger RX level	R/W	0
		1	Enables the granularity of 1 for trigger RX level		
6	TX_TRIG_GRANU1	0	Disables the granularity of 1 for trigger TX level	R/W	0
		1	Enables the granularity of 1 for trigger TX level		
5	DSR_IT	0	Disables $\overline{\text{DSR}}$ interrupt	R/W	0
		1	Enables $\overline{\text{DSR}}$ interrupt		
4	RX_CTS_DSR_WAKE_UP_ENABLE	0	Disables the wake up interrupt and clears SSR1	R/W	0
		1	Waits for a falling edge of pins RX, $\overline{\text{CTS}}$ or $\overline{\text{DSR}}$ to generate an interrupt		
3	TX_EMPTY_CTL_IT	0	Normal mode for THR interrupt (see Table 12–55)	R/W	0
		1	The THR interrupt is generated when TX FIFO and TX shift register are empty.		
2–1	DMA_MODE_2		Used to specify the DMA mode valid if SCR0 = 1	R/W	00
		00	DMA mode 0 (no DMA)		
		01	DMA mode 1 (UART_nDMA_REQ0 in TX, UART_nDMA_REQ1 in RX)		
		10	DMA mode 2 (UART_nDMA_REQ0 in RX)		
		11	DMA mode 3 (UART_nDMA_REQ0 in TX)		
0	DMA_MODE_CTL	0	The DMA_MODE is set with FCR3.	R/W	0
		1	The DMA_MODE is set with SCR2:1.		

Note: Bit 4 enables the wake-up interrupt, but this interrupt is not mapped on the IIR register. Therefore, when an interrupt occurs and if there is no interrupt pending in IIR, SSR[1] must be checked. To clear the wake-up interrupt, SCR[4] must be reset to 0.

The line control register (LCR) [6:0] defines parameters of the transmission and reception.

Table 12–49. Line Control Register (LCR)

Bit	Name	Value	Function	R/W	Reset Value
7	DIV_EN	0	Normal operating condition	R/W	0
		1	Divisor latch enable. Allows access to DLL, DLH, and other registers (refer to the registers mapping).		
6	BREAK_EN		Break control bit.	R/W	0
		0	Normal operating condition		
		1	Forces the transmitter output to go low to alert the communication terminal		
5	PARITY_TYPE2		Selects the forced parity format (if LCR3 = 1).	R/W	0
			If LCR5 = 1 and LCR4 = 0, the parity bit is forced to 1 in the transmitted and received data.		
			If LCR5 = 1 and LCR4 = 1, the parity bit is forced to 0 in the transmitted and received data.		
4	PARITY_TYPE1	0	Odd parity is generated (if bit 3 = 1).	R/W	0
		1	Even parity is generated (if bit 3 = 1).		
3	PARITY_EN	0	No parity	R/W	0
		1	Parity is generated during transmission, and the receiver checks for received parity.		
2	NB_STOP		Specifies the number of stop bits:	R/W	0
		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)		

Note: As soon as LCR[6] is set to 1, the RX line is forced to 0 and remains in this state as long as LCR[6] = 1.

Table 12–49. Line Control Register (LCR) (Continued)

Bit	Name	Value	Function	R/W	Reset Value
1:0	CHAR_LENGTH		Specify the word length to be transmitted or received.	R/W	00
		00	5 bits		
		01	6 bits		
		10	7 bits		
		11	8 bits		

Note: As soon as LCR[6] is set to 1, the RX line is forced to 0 and remains in this state as long as LCR[6] = 1.

Table 12–50. UART Mode Line Status Register (UART_LSR)

Bit	Name	Value	Function	R/W	Reset Value
7	RX_FIFO_STS	0	Normal operation.	R	1
		1	At least one parity error, framing error, or break indication in the receiver FIFO. Bit is cleared when no more errors are present in FIFO.		
6	TX_SR_E	0	Transmitter hold and shift registers are not empty.	R	1
		1	Transmitter hold and shift registers are empty.		
5	TX_FIFO_E	0	Transmit hold register is not empty	R	1
		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.		
4	RX_BI	0	No break condition	R	0
		1	A break was detected while the data being read from the RX FIFO was being received (RX input was low for one character time frame).		

Table 12–50. UART Mode Line Status Register (UART_LSR) (Continued)

Bit	Name	Value	Function	R/W	Reset Value
3	RX_FE	0	No framing error in data being read from RX FIFO	R	0
		1	Framing error occurred in data being read from RX FIFO (received data did not have a valid stop bit).		
2	RX_PE	0	No parity error in data being read from RX FIFO	R	0
		1	Parity error in data being read from RX FIFO		
1	RX_OE	0	No overrun error	R	0
		1	Overrun error has occurred. Set when the character held in the receive shift register is not transferred to the RX FIFO. This case can occur only when the receive FIFO is full.		
0	RX_FIFO_E	0	No data in the receive FIFO	R	0
		1	At least one data character in the RX_FIFO		

When the line status register (LSR) is read, LSR[4:2] reflect 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.

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.

Table 12–51. SIR Mode Line Status Register (SIR_LSR)

Bit	Name	Value	Function	R/W	Reset Value
7	THR_EMPTY	0	Transmit hold register is not empty.	R	1
		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.		
6	STS_FIFO_FULL	0	Status FIFO not full	R	0
		1	Status FIFO full		
5	RX_LAST_BYTE	0	Did not receive last byte of a frame from the FIFO	R	0
		1	Received last byte from FIFO. This bit is set when the last byte of a frame is read. Used to determine the frame boundary. Cleared by first reading the last received byte, then reading the SIR_LSR register.		
4	FRAME_TOO_LONG	0	No frame-too-long error in frame	R	0
		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.		
3	ABORT	0	No abort pattern error in frame	R	0
		1	Abort pattern received		
2	CRC	0	No CRC error in frame	R	0
		1	CRC error in the frame at the top of the STATUS FIFO (next character to be read)		
1	STS_FIFO_E	0	Status FIFO not empty	R	1
		1	Status FIFO empty		

Table 12–51. SIR Mode Line Status Register (SIR_LSR) (Continued)

Bit	Name	Value	Function	R/W	Reset Value
0	RX_FIFO_E	0	At least one data character in the RX_FIFO	R	1
		1	No data in the receive FIFO		

When the LSR is read, LSR[4:2] reflect the error bits [FL, CRC, ABORT] of the frame at the top of the STATUS FIFO (next frame status to be read). In SIR mode, the LSR bits [4:2] reflect the same values as the SFLSR bits [3:1].

Table 12–52. Supplementary Status Register (SSR)

Bit	Name	Value	Function	R/W	Reset Value
7–2	–		Reserved	R	000000
1	RX_CTS_DSR_WAKE_UP_STS	0	No falling edge event on RX, $\overline{\text{CTS}}$ and $\overline{\text{DSR}}$	R	0
		1	A falling edge occurred on RX, $\overline{\text{CTS}}$ or $\overline{\text{DSR}}$.		
0	TX_FIFO_FULL	0	TX FIFO not full	R	0
		1	TX FIFO full		

Note: Bit 1 is reset only when SCR[4] is reset to 0.

The modem control register (MCR) [3:0] controls the interface with the modem, data set, or peripheral device that is emulating the modem.

Table 12–53. Modem Control Register (MCR)

Bit	Name	Value	Function	R/W	Reset Value
7	CLKSEL	0	No action	R/W	0
		1	Divide clock input by 4.		
6	TCR_TLR	0	No action	R/W	0
		1	Enables access to the TCR and TLR registers		
5	XON_EN	0	Disable XON Any function	R/W	0
		1	Enable XON Any function		
4	LOOPBACK_EN	0	Normal operating mode	R/W	0
		1	Enable local loopback mode (internal). In this mode the MCR3:0 signals are looped back into MSR7:4. The transmit output is looped back to the receive input internally.		
3	CD_STS_CH	0	In loopback mode, forces IRQ outputs to inactive state	R/W	0
		1	In loopback mode, forces IRQ outputs to inactive state		
2	RESERVED		Reserved. This bit should always be written as 0.	R/W	0
1	RTS	0	Forces $\overline{\text{RTS}}$ output to inactive (high)	R/W	0
		1	Forces $\overline{\text{RTS}}$ output to active (low) In loopback mode controls MSR[4] If automatic RTS is enabled, the $\overline{\text{RTS}}$ output is controlled by hardware flow control.		
0	DTR	0	Forces $\overline{\text{DTR}}$ output to inactive (high)	R/W	0
		1	Forces $\overline{\text{DTR}}$ output to active (low)		

Note: Bits 5, 6, and 7 can be written only when EFR[4] = 1.

The modem status register (MSR) provides information about the current state of the control lines from the modem, data set, or peripheral device to the host (MPU or DSP). It also indicates when a control input from the modem changes state.

Table 12–54. Modem Status Register (MSR)

Bit	Name	Function	R/W	Reset Value
7–6	RESERVED	Reserved	R	
5	NDSR_STS	This bit is the complement of the $\overline{\text{DSR}}$ input. In loopback mode, it is equivalent to MCR0.	R	Input signal
4	NCTS_STS	This bit is the complement of the $\overline{\text{CTS}}$ input. In loopback mode, it is equivalent to MCR1.	R	Input signal
3–2	RESERVED	Reserved	R	0
1	DSR_STS	1: Indicates that $\overline{\text{DSR}}$ input (or MCR0 in loopback) has changed state. Cleared on a read.	R	0
0	CTS_STS	1: Indicates that $\overline{\text{CTS}}$ input (or MCR1 in loopback) has changed state. Cleared on a read.	R	0

The interrupt enable register (IER) in UART mode can be programmed to enable/disable any of the following interrupts:

- Receiver error
- RHR
- THR
- XOFF
- $\overline{\text{CTS}}$/$\overline{\text{RTS}}$ change of state from low to high

Each of these interrupts can be enabled/disabled individually. There is also a sleep mode enable bit.

Table 12–55. UART Mode Interrupt Enable Register (UART_IER)

Bit	Name	Value	Function	R/W	Reset Value
7	CTS_IT	0	Disables the $\overline{\text{CTS}}$ interrupt	R/W	0
		1	Enables the $\overline{\text{CTS}}$ interrupt		
6	RTS_IT	0	Disables the $\overline{\text{RTS}}$ interrupt	R/W	0
		1	Enables the $\overline{\text{RTS}}$ interrupt		
5	XOFF_IT	0	Disables the XOFF interrupt	R/W	0
		1	Enables the XOFF interrupt		
4	SLEEP_MODE	0	Disables sleep mode	R/W	0
		1	Enables sleep mode (stop baud rate clock when the module is inactive)		
3	MODEM_STS_IT	0	Disables the modem status register interrupt	R/W	0
		1	Enables the modem status register interrupt		
2	LINE_STS_IT	0	Disables the receiver line status interrupt	R/W	0
		1	Enables the receiver line status interrupt		
1	THR_IT	0	Disables the THR interrupt	R/W	0
		1	Enables the THR interrupt		
0	RHR_IT	0	Disables the RHR interrupt and time out interrupt	R/W	0
		1	Enables the RHR interrupt and time out interrupt		

Note: Bits 4, 5, 6, and 7 can only be written when EFR[4] = 1.

The interrupt enable register (IER) in SIR mode can be programmed to enable/disable any of the following interrupts:

- ❑ Received error
- ❑ LSR
- ❑ TX underrun
- ❑ Status FIFO
- ❑ RX overrun
- ❑ Last byte in RX FIFO
- ❑ THR
- ❑ RHR

Each of these interrupts can be enabled/disabled individually. There is also a sleep mode enable bit.

Table 12–56. SIR Mode Interrupt Enable Register (SIR_IER)

Bit	Name	Value	Function	R/W	Reset Value
7	EOF_IT	0	Disables the received EOF interrupt	R/W	0
		1	Enables the received EOF interrupt		
6	LINE_STS_IT	0	Disables the receiver line status interrupt	R/W	0
		1	Enables the receiver line status interrupt		
5	TX_UNDERRUN_IT	0	Disables the TX underrun interrupt	R/W	0
		1	Enables the TX underrun interrupt		
4	STS_FIFO_TRIG_IT	0	Disables status FIFO trigger level interrupt	R/W	0
		1	Enables status FIFO trigger level interrupt		
3	RX_OVERRUN_IT	0	Disables the RX overrun interrupt	R/W	0
		1	Enables the RX overrun interrupt		
2	LAST_RX_BYTE_IT	0	Disables the last byte of frame in RX FIFO interrupt	R/W	0
		1	Enables the last byte of frame in RX FIFO interrupt		
1	THR_IT	0	Disables the THR interrupt	R/W	0
		1	Enables the THR interrupt		
0	RHR_IT	0	Disables the RHR interrupt	R/W	0
		1	Enables the RHR interrupt		

The interrupt identification register (IIR) is a read-only register, which provides the source of the interrupt in a prioritized manner.

Table 12–57. UART Mode Interrupt Identification Register (UART_IIR)

Bit	Name	Value	Function							R/W	Reset Value
7–6	FCR_MIRROR		Mirror the contents of FCR(0) on both bits.							R	00
5–1	IT_TYPE		**Priority**	**5**	**4**	**3**	**2**	**1**	**Source**	R	00000
			1	0	0	0	1	1	Receiver line status error		
			2	0	0	1	1	0	RX time-out		
			2	0	0	0	1	0	RHR interrupt		
			3	0	0	0	0	1	THR interrupt		
			4	0	0	0	0	0	Modem interrupt		
			5	0	1	0	0	0	XOFF/special character		
			6	1	0	0	0	0	CTS, RTS, DSR change state from active (low) to inactive (high)		
0	IT_PENDING	0	An interrupt is pending (nIRQ active).							R	1
		1	No interrupt is pending (nIRQ inactive).								

The $\overline{IRQ}$ output is activated whenever one of the 8 interrupts is active.

Table 12–58. SIR Mode Interrupt Identification Register (SIR_IIR)

Bit	Name	Value	Function	R/W	Reset Value
7	EOF_IT	0	Received EOF interrupt inactive	R	0
		1	Received EOF interrupt active		
6	LINE_STS_IT	0	Receiver line status interrupt inactive	R	0
		1	Receiver line status interrupt active		
5	TX_UE_IT	0	TX underrun interrupt inactive	R	0
		1	TX underrun interrupt active		
4	STS_FIFO_IT	0	Status FIFO trigger level interrupt inactive	R	0
		1	Status FIFO trigger level interrupt active		
3	RX_OE_IT	0	RX overrun interrupt inactive	R	0
		1	RX overrun interrupt active		
2	RX_FIFO_LAST_ BYTE_IT	0	Last byte of frame in RX FIFO interrupt inactive	R	0
		1	Last byte of frame in RX FIFO interrupt active		
1	THR_IT	0	THR interrupt inactive	R	0
		1	THR interrupt active		
0	RHR_IT	0	RHR interrupt inactive	R	0
		1	RHR interrupt active		

The enhanced feature register (EFR) enables or disables enhanced features, most of which only apply to UART mode. But EFR[4] enables write accesses to FCR[5:4], the TX trigger level, which is also used in SIR mode.

Table 12–59. Enhanced Feature Register (EFR)

Bit	Name	Value	Function	R/W	Reset Value
7	AUTO_CTS_EN		Automatic CTS enable bit	R/W	0
		0	Normal operation		
		1	Automatic CTS flow control is enabled; that is, transmission is halted when the $\overline{\text{CTS}}$ pin is high (inactive).		
6	AUTO_RTS_EN		Automatic RTS enable bit	R/W	0
		0	Normal operation		
		1	Automatic $\overline{\text{RTS}}$ flow control is enabled; that is, the $\overline{\text{RTS}}$ pin goes high (inactive) when the receiver FIFO HALT trigger level, TCR3:0, is reached, and goes low (active) when the receiver FIFO restore transmission trigger level is reached.		
5	SPECIAL_CHAR_ DETECT	0	Normal operation	R/W	0
		1	Special character detect enable bit. 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 a special character has been detected.		
4	ENHANCED_EN		Enhanced functions write enable bit.	R/W	0
		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		
3–0	SW_FLOW_CONTROL		Combinations of software flow control can be selected by programming bits 3:0. See Table 12–60.	R/W	0

Table 12–60. EFR[0:3]: Software Flow Control Options

Bit 3	Bit 2	Bit 1	Bit 0	TX, RX Software Flow Controls
0	0	X	X	No transmit flow control
1	0	X	X	Transmit XON1, XOFF1
0	1	X	X	Transmit XON2, XOFF2
1	1	X	X	Transmit XON1, XON2: XOFF1, XOFF2†
X	X	0	0	No receive flow control
X	X	1	0	Receiver compares XON1, XOFF1
X	X	0	1	Receiver compares XON2, XOFF2
X	X	1	1	Receiver compares XON1, XON2: XOFF1, XOFF2†

† In these cases, the XON1 and XON2 characters or the XOFF1 and XOFF2 characters must be transmitted/received sequentially with XON1/XOFF1 followed by XON2/XOFF2.

XON1 and XON2 must be set to different values if the software flow control is enabled.

Table 12–61. XON1/Address Register 1 (XON1/ADDR1)

Bit	Name	Function	R/W	Reset Value
7–0	XON_WORD1	Used to store the 8-bit XON1 character in UART mode and ADDR1 address 1 for SIR mode.	R/W	0x00

Table 12–62. XON2/Address Register 2 (XON2/ADDR2)

Bit	Name	Function	R/W	Reset Value
7–0	XON_WORD2	Used to store the 8-bit XON2 character in UART mode and ADDR2 address 2 for SIR mode.	R/W	0x00

Table 12–63. XOFF1 Register (XOFF1)

Bit	Name	Function	R/W	Reset Value
7–0	XOFF_WORD1	Used to store the 8-bit XOFF1 character in used in UART modes.	R/W	0x00

Table 12–64. XOFF2 Register (XOFF2)

Bit	Name	Function	R/W	Reset Value
7–0	XOFF_WORD2	Used to store the 8-bit XOFF2 character in used in UART mode.	R/W	0x00

The scratchpad register (SPR) does not control the module in anyway. It is a scratchpad register to be used by the programmer to hold temporary data.

Table 12–65. Scratchpad Register (SPR)

Bit	Name	Function	R/W	Reset Value
7–0	SPR_WORD	Scratchpad register	R/W	0x00

The two divisor latch registers (DLL and DLH) store the 16-bit divisor for generation of the baud clock in the baud rate generator. DLL stores the least significant part of the divisor. DLH stores the most 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).

Table 12–66. Divisor Latch Low Register (DLL)

Bit	Name	Function	R/W	Reset Value
7–0	CLOCK_LSB	Used to store the 8-bit LSB divisor value	R/W	0x00

Table 12–67. Divisor Latch High Register (DLH)

Bit	Name	Function	R/W	Reset Value
7–0	CLOCK_MSB	Used to store the 8-bit MSB divisor value	R/W	0x00

To achieve the required baud rate, you must program DLL/DLH with the integer part of the divisor value.

Choosing the appropriate divisor value:

UART: Divisor value = Operating Frequency/(16 x baud rate).

The input frequency of the UART IrDA must be fixed to the operating frequency (here 12 MHz; no CLKSEL bit setting), and the the OSC_12M_SEL bit must be set to be able to reach the desired baud rate. Setting OSC_12M_SEL to 1 enables turning on the 6.5 division factor. For instance, 12 MHz/16/6.5 = 115200 bps; if OSC_12M_SEL is not set, the reached baud rate is either 12 MHz/16/6 or 12 MHz/16/7, which are outside permitted tolerance.

The transmission control register (TCR) stores the receive FIFO threshold levels to start/stop transmission during hardware/software flow control.

Table 12–68. Transmission Control Register (TCR)

Bit	Name	Function	R/W	Reset Value
7–4	RX_FIFO_TRIG_START	RCV FIFO trigger level to restore transmission (0-60)	R/W	0000
3–0	RX_FIFO_TRIG_HALT	RCV FIFO trigger level to halt transmission (0–60)	R/W	1111

Note: Trigger levels from 0–60 bytes are available with a granularity of four (trigger level = 4 x [4-bit register value]).

The programmer must ensure that TCR[3:0] is greater than TCR[7:4] whenever automatic RTS or software flow control is enabled to avoid spurious operation of the device.

In FIFO interrupt mode with flow control, the programmer must also 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, FIFO operation stalls. In FIFO DMA mode with flow control, this concept does not exist because a DMA request is sent each time a byte is received.

The trigger level register (TLR) stores the programmable transmit and receive FIFO trigger levels used for DMA and IRQ generation.

Table 12–69. Trigger Level Register (TLR)

Bit	Name	Function	R/W	Reset Value
7–4	RX_FIFO_TRIG_DMA	RCV FIFO trigger level	R/W	0000
3–0	TX_FIFO_TRIG_DMA	Transmit FIFO trigger level	R/W	0000

Table 12–70 and Table 12–71 summarize the different ways that can be used to set the trigger levels for the transmit FIFO and the receive FIFO, respectively.

Table 12–70. Transmit FIFO Trigger Level Setting Summary

SCR[6]	TLR[3:0]	TX FIFO Trigger Level
0	0000	Defined by FCR5:4 (either 8, 16, 32, and 56 spaces)
0	≠0000	Defined by TLR3:0 (from 4 to 60 spaces with a granularity of 4 spaces)
1	value	Defined by the concatenated value of TLR3:0 and FCR 5:4 (from 1 to 63 spaces with a granularity of 1 space). The combination of TLR3:0 = 0000 and FCR 5:4 = 00 (all zeros) is not supported (minimum one space required). All zeros result in unpredictable behavior.

Note: The protocol to set the concatenation of TLR and FCR is:
- Set SCR[6] = 0
- Set the value of threshold into FCR and TLR
- Set SCR[6] = 1

Table 12–71. Receive FIFO Trigger Level Setting Summary

SCR[7]	TLR[7:4]	TX FIFO Trigger Level
0	0000	Defined by FCR7:6 (either 8, 16, 56, and 60 characters)
0	≠0000	Defined by TLR7:4 (from 4 to 60 characters with a granularity of 4 characters)
1	value	Defined by the concatenated value of TLR7:4 and FCR 7:6 (from 1 to 63 characters with a granularity of 1 character). The combination of TLR7:4 = 0000 and FCR 7:6 = 00 (all zeros) is not supported (minimum one character required). All zeros result in unpredictable behavior.

Note: The protocol to set the concatenation of TLR and FCR is:
- Set SCR[7] = 0
- Set the value of threshold into FCR and TLR
- Set SCR[7] = 1

The mode of operation can be programmed by writing to MDR1[2:0]; therefore the mode definition 1 register (MDR1) must be programmed on start-up after configuration of the configuration registers (DLL, DLH, LCR…). The value of MDR1[2:0] must not be changed again during normal operation.

Table 12–72. Mode Definition 1 Register (MDR1)

Bit	Name	Value	Function	R/W	Reset Value
7	FRAME_END_MODE	0	Frame-length method	R/W	0
		1	Set EOT bit method		
6	–		Reserved	R/W	0
5	SCT		Stores and controls the transmission	R/W	0
		0	Starts the SIR transmission as soon as a value is written to THR		
		1	Starts the SIR transmission with the control of ACREG2		
4	–		Reserved	R	0
3	IR_SLEEP	0	SIR sleep mode disabled	R/W	0
		1	SIR sleep mode enabled		
2–0	MODE_SELECT†	000	UART mode	R/W	111
		001	SIR mode		
		111	Disable UART/default state		
			All the other values are reserved		

† The MODE_SELECT = 0x7 setting disables the UART module by disabling the FIFO and the state machine. It does not gate the functional clock to the module. The lowest power state is not achieved by setting MODE_SELECT = 0x7, but by putting the UART into sleep mode. The lowest power state is achieved when in sleep mode witht DLL = 0xFFFF and DLH = 0xFFFF.

The mode definition 2 register (MDR2) sets the trigger level for the frame status FIFO (8 entries) and must be programmed before the mode is programmed in MDR1[2:0].

Table 12–73. Mode Definition Register 2 (MDR2)

Bit	Name	Value	Function	R/W	Reset Value
7–5	–		Reserved	R/W	000
4–3	DIV_1.6M		MSB part of DIV_1.6	R/W	00
2–1	STS_FIFO_TRIG		Frame status FIFO threshold select:	R/W	00
		00	1 entry		
		01	4 entry		
		10	7 entry		
		11	8 entry		
0	–		Reserved	R/W	0

The transmit frame length registers (TXFLL and TXFLH) hold the 13-bit transmit frame length. 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.

In terms of the IrDA frame format (see Figure 12–14), the value stored in the TXFLH/TXFLL registers is the byte length from A to I.

Table 12–74. Transmit Frame Length Low Register (TXFLL)

Bit	Name	Function	R/W	Reset Value
7–0	TXFLL	LSB register used to specify the frame length	W	00000000

Table 12–75. Transmit Frame Length High Register (TXFLH)

Bit	Name	Function	R/W	Reset Value
7–5	–	Reserved	W	000
4–0	TXFLH	MSB register used to specify the frame length	W	00000

The received frame length 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, then program RXFLL and RXFLH to n + 3 in SIR mode (+3 is due to frame format with CRC and stop flag).

In terms of the IrDA frame format (see Figure 12–14), the value stored in the RXFLH/RXFLL registers is the byte length from A to EOF.

Table 12–76. Received Frame Length Low Register (RXFLL)

Bit	Name	Function	R/W	Reset Value
7–0	RXFLL	LSB register used to specify the frame length in reception	W	00000000

Offset Address (hex): 0x0D x Start Address

Table 12–77. Received Frame Length High Register (RXFLH)

Bit	Name	Function	R/W	Reset Value
7–4	–	Reserved	W	0000
3–0	RXFLH	MSB register used to specify the frame length in reception	W	0000

The status FIFO line status line register (SFLSR) 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 12–78. Status FIFO Line Status Register (SFLSR)

Bit	Name	Function	R/W	Reset Value
7–5	–	Reserved	R	000
4	OE_ERROR	1: Overrun error in RX FIFO when frame at top of FIFO was received	R	0
3	FRAME_LENGTH_ ERROR	1: Frame-length error in frame at top of FIFO	R	0
2	ABORT_DETECT	1: Abort pattern detected in frame at top of FIFO	R	0
1	CRC_ERROR	1: CRC error in frame at top of FIFO	R	0
0	–	Reserved	R	0

The resume register (RESUME) is used to clear internal flags, which halt transmission/reception when an underrun/overrun error occurs. Reading this register resumes the halted operation. This register does not physically exist and reads always as 0x00.

Table 12–79. Resume Register (RESUME)

Bit	Name	Function	R/W	Reset Value
7–0	DI	Dummy read to restart the TX or RX	R	00000000

The frame lengths of received frames are written into the status FIFO. This information can be read by reading the status FIFO registers (SFREGL and SFREGH—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.

In terms of the IrDA frame format (see Figure 12–14), the value read in the SFREGH/SFREGL registers is the byte length from A to CRC.

Table 12–80. Status FIFO Register Low (SFREGL)

Bit	Name	Function	R/W	Reset Value
7–0	SFREGL	LSB part of the frame length	R	Undefined

Table 12–81. Status FIFO Register High (SFREGH)

Bit	Name	Function	R/W	Reset Value
7–4	–	Reserved	R	0000
3–0	SFREGH	MSB part of the frame length	R	Undefined

The beginning of frame control register (BLR) [6] selects whether 0xC0 or 0xFF start patterns are to be used and when multiple start flags are required in SIR mode. If only one start flag is required, this is always 0xC0. If n start flags are required, then either (n-1) C0x0 or (n-1) 0xFF flags are sent, followed by a single 0xC0 flag (immediately preceding the first data byte).

Table 12–82. BOF Control Register (BLR)

Bit	Name	Value	Function	R/W	Reset Value
7	STS_FIFO_RESET		Status FIFO reset. This bit is self-clearing	R/W	0
6	XBOF_TYPE		SIR XBOF select	R/W	1
		0	0xFF		
		1	0xC0		
5–0	–		Reserved	R/W	000000

The beginning of frame length register (EBLR) specifies the number of BOF + XBOFs to transmit in IrDA SIR operations. Value set into this register must take into account the BOF character; to send sent 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.

Table 12–83. BOF Length Register (EBLR)

Bit	Name	Function	R/W	Reset Value
7–0	EBLR	This register allows definition of up to 176 XBOFs, the maximum required by IrDA specification.	W	00000000

Table 12–84. DIV1.6 Register (DIV16)

Bit	Name	Function	R/W	Reset Value
7–0	DIV_1.6L	Used to generate the 1.6-µs pulse	R/W	00000000

In SIR, the DIV1.6 register (DIV16) is used to generate 1.6-µs pulse encoding instead of 3/16 encoding when selected using ACREG[7]. The value of DIV_1.6 is coded on ten bits by MDR2[4:3] for its MSB and DIV_1.6[7:0] for its MSB.

In SIR mode, DIV1.6 must be programmed as follows:

DIV1.6 = [(3/(16 * baud rate)) - 1.6E-6] * FCLK_frequency

DIV1.6 = 0 is forbidden. If the calculated value Div_1.6 is between 0 and 1 the rounding must be done to 1.

With an input frequency of 13 MHz:

At 115200 bauds	DLH = 0x00	DLL = 0x07	MDR24:3 = 0x00	DIV_1.6 = 0x01
At 57600 bauds	DLH = 0x00	DLL = 0x0E	MDR24:3 = 0x00	DIV_1.6 = 0x16
At 38400 bauds	DLH = 0x00	DLL = 0x15	MDR24:3 = 0x00	DIV_1.6 = 0x2B
At 19200 bauds	DLH = 0x00	DLL = 0x2A	MDR24:3 = 0x00	DIV_1.6 = 0x6A
At 9600 bauds	DLH = 0x00	DLL = 0x55	MDR24:3 = 0x00	DIV_1.6 = 0xEB
At 2400 bauds	DLH = 0x01	DLL = 0x53	MDR24:3 = 0x03	DIV_1.6 = 0x96E5

With an input frequency of 12 MHz:

At 115200 bauds	DLH = 0x00	DLL = 0x01	MDR24:3 = 0x00	DIV_1.6 = 0x01
At 57600 bauds	DLH = 0x00	DLL = 0x02	MDR24:3 = 0x00	DIV_1.6 = 0x14
At 38400 bauds	DLH = 0x00	DLL = 0x03	MDR24:3 = 0x00	DIV_1.6 = 0x27
At 19200 bauds	DLH = 0x00	DLL = 0x06	MDR24:3 = 0x00	DIV_1.6 = 0x62
At 9600 bauds	DLH = 0x00	DLL = 0x0C	MDR24:3 = 0x00	DIV_1.6 = 0xD7
At 2400 bauds	DLH = 0x01	DLL = 0x30	MDR24:3 = 0x03	DIV_1.6 = 0x96

Table 12–85. Auxiliary Control Register (ACREG)

Bit	Name	Value	Function	R/W	Reset Value
7	PULSE_TYPE		SIR pulse-width select:	R/W	0
		0	3/16 of baud-rate pulse width		
		1	1.6-µs		
6	SD_MOD		Primary output used to configure transceivers. Connected to the SD/MODE input of transceivers.	R/W	0
		0	SD_MODE pin is set to high.		
		1	SD_MODE pin is set to low.		
5	DIS_IR_RX	0	Enables RXIR input	R/W	0
		1	Disables RXIR input for half-duplex purpose		
4	DIS_TX_UNDERRUN	0	Long stop bits cannot be transmitted, TX underrun is enabled.	R/W	0
		1	Long stop bits can be transmitted, TX underrun is disabled.		
3	–		Reserved	R	0
2	SCTX_EN		Store and controlled TX start. When MDR15 = 1 and the host writes 1 to this bit, the TX state machine starts frame transmission. This bit is self-clearing.	R/W	0

Table 12–85. Auxiliary Control Register (ACREG) (Continued)

Bit	Name	Value	Function	R/W	Reset Value
1	ABORT_EN		Frame abort. The host 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 host 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 automatically gets cleared when the host writes to the THR (TX FIFO).	R/W	0

Table 12–86. OSC 12-MHz Select Register (OSC_12M_SEL)

Bit	Name	Function	R/W	Reset Value
7–1	–	Reserved	R	0000000
0	OSC_12M_SEL†	When 1, selects 6.5 division factor with a 12-MHz system clock.	W	0

† This register is write-only and cannot be read.

Table 12–87. Module Version Register (MVR)

Bit	Name	Function	R/W	Reset Value
7–4	MAJOR_REV	Major revision number of the module	R	†
3–0	MINOR_REV	Minor revision number of the module	R	

† For example: MVR = 0x11: Version 1.1

12.8 UART/IrDA Modes of Operation

The UART/IrDA module can operate in two different modes: UART mode and slow infrared (SIR) mode.

The modules perform serial-to-parallel conversion on data characters received and parallel-to-serial conversion on data characters transmitted by the processor. The complete status of each channel of the modules and each received character/frame can be read at any time during functional operation via the line control register (LSR).

You can place the modules in an alternate mode (FIFO mode) to relieve the processor of excessive software overhead by buffering received/transmitted characters. Both the receiver and transmitter FIFOs can store up to 64 bytes of data (plus three 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 host (MPU or DSP) and the module.

12.8.1 UART Mode

The UART uses a wired interface for serial communication with a remote device.

UART modules are functionally compatible to the TL16C750 UART and are also functionally compatible with earlier designs such as the TL16C550.

UART modules can use hardware or software flow controls to manage transmission/reception. Hardware flow control significantly reduces software overhead and increases system efficiency by automatically controlling serial data flow using the $\overline{\text{RTS}}$ output and $\overline{\text{CTS}}$ input signals. Software flow control automatically controls data flow by using programmable XON/XOFF characters.

12.8.2 SIR Mode

In slow infrared (SIR) mode, data transfer takes place between the host (MPU or DSP) and peripheral devices at speeds of up to 115200 bauds. An SIR transmit frame starts with start flags (either a single C0h, multiple C0hs, or a single C0h preceded by a number of FFh flags), followed by frame data, CRC-16, and a stop flag (C1h).

BLR[6] selects whether C0h or FFh start patterns are to be used when multiple start flags are required.

The SIR transmit state machine attaches start flags, CRC-16, and stop flags. It checks the outgoing data to establish if data transparency is required.

SIR transparency is carried out if the outgoing data (between the start and stop flags) contains C0h, C1h, or 7Dh. If one of these is about to be transmitted, then the SIR state machine sends an escape character [7Dh] first, then inverts the fifth bit of the real data to be sent, and sends this data immediately after the 7Dh 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 aborts (7Dh character followed immediately by a C1h stop flag, without transparency), CRC errors, and frame-length errors. At the end of a frame reception, the host (MPU or DSP) reads the line status register (LSR) to find out the errors, if any, of the received frame.

Data can be transferred both ways simultaneously by the module, but transmit and receive must not take place at the same time according to the standard.

The infrared output in SIR mode can either be 1.6-µs or 3/16 encoding, selected by 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 two start flags at the start of each frame for back-to-back frames.

Note:

Reception supports variable-length stop bits.

12.8.2.1 CRC Generation

Figure 12–14 shows the IrDA frame format.

Figure 12–14. IrDA Frame Format

The CRC is applied on the address (A), control (C), and information (I) bytes.

Note:

The two words of CRC are written in the FIFO in reception.

12.8.2.2 Asynchronous Transparency

Before transmitting a byte, the UART IrDA controller examines each byte in the payload and the CRC field (between BOF and EOF). For each byte equal to 0xC0 (BOF), 0xC1 (EOF), 0x7D (control escape), the controller does the following.

- ❏ In transmission:

1) Inserts a control escape (CE) byte preceding the byte.

2) Complements bit 5 of the byte (that is, exclusive ORs the byte with 0x20).

The byte sent for the CRC computation is the initial byte written in the TX FIFO (before the XOR with 0x20).

- ❏ In reception:

For the A, C, I, CRC fields:

1) Compares the byte with CE byte; if not equal, sends it to the CRC detector and stores it in the RX FIFO.

2) If equal to CE, discards the CE byte.

3) Complements the bit 5 of the byte following the CE.

4) Send the complemented byte to the CRC detector and stores it in the RX FIFO.

12.8.2.3 Abort Sequence

The transmitter may decide to prematurely close a frame. The transmitting station aborts by sending the sequence 0x7dc1. The abort pattern closes the frame without a CRC field or an ending flag.

It is possible to abort a transmission frame by programming ACREG [1].

When this bit is set to 1, 7Dh and C1h are transmitted and the frame is not terminated with CRC or stop flags.

The receiver treats a frame as an aborted frame when a 7Dh character followed immediately by a C1h character has been received without transparency.

- ❏ When UART3 receives an abort sequence (0x7DC1), the abort bit (LSR[3]) is set to 1.

- ❏ When the UART3 FIFO is empty or the SFSLR register is read, the abort bit is cleared to 0 (If SFSLR register is read, abort actually reflects the status of the next frame).

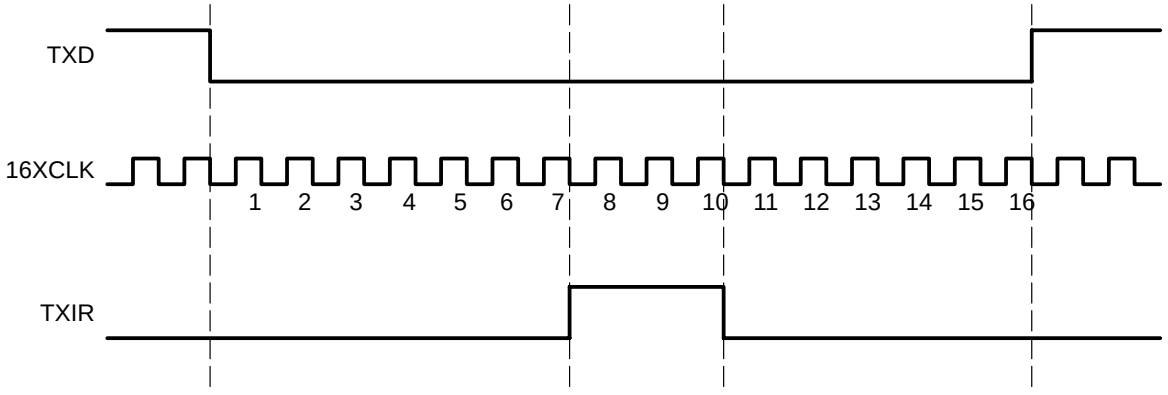
12.8.2.4 Pulse Shaping

In SIR mode both the 3/16th or the 1.6-μs pulse duration methods are supported. ACREG[7] selects the pulse-width method in transmit mode.

12.8.2.5 Encoder

Serial data from the transmit state machine is encoded to transmit data to the optoelectronics (see Figure 12–15). While the serial data input to (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 remains low, a pulse is transmitted during the 7th to the 10th clocks of each 16-clock bit cycle.

Figure 12–15. IrDA Encoder Mechanism

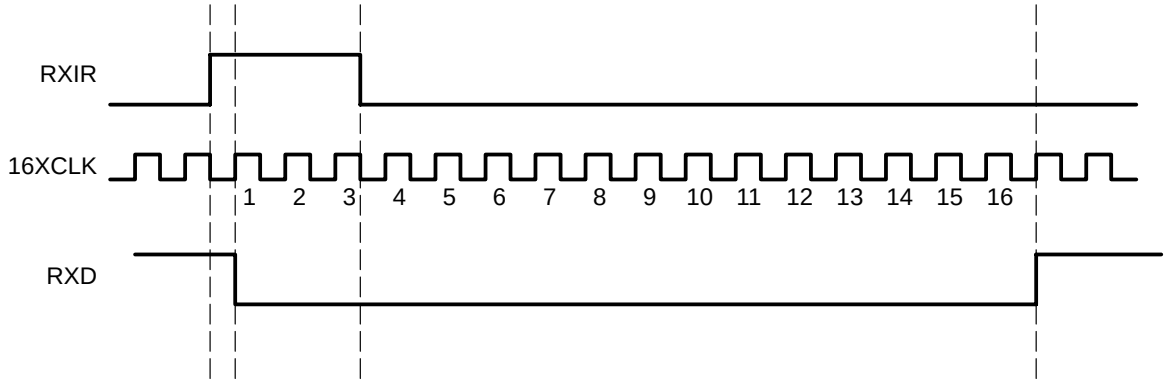


12.8.2.6 Decoder

After reset, RXD is high and the 4-bit counter is cleared (see Figure 12–16). When a rising edge is detected on RXIR, RXD falls on the next rising edge of 16XCLK with sufficient setup time. RXD remains 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.

The reception of RXIR input can be disabled with DIS_IR_RX bits of the auxiliary control register (ACREG[5]).

Figure 12–16. IrDA Decoder Mechanism



12.8.2.7 Address Checking

In SIR mode, only frames intended for the device are written to the RX FIFO, if address checking has been enabled. This avoids receiving frames not meant for this device in a multipoint infrared environment. You can program two frame addresses the UART IrDA receives with the XON1/ADDR1 and XON2/ADDR2 registers.

Selecting address1 checking is done by setting EFR[0] to 1. Address2 checking is done by setting EFR[1] to 1. Setting EFR[1:0] to 0 disables all address checking operations. If both bits are set, then the incoming frame is checked for both the private and public addresses.

If address checking is disabled, then all received frames are written into the reception FIFO.

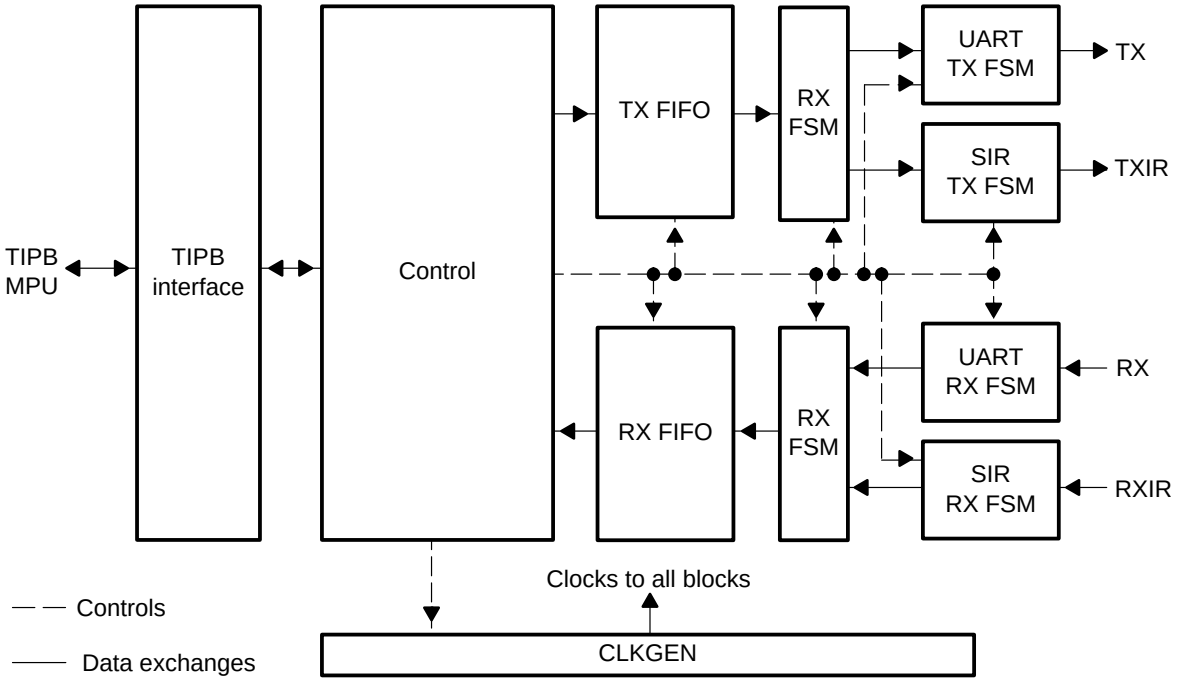
12.9 UART/IrDA Functional Description

This section provides a functional description of the UART IrDA.

12.9.1 UART/IrDA Functional Block Diagram

Figure 12–17 shows the UART/IrDA (FSM stands for finite state machine).

Figure 12–17. Functional Block Diagram



12.9.2 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 (in effect, the trigger level is the default value of one byte). The programmable trigger levels are an enhanced feature available via the trigger level register (TLR).

12.9.3 Interrupts

The UART generates interrupts on the UART_nIRQ output pin. All interrupts can be 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.

12.9.3.1 Interrupts in MODEM Mode

There are seven possible interrupts, prioritized to six different levels. When an interrupt is generated, the interrupt identification register (IIR) indicates a pending interrupt by bringing IIR[0] to logic 0, and it specifies the type of interrupt through IIR[5-1]. Table 12–88 summarizes the interrupt control functions.

Table 12–88. Generic Interrupt Functions in Modem Mode

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: All erroneous characters are read form the RX FIFO. 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	MSR1:0/ = 0	Read MSR
0 1 0 0 0 0	5	XOFF interrupt/ special character interrupt	Receive XOFF characters(s)/special character	Receive XON character(s), if XOFF interrupt/read of IIR, if special character interrupt
1 0 0 0 0 0	6	$\overline{\text{CTS}}$,$\overline{\text{RTS}}$, $\overline{\text{DSR}}$	$\overline{\text{RTS}}$ pin, $\overline{\text{CTS}}$ pin or $\overline{\text{DSR}}$ pin change state from active (low) to inactive (high).	Read IIR

Note: Once LSR[7] (RX_FIFO_STS) is set on FIFO disable (FCR[0]=0), this bit bit cannot be cleared by reading LSR. First, FCR[1] (RX_FIFO_CLERA) must be set to 1, then LSR[7] can be cleared.

The RX_FIFO_STS bit (LSR[7]) generates the interrupt for the receiver line status interrupt.

For the XOFF interrupt, if a XOFF flow character detection causes the interrupt, the interrupt is cleared by a XON flow character detection. If special character detection causes the interrupt, the interrupt is cleared by a read of the IIR.

12.9.3.2 Interrupts in SIR Mode

In the IrDA modes there are eight possible interrupts. The UART_nIRQ output is activated when any of the eight interrupts is generated (there is no priority).

Table 12–89 summarizes the interrupt control functions in SIR mode.

Table 12–89. Generic Interrupt Functions in SIR Mode

IIR Bit No.	Interrupt Type	Interrupt Source	Interrupt Reset Method
7	Received EOF	Received end-of-frame	Read IIR
6	Receiver line status interrupt	CRC, ABORT or frame-length error is written into STATUS FIFO.	Read STATUS FIFO. Read until empty—maximum eight reads required.
5	TX underrun	THR empty before EOF sent	Read RESUME register
4	Status FIFO interrupt	Status FIFO triggers level reached	Read STATUS FIFO.
3	RX overrun	Write to RHR when RX FIFO full.	Read RESUME register
2	Last byte in RX FIFO	Last byte of frame in RX FIFO	Read IIR
1	THR interrupt	TFE (THR empty) (FIFO disable) TX FIFO below trigger level (FIFO enable)	Write to THR until interrupt condition disappears.
0	RHR interrupt	DRDY (data ready) (FIFO disable) RX FIFO above trigger level (FIFO enable)	Read RHR until interrupt condition disappears.

12.9.3.3 Wake-Up Interrupt

Wake-up interrupt is a uniquely designed interrupt, enabled when 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 a wake-up interrupt occurs, the only way to clear it is to reset SCR[4] to 0.

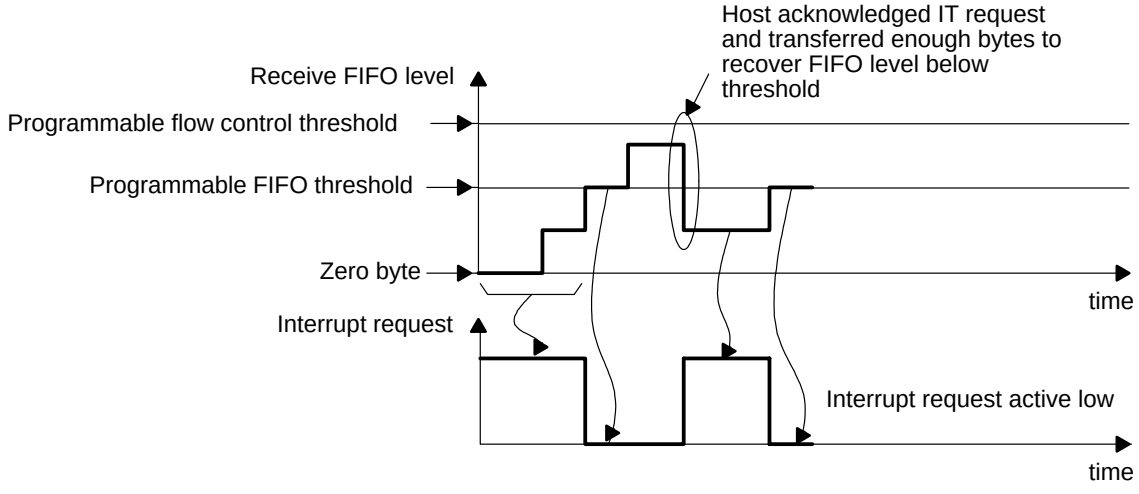
12.9.4 FIFO Interrupt Mode

In FIFO interrupt mode (FCR[0] = 1), relevant interrupts enabled via IER), the processor is informed of the status of the receiver and transmitter by an interrupt signal, $\overline{IRQ}$. These interrupts are raised when receive/transmit FIFO threshold (respectively TLR[7:4] and TLR[3:0] or FCR[7:6] and FCR[5:4]) are reached; they ask the host (MPU or DSP) to transfer data to destination (from UART module in receive mode and from any source to UART FIFO in transmit mode).

When UART flow control is enabled along with interrupt capabilities, you must ensure that the UART flow control FIFO threshold (TCR[3:0]) is greater than or equal to the receive FIFO threshold.

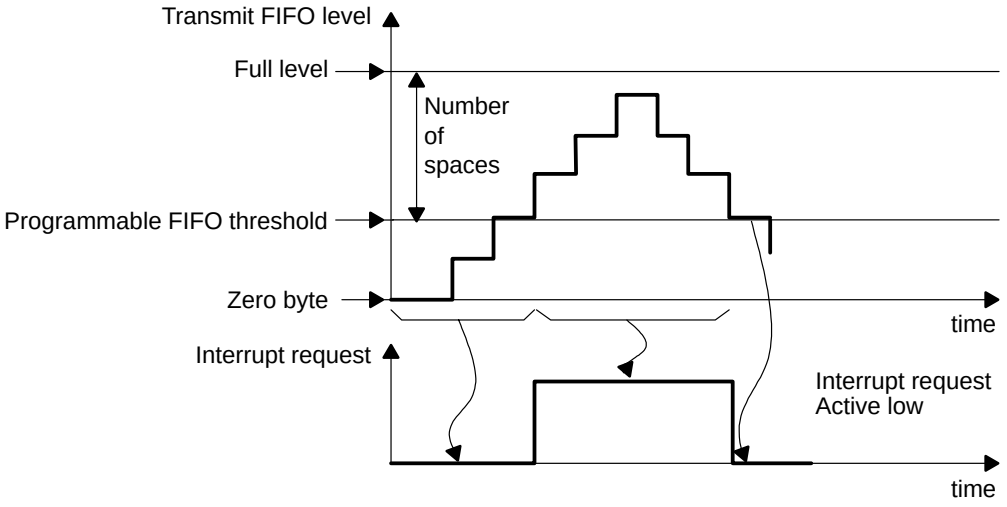
Figure 12–18 and Figure 12–19 show the receive and transmit IT operations, respectively.

Figure 12–18. 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 host (MPU or DSP) has handled enough bytes to make the FIFO level below threshold. Notice that the flow control threshold is set at a higher value than the FIFO threshold.

Figure 12–19. 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 crossed the threshold level. The interrupt line is deasserted until a sufficient number of elements has been transmitted to go below FIFO threshold.

12.9.5 FIFO Polled Mode Operation

In FIFO polled mode (FCR [0] = 0 with relevant interrupts disabled via interrupt enable register (IER)), the status of the receiver and transmitter can then be checked by polling the line status register (LSR). This mode is an alternative to the FIFO interrupt mode of operation, where the status of the receiver and transmitter is automatically known by means of interrupts sent to the host (MPU or DSP).

12.9.6 FIFO DMA Mode Operation

12.9.6.1 DMA Signaling

There are four modes of DMA operation: DMA mode 0, DMA mode 1, DMA mode 2, and DMA mode 3. They can be selected as follows.

- When SCR[0] = 0:
  - Setting FCR[3] to 0 enables DMA mode 0.
  - Setting FCR[3] to 1 enables DMA mode 1.
- When SCR[0] = 1:SCR[2:1] determine DMA mode 0 to 3 according to supplementary control register (SCR) description.

So for instance:

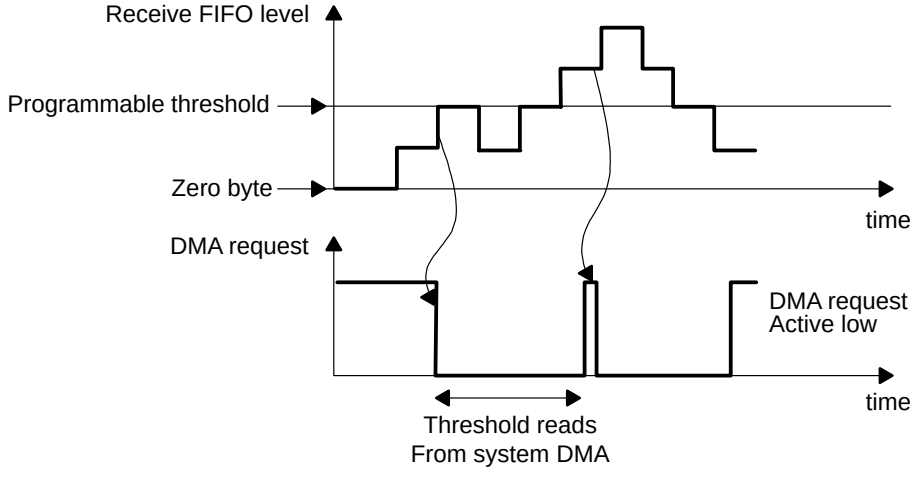
- If no DMA operation is desired, set SCR[0] to 1 and SCR[2:1] to 00 (FCR[3] is disregarded).
- If DMA mode 1 is desired, either set SCR[0] to 0 and FCR[3] to 1 or set SCR[0] to 1 SCR[2:1] to 01 (FCR[3] is disregarded).

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.

12.9.6.2 DMA Transfers (DMA Mode 1, 2, or 3)

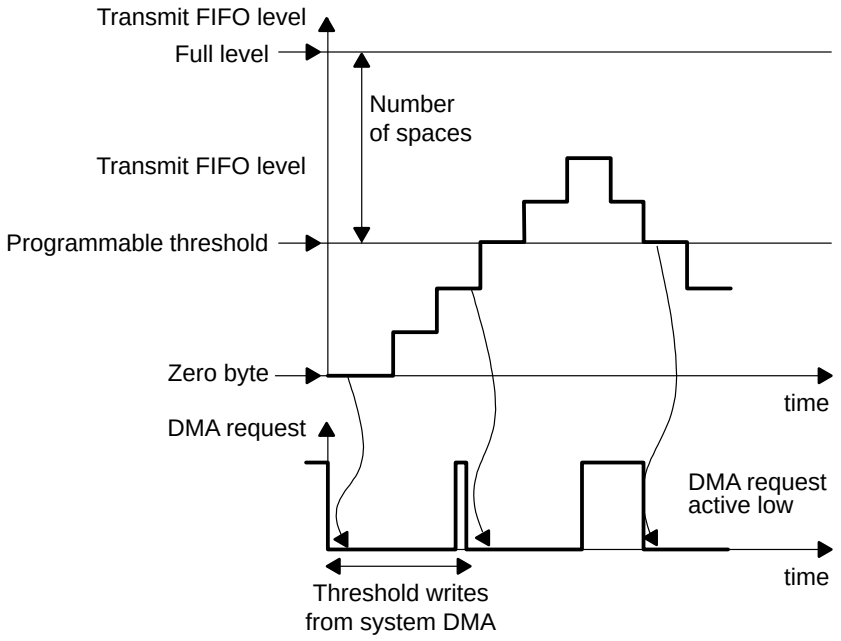
Figure 12–20 shows DMA operations at receive; Figure 12–21 shows DMA operations at transmit.

Figure 12–20. Receive FIFO DMA Request Generation



In receive mode, a DMA request is generated as soon as the receive FIFO reaches its threshold. This request is deasserted when the number of bytes defined by the threshold level has been read by the system DMA.

Figure 12–21. Transmit FIFO DMA Request Generation



In transmit mode, a DMA request is automatically asserted when FIFO is empty. This request is deasserted when the number of bytes defined by the threshold level has been written by the system DMA. The DMA request is again asserted if the FIFO is able to receive the number of bytes defined by the threshold.

12.9.7 Sleep Mode

12.9.7.1 UART Mode

Sleep mode is a low-power, enhanced feature of the UART that can be enabled by writing a 1 to IER[4] (when EFR[4] = 1).

Sleep mode is entered when:

- ❏ Serial RX data input line is idle.
- ❏ TX FIFO and TX shift register are empty.
- ❏ RX FIFO is empty.
- ❏ No interrupts are pending except THR interrupts.

Sleep mode is a good way to lower UART power consumption, but this state can be achieved only when the UART is set in modem mode. Therefore, even if UART does not have a 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, when data is written to the TX FIFO, or when there is any change in the state of the modem input pins. An interrupt can be generated on a wake up event by setting SCR[4] to 1.

Note:

Writing to the divisor latches, DLL and DLH, to set the baud clock, BCLK, must not be done during sleep mode. Disable sleep mode using IER[4] before writing to DLL or DLH.

12.9.7.2 IrDA Mode

In IrDA modes, sleep mode is enabled by writing a 1 to MDR1[3].

Sleep mode is entered when:

- ❏ Serial RXIR data input line is idle.
- ❏ TX FIFO and TX shift register are empty.
- ❏ RX FIFO is empty.
- ❏ No interrupts are 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.

12.9.8 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 4 X programmed word length + 12 bits. The receiver line is sampled midway through each bit.

  For sleep mode, the counter is reset when there is activity on the RX line.

  For the time-out interrupt, the counter only counts when there is data in the RX FIFO and the count is reset when there is activity on the RX line or when the RHR is read.

- Break condition

  When a break condition occurs, the TX line is pulled low. A break condition is activated by setting LCR[6]. 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 becomes empty (LSR[6] = 1).
  - Take a guard time according to stop bit definition.
  - Set LCR[6] to 1.

  The break condition is asserted as long as LCR[6] is set to 1.

12.9.9 Programmable Baud Rate Generator

The programmable baud generator takes any clock input and divides it by a divisor between 1 and (2^{16-1}). The CLKSEL register bit MCR[7] can be used to select the 1X or 1X/4 clock for the internal baud rate generator. The output frequency of the baud rate generator is 16x the baud rate.

You must write to the DLL register (least significant bytes) and DLH register (most significant bytes) of the baud rate divisor to program the baud rate.

Writing to these registers may result in wait states being inserted during the write access while the baud rate generator is loaded with the new value. If both registers are 0, the module is effectively disabled, and no baud clock is generated.

Note:

The programmable baud rate generator selects both the transmit and receive clock rates.

12.9.10 Hardware Flow Control

Hardware flow control is composed of automatic CTS and automatic RTS. Automatic CTS and automatic RTS can be enabled/disabled independently by programming EFR[7:6]. With automatic CTS, $\overline{\text{CTS}}$ must be active before the module can transmit data.

Automatic 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 automatic CTS and automatic 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 not enabled, overrun errors occur if the transmit data rate exceeds the receive FIFO latency.

- Automatic RTS

  Automatic RTS data flow control originates in the receiver block (see Figure 12–17). The receiver FIFO trigger levels used in automatic 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) can 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.

- Automatic 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 automatic CTS function reduces interrupts to the host system. When automatic 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 automatic CTS, the transmitter sends any data present in the transmit FIFO and a receiver overrun error can result.

12.9.11 Software Flow Control

Software flow control is enabled through the enhanced feature register (EFR) and the modem control register (MCR). Different combinations of software flow control can be enabled by setting different combinations of EFR[3-0].

There are two other enhanced features relating to software flow control:

- XON Any function [MCR(5)]: Operation resumes 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. The XOFF interrupt is cleared by a read of the IIR. The special character is transferred to the RX FIFO.

12.9.11.1 RX

When software flow control operation is enabled, the UART compares incoming data with XOFF1/2 programmed characters (in certain cases XOFF1 and XOFF2 must be received sequentially). When the correct XOFF characters are received, transmission is halted after completing transmission of the current character. XOFF detection also sets IIR(4) (if enabled via IER(5)) and causes 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, this character is treated as normal data and is written to the RX FIFO.

When XON-Any and special character detect are disabled and software flow control is enabled, no valid XON or XOFF characters are written to the RX FIFO. For example, when EFR[1:0] = 10, if XON1 and XOFF1 characters are received they do not get written to the RX FIFO.

When pairs of software flow characters are programmed to be received sequentially (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.

12.9.11.2 TX

XOFF1: Two characters are transmitted when the RX FIFO has passed the programmed trigger level TCR(3:0).

XON1: Two characters are transmitted when the RX FIFO reaches the trigger level programmed via TCR(7:4).

If, after an XOFF character has been sent, software flow control is disabled, the module transmits XON characters automatically to enable normal transmission to proceed.

The transmission of XOFF/XON(s) follows the exact 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, the 5, 6, or 7 least significant bits of XOFF1,2/XON1,2 are transmitted. 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 are never enabled simultaneously.

12.9.12 Frame Closing

There are two methods by which a transmission-frame can be properly terminated.

1) The frame-length method is selected when MDR1[7] = 0. The host (MPU or DSP) writes the frame-length value to 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.

2) The set-EOT bit method is selected when MDR1[7] = 1. The host writes 1 to ACREG[0] (EOT bit) just before it writes the last byte to the TX FIFO. When the 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.

12.9.13 Store and Controlled Transmission

In a store and controlled transmission (SCT), the host (MPU or DSP) 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, 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 is different from the normal mode, where transmission of data starts immediately after data is written to the TX FIFO. SCT is useful to send short frames without TX underrun.

12.9.14 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 (host) must:

- ❏ Reset the TX FIFO.
- ❏ Read the RESUME register—this clears the internal flag.

This functionality can be disabled or compensated for by the extension of the stop bit in transmission, in case the TX FIFO is empty.

12.9.15 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 host (MPU or DSP) 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 (host) must:

- ❏ Reset the RX FIFO.
- ❏ Read the resume register—this clears the internal flag.

12.9.16 Status FIFO

In SIR mode, a status FIFO is used to record the received frame status. When a complete frame is received, the length of the frame and the error bits associated with the frame are written into the status FIFO.

The frame length and error status can be read by reading SFREGL/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 host (MPU or DSP) uses the frame-length information to locate the frame-boundary in the received frame data. The host can screen bad frames using the error-status information and later request the sender to resend only the bad frames.

This status FIFO can be used very effectively in DMA as the host does not need to be interrupted every time a frame is received, but only whenever the programmed status FIFO trigger level is reached.

12.10 UART/IrDA Configuration Example

This section outlines the programming stages to operate one UART module with FIFO, interrupt, and no DMA capabilities. This is a three-step procedure that ensures quick start of these modules (obviously it does not cover every UART module feature). The first stage covers software reset of the module (interrupts, status, and controls); the second stage deals with FIFO configuration and enable; and the last stage deals with baud rate data and stop configuration. The procedure below is programming language agnostic.

12.11 UART Software Reset

The goal of the UART software reset is to clear IER and MCR registers, remove UART breaks (LCR[6]=0), and put module in reset (MDR1[2:0]=0x3).

The procedure of the UART is as follows:

1) Write into both the IER and MCR register (EFR[4] must first be set to 1).

2) Access the EFR register.

3) 0xBF must first be written to LCR register as follows:

   - LCR=0xBF
   - EFR[4]=1
   - LCR=0x80 (access to IER and MCR is allowed)
   - IER=0x00
   - MCR=0x00; LCR[6]=0 (UART breaks removed)
   - MDR1=0x03 (UART in reset)

12.12 UART FIFO Configuration

The goal of the UART FIFO configuration is to set trigger level for halt/restore (TCR register), set trigger level for transmit/receive (TLR register), and configure the FIFO (FCR register).

The procedure of the UART FIFO configuration is as follows:

1) Write into both the TLR and TCR registers
   - Set EFR[4] to 1
   - Set MCR[6] to 1.
2) Write into FCR.
   - Set EFR[4] to 1.

     EFR[4] = 1 has already been done in the previous section, so a simple write to MCR[6] is necessary.
3) Set TCR TLR and FCR to the desired value.

   Here accesses to TCR TLR and FCR must be disabled to avoid any further undesired write to these registers:
   - LCR=0xBF (provides access to EFR)
   - EFR[4]=0
   - LCR[7]=0
   - MCR[6]=0

12.12.1 Baud Rate Data and Stop Configuration

The goals of the baud rate and stop configuration are to configure UART data, stop (LCR register) baud rate (DLH and DLL registers), and enable UART operation. If interrupt capability is added, configuration must be added right before UART enable.

The procedure to accomplish these goals is as follows:

1) Input clock is 12 MHz, so set OSC_12M_SEL to 1.
2) Set LCR to desired value.
3) LCR[7] to 1 (access to DLH and DLL registers).
4) Set DLH and DLL.
5) LCR[7]=0 (removes access to DLH and DLL registers)
6) Set IER to desired value (sets interrupts).
7) MDR1[2:0]=0 (enables UART)

   The UART module is operational.

Chapter 13

USB Function Module

This chapter describes the components and features of the OMAP5910 universal serial bus (USB) function module.

Topic **Page**

13.1 Overview

The USB function module supports the implementation of a full-speed device fully compliant with the USB 1.1 standard. It provides an interface between the MPU core (TI925T) and the USB wire, and it handles USB transactions with minimal TI925T intervention.

The module supports one control endpoint (EP0) (IN and OUT), up to 15 IN endpoints, and up to 15 OUT endpoints. The exact endpoint configuration is software programmable. For each endpoint, the specific parameters of a configuration are the size in bytes, the direction (IN, OUT), the type (bulk/interrupt or isochronous), and the associated number.

The module also supports three DMA channels for IN endpoints and three DMA channels for OUT endpoints for either bulk/interrupt or isochronous transactions.

This chapter uses terminology defined in USB1.1 Standard. Reader familiarity with this Standard is assumed. All references to local host (LH) in this chapter refer to the MPU processor.

Figure 13–1 shows the OMAP5910 device with the USB function module highlighted. Figure 13–2 shows the connection of the USB function module within the OMAP5910 in more detail.

13.1.1 OMAP5910 Inputs/Outputs

Several configurations are possible for the USB function:

- ❏ USB function usable with internal transceiver (default configuration)
- ❏ USB function usable with external transceivers
- ❏ USB function not usable

See the details of these configurations in the section 7.14, *USB Host Controller Overview*.

13.1.2 USB Function Interrupts

The USB function generates three interrupts:

1) General USB interrupt (including endpoint 0, DMA, and device states interrupts), IRQ_GENI_ON: Connected to the MPU level 2 interrupt handler, line 20 (level-sensitive)

2) Non-ISO endpoint-specific Interrupt, IRQ_NON_ISO_ON: Connected to the MPU level 2 interrupt handler, line 30 (level-sensitive)

3) Start of frame (SOF) interrupt for ISO transactions, IRQ_ISO_ON: Connected to the MPU level 2 interrupt handler, line 29 (level-sensitive)

The IRQ_ISO_ON interrupt is also connected to the frame adjustment counter module (FAC) to count the number of frame start.

This count value can then be used by system software to adjust the duration of the two time domains with respect to each other to reduce the overflow and underflow.

Figure 13–1. USB Function Module

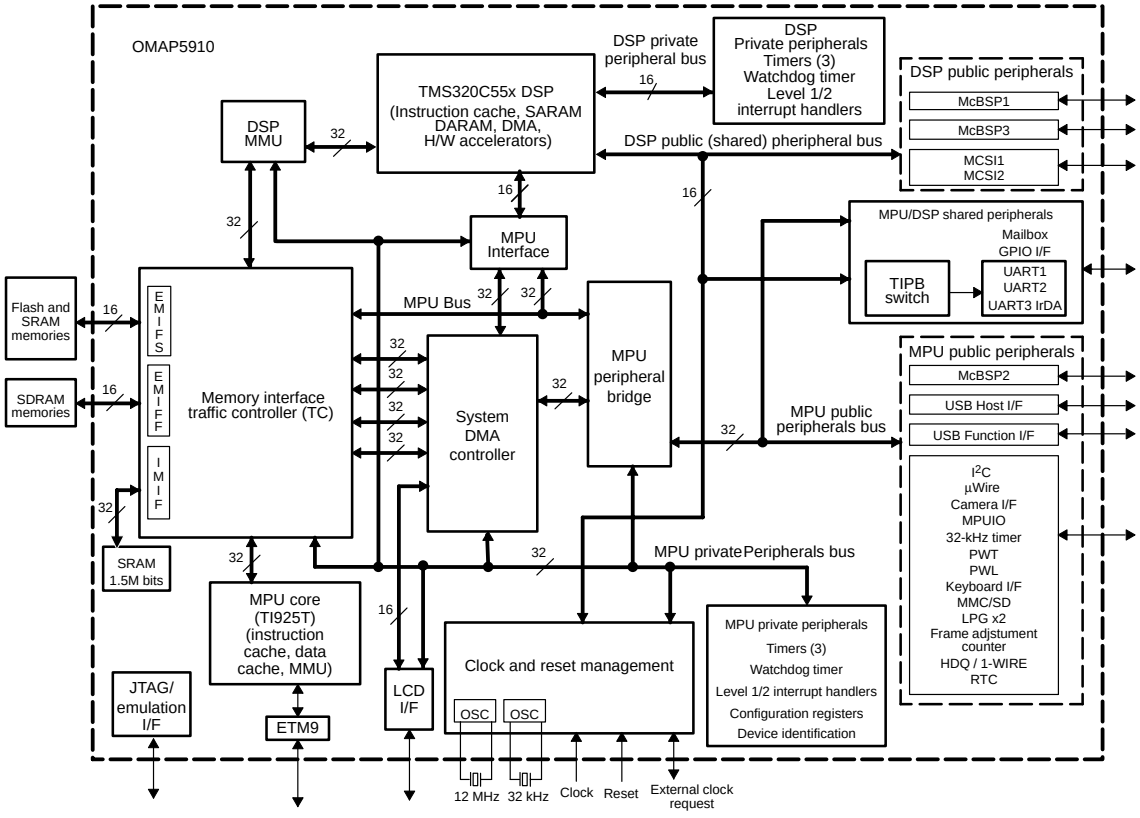
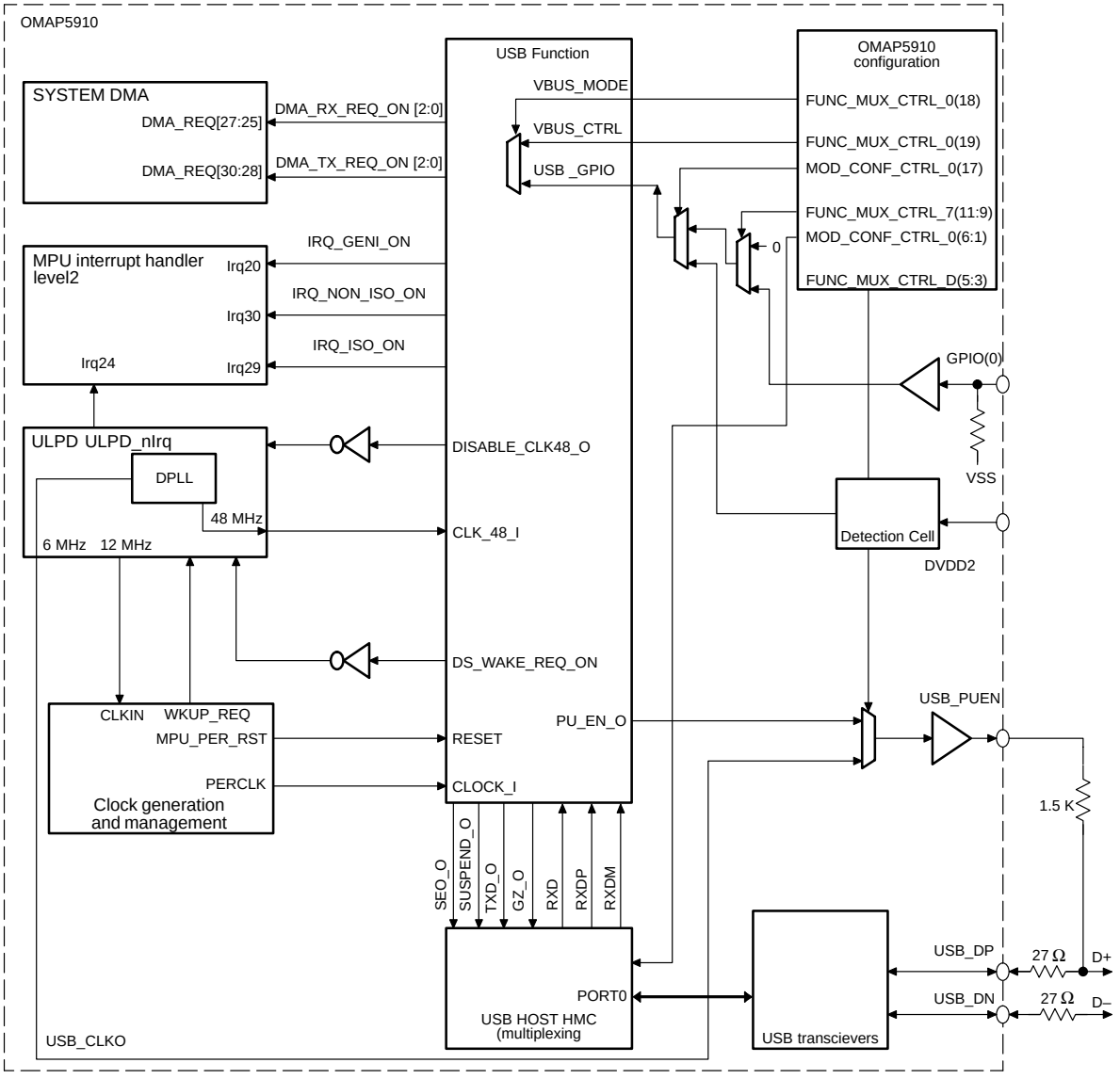


Figure 13–2. USB Function Environment



13.1.3 USB Function Clocks and Reset

The USB function has two clocks:

- An interface clock (CLOCK_I), used between the MPU TIPB and the USB function and connected to the MPU peripheral programmable clock (PERCLK), is derived by dividing CK_GEN1 (the output of DPLL1) by the value associated with the PERDIV field of the ARM_CKCTL register (0xFFFECE00).

  This is a free-running clock when the system is awake.

- A 48-MHz functional clock (CLK_48_I), which is generated by the ULPD DPLL.

  This clock can be shutdown by the USB function (DISABLE_CLK48_O) when:

  - The USB is suspended (idle state).
  - The USB is disconnected.

  This shutdown must be enabled beforehand: Set SOFF_Dis (bit 1), of the SYSCON1 register to 0 (default value).

The MPU TIPB reset (MPU_PER_RST) resets the USB function.

13.1.4 USB Function DMA Requests

The USB function can use:

- Three receive DMA channels (DMA_RX_REQ_ON [2:0]): Any endpoint number (1:15) can be selected to be assigned to one receive DMA channel.

  The DMA_RX_REQ_ON [2:0] are connected to the system DMA request [27:25].

- Three transmit DMA channels (DMA_TX_REQ_ON [2:0]): any endpoint number (1:15) can be selected to be assigned to one transmit DMA channel.

  The DMA_TX_REQ_ON [2:0] are connected to the system DMA request [30:28].

13.1.5 USB Detection

When OMAP5910 is in deep sleep mode, the USB function can detect a bus connection to an external USB host or hub and generate a deep sleep wake request (DS_WAKE_REQ_ON) to wake up the system and to get the interface clock (CLOCK_I).

The USB function can also generate a DS_WAKE_REQ_ON request while the USB is connected: if the USB function is in idle (SUSPEND state) and a resume event occurs, the DS_WAKE_REQ_ON request is generated. This 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) (see Chapter 15, *Clock Generation and System Reset Management*).

Once the interface clock is present, the USB function can generate an attached/unattached interrupt when it detects that the OMAP5910 device is connected to an external USB host or hub or when it becomes disconnected.

This attached/unattached interrupt uses the general USB interrupt line (connected to the MPU level 2 interrupt handler, line 20) and the connection state is given by the ATT bit(0) of the DEVSTAT register.

The USB function can detect whether or not the USB is connected via either software or hardware.

This choice (soft/hard) is made by the VBUS_MODE bit (18) of the OMAP5910 FUNC_MUX_CTRL_0 register:

- ❏ VBUS_MODE = 1 (default value): Software detection is selected.
- ❏ VBUS_MODE = 0: Hardware detection is selected.

13.1.5.1 Software Detection

This detection depends on the VBUS_CTRL bit (19) of the OMAP5910 FUNC_MUX_CTRL_0 register:

- ❏ VBUS_CTRL=0 (default): USB not connected
- ❏ VBUS_CTRL=1: USB connected

13.1.5.2 Hardware Detection

This detection can have two sources:

- The GPIO(0) OMAP5910 input
- The OMAP5910 input of the USB function I/O power supply (DVDD2)

The selection between these two sources is made by the USB_W2FC_VBUS_MODE bit(17) of the MOD_CONF_CTRL_0 register (bit usable only in OMAP5910 configuration mode):

- USB_W2FC_VBUS_MODE = 0 (default): GPIO(0) is selected.
- USB_W2FC_VBUS_MODE = : I/O power supply detection is selected.

GPIO0 Detection

This detection must be enabled by software as follows:

- In either the OMAP1509 or OMAP5910 configuration mode, when the VBUS_GPIO0_SELECT bit (12) (FUNC_MUX_CTRL_1 register) is set

  Or

- In the OMAP5910 configuration mode, when the CONF_GPIO0_MODE bits (11:9) (FUNC_MUX_CTRL_7 register) are set to 010

The results of this detection are as follows:

- GPIO0 input = 0: USB not connected (default value via internal pulldown)
- GPIO0 input = 1: USB connected

I/O Power Supply Detection

An analog cell is used to detect the presence or not of the USB I/O power supply (DVDD2), which permits the detection of the presence or absence of the USB.

- DVDD2 present: USB connected
- DVDD2 not present: USB not connected

13.1.6 Software Disconnect

The PULLUP_EN (0) bit of the SYSCON1 register allows the device to disconnect itself from the USB.

This bit is by default directly connected to the OMAP5910 pad (USB.PUEN), which must be connected on the application board to a 1.5-kΩ resistor.

The other pin of the resistor must be connected to the positive differential line (OMAP5910 pad USB.DP) of the USB (D+).

Thus:

- PullUp_En bit=0 (by default): The external 1.5-kΩ is seen as a pulldown on the USB D+; the external USB host cannot detect the OMAP5910 USB function.
- PullUp_En bit=1: The external 1.5-kΩ is seen as a pullup on the USB D+, the USB host detects this level and, therefore, the presence of the OMAP5910 USB function. The USB host can then configure the USB function.

The USB.PUEN signal is multiplexed inside OMAP5910 with the USB.CLKO clock. Bits (5:3) of the FUNC_MUX_CTRL_D register control this multiplexing:

- FUNC_MUX_CTRL_D(5:3)= 000 (default): USB.PUEN signal is output.
- FUNC_MUX_CTRL_D(5:3)= 001: USB.CLKO clock is output.

The USB.CLKO clock comes from the ULPD DPLL after an internal dividing by 8 (6 MHz) (see Chapter 15, *Clock Generation and System Reset Management* for more details). This clock is dedicated to an external USB HUB.

13.2 Register Map

Table 13–1 lists the USB function registers. Table 13–2 through Table 13–24 describe the register bits. The MPU base address is FFFB:4000.

Table 13–1. USB Function Module Registers

Register	Description	Access	Offset Address
REV	Revision	R	0x00
	Endpoint		
EP_NUM	Endpoint selection	R/W	0x04
DATA	Data	R/W	0x08
CTRL	Control	Set only	0x0C
STAT_FLG	Status flag	R	0x10
RXFSTAT	Receive FIFO status	R	0x14
SYSCON1	System configuration 1	R/W	0x18
SYSCON2	System configuration 2	Set only	0x1C
DEVSTAT	Device status	R	0x20
SOF	Start of frame	R	0x24
IRQ_EN	Interrupt enable	R/W	0x28
DMA_IRQ_EN	DMA interrupt enable	R/W/Clear	0x2C
IRQ_SRC	Interrupt source	R/Clear	0x30
EPN_STAT	Endpoint interrupt status	R	0x34
DMAN_STAT	DMA endpoint interrupt status	R	0x38
Reserved			0x3C
	DMA Configuration		
RXDMA_CFG	Receive channels DMA configuration	R/W	0x40
TXDMA_CFG	Transmit channels DMA configuration	R/W	0x44
DATA_DMA	DMA FIFO data	R/W	0x48
Reserved			0x4C
TXDMA0	Transmit DMA control 0	R/W	0x50

Table 13–1. USB Function Module Registers (Continued)

Register	Description	Access	Offset Address
	DMA Configuration (Continued)		
TXDMA1	Transmit DMA control 1	R/W	0x54
TXDMA2	Transmit DMA control 2	R/W	0x58
Reserved			0x5C
RXDMA0	Receive DMA control 0	R/W	0x60
RXDMA1	Receive DMA control 1	R/W	0x64
RXDMA2	Receive DMA control 2	R/W	0x68
Reserved			0x6C-0x7C
	Endpoint Configuration		
EP0	Endpoint configuration 0	R/W	0x80
EP1_RX	Receive endpoint configuration 1	R/W	0x84
EP2_RX	Receive endpoint configuration 2	R/W	0x88
…			…
EP15_RX	Receive endpoint configuration 15	R/W	0xBC
Reserved			0xC0
EP1_TX	Transmit endpoint configuration 1	R/W	0xC4
EP2_TX	Transmit endpoint configuration 2	R/W	0xC8
…			…
EP15_TX	Transmit endpoint configuration 15	R/W	0xFC

Note on register accesses:

- The local host may read from or write into registers using one of the following accesses:
  - 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.

Local host actions are required in some particular cases when reading or writing data, depending on the access mode.

13.2.1 Revision Register (REV)

The read-only revision register (REV) contains the revision number of the module. A write to this register is forbidden.

Table 13–2. Revision Register (REV)

Bit	Name	Description
15–8	–	Reserved
7–0	Rev_nb	Revision number

13.2.1.1 REV_NB

This 8-bits field indicates the revision number of the current USB function module. This value is fixed by hardware.

0x01: Revision 0.1

0x02: Revision 0.2

0x21: Revision 2.1

....

Local host (LH) and universal serial bus (USB) reset have no effect on this register.

13.2.2 Endpoint Selection Register (EP_NUM)

The read/write endpoint selection register (EP_NUM) selects and enables the endpoint that can be accessed by the local host.

Table 13–3. Endpoint Selection Register (EP_NUM)

Bit	Name	Description
15–7	–	Reserved
6	Setup_Sel	Setup FIFO select
5	EP_Sel	TX / RX FIFO select
4	EP_Dir	Endpoint direction
3–0	EP_Num	Endpoint number

13.2.2.1 Setup FIFO Select (Setup_Sel)

Set by the local host in response to a setup general USB interrupt in order to access the EP0 read-only setup FIFO when reading the DATA register. Setting this bit clears the Setup interrupt bit. When this bit is set, other EP_NUM register bits must be 0.

> CAUTION
>
> After having read the setup FIFO, the local host must clear this bit by writing a 0 to it.

0: No access

1: Access permitted

USB reset value: 0

Local hos reset value: 0

13.2.2.2 TX/RX FIFO Select (EP_Sel)

Set by the local host to access the status (STAT_FLG, RXFSTAT) and data (DATA) registers for the endpoint selected. If EP_Dir bit is set to 0, the local host can read data from endpoint RX FIFO by reading the DATA register; if EP_Dir bit is set to 1, the local host can write data into endpoint TX FIFO by writing into the DATA register. After each access to an endpoint during interrupt handling, the local host must clear this bit.

> CAUTION
>
> Before the local host 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 local host must clear this bit by writing a 0 to it.

0: No access

1: Access permitted

Value after local host or USB reset is low.

13.2.2.3 Endpoint Direction (EP_Dir)

This bit gives the direction associated with the endpoint number selected in EP_Num.

0: OUT endpoint

1: IN endpoint

Value after local host or USB reset is low.

13.2.2.4 Endpoint Number (EP_Num)

The endpoint number binary encoded in these four bits, associated to the direction given by EP_Dir bit, is the current endpoint selected. All reads and writes to the endpoint status, control, and data locations are for this endpoint.

0000: EP0

0001: EP1

….

1111: EP15.

Value after local host or USB reset is low.

13.2.3 Data Register (DATA)

The data register (DATA) is the entry point to write into a selected TX endpoint, to read data from a selected RX endpoint, or to read data from the setup FIFO. If selected endpoint direction is OUT, this register is read-only and a write into it is forbidden. If selected endpoint direction is IN, this register is write-only and a read of this register is forbidden.

Table 13–4. Data Register (DATA)

Bit	Name	Description
15–0	DATA	Transmit/receive FIFO data

13.2.3.1 Transmit/Receive FIFO Data (DATA)

EP_Dir = 0: This register contains the data received by the USB core from USB host out or setup transactions. Data can only be read successfully if the EP_Sel bit is asserted, or if Setup_Sel bit is asserted (for setup data).

EP_Dir = 1: This register contains the data written by the local host to be sent to the USB host during the next IN transaction. Data can only be written successfully if the EP_Sel bit is asserted.

Note:

Writing the DATA register when EP_Dir = 0 and reading from DATA register when EP_Dir = 1 are denied.

13.2.4 Control Register (CTRL)

This set-only control register (CTRL) controls the FIFO and status of the selected endpoint. A read access to this register always returns 0.

Note:

The endpoint 0 setup FIFO is always enabled and ready to accept setup data. No control register (CTRL) is implemented for this FIFO, because the local host cannot control it.

Table 13–5. Control Register (CTRL)

Bit	Name	Description
15–8	–	Reserved
7	Clr_Halt	Clear halt endpoint (non-isochronous)
6	Set_Halt	Set halt endpoint (non-isochronous)
5–3	–	Reserved
2	Set_FIFO_En	Set FIFO enable (non-isochronous)
1	Clr_EP	Clear endpoint
0	Reset_EP	Endpoint reset (non-Ctrl)

13.2.4.1 Clear Halt Endpoint (Clr_Halt)

Only concerns non-isochronous endpoints.

Used by the local host to clear an endpoint halt condition.

0: No action

1: Clear halt condition

Always read 0.

Note:

It is not required to set the EP_Sel bit before setting this bit. Except when this bit is set during the handling of an interrupt to the endpoint, the local host must not set the EP_Sel bit before setting the Clr_Halt bit, in order to avoid possible impacts on interrupts.

13.2.4.2 Set Halt Endpoint (Set_Halt)

Only concerns non-isochronous endpoints.

Used by the local host to halt the selected endpoint. The halted endpoint returns STALL handshakes to the USB host. The local host can disable the endpoint interrupt if it does not wish to be informed of STALL handshakes.

> CAUTION
>
> If the endpoint to halt is used by a DMA channel, the local host must disable the DMA channel before setting the halt conditions for this endpoint.

0: No action

1: Halt endpoint

Always read 0.

> **Note:**
>
> It is not required to set the EP_Sel bit before setting this bit. Except when this bit is set during the handling of an interrupt to the endpoint, the local host must not set the EP_Sel bit before setting the Set_Halt bit, in order to avoid possible impacts on interrupts.
>
> The local host must check that FIFO is empty before setting the halt feature for the endpoint. A stalled transaction has no effect in clearing the FIFO.

13.2.4.3 Set FIFO Enable (Set_FIFO_En)

Only concerns non-isochronous endpoints.

If the selected endpoint direction is IN, this bit is used by the local host to enable the USB device to transmit data from the FIFO at the next valid IN token. If the selected endpoint direction is OUT, this bit is used by the local host to enable the USB device to receive data from the USB host at the next valid OUT transaction. If not set, the device returns a NAK handshake.

Isochronous endpoint FIFOs are always enabled.

The local host must never enable endpoint 0 FIFO if not performing a control transfer. For bulk and interrupt endpoints, the FIFO must never be enabled when Set_Halt = 1 (halt feature enabled) or when RX FIFO is not empty. Furthermore, during endpoint interrupt handling, the local host must have cleared the interrupt bit before setting the Set_FIFO_En bit (to avoid masked ACK interrupts).

0: No action

1: FIFO enabled

Always read 0.

Note:

It is not required to set the EP_Sel bit before setting this bit. Except when this bit is set during the handling of an interrupt to the endpoint, the local host must not set the EP_Sel bit before setting the Set_FIFO_En bit, in order to avoid possible impacts on interrupts.

13.2.4.4 Clear Endpoint (Clr_EP)

This bit is set by the local host to clear the selected endpoint FIFO pointers and flags. This resets the FIFO pointers and the FIFO empty status bit. The FIFO enable bit and other FIFO flags are cleared upon completion of the FIFO reset. Previous transaction handshake status is also cleared. For isochronous endpoints or non-isochronous double-buffered endpoints, both foreground and background FIFO are cleared.

0: No action

1: Clear endpoint

Always read 0.

13.2.4.5 Endpoint Reset (Reset_EP)

Only concerns non-control endpoints.

Set by the local host to reset the selected endpoint. It forces an interrupt or a bulk endpoint data PID to DATA0, clears halt condition (HALT = 0), and clears FIFO (both foreground and background if endpoint is double-buffered) and previous transactions handshake status. For an isochronous endpoint, it only clears the FIFO (both foreground and background).

0: No action

1: Reset endpoint

Always read 0.

