Intel® 413808 and 413812 I/O Controllers in TPER Mode

Developer's Manual

October 2007

Order Number: 317805-001US

Contents

Figures

Tables

Revision History

Date	Revision	Description
October 2007	001	Initial Release.

1.0 Introduction

This document covers the Intel® 413808 and 413812 I/O Controllers (4138xx). Note that the 4138xx operates in multiple modes, depending on which mode, determines when this manual is applicable. (In part or whole)

With 4138xx in I/O Controller Mode, the 4138xx is a stand-alone SAS/SATA I/O Controller, the host driver interface to the 4138xx is via SLI protocol through the TPMI[1] unit. When in Operational Mode as an I/O Controller, there is no direct access to other internal units such as PBI, GPIOs, UARTs etc., all access must be done via SLI (note that some units have no SLI access API). In order to facilitate the programming of the Flash device, the PBI may be accessed via the TPMI[1] when the transport firmware is not running (cores in reset). This document is generally not applicable when in IOC mode except for those situations where access to internal registers is required. (Flash programming)

With 4138xx in TPER Mode, Third Party Embedded RAID (TPER) mode. The term "TPER"can refer to a usage model, a silicon SKU, or a version of the transport firmware. Using the term to describe a Usage Model simply means running intelligent RAID (non host-based, typically simple levels such as 0/1/10 due to limited resources) without burdening the I/O processor with external DDR2 costs. From the silicon view it is essentially a DDR2-less 81348 (4138xx/4138xx 'A' version); a SKU where the memory controller cannot be used. Because TPER mode is very similar to the 81348 from a programming model perspective, the 81348 is sometimes referenced in collateral discussing the 4138xx in TPER mode.

This manual is primarily intended for those using the 4138xx in TPER mode and much of the content is identical to that of the 81348.

See Table 11 for how the silicon and transport firmware are combined. See Figure 1, "TPER Architecture Overview" on page 37 for an Architectural Overview.

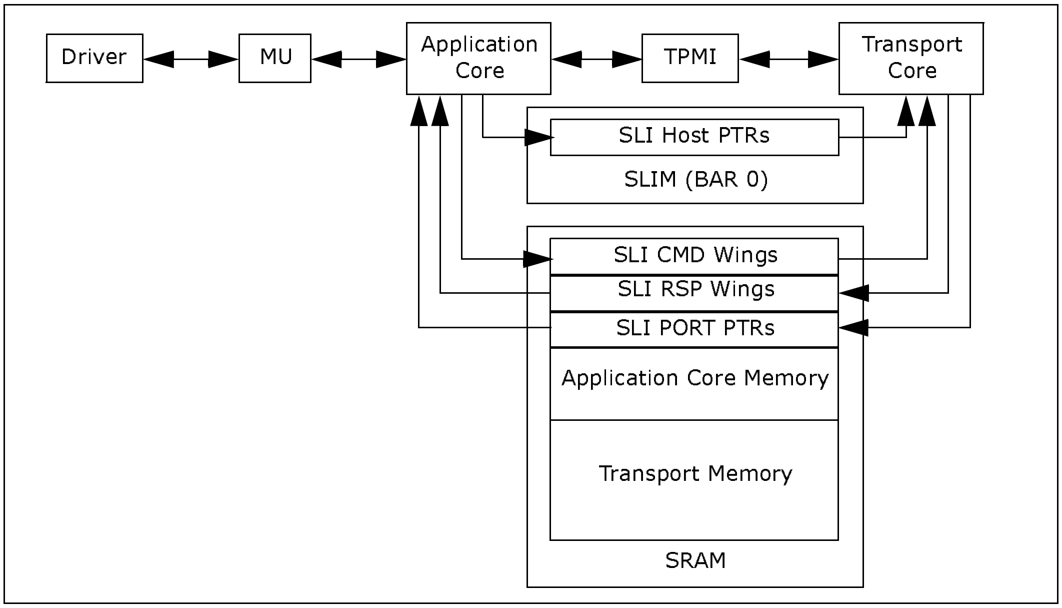
Table 1. Intel® 413808 and 413812 I/O Controllers in TPER Mode/Firmware Mapping

Silicon	Transport Firmware	
	Standard Transport FW	TPER Transport FW
81348	Functional as a full featured I/O Processor	Functional as a limited capability I/O Processor and can be upgraded to a full featured I/O Processor via firmware upgrade (aka In Place Upgrade).
4138xx in IOC mode	Functional as a full featured I/O Controller	Not Supported
4138xx in TPER mode	Not Supported	Functional as a limited capability I/O Processor

1. The TPMI specification is available through your Intel representative.

The overall high-level architecture is shown in Figure 1.

Figure 1. TPER Architecture Overview



When the 4138xx is in TPER mode, the interface to the host driver is under the control of the Application Core, and as with the 81348, the MU provides the hardware for the messaging interface. Note however, that since there is no DDR2 the MU is unable to make use of the Index Registers or Circular Queues, since these rely upon DDR2. However, the rest of the MU functionality is available.

The interface between the Application Core and the Transport Core continues to be SLI with the support of the TPMI registers. The Application Core is assigned a total of 256KB of SRAM for its use. The SLI IOCB Command and Response Rings and the SLI Port Pointers must reside in SRAM instead of DDR2. This comes out of the 256KB region assigned to the Application Core as specified by the PCB structure as communicated by the SLI Configuration Port (CONFIG_SLI_PORT) command. Complete details on the Application Core section of SRAM, including alignment requirements, addresses, etc., can be found in the *SCDL Architecture Specification, Firmware Release Notes, Sample Code* which is included in the Software Developer's Kit package.

The following is a list of other features, both silicon and firmware that are not available with the TPER usage model.

- 4138xx in TPER mode (when using 81348 silicon these features are available):
  - ADMA
  - MU circular queues, index registers.
- TPER Transport Firmware (differences from 81348 1.0 firmware features):
  - CONFIG_SAS_GPIO is not available when running TPER firmware.
  - Fewer addressable targets and outstanding I/Os supported, refer to the firmware release notes for the version in question for specifics.
  - Ring memory and port pointers allocated from application core SRAM region.
  - Any application core host messaging interface memory required allocated from application core SRAM.

1.1 Design-in Considerations

- For In Place Upgrade, the 81348 SKU must be used. The 4138xx SKU cannot be upgraded to full featured RAID. In place upgrade simply means using an 81348 SKU in IOC mode and then at some later point in time, updating the firmware and reset straps to put the 81348 into IOP mode. From the end user perspective it will appear as an in place upgrade from an I/O controller for a fully featured Intelligent RAID Subsystem.
- The following are reset straps associated with putting the 4138xx in TPER mode (additionally, of course, TPER firmware must be used for the 4138xx to operate in TPER mode). (see datasheet for complete listing of reset straps and their descriptions)
    - 4138xx mode:
      - CONTROLLER_ONLY#: 0
      - DF_SEL[2:0]: 000 (no other combinations are currently valid)
      - HOLD_X0_IN_RST#: 1
      - HOLD_X1_IN_RST#: 0
      - CFG_CYCLE_EN#: 1
    - 4138xx in TPER mode:
      - CONTROLLER_ONLY#: 1
      - DF_SEL[2:0]: 000 (no other combinations are currently valid)
      - HOLD_X0_IN_RST#: 1
      - HOLD_X1_IN_RST#: 1
      - CFG_CYCLE_EN#: 1
- The design must comprehend the additional hardware requirements needed by the full featured RAID solution. The specific hardware requirements, and the mechanism to add them, are dependent on the RAID ISV but include items such as:
    - DDR2 (connector and/or memory) and supporting components.
    - PBI modules required by the full featured RAID such as journaling NVSRAM, buzzers, additional Flash, etc.
    - Hardware Key for software/firmware feature enabling.

1.1.1 Software

- **PCI Configuration Space**: For 4138xx in TPER mode (as with 81348), the PCI configuration space presented is that of the Address Translation Unit (ATU) and it is the responsibility of the Application Core firmware to setup things such as the device ID per their design. When designing a system for In-place Upgrade, the RAID ISV must determine if/how they will have host driver compatibility between their stack running on 81348 with TPER firmware and their stack running on IOP348 with standard firmware.
- **RAID MetaData Format:** The RAID ISV is responsible for meeting any requirements with regards to Metadata format compatibility between their stack running on 81348 with TPER firmware and their stack running on 81348 with standard firmware.
- **Host interface**: Defined by the ISV, just as in 81348. When using TPER Silicon, MU Circular queues and index registers are not available. SLI interface is only relevant/available between the Application Core and Transport Core. With the same interface as 81348, with the exception that the IOCB command and response rings must be located in the Application Core SRAM space as opposed to DDR2 memory.

  For full details on SLI interface requirements for TPER, please refer to the SCDL Architecture Specification listed in Table 2, "Documentation References" on page 40.
- **SDMA (SRAM DMA) Engine:** Since there is no ADMA without DDR2, a separate DMA engine, the SRAM DMA, is provided to allow for the Application Core the ability to DMA command/status as part of its host messaging interface. The engine is different from ADMA, no chaining or special features. All access is direct register (no coprocessor access) as opposed to DDR2 memory descriptor based processing for ADMA. The SDMA chapter is found in this manual.
- **SRAM ECC:** The SRAM is a shared resource between the Application Core and the Transport Core in TPER. The Ttransport Core handles the initial enabling of ECC and scrubbing of SRAM for the entire region before the common boot code allows transfers execution to the Application Core. However, the Transport Core ignores ECC errors in the Application Core SRAM region and expects that the Application Core handles and clears ECC interrupts per specification. Similarly, the Transport Core handles all ECC interrupts for the Transport Core region of SRAM and asserts (hanger/dump) in the event that a multi-bit error occurs in its region. The SRAM chapter is found in this manual.
  - Important Notes:

    - The Application Core must ignore all ECC interrupts until after it has successfully completed CONFIG_SLI_PORT since up until that time, theTransport Firmware monitors ECC errors for the entire SRAM region.

    - In the event that the Application Ccore uses a PRG other than the TPER firmware (diagnostic overlay for internal test purposes when available), the Application Ccore must ignore the ECC interrupt for the entire SRAM region.

1.2 Documentation References

For available documentation references please refer to the following URLs:

http://developer.intel.com/design/storage/controller/docs/ioc340.htm
http://developer.intel.com/design/iio/docs/iop348.htm

Table 2 is a list of the available documentation for the 4138xx that is referenced for a TPER design.

Table 2. Documentation References

Document	Reference #
Intel® 413808 and 413812 I/O Controllers in TPER Mode Developer's Manual	317805[a]
Intel® 413808 and 413812 I/O Controllers Developer's Manual	315036[a]
Intel® 413808 and 413812 I/O Controllers in TPER Mode Datasheet	317806[a]
Intel® 413808 and 413812 I/O Controllers Datasheet	315040[a]
Intel® 413808 and 413812 I/O Controllers in TPER Mode Design Guide	317807[a]
Intel® 413808 and 413812 I/O Controllers Design Guide	315055[a]
Intel® 413808 and 413812 I/O Controllers Design Review Checklist	315046[a]
Intel® 413808 and 413812 I/O Controllers Thermal Design Considerations Application Note	315052[a]
Intel® 413808 and 413812 I/O Controllers in TPER Mode Specification Update	317808[a]
Intel® 413808 and 413812 I/O Controllers Specification Update	315043[a]
Intel® 81348 I/O Processor - Intel® 413808 and 413812 SAS/SATA I/O Controllers SSAS	356369[b] (643939[c])
SAS/SATA Command Summary (SSCS)	351156 (645843)
SCDL Architecture Specification, Firmware Release Notes, Sample Code	Delivered with firmware
TPMI Specification	TBD

a. Web document number (http://developer.intel.com/design/storage/controller/docs/ioc340.htm).
b. My SMG document number.
c. FDBL document number.

1.3 About This Document

This document is the authoritative and definitive reference for the external architecture of the Intel® 413808 and 413812 I/O Controllers in TPER Mode (4138xx), with Intel XScale® microarchitecture[2].

Intel Corporation assumes no responsibility for any errors which may appear in this document nor does it make a commitment to update the information contained herein.

Intel retains the right to make changes to these specifications at any time, without notice. In particular, descriptions of features, timings, packaging, and pin-outs does not imply a commitment to implement them. In fact, this specification does not imply a commitment by Intel to design, manufacture, or sell the product described herein.

1.3.1 How To Read This Document

This document describes the product-specific features of the 4138xx. Each chapter describes a different feature and starts with an overview followed by the theory of operation.

The reader should have a working understanding of the Peripheral Component Interconnect (PCI) Local Bus Specification, the PCI-X Addendum to the PCI Local Bus Specification and the PCI Express Specification. For more information, refer to the *PCI Local Bus Specification*, Revision 2.3, the *PCI-X Addendum to the PCI Local Bus Specification*, Revision 2.0a, and the *PCI Express Specification*, Revision 1.0a.

1.3.2 Other Relevant Documents

1. Intel® 80200 Processor based on Intel® XScale™ Microarchitecture Developer's Manual (Order Number: 273411), Intel Corporation
2. *PCI Local Bus Specification*, Revision 2.3 - PCI Special Interest Group
3. *PCI Bus Power Management Interface Specification*, Revision 1.1 - PCI Special Interest Group
4. *PCI Express Specification*, Revision 1.0a - PCI Special Interest Group

2. ARM architecture compliant.

1.4 About the Intel® 413808 and 413812 I/O Controllers in TPER Mode

The 4138xx is a single-function PCI devices that integrates two Intel XScale® processors with intelligent peripherals including a PCI bus application bridge and eight Serial-Attached SCSI (SAS) Engines. The SAS Engines on the 4138xx also support direct-attached Serial ATA (SATA) targets. The 4138xx also supports two internal busses: north internal bus and south internal bus. With the two internal busses, transactions can take place simultaneously on each bus. The north internal bus generates large burst transactions are located on the south internal bus, thus allowing the two Intel XScale® processors exclusive access to the north internal bus.

The 4138xx consolidates, into a single system:

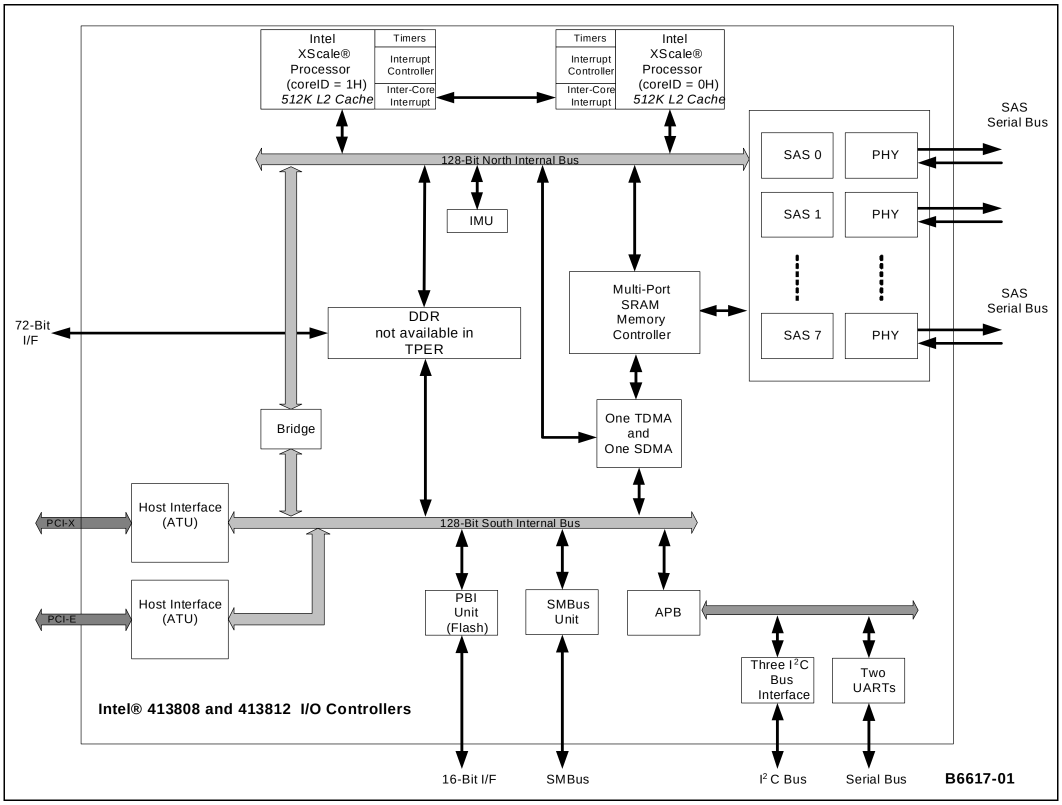
- Two Intel XScale® processors
- **Eight** Serial-Attached SCSI Links - also capable of supporting direct-attached SATA targets
- PCI - Local Memory Bus Address Translation Unit - PCI function 0
- Messaging Unit
- Inter-Processor Communication
- Inter-Processor Messaging Unit
- Third Party Messaging Interface (TPMI)
- Peripheral Bus Interface Unit (PBI)
- Integrated SRAM Memory Controller
- Performance Monitor (**PMON**)
- Two Programmable Timers per Intel XScale® processor
- Watchdog Timer per Intel XScale® processor
- **Three** I^2C Bus Interface Units
- **Two** Serial Port Units
- **Sixteen** General Purpose Input Output (GPIO) ports
- Two SGPIO busses
- Internal North Bus-South Bus Bridge

It is an integrated processor that addresses the needs of intelligent I/O Storage applications and helps reduce intelligent I/O system costs.

Both the address and data busses on the 4138xx south internal bus are byte-wise parity protected.

Figure 2 is a block diagram of the 4138xx.

Figure 2. Intel® 413808 and 413812 I/O Controllers in TPER Mode Functional Block Diagram



1.5 Intel® 413808 and 413812 I/O Controllers in TPER Mode Features

The 81348 combines two Intel XScale® processors with powerful new features to create an intelligent I/O storage processor. This single- or multi-function PCI device is fully compliant with the *PCI-X 2.0a and PCI Express 1.0a specifications*. The 81348-specific features include:

- Address Translation Unit
- Messaging Unit
- Peripheral Bus Interface Unit (PBI)
- **Three** I^{2}C Bus Interface Units
- Two SGPIO Busses
- Internal Bus Bridge
- Performance Monitoring Unit
- **Two** Serial Port Units (UARTs)
- Inter-Processor Communication
- Two Programmable Timers per core
- Watchdog Timers
- Third Party Messaging Interface (TMPI)

The 81348 microarchitecture is based upon two Intel XScale® processors. When in TPER mode, one processor is available for general application purposes (RAID). When in IOC mode, there are no processors available. The microarchitecture operates at a maximum frequency of 1.5 GHz. The instruction cache is 32 Kbytes in size and is 32-way set associative. Also, the microarchitecture includes a data cache that is 32 Kbytes and is 32-way set associative and a mini data cache that is 2 Kbytes and is 2-way set associative. Both Intel XScale® processors support unified 512-KByte Level 2 (L2) cache and is 8-way set associative.

The 81348 includes sixteen General Purpose I/O (GPIO) pins, which are used for SAS Links for activity and status indicators. Each SAS link uses two LSO pins. The 81348 also supports two SGPIO busses.

The subsections that follow briefly overview each feature. Refer to the appropriate chapter for full technical descriptions.

1.5.1 Host Interface

The 4138xx present a single function to the host When in IOC mode, the TPMI is exposed to the host and PCI configuration parameters are setup by the Transport Firmware. Some parameters are changed via use of the OEM Parameter Tool. When in TPER mode, the TPMI is not exposed to the host, the ATU is. Thus, Application Core firmware is fully responsible for setting up PCI configuration space.

1.5.2 Intel XScale® Processor

The Intel XScale® processor operates at a maximum frequency of 1.5 GHz. The instruction cache is 32 Kbytes in size and is 32-way set associative. Also, the processor includes a data cache that is 32 Kbytes and is 32-way set associative. The Intel XScale® processor supports a unified 512-KByte Level 2 (L2) cache and is 8-way set associative.

