OMAP5910 Dual-Core Processor
MPU Subsystems
Reference Guide

Literature Number: SPRU671
October 2003

IMPORTANT NOTICE

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

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

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

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

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

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

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

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

Preface

Read This First

About This Manual

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

Notational Conventions

This document uses the following conventions.

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

Related Documentation From Texas Instruments

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

OMAP5910 Dual-Core Processor MPU Subsystem Reference Guide (literature number SPRU671)

OMAP5910 Dual-Core Processor DSP Subsystem Reference Guide (literature number SPRU672)

OMAP5910 Dual-Core Processor Memory Interface Traffic Controller Reference Guide (literature number SPRU673)

OMAP5910 Dual-Core Processor System DMA Controller Reference Guide (literature number SPRU674)

OMAP5910 Dual-Core Processor LCD Controller Reference Guide (literature number SPRU675)

OMAP5910 Dual-Core Processor Universal Asynchronous Receiver/Transmitter (UART) Devices Reference Guide (literature number SPRU676)

OMAP5910 Dual-Core Processor Universal Serial Bus (USB) and Frame Adjustment Counter (FAC) Reference Guide (literature number SPRU677)

OMAP5910 Dual-Core Processor Clock Generation and System Reset Management Reference Guide (literature number SPRU678)

OMAP5910 Dual-Core Processor General-Purpose Input/Output (GPIO) Reference Guide (literature number SPRU679)

OMAP5910 Dual-Core Processor MMC/SD Reference Guide (literature number SPRU680)

OMAP5910 Dual-Core Processor Inter-Integrated Circuit (I2C) Controller Reference Guide (literature number SPRU681)

OMAP5910 Dual-Core Processor Timer Reference Guide (literature number SPRU682)

OMAP5910 Dual-Core Processor Inter-Processor Communication Reference Guide (literature number SPRU683)

OMAP5910 Dual-Core Processor Camera Interface Reference Guide (literature number SPRU684)

OMAP5905 Dual-Core Processor Multichannel Serial Interface (MCSI) Reference Guide (literature number SPRU685)

OMAP5910 Dual-Core Processor Micro-Wire Interface Reference Guide (literature number SPRU686)

OMAP5910 Dual-Core Processor Real-Time Clock (RTC) Reference Guide (literature number SPRU687)

OMAP5910 Dual-Core Processor HDQ/1-Wire Interface Reference Guide (literature number SPRU688)

OMAP5910 Dual-Core Processor PWL, PWT, and LED Peripheral Reference Guide (literature number SPRU689)

OMAP5910 Dual-Core Processor Multichannel Buffered Serial Port (McBSP) Reference Guide (literature number SPRU708)

Trademarks

OMAP and the OMAP symbol are trademarks of Texas Instruments.

Contents

Figures

Tables

MPU Subsystems

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

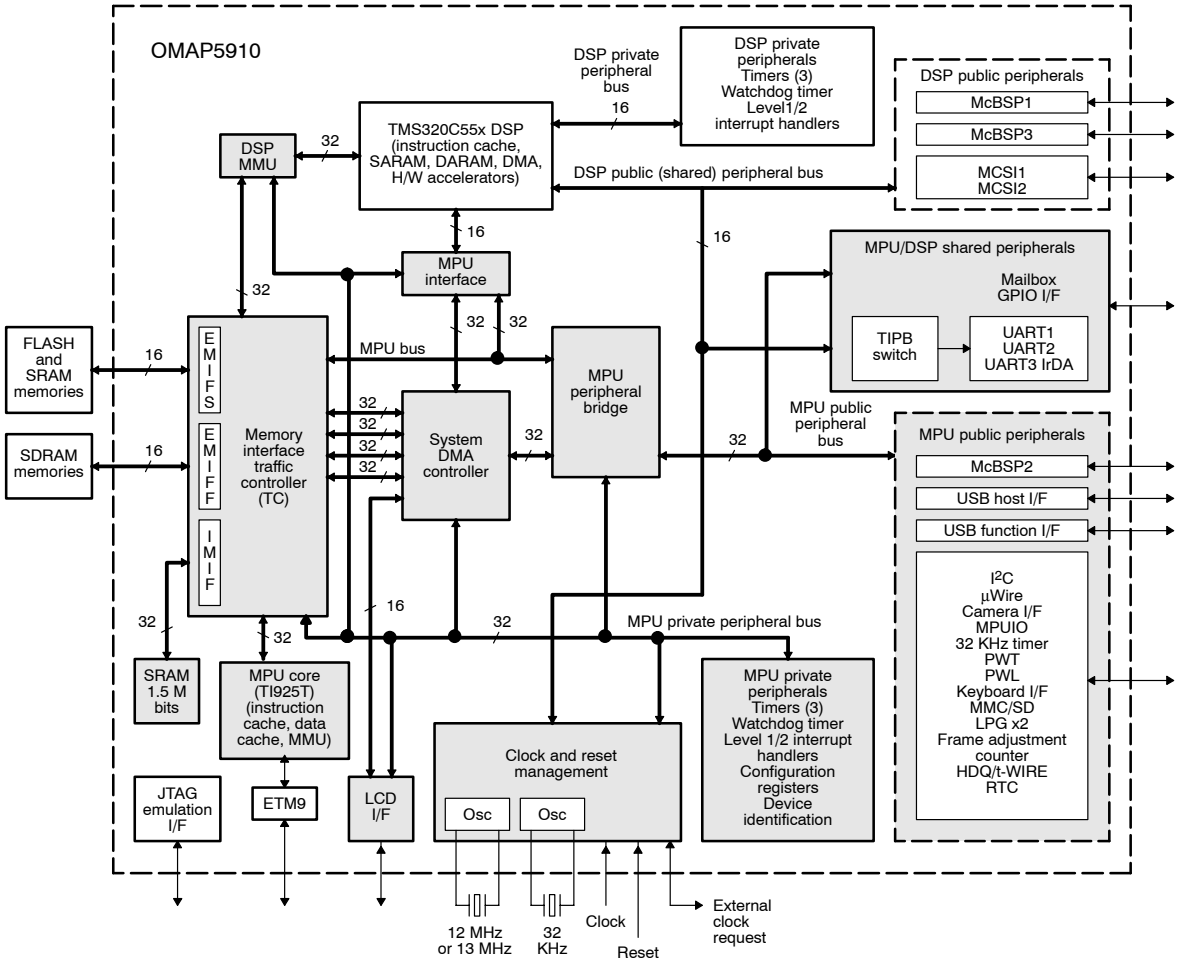
1 Introduction

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

- MPU core (see Section 2, *MPU Core*)
- Traffic controller (see SPRU673, *Memory Interface Traffic Controller Reference Guide*)
- MPU MMU (see Section 7, *MPU Memory Management Unit*)
- DSP MMU (see Section 8, *DSP Memory Management Unit*)
- System DMA controller (see SPRU674 , *System DMA Controller Reference Guide*)
- LCD controller (see SPRU675, *LCD Controller Reference Guide*)
- MPU TIPB bridge (see Section 10, *MPU TI Peripheral Bus Bridges*)
- Clock manager (see SPRU678, *Clock Generation and System Reset Management Reference Guide*)
- Interrupt handler (see Section 1. 11, MPU Interrupt Handler)
- Timers (see SPRU682, Timer Reference Guide)
- Watchdog timer (see SPRU682, Tmer Reference Guide)
- Interprocessor communication (see SPRU683, Interprocessor Communication Reference Guide)
- 1.5M-bit SRAM internal memory

Figure 1. Highlight of MPU Subsystem



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 co-processor 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.

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.

The I-cache can be independently enabled from the MMU.

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. The LOAD_MULTIPLE instruction does not perform a burst read.

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 may be 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.

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

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

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 1).

Table 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.

Table 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.

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.

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

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

To avoid missing timer interrupts, the hardware clean operation can be interrupted for the software algorithm to 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 (discarding cache data completely) is not the same as flushing (temporarily cleaning a cache by writing data to disk or out-of-cache memory).

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.

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 un-cacheable.

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 nonbufferable until the MMU is enabled. To use the write buffer, MMU must be enabled. The write buffer is always disabled when the MMU is off. However, the two functions can be enabled simultaneously with a single write to the CP15 control register.

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.

5.1 Operation

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

Table 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	Non-cacheable, 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.

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 a non-cacheable, non-buffered (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.

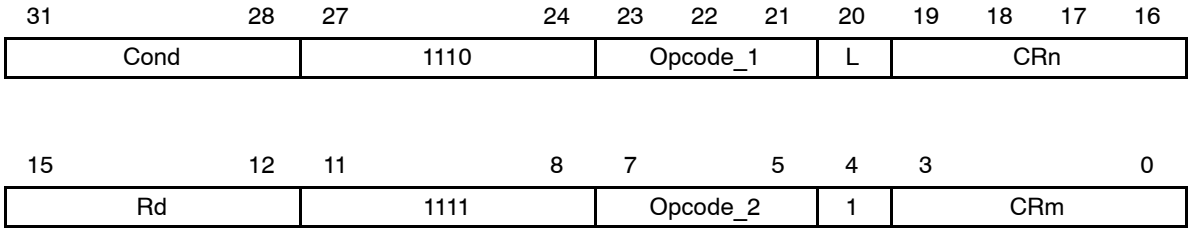
6 Coprocessor 15

TI925T operation and configuration are controlled with coprocessor instructions, configuration pins, and the MMU translation tables. The co-processor 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.

6.1 CP15 Access

The CP15 defines 16 registers. Table 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 contains the instruction bit pattern of the MCR and MRC instructions.

Figure 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.

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 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

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 should be zero (SBZ) when reading.

Writing to register 0 is ignored.

Table 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 5. CP15 ID Register

Bits	Field	Description
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 6. CP15 Cache Information Register (CIR)

Bits	Field	Value	Description
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 to operations with a 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 later in this document.
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 6. CP15 Cache Information Register (CIR) (Continued)

Bits	Field	Value	Description
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 6. CP15 Cache Information Register (CIR) (Continued)

Bits	Field	Value	Description
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 the I-cache is really a 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

The cache information 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 7. CP15 Control Register

Bits	Field	Value	Description
31–15			Reserved: These bits should not be relied upon for any particular value in these bit locations during a read (they should be masked properly). These bits should be written 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 7. CP15 Control Register (Continued)

Bits	Field	Value	Description
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 24).
8	S		System protection. This bit modifies the MMU protection system (see Table 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 7. CP15 Control Register (Continued)

Bits	Field	Value	Description
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.