13.2.5 Status Register (STAT_FLG)

The read-only status flag register provides a status of the FIFO and the results of the transaction handshakes for the selected endpoint. The eight MSB are reserved for isochronous endpoints, while the eight LSB are reserved for non-isochronous endpoints. This register cannot be read if EP_Sel bit is not asserted for the endpoint. No status flag exists for the read-only setup FIFO, which is always enabled.

> The updates for non-isochronous transactions are done at the end of each non-transparent and valid transaction to a given endpoint, if no interrupt is pending on the endpoint.

Note:

Non-transparent, non-isochronous IN transactions are those transactions responding with an ACK handshake, a STALL handshake, or optionally a NAK handshake if the Nak_En bit is asserted to 1. An ERR handshake or a NAK handshake when the Nak_En bit is 0 is considered transparent.

A write to this register has no effect.

Table 13–6. Status Register (STAT_FLG)

Bit	Name	Description
15	–	Reserved
14	Miss_In	Isochronous missed IN token for the previous frame (isochronous IN)
13	Data_Flush	Isochronous receive data flush (isochronous OUT)
12	ISO_Err	Isochronous receive data error (isochronous OUT)
11–10	–	Reserved
9	ISO_FIFO_Empty	Isochronous FIFO empty
8	ISO_FIFO_Full	Isochronous FIFO full
7	–	Reserved
6	EP_Halted	Endpoint halted flag (non-isochronous)
5	STALL	Transaction stall (non-isochronous)
4	NAK	Transaction non-acknowledge (non-isochronous)

Table 13–6. Status Register (STAT_FLG) (Continued)

Bit	Name	Description
3	ACK	Transaction acknowledge (non-isochronous)
2	FIFO_En	FIFO enable status (non-isochronous)
1	Non_ISO_FIFO_Empty	Non-isochronous FIFO empty
0	Non_ISO_FIFO_Full	Non-isochronous FIFO full

13.2.5.1 Isochronous Missed IN Token (Miss_In)

Only concerns isochronous IN endpoints.

Notifies the local host that the core missed a valid isochronous IN token during previous frame and that TX data was flushed from the FIFO instead of being transmitted to the USB host. This bit is updated on a start of frame (SOF).

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 was flushed.

Value after local host or USB reset is low.

13.2.5.2 Isochronous Receive Data Flush (Data_Flush)

Only concerns isochronous OUT endpoints.

When set, this bit indicates that data was flushed from the isochronous FIFO that was moved from the foreground to the background. This happens when the local host does not read all of the data from the foreground FIFO in a frame.

This bit is updated every frame.

0: Not significant

1: Data was flushed

Value after local host or USB reset is low.

13.2.5.3 Isochronous Receive Data Error (ISO_Err)

Only concerns isochronous OUT endpoints.

When set, this bit indicates that the isochronous 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 every frame.

0: Not significant

1: Isochronous packet received with errors

Value after local host or USB reset is low.

13.2.5.4 Isochronous FIFO Empty (ISO_FIFO_Empty)

Only concerns isochronous endpoints.

Set when the FIFO for the selected isochronous endpoint is empty, either via an appropriate write to the Clr_EP bit or the Reset_EP bit, or after successful reads from the selected FIFO.

0: Isochronous FIFO not empty

1: Isochronous FIFO empty

Value after local host or USB reset is high (FIFO empty).

13.2.5.5 Isochronous FIFO Full (ISO_FIFO_Full)

Only concerns isochronous endpoints.

Set when the FIFO for the selected isochronous endpoint is full. This condition is cleared by setting the Clr_EP bit or the Reset_EP bit, or after one successful read (by the local host or the USB host).

0: Isochronous FIFO not full

1: Isochronous FIFO full

Value after local host or USB reset is low (FIFO empty).

13.2.5.6 Endpoint Halted Flag (EP_Halted)

Only concerns non-isochronous endpoints.

This bit, when set to 1, indicates the selected endpoint is halted. The endpoint can be put into the halt state only by the local host writing the endpoint halt control bit (in response to a Set_Feature request, for instance).

0: The selected endpoint is not halted.

1: The selected endpoint is halted.

Value after local host or USB reset is low.

13.2.5.7 Transaction Stall (STALL)

Only concerns non-isochronous 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 interrupt is pending on current buffer. The core automatically returns a STALL packet if a valid IN token is received by a halted TX endpoint, if a valid OUT transaction is received by an halted RX endpoint, or if there is a request error (endpoint 0). The bit is cleared when the local host has finished handling the corresponding interrupt (at EP_Sel bit deselection).

0: No STALL handshake was returned.

1: A STALL handshake packet was returned.

Value after local host or USB reset is low.

13.2.5.8 Transmit Non-Acknowledge (NAK)

Only concerns non-isochronous endpoints with the 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 interrupt is pending on current buffer. The USB core automatically returns a NAK handshake to the USB host if a valid IN token is received by a TX endpoint or if a valid OUT transaction is received by an RX endpoint and the FIFO_En bit is not set for the endpoint. The bit is cleared when the local host has finished handling the corresponding interrupt (at EP_Sel bit deselection).

0: No NAK handshake was returned (the Nak_En bit is set).

1: A NAK handshake packet was returned and the Nak_En bit is set.

Value after local host or USB reset is low.

13.2.5.9 Transaction Acknowledge (ACK)

Only concerns non-isochronous endpoints.

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 local host has finished handling the corresponding interrupt (at EP_Sel bit deselection).

0: No ACK handshake packet was returned.

1: An ACK handshake packet was returned.

Value after local host or USB reset is low.

13.2.5.10 FIFO Enable (FIFO_En)

Only concerns non-isochronous endpoints.

This bit is asserted when the Set_FIFO_En bit is set to 1 and is cleared automatically after a transaction completes with an ACK or STALL.

0: The non-isochronous endpoint FIFO is disabled.

1: The non-isochronous endpoint FIFO is enabled.

Value after local host or USB reset is low.

13.2.5.11 Non-Isochronous FIFO Empty (Non_ISO_FIFO_Empty)

Only concerns non-isochronous endpoints.

Set when the FIFO for the selected non-isochronous endpoint is empty, either via an appropriate Clr_EP bit or the Reset_EP bit or after successful reads from the selected FIFO.

0: Non-isochronous FIFO not empty

1: Non-isochronous FIFO empty

Value after local host or USB reset is high (FIFO empty).

13.2.5.12 Non-Isochronous FIFO Full (Non_ISO_FIFO_Full)

Only concerns non-isochronous endpoints.

Set when the FIFO for the selected non-isochronous endpoint is full. This condition is cleared by setting the Clr_EP bit or the Reset_EP bit, or after one successful read (by the local host or the USB host).

0: Non-isochronous FIFO not full

1: Non-isochronous FIFO full

Value after local host or USB reset is low (FIFO empty).

13.2.6 Receive FIFO Status Register (RXFSTAT)

The read-only receive FIFO status register (RXSTAT) tells how many bytes are in the receive FIFO for the selected endpoint. A write to this register has no effect. The local host cannot read this register if EP_Sel bit is not set for the endpoint.

Note:

No receive FIFO status exists for the setup FIFO, because 8 bytes always are expected.

Table 13–7. Receive FIFO Status Register (RXSTAT)

Bit	Name	Description
15–10	–	Reserved
9–0	RXF_Count	Receive FIFO byte count

13.2.6.1 Receive FIFO Byte Count (RXF_Count)

This 10-bit field indicates the number of bytes currently in the receive FIFO.

Value after local host or USB reset is low (all 10 bits).

13.2.7 System Configuration Register 1 (SYSCON1)

The read/write system configuration 1 register (SYSCON1) provides control functions for power management and miscellaneous control for the core.

Table 13–8. System Configuration Register 1(SYSCON1)

Bit	Name	Description
15–9	–	Reserved
8	Cfg_Lock	Device configuration locked
7–5	–	Reserved
4	Nak_En	NAK enable
3	–	Reserved
2	Self_Pwr	Self-powered
1	SOFF_Dis	Shutoff disable
0	Pullup_En	External pullup enable

13.2.7.1 Device Configuration Locked (Cfg_lock)

After the local host has entered the device configuration (registers 0x20 to 0x3F), it must set this bit so that the device can be used. If 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.

Value after local host reset is low, after USB reset is unchanged (keep previous configuration).

13.2.7.2 NAK Enable (Nak_En:)

This bit can be set by the local host so that it will be signaled for NAK transaction handshake response. When this bit is set, the NAK bit is set on a NAK handshake if no 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 local host and no interrupt is asserted.

0: NAK disabled

1: NAK enabled

Value after local host or USB reset is low.

Note:

If the local host sets this bit, it must wait for a NAK interrupt before selecting the TX endpoint to write TX data.

13.2.7.3 Self-Powered (Self_Pwr)

Indicates to the USB host whether the device is bus-powered or self-powered. This is needed for a GET_DEVICE_STATUS auto-decoded request. The local host 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 chapter 9).

0: Bus-powered

1: Self-powered

Value after local host reset is low, after USB reset is unchanged.

13.2.7.4 Shutoff Disable (SOFF_Dis)

When this bit is set, it disables the power shutoff circuitry.

0: Power shutoff circuitry enabled

1: Power shutoff circuitry disabled

Value after local host reset is low, after USB reset is unchanged.

13.2.7.5 External Pullup Enable (Pullup_En)

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: Pullup disabled. USB host cannot detect the device. In this mode, the 48-MHz USB clock is forced off.

1: Pullup enabled.

Value after local host reset is low, after USB reset is high, after detach is unchanged.

13.2.8 System Configuration Register 2 (SYSCON2)

The set-only system configuration 2 register (SYSCON2) provides miscellaneous controls for the function. A read of this register always returns 0.

Table 13–9. SYSCON2 – System Configuration Register 2 (SYSCON2)

Bit	Name	Description
15–7	–	Reserved
6	Rmt_Wkp	Remote wakeup
5	Stall_Cmd	Stall command
4	–	Reserved
3	Dev_Cfg	Device configured
2	Clr_Cfg	Clear configured
1–0	–	Reserved

13.2.8.1 Remote Wakeup (Rmt_Wkp)

This set-only bit when written with a 1 initiates the remote wakeup sequence, regardless of whether or not the R_Wk_OK bit has been previously set to 1 by the USB host. So the MPU must make sure that remote wakeup has been enabled by the USB host by reading the DEVSTAT register to check that the R_Wk_OK bit [6] is set to 1 before generating a remote wakeup by writing a 1 to the Rmt_Wkp bit [6]. Reading the Rmt_Wkp bit always returns 0. Writing 0 into this bit has no effect.

0: No action

1: Initiates the remote wakeup sequence

Always read 0.

13.2.8.2 Stall Command (Stall_Cmd)

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. This set-only bit always reads 0.

0: No action

1: Stall current USB command

Always read 0.

13.2.8.3 Device Configured (Dev_Cfg)

If the local host receives a SET_CONFIGURATION with a valid configuration value and the device is in addressed state, it must write a 1 to this bit to inform the command decodes that the device has moved to the configured state. The CFG bit is set to 1 by the core.

If the device is already configured when the SET_CONFIGURATION request is received, the local host must not set this bit. If the new configuration value is 0, it must set the Clr_Cfg bit in order to move to the addressed state.

Reading this bit always returns 0. Writing 0 to this bit has no effect.

0: No action

1: Allows CFG to be set

Always read 0.

13.2.8.4 Clear Configured (Clr_Cfg)

If the local host receives a SET_CONFIGURATION with a configuration value of 0 and if device is configured, it must write a 1 to this bit to inform the command decoder that the device is now deconfigured (has moved to the addressed state). The CFG bit is cleared by the core.

Reading this bit always returns 0. Writing 0 to this bit has no effect.

0: No action

1: Allows CFG to be cleared

Always read 0.

13.2.9 Device Status Register (DEVSTAT)

The read-only device status register (DEVSTAT) provides a status reflecting the visible device states as defined in USB1.1 chapter 9. A write to this register has no effect.

This register is double buffered. If the DS_Chg_IE bit is set (interrupt enabled), the background register is moved to foreground position only after clearing any pending DS_Chg interrupts. So if there is a state change and there is still a pending DS_Chg interrupt, then recent state change is not visible because the background register was updated and not moved into foreground position.

Table 13–10. Device Status Register (DEVSTAT)

Bit	Name	Description
15–7	–	Reserved
6	R_WK_OK	Remote wakeup granted
5	USB_Reset	USB reset signaling is active
4	SUS	Suspended state
3	CFG	Configured state
2	ADD	Addressed state
1	DEF	Default state
0	ATT	Attached state

13.2.9.1 Remote Wakeup Enabled (R_WK_OK)

This 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 has granted the function the ability to assert remote wakeup.

0: Not significant

1: Remote wakeup granted

Value after local host or USB reset is low.

13.2.9.2 USB Reset Signaling (USB_Reset)

This bit returns 1 when the USB host is resetting the USB bus.

A valid USB reset resets all the endpoint FIFOS, all other control register bits except Cfg_Lock, all associated configuration registers (0x20 to 0x3F), and bits DS_Chg_IE and DS_Chg. This register (DEVSTAT) forces the device to the default state. This bit is cleared at the end of reset.

This bit is double buffered just as the other DEVSTAT bits are. If there is a pending interrupt that is not handle when a USB reset occurs, and if that interrupt is handled only when USB reset is finished, the local host does not see the USB_Reset bit going high and then low.

0: Device not being reset by USB host

1: Device is being reset by USB host

Value after local host reset is low and during USB reset is high (low after USB reset).

13.2.9.3 Suspended State (SUS)

Device is, at minimum, attached to the USB and is powered, has been reset by the USB host, and has not seen bus activity for 5 ms. It may also have a unique address and be configured for use. However, because the device is suspended, the host can not use the device function. This bit returns 1 when the USB device is in suspend state.

0: Not suspended

1: Suspended

Value after local host or USB reset is low.

13.2.9.4 Configured State (CFG)

Device is attached to the USB and powered, has been reset, has a unique address, and is configured. The host can now use the function provided by the device. This bit returns 1 when the USB device has been configured after a set 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 local host sets Clr_Cfg bit. While this bit is not set to 1, any transaction not for control EP0 is ignored. A GET_ENDPOINT_STATUS to a non-control endpoint is stalled.

0: Not configured

1: Configured

Value after local host or USB reset is low.

13.2.9.5 Addressed State (ADD)

Device is attached to the USB and powered, has been reset, and a unique device address has been assigned. This bit returns 1 after a SET_ADDRESS standard request. This bit remains set to 1 until the device becomes de-addressed.

0: Not addressed

1: Addressed

Value after local host or USB reset is low.

13.2.9.6 Default State (DEF)

This 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 depowered. Device moves into default state as soon as the USB reset is effective.

0: Not in default

1: Default

Value after local host is low and after USB reset is high.

13.2.9.7 Attached State (ATT)

This bit returns 1 when the device is attached to the USB and powered. This bit remains set to 1 until the device becomes depowered.

0: Not attached

1: Attached

Value after local host reset is low (unattached) or high (attached). After USB reset, value is high.

13.2.10 Start of Frame Register (SOF)

The read-only start of frame register (SOF) provides a frame timer status for use in isochronous communications. A write to this register is forbidden.

Table 13–11. Start of Frame Register (SOF)

Bit	Name	Description
15–13	–	Reserved
12	FT_Lock	Frame timer locked
11	TS_OK	Time stamp OK
10–0	TS	Time stamp number

13.2.10.1 Frame Timer Locked (FT_Lock)

The USB host sends out a start of frame (SOF) packet every millisecond. When a SOF packet is not received by the device due to a bus error, a local start-of-frame is generated for use by the isochronous FIFO switch.

Once the core receives two valid SOFs separated by time-frame (TF), it sets the FT_LOCK to 1 only if TF is the frame interval (IF) allowed by USB 1.1 specification (IF = [11964:12036] USB bit time). If TF is out of this interval, the FT_Lock value remains 0, and a local SOF is generated by the core.

When the FT_Lock bit 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 the The FT_Lock bit is cleared if a valid SOF is received out of the interval IF. If the core receives a valid SOF in this interval, the frame timer locks to the new frame-time. If the core does not receive a valid SOF, the frame timer remains lock to TF.

When the FT_Lock is cleared, a local SOF is generated after 12036 USB bit times if no valid SOF has been received, and the FT_Lock remains 0.

0: Frame timer is not locked.

1: Frame timer is locked.

Value after local host or USB reset is low.

Each time a valid SOF is received by the core out of allowed interval IF, a local SOF is generated and isochronous FIFO switch.

13.2.10.2 Time Stamp OK(TS_OK)

This bit indicates that the time stamp in the TS field 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: Time stamp is invalid.

1: Time stamp is valid.

Value after local host or USB reset is low.

13.2.10.3 Time Stamp Number(TS)

This field returns the time stamp from last USB host valid SOF packet. The frame number is valid if the TS_OK is 1. In case of a SOF miss, this value is not updated and TS_OK is cleared.

Value after local host or USB reset is low (all 11 bits).

13.2.11 Interrupt Enable Register (IRQ_EN)

The read/write interrupt enable register (IRQ_EN) enables all non-DMA interrupts (control, state changed, isochronous, non-isochronous).

Table 13–12. Interrupt Enable Register (IRQ_EN)

Bit	Name	Description
15–8	–	Reserved
7	SOF_IE	Start-of-frame interrupt enable
6	–	Reserved
5	EPn_RX_IE	Receive endpoint n interrupt enable (non-isochronous)
4	EPn_TX_IE	Transmit endpoint n interrupt enable (non-isochronous)
3	DS_Chg_IE	Device state changed interrupt enable
2–1	–	Reserved
0	EP0_IE	EP0 transactions interrupt enable

When a bit position is set to 1 by the local host, an interrupt is signaled to the local host 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 local host.

0: Interrupt disabled

1: Interrupt enabled

Value after local host or USB reset is low for all bits except the DS_Chg_IE bit, which remains unchanged after a reset from USB host.

13.2.12 Interrupt Source Register (IRQ_SRC)

The read/clear-only interrupt source register (IRQ_SRC) has for function to identify and clear the source of the interrupt signaled by a set flag.

Table 13–13. Interrupt Source Register (IRQ_SRC)

Bit	Name	Description
15–11	–	Reserved
10	TXn_Done	Transmit DMA channel n done interrupt flag (non-isochronous)
9	RXn_Cnt	Receive DMA channel n transactions count interrupt flag (non-isochronous)
8	RXn_EOT	Receive DMA channel n end of transfer interrupt flag (non-isochronous)
7	SOF	Start-of-frame interrupt flag
6	–	Reserved
5	EPn_RX	EPn OUT transactions interrupt flag
4	EPn_TX	EPn IN transactions interrupt flag
3	DS_Chg	Device state changed interrupt flag
2	Setup	Setup transaction interrupt flag
1	EP0_RX	EP0 OUT transactions interrupt flag
0	EP0_TX	EP0 IN transactions interrupt flag

Common to all bits:

The local host can only clear a set bit location by writing a 1 into the bit location (except for Setup bit, which is automatically cleared by the core). A write of 0 has no effect.

When a bit location is set to 1 by the core, an interrupt is signaled to the local host if the interrupt was enabled.

0: No interrupt

1: Interrupt signaled

Value after local host or USB reset is low, except for the DS_Chg bit, which is high after a USB reset.

13.2.12.1 Transmit DMA CH.n Done Interrupt Flag (TXn_Done)

Only for non-isochronous DMA transfer. This bit is never set for isochronous DMA transfer.

This bit is set automatically by the core when a transmit DMA channel has completed the programmed transfer by servicing the last IN transaction from the USB host. This is when TXn_TSC (transfer size counter) equals 0 and the last IN transaction completes with an ACK. When this bit is asserted, the local host must read the DMAN_STAT register to identify the endpoint number for which the transfer completed.

The endpoint interrupt the EPn_TX is never set for the assigned endpoint to TX DMA channel n.

0: No action

1: Non-isochronous transmit DMA transfer for a channel has ended.

Value after local host or USB reset is low.

13.2.12.2 RX DMA CH.n Transactions Count Interrupt Flag (RXn_Cnt)

Only for non-isochronous DMA transfer. This bit is never set for isochronous DMA transfer.

This bit is set automatically by the core during an active receive DMA transfer each time RXn_TC equals 0 after an OUT transaction with ACK status. This bit is set after RX DMA data has been read (end of DMA request). When this bit is asserted, the local host must read the DMAN_STAT register to identify the endpoint number for which the transfer completed. An RXn_Cnt interrupt is asserted also if the RXn_Stop bit is set; in this case, both the RXn_EOT and the RXn_Cnt are asserted.

0: No action

1: Non-isochronous receive DMA transfer for a channel has reached transactions count level.

Value after local host or USB reset is low.

13.2.12.3 Receive DMA CH.n EOT Interrupt Flag (RXn_EOT)

Only for non-isochronous DMA transfer. This bit is never set for isochronous DMA transfer.

This bit is set automatically by the core when a receive DMA channel has detected an end of transfer (EOT) packet during the last OUT transaction from the USB host. This bit is set after RX DMA data has been read (end of DMA request). When this happens, the DMA-assigned endpoint FIFO is kept disabled (the FIFO_En = 0) to avoid receiving a new packet data from the USB host. The local host can grant another DMA transfer to the same endpoint by simply enabling the FIFO again (the FIFO_En = 1).

An end of transfer is detected when the core receives a data packet whose size is less than the configured endpoint FIFO size (or empty) or when RXn_TC equals 0 after an OUT transaction with ACK status and the RXn_Stop bit is set.

When this bit is asserted, the local host must read the DMAn_RX_IT_src to identify the endpoint number for which the transfer completed and must read the DMAn_RX_SB to be informed of an odd number of bytes received during the last transaction (useful for 16-bit read access from DATA_DMA register).

The endpoint interrupt EPn_RX bit is never set for the assigned endpoint to RX DMA channel.

0: No action

1: Non-isochronous receive DMA transfer for a channel has ended.

Value after local host or USB reset is low.

13.2.12.4 Start Of Frame Interrupt Flag (SOF)

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 to the SOF bit location clears the flag. Writing a 0 has no effect.

In accordance with the USB1.1 spec, if an SOF is not received or is corrupted, the core still sets this flag at the same rate (if bit FT_Lock = 1) or after 12043 USB bit times (if bit FT_lock = 0).

0: No action

1: Start of frame packet received (or internal SOF)

Value after local host or USB reset is low.

13.2.12.5 OUT Transaction Endpoint n Interrupt Flag (EPn_RX)

Only concerns non-isochronous endpoints.

This bit is automatically set by the core when a handshake sequence occurs for an OUT transaction to an interrupt of bulk endpoint (NAK with the Nak_En bit set, ACK, or STALL). The local host must read EPN_STAT register to identify the endpoint causing the interrupt.

0: No action

1: OUT transaction detected on an endpoint.

Value after local host or USB reset is low.

13.2.12.6 IN Transaction Endpoint n Interrupt Flag (EPn_TX)

Only concerns non-isochronous endpoints.

This bit is automatically set by the core when a handshake sequence occurs for an IN transaction to an interrupt of bulk endpoint (NAK with the Nak_En bit set, ACK or STALL). The local host must read EPN_STAT register to identify the endpoint causing the interrupt.

0: No action

1: IN transaction detected on an endpoint.

Value after local host or USB reset is low.

13.2.12.7 Device State Changed Interrupt Flag (DS_Chg)

This bit is automatically set by the core 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 local host reset is low and after USB reset is high.

13.2.12.8 Setup Transaction Interrupt Flag (Setup)

This bit is automatically set by the core when a valid setup transaction completes one control endpoint for a non-autodecoded control request and is cleared automatically by the core when the local host sets the Setup_Sel bit when reading setup data. A write of 1 to it has no effect.

0: No action

1: Valid setup transaction occurred on endpoint 0.

Value after local host or USB reset is low.

13.2.12.9 OUT Transaction Endpoint 0 Interrupt Flag (EP0_RX)

This bit is set automatically by the core when a handshake sequence occurs for a non-autodecoded OUT transaction to control endpoint (NAK with the Nak_En bit set, ACK, or STALL).

0: No action

1: OUT transaction on EP0

Value after local host or USB reset is low.

13.2.12.10 IRQ_SRC[0].EP0_TX: IN Transaction Endpoint 0 Interrupt Flag

This bit is set automatically by the core when a handshake sequence occurs for a non-autodecoded IN transaction to control endpoint (NAK with the Nak_En bit set, ACK, or STALL).

0: No action

1: IN transaction on EP0

Value after local host or USB reset is low.

13.2.13 Non-Isochronous Endpoint Interrupt Status Register (EPN_STAT)

The read-only non-isochronous endpoint interrupt status register (ENP_STAT) identifies the non-isochronous endpoint causing an EPn interrupt. A write into it is forbidden.

If a non-transparent transaction occurs before a previous one on another endpoint in the same direction has been handled by the local host, the second interrupt is asserted only after first one has been cleared by the local host and EPN_STAT is updated with the corresponding interrupt assertion.

Table 13–14. Non-Isochronous Endpoint Interrupt Status Register (EPN_STAT)

Bit	Name	Description
15–12	–	Reserved
11–8	EPn_RX_IT_src	Receive endpoint interrupt source (non-isochronous)
7–4	–	Reserved
3–0	EPn_TX_IT_src	Transmit endpoint interrupt source (non-isochronous)

13.2.13.1 Receive Endpoint Interrupt Source (EPn_RX_IT_src)

Only concerns non-isochronous endpoints. When the EPn_RX flag bit is set, the endpoint causing the interrupt condition is encoded in these four register bits. When the EPn_RX flag bit is cleared, the four bits read as 0.

0000: No receive endpoint interrupt is pending.

0001: EP1

….

1111: EP15

Value after local host or USB reset is low (all 4 bits).

13.2.13.2 Transmit Endpoint Interrupt Source (EPn_TX_IT_src)

Only concerns non-isochronous endpoints.

When the EPn_TX flag is set, the endpoint causing this flag to be set is encoded in these four register bits. When the EPn_TX flag is cleared, the four bits read as 0.

0000: No transmit endpoint interrupt is pending.

0001: EP1

….

1111: EP15

Value after local host or USB reset is low (all 4 bits).

13.2.14 Non-Isochronous DMA Interrupt Status Register (DMAN_STAT)

The read-only non-isochronous DMA interrupt status register (DMAN_STAT) identifies the endpoint causing a DMA interrupt. A write into it is forbidden.

If a DMA interrupt occurs before a previous one on another endpoint in the same direction has been handled by the local host, the second interrupt is asserted only after first one has been cleared by the local host and DMAN_STAT is updated when the corresponding interrupt is asserted.

Table 13–15. Non-Isochronous DMA Interrupt Status Register (DMAN_STAT)

Bit	Name	Description
15–11	–	Reserved
12	DMAn_RX_SB	DMA receive single byte (non-isochronous)
11–8	DMAn_RX_IT_src	DMA receive interrupt source (non-isochronous)
7–4	–	Reserved
3–0	DMAn_TX_IT_src	DMA transmit interrupt source (non-isochronous)

13.2.14.1 DMA Receive Single Byte (DMAn_RX_SB)

Only concerns non-isochronous endpoints (isochronous endpoints receive a constant number of bytes).

This bit is set when the RXn_EOT interrupt is asserted and the core receives an odd number of bytes during the last transaction. This bit determines the exact number of bytes received in case of a 16-bit read access from DATA_DMA register. When the RXn_EOT flag is cleared, this bit read as 0.

0: No EOT DMA interrupt is pending, or core received an even number of bytes during last transaction.

1: An EOT DMA interrupt is pending, and an odd number of bytes was received during last transaction.

Value after local host or USB reset is low.

13.2.14.2 DMA Receive Interrupt Source (DMAn_RX_IT_src)

Only concerns non-isochronous endpoints.

When the EPn_RX flag bit is set, the endpoint causing this flag to be set is encoded in these four register bits. When the EPn_RX flag bit is cleared, the four bits read as 0.

0000: No receive DMA interrupt is pending.

0001: EP1

….

1111: EP15

Value after local host or USB reset is low (all 4 bits).

13.2.14.3 DMA Transmit Interrupt Source (DMAn_TX_IT_src)

Only concerns non-isochronous endpoints.

When the EPn_TX flag is set, the endpoint causing this flag to be set is encoded in these four register bits. When the EPn_TX flag is cleared, the four bits read as 0.

0000: No transmit DMA interrupt is pending.

0001: EP1

….

1111: EP15

Value after local host or USB reset is low (all 4 bits).

13.2.15 Receive DMA Channels Configuration Register (RXDMA_CFG)

The read/write receive DMA channels configuration register (RXDMA_CFG) 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 the Cfg_Lock bit is set.

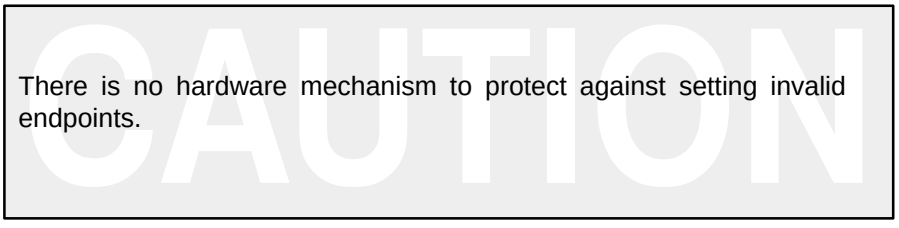


Table 13–16. Receive DMA Channels Configuration Register (RXDMA_CFG)

Bit	Name	Description
15–12	–	Reserved
11–8	RXDMA2_EP	Receive endpoint number for DMA channel 2
7–4	RXDMA1_EP	Receive endpoint number for DMA channel 1
3–0	RXDMA0_EP	Receive endpoint number for DMA channel 0

13.2.15.1 Receive Endpoint Number for DMA Channel 2 (RXDMA2_EP)

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

Value after local host or USB reset is low (all 4 bits).

13.2.15.2 Receive Endpoint Number for DMA Channel 1 (RXDMA1_EP)

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

Value after local host or USB reset is low (all 4 bits).

13.2.15.3 Receive Endpoint Number for DMA Channel 0 (RXDMA0_EP)

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

Value after local host or USB reset is low (all 4 bits).

13.2.16 Transmit DMA Channels Configuration Register (TXDMA_CFG)

The read/write transmit DMA channels configuration register (TXDMA_CFG) enables the three possible DMA transmit channels and selects the endpoint number that is assigned to each of these DMA channels. An endpoint used by a TX DMA channel must have been configured through register EPn_TX. TXDMA_CFG register can be filled when the Cfg_Lock bit is set.

> **CAUTION**
>
> There is no hardware mechanism to protect against setting invalid endpoints.

Table 13–17. Transmit DMA Channels Configuration Register (TXDMA_CFG)

Bit	Name	Description
15–12	–	Reserved
11–8	TXDMA2_EP	Transmit endpoint number for DMA channel 2
7–4	TXDMA1_EP	Transmit endpoint number for DMA channel 1
3–0	TXDMA0_EP	Transmit endpoint number for DMA channel 0

13.2.16.1 Transmit Endpoint Number for DMA Channel 2 (TXDMA2_EP)

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

Value after local host or USB reset is low (all 4 bits).

13.2.16.2 Transmit Endpoint Number for DMA Channel 1 (TXDMA1_EP)

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

Value after local host or USB reset is low (all 4 bits).

13.2.16.3 Transmit Endpoint Number for DMA Channel 0 (TXDMA0_EP)

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

Value after local host or USB reset is low (all 4 bits).

13.2.17 DMA FIFO Data Register (DATA_DMA)

The DMA FIFO data register (DATA_DMA) 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 13–18. DMA FIFO Data Register (DATA_DMA)

Bit	Name	Description
15–0	DATA_DMA	DMA FIFO data

13.2.17.1 DMA FIFO Data(DATA_DMA)

When an RX DMA request is active for a channel (only one active at a given time), this register contains the data received by the core from 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.

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.

The local host must not access this register directly; however, there is no hardware mechanism to protect from such access. Do not attempt access into this register during DMA request handling.

13.2.18 Transmit DMA Control Registers (TXDMA0...TXDMA2)

The read/write transmit DMA control registers (TXDMA...TXDMA2) control the operation of the transmit DMA channel n (n = 0, 1, 2).

Table 13–19. Transmit DMA Control Registers (TXDMA...TXDMA2)

Bit	Name	Description
15	TXn_EOT	Transmit DMA channel n end of transfer
14	TXn_Start	Transmit DMA channel n start
13–10	–	Reserved
9:0	TXn_TSC	Transmit DMA channel n transfer size counter

13.2.18.1 Transmit DMA Ch.n End of Transfer (TXn_EOT)

This bit can be either 0 or 1 for BULK DMA transfer.

When set to 1 by the local host, it signals to the core that the transfer size set in TXn_TSC is in bytes. A TX done interrupt (the TXn_Done) is asserted with the last IN transaction. If the number of bytes set in 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 local host or USB reset is low.

13.2.18.2 Transmit DMA Ch.n Start (TXn_Start)

Set by the local host to tell the device that the main DMA system is ready to transmit the number of bytes or buffers. Once set, the DMA transfer cannot be interrupted, except if the local host clears endpoint in TXDMA_CFG register (see part 8.8). Writing 0 to this bit has no effect and a read of this bit always returns 0. The TXn_Done interrupt bit is asserted when the DMA transfer ends.

0: No action

1: DMA transfer start

Always reads 0.

13.2.18.3 Transmit DMA Ch.n Transfer Size Counter (TXn_TSC)

The binary-encoded value from 0 to 1023, which is written by the local host into this register, corresponds to the number of bytes or number of buffer transfers (function of TXn_EOT), which is transmitted by the transmit DMA channel n. When read, the register reflects the number of bytes/buffers the USB device has left to transmit. This read mode is only provided for software debug purposes.

> For isochronous transfer, the user must verify that the set value does not exceed the isochronous FIFO size for the endpoint. There is no hardware mechanism to protect from this situation. If it happens, results are unpredictable.
>
> For bulk transfer, when TXn_EOT = 0 a set value of TXn_TSC = 0 means 1024 buffers and not 0. The counter then operates in the following way: 000, 3FF, 3FE, …001, 000, stop. When TXn_EOT = 1, a set value of TXn_TSC = 0 a NULL packet is sent in response to the next IN token.

Value after local host or USB reset is low (all 10 bits).

13.2.19 Receive DMA Control Registers (RXDMA...RXDMA2)

These read/write receive DMA control registers enable monitoring of incoming OUT transactions during DMA transfer on channel n (n=0,1,2).

Table 13–20. Receive DMA Control Registers (RXDMA0...RXDMA2)

Bit	Name	Description
15	RXn_Stop	Receive DMA channel n transfer stop
14–8	–	Reserved
7–0	RXn_TC	Receive DMA channel n transactions count

13.2.19.1 Receive DMA Ch.n Transfer Stop (RXn_Stop)

When this bit is set, an RXn_EOT interrupt is asserted to the local host after n OUT transactions where n is the encoded binary value + 1 programmed into RXn_TC field. This register is used when no smaller than buffer size packet is received at an end-of-transfer (EOT) and the local host expects a given amount of data for the transfer.

> 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 local host must set the Set_FIFO_En for the endpoint to reenable the channel.

Value after local host or USB reset is low.

13.2.19.2 Receive DMA Ch.n Transactions Count (RXn_TC)

The local host can ask for an interrupt each n OUT transactions where n is the encoded binary value + 1 programmed into RXn_TC field. This register must be programmed to the desired transactions watermark limit prior to enabling the DMA transfer for the receive DMA channel n.

> A reached watermark does not disable an active DMA transfer if RXn_Stop was not set. If RXn_Stop was set for the transfer, both RXn_Cnt and RXn_EOT interrupts are asserted.
>
> A read of this register returns the number of transactions remaining before the RXn_Cnt interrupt flag is asserted. This read mode is only provided for software debug purposes.

Value after local host or USB reset is low (all 8 bits).

13.2.20 Endpoint 0 Configuration Register (EP0)

The read/write endpoint 0 configuration register (EP0) gives the device configuration for control endpoint 0.

Table 13–21. Endpoint 0 Configuration Register (EP0)

Bit	Name	Description
15–14	–	Reserved
13–12	EP0_Size	Endpoint 0 FIFO size
11	–	Reserved
10–0	EP0_ptr	Endpoint 0 pointer

13.2.20.1 Endpoint 0 FIFO Size (EP0_Size)

This field contains the endpoint 0 FIFO size value and must match the value sent by the local host to the USB host during the GET_DEVICE_ DESCRIPTOR request preceding configuration phase. Status flags (the Non_ISO_FIFO_Empty, the Non_ISO_FIFO_Full) and overrun/underrun conditions are based on this value for all IN and OUT transactions to endpoint 0.

The local host must fill this field before setting the Cfg_Lock bit.

00: 8 bytes

01: 16 bytes

10: 32 bytes

11: 64 bytes

Value after local host reset is low (both bits), after USB reset is unchanged.

13.2.20.2 Endpoint 0 Pointer (EP0_ptr)

This field contains the address of the endpoint 0 pointer. Value 0x000 is forbidden (reserved for setup FIFO).

0x000: address = BASE (forbidden)

0x001: address = BASE + 8 bytes

0x002: address = BASE + 16 bytes

0x003: address = BASE + 24 bytes

...

0x0FF: address = BASE + 2040 bytes

Value after local host reset is low (all bits), after USB reset is unchanged.

Set the pointer value higher than 0xFF, because the memory size is 2K bytes. A pointer value equal to 0xFF corresponds to 2040 bytes: addressing upper bytes results in memory overlap (see Section 13.4, *Device Initialization*).

13.2.21 Receive Endpoint Configuration Registers (EP1_RX...EP15_RX)

The read/write receive endpoint configuration registers (EP1_RX...EP15_RX) give the device configuration for non-control receive endpoint n (n: 115). The endpoints size fields must match values sent by the local host to the USB host in response to the GET_CONFIGURATION_DESCRIPTOR during configuration phase.

The local host must fill this field before setting the Cfg_Lock bit and must not change the values once Cfg_Lock bit is set.

Table 13–22. Receive Endpoint n Configuration Registers (EP1_RX...EP15_RX)

Bit	Name	Description
15	EPn_RX_Valid	Receive endpoint n valid
14	EPn_RX_Size/Db	Receive non-isochronous endpoint n double-buffer (Db) Or receive isochronous endpoint n size[2]
13–12	EPn_RX_Size	Receive endpoint n size
11	EPn_RX_Iso	Receive isochronous endpoint n
10–0	EPn_RX_ptr	Receive endpoint n pointer

13.2.21.1 Receive Endpoint n Valid (EPn_RX_Valid)

This bit must be set by the local host to allow receive endpoint n to be used for USB transfers as part of the device configuration. If not set, all transactions to this endpoint are ignored by the core.

1: Receive endpoint n is part of the device configuration.

0: Receive endpoint n does not exist for this configuration.

Value after local host reset is low, after USB reset is unchanged.

13.2.21.2 Receive Endpoint n Double-Buffer (EPn_RX_Db)

This bit is only for non-isochronous endpoints. For isochronous endpoints, which are always double-buffered, this bit is endpoint size MSB.

This bit must be set by the local host to allow double buffering for receive non-isochronous endpoint n. This is used to reduce number of transactions resulting in NAK handshake.

1: Double buffer used for non-isochronous receive endpoint n.

0: No double buffer for non-isochronous receive endpoint n.

Value after local host reset or USB reset is unchanged.

13.2.21.3 Receive Endpoint n Size (EPn_RX_Size)

This paragraph includes description of EPn_RX.[14] bit for isochronous endpoints.

This field contains the endpoint n FIFO size value. Status flags (the Non_ISO_FIFO_Empty, the Non_ISO_FIFO_Full, the ISO_FIFO_Empty, the ISO_FIFO_Full) and overrun and underrun conditions are based on this value for all OUT transactions to endpoint n (see Table 13–23).

Table 13–23. Endpoint n Size Values

Non-Isochronous [13:12]	Isochronous [14:12]
00: 8 bytes	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
	Reserved

Value after local host reset or USB reset is unchanged.

13.2.21.4 Receive Isochronous Endpoint n(EPn_RX_Iso)

This 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.

Value after local host reset or USB reset is unchanged.

13.2.21.5 Receive Endpoint n Pointer (EPn_RX_ptr)

This field contains the address of the receive endpoint n pointer. Value 0x000 is forbidden (reserved for setup FIFO).

> CAUTION
>
> For isochronous endpoints or for non-isochronous endpoints that allow double-buffering, 2*RX buffer size must be reserved for ping-pong.

0x000: address = BASE (forbidden)

0x001: address = BASE + 8 bytes

0x002: address = BASE + 16 bytes

0x003: address = BASE + 24 bytes

...

0x0FF: address = BASE + 2040 bytes

Value after local host reset or USB reset is unchanged.

Set the pointer value higher than 0xFF, because the memory size is 2K bytes. A pointer value equal to 0xFF corresponds to 2040 bytes: addressing upper bytes results in memory overlap (see Section 13.4, *Device Initialization*).

13.2.22 Transmit Endpoint Configuration Registers (EP1_TX...EP15_TX)

The read/write transmit endpoint configuration registers (EP1_TX... EP15_TX) configure the device for noncontrol transmit endpoint n (n: 115). The endpoint size fields must match the values sent by the local host to the USB host in response to the GET_CONFIGURATION_DESCRIPTOR during configuration phase.

The local host must fill this field before setting the Cfg_Lock bit and must not change the values once the Cfg_Lock bit is set.

Table 13–24. Transmit Endpoint Configuration Registers (EP1_TX...EP15_TX)

Bit	Name	Description
15	EPn_TX_Valid	Transmit endpoint n valid
14	EPn_TX_Size/Db	Transmit non-isochronous endpoint n double-buffer or transmit isochronous endpoint n size[2]
13–12	EPn_TX_Size	Transmit endpoint n size
11	EPn_TX_Iso	Transmit isochronous endpoint n
10–0	EPn_TX_ptr	Transmit endpoint n pointer

13.2.22.1 EPn_TX[15].EPn_TX_Valid: Transmit Endpoint n Valid

This bit must be set by the local host to allow transmit endpoint n to be used for USB transfers as part of the device configuration. If not set, all transactions to this endpoint are ignored by the core.

1: Transmit endpoint n is part of the device configuration.

0: Transmit endpoint n does not exist for this configuration.

Value after local host reset is low, after USB reset is unchanged.

13.2.22.2 Transmit Endpoint n Double-Buffer(EPn_TX_Db)

This bit is only for non-isochronous endpoints used in DMA mode. For isochronous endpoints, which are always double buffered, this bit is the endpoint size MSB. For non-isochronous endpoints which are not used in a DMA transfer, double-buffering is not provided.

This bit must be set by the local host to allow double buffering for transmit non-isochronous endpoint n, when used in a DMA transfer. This is used to reduce number of transactions resulting in NAK handshake.

1: Double buffer used for non-isochronous transmit endpoint n.

0: No double buffer for non-isochronous transmit endpoint n.

Value after local host or USB reset is unchanged.

13.2.22.3 Transmit Endpoint n Size (EPn_TX_Size)

EPn_TX.[14] bit description only applies for isochronous 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 (see Table 13–23, *Endpoint n Size Values*).

Value after local host reset or USB reset is unchanged.

13.2.22.4 Transmit Isochronous Endpoint n (EPn_TX_Iso)

This 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.

Value after local host or USB reset is unchanged.

13.2.22.5 Transmit Endpoint n Pointer (EPn_TX_ptr)

This field contains the address of the transmit endpoint n pointer.

> **CAUTION**
>
> For isochronous endpoints or for non-isochronous 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

Value after local host reset or USB reset is unchanged.

Set the pointer value higher than 0xFF, because the memory size is 2K bytes. A pointer value equal to 0xFF corresponds to 2040 bytes: addressing upper bytes results in memory overlap (see Section 13.4, *Device Initialization*).

13.3 USB Transactions

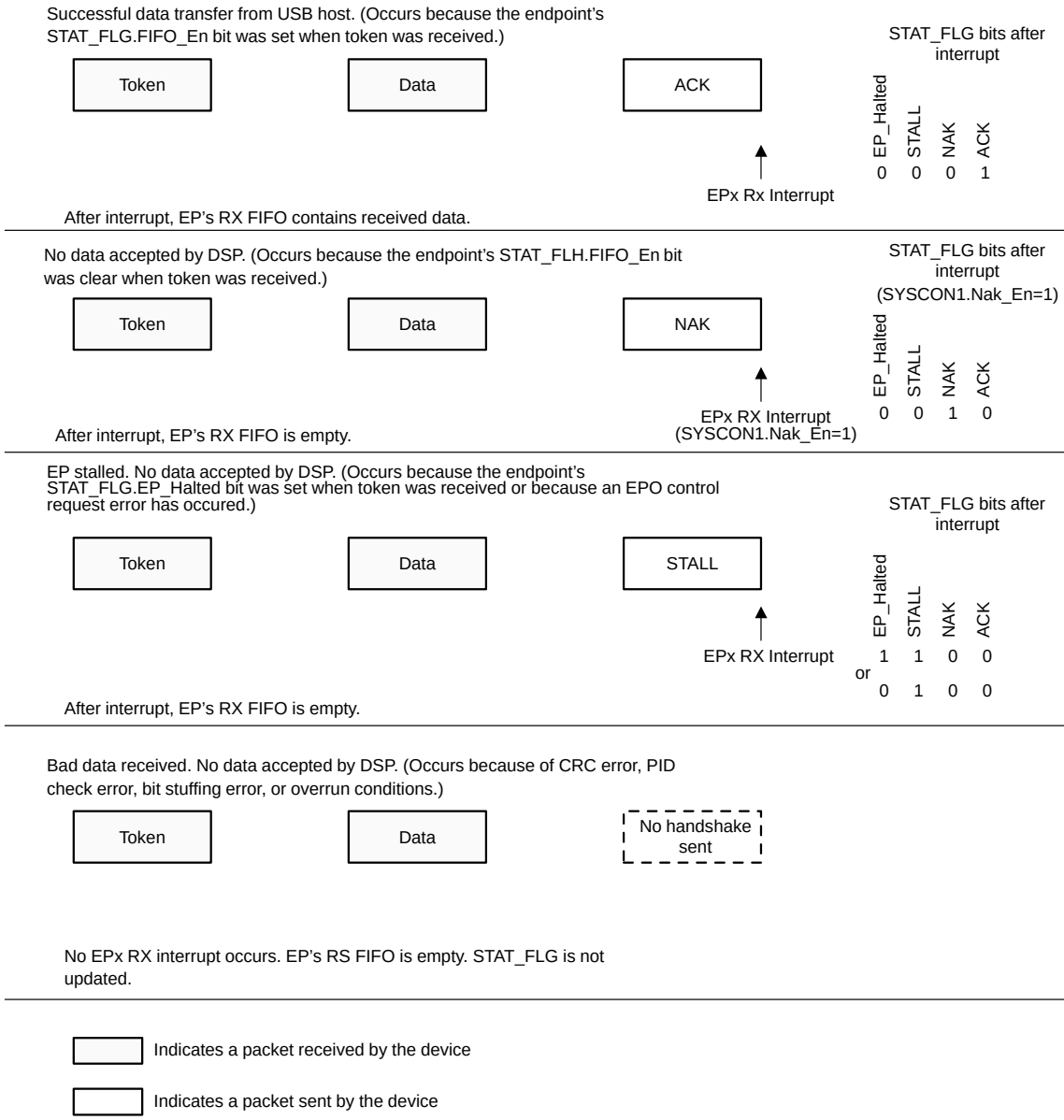
There is an interrupt to the local host at the end of a USB transaction if the local host has actions to perform. Isochronous transactions are an exception, because isochronous interrupt information is available at start of frame interrupts. The local host ISR code determines which endpoint and direction caused the interrupt and acts appropriately. The following sections 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 local host are not taken into account in this section. 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, etc. 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.

13.3.1 Non-Isochronous, Non-Setup OUT (USB HOST –> LH) Transactions

Non-isochronous, non-setup OUT transactions refer to USB transactions where data is moved from the USB host to the local host and where the USB handshaking protocols are in effect and data transmission is guaranteed. These types of transactions apply to all OUT transactions on bulk and interrupt endpoint types, and to non-setup transactions on control endpoints.

Figure 13–3 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: ACK (good data received), NAK (device not ready to receive data), and STALL (device in a condition where the endpoint cannot handle OUT transactions). The last case shows an abnormal case where the token packet or the data packet was received with errors. The RX FIFO only contains valid receive data under the first, ACK, case.

Figure 13–3. Non-Isochronous, Non-Control OUT Endpoint Handshaking Conditions



13.3.1.1 Non-Isochronous, Non-Control OUT Endpoint Handshaking Conditions

The Set_FIFO_En bit provides the main control for the ability to allow successful OUT transaction data reception for the endpoint. If at the beginning of an OUT transaction to an endpoint the 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 USB host to indicate that the data was received correctly (this is the top case shown in Figure 13–3). If, however, the 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 13–3).

It is important to note that 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 local host code must be careful not to read too much data. Local host code must read the RXF_Count value before reading data from the RX FIFO.

After a USB OUT transaction to an endpoint where the data is accepted (ACKed), the hardware clears the endpoint's FIFO_En bit. Once the local host 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 Set_FIFO_En bit. Local host software can use the 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 local host and the STAT_FLG is updated. In response to the endpoint interrupt, the local host must read EPN_STAT register to identify the endpoint causing the interrupt, then write a 1 to the interrupt bit to clear it. The local host must set EP_Num to the endpoint number and EP_Sel to 1, then read the endpoint status. The ACK bit is set to indicate that the endpoint received a transaction to which the USB module signaled ACK handshaking.

If the FIFO_Empty is cleared, the host 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 local host knows the number of bytes to read from RX FIFO by reading the RXF_Count value. The local host can then read RX data from DATA register. Once the local host has read the data from the FIFO, it sets the 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_Sel bit. This clears the ACK bit for this endpoint and allows the next transaction status to be written to the STAT_FLG register.

Non-Acknowledged Transactions (NAK)

The device can be configured via the Nak_En bit, either to inform the local host of a NAKed transaction or not. If the NAK_EN bit is cleared, no interrupt is asserted to the local host if an OUT transaction completes with a NAK handshake and the NAK bit not set. If the Nak_En bit is set, the USB module issues an endpoint-specific interrupt to the local host at completion of an OUT transaction to an endpoint and the NAK bit is set. In response to the endpoint interrupt, the local host must read EPN_STAT register to identify the endpoint causing the interrupt then write a 1 to the interrupt bit to clear it. The local host must set EP_Num to the endpoint number and EP_Sel to 1 then read the endpoint status. The NAK bit is set to indicate that the endpoint received a transaction to which the USB module signaled NAK handshaking.

The local host must set the 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_Sel bit. This clears the NAK bit for this endpoint and allows the next transaction status to be written to the STAT_FLG register.

13.3.1.2 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's 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's STALL bit is set and an endpoint-specific interrupt is generated for the endpoint. The FIFO_En bit is of lower priority than the EP_Halted; when the EP_Halted bit is set, transactions to the RX endpoint are stalled, regardless of the FIFO_En value. If the FIFO_En bit is set, the 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 local host must read EPN_STAT register to identify the endpoint causing the interrupt, then write a 1 to the interrupt bit to clear it. The local host must set EP_Num to the endpoint number and EP_Sel to 1, then read the endpoint status. The STALL bit is set to indicate that the endpoint received a transaction to which the USB module signaled STALL handshaking.

If the EP_Halted has been set by the local host and can be removed, the local host must set the Clr_Halt to clear the condition and set the Set_FIFO_En to allow the next USB OUT transaction to the endpoint to be placed into the RX

FIFO. If the 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 local host has no action to perform and must clear the EP_Sel bit. This clears the STALL bit for this endpoint and allows the next transaction status to be written to 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 local host (the fourth case shown in Figure 13–3). Additionally, the endpoint RX FIFO is not filled and the FIFO_En bit is not cleared. If the local host 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 an ACK handshake to the USB host, regardless of the FIFO_En bit (per USB spec). Data is ignored, and no interrupt is asserted to the local host.

This error occurs if an ACK handshake from the previous OUT transaction is received corrupted by the USB host.

13.3.1.3 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 local host. Additionally, the endpoint RX FIFO is not filled, and the FIFO_En bit is not cleared.

The local host must not read more data from the RX FIFO than the amount indicated by RXF_Count.

13.3.2 Non-Isochronous IN (LH–>USB HOST) Transactions

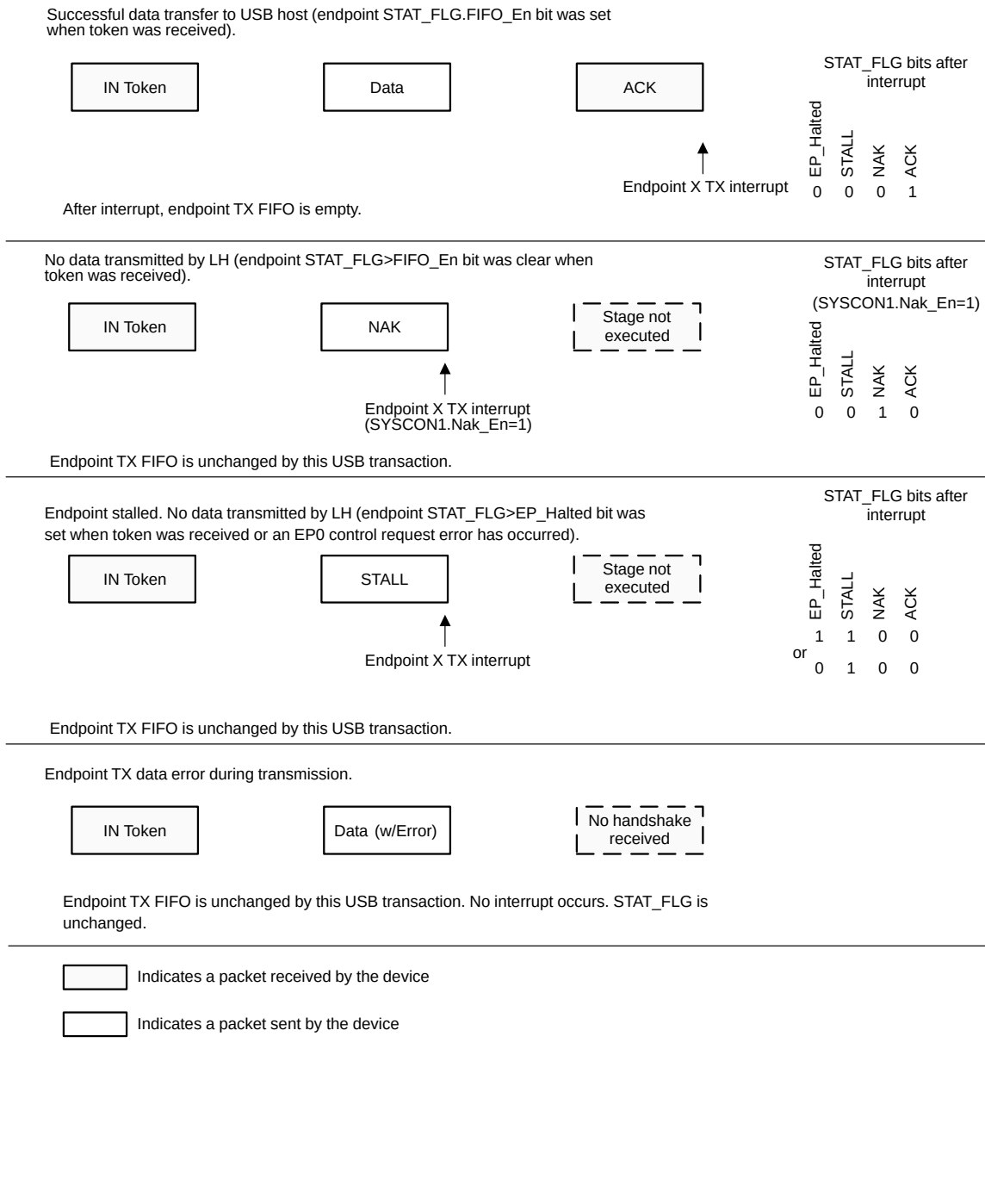
Non-isochronous IN transactions refer to USB transactions where data is moved from the local host to the USB host where the USB handshaking protocols are in effect and data transmission is guaranteed. These transactions are the IN transactions that occur on control, bulk, and interrupt endpoints. These transactions do not guarantee USB bandwidth.

To provide data for an endpoint IN transaction, the local host code writes the transmit data into the endpoint transmit FIFO. Local host 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's 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 the FIFO_En bit is set when an IN transaction to the endpoint occurs, a 0-byte data packet is sent.

Once the endpoint's previous transmit activity is taken care of, the local host code gains access to endpoint's FIFO and status by setting EP_Sel bit. Then the local host 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, local host code sets the Set_FIFO_En bit to allow the USB to use the endpoint's TX FIFO and then clears the 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 13–4 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 interrupts are generated, and the resulting STAT_FLG bits for the endpoint. The top three cases show the normal USB handshaking: ACK (data sent by USB module and received properly by the USB host), NAK (device not ready to send data to USB host), and STALL (device in a condition where the endpoint cannot handle IN transactions). The last case shows an abnormal case where there is an error either in the token packet received by the core, or in the data packet received by the USB host.

Figure 13–4. Non-Isochronous IN Transaction Phases and Interrupts



13.3.2.1 Non-Isochronous IN Endpoint Handshaking

Per the USB spec for IN transactions, the USB host may only provide 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 13–4), and the second indicates that the host received a garbled data packet (last case shown in Figure 13–4).

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 local host (see first case in Figure 13–4). In response to the endpoint interrupt, the local host must read EPN_STAT register to identify the endpoint causing the interrupt, then write a 1 to the interrupt bit to clear it. The local host mustset EP_Num to the endpoint number, EP_Dir to 1 (to signal an IN endpoint), and EP_Sel to 1, then read the endpoint status. The ACK bit is set to indicate that the endpoint received an ACK handshake from the USB host and that the TX FIFO is empty (because any data that was in the TX FIFO was transmitted during the IN transaction).

If the local host has more data to transmit to the USB host, it must fill the TX FIFO following the process indicated above. It must then clear the EP_Sel bit. This clears the ACK bit for this endpoint and allows the next transaction status to be written to the STAT_FLG register.

Non-must Transactions (NAK)

For the case where the local host 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 (as shown in the second case in Figure 13–4). Readiness to transmit data is signaled via the endpoint's FIFO_En bit; when 1, it indicates that data in the TX FIFO can be sent to the USB host. When the endpoint's FIFO_En bit is 0 and an IN transaction to the endpoint occurs, a NAK handshake is sent, indicating that the local host is not ready to handle the request.

If the 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 local host is generated. If the Nak_En bit is set when the NAK handshake is sent in the data packet portion of the transaction to the IN endpoint, the NAK bit is set and an endpoint-specific interrupt to the local host is generated.

In response to the endpoint interrupt, the local host must read the EPN_STAT register to identify the endpoint causing the interrupt, then write a 1 to the interrupt bit to clear it. The local host must set EP_Num to the endpoint number, EP_Dir to 1 (to signal an IN endpoint), and EP_Sel to 1, then read the endpoint status. The NAK bit is set to indicate that the endpoint sent a NAK handshake to the USB host. If the local host has data to transmit to the USB host, it must fill the TX FIFO following the process indicated above. The local host must then clear the EP_Sel bit. This clears the NAK bit for this endpoint and allows the next transaction status to be written to the STAT_FLG register. Signaling NAK does not cause the endpoint's TX FIFO to be cleared (since the local host 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 may not be a good mechanism in cases where the local host is not able to service a request for long periods of time.

13.3.2.2 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's EP_Halted flag bit is set (as shown in the third case in Figure 13–4) or if a request error occurs (control transaction only). A STALL handshake indicates that the device endpoint is in a condition where it is not able to transfer data and that the USB host must not retry the transaction. The device typically requires intervention via some other mechanism to clear the condition, typically a control transfer via endpoint 0. The local host can set the endpoint EP_Halted bit by writing the appropriate value in the EP_NUM register to select it. It can then set the endpoint's Set_Halt bit and clear it by selecting the endpoint and setting the endpoint's Clr_Halt bit. When the endpoint EP_Halt bit 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 STALL bit is set, and an endpoint-specific interrupt to the local host is generated.

In response to the endpoint interrupt, the local host must read EPN_STAT register to identify the endpoint causing the interrupt, then write a 1 to the interrupt bit to clear it. The local host must set EP_Num to the endpoint number, EP_Dir to 1 (to signal an IN endpoint), and EP_Sel to 1, then read the endpoint status. The STALL bit is set to indicate that the endpoint sent a STALL handshake to the USB host. The local host must then clear the EP_Sel bit. This clears the STALL bit for this endpoint and allows the next transaction status to be written to 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 local host occurs for transactions with corrupted packets. If the local host 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 (case of error during transmission), the USB function module detects after a time-out that an error has occurred. The data to transmit is still in the TX FIFO, and can be resent during next IN transaction, the FIFO_En bit is not cleared, and no interrupt is asserted to the local host.

13.3.2.3 Non-Isochronous IN Endpoint FIFO Error Conditions

The local host cannot write more data into the TX FIFO than the configured FIFO size.

13.3.3 Isochronous OUT (USB HOST–> LH) Transactions

Isochronous OUT transactions are USB transactions where a given amount of data is transferred from the USB host to the USB function module device every 1-ms USB frame. No USB handshaking is provided, and no endpoint-specific interrupt to the local host is generated at completion of an isochronous OUT transaction. The local host is responsible for handling isochronous OUT data at each start of frame (SOF) interrupt.

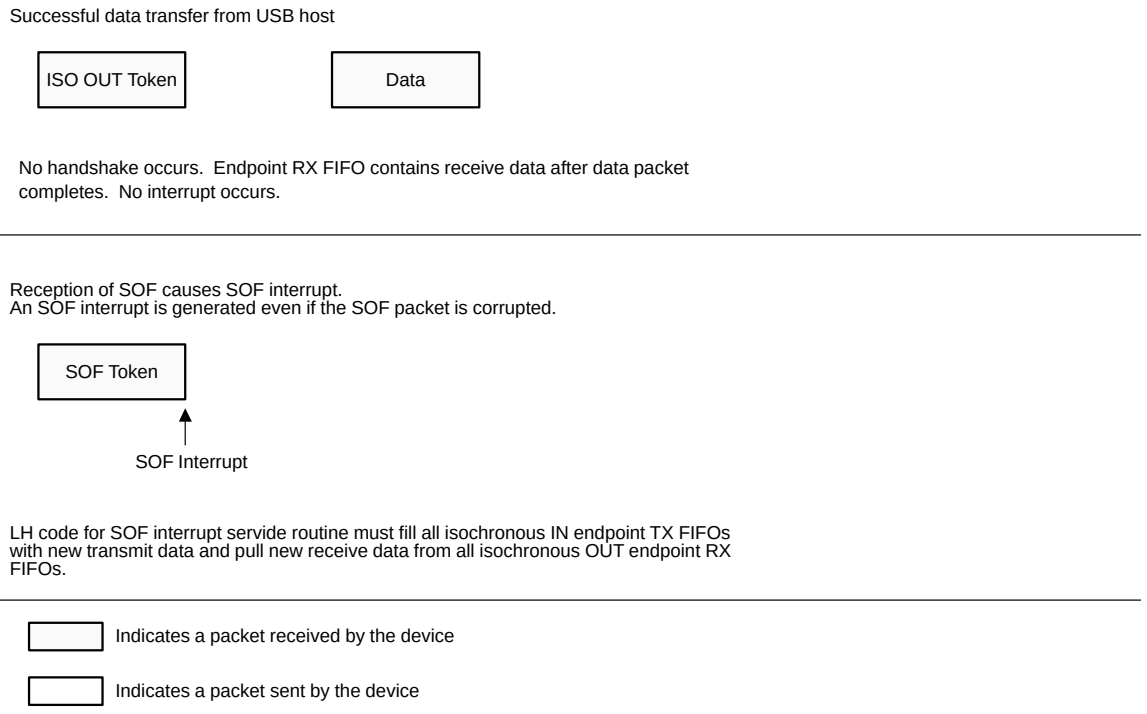
At every SOF interrupt, the local host code must select the endpoint for each isochronous OUT endpoint by writing the appropriate value into the EP_NUM register and checking the ISO_FIFO_Empty bit. If the RX FIFO contains data, code must read the RXF_Count value (if the number of bytes to read from RX FIFO is not known), read all the bytes from RX FIFO via the DATA register, then clear the EP_Sel bit.

Because the USB transaction for the isochronous endpoint can occur at any time during the 1-ms USB frame, the USB interface implements 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 local host 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, while the foreground FIFOs are enabled to the local host.

Figure 13–5 shows the two phases (isochronous OUT token and data) of an isochronous OUT data transfer in the top portion of the figure. No endpoint-specific interrupt to the local host is generated for the isochronous OUT transaction. The data for isochronous endpoints are instead handled by the local host at each start of frame (SOF) interrupt, which is shown as the second case in Figure 13–4.

Figure 13–5. Isochronous OUT Transaction Phases and Interrupts



13.3.3.1 Isochronous OUT Endpoint Handshaking

Because isochronous endpoint transactions have no handshake packets, the STALL, the NAK, and the ACK bits for isochronous endpoints always return 0. Because there is no handshake, the endpoint-specific interrupt for isochronous endpoints is not used.

13.3.3.2 Isochronous OUT Transaction Error Conditions

If the local host fails to read all of the data in the isochronous OUT endpoint foreground FIFO by the time that the foreground and background FIFOs are switched (at the next SOF), the endpoint FIFO that is being switched to the background is flushed and the Data_Flush bit is asserted for the duration of the next frame.

There is no special indication for the case where the USB host does not provide a transaction to an isochronous 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 isochronous OUT transaction also results in an empty FIFO and can not be distinguished from a missed isochronous OUT transaction.

If an isochronous OUT transaction occurs with data error (CRC, PID check, bit stuffing), the RX FIFO is empty at the next SOF interrupt and the ISO_Err bit is asserted for the duration of the next frame.

13.3.3.3 Isochronous OUT Endpoint FIFO Error Conditions

The local host must never read more data than value given by the RXF_Count.

If the USB host sends more data than the FIFO can contain, the FIFO is cleared and the ISO_Err bit is set at the next SOF interrupt. A properly configured USB system does not do this.

Both foreground and background isochronous FIFOs are cleared when the Clr_EP bit is set.

13.3.4 Isochronous IN (LH–>USB HOST) Transactions

Isochronous IN transactions are USB transactions where a given amount of data is transferred from the USB function module device to the USB host every 1-ms USB frame. No handshaking is provided.

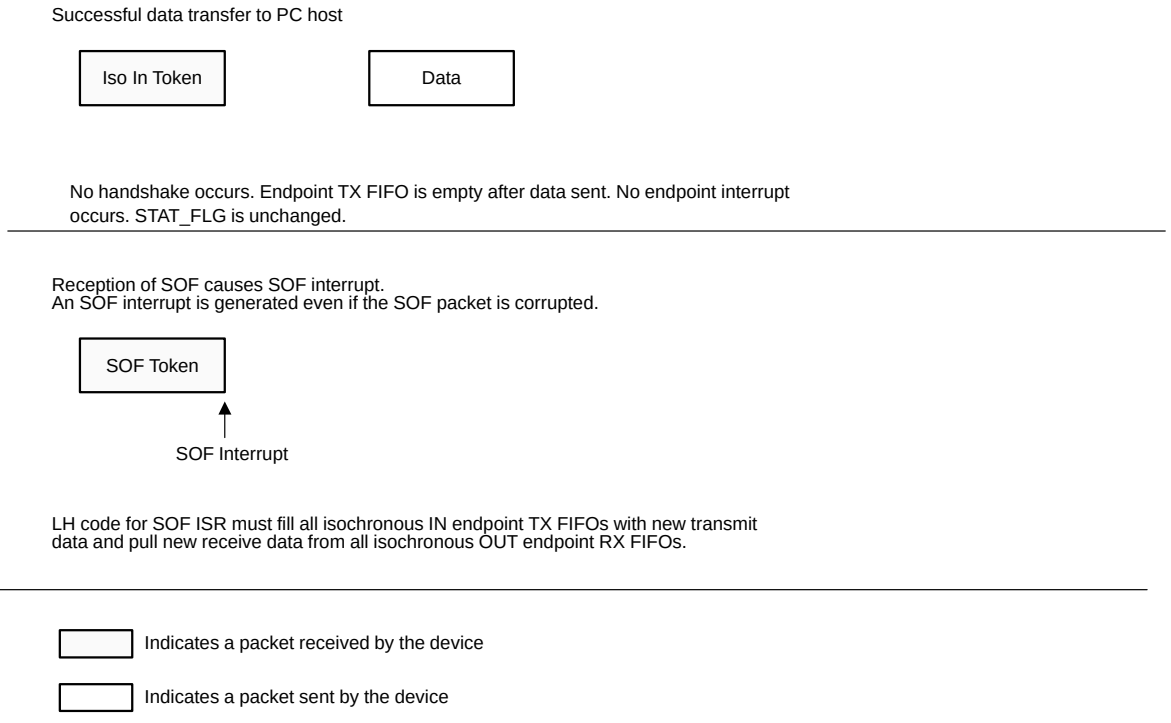
The USB module provides double buffering of data for isochronous IN endpoints; the background FIFO is used as the source of data for IN transactions to the isochronous endpoint, and the foreground FIFO can be written to by the local host. When an IN transaction to an isochronous endpoint occurs, the USB module sends all data found in the endpoint background TX FIFO. The local host 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, local host code selects the endpoint (via EP_NUM register), then fills the endpoint TX FIFO (via DATA register). Once all the transmit data has been written to the FIFO, the local host code must clear the EP_Sel bit.

Because the USB transaction for the isochronous endpoint can occur at any time during the 1-ms USB 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 local host 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 isochronous endpoints implement double buffering, isochronous endpoints do not control access to the FIFOs via the Set_FIFO_En bit; the Set_FIFO_En and the FIFO_En bits are not implemented for isochronous IN endpoints.

Figure 13–6 shows the transaction phases associated with isochronous IN transactions and the SOF transaction. No endpoint-specific interrupt to the local host is generated as a result of an isochronous IN transaction. There is no handshake phase. The SOF transaction causes an SOF interrupt to the local host; it is assumed that the local host refills the isochronous IN endpoint transmit FIFO at each SOF interrupt.

Figure 13–6. Isochronous IN Transaction Phases and Interrupts



13.3.4.1 Isochronous IN Endpoint Handshaking

Because isochronous endpoint transactions have no handshake packets, the STALL, the NAK, and the ACK bits for isochronous endpoints always return 0. Because there is no handshake, there is no endpoint-specific interrupt to the local host to report handshake results for isochronous endpoints.

13.3.4.2 Isochronous IN Transaction Error Conditions

If the USB host did not successfully complete an isochronous IN transaction in the previous frame and if data were present in the TX FIFO to be sent at the IN transaction, the Miss_In bit is asserted for the duration of the following frame. If the isochronous IN endpoint is cleared in the middle of a USB transaction to the background FIFO, a bit-stuffing error is forced for the isochronous transaction.

13.3.4.3 Isochronous IN Endpoint FIFO Error Conditions

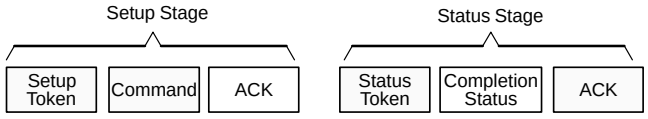
If the local host attempts to overfill the configured endpoint FIFO, data written to the DATA register after the TX FIFO is full is lost, but any data that was successfully put into the FIFO is transmitted when that FIFO is the background FIFO and an IN transaction for that endpoint occurs. Since an isochronous TX FIFO is cleared automatically on the toggle from background to foreground, there is no reason to clear the FIFO. However, if the local host does not wish to send the data it wrote, clearing the endpoint is the only mechanism to do this.

13.3.5 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 function module is capable of autodecoding some control write and control read transfers. These operations are summarized in Figure 13–7 and Figure 13–8. If an IN or OUT transaction is received in a control request, this transaction is automatically stalled by the core.

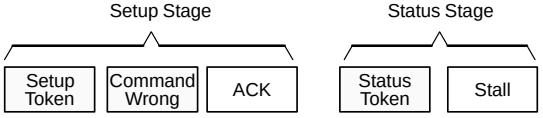
Figure 13–7. Stages and Transaction Phases of Autodecoded Control Transfers

Autodecode control write transfer–correct status
(set address, clear/set device/interface feature).



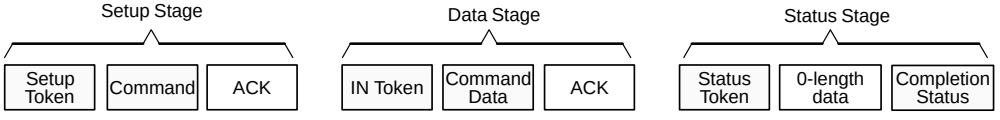
Interrupt occurs/status flags are not updated.

Autodecode control read transfers-–request error
(due to wrong setup data).



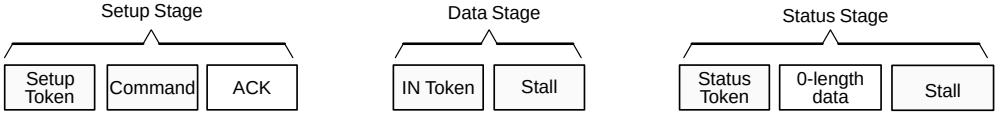
Interrupt occurs/status flags are not updated.

Autodecode control read transfers-–correct status
(get device/endpoint status)



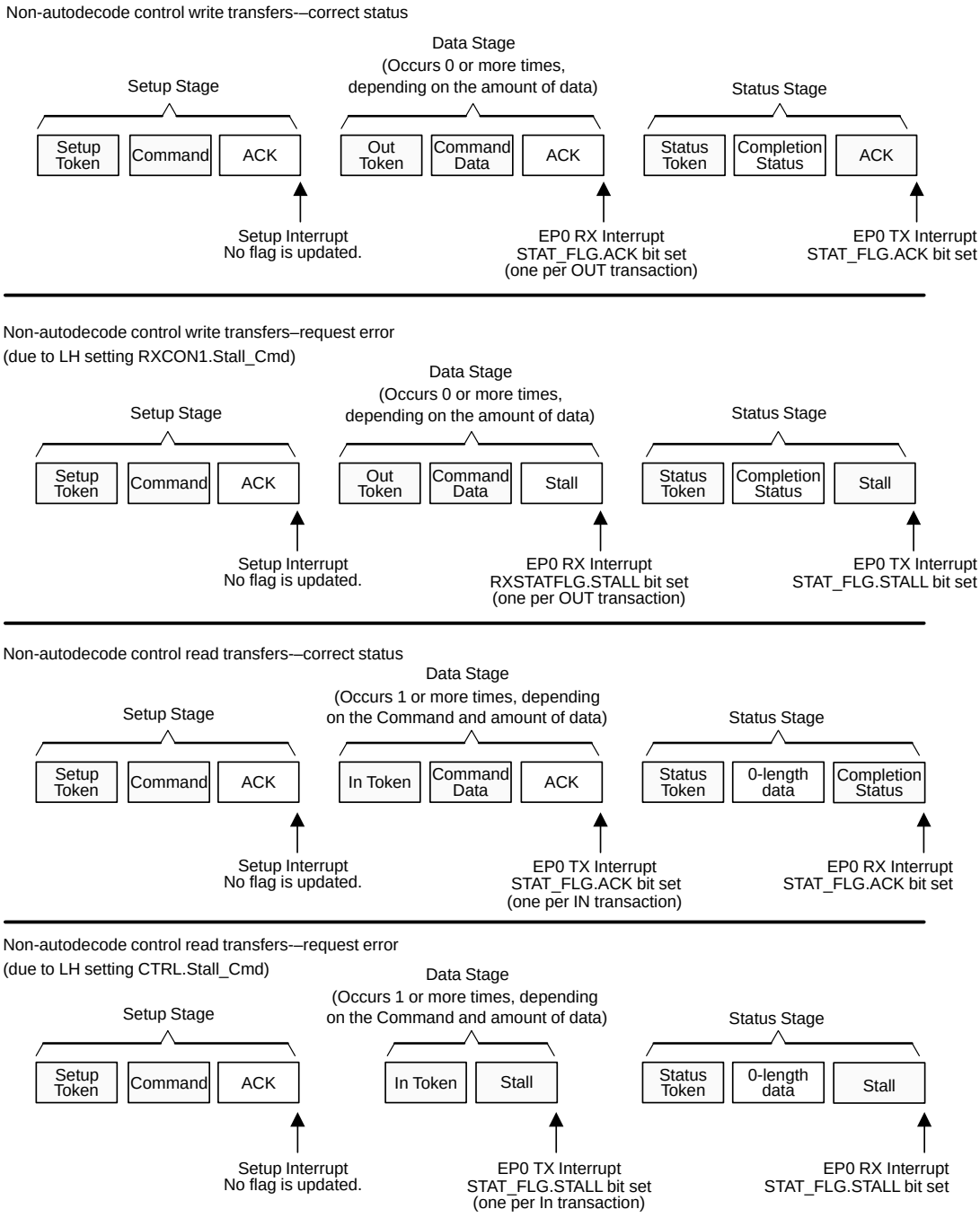
Interrupt occurs/status flags are not updated.

Autodecode control read transfers-–request error
(due to wrong setup or command data)



Interrupt occurs/status flags are not updated.

Figure 13–8. 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, have specific USB protocol meaning, and are not handled automatically by the core. The USB function block automatically provides an ACK handshake for the setup stage transaction, but the data and status stage transaction handshaking is accomplished using the normal RX and TX control bits that affect transaction handshaking. A general USB interrupt to the local host occurs at the end of each transaction of each stage of a control transfer. The local host must perform the following actions to act on non-autodecoded control transfers:

- ❏ Process the setup phase USB interrupt. The local host reads the control transfer command from the setup FIFO and decodes the command. For control reads, the local host fetches the requested read data and places it (or as much of the read data as will fit) into the endpoint 0 FIFO, then enables the endpoint 0 FIFO. For control writes, the local host code only enables the endpoint 0 FIFO. Local host 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 local host must set the endpoint 0 control bits to signal the desired stage status to the host.

- ❏ Process the status stage endpoint 0 general USB interrupt. The local host 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, have specific USB protocol meaning, and are handled automatically by the USB function block without any intervention by the local host. The USB function 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 local host during autodecoded control transfers.

If no request defined by the USB1.1 specification is associated with the data of the setup phase, the request is stalled by the core and the local host 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 if 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 local host (if the local host had control of the FIFO). New setup data is placed into the setup FIFO, and the setup interrupt flag is set (Setup bit).

In response to the setup interrupt, the local host must select the setup FIFO by setting Setup_sel bit. This clears the setup flag. The local host must then read 8 bytes from the setup FIFO, clear the Setup_Sel bit, and check that the setup bit has not been reset by a new setup transaction. If the setup flag is asserted, the local host must discard the previously read data and handle the new setup packet as explained above. Thus the local host never misses a new occurring setup transaction (this is per USB 1.1 specification).

13.3.5.1 Autodecoded Control Write Transfers

For set address control write transfers, the USB address provided in the setup token is captured to the USB module device address register. If the new address is different from 0, the device moves into addressed state (the ADD bit 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's remote-wakeup feature, the 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 USB 1.1 specification).

Per USB 1.1 specification, a SET_ADDRESS request is effective after the status stage of the request, even if the status stage does not end with an ACK handshake. SET/CLEAR_FEATURE requests are effective after the setup stage, even if no status stage occurs.

Autodecoded Control Write Transfer Handshaking

The USB function module automatically provides ACK handshaking for all transactions of all stages of autodecoded control write transfers, except if a corrupted packet is received, which is ignored by the USB module. The Set_FIFO_En and Stall_Cmd bits have no effect on the 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.

13.3.5.2 Autodecoded Control Read Transfers

Autodecoded control reads include the standard device request to get the endpoint and device status. These control read transfers access information that is kept in registers inside the USB module, so local host code is not involved in filling the read data into the TX FIFO.

The USB module returns the currently selected endpoint's status information (depending on the index 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 local host receives no interrupt.

Autodecoded Control Read Transfer Handshaking

The USB function 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 is ignored by the USB module. The FIFO_En and 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 local host. 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 in which 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.

13.3.5.3 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 a local host general USB interrupt with the setup flag set. The local host must respond to this general USB interrupt by setting the Setup_Sel bit, which clears the setup interrupt flag. The local host must then read 8 bytes from the setup FIFO via the DATA register, clear EP_Sel bit, and check the setup flag. If the setup flag is set, the local host must discard the setup data it has just read and handle the new setup data packet following the same scheme. If the setup flag is cleared, the local host code must interpret this request information and performs any application-specific activity needed due to the setup stage request (see Figure 13–8). If there is one or more data stages for the transfer, the local host must set the 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 apply as for non-isochronous, non-control OUT endpoints. The local host can cause NAK, STALL, or ACK signaling for the data stage transactions. If ACK was signaled on a given general USB interrupt, the local host 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 FIFO_En. The local host may delay signaling completion of the control write transfer by forcing NAK handshaking to the host during the status stage (by holding the FIFO_En bit equal to 0), or by causing ACK handshaking by setting the Set_FIFO_En bit to 1 (with an empty endpoint 0 FIFO). An endpoint 0 TX general USB interrupt is sent to the local host at completion of the status stage.

After a SET_CONFIGURATION request, the device moves into addressed or configured state as soon as the local host sets the Dev_Cfg or the Clr_Cfg bits.

Specific Local Host Required Actions

If the device receives a valid set endpoint halt feature request, it must set the appropriate Set_Halt control bit.

If the device receives a valid clear endpoint halt feature request, it must set the appropriate Reset_EP bit to clear halt condition, set FIFO flags, and reset data PID to DATA0 for the endpoint. If specified endpoint number is 0, the local host has only to set the Clr_Halt bit to clear halt condition.

If the device receives a valid set configuration request, it must reset all endpoints by setting the Reset_EP control bits, set the Self_Pwr bit to the appropriate value, and 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 local host must write 1 to the Dev_Cfg bit to allow the device to move into the configured state (the CFG bit set). If the device was configured when the set configuration was received, and new configuration value is 0, the local host must write 1 to the Clr_Cfg bit to allow the device to move back into the addressed state (the CFG bit cleared).

If the device receives a valid set interface request, it must reset all endpoints used by the interface set by setting the Reset_EP control bits and must set the halt conditions for endpoints not used by this interface.

Other local host 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. No interrupt is generated.

Data stage handshaking for non-autodecoded control write transfers is dependant on the endpoint 0's FIFO_En, EP_Halted, and Stall_Cmd bits. The local host may delay completion of any transaction of the data stage by signaling NAK (via the Set_FIFO_En bit not set). The USB specification requires that once a STALL is signaled in a control transfer, it must be signaled on that endpoint until the next setup token is received. Either the Stall_Cmd or the Set_Halt (reflected in the EP_Halted register bit) register bits provide this functionality. Also note that the EP_Halted bit does not reflect the forced STALL caused by the Stall_Cmd bit; it retains its previous value.

Status stage handshaking is controlled by the endpoint 0's FIFO_En and Stall_Cmd bits. Successful completion of a non-autodecoded control write transfer is indicated by the USB function 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. While NAK handshaking can be used to indicate delays in completion of the requested control write, the USB host may 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 local host cannot deal with itself, the local host 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 the endpoint 0 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.

13.3.5.4 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 local host general USB interrupt with the setup flag set. The local host must respond to this general USB interrupt by setting the Setup_Sel bit, which clears the setup interrupt flag. The local host must then read 8 bytes from the setup FIFO via the DATA register, clear EP_Sel bit, and check the setup flag. If the setup flag is set, the local host must discard the setup data it has just read, and handle the new setup data packet following the same scheme. If the setup flag is cleared, the local host code interprets this request information and then prepares data for the IN transaction that follows. 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 Set_FIFO_En bit.

The data stage of a control read transfer consists of one or more IN transactions. Transaction handshaking and interrupt generation apply as for non-isochronous, non-control IN endpoints; the local host can cause NAK, STALL, or ACK signaling for the data stage transactions. At endpoint 0 TX general USB interrupts, local host code must move more data to the endpoint 0 FIFO until the last bytes of the requested data has 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 local host 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 0-length data packet, and the local host code must return its completion status for the control read standard request via standard handshaking mechanisms.

In the case of returning exactly what the host requested when the 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.

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, setup transaction is ignored (error).

Data stage handshaking for non-autodecoded control read transfers is dependant on the endpoint 0 FIFO_En, EP_Halted, and 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 Stall_Cmd and the Set_Halt (reflected through the EP_Halted register bit) register bits provide this functionality. The EP_Halted does not reflect the forced STALL caused by the Stall_Cmd bit; it retains its previous value.

Status stage is controlled by the FIFO_En and the 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 function module 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 STALL is returned by the core, and an interrupt is asserted.

Non-Autodecoded Control Read Transfer Error Conditions

If an error occurs that the local host cannot handle itself while handling the control read, the local host 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's 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 function 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 local host.

13.3.5.5 Autodecoded Versus Non-Autodecoded Control Requests

Table 13–25. Autodecoded Versus Non-Autodecoded Control Requests

Request	Recipient	Status	LH Required Action	Device Behavior if Device not Configured
GET_ STATUS	Device	Autodecoded	None	Device status is returned (the Self_Pwr and R_WK_OK bits).
	Interface	Non-autodecoded	The local host must stall the command (via the Stall_ Cmd bit) if interface number is not correct. No feature is defined for interface.	Command is passed to the local host.
	Endpoint	Autodecoded	None	The core automatically stalls the command if endpoint number is different from 0.

Table 13–25. Autodecoded Versus Non-Autodecoded Control Requests (Continued)

Request	Recipient	Status	LH Required Action	Device Behavior if Device not Configured
CLEAR/SET FEATURE	Device	Autodecoded	None (DS_Chg interrupt is asserted to the local host after any the R_WK_OK bit modification).	The core handles the request.
	Interface	Autodecoded	None (No feature is defined in USB 1.1 spec for interface. These requests are stalled).	Command is stalled.
	Endpoint	Non-autodecoded	The local host must stall the command (via the Stall_Cmd bit) if endpoint number/type/direction is not correct. The local host must reset the endpoint after having handled the pending transactions (if CLEAR) or set halt condition (if SET). For EP 0, local host only has to clear or set halt condition: FIFO and data PID are always correct for next setup.	Command is passed to the local host.
SET_ ADDRESS	Device	Autodecoded	None (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 local host).	❑ Default: device moves into 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 local host must write descriptor data into endpoint 0 FIFO.	Command is passed to the local host.

Table 13–25. Autodecoded Versus Non-Autodecoded Control Requests (Continued)

Request	Recipient	Status	LH Required Action	Device Behavior if Device not Configured
SET_ DESCRIPTOR	All	Non-autode-coded	The local host must stall the command (via the Stall_Cmd bit) if it does not support set descriptor requests.	Command is passed to the local host.
GET/SET CONFIGURATION	Device	Non-autode-coded	The local host must stall the command (via the Stall_Cmd bit) if configuration number is not correct. If the request is SET_CONFIG, the local host must reset all endpoints, halt endpoints not used by the default interface setting, set the Self_Pwr value if device is self-powered for the configuration set, and then set the Dev_Cfg bit (if config nb is not 0), or set the Clr_Cfg bit (if config nb is 0) before allowing status stage to complete. The device goes 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 local host.	Command is passed to the local host.
GET/SET INTERFACE	Interface	Non-autode-coded	The local host must stall the command (via the Stall_Cmd bit) if interface/setting number is not correct. If the request is SET_INTER–FACE, the local host must reset endpoints used by the interface, and then halt endpoints not used by the interface setting, before allowing status stage to complete.	Command is passed to the local host.
SYNCH_ FRAME	Endpoint	Non-autode-coded	The local host 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 local host.

- Transactions on endpoints other than zero are ignored if the device is not configured (addressed state).
- If some endpoints are not used by the interface currently set, transactions on these endpoints are not ignored; the local host must set the Halt feature for the endpoint. This does not happen if the USB host works correctly.
- If endpoint 0 is halted, per USB 1.1 specification (see USB 1.1 specification, section 9.4.5: Get_Status), all requests are stalled except GET_STATUS, CLEAR_FEATURE, and SET_FEATURE requests.
- Requests are handled per specification USB 1.1, when specified in this specification, but many device reactions are not specified by USB 1.1.

13.3.5.6 Note on Control Transfers Data Stage Length

The control transfer data stage length is indicated in setup data packet.

During control reads, if the USB host requests more data than indicated in setup packet, an 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 local host will not have enabled the RX FIFO.

During control writes, if the USB host sends more bytes than indicated in setup packet, the transfer is STALLed. If the USB host sends less 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 local host will not have enabled the TX FIFO.

13.4 Device Initialization

To allow communication between the device and a USB host, the local host must configure the device by filling the configuration registers.

For each endpoint, the local host must write to dedicated register:

- Endpoint size
- Whether or not double buffering is allowed for endpoint
- Endpoint type (isochronous or non-isochronous)
- Address of the pointer

The RAM has a specified size (2048 bytes), and the local host can choose its configuration by setting appropriate value. Figure 13–9 shows an example of RAM organization.

Once the endpoints are configured, the local host must set the Cfg_Lock bit. If this bit is not set, all transactions are ignored by the core. Then, when the local host is ready to communicate with the USB host, it must set the Pullup_En bit. The local host can wait until the DS_Chg Attach interrupt has been detected and handled before setting the Pullup_En bit. The USB host does not detect the device until this bit is set.

Figure 13–10 and Figure 13–11 are flowcharts of the configuration phase.