1.5.3 Internal Busses

The 4138xx is architected around two internal busses: north internal bus and south internal bus. The two busses use the same bus protocol. The north internal bus is 128-bit wide and operates at speed up to 400 MHz.

The south internal bus is 128-bits wide and operates at speeds up to 400 MHz. The south internal bus provides data paths for large DMA generated burst transactions.

Both the internal address and data busses on the south internal bus are parity protected on a byte-wise basis.

1.5.4 Application DMA Controller

ADMA is not available in the 4138xx in any mode.

1.5.5 Address Translation Unit

The Address Translation Unit (ATU) allows PCI transactions direct access to the local memory. The ATU provides the interface for the RAID Controller PCI function. The ATU supports transactions between PCI address space and the internal address space. Address translation is controlled through programmable registers accessible from both the PCI interface and the Intel XScale® processor. Dual access to registers allows flexibility in mapping the two address spaces. The ATU also supports the extended capability configuration headers.

1.5.6 Messaging Unit

The Messaging Unit (MU) provides data transfer between the PCI system and the 4138xx. It uses interrupts to notify each system when new data arrives. The MU has the following messaging mechanisms:

- Message Registers
- Doorbell Registers

Each allows a host processor or external PCI device and the 4138xx to communicate through message passing and interrupt generation. The MU in conjunction with the ATU in TPER mode.

1.5.7 DDR Memory Controller

DDR is not available on the 4138xx.

1.5.8 Peripheral Bus Interface

The Peripheral Bus Interface Unit is a data communication path to the Flash memory components or other peripherals of 4138xx hardware system. Note, that Flash parts must be compatible with the transport firmware. See the System/Software Architecture Specfication and Design Guide Checklist for more information on supported Flash parts. The PBI includes support for either 8/16 bit devices. To perform these tasks at high bandwidth, the bus features a burst transfer capability which allows successive 8/16-bit data transfers.

1.5.9 Performance Monitoring Unit

The Performance Monitoring Unit allows various events on the 4138xx to be monitored.

1.5.10 I^2C Bus Interface Unit

There are three I^2C (Inter-Integrated Circuit) Bus Interface Units that allow the Intel XScale® processor to serve as a master and slave device residing on the I^2C bus. The I^2C unit uses a serial bus developed by Philips Semiconductor consisting of a two-pin interface. The bus allows 4138xx to interface to other I^2C peripherals and microcontrollers for system management functions. It requires a minimum of hardware for an economical system to relay status and reliability information on the I/O subsystem to an external device. Also refer to *I^2C Peripherals for Microcontrollers* (Philips Semiconductor).

1.5.11 UART Unit

The 4138xx includes two UART units. The UART Unit allows the two Intel XScale® processors to serve as a master and slave device residing on the UART bus. The UART unit uses a serial bus consisting of a two-pin interface. The bus allows 4138xx to interface to other peripherals and microcontrollers. Also refer to *16550 Device spec* (National Semiconductor).

1.5.12 Interrupt Controller Unit

Each Intel XScale® processor supports an Interrupt Controller Unit. The Interrupt Controller Unit (ICU) aggregates interrupt sources both external and internal of sources of 4138xx to the Intel XScale® processor. The ICU supports high performance interrupt processing with direct interrupt service routine vector generation on a per source basis. Each source has programmability for masking, core processor interrupt input, and priority.

1.5.13 Internal Bus System Controller

Each internal bus (north and south) employs a internal System Controller. The internal System Controller observes all the address or data bus request from requestors and completors connected to the internal bus. The internal System Controller includes features to handle: internal address bus arbitration, internal data bus arbitration, framing Address bus cycles, framing Data bus cycles, and provides the shared address and shared data paths from/to units.

1.5.14 Inter-Processor Communication

All intern processor communications on the 4138xx are over the internal bus.

1.5.15 Inter-Processor Messaging Unit

The IPMU is not available on the 4138xx.

1.5.16 Timers

The 4138xx supports two programmable 32-bit timers per processor. The 4138xx also supports one watchdog timer per processor.

1.5.17 GPIO

The 4138xx includes sixteen General Purpose I/O (GPIO) pins.

1.5.18 FSENG

The FSENG block contains the Serial-Attached SCSI(SAS) and SATA engines. The 81348 contains up to **eight** engines. And each engine is composed of the transport, link, PHY and physical layers. This unit is for use by the Transport Firmware only.

1.6 Terminology and Conventions

1.6.1 Representing Numbers

All numbers in this document can be assumed to be Base10 unless designated otherwise. In text, numbers in Base16 are represented as “nnnH”, where the “H” signifies hexadecimal. In pseudo code descriptions, hexadecimal numbers are represented in the form 0x1234ABCD. Binary numbers are not explicitly identified but are assumed when bit operations or bit ranges are used.

1.6.2 Fields

A *reserved* field is a field that may be used by an implementation. When the initial value of a reserved field is supplied by software, this value must be zero. Software should not modify reserved fields or depend on any values in reserved fields.

A *read/write* field can written to a new value following initialization. This field can always be read to return the current value.

A *read only* field can be read to return the current value. Writes to *read only* fields are treated as no-op operations and does not change the current value nor result in an error condition.

A *read/clear* field can also be read to return the current value. A write to a *read/clear* field with the data value of 0 causes no change to the field. A write to a *read/clear* field with a data value of 1 causes the field to be cleared (reset to the value of 0). For example, when a *read/clear* field has a value of F0H, and a data value of 55H is written, the resultant field is A0H.

A *read/set* field can also be read to return the current value. A write to a *read/set* field with the data value of 0 causes no change to the field. A write to a *read/set* field with a data value of 1 causes the field to be set (set to the value of 1). For example, when a *read/set* field has a value of F0H, and a data value of 55H is written, the resultant field ia F5H.

A *writeonce/readonly* field can be written to a new value **once** following initialization. After the this write has occurred, the *writeonce/readonly* field treats all subsequent writes as no-op operations and does not change the current value or result in an error condition. The field can always be read to return the current value.

1.6.3 Specifying Bit and Signal Values

The terms *set* and *clear* in this specification refer to bit values in register and data structures. When a bit is set, its value is 1; when the bit is clear, its value is 0. Likewise, *setting* a bit means giving it a value of 1 and *clearing* a bit means giving it a value of 0.

The terms *assert* and *deassert* refer to the logically active or inactive value of a signal or bit, respectively.

1.6.4 Signal Name Conventions

All signal names use the PCI signal name convention of using the "#" symbol at the end of a signal name to indicate that the signal's active state occurs when it is at a low voltage. The absence of the "#" symbol indicates that the signal's active state occurs when it is at a high voltage.

1.6.5 Terminology

To aid the discussion of the 81348 architecture, the following terminology is used:

Downstream	At or toward a PCI bus with a higher number (after configuration)
DWORD	32-bit data word
QWORD	64-bit data word
word	32-bit data word
Host processor	Processor located upstream from the 81348
Local processor	Intel XScale® processor within the 81348
Local bus	81348 Internal Bus
Local memory	Memory subsystem on the Intel XScale® processor DDR SDRAM or Peripheral Bus Interface busses
Upstream	At or toward a PCI bus with a lower number (after configuration)

2.0 Address Translation Unit (PCI-X)

This chapter describes the operation modes, setup, and implementation of the module which interfaces between the PCI bus and the Intel® 413808 and 413812 I/O Controllers in TPER Mode (4138xx) internal bus.

2.1 Overview

As indicated in Figure 3, the Address Translation Unit (ATU) — the interface between the PCI bus and the on-chip internal bus — consists of the Address Translation Unit (ATU) and the Expansion ROM Unit.

The ATU supports both inbound and outbound address translation. The ATU provides access between the PCI bus and the 4138xx internal bus.

Transactions initiated on the PCI bus and targeted at the 4138xx internal bus are referred to as *inbound transactions* (PCI to internal bus). Transactions initiated on the 4138xx internal bus and targeted at the PCI bus are referred to as *outbound transactions* (internal bus to PCI). The ATU accepts multiple inbound or outbound transactions and processes them simultaneously.

During inbound transactions, the ATU converts PCI addresses (initiated by a PCI bus master) to Internal Bus Addresses and initiates the data transfer on the 4138xx internal bus. During outbound transactions, the ATU converts internal bus addresses to PCI addresses and initiates the data transfer on the PCI bus.

The Expansion ROM provides the PCI mechanism for downloading device/board driver code during system boot sequence. It consists of a separate inbound address range which accesses a Flash EPROM device connected through the 4138xx memory controller. Refer to the *PCI Local Bus Specification*, Revision 2.3 for details of Expansion ROM usage.

The Address Translation Unit and the Expansion ROM Translation Unit represent a single function of the multi-function 4138xx PCI device.

The ATU supports the following PCI operating modes and bus widths delivering up to 2133 Mbytes/sec of bandwidth:

— Conventional Modes: PCI 33, PCI 66

— PCI-X Modes: Mode 1 (PCI-X 66, PCI-X 133), Mode 2 (PCI-X 266)[3]

— Bus Widths: 64-bit, 32-bit

In Mode 2, all transaction phases are ECC protected (when enabled). In the remaining PCI-X and Conventional modes supported by 4138xx, all transaction phases are parity protected (when enabled).

On the internal interface, the ATU implements the 4138xx internal bus protocol which provides for a maximum of 4800 Mbytes/sec of bandwidth.

3. PCIX is not supported in TPER mode.

The 4138xx meets the standard requirements to be considered "Hot-Swap Silicon" detailed in the *Compact PCI Hot-Swap Specification*, Revision 2.1.

Address and data are protected by byte-wise parity on the internal bus.

The ATU includes four extended capability headers that implement Power Management capability as defined by the *PCI Bus Power Management Interface Specification*, Revision 1.1, MSI capability as defined by *PCI Local Bus Specification*, Revision 2.3, Hot-Swap capability as defined by the *Compact PCI Hot-Swap Specification*, Revision 2.1, and PCI-X capability as defined by *PCI-X Protocol Addendum to the PCI Local Bus Specification,* Revision 2.0.

The functionality of the ATU is described in the following sections. The ATU has a memory-mapped register interface that is visible from either the PCI interface, the internal bus interface, or both.

Figure 3. ATU Block Diagram

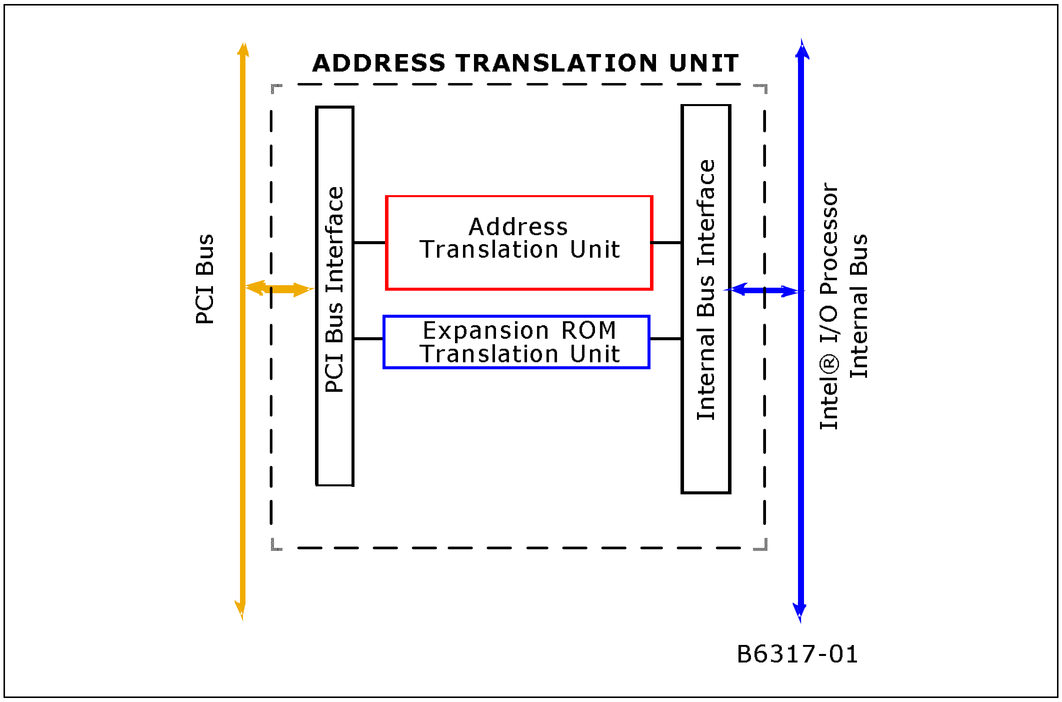
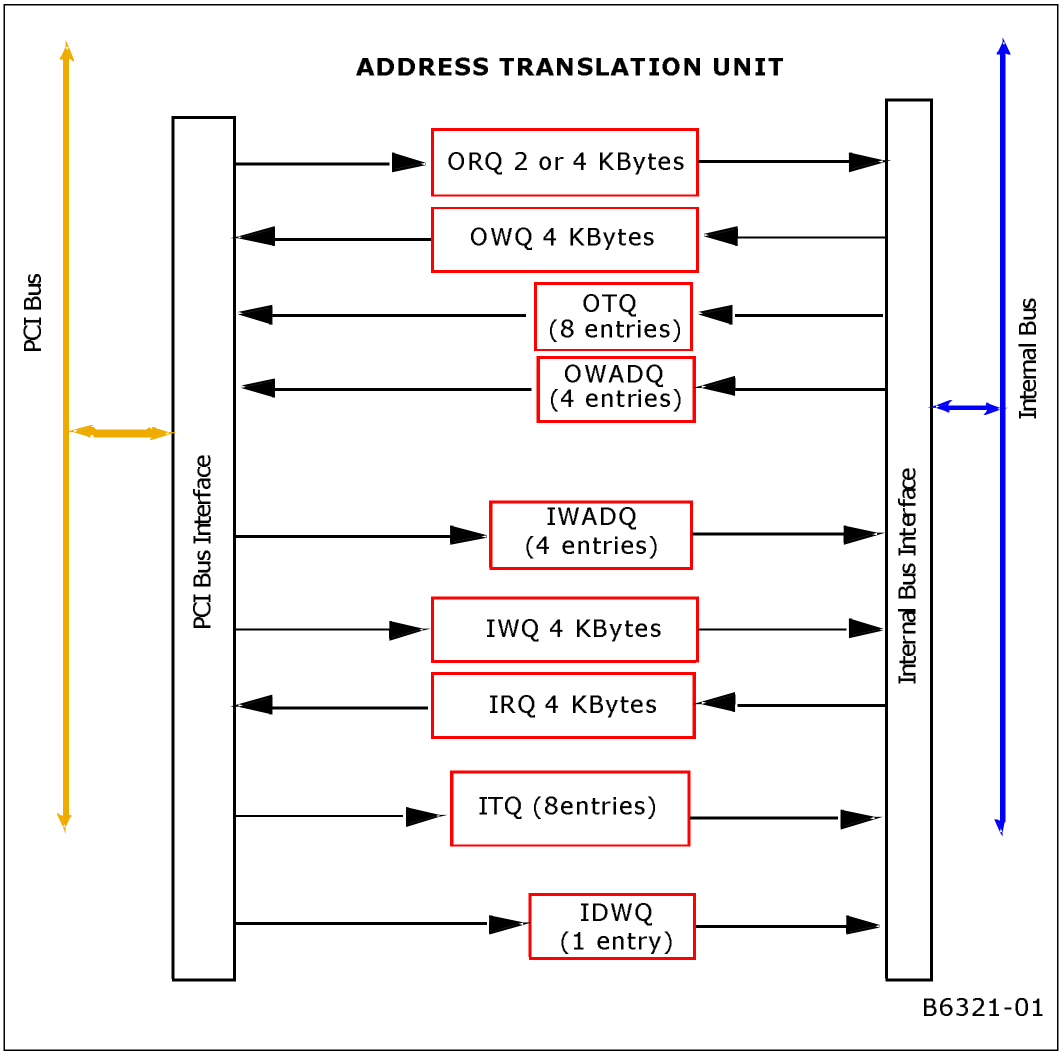


Figure 4. ATU Queue Architecture Block Diagram



2.2 ATU Address Translation

The ATU allows PCI masters on the PCI bus to initiate transactions to the 4138xx internal bus and allows the Intel XScale® processor (ARM* architecture compliant) to initiate transactions to the PCI bus.

The ATU implements an address windowing scheme to determine which addresses to claim and translate to the destination bus.

- The address windowing mechanism for inbound translation is described in Section 2.2.1.1, "Inbound Address Translation" on page 56
- The address windowing mechanism for outbound translation is described in Section 2.2.2, "Outbound Transactions- Single Address Cycle (SAC) Internal Bus Transactions" on page 67 and Section 2.2.3, "Outbound Write Transaction" on page 72

The ATU has the ability to accept up to eight inbound PCI read transactions and four inbound PCI write transactions simultaneously. Also, the ATU has the ability to accept up to eight outbound internal bus read transactions and four outbound internal bus write transactions simultaneously. Refer to Figure 4 and Section 2.6 for details of the ATU queue architecture.

The ATU unit allows for recognition and generation of multiple PCI cycle types. Table 3 shows the PCI and PCI-X commands supported for both inbound and outbound ATU transactions. The type of operation seen by the ATU on inbound transactions is determined by the PCI master who initiates the transaction. Claiming an inbound transaction depends on the address range programmed within the inbound translation window. The type of transaction used by the ATU on outbound transactions generated by the core processor is determined by the internal bus address and the outbound windowing scheme.

ATU supports the 64-bit addressing specified by the *PCI Local Bus Specification*, Revision 2.3. This 64-bit addressing extension is supported for both inbound and outbound data transactions. This is in addition to the 64-bit data extensions supported by the 4138xx.

ATU does not support exclusive access using the PCI LOCK# signal. Also, the ATU does not insure atomicity for outbound transactions.

Note: In conventional PCI Mode, the ATUX will pre-fetch data for read transactions on the internal bus as defined in Section 2.6.1.2, "Inbound Read Queue Structure" on page 84. For internal bus targets that are not capable of supporting large byte-counts as indicated in Table 10, "Inbound Read Prefetch Data Sizes" on page 84, these internal bus targets must be accessed using a non-prefetchable PCI window. This requirement also includes access made to the north internal bus targets via the internal bus bridge. The internal bus bridge supports 32-byte data queues, and any access made via the internal bus bridge that is greater than 32 bytes will result in an internal bus target abort.

Table 3. ATU Command Support

PCI Command Encoding	PCI Command Type	PCI-X Command Type	Claimed on Inbound Transactions on PCI Bus	Generated by Outbound Transactions on PCI Bus	Valid Internal Bus Command
0000	Interrupt Acknowledge	Interrupt Acknowledge	No	No	Reserved
0001	Special Cycle	Special Cycle	No	No	Reserved
0010	I/O Read	I/O Read	No[1]	**Yes**	Reserved
0011	I/O Write	I/O Write	No[1]	**Yes**	Reserved
0100	Reserved	Reserved	No	No	Reserved
0101	Reserved	Device ID Message[2]	No	No	Reserved
0110	Memory Read	Memory Read DWORD	**Yes**	**Yes**	Read
0111	Memory Write	Memory Write	**Yes**	**Yes**	Write
1000	Reserved	Alias to Memory Read Block	**Yes**	No	Read
1001	Reserved	Alias to Memory Write Block	**Yes**	No	Write
1010	Configuration Read	Configuration Read	**Yes**	**Yes**	Read
1011	Configuration Write	Configuration Write	**Yes**	**Yes**	Write
1100	Memory Read Multiple	Split Completion	**Yes**	**Yes**	Split Completion
1101	Dual Address Cycle	Dual Address Cycle	**Yes**	**Yes**	Reserved
1110	Memory Read Line	Memory Read Block	**Yes**	**Yes**[3]	Read
1111	Memory Write and Invalidate	Memory Write Block	**Yes**	Yes	Write

Notes:
1. The ATUX function itself does not claim I/O Transactions.
2. PCI-X mode 2 only
3. PCI-X mode only

Inbound and outbound ATU transactions are best described by the data flows used on the PCI bus and the 4138xx internal bus during read and write operations. The following sections describe read and write operations for inbound ATU transactions (PCI to internal bus) and outbound transactions (internal bus to PCI).