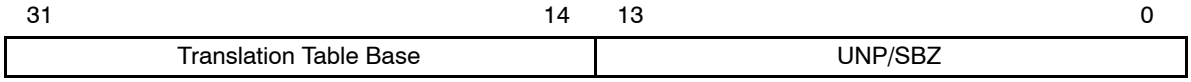
The following code segment example shows correct MMU enabling which takes into account the latency to transition to virtual addressing:

```
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.

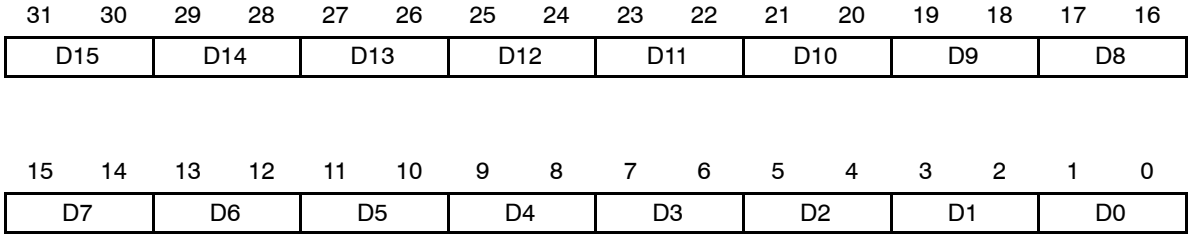
Figure 3. Format of the CP15 Translation Table Base Register



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 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 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 8. Domain Configuration

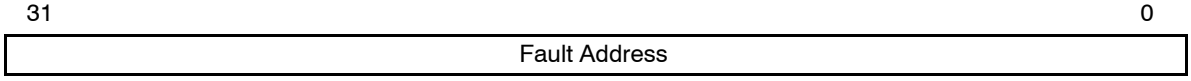
Value	Access Type	Description
0b00	No access	Any access generates a domain fault.
0b01	Client	Access rights are checked against the permission given by the page descriptor.
0b10	Reserved	Behaves like no access
0b11	Manager	The access rights are not checked; permission faults cannot be generated.

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 9. CP15 Fault Status Register

Bits	Field	Description
31–9	UNP/SB	Reserved: The values in these particular bit locations cannot be relied upon during a read (they must be masked properly). These bits should be written 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 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 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.

6.2.2 Cache Operations

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

Table 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 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 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

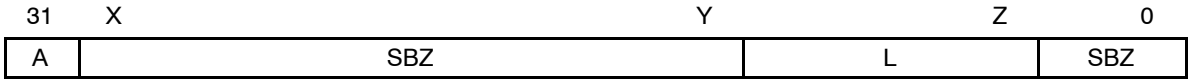
Notes: 1) Flush I- and D-cache operations invalidate all entries in the I-cache and D-cache respectively. The flush D-cache also discards any dirty lines present in the D-cache.

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 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 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:

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

$y = 8 + CIR[20-18] - CIR[17-15]$

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

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.

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 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

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.

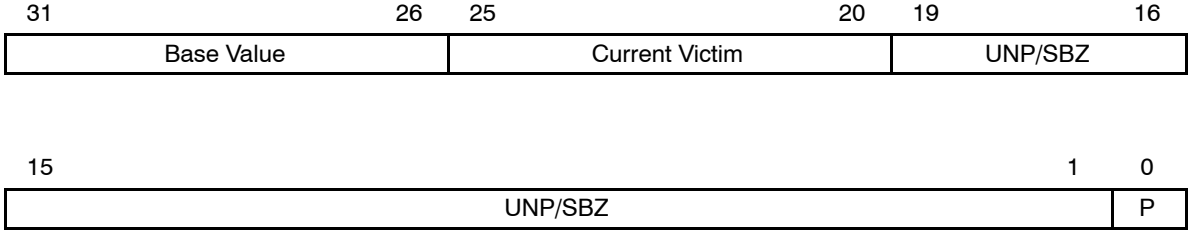
Reading register 10 returns the value of the TLB victim counter base value register, the current value of the victim counter, and the state of the preserved bit. Bits 20−1 are unpredictable when read.

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 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 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).

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.

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 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.

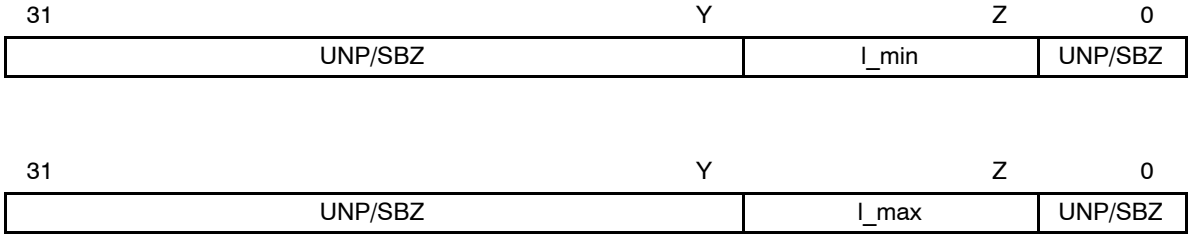
Table 14. TI925T Configuration Register

Bits	Field	Value	Description
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 14. TI925T Configuration Register (Continued)

Bits	Field	Value	Description
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 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 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 9. Format of the Thread-ID Register

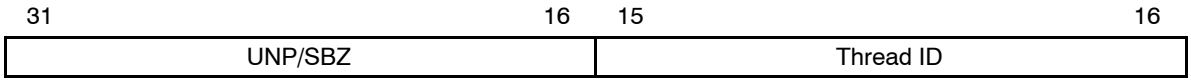


Table 15. TI925T_status Register

Bits	Field	Description
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.

Table 15. TI925T_status Register (Continued)

Bits	Field	Description
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.

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.

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 access is permitted. If 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 the TI925T.

In the event of a TLB miss (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 in the TLB at a random location.

Note:

Because 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).

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 or coarse tables).
- The second-level tables can hold large, small, and tiny page translations entries.

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 sub-pages (single access permission rights).

7.4 MMU Program-Accessible Registers

The system control coprocessor (CP15) registers listed in Table 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 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) hold a 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 6, *Coprocessor 15* for more details on the CP15).

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.

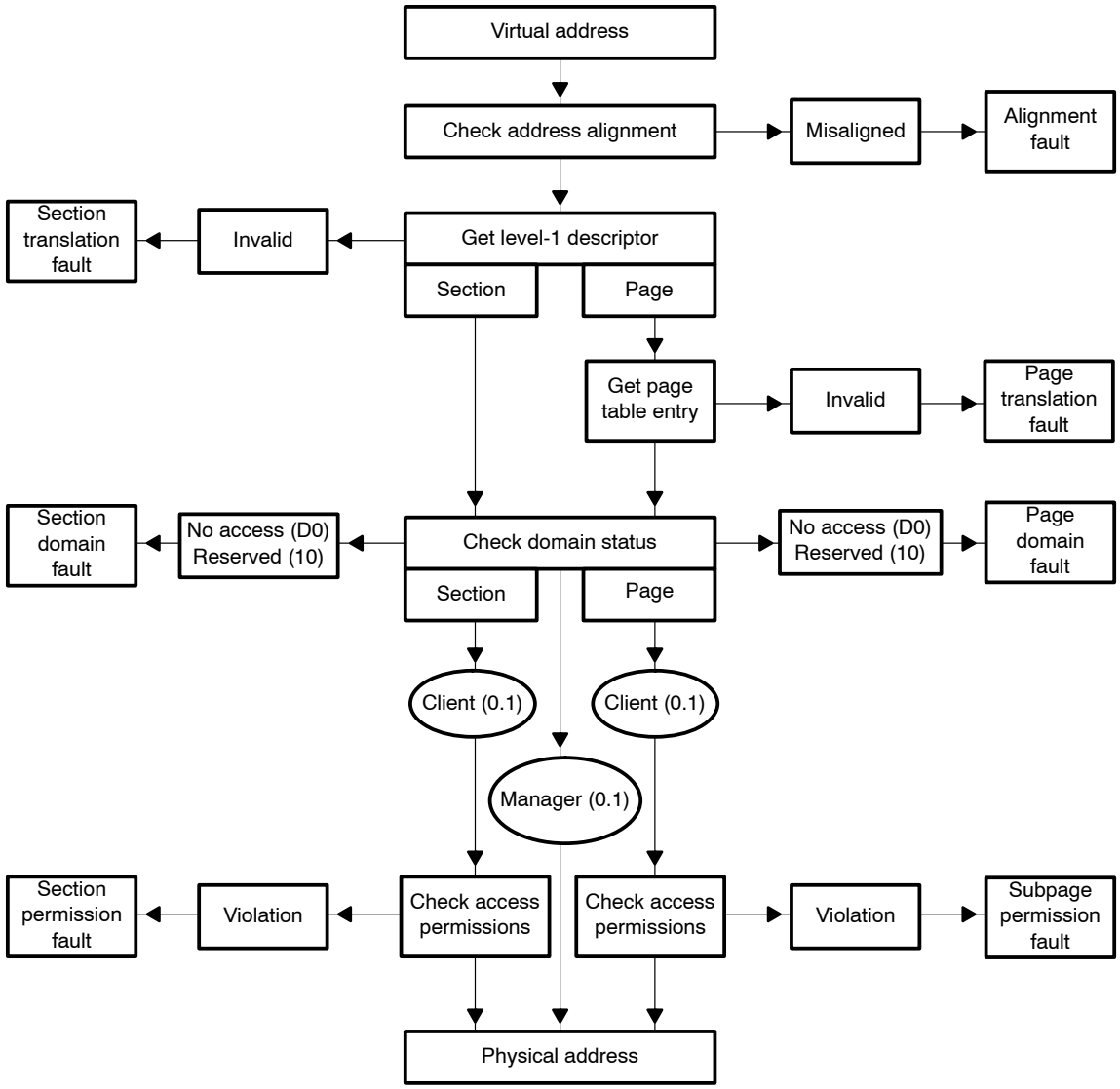
There are four routes by which the address translation (hence access permission) takes place. The route taken depends on whether the address in question has been marked as a section-mapped access or a page-mapped access. There are three sizes of page-mapped access (large, small, and tiny pages). However, 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.

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 10).

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

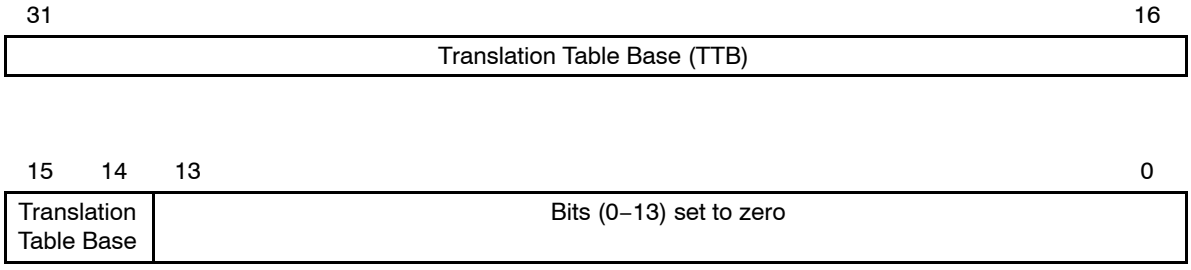
Figure 10. Address Translation Process



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 11. Translation Table Base Register

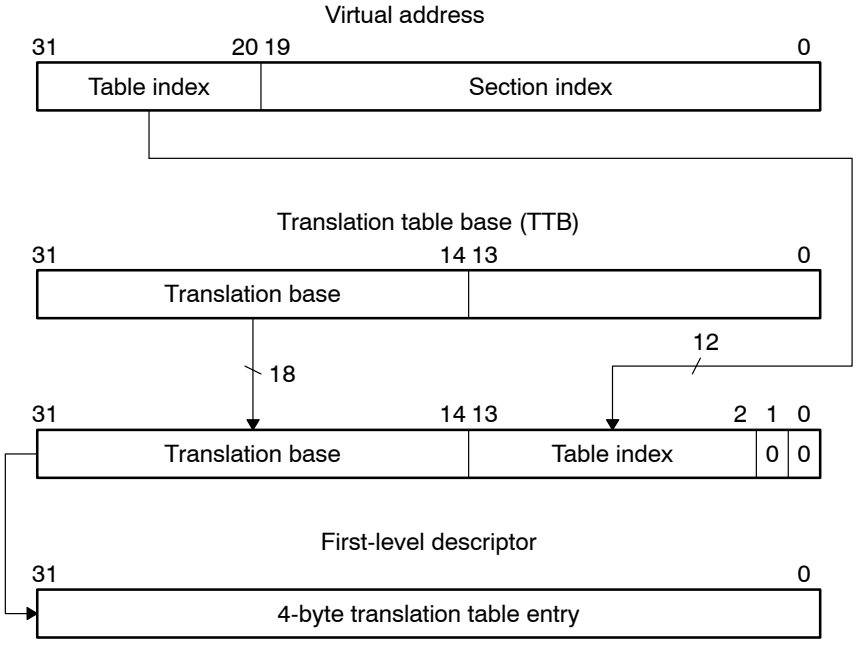


The translation table has up to 4096, 32-bit entries, each describing 1M byte of virtual memory. This allows the addressing of up to 4G bytes of virtual memory.

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 12) by accessing the translation table level-1 descriptors (see Section 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.

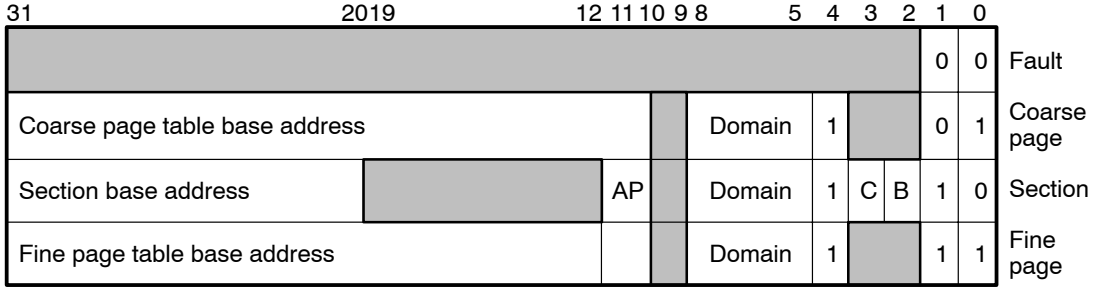
Figure 12. Accessing the Translation Table Level-1 Descriptors



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 13.

Figure 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 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 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 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 19. Level-1 Coarse Page Table Descriptor

Bits	Field	Decription
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.

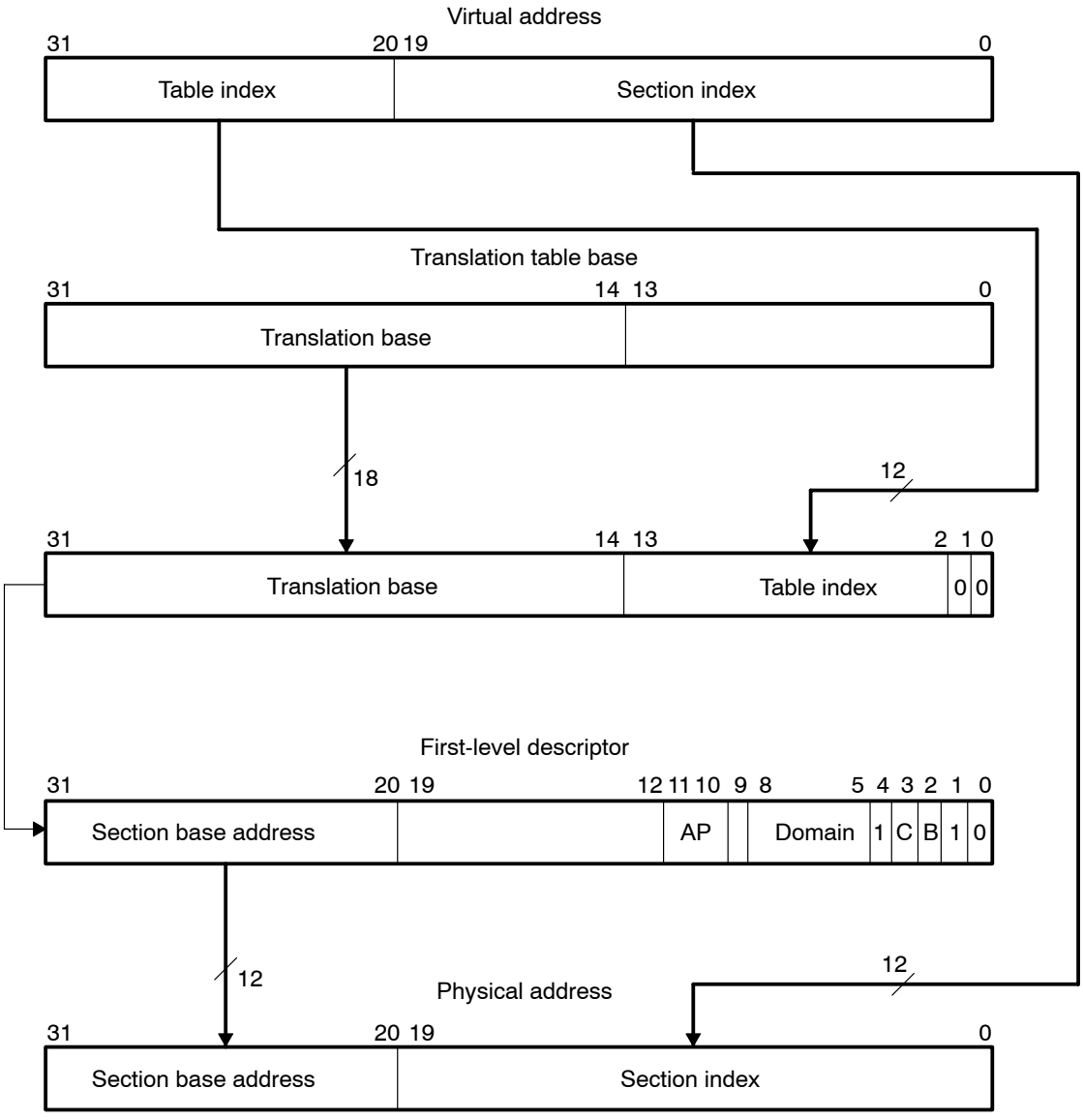