Figure 13–9. Example of RAM Organization

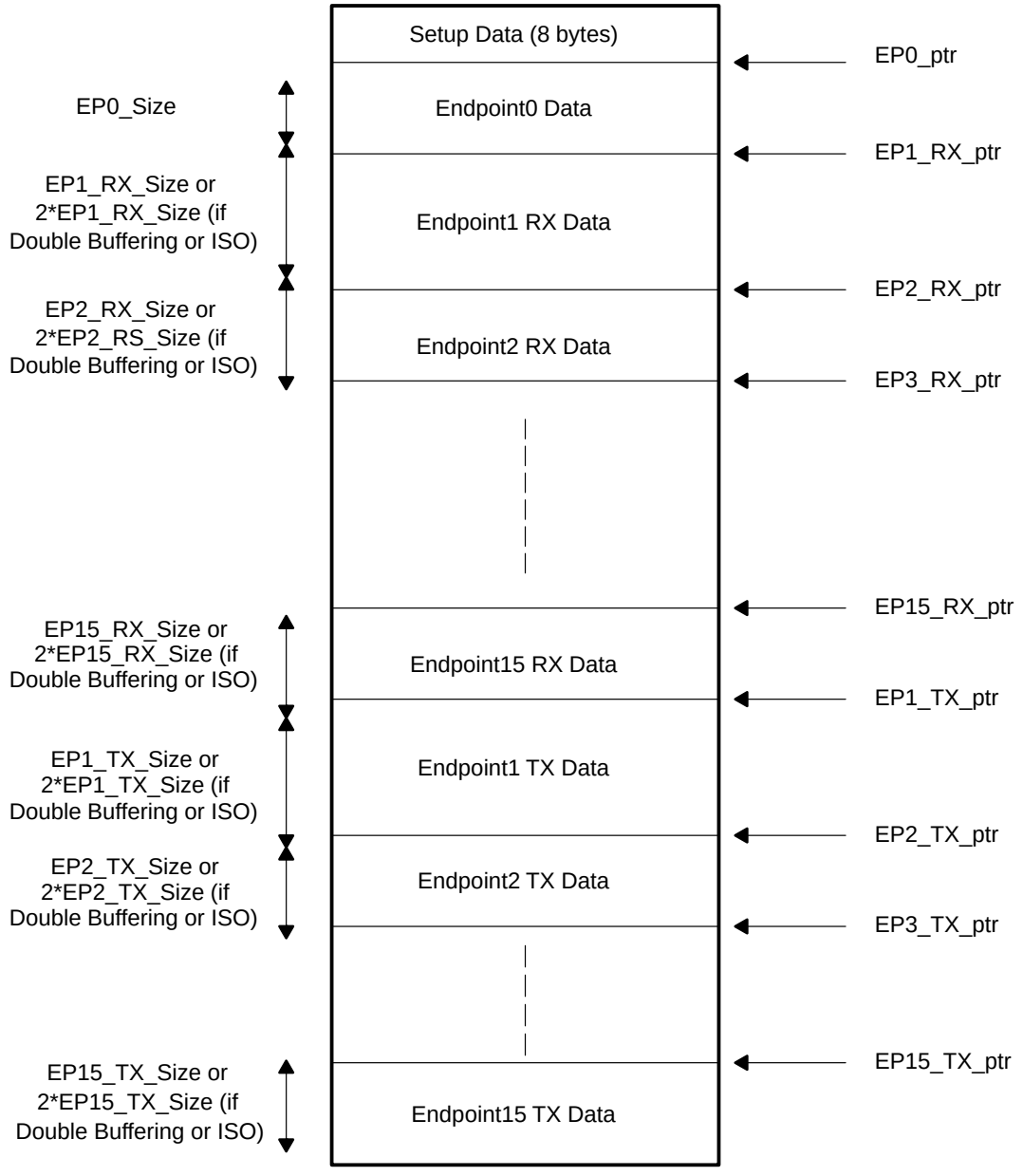


Figure 13–10. Device Configuration Routine

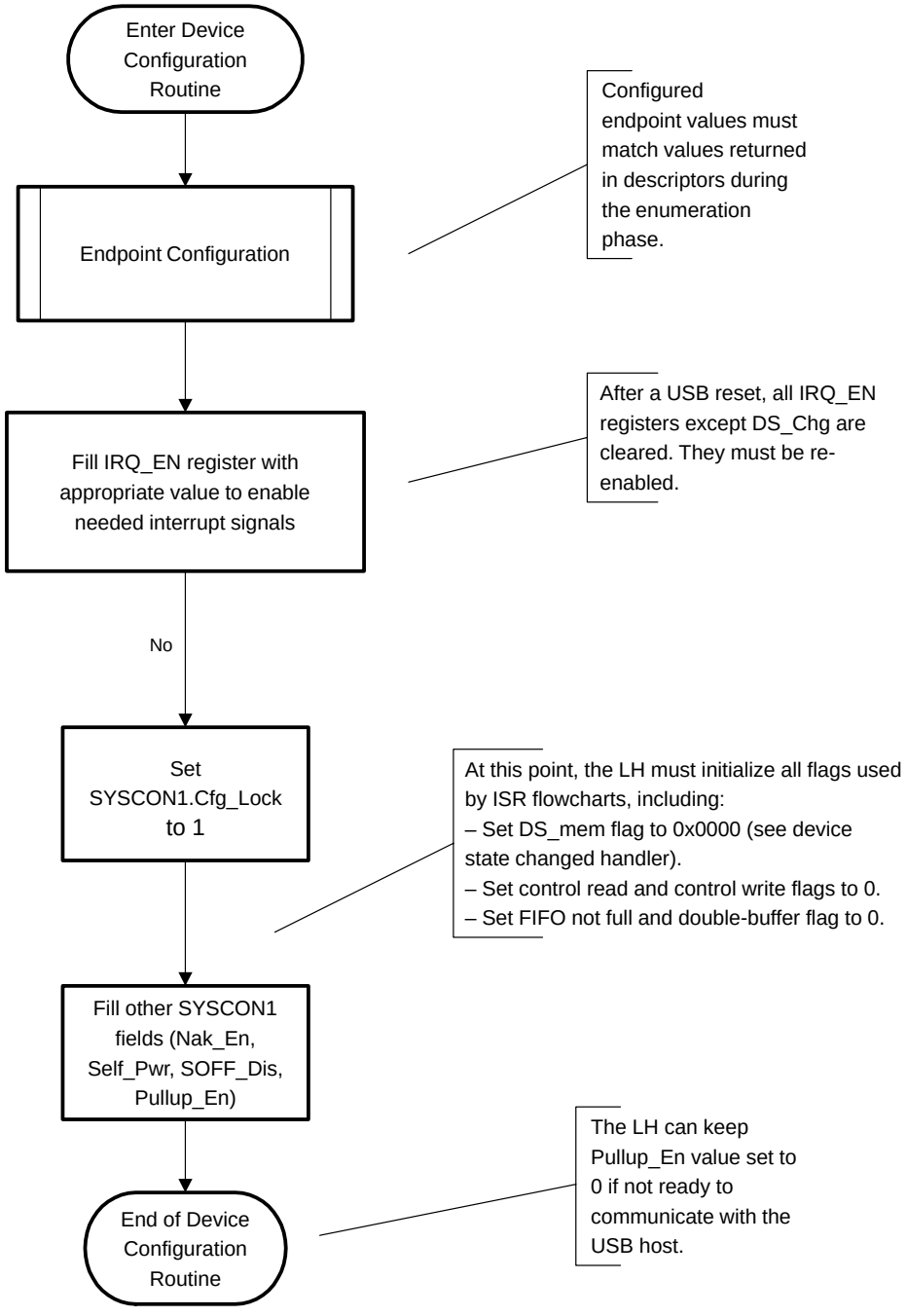
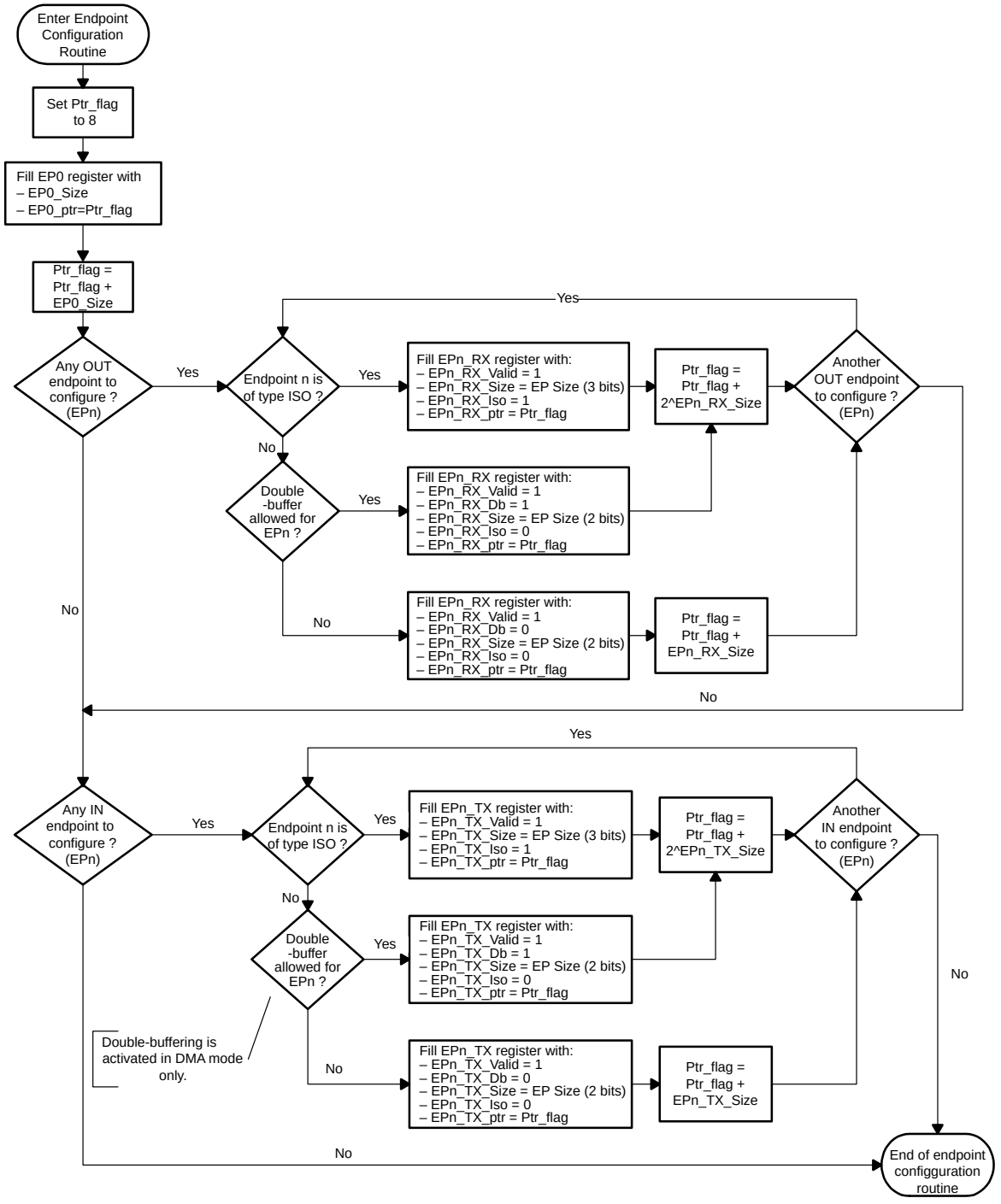


Figure 13–11. Endpoint Configuration Routine



13.5 Preparing for Transfers

To avoid NAK handshakes for the first transaction on an endpoint, the local host must prepare the endpoint FIFO for receiving or transferring data. After the first transaction, the FIFO is enabled during the interrupt handling.

For receive endpoints, this phase consists of enabling the FIFO to receive data from the USB host. If double buffering is allowed for the endpoint, setting the Set_FIFO_En bit enables both FIFOs. Therefore, it is not possible to allow a single transaction when double buffering is used.

The local host enters the Prepare for USB RX Transfers routine, shown in Figure 13–12, after the enumeration phase and then properly reacts to endpoint interrupts. Whether double buffering is allowed or not is transparent to the local host, unless both FIFOs are cleared through the Clr_EP or the Reset_EP bits. In that case, and in the case where the local host finishes handling an interrupt without having set the Set_FIFO_En bit, the local host must reenter the Prepare for USB RX Transfers routine.

For transmit endpoints, the local host enters the Prepare for USB Transfer on endpoint n routine each time a new file must be transmitted from endpoint n to USB host. The local host must not enter this routine until data written into TX FIFO from previous transfer has been received successfully by the USB host (ACK interrupt received), unless TX FIFO is cleared through the Clr_EP or the Reset_EP bits.

This does not apply to endpoint 0, which is not used before a setup interrupt occurs. At setup interrupt, the local host reacts appropriately and enables the EP0 FIFO only if necessary.

Figure 13–12. Prepare for USB RX Transfer Routine

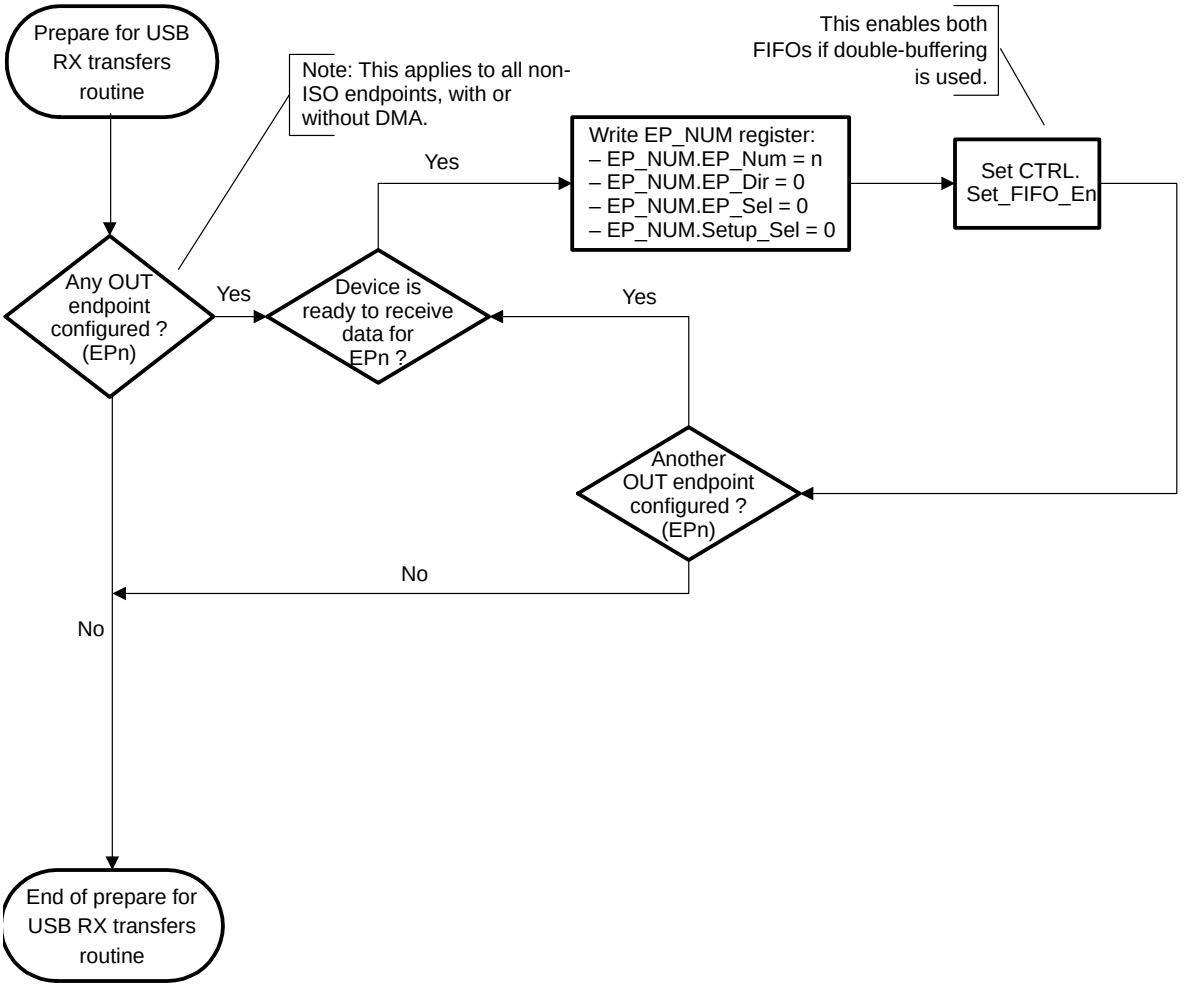
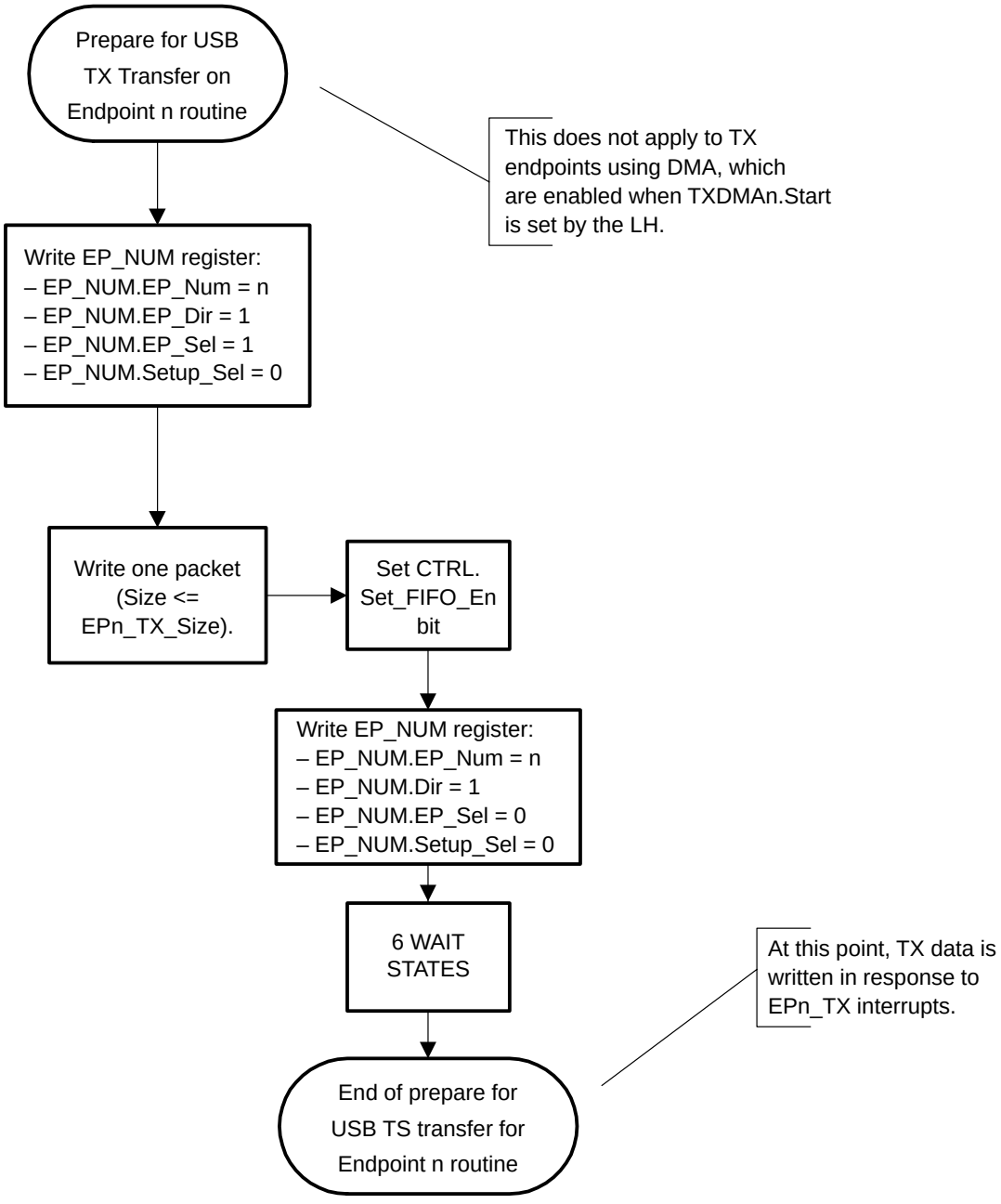


Figure 13–13. Prepare for TX Transfer on Endpoint n Routine



Note: No double buffering in non-DMA TX mode.

13.6 Interrupt Service Routine (ISR) Flowcharts

The flowcharts in this section give general operational guidelines for USB ISR processing. System-architecture-specific details are left to the engineers who write the local host and USB host code. One USB-specific interrupt register is provided (IRQ_SRC) to keep track of the interrupts to handle. These interrupts can be of the following types:

- General USB interrupts (including endpoint 0, DMA, and device states interrupts)
- Non-isochronous endpoint-specific interrupt
- Start of frame (SOF) interrupt for isochronous transactions

The general USB interrupt ISR must handle non-autodecoded control transfers on endpoint 0 and some special interrupts generated due to USB device state modifications or DMA transfers. The ISR for the endpoint-specific interrupt must handle interrupts from the USB module that are generated due to 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 below to give a guideline for how to handle the interrupts related to the USB function module. The flowcharts in this section assume that the Nak_En bit is cleared.

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 show reads from these application buffers for IN transactions on TX endpoints, and they show writes to these application buffers for OUT transactions on RX endpoints.

13.6.1 Important Note on USB Interrupts

When an endpoint interrupt is asserted, the local host sets the EP_Sel bit to 1. The local host must finish the interrupt handling before clearing EP_Sel bit, because clearing this bit clears the corresponding status bit in the status flag register (ACK, NAK, STALL). When an interrupt is pending on an endpoint, the local host must not select then deselect the endpoint without handling the interrupt, because this clears the pending transaction status flags. The local host does not need to set EP_Sel to 1 when setting the Set_FIFO_En, the Set_Halt and the 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 endpoint halt feature was set), so that the local host never misses

an ACKed transaction. If double buffering is used for an endpoint, the status flag register 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 local host does not need to set the Nak_En bit during normal operation. However, this bit must be set when the local host finishes handling an endpoint interrupt without having set the corresponding Set_FIFO_End bit. During TX transaction, if the Nak_En bit is set, the local host must wait for a NAK interrupt to write the TX data, to avoid a possible conflict caused by reception of a NAK interrupt while the local host is writing the TX data.

13.6.2 Parsing the General USB Interrupt

The general USB interrupt 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-isochronous DMA transfers in either receive or transmit mode. Multiple interrupts may be active at any time, and all interrupts must be dealt with by the ISR before returning from the ISR. Figure 13–14 shows an appropriate flowchart for parsing the general USB interrupts.

13.6.3 Setup Interrupt Handler

A separate interrupt flag exists for setup transactions, so that the local host cannot miss a setup transaction, even if it occurs during data or status phase of another transfer (case of aborted transfer). The setup parsing function captures the control transfer request information for use in determining which USB bus activity is needed and controlling how the local host must generate or respond to the control transfer. This information includes:

- bmRequestType
- bmRequest
- wValue
- wIndex
- wLength

The setup interrupt handler shown in Figure 13–15 is responsible for processing setup transactions occurring on endpoint 0. It calls the routine that parses the control transfer request information, shown in Figure 13–16 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 13–14. General USB Interrupt ISR Source Parsing Flowchart

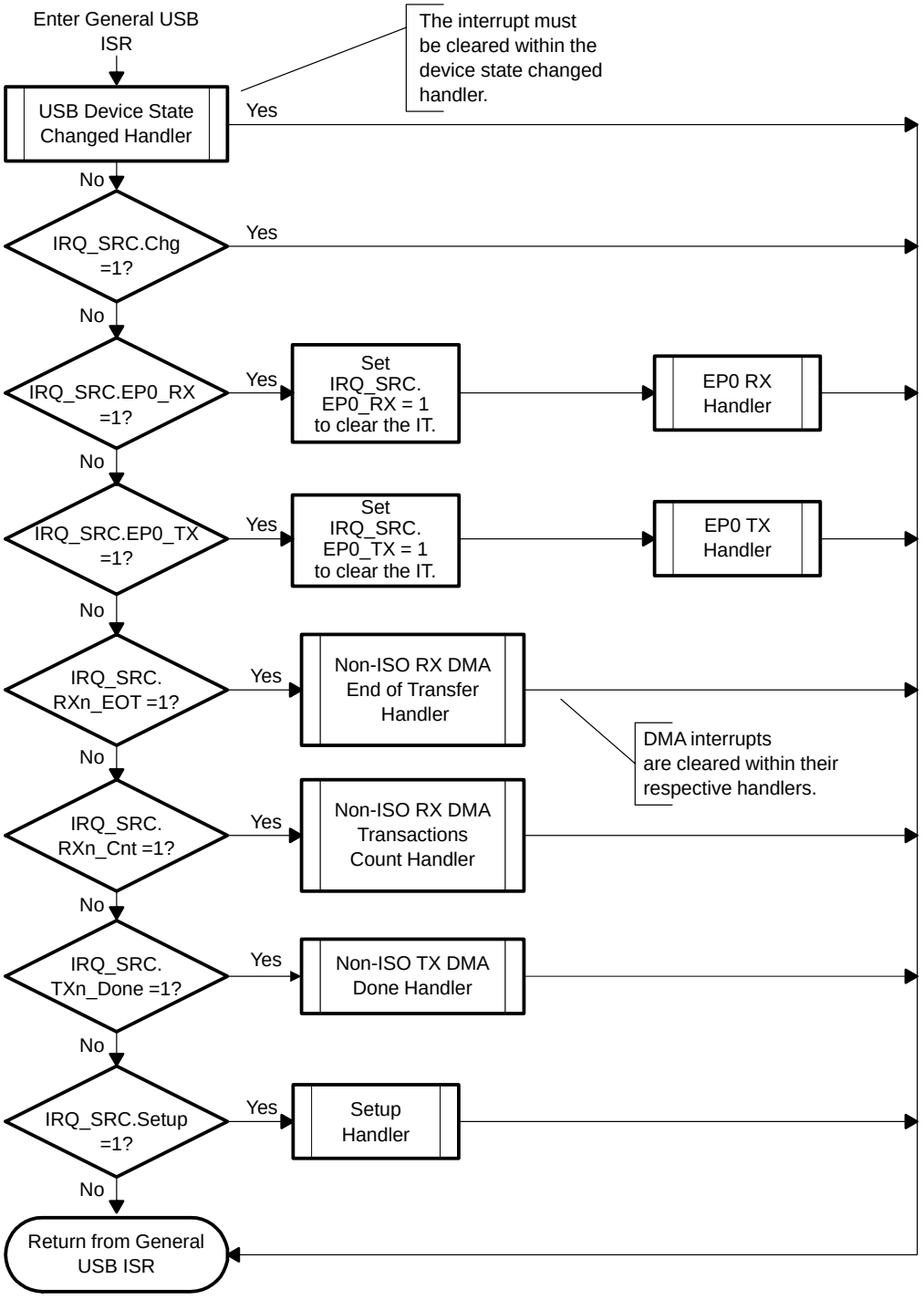


Figure 13–15. Setup Interrupt Handler

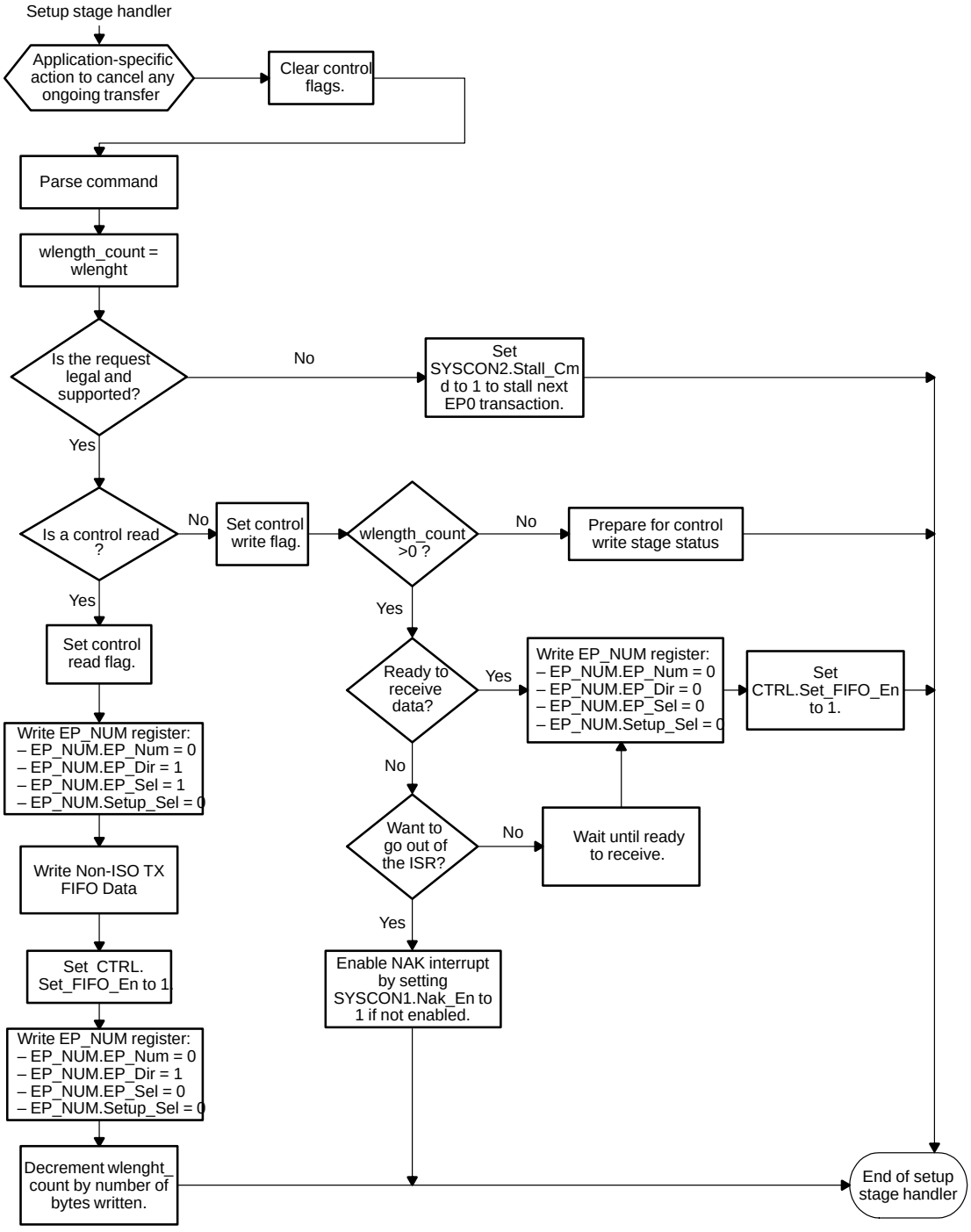
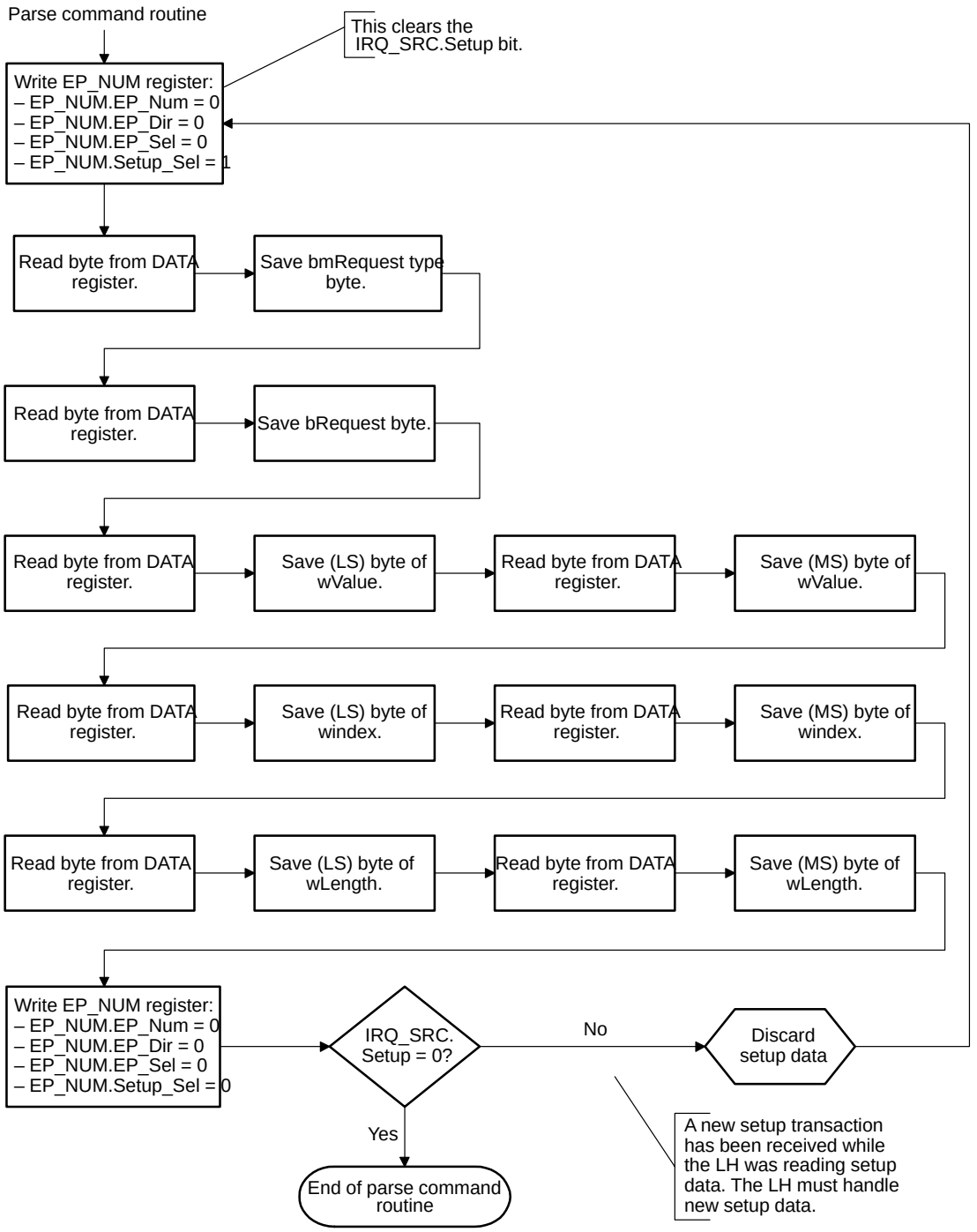


Figure 13–16. Parse Command Routine (Setup Stage Control Transfer Request)



13.6.4 Endpoint 0 RX Interrupt Handler

The endpoint 0 RX portion of the general USB interrupt handler, shown in Figure 13–17, must handle general USB interrupts related to control OUT transactions on endpoint 0. Notice that no EP0 interrupt is generated for autodecoded control transfers.

13.6.5 Endpoint 0 TX Interrupt Handler

The endpoint 0 TX portion of the general USB interrupt handler, shown in Figure 13–19, 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.

Figure 13–17. Endpoint 0 RX Interrupt Handler

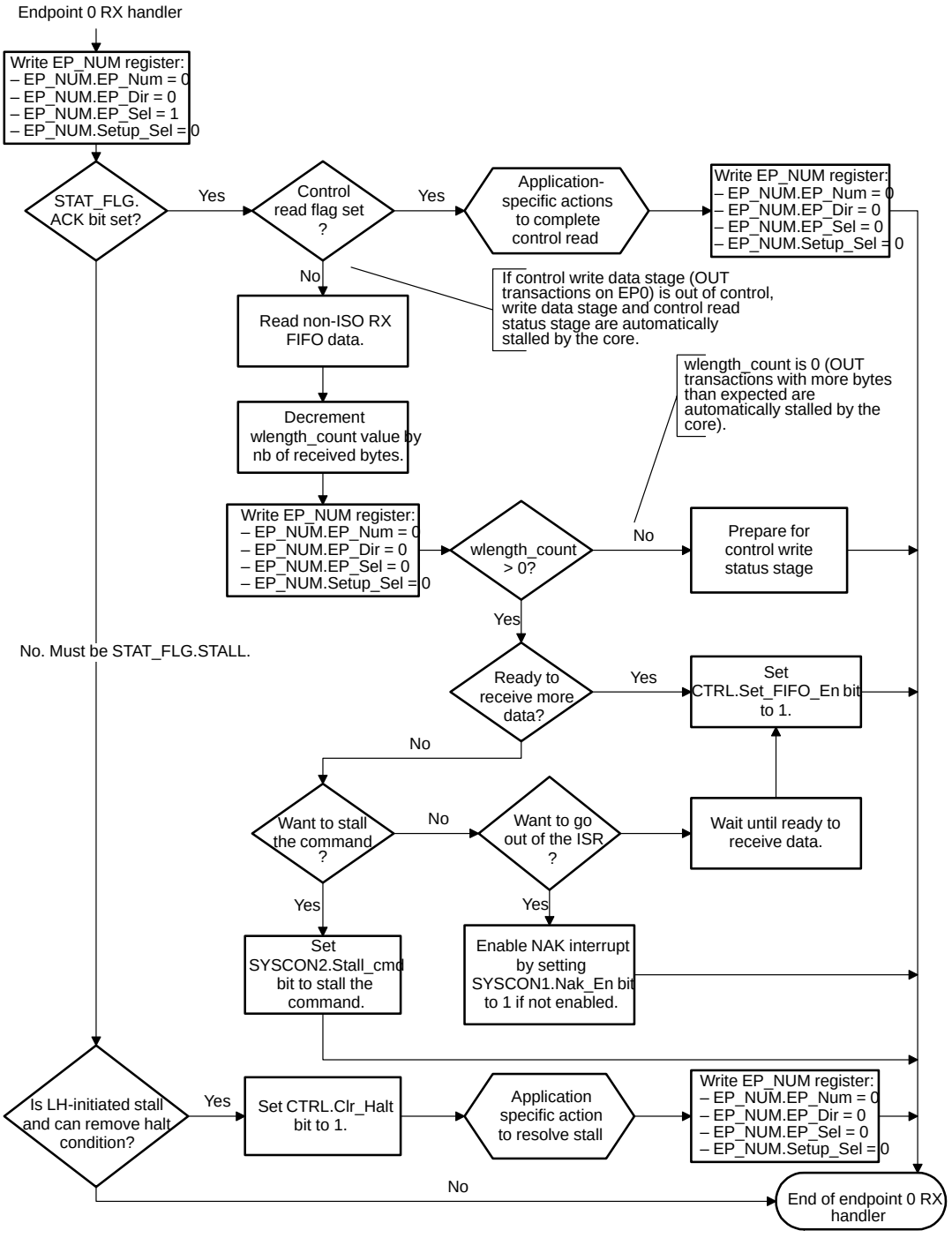


Figure 13–18. Prepare for Control Write Status Stage Routine

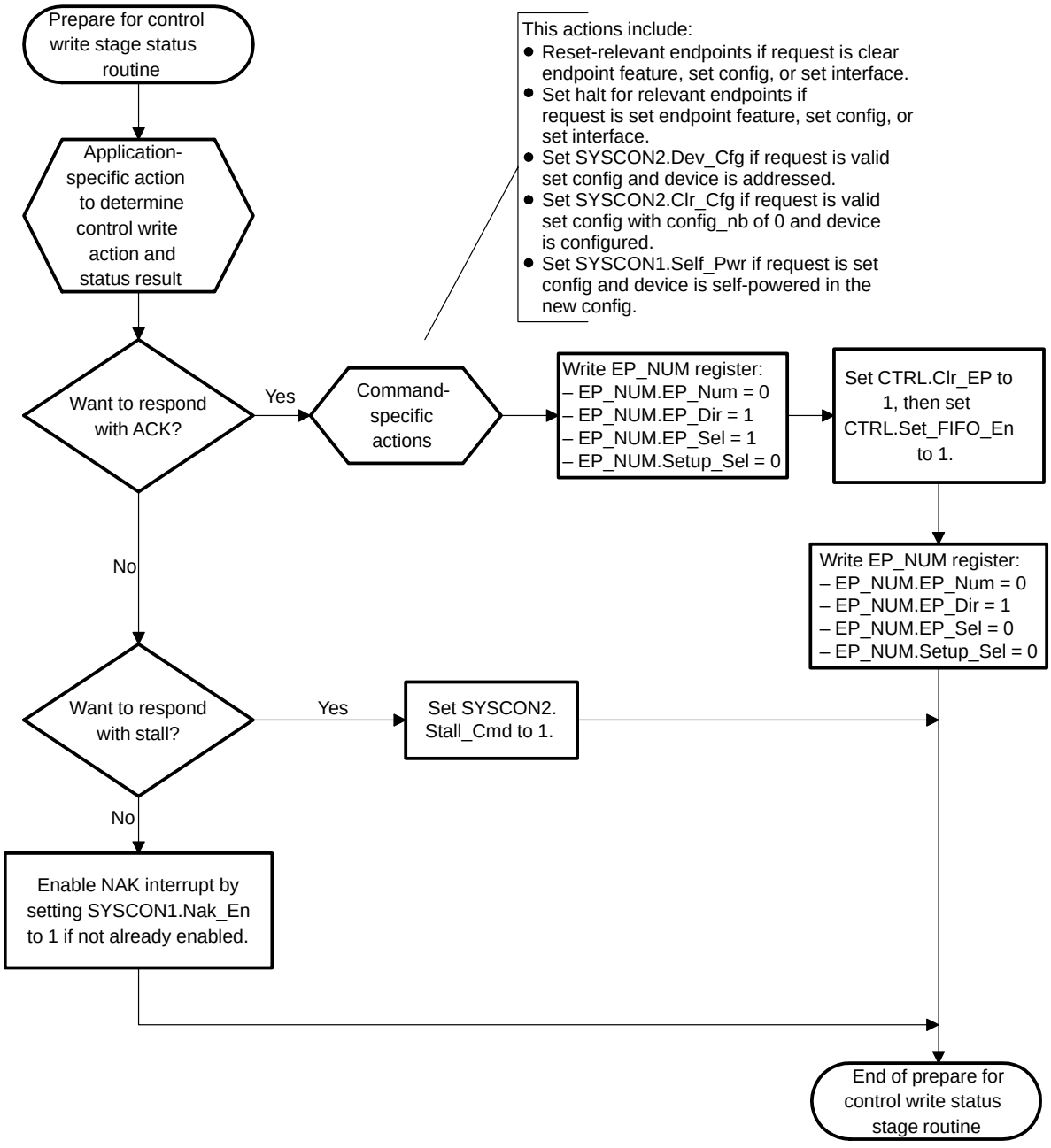


Figure 13–19. Endpoint 0 TX Interrupt Handler

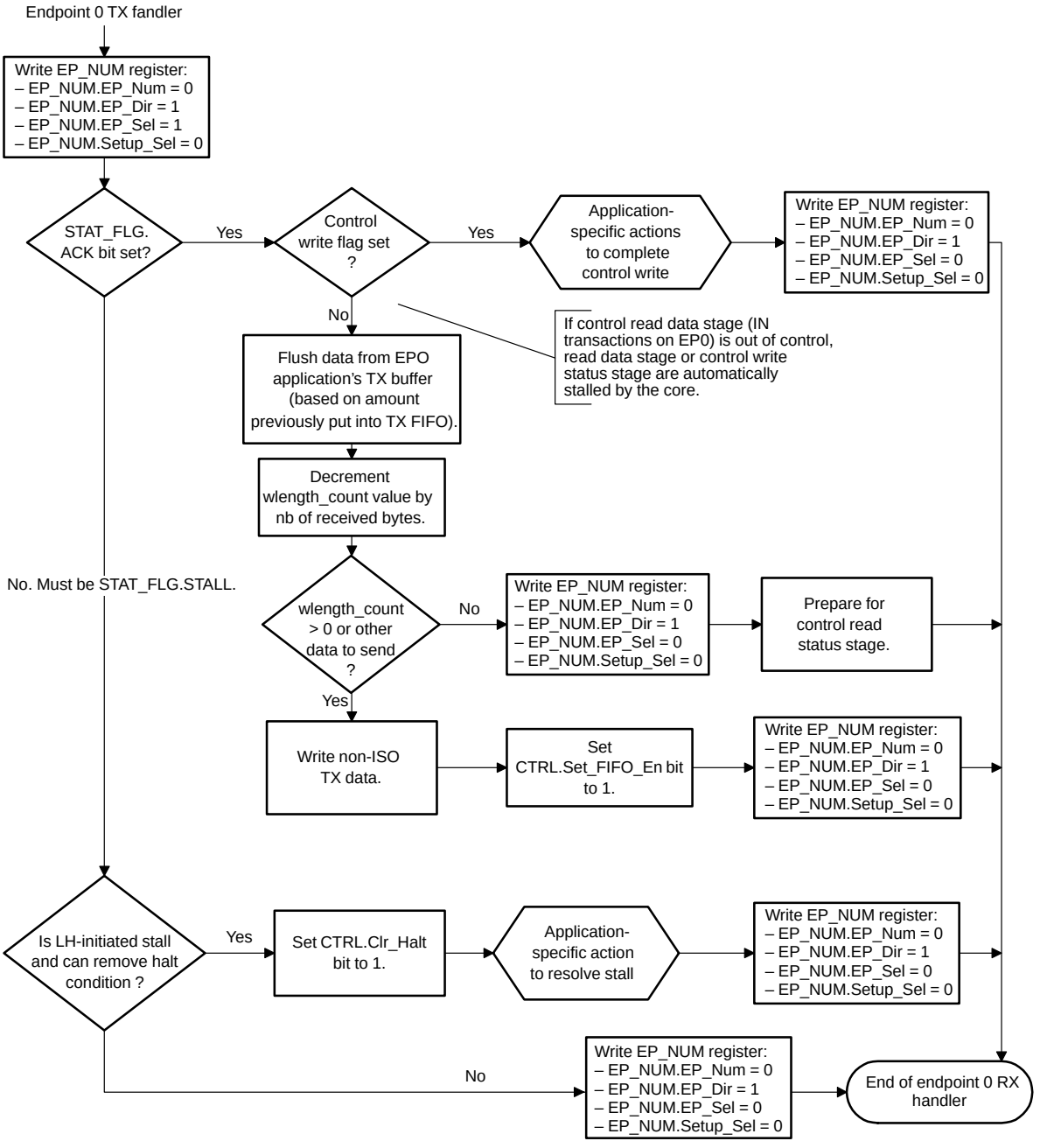
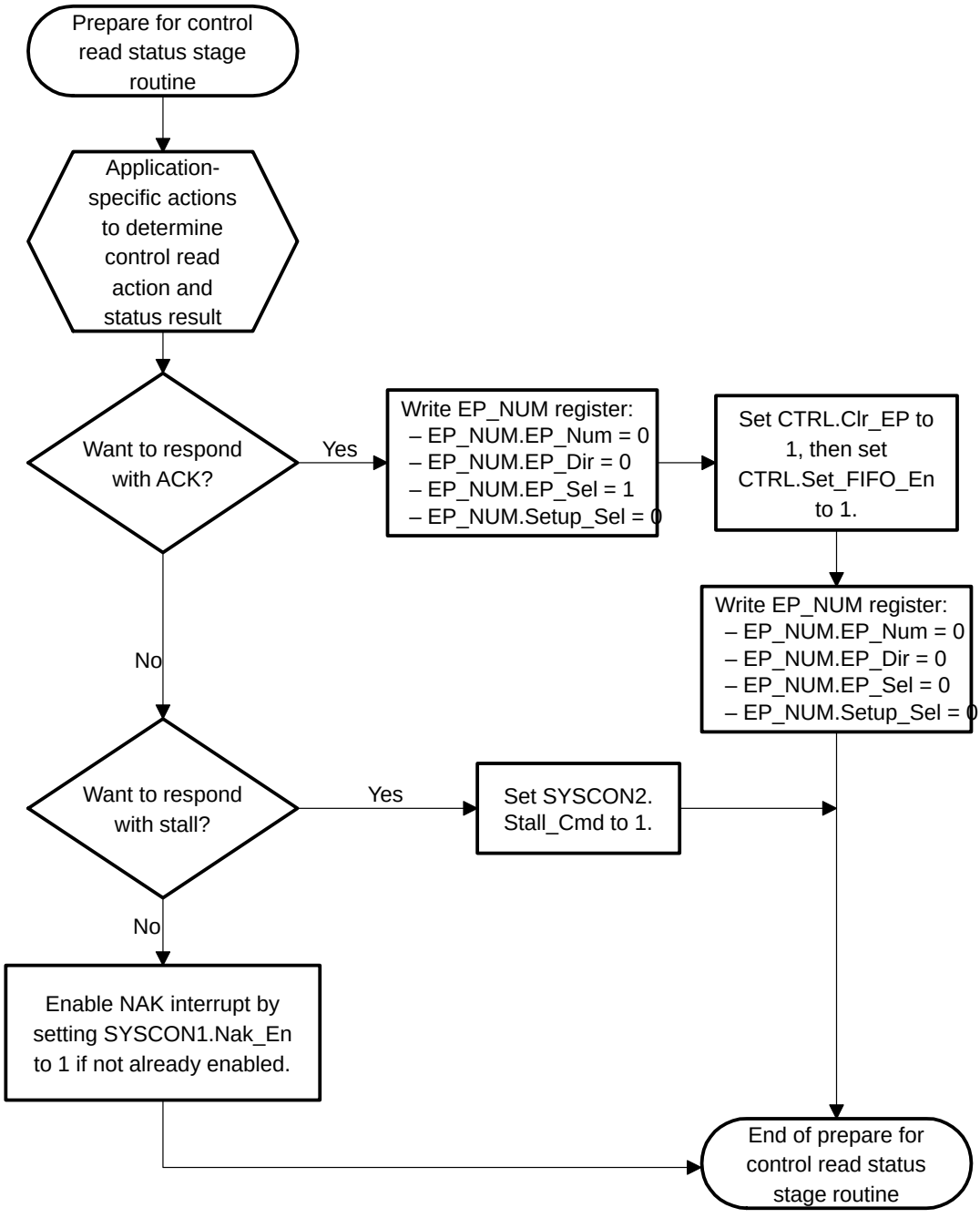


Figure 13–20. Prepare for Control Read Status Stage Routine



13.6.6 Device States Changed Handler

This section describes how USB device states and transitions states are decoded by the USB function and how they can be handled.

The state machine (see Figure 13–21) moves the USB function device from one state to another state with respect to USB1.1 specification. Attach/ unattach transition is not shown in the transition flow.

Since the SET_CONFIGURATION is not decoded by the core, the local host has the responsibility to distinguish a SET_CONFIGURATION with a non-valid configuration value from other SET_CONFIGURATION requests and to set the Dev_Cfg only if configuration value is valid (value 0 is non-valid), when device is in addressed state. When device is in configured state, the local host has the responsibility to set the Clr_Cfg if configuration number is 0 so that the device moves to addressed state.

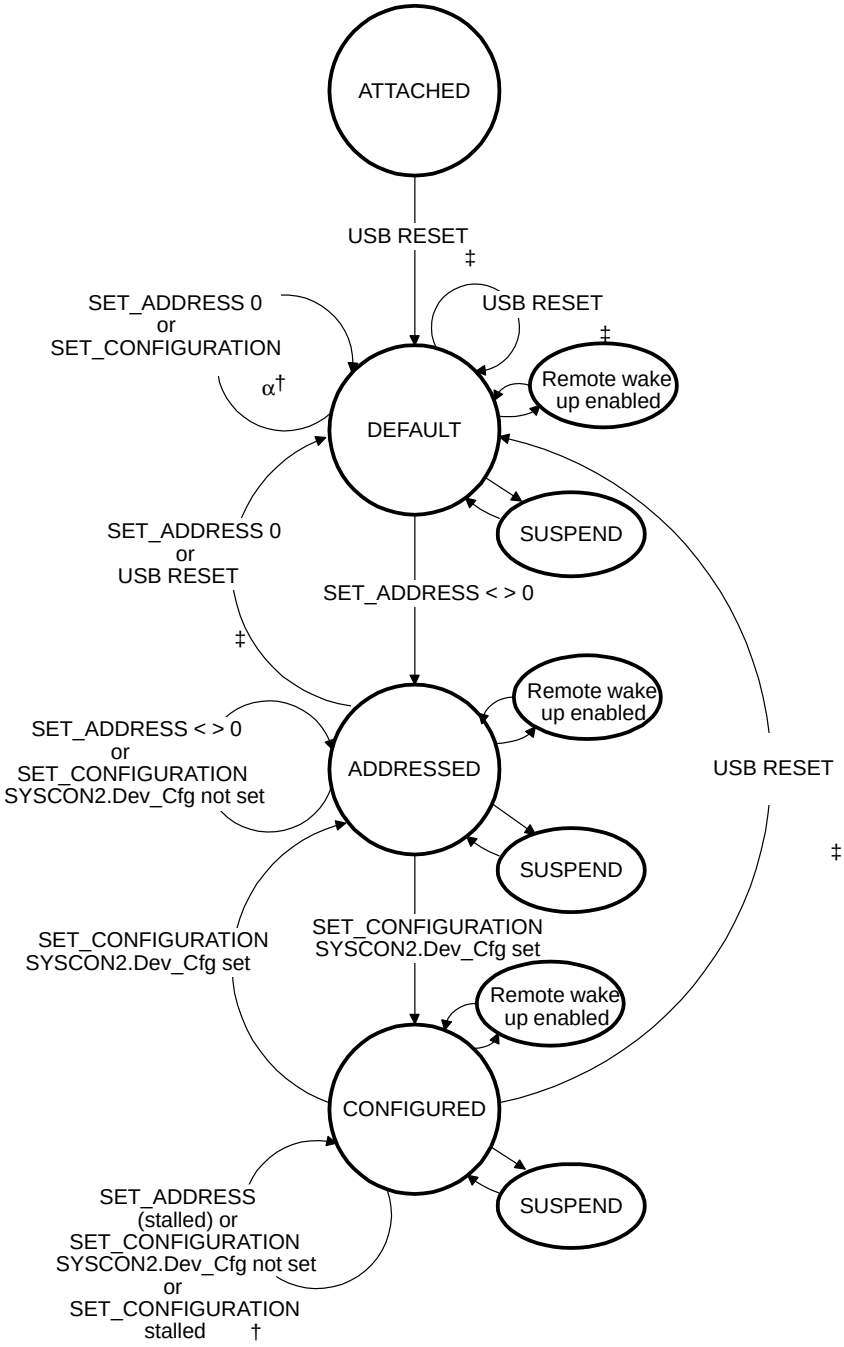
Device states are visible in DEVSTAT register and are decoded as follows:

- Attached: The device is attached to the USB and powered.
- Default: The device is attached to the USB, powered, and reset.
- Addressed: The device is attached to the USB, powered, reset, and an address has been assigned. The device moves into the addressed state after a SET_ADDRESS request with an address number different of 0.
- Configured: The device is attached to the USB, powered, reset, has an address different from 0, and is configured. The device moves into the configured state after a valid SET_CONFIGURATION request only if the local HOST has set the Dev_Cfg bit (meaning the configuration is valid).
- Suspended: 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, respectively.

Any change in the DEVSTAT register bits triggers a device change interrupt (the 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 local host sets the Clr_Cfg or the DEV_Cfg bit.

Figure 13–21. USB Function Device State Transitions

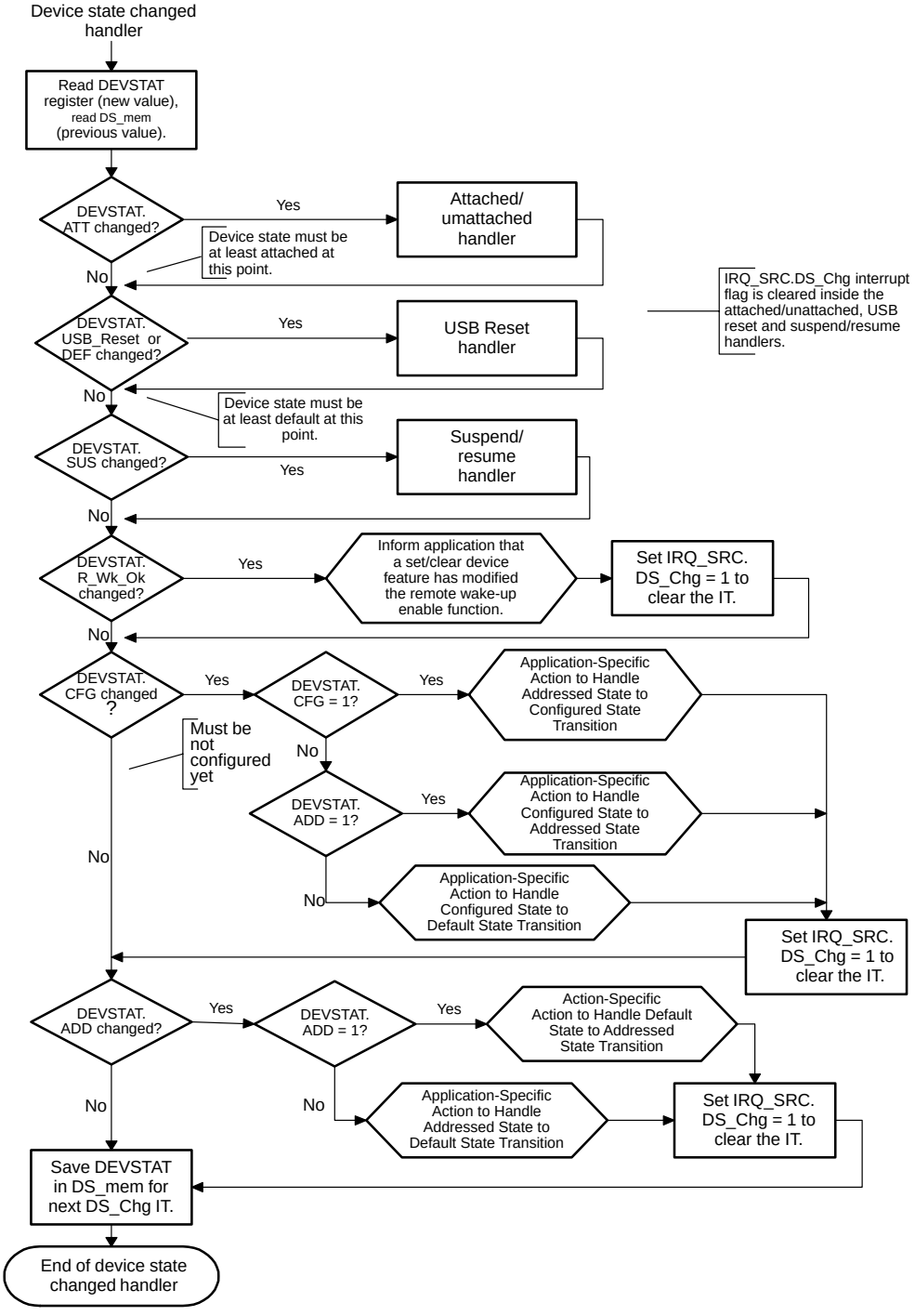


Behavior not specified by USB 1.1 specifications (see chapter 9)

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.

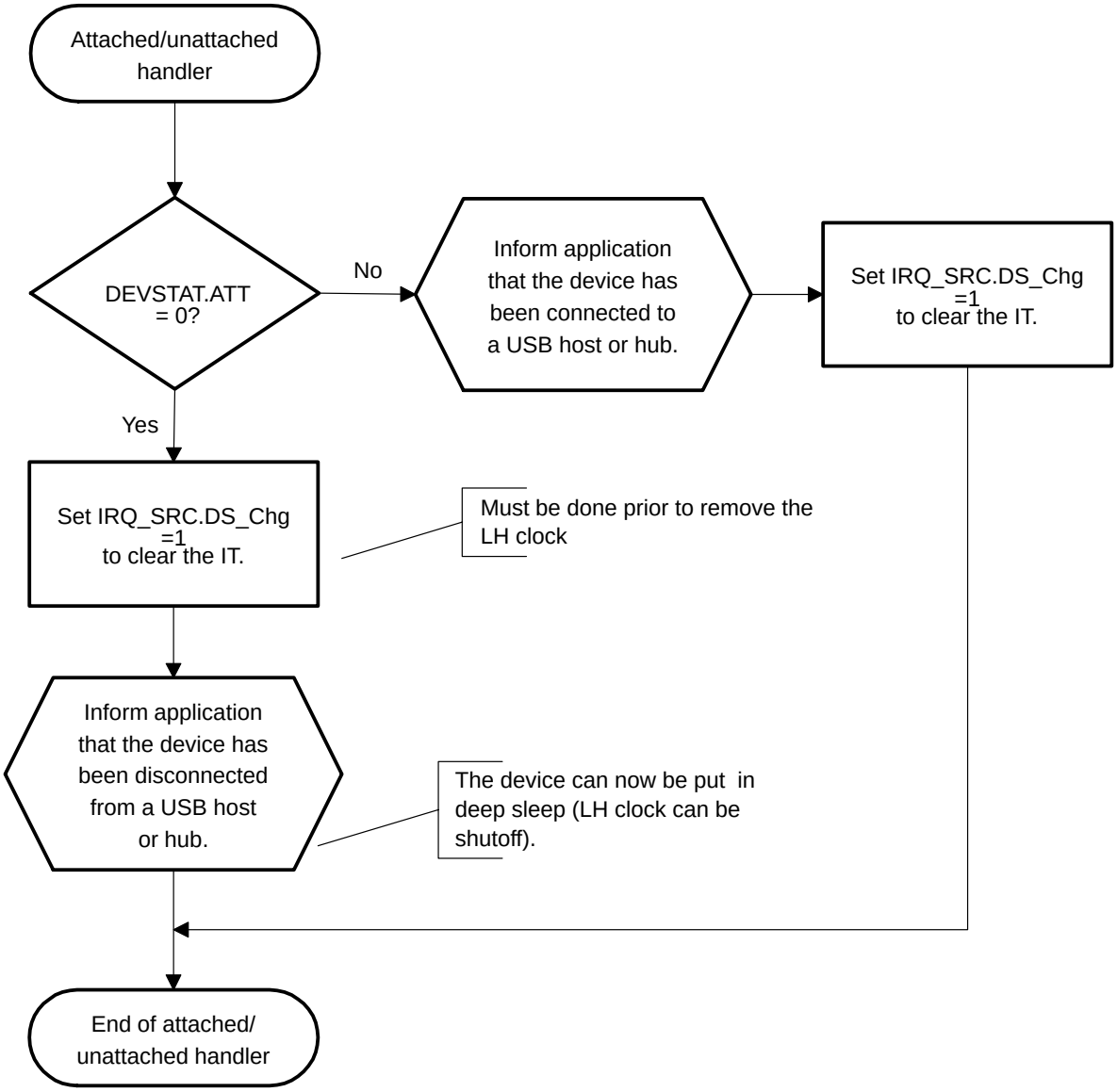
Figure 13–22. Typical Operation for USB Device State Changed Interrupt Handler



13.6.7 Device States Attached/Unattached Handler

The device attached/unattached interrupt (Figure 13–23) occurs either when the device etects it is connected to the USB host or Hub (VBUS is on) or when it becomes disconnected (VBUS is off). The local host can use this interrupt to put the evice into deep sleep or to initialize any application-specific information relating to the USB device.

Figure 13–23. Attached/Unattached Handler



13.6.8 USB Reset Interrupt Handler

When a USB reset occurs, the USB module generates a general USB interrupt to the local host (see Figure 13–24 and Figure 13–25). The local host 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 of 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

13.6.9 Suspend/Resume Interrupt Handler

When a USB suspend/resume general USB interrupt occurs (see Figure 13–26), the USB module has either entered or left suspend mode. The local host code must determine which and react appropriately.

The suspend sense hardware is implemented to trigger only after 5 ms of bus IDLE. This forces compliance with the USB Spec Version 1.1 t_{WTRSM} timing parameter (3 ms of IDLE to identify suspend, 2 ms before remote device can signal resume).

If the local host wants to wake the device from suspend mode and remote wakeup enable is set (bit R_WK_OK = 1), it must first turn its clock on (if stopped) then set the Rmt_Wkp. The device then resumes.

If shutoff is enabled (the SOFF_Dis = 0), the 48-MHz clock is automatically shut off at suspend and turned on at resume (USB host or local host driven). Setting the SOFF_Dis bit is part of the device configuration; however, the local host can modify its value at suspend interrupt time if necessary.

13.6.10 Parsing the Non-Isochronous Endpoint-Specific Interrupt

The endpoint-specific interrupt ISR (Figure 13–27) must parse the interrupt identifier registers IRQ_SRC to determine the interrupts that are active (EPn_RX, EPn_TX, or both). The two interrupts can be active at any time, and must be dealt with by the ISR before returning from the ISR. The ISR must then read EPN_STAT register to determine the endpoint causing the interrupt. For each direction, only one endpoint interrupt can be active at a time.

Figure 13–24. USB Reset Handler Flowchart I

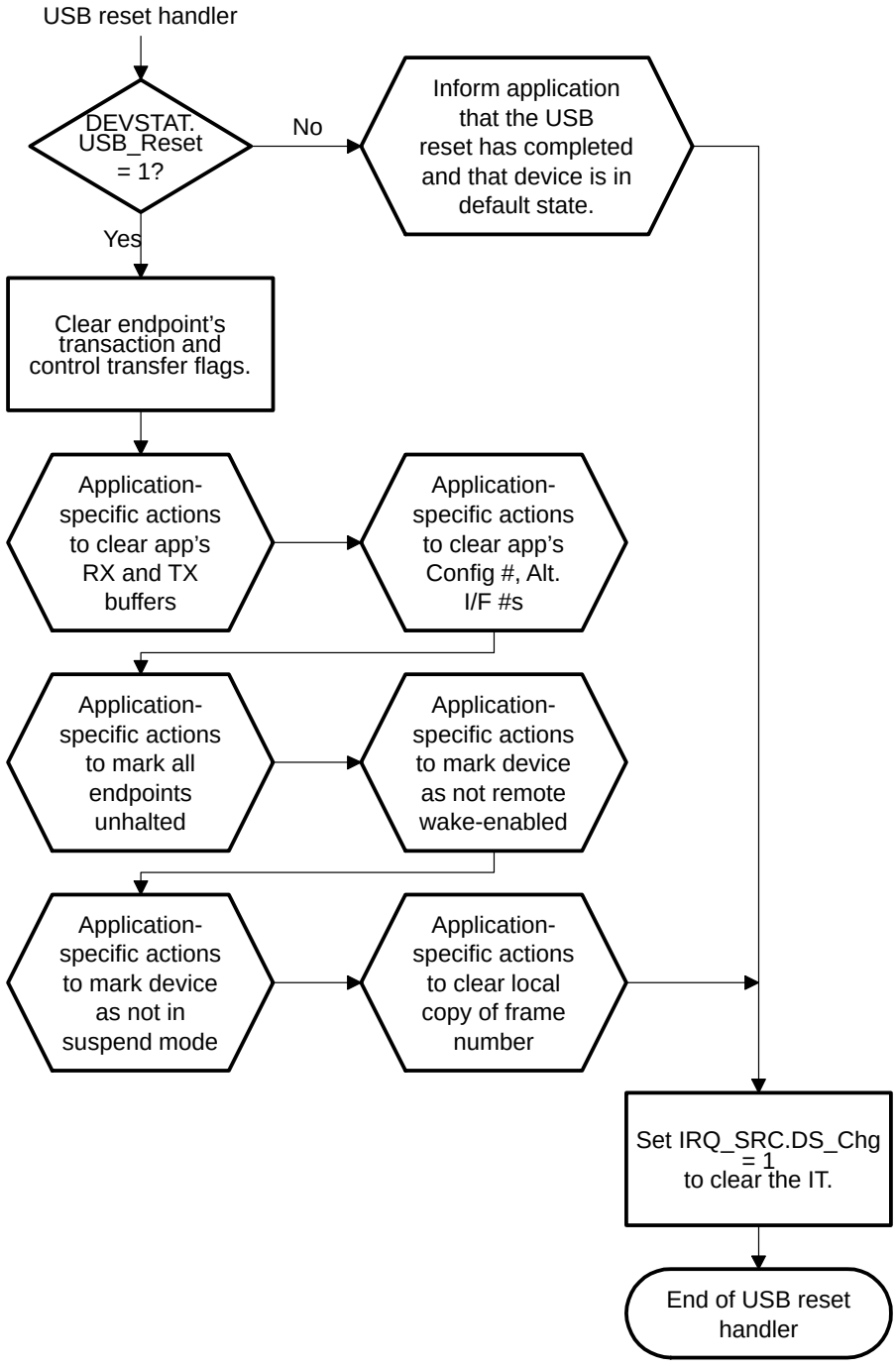


Figure 13–25. USB Reset Handler Flowchart II

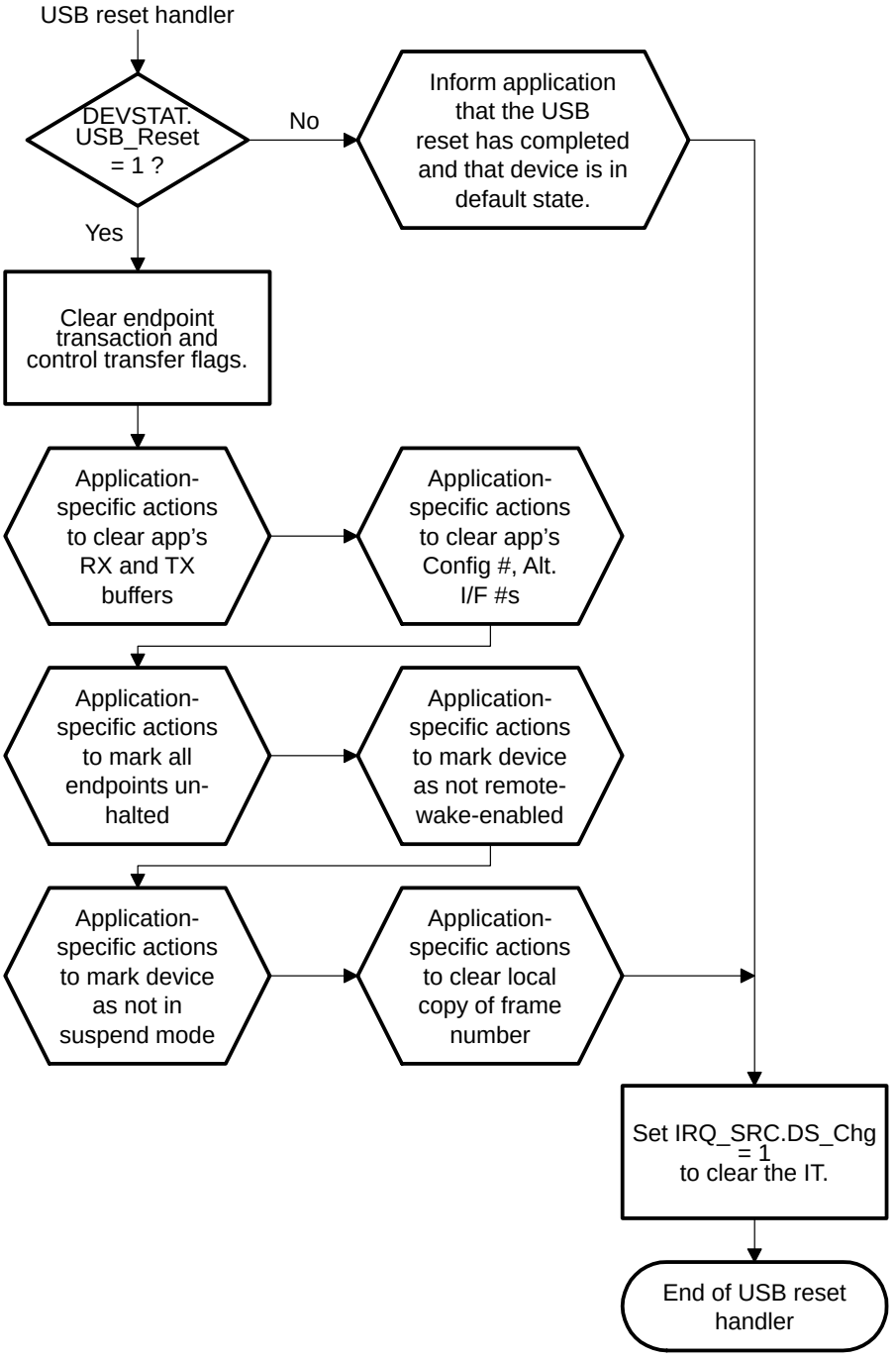


Figure 13–26. Typical Operation for USB Suspend/Resume General USB Interrupt Handler

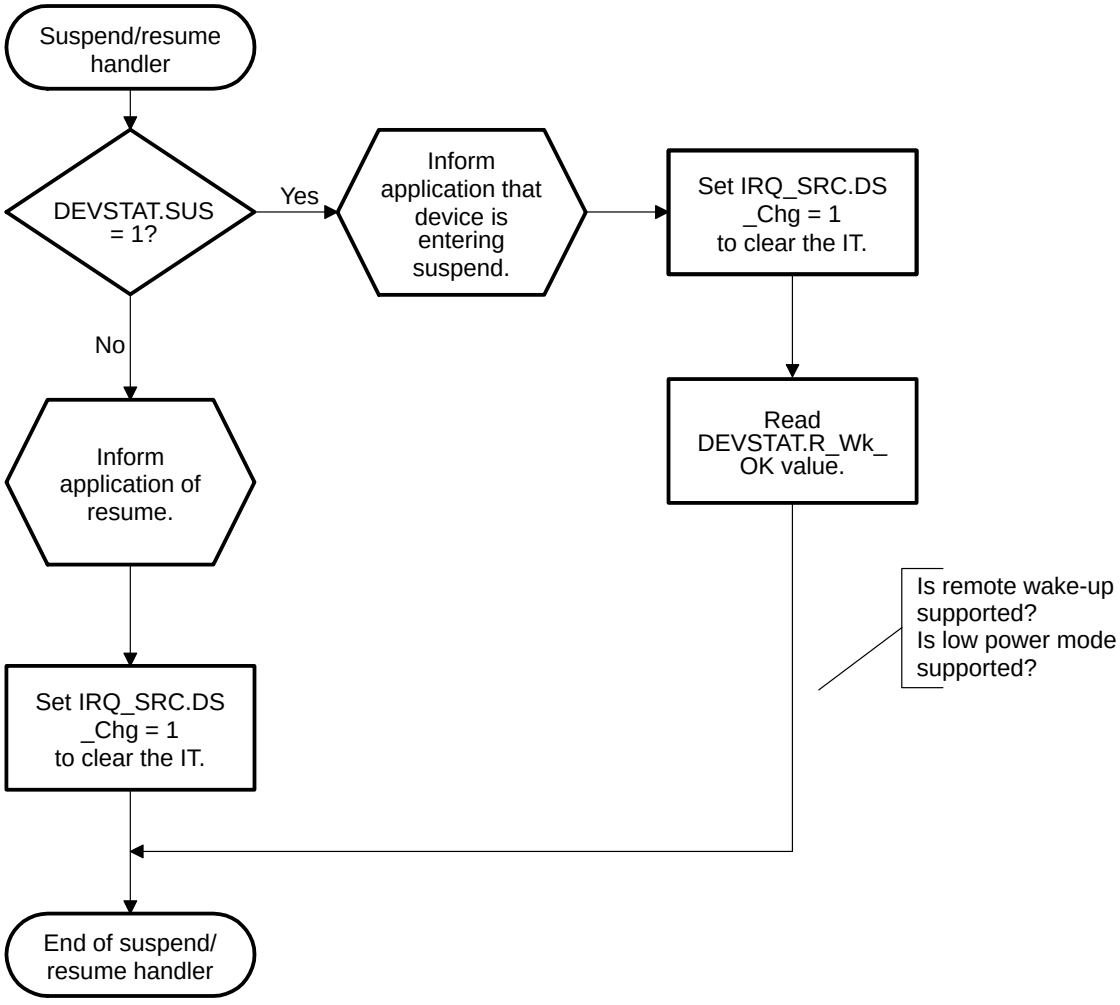
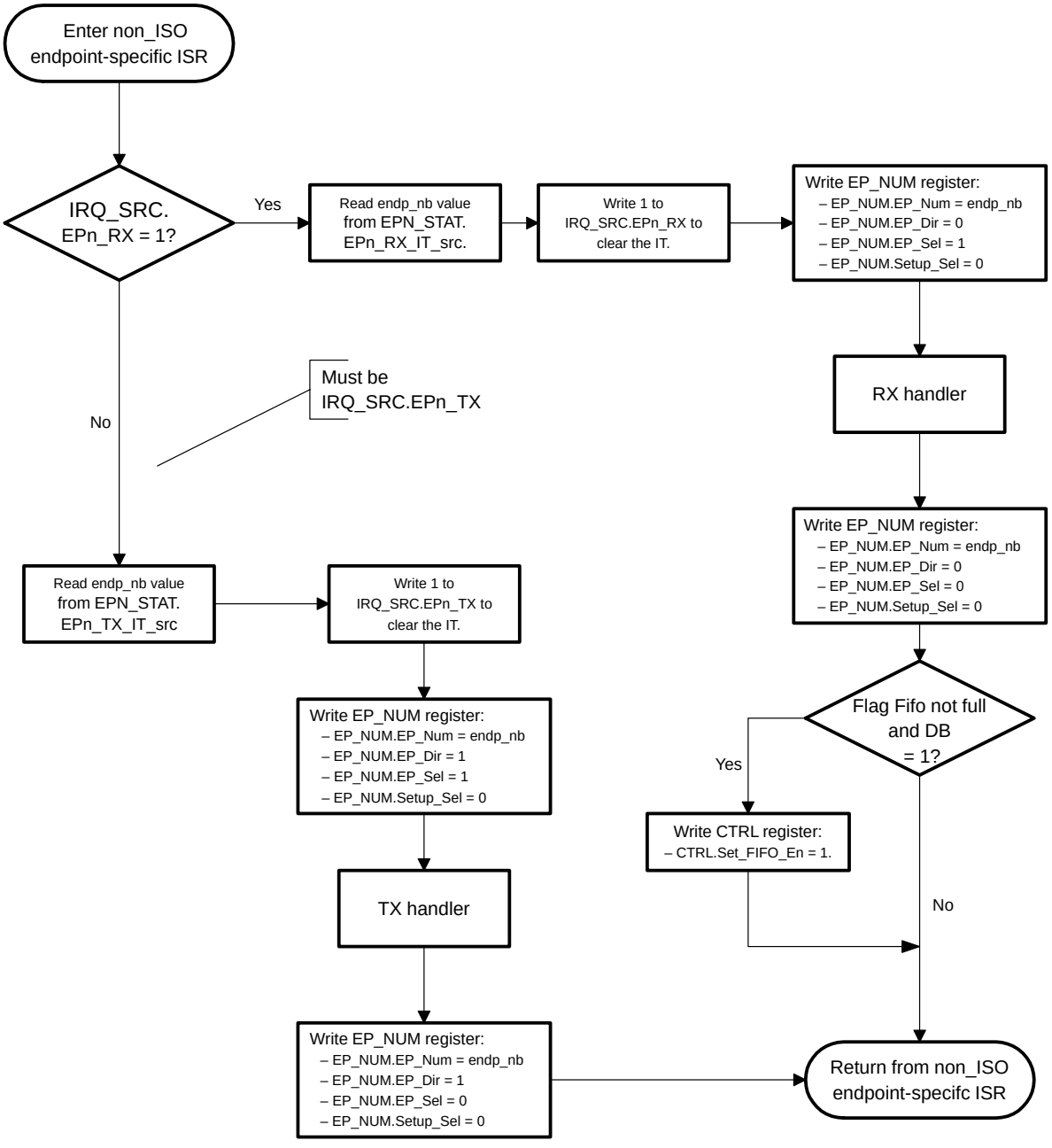


Figure 13–27. Non-Isochronous Endpoint-Specific (Except ER 0) ISR Flowchart



13.6.11 Non-Isochronous, Non-Control OUT Endpoint Receive Interrupt Handler

Figure 13–28 shows the operations necessary to handle non-isochronous, 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-isochronous, non-control receive endpoints, where the only differences are in EP_NUM register value set for the selection of the proper application RX data buffer in the read non-isochronous RX FIFO data routine (see Figure 13–29).

This flowchart does not attempt to document control endpoint 0 receive interrupts, which are discussed separately due to the more complex three-stage transfer mechanism used for control writes.

13.6.12 Non-Isochronous, Non-Control IN Endpoint Transmit Interrupt Handler

Figure 13–30 shows the operations necessary to handle non-isochronous, 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-isochronous, non-control transmit endpoints, where the only differences are in EP_NUM register value set for the routine selection of the application TX buffer (see Figure 13–31).

This flowchart does not attempt to document control endpoint 0 transmit interrupts, which are discussed separately due to the more complex three-stage transfer mechanism used for control reads.

13.6.13 SOF Interrupt Handler

SOF interrupts to the local host occur once per USB frame. The local host 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 13–32 shows the SOF ISR flowchart. The read isochronous RX FIFO data and write isochronous TX FIFO data procedures are shown in Figure 13–33 and Figure 13–34, respectively.