2.2.1 Inbound Transactions

Inbound transactions which target the ATU are translated and executed on the 4138xx internal bus. As a PCI target, the ATU is capable of accepting all PCI memory read and write operations as either a 32-bit or a 64-bit PCI target. In the conventional PCI mode *Memory Write* and *Memory Write and Invalidate* operations are performed as posted operations and all memory read operations are performed as delayed reads. In the PCI-X mode *Memory Write, Memory Write Block,* and *Alias to Memory Write Block* operations are performed as posted operations and *Memory Read DWORD, Memory Read Block,* and *Alias to Memory Read Block* operations are executed as split transactions. The ATU is capable of accepting configuration read and write cycles. In the conventional PCI mode, *Configuration Writes* are performed as delayed memory write operations and *Configuration Reads* are performed as delayed read operations. In the PCI-X mode, both *Configuration Writes* and *Configuration Reads* are performed as split transactions.

Inbound memory write transactions have their addresses entered into the inbound write address queue (IWADQ) and data entered into the inbound write data queue (IWQ). The IWQ/IWADQ pair are capable of holding up to 4 write operations up to the size of the data queue. Inbound configuration writes use the inbound delayed write queue (IDWQ) for address and data. Refer to Section 2.6 for details of queue operation. Inbound read operations (memory and configuration) have their address entered into the inbound transaction queue (ITQ) and the data is returned to the PCI master in the inbound read queue (IRQ). The ITQ is capable of holding up to 8 delayed read requests (split read requests when operating in the PCI-X mode).

In the conventional PCI mode, for inbound transactions, the ATU is a slave on the PCI bus and is a requester on the internal bus. PCI slave operation is defined in the *PCI Local Bus Specification*, Revision 2.3. In the PCI-X mode, for inbound transactions, the ATU initially is a target on the PCI bus and becomes an initiator when performing split completion transactions, and is an initiator on the internal bus.

Operation of the internal bus is defined in Chapter 7.0, "System Controller (SC) and Internal Bus Bridge". Specific operation of the ATU as master on the internal bus is defined in Section 2.2.6.

2.2.1.1 Inbound Address Translation

The ATU allows external PCI bus initiators to directly access the internal bus. These PCI bus initiators can read or write 4138xx memory-mapped registers or 4138xx local memory space. The process of inbound address translation involves two steps:

1. Address Detection.
    a. Determine when the 32-bit PCI address (64-bit PCI address during DACs) is within the address windows defined for the inbound ATU.
    b. Claim the PCI transaction with medium DEVSEL# timing in the conventional PCI mode and with Decode A DEVSEL# timing in the PCI-X mode.
2. Address Translation.
    a. Translate the 32-bit PCI address (lower 32-bit PCI address during DACs) to a 36-bit 4138xx internal bus address.

The ATU uses the following registers in inbound address window 0 translation:

- Inbound ATU Base Address Register 0
- Inbound ATU Limit Register 0
- Inbound ATU Translate Value Register 0
- Inbound ATU Upper Translate Value Register 0

The ATU uses the following registers in inbound address window 1 translation:

- Inbound ATU Base Address Register 1
- Inbound ATU Limit Register 1
- Inbound ATU Translate Value Register 1
- Inbound ATU Upper Translate Value Register 1

The ATU uses the following registers in inbound address window 2 translation:

- Inbound ATU Base Address Register 2
- Inbound ATU Limit Register 2
- Inbound ATU Translate Value Register 2
- Inbound ATU Upper Translate Value Register 2

The ATU uses the following registers in inbound address window 3 translation:

- Inbound ATU Base Address Register 3
- Inbound ATU Limit Register 3
- Inbound ATU Translate Value Register 3
- Inbound ATU Upper Translate Value Register 3

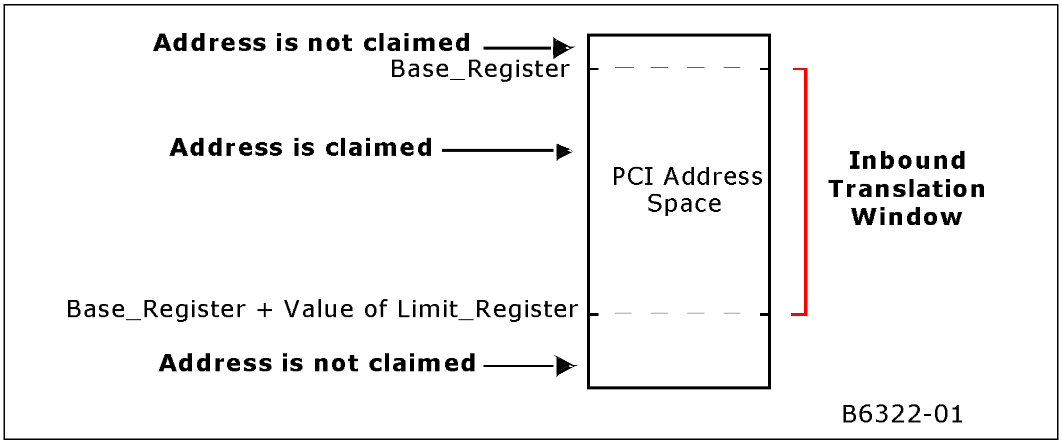
Inbound address detection is determined from the 32-bit PCI address, (64-bit PCI address during DACs) the base address register and the limit register. In the case of DACs none of the upper 32-bits of the address is masked during address comparison. The algorithm for detection is:

Equation 1. Inbound Address Detection

```
When PCI_Address [31:0] & Limit_Register[31:0] == Base_Register[31:0] and
PCI_Address [63:32] == Base_Register[63:32] (for DACs only) the PCI Address is claimed by the Inbound
ATU.
```

Figure 5 shows an example of inbound address detection.

Figure 5. Inbound Address Detection



The incoming 32-bit PCI address (lower 32-bits of the address in case of DACs) is bitwise ANDed with the associated inbound limit register. When the result matches the base register (and upper base address matches upper PCI address in case of DACs), the inbound PCI address is detected as being within the inbound translation window and is claimed by the ATU.

Note: By default, the first 8 Kbytes of the ATU inbound address translation window 0 are reserved for the Messaging Unit. See Chapter 4.0, "Messaging Unit".

Once the transaction is claimed, the address must be translated from a PCI address to a 36-bit internal bus address. In case of DACs upper 32-bits of the address is simply discarded and only the lower 32-bits are used during address translation. The algorithm is:

Equation 2. Inbound Translation

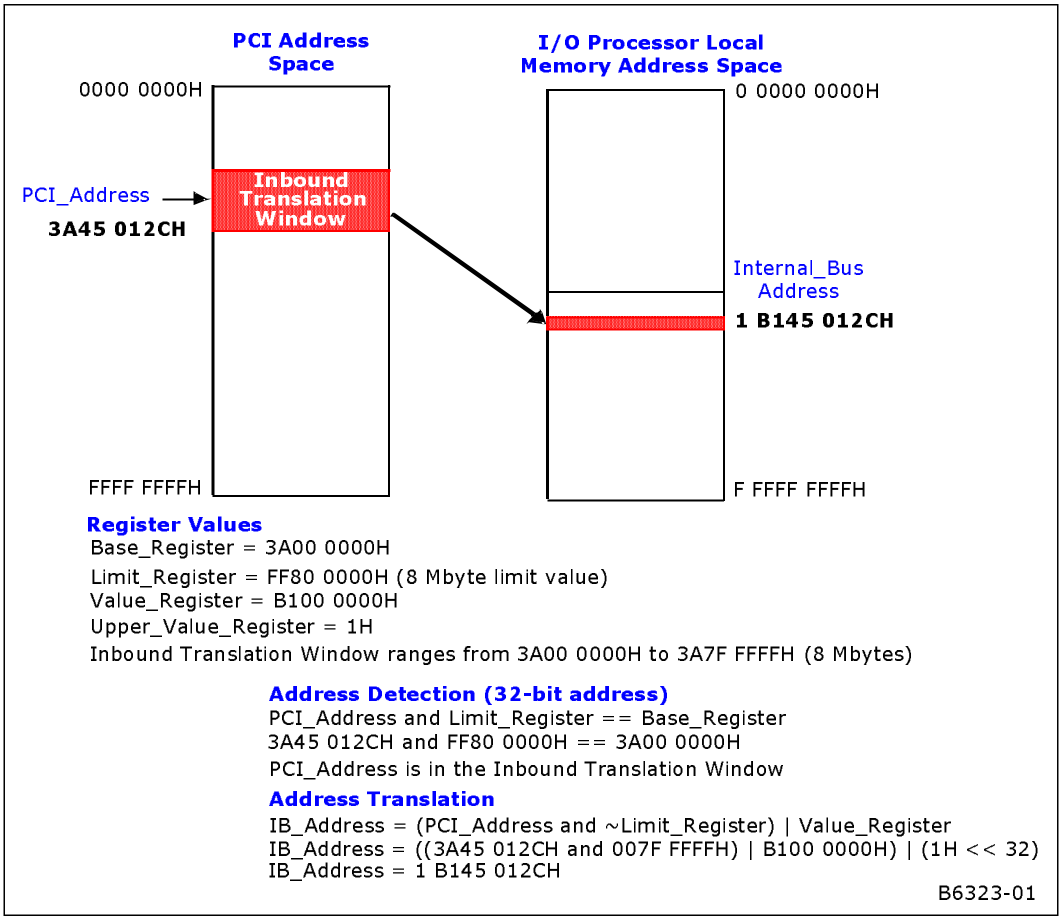
Intel® 413808 and 413812 I/O Controllers Internal Bus Address = ((PCI_Address[31:0] & ~Limit_Register[31:0]) | ATU_Translate_Value_Register[31:0]) | (ATU_Upper_Translate Value_Register[3:0] << 32).

The incoming 32-bit PCI address (lower 32-bits in case of DACs) is first bitwise ANDed with the bitwise inverse of the limit register. This result is bitwise ORed with the ATU Translate Value, which is then ORed with the 4-bit ATU Upper Translate Value left shifted by 32; the result is the 36-bit internal bus address. This translation mechanism is used for all inbound memory read and write commands excluding inbound configuration read and writes. Inbound configuration cycle translation is described in Section 2.2.1.4, "Inbound Configuration Cycle Translation" on page 64.

In the PCI mode for inbound memory transactions, the only burst order supported is Linear Incrementing. For any other burst order, the ATU signals a Disconnect after the first data phase. The PCI-X supports linear incrementing only, and hence the above situation is never encountered in the PCI-X mode.

Figure 6 shows an inbound translation example for 32-bit addressing. This example would hold true for an inbound transaction from PCI bus.

Figure 6. Inbound Translation Example



2.2.1.2 Inbound Write Transaction

An inbound write transaction is initiated by a PCI master and is targeted at either 4138xx local memory or a 4138xx memory-mapped register.

Data flow for an inbound write transaction on the PCI bus is summarized as:

- The ATU claims the PCI write transaction when the PCI address is within the inbound translation window defined by the ATU Inbound Base Address Register (and Inbound Upper Base Address Register during DACs) and Inbound Limit Register.
- When the IWADQ has at least one address entry available and the IWQ has at least one buffer available, the address is captured and the first data phase is accepted.
- The PCI interface continues to accept write data until one of the following is true:
  - The initiator performs a disconnect.
  - The transaction crosses a buffer boundary.
- When an uncorrectable address error is detected during the address phase of the transaction, the uncorrectable address error mechanisms are used. Refer to Section 2.7.1 for details of the uncorrectable address error response.
- When operating in the PCI-X mode when an uncorrectable attribute error is detected, the uncorrectable attribute error mechanism described in Section 2.7.1 is used.
- When an uncorrectable data error is detected while accepting data, the slave interface sets the appropriate bits based on PCI specification. No other action is taken. Refer to Section 2.7.3.6 for details of the inbound write uncorrectable data error response.

Once the PCI interface places a PCI address in the IWADQ, when IWQ has received data sufficient to cross a buffer boundary or the master disconnects on the PCI bus, the ATUs internal bus interface becomes aware of the inbound write. When there are additional write transactions ahead in the IWQ/IWADQ, the current transaction remains posted until ordering and priority have been satisfied (Refer to Section 2.6.3) and the transaction is attempted on the internal bus by the ATU internal master interface. The ATU does not insert target wait states nor do data merging on the PCI interface, when operating in the PCI mode.

In the PCI-X mode memory writes are always executed as immediate transactions, while configuration write transactions are processed as split transactions. The ATU generates a Split Completion Message, (with Message class = 0h - Write Completion Class and Message index = 00h - Write Completion Message) once a configuration write is successfully executed.

Also, when operating in the PCI-X mode a write sequence may contain multiple write transactions. The ATU handles such transactions as independent transactions.

Data flow for the inbound write transaction on the internal bus is summarized as:

- The ATU internal bus master requests the internal bus when IWADQ has at least one entry with associated data in the IWQ.
- When the internal bus is granted, the internal bus master interface initiates the write transaction by driving the translated address onto the internal bus. For details on inbound address translation, see Section 2.2, “ATU Address Translation” on page 53.
- When an internal bus target does not claim write transaction, a master abort condition is signaled on the internal bus. The current transaction is flushed from the queue and **SERR#** may be asserted on the PCI interface.
- The ATU initiator interface attempts a 128-bit wide transfer on the internal bus. When the target that claims the request does not support 128-bit wide transfers, a 64-bit wide transfer is used. Transfers of use internal bus byte enables to mask the bytes not written in each data phase. Write data is transferred from the IWQ to the internal bus when data is available and the internal bus interface retains internal bus ownership. Refer to Chapter 7.0, “System Controller (SC) and Internal Bus Bridge” for details of internal bus operation.
- The internal bus interface stops transferring data from the current transaction to the internal bus when one of the following conditions becomes true:
  — The data from the current transaction has completed (satisfaction of byte count). An initiator termination is performed and the bus returns to idle.
  — A Master Abort is signaled on the internal bus. **SERR#** may be asserted on the PCI bus. Data is flushed from the IWQ.

2.2.1.3 Inbound Read Transaction

An inbound read transaction is initiated by a PCI initiator and is targeted at either 4138xx local memory or a 4138xx memory-mapped register space. The read transaction is propagated through the inbound transaction queue (ITQ) and read data is returned through the inbound read queue (IRQ).

When operating in the conventional PCI mode, all inbound read transactions are processed as delayed read transactions. When operating in the PCI-X mode, all inbound read transactions are processed as split transactions. The ATUs PCI interface claims the read transaction and forwards the read request through to the internal bus and returns the read data to the PCI bus. Data flow for an inbound read transaction on the PCI bus is summarized in the following statements:

- The ATU claims the PCI read transaction when the PCI address is within the inbound translation window defined by ATU Inbound Base Address Register (and Inbound Upper Base Address Register during DACs) and Inbound Limit Register.
- When operating in the conventional PCI mode, when the ITQ is currently holding transaction information from a previous delayed read, the current transaction information is compared to the previous transaction information (based on the setting of the DRC Alias bit in Section 2.14.40, "ATU Configuration Register - ATUCR" on page 177). When there is a match and the data is in the IRQ, return the data to the master on the PCI bus. When there is a match and the data is not available, a Retry is signaled with no other action taken. When there is not a match and when the ITQ has less than eight entries, capture the transaction information, signal a Retry and initiate a delayed transaction. When there is not a match and when the ITQ is full, then signal a Retry with no other action taken.
  — When an uncorrectable address error is detected, the uncorrectable address response defined in Section 2.7 is used.
- When operating in the conventional PCI mode, once read data is driven onto the PCI bus from the IRQ, it continues until one of the following is true:
  — The initiator completes the PCI transaction. When there is data left unread in the IRQ, the data is flushed.
  — An internal bus Target Abort was detected. In this case, the Q-word associated with the Target Abort is never entered into the IRQ, and therefore is never returned.
  — Target Abort or a Disconnect with Data is returned in response to the Internal Bus Error.
  — The IRQ becomes empty. In this case, the PCI interface signals a Disconnect with data to the initiator on the last data word available.
- When operating in the PCI-X mode, when ITQ is not full, the PCI address, attribute and command are latched into the available ITQ and a Split Response Termination is signalled to the initiator.
- When operating in the PCI-X mode, when the transaction does not cross a 1024 byte aligned boundary, then the ATU waits until it receives the full byte count from the internal bus target before returning read data by generating the split completion transaction on the PCI-X bus. When the read requested crosses at least one 1024 byte boundary, then ATU completes the transfer by returning data in1024 byte aligned chunks.

- When operating in the PCI-X mode, once a split completion transaction has started, it continues until one of the following is true:
  - The requester (now the target) generates a Retry Termination, or a Disconnection at Next ADB (when the requester is a bridge)
  - The byte count is satisfied.
  - An internal bus Target Abort was detected. The ATU generates a Split Completion Message (message class=2h - completer error, and message index=81h - internal bus target abort) to inform the requester about the abnormal condition. The ITQ for this transaction is flushed. Refer to Section 2.7.1.
  - An internal bus Master Abort was detected. The ATU generates a Split Completion Message (message class=2h - completer error, and message index=80h - Master abort) to inform the requester about the abnormal condition. The ITQ for this transaction is flushed. Refer to Section 2.7.1
- When operating in the conventional PCI mode, when the master inserts wait states on the PCI bus, the ATU PCI slave interface waits with no premature disconnects.
- When an uncorrectable data error occurs signified by **PERR#** asserted from the initiator, no action is taken by the target interface. Refer to Section 2.7.3.5.
- When operating in the conventional PCI mode, when the read on the internal bus is target-aborted, either a target-abort or a disconnect with data is signaled to the initiator. This is based on the ATU ECC Target Abort Enable bit (bit 0 of the ATUIMR for ATU). When set, a target abort is used, when clear, a disconnect is used.
- When operating in the PCI-X mode, when the transaction on the internal bus resulted in a target abort, the ATU generates a Split Completion Message (message class=2h - completer error, and message index=81h - internal bus target abort) to inform the requester about the abnormal condition. The ITQ for this transaction is flushed. Refer to Section 2.7.1.
- When operating in the conventional PCI mode, when the transaction on the internal bus resulted in a master abort, the ATU returns a target abort to inform the requester about the abnormal condition. The ITQ for this transaction is flushed. Refer to Section 2.7.1
- When operating in the PCI-X mode, when the transaction on the internal bus resulted in a master abort, the ATU generates a Split Completion Message (message class=2h - completer error, and message index=80h - internal bus master abort) to inform the requester about the abnormal condition. The ITQ for this transaction is flushed. Refer to Section 2.7.1.
- When operating in the PCI-X mode, when the Split Completion transaction completes with either Master-Abort or Target-Abort, the requester is indicating a failure condition that prevents it from accepting the completion it requested. In this case, since the Split Request addresses a location that has no read side effects, the ATU must discard the Split Completion and take no further action.

The data flow for an inbound read transaction on the internal bus is summarized in the following statements:

- The ATU internal bus master interface requests the internal bus when a PCI address appears in an ITQ and transaction ordering has been satisfied. When operating in the PCI-X mode the ATU does not use the information provided by the Relax Ordering Attribute bit. That is, ATU always uses conventional PCI ordering rules.
- Once the internal bus is granted, the internal bus master interface drives the translated address onto the bus. When a Retry is signaled, the request is repeated. When a master abort occurs, the transaction is considered complete and a target abort is loaded into the associated IRQ for return to the PCI initiator (transaction is flushed once the PCI master has been delivered the target abort).
- Once the translated address is on the bus and the transaction has been claimed, the internal bus target starts returning data using a split response. Read data is continuously received by the IRQ until one of the following is true:
  - The full byte count requested by the ATU read request is received. The internal bus completer’s initiator interface performs an initiator completion in this case.
  - A partial byte count requested by the ATU read request is received. The completer’s internal bus initiator interface performs an initiator completion in this case. Also, the completer reacquires the internal bus to deliver the remaining read data byte count to the ATU.
  - When operating in the conventional PCI mode, a Target Abort is received on the internal bus from the internal bus target. In this case, the transaction is aborted and the PCI side is informed.
  - When operating in the PCI-X mode, a Target Abort is received on the internal bus from the internal bus target. In this case, the transaction is aborted. The ATU generates a Split Completion Message (message class=2h - completer error, and message index=81h - internal bus target abort) on the PCI bus to inform the requester about the abnormal condition. The ITQ for this transaction is flushed.