Table 20. Level-1 Section Descriptor

Bits	Field	Decription
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 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.

7.6.4 Translating Section References

Figure 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.

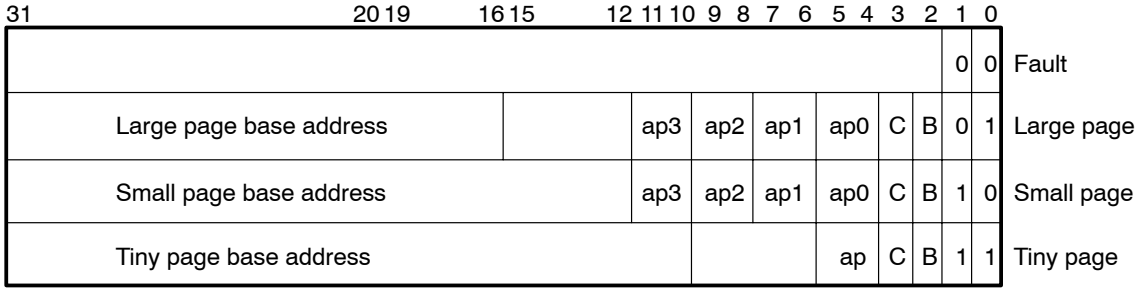
Figure 14. Section Translation



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 15 shows the format of level-2 descriptors.

Figure 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 21. Level-2 Section Descriptor

Bits	Field	Decription
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	Access permissions (ap3-ap0) must be specified 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 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.

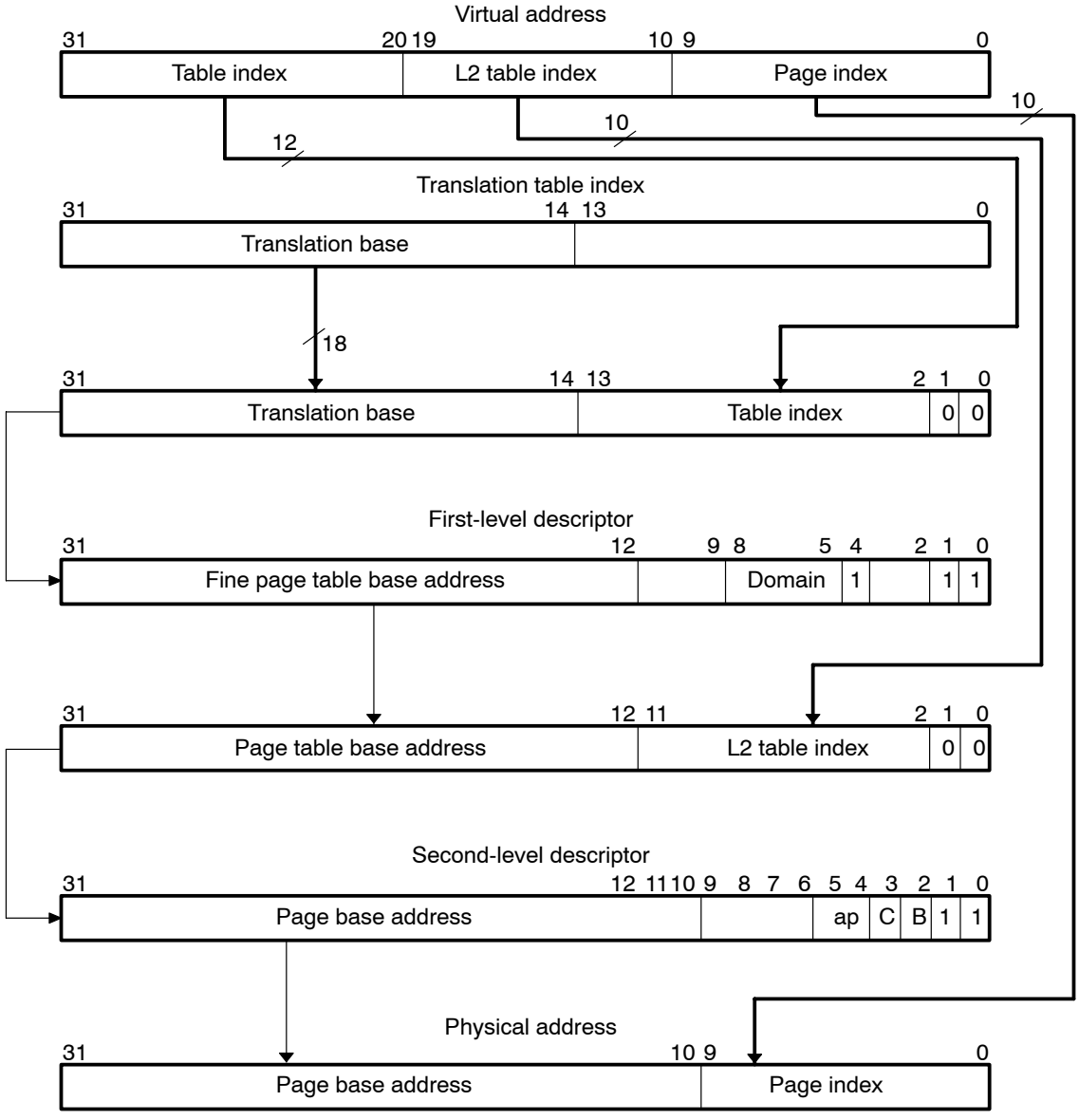
Table 22. Interpreting Page Table Entry Bits 1–0

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

7.6.6 Translating Tiny Pages References

Figure 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.

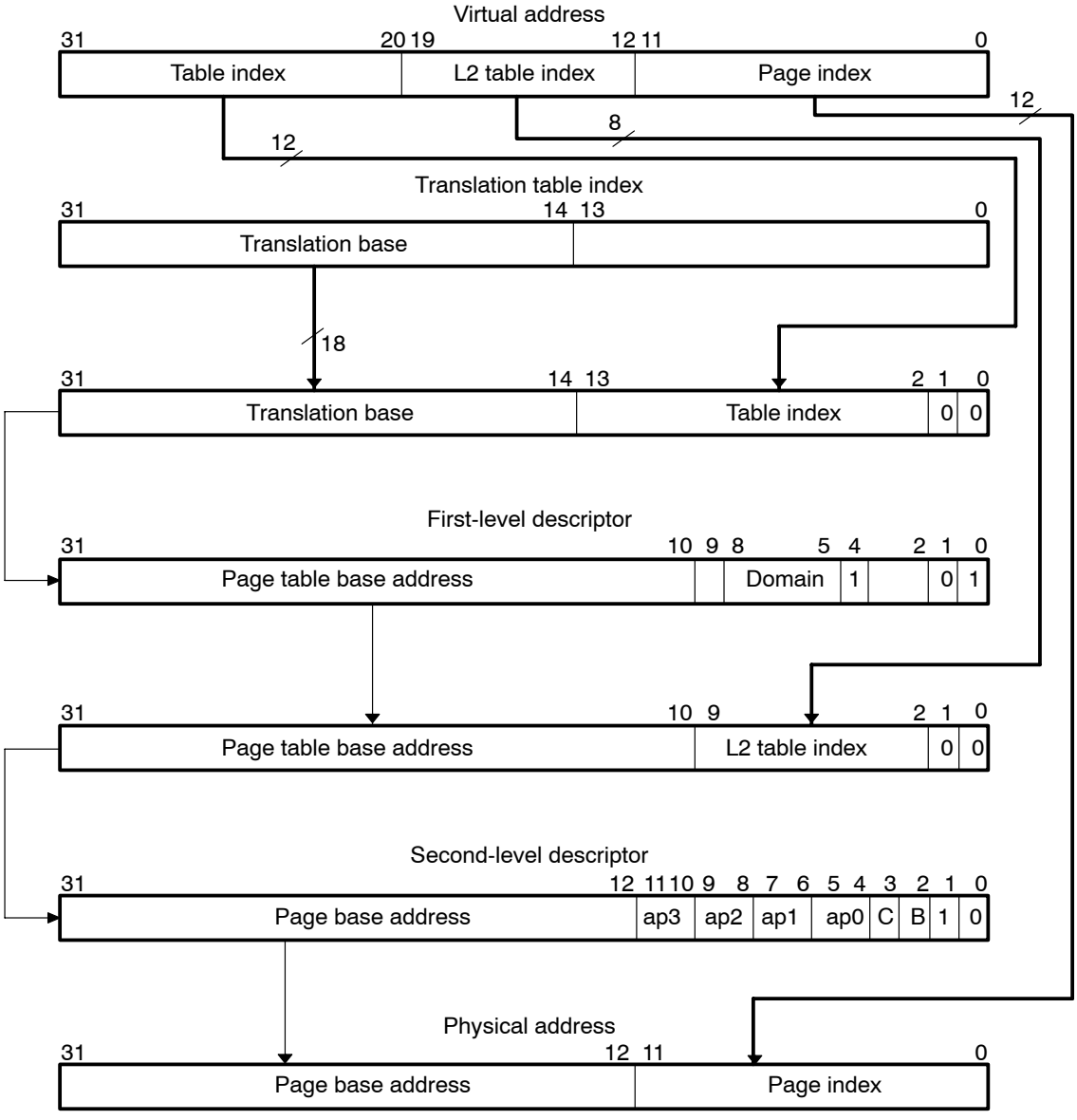
Figure 16. Tiny Page Translation



7.6.7 Translating Small Page References

Figure 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 17. Small Page Translation



7.6.8 Translating Large Page References

Figure 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.

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.

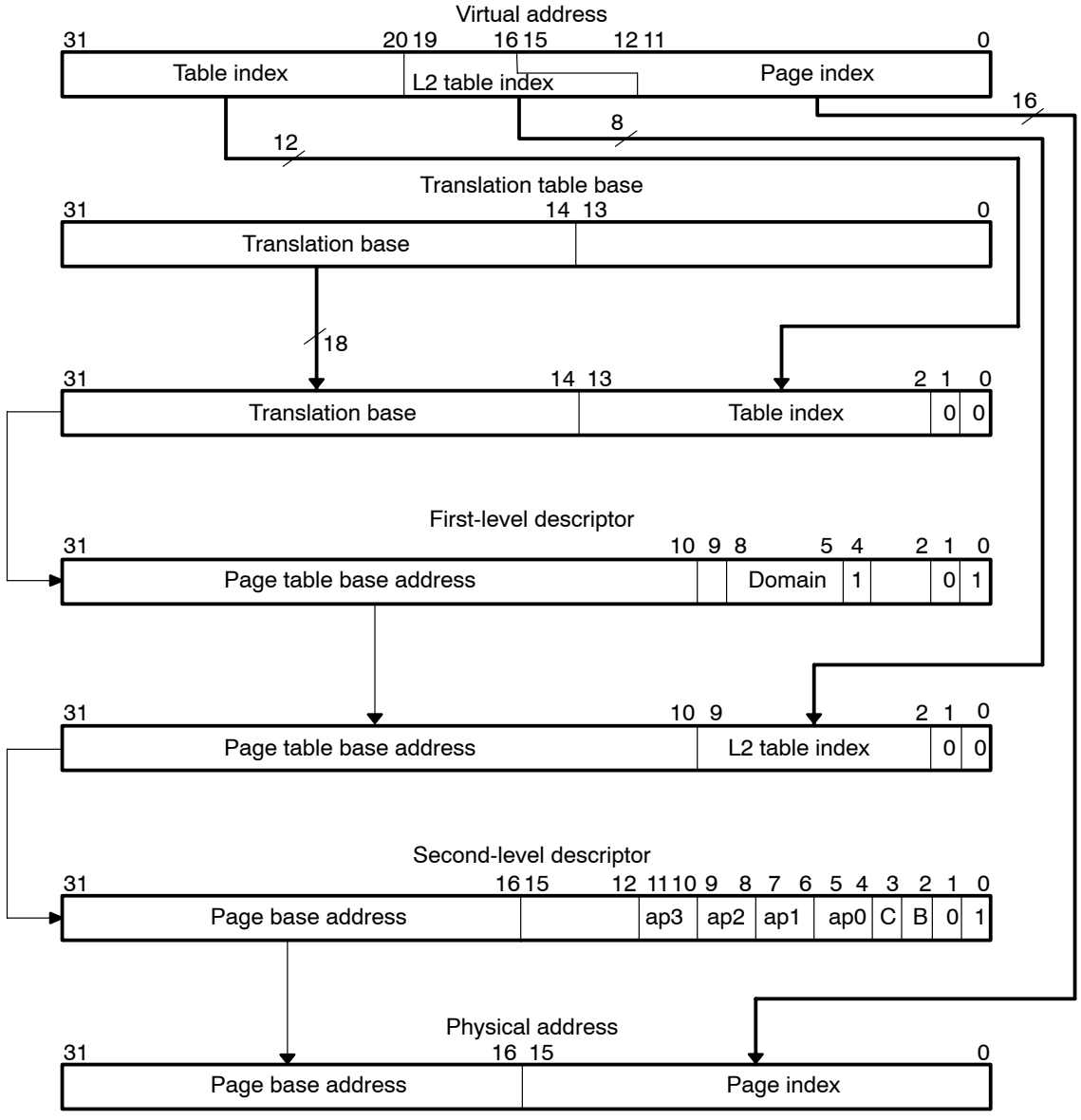
When the MMU is off, the only fault generated is the alignment fault.

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 pre-fetch 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.

MPU instructions are prefetched, so a prefetch abort simply flags the instruction as it enters the instruction pipeline. An abort does not occur, because a previously fetched instruction could render the rest of the pipe information 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 18. Large Page Translation



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 (VA) 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 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 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.

7.9 Domain Access Control

MMU accesses are primarily controlled via domains. There are 16 domains, and each domain is defined by a 2-bit field. 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 19. Domain Access Control Register Format

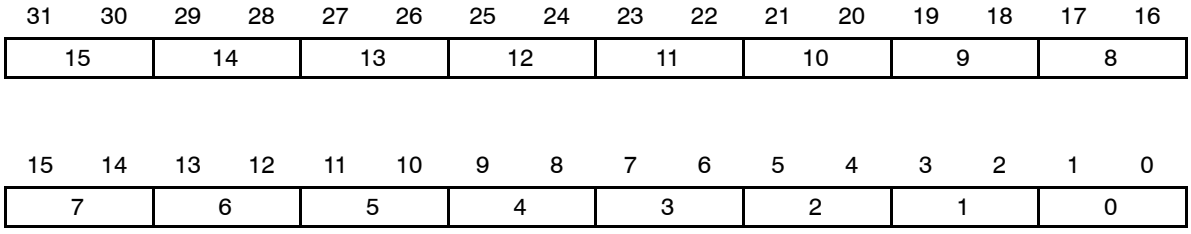


Table 24 defines how the bits within each domain are interpreted to specify the access permissions.

Table 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.

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 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

Table 25. Interpreting Access Permission (Continued)

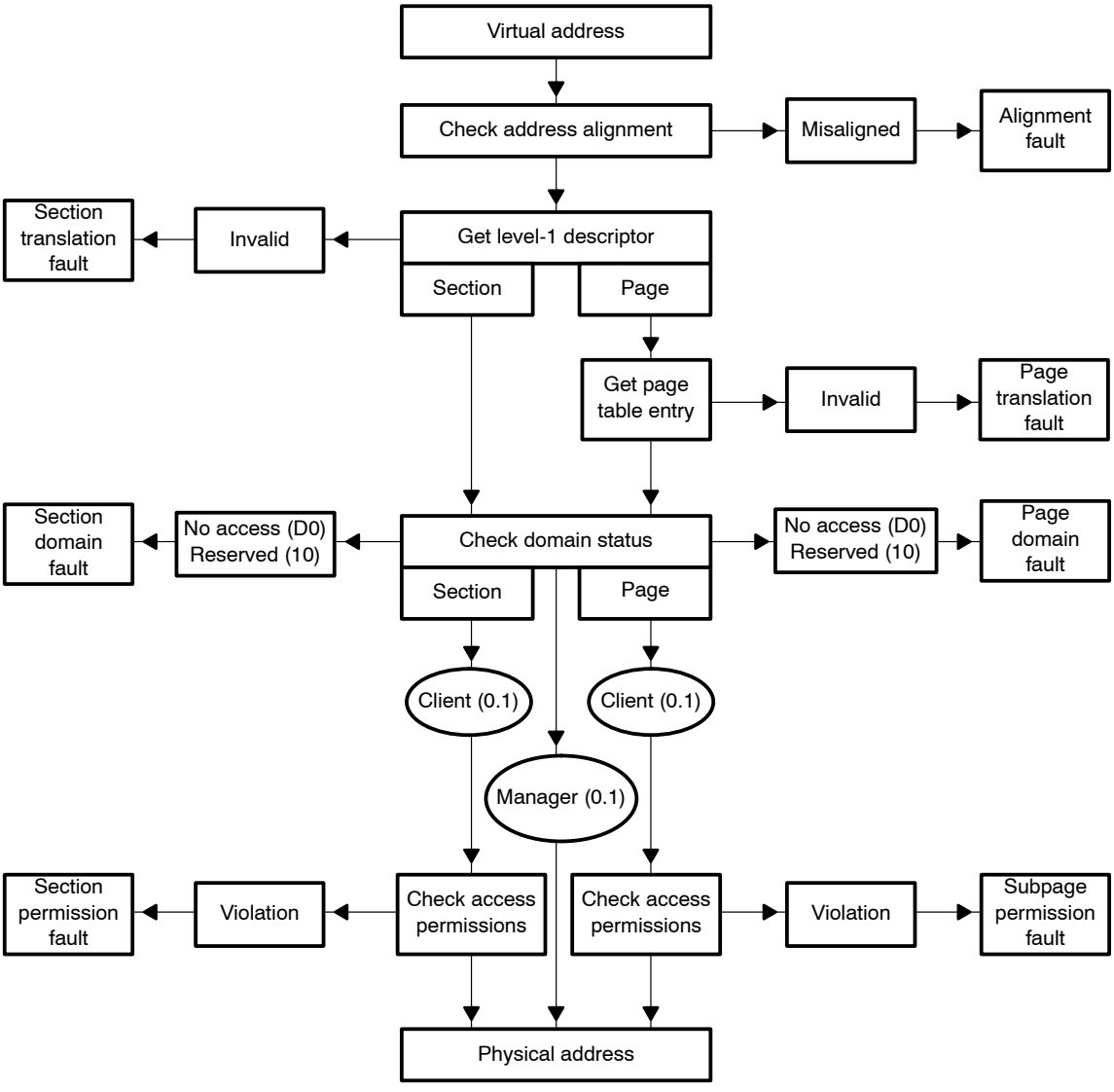
Domain	AP	S	R	Supervisor	User	Description
01	00	1	0	Read only	No access	Supervisor read only permitted
01	00	0	1	Read only	Read only	Any write generates a permission fault.
01	00	1	1	Reserved	Reserved	Generates a permission fault
01	01	x	x	Read/write	No access	Access allowed only in supervisor mode.†
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 the 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.

7.11 Fault Checking Sequence

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

Figure 20. Sequence for Checking Faults



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.

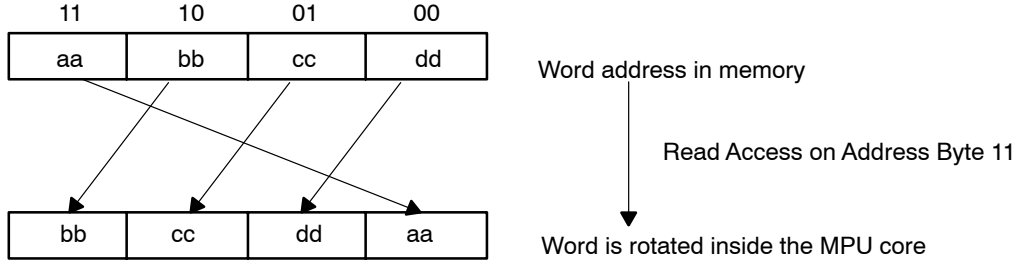
Instruction fetches do not generate alignment faults; they always access memory on 32-bit word boundaries.

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 21 is an example of read-word access on byte 11.

Figure 21. Nonaligned Read-Word Access



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.

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).

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 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.

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.

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

- 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.

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.

8 DSP Memory Management Unit

The MMU is used when the DSP software accesses external memory. This memory can be mapped on any OMAP5910 address space, on the internal SRAM, or on an external SDRAM. The DSP MMU translates addresses coming from the DSP (virtual addresses) to addresses mapped by the Traffic Controller. The MMU is used when the DSP accesses external memory.

The DSP MMU can map 16M bytes of DSP addresses (virtual addresses) to any area in the entire 4G bytes of system memory (physical address). The physical address space includes internal and external memory acessed through the Traffic Controller.

If memory protection or memory access violation occurs, the DSP MMU generates an IRQ_28 interrupt to the MPU second-level interrupt handler. The cause of the violation can be found in the MMU fault address and fault status registers. Both MPU and DSP can configure the DSP MMU using the TI Peripheral Bus registers. Typically, the MPU initializes the DSP MMU at boot time. The DSP MMU registers are listed in Table 26.

Table 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

Table 26. DSP Memory Management Unit Registers (Continued)

Name	Description	R/W	Size	Address	Reset Value
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
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 27. Prefetch Register (PREFETCH_REG)) – Offset Address (hex): 00

Bits	Description	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 28. Prefetch Status Register (WALKING_ST_REG) – Offset Address (hex): 04

Bits	Description	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 29. Control Register (CNTL_REG) – Offset Address (hex): 08

Bits	Description	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, the walking table logic is enabled. 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 30. Fault Address Register MSB (FAULT_AD_H_REG) – Offset Address (hex): 0C

Bits	Description	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 31. Fault Address Register LSB (FAULT_AD_L_REG) – Offset Address (hex): 10