Figure 13–28. Non-Isochronous Non-Control Endpoint Receive Interrupt Handler

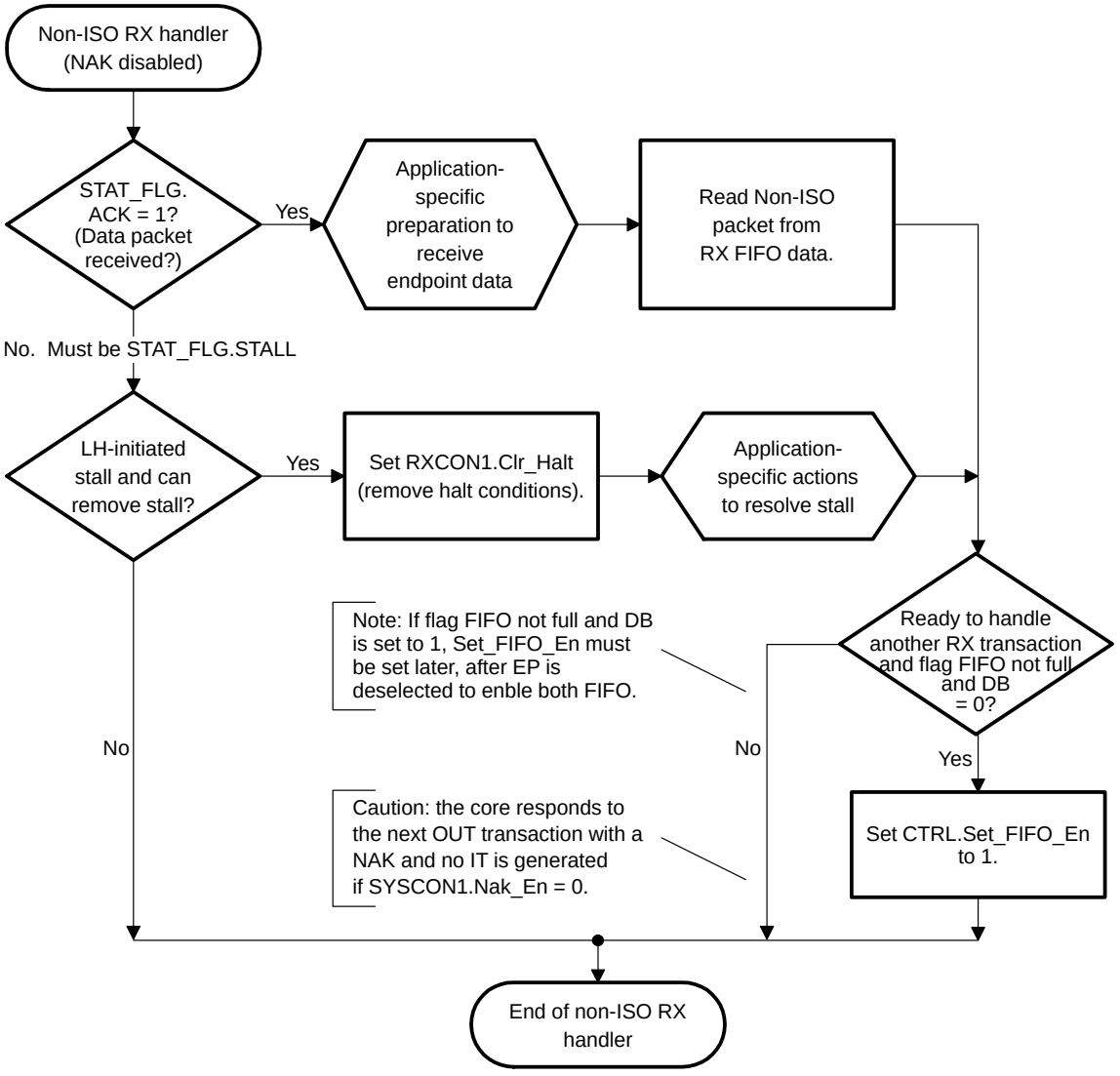


Figure 13–29. Read Non-Isochronous RX FIFO Data Flowchart

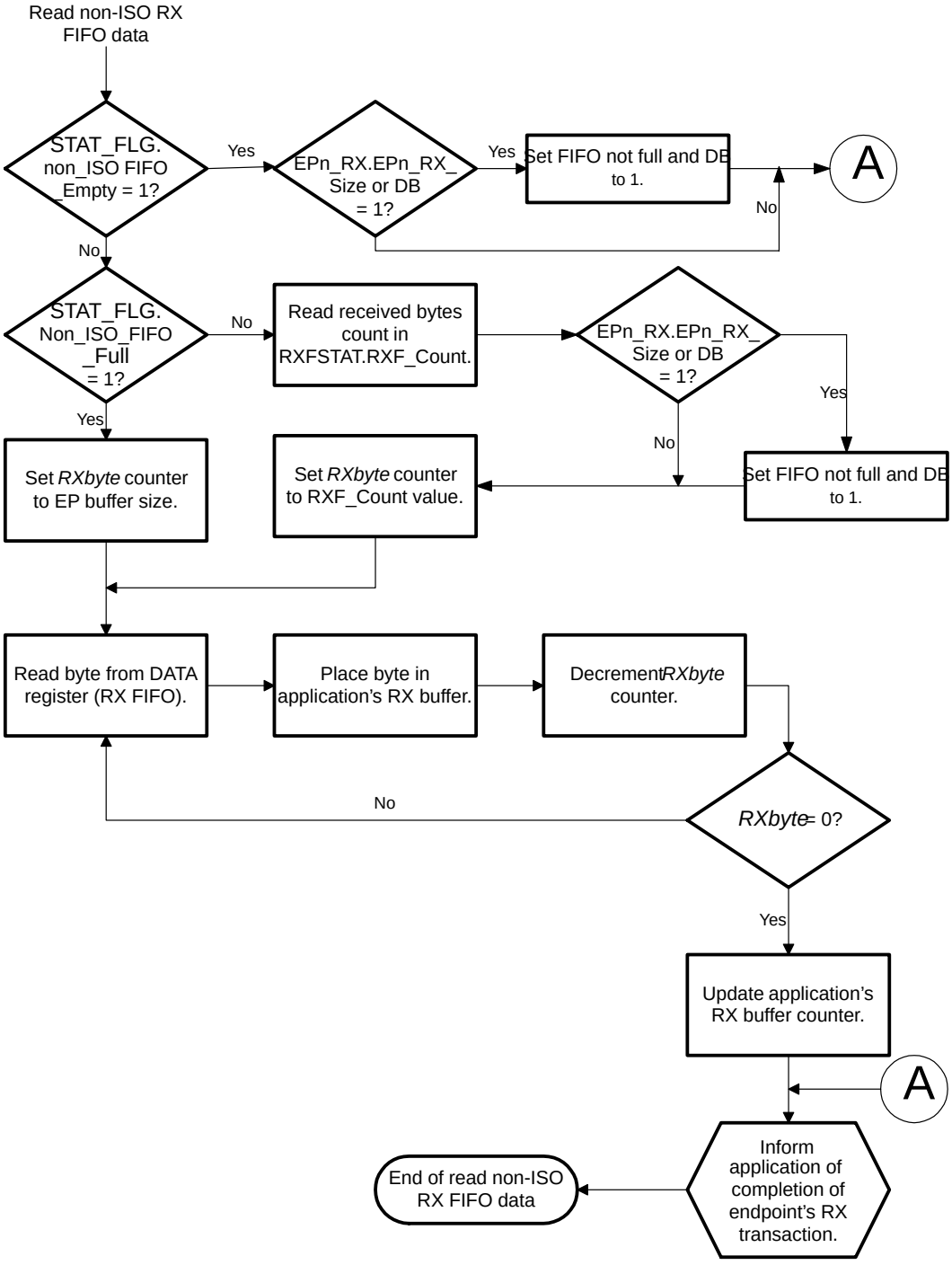


Figure 13–30. Non-Isochronous Non-control Endpoint Transmit Interrupt Handler

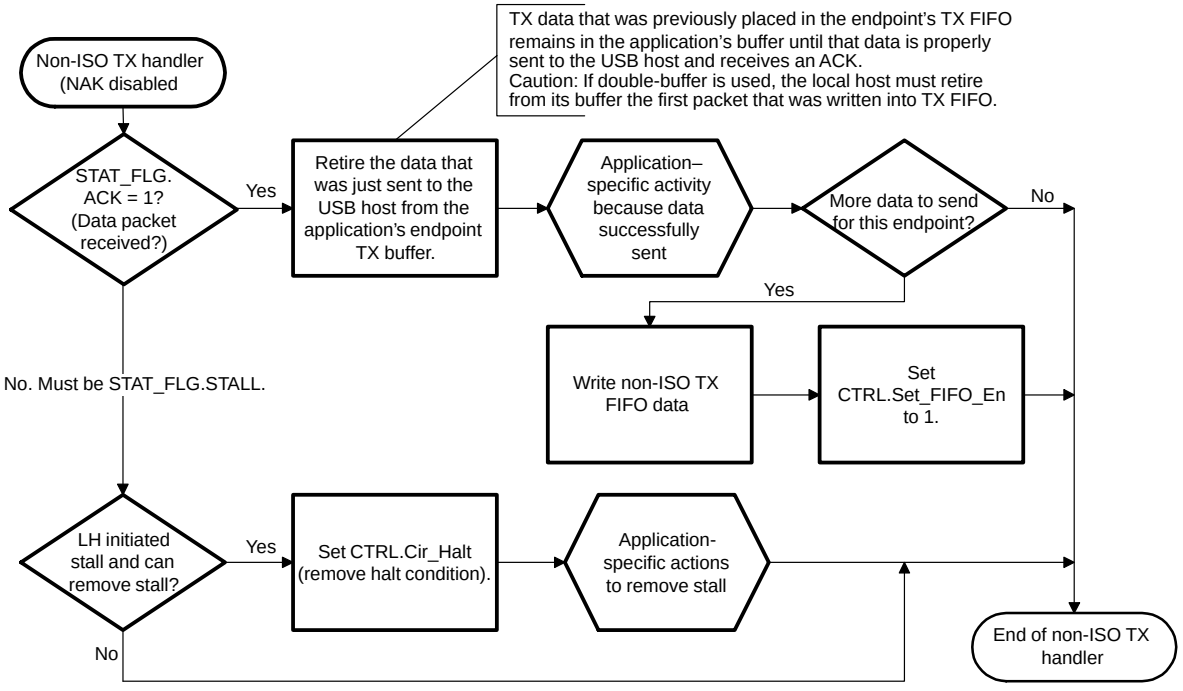


Figure 13–31. Write Non-Isochronous TX FIFO Data Flowchart

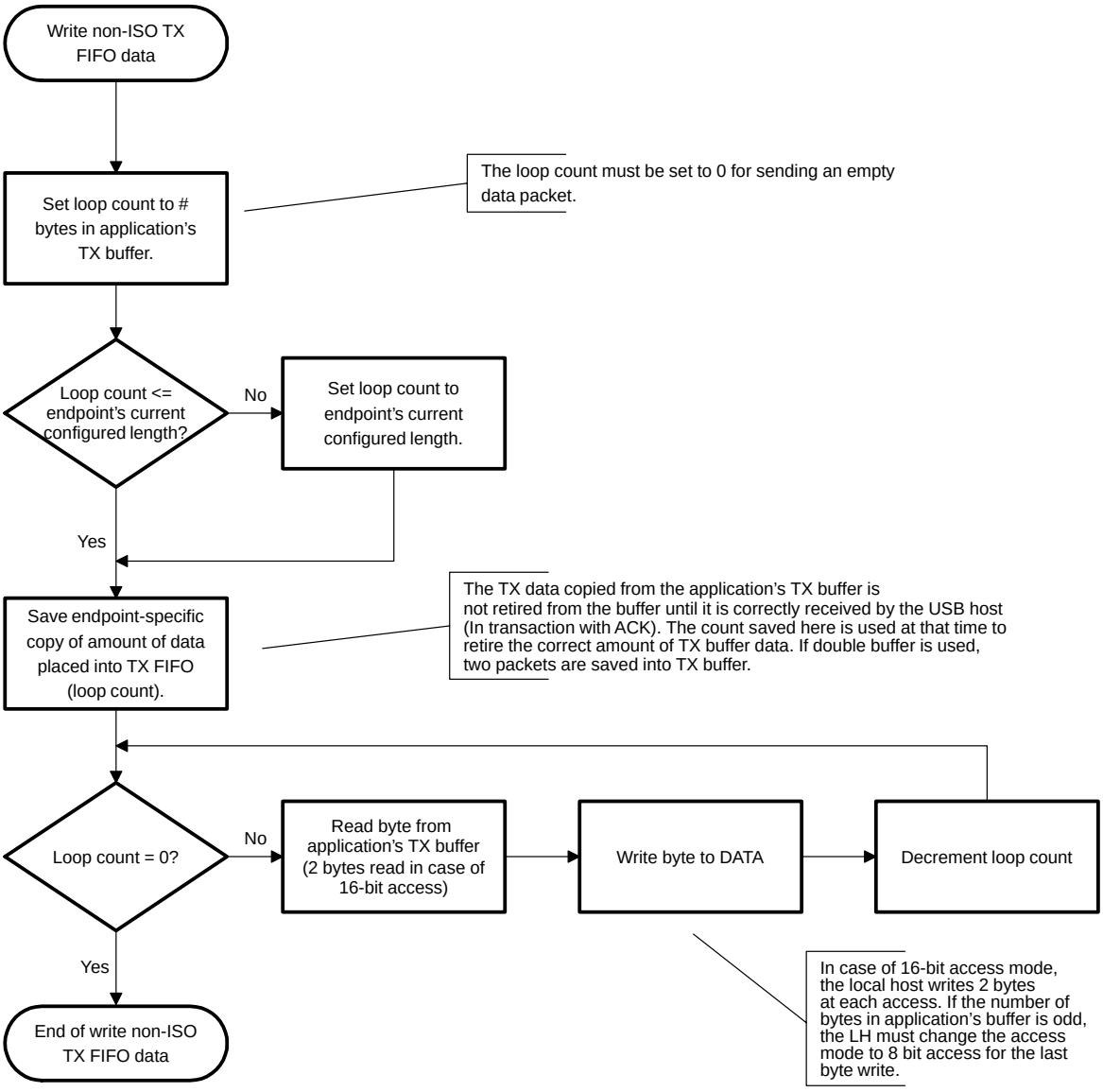


Figure 13–32. SOF Interrupt Handler Flowchart

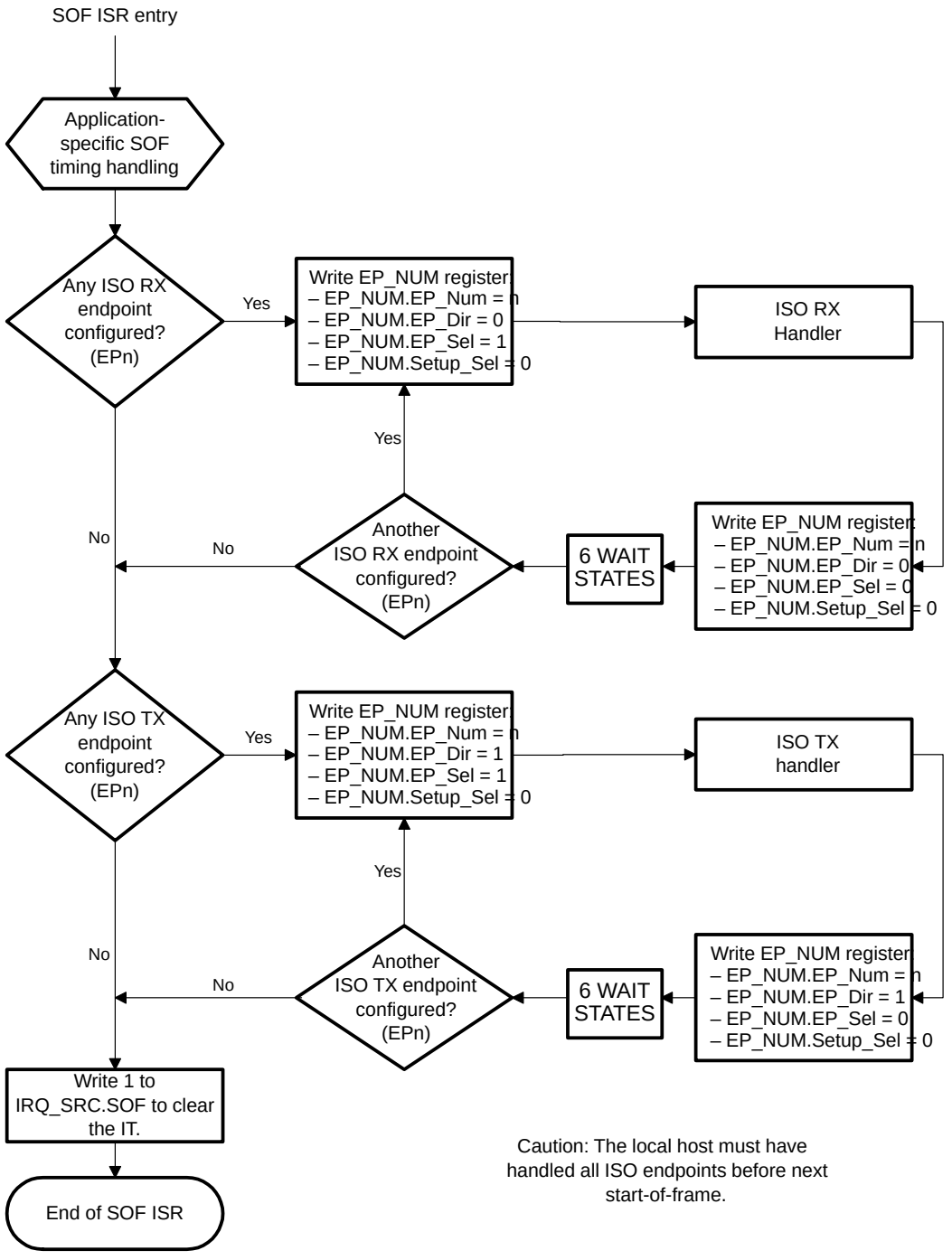


Figure 13–33. Read Isochronous RX FIFO Data Flowchart

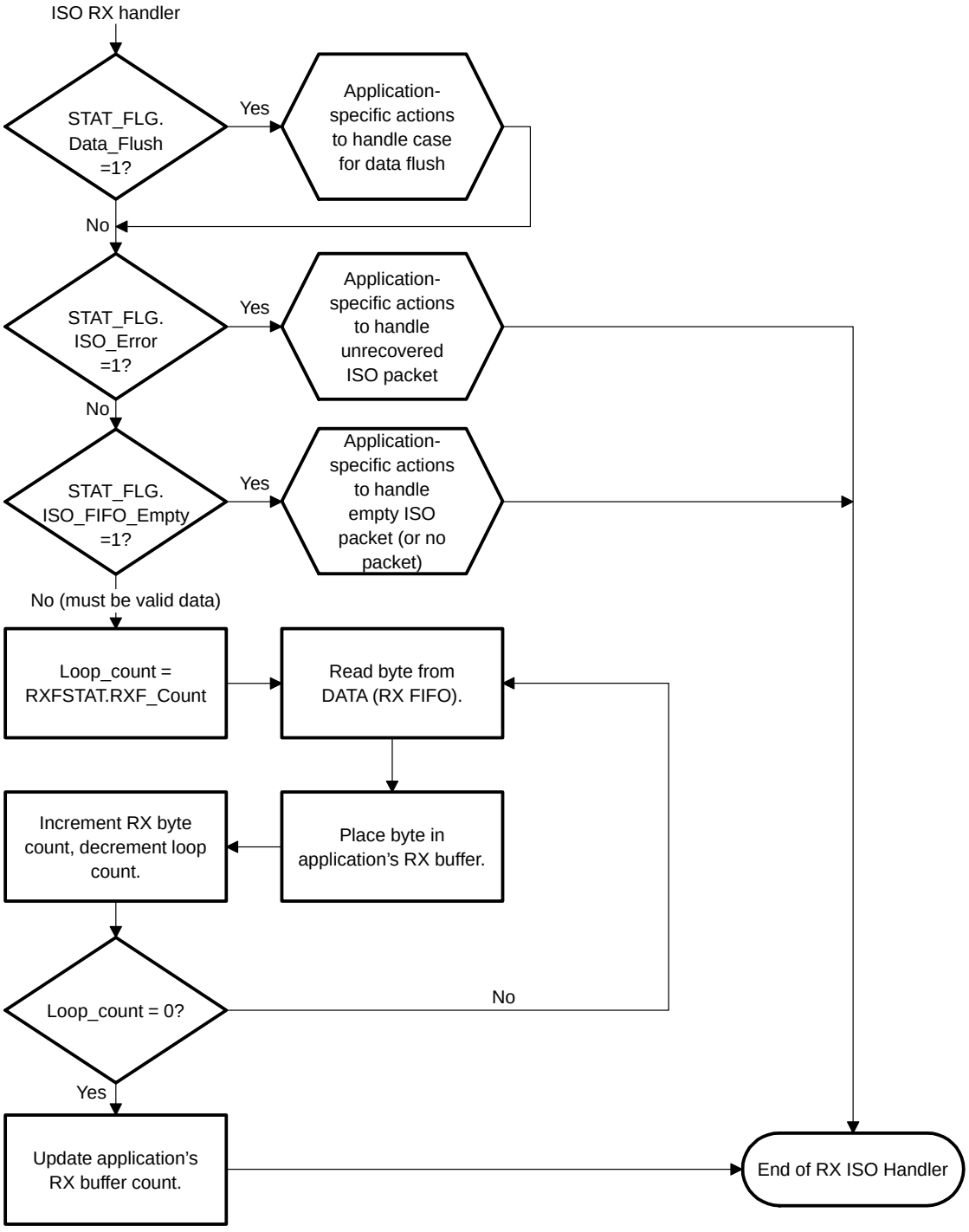
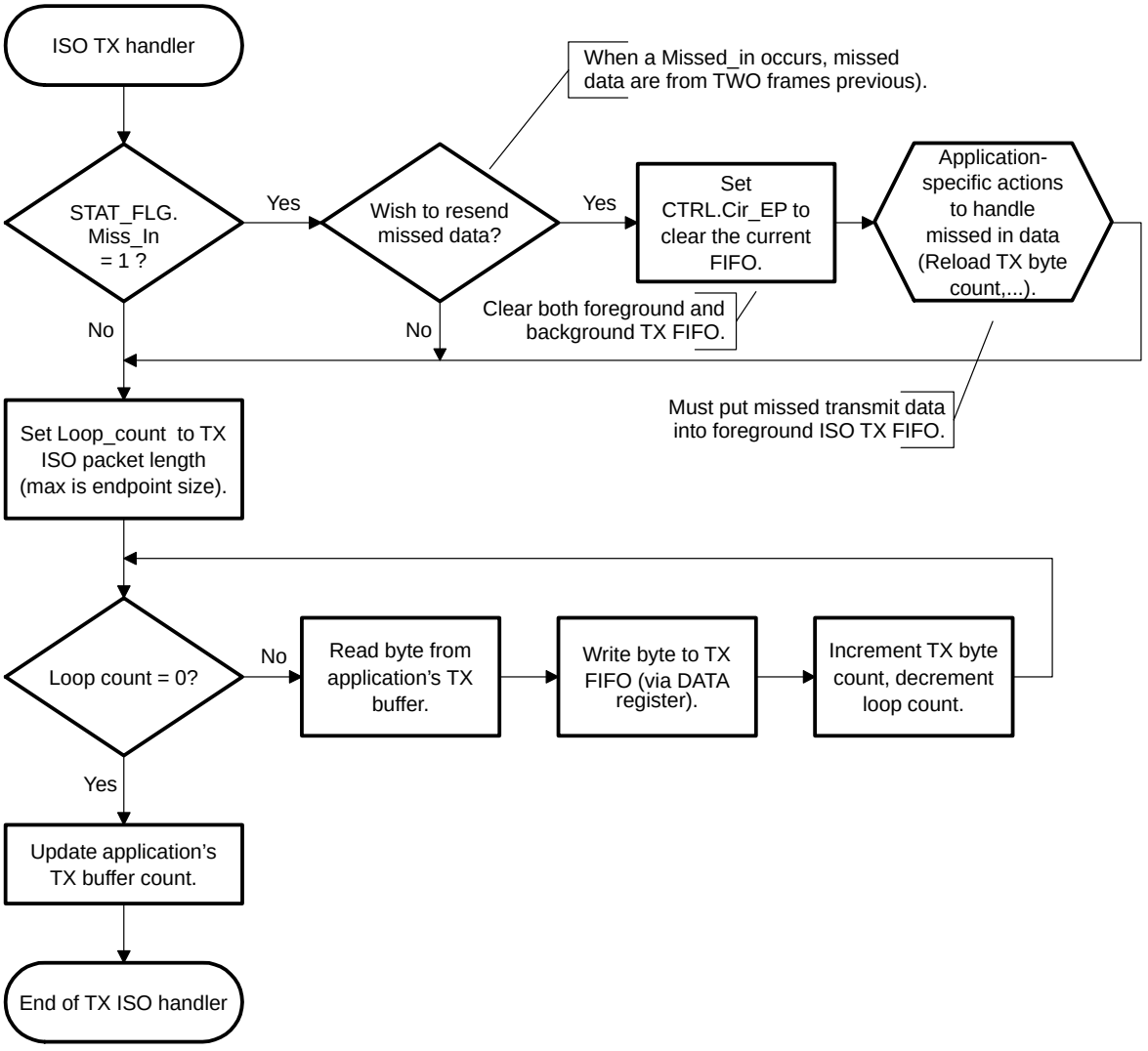


Figure 13–34. Write Isochronous TX FIFO Data Flowchart



13.6.14 Summary of USB-Related Interrupts

Table 13–26. USB Interrupt Type by Endpoint Type

	General USB IRQs				EP-Specific IRQs		SOF
Interrupt Type	**Setup (EP0)**	**Control (EP0) Out**	**Control (EP0) In**	**Other**	**Bulk or Interrupt Out**	**Bulk or Interrupt In**	**(Isochronous) SOF**
Transaction ACKd		√	√		√	√	
Transaction NAKed (if enabled)		√	√		√	√	
Transaction STALLed		√	√		√	√	
Setup	√						
SOF							√
Device State Changed				√			
RX DMA EOT (non_ISO)				√			
RX DMA Trans Count (non_ISO)				√			
TX DMA Done (non_ISO)				√			

13.7 DMA Operation

The USB function module provides support for six DMA channels. Three receive DMA channels are reserved for OUT transfers (isochronous or non-isochronous) and three transmit DMA channels are reserved for IN transfers (isochronous or non-isochronous). It is not possible to operate DMA transactions on control EP0.

> The local host must not access an endpoint used in a DMA transfer through EP_NUM, CTRL and STAT_FLG registers (in DMA, this remark applies after the local host has set the Set_FIFO_En bit to enable the RX DMA transfer). In particular, the local host must not set the halt feature while the endpoint is selected in the RXDMA_CFG register.

13.7.1 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 RXDMAn_EP fields (a 0 value means the DMA channel is deselected). Received OUT data must be read when a RX DMA request is active, through the register DATA_DMA. The RX FIFO accessed is that of the endpoint for which DMA request is active (only one RX DMA request active at a time).

13.7.2 Non-Isochronous OUT (USB HOST –> LH) DMA Transactions

During non-isochronous transfers to a DMA-operated OUT endpoint, a request to the local host DMA controller is generated when data has been placed into the endpoint FIFO and must be read. Notice that 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-isochronous OUT transfers.

- End of transfer interrupt (the 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 RXn_TC field, if Rxn_Stop bit has been set by the local host.

After an end of transfer interrupt, the local host must set the Set_FIFO_En for the endpoint to reenable the channel.

> **CAUTION**
>
> The local host must not initiate a new RX DMA transfer until it receives an end-of-transfer interrupt.

❑ Transactions count interrupt (the RXn_Cnt)

This interrupt performs watermark control. It can be used by the local host to monitor the file size of incoming transfers and take appropriate actions if, for instance, the file being received exceeds an expected size.

> **CAUTION**
>
> 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-isochronous endpoint. An RXn_Count interrupt is asserted even if RXn_Stop has been set: in that case, both RXn_Count and RXn_EOT interrupts are asserted.

The transaction count watermark is programmed in the RXDMAn register.

Figure 13–35. Non-Isochronous RX DMA Transaction Example (RX_TC = 2)

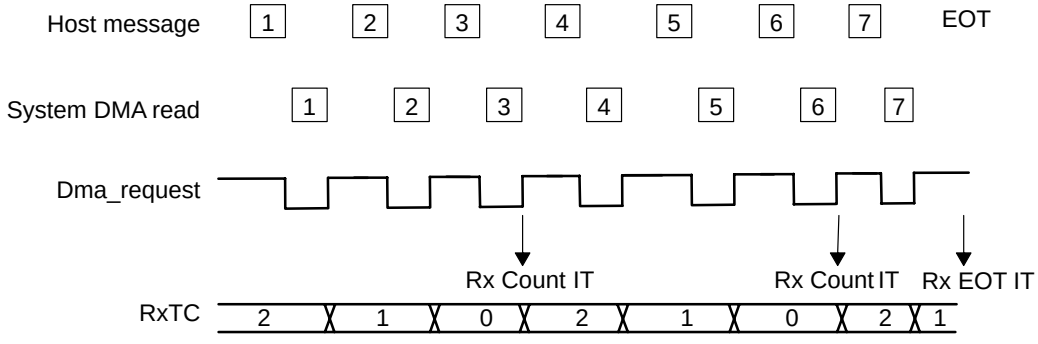


Figure 13–36. Non-Isochronous RX DMA Start Routine

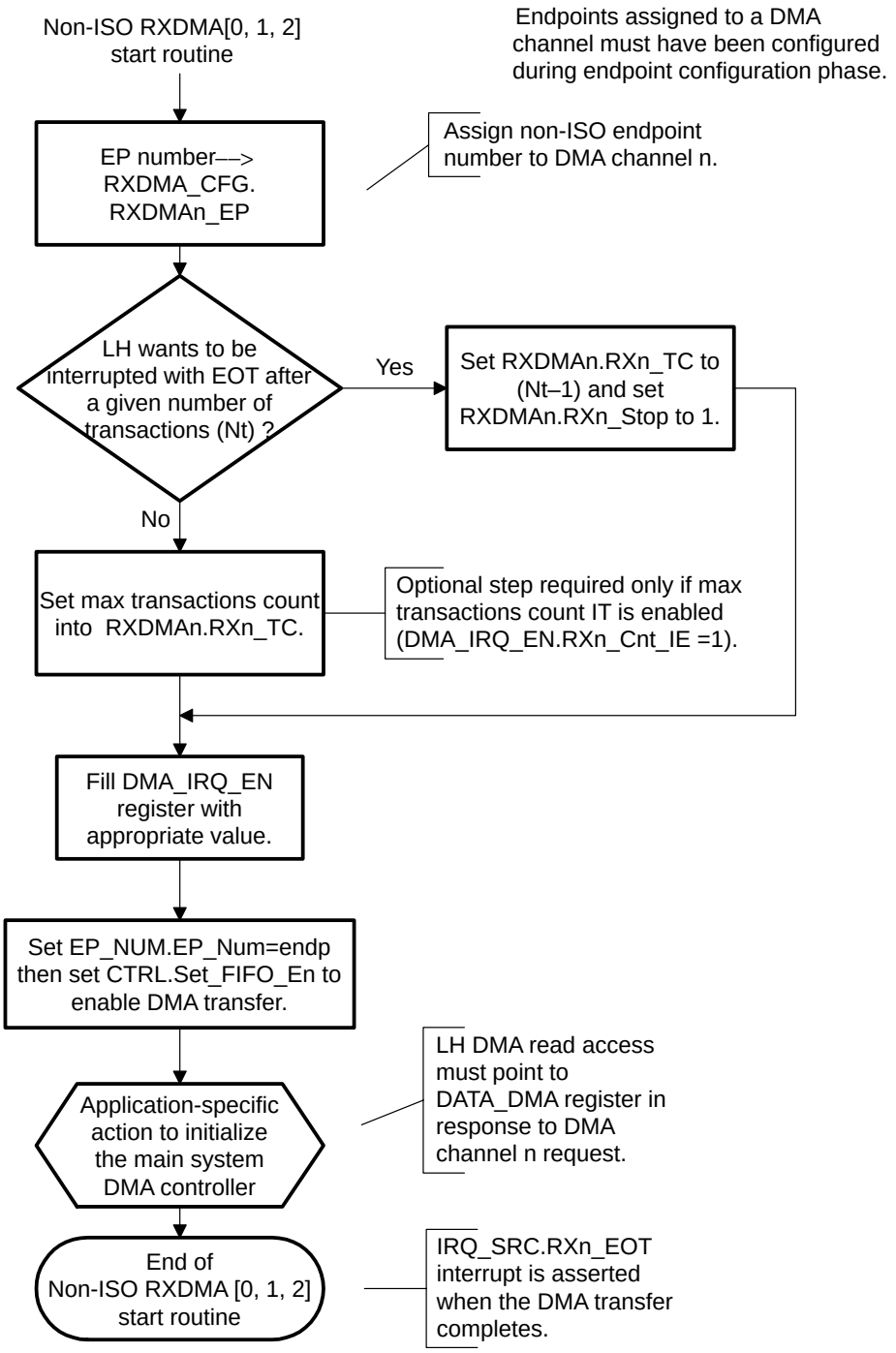


Figure 13–37. Non-Isochronous RX DMA EOT Interrupt Handler

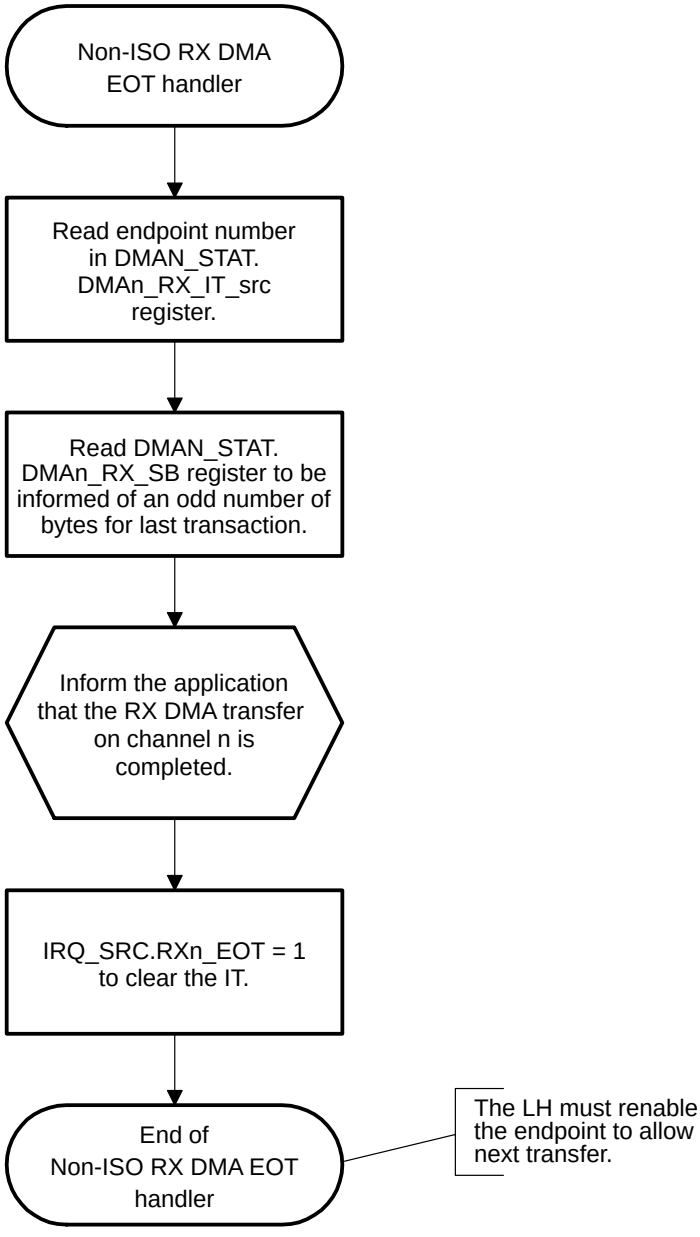
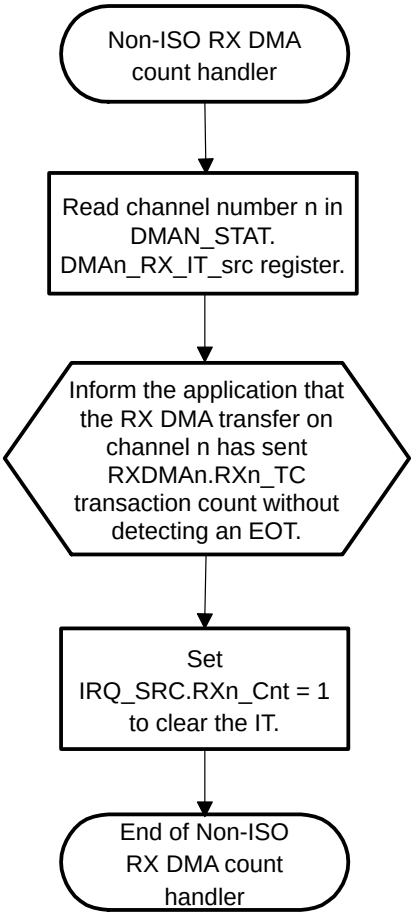


Figure 13–38. Non-Isochronous RX DMA Transaction Count Interrupt Handler



13.7.3 Isochronous OUT (USB HOST –> LH) DMA Transactions

During isochronous transfers to a DMA-operated OUT endpoint, a request to the local host 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 isochronous OUT endpoints.

Figure 13–39. Isochronous RX DMA Transaction

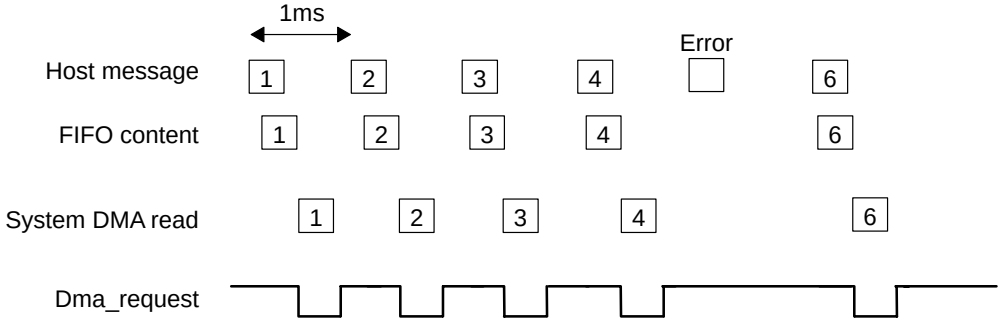
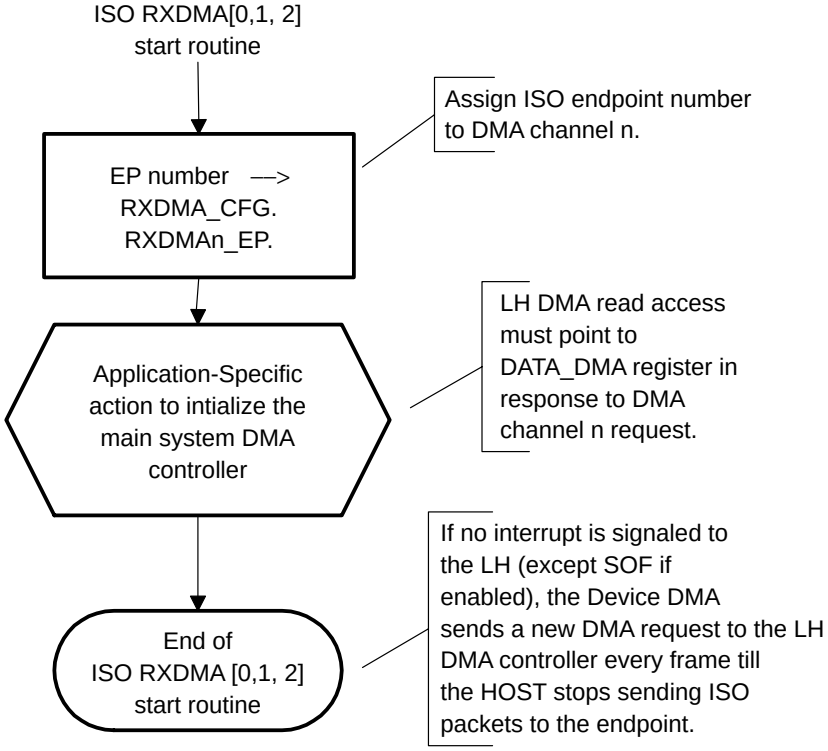


Figure 13–40. Isochronous RX DMA Start Routine



13.7.4 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 nonzero value in 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 isochronous or non-isochronous 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).

13.7.5 Non-Isochronous IN (LH –> USB HOST) DMA Transactions

Non-isochronous (bulk) TX DMA file transfers are virtually unlimited in size. The flowcharts depicted in Figure 13–42 and Figure 13–43 show how to handle small, medium, or large file transfers.

TXDMA0, TXDMA1, and TXDMA2 registers operate for non-isochronous endpoints in the following manner. The transfer size counter (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 local host 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 packets to be written, until 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.

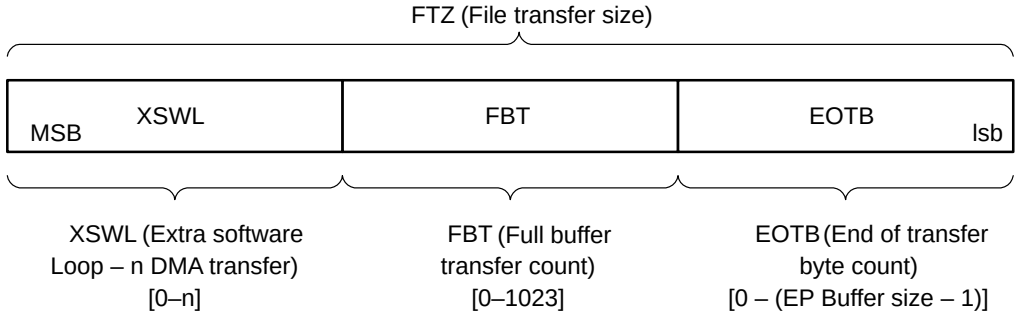
A DMA transmit transfer done interrupt is signaled to the local host after the last IN transaction completes successfully. This is after the START bit was set and after TXn_TSC equals 0 for the selected DMA channel.

> **CAUTION**
>
> The local host must not initiate a new TX DMA transfer until it receives a TX_Done interrupt.

Small file transfer less than 1024 bytes can be achieved in a single DMA pass signaled by a single interrupt completion. File size equal or greater than 1024 bytes needs two or more DMA passes signaled by an interrupt completion after each pass.

The size of the file to transfer (FTZ) can be assimilated conceptually as a concatenation of three arguments as shown in Figure 13–41.

Figure 13–41. File Transfer Size



The flowchart in Figure 13–42 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 local host via a DONE interrupt, as depicted in Figure 13–43. The start routine and the associated interrupt handler are tightly coupled.

An example would be 100603 bytes to transfer via 32 bytes IN bulk endpoint, which gives XSWL = 0x3, FBT = 0x47, EOTB = 0x1b. In this example, five passes of DMA transfer, signaled by five TXn_Done interrupts, are required, as follows:

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

Figure 13–42. Non-Isochronous TX DMA DMA Start Routine

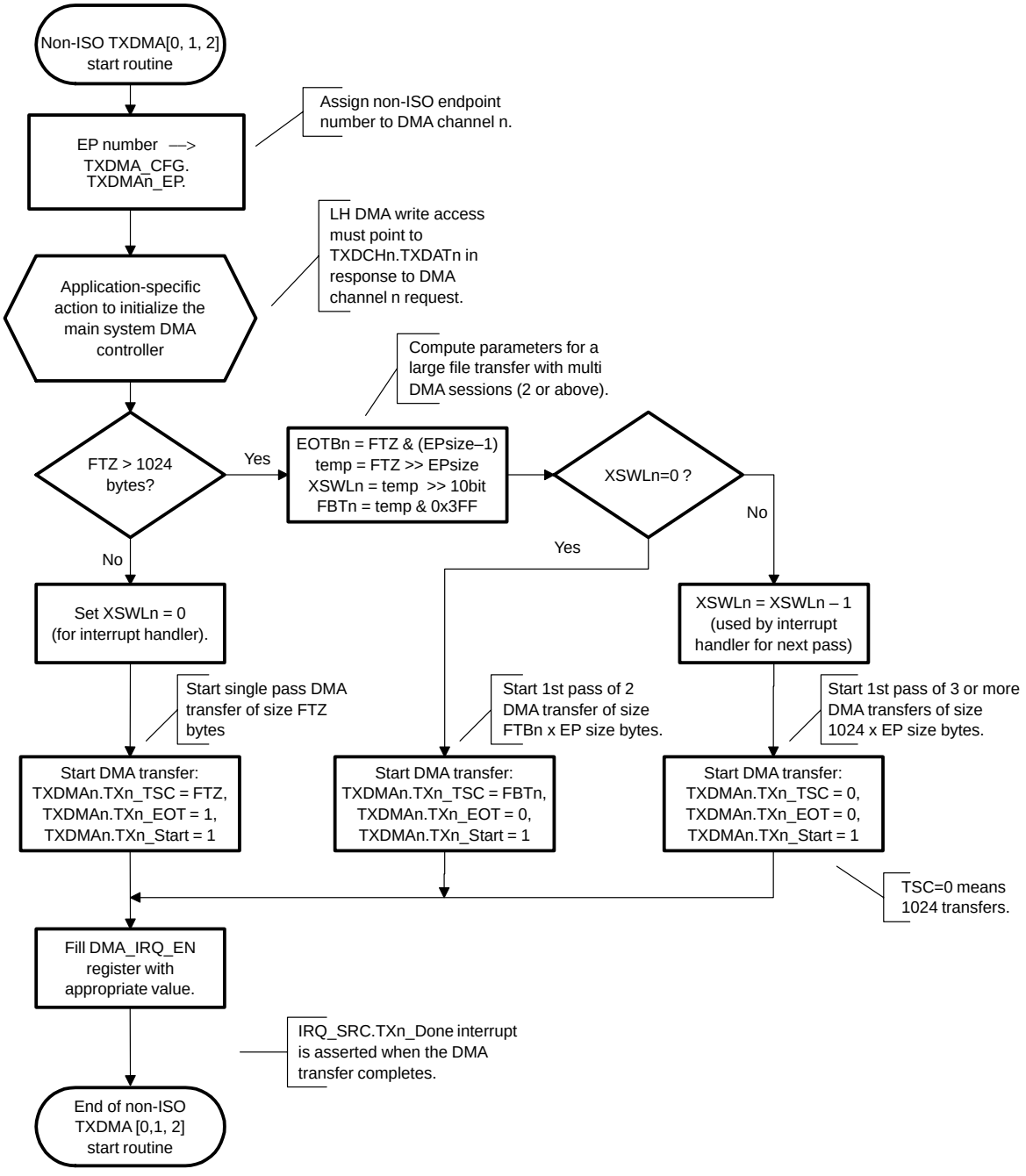
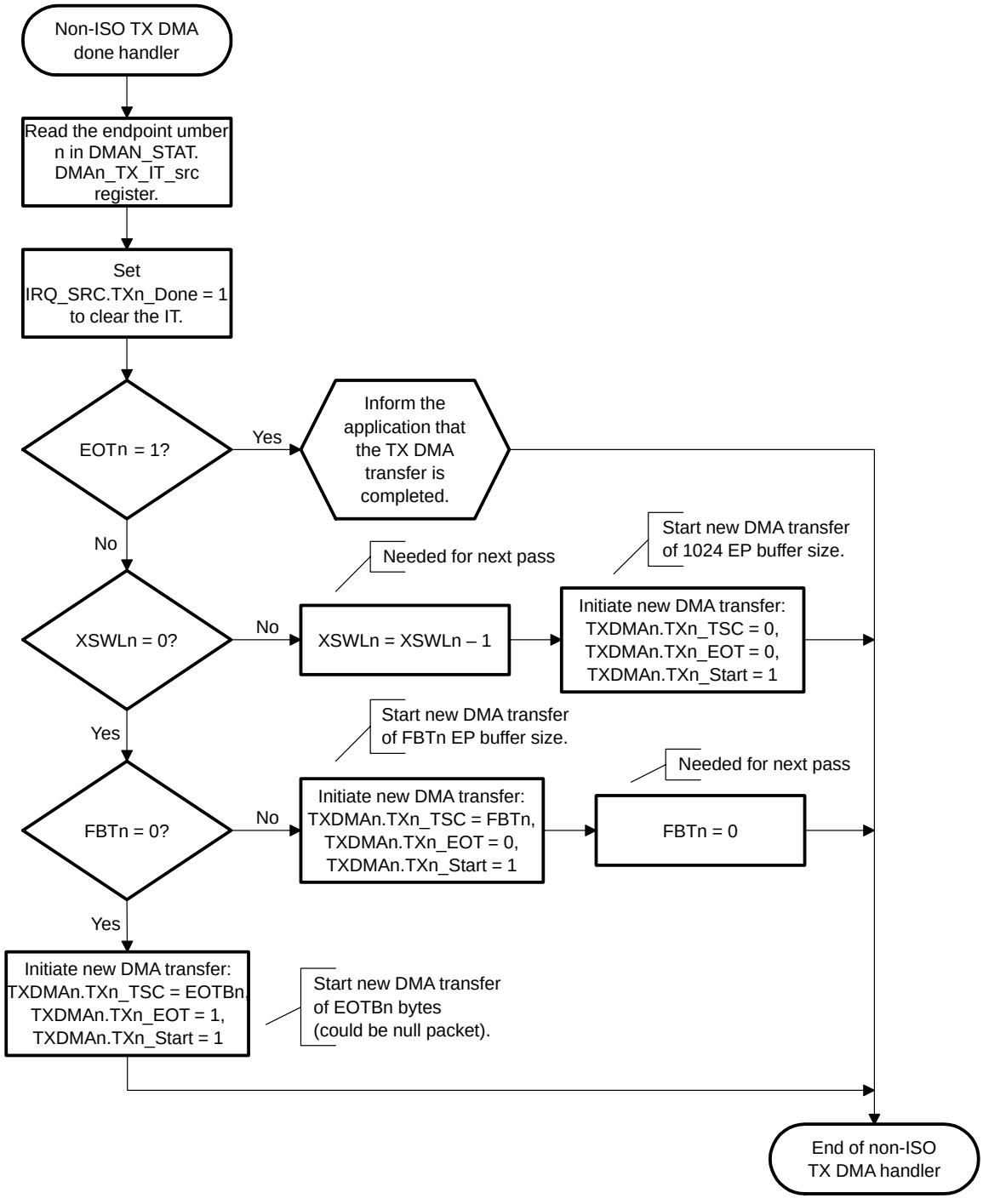


Figure 13–43. Non-Isochronous TX DMA Done Interrupt Handler



13.7.6 Isochronous IN (USB HOST –> LH) DMA Transactions

For isochronous endpoints (Figure 13–44), the transfer size counter (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 results are unpredictable.

A request to the local host main DMA controller is generated when the endpoint buffer is empty initially; after that, the START bit is set and then set again after each SOF (every 1 ms). The request is removed when the number of bytes written in the buffer matches TXn_TSC value.

During isochronous transfers to a DMA-operated IN endpoint, a request to the local host 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 DMA transfer.

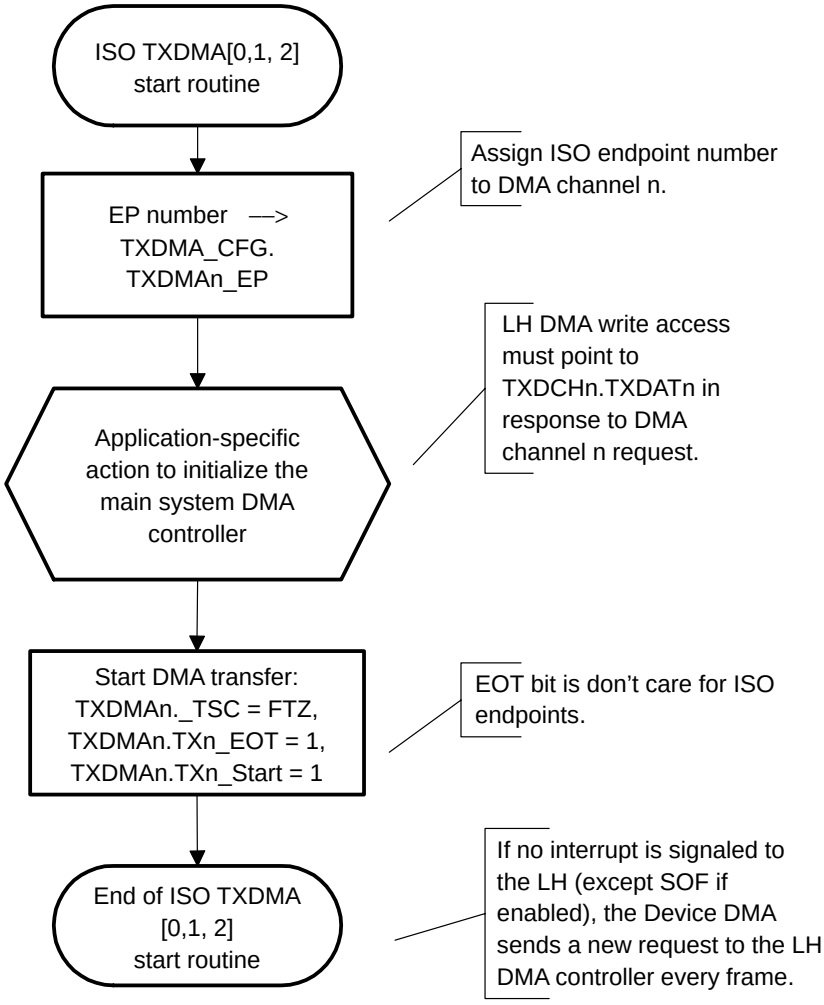
No interrupt is signaled to the local host during DMA operation to isochronous IN endpoints.

13.7.7 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; request must be serviced imperatively.

- For a USB TX DMA transfer, a DMA synchronization event is sent for each frame (DMA_CCR register FS field set to 1). Here the software decides the size of the USB transfer, so the DMA transfer is known in advance. The software must program the system DMA and the USB function to this transfer size.
- For a USB RX DMA transfer, a DMA synchronization event is sent for each element (DMA_CCR register FS field set to 0). Here the transfer size is unknown, and the DMA transfer is undetermined. Even if the USB function is programmed to transfer n elements, a RX DMA request is generated for each element.

Figure 13–44. Isochronous TX DMA Start Routine



13.7.8 Note on DMA Channel Deconfiguration

It is recommended that the local host wait for an EOT (RX) or a done (TX) interrupt before disabling the channel by writing a value 0 in TX/RXDMA_CFG register. However, if needed by the application, the local host can deselect the endpoint number in the TX/RXDMA_CFG register during a DMA transfer. The behavior is as follows:

- For RX transfer:

  If an RX DMA request is active for the endpoint when endpoint is unselected, deconfiguration is effective only at the end of the RX DMA request (that is, after all the DMA data has been read). When double buffering is used, the deconfiguration is effective after both buffers have been read (if both buffers were not empty at deselection). An EOT interrupt is asserted if an end-of-transfer is detected.

  If RX DMA request is not active when deselection occurs, the effect is immediate.

- For TX transfer:

  If request is active when the endpoint is unselected, deconfiguration is effective after the TX DMA request has been handled and the TX data has 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 TSC bit value is 0 after the transaction.

  If 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 TXDMAn_TSC value is 0 at this point, no TX_Done interrupt is asserted. If TX_Done interrupt had already been asserted when endpoint is deselected, configuration is effective only after the TX_Done interrupt handling.

  TX/RXDMAn_EP reflects endpoint value until deconfiguration is effective. The local host must read this register to know if channel has been disabled or not yet. It must wait until read value is 0 before performing other actions to the endpoint. After effective deconfiguration, all transactions to the endpoint generates an endpoint-specific interrupt (if non-transparent).

  If the selected endpoint is isochronous, deconfiguration is effective after the TX/RX request has been serviced and the following isochronous transactions are handled at SOF interrupt through endpoint registers (EP_NUM and STAT_FLG).

13.8 Power Management

The flowchart in Figure 13–45 shows the values assigned to USB function signals concerned with power management in the functioning of the device state. These signals are:

- PUEN_O

  Pullup enable signal, which always reflects the Pullup_En register bit.

- SHUTOFF_O

  Power circuitry shutoff signal, controlled by the core and the SOFF_Dis bit.

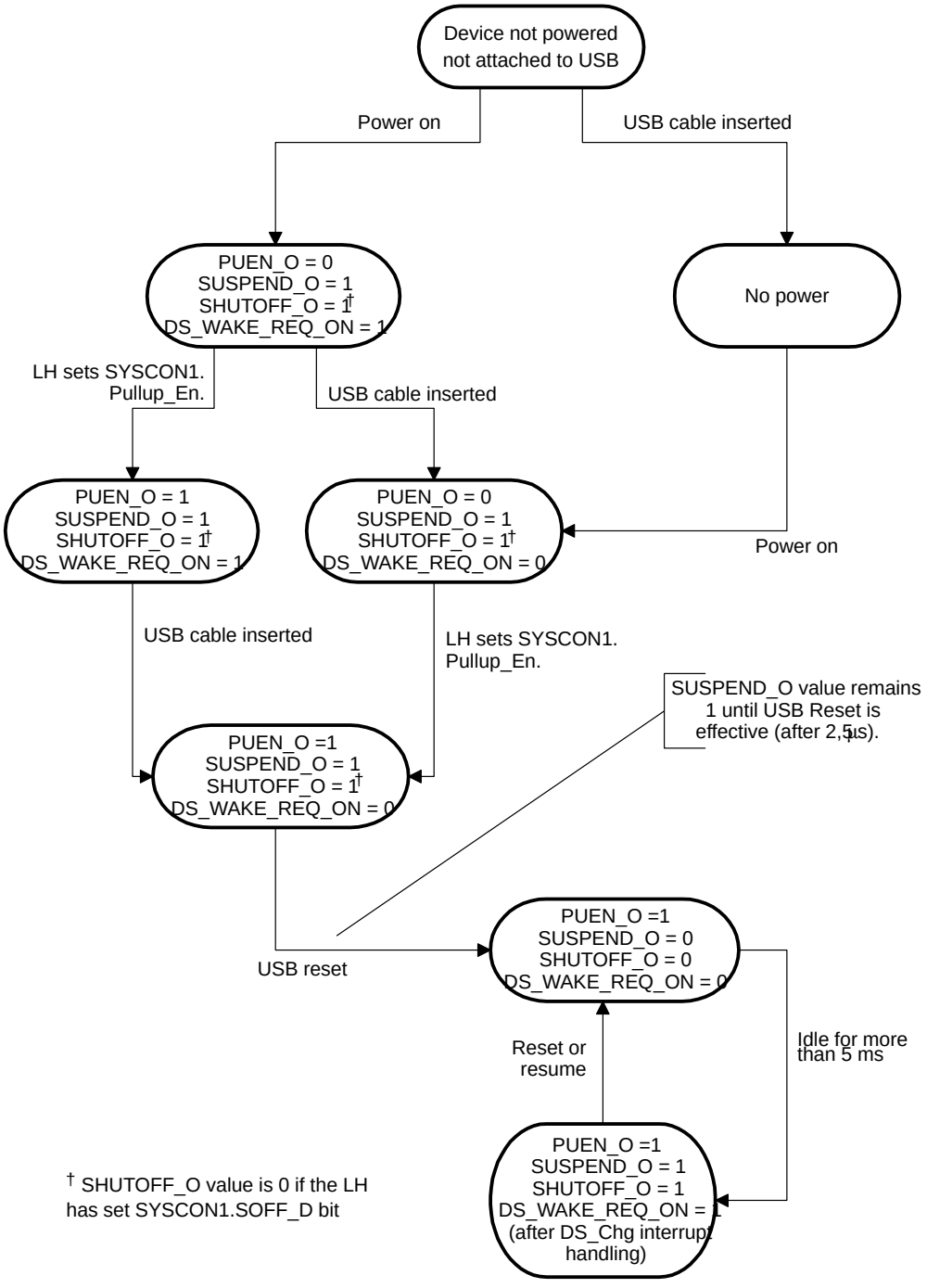
- DS_WAKE_REQ_ON

  Deep sleep wake request, asserted low when interface clock is needed.

- SUSPEND_O

  Suspend signal, asserted high when the device is in suspend mode.

Figure 13–45. Power Management Signal Values



Chapter 14

Universal Serial Bus Host

This chapter describes the universal serial bus (USB) host of the OMAP5910 multimedia processor.

Topic **Page**

14.1 USB Host Controller

The OMAP5910 USB host controller (HC) is a three-port controller that communicates with USB devices at the USB low-speed (1.5M bit/s maximum) and full-speed (12M bit/s maximum) data rates. It is compatible with the *Universal Serial Bus Specification Revision 1.1* 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 OMAP5910 USB host controller are already familiar with the *USB Specification* and *OHCI Specification for USB*.

The OMAP5910 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 OMAP5910 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.

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 OMAP5910 USB host controller accesses these data structures without direct intervention by the processor using the OMAP5910's local bus. These data structures and data buffers can be located in internal or external system RAM. The local bus MMU allows the USB host controller to access the full address range of internal and external memories.

The OMAP5910 USB host controller is connected to the OMAP5910 MPU public peripheral bus for MPU access to registers. The USB host controller gains access to the data structures in OMAP5910 system memory via the internal OMAP5910 local bus (LB) interface. 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.

Flexible multiplexing of signals from the OMAP5910 USB host controller, the OMAP5910 USB function controller, and other OMAP5910 peripherals allows a wide variety of system-level USB functions. Notice in Figure 14–2 that many of the pins can be used for USB-related signals or for signals from other OMAP5910 peripherals. The OMAP5910 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 OMAP5910 USB signal multiplexing selects how the signals associated with the three OMAP5910 USB host ports and the OMAP5910 USB function controller can be brought out to OMAP5910 pins. Notice that the

OMAP5910 pins associated with the integrated USB transceiver are only available for use by the OMAP5910 USB host controller or OMAP5910 USB function controller and are not shared with any other peripherals.

Figure 14–1 shows the OMAP5910 device with the USB host controller highlighted. Figure 14–2 shows the OMAP5910 USB host controller.

Figure 14–1. OMAP5910 USB Host Controller Block Diagram

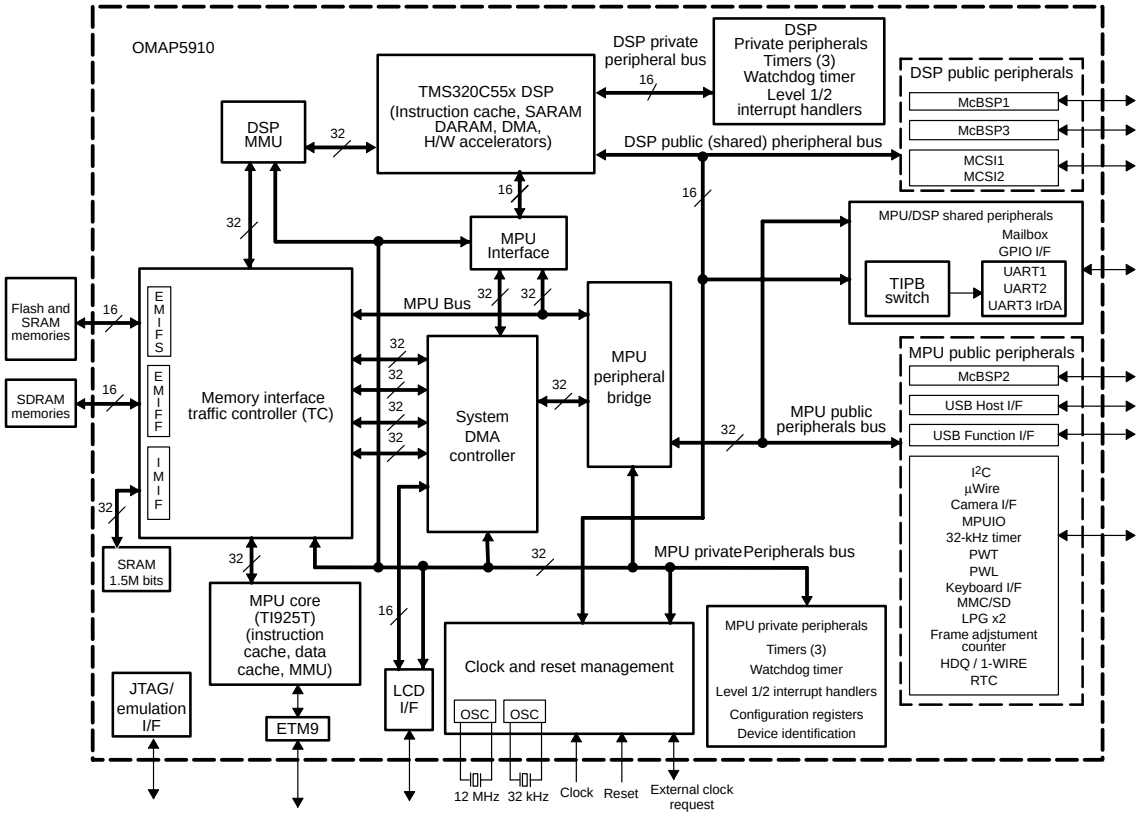
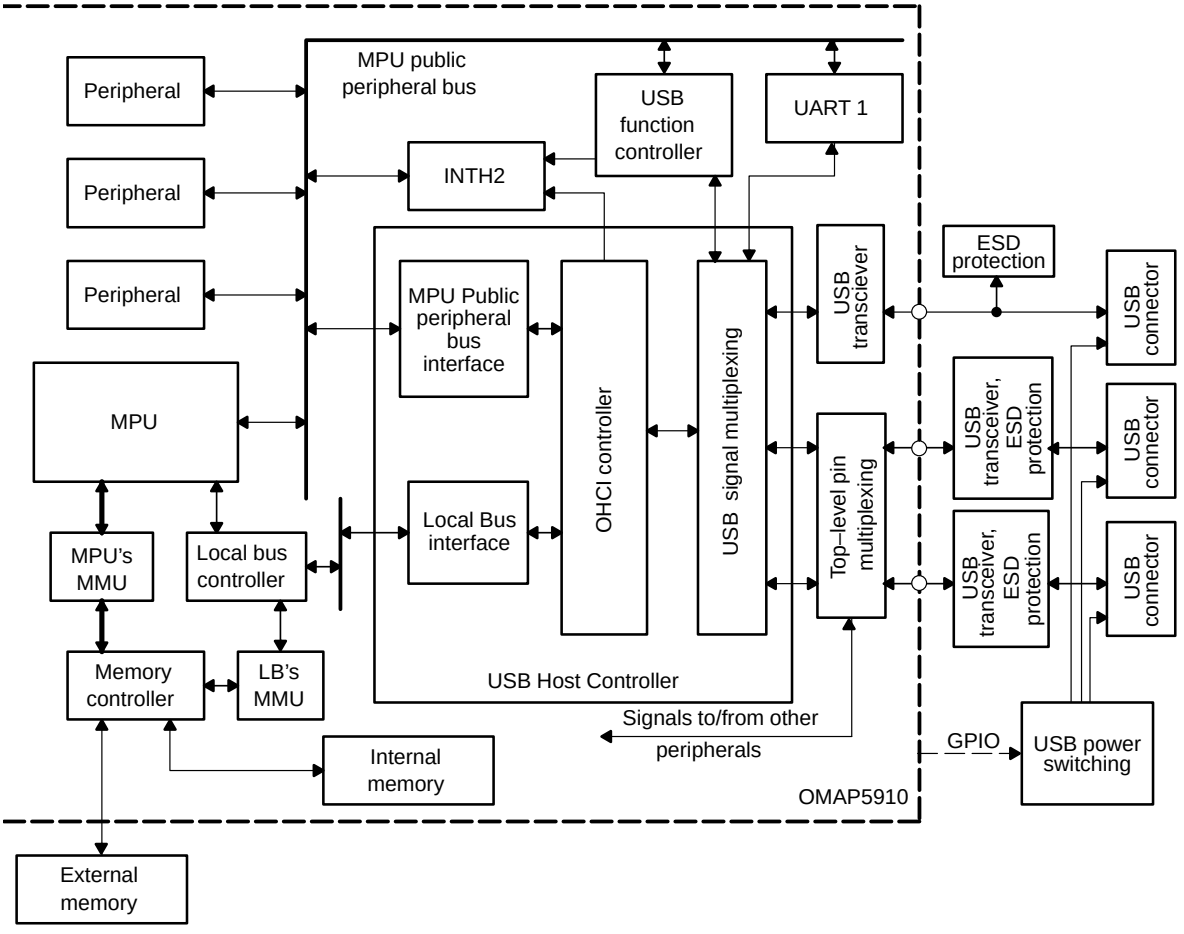


Figure 14–2. OMAP5910 USB Host Controller



14.2 USB Open Host Controller Interface Functionality

14.2.1 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 1.1*, define most of the USB functionality that the OMAP5910 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*. OMAP5910 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.

14.2.2 OMAP5910 USB Host Controller Differences from *OHCI Specification for USB*

The OMAP5910 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 OMAP5910 system memory addressing mechanisms and the effects of OMAP5910 pin multiplexing options.

14.2.2.1 Power Switching Output Pins Not Supported

The OMAP5910 device does not provide pins that can be controlled directly by the USB host controller OHCI port power control features. The OHCI RhPortStatus(n) 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 other GPIO pins or implementation-specific control mechanisms to control VBUS switching.

14.2.2.2 Overcurrent Protection Input Pins Not Supported

The OMAP5910 device does not provide any pins that allow the USB host controller OHCI RhPortStatus(n) overcurrent protection status bits to be directly controlled by external hardware.

Users can use GPIO pins or other implementation-specific control mechanisms to report port overcurrent information to the USB host controller driver.

14.2.2.3 HMC_MODE and Top-Level Pin Multiplexing and OHCI Registers

The USB signal multiplexing modes provide selections where 0, 1, 2, or 3 USB host controller ports can be brought to OMAP5910 pins. The OHCI RhDescriptorA register always reports three available USB host ports, regardless of the CONF_MOD_USB_HOST_HMC_MODE_R field of the MOD_CONF_CTRL_0 register or top-level pin multiplexing settings. When the CONF_MOD_USB_HOST_HMC_MODE_R field setting of the MOD_CONF_CTRL_0 register disables a USB host controller port, the USB host controller sees that port as unattached.

When OMAP5910 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.

14.2.2.4 No Ownership Change Interrupt

The OMAP5910 USB host controller does not implement the OHCI ownership change interrupt.

14.2.2.5 Valid Address Ranges for Pointers to Data Structures

The mechanism that allows the OMAP5910 USB host controller to access USB endpoint descriptor (ED), transfer descriptor (TD) , and HCCA data structures in system memory places certain requirements on the registers that point to data structures in system memory and on the pointers within those data structures. Details can be found in Section 14.6, *USB Host Controller Access to System Memory*.

14.2.3 OMAP5910 Implementation of *OHCI Specification for USB*

14.2.3.1 Isochronous TD OFFSETX/PSWX Values

The OMAP5910 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.

14.2.3.2 OMAP5910 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 local bus addresses of valid endpoint descriptors.

The OMAP5910 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 OMAP5910 USB host controller attempts to access using the local bus virtual address of 0, which causes an unrecoverable error. Registers HostUEAddr, HostUEStatus, and HostTimeoutCtrl are updated in this case.

14.3 USB Host Controller Registers

Most of the OMAP5910 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.

Note:

The USB host controller registers must be accessed using 32-bit data operations. Use of smaller data access sizes may result in unexpected operation of the USB host controller. The USB host controller registers and the USB host controller data structures are organized for little endian operation mode because the TI925T MPU processor on the OMAP5910 device must use little endian mode.

The OMAP5910 USB host controller registers are listed in Table 14–1. Table 14–2 through Table 14–29 describe specific register bits.

Table 14–1. 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	32	FFFB:A014h
HcHCCA	Local bus virtual address of HCCA‡	R/W	32	FFFB:A018h
HcPeriodCurrentED	Local bus virtual address of current periodic endpoint descriptor‡	R/W	32	FFFB:A01Ch
HcControlHeadED	Local bus virtual address of head of control endpoint descriptor list‡	R/W	32	FFFB:A020h
HcControlCurrentED	Local bus virtual address of current control endpoint descriptor‡	R/W	32	FFFB:A024h

† 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 local bus virtual addresses used in these registers. See Section 14.6.1, *Local Bus Addressing*.
§ This register provides control and status for the OMAP5910 pins associated with the USB transceiver for some HMC_MODE values.
¶ This register provides control and status for the OMAP5910 pins associated with USB port 1 for some HMC_MODE values.
# This register provides control and status for the OMAP5910 pins associated with USB port 2 for some HMC_MODE values.

Table 14–1. USB Host Controller Registers (Continued)

Name	Description	R/W	Size†	Address
HcBulkHeadED	Local bus virtual address of head of bulk endpoint descriptor list‡	R/W	32	FFFB:A028h
HcBulkCurrentED	Local bus virtual of current bulk endpoint descriptor‡	R/W	32	FFFB:A02Ch
HcDoneHead	Local bus virtual 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
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 timeout 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 local bus virtual addresses used in these registers. See Section 14.6.1, *Local Bus Addressing*.
§ This register provides control and status for the OMAP5910 pins associated with the USB transceiver for some HMC_MODE values.
¶ This register provides control and status for the OMAP5910 pins associated with USB port 1 for some HMC_MODE values.
# This register provides control and status for the OMAP5910 pins associated with USB port 2 for some HMC_MODE values.

The other revision number register reports the revision number of the *OHCI Specification for USB* upon which the USB host controller is based.

Table 14–2. OHCI Revision Number Register (HcRevision)

Bit	Name	Description	Type	Reset Value
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 controls the operating mode of the USB host controller.

Table 14–3. HC Operating Mode Register (HcControl)

Bit	Name	Value	Description	Type	Reset Value
31–11	Reserved		Reserved		
10	RWE		Remote wake-up enable. This bit has no effect in OMAP5910. The OMAP5910 USB host controller does not provide a processor wake-up mechanism.	R/W	0
9	RWC		Remote wake up connected. This bit has no effect in OMAP5910. The OMAP5910 USB host controller does not provide a processor wake-up mechanism.	R/W	0
8	IR		Interrupt routing. The OMAP5910 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

Table 14–3. HC Operating Mode Register (HcControl) (Continued)

Bit	Name	Value	Description	Type	Reset Value
7–6	HCFS		Host controller functional state:	R/W	00
		00	USBReset		
		01	USBResume		
		10	USBOperational		
		11	USBSuspend		
			A transition to USBOperational causes SOF generation to begin in 1 ms. The USB host controller may automatically transition from USBSuspend to USBResume if a downstream resume is received. The USB host controller enters USBSuspend after a software reset. The USB host controller enters USBReset after a hardware reset. The USBReset state resets the root hub and causes downstream signaling of USBReset.		
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 HcBulk-CurrentED to point to an ED still on the list before it reenables 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
		0	Control ED list is not processed in the next 1-ms frame. Host controller driver may modify the control ED list. If driver removes the ED pointed to bythe HcControlCurrentED from the ED list, it must updateHcControlCurrentED to point to an ED still on the list before it reenables 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		

Table 14–3. HC Operating Mode Register (HcControl) (Continued)

Bit	Name	Value	Description	Type	Reset Value
3	IE		Isochronous enable	R/W	0
		0	Enables processing of isochronous EDs.		
		1	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 might not occur in the current frame but is enabled in the next frame.		
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 14–4. HC Command and Status Register (HcCommandStatus)

Bit	Name	Description	Type	Reset Value
31–18	Reserved	Reserved		
17–16	SOC	Scheduling overrun count 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.	R	00
15–4	Reserved	Reserved		
3	OCR	Ownership change request This bit is set by the host controller driver to gain ownership of the host controller. OMAP5910 does not support SMI interrupts, so no ownership change interrupt occurs.	R/W	0
2	BLF	Bulk list filled 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.	R/W	0
1	CLF	Control list filled 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.	R/W	0
0	HCR	Host controller reset 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 USBSuspend 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.	R/W	0

The HC interrupt status register reports the status of the USB host controller internal interrupt sources.

Table 14–5. HC Interrupt Status Register (HcInterruptStatus)

Bit	Name	Description	Type	Reset Value
31	Reserved	Reserved		
30	OC	Ownership change The OMAP5910 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 local 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

Table 14–5. HC Interrupt Status Register (HcInterruptStatus) (Continued)

Bit	Name	Description	Type	Reset Value
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
1	WDH	Write done head When 1 indicates that the USB host controller has updated the 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 enables various OHCI interrupt sources to generate interrupts to the OMAP5910 level 2 interrupt handler.

Table 14–6. HC Interrupt Enable Register (HcInterruptEnable)

Bit	Name	Description	Type	Reset Value
31	MIE	Master interrupt enable When 1, allows other enabled OHCI interrupt sources to propagate to the OMAP5910 level 2 interrupt controller. When 0, OHCI interrupt sources are ignored and no USB host controller interrupts are propagated to the OMAP5910 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 OMAP5910.	R	0
29–7	Reserved	Reserved		

Table 14–6. HC Interrupt Enable Register (HcInterruptEnable) (Continued)

Bit	Name	Description	Type	Reset Value
6	RHSC	Root hub status change When 1 and MIE is 1, allows root hub status change interrupts to propagate to the OMAP5910 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
5	FNO	Frame number overflow When 1 and MIE is 1, allows frame number overflow interrupts to propagate to the OMAP5910 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 OMAP5910 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 OMAP5910 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

Table 14–6. HC Interrupt Enable Register (HcInterruptEnable) (Continued)

Bit	Name	Description	Type	Reset Value
2	SF	Start of frame When 1 and MIE is 1, allows start of frame interrupts to propagate to the OMAP5910 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
1	WDH	Write done head When 1 and MIE is 1, allows write done head interrupts to propagate to the OMAP5910 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 OMAP5910 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 14–7. HC Interrupt Disable Register (HcInterruptDisable)

Bit	Name	Description	Type	Reset Value
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 OMAP5910.	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
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

Table 14–7. HC Interrupt Disable Register (HcInterruptDisable) (Continued)

Bit	Name	Description	Type	Reset Value
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 local bus virtual address of the beginning of the HCCA.

Table 14–8. HC HCAA Address Register (HcHCCA)

Bit	Name	Description	Type	Reset Value
31–8	HCCA	See Section 14.6.1, *Local Bus Addressing,* for the restrictions on local bus virtual addresses.	R/W	0
7–0	Reserved	Reserved	R	0

The HC current periodic register defines the local bus virtual address of the next endpoint descriptor (ED) on the periodic ED List.

Table 14–9. HC Current Periodic Register (HcPeriodCurrentED)

Bit	Name	Description	Type	Reset Value
31–4	PCED	Local bus virtual address of current ED on the periodic ED list. This field represents bits 31:4 of the local bus virtual 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 14.6.1, *Local Bus Addressing,* for the restrictions on local bus virtual addresses.	R/0	0x0000 000
3–0	Reserved	Reserved	R	0x0

The HC head control register defines the local bus virtual address of the head ED of the control ED list.

Table 14–10. HC Head Control Register (HcControlHeadED)

Bit	Name	Description	Type	Reset Value
31–4	CHED	Local bus virtual address of head ED on the control ED list This field represents bits 31:4 of the local bus virtual 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 See Section 14.6.1, *Local Bus Addressing,* for the restrictions on local bus virtual addresses.	R/W	0
3–0	Reserved	Reserved	R	0x0

The HC current control register defines the local bus virtual address of the next ED on the control ED list.

Table 14–11. HC Current Control Register (HcControlCurrentED)

Bit	Name	Description	Type	Reset Value
31–4	CCED	Local bus virtual address of current ED on the control ED list This field represents bits 31:4 of the local bus virtual 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 See Section 14.6.1, *Local Bus Addressing,* for the restrictions on local bus virtual 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	0
3–0	Reserved	Reserved	R	0x0

The HC head bulk register defines the local bus virtual address of the head ED on the bulk ED list.

Table 14–12. HC Head Bulk Register (HcBulkHeadED)

Bit	Name	Description	Type	Reset Value
31–4	BHED	Local bus virtual address of head ED on the bulk ED list This field represents bits 31:4 of the local bus virtual 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 See Section 14.6.1, *Local Bus Addressing,* for the restrictions on local bus virtual addresses.	R/W	0
3–0	Reserved	Reserved	R	0x0

The HC current bulk register defines the local bus virtual address of the next ED on the bulk ED list.

Table 14–13. HC Current Bulk Register (HcBulkCurrentED)

Bit	Name	Description	Type	Reset Value
31–4	BCED	Local bus virtual address of current ED on the bulk ED list This field represents bits 31:4 of the local bus virtual 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 See Section 14.6.1, *Local Bus Addressing,* for the restrictions on local bus virtual 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. This register is automatically updated by the USB host controller.	R/W	0
3–0	Reserved	Reserved	R	0x0

The HC head done register defines the local bus virtual address of the current head of the done TD queue.

Table 14–14. HC Head Done Register (HcDoneHead)

Bit	Name	Description	Type	Reset Value
31–4	DH	Local bus virtual address of the last TD that was added to the done queue. This field represents bits 31:4 of the local bus virtual 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. See See Section 14.6.1, *Local Bus Addressing,* for the restrictions on local bus virtual addresses. A value of 0x00000000 indicates that there are no TDs on the done queue. This register is automatically updated by the USB host controller.	R	0x0000000
3–0	Reserved	Reserved	R	0x0

The HC frame interval register defines the number of 12-MHz clock pulses in each USB frame.

Table 14–15. HC Frame Interval Register (HcFmInterval)

Bit	Name	Description	Type	Reset Value
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	0
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 may 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 14–16. HC Frame Remaining Register (HcFmRemaining)

Bit	Name	Description	Type	Reset Value
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	0

The HC frame number register reports the current USB frame number.

Table 14–17. HC Frame Number Register (HcFmNumber)

Bit	Name	Description	Type	Reset Value
31–16	Reserved	Reserved		
15–0	FN	Frame number 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.	R	0

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 14–18. HC Periodic Start Register (HcPeriodicStart)

Bit	Name	Description	Type	Reset Value
31–14	Reserved	Reserved		
13–0	PS	Periodic start The host controller driver must program this value to be about 10% less than the frame interval field value so that control and bulk EDs have priority for the first 10% of the frame; then periodic EDs have priority for the remaining 90% of the frame.	R/W	0

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 14–19. HC Low-Speed Threshold Register (HcLSThreshold)

Bit	Name	Description	Type	Reset Value
31–14	Reserved	Reserved		
13–0	LST	Low-speed threshold This field defines the number of full-speed bit times in the frame after which the USB host controller may not 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.	R/W	0x0628

The HC root hub A register defines several aspects of the USB host controller root hub functionality.

Table 14–20. HC Root Hub A Register (HcRhDescriptorA)

Bit	Name	Value	Description	Type	Reset Value
31–24	POTPG		Power-on to power-good time	R/W	0xA
			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 OMAP5910 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.		
23–13	Reserved		Reserved		
12	NOCP		No overcurrent protection	R/W	1
			When 1, this bit indicates that the USB host controller does not implement overcurrent protection inputs. OMAP5910 does not provide signals to allow connection of external overcurrent indication signals to the USB host controller, so this bit defaults to 1.		
11	OCPM		Overcurrent protection mode	R/W	0
			OMAP5910 does not provide overcurrent protection input signals, so this bit has no effect.		

Table 14–20. HC Root Hub A Register (HcRhDescriptorA) (Continued)

Bit	Name	Value	Description	Type	Reset Value
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 OMAP5910 does not provide connections from the USB host controller to control external VBUS switching, this bit defaults to 1.		
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.		
			Because OMAP5910 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	R	0x03
			This register defaults to 3 to indicate three downstream ports.		
			The USB signal multiplexing mode and OMAP5910 top-level pin multiplexing features may place the OMAP5910 device in a mode where 0, 1, 2, or 3 of the USB host controller downstream ports are actually usable. This register reports three regardless of USB signal multiplexing mode and OMAP5910 top-level pin multiplexing mode.		
			See Section 14.5, *USB Pin Multiplexing*, for information on USB signal multiplexing.		

The HC root hub B register defines several aspects of the USB host controller root hub functionality.

Table 14–21. HC Root Hub B Register (HcRhDescriptorB)

Bit	Name	Description	Type	Reset Value
31–16	PPCM	Port power control mask	R/W	0x0000
		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.		
		OMAP5910 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.		
15–0	DR	Device removable	R/W	0x0000
		Each bit defines whether a corresponding downstream port has a removable or nonremovable device. A cleared bit indicates the corresponding port may have a removable device attached. A set bit indicates that the corresponding port has a nonremovable 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.		

The HC root hub status register reports the USB host controller root hub status.

Table 14–22. HC Root Hub Status Register (HcRhStatus)

Bit	Name	Description	Type	Reset Value
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.	R/W	0
16	LPSC	Local power status change This bit defaults to 0 since the root hub does not support the local power status feature. Write of 0 has no effect. Write of 1 sets the PortPowerStatus bits for all ports if PowerSwitchingMode is 0. A write of 1 sets PortPowerStatus bits for ports with their corresponding PortPowerControlMask bits cleared if PowerSwitchingMode is 1.	R/W	0
15	DRWE	Device remote wake-up enable When 1, this bit enables a ConnectStatusChange event to be treated as a resume event, which causes a transition from USBSuspend to USBResume state, and sets the ResumeDetected interrupt status bit. When 0, ConnectStatusChange events do not cause a transition from USBSuspend to USBResume state and the ResumeDetected 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 14–22. HC Root Hub Status Register (HcRhStatus) (Continued)

Bit	Name	Description	Type	Reset Value
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 OMAP5910 does not provide signals for external hardware to report overcurrent status to the USB host controller, this bit is always 0.	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 PortPowerControlMask bit is 0. Because OMAP5910 does not provide signals from the USB host controller to external VBUS switching circuitry, this bit has no effect.	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 OMAP5910 USB port associated with the OMAP5910 integrated USB transceiver according to the HMC_MODE value. See Section 14.5, *USB Pin Multiplexing*.

Table 14–23. HC Port 1 Status and Control Register (HcRhPortStatus1)

Bit	Name	Description	Type	Reset Value
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 OMAP5910 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.	R/W	0
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

Table 14–23. HC Port 1 Status and Control Register (HcRhPortStatus1) (Continued)

Bit	Name	Description	Type	Reset Value
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 CurrentConnectStatus is 0 when a SetPortReset, SetPortEnable, or SetPortSuspend 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 nonremovable 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 only valid when port 1 CurrentConnectStatus is 1. The host controller driver can write a 1 to this bit to clear the port 1 PortPowerStatus. A write of 0 to this bit has no effect. OMAP5910 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
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 OMAP5910 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 1 port power status bit. A write of 0 has no effect.	R/W	1
7–5	Reserved	Reserved		

Table 14–23. HC Port 1 Status and Control Register (HcRhPortStatus1) (Continued)

Bit	Name	Description	Type	Reset Value
4	PRS/SPR	Port 1 port reset status/set port reset When read as 1, indicates that port 1 is receiving the USB reset signaling. 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. OMAP5910 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
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 CurrentConnectStatus 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 CurrentConnectState is 0, a write of 1 instead sets ConnectStatusChange 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

Table 14–23. HC Port 1 Status and Control Register (HcRhPortStatus1) (Continued)

Bit	Name	Description	Type	Reset Value
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 CurrentConnectStatus 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 value, HcRhPortStatus2 can provide status and control for the OMAP5910 USB Port 1 pins:

- CLK32K_OUT/USB1.SPEED
- BOOT/USB1.SUSP
- RST_HOST_OUT/USB1_SE0
- MCBSP.CLK/USB1_TXEN
- MCSI1.DOUT/USB1.TXD
- MCSI1.SYNC/USB1.VP
- MCSI1.CLK/USB1.VM
- MCSI1.DIN/USB1.RCV

Table 14–24. HC Port 2 Status and Control Register (HcRhPortStatus2)

Bit	Name	Description	Type	Reset Value
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 OMAP5910 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.	R/W	0
18	PSSC	Port 2 suspend status change 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
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

Table 14–24. HC Port 2 Status and Control Register (HcRhPortStatus2) (Continued)

Bit	Name	Description	Type	Reset Value
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 CurrentConnectStatus is 0 when a SetPortReset, SetPortEnable, or SetPortSuspend 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 only set 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 only valid when port 2 CurrentConnectStatus is 1. The host controller driver can write a 1 to this bit to clear the port 2 PortPowerStatus. A write of 0 to this bit has no effect. The OMAP5910 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
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 OMAP5910 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 has 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.	R/W	1
7–5	Reserved	Reserved		

Table 14–24. HC Port 2 Status and Control Register (HcRhPortStatus2) (Continued)

Bit	Name	Description	Type	Reset Value
4	PRS/SPR	Port 2 port reset status/set port reset When read as 1, indicates that port 2 is receiving the USB reset signaling. 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 cause 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 OMAP5910 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.	R/W	0
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 CurrentConnectStatus 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 CurrentConnectState is 0, a write of 1 instead sets ConnectStatusChange 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

Table 14–24. HC Port 2 Status and Control Register (HcRhPortStatus2) (Continued)

Bit	Name	Description	Type	Reset Value
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 CurrentConnectStatus 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 value, HcRhPortStatus2 can provide status and control for the OMAP5910 USB port 2 pins:

- MCSI2.CLK/USB2_SUSP
- UART2.RTS/USB2_SE0
- MCSI2.DOUT/USB2.TXEN
- UART2.TX/USB2.TXD
- MCSI2.DIN/USB2.VP
- UART2.RX/USB2.VM
- UART2.CTS/USB2.RCV.

Table 14–25. HC Port 3 Status and Control Register (HcRhPortStatus3)

Bit	Name	Description	Type	Reset Value
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 OMAP5910 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.	R/W	0
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

Table 14–25. HC Port 3 Status and Control Register (HcRhPortStatus3) (Continued)

Bit	Name	Description	Type	Reset Value
16	CSC	Port 3 connect status change This bit indicates, when 1, that the port 3 current connect status has changed due to a connect or disconnect event. If CurrentConnectStatus is 0 when a SetPortReset, SetPortEnable, or SetPortSuspend 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 only set 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 CurrentConnectStatus is 1. The host controller driver may write a 1 to this bit to clear the port 3 PortPowerStatus. A write of 0 to this bit has no effect. OMAP5910 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
8	PPS/SPP	Port 3 port 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 OMAP5910 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 may track the current power state using the port power status bit and other power control bits, but those bits has no direct effect on external port power control. A write of 1 to this bit sets the port 3 port power status bit. A write of 0 has no effect.	R/W	1
7–5	Reserved	Reserved		

Table 14–25. HC Port 3 Status and Control Register (HcRhPortStatus3) (Continued)

Bit	Name	Description	Type	Reset Value
4	PRS/SPR	Port 3 port reset status/set port reset When read as 1, indicates that port 3 is receiving the USB reset signaling. When read as 0, USB reset is not being sent to port 3. A write of 1 to this bit sets the port 3 port reset status bit and cause 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 port overcurrent indicator/clear suspend status When read as 1, indicates that a port 3 port overcurrent condition has occurred. When 0, no port 3 port overcurrent condition has occurred. The OMAP5910 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 port suspend status is 1 causes resume signaling on port 3. A write of 1 when port 3 port suspend status is 0 has no effect. A write of 0 has no effect.	R/W	0
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 CurrentConnectStatus is 1, a write of 1 to this bit sets the port 3 port suspend status bit and places port 3 in USB suspend state. If CurrentConnectState is 0, a write of 1 instead sets ConnectStatusChange 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

Table 14–25. HC Port 3 Status and Control Register (HcRhPortStatus3) (Continued)

Bit	Name	Description	Type	Reset Value
1	PES/SPE	Port 3 port 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 CurrentConnectStatus is 1 sets the port 3 port 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 nonremovable device on port 3. A write of 1 to this bit clears the port 3 port enable bit. A write of 0 to this bit has no effect.	R/W	0

The host UE address register reports the local bus virtual address of the last local bus access which 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 14–26. Host UE Address Register (HostUEAddr)

Bit	Name	Description	Type	Reset Value
31–0	UE_ADDR	Unrecoverable error address This register captures the local bus virtual address of any local 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 a local bus access that resulted in an unrecoverable error. This register is not affected by local bus timeouts occurring when the MPU, DSP, or DMA controller attempts to access a local bus slave peripheral.	R	0x0000 0000

The host UE status register reports the local 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 14–27. Host UE Status Register (HostUEStatus)

Bit	Name	Description	Type	Reset Value
31–1	Reserved	Reserved	R	xxxxxxxx
0	UEAccess	Access type when unrecoverable error occurred When an unrecoverable error occurs due to timeout of a local bus write, this bit is set. When an unrecoverable error occurs due to timeout of a local 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 a local bus access that resulted in an unrecoverable error. This register is not affected by local bus time-outs occurring when the MPU, DSP, or DMA controller attempts to access a local bus slave peripheral.		0

The host time-out control register controls the USB host controller local bus time-out mechanism. This register is not defined by the OHCI specification.

Table 14–28. Host Time-out Control Register (HostTimeoutCtrl)

Bit	Name	Description	Type	Reset Value
31–1	Reserved	Reserved		
0	TO_DIS	Local bus time-out disable When 1, the USB host controller local 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 local bus clocks for completion of a local bus access to system memory. If the local bus cycle does not complete in that time, the USB host controller signals an unrecoverable error. This bit has no effect on MPU, DSP, or DMA controller accesses to local bus slave peripherals.	R/W	0

The host revision register returns the revision number for the OMAP5910 USB host controller. This register is not defined by the OHCI specification.

Table 14–29. Host Revision Register (HostRevision)

Bit	Name	Description	Type	Reset Value
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 OMAP5910 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 OMAP5910 USB host controller version implements a minor revision number of 0.	R	xx

14.3.1 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.

14.3.2 Endianism and USB Host Controller Registers

The OMAP5910 USB host controller assumes that all MPU accesses to its registers are 32-bit accesses. This restriction means that the host controller driver software may operate in either big-endian or little-endian without having to perform endian conversion on USB host controller register accesses. Software that uses 16-bit or 8-bit accesses to USB host controller registers does not work correctly, regardless of processor endianism mode.

The processor endianism does affect how the software must access the USB data structures and USB data buffers in system memory. See Section 14.6.5, *Endianism and USB Host Controller Access to System Memory* for details on endianism, data buffers, and data structures.

14.3.3 USB Host Controller Registers, USB Reset, and USB Clocking

When the USB host controller is not clocked (because the MOD_CONF_CTRL_0 register CONF_MOD_USB_HOST_HHC_UHOST_EN_R bit is 0), 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.

The USB host controller completes its reset within about 72 clock cycles after CONF_MOD_USB_HOST_HHC_UHOST_EN_R is active and the ULPD begins providing clock to the USB host controller. 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.

14.4 USB Host Controller Interrupt Sources

14.4.1 OHCI Interrupts

The OMAP5910 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.

14.4.1.1 OHCI Scheduling Overrun Interrupt

The OHCI scheduling overrun interrupt is supported as described in the *OHCI Specification for USB*.

14.4.1.2 OHCI HcDoneHead Writeback Interrupt

The OHCI HcDoneHead writeback interrupt is supported as described in the *OHCI Specification for USB*.

14.4.1.3 OHCI Start Of Frame Interrupt

The OHCI start of frame interrupt is supported as described in the *OHCI Specification* for USB.

14.4.1.4 OHCI Resume Detect Interrupt

The OHCI resume detect interrupt is supported as described in the *OHCI Specification for USB*.

14.4.1.5 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 a local bus read or local bus write within 4096 local bus clocks when the USB host local bus timeout feature is enabled (see Table 14–28, *Host Timeout Control Register (HostTimeoutCtrl*). When a local bus timeout 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.

14.4.1.6 OHCI Frame Number Overflow

The OHCI frame number overflow interrupt is supported as described in the *OHCI Specification for USB*.

14.4.1.7 OHCI Root Hub Status Change

The OHCI root hub status change interrupt is supported as described in the *OHCI Specification for USB*. The OMAP5910 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 due to a port overcurrent event.

14.4.1.8 OHCI Ownership Change Interrupt

The optional OHCI ownership change interrupt is not supported.

14.4.2 Local Bus MMU Interrupts

Interrupts from the local bus MMU to the MPU level 1 interrupt handler IRQ_17 input occur if the USB host controller attempts an access which the local bus MMU cannot perform. This interrupt can be an important tool when debugging a USB host controller driver and can be used to help identify operational faults in a final product.

This interrupt does not occur if the USB host controller attempts to access a local bus virtual address that is in the range 0x00000000 to 0x2FFFFFFF or 0x40000000 to 0xFFFFFFFF. For more detail on the local bus MMU, see Section 14.6, *USB Host Controller Access to System Memory*.

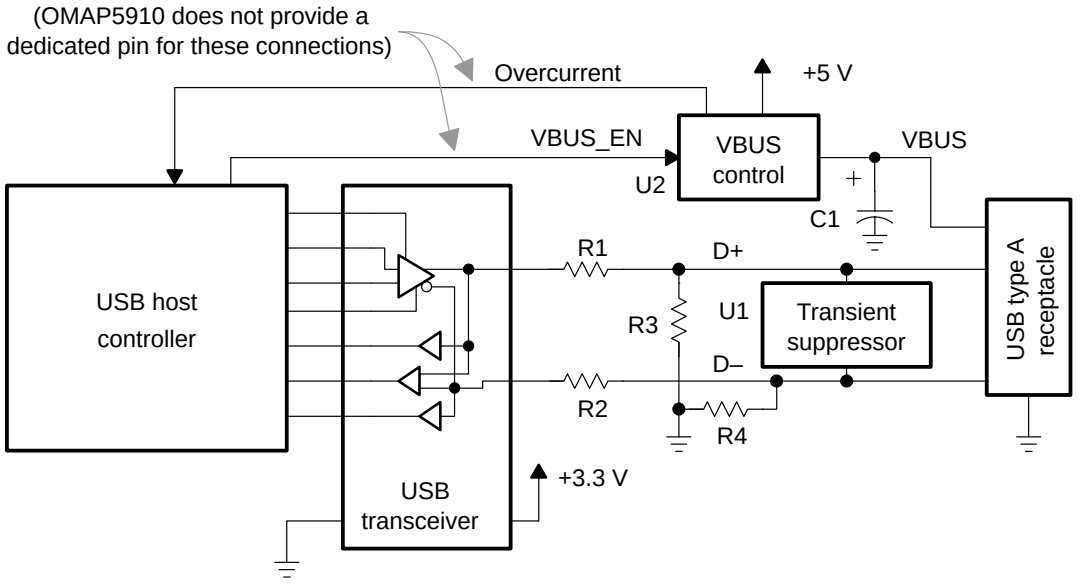
14.5 USB Pin Multiplexing

OMAP5910 USB signal multiplexing provides five main types of signaling: USB host and USB function with USB transceivers, USB host and USB function without USB transceivers, and UART1. This section describes first general external connectivity for these types of multiplexing and then the specific OMAP5910 USB multiplexing modes along with specific OMAP5910 connectivity for each mode.