To support *PCI Local Bus Specification*, Revision 2.0 devices, the ATU can be programmed to ignore the memory read command (Memory Read, Memory Read Line, and Memory Read Multiple) when trying to match the current inbound read transaction with data in a DRC queue which was read previously (DRC on target bus). When the Read Command Alias Bit in the ATUCR register is set, the ATU does not distinguish the read commands on transactions. For example, the ATU enqueues a DRR with a Memory Read Multiple command and performs the read on the internal bus. Some time later, a PCI master attempts a Memory Read with the same address as the previous Memory Read Multiple. When the Read Command Bit is set, the ATU would return the read data from the DRC queue and consider the Delayed Read transaction complete. When the Read Command bit in the ATUCR was clear, the ATU would not return data since the PCI read commands did not match, only the address.

2.2.1.4 Inbound Configuration Cycle Translation

The 4138xx ATU only accepts Type 0 configuration cycles with a function number of zero when bit[7] of the ATUHTR (see Section 2.14.11, "ATU Header Type Register - ATUHTR" on page 153) is cleared or function numbers of zero and one when bit[7] of the ATUHTR is set.

The ATU is configured through the PCI bus. When operating in conventional PCI mode, all inbound configuration cycles are processed as delayed transactions. When operating in PCI-X mode, all inbound configuration cycles are processed as split transactions. The translation mechanism for inbound configuration cycles is defined by the *PCI Local Bus Specification*, Revision 2.3.

The ATU configuration space is selected by a PCI configuration command and claims access (by asserting **P_DEVSEL#**) when the **P_IDSEL** pin is asserted, the PCI command indicates a configuration read or write, and address bits **P_AD[1:0]** are 00_2 all during the address phase. The ATU interface ignores any configuration command (**P_IDSEL** active) where **P_AD[1:0]** are not 00_2 (e.g. Type 1 commands). During the configuration access address phase, the PCI address is divided into a number of fields to determine the actual configuration register access. These fields, in combination with the byte enables during the data phase create the unique encoding necessary to access the individual registers of the configuration address space:

- **P_AD[7:2] -** Register Number. Selects one of 64 DWORD registers in the ATU PCI configuration address space.
- **P_C/BE[3:0]# -** Used in data phase. Selects which actual configuration register is used within the DWORD address. Creates byte addressability of the register space.
- **P_AD[10:8]** - Function Number. Used to select which function of a multi-function device is being accessed. The ATU is function 0 and therefore it only responds to 000_2 in this bit field and ignore all other bit combinations.
- **P_AD[27:24]** - Upper Register Number. In PCI-X Mode 2, Upper Register Number and Lower Register Number combine to select one of 1024 DWORD registers in the ATU PCI configuration address space.

Note: In PCI-X Mode 2, the ATU does not support any extended capabilities list items starting at offset 100H indicated by a Null Enhanced Capability Header at offset 100H (i.e., Enhanced Capability Header with a Capability ID of 0000H, a Capability Version of 0H, and a Next Capability offset of 000H).

ATU configuration address space starts at internal address 3100H. Therefore, **P_AD[7:2]** equal to 000000_2 equates to address 3100H and **P_AD[7:2]** equal to 000001_2 results in address 3104H and so on.

For inbound configuration reads, IRQ and ITQ are used in the same manner as inbound memory read operations. The internal bus cycle that results are a 32-bit transaction.

For inbound configuration writes, ATU adds a delayed write data queue (IDWQ), which holds data the same way as the IWQ. Transaction information from the configuration write operation on the PCI interface is captured into the IDWQ (when full, a Retry is signaled). Data from delayed write (split write in PCI-X mode) request cycle is latched into IDWQ and forwarded to the internal bus interface. Once transaction ordering and priority have been satisfied, the internal bus master interface requests the internal bus and delivers write data to the target as defined in Section 2.2.1.2.

Status of the internal bus transaction is returned to the PCI bus, PCI initiator. When operating in conventional PCI mode, the initiator retry cycle is accepted once the write has been completed on the internal bus and status has been captured for return to the PCI master. When operating in PCI-X mode, a Split Completion Message (message class=0h and message index= 00h - Normal Completion) is generated on the PCI bus, once the write has been completed on the internal bus and status has been latched for return to the PCI master. Since Master Aborts and Target Aborts cannot occur during internal configuration cycles, normal completion status is returned. Data from PCI completion transaction is discarded.

2.2.1.5 Discard Timers

The ATU implements discard timers for inbound delayed transactions. These timers prevent deadlocks when the initiator of a retried delayed transaction fails to complete the transaction within 2^{10} or 2^{15} PCI clock cycles on the initiating bus when operating in the conventional PCI mode. The timer starts counting when the delayed request becomes a delayed completion by completing on the internal bus and all passing rules are satisfied. When the originating master on the PCI bus has not retried the transaction before the timer expires, the completion transaction is discarded.

Discard timer values are controlled by the ATU Configuration Register's Discard Timer Value bit. The ATU queues covered by discard timers are the IRQ and the IDWQ. After discarding a transaction, the ATU must set the Discard Timer Status bit in the ATU Configuration Register. The ATU does *not* assert the **SERR#** signal after discarding a transaction.

2.2.2 Outbound Transactions- Single Address Cycle (SAC) Internal Bus Transactions

Outbound transactions initiated by the 4138xx core processor are directed to the PCI interface through the ATU. The core processor always generates Single Address Cycles on the internal bus. As a PCI master, the ATU is capable of PCI I/O transactions, PCI memory reads (in case of conventional PCI), memory read DWORD (in case of PCI-X), PCI memory writes, configuration reads and writes, and DAC cycles. Outbound memory transactions are always attempted as 64-bit PCI transactions. Outbound memory write operations are performed as posted operations and outbound memory read operations are all performed as split read operations.

Outbound transactions use a separate set of queues from inbound transactions. Outbound write operations have their address entered into the outbound write address queue (OWADQ) and their data into the outbound write queue (OWQ). Outbound read transactions, performed as split transactions, use the Outbound Transaction Queue (OTQ) to store address, and get data returned into the outbound read queue (ORQ). Refer to Section 2.6.2 for details of outbound queue architecture. Outbound configuration transactions use a special outbound port structure. Refer to Section 2.2.3 for details.

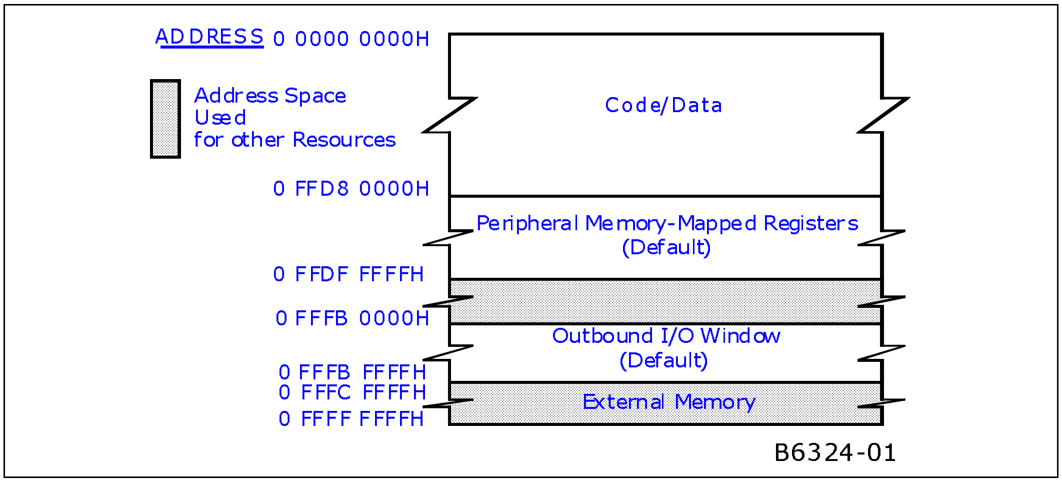
For outbound write transactions, the ATU is a target on the internal bus and initiator on the PCI bus. For outbound read transactions, the ATU is a completer on the internal bus (initially accepts the split read as a target and then provides read data by initiating a split completion). Internal bus operation is defined in Chapter 7.0, “System Controller (SC) and Internal Bus Bridge”. ATU specific internal bus operation is defined in Section 2.2.6.

2.2.2.1 Outbound Address Translation - Internal Bus Transactions

In addition to providing the mechanism for inbound translation, the ATU translates Intel XScale® processor-initiated cycles to the PCI bus. This is known as *outbound address translation*. Outbound transactions are processor or ADMA transactions targeted at the PCI bus. The ATU internal bus target interface claims internal bus cycles and completes the cycle on the PCI bus on behalf of the Intel XScale® processor or ADMA.

Figure 7 shows 4 Gbyte memory section 0 (Internal Bus Address [35:32] = 0000_2) of the 4138xx memory map with all reserved address locations highlighted. By default, the 64KByte outbound I/O window is from 0.FFFB.0000H to 0.FFFB.FFFFH while the PMMR registers, by default, reside from 0 FFD8.0000H to 0 FFDF.FFFFH.

Figure 7. 4 Gbyte Section 0 of the Internal Bus Memory Map



By default, Outbound Memory Window 0, Outbound Memory Window 1, Outbound Memory Window 2, and Outbound Memory Window 3 reside in 4 Gbyte memory sections 1, 2, 3 and 4 respectively, of the 64 Gbyte Internal Bus address space.

The ATU response to Outbound Transactions is globally controlled by the ATU Configuration Register Outbound Enable bit, as well as the Bus Master Enable bit in each function. When the Outbound Enable bit is deasserted, the internal bus outbound transaction master-abort are not forwarded to the PCI Express Domain. When the Outbound Enable bit is asserted, the relevant Bus Master Enable bit for each function determines appropriate response to an outbound transaction.

Table 4 describes the Outbound ATUs behavior for the different combinations of these control bits.

Table 4. Outbound Address Translation Control

Outbound Response	Outbound Enable[a] (ATUCR[1])	Bus Master Enable[b]
Master-Abort	0	0
Master-Abort	0	1
Retry	1	0
Claim[c]	1	1

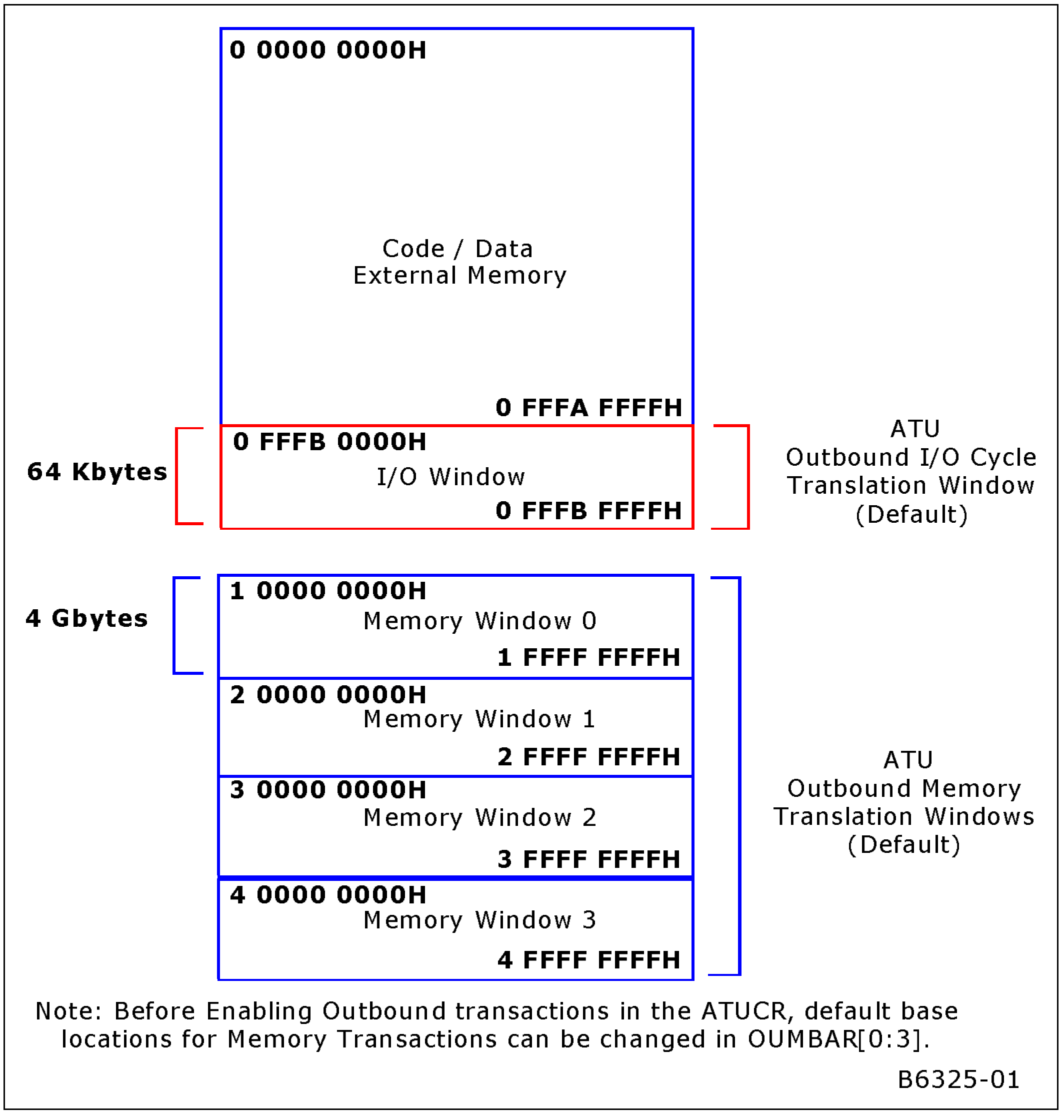
a. In addition, the outbound memory windows need to be individually enabled in order to claim. When, disabled, the outbound memory windows does not claim. By default, Outbound Memory Window 0, and Outbound Memory Window 1 are enabled. Outbound Memory Windows 2 and 3 are disabled by default.
b. In a multi-function configuration, each function independently controls its own Bus Master Enable bit.
c. The ATU may respond with a Retry in this case when the Outbound Transaction Queues are full.

2.2.2.2 Outbound Address Translation Windows

Inbound translation involves a programmable inbound translation window consisting of a base and limit register and a value register for PCI to internal bus translation. The outbound address translation windows use a similar methodology except that the outbound translation window limit sizes are fixed in the 4138xx internal bus address space; this removes the need for separate limit registers.

Figure 8 on page 69 illustrates the five outbound address translation windows.

Figure 8. Outbound Address Translation Windows



ATU has four 4 Gbyte outbound memory translation windows and one 64 Kbyte outbound I/O translation window. By default, Outbound Memory Window 0 (OUMBAR0), Outbound Memory Window 1 (OUMBAR1), Outbound Memory Window 2 (OUMBAR2), and Outbound Memory Window 3 (OUMBAR3) reside in 4 Gbyte memory sections 1, 2, 3, and 4, respectively. The default location of the 64 KByte outbound I/O window range is from 0.FFFB.0000H to 0.FFFB.FFFFH. The following registers are used to specify the five 4 Gbyte windows for claiming Outbound Memory transactions:

- Outbound Upper Memory Base Address Register 0 (OUMBAR0)
  - Default Value equal to 01H.
- Outbound Upper Memory Base Address Register 1 (OUMBAR1)
  - Default Value equal to 02H.
- Outbound Upper Memory Base Address Register 2 (OUMBAR2)
  - Default Value equal to 03H.
- Outbound Upper Memory Base Address Register 3 (OUMBAR3)
  - Default Value equal to 04H.
- Outbound I/O Base Address Register (OIOBAR)
  - Default Value equal to 0FFF B000H

An internal bus cycle with an address within one outbound window initiates a read or write cycle on the PCI bus. The PCI cycle type depends on which translation window the local bus cycle "hits". The read or write decision is based on the internal bus cycle type.

ATU has windows dedicated to the following outbound PCI/PCI-X transaction types:

- Memory reads and Memory writes - Memory Window
- I/O reads and writes - I/O Window

Table 5. Internal Bus-to-PCI Command Translation for Memory Windows

Internal Bus Command	Conventional PCI Command	PCI-X Command
Write[a]	Memory Write	Memory Write
Write[b]	Memory Write and Invalidate[c] (DMA) Memory Write (Memory Window 0-3)	Memory Write Block (DMA) Memory Write (Memory Window 0-3)
Read[d]	Memory Read	Memory Read DWORD
Read[e]	Memory Read Multiple	Memory Read Block

a. The internal bus request does **not** cross a QWORD address boundary.
b. The internal bus request does cross a QWORD address boundary
c. The ATU converts Write to Memory Write and Invalidate when the following four conditions are met, otherwise the Write is converted to a Memory Write:
   1.) Memory Write and Invalidate transactions are enabled in the "ATU Command Register - ATUCMD".
   2.) The Cache Line size is set to 8 or 16 DWORDS in the "ATU Cacheline Size Register - ATUCLSR".
   3.) Starting address of the Outbound PCI bus request is aligned to the cache line size.
   4.) Byte count intended for the Outbound PCI bus request is a multiple of the cache line size.
d. The internal bus request does **not** cross a DWORD address boundary.
e. The internal bus request does cross a DWORD address boundary

Table 6. Internal Bus-to-PCI Command Translation for I/O Window

Internal Bus Command[a]	Conventional PCI Command	PCI-X Command
Write	I/O Write	I/O Write
Read	I/O Read	I/O Read

a. User should designate memory region containing I/O Window as non-cacheable and non-bufferable from Intel XScale® processor. This insures all load/stores to I/O Window are of DWORD quantities. In the event that the user inadvertently issues a read to the I/O Window which crosses a DWORD address boundary, the ATU target aborts the transaction. Only bytes 3:0 are relevant dependent on the Byte Enables.

The translation portion of outbound ATU transactions is accomplished with a value register in the same manner as inbound translations. Each outbound memory window is associated with one translation register which provides the upper translation addresses (OUMWVR0-3). When the corresponding OUMWVRx register is all-zero a SAC transaction is generated on the PCI bus. Otherwise, a DAC is generated on the PCI bus using the value in the OUMWVRx register for the upper 32-bit address. ATU uses the following registers during outbound address translation:

- Outbound Upper 32-bit Memory Window Value Register 0 (OUMWVR0)
- Outbound Upper 32-bit Memory Window Value Register 1 (OUMWVR1)
- Outbound Upper 32-bit Memory Window Value Register 0 (OUMWVR2)
- Outbound Upper 32-bit Memory Window Value Register 1 (OUMWVR3)
- Outbound I/O Window Value Register (OIOWVR)
- Outbound Configuration Cycle Address Register (OCCAR)

See Section 2.14 for details on outbound translation register definition and programming constraints.

The translation algorithm used, as stated, is very similar to inbound translation. For memory transactions, the algorithm is:

Equation 3. Outbound Address Translation

```
PCI Address = (Internal_Bus_Address & 0.FFFF.FFFFH) | (Upper_Window_Value_Register << 32)
```

For memory transactions, the internal bus address is bitwise ANDed with the inverse of 4 Gbytes which clears the upper 4 bits of the 36 bit address. The result is bitwise ORed with the outbound upper window value register left shifted by 32 to create the Upper 32-bits of the PCI address. When the Upper 32-bits of the PCI Address equals 0000 0000H, the ATU generates a SAC transaction on the PCI bus, otherwise, a DAC transaction is used.