Bits	Description	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 32. Fault Status Register (F_ST_REG)) – Offset Address (hex): 14

Bits	Description	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 33. IT Acknowledge Register (IT_ACK_REG) – Offset Address (hex): 18

Bits	Description	Size	Access	Value at Hardware Reset
15–1	Reserved	15		
0	a write of 1 to this bit acknowledges the interrupt and clears the bit sutomatically. A write of 0 has no effect.	1	W	0

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

Bits	Description	Size	Access	Value at Hardware Reset
15–0	MSB of TTB	16	R	0

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

Bits	Description	Size	Access	Value at Hardware Reset
15−7	LSB of TTB	9	R	0
6−0	Reserved	7		

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

Bits	Description	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 37. Load Entry in TLB Register (LD_TLB_REG) − Offset Address (hex): 28

Bits	Description	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 38. CAM Entry Register MSB (CAM_H_REG) − Offset Address (hex): 2C

Bits	Description	Size	Access	Value at Hardware Reset
15−6	Reserved	10		
5−0	Table index level-1 MSB	6	R/W	0

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

Bits	Value	Description	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
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 40. RAM Entry Register MSB (RAM_H_REG) − Offset Address (hex): 34

Bits	Description	Size	Access	Value at Hardware Reset
15−0	MSB physical address	16	R/W	0

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

Bits	Description	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 42. Global Flush Register (GFLUSH_REG) – Offset Address (hex): 3C

Bits	Description	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 43. Individual Flush Register (FLUSH_ENTRY_REG) – Offset Address (hex):40

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

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

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

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

Bits	Value	Description	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
3		Preserved bit	1	R/W	0
	0	CAM entry not preserved			

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

Bits	Value	Description	Size	Access	Value at Hardware Reset
	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 46. RAM Entry Register MSB (READ_RAM_H_REG) – Offset Address (hex): 4C

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

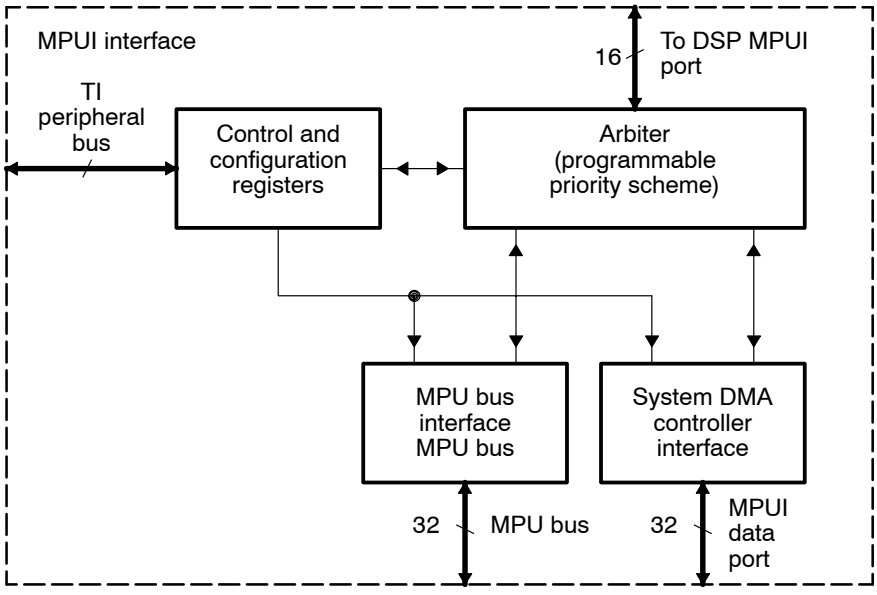
Table 47. RAM Entry Register LSB (READ_RAM_L_REG) – Offset Address (hex): 50

Bits	Description	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		

9 MPU Interface

The MPU interface (MPUI) allows the TI925T and the system DMA controller to communicate with the DSP and its peripherals (except the private peripherals) via the DSP MPUI port (part of the DSP); see Figure 22. Through the MPU interface (MPUI), the MPU and the system DMA controller can access the entire DSP memory space (16M bytes) including the 128K byte DSP I/O space. Figure 1−22 shows the block diagram of the MPUI.

Figure 22. MPUI Simplified Block Diagram



9.1 Functional Features

The MPUI supports the following features:

- Four access modes:
  - Shared-access mode (SAM) for SARAM, DARAM, memory interface access
  - Shared-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, and system DMA) 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 the DSP's entire 16 M byte memory space
- 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 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: a portion of the SRAM blocks can be shared by the DSP and MPU through the MPUI. The MPUI configuration register (DSP_MPUI_CONFIG) defines the memory blocks (SARAM 0,1,2....) to be shared (see Table 1–56).
- 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(Host-Only Mode) is more efficient than SAM(Single-Access Mode), 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

The MPUI port can access only memory space inside the DSP. Accessing via the DSP MMU is prohibited. The MPU system and system DMA must access EMIF, EMIFF, and IMIF through the traffic controller.

9.2 MPUI Registers

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

Table 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 49. Control Register (CTRL_REG) − Offset: x00

Bits	Value	Description	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			
	01	Word swap only for non-MPUIMEM accesses			
	10	Word swap only for MPUIMEM 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. Note: the lower the number, the higher the priority.	3	R/W	000
	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-MPUIMEM accesses			
	10	Byte swap for all accesses			
	11	Byte swap only for MPUIMEM accesses			
15−8		MPUI bus access time out	8	R/W	0xFF
7−4		Division factor of MPUIF_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 49. Control Register (CTRL_REG) – Offset: x00 (Continued)

Bits	Value	Description	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, other modules such as the TIPB Bridge can also generate the IRQ_ABORT interrupt.

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

Bits	Description	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 51. Debug Data Register (DEBUG_DATA) – Offset: x08

Bits	Description	Size	Access	Value at Hard-ware Reset
31–0	The 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 52. Debug Flag Register (DEBUG_FLAG) – Offset: x0C

Bits	Value	Description	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 52. Debug Flag Register (DEBUG_FLAG) – Offset: x0C (Continued)

Bits	Value	Description	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 53. Status Register (STATUS_REG) – Offset: x10

Bits	Value	Description	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 53. Status Register (STATUS_REG) – Offset: x10 (Continued)

Bits	Value	Description	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 54. DSP Status Register (DSP_STATUS_REG) – Offset: x14

Bits	Value	Description	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 54. DSP Status Register (DSP_STATUS_REG) – Offset: x14 (Continued)

Bits	Value	Description	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 55. DSP Boot Configuration Register (DSP_BOOT_CONFIG) – Offset: x18