14.5.1 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 USB-type A receptacle, power on the VBUS signal (may be switched or unswitched power), transient suppression, pulldown resistors, and USB-compatible downstream port transceiver. These elements are shown in Figure 14–3.

Figure 14–3. Typical USB Host Connections



R1, R2	Value depends on transceiver
R3, R4	15K Ohm +/– 5%
C1	Low ESR cap, minimum 120 uF
U1	Transient suppressor, such as SN65220, SN65240, or SN75240
U2	Power switch, such as TPS2014 or TPS2015

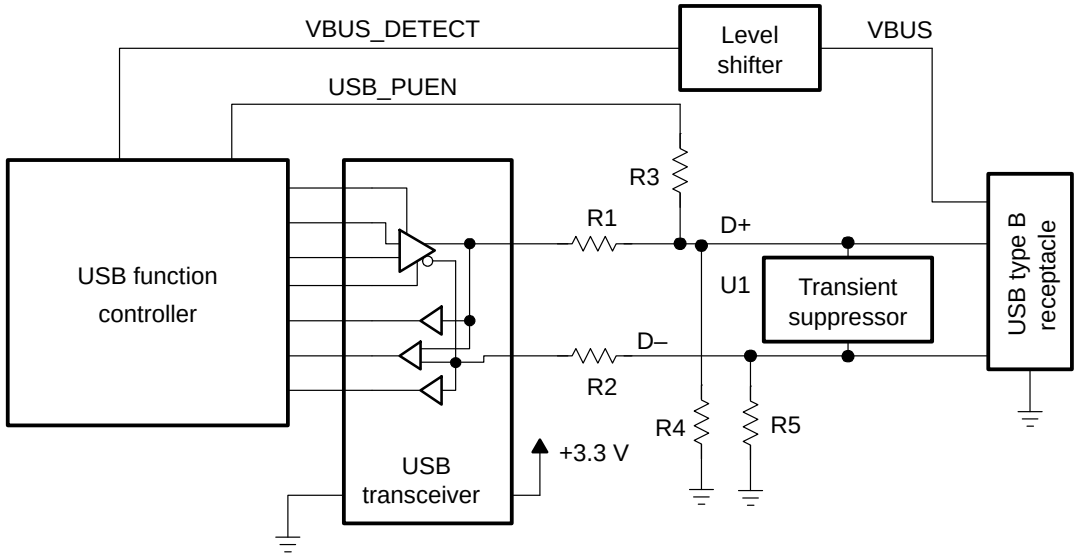
There are at least three different types of signaling used by commercially-available USB transceivers to interface to a USB function controller. OMAP5910 is designed for use with USB transceivers that use TXSE0 and TXD signaling. OMAP5910 is not designed to interface directly with USB transceivers that use bidirectional signals between the transceiver and the USB host controller or with USB transceivers that use TXD+ and TXD– signaling.

Because OMAP5910 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, OMAP5910 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.

14.5.2 USB Function Controller Connectivity With USB Transceivers

To provide a robust USB solution, a system that provides a USB function controller must implement certain features. These features include a USB-type 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 14–4.

Figure 14–4. Typical USB Function Connections



R1, R2	Value depends on transceiver
R3	1.5K Ohm +/– 5%
R4, R5	Weak pulldown (optional, see text)
C1	Low ESR cap, minimum 120 uF
U1	Transient suppressor, such as SN65220, SN65240, or SN75240

There are at least three different types of signaling used by commercially-available USB transceivers to interface to a USB function controller. OMAP5910 is designed mainly for use with USB transceivers that use TXSE0 and TXD signaling. OMAP5910 is not designed to interface directly with USB transceivers which use bidirectional signals between the transceiver and the USB host controller, or with USB transceivers that use TXD+ and TXD– signaling.

Figure 14–4 shows optional weak pulldown resistors R4 and R5. These can be used to hold the D+ and D- signals at voltages below the USB transceiver V_{IL} level when there is no USB host connected to the USB Type B connector. By keeping D+ and D- voltages below V_{IL}, the USB transceiver I_{DDQ} can be reduced. Choice of value for R4 and R5 must be made carefully so that the circuit meets the requirements of the *USB Specification*.

The OMAP5910 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.

14.5.3 On-Board Transceiverless Connection Using OMAP5910 Transceiverless Link Logic

The transceiverless link logic feature of the OMAP5910 USB signal multiplexing enables connection of the OMAP5910 device to an external, on-board USB host controller or external on–board USB function controller, without the use of USB transceivers or associated circuitry. When the transceiverless lin logic is used, both of the USB transceivers, the series resistors, pullup and pulldown resistors, VBUS switching components, and USB connectors and cables that normally are used between a USB host controller and the downstream USB function controller are removed.

The transceiverless link logic signaling system is not suitable for use across a cable. It is intended only for use when the OMAP5910 device is used with an external USB integrated circuit which is on the same board.

When using the transceiverless link logic, six of the external USB integrated circuit pins that normally connect to a USB transceiver connect instead directly to OMAP5910 device pins. Signaling on these pins use CMOS levels.

Transceiverless link logic can be compared to a normal USB implementation as shown in Figure 14–5 and Figure 14–6. Figure 14–5 shows OMAP5910 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 OMAP5910 transceiverless link logic. Figure 14–6 shows OMAP5910 used as a USB function controller, with the top portion of the diagram showing a transceiver-based solution and the bottom portion showing a transceiverless solution using the OMAP5910 transceiverless link logic.

The transceiverless link logic function in the OMAP5910 device interprets the transmit control signals from the external USB integrated circuit and similar signals from the OMAP5910 USB host controller or OMAP5910 USB function 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 OMAP5910 USB host controller or OMAP5910 USB function controller.

Figure 14–5. OMAP5910 USB Host Controller Connection—With and Without the OMAP5910 Transceiverless Link Logic

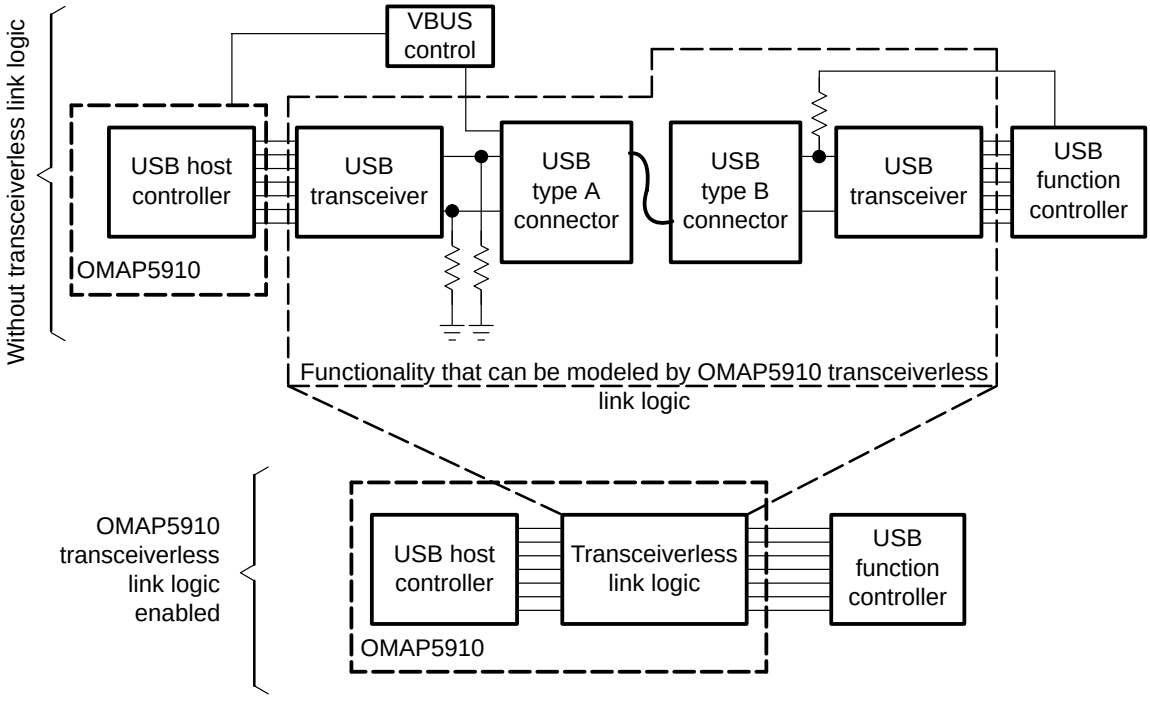
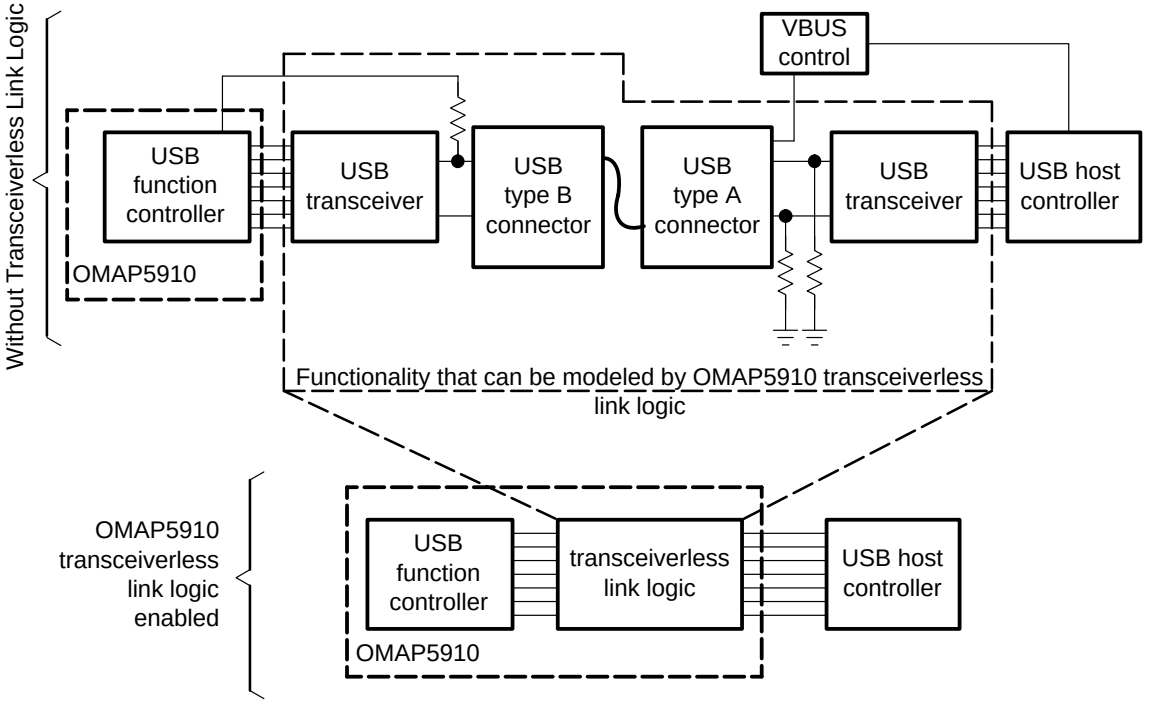


Figure 14–6. OMAP5910 USB Function Connection—With and Without the OMAP5910 Transceiverless Link Logic



14.5.4 USB Signal Multiplexing Mode Diagrams

The OMAP5910 USB signal multiplexing mechanisms provide a wide variety of options for bringing USB functionality to the OMAP5910 pins. These options are listed in Table 14–30 and are shown in Figure 14–7 through Figure 14–31. Each of these figures shows the external connectivity used to implement a typical system using one of the USB signal multiplexing modes. Each diagram assumes that OMAP5910 top-level signal multiplexing has been initialized to select the USB signal multiplexer as the source/destination for those signals shown as actively controlled by the USB signal multiplexing box. Top level pin muxing is configured via the OMAP5910 configuration registers described in section 6.8. In the figures, the items shown in gray do not receive or control OMAP5910 USB related pin signals for the HMC_MODE shown. For each configuration, appropriate external hardware is also required. This may include pullup or pulldown resistors, series resistors, USB transceiver devices, ESD protection devices, power switching circuitry, and USB connectors.

Table 14–30. USB Signal Multiplexing Modes

CONF_MOD_USB_HOST_HMC_MODE_R		USB Functions Available to OMAP5910		
		Integrated USB Transceiver Pins (UXB0.x Pins)	**Pin Group 1**	**Pin Group 2**
0		USB function		
1		USB host port 1	USB host port 2	USB host port 3
2		USB host port 1	UART 1†	USB host port 3
3		USB host port 1	USB function	USB host port 3
4		USB function	USB host port 2	USB host port 3
5		USB host port 1		USB host port 3
6		USB function		USB host port 3
7		USB host port 1	‡	‡
8	Reserved. Do not use.	Reserved	Reserved	Reserved
9		USB host port 1	USB host port 2	USB host port 3 with TLL
10		USB host port 1	UART 1†	USB host port 3 with TLL
11		USB host port 1	USB function	USB host port 3 with TLL
12		USB function	USB host port 2	USB host port 3 with TLL
13		USB host port 1		USB host port 3 with TLL
14		USB function		USB host port 3 with TLL
15		USB host port 1	USB host port 2	USB host port 3 with TLL
16		USB host port 1		

† CONF_MOD_USB_HOST_HMC_MODE_R values that select UART 1 bring UART1 CTS, RX, and TX signals to pins that can, in other CONF_MOD_USB_HOST_HMC_MODE_R values, be used for USB.

‡ CONF_MOD_USB_HOST_HMC_MODE_R 7 provides an internal signal path from six of the USB-related OMAP5910 input pins to six of the USB-related OMAP5910 output pins.

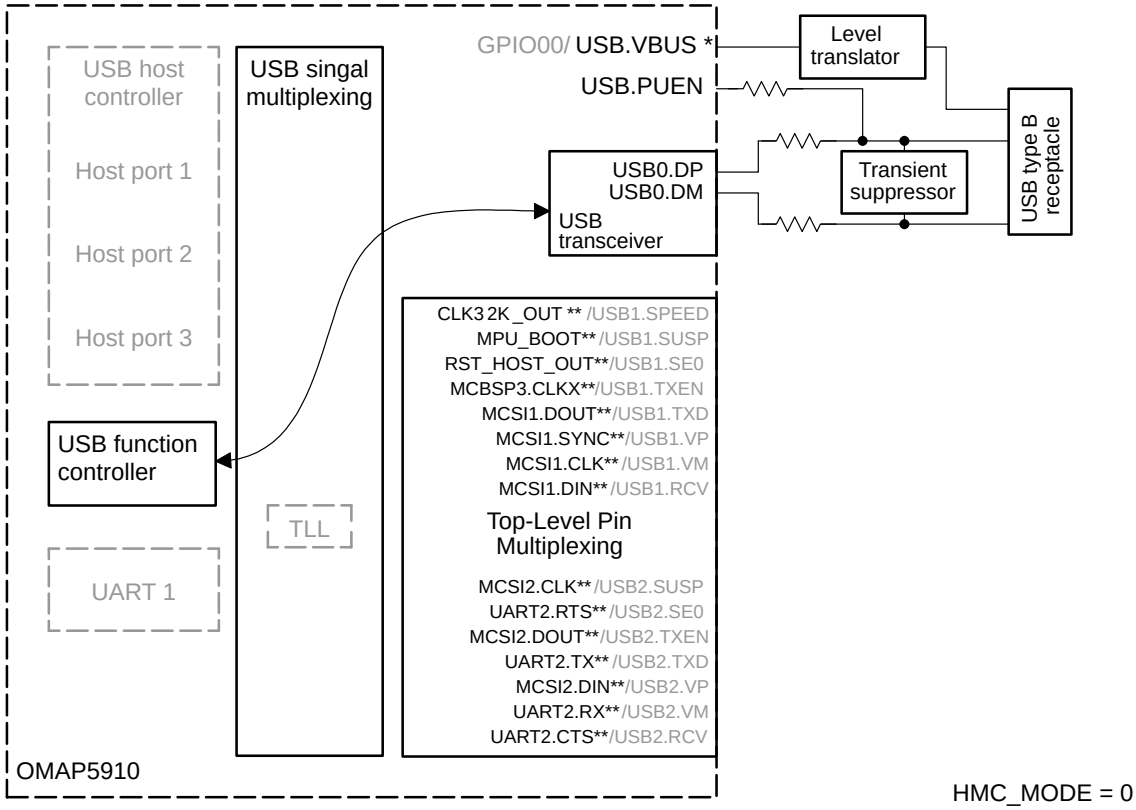
§ CONF_MOD_USB_HOST_HMC_MODE_R 21 provides an internal signal path from USB host controller Port 3 to the OMAP5910 USB function via the transceiverless link logic.

Table 14–30. USB Signal Multiplexing Modes (Continued)

CONF_MOD_USB_HOST_HMC_MODE_R	USB Functions Available to OMAP5910		
	Integrated USB Transceiver Pins (UXB0.x Pins)	**Pin Group 1**	**Pin Group 2**
17	USB host port 1	USB host port 2	
18	USB host port 1	UART 1†	
19	USB host port 1	USB function	
20	USB function	USB host port 2	
21§	USB host port 1		
22			
23	USB host port 1	USB host port 2	USB host port 3 with TLL (TXD– signaling)
24	USB host port 1	UART 1†	USB host port 3 with TLL (TXD– signaling)
25	USB host port 1	USB function	USB host port 3 with TLL (TXD– signaling)
26–31			

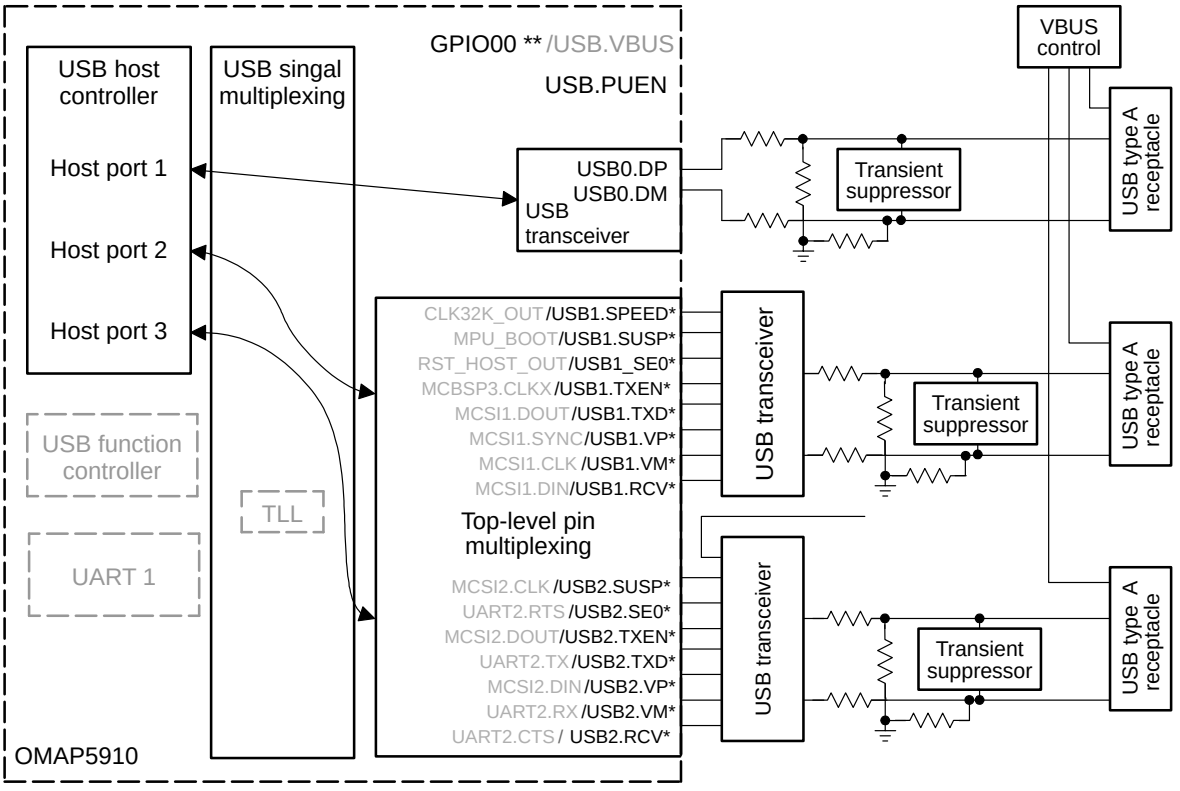
† CONF_MOD_USB_HOST_HMC_MODE_R values that select UART 1 bring UART1 CTS, RX, and TX signals to pins that can, in other CONF_MOD_USB_HOST_HMC_MODE_R values, be used for USB.
‡ CONF_MOD_USB_HOST_HMC_MODE_R 7 provides an internal signal path from six of the USB-related OMAP5910 input pins to six of the USB-related OMAP5910 output pins.
§ CONF_MOD_USB_HOST_HMC_MODE_R 21 provides an internal signal path from USB host controller Port 3 to the OMAP5910 USB function via the transceiverless link logic.

Figure 14–7. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 0



* Assumes pins have configured for USB functionality.
** Unused USB functional pins can be configured for non-USB functionality.

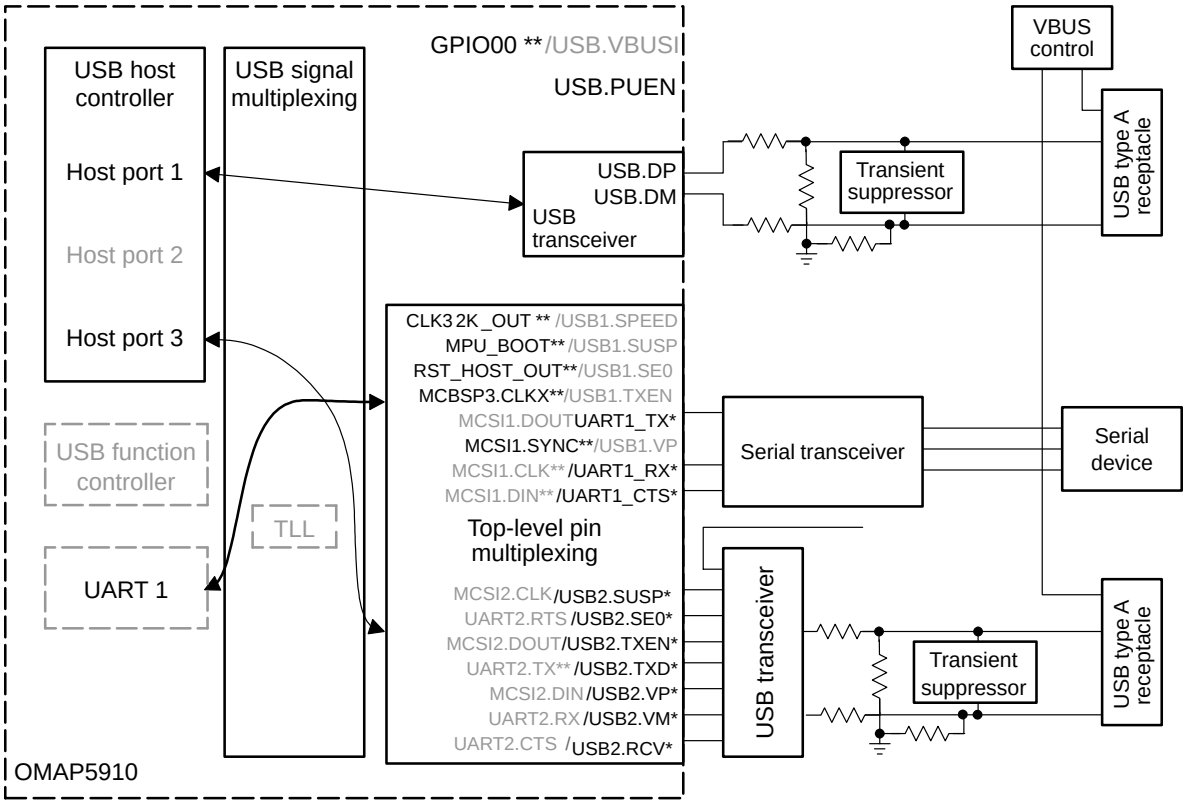
Figure 14–8. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 1



* Assumes pins have configured for USB functionality.
** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 1

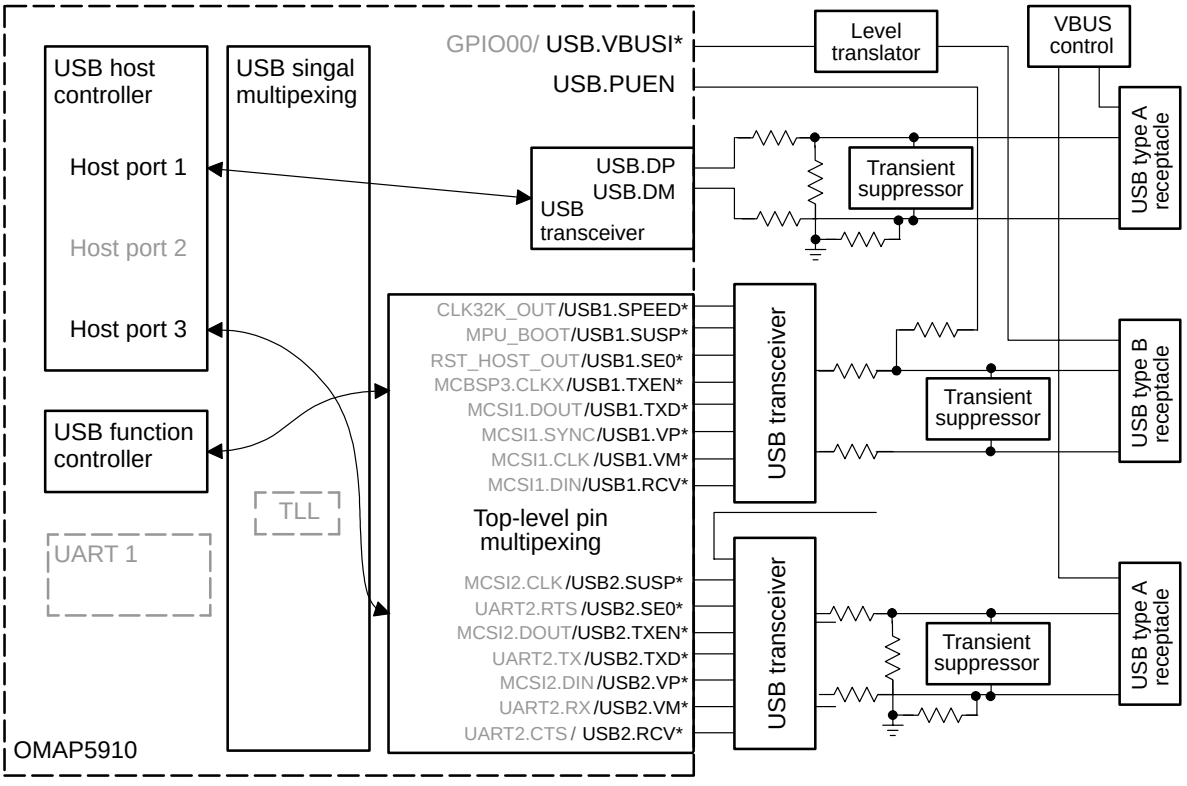
Figure 14–9. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 2



* Assumes pins have configured for USB functionality.
** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 2

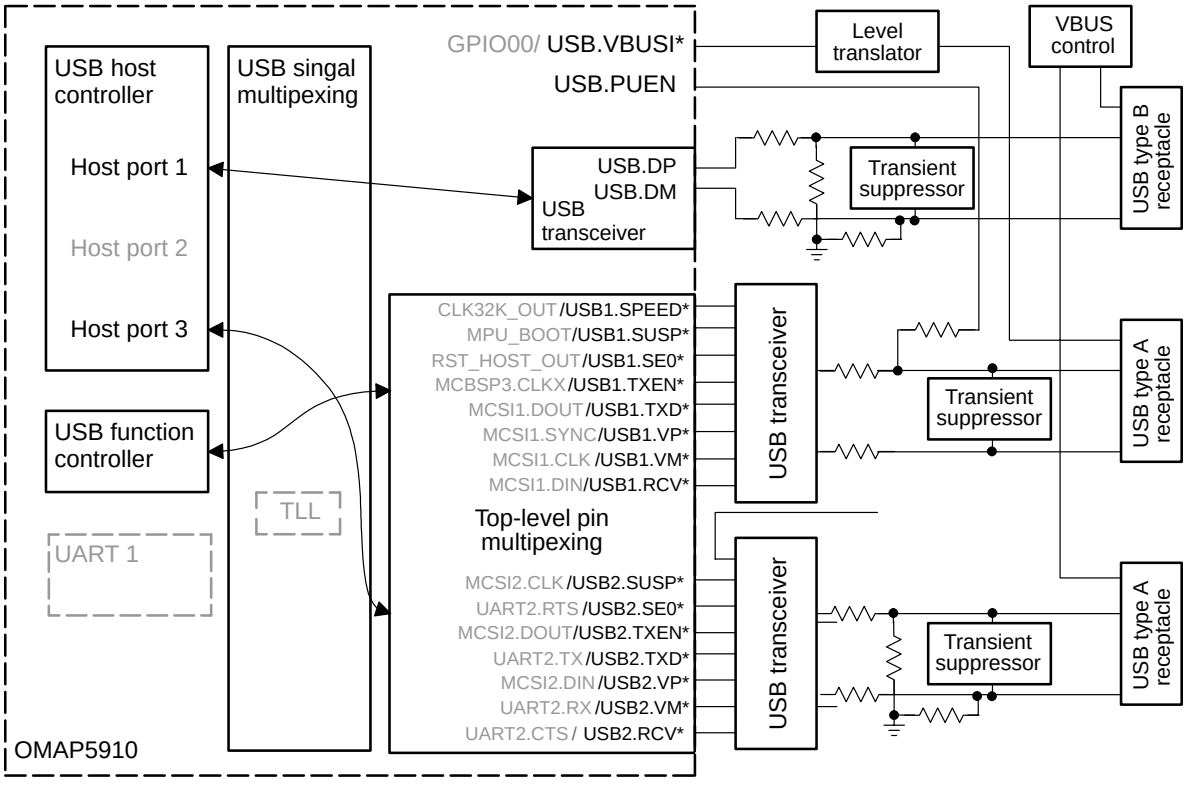
Figure 14–10. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 3



* Assumes pins have configured for USB functionality.

HMC_MODE = 3

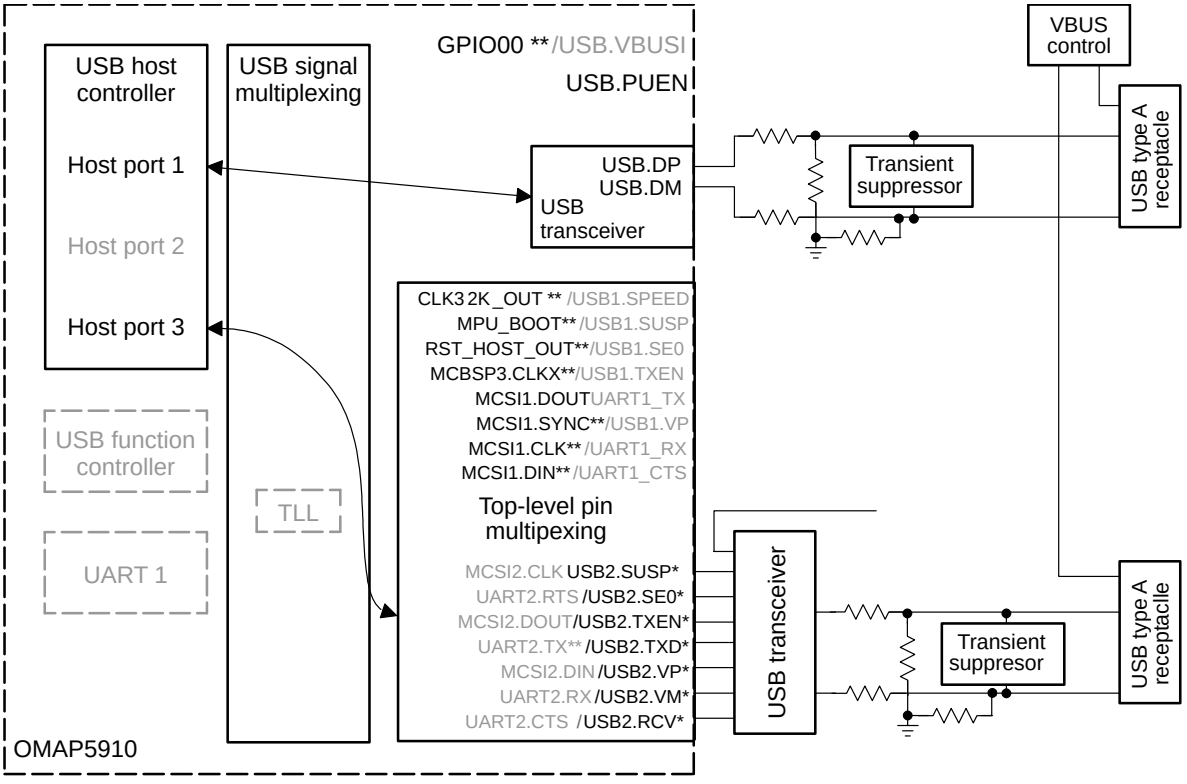
Figure 14–11. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 4



* Assumes pins have configured for USB functionality.

HMC_MODE = 3

Figure 14–12. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 5

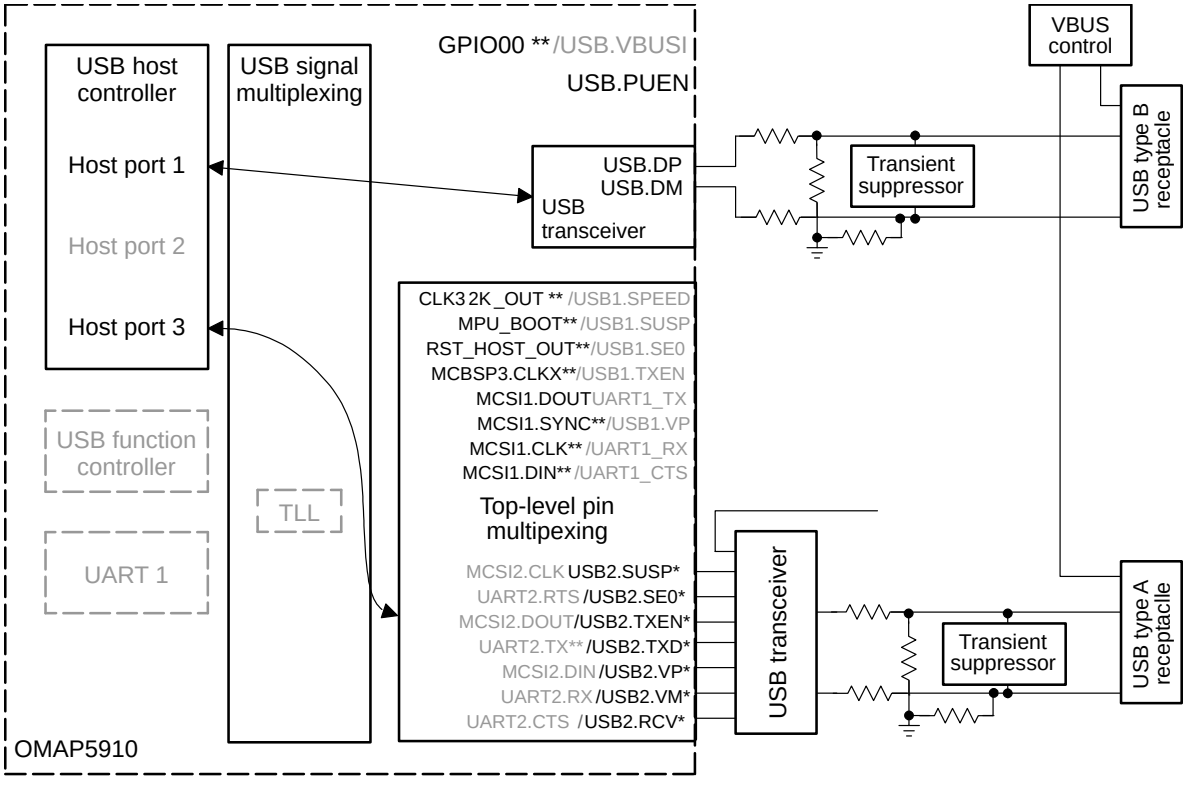


* Assumes pins have configured for USB functionality.

** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 5

Figure 14–13. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 6

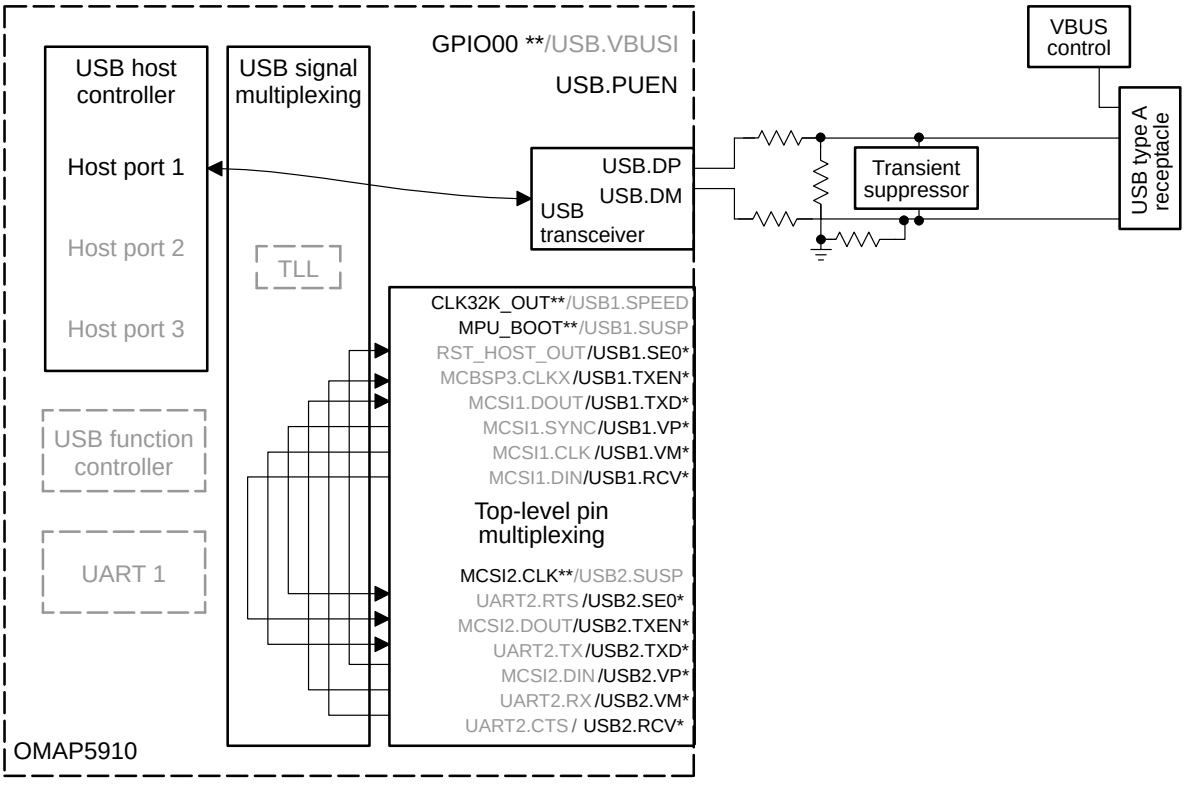


* Assumes pins have configured for USB functionality.

** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 5

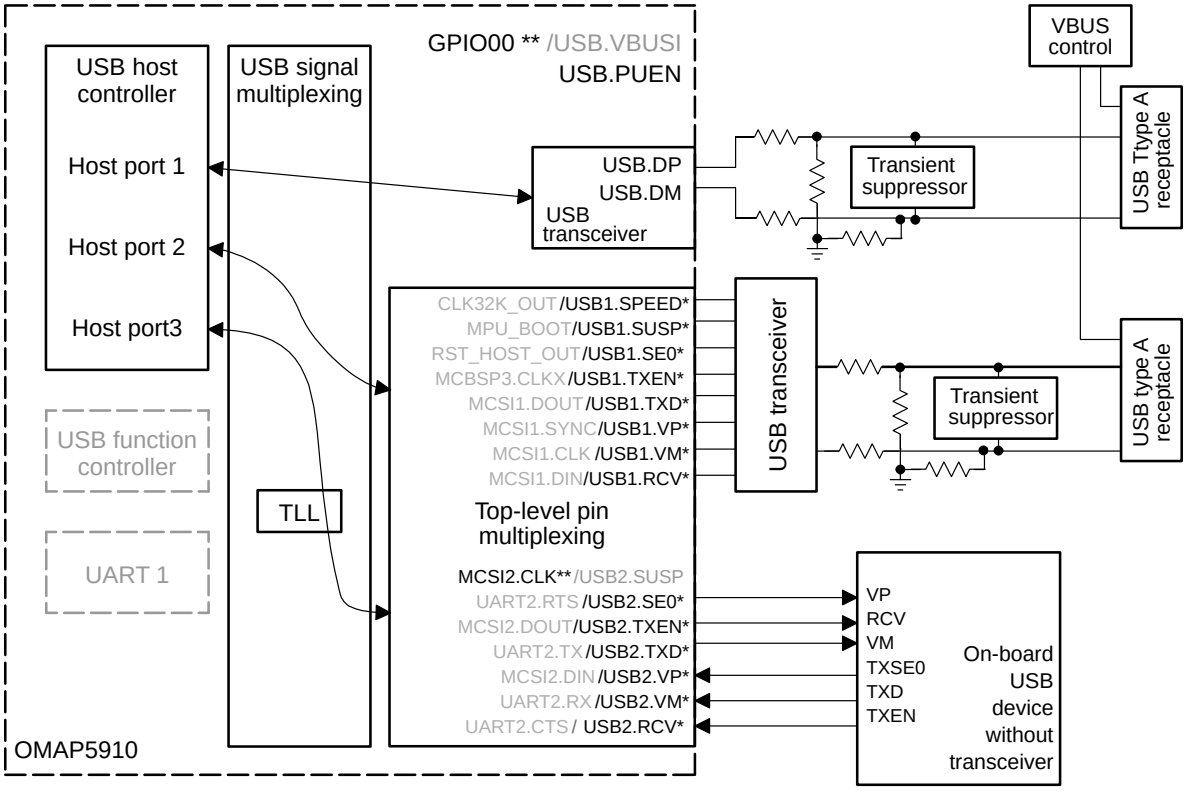
Figure 14–14. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 7



* Assumes pins have configured for USB functionality.
** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 7

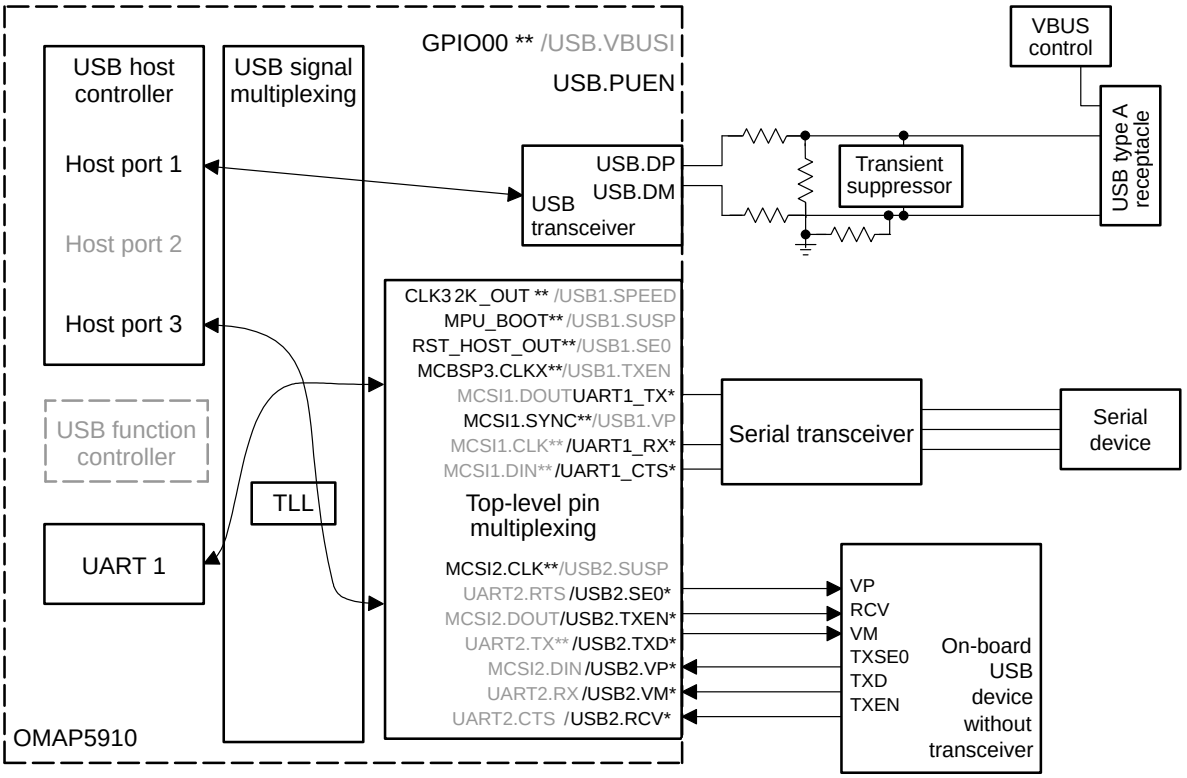
Figure 14–15. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 9



* Assumes pins have configured for USB functionality.
** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 9

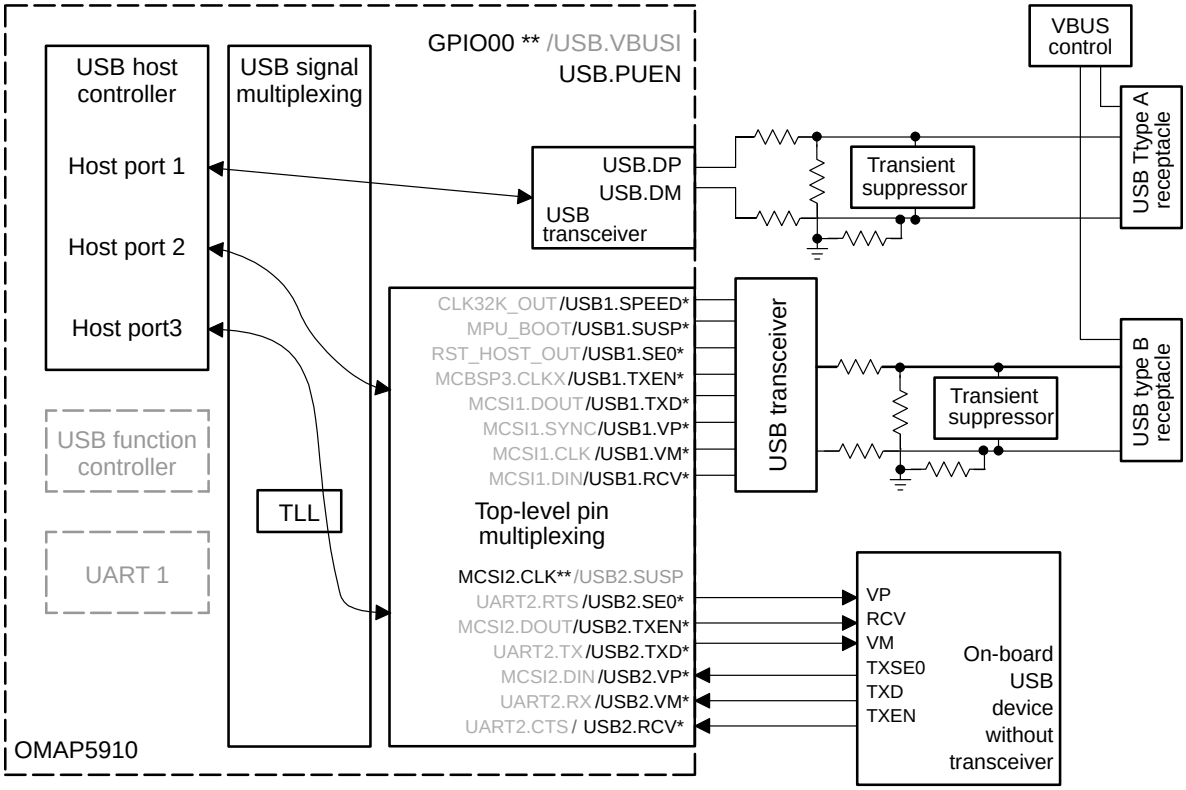
Figure 14–16. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 10



* Assumes pins have configured for USB functionality.
** used USB functional pins can be configured for non-USB functionality.

HMC_MODE = 10

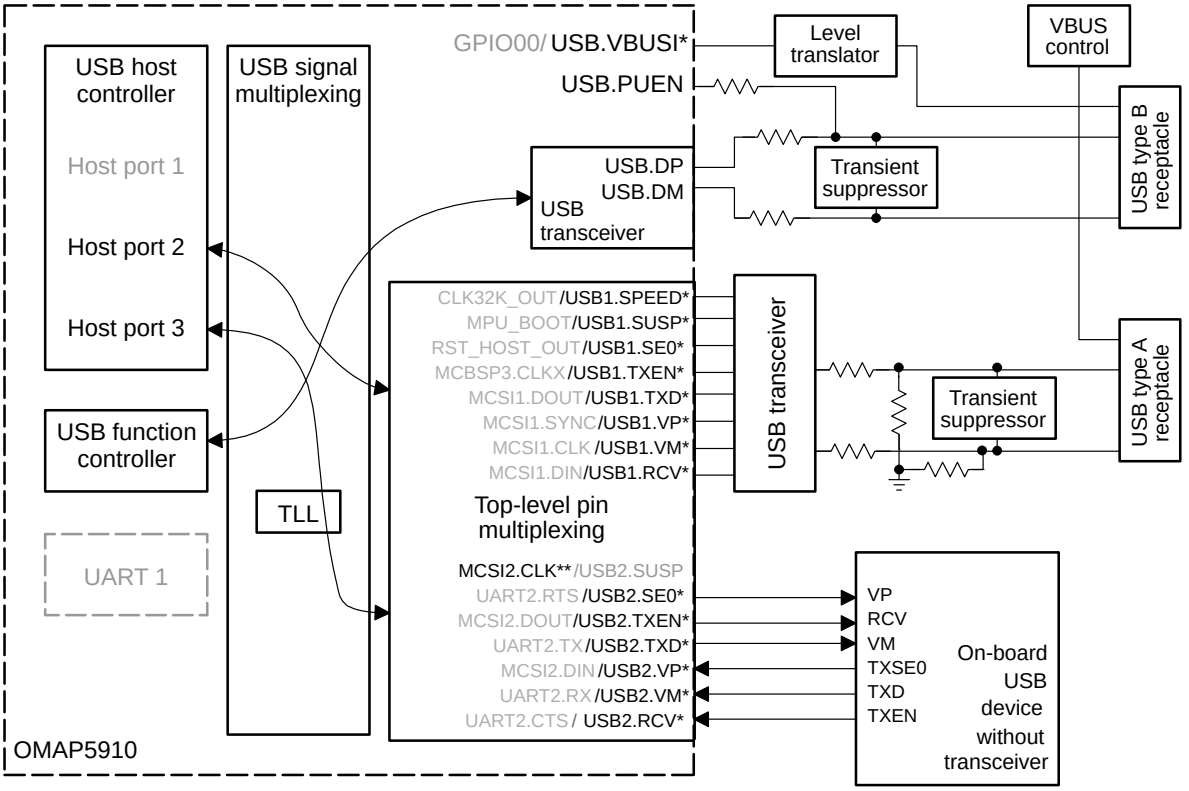
Figure 14–17. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 11



* Assumes pins have configured for USB functionality.
** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 9

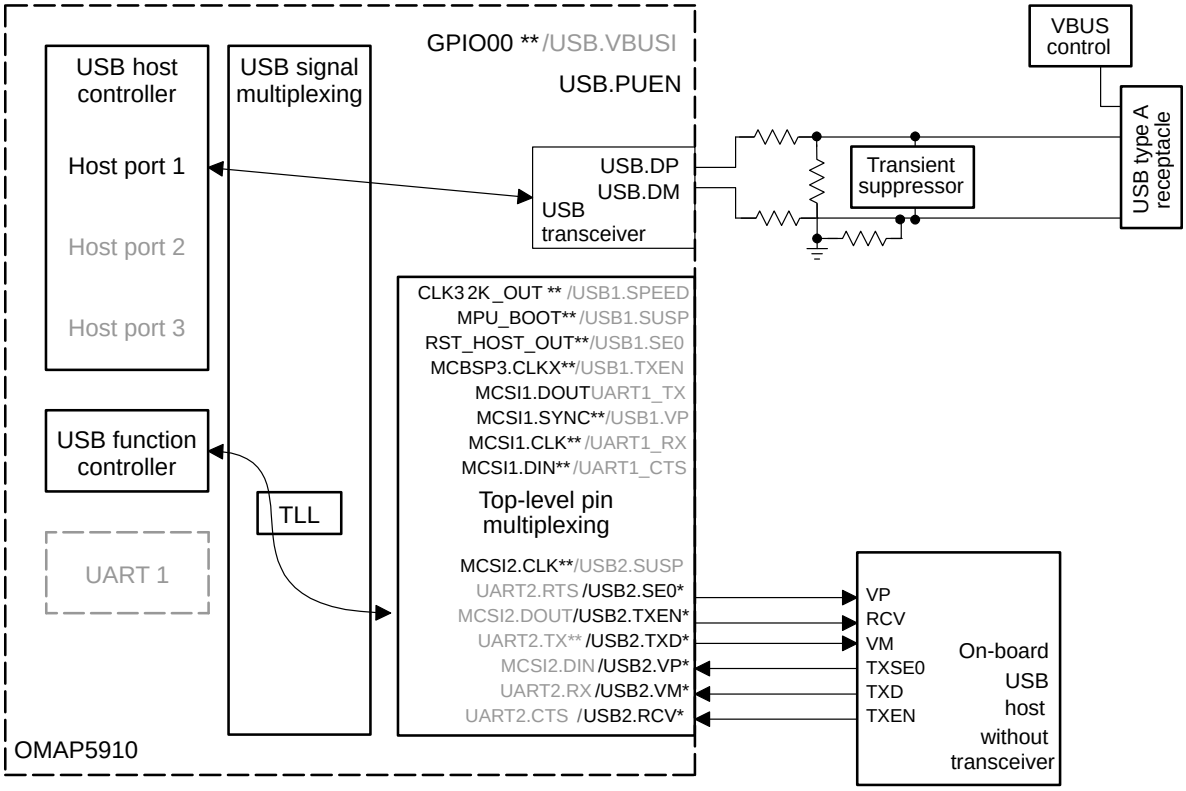
Figure 14–18. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 12



* Assumes pins have configured for USB functionality.
** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 12

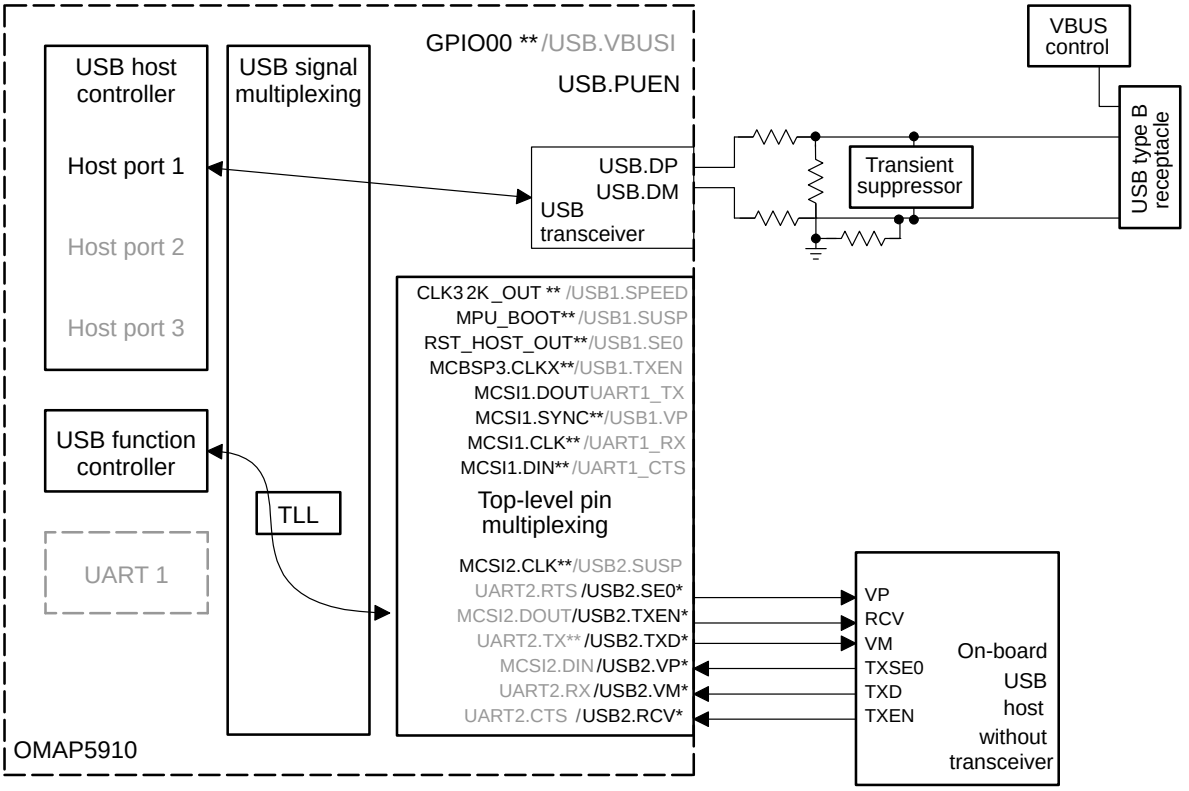
Figure 14–19. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 13



* Assumes pins have configured for USB functionality.
** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 13

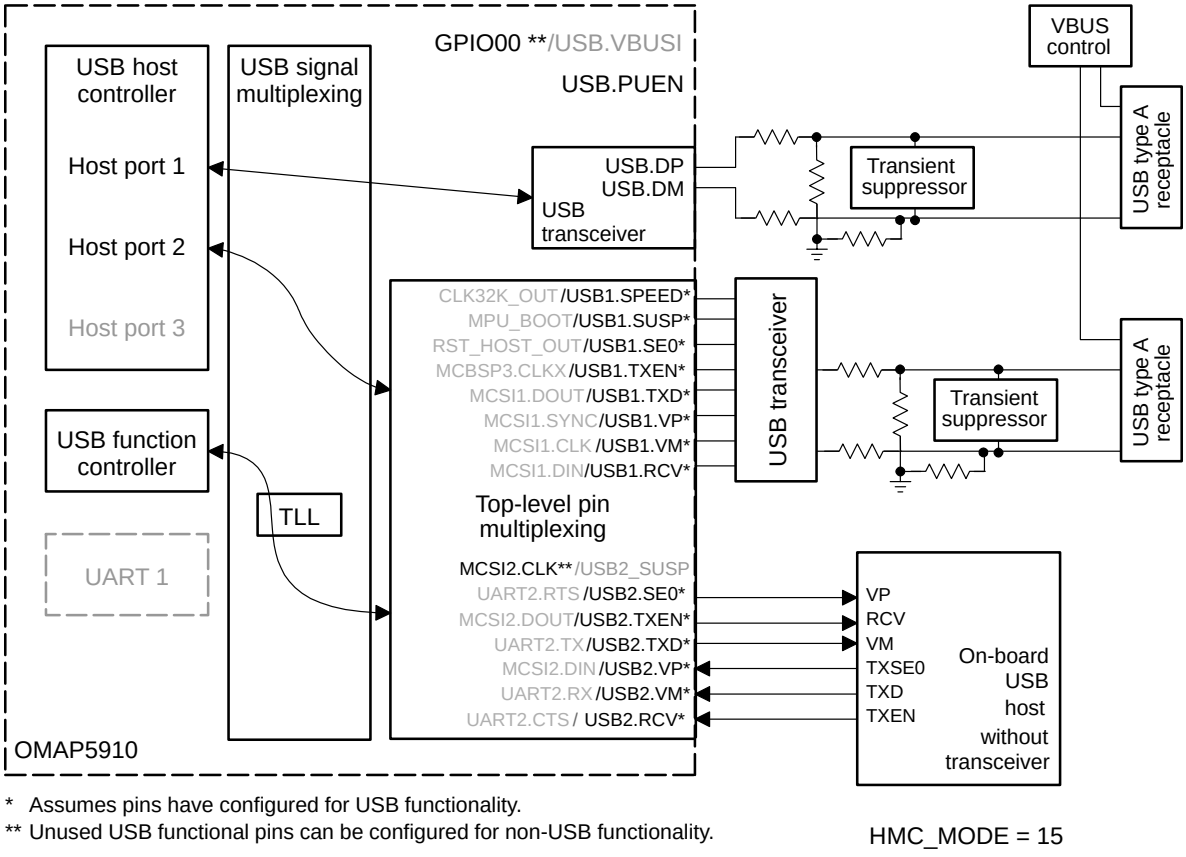
Figure 14–20. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 14



* Assumes pins have configured for USB functionality.
** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 14

Figure 14–21. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 15

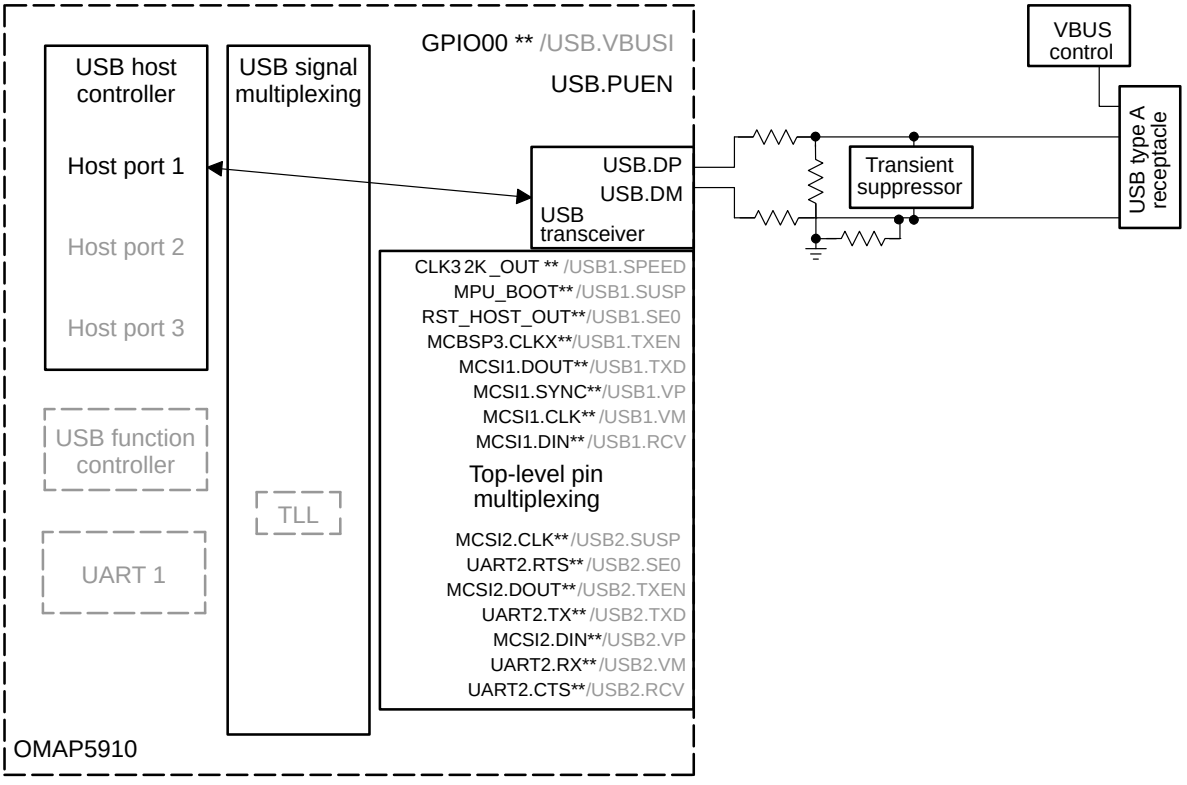


* Assumes pins have configured for USB functionality.

** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 15

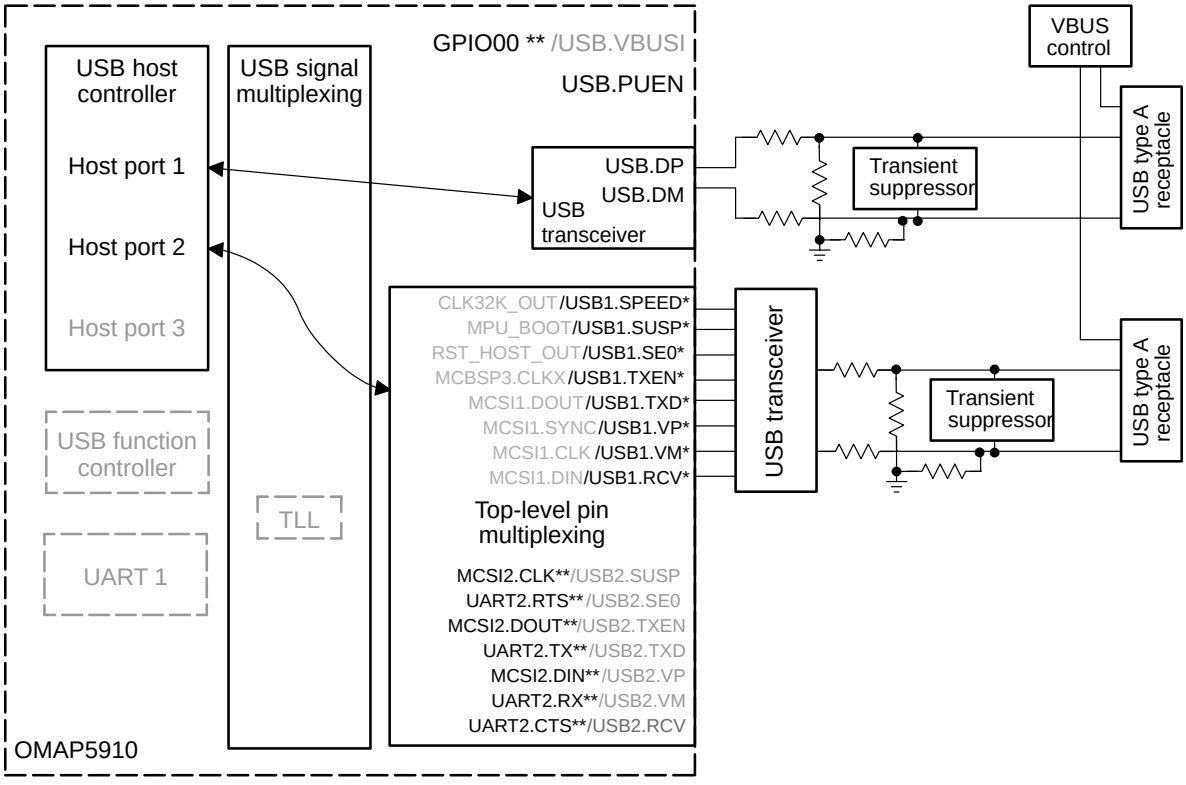
Figure 14–22. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 16



* Assumes pins have configured for USB functionality.
** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 16

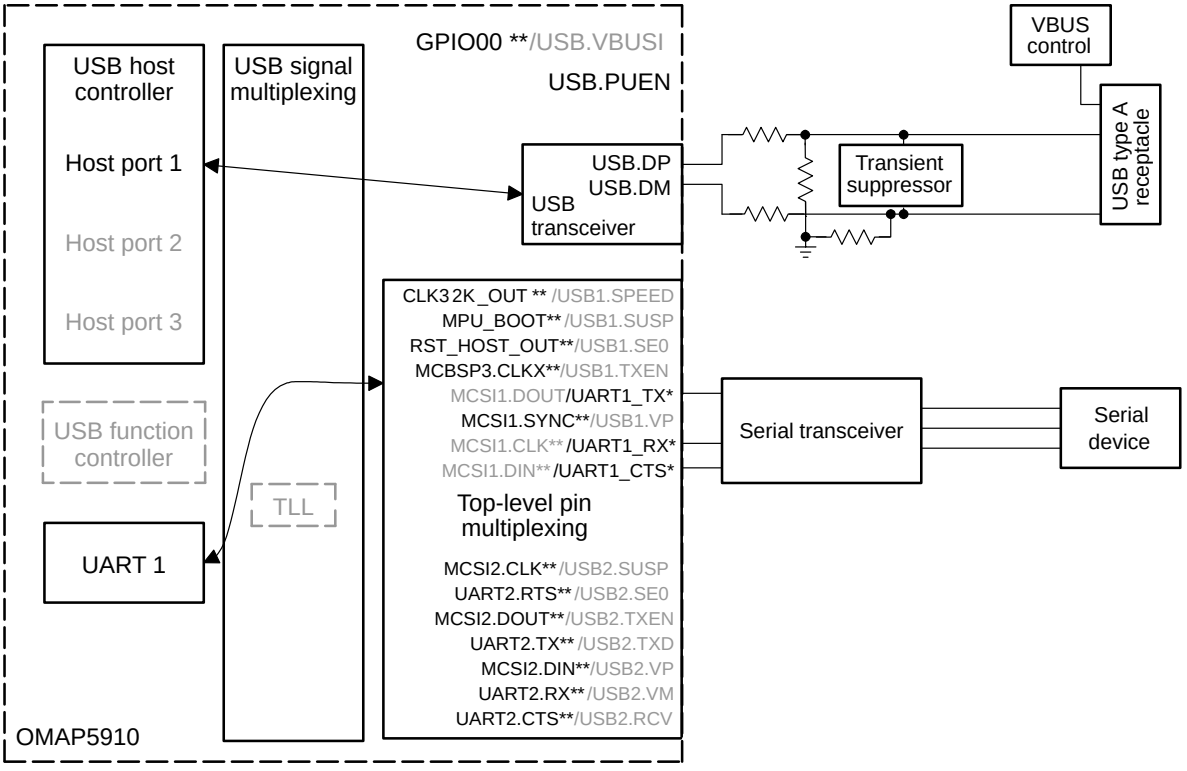
Figure 14–23. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 17



* Assumes pins have configured for USB functionality.
** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 17

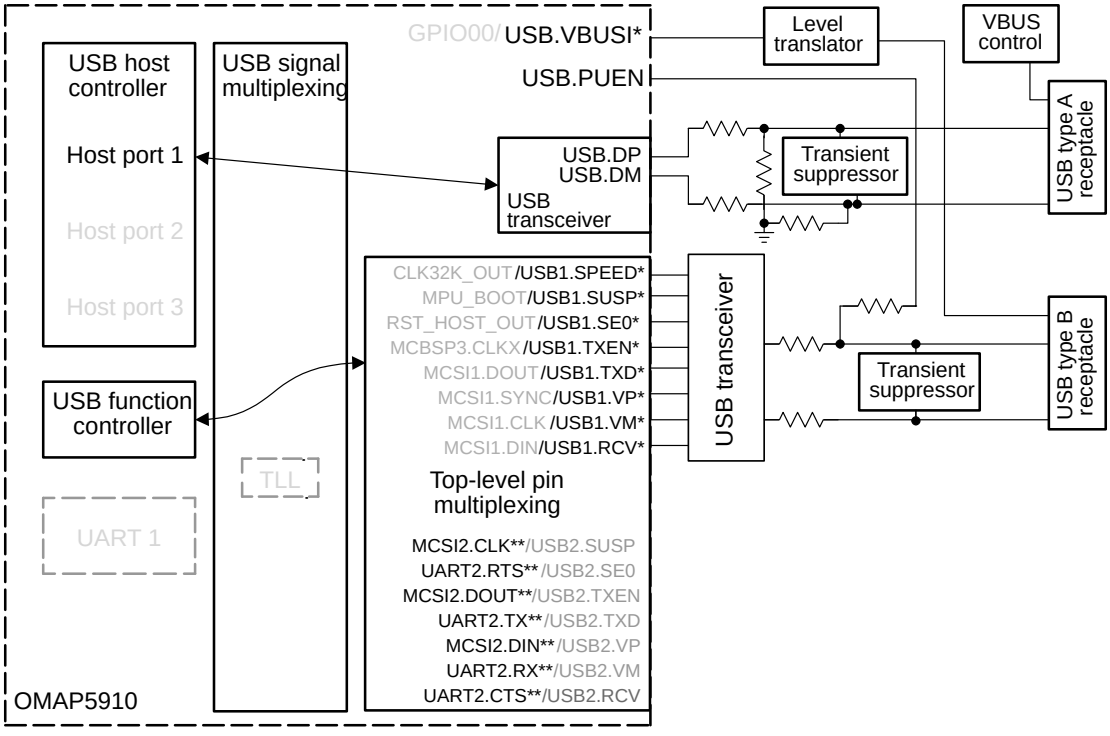
Figure 14–24. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 18



* Assumes pins have configured for USB functionality.
** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 18

Figure 14–25. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 19

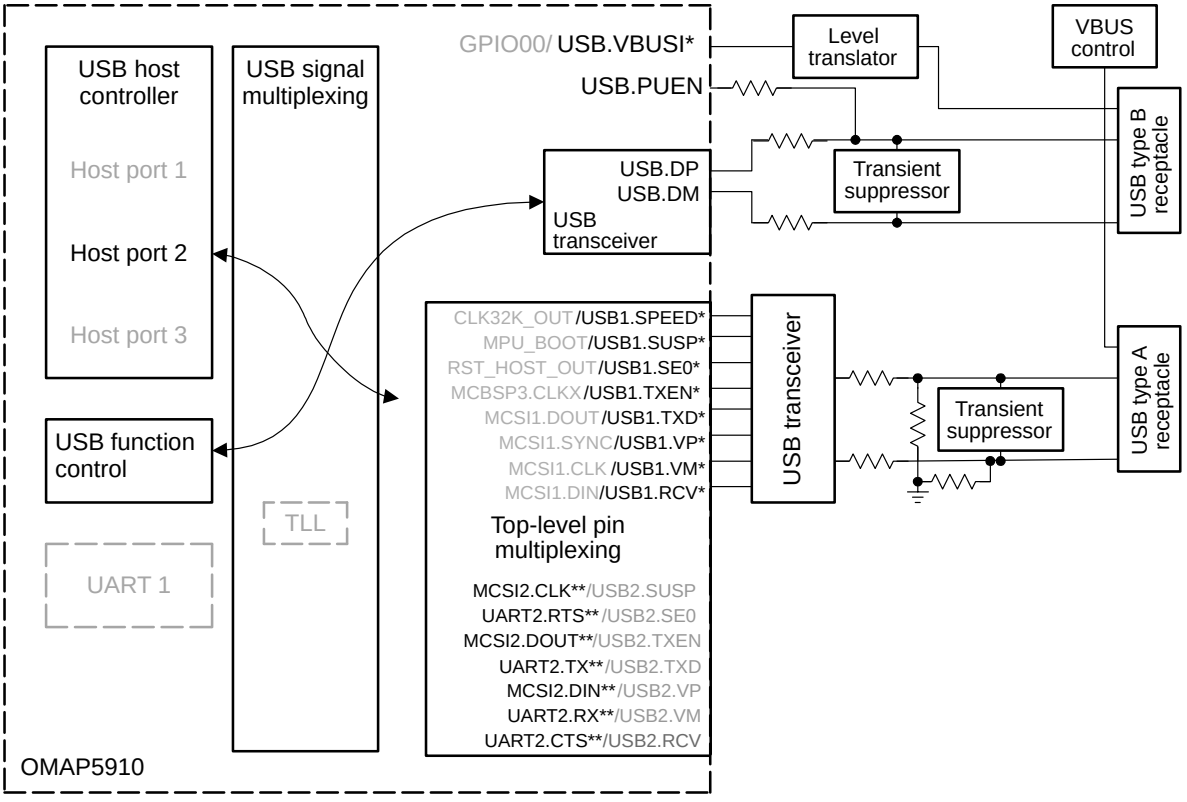


* Assumes pins have configured for USB functionality.

** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 19

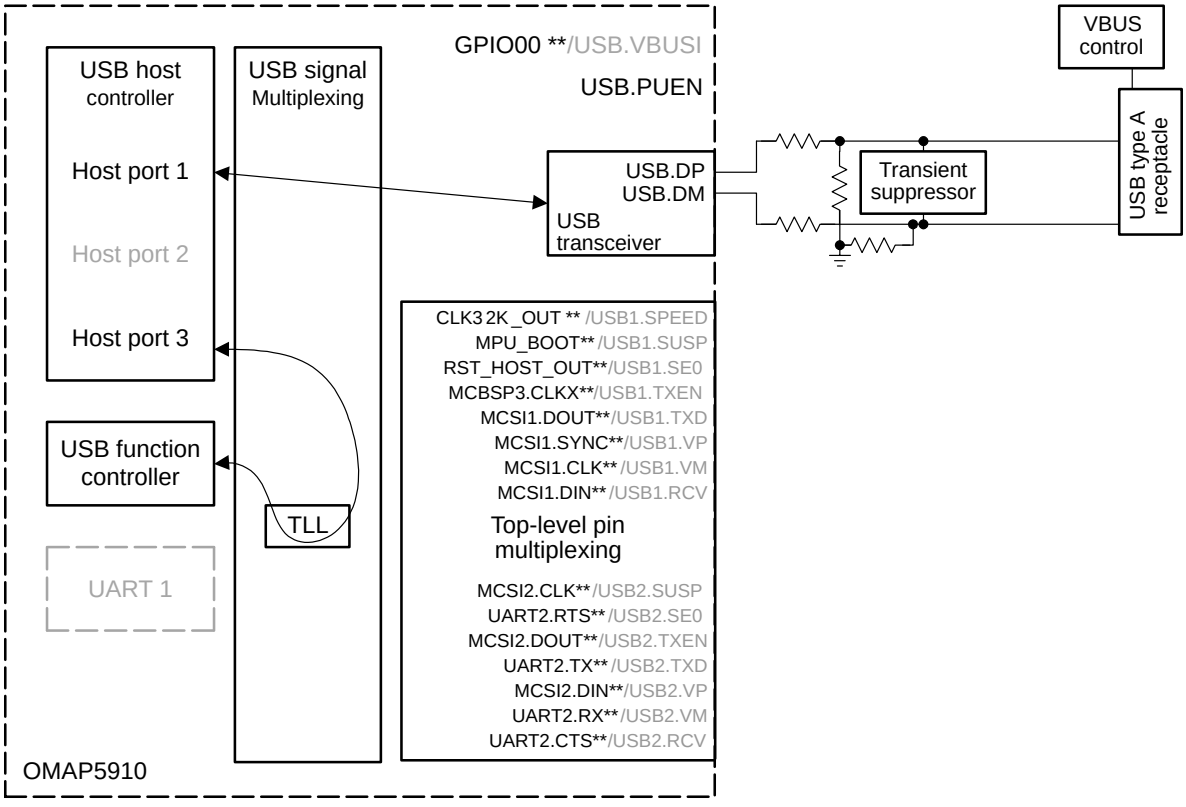
Figure 14–26. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 20



* Assumes pins have configured for USB functionality.
** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 20

Figure 14–27. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 21

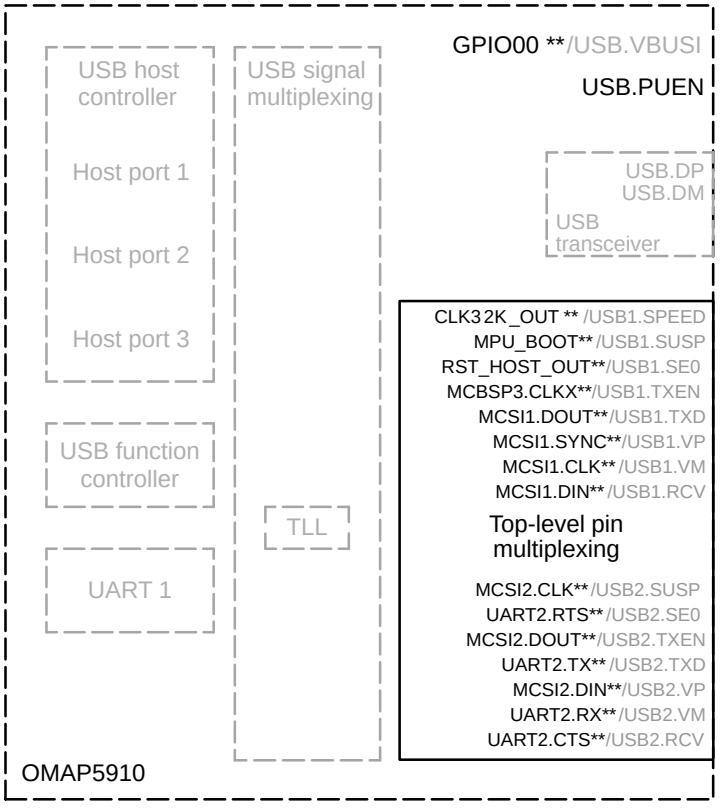


* Assumes pins have configured for USB functionality.

** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 21

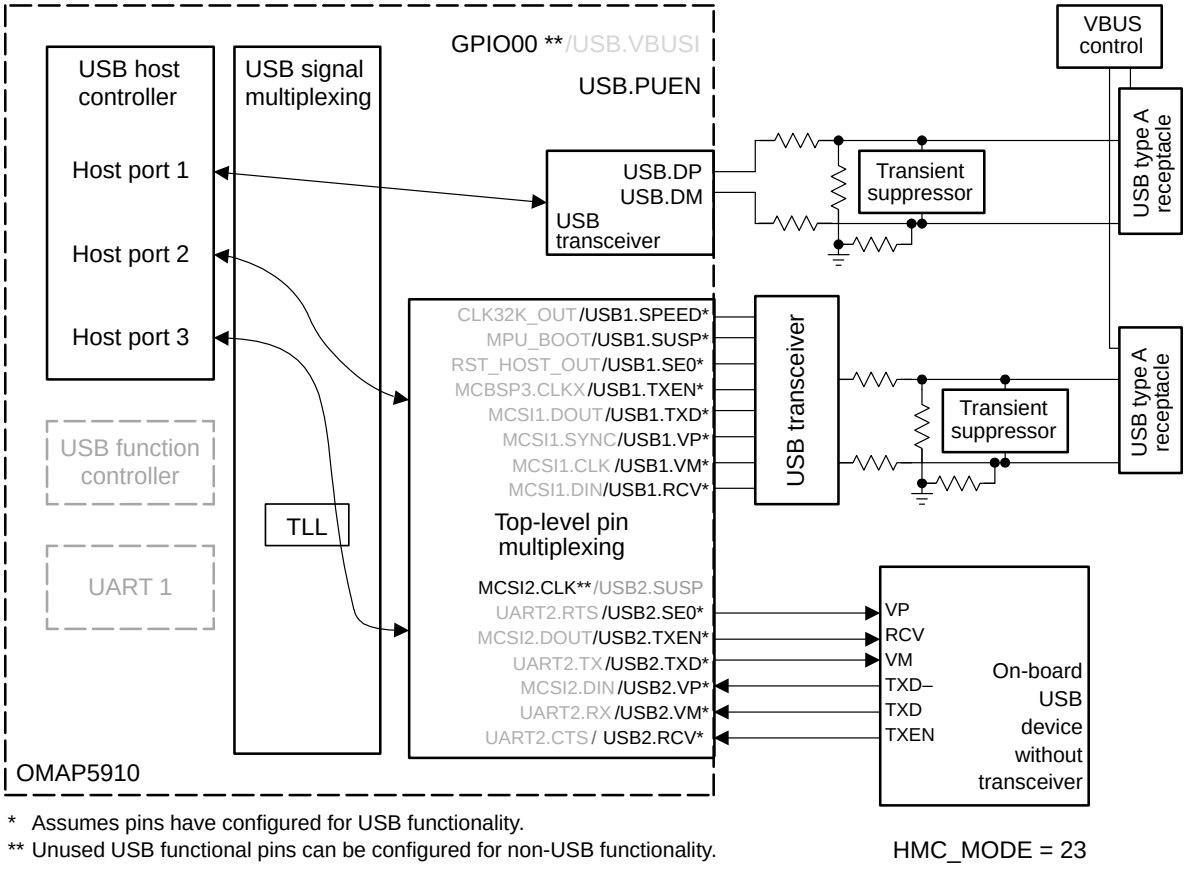
Figure 14–28. OMAP5910 Configured for HMC_MODEs 22, 26-31



HMC_MODE = 22
HMC_MODE = 26
HMC_MODE = 27
HMC_MODE = 28
HMC_MODE = 29
HMC_MODE = 30
HMC_MODE = 31

* Assumes pins have configured for USB functionality.
** Unused USB functional pins can be configured for non-USB functionality.

Figure 14–29. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 23 (Transceiverless Connection Uses TXD+, TXD- Signaling)

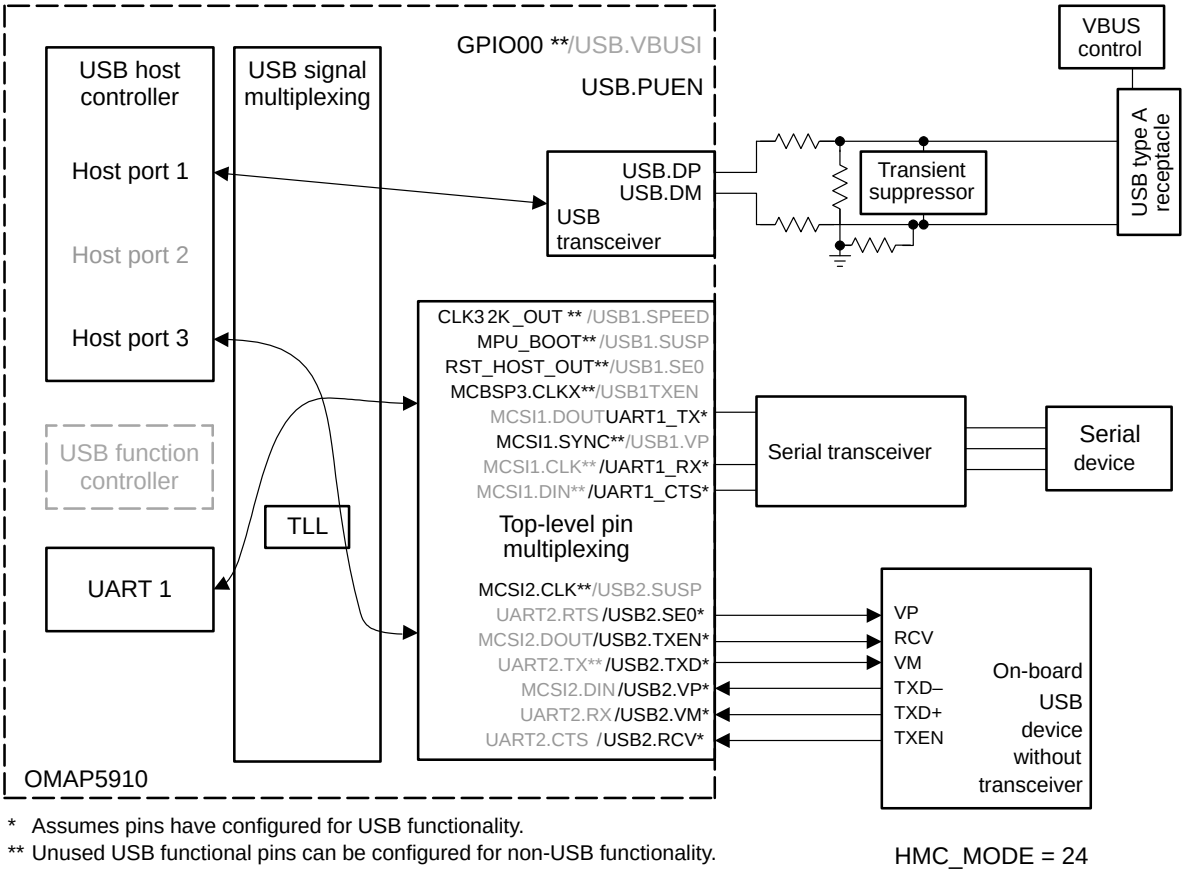


* Assumes pins have configured for USB functionality.

** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 23

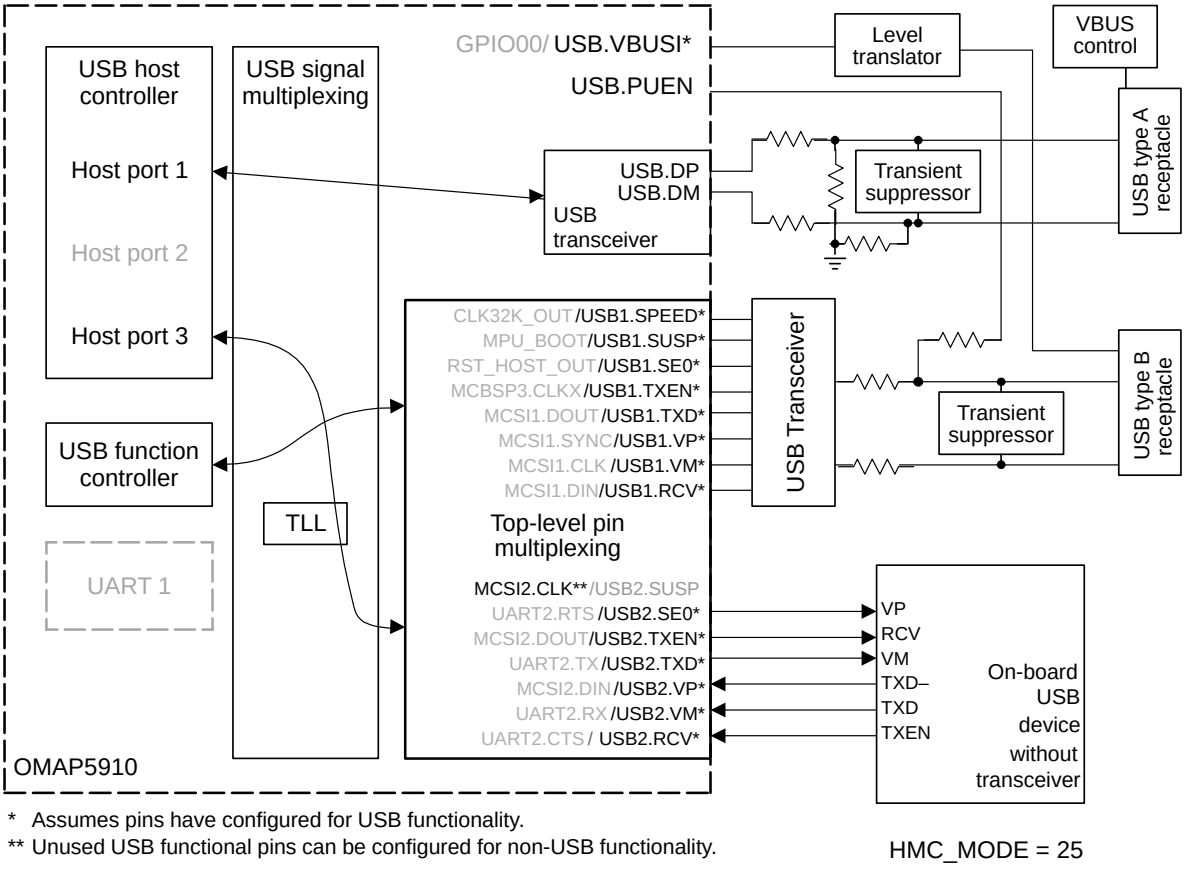
Figure 14–30. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 24 (Transceiverless Connection Uses TXD+, TXD- Signaling)



* Assumes pins have configured for USB functionality.
** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 24

Figure 14–31. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 25 (Transceiverless Connection Uses TXD+, TXD- Signaling)



* Assumes pins have configured for USB functionality.
** Unused USB functional pins can be configured for non-USB functionality.

14.5.5 Ports Shown as Unconnected

Many of the multiplexing modes show cases where a USB host port or the USB function controller are not connected to pins. When a USB signal multiplexing mode is selected that does not connect to a USB host controller port, that host controller port sees signaling that implies no external device is connected. When a USB signal multiplexing mode is selected that does not connect to the USB function controller port, the USB function controller port sees USB single-ended 0 signaling, which implies a USB reset.

14.5.6 Conflicts Between USB Signal Multiplexing and Top-Level Multiplexing

When OMAP5910 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 OMAP5910 top-level signal multiplexing is used.

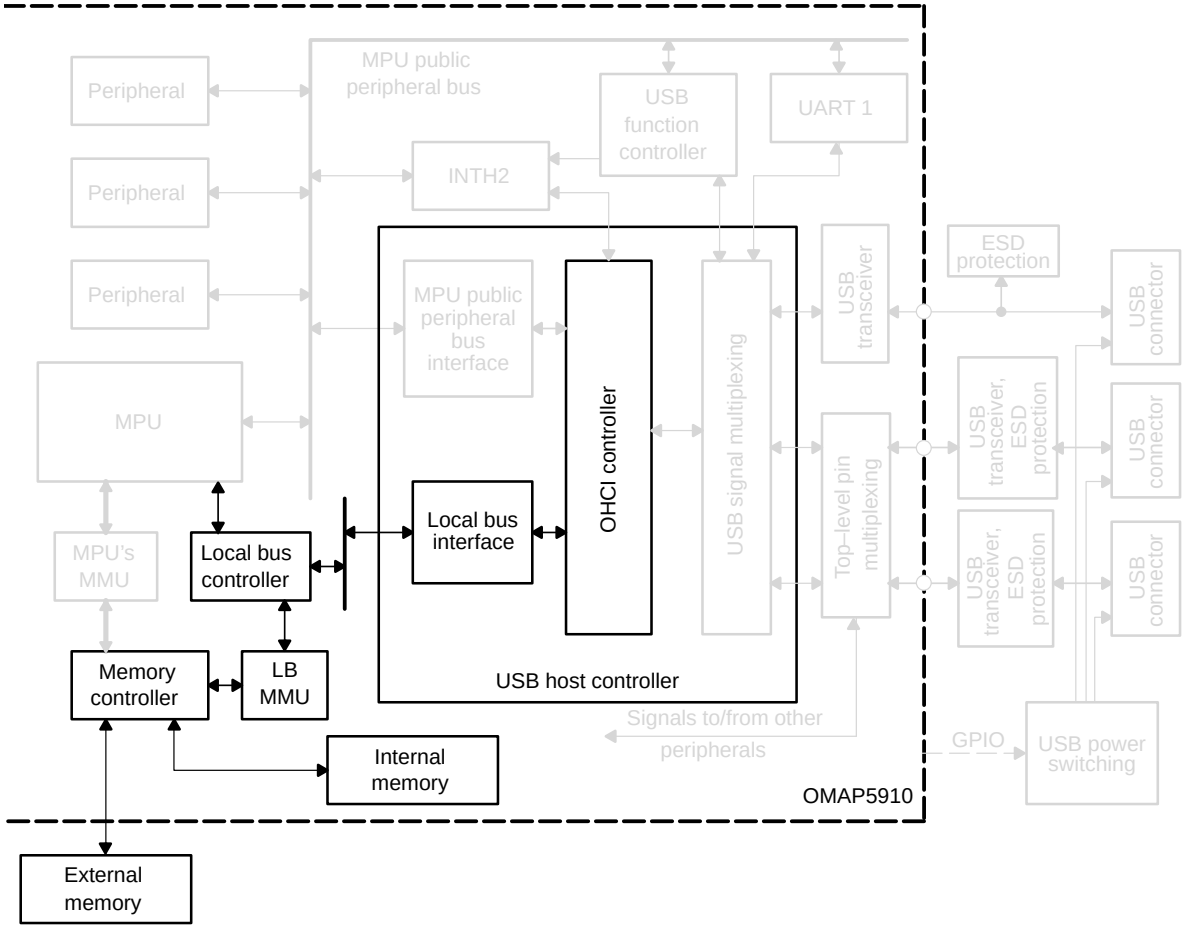
When OMAP5910 top-level signal multiplexing selects non-USB functionality for a pin but the USB signal multiplexing is set to use that pin as an input, the OMAP5910 top-level signal multiplexing presents a low level to the USB signal multiplexer.

It may be useful to select a CONF_MOD_HOST_HMC_MODE_R value that brings some USB signals to the OMAP5910 top-level signal multiplexing, but then set the top-level signal multiplexing to ignore those USB signals.

14.6 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 OMAP5910 local bus controller allows the USB host controller to access OMAP5910 system memory, as shown in Figure 14–32.

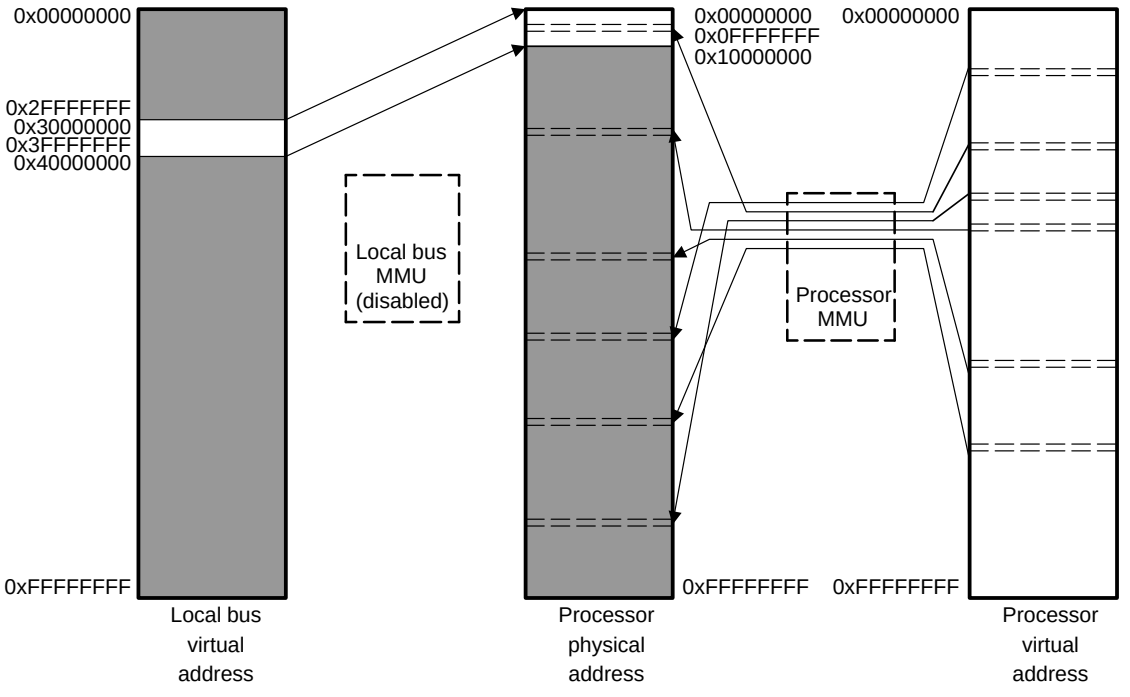
Figure 14–32. OMAP5910 USB Host Controller Data Path to System Memory



14.6.1 Local Bus Virtual Addressing

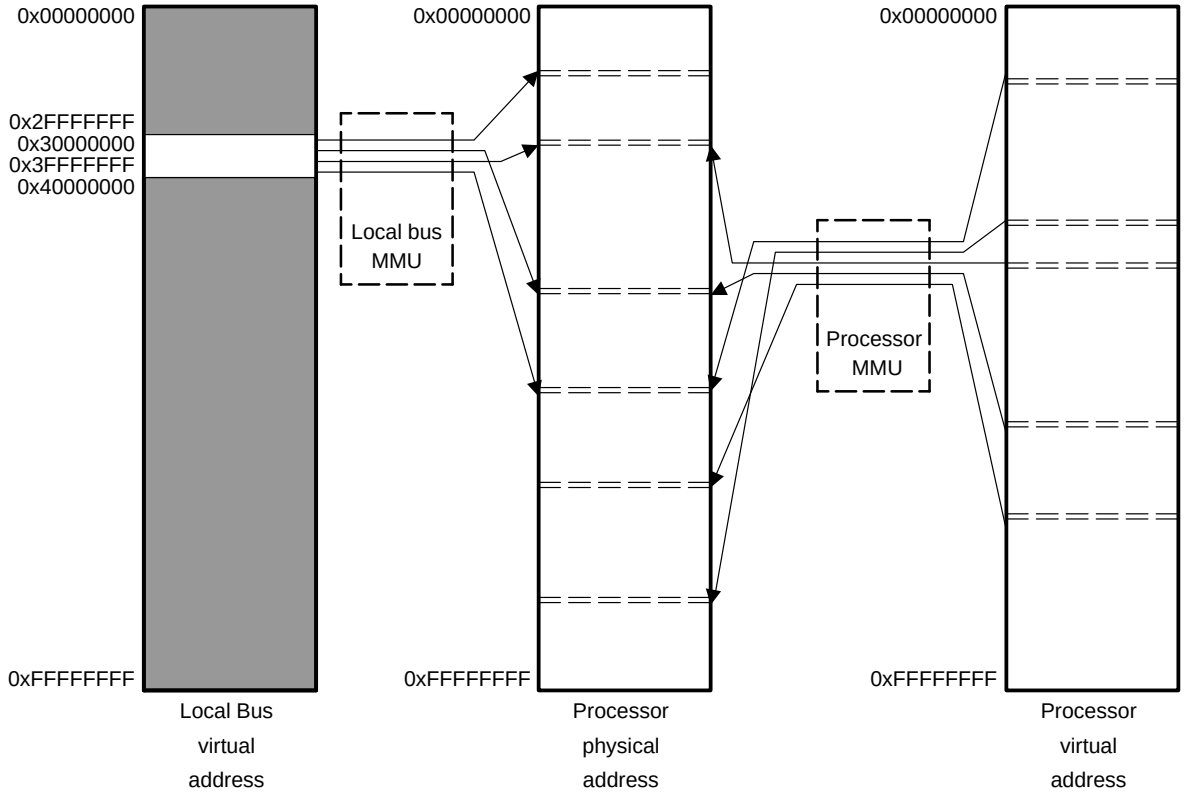
The OMAP5910 local bus implementation requires the USB host controller to use local bus virtual addresses with bits 31-28 set to 0011b, which gives the USB host controller a usable local bus virtual address range of 0x30000000 to 0x3FFFFFFF (a total range of 256 MBytes). If the local bus MMU is disabled, the local bus virtual address range from 0x30000000 to 0x3FFFFFFF automatically maps to processor address range 0x00000000 to 0x0FFFFFFF. With the local bus MMU disabled, it is not possible to access OMAP5910 physical addresses outside of this range. This is shown in Figure 14–33.

Figure 14–33. Relationships Between Processor Virtual Address, Processor Physical Address, and Local Bus Virtual Address with Local Bus MMU Disabled



When the OMAP5910 local bus MMU is enabled, the USB host controller is not limited to processor physical addresses between 0x00000000 and 0x0FFFFFFF. The local bus MMU can be programmed to devirtualize the local bus addresses in the local bus virtual address range 0x30000000-0x3FFFFFFF to any processor physical address. The relationships between local bus virtual addresses, processor virtual addresses, and processor physical addresses is shown in Figure 14–34.

Figure 14–34. Relationships Between Processor Virtual Address, Processor Physical Address, and Local Bus Virtual Address with Local Bus MMU Enabled



If any portion of the USB host controller data structures are stored in memory that is not in the physical address range 0x00000000 to 0x0FFFFFFF, then the local bus MMU must be programmed to provide a different mapping between local bus virtual addresses and physical addresses. A system implementation that does not implement any physical RAM in the physical address range 0x00000000 to 0x0FFFFFFF must enable the local bus MMU in order to use the OMAP5910 USB host controller.

Initialization of the local bus MMU is done using the same sorts of operations as are used for initializing the MPU MMU, except addressing the local bus MMU instead of the MPU MMU.

The processor local bus interface ignores any local bus activity in the local bus virtual address range from 0x0 to 0x2FFFFFFF, and the local bus virtual address range from 0x40000000 to 0xFFFFFFFF. Should the USB host request such an access (through improper setup of the USB host controller registers or improper initialization of endpoint descriptors, transfer descriptors, or the HCCA), the USB host local bus interface times out and signals an

unrecoverable error via the USB host interrupt mechanisms. If the USB host local bus time-out feature is disabled, the USB host controller instead waits indefinitely for completion of the local bus access and therefore locks up the local bus.

14.6.2 Cache Coherency in OHCI Data Structures and Data Buffers

The OMAP5910 traffic controller does not provide mechanisms to flush (or writeback) the MPU cache when a DMA controller or local bus access to system memory occurs. Because there is no forced coherency mechanism, the system implementation must ensure that the OMAP5910 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 the USB host controller could 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 noncacheable system memory. In this case, every MPU access directly accesses physical memory, so there is not a 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 Chapter 2, *MPU Subsystem.*

14.6.3 Local Bus Addressing and OHCI Data Structure Pointers

Because of the limitations described in Section 14.6.1, *Local Bus Virtual Addressing,* care is needed when programming the USB host controller OHCI registers that are pointers to data structures and when initializing the pointers inside those data structures. The USB host controller OHCI registers that point to the HCCA and the ED lists must be programmed with values that are local bus virtual addresses that correspond to the physical addresses of the particular data structure. In most cases, the USB host controller OHCI registers that point to data structures in system memory are not programmed with the address that the MPU software uses to access those data structures.

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 local bus virtual address that points to the most recent transfer descriptor that has been retired. As such, the host controller driver software must be able to convert from the local bus virtual address back to a 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 addressing depends on the settings used in the MPU MMU and local bus MMU; that is, functionality is system-dependent. The conversion functions are described in general terms in the next section.

14.6.3.1 MPUVAtoLBVA()—MPU Virtual Address to Local Bus Virtual Address Conversion Function

This function converts a MPU virtual address to the local bus virtual address that points to the same physical address in system memory. This output from this function is needed in programming the USB host controller registers that point to data structures in system memory and in programming pointers within the ED, TD, and HCCA data structures.

This function must understand the MPU MMU and local bus MMU programming and local bus virtual address restrictions. This routine is generally implemented in a two-step process—MPU virtual address to physical address, then physical address to local bus virtual address. Those two steps are described in section 14.6.3.3, *MPUVAtoPA()—MPU Virtual Address to Physical Address Conversion Function* and section 14.6.3.5, *PAtoLBVA()—Physical Address to Local Bus Virtual Address Conversion Function*.

It is advisable to implement a checking function in this routine that verifies that the resulting local bus virtual address is in the usable local bus virtual address range of 0x30000000 to 0x3FFFFFFF. If the USB host controller attempts to access a local bus virtual address outside of the valid range, an OHCI unrecoverable error occurs if the USB host controller local bus time-out feature is enabled. If the time-out feature is disabled and the USB host controller attempts an access outside of the valid range, the USB host controller local bus interface locks up and the USB host controller is not able to access the system memory data structures needed to issue USB packets.

Take care when converting null pointers. The OHCI USB host controller uses null pointers to indicate the end of lists. Some convention must be used to indicate an MPU null pointer and properly convert that MPU null pointer to a local bus virtual address of 0x00000000.

14.6.3.2 LBVAtoMPUVA()—Local Bus Virtual Address to MPU Virtual Address Conversion Function

This function converts a local bus virtual address to a MPU virtual address. This function is used when the host controller driver must traverse the ED and TD linked lists in system memory, or in traversing the done TD list.

This function must understand both the MPU MMU and local bus MMU programming and local bus virtual address restrictions. This routine is generally implemented in a two step process—local bus virtual address to physical address, then physical address to MPU virtual address. Those two steps are described in section 14.6.3.4, *LBVAtoPA()—Local Bus Virtual Address to Physical Address Conversion Function* and section 14.6.3.6, *PAtoMPUVA()—Physical Address to MPU Virtual Address Conversion Function*.

Take care when converting null pointers. The OHCI USB host controller uses null pointers to indicate the end of lists. Some convention must be used to indicate an MPU null pointer and properly convert a local bus virtual address of 0x000000 to that MPU null pointer value.

14.6.3.3 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.

14.6.3.4 LBVAtoPA()—Local Bus Virtual Address to Physical Address Conversion Function

This function converts a local bus virtual address to the equivalent system physical address. This function must understand the way that the system software has configured the local bus MMU. This function must provide a conversion that has a result that is identical to the conversion done in hardware by the local bus MMU.

This function must also understand the OMAP5910 local bus limitation that requires local bus virtual addresses to be in the range 0x30000000 to 0x3FFFFFFF.

14.6.3.5 PAtoLBVA()—Physical Address to Local Bus Virtual Address Conversion Function

This function is a reverse version of the local bus virtual address to physical address conversion function. It accepts a processor physical address as its argument and returns the equivalent local bus virtual address.

It is possible to program the local bus MMU to allow two (or more) different local bus virtual addresses to map to the same physical address. System software implementers must be careful to avoid that situation.

It is advisable to implement a checking function in this routine that verifies that the resulting local bus virtual address is in the usable local bus virtual address range of 0x30000000 to 0x3FFFFFFF. If the USB host controller attempts to access a local bus virtual address outside of the valid range, an OHCI unrecoverable error occurs if the USB host controller local bus time-out feature is enabled. If the time-out feature is disabled and the USB host controller attempts an access outside of the valid range, the USB host controller local bus interface locks up and the USB host controller is not able to access the system memory data structures needed to issue USB packets.

14.6.3.6 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.

14.6.3.7 Physical, MPU Virtual, and Local Bus Virtual Addresses—an Example

An example helps show the requirements. Consider a system where the MPU places the HCCA and ED and TD lists in SRAM on OMAP5910 CS0 and where the USB data buffers are in external SDRAM. The MPU MMU is set up, in part, as shown in Table 14–31.

Table 14–31. MPU MMU Programming for Address Conversion Example

MPU Virtual Address	Physical Address	Page Size
0x00000000 to 0x000000FFF	0x0005E000 to 0x0005EFFF	Small (4K bytes)
0x00010000 to 0x0001FFFF	0x10170000 to 0x1017FFFF	Large (64K bytes)

Assume that the system software has allocated the following MPU addresses for some OHCI data structures, as shown in Table 14–32.

Table 14–32. MPU Memory Allocations for Address Conversion Example

OHCI Structure	MPU Virtual Address
HHCA base address	0x00000700
First bulk ED on the ED list	0x00000140
First control ED on the ED list	0x00000150
First TD for first ED on bulk list	0x00000D90
First TD for first ED on control ED list	0x00000DA0
Data buffer start for first TD on first ED of bulk list	0x00010007
Data buffer start for first TD on first ED of control list	0x00013423

The corresponding physical addresses for these OHCI structures are shown in Table 14–33.

Table 14–33. Physical Addresses for Address Conversion Example

OHCI Structure	MPU Virtual Address	Physical Address
HHCA base address	0x00000700	0x0005E700
First bulk ED on the ED list	0x00000140	0x0005E140
First control ED on the ED list	0x00000150	0x0005E150
First TD for first ED on bulk list	0x00000D90	0x0005ED90
First TD for first ED on control ED list	0x00000DA0	0x0005EDA0
Data buffer start for first TD on first ED of bulk list	0x00010007	0x10170007
Data buffer start for first TD on first ED of control list	0x00013423	0x10173423

Note: Assumes the MPU MMU initialization shown in Table 14–31 and the MPU memory allocations shown in Table 14–32.

Because the USB data buffers are to be stored outside of the physical address range that may be reached by the USB host controller when the local bus MMU is disabled, it is necessary to enable and program the local bus MMU. A possible way to program the local bus MMU is shown in Table 14–34.

Table 14–34. Local Bus MMU Programming for Address Conversion Example

Local Bus Virtual Address	Physical Address	Page Size
0x30F00000 to 0x30F0FFFF	0x0005E000 to 0x0005EFFF	Small (4 KBytes)
0x32100000 to 0x3210FFFF	0x10170000 to 0x1017FFFF	Large (64 KBytes)

Given this local bus MMU virtual address to physical address mapping, the local bus virtual addresses in Table 14–35 are needed to program the USB host controller OHCI registers and some of the pointers in the ED and TD structures.

Table 14–35. Local Bus Virtual Addresses for Address Conversion Example

OHCI Structure	MPU Virtual Address	Physical Address	Local Bus Virtual Address
HHCA base address	0x00000700	0x0005E700	0x30F00700
First bulk ED on the ED list	0x00000140	0x0005E140	0x30F00140
First control ED on the ED list	0x00000150	0x0005E150	0x30F00150
First TD for first ED on bulk list	0x00000D90	0x0005ED90	0x30F00D90
First TD for first ED on control ED list	0x00000DA0	0x0005EDA0	0x30F00DA0
Data buffer start (CBP) for first TD on first ED of bulk list	0x00010007	0x10170007	0x32100007
Data buffer start (CBP) for first TD on first ED of control list	0x00013423	0x10173423	0x32103423

Given all of the information in Table 14–35, the MPU performs the initializations described in Table 14–36.

Table 14–36. Some Data Structure Initializations for Address Conversion Example

Item initialized	Value	Type of Value
HcHCCA register	0x30F00700	Local bus virtual address
HcBulkHeadED register	0x30F00140	Local bus virtual address
HcControlHeadED register	0x30F00150	Local bus virtual address
TDQueueHeadPointer for first bulk ED	0x30F00D90	Local bus virtual address
TDQueueHeadPointer for first control ED	0x30F00DA0	Local bus virtual address
CBP for first TD of first bulk ED	0x32100007	Local bus virtual address
CBP for first TD of first control ED	0x32103423	Local bus virtual address

To properly initialize the EDs and TDs mentioned here, the MPU software must allocate memory for the data structure and initialize the various fields. To initialize the address pointers in the structure, it begins by getting the MPU virtual address of the item that is pointed to by the data structure pointer. It converts that value to the corresponding local bus virtual address and places that value into the data structure pointer field.

For example, if the USB host controller driver must add a TD to the TD list in the first bulk ED, it must locate the last TD on the TD list and update it with the new TD information. It must then change the NextP pointer of the last TD to point to the local bus virtual address of a newly allocated empty TD. To traverse the TD list to find the last TD, the host controller driver perhaps starts by traversing the ED list. It would start by reading the HcBulkHeadED register, which returns a local bus virtual address. It converts this value to the corresponding MPU virtual address and reads the ED. If the ED is for a different function number or different endpoint number or direction, the driver reads the NextED pointer, converts the value from a local bus virtual address to a MPU virtual address, and checks this new ED for function number and endpoint number. Once it finds an ED that matches in function number and endpoint number (and perhaps endpoint direction), it begins traversing the TD list associated with the ED.

To traverse the TD list, the driver begins by saving a copy of the TailP value from the ED and by reading the HeadP value from the ED, which is a local bus virtual address. The driver converts this to a MPU virtual address, and reads

the TD. If the NextTD pointer is not the same as the saved copy of the TailP value, the TD is the not last TD in the list, and the driver converts the NextTD local bus virtual address to a MPU virtual address, fetches that TD, and compares the NextTD pointer to the saved TailP value. This process continues until the driver finds a TD with a NextTD value that matches the saved TD value.

Once the last TD on the correct list has been found, the driver copies the new TD information into the last TD, allocates memory for a new TD, converts the address of the new TD to a local bus virtual address, and updates the NextTD value in the last TD to point to the newly allocated TD. It also updates the TailP value in the ED to the local bus virtual address of the newly allocated TD.

14.6.4 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 and must not mistake 0x00000000, the null pointer, for local bus virtual address 0x30000000, which may point to a valid location in physical system memory.

14.6.5 Endianism and USB Host Controller Access to System Memory

The *OHCI Specification for USB* defines a little-endian controller. Since the OMAP5910 USB host controller is OHCI-compliant, it is defined for use in little-endian systems. The OMAP5910 MPU core and subsystem are also little endian.

14.6.5.1 Endianism and OHCI Endpoint and Transfer Descriptors

OHCI endpoint and transfer descriptors are implemented as shown in the *OHCI Specification for USB*, with the bit positions in the endpoint and transfer descriptor data structures representing bit positions on the physical 32-bit data bus.

If using C structures for the EDs and TDs, be careful of data alignment of the structures and the implications of the MPU's little endianism. The *OHCI Specification for USB* requires that EDs and TDs be aligned at 32-bit boundaries in system memory (local bus virtual address LS 4 bits must be 0). It may be useful to validate that your data structures are being properly initialized in memory by performing 32-bit reads to the initialized data structures and comparing the 32-bit read values to the intended values within the bit-fields. However, because the USB host controller and the MPU core both use little endian addressing, this should not arise any issues on the OMAP5910 device.

14.6.5.2 Endianism and OHCI Data Buffers

The OMAP5910 MPU subsystem only supports little endian operation. The USB host controller assumes little-endian data addressing.

Little endian addressing imposes the following data positions within a 32-bit aligned, 32-bit data value in memory, as shown in Table 14–37.

Table 14–37. Little Endian Data Alignment Within 32-Bit Word

		Bit Position Within 32-Bit Word			
Address Offset	**Data Size**	**31 24**	**23 16**	**15 8**	**7 0**
0x0	1				First byte
0x0	2			Second byte	First byte
0x0	3		Third byte	Second byte	First byte
0x0	4	Fourth byte	Third byte	Second byte	First byte
0x1	1			First byte	
0x1	2		Second byte	First byte	
0x1	3	Third byte	Second byte	First byte	
0x2	1		First byte		
0x2	2	Second byte	First byte		
0x3	1	First byte			

14.7 OMAP5910 Local Bus

The OMAP5910 local bus supports both slave and master peripherals. This bus allows the MPU, DSP, and DMA controller to access local bus slave peripherals and allows local bus master peripherals to move data to and from system memory.

The OMAP5910 device does not implement any local bus slave peripherals. The local bus interface of the OMAP5910 USB host controller implements a local bus master peripheral which enables the USB host controller to access system memory via the OMAP5910 local bus and the OMAP5910 traffic controller.

14.7.1 LB Register Descriptions

Table 14–38 lists the registers associated with local bus (LB) control and status. Table 14–39 through Table 14–47 describe specific register bits.

The LB_CLOCK_DIV register has a direct impact on the ability of the USB host controller to access OMAP5910 system memory. The remaining OMAP5910 local bus control and status registers have no direct effect, because they control MPU, DSP, and DMA controller accesses to slave peripherals addressed via local bus and there are no slave peripherals on the OMAP5910 local bus.

The local bus memory management unit and its registers are discussed separately in section 14.8, *OMAP5910 Local Bus MMU*.

Table 14–38. Local Bus Control Registers

Name	Description	R/W	Size†	Address
LB_MPU_TIMEOUT	LB MPU time-out	R/W	32	FFFE:C100h
LB_HOLD_TIMER	LB hold timer	R/W	32	FFFE:C104h
LB_PRIORITY_REG	LB priority	R/W	32	FFFE:C108h
LB_CLOCK_DIV	LB clock divider	R/W	32	FFFE:C10Ch
LB_ABORT_ADD	LB abort address	R	32	FFFE:C110h
LB_ABORT_DATA	LB abort data	R	32	FFFE:C114h
LB_ABORT_STATUS	LB abort status	R	32	FFFE:C118h
LB_IRQ_OUTPUT	LB IRQ output	R/W	32	FFFE:C11Ch
LB_IRQ_INPUT	LB IRQ input	R	32	FFFE:C120h

† Access to these registers must be by 32-bit reads or 32-bit writes. Use of other access sizes may result in undefined operation.

14.7.2 LB MPU Time-out Register (LB_MPU_TIMEOUT)

This register controls the maximum number of local bus clocks allowed for an access by the OMAP5910 MPU, DSP, or DMA controller to a local bus slave device before signaling a local bus abort. This register has no effect on OMAP5910 USB host controller accesses to system memory.

Table 14–39. LB MPU Time-out Register (LB_MPU_TIMEOUT)

Bit	Name	Description	Type	Reset Value
31–8	Reserved	Reserved	R	0
7–0	TIMEOUT	Local bus slave access time-out Number of local bus clocks to wait for completion of a local bus cycle initiated by the MPU, DSP, or DMA before signaling a local bus abort. This time-out does not apply to local bus cycles initiated by the OMAP5910 USB host controller. Because there are no local bus slave peripherals in the OMAP5910 device, this register can be set to a low value like 3. This causes the local bus controller to issue a local bus abort interrupt if the DSP, MPU, or DMA controller mistakenly attempts to access a physical address associated with the local bus slave memory space.	R/W	0xFF

14.7.3 LB Hold Timer Register (LB_HOLD_TIMER)

This register configures the local bus controller bus request and bus hold functions. This register has no effect on OMAP5910 USB host controller accesses to system memory.

Table 14–40. LB Hold Timer Register (LB_HOLD_TIMER)

Bit	Name	Description	Type	Reset Value
31–10	Reserved	Reserved	R	0
9	BREQ_TIMER_EN	Local bus request timer enable Because there are no local bus slave peripherals on OMAP5910, it is recommended that this register be set to 1 to enable the request timer. This helps prevent lockup of the local bus.	R/W	0
8	HOLD_TIMER_EN	Local bus hold timer enable Because there are no local bus slave peripherals on OMAP5910, it is recommended that this register be set to 1 to enable the HOLD timer. This helps prevent lockup of the local bus.	R/W	0
7–0	HOLD_TIMER	Local bus hold timer value Because there are no local bus slave peripherals on OMAP5910, it is recommended that this register be set to a small number like 5 so that the hold timer does not need to wait a long time before signaling an abort. This speeds return of the local bus abort interrupt should the DSP, MPU, or DMA controller accidentally access the local bus.	R/W	0

14.7.4 LB Priority Register (LB_PRIORITY_REG)

The priority register is not used in OMAP5910.

Table 14–41. LB Priority Register (LB_PRIORITY_REG)

Bit	Name	Description	Type	Reset Value
31–0	Reserved	Reserved for future expansion. These bits should always be written as 0.	R	0

14.7.5 LB Clock Divider Register (LB_CLOCK_DIV)

This register controls local bus clocking. This clocking affects accesses by the USB host controller to system memory, as well as DMA controller, DSP, and MPU accesses to local bus slave peripherals. Because OMAP5910 does not implement any local bus slave peripherals, the local bus clock rate mainly affects the USB host controller accesses to system memory.

This register also provides masks for external local bus interrupts and local bus abort interrupts that can be routed to the MPU level 1 interrupt controller.

The clock division register can be read in user and supervisor modes and can be written in supervisor mode only.

Table 14–42. LB Clock Divider Register (LB_CLOCK_DIV)

Bit	Name	Value	Description	Type	Reset Value
31–8	Reserved		Reserved	R	0
7	LB_ABORT_MASK		Local bus abort interrupt mask	R/W	1
		0	Local bus abort interrupt is enabled.		
		1	Local bus abort interrupt is disabled.		
			When enabled and the DSP, MPU, or DMA controller attempts to access a local bus slave peripheral, the local bus time-out counter counts down to zero and then signal a local bus abort. When disabled and the DSP, MPU, or DMA controller attempts to access a local bus slave peripheral, the local bus time-out counter does not count and the cycle locks up the local bus.		
			Since OMAP5910 does not implement any local bus slave peripherals, it is recommended that this bit be cleared and that system software trap the local bus interrupt at MPU level 1 interrupt handler IRQ29 input.		
6–3	LB_IRQ_IN_MASK		Local bus interrupt input mask	R/W	0xF
			These bits enable (when 0) or disable (when 1) local bus interrupt inputs.		
			OMAP5910 does not provide interrupt sources for these interrupts, so it is recommended that these bits remain set to 1 to ensure future compatibility.		

Table 14–42. LB Clock Divider Register (LB_CLOCK_DIV) (Continued)

Bit	Name	Value	Description	Type	Reset Value
2–0	LB_CLK_DIV		Local bus clock divisor:	R/W	4
		010	Local bus clock is transfer controller clock divided by 2.		
		100	Local bus clock is transfer controller clock divided by 4.		
		110	Local bus clock is transfer controller clock divided by 6.		
			Other values: Reserved		
			These bits control the local bus clock rate. The clock for the local bus is derived from the OMAP5910 traffic controller clock. The local bus clock can be set to be transfer controller clock divided by 2, divided by 4, or divided by 6. All other values are reserved and must not be used.		
			This field must be set to a suitable value to allow USB host controller access to system memory. The USB host controller supports a maximum local bus clock frequency of 50 MHz. Be sure that the divisor you select does not result in a local bus clock frequency greater than 50 MHz.		

14.7.6 LB Abort Address Register (LB_ABORT_ADD)

This register captures the local bus address if the MPU, DSP, or DMA controller attempts to access a local bus slave peripheral and a local bus time-out occurs.

This register is not affected by OMAP5910 USB host controller accesses to system memory.

Table 14–43. LB Abort Address Register (LB_ABORT_ADD)

Bit	Name	Description	Type	Reset Value
31–0	LB_ABORT_ADD	Local bus address of last aborted local bus cycle When the DMA controller, MPU, or DSP attempts to access a local bus slave peripheral and a local bus abort occurs, the address of the cycle is captured to this register.	R	0xFFFF FFFF

14.7.7 LB Abort Data Register (LB_ ABORT_DATA)

This register reports the data bus value for the last DMA controller, DSP, or MPU local bus cycle that was aborted.

This register is not affected by OMAP5910 USB host controller accesses to system memory.

Table 14–44. LB Abort Data Register (LB_ ABORT_DATA)

Bit	Name	Description	Type	Reset Value
31–0	LB_ABORT_DATA	Data for aborted local bus access attempt This register reflects the data seen on the local bus when a local bus access is aborted.	R	0xFFFF FFFF

14.7.8 LB Abort Status Register (LB_ABORT_STATUS)

This register provides the local bus cycle status for the last aborted local bus cycle that was issued by the DMA controller, DSP, or MPU. This register is not affected by OMAP5910 USB host controller accesses to system memory.

Table 14–45. LB Abort Status Register (LB_ABORT_STATUS)

Bit	Name	Description	Type	Reset Value
31–8	Reserved	Reserved	R	0
7–4	LB_BE	Byte enables from aborted local bus access Active-low byte enables seen at the last aborted local bus access that was issued by the DSP, DMA controller, or MPU.	R	0
3	LB_RD	Cycle type from aborted local bus access Indicates, when 1, that the last aborted local bus access was a read. When 0, indicates that the last aborted local bus access was a write.	R	0
2	LB_DMA	DMA sourced aborted local bus access When 1, indicates that the last aborted local bus access was sourced by the DMA controller.	R	0
1	LB_DSP	DSP sourced aborted local bus access When high, indicates that the last aborted local bus access was sourced by the DSP.	R	0
0	LB_MPU	MPU sourced aborted local bus access When high, indicates that the last aborted local bus access was sourced by the MPU controller.	R	0

14.7.9 LB IRQ Output Register (LB_IRQ_OUTPUT)

This register is reserved for future expansion and should be left at 0.

Table 14–46. LB IRQ Output Register (LB_IRQ_OUTPUT)

Bit	Name	Description	Type	Reset Value
31–0	Reserved	Reserved	R	0

14.7.10 LB IRQ Input Register (LB_IRQ_INPUT)

This register reports the status of local bus interrupts, including interrupts from external sources (which are not used in OMAP5910) and the status of the local bus abort interrupt.

This register has no effect on OMAP5910 USB host controller accesses to system memory and is not affected by USB host controller accesses to system memory.

Table 14–47. LB IRQ Input Register (LB_IRQ_INPUT)

Bit	Name	Description	Type	Reset Value
31–5	Reserved	Reserved	R	0
4	ABORT_STAT	Local bus abort status 0: Local bus abort has occurred since last read of LB_IRQ_INPUT. 1: Local bus abort has not occurred since last read of LB_IRQ_INPUT. Reading this register sets this bit to 1. Writes have no effect. A local bus abort event causes an interrupt to the MPU level 1 interrupt handler if the LB_ABORT_MASK bit of the LB_CLOCK_DIV register is set to 0. Because the DMA controller, DSP, and MPU could mistakenly attempt to access a local bus slave peripheral address, it is recommended that system software trap the local bus abort interrupt and signal a system error should one occur.	R	1
3–0	IRQ_IN_STAT	Reserved for future expansion. These bits can be ignored.	R	0xF

14.7.11 Local Bus Initialization

Proper operation of the OMAP5910 local bus (for USB host controller access to system memory) requires that the FUNC_MUX_CTRL_0 register be properly initialized. FUNC_MUX_CTRL_0 bits 1:0 must be set either to 00b or to 11b in order to support correct local bus activity.

The local bus clock rate must be chosen to provide a suitable clock to the USB host controller. The local bus clock rate is controlled by the LB_CLOCK_DIV register LB_CLK_DIV field and is generated by dividing the OMAP5910 transfer controller clock by 2, 4, or 6. The local bus clock rate must be programmed to provide a local bus clock frequency that is 50 MHz or slower.

14.7.12 Local Bus Virtual Addressing

When an MPU, DSP, or DMA access to the local bus causes a local bus abort interrupt, the interrupt service routine for MPU level 1 interrupt IRQ29 must read the LB IRQ input register (LB_IRQ_INPUT) to clear the IRQ and then must clear the IRQ at the MPU level 1 interrupt handler input IRQ29.

14.8 OMAP5910 Local Bus MMU

The local bus memory management unit (MMU) is used by the local bus interface for address management of bus cycles initiated by the OMAP5910 USB host controller. The local bus MMU manages local bus virtual addresses, including conversion of local bus virtual addresses to physical addresses and monitoring of access permissions. The local bus MMU can be initialized by the MPU to allow the USB host controller to access the full range of OMAP5910 system memory. A detailed functional description of the MMU architecture can be found in Chapter 2, *MPU Subsystem*.

The local bus MMU includes the following basic blocks:

- A 32-entry translation look aside buffer (TLB)
- Walking table logic
- Registers for recording fault status and fault address

When properly configured and enabled, the walking table logic automatically performs the address conversion from local bus virtual address to physical address (for USB host controller accesses to system memory). Alternately, the walking table logic can be disabled, which allows the MPU to perform local bus virtual address translation under software control.

The local bus MMU is configured via registers which are on the MPU private peripheral bus. The MPU is responsible for correctly configuring the local bus MMU memory mapping functions.

The local bus MMU operates much like the MPU and DSP MMUs. The local bus MMU does not use the upper 4 bits of the local bus virtual address (they are ignored by the local bus MMU hardware) and does not support the prefetch feature.

14.8.1 OMAP5910 Local Bus MMU Registers

The OMAP5910 local bus MMU registers are listed in Table 14–48. Table 14–49 through Table 14–67 describe the specific register bits.

Table 14–48. Local Bus MMU Registers

Name	Description	User Access	Supervisor Access	Size	Address
Reserved	Reserved	R	R/W	16	FFFE:C200h
LB_MMU_WALKING_ ST_REG	LB MMU walking status	R	R/W	16	FFFE:C204h
LB_MMU_CNTL_REG	LB MMU control	R	R/W	16	FFFE:C208h
LB_MMU_FAULT_AD_ H_REG	LB MMU fault address high	R	R	16	FFFE:C20Ch
LB_MMU_FAULT_AD_ L_REG	LB MMU fault address low	R	R	16	FFFE:C210h
LB_MMU_FAULT_ST_REG	LB MMU fault status	R	R	16	FFFE:C214h
LB_MMU_IT_ACK_REG	LB MMU interrupt acknowledge	W	W	16	FFFE:C218h
LB_MMU_TTB_H_REG	LB MMU TTB high	R	R/W	16	FFFE:C21Ch
LB_MMU_TTB_L_REG	LB MMU TTB low	R	R/W	16	FFFE:C220h
LB_MMU_LOCK_REG	LB MMU lock counter	R/W	R/W	16	FFFE:C224h
LB_MMU_LD_TLB_REG	LB MMU TLB load/read	R/W	R/W	16	FFFE:C228h
LB_MMU_CAM_H_REG	LB MMU CAM high	R/W	R/W	16	FFFE:C22Ch
LB_MMU_CAM_L_REG	LB MMU CAM low	R/W	R/W	16	FFFE:C230h
LB_MMU_RAM_H_REG	LB MMU RAM high	R/W	R/W	16	FFFE:C234h
LB_MMU_RAM_L_REG	LB MMU RAM low	R/W	R/W	16	FFFE:C238h
LB_MMU_GFLUSH_REG	LB MMU global flush	R/W	R/W	16	FFFE:C23Ch
LB_MMU_FLUSH_ENTRY_ REG	LB MMU flush entry	R/W	R/W	16	FFFE:C240h
LB_MMU_READ_CAM_H_ REG	LB MMU CAM read high	R/W	R/W	16	FFFE:C244h
LB_MMU_READ_CAM_ L_REG	LB MMU CAM read low	R/W	R/W	16	FFFE:C248h

Table 14–48. Local Bus MMU Registers (Continued)

Name	Description	User Access	Supervisor Access	Size	Address
LB_MMU_READ_RAM_H_REG	LB MMU RAM read high	R/W	R/W	16	FFFE:C24Ch
LB_MMU_READ_RAM_L_REG	LB MMU RAM read low	R/W	R/W	16	FFFE:C250h

This register reports the local bus MMU walking table status.

Table 14–49. LB MMU Walking Status Register (LB_MMU_WALKING_ST_REG)

Bit	Name	Function	Access		Hardware Reset Value
			User	Sup	
15–:2	Reserved	Reserved	-	-	-
1	Wtl_working	When 1, the walking table is active.	R	R/W	0
0	Reserved	Reserved	-	-	-

The LB MMU control register controls the local bus MMU reset and enable functions.

Table 14–50. LB MMU Control Register (LB_MMU_CNTL_REG)

Bit	Name	Function	Access		Hardware Reset Value
			User	Sup	
15–6	Reserved	Reserved	-	-	-
5	Burst_16_mngt_en	When 1, enables 16 bit burst access management	R	R/W	0
4–3	Reserved	Reserved	R	R	00
2	Wtl_en	When 1, enables the walking table logic. When 0, the walking table is disabled and access to the TLB and lock counter are disabled.	R	R/W	0
1	MMU_en	Local bus MMU enable 0: Local bus MMU is disabled. 1: Local bus MMU is enabled. When 0, the local bus MMU is disabled and local bus virtual addresses in the range 0000:0000h to 0FFF:FFFF are mapped directly to physical address range 0000:0000h to 00FF:FFFFh.	R	R/W	0
0	Reset_sw	When 0, holds the local bus MMU in reset. When 1, the local bus MMU is not held in reset. Software must set this bit to allow the MMU to function.	R	R/W	0

The LB MMU fault address registers report the local bus virtual address of the last local bus access which caused a local bus MMU fault.

Table 14–51. LB MMU Fault Address High Register (LB_MMU_FAULT_AD_H_REG)

Bit	Name	Function	Access		Hardware Reset Value
			User	Sup	
15–0	Fault_address_MSB	Most significant 16 bits of the local bus address that caused a local bus MMU fault. The most significant 4 bits are always 0000.	R	R	0x0000

Table 14–52. LB MMU Fault Address Low Register (LB_MMU_FAULT_AD_L_REG)

Bit	Name	Function	Access		Hardware Reset Value
			User	Sup	
15–0	Fault_address_LSB	Least significant 16 bits of the local bus address that caused a local bus MMU fault	R	R	0x0000

The LB MMU fault status register provides information on the type of fault encountered at the last local bus MMU fault.

Table 14–53. LB MMU Fault Status Register (LB_MMU_FAULT_ST_REG)

Bit	Name	Function	Access		Hardware Reset Value
			User	Sup	
15–3	Reserved	Reserved	-	-	-
2	Perm_fault	Permission fault. Active high	R	R	0
1	Tlb_miss	TLB miss (when WTB disabled). Active high	R	R	0
0	Trans_fault	Translation fault (invalid descriptor). Active high	R	R	0

The LB MMU register is used to acknowledge the local bus MMU interrupt at the local bus MMU interrupt generator. Acknowledging the interrupt at the local bus MMU interrupt generator causes the generator to deassert its interrupt indication.

In response to a local bus MMU interrupt, the interrupt service routine must:

1) Read LB_MMU_FAULT_AD_L_REG and LB_MMU_FAULT_AD_H_REG.
2) Read LB_MMU_FAULT_ST_REG.
3) Determine how to respond to the faulty access.
4) Acknowledge the interrupt by writing a 1 to LB_MMU_IT_ACK_REG.
5) Acknowledge the interrupt at MPU level 1 interrupt handler IRQ17.

Table 14–54. LB MMU Interrupt Acknowledge Register (LB_MMU_IT_ACK_REG)

Bit	Name	Function	Access		Hardware Reset Value
			User	Sup	
15–1	Reserved	Reserved	-	-	-
0	It_ack	Write a 1 to this bit to acknowledge the interrupt. A write of 0 has no effect; a write of 1 clears the bit automatically.	W	W	0

The LB TTB address registers define the physical address of the local bus MMU translation table base (TTB).

Table 14–55. LB MMU TTB Address High Register (LB_MMU_TTB_H_REG)

Bit	Name	Function	Access		Hardware Reset Value
			User	Sup	
15–0	TTB_REG_H	Most significant 16 bits of physical address of translation table base address	R	R/W	0

Table 14–56. LB MMU TTB Address Low Register (LB_MMU_TTB_L_REG)

Bit	Name	Function	Access		Hardware Reset Value
			User	Sup	
15–0	TTB_REG_L	Least significant 16 bits of physical address of translation table base address. Bits 9-0 must always be 0.	R	R/W	0

Table 14–57. LB MMU Lock Counter Register (LB_MMU_LOCK_REG)

Bit	Name	Function	Access		Hardware Reset Value
			User	Sup	
15–10	Base_value	Locked entries base value	R/W	R/W	0
9–4	Current_victim	Current entry pointed to by the WTL. Base_value <= Current_victim <= 31	R/W	R/W	0
3–0	Unused				

To write one entry into the local bus MMU TLB, use the following procedure:

1) Disable the table walking logic (if not already disabled).
2) Write the appropriate value to LB_MMU_CAM_H_REG.
3) Write the appropriate value to LB_MMU_CAM_L_REG.
4) Write the appropriate value to LB_MMU_RAM_H_REG.
5) Write the appropriate value to LB_MMU_RAM_L_REG.
6) Write the TLB entry number to be modified into Current_victim: number must be equal to or greater than the current value of Base_value and must be less than or equal to 31.
7) Write the LD_TLB_REG register with the Ld_tlb_item bit set.
8) Enable the table walking logic (if necessary).

To read one local bus MMU TLB entry, use the following procedure:

1) Disable the table walking logic (if not already disabled).
2) Write the TLB entry number of the TLB entry to be read into Current_victim: number must be equal to or greater than the current value of Base_value and must be less than or equal to 31.
3) Write the LD_TLB_REG register with the Rd_tlb_item bit set.
4) Read the value from LB_MMU_READ_CAM_H.
5) Read the value from LB_MMU_READ_CAM_L.
6) Read the value from LB_MMU_READ_RAM_H.
7) Read the value from LB_MMU_READ_RAM_L.
8) Enable the table walking logic (if necessary).

This register controls reading and writing to the local bus MMU TLB.

Table 14–58. Local Bus MMU TLB Read/Write Register

Bit	Name	Function	Access		Hardware Reset Value
			User	Sup	
15–2	Reserved	Reserved	–	–	–
1	Rd_tlb_item	When this bit is set, causes the TLB data pointed to by the lock counter to be read. Writing a 0 has no effect. Always reads as 0.	R/W	R/W	0
0	Ld_tlb_item	When this bit is set, causes the TLB to be written. Writing a 0 has no effect. Always reads as 0	R/W	R/W	0

The LB MMU content addressable memory (CAM) access registers can be used to access data in the local bus MMU content addressable memory. The VA_tag values are compared against the local bus virtual address when a local bus access occurs. Bits 31-28 of the local bus virtual address are ignored in the address comparison.

Note:

The USB host controller can malfunction if the local bus MMU is programmed with tiny pages. See Section 14.8.2.1, *Local Bus MMU Page Size and the USB Host Controller,* for more information.

Table 14–59. Local Bus MMU CAM High Register

Bit	Name	Function	Access		Hardware Reset Value
			User	Sup	
15–6	Reserved	Reserved	–	–	–
5–0	VA_tag_l1_H	The most significant six bits of the local bus virtual address, which are used for this entry level 1 table index. Because the local bus MMU ignores bits 31-28 of the local bus virtual address, these 6 bits correspond to bits 27-22 of the 32-bit local bus virtual address used by the USB host controller.	R/W	R/W	0

Table 14–60. Local Bus MMU CAM Low Register

Bit	Name	Value	Function	Access		Hardware Reset Value
				User	Sup	
15–14	VA_tag_l1_L		Least significant two bits of the local bus virtual address to be used for this entry level 1 table index. Because the local bus MMU ignores bits 31-28 of the local bus virtual address, VA_tag_l1_L correspond to bits 21-20 of the 32-bit local bus virtual address used by the USB host controller.	R/W	R/W	0
13–4	VA_tag_l2		Bits of the local bus virtual address to be used for the level 2 table index lookup (depending on page size). For tiny pages, bits 13:4 of this register are compared against local bus virtual address bits 19-10. For small pages, bits 13:6 are compared against local bus virtual address bits 19-12. For large pages, bits 13:10 are compared against local bus virtual address bits 19-16. For sections, this field is ignored.	R/W	R/W	0
3	P		Preserved bit. When 1, CAM entry is preserved. When 0, CAM entry is not preserved.	R/W	R/W	0
2	V		Valid bit.	R	R	0
		0	CAM entry is not valid			
		1	CAM entry is valid			
1–0	SLST		Page size associated with CAM entry:	R/W	R/W	0
		00	Section (1MB)			
		01	Large page (64KB)			
		10	Small page (4KB)			
		11	Tiny page (1KB)			

The LB MMU RAM registers provide information on the physical address associated with a CAM entry that defines a page of memory in the local bus virtual address space.

Table 14–61. Local Bus MMU RAM High Register

Bit	Name	Function	Access		Hardware Reset Value
			User	Sup	
15–0	Ram_MSB	Most significant 16 bits of the physical address that corresponds to a local bus virtual address	R/W	R/W	0

Table 14–62. Local Bus MMU RAM Low Register

Bit	Name	Value	Function	Access		Hardware Reset Value
				User	Sup	
15–10	Ram_LSB		Least significant six bits of the physical address that corresponds to a local bus virtual address	R/W	R/W	0
9–8	AP		Access permission bits	R/W	R/W	0
		00	No access. Any local bus access to this page causes a permission fault.			
		01	No access. Any local bus access to this page causes a permission fault.			
		10	Read access only. Any local bus write access to this page causes a permission fault			
		11	Full access. Any local bus access to this page can complete without a permission fault.			
7–0	Unused					

The LB MMU global flush register allows flushing of all nonprotected TLB entries.

Note:

Flushing the whole TLB does not change the base_value or the victim_counter fields of the LB_MMU_LOCK_REG register.

Table 14–63. Local Bus MMU Global Flush Register

Bit	Name	Function	Access		Hardware Reset Value
			User	Sup	
15–1	Reserved	Reserved	–	–	–
0	Global_flush	When written with a 1, all nonprotected TLB entries are flushed. Has no effect when written with a 0. Always returns 0 on read.	R/W	R/W	0

The LB MMU entry flush register allows flushing of individual local bus MMU TLB entries.

Table 14–64. Local Bus MMU Entry Flush Register

Bit	Name	Function	Access		Hardware Reset Value
			User	Sup	
15–1	Reserved	Reserved	–	–	–
0	flush_entry	When written with a 1, flushes the TLB entry pointed to by the virtual address in LB_MMU_CAM_H_REG and LB_MMU_CAM_L_REG, even if the entry is set as a protected entry. Has no effect when written with a 0. Always reads as 0.	R/W	R/W	0

The LB MMU CAM read registers allow reading of local bus MMU CAM entries. See the LB_MMU_LD_TLB_REG description.

Table 14–65. Local Bus MMU CAM Read High Register

Bit	Name	Value	Function	Access		Hardware Reset Value
				User	Sup	
15–6	Reserved		Reserved	–	–	–
5–0	VA_tag_l1_H		Most significant six bits of the local bus virtual address to be used for this entry level 1 table index	R/W	R/W	0
15–14	VA_tag_l1_L		Least significant two bits of the local bus virtual address which are used for this entry level 1 table index	R/W	R/W	0
13–4	VA_tag_l2		Bits of the local bus virtual address which may be used for the level 2 table index lookup (depending on page size).	R/W	R/W	0
			For tiny pages, bits 13:4 are used.			
			For small pages, bits 13:6 are used.			
			For large pages, bits 13:10 are used.			
			For sections, this field is ignored.			
3	P		Preserved bit. When 1, CAM entry is preserved. When 0, CAM entry is not preserved.	R/W	R/W	0
2	V		Valid bit. When 1, CAM entry is valid. When 0, CAM entry is not valid	R	R	0
1–0	SLST		Page size associated with CAM entry:	R/W	R/W	0
		00	Section (1MB)			
		01	Large page (64KB)			
		10	Small page (4KB)			
		11	Tiny page (1KB)			

The LB MMU RAM read registers are used to read the local bus MMU TLB RAM. See the LB_MMU_LD_TLB_REG description.

Table 14–66. Local Bus MMU RAM Read High Register

Bit	Name	Function	Access		Hardware Reset Value
			User	Sup	
15–0	Ram_MSB	Most significant 16 bits of the physical address that corresponds to a local bus virtual address	R/W	R/W	0

Table 14–67. Local Bus MMU RAM Read Low Register

Bit	Name		Function	Access		Hardware Reset Value
				User	Sup	
15–10	Ram_LSB		Least significant six bits of the physical address that corresponds to a local bus virtual address	R/W	R/W	0
9–8	AP		Access permission bits	R/W	R/W	0
		00	No access. Any local bus access to this page causes a permission fault.			
		01	No access. Any local bus access to this page causes a permission fault.			
		10	Read access only. Any local bus write access to this page causes a permission fault.			
		11	Full access. Any local bus access to this page can complete without a permission fault.			
7–0	Unused					

14.8.2 Local Bus MMU Programming for USB Host Controller Operation

14.8.2.1 Local Bus MMU Page Size and the USB Host Controller

The OHCI USB host controller assumes that page size is no smaller than 4K bytes. The OHCI USB host controller allows a transfer descriptor data buffer to split between two non-contiguous 4K byte pages in local bus virtual address space. If a transfer descriptor current buffer pointer is in a different 4K-byte page of local bus virtual address space than the transfer descriptor buffer end, the OHCI USB host controller reads from or writes to that buffer starting at the current buffer pointer and increasing until it gets to the last byte of the 4K aligned block of local bus virtual address space. It then switches to a local bus virtual address, which has the same 20 upper bits as the transfer descriptor buffer end register, with 12 lower bits at 0.

Because of this behavior, it is strongly preferred that the mapping of local bus virtual addresses to processor physical addresses be done via 4K-byte pages (or larger). If tiny pages (1K byte per page) are used, they must be used in groups of four tiny pages that are contiguous both in local bus virtual address space and in processor physical address space, with the first of the tiny pages aligned on a 4K-byte boundary in both processor physical address space and local bus virtual address space. When using local bus MMU tiny pages, failure to allocate and align groups of four tiny pages as described here results in malfunction of the USB host controller memory access operations.

14.8.2.2 Local Bus MMU and Page Protection

The access protection features of the local bus MMU are similar to the protection mechanisms provided by the MPU MMU. If a local bus access attempts to access a local bus virtual address within a local bus MMU page that is marked as protected or attempts to write to a page that is marked as read-only, an interrupt is issued to the MPU level 1 interrupt handler.

14.8.2.3 Local Bus MMU Page Miss

A local bus access that has an address within the valid local bus address range but does not have a corresponding page mapping in the local bus MMU causes the local bus MMU to issue an interrupt to the MPU level 1 interrupt handler.

The MPU can make use of the local bus MMU page miss interrupt to allow software-based conversion between local bus virtual addresses and physical addresses. If using such a software mechanism, be aware that the software overhead must be considered in relation to the local bus time-out counter. If the software cannot provide the converted physical address to the local bus MMU quickly enough for the USB host controller local bus access to complete

before the local bus time-out counter completes its count, the unrecoverable error OHCI interrupt is signaled and the USB host controller stops performing USB packet transactions. It is possible to disable the USB host controller local bus time-out counter, but then there is no protection against the local bus lock-up that occurs if the USB host controller attempts to access a local bus virtual address that is outside of the valid local bus virtual address range of 3000:0000h to 3FFF:FFFFh.

14.9 USB Host Controller Reset and Clock Control

14.9.1 USB Host Controller Clock Control

The OMAP5910 clock generation and system reset management module (ULPD) provides a 48-MHz clock to the USB host controller. This clock can be stopped by software to reduce USB host controller power consumption when USB host controller operation is not needed.

Clocking for the local bus is controlled by a different mechanism. When the USB host controller needs to access system memory, the local bus must be operating.

14.9.2 Initializing ULPD to Generate the 48-MHz Clock

The ULPD module generates 48 MHz for the USB host controller using either a digital PLL (DPLL) or an analog PLL (APLL). The USB host controller receives a clock from the ULPD module when the CONF_MOD_USB_HOST_HHC_UHOST_EN_R bit is set. This register bit provides the clock request from the USB host controller to the ULPD.

The ULPD resets to a mode where the 48-MHz clock is generated by the DPLL. This sequence can be used to disable the DPLL and enable the APLL:

1) Clear the ULPD DPLL enable bit.
2) Wait until the DPLL lock bit goes to 0 to indicate that the DPLL is no longer locked (ULPDs DPLL_CTRL_REG register LOCK bit).
3) Set the APLL enable bit (ULPDs APLL_CTRL_REG register, APLL_NDPLL_SWITCH bit). (When this bit is 0, the DPLL output is selected rather than the APLL output).
4) Wait until the APLL global lock bit is 1 (ULPDs register, GLOBAL_LOCK bit).

When the ULPD module selects the APLL, the DPLL is shut off and all OMAP5910 modules that use 48 MHz receive 48 MHz from the APLL. When the ULPD module selects the DPLL, the APLL is shut off and all OMAP5910 modules that use 48 MHz receive 48 MHz from the DPLL.

14.9.3 USB Host Controller Hardware Reset

Reset of the USB host controller is provided by the ULPD module. The PER_EN bit in the MPU reset control 2 register controls the reset to many OMAP5910 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 OMAP5910 ULPD module) be active and that the OMAP5910 configuration register CONF_MOD_USB_HOST_HHC_UHOST_EN_R bit be set 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 PER_EN bit set, CONF_MOD_USB_HOST_HHC_ UHOST_EN_R bit 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 the correct, reset the values and continue reading until the proper reset values are seen.

The CONF_MOD_USB_HOST_HHC_UHOST_EN_R 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.

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 the MOD_CONF_CTRL_0 register CONF_MOD_USB_HOST_HHC _UHOST_EN_R bit is set. It must then wait until reads of both the HcRevision register and the HcHCCA register return their correct reset default values.

14.9.4 USB Host Controller OHCI Reset

The *OHCI Specification for USB* provides the HCR bit in the HCCommand Status register, which resets the OHCI controller. This reset may be used to reset the OHCI functionality and has no effect on the USB host controller local bus and MPU public peripheral bus interfaces.

14.9.5 USB Host Controller Power Management

Power management of the OMAP5910 USB host controller is limited to disabling the clock to the USB host controller using the MOD_CONF_CTRL_0 register CONF_MOD_USB_HOST_HHC_UHOST_EN_R bit. When this bit is 0, the USB host controller clocks are disabled and the USB host controller is held in reset. The USB signal multiplexing controlled by CONF_MOD_USB_HOST_HMC_MODE_R is not affected, so a CONF_MOD_USB_HOST_HMC_MODE_R setting that multiplexes USB function controller and/or UART1 signals to OMAP5910 top-level multiplexing can still make use of the USB function controller and/or UART1.

When the OMAP5910 host controller's 48-MHz clock is disabled, all USB host controller OHCI registers and the HostUEAddr, HostUEStatus, HostTimeoutCtrl, and HostRevision registers are inaccessible.

14.9.6 Local Bus Clock

Proper operation of the USB host controller requires that the local bus clock be enabled. For details, see Section 14.7.11, *Local Bus Initialization*.

14.10 OMAP5910 USB Hardware Considerations

14.10.1 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.

14.10.2 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.

14.10.3 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 OMAP5910 device provides the input pin USB_VBUSI, which is provided to the OMAP5910 USB function controller. This input is a CMOS input that is not rated for the full VBUS range specified by the USB specification. An external signal level converter is required to convert the VBUS signal range to a range suitable for the USB_VBUSI pin.

14.10.4 USB D+ Pullup Enable for USB Function Controller

When using a USB signal multiplexing mode that provides USB function controller signals to OMAP5910 pins, the OMAP5910 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 should be active and is driven high when the pullup should be 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 2, the USB.PUEN pin is driven high when the pullup should be active and is driven low when the pullup should be inactive. In this mode of operation, the pullup resistor may be connected directly between the OMAP5910 USB.PUEN and the D+ signal.

When the USB.PUEN pin is set for top-level multiplexing configuration 3, the USB.PUEN pin is driven high when the pullup should be active and is placed in high impedance mode the pullup should be inactive. In this mode of operation, the pullup resistor can be connected directly between the OMAP5910 USB.PUEN and the D+ signal.

14.10.5 Port Passthrough Mode

When MOD_CONF_CTRL_0 register CONF_MOD_USB_HOST_HMC_MODE_R is 7 (with appropriate top level signal multiplexing settings), the signals from six OMAP5910 input pins are passed to six OMAP5910 output pins. This mode is described in Table 14–68.

Table 14–68. CONF_MOD_USB_HOST_HMC_MODE_R=7 Internal Connectivity

HMC_MODE 7		
Input Pin Name		**Output Pin Name**
MCSI1.DIN/USB2.VP	is logically connected to	RST_HOST_OUT/USB1_SE0
UART2.CTS/USB2.RCV	is logically connected to	MCBSP.CLK/USB1_TXEN
UART2.RX/USB2.VM	is logically connected to	MCSI1.DOUT/USB1.TXD
MCSI1.SYNC/USB1.VP	is logically connected to	UART2.RTS/USB2_SE0
MCSI1.DIN/USB1.RCV	is logically connected to	MCSI2.DOUT/USB2.TXEN
MCSI1.CLK/USB1.VM	is logically connected to	UART2.TX/USB2.TXD

14.10.6 UART1 Connectivity when CONF_MOD_USB_HOST_HMC_MODE_R = 2, 10, 18, and 24

The USB signal multiplexing provides modes that allow UART1 signals to be brought to OMAP5910 pins. This multiplexing is separate from UART1 multiplexing that is provided by OMAP5910 top-level pin multiplexing. When CONF_MOD_USB_HOST_HMC_MODE_R is set to 2, 10, 18, or 24, three signals to/from UART1 are provided to OMAP5910 pins when the OMAP5910 top-level pin multiplexing for those pins selects the USB signal multiplexer. These signal assignments are described in Table 14–69.

Table 14–69. CONF_MOD_USB_HOST_HMC_MODE_R = 2, 10, 18, and 24 UART Signal Assignments

OMAP5910 Pin Name	Signal Name	Direction
MCSI1.DOUT/USB1.TXD	UART1_TX	Output
MCSI1.BLK/USB1.VM	UART1_RX	Input
MCSI1.DIN/USB1.RCV	UART1_CTS	Input

14.10.7 MPU_BOOT Signal Sharing

The OMAP5910 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 may 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.

14.10.8 USB D+, D– Pulldown for USB Function Controller

System implementations which use the OMAP5910 USB function controller and are sensitive to supply current requirements may wish to 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 resistor depends on transceiver characteristics, and board layout and must be designed to meet USB D+ and D– signal requirements.

Chapter 15

Clock Generation and System Reset Management

This chapter describes clock generation and system reset for the OMAP5910 multimedia processor.

Topic **Page**

15.1 Introduction

The clock generator and reset management module supplies clocks and resets to the entire OMAP5910 device.

Figure 15–1 shows the OMAP5910 with the clock generator and reset management area highlighted. Figure 15–2 shows the OMAP5910 clock scheme.

15.1.1 Clock Generation and System Reset Control

In the OMAP5910 device, clock generation and system reset are controlled by several modules, as shown in Figure 15–3.

There are three major components of this circuitry: the ultralow-power device (ULPD), the reset management module, and the clock generation and management module.

Figure 15–1. OMAP5910 Device Clock and Reset Management

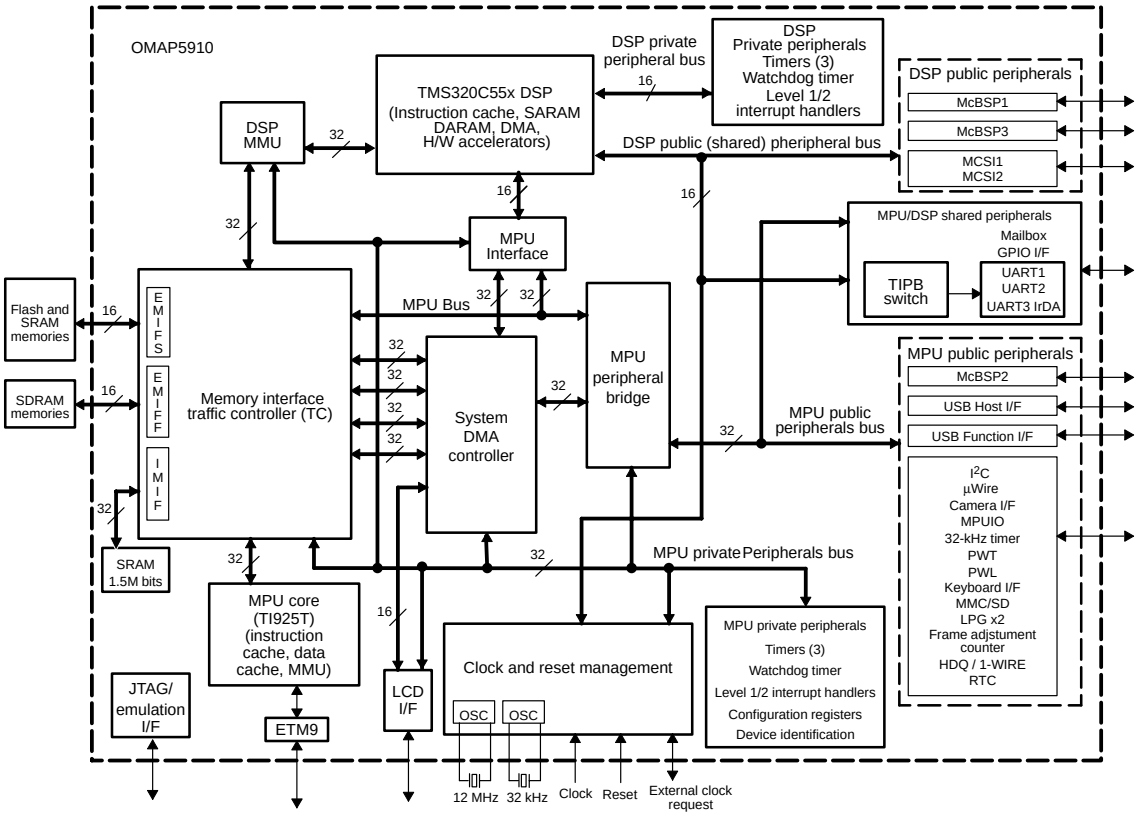


Figure 15–2. OMAP5910 Clock Scheme

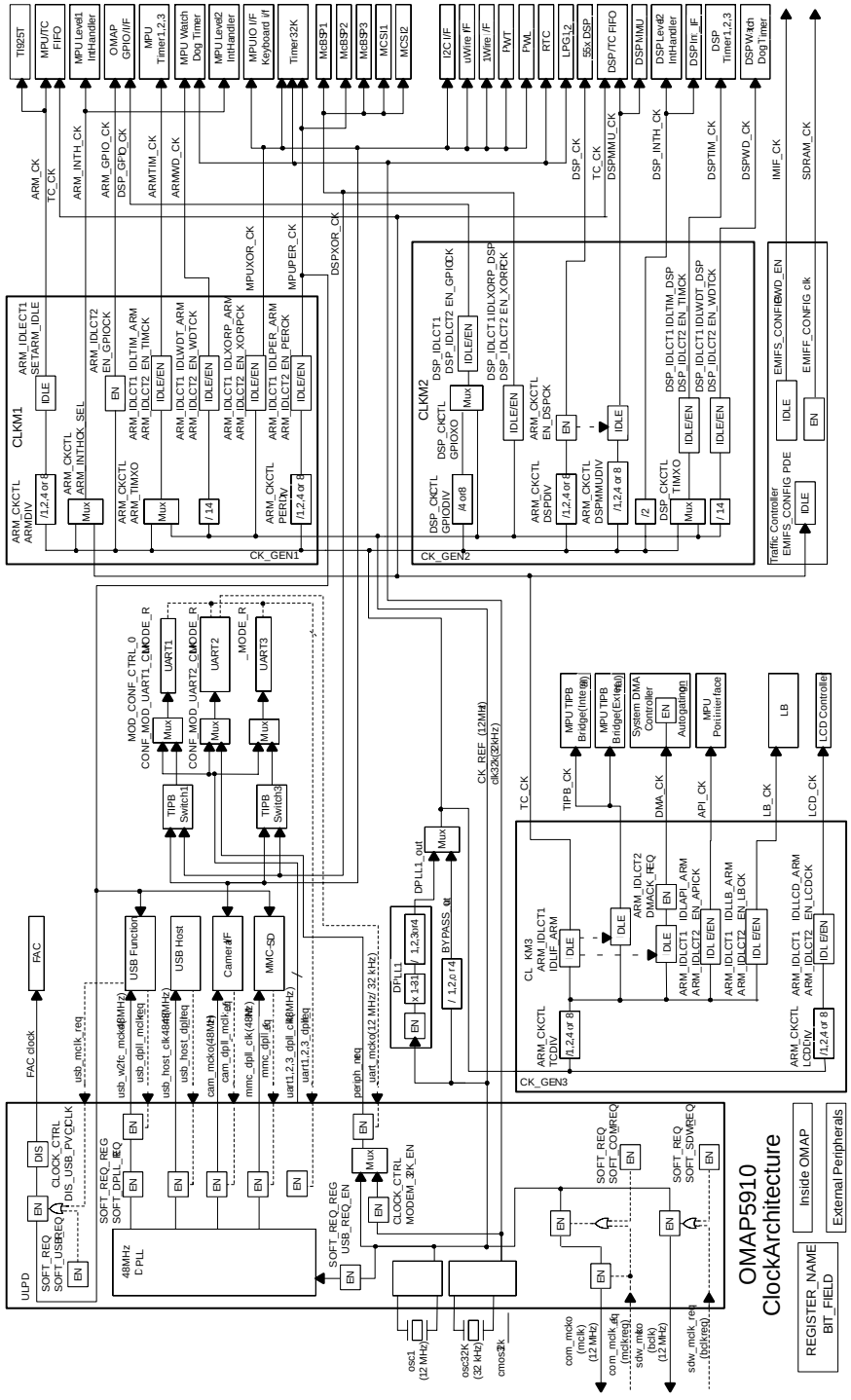
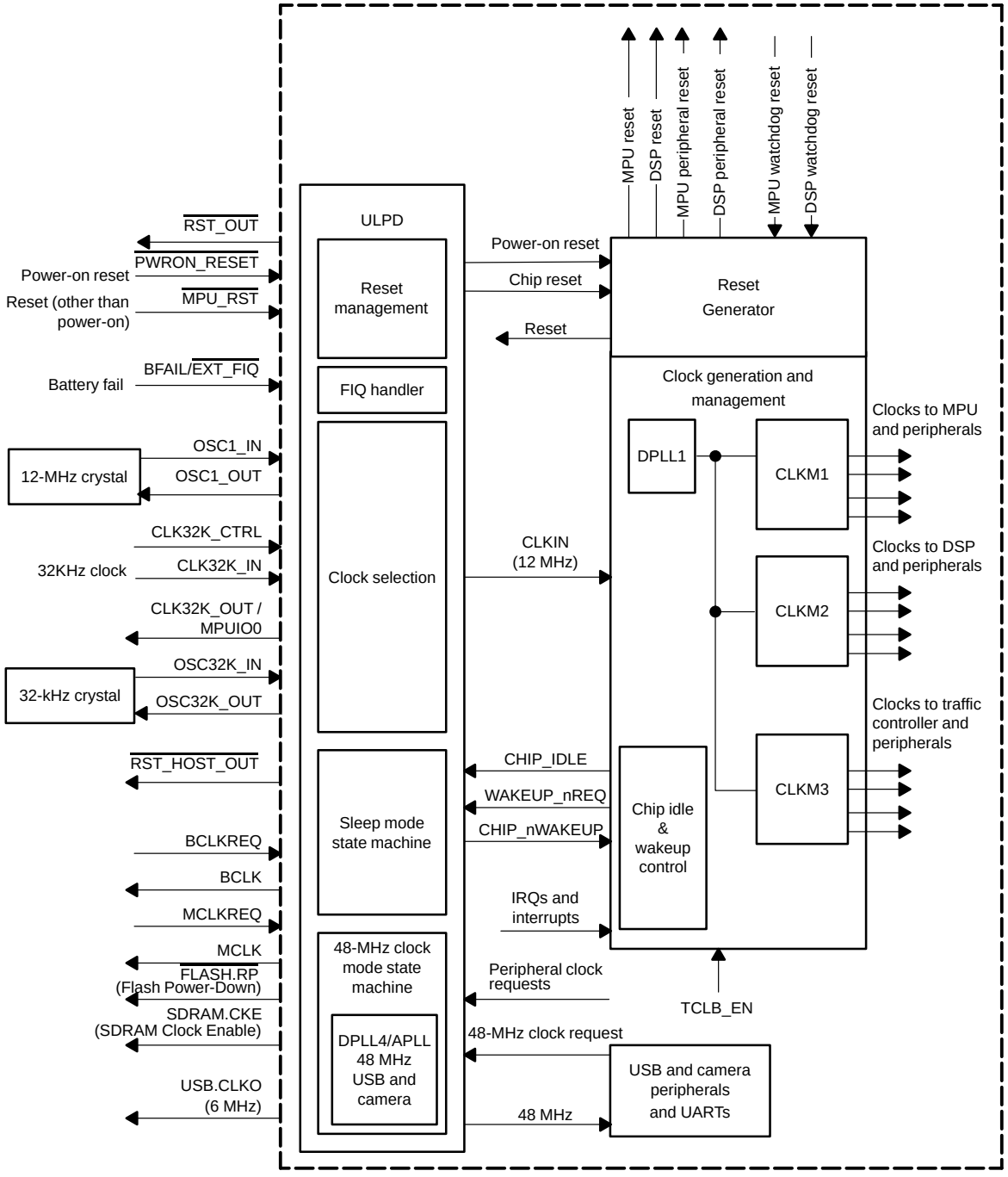


Figure 15–3. Modules Controlling Clock and Reset Management



15.1.1.1 ULPD Module

The ULPD module is an embedded peripheral controlled by the internal MPU with the following functions:

- Performs the state transition of the different power modes:
  - Awake: 32-kHz and 12-MHz clocks are on, and 12 MHz is fed into the clock generation module (those PLLs can be turned on or off).
  - Big sleep mode: 32-kHz and 12-MHz clocks are on, 12 MHz is not fed into the clock generation module, and PLLs are off.
  - Deep sleep mode: 32-kHz clock is on, and 12-MHz clock and PLLs are off.
- Performs the power-on reset of the chip
- Calibrates an external quartz based oscillator (32 kHz)
- Performs the wake up of the 12-MHz OSC1 oscillator to provide the OSC1 clock to an external device. An external clock request or peripheral wake-up request turns on the OSC1 oscillator but is not treated as an interrupt.
- Performs a 12-MHz/32-kHz switch for peripherals that need to switch to 32 kHz
- Generates the functional reset signal used by the reset module
- Manages the 48-MHz DPLL and APLL on/off
- Processes battery-failed signal to generate external shutdown signal ($\overline{\text{RST\_HOST\_OUT}}$)

15.1.1.2 Reset Module

The reset module has the following functions:

- Provides internal global reset and software reset
- Performs reset control for peripheral bus peripherals
- Monitors internal and external reset (for example, watchdog timer time-out)
- Monitors system and reset status

15.1.1.3 Clock Generation and Management Module

The clock generation and management module provides the following features:

- Programmable clocking scheme (synchronous and synchronous scalable modes) and power-up defaults to fully synchronous mode
- Setup and configuration controlled by both the DSP and MPU processors
- A single-reference clock input to one DPLL from an external source (CLKIN). The PLL modes are configurable: lock, bypass, and idle.
- Programmable clocks, from CLKM1, with clock and idle control capability to the MPU and its subsystem:
  - GPIO
  - Timers
  - Other peripherals
- Programmable clock, from CLKM2, with clock and idle control capability to the DSP and its subsystem:
  - GPIO
  - Timers
  - Other peripherals
- Programmable clock, from CLKM3, with clock and idle control capability to the memory interface traffic controller (TC), including the following modules:
  - MPU interface (MPUI)
  - System DMA controller
  - LCD controller
  - Local bus
  - MPU peripheral bridges
  - Two internal MPU TI peripheral bridges to minimize access latency
- Programmable power saving and idle mode controls for the MPU, the DSP, the TC, and their respective subdomains
- Low-frequency clocks (reference clock/14) to supply watchdog timers for the DSP and MPU
- DMA clock request mechanism (provides DMA clock during data transfer only)
- Power control for external device reset/power on (flash)
- Idle sequence controls (MPU clock domain, DSP clock domain, and TC clock domain)
- Programmable idle modes (MPU and DSP) for different applications

- ❏ Wake up initiated by interrupts (MPU, DSP, and TCLB_EN pin) or DMA requests (TC and peripheral bus) in the idle mode
- ❏ Unmasked interrupt events enabled to wake up the device during idle modes

15.1.1.4 Memory-Mapped Registers

The application program controls the clock generation, reset, and power-saving modes via a set of memory-mapped registers (nine 16-bit registers for MPU subsystem and seven 16-bit registers for the DSP subsystem). These registers are accessible by the MPU or the DSP processors.

The MPU is the master of the OMAP5910 device at all times, and it controls the activities in the MPU, DSP, and TC domains.

The DSP controls DSP peripheral activities.

15.1.1.5 Clock Domains

The OMAP5910 is partitioned into three clock domains, each with its own clock manager:

- ❏ MPU domain (CLKM1)
- ❏ DSP domain (CLKM2)
- ❏ TC domain (CLKM3)

Clock domains use a common DPLL to provide a synchronous clock. The different clocking configurations are discussed in detail later in this chapter.

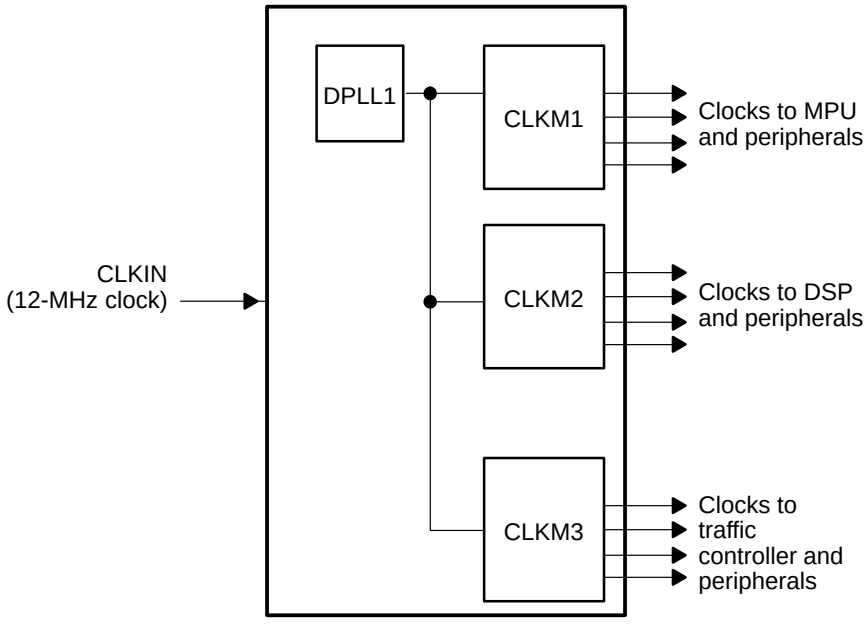
The external clock source (OSC1_IN) frequency must be 12 MHz to ensure proper operation for the USB.

15.2 Clock Generation

Figure 15–4 shows the basic building blocks of the clock generators and system reset module. This module consists of:

- One DPLL—frequency synthesizers (frequency lock but not phase lock)
- Control register file (CLKREG)—clock generator, system reset, idle, and wake-up controls
- Three CLKMs—clock generation and wake-up controls

Figure 15–4. Clock Generation and System Reset Module



15.2.1 Clocking Schemes

The clock generator supports two clocking schemes to provide performance flexibility and power-saving capabilities to the system. The clocking schemes are programmable. Power-up mode defaults to fully synchronous mode. Table 15–1 shows the clocking scheme selection, and Table 15–2 shows the CLKM source selection.

Table 15–1. Clocking Scheme Selection

Clock_Select	Clocking Scheme	Remarks
000	Full synchronous	Default, bypass FIFO logic. TC=DSPMMU=MPU, DSP = 1x or 2x of DSPMMU
001	Reserved	Do not use this setting
010	Synchronous scalable	Use FIFO logic between MPU and TC, DSP MMU and TC
Others	Reserved	Do not use this settings

Note: In all the above cases, the frequency of the DSP can be 1x or 2x that of DSP MMU.

Table 15–2. CLKM Source Selection—Set via the MPU System Status Register

Clock Select	Operating Mode	CLKM1 Input Clock Source	CLKM2 Input Clock Source	CLKM3 Input Clock Source	Remarks
000	Fully synchronous	DPLL1/N	DPLL1/O	DPLL1/N	Notes 1, 2, 4
010	Synchronous scalable	DPLL1/M	DPLL1/N	DPLL1/O	Notes 3, 4

Notes: 1) If you select the fully synchronous mode, you must program the divide-down bits so that MPUDIV, DSPMMUDIV and TCDIV are all equal. Further, DSPMMUDIV must be 1x or 1/2x that of DSPDIV.

2) CLKGEN1 = CLKGEN3 = DPLL1/N, CLKGEN2 = CLKGEN1 or 2*CLKGEN1 = DPLL1/O

3) M, N =< O, and O is a multiple (1, 2, 4, 8) of M, N.

4) The DSP MMU cannot run above the maximum speed of TC.

15.2.2 Operating Modes

The OMAP5910 device supports the following operating modes:

- Fully synchronous

  MPU, DSP MMU, and TC run at the same clock period, and DSP MMU is 1x or 1/2x of DSP. For example, DPLL1 output can be 120 MHz; MPU, DSP MMU, and TC can be 60 MHz; and DSP can be 120 MHz.

  On power up, the OMAP5910 device is always in synchronous mode, where MPU, DSP MMU, TC, and DSP are all at the same speed.

- Synchronous scalable

  MPU, DSP MMU, and TC are synchronous, but MPU and DSP MMU are multiples (1x, 2x, 4x, or 8x) of TC. The DSP must be 1x or 2x of the DSP MMU. For example, DPLL1 clock can be 120 MHz, MPU can run at 120 MHz, DSP MMU can run at 60 MHz, TC can run at 30 MHz, and DSP can run at 60 MHz or 120 MHz. In this mode, the clock-feeding mechanism (to each respective domain) is similar to that of the fully synchronous mode, with the exception that the clocks are synchronous but are multiples of each other. The input clock is from DPLL1, and the clock is multiplied/ divided by the CLKM (1, 2, 3) as in the following example (assuming the output of DPLL1 is 120 MHz):

  - CLKM1 output: 120 MHz/2
  - CLKM2 output: 120 MHz/1
  - CLKM3 output: 120 MHz/4

  Divider circuitry is implemented in each CLKM.

Note:

In synchronous scalable mode, the traffic controller clock must have the same or a slower frequency as the MPU and the DSPMMU clock.

At reset, the fully synchronous mode is selected (default). After the OMAP5910 device is up and running, the application software can write to the control registers via the CLOCK_SELECT (2:0) bits in the MPU system status register (ARM_SYSST) to switch to a desired mode of operation. However, you should use the system software to save the context before switching modes. For information about the switching procedure, see Appendix B, *Switching Clock Modes*.

The DSP_MMU clock must obey all of the following rules:

1) TC clock frequency always must be equal to or less than DSP, DSPMMU, and MPU clocks.

2) DSP_MMU clock must be 1x or 1/2x of DSP clocks.

3) DSPMMU_DIV can be /1, /2, /4, /8, but TC_DIV and DSP_DIV must obey 1 and 2.

4) DSP_MMU clock frequency cannot be more than the maximum speed of the TC.

15.2.3 External Master Mode

This mode allows bypass of the 12-Mhz on-chip oscillator in systems where the 12-MHz clock is provided externally by a master device. The procedure for utilizing this mode is as follows:

1) During power-on reset, OMAP5910 is in deep sleep:

   - 12-MHz on-chip oscillator is disabled.
   - MCLKREQ pin is an input.

2) After power-on reset, OMAP5910 is awake.

   - 12-MHz oscillator is bypassed.
   - MCLKREQ pin is an input.
   - 12-Mhz clock is provided externally.

3) Switch to external master mode by setting (FUNC_MUX_CTRL_B(20:18) = 001.

   - The 12-MHz oscillator is bypassed (disabled).
   - MCLKREQ pin is now the EXT_MASTER_REQ pin, which drives to 1.
   - 12-Mhz clock is provided externally.

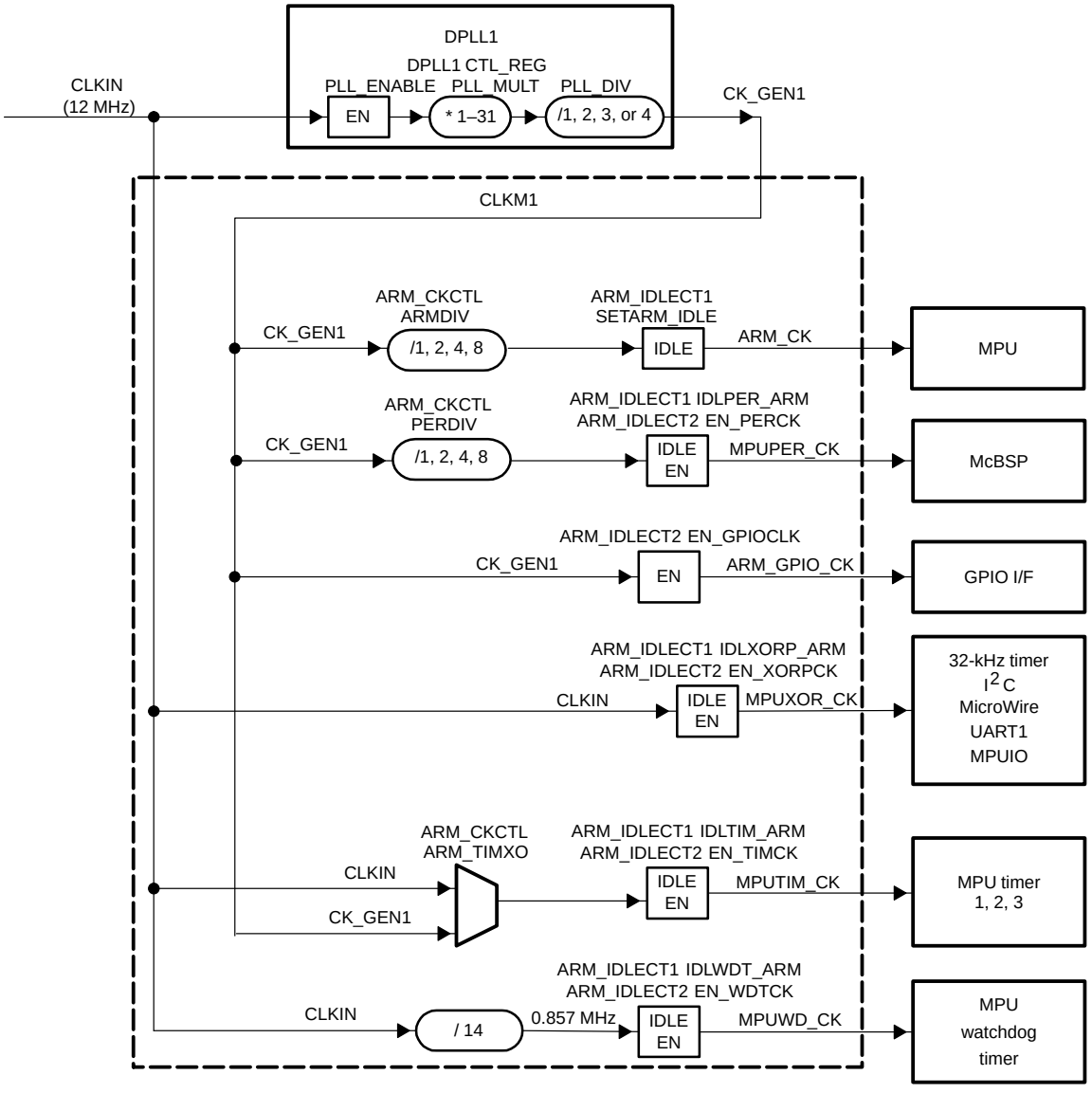
4) If OMAP switches into deep sleep:

   - EXT_MASTER_REQ drives to 0 to indicate that the external 12-MHz clock is not needed.
   - The 12-MHz clock can then switch off externally.