For I/O transactions, the algorithm is:

Equation 4. I/O Transactions

```
PCI Address = (Internal_Bus_Address & 0.0000.FFFFH) | Window_Value_Register
```

For I/O transactions, the internal bus address is bitwise ANDed with the inverse of 64 Kbytes which clears the upper 20 bits of address. Address aliasing is prevented by the outbound window value registers which only allow values on boundaries equivalent to the window’s length.

2.2.3 Outbound Write Transaction

An outbound write transaction is initiated by the Intel XScale® processor[4] or by one of the ADMA channels and is targeted at a PCI target on the PCI bus. The outbound write address and write data are propagated from the 4138xx internal bus to a PCI bus through OWADQ and OWQ, respectively.

The ATUs internal bus target interface claims the write transaction and forwards the write data through to the targeted PCI bus. The data flow for an outbound write transaction on the internal bus is summarized in the following statements:

- For Single Address Cycles (SACs), ATU internal bus target interface latches the address from the internal bus into the OWADQ when that address is inside one of the outbound translate windows (see Section 2.6) and the OWQ is not full.
- For Dual Address Cycles (DACs), ATU internal bus target interface latches address from the internal bus into the OWADQ when the OWQ is not full and OWADQ is not full.
- Once outbound address is latched, internal bus target interface stores write data into the OWQ until the internal bus transaction completes or the reaches a buffer boundary. The initiator of the transaction is disconnected at an ADB when the transaction reaches a buffer boundary.
- When the OWADQ is full, the target interface signals a Retry on the internal bus to the outbound cycle initiator.
- When OWADQ latches the address and corresponding data is latched in a buffer in OWQ, the outbound cycle is enabled for transmission on the PCI Bus and PCI interface requests PCI bus.

4. For best performance, the user should designate the two Outbound Memory Windows as non-cacheable and bufferable from the Intel XScale® processor. This assignment enables the Intel XScale® processor to issue multiple outstanding transactions to the Outbound Memory Windows, thereby, taking full advantage of the ATU outbound queue architecture. However, the user needs to be aware that the Outbound ATU queue architecture does not maintain strict ordering between read and write requests as described in Table 14, "ATU Outbound Data Flow Ordering Rules" on page 88. In the event that the user requires strict ordering to be maintained. In the event that the user requires strict ordering to be maintained, the user must change the designation of this region of memory to be non-cacheable/non-bufferable and enforce the requirement in software.

The PCI interface is responsible for completing the outbound write transaction with the PCI address translated from the OWADQ and the data in the OWQ. The data flow for an outbound write transaction on the PCI bus is summarized in the following statements:

- ATU PCI interface requests PCI bus, when completed internal bus transaction is in OWADQ and data associated with transfer in OWQ. Once bus is granted, PCI master interface writes PCI translated address from OWADQ to PCI bus and waits for transaction to be claimed.
- When Master Abort seen during address phase, transaction flushed and OWADQ/OWQ are cleared. Section 2.7.5 has full details on PCI master abort conditions during outbound transactions.
- In conventional PCI mode, once PCI write transaction is claimed, the PCI interface transfers data from the OWQ to the PCI bus until one of the following is true:
  - — PCI target signals a Retry or Disconnect. The ATU PCI master attempts to reacquire the PCI bus to complete the write transaction.
  - — **GNT#** signal is deasserted and master latency timer has expired. In this case, master interface attempts to reacquire PCI bus and complete write transaction.
  - — PCI target signals a Target-Abort. In this case, OWQ and OTQ are cleared and transaction aborted. Appropriate error bits are set as defined in Section 2.7.6.
  - — Transaction terminates normally by transferring all data (full byte count) associated with it. The write address is removed from the OWADQ and the interface returns to idle.
- In PCI-X mode, once the PCI memory write transaction is claimed, the PCI interface transfers data from the OWQ to the PCI bus until one of the following is true:
  - — The PCI target signals a Retry or Single Data Phase Disconnect. The ATU PCI initiator attempts to reacquire the PCI bus to complete the write transaction.
  - — Reacquire the PCI bus to complete the write transaction.
  - — **GNT#** signal is deasserted and the master latency timer has expired. In this case, the master interface attempts to reacquire the PCI bus and complete the write transaction.
  - — PCI target signals a Target-Abort. In this case, the OWQ and OWADQ are cleared and the transaction is aborted. The appropriate error bits are set as defined in Section 2.7.6.
  - — Transaction terminates normally with Satisfaction of Byte Count. The write address is removed from the OWADQ and the interface returns to idle.
- In the PCI-X mode, once the PCI I/O write transaction is claimed, the PCI interface transfers data from the OWQ to the PCI bus until one of the following is true:
  - — PCI target signals a Retry. The ATU PCI initiator attempts to reacquire the PCI bus to complete the write transaction.
  - — Transaction terminates normally with Satisfaction of Byte Count or with Single Data Phase Disconnect. The write address is removed from the OWADQ and the interface returns to idle.
  - — PCI target signals a Target-Abort. In this case, the OWQ and OWADQ are cleared and the transaction is aborted. The appropriate error bits are set as defined in Section 2.7.6.
  - — Transaction terminates with Split Response Termination. The write address is removed from the OWADQ and the interface returns to idle only when it receives the corresponding Split Completion Message.

  When an uncorrectable data error is encountered (**PERR#** detected), the master interface continues writing data to clear the queue.

  In the conventional PCI mode when the PCI target deasserts **TRDY#**, no action is taken by the ATU master other than inserting wait states.

2.2.4 Outbound Read Transaction

An outbound read transaction is initiated by the Intel XScale® processor[5] or an ADMA channel and is targeted at a PCI slave on the PCI bus. The read transaction is propagated through the outbound transaction queue (OTQ) and read data is returned through the outbound read queue (ORQ).

The ATUs internal bus target interface claims the Memory Read Dword and Memory Read Block and Alias to Memory Read Block transaction and forwards the read request through to the PCI bus and returns the read data to the internal bus.

The data flow for an outbound read transaction on the internal bus is summarized in the following statements:

- For Single Address Cycles (SACs), the ATU internal bus interface latches the internal bus address when the address is inside an outbound address translation window and OTQ is not full. For Dual Address Cycles (DACs), ATU internal bus target interface latches the address from the internal bus into OTQ irrespective of the address. All read transactions are handled as split transactions. When OTQ is full (previous outbound transactions in progress), the internal bus interface signals a Retry to the transaction initiator.
- When during the completion cycle on the PCI interface, a master abort is encountered, a flag is set and the ATU aborts the completion to the internal bus requester. The OTQ is cleared of the transaction.
- When during the completion cycle on the PCI interface, a target abort is encountered, a flag is set and the ATU aborts the completion to the internal bus requester. The OTQ is cleared of the transaction.
- Once the transaction completes on the PCI bus, the ATU generates a split completion transaction to return data to the internal bus requester.
- When operating in the PCI-X mode, ATU may receive a split completion error message when attempting to read data on the PCI bus. In this case, ATU notifies the internal bus requester about the error by aborting the completion to the internal bus requester. The OTQ is cleared of the transaction.

The data flow for an outbound read transaction on the PCI bus is summarized in the following statements:

- The ATU PCI interface requests the PCI bus when the head of the OTQ has at least one entry and the ordering rules are satisfied. Once the bus is granted, the PCI interface transfers the PCI translated address from the OTQ to the PCI bus and wait for the transaction to be claimed.
- When no **DEVSEL#** is asserted, a master abort is signaled. This is passed through to the internal bus target interface.
- When a target abort is signaled from the PCI target, the target abort is returned to the internal bus and the PCI interface returns to idle.
- When operating in the PCI-X mode the read transaction may terminate as a split response termination. Then the ATU receives data during the corresponding split completion transaction. When an error occurs, the ATU may receive a split completion error message.

---

5. For best performance, designate the two Outbound Memory Windows as non-cacheable and bufferable from the Intel XScale® processor. This assignment enables the Intel XScale® processor to issue multiple outstanding transactions to the Outbound Memory Windows, thereby, taking full advantage of ATU outbound queue architecture. However, be aware that the Outbound ATU queue architecture does not maintain strict ordering between read and write requests as described in Table 14, "ATU Outbound Data Flow Ordering Rules" on page 88. In the event that the user requires strict ordering to be maintained, the user must change the designation of this region of memory to be non-cacheable/non-bufferable and enforce the requirement in software.

2.2.5 Outbound Configuration Cycle Translation

Outbound ATU provides a port programming model for outbound configuration cycles.

Performing an outbound configuration cycle to the PCI bus involves up to two internal bus cycles:

1. Writing Outbound Configuration Cycle Address Register (OCCAR) with PCI address used during configuration cycle. See the *PCI-X Protocol Addendum to the PCI Local Bus Specification,* Revision 2.0 for information regarding configuration address cycle formats. This IB bus cycle enables the transaction.
2. Writing or reading Outbound Configuration Cycle Data Register (OCCDR). A read causes a configuration cycle read to the PCI bus with the address in the outbound configuration cycle address register. Note that the Internal Bus read is executed as a split transaction. Similarly, a write initiates a configuration cycle write to PCI with the write data from the second processor cycle. Configuration cycles are non-burst and restricted to a single 32-bit word cycle. [6]

When the Configuration Cycle Data Register is written, data is latched and forwarded to the PCI bus with the internal target issuing a single data phase disconnect with 32-bit data only. This cycle does not receive an **ACK64#** from the ATU and therefore is defined as 32-bit only.

Note, the programming model uses the register interface for outbound configuration cycles, from a hardware standpoint, the address is entered into OTQ (reads) or OWADQ (writes), configuration write data goes through OWQ and configuration read data is returned in the ORQ.

Note: Outbound configuration cycle data registers are not physical registers. They are 4138xx memory mapped addresses used to initiate a transaction with the address in the associated address register.

2.2.5.1 PCI-X Mode 1 Considerations for Outbound Configuration Cycles

Configuration cycle address Bits 15:11 for Type 0 configuration cycles are defined differently for Conventional versus PCI-X modes. When 4138xx software programs OCCAR to initiate a Type 0 configuration cycle, always load OCCAR based on the PCI-X definition for Type 0 configuration cycle address. In Conventional mode, 4138xx clears OCCAR bits 15:11 prior to initiating an outbound Type 0 configuration cycle.

During the attribute phase of a Type 0 configuration transaction, the Secondary Bus Number field (bits 7:0) is set equal to the Requester Bus Number (bits 15:8 of the “PCI-X Status Register - PCIXSR” on page 193).

6. The designate the memory region containing OCCDR as non-cacheable and non-bufferable from the Intel XScale® processor. This insures that all load/stores to OCCDR are only of DWORD quantities. In event the user inadvertently issues a read to OCCDR that crosses a DWORD address boundary, the ATU target aborts the transaction. All writes are terminated with a Single-Phase-Disconnect and only bytes 3:0 is relevant.

2.2.5.2 PCI-X Mode 2 Considerations for Outbound Configuration Cycles

In addition to the PCI-X Mode 1 changes relative to Conventional PCI mode, for PCI-X Mode 2, the definition for bits 31:24 of the configuration address has changed. Bits 31:28 are Reserved while bits 27:24 represent the enhanced configuration cycle upper register address providing up to 4 Kbytes of configuration register space. In addition, a consequence of this change is that Device Numbers 15:8 are not longer available. For Mode 2 implementations, the System hardware is restricted to Device Numbers 7:0. Software needs to read the PCSR to confirm that the PCI interface is operating in Mode 2 in order to set bits 27:24 properly.

2.2.5.3 Outbound Configuration Cycle Error Conditions

Master aborts during outbound configuration reads result in ATU aborting the read completion the on internal bus.

Target aborts during outbound configuration reads result in ATU aborting the read completion on the internal bus.

Uncorrectable errors during outbound configuration reads result in ATU aborting the read completion on the internal bus.

Uncorrectable errors detected by target of an outbound configuration write may result in the ATU receiving either of the two Split Completion Write Uncorrectable Data Error Messages (with message class=2h -completer error and message index=01h - split write uncorrectable data error or with message class=1h - bridge error and message index=02h - write uncorrectable data error) on the PCI bus. When Parity Checking is enabled, the ATU sets error bits in the ATUSR and the PCIXSR. The Intel XScale® processor is interrupted when the Split Completion Error and/or Master Data Parity interrupt(s) are unmasked.

2.2.6 Internal Bus Operation

Complete internal bus operation of the 4138xx is defined in Chapter 7.0, "System Controller (SC) and Internal Bus Bridge". The ATU acts as both internal bus master and internal bus slave device. A summary of ATU internal bus specific operation follows for both master and slave operation

2.3 Big Endian Byte Swapping

Each memory and I/O window has an associated byte swapping enable located in the following address translation registers:

- bit 0 of Inbound Address Translate Value Register 0-3 (IATVR0-3)
- bit 0 of Inbound Expansion ROM Translate Value Register (ERTVR)
- bit 0 of Outbound I/O Window Translate Value Register (OIOWTVR)
- bit 27 of Outbound Upper Memory BAR 0-3 (OUMBAR0-3)

Note: The Messaging Unit (MU) Memory is mapped in PCI Window 0 (ATU Base Address Register 0) along with the MSI-X table structures. Byte swapping should not be enabled for BAR0 when using MSI-X.

2.3.1 Inbound Byte Swapping

When enabled, the swapping occurs as described in Figure 9, "Inbound Byte Swapping for 32-bit PCI" on page 78 and Figure 10, "Inbound Byte Swapping for 64-bit PCI" on page 78. The bytes are swapped within a DWORD and byte swapping is performed for all transactions regardless of byte count.

Figure 9. Inbound Byte Swapping for 32-bit PCI

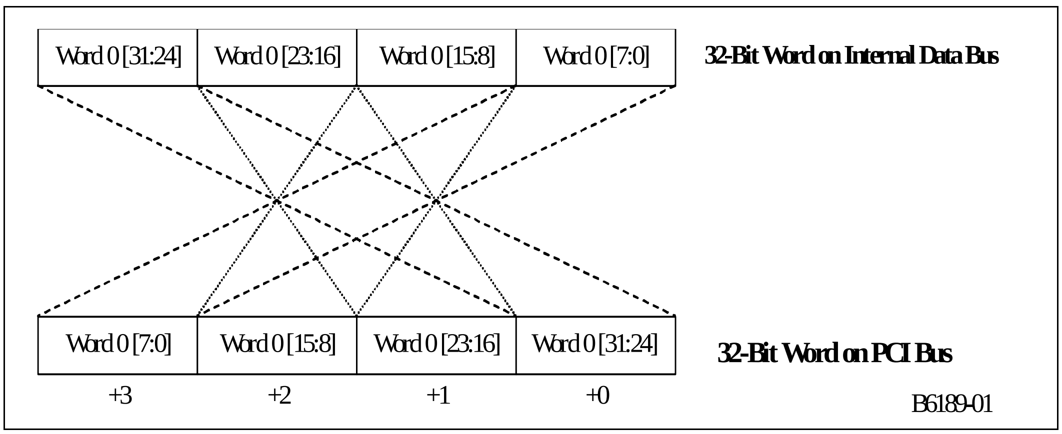
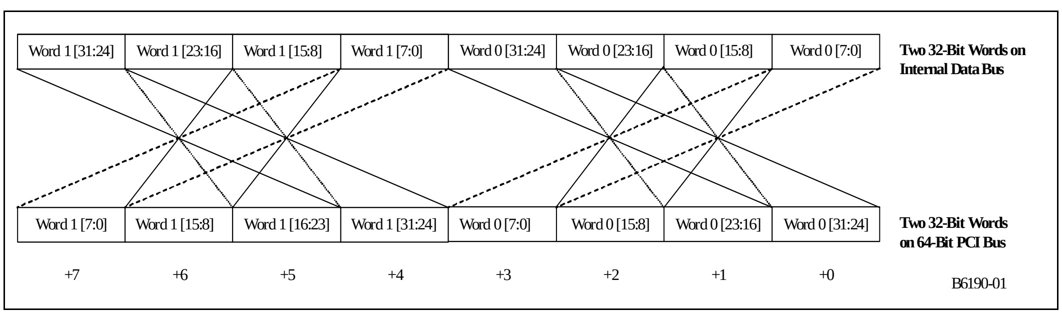


Figure 10. Inbound Byte Swapping for 64-bit PCI



2.3.2 Outbound Byte Swapping

When enabled, the swapping occurs as described in Figure 11, "Outbound Byte Swapping for Transaction with Byte Count of 1" on page 79, Figure 12, "Outbound Byte Swapping for Transaction with Byte Count of 2" on page 79, and Figure 13, "Outbound Byte Swapping for Transaction with Byte Count of 3 or Larger" on page 79. The bytes are swapped within a 32-bit DWORD and the type of byte swapping performed is determined by the transaction byte count. For Byte Count of 3 or larger transactions, no byte swapping is performed.

Note: The byte swapping capability of the ADMA unit should be used to swap bytes within each DWORD for PCI-to-Memory Read/Write DMA transfers.

Figure 11. Outbound Byte Swapping for Transaction with Byte Count of 1

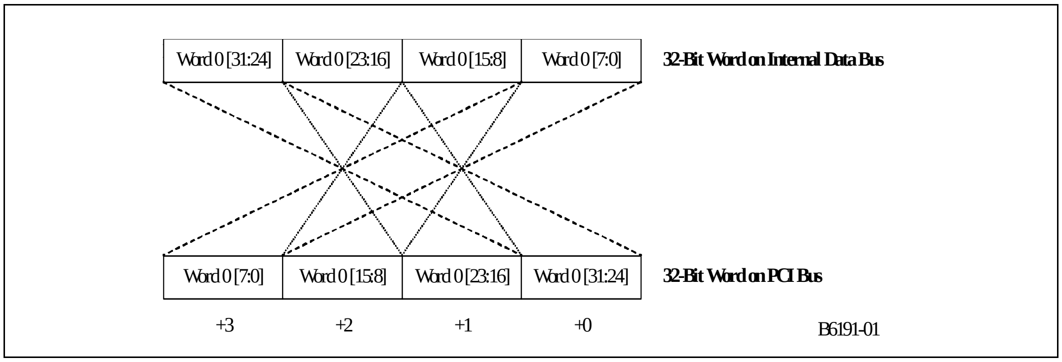


Figure 12. Outbound Byte Swapping for Transaction with Byte Count of 2

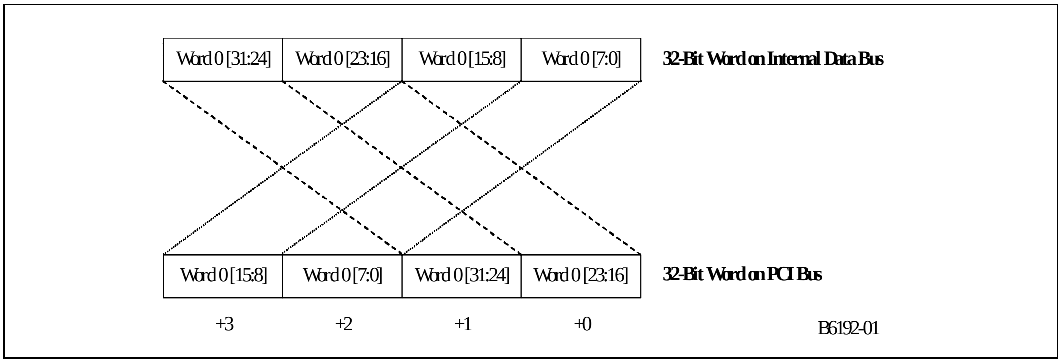
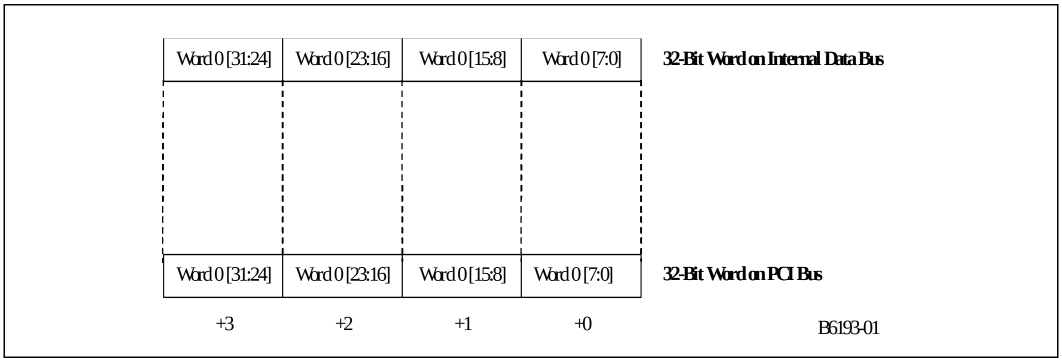


Figure 13. Outbound Byte Swapping for Transaction with Byte Count of 3 or Larger



2.4 CompactPCI Hot-Swap

The 4138xx meets the standard requirements to be considered "Hot-Swap Silicon" detailed in the *Compact PCI Hot-Swap Specification*, Revision 2.1. This includes a dedicated pin interface and extended capability header.