Bits	Description	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	4	R/W	0

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

Bits	Description	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 the formula: API_SIZE/2) * 8K bytes, starting from SARAM0	16	R/W	0xFFFF

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

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

	SARAM		
APISIZE (15..0)	11–8	7–4	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

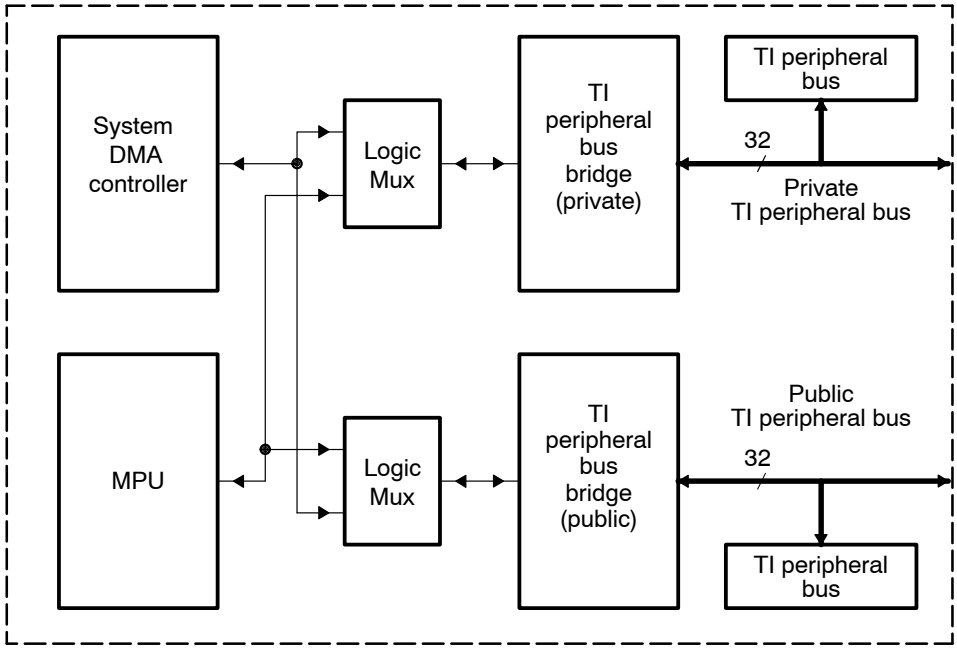
Notes: 1) 0: Shared-access RAM
2) 1: Host-only RAM (no DSP access)

10 MPU TI Peripheral Bus Bridges

The MPU TI peripheral bus (TIPB) bridges (see Figure 23) connect the TI925T to its peripherals. Two MPU TIPBs, one private and one public, are implemented to reduce access latency and improve system performance. Concurrent transfers are possible if there are no resource conflicts; for example, when 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 23. MPU TI Peripheral Bus Bridge Connections



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.

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.

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, bits 3–0 (strobe 0) and bits 7–4 (strobe 1) of the TIPB control register (TIPB_CNTL) can be used to configure the access factor and consequently the strobe frequencies (as shown in Table 58).

Table 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.

10.4 MPU Posted Write

The MPU can perform a posted write. When posted write is enabled inside the MPU_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.

10.5 Pipeline Mode

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

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.

10.7 TIPB Bridge Registers

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

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

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

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

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

Bits	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	Reserved – always set to 1.	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 65. Address Debug Register (ADDRESS_DBG) – Offset: x10

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

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

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

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

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

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

Bits	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

11 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 24. 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.

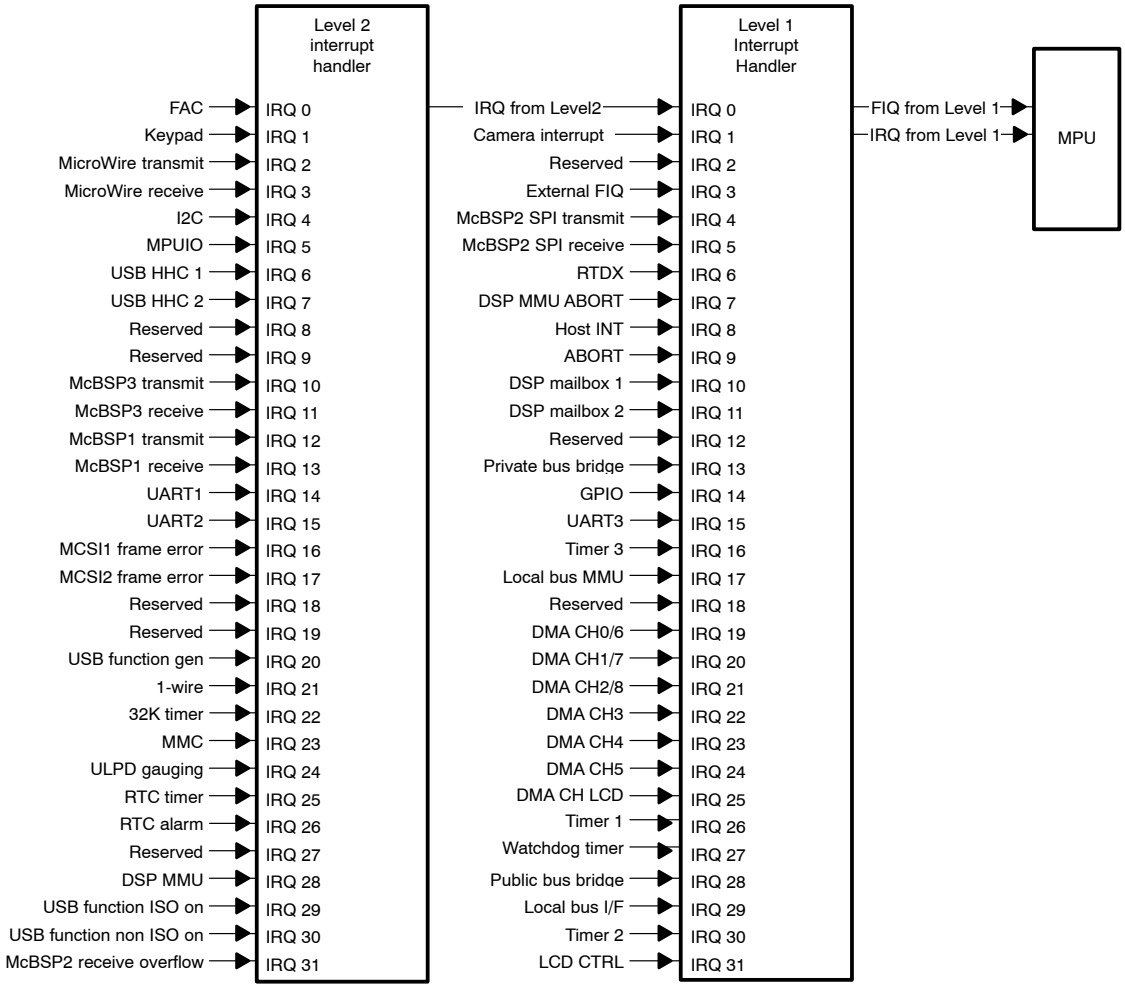
11.1 MPU Level-1 Interrupt Handler

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 76). All interrupts are maskable (individually enabled and disabled via the mask interrupt register (MIR)—see Table 72) with an internal register. The interrupt source information can be read back

from the ITR register (see Table 71, Table 72, and Table 73). 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 24) 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 24. MPU Interrupt Handlers



11.2 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.

12 Level-1 and Level-2 Interrupt Mapping

Table 69 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 69. 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 69. 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 69. 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.

13 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 70). Table 71 through Table 77 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 70. 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

Table 70. Interrupt Handler Registers (Continued)

Name	Description	R/W	Bits	Offset	Reset Value
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
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 71. Interrupt Input Register (ITR)

Bits	Field	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 72. Mask Interrupt Register (MIR)

Bits	Field	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 73. Binary-Coded Source IRQ Register (SIR_IRQ_CODE)

Bits	Field	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 74. Binary-Coded Source FIQ Register (SIR_FIQ_CODE)

Bits	Field	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 75. Control Register (CONTROL_REG)

Bits	Field	Description	Reset Value
1	NEW_FIQ_AGR	New FIQ agreement. Writing a 1 resets FIQ output, clears source FIQ register, and enables new IRQ generation.	0
0	NEW_IRQ_AGR	New IRQ agreement. Writing a 1 resets IRQ output, clears source IRQ register, and enables new IRQ generation.	0

Table 76. Interrupt Level Registers (ILR0...ILR31)

Bits	Field	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 77. Interrupt Set Register (ISR)

Bits	Field	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

14 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.

14.1 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)

14.2 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.

14.3 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.

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 78.

Table 78. 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.

14.4 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 SPRU680 MMC/SD Reference Guide.

15 OMAP5910 Configuration Registers

Table 79 lists the 32-bit read/write configuration registers. Table 80 through Table 102 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 79. Configuration Registers

Registers	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 80. Functional Multiplexing Control 0 Register (FUNC_MUX_CTRL_0)

Bits	Field	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	Dynamic 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 80. Functional Multiplexing Control 0 Register (FUNC_MUX_CTRL_0) (Continued)

Bits	Field	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 (7) 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 80. Functional Multiplexing Control 0 Register (FUNC_MUX_CTRL_0) (Continued)