15.2.4 CLKM1

CLKM1 controls the clock distribution and idle modes of the MPU subsystem, plus associated private and public peripherals (see Figure 15–5).

Figure 15–5. MPU Clock Distribution



The MPU clock (see Figure 15–5) has the following domains (CLKM1):

- MPU processor clock: MPU_CK, which is CLK_GEN1 divided by 1, 2, 4, or 8, as programmed via the ARMDIV bits of the MPU clock control register (ARM_CKCTL). The idle mode of the MPU is controlled by the SETARM_IDLE bit of the MPU idle mode entry 1 register (ARM_ IDLECT1).
- MPU peripheral clocks:
  - MPUXOR_CK, which is derived from CK_REF
  - MPUPER_CK, which is CLK_GEN1 divided by 1, 2, 4 or 8, as programmed via the PERDIV bits of the MPU clock control register (ARM_CKCTL)

    The MPUPER_CLK clock is enabled by the EN_PERCK bit of the MPU idle mode entry 2 register (ARM_IDLECT2) and the MPUX-OR_CK clock by the EN_XORPCK bit.
- MPU watchdog timer clock (low frequency, derived from CK_REF/14): Called either CK_CLKIN14 or MPUWD_CK. This clock is enabled by the EN_WDTCK bit of the MPU idle mode entry 2 register (ARM_IDLECT2). The IDLE mode is controlled by the IDLWDT_ARM bit of the MPU idle mode entry 1 register (ARM_IDLECT1). The clock cannot be disabled or idled while in the watchdog mode.
- MPU internal timers clock: MPUTIM_CK, which is derived from either CK_GEN1 or CK_REF, as selected by the ARM_TIMXO bit of the MPU clock control register (ARM_CKCTL). The IDLE mode of the MPU timers is controlled by the IDLTIM_ARM bit of the MPU idle mode entry 1 register (ARM_IDLECT1) and is enabled by the EN_TIMCK bit of the MPU idle mode entry 2 register (ARM_IDLECT2).
- MPU GPIO, MPU_GPIO_CLK, which is equal to CK_GEN1. This clock is enabled by the EN_GPIOCK bit of the the MPU idle mode entry 2 register (ARM_IDLECT2).

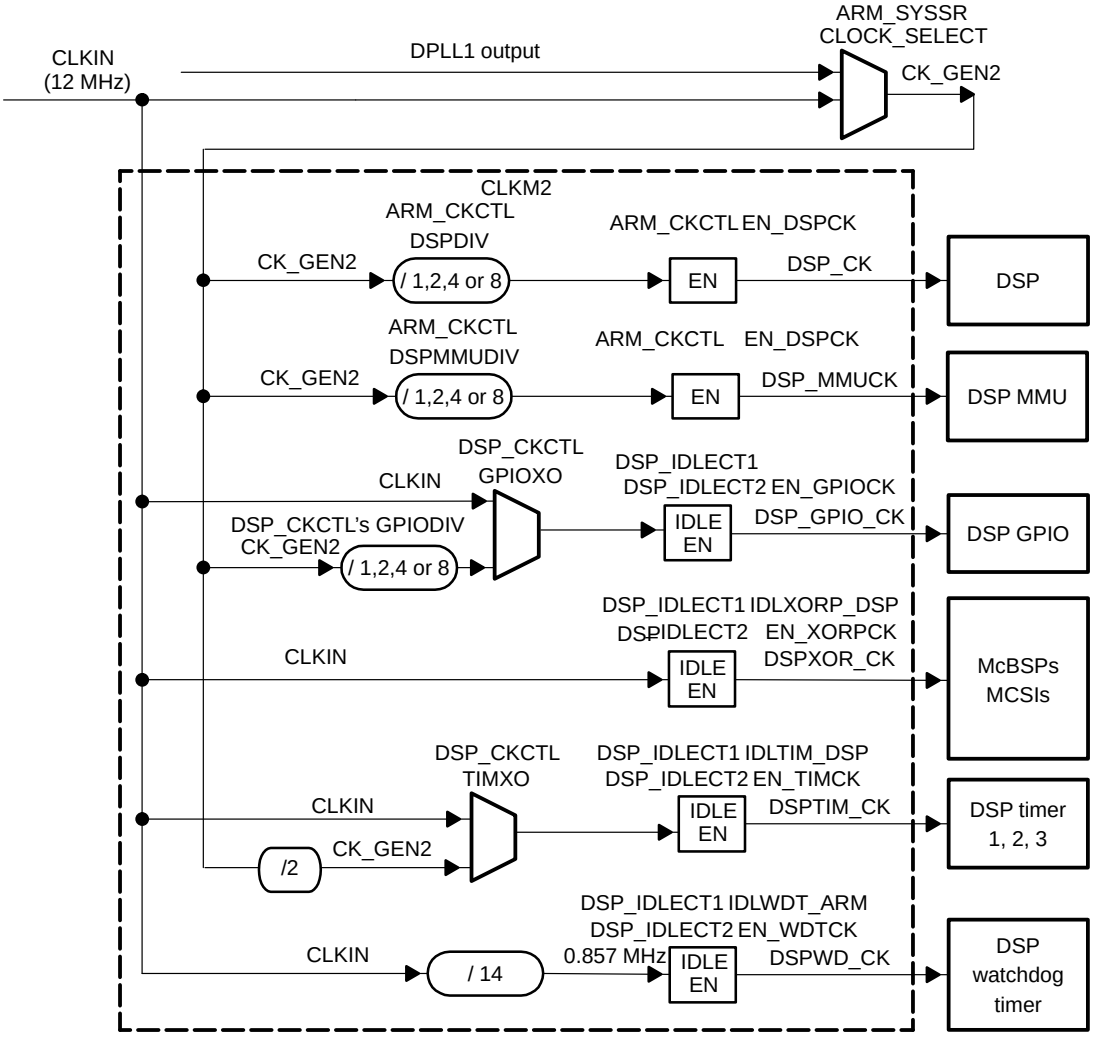
15.2.5 CLKM2

CLKM2 controls clock distribution and idle modes of the DSP subsystem plus associated private and shared peripherals. As shown in Figure 15–6, the CLKM2 circuitry provides separate clock signals for the DSP internal peripherals (GPIO, watchdog timer, and timers) and public peripherals.

Note:

CK_GEN2 represents the output of DPLL1 or CLK_REF, depending on clocking mode enabled.

Figure 15–6. DSP Clock Distribution



Clock signals for each clock domain of the DSP subsystem are as follows:

- MPU-controlled:
  - DSP processor clock: DSP_CK, which is CK_GEN2 divided by 1, 2, 4, or 8, as programmed via the DSPDIV bits of the MPU clock control register (ARM_CKCTL). The enabling of DSP_CK while the DSP is in the reset state is controlled by the EN_DSPCK bit of the MPU clock control register (ARM_CKCTL).
  - DSP MMU clock: DSPMMU_CK, as derived from CK_GEN2 divided by 1, 2, 4, 8, as programmed by the DSPMMUDIV bits of the MPU clock control register (ARM_CKCTL). Take care in selecting clocking schemes so as to not exceed the maximum frequency of the DSP MMU. See the OMAP5910 device datasheet for absolute timing limits.
- DSP-controlled:
  - DSP GPIO (DSP_GPIO_CK), as derived from CK_GEN2 divided by 1, 2, 4, 8 (as programmed by DSP_CKCTL GPIODIV) or CLKIN, as selected by the GPIOXO bit of the DSP clock control register (DSP_CKCTL). The clock is enabled by the EN_GPIOCLK bit of the DSP idle mode entry 2 register (DSP_IDLECT2). The IDLE mode is controlled by the IDLG–PIO_DSP bit of the DSP idle mode entry 1 register (DSP_IDLECT1).
  - DSP public peripherals (McBSPs, MCSIs): DSPXOR_CK, which is derived from CLKIN. The clock is enabled by the EN_XORPCK bit of the DSP idle mode entry 2 register (DSP_IDLECT2). The IDLE mode is controlled by the IDLXORP_DSP bit of the DSP idle mode entry 1 register (DSP_IDLECT1).
  - DSP internal timers: DSPTIM_CK is selected from either CK_GEN2/2 or CLKIN via the DSP_TIMXO bit of the DSP clock control register (DSP_CKCTL). The clock is enabled by the EN_TIMCLK bit of the DSP idle mode entry 2 register (DSP_IDLECT2). The IDLE mode is controlled by the IDLTIM_DSP bit of the DSP idle mode entry 1 register (DSP_IDLECT1).
  - DSP watchdog timers (low frequency, derived from CLKIN divided by 14): DSPWD_CK. The clock is enabled by the EN_WDCLK bit of the DSP idle mode entry 2 register (DSP_IDLECT2). The IDLE mode is controlled by the IDLWDT_DSP bit of the DSP idle mode entry 1 register (DSP_IDLECT1). The watchdog timer clock can only be disabled or idled when not in watchdog mode.

After reset, the highest frequency option (CK_GEN2 divided by 1) is selected for GPIO and DSPPER clocks. The software application program can alter these divisors at any time during operation by writing to the GPIODIV or DSP_PERDIV bits in the DSP clock control register (DSP_CKCTL).

The clock generator output (CK_GEN2) delivers a 50% duty cycle to the DSP subsystem clock distribution module (CLKM2). This module provides additional clock scaling, routing, and idle/reset control to the DSP to individual components in the DSP clock domain.

The CK_GEN2 clock works in conjunction with the idle and wake-up control logic to produce the DSP_CK clock signal that drives the DSP subsystem, the DSP MMU, and the DSP interrupt modules.

At reset, the CK_GEN2 is in the bypass mode, so it supplies (CLKM2) a clock of the same 12-MHz frequency as CLKIN. After the global reset period, the MPU application program can change the clock frequency through the CK_GEN2 control register.

DSP_CK is enabled at reset until the DSP is in reset state. The EN_DSPCK bit (located in the clock control register ARM_CKCTL) allows the MPU to turn off the DSP_CK while the DSP is held in a reset state.

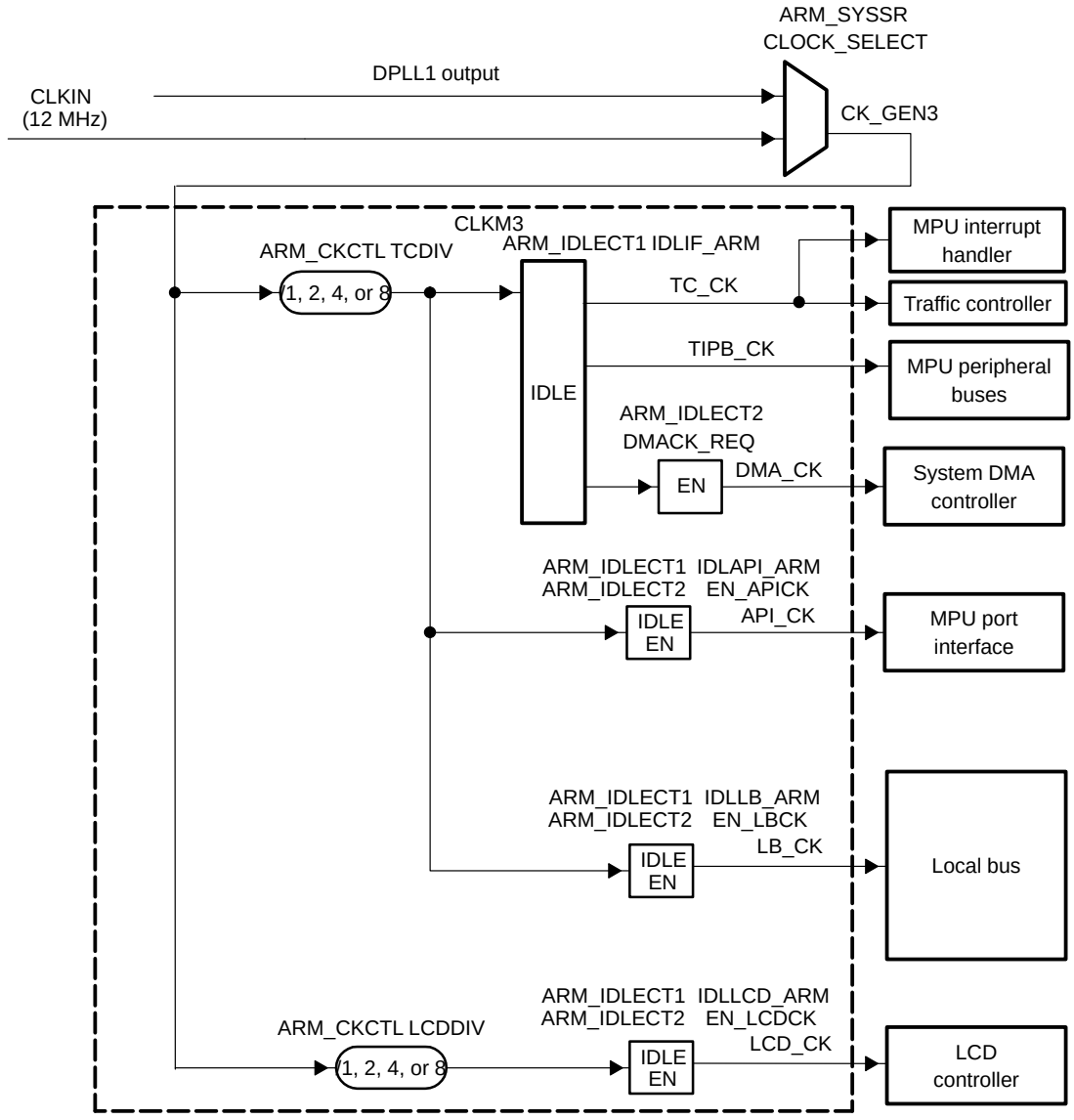
A free-running counter/divider receives the CK_GEN2 signal and makes available four taps where a 50% duty-cycle clock and the clock divided by 1, 2, 4, and 8 can be selected. A multiplexer set through the DSPDIV bits in the clock control register (ARM_CKCTL) selects the clock frequency that applies to the DSP clock domain.

At reset, the higher frequency (divide by 1) is selected. The software application program (accessing the control register file) can change the divider ratio by writing to the DSPDIV [1–0] bits at any time while the OMAP5910 device is running. A synchronization mechanism is implemented to remove any spikes while the clock frequency is changing (disable the clock first, change the DSPDIV bits, and then enable the clock).

15.2.6 CLKM3

CLKM3 controls clock distribution and idle modes of the traffic controller and various system-level clock domains. The traffic controller clock, CLKM3 (see Figure 15–7), has the following domains.

Figure 15–7. Traffic Controller Clock Distribution



Many of the following clocks are the same as the traffic controller clock (TC_CK) in terms of their frequencies, but not their IDLE controls. Each of the clocks has separate IDLE control logic.

- Traffic controller clock, TC_CK, is derived from CK_GEN3 divided by 1, 2, 4, or 8, as programmed via the TCDIV bits of the MPU clock control register (ARM_CKCTL). The MPU interrupt handler uses the TC_CK clock, as set by the ARM_INTHCK_SEL bit of the MPU clock control register (ARM_CKCTL). The IDLE mode is controlled by the IDLIF_ARM bit of the MPU idle mode entry 1 register (ARM_IDLECT1).

- Local bus and local bus MMU clock, LB_CK, is the same as TC_CK. The IDLE mode is controlled by the IDLLB_ARM bit of the MPU idle mode entry 1 register (ARM_IDLECT1). The LB_CK is enabled by the EN_LBCK bit of the MPU idle mode entry 2 register (ARM_IDLECT2).

- MPU port interface (MPUI) clock is dependent not only on the IDLAPI_ARM bit of the MPU idle mode entry 1 register (ARM_IDLECT1), but also on DSP_IDLE.

- The system DMA controller clock, DMA_CK, is the same as TC_CK. The IDLE mode is controlled by the IDLIF_ARM bit of the MPU idle mode entry 1 register (ARM_IDLECT1), and the clock is enabled by the DMACK_REQ bit of the MPU idle mode entry 2 register (ARM_IDLECT2).

- The MPU peripheral bridge clock, TIPB_CK, is the same as TC_CK. The IDLE mode is controlled by the IDLIF_ARM bit of the MPU idle mode entry 1 register (ARM_IDLECT1).

- LCD controller clock, LCD_CK, is derived from CK_GEN3 divided by 1, 2, 4, or 8, as programmed via the LCDDIV bit of the MPU clock control register (ARM_CKCTL). This clock is enabled by the EN_LCDCK bit of the MPU idle mode entry 2 register (ARM_IDLECT2). The IDLE mode is controlled by the IDLLCD_ARM bit of the MPU idle mode entry 1 register (ARM_IDLECT1).

15.2.7 Clock Distribution and Synchronization

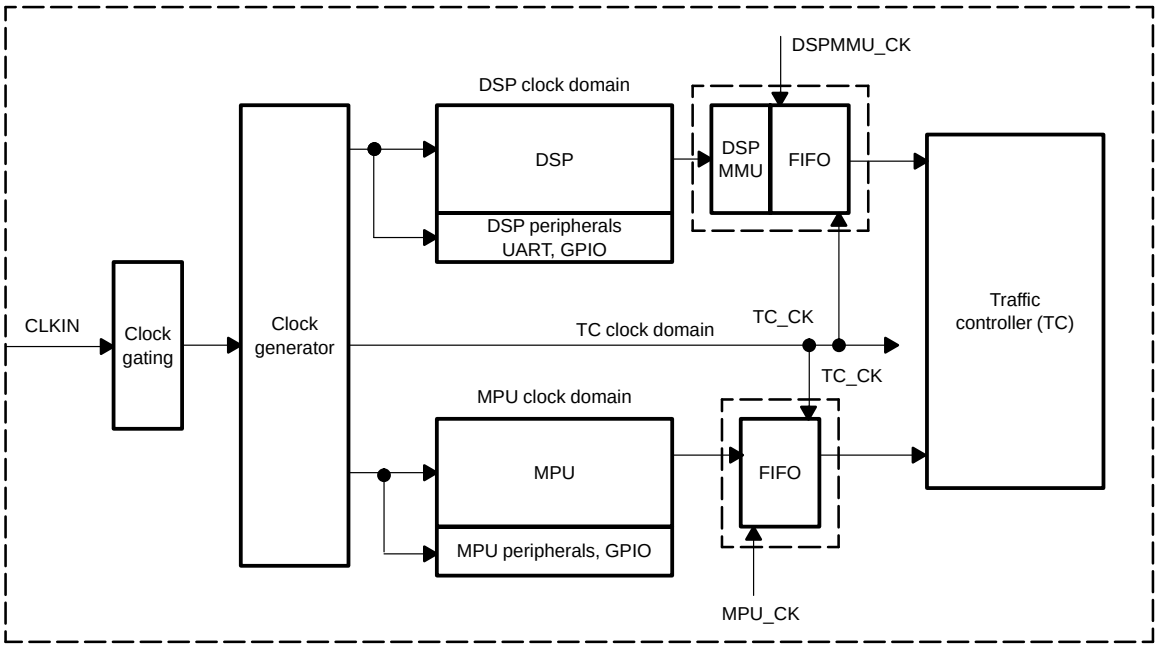
When any of the clock domains are running at different frequencies with respect to each other, FIFOs are used to buffer data being transferred between domains (see Figure 15–8). This is necessary to ensure that data being sent from a fast domain is buffered as a slow domain receives it. This buffering introduces latencies as data is passed between domains. Thus, this buffering can be bypassed if it is not needed (that is, when domains are running at the same speeds).

- For the fully synchronous clocking scheme, MPU_CK = DSPMMU_CK = TC_CK; the FIFO logic is bypassed between TC and MPU, TC and DSPMMU.
- For the synchronous scalable clocking scheme, FIFO logic is used for both processors.

Note:

TC_CK clock must be slower than or equal to the MPU_CK and DSP MMU clock speed.

Figure 15–8. OMAP5910 Clock Distribution and Synchronization

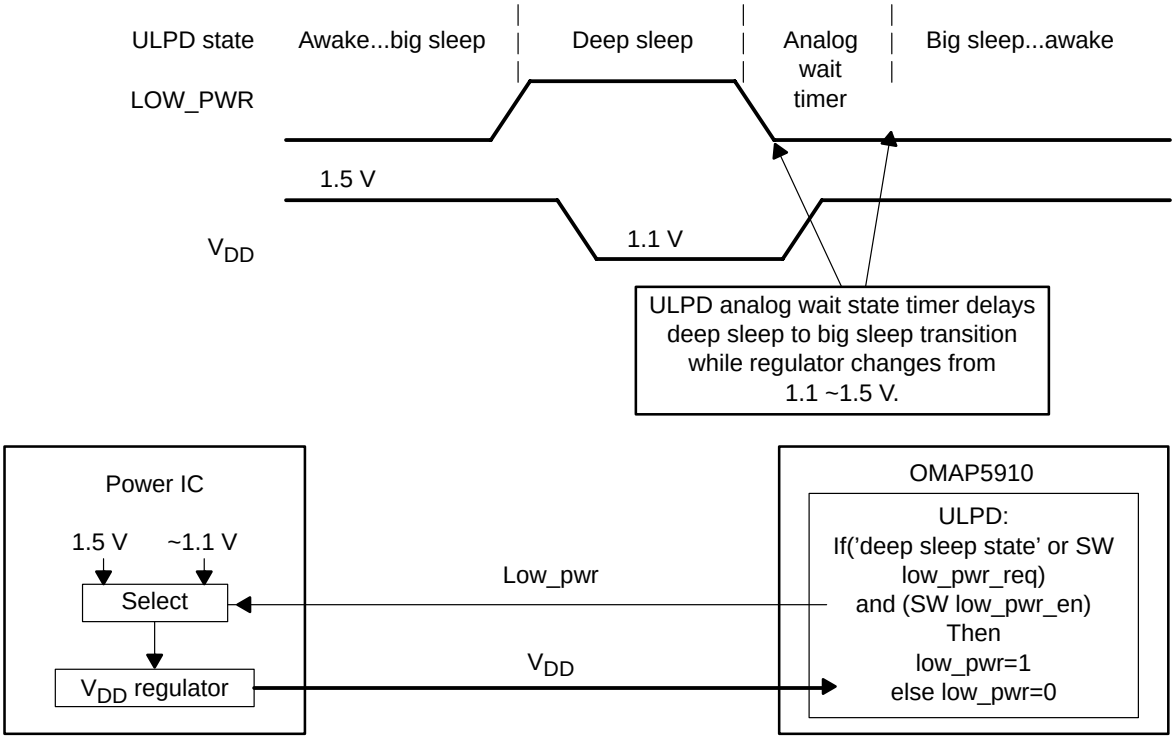


15.2.8 Low-Power Mode

The OMAP5910 LOW_PWR I/O is an active-high request for LOW_PWR mode (see Figure 15–9). It automatically requests external regulators to go to standby (low-power mode) or to lower the VDD core voltage during deep sleep mode. A software request, through bit 1 of the power control register (POWER_CTRL_REG), is available for debug and future support of the low voltage operational mode.

The LOW_PWR signal is multiplexed on the MPUIO5 ball. To get it on this ball, you must set bits (14:12) of the FUNC_MUX_CTRL_7 register to 001.

Figure 15–9. Low-Voltage Mode



15.3 Power Management

There are three clock domains in the OMAP5910 device. Each clock domain has its own clock management unit and can be put into idle mode (power saving mode) when unused without affecting the rest of OMAP5910 device functionality. In addition, the ULPD provides low-power modes that affect the entire device, not just the individual domains (see Figure 15–10).

A chip idle occurs when the DSP is idled, the MPU requests an idle, and the TC domain has no remaining transactions. Chip idle causes the ULPD to initiate big sleep mode. In big sleep mode, DPLL 1 is turned off, but the 12-MHz clock is still active. If the 12-MHz clock is not needed, then the ULPD can initiate a further transition to deep sleep mode, which turns the 12-MHz clock off internally.

When an unmasked interrupt event occurs, a request is performed by the wake-up control module. When receiving the request, if the device is in deep sleep mode, the ULPD brings the device out of deep sleep mode. Once the 12-MHz clock is stable, the ULPD brings the device into awake mode.

To reduce the wake-up time, a special hardware request is implemented in parallel to interrupts to wake-up the ULPD whenever an interrupt occurs. This request is generated by peripherals as USB or UART. When receiving the hardware request, the ULPD brings the device out of the deep sleep mode to wakeup the 12-MHz clock. When the clock is awake, the ULPD goes to big sleep and awakes if a request is received from wake-up control module. Figure 15–11 shows the wakeup control module.

Deep sleep mode is the entry state of the device when a power-on reset occurs. Such a reset acts as a wake-up request, causing the device to transition to the awake state.

The 12-MHz clock may be required to clock signals out of the device (such as the MCLK pin) or to ULPD DPLL (used for USB and other internal peripherals) in either the big sleep or awake states. The deep sleep state can not be entered if there is need for the 12-MHz clock.

The power management state machine runs at 32 kHz. All control signals of this state machine are resynchronized on the 32-kHz clock. The 32-kHz clock is always on.

Table 15–3 lists the peripherals and external signals which can wake up OMAP5910 from deep sleep.

Figure 15–10. Power Management State Machine

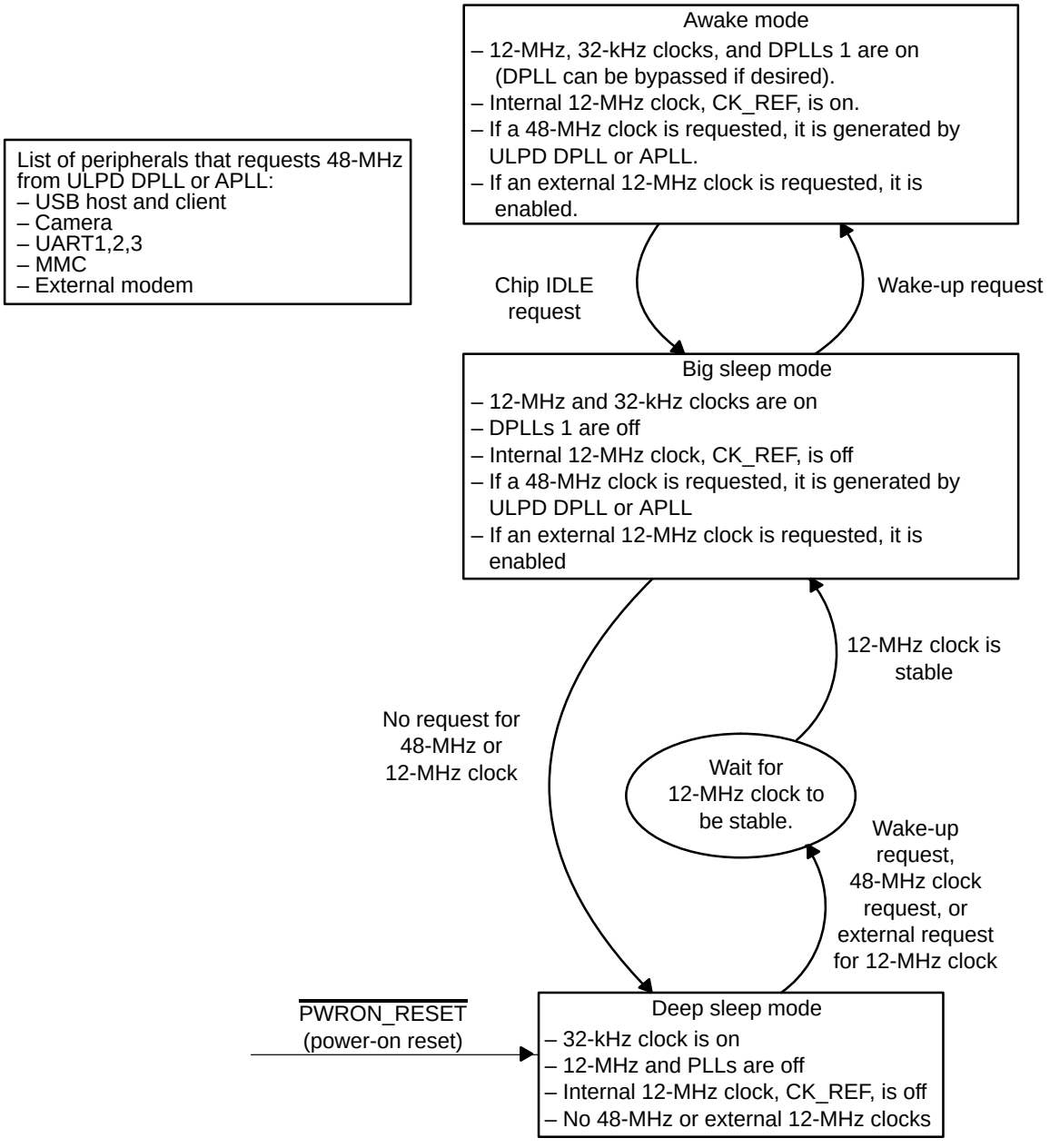


Figure 15–11. Wake-up Control Module

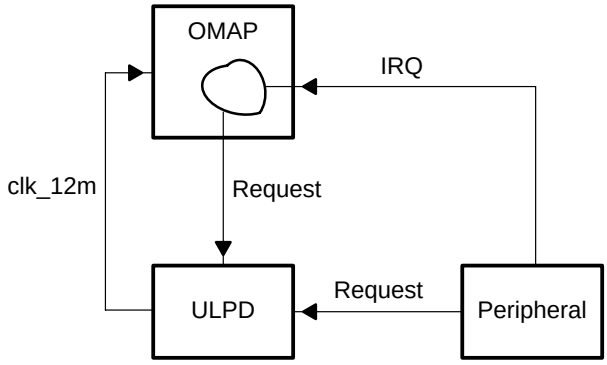


Table 15–3. OMAP5910 Wake-Up Peripherals and External Signals

Requestor	Mechanism	Transition from Deep Sleep to ?
Power on reset (external signal)	Cold reset	AWAKE
$\overline{\text{MPU\_RESET}}$ (external signal)	Warm reset	AWAKE
MPUIO keyboard	MPU interrupt	AWAKE
MPUIO GPIO	MPU interrupt	AWAKE
Timer32K	MPU interrupt	AWAKE
UART2 RX detection	Peripheral request generated to ULPD	AWAKE
RTC	MPU interrupt	AWAKE
USB cable insertion	USB request to ULPD which generates MPU interrupt	BIG SLEEP then AWAKE via the MPU interrupt
UART1/2/3 (wait for a falling edge on the RX, DSR or CTS signals)	MPU/DSP interrupt	AWAKE
MCLK_REQ (external signal)	Request to ULPD	BIG SLEEP
BCLK_REQ (external signal)	Request to ULPD	BIG SLEEP
External DMA request (external signal)	Asynchronous request from TC	AWAKE

15.3.1 DSP Idle Modes

Two DSP registers are used to configure and check the idle modes for the DSP subdomains.

- The idle configuration register (ICR) specifies which clock domains get put into IDLE by the next execution of the IDLE instruction.
- The idle status register (ISR) indicates which subdomains are currently in IDLE mode.

The six different subdomains are:

- DSP core subdomain (DSP CORE/SARAM/DARAM): ICR and ISR bit 0
- DSP DMA controller subdomain (DMA/SARAM/DARAM): ICR and ISR bit 1
- I-cache subdomain (I-cache): ICR and ISR bit 2
- Peripherals subdomain (peripherals outside the DSP): ICR and ISR bit 3:

  Peripherals can be individually controlled (shut off/enabled) by programming the control registers in the clock generation management module (CLKM). To maximize power conservation, seven different peripheral idle modes are defined (UART, GPIO, timers, watchdog timer). Each one can be individually activated and deactivated by software.

  Two different strategies are used to control the clock that feeds the DSP peripherals:

  - The clock is shut off/activated according to the DSP idle mode or application-specific environment (disable the peripheral clocks when the DSP in not in idle). Peripherals connected to this clock cannot request DMA transfers during the DSP idle mode.
  - The clock is never shut off (input reference clock).

    In either case, the DSP peripheral clocks are directly shut off/activated by the DSP software.

- DPLL subdomain (DSP input clock + DSP interrupt handler): ICR and ISR bit 4

  Setting up the DSP DPLL idle mode sends a DSP_IDLE signal to the clock generation management module (CLKM), which disables the input clock to the DSP.

- DSP EMIF subdomain: ICR and ISR bit 5

15.3.1.1 Putting the DSP in IDLE

Perform this procedure to put the DSP in idle mode:

1) Turn off the DSP peripheral clocks.

2) Disable the DSP watchdog timer.

3) Enable the desired DSP interrupts (interrupt mask).

4) Enable the DSP global interrupt (INTM bit), if required.

5) Switch the DSP to shared-access mode (SAM) (it is in SAM, but switch mode to signal the CPU). In SAM, the public peripheral bus is shared between the MPU and DSP.

6) Write to ICR registers to disable all clock domains or a particular clock domain (write a 1 to disable).

7) Switch the DSP back to host-only mode (HOM). In HOM, the public peripheral bus is owned exclusively by the MPU (usually because the DSP is in idle mode or about to go to idle mode).

8) Execute the IDLE instruction.

   DSP goes to sleep. If INTM is set, ISR is not executed after wake up; the interrupt simply wakes the DSP up. The program continues just after the IDLE instruction; otherwise, ISR is executed.

When the DSP is awakened (by an enabled interrupt or an external event such as reset), perform the following procedure:

1) Disable the global interrupt (INTM), if required.

2) Switch the DSP to SAM. The public peripheral bus is now shared by the DSP and MPU.

3) Write a 0 to the respective ICR bit to clear the idle bit.

4) Execute the IDLE instruction.

Note:

To set the DPLL subdomain to idle, switch off the other clock domains (CPU, DMA cache); otherwise, the DSP peripheral cancels the idle request with a bus error indicating that the configuration is not allowed.

15.3.2 MPU Idle Modes

A clock management register, the MPU idle mode entry 1 register (ARM_ IDLECT1), controls the different clock domains (clock enables) during the idle state and allows the user to put different parts of the MPU clock domain into the idle mode, if desired.

Three different subdomains are defined: the MPU subdomain, the DPLL subdomain, and the MPU peripheral subdomain.

15.3.2.1 MPU Subdomain (MPU + MPU Interrupt Handler)

MPU can go into the idle mode in two ways:

- By executing the CP15 instruction wait-for-interrupt*:* Executed by an OS kernel. By configuration, this instruction provokes an Idle1 or an Idle2 mode.
- By setting the SETMPU_IDLE bit of ARM_IDLECT1 (the idle control register in clock generator and system-reset module): Programmed by a process application. Setting these bits (active) allows the MPU to enter in an Idle1 or an Idle2 mode. This is the recommended method to use.

  The MPU clock restarts upon an enabled interrupt request or system reset.

Wait For Interrupt Instruction

When the MPU CP15 instruction is used, the system software does not need to take care of adding any extra cycles to wait after the instruction being executed (the MPU itself takes care of this). When this instruction is executed, the MPU stops its ongoing operations and sends a sleep acknowledge signal to the OMAP clock reset module to request stopping the clock. The MPU does not execute any other access after executing the wait for interrupt instruction. The MPU wakes up when an interrupt occurs and executes the subsequent instructions after servicing the interrupt.

Set Bit 11 of ARM_IDLECT1

After this bit is set through software, the software must wait for a certain number of cycles to ensure no other MPU requests occur before the MPU is idled. This is because there is a latency between the MPU TIPB write to the IDLECT1 register and the time when the MPU sleep request to the MPU core actually goes high. Also, the software must account for the clock cycles required for the MPU to send an acknowledge signal back to the OMAP clock reset module, depending on what operations it was performing.

The following rules ensure that sufficient time is allowed:

- Clock reset module needs at least four reference clock cycles (12-MHz clock) and two MPU clock cycles to send a sleep request to the MPU.
- The MPU needs at least three MPU clock cycles to send back an acknowledge after receiving the sleep request. This is true even if there are no access requests coming from MPU before or after the sleep request. The sleep request is generated from the register write to IDLECT1 Bit 11.
- Add NOP instructions to make sure there is no request coming from the MPU by considering the worst case scenario. This scenario is: MMU and I–cache enabled, a MMU TLB miss (requiring a L1 and L2 fetch), and an I–cache miss, as follows:
  - Four read strobes for the instruction fetch (a line load of 4 words)
  - One read strobe for TLB Miss L1 descriptor fetch
  - One read strobe for TLB Miss L2 descriptor fetch

  This requires six read strobes, which need N number of MPU clock cycles. N depends on the type of memory from which the reads are being made.
- There is a software solution to avoid these extra N MPU clock cycles due to the read strobes. This solution requires only I-Cache be enabled. By changing the loop count value (CMP R2) we can increase/decrease number of cycles, as shown in Figure 15–12.

Figure 15–12. Code Example

```
    .state16            ; thumb mode
    .ref   edata        ; defined by armas
    .global $arm_idle
$arm_idle:
    push   {lr}
    push   {r1–r7}
    adr    r4, into_32_bis
    bx     r
    nop
    nop
    nop
    .state32            ; arm mode
into_32_bis:
    LDR    R1,ARM_IDLECT1
    MOV    R3,#1
    MOV    R3,R3,LSL # 11
    MOV    R2,#0
    LDR    R0,[R1]
    ORR    R0,R0,R3
; This is the loop that will wait for at least 100 cycles
; before issuing next request from ARM. On the first run of the loop only Icache
; gets loaded with the loop and the next 2 instructions but write to SYSST does
not occur
; In the 2nd run of the loop only write to IDLE_CT1 happens and after that ARM
runs the loop from
;Icache so no request goes out
LOOP   CMP    R2,#1
       STREQ  R0,[R1]
       ADD    R2,R2,#1
       CMP    R2,#16
       BNE    LOOP
```

Figure 15–12. Code Example (Continued)

```
the_end:
   adr r2, into_16_bis + 1
   bx  r2
   .state16
into_16_bis:
   nop
   nop
   nop
   nop
   nop
   nop
   nop
   nop
   nop
   nop
   pop    {r1–r7}
   pop    {pc}
;
CONSTANT TABLE
;
ARM_IDLECT1              .long 0xFFFECE04
```

Example:

- If the reference clock speed = 12 MHz, then MPU clock speed = 96 MHz
- If the above routine is used after the write to ARM_IDLECT1 register, you need 4 * 96/12 + 2 + 3 = 37 clock cycles

To achieve this, run the loop seven times.

15.3.2.2 DPLL Subdomain

DPLL1 is disabled when all of the clock domains using DPLL1 are disabled**.**

Setting up the IDLDPLL_MPU bit enables DPLL1 to enter the idle mode when the following conditions are met; otherwise, DPLL1 remains active.

- ❏ DSP is set in the global idle mode.
- ❏ MPU is set in the idle mode (either from request or from wait-for-interrupt).
- ❏ There is no active DMA transaction and there is no activity on the internal local bus, as indicated by the internal TCLB_EN signal
- ❏ No peripheral post write is queued.
- ❏ The peripheral clocks are stopped.
- ❏ There are no pending interrupts.

15.3.2.3 Peripheral Subdomain

Two clocks feed the MPU peripherals:

- ❏ The clock that is shut off/activated according to the MPU idle mode. Peripherals connected to this clock cannot request DMA transfers during the MPU idle mode.
- ❏ The clock that is never shut off (input reference clock)

In either case, the MPU peripheral clocks are directly shut off/activated by the MPU software.

15.3.3 Traffic Controller Idle Modes

A clock management register, ARM_IDLECT1, controls different clock subdomains (clock enables) during the idle state and allows the user to put different parts of the traffic controller into idle mode, if desired.

Several different subdomains are defined.

- ❏ System DMA subdomain

  To optimize power consumption, the system DMA controller clock (DMA_CK) can be shut off/activated according to the DMA activity (provides clock during data transfer only) or it can be shut off only when MPU goes to idle.

- Traffic controller subdomain

  The traffic controller clock (TC_CK) is shut off if the MPU and the DSP DPLLs are in idle mode and there is no DMA transfer. When the clock must be shut off, the memory interface completes the current memory transaction before notifying the CLKM3 to shut off a clock. The SDRAM is placed in self-refresh mode before shutting off the SDCLK_EN clock, and there is no internal local bus activity, as indicated by the internal TCLB_EN signal.

- MPU peripheral bridge domain

  The MPU peripheral bridge clock (TIPB_CK) is shut off if the MPU is in idle mode and there is no DMA transfer.

- LCD domain

  The LCD clock (LCD_CK) is shut off/activated according to the traffic controller idle mode or forced off with the enable bit.

- MPU MPUI domain

  The MPUI clock (API_CK) is shut off if the MPU and the DSP DPLLs are in idle mode and there is no DMA transfer (when the clock is required to be shut off, the memory interface completes the current memory transaction before notifying the CLKM3 to shut off the clock). The MPUI clock can also be forced off with the enable bit (the MPUI clock can only be shut off by the enable bit in current implementation).

Note:

To idle the traffic controller, the system software must ensure that the current transfers are completed and their clock domains are turned off before going to idle.

The traffic controller clock restarts upon either an MPU or DSP interrupt request, a DMA request, or activity on the internal buses.

15.3.3.1 DPLL Idle Procedure

In the event that only the input reference clock is needed (that is, only timer/watchdog are active), then the DPLL can be set to idle mode. This procedure applies to DPLL. The DPLL clock is stopped if all the domains that use it as a source are stopped.

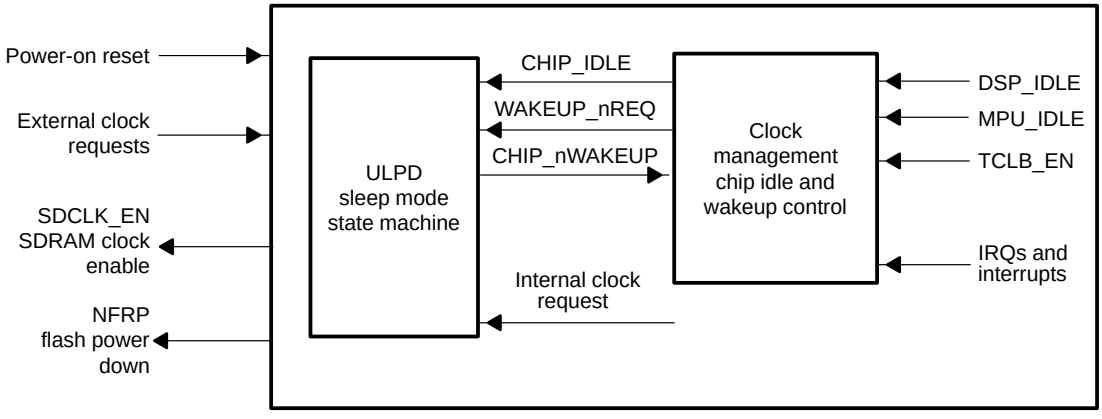
Setting the IDLDPLL_ARM bits (in ARM_IDLECT1 register) to logic 1 forces the corresponding DPLL to enter the idle mode whenever the DPLL output clocks *(CK_GEN1, CK_GEN2, CK_GEN3)* are not being used.

The DPLL idle control logic receives signals from clock generation modules, indicating when the output clock can be stopped. The DPLL idle mode is entered when the IDLDPLL_ARM bits in the MPU idle mode entry 1 register (ARM_IDLECT1) are set to logic 1.

15.3.4 Chip Idle and Wake-Up Control

In the status request register (STATUS_REQ_REG), both the CHIP_IDLE and WAKEUP_nREQ signals can be used by the ULPD idle logic to control the input clock source. The CHIP_IDLE signal remains high until the DPLLs acknowledge that they have entered the IDLE mode. The WAKEUP_nREQ signal goes active as soon as one of the chip wake-up conditions occurs (MPU or DSP interrupt, DMA request, local bus activity). The CHIP_IDLE signal can be used by the ULPD to decide when to stop CLKIN (CK_REF) to the clock generation module, and it uses the WAKEUP_nREQ signal to restart the clock (see Figure 15–13).

Figure 15–13. Chip Idle and Wake-Up Control



In the status request register (STATUS_REQ_REG), the following signals are used in chip idle and wake-up control:

- CHIP_nWAKEUP: CHIP_nWAKEUP is the acknowledge of ULPD when CHIP_IDLE signal is active. It indicates to the clock management module that the ULPD has left the awake mode and the 12-MHz clock, CLKIN, is off.
- When CHIP_IDLE becomes inactive, the ULPD the releases CHIP_nWAKEUP signal to indicate that the ULPD is in awake mode and CLKIN is back on. This signal provides the ULPD with more control of the OMAP5910 device wake up.

- When the MPU idle mode entry 1 register (ARM_IDLECT1) WKUP_MODE field = 0, the OMAP5910 device does not wake up, even if one of the above wake-up conditions occurs, until the CHIP_nWAKEUP goes active.

  - MPU interrupt
  - DSP interrupt
  - Internal local bus activity (TCLB_EN signal = 1)

  In chip idle, if this signal is inactive (high), the OMAP5910 device does not wake up (prevent from waking up), even if one of the above wake-up conditions has occurred. This signal has no effect when the OMAP5910 device is not in chip idle. When the MPU idle mode entry 1 register (ARM_IDLECT1) WKUP_MODE = 0, the OMAP5910 device does not wake up, even if one of the above wake-up conditions has occurred.

- WAKEUP_nREQ: request to the ULPD to wake up the chip. This signal is looked at only when the CHIP_IDLE signal is high. Note that WAKEUP_nREQ is asserted by OMAP whenever there any wake-up condition occurs, even when OMAP is not in CHIP_IDLE.

- $\overline{\text{FLASH.RP}}$ and SDRAM.CKE: Power control for external devices (for example, flash and SDRAM)

- Power-on reset: Must be valid until power and input clock are stable

- CHIP_IDLE: Chip idle mode. Indicates that all internal clocks are stopped. This internal signal is active regardless the state of the external wake-up control feature. This pin is asserted (high) when all DPLL ACKs are returned and all the peripherals using CK_REF are disabled.

- TCLB_EN: TC local bus enable, enables restart of the clock to the TC when the idle mode is set.

- DSP interrupts (internal signal is DSP_nIRQ): Interrupt request from the DSP interrupt handler. Asserts when an unmasked interrupt has been asserted.

- MPU IRQ (internal signal is nIRQ_SET*):* Interrupt request from the MPU interrupt handler. Asserts when an unmasked interrupt has been asserted. When MPU is awake, must remain low until MPU clock restart.

- DSP_IDLE—DSP idle command

- MPU_IDLE—MPU idle command

15.3.4.1 Chip Idle Mode

In the event that no clock signal is necessary (chip idle), all clock domains generated either from CK_GEN (1,2,3) or CK_REF (CLKIN) are stopped. The external reference clock source can be stopped as well.

The CHIP_IDLE signal is asserted when all internal system clocks are disabled and after the DPLL idle state has been acknowledged. It is deasserted when a wake-up condition is detected (CHIP_nWAKEUP and one of the wake-up conditions are active when external wake-up control is enabled or just one of the wake-up conditions is active when wake-up control is not enabled). It takes some synchronization CK_REFs for CHIP_IDLE to go low after a wake-up condition is detected.

Note:

In addition to the DPLL timing, the clock source start-up time can affect the OMAP5910 system response. Use deep sleep mode for long sleep periods only.

15.3.4.2 Chip Idle Procedure

1) Set TC_EMIF_SLOW_IF_CONFIG_REG = 0x0000000C and TC_EMIF_FAST_SDRAM_CONFIG_REG = 0x0C618800

2) Disable the MPU watchdog timer by first writing 0x00f5 to the WDTIMER_TIMER_MODE_REG and then writing 0x00A0 (this is to prevent a watchdog reset from being generated that results in a global reset.)

3) Set up the MPU interface (write to the MPUI control setup and DSP boot registers).

4) Set API_SIZE_REG = 0x0000 (only need to set the bits that correspond to the SARAMs with DSP code to 0).

5) Enable the MPU interrupts and unmask these interrupts by writing a 0 to the corresponding bit in the MIR-mask interrupt register in the MPU interrupt handler. Write to the corresponding interrupt interrupt-level register (ILR) to set the priority, edge sensitivity, and whether the interrupt is routed to the IRQ or FIQ. Besides the interrupt that is used to wake up the MPU out of idle, you also can enable a DSP mailbox interrupt to the MPU. This DSP2MPU mailbox interrupt service routine is used to put the MPU into idle. The reason to use this mailbox interrupt is that the MPUI clock is needed by the DSP to switch between SAM/HOM. The MPU cannot go to idle until this is done, because only the MPU can write to the EN_APICK bit of the MPU idle mode entry 2 register (ARM_IDLECT2).

6) Take the DSP out of reset. First, write the MPU reset control 1 register (ARM_RSTCT1) DSP_RST field = 1. Set the MPU reset control 1 register (ARM_RSTCT1) DSP_EN to = 1. The DSP_RST bit controls the MCU reset, and the DSP_EN bit controls the reset signal of the DSP.

7) In the DSP code, enable and unmask the DSP interrupts.

8) Disable the DSP watchdog timer (that is, take it out of watchdog mode) and disable the DSP peripheral clocks setting the DSP idle mode entry 2 register (DSP_IDLECT2) to 0x0000.

9) Write to the DSP mailbox registers to generate the interrupt to the MPU. In the corresponding MPU interrupt service routine, add a wait loop to make sure the DSP has gone to idle before disabling the MPUI clock.

10) In the MPU interrupt service routine, clear the DSP2MPU mailbox interrupt, then disable the ARMGPIO_CK, LB_CK, and LCD_CK by writing the MPU idle mode entry 2 register (ARM_IDLECT2) = 0x0087 (can also write 0x0000, which disables the MPU peripheral clocks instead of letting them go to IDLE, only after MPU goes to IDLE using the MPU idle mode entry 1 register (ARM_IDLECT1) IDL_ARM). Put the MPU into IDLE by writing MPU idle mode entry 1 register (ARM_IDLECT1) = 0x0FFF, which sets the SETARM_IDLE bit. This also sets the IDLIF bits, which allow the MPU peripherals to go to IDLE when the MPU goes to idle, and sets the IDLDPLL_ARM bit, which allows the DPLLs to go to idle.

11) Back in the DSP code, after writing to the DSP mailbox register in step 10, switch the DSP TIPB and MPUI to SAM. Then, write 0xFF to the DSP ICR register. Switch the DSP back to HOM and execute the idle instruction. See the section on putting the DSP in idle mode for descriptions of the SAM and HOM.

    The DSP and MPU domains go to idle, and then the TC also goes to idle. Then all 3 DPLLs go to idle, and the CHIP_IDLE signal goes active high, which indicates to the external system that the OMAP5910 input clock can be disabled, assuming there are no wake-up conditions at that time indicated by the WAKEUP_nREQ signal.

15.3.4.3 Wake-Up Procedure

An interrupt request (not masked) (either to the MPU or to the DSP), a DMA clock request, or a logical 1 at the TCLB_EN signal exits the idle mode. In addition, when all internal clocks are stopped (chip-idle mode), the wake-up procedure can be controlled via the ULPD. The WKUP_MODE bit of the MPU idle mode entry 1 register (ARM_IDLECT1) defines this wake-up option.

To give the ULPD wake-up control, the WKUP_MODE bit must be cleared to 0 before entering the idle mode.

In the OMAP5910 device, the WKUP_MODE bit is controlled by the MPU in the MPU idle mode entry 1 register (ARM_IDLECT1). The DSP has no control of this bit.

When the WKUP_MODE bit value is set to logic 1, a single wake-up condition (as defined in the following list) initiates a chip wake-up procedure.

1) nIRQ_SET: Upon an interrupt request, the MPU interrupt handler initiates the restart of the ARM_CK, ARM_INTH_CK, TIPB_CKs, DMA_CK, and TC_CK clocks (depending on the setting of the MPU idle mode entry 1/2 registers (ARM_IDLECT1/2), peripherals clocks can also restart). If the idle mode was entered from the SETARM_IDLE bit, then the bit is cleared to 0.

2) DSP_nIRQ: Upon an interrupt request, the DSP interrupt handler initiates the restarting of the MPUI clock (if MPUICK_EN is not set to 0), DSP_CK, DSP_INTH_CK, TC_CK clocks (depending on the setting of the MPU idle mode entry 1/2 registers (ARM_IDLECT1/2), peripheral clocks can also restart).

3) TCLB_EN signal: When the internal TCLB_EN signal goes active, the TC_CK and LB_CK clocks restart. The TC_CK/LB_CK clocks keep running as long local bus activity occurs.

4) When the system DMA controller receives an asynchronous request from the traffic controller, DMA_CK/TC_CK/LB_CK and DMA_CK/TC_CK/ LB_CK are enabled to keep running as long as the DMA operates.

5) When the system DMA controller receives a request from the TIPB bridge, it enables TC_CK/TIPB_CKs/DMA_CK and TC_CK/TIPB_CKs/DMA_CK to keep running as long as the DMA operates.

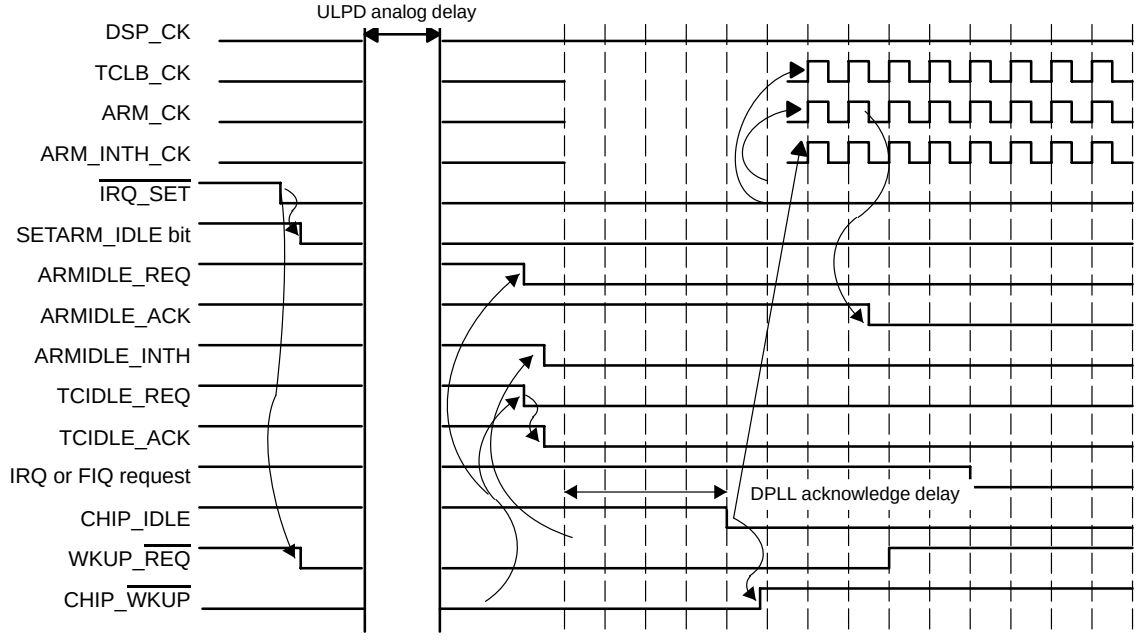
Note:

The internal signals that cause the wake-ups are asynchronous and do not need a running clock to be activated. When the WKUP_MODE bit value is a logical 0 and the CHIP_IDLE signal is active, this condition indicates the entire chip is in deep sleep mode. The combination of one of the above conditions and a CHIP_nWAKEUP request from the ULPD is required to exit the chip-idle mode. If WKUP_MODE = 1, then the ULPD is not required to exit the chip-idle mode, only one of the above wake-up conditions.

If the CHIP_IDLE signal is inactive (and at least one of the internal clocks is running), the CHIP_nWAKEUP signal is disabled and a single wake-up condition (not ULPD controlled) brings the DSP or MPU system out of idle mode. A global system reset brings the OMAP5910 device out of idle mode, regardless of the WAKEUP_MODE bit value, ULPD control, or the interrupt status.

Figure 15–14 illustrates an ULPD-controlled wake-up sequence (assuming the DSP and the MPU clock domains have the same clock frequency, CK_REF). The wake-up is initiated from an interrupt to the MPU, whereas the DSP remains in idle mode.

Figure 15–14. ULPD Controlled Wake-Up Sequences



15.3.5 Power-Saving Capability

The OMAP5910 device has several power-saving modes that help to reduce the operating current by stopping the clock signals of the unused (inactive) domain(s). The idle controls to the DSP, the MPU, and the traffic controller provide a flexible and an efficient power-saving mechanism. The following list of power-saving modes is given as an example only, because other modes are possible. The system software can program the OMAP5910 device to operate in any of these modes for a specific application.

- Mode 0: The DSP is partially in the idle mode (see DSP idle protocol).
- Mode 1: The DSP is in the global idle mode.
- Mode 2: The MPU is in the idle mode (DSP is still running) for both:
  - System DMA controller is active.
  - System DMA controller is not active.
- Mode 3: Both MPU and DSP are set in the idle mode (peripherals are still active) for both:
  - System DMA controller is active.
  - System DMA controller is not active.
- Mode 4: The MPU, the DSP, and traffic controller are stopped (the traffic controller can be stopped only if both the MPU and the DSP are in idle) for both:
  - Peripheral modules are individually stopped.
  - All peripherals are stopped.
- Mode 5: The MPU, the DSP, peripherals, and DPLL (1) are stopped; however, the timer/watchdog (or OS-timer) is still active.
- Mode 6 (chip idle): The MPU, the DSP, peripherals, DPLL (1), and timers are stopped, while the ULPD clock source remains the only active clock signal.
- Mode 7 (deep sleep): All internal system clocks (the MPU, the DSP, peripherals, DPLL (1) and timers) and the ULPD reference clock source are stopped, leaving the OMAP5910 device in a static state in which it consumes the lowest possible power.

In all power-saving modes, the OMAP5910 device retains all RAM data (keeps the memory data), and the register configuration values (for example, the frequency selection is maintained, etc.). The data to output terminals is also maintained, and input terminals are set to logic low or logic high (not floating) to reduce the current from flowing through the input logic.

The OMAP5910 device exits from the power-saving modes by means of a reset or any interrupt (with or without additional external control from the CHIP_nWAKEUP pin). A wake-up interrupt must be enabled (not masked off) to bring the OMAP5910 device out of the power-saving mode.

15.3.6 ULPD Power Management State Machine

The power management function of the ultralow-power module (ULPD), handles the high-frequency oscillator on/off sequences. It is a state machine that can stop the oscillator and restart it on a wake-up signal. Set-up times of the oscillator are taken into account in order to stop/restart internal clocks in a clean manner. The ULPD state-machine uses the 32-kHz clock.

- The ULPD DPLL and APLL for the 48-MHz USB clock is handled in the ULPD module.
- ULPD DPLL is a x4 digital PLL.
- APLL is a x48 analog PLL. The input clock ref is 1 MHz, based on either a 12-MHz system clock divided by 12 or optionally on a 13-MHz system clock divided by 13.
- The switch between DPLL and APLL is controlled by software through a TIPB register of ULPD.

15.3.6.1 Gauging the 32-kHz Oscillator

As the 32-kHz oscillator exact frequency is unknown, it can be gauged by comparing the 32-kHz oscillator with a high-frequency clock (12-MHz oscillator, ULPD DPLL, or external clock) during any active period. Gauging is only necessary in specific applications where it is important to know the exact frequency of the 32-kHz oscillator.

There is a 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). You must first disable the gauging (GAUGING_CTRL_REG[0] = 0), then read the high-frequency counter and the 32-kHz counter value.

15.3.6.2 Gauging Versus High-Frequency Clock

To gauge the 32-kHz oscillator, two counters clocked on the 32-kHz clock and the high-frequency clock are concurrently running during the gauging period. The high1–frequency clock is selected among the 12-MHz clock, the DPLL, and the external clock. At the end of the gauging period, the number of 32-kHz clocks and the number of high-frequency clocks are calculated as follows:

- Nb_32kHz = counter_32_msb * 65536 + counter_32_lsb:
  - counter_32_msb is the MSB value of 32-kHz counter.
  - counter_32_lsb is the LSB value of 32-kHz counter.
- Nb_hi_freq = counter_hi_freq_msb * 65536 + counter_hi_freq_lsb:
  - counter_hi_freq_msb is the MSB value of high-frequency counter.
  - counter_hi_freq_lsb is the LSB value of high-frequency counter.

Use the following procedure to gauge the 32-kHz clock versus the high-frequency clock:

1) Select gauging versus high-frequency clock:
   - Write gauging_ctrl[0] = 0 to select gauging versus 12-MHz clock.

     or
   - Write gauging_ctrl[0] = 1 to select gauging versus external or DPLL clock.
2) Start gauging by writing gauging_ctrl[1] = 0.
3) Wait a few seconds.
4) On reception of the gauging interrupt (low-level sensitive interrupt), stop gauging by writing gauging_ctrl[1] = 0.
5) Check the overflow of the 32-kHz counter:

   Read it_status[2].
6) Check the overflow of the high frequency counter:

   Read it_status[1].
7) If an overflow occurred during the process, then return to step 3 to restart the gauging and reduce the waiting time. Else continue to next step.
8) Read 32-kHz counter value:

   Read counter_32_msb and counter_32_lsb registers.

9) Read hi_freq counter value:

   Read counter_hi_freq_msb and counter_hi_freq_lsb registers.

10) Compare values of the counter and proceed with calibration of the 32-kHz counter.

At the end of the gauging operation, an END_GAUGING interrupt informs the MCU that gauging is stopped and values of the counters are ready to be read. This interrupt is low-level sensitive. It is cleared on the reading of these registers.

Two other low-level-sensitive interrupts indicate whether an overflow occurred on one of the two counters during the gauging operation. One interrupt is dedicated for the 32-kHz counter, the second one for the high-frequency counter. They are also cleared on the read of the interrupt status register.

15.3.6.3 Control of 32-kHz Oscillator

The 32-kHz oscillator start-up time is configurable via the bits MOD_32KOSC_SW_R bits of the module configuration control 0 register (MOD_CONF_CTRL_0) in OMAP5910 configuration.

The 32kHz clock source can come from either the on-chip 32-kHz oscillator or from an external 32-kHz clock oscillator providing a clock onto the CLK32K_IN input pin. Clock source selection depends upon the state of the CLK32K_CTRL input pin. If this pin is driven (or tied) high, the on-chip 32-kHz oscillator is enabled as the clock source. If the pin is driven (or tied) low, then the clock must be provided externally on the CLK32K_IN pin.

MOD_32KOSC_SW_R is a 4-bit field that sets the performance control switches of the oscillator 32-kHz (SW1, SW2, SW3, SW4). Table 15–4 lists the recommended control switch settings:

Table 15–4. Recommended Control Switch Settings

Oscillator Performance	SW4	SW3	SW2	SW1
Least current	1	0	0	0
Fast startup	1	0	1	1

15.3.6.4 Battery Failed

A battery-failed event is indicated via the BFAIL/$\overline{\text{EXT\_FIQ}}$ signal connected to the ULPD, which performs a power-down action. The ULPD processes the incoming BFAIL/$\overline{\text{EXT\_FIQ}}$ signal to an external device (via $\overline{\text{RST\_HOST\_OUT}}$) to decrease a programmable counter and to generate a shutdown signal when the counter reaches zero. It also creates an interrupt on MPU level 1. The release of $\overline{\text{RST\_HOST\_OUT}}$ is controlled by software. You can use the SW_NSHUTDOWN bit in the power control register (POWER_CTRL_REG) to toggle the state of the $\overline{\text{RST\_HOST\_OUT}}$ pin to high or low level. A power-on reset condition also causes $\overline{\text{RST\_HOST\_OUT}}$ to be active low.

15.3.6.5 Big Sleep and Deep Sleep Mode

To go into deep sleep mode, the MPU and DSP must release and unmask all interrupts that can awake the chip during this mode and they must mask all interrupts that do not awake the chip during deep sleep mode. Then the MPU enters big sleep mode. When the device reaches the idle state, it informs power management to enter into sleep mode by setting the CHIP_IDLE signal. The state machine cuts the 12-MHz OSC1 oscillator only if external devices do not request the oscillator clock.

15.3.6.6 Power-On Reset

The $\overline{\text{PWRON\_RESET}}$ signal is used to reset the entire device. This signal is resynchronized on the 32-kHz to achieve a clean reset. Then the ULPD module initiates the internal reset sequence (minimum of two full 32-kHz clock periods is recommended). The 32-kHz logic within the ULPD module is reset asynchronously.

15.3.6.7 Interrupt Wake-Up

When at least one unmask interrupt has occurred during deep sleep, a wake-up sequence is performed. In this case, the chip must leave the deep-sleep mode as soon as possible to process this interrupt. To respect the schedule of the clock enables, the following wake-up sequence is processed:

1) When the clock management interrupt handlers detect an interrupt, the internal WAKEUP_nREQ signal is asserted.

2) The 12-MHz OSC1 oscillator is enabled and the setup counter is loaded with the setup_oscillator value.

3) When the setup counter reaches zero, the CHIP_nWAKEUP signal is used to inform the device to leave big sleep mode.

15.3.6.8 Functional Reset Generation

The ULPD generates the functional reset of the device internally from the $\overline{\text{PWRON\_RESET}}$ signal and holds it active low for a minimum of 20 REF_CK (12-MHz) clock cycles.

15.3.7 32-kHz Oscillator

The 32-kHz oscillator is always on and uses a 32-kHz external quartz. The modules working with the 32-kHz clock are:

- ❑ 32-kHz timer
- ❑ Power management
- ❑ UART communication with communication processor
- ❑ PWL
- ❑ MPUIO (debouncing)/keyboard (keypad)

15.3.8 12-MHz Oscillator

This oscillator is to be used with a 12-MHz external quartz, which is used by the clock and reset management module and is provided to the clock ref of DPLL1 for MPU, DSP, TC, and peripherals. It allows the generation of the 48 MHz required by USB, camera, MMC, and UARTs. The ULPD DPLL and APLL, located in the ULPD, provide the 48 MHz. By default, the USB uses the APLL. This 12-MHz clock is used as the input clock of the ULPD. The 12-MHz oscillator is on during awake and big sleep mode. It is off during deep-sleep mode. The power management module handles the wake-up sequence.

When using the on-chip oscillator (normal mode), either a 12-MHz crystal or a 13-MHz crystal can be connected to the OSC1_IN and OSC1_OUT pins so that the on-chip oscillator generates a 12-MHz or a 13-MHz clock reference to the OMAP device. In external master mode, the on-chip oscillator is disabled and a 12-MHz or 13-MHz reference clock must be provided by an external oscillator connected to the OSC1_IN pin.

In both of these cases, the APLL can be configured to generate the 48-MHz clock from either a 12-MHz or a 13-MHz reference clock. If a 13-MHz reference clock is used, then use the APLL to generate the 48-MHz clock, as opposed to using the ULPD DPLL. The ULPD DPLL can only generate a 48-MHz clock when the reference is 12 MHz.

Note: 13-MHz Clock

If a 13-MHz reference clock is used, then any timings detailed in this document (peripheral clocks, etc.) which are calculated based on the 12-MHz reference clock must be recalculated using the 13-MHz value.

15.3.9 Reset Protocol

The OMAP5910 device system reset is accomplished with a combination of hardware and software control. Individual components (or modules) can be reset by software.

There are five different sources that can cause a system reset. Three of them are internally generated, and two of them are input from the external pin. These sources are: the cold reset ($\overline{\text{PWRON\_RESET}}$ reset pin), the warm reset (from either the $\overline{\text{MPU\_RST}}$ pin or software-generated), and the watchdog reset (MPU and DSP).

- The $\overline{\text{PWRON\_RESET}}$ signal must be asserted for at least two 32-kHz clock periods to be recognized. $\overline{\text{PWRON\_RESET}}$ indicates a power-on reset of the device. When $\overline{\text{PWRON\_RESET}}$ is asserted low, the internal power on reset and warm reset of MPU are active low until the 12-MHz clock (REF_CK) is on. Then power-on reset is released after 20 REF_CK cycles, and warm reset is released after 30 additional cycles.

- When the device is awake, $\overline{\text{MPU\_RST}}$ controls the warm reboot of the MPU. The external reset signal must be asserted for 30 REF_CK cycles to be recognized. The reset signal is synchronized before feeding to the reset manager module (RSTM) that generates the internal reset signals within the OMAP5910 device. The internal reset is asserted for at least 64 REF_CK cycles, and that clock must be running when $\overline{\text{MPU\_RST}}$ is asserted.

Some configuration registers are only reset to their default values by certain types of resets:

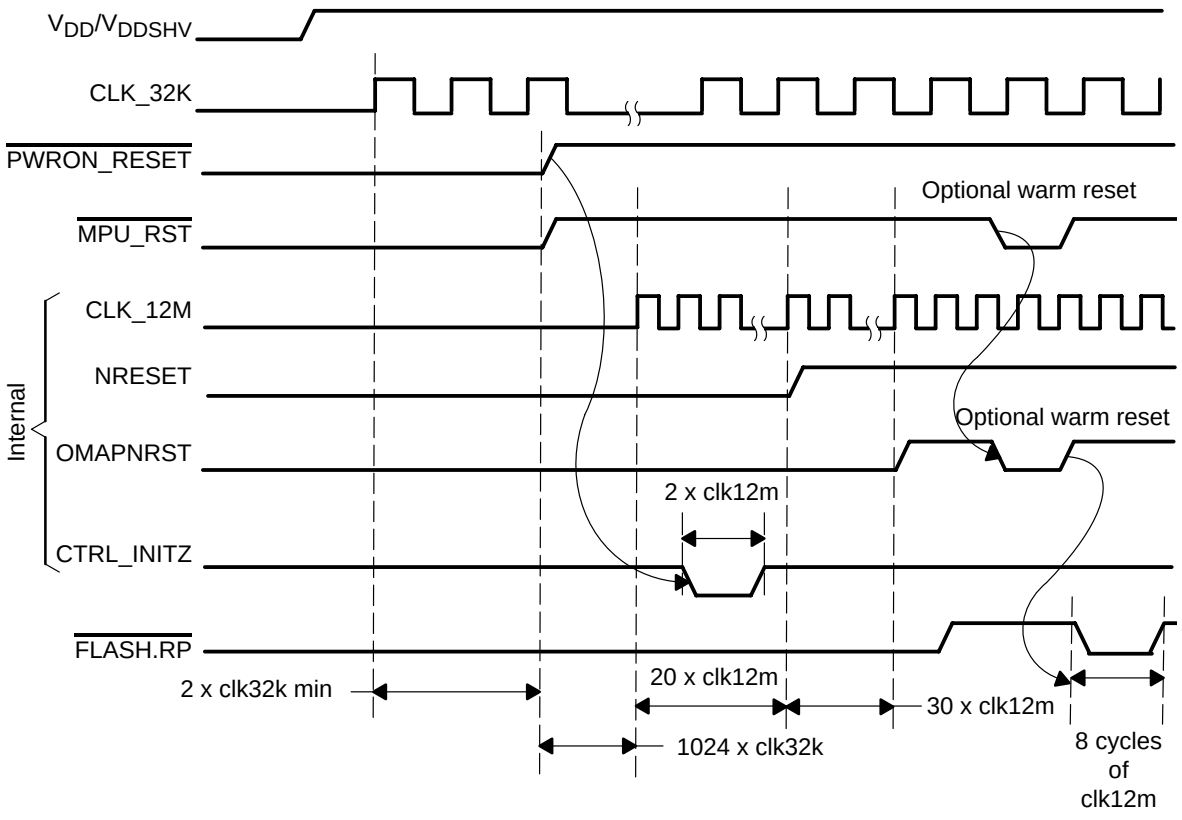
The following registers are only reset by power-on reset:

- ULPD
- Functional multiplexing configuration
- MPU level 2 interrupt handler
- MPUIO (the output register (OUTPUT_REG)
- Input/output control register (IO_CNTL))
- MCSI (the main parameters register (MAIN_PARAMETERS__REG))

DPLL registers are reset by all the resets, both hardware and software

The OMAP5910 device global reset sequence with the behavior of the $\overline{\text{FLASH.RP}}$ pin is shown in Figure 15–15.

Figure 15–15. External Power Control During A Reset Sequence



15.3.9.1 Cold Reset

Cold reset is in response to the assertion of the external reset signal (PWRON_RESET). When the reset is initiated from the pin, the OMAP5910 device is held in reset until the pin goes inactive. The reset module generates reset signals to the respective modules. All modules are put to a known state, and the RAM data is in an unknown state.

During the power-up reset, the DSP and the DSP subsystem are held in reset mode (by hardware). The MPU boots from CS0 or CS3. Software then writes to the control registers to release the reset of the DSP subsystem once the MPU is up and running.

15.3.9.2 Warm Reset

A warm reset can be generated from either the $\overline{\text{MPU\_RST}}$ pin or through system software. There are several types of warm reset:

- Global: Resets the DSP, the MPU, and all internal modules through the SW_RST field, the ARM_RST field, or the DSP_EN field of the MPU reset control 1 register (ARM_RSTCT1).
- DSP: Resets the DSP system, with the exception of the configuration setting. The reset of the associated internal peripherals (DSP interrupt handler, timers, UART, GPIO, etc.) is controlled from the peripheral reset signal issued by the DSP. This is done through the SW_RST field, the DSP_RST field, or the DSP_EN field of the MPU reset control 1 register (ARM_RSTCT1).
- MPU: Resets the MPU subsystem. The signal_reset causes an MPU reset, as does the ARM_RST field of the MPU reset control 1 register (ARM_RSTCT1).
- MCU: Resets the peripheral interrupt priority encoder registers and part of the MPUI control logic.

15.3.9.3 Watchdog Reset (DSP and MPU)

- MPU watchdog: A system reset (global reset) is generated when the down-counter underflows (assuming the MPU watchdog timer is configured as a watchdog timer).
- DSP watchdog: A DSP reset (reset the DSP and the DSP MMU) is generated when the down-counter underflows (assuming the DSP watchdog timer is configured as a watchdog timer)

15.3.9.4 Warm Reset via $\overline{\text{MPU\_RST}}$

As mentioned earlier, one type of warm reset (also called warm boot) is caused by the $\overline{\text{MPU\_RST}}$ input pin being driven active low. In a portable application initiating a warm-boot via $\overline{\text{MPU\_RST}}$ is a very convenient feature since the removal of battery sourced power is sometimes difficult or impossible within the application. Such a warm reset is also useful in line–powered systems when a system restart is desired without cycling power.

The $\overline{\text{MPU\_RST}}$ input pin has the following characteristics:

- When driven active low, this pin activates a warm reset, forcing the MPU processor to reboot.

- ❏ The $\overline{\text{MPU\_RST}}$ input pin has a hysteresis type input buffer.
- ❏ De-bouncing is not implemented
- ❏ The pin has a minimum pulse width requirement (refer to appropriate data-sheet all timing requirements).

When a warm reset condition occurs, the $\overline{\text{RST\_OUT}}$ output pin is asserted active low. The $\overline{\text{RST\_OUT}}$ output pin is always enabled (the $\overline{\text{RST\_OUT}}$ function cannot be disabled). Refer to the appropriate device datasheet for complete timing characteristics of $\overline{\text{RST\_OUT}}$ relative to $\overline{\text{MPU\_RST}}$.

The assertion of $\overline{\text{MPU\_RST}}$ (warm reset) has the following effects:

- ❏ Always forces a re-boot of the MPU processor
- ❏ Shared GPIO logic and registers are reset.
- ❏ ULPD registers are reset except for the following cases:
  - ■ IT_STATUS_REG: pending interrupts could possibly survive the warm-reset and become posted again after the reset condition.
  - ■ DPLL register is not reset.
  - ■ SETUP_ANALOG_CELL3_ULPD1_REG is not reset.
- ❏ ARMIO_CTL and DATA_OUT registers associated with MPUIO logic are NOT reset and retain their previous values.
- ❏ The appropriate bits within the ARM_SYSST register are set to indicate that the reset event was due to a warm-boot (the MPU can read ARM_SYSST to differentiate a warm-boot from a power on reset).
- ❏ Self-refresh of external SDRAM is initiated if this function is enabled via the appropriate Traffic Controller registers.

The following registers /logic are unaffected by a warm reset condition and retain their previous state:

- ■ OMAP5910 Configuration Registers associated with device pin multiplexing
- ■ OMAP5910 Configuration Registers which enable/disable pullup/ pulldown resistors on the device pins.
- ■ MPU Level 2 interrupt handler registers
- ■ MPUIO control and data registers
- ■ MCSI1 and MCSI2's MAIN_PARAMETERS_REG registers.

- RTC (on–chip Real–time–clock) registers and logic.

Since only selective peripheral registers are reset by a warm-reset via MPU_RST assertion, it is recommended that the safest approach in using the MPU_RST signal for warm-reset is to always perform a complete system re-initialization at re-boot.

The OMAP5910 device implements a single low power pin (LOW_PWR) that indicates to external logic that the device is in low power mode or deep sleep mode. The LOW_PWR signal is multiplexed on the same pin as the MPUIO5 signal, so to utilize the low power function, user software needs to configure the pin appropriately as LOW_PWR. A warm reset condition (MPU_RST active) has the following effects on the LOW_PWR output pin:

Given a warm reset condition when OMAP5910 is awake:

- Multiplexing logic responsible for driving LOW_PWR onto the correct pin is not reset.
- The LOW_PWR pin remains low

Given a warm reset condition when OMAP5910 is in low power or deep sleep mode:

- Multiplexing logic responsible for driving LOW_PWR onto the correct pin is not reset.
- The LOW_PWR pin transitions from high to low.

OMAP5910 implements a deep sleep mode wherein the 12 MHz oscillator is powered down. A MPU_RESET event will alert the ULPD module, which will turn on the 12 MHz oscillator. When this oscillator's clock is stable, the re-boot of the ARM9 can begin. The time required for oscillator stability is defined by the value programmed in the analog delay counter.

15.3.10 Power Control for External Devices

The idle and wake-up control module implements a power control for external devices through the FLASH.RP output pin. The FLASH.RP signal is asserted low for eight input clock cycles (Trl) when a global reset occurs.

Before an idle/wake-up sequence entry, the external power control can be enabled/disabled and the time can be programmed depending on the system application. When a global reset is asserted, the timing of the FLASH.RP signal is fixed as shown in Figure 15–15. The eight cycle count starts from when the MPU_RST pin is detected high (OMAPNRST). This means that there

are actually a few extra cycles after the $\overline{\text{MPU\_RST}}$ pin is deasserted high for synchronization before the cycle count starts.

Note:

The $\overline{\text{FLASH.RP}}$ signal is low when either $\overline{\text{PWRON\_RESET}}$ or $\overline{\text{MPU\_RST}}$ is low, and it stays low for an additional ~8 CLKINs after $\overline{\text{PWRON\_RESET}}$ and $\overline{\text{MPU\_RST}}$ are released.

In addition, $\overline{\text{FLASH.RP}}$ goes low if the TC is in IDLE mode and ARM_EWUPCT REPWR_EN field = 0.

15.3.11 Configuring Clocks After a Reset

After a reset, the device is in the fully synchronous clocking mode. DPLL1 is selected as the source for CLK_GEN1, CLK_GEN2, and CLK_GEN3. The DPLL1 is disabled, so the device is running at the CK_REF frequency. Set the domains to operate at the desired frequencies as follows:

- ❏ Select the desired clocking mode via the CLOCK_SELECT bit of the MPU system status register (ARM_SYSST).
- ❏ Program each of the division modes for the DPLLs for the clock domains.
- ❏ Program the DPLLs and enable them.
- ❏ Program each domain-defined enable bit (discussed in section 15.4, *Clock Generation and Reset Control Registers*). Some peripherals have additional enables for their local clocks (discussed in various peripheral chapters).

After a reset, the application software can write to the control registers via CLOCK_SELECT (2:0) bits of the MPU system status register (ARM_SYSST) to switch to a desired mode of operation. However, use the system software to save the context before switching modes. For information about the switching procedure, see Appendix B, *Switching Clock Modes*.

15.4 Clock Generation and Reset Control Registers

The clock generation and system reset module contains 16-bit registers for the following functions:

- Reset control
- System clocks
- Power-saving mode
- Wake-up control
- Operations
- CLKOUT pins

These registers are partitioned into MPU (see Table 15–5) and DSP (see Table 15–17) groups.

Note:

All MPU clock generator and system reset control registers are 32-bit accessed and all are 32-bit word aligned. All DSP control registers are 16-bit accessed by the DSP and 32-bit word aligned.

- The MPU address for these registers starts at address(hex): FFFFCE80.

Note:

All registers dedicated to the MPU are write-accessed in supervisor mode only.

- Some registers are dedicated to the DSP subsystem and can be monitored by the DSP only. Those registers are mapped to the DSP memory space starting at DSP word address (hex): 004000. They can also be accessed by the MPU through the MPUI interface.

The remaining registers are controlled by the MPU only. They are memory mapped to the MPU memory space starting at address (hex): FFFECE00.

The physical address of a register is the starting address (defined by the system) plus the offset address (given in Table 15–5 and following registers).

Each processor can read its associated registers at any time without affecting ongoing operations, and the registers can be written via their bits.

Table 15–5 lists the MPU clock/reset/power mode control registers.

Bit Width: 32

In the OMAP5910 device, the MPU is the master at all times; it has complete control of the clock generator and system reset module.

Table 15–5. MPU Clock/Reset/Power Mode Control Registers – Base Address: FFFE:CE00

Register Name	Descriptions	R/W	Size	Offset	Reset Value
ARM_CKCTL	Defines frequency for MPU, LCD, LCLB, MPUPER clocks	R/W	16 bits	x00	0x0000 3000
ARM_IDLECT1	Enables and defines idle mode entry for each clock domain	R/W	16 bits	x04	0x0000 0400
ARM_IDLECT2	Controls clock domains individually	R/W	16 bits	x08	0x0000 0100
ARM_EWUPCT	Delay from external device restore power with reference to MPU clock	R/W	16 bits	x0C	0x0000 003F
ARM_RSTCT1	Initiates S/W reset to MPU and DSP	R/W	16 bits	x10	0x0000 0000
ARM_RSTCT2	Set PER_EN signal	R/W	16 bits	x14	0x0000 0000
ARM_SYSST	Contains system information such as reset status flags, processor state	R/W	16 bits	x18	0x0000 0038
ARM_CKOUT2	Reserved			0x20	

The MPU clock control register (ARM_CKCTL) defines the frequency for ARM_CK, DSPMMU_CK, TC_CK, DSP_CK, LCD_CK, and MPUPER_CK.

Table 15–6. MPU Clock Control Register (ARM_CKCTL)

Bit	Name	Value	Description	Type	Reset Value
15	RESERVED		Reading this bit gives an undefined value, and writing to it has no effect.		
14	ARM_INTHCK_SEL		This bit controls which clock is used for ARM_INTH_CK.	R/W	0
		0	TC_CK (this is default and must not be changed)		
		1	Reserved		

Note: If you select the fully synchronous mode, then it is your responsibility to program the divide-down bits so that ARMDIV, DSPDIV DSPMMUDIV, and TCDIV are all equal. At reset, these divide-down bits are all defaulted to divide by 1.

In any mode, the DSPDIV and DSPMMUDIV must be set so that the DSPMMU_CK is either = to DSP_CK or DSP_CK/2.

In synchronous scalable mode, you must make sure that the DSPMMUDIV and ARMDIV are greater than or equal to TCDIV.

Table 15–6. MPU Clock Control Register (ARM_CKCTL) (Continued)

Bit	Name	Value	Description	Type	Reset Value
13	EN_DSPCK		This bit allows turning on DSP_CK while DSP is still in a reset state.	R/W	1
		0	Disable DSP_CK to be turned off during reset state.		
		1	Enable DSP_CK to be turned on during reset state.		
12	ARM_TIMXO		Selects either CK_GEN1 or input reference clock (CLKIN) to supply internal MPU timers.	R/W	1
		0	The ARMTIM_CK clock frequency is input reference clock.		
		1	ARMTIM_CK frequency is issued from CK_GEN1.		
11–10	DSPMMUDIV (1:0)		These read/write bits define prescaler value from frequency of C*K_GEN2* to DSPMMU clock domain clock (DSPMMU_CK).	R/W	0
9–8	TCDIV (1:0)		These read/write bits define prescaler value from frequency of CK_GEN3 to TC clock domain clock (TC_CK).	R/W	0
7–6	DSPDIV (1:0)		These read/write bits define prescaler value from frequency of CK_GEN2 to DSP clock domain clock (DSP_CK).	R/W	0
5–4	ARMDIV (1:0)		These read/write bits define prescaler value from frequency of CK_GEN1 to MPU clock domain clock (ARM_CK).	R/W	0

Note: If you select the fully synchronous mode, then it is your responsibility to program the divide-down bits so that ARMDIV, DSPDIV DSPMMUDIV, and TCDIV are all equal. At reset, these divide-down bits are all defaulted to divide by 1.

In any mode, the DSPDIV and DSPMMUDIV must be set so that the DSPMMU_CK is either = to DSP_CK or DSP_CK/2.

In synchronous scalable mode, you must make sure that the DSPMMUDIV and ARMDIV are greater than or equal to TCDIV.

Table 15–6. MPU Clock Control Register (ARM_CKCTL) (Continued)

Bit	Name	Value	Description	Type	Reset Value
3–2	LCDDIV (1:0)		These read/write bits define prescaler value from frequency of CK_GEN3 to LCD controller clock signal (LCD_CK).	R/W	0
1–0	PERDIV (1:0)		These read/write bits define prescaler value from frequency of CK_GEN1 to peripheral clock domain (MPUPER_CK)	R/W	0

Note: If you select the fully synchronous mode, then it is your responsibility to program the divide-down bits so that ARMDIV, DSPDIV DSPMMUDIV, and TCDIV are all equal. At reset, these divide-down bits are all defaulted to divide by 1.

In any mode, the DSPDIV and DSPMMUDIV must be set so that the DSPMMU_CK is either = to DSP_CK or DSP_CK/2.

In synchronous scalable mode, you must make sure that the DSPMMUDIV and ARMDIV are greater than or equal to TCDIV.

Table 15–7 lists the frequency selections for TC_CK and LCD_CK clocks.

Table 15–7. TC_CK and LCD_CK Frequency Selections

TCDIV(1) LCDDIV(1)	TCDIV(0) LCDDIV(0)	TC_CK Frequency LCD_CK Frequency
0	0	CK_GEN3/1
0	1	CK_GEN3/2
1	0	CK_GEN3/4
1	1	CK_GEN3/8

Table 15–8 lists the frequency selection for DSP_CK clocks.

Table 15–8. DSP_CK Frequency Selections

DSPDIV(1)	DSPDIV(0)	DSP_CK Frequency
0	0	CK_GEN2/1
0	1	CK_GEN2/2
1	0	CK_GEN2/4
1	1	CK_GEN2/8

Table 15–9 lists the frequency selection for ARM_CK and MPUPER_CK clocks.

Table 15–9. ARM_CK and MPUPER_CK Frequency Selections

ARMDIV(1) PERDIV(1)	ARMDIV(0) PERDIV(0)	ARM_CK Frequency MPUPER_CK Frequency
0	0	CK_GEN1/1
0	1	CK_GEN1/2
1	0	CK_GEN1/4
1	1	CK_GEN1/8

The MPU idle mode entry 1 register (ARM_IDLECT1) enables and defines the idle mode entry to each clock domain.

Table 15–10. MPU Idle Mode Entry 1 Register (ARM_IDLECT1)

Bit	Name	Value	Description	Type	Reset Value
15–12	RESERVED		Reading these bits gives undefined value. Writing them has no effect.		
11	SETARM_IDLE		Initiates MPU idle mode when written to a logical 1 and is cleared by a global reset or an interrupt request (nIRQ_SET) from interrupt handler to MPU processor:	R/W	1
		0	MPU active (or in idle mode, set via wait-for-interrupt instruction)		
		1	MPU in idle mode		

Note: When the timer/watchdog is configured as watchdog timer, the clock is never shutdown, regardless of the value of the IDLWDT_ARM bit.