Hot-Swap Control and Status Register is implemented via the Extended Capability Pointer mechanism in the ATUs configuration space (Section 2.14.61, "CompactPCI Hot-Swap Capability ID Register").

Provides Software Connection Control Resources for ENUM#, Hot-Swap Switch, blue LED and Device Hiding.

Handle Switch de-bouncing is implemented.

See Section 2.4.1.1, "Compact PCI Hot-Swap Mode Select" for a description of **HS_SM#** and HS_FREQ[1:0] requirements.

2.4.1 Pin Interface

Table 7. Compact PCI Hot-Swap

Signal	Width	Type	Description
HS_ENUM#	1	Od-O	Compact PCI Hot-Swap Event: Conditionally asserted to notify the system host that either a board has been freshly inserted or is about to be extracted. This signal informs the system host that the configuration of the system has changed. The system host then performs any necessary maintenance such as installing or quiesing a device driver.
HS_LSTAT	1	I	Compact PCI Hot-Swap Latch Status: An input indicating the state of the ejector switch. 0 = Indicates the ejector switch is closed. 1 = Indicates the ejector switch is open. When Compact PCI Hot-Swap is not supported, this signal must be tied low.
HS_LED_OUT	1	O	LED Output: 4138xx outputs a logic one to illuminate the Hot-Swap LED.
HS_SM#	1	I	Hot-Swap Startup Mode (Strap): This strap is sampled at the rising edge of P_RST# to indicate Hot-Swap Mode. When 1b and Configuration Retry Mode enabled (bit 2 of the PCSR), indicates 4138xx retries all Configuration transactions. When 0b and Configuration Retry Mode enabled (bit 2 of the PCSR), indicates 4138xx does not claim Configuration transactions and the bus mode is determined by the HS_FREQ[1:0] straps.
HS_FREQ[1:0]	2	I	This pins are reserved for determining Bus frequency and mode during a PCI-X Hot-Swap event and are only valid when **HS_SM#** = 0. The bus frequency and mode are described below: 11 => PCI Mode, 33 or 66 MHz. Use P_M66EN to determine frequency 10 => PCI-X Mode, 66 MHz 01 => PCI-X Mode, 100 MHz 00 => PCI-X Mode, 133 MHz These pins are not used for non-Hot-Swap systems.
Total	6		

2.4.1.1 Compact PCI Hot-Swap Mode Select

HS_SM# must be asserted (0b) to enable Hot-Swap functionality.

HS_FREQ[1:0] pins allow the 4138xx to determine the cPCI backplane operating frequency without needing to see a PCI-X initialization pattern. These pins are only valid when **HS_SM#** is sampled as 0b during P_RST#.

Table 8. HS_FREQ Encoding [a]

HS_FREQ[1:0]	P_M66EN	Operating Mode	Bus Frequency	PCSR[19:16]
11	0	PCI	33 MHz	1111
11	1	PCI	66 MHz	1111
10	-	PCI-X (Mode 1)	66 MHz	1110
01	-	PCI-X (Mode 1)	100 MHz	1101
00	-	PCI-X (Mode 1)	133 MHz	1100

a. Hot-Swap is not supported in PCI-X Mode 2.

2.5 Expansion ROM Translation Unit

The inbound ATU supports one address range (defined by a base/limit register pair) used for the Expansion ROM. Refer to the *PCI Local Bus Specification*, Revision 2.3 for details on Expansion ROM format and usage.

During a powerup sequence, initialization code from Expansion ROM is executed once by the host processor to initialize the associated device. The code can be discarded once executed. Expansion ROM registers are described in Section 2.14.37 through Section 2.14.39.

The inbound ATU supports an inbound Expansion ROM window which works like the inbound translation window. A read from the expansion ROM windows is forwarded to the internal bus. The address translation algorithm is the same as the inbound translation; see Section 2.2.1.1, "Inbound Address Translation" on page 56. As a PCI target, the Expansion ROM interface behaves as a standard ATU interface and is capable of returning a 64-bit access by the assertion of **ACK64#** in response to a 64-bit request.

The Expansion ROM unit uses the ATU inbound transaction queue and the inbound read data queue.

When operating in the conventional PCI mode, the address of the inbound delayed read cycle is entered into the ITQ queue and the delayed read completion data is returned in the IRQ. That is, inbound reads to the Expansion ROM window are handled as delayed transactions on the PCI bus.

When operating in the PCI-X mode, the address of the inbound read cycle is entered into the ITQ queue and the read completion data is returned in the IRQ. That is, inbound reads to the Expansion ROM window are handled as split transactions on the PCI bus. The internal bus initiator interface fills the IRQ read queue with the full byte count before generating the split completion transaction on the PCI bus. That is, the ATU generates a Read request on the internal bus with byte count set to the byte count specified in the PCI read. The PBI (Flash Interface) returns data in:

- either one or more 1024 byte split completion transactions when byte count is greater than or equal to 1024 bytes or
- one split completion with the full byte count when byte count is less than 1024 bytes.

Expansion ROM writes are not supported and result in a Target Abort.

2.6 ATU Queue Architecture

ATU operation and performance depends on the queueing mechanism implemented between the internal bus interface and PCI bus interface. As indicated in Figure 4, the ATU queue architecture consists of separate inbound and outbound queues. The function of each queue is described in the following sections.

2.6.1 Inbound Queues

The inbound data queues of the ATU support transactions initiated on a PCI bus and targeted at either 4138xx local memory or a 4138xx memory mapped register. Table 9 details the name and sizes of the ATU inbound data queues.

Table 9. Inbound Queues

Queue Mnemonic	Queue Name	Queue Size (Bytes)
IWQ	Inbound Write Data Queue	4 KBytes (4*1KB)
IWADQ	Inbound Write Address Queue	4 Transaction Addresses
IRQ	Inbound Read Data Queue	4 KBytes (4*1KB)
IDWQ	Inbound Delayed Write address/data Queue	1 Transaction
ITQ	Inbound Transaction Queue	8 Addresses/Commands

2.6.1.1 Inbound Write Queue Structure

The ATU Inbound Write Queues consist of the inbound write data queue and the inbound write address queue. The inbound write data queue holds the data for memory write transactions moving from a PCI Bus to the internal bus and the address queue holds the corresponding address of the transactions in the data queues. The inbound write queue, IWQ, has a queue depth of 4 KBytes and moves write transactions from the PCI bus to the internal bus. The corresponding address queue, IWADQ, is capable of holding 4 address entries. The queue pair is capable of holding up to 4 memory write (or MWI when operating in the conventional PCI mode) transactions.

The following rules apply to the PCI bus interface and govern the acceptance of data into IWQ and address into the tail of the IWADQ:

- A memory write operation claimed by the target PCI interface on the PCI bus is accepted into the address and data queues when the IWADQ has at least one address entry available.
- When operating in the conventional PCI mode, when the IWQ reaches a full state while filling, a disconnect with data is signaled to the master of the transaction.
- When operating in the PCI-X mode, when the IWQ reaches a full state while filling, a disconnect at next ADB is signaled to the master of that transaction.

Memory write transactions are drained from the head of the queue when the master interface has acquired bus ownership and transaction ordering and priority have been satisfied (see Section 2.6.3). A memory write transaction is considered drained from the queue when the entire amount of data entered on the PCI bus has been accepted by the internal bus target. Error conditions resulting in the cancellation of a write transaction only flush the transaction at the head of the data and address queue. All other transactions within the queues are considered still valid.

2.6.1.2 Inbound Read Queue Structure

The inbound read queues are responsible for retrieving data from local memory and returning it to the PCI bus in response to a read transaction initiated from a PCI master. When operating in the conventional PCI mode, reads are handled as delayed transactions. When operating in the PCI-X mode reads are handled as split transactions. The address of the transactions are held in the ITQ. Up to 8 read requests can be stored in the ITQ. The read data is returned through IRQ.

When operating in the conventional PCI mode, the IRQ holds data from four PCI bus read transactions. The read request cycle on PCI latches the read command and the address into the ITQ when the cycle is first initiated by the PCI master. The ATU internal bus initiator interface takes the translated address and the command and performs a read on the internal bus. Reads can be any of the PCI memory read command types using the ATU inbound translation or an inbound configuration read using the specific configuration cycle translation. The data from the read on the internal bus is stored in the IRQ until the PCI master initiates a read cycle that matches the initial request cycle in both command and address. Any data left in an IRQ after the delivery of a completion cycle on PCI is flushed. This is possible since all internal bus memory is considered prefetchable with no read side effects.

When operating in the PCI-X mode, the IRQ may hold data from up to four PCI bus read transactions. The read request cycle on PCI latches the read command and the address into the ITQ when the cycle is first initiated by the PCI master. The ATU internal bus initiator interface takes the translated address and the command and performs a read on the internal bus. Reads can be any of the PCI memory read command types using the ATU inbound translation or an inbound configuration read using the specific configuration cycle translation. Once read data is available in the IRQ, the ATU generates one or more split completions to return read data to the PCI requester.

When operating in the conventional PCI mode, the exact amount of data (byte count) read by the master state machine on the internal bus interface depends upon the read command used and how much data the Internal Bus target device delivers. Table 10 shows the amount of data attempted to be read for the different memory read commands for the ATU, when operating in the conventional PCI mode.

Internal bus error conditions override all prefetch amounts (i.e., a master-abort and target-abort conditions).

Table 10. Inbound Read Prefetch Data Sizes

PCI Read Command	Prefetch Size (Bytes)
Memory Read	4 to 32
Memory Read Line	4 to 128
Memory Read Multiple	4 to 1024

2.6.1.3 Inbound Delayed Write Queue

The IDWQ is used specifically for inbound configuration write cycles to the ATU. I/O Write transactions are not accepted by the ATU and result in a Master Abort.

The IDWQ contains both the address and data of a configuration write cycle. When operating in the conventional PCI mode, the configuration writes are handled as delayed writes. When operating in the PCI-X mode, the configuration writes are handled as split writes. When the write cycle is initiated on the PCI bus, the address and data are entered into the 8 byte queue, and forwarded to the internal bus. The transaction is forwarded to the internal bus once transaction ordering has been satisfied. The status of the transaction (normal completion) is maintained in the IDWQ for return to the PCI master on the initiating bus. When operating in the PCI-X mode, a write completion message is generated by the ATU to indicate the successful execution of the configuration write transaction.

The IDWQ can only hold 32-bits of data and should never be accessed with **REQ64#** asserted per the *PCI Local Bus Specification*, Revision 2.3 which states that "only memory transactions support 64-bit data transfers". In addition, the cycle should always return only 32-bits of data on the internal bus.

2.6.1.4 Inbound Transaction Queues Command Translation Summary

Table 11. PCI to Internal Bus Command Translation for All Inbound Transactions

PCI Command	Conventional Mode	PCI-X Mode	Internal Bus Command
Memory Write	a	a	Write
Memory Write and Invalidate	a		Write
Memory Write Block		a	Write
Alias to Memory Write Block		a	Write
Memory Read	a		Read
Memory Read Line	a		Read
Memory Read Multiple	a		Read
Memory Read Block		a	Read
Alias to Memory Read Block		a	Read
Memory Read DWORD		a	Read
Configuration Read	a	a	Read
Configuration Write	a	a	Write
Split Read Completion		a	Data Transfer
Split Write Completion		a	None
Split Completion Error Message		a	Data Transfer Abort
All Other Commands Not Claimed by the ATU	a	a	N/A

2.6.2 Outbound Queues

The outbound queues of the ATU are used to hold read and write transactions from the core processor directed at the PCI bus. Each ATU outbound queue structure has a separate read queue, write queue, and address queue. Table 12 contains information about ATU outbound queues.

Table 12. Outbound Queues

Queue Mnemonic	Queue Name	Queue Size (Bytes)
OWQ	Outbound Write Data Queue	4 KBytes (4*1024B)
OWADQ	Outbound Write Address Queue	4 Transaction Addresses
ORQ	Outbound Read Data Queue	2 or 4 KBytes (4* 512B or 4*1024B)[a]
OTQ	Outbound Transaction Queue	8 Addresses/Commands

a. The ORQ can be throttled between 2 Kbytes or 4 Kbytes depending on the setting of the Maximum Memory Read Byte Count (MMRBC) field of the PCI-X Command register (see Section 2.14.55, "PCI-X Command Register - PCIXCMD" on page 191). When the MMRBC is set to 512 bytes (default value), the ORQ is only capable of handling 2 Kbytes of SRC data, otherwise, the ORQ handles 4 Kbytes of SRC data.

The outbound queues are capable of holding outbound memory read, memory write, I/O read, and I/O write transactions. The type of transaction used is defined by the internal bus address and the command used on the internal bus. See Section 2.2.2 and Section 2.2.3 for details on outbound address translation.

When an internal bus agent initiates an outbound write transaction, the address is entered into the OWADQ (when not full). The data from the internal bus write is then entered into the OWQ and the transaction is forwarded to the PCI bus. When the write completes (or an error occurs), the address is flushed from the OWADQ. Data is flushed only for the master abort or target abort cases.

For outbound reads, the address is entered into the OTQ (when not full) and a split response termination is signaled to the requester on the internal bus. Read data is fetched and returned to the requester on the internal bus.

2.6.2.1 Relaxed Ordering and No Snoop Outbound Request Attributes

In PCI-X mode, the ATU may set the Relaxed Ordering (RO) bit 29 of the Requester Attributes and/or the No Snoop (NS) bit 30 of the Requester Attributes for an outbound request.

For any other outbound requests, the NS and RO attribute bits are set to 0.

Note: The *PCI-X Protocol Addendum to the PCI Local Bus Specification,* Revision 2.0 permits the ATU to set the RO bit to '1' in the Requester Attributes only when enabled by bit 1 of the "PCI-X Command Register - PCIXCMD" on page 191.

2.6.3 Transaction Ordering

Because the ATU can process multiple transactions, they must maintain proper ordering to avoid deadlock conditions and improve throughput. The ATU transaction ordering rules used by the 4138xx are listed in Table 13 for the inbound direction and Table 14 on page 88 for the outbound direction. The tables are based on the direction the transaction is moving, i.e. the data for outbound delayed read moves in the same direction as the data for an inbound write or the address/command for an inbound read. When operating in the PCI-X mode, the ATU ignores the Relaxed Ordering Attribute.

Note: Outbound Non-Posted Writes are the result of Internal Bus Memory writes that are claimed by either the I/O translation window or the Outbound Configuration Cycle Data Register - OCCDR. Though these write requests arrive on the PCI bus as non-posted write requests, it is important to note that from the Intel XScale® processor point of view, these internal bus memory write requests are posted into the Outbound ATU transaction queue. Furthermore, in PCI-X mode non-posted write requests have the potential to be split. Thus, even though a split write completion may be returned to the ATU on the PCI bus for a given outbound non-posted write request, the split write completion does not passed back through to the internal bus. Additionally, strong ordering between outbound memory (posted) write requests and outbound non-posted write requests are ***not*** maintained as indicated in Table 14 on page 88.

For best performance, the user should designate the two Outbound Memory Windows as non-cacheable and bufferable from the Intel XScale® processor. This assignment enables the Intel XScale® processor to issue multiple outstanding transactions to the Outbound Memory Windows, thereby, taking full advantage of the ATU outbound queue architecture. However, the user needs to be aware that the Outbound ATU queue architecture does not maintain strict ordering between read and write requests as described in Table 14, "ATU Outbound Data Flow Ordering Rules" on page 88. In the event that the user requires strict ordering to be maintained, the user must change the designation of this region of memory to be non-cacheable/non-bufferable and enforce the requirement in software.

Table 13. ATU Inbound Data Flow Ordering Rules

Row Pass Column?[a]	ATU Inbound Writes	Inbound Delayed Read Request (PCI mode)	Inbound Split Read Request (PCI-X mode)	Inbound Configuration Write Request	Outbound Split Read Completion
ATU Inbound Writes	No	Yes	Yes	Yes	Yes
Inbound Delayed Read Request (PCI mode)	No	No	NA	No	Yes
Inbound Split Read Request (PCI-X mode)	No	NA	No	No	Yes
Inbound Configuration Write Request	No	No	No	NA	Yes
Outbound Split Read Completion	No	Yes	Yes	Yes	Yes

a. Outbound Non-Posted Write Completions are not included in this table since these transactions are never passed back to the Internal Bus Requester (Intel XScale® processor). The reason is that from the Intel XScale® processor's point of view, these write requests are posted into the Outbound ATU transaction queue.

Table 14. ATU Outbound Data Flow Ordering Rules

Row Pass Column?	Outbound Write		Outbound Read Request	Inbound Read Completion		Inbound Write Completion	
	Posted	Non-Posted		DRC	SRC	DWC[a]	SWC[b]
Outbound Posted Write Request	No	Yes	Yes	Yes	Yes	Yes	Yes
Outbound Non-posted[c] Write Request	No	No	No	Yes	Yes	Yes	Yes
Outbound Read Request	No	No	No	Yes	Yes	Yes	Yes
Inbound Delayed Read Completion (DRC)	No	Yes	Yes	Yes	NA	Yes	NA
Inbound Split Read Completion (SRC)	No	Yes	Yes	NA	Yes	NA	No
Inbound Delayed Write Completion (DWC)	No	Yes	Yes	Yes	NA	NA	NA
Inbound Split Write Completion (SWC)	No	Yes	Yes	NA	Yes	NA	NA

a. Since the Inbound DWR transaction queue is one-deep, the passing rule associated with DWC vs. DWC is moot (i.e., NA).
b. Since the Inbound SWR transaction queue is one-deep, the passing rule associated with SWC vs. SWC is moot (i.e., NA).
c. Outbound Non-posted writes include I/O writes and Configuration writes.