Bits	Field	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/$\overline{\text{EXT\_FIQ}}$ OMAP5910 input pin	R/W	0x0
		1	Allows AND gating of OMAP5910 inputs with BFAIL/$\overline{\text{EXT\_FIQ}}$ 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/$\overline{\text{EXT\_FIQ}}$ OMAP5910 input pin	R/W	0x0
		1	Allows AND gating of outputs with BFAIL/$\overline{\text{EXT\_FIQ}}$ 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 81. Functional Multiplexing Control 1 Register (FUNC_MUX_CTRL_1)

Bits	Field	Description	R/W	Reset Value
31–0	RESERVED	Reserved. These bits must always be written as 0.	R/W	0x00000000

Table 82. Functional Multiplexing Control 2 Register (FUNC_MUX_CTRL_2)

Bits	Field	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 83. Compatibility Mode Control 0 Register (COMP_MODE_CTRL_0)

Bits	Field	Description	R/W	Reset Value
31–16	RESERVED	Reserved. 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 84. Functional Multiplexing Control 3 Register (FUNC_MUX_CTRL_3)

Bits	Field	Description	R/W	Reset Value
31–0	RESERVED	Reserved. These bits must always be written as 0.	R/W	0x0

Table 85. Functional Multiplexing Control 4 Register (FUNC_MUX_CTRL_4)

Bits	Field	Description	R/W	Reset Value
31–30	RESERVED	Reserved. 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
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

Table 85. Functional Multiplexing Control 4 Register (FUNC_MUX_CTRL_4) (Continued)

Bits	Field	Description	R/W	Reset Value
20–18	RESERVED	Reserved. 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. These bits must always be written as 0.	R/W	0x0

Table 86. Functional Multiplexing Control 5 Register (FUNC_MUX_CTRL_5)

Bits	Field	Description	R/W	Reset Value
31–30	RESERVED	Reserved. 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
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

Table 86. Functional Multiplexing Control 5 Register (FUNC_MUX_CTRL_5) (Continued)

Bits	Field	Description	R/W	Reset Value
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
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 87. Functional Multiplexing Control 6 Register (FUNC_MUX_CTRL_6)

Bits	Field	Description	R/W	Reset Value
31–30	RESERVED	Reserved. 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
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

Table 87. Functional Multiplexing Control 6 Register (FUNC_MUX_CTRL_6) (Continued)

Bits	Field	Description	R/W	Reset Value
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 88. Functional Multiplexing Control 7 Register (FUNC_MUX_CTRL_7)

Bits	Field	Description	R/W	Reset Value
31–21	RESERVED	Reserved. 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 89. Functional Multiplexing Control 8 Register (FUNC_MUX_CTRL_8)

Bits	Field	Description	R/W	Reset Value
31–30	RESERVED	Reserved. 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. 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 $\overline{\text{UWIRE.CS3}}$ 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 $\overline{\text{UWIRE.CS0}}$ 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 90. Functional Multiplexing Control 9 Register (FUNC_MUX_CTRL_9)

Bits	Field	Description	R/W	Reset Value
31–30	RESERVED	Reserved. 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. 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. 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 91. Functional Multiplexing Control A Register (FUNC_MUX_CTRL_A)

Bits	Field	Description	R/W	Reset Value
31–27	RESERVED	Reserved. 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. 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 91. Functional Multiplexing Control A Register (FUNC_MUX_CTRL_A) (Continued)

Bits	Field	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 92. Functional Multiplexing Control B Register (FUNC_MUX_CTRL_B)

Bits	Field	Description	R/W	Reset Value
31–21	RESERVED	Reserved. 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. 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 92. Functional Multiplexing Control B Register (FUNC_MUX_CTRL_B) (Continued)

Bits	Field	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. These bits must always be written as 0.	R/W	0x0

Table 93. Functional Multiplexing Control C Register (FUNC_MUX_CTRL_C)

Bits	Field	Description	R/W	Reset Value
31–30	RESERVED	Reserved. 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 93. Functional Multiplexing Control C Register (FUNC_MUX_CTRL_C) (Continued)

Bits	Field	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. 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. 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 94. Functional Multiplexing Control D Register (FUNC_MUX_CTRL_D)

Bits	Field	Description	R/W	Reset Value
31–15	RESERVED	Reserved. 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. 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 $\overline{\text{FLASH.CS2}}$ at reset. The control for this I/O is forced to 000 at reset.	R/W	0x0
5–0	RESERVED	Reserved. These bits must always be written as 0.	R/W	0x0

Table 95. Pulldown Control 0 Register (PULL_DWN_CTRL_0)

Bits	Field	Value	Description (see Note)	R/W	Reset Value
31–29	RESERVED		Reserved. 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. 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

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 95. Pulldown Control 0 Register (PULL_DWN_CTRL_0) (Continued)

Bits	Field	Value	Description (see Note)	R/W	Reset Value
		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. 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		
20–17	RESERVED		Reserved. 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. 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 96. Pulldown Control 1 Register (PULL_DWN_CTRL_1)

Bits	Field	Value	Description (See Note)	R/W	Reset Value
31–30	RESERVED		Reserved. 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. These bits must always be written as 0.	R/W	0x0
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		

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 96. Pulldown Control 1 Register (PULL_DWN_CTRL_1) (Continued)

Bits	Field	Value	Description (See Note)	R/W	Reset Value