Table 15–10. MPU Idle Mode Entry 1 Register (ARM_IDLECT1) (Continued)

Bit	Name	Value	Description	Type	Reset Value
10	WKUP_MODE		Enables MPU to exit idle mode from an interrupt and triggers an event on CHIP_nWAKEUP signal:	R/W	1
		0	After interrupt asserted, MPU idle mode is exited upon a low level at CHIP_nWAKEUP pin. The wake-up conditions wake up OMAP5910 device out of CHIP_IDLE only if CHIP_nWAKEUP is active.		
		1	Idle mode exited upon an MPU interrupt (regardless of CHIP_nWAKEUP signal). Also, any wake-up condition wakes up OMAP5910 device out of CHIP_IDLE, regardless of value on CHIP_nWAKEUP signal.		
9	IDLTIM_ARM		Selects idle entry mode for internal MPU timer clock:	R/W	0
		0	Clock supplied to timers remains active when MPU enters idle mode (ARM_CK stopped)		
		1	Timer clock stopped in conjunction with MPU clock when idle mode entered		
8	RESERVED		Reserved. To prevent errant behavior, always write this bit as 0.	R/W	0
7	IDLDPLL_ARM		Enables DPLL1 to enter idle mode when following conditions are met: ❑ DSP set in global-idle mode ❑ MPU set in idle mode (either from request or wait-for-interrupt) ❑ No active DMA transaction or TCLB_EN signal inactive ❑ No peripheral bus posted write queued ❑ Peripheral clocks stopped	R/W	0
		0	DPLL1 remains active when above conditions occur.		
		1	DPLL1 enters idle mode when above conditions are met.		

Note: When the timer/watchdog is configured as watchdog timer, the clock is never shutdown, regardless of the value of the IDLWDT_ARM bit.

Table 15–10. MPU Idle Mode Entry 1 Register (ARM_IDLECT1) (Continued)

Bit	Name	Value	Description	Type	Reset Value
6	IDLIF_ARM		Enables local-bus, peripheral bridge, system DMA controller, and traffic controller of the MPU subsystem to enter idle mode whenever the MPU sets SET_IDLE bit or executes a wait-for-interrupt instruction:	R/W	0
		0	Clocks TIPB_CK, DMA_CK, and TC_CK remain active when the MPU enters idle mode.		
		1	Clocks TIPB_CK, DMA_CK, and TC_CK are stopped in conjunction with the MPU clock when idle mode is entered.		
5–3	RESERVED		Reserved. To prevent errant behavior, always write this bit as 0.	R/W	0
2	IDLPER_ARM		Selects idle entry mode for peripheral clock (MPUPER_CK)	R/W	0
		0	Clock remains active when MPU enters idle mode.		
		1	Clock stopped in conjunction with MPU idle mode entry.		
1	IDLXORP_ARM		Selects idle entry mode for 32-k or gp timer (MPU TIPB) and peripheral clock (MPUXOR_CK):	R/W	0
		0	OS timer and MPUXOR_CK clock remain active when MPU enters idle mode.		
		1	OS timer and MPUXOR_CK clock are stopped in conjunction with MPU clock when idle mode is entered.		
0	IDLWDT_ARM		Selects idle entry mode for internal timer/watchdog connected to MPU peripheral bus:	R/W	0
		0	Clock supplied to timer/watchdog remains active when MPU enters idle mode.		
		1	Timer/watchdog clock stopped in conjunction with MPU clock when idle mode is entered.		

Note: When the timer/watchdog is configured as watchdog timer, the clock is never shutdown, regardless of the value of the IDLWDT_ARM bit.

The MPU idle mode entry 2 register (ARM_IDLECT2) controls the clock domains independently of the MPU state.

Table 15–11. MPU Idle Mode Entry 2 Register (ARM_IDLECT2)

Bit	Name	Value	Description	Type	Reset Value
15–11	RESERVED		Reading these bits gives undefined values. Writing to them has no effect.		0x08
10	RESERVED		Reserved. This bit should always be written as 0.	R/W	0
9	EN_ GPIOCK		Enables clock of MPU GPIO connected to MPU TIPB:	R/W	0
		0	MPU GPIO clock stopped—bit must be set to logic 1 to enable clock activity		
		1	MPU GPIO clock active		
8	DMACK_REQ		Disables permanently-supplied-clock to system DMA controller to function on a clock request basis:	R/W	1
		0	DMA clock shutdown when idle mode is entered if IDLIF_ARM = 1		
		1	DMA clock stopped by default (reactivated upon DMA requests only)		
7	EN_ TIMCK		Enables clock of MPU timer connected to MPU TIPB:	R/W	0
		0	MPU timer clock is stopped—bit must be set to logic 1 to enable clock activity		
		1	MPU timer clock active and can be stopped depending on IDLTIM_ARM bit of MPU idle mode entry 1 register (ARM_IDLECT1)		
6	EN_ APICK		Enables clock of MPUI clock:	R/W	0
		0	MPUI clock stopped—bit must be set to logic 1 to enable clock activity		
		1	MPUI clock active		
5	RESERVED		Reserved. This bit should always be written as 0.	P R/W	0

Note: When the timer/watchdog is configured as watchdog timer, the clock is never shutdown, regardless the value of the IDLWDT_ARM bit or the EN_WDTCK bit.

Table 15–11. MPU Idle Mode Entry 2 Register (ARM_IDLECT2) (Continued)

Bit	Name	Value	Description	Type	Reset Value
4	EN_ LBCK		Enables clock of local bus clock:	P R/W	0
		0	Clock stopped—it must be set to logic 1 to enable clock activity.		
		1	Clock active		
3	EN_ LCDCK		Enables clock of LCD controller connected to MPU TIPB:	P R/W	0
		0	Clock stopped—bit must be set to logic 1 to enable clock activity		
		1	Clock active		
2	EN_PERCK		Enables peripheral clock (MPUPER_CK):	P R/W	0
		0	Clock stopped—bit must be set to logic 1 to authorize clock activity		
		1	Clock active and can be stopped depending on IDLTIM_ARM bit of MPU idle mode entry 1 register (ARM_IDLECT1)		
1	EN_XORPCK		Enables clock of OS timer connected to MPU TIPB and CLKIN reference peripheral clock (XORP_CK):	P R/W	0
		0	OS timer clock and external XORP_CK clock stopped—bit must be set to logic 1 to authorize clock activity		
		1	OS timer clock and external XORP_CK clock active and can be stopped depending on IDLXORP_ARM bit of MPU idle mode entry 1 register (ARM_IDLECT1)		
0	EN_WDTCK		Enables clock of timer/watchdog connected to MPU TIPB:	P R/W	0
		0	Clock stopped—bit must be set to logic 1 to authorize clock activity		
		1	Clock supplied to timer/watchdog is active and can be stopped depending on IDLWDT_ARM bit of MPU idle mode entry 1 register (ARM_IDLECT1)		

Note: When the timer/watchdog is configured as watchdog timer, the clock is never shutdown, regardless the value of the IDLWDT_ARM bit or the EN_WDTCK bit.

The MPU external wake-up register (ARM_EWUPCT) enables the WAKEUP signal and defines the delay from the external device to restore power with reference to the MPU clock restarting when the idle mode is exited.

Table 15–12. MPU External Wake-up Register (ARM_EWUPCT)

Bit	Name	Value	Description	Type	Reset Value
15–6	RESERVED		Reading these bits gives undefined values. Writing them has no effect.		
5	REPWR_EN		Enables external power control feature:	R/W	1
		0	$\overline{\text{FLASH.RP}}$ pin is set to logic low (Vol) when traffic controller (TC) is in idle mode.		
		1	$\overline{\text{FLASH.RP}}$ pin is not activated when TC idle mode is entered		
4–0	EXTPW(4:0)		Define delay from $\overline{\text{PWRON\_RESET}}$ pin going high to clocks restarting: Delay is calculated as follows: t_W (Wake-up time) = [EXTPWR(field value) ± 1] x *CLKIN* (period) With EXTPWR = 0 to 31	R/W	1

The MPU reset control 1 register (ARM_RSTCT1) initiates the software reset to the DSP and to the MPU.

Table 15–13. MPU Reset Control 1 Register (ARM_RSTCT1)

Bit	Name	Value	Description	Type	Reset Value
15–4	RESERVED		Reading these bits gives undefined values. Writing to them has no effect.		
3	SW_RST		Resets both DSP, MPU, and peripherals (bit is always read 0):	R/W	0
		0	DSP, MPU, and peripheral clock domain enabled		
		1	DSP, MPU, and peripherals reset—once set to logic 1 by MPU processor, this bit returns to logic 0 once reset completes.		
2	DSP_RST		Resets priority registers (TIPB module), EMIF configuration registers, and MPUI control logic (partially) in DSP. This bit is set by external reset pins and is released by writing a logic 1 in register (use for MPUI boot).	R/W	0
		0	Priority, EMIF configuration registers, and MPUI are reset.		
		1	Priority and EMIF configuration registers can be programmed.		
1	DSP_EN		Resets DSP:	R/W	0
		0	Resets DSP, excluding configuration setting, and maintains reset state as long as this bit is asserted low		
		1	Enables DSP—after global reset, bit must be set to a logical 1 to enable DSP.		
0	ARM_RST		Resets MPU (bit is always read 0):	R/W	0
		0	MPU clock domain enabled		
		1	MPU reset—once set to logic 1 by MPU processor, bit returns to logic 0 on next cycles.		

Note: Writing the DSP_EN bit to 0 and ARM_RST bit to 1 together initiate a global software reset.

The MPU reset control 2 register (ARM_RSTCT2) sets the PER_EN signal that resets peripherals attached to the MPU.

Table 15–14. MPU Reset Control 2 Register (ARM_RSTCT2)

Bit	Name	Value	Description	Type	Reset Value
15–1	RESERVED		Reading these bits gives undefined values. Writing to them has no effect.		
0	PER_EN		Controls MPUPER_nRST signal used to reset and/or enable peripherals connected to MPU TIPB:	R/W	0
		0	ARMPER_nRST signal active		
		1	ARMPER_nRST signal inactive		

The MPU system status register (ARM_SYSST) contains the system information such as processor state, chip configuration, and reset status flags.

Table 15–15. MPU System Status Register (ARM_SYSST)

Bit	Name	Value	Description	Type	Reset Value
15–14	RESERVED		Reading these bits gives undefined values. Writing to them has no effect.		
13–11	CLOCK_SELECT (2–0)		The CLOCK_SELECT bits indicate the current clocking scheme selection: the application can switch OMAP5910 clocking scheme by writing to these bits. These bits are at logic 0 after reset (select fully synchronous clocking scheme) (see Table 15–16).	R/W	0
10–7	RESERVED				
6	IDLE_DSP		Indicates DSP state:	R	0
		0	DSP active		
		1	DSP in global idle state		
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 TIPB interface:	R/C	0
		0	No power-on reset detected		
		1	A power-on reset occurred		

Table 15–15. MPU System Status Register (ARM_SYSST) (Continued)

Bit	Name	Value	Description	Type	Reset Value
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 TIPB:	R/C	0
		0	No external reset detected		
		1	An external reset asserted		
3	ARM_MCRST		Indicates whether or not an MPU reset has occurred. This bit is cleared to 0 upon a reset pulse asserted at CHIP_nRESET signal, or by writing to it a logic 0. This bit cannot be written to logic 1 from TIPB interface:	R/C	0
		0	MPU has not been reset.		
		1	MPU has been reset.		
2	ARM_WDRST		Indicates whether or not reset has been asserted due to an MPU timer/watchdog underflow. This bit is cleared to logic 0 upon a reset pulse asserted at CHIP_nRESET signal, or by writing it to logic 0. This bit cannot be written to logic 1 from peripheral bus interface:	R/C	0
		0	MPU timer/watchdog underflow has not occurred.		
		1	MPU timer/watchdog underflow has generated reset.		
1	GLOB_SWRST		Indicates whether or not reset has been asserted due to a global software reset (DSP_EN and ARM_RST set together). This bit is cleared to logic 0 upon an external reset pulse asserting at CHIP_nRESET signal, or by writing it to logic 0. This bit cannot be written to logic 1 from peripheral bus interface:	R/C	0
		0	A global software reset has not been requested.		
		1	A global software reset has been requested.		

Table 15–15. MPU System Status Register (ARM_SYSST) (Continued)

Bit	Name	Value	Description	Type	Reset Value
0	DSP_WDRST		Indicates whether or not reset has been asserted due to a DSPs timer/watchdog underflow. This bit is cleared to logic 0 upon an reset pulse asserting at CHIP_nRESET signal, or by writing it to logic 0. This bit cannot be written to logic 1 from peripheral bus interface:	R/C	0
		0	DSP timer/watchdog underflow has not occurred.		
		1	DSP timer/watchdog underflow has generated reset.		

Table 15–16 lists the clocking schemes for the MPU system status register (ARM_SYSST).

Table 15–16. Clocking Schemes for OMAP5910

CLOCK_SELECT (2)	CLOCK_SELECT (1)	CLOCK_SELECT (0)	CLOCK SCHEME
0	0	0	Fully synchronous
0	0	1	Reserved
0	1	0	Synchronous scalable
0	1	1	Reserved
1	x	x	Reserved

DSP Base word address: 0x4000 – Bit Width: 16

Table 15–17. DSP Clock/Reset/Power Mode Control Registers

Register Name	Descriptions	R/W	Size	DSP Address	MPU Address	Reset Value
DSP_IDLECT1	DSP idle select 1	R/W	16 bits	00:4002	E100:8004	0x0000
DSP _IDLECT2	DSP idle select 2	R/W	16 bits	00:4004	E100:8008	0x0000
Reserved				00:4006		0x0000
Reserved				00:4008		0x0000
DSP _RSTCT2	DSP reset control	R/W	16 bits	00:400A	E100:8014	0x0000
DSP _SYSST	DSP system status	R/W	16 bits	00:400C	E100:8018	0x0000
Reserved				00:400E		0x0000
Reserved				00:4010		0x0000

The DSP domain peripheral clock setup and the external module reset functions are controlled by the DSP through these registers.

The DSP clock control register (DSP_CKCTL) defines the frequency selection for the DSP_GPIO_CK and the DSPTIM_CK.

Table 15–18. DSP Clock Control Register (DSP_CKCTL) – Offset Address: 0x00

Bit	Name	Value	Description	Type	Reset Value
15–9	RESERVED		Reading these bits gives an undefined value. Writing to them has no effect.		
8	TIMXO		Selects either a CK_GEN2 frequency clock or input reference clock (CLKIN) to supply DSP timer peripherals.	R/W	1
		0	The DSPTIM_CK clock frequency is the input reference clock.		
		1	DSPTIM_CK frequency is issued from CK_GEN2/2.		

Table 15–18. DSP Clock Control Register (DSP_CKCTL) – Offset Address: 0x00 (Continued)

Bit	Name	Value	Description	Type	Reset Value
7	GPIOXO		Selects either a subfrequency issued from CK_GEN2 or input reference clock (CLKIN) to supply GPIO peripheral.	R/W	1
		0	The DSP_GPIO_CK clock frequency is the input reference clock.		
		1	DSP_GPIO_CK frequency is issued from CK_GEN2 and defined by the GPIODIV field value.		
6–5	GPIODIV(1:0)		These read/write bits define prescaler value from CK_GEN2 to the GPIO clock signal (GPIO_CK).	R/W	0
4–0	RESERVED		Reserved. These bits should always be written as 0.		

Table 15–19 lists the selection for the DSP_GPIO_CK (GPIOXO = 1).

Table 15–19. GPIO_CK Selections

GPIODIV(1)	GPIODIV(0)	DSP_GPIO_CK Frequency
0	0	CK_GEN2/1
0	1	CK_GEN2/2
1	0	CK_GEN2/4
1	1	CK_GEN2/8

The DSP idle mode entry 1 register (DSP_IDLECT1) enables and defines the idle mode entry/exit to each clock domain.

Table 15–20. DSP Idle Mode Entry 1 Register (DSP_IDLECT1) – Offset Address: 0x04

Bit	Name	Value	Description	Type	Reset Value
15–9	RESERVED		Reading these bits gives undefined values. Writing to them has no effect.		
8	IDLTIM_DSP		Selects idle entry mode for internal DSP timer clock (DSPTIM_CK).	R/W	0
		0	The DSPTIM_CK clock remains active when DSP enters idle mode.		
		1	The DSPTIM_CK clock is stopped in conjunction with DSP clock when idle mode is set (DSP_IDLE signal asserted high).		
7	RESERVED			R/W	0
6	RESERVED		Reserved. To prevent errant behavior, this bit should always be written as 1.	R/W	1
5–2	RESERVED		Reserved. To prevent errant behavior, these bits should always be written as o.	R/W	0
1	IDLXORP_DSP		Selects idle entry mode for reference peripheral clock (DSPXOR_CK).	R/W	0
		0	The DSPXOR_CK clock remains active when DSP enters idle mode.		
		1	The DSPXOR_CK clock is stopped in conjunction with DSP clock when idle mode is entered.		
0	IDLWDT_DSP		Selects idle entry mode for timer/watchdog connected to DSP TIPB.	R/W	0
		0	The clock supplied to timer/watchdog remains active when the DSP enters idle mode.		
		1	The timer/watchdog clock is stopped in conjunction with the DSP clock when idle mode is entered.		

Note: When the timer/watchdog is configured as watchdog timer, the clock is never shutdown, regardless of the value of the IDLWDT_DSP bit.

The DSP idle mode entry 2 register (DSP_IDLECT2) disables the clock domains individually and independently of the DSP state.

Table 15–21. DSP Idle Mode Entry 2 Register (DSP_IDLECT2) – Offset Address: 0x08

Bit	Name	Value	Description	Type	Reset Value
15–6	RESERVED		Reading these bits gives undefined values. Writing to them has no effect.		
5	EN_TIMCK		Enables DSP timer clock (DSPTIM_CK)	R/W	0
		0	DSPTIM_CK clock is stopped. This bit must be set to logic 1 to enable clock activity.		
		1	DSPTIM_CK clock is active and can be stopped, depending on DSP_IDLECT1 IDLTIM_DSP bit.		
4	EN_GPIOCK		Enables GPIO peripheral clock (GPIO_CK).	R/W	0
		0	GPIO_CK clock is stopped. The bit must be set to logic 1 to enable clock activity.		
		1	GPIO_CK clock is active. The GPIO_CK clock must be disabled by setting EN_GPIOCK = 0 before sleep modes can be entered.		
3–2	RESERVED		Reserved. These bits should always be written as 0.		
1	EN_XORPCK		Enables DSPXOR_CK reference clock.	R/W	0
		0	The DSPXOR_CK clock is stopped. The bit must be set to logic 1 to enable clock activity.		
		1	The DSPXOR_CK clock is active and can be stopped, depending on DSP_IDLECT1 IDLXORP_DSP bit.		
0	EN_WDTCK		Enables DSPWDT_CK reference clock.	R/W	0
		0	The DSPWDT_CK clock is stopped. The bit must be set to logic 1 to enable clock activity.		
		1	The DSPWDT_CK clock is active and can be stopped, depending on DSP_IDLECT1 IDLWDT_DSP bit.		

The DSP reset control 2 register (DSP_RSTCT2) sets the PER_EN signal that resets peripherals attached to the DSP.

Table 15–22. DSP Reset Control 2 Register (DSP_RSTCT2) – Offset Address: 0x14

Bit	Name	Value	Description	Type	Reset Value
15–1	RESERVED		Reading these bits gives undefined values. Writing to them has no effect.		
0	PER_EN		This read/write bit controls the DSPPER_nRST signal that can be used to reset and/or enable peripherals connected to the DSP.	R/W	0
		0	Writing a logic 0 sets DSPPER_nRST signal to active.		
		1	Writing a logic 1 sets DSPPER_nRST signal to inactive.		

The DSP system status register (DSP_SYSST) contains the system information.

Table 15–23. DSP System Status Register (DSP_SYSST) – Offset Address: 0x18

Bit	Name	Value	Description	Type	Reset Value
15–14	RESERVED		Reading these bits gives undefined values. Writing to them has no effect.		
13–11	CLOCK_SELECT (2–0)		These read-only bits reflect CLOCK_SELECT and indicate current clocking scheme selection.	R	0
10–7	RESERVED				
6	IDLE_ARM		This read-only bit indicates MPU state.	R	0
		0	MPU active		
		1	MPU in idle state		

Notes: 1) This bit is only to be used for test/debug purposes only.

2) In the OMAP5910 device, the DSP_EN and ARM_RST bits (located in ARM_RSTCT1) must be set together to activate the global software reset. Setting the SW_RST bit only (DSP_RSTCT1) results in global software reset flag.

Table 15–23. DSP System Status Register (DSP_SYSST) – Offset Address: 0x18 (Continued)

Bit	Name	Value	Description	Type	Reset Value
5	POR		This read/clear-only status bit 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 peripheral bus interface.	R/C	0
		0	No power-on reset has been detected.		
		1	A power-on reset has occurred.		
4	EXT_RST		This read/clear-only status bit 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 TIPB interface.	R/C	0
		0	No external reset detected.		
		1	An external reset has been asserted.		
3	DSP_ARM_RST		This read/write bit is used by DSP to hold MPU in reset.	R/C	0
		0	The MPU is enabled. This is default value after reset.		
		1	Reset MPU		
2	ARM_WDRST		This read/clear-only status bit indicates whether or not reset has been asserted due to a MPU timer/watchdog underflow. This bit is cleared to logic 0 upon an external reset pulse asserting at CHIP_nRESET signal, or by writing it to logic 0. This bit cannot be written to logic 1 from peripheral bus interface.	R/C	0
		0	MPU timer/watchdog underflow has not occurred.		
		1	MPU timer/watchdog underflow has generated reset.		

Notes: 1) This bit is only to be used for test/debug purposes only.

2) In the OMAP5910 device, the DSP_EN and ARM_RST bits (located in ARM_RSTCT1) must be set together to activate the global software reset. Setting the SW_RST bit only (DSP_RSTCT1) results in global software reset flag.

Table 15–23. DSP System Status Register (DSP_SYSST) – Offset Address: 0x18 (Continued)

Bit	Name	Value	Description	Type	Reset Value
1	GLOB_SWRST		This read/clear-only status bit indicates whether or not reset has been asserted due to a global software reset. This bit is cleared to logic 0 upon an external reset pulse asserting at CHIP_nRESET signal, or by writing it to logic 0. This bit cannot be written to logic 1 from TIPB interface. See Note 2.	R/C	0
		0	A global software reset has not been requested.		
		1	A global software reset has been requested.		
0	DSP_WDRST		This read/clear-only status bit indicates whether or not reset has been asserted due to a DSP timer/watchdog underflow. This bit is cleared to logic 0 upon an external reset pulse at CHIP_nRESET signal or by writing it to logic 0. This bit cannot be written to logic 1 from peripheral bus interface.	R/C	0
		0	DSP timer/watchdog underflow has not occurred.		
		1	DSP timer/watchdog underflow has generated reset.		

Notes: 1) This bit is only to be used for test/debug purposes only.

2) In the OMAP5910 device, the DSP_EN and ARM_RST bits (located in ARM_RSTCT1) must be set together to activate the global software reset. Setting the SW_RST bit only (DSP_RSTCT1) results in global software reset flag.

15.4.1 DPLL Operation Mode Registers

The digital phase-locked loop (DPLL) can be operated either in bypass mode, in lock mode, or in idle mode.

- In lock mode, the DPLL synthesizes a frequency clock (CLKOUT) from a fixed reference clock signal (CLKREF). The output frequency is an integer multiple or fractional multiple (m/n, respectively PLL_MULT and PLL_DIV bit field) of the input reference. With $1 < m < 31$ and $1 < n < 4$, the frequency output ranges from CLKREF /4 to 31x CLKREF.
- In bypass mode, the DPLL output clock in bypass mode can be CLKREF (input clock), CLKREF/2, or CLKREF/4 depending on the BYPASS_DIV bit field value.
- In idle mode, the DPLL circuitry is disabled. The output clock is held in a high static level and the configuration data is maintained.

The mode (bypass or lock) and the frequency selections are defined via memory mapped control register bits. Programming is performed through the TIPB and is initiated the MPU.

The idle mode is entered upon an asynchronous idle signal request (idle).

Note:

The DPLL idle mode entry/exit timing is dependent upon the input/output frequency ratio selection.

The input reference clock signal must be active for (at least) 24 input clock cycles from the idle request (idle rising edge) before the DPLL idle setup has completed (idle_ack high).

Once in idle mode, the clock source can be stopped (either in a high or a low level), but the reference clock source must be restarted before releasing the idle mode.

When the idle mode is exited, the DPLL is set in bypass mode and the output clock signal is valid after a maximum of 10 input reference clock cycles. If the DPLL was synthesizing a frequency prior to enter the idle state, then the DPLL switches from the bypass mode (frequency set /BYPASS_DIV) to the synthesizer mode (frequency set/PLL_MULT and PLL_DIV) when the lock state is reacquired.

Table 15–24 lists the DPLL control registers. Table 15–25 describes the register bits.

The MPU base addresses for the DPLL control registers are:

DPLL1: 0xFFFE:CF00
Bit width: 16 bits

Writing to the control register (CTL_REG) 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 being able to change the multiple or divide values without reentering the DPLL lock sequence.

Table 15–24. DPLL Control Registers

Register Name	Descriptions	R/W	Size	Offset	Reset Value
CTL_REG	DPLL control register	R/W	16 bits	x00	0x2002

Table 15–25. DPLL Control Register (CTL_REG)

Bit	Name	Description	Type	Reset Value
15–14	RESERVED	This bit is reserved and set to 0.		
13	IOB	Initialize on break. When high, DPLL switches to bypass mode and starts a new locking sequence if DPLL core indicates that it lost lock. When set low, DPLL continues to output synthesized clock even if core indicates it has lost lock, but BREAKLN is active low. The power-on value is 1.		
12	RESERVED	This bit is reserved and set to 0.		
11–7	PLL_MULT(4–0)	The DPLL multiply value. The maximum clock out frequency is 31*CLKREF.		
6–5	PLL_DIV(1:0)	The DPLL divide value. PLL_DIV(1:0) = 00: CLKOUT = CLKREF 01: CLKOUT = CLKREF/2 10: CLKOUT = CLKREF/3 11: CLKOUT = CLKREF/4 The minimum CLKOUT frequency is 0.25 * CLKREF. When PLL_MULT(4:0) is equal to 0 or 1, CLKOUT is not synthesized by DPLL but by simply a divided down version of CLKREF. Affects lock mode only.		
4	PLL_ENABLE	Setting PLL_ENABLE bit to 1 requests DPLL to enter LOCK mode. It enters LOCK mode only after it has synthesized desired frequency. Clearing bit to 0 causes DPLL to switch back to bypass mode.		
3–2	BYPASS_DIV(1:0)	Determines clkout frequency when in BYPASS mode. BYPASS_DIV(1:0) = 00: CLKOUT = CLKREF 01: CLKOUT = CLKREF/2 1X: CLKOUT = CLKREF/4		
1	BREAKLN	When BREAKLN = 0, DPLL has broken lock for some unknown reason. If and when lock condition is restored or a write to control register occurs, BREAKLN returns to 1.		
0	LOCK	When LOCK = 1, DPLL is in lock mode and clkout is desired synthesized clock frequency. When LOCK = 0, DPLL is in bypass mode and clkout contains a divided down output clock. If PLL_MULT = 0 or 1, oscillators in DPLL_core are not activated and duty cycle is not ensured.		

Table 15–26 lists the ULPD registers. Table 15–27 through Table 15–41 describe the register bits.

Bit Width: 32

Table 15–26. ULPD Registers – MPU Base Address: FFFE:0800

Name	Descriptions	R/W	Size	Offset	Reset Value
COUNTER_32_LSB	Lower value of number of ticks from 32-kHz clock	R	16 bits	x00	0x0001
COUNTER_32_MSB	Upper value of number of ticks from 32-kHz clock	R	16 bits	x04	0x0000
COUNTER_HIGH_FREQ_LSB	Lower value of number of ticks from high frequency clock	R	16 bits	x08	0x0001
COUNTER_ HIGH_FREQ _MSB	Upper value of number of ticks from high frequency clock	R	16 bits	x0C	0x0000
GAUGING_CTRL_REG	Drives gauging functionality	R/W	16 bits	0x10	0x0000
IT_STATUS_REG	Interrupt status register	R	16 bits	0x14	0x0000
Reserved			8 bits	0x18	0x01
Reserved			8 bits	0x1C	0x01
Reserved			8 bits	0x20	0x01
SETUP_ANALOG_CELL3_ULPD1_REG	Number of 32-kHz clocks to wake up	R/W	16 bits	0x24	0x03FF
Reserved			8 bits	0x2C	0x01
Reserved			8 bits	0x28	0x01
CLOCK_CTRL_REG	Manages clock output and inactive values	R/W	16 bits	0x30	0x0000
SOFT_REQ_REG	Manage software clock requests	R/W	16 bits	X34	0x0000
COUNTER_32_FIQ_REG	Number of 32-kHz clocks to delay active modem shut down signal after receiving an active EXT_FIQ signal	R/W	16 bits	X38	0x0001
DPLL_CTRL_REG	48-MHz DPLL	R/W	16 bits	X3C	0x2211
STATUS_REQ_REG	Status of hardware requests	R	16 bits	0x40	U

Table 15–26. ULPD Registers – MPU Base Address: FFFE:0800 (Continued)

Name	Descriptions	R/W	Size	Offset	Reset Value
LOCK_TIME_REGISTER	Defines lock time when APLL is selected	R/W	16 bits	0x48	0x960
APLL_CTRL_REG	This register allows switch between APLL and DPLL and controls all input of APLL.	R/W	16 bits	0x4C	U
POWER_CTRL_REG	Power control register	R/W	16 bits	0x50	0x8

The counter 32 LSB register (COUNTER_32_LSB_REG) represents the lower value of the number of ticks from the 32-kHz clock during gauging time.

Table 15–27. Counter 32 LSB Register (COUNTER_32_LSB_REG)

Bit	Name	Type	Reset Value
15–0	COUNTER_32_LSB	R	0x0001

The counter 32 MSB register (COUNTER_32_MSB_REG) represents the upper value of the number of ticks from the 32-kHz clock during gauging time.

Table 15–28. Counter 32 MSB Register (COUNTER_32_MSB_REG)

Bit	Name	Type	Reset Value
15–0	COUNTER_32_LSB	R	0x0000

The counter high frequency LSB register (COUNTER_HIGH_FREQ_LSB_REG) represents the lower value of the number of ticks of the high-frequency clock during gauging time.

Table 15–29. Counter High Frequency LSB Register (COUNTER_HIGH_FREQ_LSB_REG)

Bit	Name	Type	Reset Value
15–0	COUNTER_HIGH_FREQ_LSB	R	0x0001

The counter high frequency MSB register (COUNTER_HIGH_FREQ_MSB_REG) represents the upper value of the number of ticks from the high-frequency clock during gauging time.

Table 15–30. Counter High Frequency MSB Register (COUNTER_HIGH_FREQ_MSB_REG)

Bit	Name	Type	Reset Value
15–0	COUNTER_HIGH_FREQ_LSB	R	0x0000

The gauging control register (GAUGING_CTRL_REG) controls the gauging activity. It start/stops it and selects the clock used as the high-frequency clock.

Table 15–31. Gauging Control Register (GAUGING_CTRL_REG)

Bit	Name	Value	Description	Type	Reset Value
1	SELECT_HI_ FREQ_CLOCK	0	Use 12-MHz clock for high frequency clock	R/W	0
		1	Reserved. Do not use this setting.		
0	GAUGING_EN	0	Stop gauging	R/W	0
		1	Enable gauging		

The setup analog cell3 ULPD1 register (SETUP_ANALOG_CELL3_ULPD1_REG) provides the number of 32-kHz clock periods until the ULPD wakes up the device when a wake-up request is made. See Figure 15–14 (page 15-37).

Table 15–32. Setup Analog Cell3 ULPD1 Register (SETUP_ANALOG_CELL3_ULPD1_REG)

Bit	Name	Type	Reset Value
15–0	SETUP_ANALOG_CELL3	R/W	0x3FF

The interrupt status register (IT_STATUS_REG) provides the status and source of ULPD interrupts.

Table 15–33. Interrupt Status Register (IT_STATUS_REG)

Bit	Name	Description	Type	Reset Value
3	IT_WAKEUP_USB	Wake-up interrupt from USB function is shared with ULPD gauging interrupt (MPU level 2 interrupt IRQ_24).	R	0x0
2	OVERFLOW_32	Overflow occurred on 32-kHz counter during gauging.	R	0x0
1	OVER-FLOW_HI_FREQ	Overflow occurred in high-frequency counter during gauging versus high frequency clock.	R	0x0
0	IT_GAUGING	End of gauging interrupt. Informs the MPU that gauging is stopped and that it can read value of high and low frequency counters.	R	0x0

The clock control register (CLOCK_CTRL_REG) manages clock output and inactive values.

Table 15–34. Clock Control Register (CLOCK_CTRL_REG)

Bit	Name	Value	Description	Type	Reset Value
5	DIS_USB_PVCI_ CLK	0	Enables USB function clock for FAC counter	R/W	0x0
		1	Disables USB function clock for FAC counter		
4	USB_MCLK_EN	0	Disables USB.CLKO	R/W	0
		1	Enables USB.CLO		
3	RESERVED		Reserved. This bit should always be written as 0.	R/W	0x0
2	SDW_MCLK_INV	0	BCLK is low when inactive.	R/W	0
		1	BCLK is high when inactive.		
1	COM_MCLK_ INV	0	MCLK is low when inactive.	R/W	0
		1	MCLK is high when inactive		
0	MODEM_32K_ EN	0	Disables 32-kHz on UART clock	R/W	0
		1	Enables 32-kHz on UART clock		

The software clock request register (SOFT_REQ_REG) manages software clock requests.

Table 15–35. Software Clock Request Register (SOFT_REQ_REG)

Bit	Name	Value	Description	Type	Reset Value
4	USB_REQ_EN	0	Disables USB function hardware DPLL request	R/W	0x1
		1	Enables USB function hardware DPLL request		
3	SOFT_USB_REQ	0	No software request for clocking on USB.CLK0	R/W	0
		1	Software request for clocking on USB.CLK0		
2	SOFT_SDW_REQ	0	No software request for clocking on BCLK	R/W	0
		1	Software request for clocking on BCLK		
1	SOFT_COM_REQ	0	No software request for clocking on MCLK	R/W	0
		1	Software request for clocking on MCLK		
0	SOFT_DPLL_REQ	0	Software request for clocking on 48-MHz DPLL	R/W	0
		1	Software request for clocking on 48-MHz DPLL (except no software request possible when PLL_ENABLE = 0 in DPLL_CTRL_REG)		

The counter 32 FIQ register (COUNTER_32_FIQ_REG) represents the number of 32-kHz clocks to delay before activating $\overline{\text{RST\_HOST\_OUT}}$ after receiving an active BFAIL/$\overline{\text{EXT\_FIQ}}$ signal.

Table 15–36. Counter 32 FIQ Register (COUNTER_32_FIQ_REG)

Bit	Name	Type	Reset Value
7–0	COUNTER_32_FIQ	R/W	0x01

This is the control register for the 48-MHz DPLL.

The reset multiply and divide settings are fixed for x4 operation to achieve a 48-mHz clock. The DPLL must always remain enabled when it is used to generate the 48-mHz clock for USB, UARTs, or other peripherals

Writing to the DPLL control register (DPLL_CTRL_REG) causes the DPLL to switch immediately 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 being able to change the multiple or divide values without reentering the DPLL lock sequence.

Table 15–37. DPLL Control Register (DPLL_CTRL_REG)

Bit	Name	Description	Type	Reset Value
15–14	RESERVED	This bit is reserved and set to 0.		
13	IOB	Initialize on break. When high, DPLL switches to bypass mode and starts a new locking sequence if DPLL core ever indicates that it lost lock. When set low, DPLL continues to output synthesized clock, even if core indicates it has lost lock, but BREAKLN is active low. The power-on value is 1.	R/W	1
12	RESERVED	This bit is reserved and set to 0.		
11–7	PLL_MULT(4:0)	This bit is reserved and always written to its reset value.	R/W	0x0
6–5	PLL_DIV(1:0)	This bit is reserved and always written to its reset value.	R/W	0x0
4	PLL_ENABLE	Setting PLL_ENABLE bit to 1 requests DPLL to enter LOCK mode. It enters lock mode only after it has synthesized desired frequency. Clearing bit to 0 causes DPLL to switch back to the bypass mode.	R/W	1
3–2	BYPASS_DIV(1:0)	This bit is reserved and always written to its reset value.	R/W	0x0
1	BREAKLN	When BREAKLN = 0 , DPLL has broken lock for some unknown reason. If and when lock condition is restored or a write to control register occurs, BREAKLN returns to 1.	R	0x1
0	LOCK	When LOCK = 1, DPLL is in lock mode and clkout is desired synthesized clock frequency. When LOCK = 0, DPLL is in bypass mode and clkout contains a divided down output clock.	R	0x0

The status request register (STATUS_REQ_REG) indicates the status of hardware requests. The reset value of these registers depends on what requests are being made.

Table 15–38. Status Request Register (STATUS_REQ_REG)

Bit	Name	Value	Description	Type	Reset Value
13	MODEM_ NSHUTDOWN		Status of $\overline{\text{RST\_HOST\_OUT}}$ output pin	R	
12	MMC_DPLL_REQ		DPLL wake-up request from MMC	R	
11	UART3_DPLL_REQ		DPLL wake-up request for UART3	R	
10	UART2_DPLL_REQ		DPLL wake-up request for UART2	R	
9	UART1_DPLL_REQ		DPLL wake-up request for UART1	R	
8	USB_HOST_ DPLL_REQ		12-MHz clock for DPLL wake-up requested by USB host	R	
7	CAM_DPLL_ MCLK_REQ	0	No request for 48-MHz DPLL wake-up by camera interface	R	
		1	Indicates request for 48-MHz DPLL wake-up by camera interface		
6	USB_DPLL_ MCLK_REQ	0	No request for 48-MHz DPLL wake-up by USB interface	R	
		1	Indicates request for 48-MHz DPLL wake-up by USB interface		
5	USB_ MCLK_REQ	0	No clock request by USB host	R	
		1	Indicates clock request by USB host		
4	SDW_ MCLK_REQ	0	No clock request by BCLKREQ	R	
		1	Indicates clock request by BCLKREQ		
3	COM_ MCLK_REQ	0	No clock request by MCLKREQ	R	
		1	Indicates clock request by MCLKREQ		
2	PERIPH_nREQ		Indicates status of internal peripheral clock request signal. This is an active-low signal.	R	
1	WAKEUP_nREQ		Indicates status of internal WAKEUP_nREQ signal. This is an active-low signal.	R	
0	CHIP_IDLE		Indicates status of internal CHIP_IDLE signal	R	

The lock time register (LOCK_TIME) allows fixing the lock time when the APLL is used. It represents the number of CLK (12-MHz or 13-MHz) periods required to activate the APLL lock.

Table 15–39. Lock Time Register (LOCK_TIME)

Bit	Name	Description	Type	Reset Value
15–0	LOCK_TIME	Indicates number of CLK (12-MHz or 13-MHz) periods to wait for activate lock when APLL is used. The reset value corresponds at a lock of 200 µs for a 12-MHz CLK.	R/W	0x960

APLL control register (APLL_CTRL_REG) allows the switch between the APLL and the DPLL. It controls all the input of the APLL.

Table 15–40. APLL Control Register (APLL_CTRL_REG)

Bit	Name	Function	R/W	Reset Value
15:4	Reserved	Reserved. These bits should always be written as 0.	R	0xx
3	SEL	Bit used to select correct divider so the APLL can generate a 48-mHz clock from either a 12-mHz or 13-mHz reference source. Bit defaults to the 12-mHz reference setting. 0: Divide by 13 provides 1-mHz clock to APLL for 48-mHz generation (if reference clock is 13-mHz) 1: Divide by 12 provides 1-mHz clock to APLL for 48-mHz generation (if reference clock is 12-mHz)	R/W	0x1
2-1	RESERVED	Reserved. These bits should always be written as 0.	R/W	0x0
0	APLL_NDPLL_SWITCH	It allows switch between APLL and DPLL. When 0, use DPLL; when 1, use APLL. By default, use DPLL.	R/W	0x0

Table 15–41. Power Control Register (POWER_CTRL_REG)

Bit	Name	Value	Description	Type	Reset Value
15–4	RESERVED		Reserved	R	Unknown
3	SW_NSHUTDOWN		Software generation of $\overline{\text{RST\_HOST\_OUT}}$. This bit controls the state of $\overline{\text{RST\_HOST\_OUT}}$ pin when SW_RST bit is 1.	R/W	0x1
		0	$\overline{\text{RST\_HOST\_OUT}}$ is active low		
		1	$\overline{\text{RST\_HOST\_OUT}}$ is inactive high		
2	SW_RST		Released hardware generation of $\overline{\text{RST\_HOST\_OUT}}$	R/W	0x0
		0	State of $\overline{\text{RST\_HOST\_OUT}}$ pin depends on BFAIL/EXT_FIQ and 32k counter.		
		1	State of $\overline{\text{RST\_HOST\_OUT}}$ pin is equal to level of SW_NSHUTDOWN bit.		
1	LOW_PWR_REQ		Low power software request. When this bit and the LOW_PWR_EN bit are high, the LOW_PWR pin is driven active high.	R/W	0x0
0	LOW_PWR_EN		Low power enable bit. Disable by default. This bit enables the usage of the LOW_PWR output pin.	R/W	0x0
		0	During deep sleep, or if LOW_PWR_REQ is high, the LOW_PWR pin is driven active high.		
		1	LOW_PWR pin is always driven low.		

Table 15–42 lists the DSP idle registers. Table 15–43 and Table 15–44 describe the register bits.

Table 15–42. DSP Idle Registers

Register Name	Descriptions	R/W	Size	Offset	Reset Value
ICR	DSP idle configuration register	R/W	16 bits	x01	0x0000
ISR	DSP idle status register	R	16 bits	x02	0x0000

The DSP idle configuration register (ICR) indicates the DSP subdomains that are placed in idle mode.

Table 15–43. DSP Idle Configuration Register (ICR)

Bit	Name	Value	Description	Type	Reset Value
15–6	RESERVED				0
5	EMIF_IDLE_DOMAIN	0	No request to idle DSP EMIF	R/W	0
		1	Request to place DSP EMIF in idle		
4	DPLL_IDLE_DOMAIN	0	No request to idle DSP DPLL	R/W	0
		1	Request to place DSP DPLL in idle		
3	PER_IDLE_DOMAIN	0	No request to idle DSP peripherals	R/W	0
		1	Request to place DSP peripherals in idle		
2	CACHE_IDLE_DOMAIN	0	No request to idle DSP I-cache	R/W	0
		1	Request to place DSP I-cache in idle		
1	DMA_IDLE_DOMAIN	0	No request to idle DSP DMA controller	R/W	0
		1	Request to place DSP DMA controller in idle		
0	CPU_IDLE_DOMAIN	0	No request to idle DSP core and memory	R/W	0
		1	Request to place DSP core, SARAM, and DARAM in idle		

The DSP idle status register (ISR) indicates the DSP subdomains that have been placed in idle mode.

Table 15–44. DSP Idle Status Register (ISR)

Bit	Name	Value	Description	Type	Reset Value
15–6	RESERVED				
5	EMIF_IDLE_STATUS	0	DSP EMIF not in idle	R	0
		1	DSP EMIF in idle		
4	DPLL_IDLE_STATUS	0	DSP DPLL not in idle	R	0
		1	DSP DPLL in idle		
3	PER_IDLE_STATUS		DSP peripherals not in idle	R	0
		1	DSP peripherals in idle		
2	CACHE_IDLE_STATUS	0	No request to idle DSP I-cache not in idle	R	0
		1	DSP I-cache in idle		
1	DMA_IDLE_STATUS	0	DSP DMA controller not in idle	R	0
		1	DSP DMA controller in idle		
0	CPU_IDLE_STATUS	0	DSP core and memory not in idle	R	0
		1	DSP core, SARAM and DARAM in idle		

Appendix A

Input/Output Descriptions

This appendix describes OMAP5910 inputs and outputs (I/O) and OMAP5910 functional multiplexing, which include:

- I/O signals ordered by their functions (Table A–1)
- Functional multiplexing control bits for each ball (Table A–2)

Consult the OMAP5910 Data Manual (literature number SPRS197) for additional information, including I/O pad reset status, buffer types and boundary scan, pullup/pulldown and gating/inhibiting information.

Topic **Page**

A.1 I/O Signals

Table A–1 identifies the input and output signals for the OMAP5910 device.

Some signals are available on multiple pins via pin multiplexing configuration settings.

Table A–1. Input and Output Signals for the OMAP5910 Device

Signals	Description	Ballout
FLASH		
$\overline{\text{FLASH.WP}}$	FLASH write protect	V4
$\overline{\text{FLASH.WE}}$	FLASH write enable	W2
$\overline{\text{FLASH.RP}}$	FLASH power down for TI/reset for Intel	W1
$\overline{\text{FLASH.OE}}$	FLASH output enable	U4
FLASH.D[15]	FLASH data bit 15	V3
FLASH.D[14]	FLASH data bit 14	T4
FLASH.D[13]	FLASH data bit 13	U3
FLASH.D[12]	FLASH data bit 12	U1
FLASH.D[11]	FLASH data bit 11	P8
FLASH.D[10]	FLASH data bit 10	T3
FLASH.D[9]	FLASH data bit 9	T2
FLASH.D[8]	FLASH data bit 8	R4
FLASH.D[7]	FLASH data bit 7	R3
FLASH.D[6]	FLASH data bit 6	R2
FLASH.D[5]	FLASH data bit 5	P7
FLASH.D[4]	FLASH data bit 4	P4
FLASH.D[3]	FLASH data bit 3	P2
FLASH.D[2]	FLASH data bit 2	N7
FLASH.D[1]	FLASH data bit 1	N2
FLASH.D[0]	FLASH data bit 0	N4
FLASH.CLK	FLASH clock	N3
$\overline{\text{FLASH.CS3}}$	FLASH chip select bit 3	N8
$\overline{\text{FLASH.CS2}}$	FLASH chip select bit 2	M4

Table A–1. Input and Output Signals for the OMAP5910 Device (Continued)

Signals	Description	Ballout
FLASH (continued)		
FLASH.CS1	FLASH chip select bit 1	M3
FLASH.CS0	FLASH chip select bit 0	M7
FLASH.BE[1]	FLASH byte enable bit 1	M8
FLASH.BE[0]	FLASH byte enable bit 0	L3
FLASH.ADV	FLASH address valid	L4
FLASH.A[24]	FLASH address bit 24	L7
FLASH.A[23]	FLASH address bit 23	K3
FLASH.A[22]	FLASH address bit 22	K4
FLASH.A[21]	FLASH address bit 21	L8
FLASH.A[20]	FLASH address bit 20	J1
FLASH.A[19]	FLASH address bit 19	J3
FLASH.A[18]	FLASH address bit 18	J4
FLASH.A[17]	FLASH address bit 17	J2
FLASH.A[16]	FLASH address bit 16	K7
FLASH.A[15]	FLASH address bit 15	H3
FLASH.A[14]	FLASH address bit 14	H4
FLASH.A[13]	FLASH address bit 13	K8
FLASH.A[12]	FLASH address bit 12	G2
FLASH.A[11]	FLASH address bit 11	G3
FLASH.A[10]	FLASH address bit 10	G4
FLASH.A[9]	FLASH address bit 9	F3
FLASH.A[8]	FLASH address bit 8	J7
FLASH.A[7]	FLASH address bit 7	E3
FLASH.A[6]	FLASH address bit 6	F4
FLASH.A[5]	FLASH address bit 5	D2
FLASH.A[4]	FLASH address bit 4	E4

Table A–1. Input and Output Signals for the OMAP5910 Device (Continued)

Signals	Description	Ballout
FLASH (continued)		
FLASH.A[3]	FLASH address bit 3	C1
FLASH.A[2]	FLASH address bit 2	D3
FLASH.A[1]	FLASH address bit 1	J8
FLASH.RDY	FLASH ready for TI/wait for Intel	H7
$\overline{\text{FLASH.BAA}}$	FLASH burst advance acknowledge	M4
SDRAM		
$\overline{\text{SDRAM.WE}}$	SDRAM write enable	C3
$\overline{\text{SDRAM.RAS}}$	SDRAM row address srtobe	A2
$\overline{\text{SDRAM.DQMU}}$	SDRAM upper byte mask	D4
$\overline{\text{SDRAM.DQML}}$	SDRAM lower byte mask	B3
SDRAM.D[15]	SDRAM data bit 15	D5
SDRAM.D[14]	SDRAM data bit 14	C4
SDRAM.D[13]	SDRAM data bit 13	B4
SDRAM.D[12]	SDRAM data bit 12	D6
SDRAM.D[11]	SDRAM data bit 11	C5
SDRAM.D[10]	SDRAM data bit 10	H8
SDRAM.D[9]	SDRAM data bit 9	C6
SDRAM.D[8]	SDRAM data bit 8	B6
SDRAM.D[7]	SDRAM data bit 7	D7
SDRAM.D[6]	SDRAM data bit 6	C7
SDRAM.D[5]	SDRAM data bit 5	D8
SDRAM.D[4]	SDRAM data bit 4	B8
SDRAM.D[3]	SDRAM data bit 3	G8
SDRAM.D[2]	SDRAM data bit 2	C8
SDRAM.D[1]	SDRAM data bit 1	G9
SDRAM.D[0]	SDRAM data bit 0	B9

Table A–1. Input and Output Signals for the OMAP5910 Device (Continued)

Signals	Description	Ballout
SDRAM (continued)		
SDRAM.CKE	SDRAM power down control signal	D9
SDRAM.CLK	SDRAM clock	C9
$\overline{\text{SDRAM.CAS}}$	SDRAM column address srtobe	H9
SDRAM.BA[1]	SDRAM bank select 1	D10
SDRAM.BA[0]	SDRAM bank select 0	C10
SDRAM.A[12]	SDRAM address bit 12	G10
SDRAM.A[11]	SDRAM address bit 11	H10
SDRAM.A[10]	SDRAM address bit 10	C11
SDRAM.A[9]	SDRAM address bit 9	D11
SDRAM.A[8]	SDRAM address bit 8	G11
SDRAM.A[7]	SDRAM address bit 7	C12
SDRAM.A[6]	SDRAM address bit 6	D12
SDRAM.A[5]	SDRAM address bit 5	H11
SDRAM.A[4]	SDRAM address bit 4	C13
SDRAM.A[3]	SDRAM address bit 3	D13
SDRAM.A[2]	SDRAM address bit 2	G12
SDRAM.A[1]	SDRAM address bit 1	C14
SDRAM.A[0]	SDRAM address bit 0	B14
LCD Interface		
LCD.VS	LCD vertical synchronization	D14
LCD.HS	LCD horizental synchronization	H12
LCD.AC	LCD ac-bias or output enable for connection to LCD	B15
LCD.PCLK	LCD pixel clock	C15
LCD.P[15]	LCD pixel data bit 15	D15
LCD.P[14]	LCD pixel data bit 14	C16
LCD.P[13]	LCD pixel data bit 13	A17

Table A–1. Input and Output Signals for the OMAP5910 Device (Continued)

Signals	Description	Ballout
LCD Interface (continued)		
LCD.P[12]	LCD pixel data bit 12	G13
LCD.P[11]	LCD pixel data bit 11	B17
LCD.P[10]	LCD pixel data bit 10	C17
LCD.P[9]	LCD pixel data bit 9	D16
LCD.P[8]	LCD pixel data bit 8	D17
LCD.P[7]	LCD pixel data bit 7	C18
LCD.P[6]	LCD pixel data bit 6	B19
LCD.P[5]	LCD pixel data bit 5	A20
LCD.P[4]	LCD pixel data bit 4	H13
LCD.P[3]	LCD pixel data bit 3	G14
LCD.P[2]	LCD pixel data bit 2	C19
LCD.P[1]	LCD pixel data bit 1	B21
LCD.P[0]	LCD pixel data bit 0	D18
McBSP1 Interface		
MCBSP1.CLKS	McBSP1 clock input	G20
MBSP1.CLKX	McBSP1 bit clock	G21
MCBSP1.FSX	McBSP1 frame synchronization	H15, H18
MCBSP1.DX	McBSP1 data ouput	H18, H15
MCBSP1.DR	McBSP1 data input	H20
MCSI1 Interface		
MCSI1.DOUT	MCSI1 data output	W14
MCSI1.SYNC	MCSI1 frame synchronization	V13
MCSI1.CLK	MCSI1 bit clock	AA13
MCSI1.DIN	MCSI1 data input	W13

Table A–1. Input and Output Signals for the OMAP5910 Device (Continued)

Signals	Description	Ballout
McBSP2 Interface		
MCBSP2.DR	McBSP2 data input	P10, AA5
MCBSP2.FSX	McBSP2 transmit frame synchronization	W7
MCBSP2.CLKR	McBSP2 receive clock	V7
MCBSP2.CLKX	McBSP2 transmit clock	Y6
MCBSP2.FSR	McBSP2 receive frame synchronization	W6
MCBSP2.DX	McBSP2 data output	AA5, P10
MCSI2 interface		
MCSI2.CLK	MCSI2 clock	Y10
MCSI2.DIN	MCSI2 data input	AA9
MCSI2.DOUT	MCSI2 data output	W9
MCSI2.SYNC	MCSI2 frame synchronization	V9
UART1 Interface		
UART1.RX	UART1 receive data	V14
UART1.TX	UART1 transmit data	Y14
UART1.RTS	UART1 request to send	AA15
UART1.CTS	UART1 clear to send	R14
UART1.DSR	UART1 data set ready	U18, R13
UART1.DTR	UART1 data terminal ready	W21, Y13
UART3 Interface		
UART3.RX	UART3 receive data	L14, K19
UART3.TX	UART3 transmit data	M18, K18
UART3.RTS	UART3 request to send in UART mode SD_MODE in IRDA mode.	Y13, R19, K14
UART3.CTS	UART3 clear to send	R13, K15
UART3.DSR	UART3 data set ready	U18
UART3.DTR	UART3 data terminal ready	W21

Table A–1. Input and Output Signals for the OMAP5910 Device (Continued)

Signals	Description	Ballout
UART2 Interface		
UART2.RX	UART2 receive data	R9, L14
UART2.TX	UART2 transmit data	V6, M18
UART2.RTS	UART2 request to send	W5, M19
UART2.CTS	UART2 clear to send	Y5,L15
UART2.BCLK	UART2 baud clock	Y4
Clock COM Interface		
MCLK	M-CLK (master clock) output (12/48 MHz)	Y9
MCLKREQ	Request for the M-CLK	R10
BCLK	BCLK general-purpose clock output	Y13
BCLKREQ	BCLK request input	R13
GPIO		
GPIO15	General purpose I/O 15	M20
GPIO14	General purpose I/O 14	N21
GPIO13	General purpose I/O 13	N19
GPIO12	General purpose I/O 12	N18, W6
GPIO11	General purpose I/O 11	N20, V7
GPIO9	General purpose I/O 9	W8
GPIO8	General purpose I/O 8	Y8
GPIO7	General purpose I/O 7	M15, Y5, V9
GPIO6	General purpose I/O 6	P19
GPIO4	General purpose I/O 4	P20
GPIO3	General purpose I/O 3	P18
GPIO2	General purpose I/O 2	M14
GPIO1	General purpose I/O 1	R19
GPIO0	General purpose I/O 0	R18

Table A–1. Input and Output Signals for the OMAP5910 Device (Continued)

Signals	Description	Ballout
MPU I/O		
MPUIO12	MPU input/output 12	L19
MPUIO11	MPU input/output 11	W10
MPUIO7	MPU input/output 7	V10
MPUIO6	MPU input/output 6	W11
MPUIO5	MPU input/output 5	T20, W5
MPUIO4	MPU input/output 4	T19
MPUIO3	MPU input/output 3	V8
MPUIO2	MPU input/output 2	N15
MPUIO1	MPU input/output 1	U19
MPUIO0	MPU input/output 0	Y12
I^2C Interface		
SCL	I^2C master serial clock	T18
SDA	I^2C serial bidirectional data	V20
UWIRE Interface		
UWIRE.SDI	UWIRE serial data input	U18, J14
UWIRE.SDO	UWIRE serial data output	W21, H19
UWIRE.SCLK	UWIRE serial clock	V19, J15
$\overline{\text{UWIRE.CS0}}$	UWIRE serial chip select 0	N14, J18
$\overline{\text{UWIRE.CS3}}$	UWIRE serial chip select 3	P15, J19
McBSP3 Interface		
MCBSP3.DR	McBSP3 data input	AA17, U18
MCBSP3.DX	McBSP3 data output	P14, W21
MCBSP3.FSX	McBSP3 frame synchronization	N18, P18, P19, P20
MCBSP3.CLKX	McBSP3 clock	W16, N14

Table A–1. Input and Output Signals for the OMAP5910 Device (Continued)

Signals	Description	Ballout
Miscelleanous		
MPU_BOOT	MPU boot mode	AA17
RST_HOST_OUT	Reset signal	P14
CLK32K_IN	32 kHz clock input	P13
CLK32K_OUT	32 kHz clock output	Y12
CLK32K_CTRL	32 kHz clock Selection	AA20
$\overline{\text{RST\_OUT}}$	Global reset output of MPU subsystem	W15
$\overline{\text{PWRON\_RESET}}$	Power on reset	G19
$\overline{\text{MPU\_RST}}$	Warm boot reset to TI925T only	V15
$\overline{\text{EXT\_DMA\_REQ1}}$	External DMA request 1	T19
$\overline{\text{EXT\_DMA\_REQ0}}$	External DMA request 0	N15
LOW_PWR	Low power request	T20
BFAIL/$\overline{\text{EXT\_FIQ}}$	Battery voltage failure detection and/or external FIQ input	W19
IRQ_OBS	Interrupt observability output	M18
DMA_REQ_OBS	DMA request observability output	L14
USB Integrated Transceiver Pins		
USB.DP	USB differential (+) line	P9
USB.DM	USB differential (–) line	R8
USB.PUEN	USB clock output (6 MHz)	W4
USB.CLKO	USB pullup enable	W4
USB.VBUS	USB VBUS detect input	R18
Camera Interface		
CAM.EXCLK	Camera clock output	H19
CAM.LCLK	Camera image data latch clock	J15
CAM.D[7]	Camera digital image data bit 7	J18
CAM.D[6]	Camera digital image data bit 6	J19
CAM.D[5]	Camera digital image data bit 5	J14

Table A–1. Input and Output Signals for the OMAP5910 Device (Continued)

Signals	Description	Ballout
Camera Interface (continued)		
CAM.D[4]	Camera digital image data bit 4	K18
CAM.D[3]	Camera digital image data bit 3	K19
CAM.D[2]	Camera digital image data bit 2	K15
CAM.D[1]	Camera digital image data bit 1	K14
CAM.D[0]	Camera digital image data bit 0	L19
CAM.VS	Camera vertical synchronization	L18
CAM.HS	Camera horizontal synchronization	L15
CAM.RSTZ	Camera module reset	M19
MMC/SD Interface		
MMC.DAT3	SD card data bit 3	W11
MMC.DAT2	SD card data bit 2	W10, M15
MMC.DAT1	SD card data bit 1	V10
MMC.DAT0_SPI.DI	MMC or SD card data bit 0/SPI serial input	R11
MMC.CLK	MMC/SD clock	V11
MMC.CMD_SPI.DO	MMC/SD command/SPI serial output	P11
SPI.CS3	SPI chip select 3	P18
SPI.CS2	SPI chip select 2	P20
SPI.CS1	SPI chip select 1	P19
SPI.RDY	SPI ready	R18
SPI.CLK	SPI clock	M14
Oscillators		
OSC1_IN	12 MHz quartz connection (XI)	Y2
OSC1_OUT	12 MHz quartz connection (XO)	W3
OSC32K_IN	32 kHz quartz connection (XI)	W12
OSC32K_OUT	32 kHz quartz connection (XO)	R12

Table A–1. Input and Output Signals for the OMAP5910 Device (Continued)

Signals	Description	Ballout
Configuration Interface		
CONF	OMAP5910 configuration Input	V18
STAT_VAL/$\overline{\text{WKUP}}$	MMC slect/wakeup input	Y17
Keyboard Interface		
KB.C[7]	Keyboard matrix column 7	V19
KB.C[6]	Keyboard matrix column 6	P15
KB.C[5]	Keyboard matrix column 5	C20
KB.C[4]	Keyboard matrix column 4	C21
KB.C[3]	Keyboard matrix column 3	E18
KB.C[2]	Keyboard matrix column 2	D19
KB.C[1]	Keyboard matrix column 1	D20
KB.C[0]	Keyboard matrix column 0	F18
KB.R[7]	Keyboard matrix row 7	M20
KB.R[6]	Keyboard matrix row 6	N21
KB.R[5]	Keyboard matrix row 5	N19
KB.R[4]	Keyboard matrix row 4	E19
KB.R[3]	Keyboard matrix row 3	E20
KB.R[2]	Keyboard matrix row 2	H14
KB.R[1]	Keyboard matrix row 1	F19
KB.R[0]	Keyboard matrix row 0	G18
JTAG Interface		
TDI	Test data input	Y19
TDO	Test data output	AA19
TMS	Test mode select	V17
TCK	Test clock	W18
$\overline{\text{TRST}}$	Test reset	Y18

Table A–1. Input and Output Signals for the OMAP5910 Device (Continued)

Signals	Description	Ballout
JTAG Interface (continued)		
EMU0	Test emulation 0	V16
EMU1	Test emulation 1	W17
Trace Interface		
ETM.SYNC	ETM9 trace synchronization	H19
ETM.CLK	ETM9 trace clock	J15
ETM.D[7]	ETM9 trace packet bit 7	J18
ETM.D[6]	ETM9 trace packet bit 6	J19
ETM.D[5]	ETM9 trace packet bit 5	J14
ETM.D[4]	ETM9 trace packet bit 4	K18
ETM.D[3]	ETM9 trace packet bit 3	K19
ETM.D[2]	ETM9 trace packet bit 2	K15
ETM.D[1]	ETM9 trace packet bit 1	K14
ETM.D[0]	ETM9 trace packet bit 0	L19
ETM.PSTAT[2]	ETM9 trace pipe state bit 2	L18
ETM.PSTAT[1]	ETM9 trace pipe state bit 1	L15
ETM.PSTAT[0]	ETM9 trace pipe state bit 0	M19
HDQ/1-Wire Interface		
HDQ	HDQ 1-Wire interface pin	N20
LED Pulse Generators		
LED1	LED pulse generator 1 output	P18
LED2	LED pulse generator 2 output	T19
PWM Interface		
PWT	Pulse width tone	M18
PWL	Pulse width light	L14

Table A–1. Input and Output Signals for the OMAP5910 Device (Continued)

Signals	Description	Ballout
USB Pin Group #1		
USB1.SUSP	USB 1 bus segment suspend control	AA17
USB1.VM	USB 1 Vminus receive data	AA13
USB1.SE0	USB 1 single ended zero	P14
USB1.TXEN	USB 1 transmit enable	W16
USB1.SPEED	USB 1 bus segment speed control	Y12
USB1.VP	USB 1 Vplus receive data	V13
USB1.TXD	USB 1 transmit data	W14
USB1.RCV	USB 1 receive data	W13
USB Pin Group #2		
USB2.VP	USB 2 Vplus receive data	AA9
USB2.VM	USB 2 Vminus receive data	R9
USB2.SE0	USB 2 single ended zero	W5
USB2.TXEN	USB 2 transmit enable	W9
USB2.SUSP	USB 2 bus segment suspend control	Y10
USB2.RCV	USB 2 receive data	Y5
USB2.TXD	USB 2 transmit data	V6

A.2 I/O Functional Multiplexing

Table A–2 provides the input/output configuration programming for signal multiplexing on each pin and for enabling/disabling internal pullup and pull-down resistors on each pin.

Table A–2. Configuration Programming

Ballout	Pin-by-Pin Multiplexing OMAP5910 Configuration Register	Register Offset	Register Field	Value	Signal on the Top	Pin-by-Pin Pullup/Down OMAP5910 Configuration Register	Register Offset	Register Field
A1								
B2								
C3								
A2								
D4								
B3								
A3								
D5								
C4								
B4								
D6								
C5								
B5								
A5								
H8								
C6								
B6								
D7								
C7								
B7								

Note: When a row is empty, it means that there is no:
– Functional multiplexing on this pin
– Software configurable pullup/pulldown on this pin

Table A–2. Configuration Programming (Continued)

Ballout	Pin-by-Pin Multiplexing OMAP5910 Configuration Register	Register Offset	Register Field	Value	Signal on the Top	Pin-by-Pin Pullup/Down OMAP5910 Configuration Register	Register Offset	Register Field
A7								
G8								
D8								
B8								
C8								
G9								
B9								
D9								
A9								
C9								
B10								
H9								
D10								
C10								
G10								
H10								
A11								
C11								
D11								
G11								
C12								
D12								
H11								

Note: When a row is empty, it means that there is no:
– Functional multiplexing on this pin
– Software configurable pullup/pulldown on this pin

Table A–2. Configuration Programming (Continued)

Ballout	Pin-by-Pin Multiplexing OMAP5910 Configuration Register	Register Offset	Register Field	Value	Signal on the Top	Pin-by-Pin Pullup/Down OMAP5910 Configuration Register	Register Offset	Register Field
B12								
A13								
C13								
D13								
B13								
G12								
C14								
B14								
A15								
D14								
H12								
B15								
C15								
D15								
B16								
C16								
A17								
G13								
B17								
C17								
D16								
B18								
D17								

Note: When a row is empty, it means that there is no:
– Functional multiplexing on this pin
– Software configurable pullup/pulldown on this pin

Table A–2. Configuration Programming (Continued)

Ballout	Pin-by-Pin Multiplexing OMAP5910 Configuration Register	Register Offset	Register Field	Value	Signal on the Top	Pin-by-Pin Pullup/Down OMAP5910 Configuration Register	Register Offset	Register Field
A19								
C18								
B19								
A20								
H13								
G14								
A21								
B20								
C19								
B21								
D18								
C20								
C21								
E18								
D19								
D20								
F18								
E19								
E20								
E21								
H14								
F19								
F20								

Note: When a row is empty, it means that there is no:
– Functional multiplexing on this pin
– Software configurable pullup/pulldown on this pin

Table A–2. Configuration Programming (Continued)

Ballout	Pin-by-Pin Multiplexing OMAP5910 Configuration Register	Register Offset	Register Field	Value	Signal on the Top	Pin-by-Pin Pullup/Down OMAP5910 Configuration Register	Register Offset	Register Field
G18								
G19								
G20								
G21								
H15	FUNC_MUX_CTRL_4	0x14	14:12	000	MCBSP1.FSX			
				001	MCBSP1.DX			
H18	FUNC_MUX_CTRL_4	0x14	17:15	000	MCBSP1.DX			
				001	MCBSP1.FSX			
H20						PULL_DOWN_CTRL_0	0x40	16
H19	FUNC_MUX_CTRL_4	0x14	23:21	000	CAM.EXCLK			
				001	ETM.SYNC			
				010	UWIRE.SDO			
J15	FUNC_MUX_CTRL_4	0x14	26:24	000	CAM.LCLK			
				001	ETM.CLK			
				010	UWIRE.SCLK			
J20								
J18	FUNC_MUX_CTRL_4	0x14	29:27	000	CAM.D[7]			
				001	ETM.D[7]			
				010	$\overline{\text{UWIRE.CS0}}$			

Note: When a row is empty, it means that there is no:
– Functional multiplexing on this pin
– Software configurable pullup/pulldown on this pin

Table A–2. Configuration Programming (Continued)

Ballout	Pin-by-Pin Multiplexing OMAP5910 Configuration Register	Register Offset	Register Field	Value	Signal on the Top	Pin-by-Pin Pullup/Down OMAP5910 Configuration Register	Register Offset	Register Field
J21								
J19	FUNC_MUX_ CTRL_5	0x18	2:0	000	CAM.D[6]			
				001	ETM.D[6]			
				010	UWIRE.CS3			
K20								
J14	FUNC_MUX_ CTRL_5	0x18	5:3	000	CAM.D[5]	PULL_DOWN_ CTRL_0	0x40	21
				001	ETM.D[5]			
				010	UWIRE.SDI			
K18	FUNC_MUX_ CTRL_5	0x18	8:6	000	CAM.D[4]			
				001	ETM.D[4]			
				010	UART3.TX			
K19	FUNC_MUX_ CTRL_5	0x18	11:9	000	CAM.D[3]	PULL_DOWN_ CTRL_0	0x40	23
				001	ETM.D[3]			
				010	UART3.RX			
K15	FUNC_MUX_ CTRL_5	0x18	14:12	000	CAM.D[2]	PULL_DOWN_ CTRL_0	0x40	24
				001	ETM.D[2]			
				010	UART3.CTS			
K14	FUNC_MUX_ CTRL_5	0x18	17:15	000	CAM.D[1]			
				001	ETM.D[1]			
				010	UART3.RTS			

Note: When a row is empty, it means that there is no:
– Functional multiplexing on this pin
– Software configurable pullup/pulldown on this pin

Table A–2. Configuration Programming (Continued)

Ballout	Pin-by-Pin Multiplexing OMAP5910 Configuration Register	Register Offset	Register Field	Value	Signal on the Top	Pin-by-Pin Pullup/Down OMAP5910 Configuration Register	Register Offset	Register Field
L21								
L19	FUNC_MUX_CTRL_5	0x18	20:18	000	CAM.D[0]			
				001	ETM.D[0]			
				010	MPUIO12			
L18	FUNC_MUX_CTRL_5	0x18	23:21	000	CAM.VS			
				001	ETM.PSTAT[2]			
L15	FUNC_MUX_CTRL_5	0x18	26:24	000	CAM.HS	PULL_DOWN_CTRL_0	0x40	28
				001	ETM.PSTAT[1]			
				010	UART2.CTS			
M19	FUNC_MUX_CTRL_5	0x18	29:27	000	CAM.RSTZ			
				001	ETM.PSTAT[0]			
				010	UART2.RTS			
M18	FUNC_MUX_CTRL_6	0x1C	2:0	000	UART3.TX			
				001	UART3.TX			
				010	PWT			
				011	Reserved			
				100	UART2.TX			

Note: When a row is empty, it means that there is no:
– Functional multiplexing on this pin
– Software configurable pullup/pulldown on this pin

Table A–2. Configuration Programming (Continued)

Ballout	Pin-by-Pin Multiplexing OMAP5910 Configuration Register	Register Offset	Register Field	Value	Signal on the Top	Pin-by-Pin Pullup/Down OMAP5910 Configuration Register	Register Offset	Register Field
L14	FUNC_MUX_CTRL_6	0x1C	5:3	000	UART3.RX			
				001	PWL			
				010	Reserved			
				011	UART2.RX			
M20	FUNC_MUX_CTRL_6	0x1C	8:6	000	GPIO15	PULL_DOWN_CTRL_1	0x44	0
				001	KB.R[7]			
N21	FUNC_MUX_CTRL_6	0x1C	11:9	000	GPIO14	PULL_DOWN_CTRL_1	0x44	1
				001	KB.R[6]			
N19	FUNC_MUX_CTRL_6	0x1C	14:12	000	GPIO13	PULL_DOWN_CTRL_1	0x44	2
				001	KB.R[5]			
N18	FUNC_MUX_CTRL_6	0x1C	17:15	000	GPIO12	PULL_DOWN_CTRL_1	0x44	3
				001	MCBSP3.FSX			
N20	FUNC_MUX_CTRL_6	0x1C	20:18	000	GPIO11	PULL_DOWN_CTRL_1	0x44	4
				001	HDQ			
M15	FUNC_MUX_CTRL_6	0x1C	23:21	000	GPIO7	PULL_DOWN_CTRL_1	0x44	5
				001	MMC.DAT2			

Note: When a row is empty, it means that there is no:
- Functional multiplexing on this pin
- Software configurable pullup/pulldown on this pin

Table A–2. Configuration Programming (Continued)

Ballout	Pin-by-Pin Multiplexing OMAP5910 Configuration Register	Register Offset	Register Field	Value	Signal on the Top	Pin-by-Pin Pullup/Down OMAP5910 Configuration Register	Register Offset	Register Field
P19	FUNC_MUX_CTRL_6	0x1C	26:24	000	GPIO6	PULL_DOWN_CTRL_1	0x44	6
				001	SPI.CS1			
				010	MCBSP3.FSX			
P20	FUNC_MUX_CTRL_6	0x1C	29:27	000	GPIO4	PULL_DOWN_CTRL_1	0x44	7
				001	SPI.CS2			
				010	MCBSP3.FSX			
R21								
P18	FUNC_MUX_CTRL_7	0x20	2:0	000	GPIO3	PULL_DOWN_CTRL_1	0x44	8
				001	SPI.CS3			
				010	MCBSP3.FSX			
				011	LED1			
M14	FUNC_MUX_CTRL_7	0x20	5:3	000	GPIO2	PULL_DOWN_CTRL_1	0x44	9
				001	SPI.CLK			
R20								
R19	FUNC_MUX_CTRL_7	0x20	8:6	000	GPIO1	PULL_DOWN_CTRL_1	0x44	10
				001	UART3.RTS			

Note: When a row is empty, it means that there is no:
– Functional multiplexing on this pin
– Software configurable pullup/pulldown on this pin

Table A–2. Configuration Programming (Continued)

Ballout	Pin-by-Pin Multiplexing OMAP5910 Configuration Register	Register Offset	Register Field	Value	Signal on the Top	Pin-by-Pin Pullup/Down OMAP5910 Configuration Register	Register Offset	Register Field
R18	FUNC_MUX_CTRL_7	0x20	11:9	000	GPIO0	PULL_DOWN_CTRL_1	0x44	11
				001	SPI.RDY			
				010	USB.VBUS			
T20	FUNC_MUX_CTRL_7	0x20	14:12	000	MPUIO5	PULL_DOWN_CTRL_1	0x44	12
				001	LOW_PWR			
T19	FUNC_MUX_CTRL_7	0x20	17:15	000	MPUIO4	PULL_DOWN_CTRL_1	0x44	13
				001	$\overline{\text{EXT\_DMA\_REQ0}}$			
				010	LED2			
U21								
N15	FUNC_MUX_CTRL_7	0x20	20:18	000	MPUIO2	PULL_DOWN_CTRL_1	0x44	14
				001	$\overline{\text{EXT\_DMA\_REQ0}}$			
U20								
U19								
T18								
V20								
U18	FUNC_MUX_CTRL_8	0x24	2:0	000	UWIRE.SDI	PULL_DOWN_CTRL_1	0x44	18
				001	UART3.DSR			
				010	UART1.DSR			
				011	MCBSP3.DR			

Note: When a row is empty, it means that there is no:
– Functional multiplexing on this pin
– Software configurable pullup/pulldown on this pin

Table A–2. Configuration Programming (Continued)

Ballout	Pin-by-Pin Multiplexing OMAP5910 Configuration Register	Register Offset	Register Field	Value	Signal on the Top	Pin-by-Pin Pullup/Down OMAP5910 Configuration Register	Register Offset	Register Field
W21	FUNC_MUX_CTRL_8	0x24	5:3	000	UWIRE.SDO			
				001	UART3.DTR			
				010	UART1.DTR			
				011	MCBSP3.DX			
V19	FUNC_MUX_CTRL_8	0x24	8:6	000	UWIRE.SCLK			
				001	KB.C[7]			
W20								
Y21								
N14	FUNC_MUX_CTRL_8	0x24	11:9	000	$\overline{\text{UWIRE.CS0}}$			
				001	$\overline{\text{UWIRE.CS0}}$			
				010	MCBSP3.CLKX			
P15	FUNC_MUX_CTRL_8	0x24	14:12	000	$\overline{\text{UWIRE.CS3}}$			
				001	$\overline{\text{UWIRE.CS3}}$			
				010	KB.C[6]			
AA21								
Y20								
W19								
AA20								
V18						PULL_DOWN_CTRL_3	0x4C	9

Note: When a row is empty, it means that there is no:
- Functional multiplexing on this pin
- Software configurable pullup/pulldown on this pin

Table A–2. Configuration Programming (Continued)

Ballout	Pin-by-Pin Multiplexing OMAP5910 Configuration Register	Register Offset	Register Field	Value	Signal on the Top	Pin-by-Pin Pullup/Down OMAP5910 Configuration Register	Register Offset	Register Field
Y19						PULL_DOWN_CTRL_3	0x4C	10
AA19								
V17						PULL_DOWN_CTRL_3	0x4C	11
W18						PULL_DOWN_CTRL_3	0x4C	12
Y18						PULL_DOWN_CTRL_3	0x4C	13
V16						PULL_DOWN_CTRL_1	0x44	25
W17						PULL_DOWN_CTRL_1	0x44	26
Y17								
AA17	FUNC_MUX_CTRL_8	0x24	29:27	000	MPU_BOOT	PULL_DOWN_CTRL_1	0x44	27
				001	MCBSP3.DR			
				010	USB1_SUSP			
P14	FUNC_MUX_CTRL_9	0x28	2:0	000	RST_HOST_OUT			
				001	MCBSP3.DX			
				010	USB1_SE0			
W16	FUNC_MUX_CTRL_9	0x28	5:3	000	MCBSP3.CLKX	PULL_DOWN_CTRL_1	0x44	29
				001	MCBSP3.CLKX			
				010	USB1_TXEN			
Y16								

Note: When a row is empty, it means that there is no:
– Functional multiplexing on this pin
– Software configurable pullup/pulldown on this pin

Table A–2. Configuration Programming (Continued)

Ballout	Pin-by-Pin Multiplexing OMAP5910 Configuration Register	Register Offset	Register Field	Value	Signal on the Top	Pin-by-Pin Pullup/Down OMAP5910 Configuration Register	Register Offset	Register Field
V15								
W15								
Y15								
AA15	FUNC_MUX_ CTRL_9	0x28	14:12	000	UART1.RTS			
				001	UART1.RTS			
R14						PULL_DOWN_ CTRL_2	0x48	1
V14						PULL_DOWN_ CTRL_2	0x48	2
Y14	FUNC_MUX_ CTRL_9	0x28	23:21	000	UART1.TX			
				001	UART1.TX			
W14	FUNC_MUX_ CTRL_9	0x28	26:24	000	MCSI1. DOUT			
				001	USB1.TXD			
R13	FUNC_MUX_ CTRL_9	0x28	29:27	000	UART3. CLKREQ	PULL_DOWN_ CTRL_2	0x48	5
				001	UART3.CTS			
				010	UART1.DSR			
Y13	FUNC_MUX_ CTRL_A	0x2C	2:0	000	UART3. BCLK			
				001	UART3.RTS			
				010	UART1. DTR			

Note: When a row is empty, it means that there is no:
- Functional multiplexing on this pin
- Software configurable pullup/pulldown on this pin

Table A–2. Configuration Programming (Continued)

Ballout	Pin-by-Pin Multiplexing OMAP5910 Configuration Register	Register Offset	Register Field	Value	Signal on the Top	Pin-by-Pin Pullup/Down OMAP5910 Configuration Register	Register Offset	Register Field
V13	FUNC_MUX_CTRL_A	0x2C	5:3	000	MCSI1.SYNC	PULL_DOWN_CTRL_2	0x48	7
				001	USB1.VP			
AA13	FUNC_MUX_CTRL_A	0x2C	8:6	000	MCSI1.CLK	PULL_DOWN_CTRL_2	0x48	8
				001	USB1.VM			
W13	FUNC_MUX_CTRL_A	0x2C	11:9	000	MCSI1.DIN	PULL_DOWN_CTRL_2	0x48	9
				001	USB1.RCV			
Y12	FUNC_MUX_CTRL_A	0x2C	14:12	000	CLK32K_OUT			
				001	MPUIO0			
				010	USB1.SPEED			
P13								
V12								
W12								
R12								
P12								
AA11								
W11	FUNC_MUX_CTRL_D	0x38	14:12	000	MMC.DAT3	PULL_DOWN_CTRL_3	0x4C	8
				001	ms_remove			
				010	MPUIO6			
V11								
R11								

Note: When a row is empty, it means that there is no:
– Functional multiplexing on this pin
– Software configurable pullup/pulldown on this pin

Table A–2. Configuration Programming (Continued)

Ballout	Pin-by-Pin Multiplexing OMAP5910 Configuration Register	Register Offset	Register Field	Value	Signal on the Top	Pin-by-Pin Pullup/Down OMAP5910 Configuration Register	Register Offset	Register Field
W10	FUNC_MUX_CTRL_A	0x2C	20:18	000	MMC.DAT2	PULL_DOWN_CTRL_2	0x48	12
				001	VSS (Forced to 'Z')			
				010	MPUIO11			
V10	FUNC_MUX_CTRL_A	0x2C	26:24	000	MMC.DAT1	PULL_DOWN_CTRL_2	0x48	14
				001	ms_ins			
				010	MPUIO7			
P11						PULL_DOWN_CTRL_2	0x48	15
Y10	FUNC_MUX_CTRL_B	0x30	5:3	000	MCSI2.CLK	PULL_DOWN_CTRL_2	0x48	17
				001	USB2.SUSP			
AA9	FUNC_MUX_CTRL_B	0x30	8:6	000	MCSI2.DIN	PULL_DOWN_CTRL_2	0x48	18
				001	USB2.VP			
W9	FUNC_MUX_CTRL_B	0x30	11:9	000	MCSI2.DOUT			
				001	USB2.TXEN			
V9	FUNC_MUX_CTRL_B	0x30	14:12	000	MCSI2.SYNC	PULL_DOWN_CTRL_2	0x48	20
				001	GPIO7			
Y9								

Note: When a row is empty, it means that there is no:
- Functional multiplexing on this pin
- Software configurable pullup/pulldown on this pin

Table A–2. Configuration Programming (Continued)

Ballout	Pin-by-Pin Multiplexing OMAP5910 Configuration Register	Register Offset	Register Field	Value	Signal on the Top	Pin-by-Pin Pullup/Down OMAP5910 Configuration Register	Register Offset	Register Field
R10	FUNC_MUX_ CTRL_B	0x30	20:18	000	UART2. CLKREQ	PULL_DOWN_ CTRL_2	0x48	22
				001	EXT_ MASTER_ REQ			
W8						PULL_DOWN_ CTRL_2	0x48	23
Y8						PULL_DOWN_ CTRL_2	0x48	24
AA7								
V8						PULL_DOWN_ CTRL_2	0x48	25
P10	FUNC_MUX_ CTRL_C	0x34	2:0	000	MCBSP2.DR	PULL_DOWN_ CTRL_2	0x48	26
				001	MCBSP2.DX			
Y7								
W7						PULL_DOWN_ CTRL_2	0x48	27
V7	FUNC_MUX_ CTRL_C	0x34	8:6	000	MCBSP2. CLKR	PULL_DOWN_ CTRL_2	0x48	28
				001	GPIO11			
Y6						PULL_DOWN_ CTRL_2	0x48	29
W6	FUNC_MUX_ CTRL_C	0x34	14:12	000	MCBSP2. FSR	PULL_DOWN_ CTRL_2	0x48	30
				001	GPIO12			

Note: When a row is empty, it means that there is no:
- Functional multiplexing on this pin
- Software configurable pullup/pulldown on this pin

Table A–2. Configuration Programming (Continued)

Ballout	Pin-by-Pin Multiplexing OMAP5910 Configuration Register	Register Offset	Register Field	Value	Signal on the Top	Pin-by-Pin Pullup/Down OMAP5910 Configuration Register	Register Offset	Register Field
AA5	FUNC_MUX_CTRL_C	0x34	17:15	000	MCBSP2.DX	PULL_DOWN_CTRL_2	0x48	31
				001	MCBSP2.DR			
R9	FUNC_MUX_CTRL_C	0x34	20:18	000	UART2.RX	PULL_DOWN_CTRL_3	0x4C	0
				001	USB2.VM			
Y5	FUNC_MUX_CTRL_C	0x34	23:21	000	UART2.CTS	PULL_DOWN_CTRL_3	0x4C	1
				001	USB2.RCV			
				010	GPIO7			
W5	FUNC_MUX_CTRL_C	0x34	26:24	000	UART2.RTS			
				001	UART2.RTS			
				010	USB2.SE0			
				011	MPUIO5			
V6	FUNC_MUX_CTRL_C	0x34	29:27	000	UART2.TX			
				001	UART2.TX			
				010	USB2.TXD			
Y4								
V5								
AA3								
W4	FUNC_MUX_CTRL_D	0x38	5:3	000	USB.PUEN			
				001	USB.CLKO			
Y3								

Note: When a row is empty, it means that there is no:
– Functional multiplexing on this pin
– Software configurable pullup/pulldown on this pin

Table A–2. Configuration Programming (Continued)

Ballout	Pin-by-Pin Multiplexing OMAP5910 Configuration Register	Register Offset	Register Field	Value	Signal on the Top	Pin-by-Pin Pullup/Down OMAP5910 Configuration Register	Register Offset	Register Field
AA2								
P9								
R8								
AA1								
Y2								
W3								
Y1								
V4								
W2								
W1								
U4								
V3								
V2								
T4								
U3								
U2								
U1								
P8								
T3								
T2								
R4								
R3								
R2								

Note: When a row is empty, it means that there is no:
- Functional multiplexing on this pin
- Software configurable pullup/pulldown on this pin

Table A–2. Configuration Programming (Continued)

Ballout	Pin-by-Pin Multiplexing OMAP5910 Configuration Register	Register Offset	Register Field	Value	Signal on the Top	Pin-by-Pin Pullup/Down OMAP5910 Configuration Register	Register Offset	Register Field
R1								
P7								
P4								
P2								
P3								
N7								
N2								
N4								
N1								
N3								
M2								
N8								
M4	FUNC_MUX_ CTRL_D	0x38	8:6	000	$\overline{\text{FLASH.CS2}}$			
				001	$\overline{\text{FLASH.BAA}}$			
M3								
M7								
M8								
L1								
L3								
L4								
L7								
K3								
K4								

Note: When a row is empty, it means that there is no:
– Functional multiplexing on this pin
– Software configurable pullup/pulldown on this pin

Table A–2. Configuration Programming (Continued)

Ballout	Pin-by-Pin Multiplexing OMAP5910 Configuration Register	Register Offset	Register Field	Value	Signal on the Top	Pin-by-Pin Pullup/Down OMAP5910 Configuration Register	Register Offset	Register Field
L8								
K2								
J1								
J3								
J4								
J2								
K7								
H3								
H2								
G1								
H4								
K8								
G2								
G3								
G4								
F2								
F3								
E1								
J7								
E2								
E3								
F4								
D2								

Note: When a row is empty, it means that there is no:
– Functional multiplexing on this pin
– Software configurable pullup/pulldown on this pin

Table A–2. Configuration Programming (Continued)

Ballout	Pin-by-Pin Multiplexing OMAP5910 Configuration Register	Register Offset	Register Field	Value	Signal on the Top	Pin-by-Pin Pullup/Down OMAP5910 Configuration Register	Register Offset	Register Field
E4								
C1								
D3								
C2								
B1								
J8								
H7								
E5								

Note: When a row is empty, it means that there is no:
– Functional multiplexing on this pin
– Software configurable pullup/pulldown on this pin

Appendix B

Switching Clock Modes

This appendix describes the programming guidelines for switching clock modes in the OMAP5910 device.

Topic **Page**

B.1 Switching Procedure

Perform the following procedure to switch OMAP5910 clock modes:

1) Make sure the DSP clock is enabled.
2) Make sure there are no active transfers on interfaces (EMIFS, EMIFF, TIPB, IMIF, MPUI, etc.).
3) Make sure there are no active DSP transactions being performed.
4) Disable the MPU D-cache/MMU.
5) Make sure the MPU clock control register (ARM_CKCTL), clock dividers, and the DPLL_REG have correct contents for the clock mode the system is being switched to.
    a) Switch from SYNC mode to SYNCSCALE mode:
        i) Make sure that the frequency of the traffic controller is always less than the maximum frequency of the traffic controller.
        ii) Change the clock mode.
        iii) Program the clock dividers.
        iv) Program DPLL to frequency desired.
    b) Switch from SYNCSCALE to SYNC mode:
        i) Program the DPLL to the desired frequency in synchronous mode.
        ii) Program all clock dividers to be equal.
        iii) Change the clock mode to SYNC mode.
6) After the MPU write to the MPU system status register (ARM_SYSST) (0x18) to switch modes, there must be no requests from the MPU to the traffic controller for the next 100 MPU cycles (see Section B.2, *Main Code*, and Section B.3, *Delay Procedure*).
7) Make sure all read and write accesses to the clock reset registers are 16-bit accesses.

B.2 Main Code

The following is the main code for switching modes:

```
main()
{
   …
…..
// Enable Icache
   INT_SetSupervisor();
   ARM_WRITE_REG1(I_bit);
   INT_SetUser();
// Enable DSP Clock
   MCU_CKCTL = 0x2000;
   switch_mode(CLOCK_MODE_SYNC_SCALE);
// Passing in 0x1000
….
….
}
```

B.3 Delay Procedure

Use the following software routine to create a delay of 100 clock cycles after a switch mode write.

Ensure the I-cache is enabled during switching modes.

```
      state16               ; thumb mode
      .ref  edata           ; defined by armas
      .global $switch_mode
$switch_mode:
       push    {lr}
       push    {r1-r7}
       adr     r4, into_32_bis
       bx      r4
   nop
   nop
   nop
       .state32               ; arm mode
into_32_bis:
       ;
      LDR   R1,ARM_SYSST
      MOV   R3,#0
      MOV   R2,#0
; This is the loop that will wait for at least 100 cycles
; before issuing next request from MPU. On the first run
of the loop only Icache
; gets loaded with the loop and the next 2 instructions
but write to SYSST does not occur
; In the 2nd run of the loop only write to SYSST happens
and after that MPU runs the loop from
;Icache so no request goes out
LOOP     CMP   R2,#1
         STREQ R0,[R1]
         ADD   R2,R2,#1
         CMP   R2,#16
         BNE   LOOP
the_end:
```

```
          adr    r2, into_16_bis + 1
          bx     r2
   .state16
into_16_bis:
       nop
       nop
       nop
       nop
       nop
       nop
       nop
       nop
       nop
       nop
       pop   {r1-r7}
       pop   {pc}
;*********************************************************
;* CONSTANT TABLE                                        *
;*********************************************************
ARM_SYSST       .long 0xFFFECE18
```

Index

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W