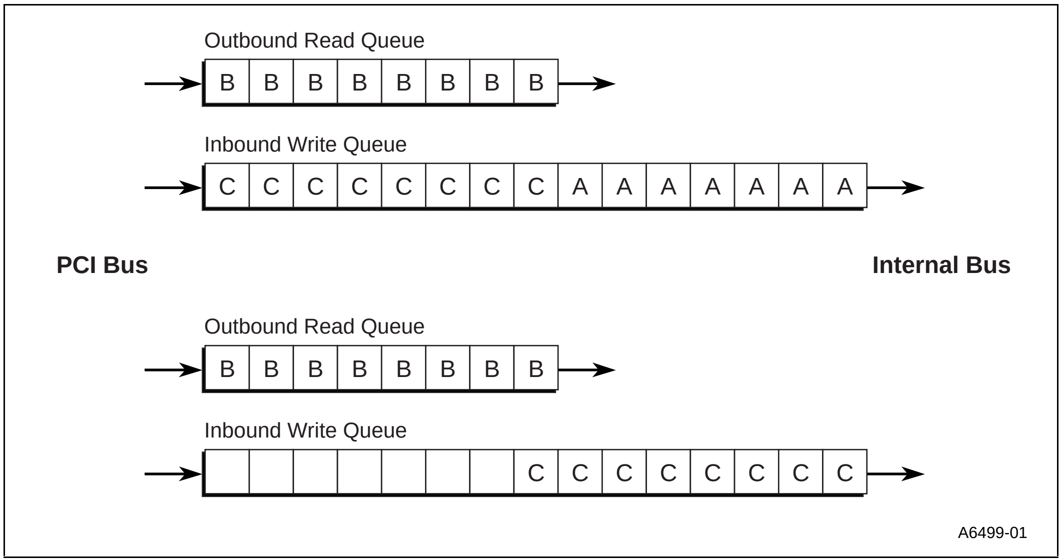
Definitions of the terms used in Table 13 and Table 14 are as follows. PCI terms are noted in parenthesis:

- Inbound Write (PMW) - Data from a write cycle initiated on PCI and targeted at the internal bus. Note that the address is in a separate transaction queue and is not referenced.
- Inbound Read Request (DRR-PCI mode and SRR-PCI-X mode) - address information from a read transactions retried or split on the PCI bus. Mastered on the internal bus to retrieve data for the Inbound Read Completion.
- Inbound Configuration Write Request - (DWR- PCI mode and SWR - PCI-X mode) - The address and data associated with a configuration write transaction from PCI and targeted at the ATU PCI configuration address space. Once completed on the internal bus, creates an Inbound Configuration Write Completion.
- Outbound Read Completion (SRC) - The data read on PCI in the process of being returned to the internal bus. This data is the completion cycle that results from an Outbound Read Request.
- Outbound Write (PMW) - The address and data from a write initiated on the internal bus and eventually completing on the PCI bus.
- Outbound Read Request (SRR) - The address/command of a split read cycle initiated on the internal bus. The read data is returned in the Outbound Read Completion cycle.
- Inbound Read Completion (DRC-PCI mode and SRC-PCI-X mode) - The data read on the internal bus in the process of being returned to the PCI bus. This data is the completion cycle for an Inbound Read Request.
- Inbound Configuration Write Completion (DWC-PCI mode and SWC-PCI-X mode) - The status of an inbound write configuration cycle traveling from the internal bus back towards the PCI bus.

These transaction ordering rules define the way idata moves in both directions through the ATU. In Table 13 and Table 14 a **NO** response in a box means that based on ordering rules, the current transaction (the row) can not pass the previous transaction (the column) under any circumstance. A **Yes** response in the box means that the current transaction is *allowed* to pass the previous transaction but is not required to, based on whether a consistent view of data or prevention of deadlocks is needed.

In the case of inbound write operations, multiple transactions may exist within the IWQ and the corresponding IWADQ at any point in time. The ordering of these transactions is based on a time stamp basis. Transactions entering the queue are stamped with a relative time in relation to all other transactions moving in a similar direction.

Example 1. Inbound Queue Completion



In Example 1 on page 89, the inbound write and outbound read queues of the ATU are shown. In this example, transaction A entered the write queue at **Time 0**. Next, the ATU entered read data into the outbound read queue at **Time 1** (Transaction B). Finally, before the previous transactions could be cleared, another inbound write, Transaction C, was entered into the IWQ. The ordering in Table 13 states that nothing can pass an inbound write and therefore Transaction A must complete on the internal bus before Transaction B since an outbound read completion can not pass an inbound write. Also, Transaction A must complete before Transaction C since an inbound write can not pass another inbound write. Once Transaction A completes, Transaction C moves to the head of the IWQ. The two transactions at the head of the queues moving data in an inbound direction are now Transaction C, an inbound write, and Transaction B, an outbound read completion. Ordering states that an inbound write may pass an outbound read completion. This means that the arbitration mechanism now takes over to decide which completes. Note that ordering enforced the completion of Transaction A but arbitration dictated the completion of Transactions B and C.

The first action performed to determine which transaction is allowed to proceed (either inbound or outbound) is to apply the rules of ordering as defined in Table 13 and Table 14. Any box marked **No** must be satisfied first. For example, when an inbound read request is in ITQ and it was latched *after* the data in the IDWQ arrived (this is a configuration write), then ordering states that an Inbound Read Request may not pass an Inbound Configuration Write Request. Therefore, the Inbound Configuration Write Request must be cleared out of IDWQ before the Inbound Read Request is attempted on the internal bus. Once transaction ordering is satisfied, the boxes marked **Yes** are now resolved.

2.6.3.1 Transaction Ordering Summary

Table 15 and Table 16, define transaction ordering in relation to token assignment of the priority mechanism (this is discussed in Section 2.6.3). These tables are read as follows:

1. As the transaction enters the head of the respective queue, the question in column 2 is asked.
2. When all the answers in column 3 for a given transaction type assigns a token to the transaction at the head of the queue, a token is assigned. Otherwise, no token is assigned signifying that transaction ordering must first be satisfied. Any transaction with a token may be initiated on the bus.

Table 15. Inbound Transaction Ordering Summary

Transaction at Head of Queue	Question	Answer	Action
Inbound Write in IWQ	Is there an Inbound Write Request with an earlier time stamp?	Yes	Do Not Assign Token Allow previous Transaction to Complete
		No	Assign Token
Inbound Read Request in ITQ	Is there an Inbound Write with an earlier time stamp?	Yes	Do Not Assign Token Allow previous Transaction to Complete
		No	Assign Token
	Is there an Inbound Read Request with an earlier time stamp?	Yes	Do Not Assign Token Allow previous Transaction to Complete
		No	Assign Token
	Is there an Inbound Configuration Write Request with an earlier time stamp?	Yes	Do Not Assign Token Allow previous Transaction to Complete
		No	Assign Token
Inbound Configuration Write Request in IDWQ	Is there an Inbound Write with an earlier time stamp?	Yes	Do Not Assign Token Allow previous Transaction to Complete
		No	Assign Token
	Is there an Inbound Read Request with an earlier time stamp?	Yes	Do Not Assign Token Allow previous Transaction to Complete
		No	Assign Token
Outbound Read Completion in ORQ	Is there an Inbound Write with an earlier time stamp?	Yes	Do Not Assign Token Allow previous Transaction to Complete
		No	Assign Token

Table 16. Outbound Transaction Ordering Summary

Transaction at Head of Queue	Question	Answer	Action
Outbound Write in OWQ	Is there an Outbound Write Request with an earlier time stamp?	Yes	Do Not Assign Token Allow previous Transaction to Complete
		No	Assign Token
Outbound Read Request in OTQ	Is there an Outbound Write with an earlier time stamp?	Yes	Do Not Assign Token Allow previous Transaction to Complete
		No	Assign Token
	Is there an Outbound Read Request with an earlier time stamp?	Yes	Do Not Assign Token Allow previous Transaction to Complete
		No	Assign Token
Inbound Configuration Write Completion in IRQ	Is there an Outbound Posted Write with an earlier time stamp?	Yes	Do Not Assign Token Allow previous Transaction to Complete
		No	Assign Token
	Is there an Inbound Read Completion with an earlier time stamp?	Yes	Do Not Assign Token and allow previous Transaction to Complete when in PCI-X Mode. Assign Token when in Conventional Mode
		No	Assign Token
Inbound Read Completion in IRQ	Is there an Outbound Posted Write with an earlier time stamp?	Yes	Do Not Assign Token Allow previous Transaction to Complete
		No	Assign Token
	Is there an Inbound Read Completion with an earlier time stamp?	Yes	Do Not Assign Token and allow previous Transaction to Complete when in PCI-X Mode. Assign Token when in Conventional Mode
		No	Assign Token
	Is there a Configuration Write Completion with an earlier time stamp?	Yes	Do Not Assign Token and allow previous Transaction to Complete when in PCI-X Mode. Assign Token when in Conventional Mode
		No	Assign Token

2.6.4 Byte Parity Checking and Generation

The internal bus interface of the ATU supports byte-wise parity protection on the internal bus. This includes A_PARITY[4:0] and D_PARITY[15:0] on the address bus (A[35:0]) and the data bus (D[127:0]) respectively.

For an outbound write request (or inbound read completion) the internal bus interface verifies the write request address parity and data parity on the data cycles.

For an inbound read completion, the internal bus interface verifies the read completion data parity on the data cycles.

For an inbound write request, the ATU computes and appends write request address parity and data parity data parity prior to delivering the request to the internal bus.

For an inbound read request, the internal bus interface computes and appends read request address parity prior to delivering the request to the internal bus.

Note: The ATU forwards data parity across to the other interface. When a data parity error occurs on the internal bus interface, the same data issues on the PCI bus with either bad parity or uncorrectable ECC error. For inbound transactions the bad parity or uncorrectable ECC errors results in the ATU marking that data bad by corrupting the parity on the internal bus.

2.6.4.1 Parity Generation

Data parity signals include byte enables in the calculation. Table 17 lists the data bits that are used for the parity calculation. The parity bits are calculated by bit XORing the data bits as shown in Table 17. As an example, the parity calculation for the lowest order byte of the data bus D[7:0] is calculated as follows:

Equation 5. D_PARITY0 = D[0] XOR D[1] XOR D[2] XOR D[3] XOR D[4] XOR D[5] XOR D[6] XOR D[7] XOR WBE[0]

Table 17. Parity Generation

Address/Data Parity Bit	Address/Data Bus	Address/Data Parity Bit	Address/Data Bus
A_PARITY4	A[35:32]	D_PARITY9	D[79:72], WBE[9]
A_PARITY3	A[31:24]	D_PARITY8	D[71:64], WBE[8]
A_PARITY2	A[23:16]	D_PARITY7	D[63:56], WBE[7]
A_PARITY1	A[15:8]	D_PARITY6	D[55:48], WBE[6]
A_PARITY0	A[7:0]	D_PARITY5	D[47:40], WBE[5]
D_PARITY15	D[127:120], WBE[15]	D_PARITY4	D[39:32], WBE[4]
D_PARITY14	D[119:112], WBE[14]	D_PARITY3	D[31:24], WBE[3]
D_PARITY13	D[111:104], WBE[13]	D_PARITY2	D[23:16], WBE[2]
D_PARITY12	D[103:96], WBE[12]	D_PARITY1	D[15:8], WBE[1]
D_PARITY11	D[95:88], WBE[11]	D_PARITY0	D[7:0], WBE[0]
D_PARITY10	D[87:80], WBE[10]		

2.6.4.2 Parity Checking

On an outbound request, address parity is checked on the address bus A[35:0]. The parity bits are checked by first bit XORing the address bits shown in Table 17 with the corresponding address parity bits, and then verifying when the result of each of the XORed operations is equal to zero. As an example, the parity calculation for the lowest order byte of the address bus A[7:0] is carried as follows:

Equation 6. PARITY_RESULT = A_PARITY0 XOR A[0] XOR A[1] XOR A[2] XOR A[3] XOR A[4] XOR A[5] XOR A[6] XOR A[7]

The parity logic uses the following algorithm. This algorithm logs the error when an error is detected.

```
check address parity
    if parity is good
        done
    else {error}
        create an error log
        Interrupt the core (if enabled)
```

On an outbound write request, data parity is checked on the data bus D[127:0]. The parity bits are checked by first bit XORing the data bits shown in Table 17 with the corresponding data parity bits, and then verifying when the result of each of the XORed operations is equal to zero. As an example, the parity calculation for the lowest order byte of the data bus D[7:0] is carried as follows:

Equation 7. PARITY_RESULT = D_PARITY0 XOR D[0] XOR D[1] XOR D[2] XOR D[3] XOR D[4] XOR D[5] XOR D[6] XOR D[7] XOR WBE[0]

A non-zero result from the above operation indicates a parity error.

The parity logic uses the following algorithm, and this algorithm logs the error when an error is detected.

```
check data parity
    if parity is good
        done
    else {error}
        create an error log
        Interrupt the core (if enabled)
```

2.6.4.3 Parity Disabled

When software disables parity, the ATU does generate the parity byte for read completions, but does not check the address and data byte parity.

2.7 ATU Error Conditions

PCI and internal bus error conditions cause ATU state machines to exit normal operation and return to idle states. In addition, status bits are set to inform error handling code of exact cause of error condition. Error conditions and status can be found in the ATUSR. The basic flow for a PCI error is as follows:

- Set the bit in the ATU Status Register which corresponds to the error condition (master abort, target abort, etc.)
- Set the bit in the ATU Interrupt Status Register which corresponds to the error condition (master abort, target abort, etc.). This function is maskable for all PCI error conditions.
- The setting of the bit in the ATU Interrupt Status Register results in an interrupt being driven to the Intel XScale® processor.

Error conditions on one side of the ATU are generally propagated to the other side of the ATU and have different effects depending on the error. Error conditions and their effects are described in the following sections.

PCI bus error conditions and the action taken on the bus are defined within the *PCI Local Bus Specification*, Revision 2.3, and the *PCI-X Protocol Addendum to the PCI Local Bus Specification,* Revision 2.0. The ATU adheres to the error conditions defined within the PCI specification for both requester and target operation. Error conditions on the internal bus are caused by an ECC error from the Memory Controller, (see Section 8.4, “ECC Interrupts/Error Conditions” on page 531 for details on memory controller error conditions), an Internal Bus Byte Parity Error, or by incorrect addressing resulting in an internal master abort. All actions on the PCI Bus for error situations are dependent on the error control bits found in the ATU Command Register (see Section 2.14.5, “ATU Command Register - ATUCMD” on page 148) for both Conventional and PCI-X modes. For PCI-X mode, the error response is also dependent on an error control bit in the PCI-X Command Register (see Section 2.14.55, “PCI-X Command Register - PCIXCMD” on page 191). In addition, for PCI-X Mode 2 only, the error response also depends on the ECC Control and Status Register (see Section 2.14.57, “ECC Control and Status Register - ECCCSR” on page 195).

The 4138xx operates in parity mode (when enabled) for conventional PCI (PCI-33, PCI-66), and PCI-X Mode 1 (PCI-X 66, PCI-X 133). For PCI-X Mode 2, the 4138xx functions in ECC mode. Parity errors, single-bit ECC errors (when correction is disabled), and multi-bit ECC errors are uncorrectable errors. In ECC mode, all single-bit errors are corrected when error correction is enabled and the transaction completes.

The following sections detail all ATU error conditions on the PCI and 4138xx internal bus, action taken on these conditions, and status and control bits associated with error handling.

2.7.1 Uncorrectable Address and Uncorrectable Attribute Errors on the PCI Interface

The ATUs must detect and report uncorrectable address and attribute (PCI-X mode only) errors for transactions on the PCI bus. When an uncorrectable address or attribute error occurs on the PCI interface of the ATU, the 4138xx performs the following actions based on the constraints specified:

- In Conventional mode, when the Parity Error Response bit in ATUCMD is set, the ATU ignores (Master-Abort) the transaction by not asserting **DEVSEL#**. When clear, the transaction proceeds normally.
- In PCI-X mode, when the Parity Error Response bit in ATUCMD is set, the ATU completes the transaction on the PCI bus as when no error had occurred, but the request or completion is not forwarded to the internal bus. When clear, the transaction proceeds normally.
- Assert **SERR#** when the **SERR#** Enable bit and the Parity Error Response bit in the ATUCMD are set. When the ATU asserts **SERR#,** additional actions is taken:
  - Set the **SERR#** Asserted bit in the ATUSR
  - When the ATU **SERR#** Asserted Interrupt Mask Bit in the ATUIMR is clear, set the **SERR#** Asserted bit in the ATUISR. When set, no action.
  - When the ATU **SERR#** Detected Interrupt Enable Bit in the ATUCR (see Section 2.14.40, "ATU Configuration Register - ATUCR" on page 177) is set, set the **SERR#** Detected bit in the ATUISR. When clear, no action.
- Set the Detected Uncorrectable Address or Attribute Error bit in PCSR (PCI Configuration and Status Register).
- Set the Detected Parity Error bit in the ATUSR. When the ATU sets the Detected Parity Error bit, additional actions is taken:
  - When the ATU Detected Parity Error Interrupt Mask bit in the ATUIMR is clear, set the Detected Parity Error bit in the ATUISR. When set, no action.
- For PCI-X Mode 2, update the "ECC Control and Status Register - ECCCSR" on page 195, the "ECC First Address Register - ECCFAR" on page 198, the "ECC Second Address Register - ECCSAR" on page 199, and the "ECC Attribute Register - ECCAR" on page 200 for the transaction

Note: The Detected Parity Error bit with its' associated interrupt along with the Detected Uncorrectable Address or Attribute Error bit provides software with the ability to distinguish between an Uncorrectable Address or Attribute error versus an Uncorrectable Data Error during a Detected Parity error interrupt.

2.7.2 Correctable Address and Correctable Attribute Errors on the PCI Interface

In PCI-X Mode 2 (when single-bit correction is enabled), the ATUs must detect and report correctable address and attribute (PCI-X mode only) errors for transactions on the PCI bus. When a correctable address or attribute error occurs on the PCI interface of the 4138xx, the ATU performs the following actions based on the constraints specified:

- The error is corrected and the ATU completes the transaction on the PCI bus as when no error had occurred. Then, the transaction is forwarded to the internal bus normally.
- Update the "ECC Control and Status Register - ECCCSR" on page 195, the "ECC First Address Register - ECCFAR" on page 198, the "ECC Second Address Register - ECCSAR" on page 199, and the "ECC Attribute Register - ECCAR" on page 200 for the transaction.
  - When the ATU Detected Correctable Error Interrupt Mask bit in the ATUIMR is clear, set the Detected Correctable Error bit in the ATUISR. When set, no action.

Note: The ECCCSR provides information on the transaction phase in which the correctable error occurred.

2.7.3 Uncorrectable Data Errors on the PCI Interface

Two kinds of uncorrectable data errors can occur on the PCI interface: errors as an initiator and errors as a target.

Errors encountered as an initiator:

— Outbound Read Request

— Outbound Write Request

— Inbound Read Completions

— Inbound Configuration Write Completion Messages

As an initiator, the ATU provides an error response for uncorrectable data errors on outbound reads, and uncorrectable data errors occurring at the target for outbound writes. However, there is no error response for uncorrectable data errors on inbound configuration write completion messages and inbound read completions.

Errors encountered as a target:

— Inbound Read Request (Immediate Data Transfer)

— Inbound Write Request

— Outbound Read Completions

— Outbound Split (I/O or Configuration) Uncorrectable Write Data Error Messages

— Inbound Configuration Write

— Split Completion Messages

As a target, the ATU provides an error response for uncorrectable data errors on inbound writes, outbound read completions, outbound uncorrectable split write data error messages, inbound configuration writes, and split completion messages. However, there is no error response for uncorrectable data errors on inbound reads.

2.7.3.1 Outbound Read Request Uncorrectable Data Errors

2.7.3.1.1 Immediate Data Transfer

As an initiator, the ATU may encounter this error condition in Conventional or PCI-X mode when the target transfers data immediately rather than signalling a Retry[7] (Conventional Delayed Read Request) or a Split Response Termination (PCI-X Split Read Request).

Uncorrectable data errors occurring during read operations initiated by the ATU are recorded, **PERR#** is asserted (when enabled) and **SERR#** is asserted (when enabled). Specifically, the following actions with the given constraints are taken by the ATU:

- **PERR#** is asserted two clocks cycles (three clock cycles when operating in the PCI-X mode) following the data phase in which the uncorrectable data error is detected on the bus. This is only done when the Parity Error Response bit in the ATUCMD is set. When the ATU asserts **PERR#**, additional actions are taken:
  - — The Master Parity Error bit in the ATUSR is set.
  - — When the ATU PCI Master Parity Error Interrupt Mask Bit in the ATUIMR is clear, set the PCI Master Parity Error bit in the ATUISR. When set, no action.
  - — When the ATU is operating in the PCI-X mode, the **SERR#** Enable bit in the ATUCMD is set, and the Uncorrectable Data Error Recover Enable bit in the PCIXCMD register is clear, assert **SERR#**, otherwise no action. When the ATU asserts **SERR#,** additional actions are taken:

    Set the **SERR#** Asserted bit in the ATUSR.