24–19	RESERVED		Reserved. 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	RESERVED		Reserved	R/W	0x0
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.		

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 96. Pulldown Control 1 Register (PULL_DWN_CTRL_1) (Continued)

Bits	Field	Value	Description (See Note)	R/W	Reset Value
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.		
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.		

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 96. Pulldown Control 1 Register (PULL_DWN_CTRL_1) (Continued)

Bits	Field	Value	Description (See Note)	R/W	Reset Value
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.		
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.		

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 96. Pulldown Control 1 Register (PULL_DWN_CTRL_1) (Continued)

Bits	Field	Value	Description (See Note)	R/W	Reset Value
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.		
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.		

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 96. Pulldown Control 1 Register (PULL_DWN_CTRL_1) (Continued)

Bits	Field	Value	Description (See Note)	R/W	Reset Value
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.		
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.		

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 96. Pulldown Control 1 Register (PULL_DWN_CTRL_1) (Continued)

Bits	Field	Value	Description (See Note)	R/W	Reset Value
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 97. Pulldown Control 2 Register (PULL_DWN_CTRL_2)

Bits	Field	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		

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 97. Pulldown Control 2 Register (PULL_DWN_CTRL_2) (Continued)

Bits	Field	Value	Description (See Note)	R/W	Reset Value
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		
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		

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 97. Pulldown Control 2 Register (PULL_DWN_CTRL_2) (Continued)

Bits	Field	Value	Description (See Note)	R/W	Reset Value
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.		
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.		

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 97. Pulldown Control 2 Register (PULL_DWN_CTRL_2) (Continued)

Bits	Field	Value	Description (See Note)	R/W	Reset Value
21	RESERVED		Reserved. 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. 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		
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		

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 97. Pulldown Control 2 Register (PULL_DWN_CTRL_2) (Continued)

Bits	Field	Value	Description (See Note)	R/W	Reset Value
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. 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		
11–10	RESERVED		Reserved. 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		

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 97. Pulldown Control 2 Register (PULL_DWN_CTRL_2) (Continued)

Bits	Field	Value	Description (See Note)	R/W	Reset Value
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. These bits must always be written as 0.	R/W	0x0
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. 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 97. Pulldown Control 2 Register (PULL_DWN_CTRL_2) (Continued)

Bits	Field	Value	Description (See Note)	R/W	Reset Value
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. 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 98. Pulldown Control 3 Register (PULL_DWN_CTRL_3)

Bits	Field	Value	Description	R/W	Reset Value
31–14	RESERVED		Reserved. 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		

Table 98. Pulldown Control 3 Register (PULL_DWN_CTRL_3) (Continued)

Bits	Field	Value	Description	R/W	Reset Value
		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		
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. 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		

Table 98. Pulldown Control 3 Register (PULL_DWN_CTRL_3) (Continued)

Bits	Field	Value	Description	R/W	Reset Value
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 99. Gate and Inhibit Control 0 Register (GATE_INH_CTRL_0)

Bits	Field	Value	Description	R/W	Reset Value
31–4	RESERVED		Reserved. 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		
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

Table 99. Gate and Inhibit Control 0 Register (GATE_INH_CTRL_0) (Continued)

Bits	Field	Value	Description	R/W	Reset Value
1	CONF_ SOFTWARE_BVLZ_R		This bit controls software gating and inhibiting of the OMAP5910 I/O, which are gated or inhibited by BFAIL/$\overline{\text{EXT\_FIQ}}$. 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/$\overline{\text{EXT\_FIQ}}$ 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 100. Voltage Control 0 Register (VOLTAGE_CTRL_0)

Bits	Field	Value	Description	R/W	Reset Value
31–3	RESERVED		Reserved. 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 100. Voltage Control 0 Register (VOLTAGE_CTRL_0) (Continued)

Bits	Field	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 101. 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 102. Module Configuration Control 0 Register (MOD_CONF_CTRL_0)

Bits	Field	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		

Table 102. Module Configuration Control 0 Register (MOD_CONF_CTRL_0) (Continued)

Bits	Field	Value	Description	R/W	Reset Value
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.		
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.		

Table 102. Module Configuration Control 0 Register (MOD_CONF_CTRL_0) (Continued)

Bits	Field	Value	Description	R/W	Reset Value
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.		
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.		

Table 102. Module Configuration Control 0 Register (MOD_CONF_CTRL_0) (Continued)

Bits	Field	Value	Description	R/W	Reset Value
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. 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.		
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.		

Table 102. Module Configuration Control 0 Register (MOD_CONF_CTRL_0) (Continued)

Bits	Field	Value	Description	R/W	Reset Value
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. 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 102. Module Configuration Control 0 Register (MOD_CONF_CTRL_0) (Continued)

Bits	Field	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 102. Module Configuration Control 0 Register (MOD_CONF_CTRL_0) (Continued)

Bits	Field	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 an attached device with the bus in an IDLE condition.		
6−1	CONF_MOD_USB_HOST_HMC_MODE_R		USB_HHC port multiplexing control. See SPRU677, *USB Reference Guide*, 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 SPRU677, *USB Reference Guide*.		
0	RESERVED		Reserved. This bit should always be written as 0.	R/W	0x0

16 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.

16.1 Identification Code Register

The identification code register (IDCODE), shown in Table 103, 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 103. 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 104.

Table 104. 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

16.2 Die Identification (ID)

An electrically readable die ID permits tracing of individual dies back to manufacturing data.

The die ID is a 64-bit code.

The die ID can be read by software via the private TIPB (see Table 105).

Table 105. 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

17 MPU Private Peripherals Overview

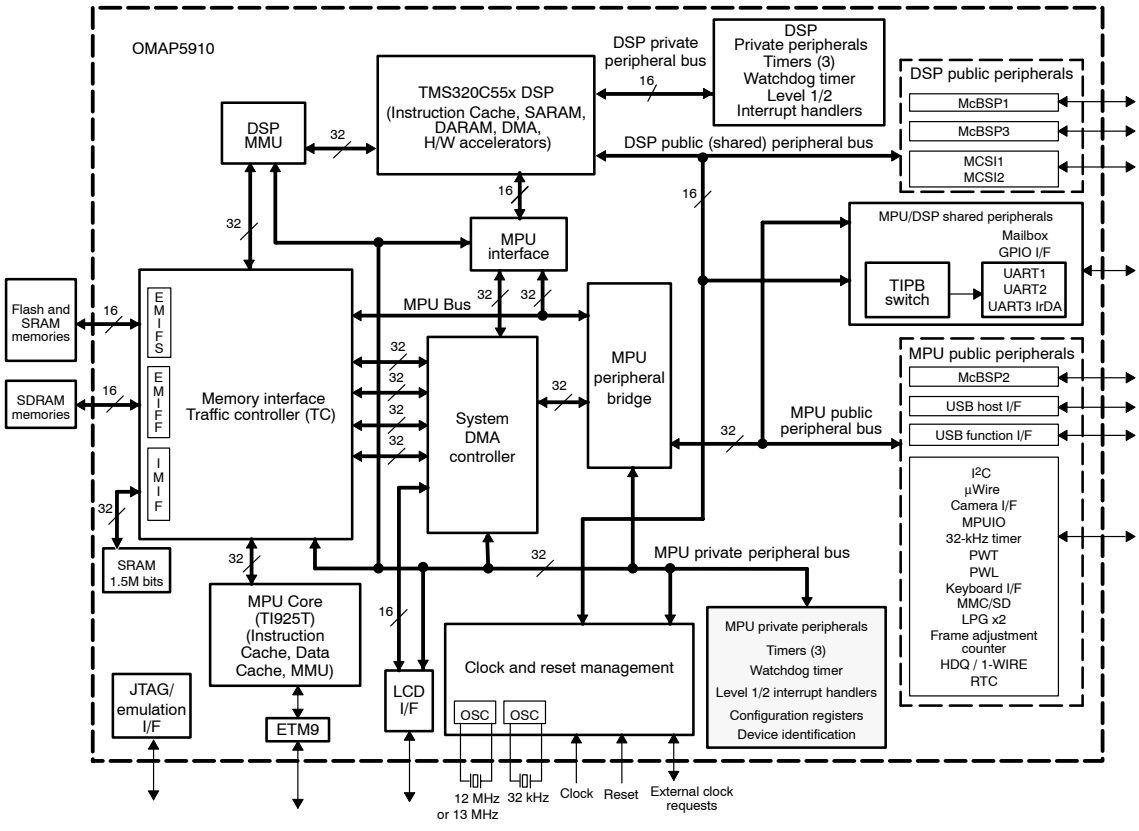
Three standard peripherals are attached to and accessible only by the TI925T RISC processor private bus (TIPB) to provide housekeeping functions for the operating system (OS) and applications. These peripherals include timers, a watchdog timer, and interrupt handlers.

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 25 shows the OMAP5910 device with the MPU private peripherals highlighted. For more information on the MPU private peripherals see:

SPRU682, OMAP Timer Reference Guide

Figure 25. MPU Private Peripherals



18 MPU Public Peripherals Overview

Figure 26 shows the OMAP5910 device with the MPU public peripherals highlighted. For more information about the MPU public peripherals see:

SPRU681, OMAP5910 I^{2}C Controller Reference Guide

SPRU686, OMAP5910 Micro-Wire Interface Reference Guide

SPRU684, OMAP5910 Camera Interface Reference Guide

SPRU683, OMAP5910 Inter-Processor Communications Reference Guide

SPRU682, OMAP5910 Timer Reference Guide

SPRU689, OMAP5910 PWL, PWT, and LED Reference Guide

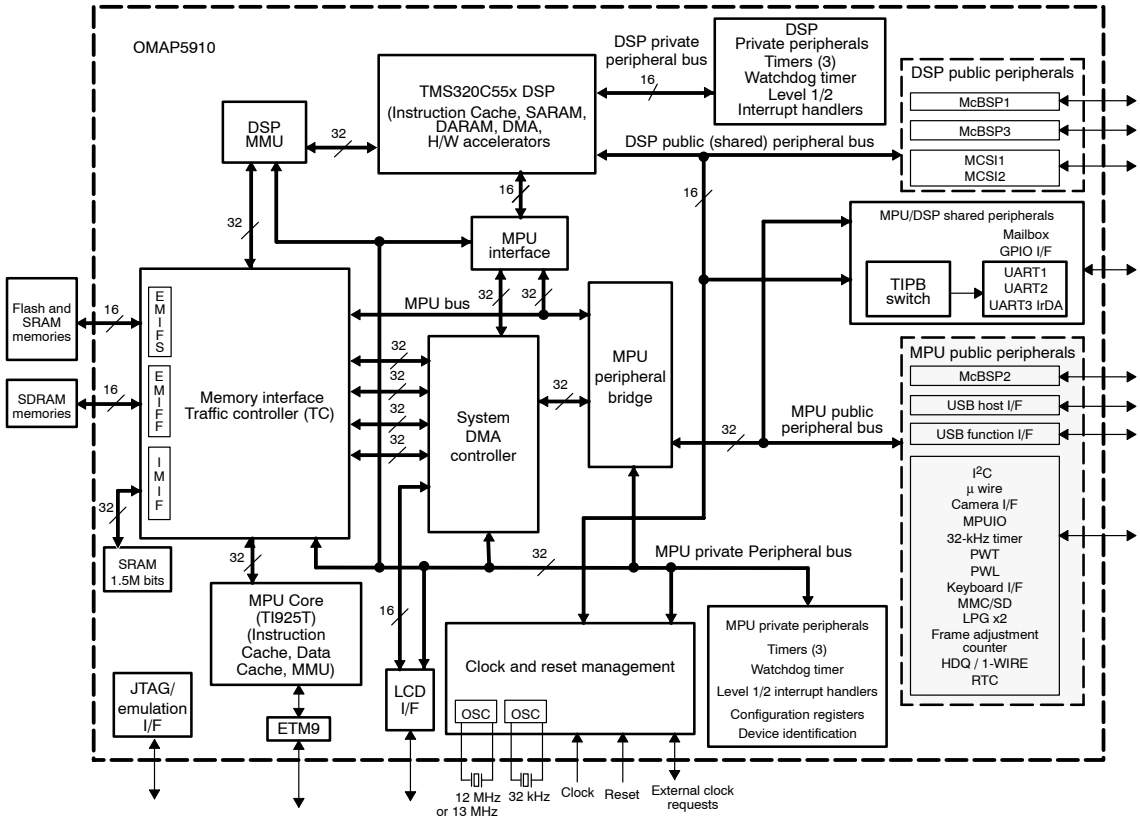
SPRU680, OMAP5910 MMC/SD Reference Guide

SPRU677, OMAP5910 USB and FAC Reference Guide

SPRU688, OMAP5910 HDQ/1-Wire Interface Reference Guide

SPRU687, OMAP5910 RTC Reference Guide

Figure 26. MPU Public Peripherals Area



19 MPU/DSP Peripherals Overview

The OMAP5910 device has five peripherals that appear on both MPU and DSP public peripheral buses:

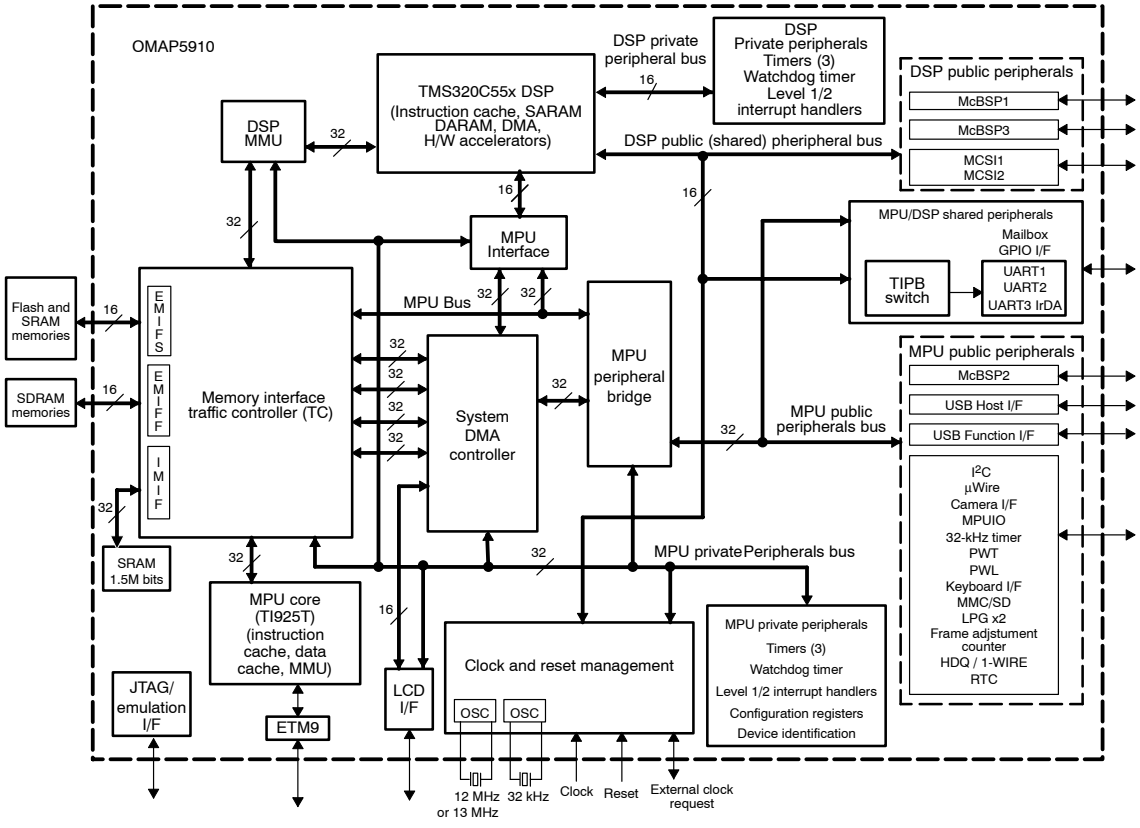
- ❏ Mailbox registers for interprocessor communication
- ❏ General-purpose I/O (GPIO)
- ❏ UART1
- ❏ UART2
- ❏ UART/IrDA

Figure 27 shows the OMAP5910 device with the MPU/DSP peripherals highlighted. For more information on MPU/DSP peripherals see:

SPRU676, OMAP5910 UART Device Reference Guide

SPRU679, OMAP5910 GPIO Reference Guide

Figure 27. Highlight of MPU/DSP Peripherals



20 Endianism Conversion

The TI925T operates in little endian mode and the DSP operates in big-endian mode, so shared data must be converted to their respective formats before any processing is done. Table 106 and Table 107 illustrate the little- and big-endian data formats. In each case, the data is a reference to the little-endian format in which the bytes are numbered from right to left.

Cases when endianism must be addressed:

- ❏ When the DSP accesses outside the DSP subsystem through its MIMU
- ❏ When the MPU accesses the DSP subsystem through the MPUI

Endian 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.

Endian 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 106. 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 107. 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

20.1 Conversion Through the DSP MMU

Swapping buffers are implemented at the boundary between the DSP and the DSP MMU (see Figure 28). 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 107), but the bytes are swapped in order to recognize the data in little-endian mode (see Table 106).

- 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.

- A 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 108), but the 16-bit words are swapped in order to reorganize the 32-bit data in little-endian mode.

- A 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.

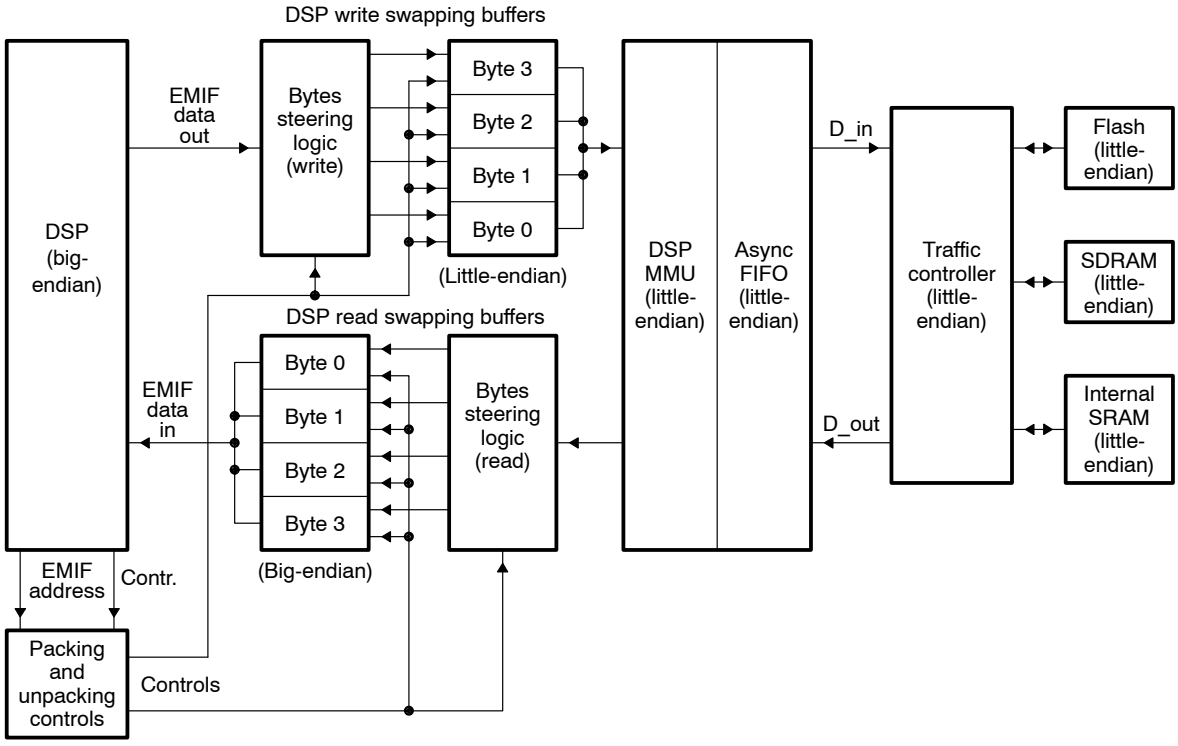
Table 108. DSP Data Format

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 28 shows the endian conversion at the DSP MMU interface boundary. The byte and word swapping is done by decoding the data width, 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 28. DSP Endian Conversion, 32-Bit Aligned Data



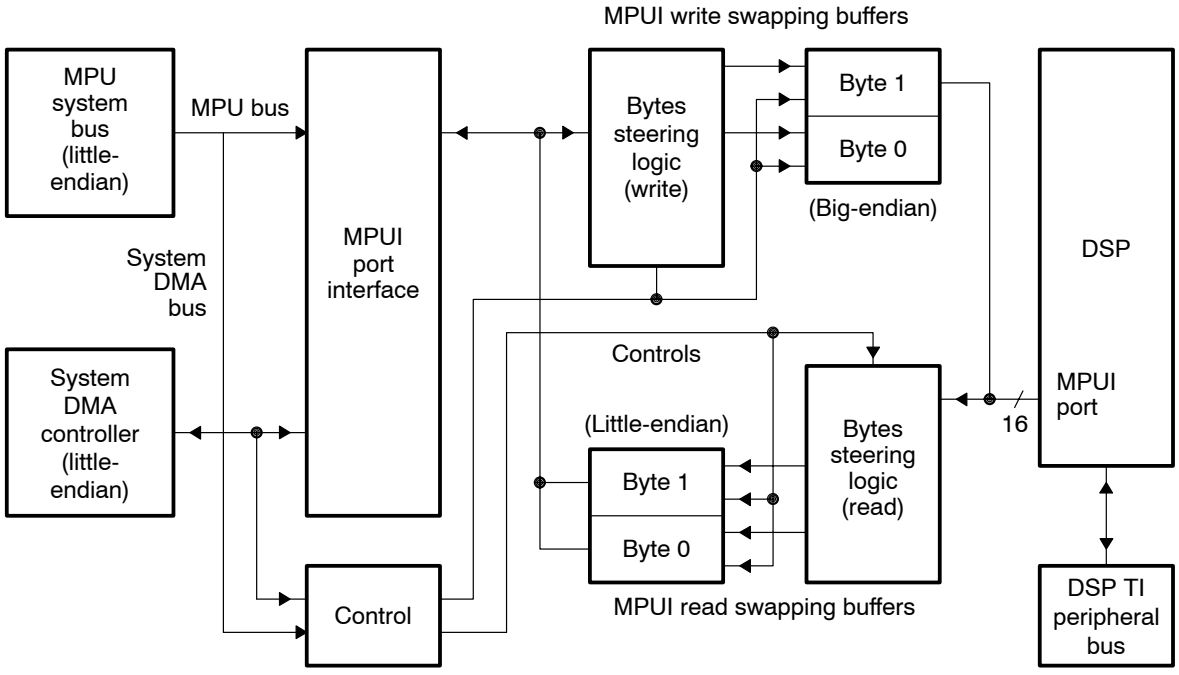
Note: The steering logic puts the byte/word/double-word in appropriate formats.

20.2 Conversion Through the MPUI

Swapping buffers are implemented at the boundary between the DSP and the MPUI (See Figure 29).

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 29. 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.

21 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.

21.1 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)

21.2 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)

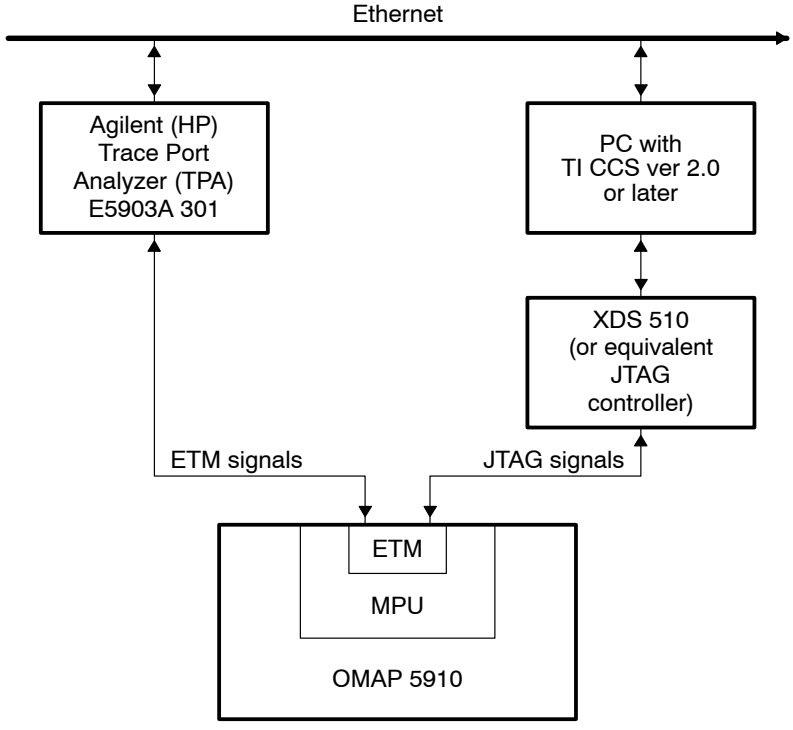
    When this rate is selected, the trace port analyzer (TPA) samples the trace data signals on both the rising and the falling edges of the TRACECLK. Bit 13 of the ETM control register enables selection of this rate.

The ETM trace signals are multiplexed with the camera interface pins on the OMAP5910 device. The default value upon reset is the camera interface.

21.3 Operation

Figure 30 shows how the OMAP5910 ETM is used in a system setup for capture of trace data.

Figure 30. 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.

21.4 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*.

Index

A

B

C

D

E

F

I

L

M

S

T

V

W