    When the ATU **SERR#** Asserted Interrupt Mask Bit in the ATUIMR is clear, set the **SERR#** Asserted bit in the ATUISR. When set, no action.

    When the ATU **SERR#** Detected Interrupt Enable Bit in the ATUCR is set, set the **SERR#** Detected bit in the ATUISR. When clear, no action.
  - — The read completion are aborted on the internal bus of the 4138xx.
- The Detected Parity Error bit in the ATUSR is set. When the ATU sets this bit, additional actions are taken:
  - — When the ATU Detected Parity Error Interrupt Mask bit in the ATUIMR is clear, set the Detected Parity Error bit in the ATUISR. When set, no action.
- For PCI-X Mode 2, update the "ECC Control and Status Register - ECCCSR" on page 195, the "ECC First Address Register - ECCFAR" on page 198, the "ECC Second Address Register - ECCSAR" on page 199, and the "ECC Attribute Register - ECCAR" on page 200 for the transaction.

7. Retry terminations may also be signaled in PCI-X mode when the target is too busy to handle the current request. However, this is not the same as a Delayed Read Request in Conventional PCI mode since the requester is not required or expected by the target to return with the same read request.

2.7.3.1.2 Split Response Termination

As an initiator, the ATU may encounter this error condition in PCI-X mode when the target signals a Split Response Termination.

Parity errors occurring during Split Response Terminations of Read Requests by the ATU are recorded, **PERR#** is asserted (when enabled) and **SERR#** is asserted (when enabled). Specifically, the following actions with the given constraints are taken by the ATU:

- **PERR#** is asserted two clocks cycles (three clock cycles when operating in the PCI-X mode) following the Split Response Termination in which the parity error is detected on the bus. This is only done when the Parity Error Response bit in the ATUCMD is set. When the ATU asserts **PERR#**, additional actions are taken:
  - — The Master Parity Error bit in the ATUSR is set.
  - — When the ATU PCI Master Parity Error Interrupt Mask Bit in the ATUIMR is clear, set the PCI Master Parity Error bit in the ATUISR. When set, no action.
  - — When the **SERR#** Enable bit in the ATUCMD is set, and the Uncorrectable Data Error Recover Enable bit in the PCIXCMD register is clear, assert **SERR#**, otherwise no action. When the ATU asserts **SERR#,** additional actions are taken:

    Set the **SERR#** Asserted bit in the ATUSR.

    When the ATU **SERR#** Asserted Interrupt Mask Bit in the ATUIMR is clear, set the **SERR#** Asserted bit in the ATUISR. When set, no action.

    When the ATU **SERR#** Detected Interrupt Enable Bit in the ATUCR is set, set the **SERR#** Detected bit in the ATUISR. When clear, no action
- The Detected Parity Error bit in the ATUSR is set. When the ATU sets this bit, additional actions are taken:
  - — When the ATU Detected Parity Error Interrupt Mask bit in the ATUIMR is clear, set the Detected Parity Error bit in the ATUISR. When set, no action.
- For PCI-X Mode 2, update the "ECC Control and Status Register - ECCCSR" on page 195, the "ECC First Address Register - ECCFAR" on page 198, the "ECC Second Address Register - ECCSAR" on page 199, and the "ECC Attribute Register - ECCAR" on page 200 for the transaction.

The Outbound Read Request remains enqueued in the ATU since the completer is initiating completion transactions that are associated with this request.

2.7.3.2 Outbound Write Request Uncorrectable Data Errors

2.7.3.2.1 Outbound Writes that are not MSI (Message Signaled Interrupts)

As an initiator, the ATU may encounter this error condition when operating in either the Conventional or PCI-X modes.

Uncorrectable Data Errors occurring during write operations initiated by the ATU may record the assertion of **PERR#** from the target on the PCI Bus. In PCI-X mode, this includes the assertion of **PERR#** from the target on the PCI Bus following the split response termination of a non-posted write request. When an error occurs, the ATUs continue writing data to the target to clear the OWQ of the current outbound write transaction. Specifically, the following actions with the given constraints are taken by the ATU:

- When **PERR#** is sampled active and the Parity Error Response bit in the ATUCMD is set, set the Master Parity Error bit in the ATUSR. When the Parity Error Response bit in the ATUCMD is clear, no action is taken. When the Master Parity Error bit in the ATUSR is set, additional actions are taken:
  - When the ATU PCI Master Parity Error Interrupt Mask Bit in the ATUIMR is clear, set the PCI Master Parity Error bit in the ATUISR. When set, no action.
  - When the ATU is operating in the PCI-X mode, the **SERR#** Enable bit in the ATUCMD is set, and the Uncorrectable Data Error Recover Enable bit in the PCIXCMD register is clear, assert **SERR#**, otherwise no action. When the ATU asserts **SERR#,** additional actions are taken:

    Set the **SERR#** Asserted bit in the ATUSR

    When the ATU **SERR#** Asserted Interrupt Mask Bit in the ATUIMR is clear, set the **SERR#** Asserted bit in the ATUISR. When set, no action.

    When the ATU **SERR#** Detected Interrupt Enable Bit in the ATUCR is set, set the **SERR#** Detected bit in the ATUISR. When clear, no action

Outbound uncorrectable write data errors, do not result in a master completion. In addition, when the target terminates the transaction (disconnect), the ATU master must reinitiate the transaction to clear the data from the OWQ.

2.7.3.2.2 MSI Outbound Writes

As an initiator, the ATU may encounter this error condition when operating in either the Conventional or PCI-X modes.

Uncorrectable Data Errors occurring during MSI write operations initiated by the ATU may record the assertion of **PERR#** from the target on the PCI Bus. When an error occurs, the ATU completes the transaction normally. Then, the following actions with the given constraints are taken by the ATU:

- When **PERR#** is sampled active and the Parity Error Response bit in the ATUCMD is set, set the Master Parity Error bit in the ATUSR. When the Parity Error Response bit in the ATUCMD is clear, no action is taken. When the Master Parity Error bit in the ATUSR is set, additional actions are taken:
  - When the ATU PCI Master Parity Error Interrupt Mask Bit in the ATUIMR is clear, set the PCI Master Parity Error bit in the ATUISR. When set, no action.
  - When the **SERR#** Enable bit in the ATUCMD is set, assert **SERR#**, otherwise no action. When the ATU asserts **SERR#,** additional actions are taken:

    Set the **SERR#** Asserted bit in the ATUSR.

    When the ATU **SERR#** Asserted Interrupt Mask Bit in the ATUIMR is clear, set the **SERR#** Asserted bit in the ATUISR. When set, no action.

    When the ATU **SERR#** Detected Interrupt Enable Bit in the ATUCR is set, set the **SERR#** Detected bit in the ATUISR. When clear, no action.

2.7.3.3 Inbound Read Completions Uncorrectable Data Errors

As an initiator, ATU may encounter this error condition when operating in PCI-X mode.

When as the completer of a Split Read Request the ATU observes **PERR#** assertion during the split completion transaction**,** the ATU attempts to complete the transaction normally and no further action are taken.

2.7.3.4 Inbound Configuration Write Completion Message Uncorrectable Data Errors

As an initiator, ATU may encounter this error condition when operating in PCI-X mode.

When as the completer of a Configuration (Split) Write Request the ATU observes **PERR#** assertion during the split completion transaction**,** the ATU attempts to complete the transaction normally and no further action are taken.

2.7.3.5 Inbound Read Request Uncorrectable Data Errors

2.7.3.5.1 Immediate Data Transfer

As a target, the ATU may encounter this error when operating in the Conventional PCI or PCI-X modes.

Inbound read uncorrectable data errors occur when read data delivered from the IRQ is detected as having bad parity by the initiator of the transaction who is receiving the data. The initiator may optionally report the error to the system by asserting **PERR#**. As a target device in this scenario, no action is required and no error bits are set.

2.7.3.5.2 Split Response Termination

As a target, the ATU may encounter this error when operating in the PCI-X mode.

Inbound read uncorrectable data errors occur during the Split Response Termination. The initiator may optionally report the error to the system by asserting **PERR#**. As a target device in this scenario, no action is required and no error bits are set.

2.7.3.6 Inbound Write Request Uncorrectable Data Errors

As a target, ATU may encounter this error when operating in Conventional or PCI-X modes.

Uncorrectable Data errors occurring during write operations received by the ATU may assert **PERR#** on the PCI Bus. When an error occurs, the ATU continues accepting data until the initiator of the write transaction completes or a queue fill condition is reached. Specifically, the following actions with the given constraints are taken by the ATU:

- **PERR#** is asserted two clocks cycles (three clock cycles when operating in PCI-X mode) following the data phase in which uncorrectable data error is detected on the bus. This is only done when the Parity Error Response bit in ATUCMD is set.
- The Detected Parity Error bit in the ATUSR is set. When the ATU sets this bit, additional actions are taken:
  - When the ATU Detected Parity Error Interrupt Mask bit in the ATUIMR is clear, set the Detected Parity Error bit in the ATUISR. When set, no action.
- For PCI-X Mode 2, update the "ECC Control and Status Register - ECCCSR" on page 195, the "ECC First Address Register - ECCFAR" on page 198, the "ECC Second Address Register - ECCSAR" on page 199, and the "ECC Attribute Register - ECCAR" on page 200 for the transaction.

2.7.3.7 Outbound Read Completion Uncorrectable Data Errors

As a target, the ATU may encounter this error when operating in the PCI-X mode.

Uncorrectable Data errors occurring during read completion transactions that are claimed by the ATU are recorded, **PERR#** is asserted (when enabled) and **SERR#** is asserted (when enabled). Specifically, the following actions with the given constraints are taken by the ATU:

- **PERR#** is asserted two clocks cycles (three clock cycles when operating in the PCI-X mode) following the data phase in which the uncorrectable data error is detected on the bus. This is only done when the Parity Error Response bit in the ATUCMD is set. When the ATU asserts **PERR#**, additional actions are taken:
  - The Master Parity Error bit in the ATUSR is set.
  - When the ATU PCI Master Parity Error Interrupt Mask Bit in the ATUIMR is clear, set the PCI Master Parity Error bit in the ATUISR. When set, no action.
  - When the **SERR#** Enable bit in the ATUCMD is set, and the Uncorrectable Data Error Recover Enable bit in the PCIXCMD register is clear, assert **SERR#**, otherwise no action. When the ATU asserts **SERR#,** additional actions are taken:

    Set the **SERR#** Asserted bit in the ATUSR.

    When the ATU **SERR#** Asserted Interrupt Mask Bit in the ATUIMR is clear, set the **SERR#** Asserted bit in the ATUISR. When set, no action.

    When the ATU **SERR#** Detected Interrupt Enable Bit in the ATUCR is set, set the **SERR#** Detected bit in the ATUISR. When clear, no action.
  - The read completion is aborted on the internal bus of the 4138xx.
- The Detected Parity Error bit in the ATUSR is set. When the ATU sets this bit, additional actions are taken:
  - When the ATU Detected Parity Error Interrupt Mask bit in the ATUIMR is clear, set the Detected Parity Error bit in the ATUISR. When set, no action.
- For PCI-X Mode 2, update the “ECC Control and Status Register - ECCCSR” on page 195, the “ECC First Address Register - ECCFAR” on page 198, the “ECC Second Address Register - ECCSAR” on page 199, and the “ECC Attribute Register - ECCAR” on page 200 for the transaction.

2.7.3.8 Outbound Split Write Uncorrectable Data Error Message

The ATU claims a Split Completion Error Message that indicates an uncorrectable data error has occurred on one of the ATUs non-posted (I/O or Configuration) write requests (Message Class = 2h, Message Index = 01h -- Uncorrectable Split Write Data Error or Message Class = 1h, Message Index = 02h --Uncorrectable Write Data Error).

When the ATU receives a Split Completion Error Message indicating an uncorrectable data error for an outstanding Configuration or I/O (split) write request, the error is recorded, and **SERR#** is asserted (when enabled). Specifically, the following actions with the given constraints are taken by the ATU:

- When the Parity Error Response bit in the ATUCMD is set, these actions are taken:
  - The Master Parity Error bit in the ATUSR is set.
  - When the ATU PCI Master Parity Error Interrupt Mask Bit in the ATUIMR is clear, set the PCI Master Parity Error bit in the ATUISR. When set, no action.
  - When the **SERR#** Enable bit in the ATUCMD is set, and the Uncorrectable Data Error Recover Enable bit in the PCIXCMD register is clear, assert **SERR#**, otherwise no action. When the ATU asserts **SERR#,** additional actions are taken:

    Set the **SERR#** Asserted bit in the ATUSR.

    When the ATU **SERR#** Asserted Interrupt Mask Bit in the ATUIMR is clear, set the **SERR#** Asserted bit in the ATUISR. When set, no action.

    When the ATU **SERR#** Detected Interrupt Enable Bit in the ATUCR is set, set the **SERR#** Detected bit in the ATUISR. When clear, no action.
- The Received Split Completion Error Message bit in the PCIXSR is set (based on bit 30 of the completer attributes being set). When the ATU sets this bit, additional actions are taken:
  - When the ATU Received Split Completion Error Message Interrupt Mask bit in the ATUIMR is clear, set the Received Split Completion Error Message bit in the ATUISR. When set, no action.
- The transaction associated with the Split Completion Error Message is discarded.

2.7.3.9 Inbound Configuration Write Request

As a target, the ATU may encounter this error when operating in the Conventional or PCI-X modes.

2.7.3.9.1 Conventional PCI Mode

To allow for correct data parity calculations for delayed write transactions, the ATU delays the assertion of **STOP#** (signalling a Retry) until **PAR** is driven by the master. A parity error during a delayed write transaction (inbound configuration write cycle) can occur in any of the following parts of the transactions:

- During the initial Delayed Write Request cycle on the PCI bus when the ATU latches the address/command and data for delayed delivery to the internal configuration register.
- During the Delayed Write Completion cycle on the PCI bus when the ATU delivers the status of the operation back to the original master.

The 4138xx ATU PCI interface has the following responses to a delayed write parity error for inbound transactions during Delayed Write Request cycles with the given constraints:

- When the Parity Error Response bit in the ATUCMD is set, the ATU asserts **TRDY#** (disconnects with data) and two clock cycles later asserts **PERR#** notifying the initiator of the parity error. The delayed write cycle is not enqueued and forwarded to the internal bus.
- When the Parity Error Response bit in the ATUCMD is cleared, the ATU retries the transaction by asserting **STOP#** and enqueues the Delayed Write Request cycle to be forwarded to the internal bus. **PERR#** is not asserted.
- The Detected Parity Error bit in the ATUSR is set. When the ATU sets this bit, additional actions are taken:
  - — When the ATU Detected Parity Error Interrupt Mask bit in the ATUIMR is clear, set the Detected Parity Error bit in the ATUISR. When set, no action.
- For PCI-X Mode 2, update the "ECC Control and Status Register - ECCCSR" on page 195, the "ECC First Address Register - ECCFAR" on page 198, the "ECC Second Address Register - ECCSAR" on page 199, and the "ECC Attribute Register - ECCAR" on page 200 for the transaction.

For the original write transaction to be completed, the initiator retries the transaction on the PCI bus and the ATU returns the status from the internal bus, completing the transaction.

For the Delayed Write Completion transaction on the PCI bus where a data parity error occurs and therefore does not agree with the status being returned from the internal bus (i.e. status being returned is normal completion) the ATU performs the following actions with the given constraints:

- When the Parity Error Response Bit is set in the ATUCMD, the ATU asserts **TRDY#** (disconnects with data) and two clocks later asserts **PERR#**. The Delayed Completion cycle in the IDWQ remains since the data of retried command did not match the data within the queue.
- The Detected Parity Error bit in the ATUSR is set. When the ATU sets this bit, additional actions are taken:
  - — When the ATU Detected Parity Error Interrupt Mask bit in the ATUIMR is clear, set the Detected Parity Error bit in the ATUISR. When set, no action.

2.7.3.9.2 PCI-X Mode

Uncorrectable Data errors occurring during configuration write operations received by the ATU may cause **PERR#** assertion and delivery of a Split Completion Error Message on the PCI Bus. When an error occurs, the ATU accepts the write data and complete with a Split Response Termination. Specifically, the following actions with the given constraints are then taken by the ATU:

- When the Parity Error Response bit in the ATUCMD is set, **PERR#** is asserted three clocks cycles following the Split Response Termination in which the uncorrectable data error is detected on the bus. When the ATU asserts **PERR#**, additional actions are taken:
  - An Uncorrectable Split Write Data Error message (with message class=2h - completer error and message index=01h - Uncorrectable Split Write Data Error) is initiated by the ATU on the PCI bus that addresses the requester of the configuration write.
  - When the Initiated Split Completion Error Message Interrupt Mask in the ATUIMR is clear, set the Initiated Split Completion Error Message bit in the ATUISR. When set, no action.
  - The Split Write Request is not enqueued and forwarded to the internal bus.
- The Detected Parity Error bit in the ATUSR is set. When the ATU sets this bit, additional actions are taken:
  - When the ATU Detected Parity Error Interrupt Mask bit in the ATUIMR is clear, set the Detected Parity Error bit in the ATUISR. When set, no action.
- For PCI-X Mode 2, update the "ECC Control and Status Register - ECCCSR" on page 195, the "ECC First Address Register - ECCFAR" on page 198, the "ECC Second Address Register - ECCSAR" on page 199, and the "ECC Attribute Register - ECCAR" on page 200 for the transaction.

2.7.3.10 Split Completion Messages

As a target, the ATU may encounter this error when operating in the PCI-X mode.

Uncorrectable Data errors occurring during Split Completion Messages claimed by the ATU may assert **PERR#** (when enabled) or **SERR# (when** enabled) on the PCI Bus. When an error occurs, the ATU accepts the data and complete normally. Specifically, the following actions with the given constraints are taken by the ATU:

- **PERR#** is asserted three clocks cycles following the data phase in which the uncorrectable data error is detected on the bus. This is only done when the Parity Error Response bit in the ATUCMD is set. When the ATU asserts **PERR#**, additional actions are taken:
  - — The Master Parity Error bit in the ATUSR is set.
  - — When the ATU PCI Master Parity Error Interrupt Mask Bit in the ATUIMR is clear, set the PCI Master Parity Error bit in the ATUISR. When set, no action.
  - — When the **SERR#** Enable bit in the ATUCMD is set, and the Uncorrectable Data Error Recover Enable bit in the PCIXCMD register is clear, assert **SERR#**; otherwise no action is taken. When the ATU asserts **SERR#,** additional actions are taken:

    Set the **SERR#** Asserted bit in the ATUSR.

    When the ATU **SERR#** Asserted Interrupt Mask Bit in the ATUIMR is clear, set the **SERR#** Asserted bit in the ATUISR. When set, no action.

    When the ATU **SERR#** Detected Interrupt Enable Bit in the ATUCR is set, set the **SERR#** Detected bit in the ATUISR. When clear, no action.
- When the SCE bit (Split Completion Error -- bit 30 of the Completer Attributes) is set during the Attribute phase, the Received Split Completion Error Message bit in the PCIXSR is set. When the ATU sets this bit, additional actions are taken:
  - — When the ATU Received Split Completion Error Message Interrupt Mask bit in the ATUIMR is clear, set the Received Split Completion Error Message bit in the ATUISR. When set, no action.
- The Detected Parity Error bit in the ATUSR is set. When the ATU sets this bit, additional actions are taken:
  - — When the ATU Detected Parity Error Interrupt Mask bit in the ATUIMR is clear, set the Detected Parity Error bit in the ATUISR. When set, no action.
- For PCI-X Mode 2, update the “ECC Control and Status Register - ECCCSR” on page 195, the “ECC First Address Register - ECCFAR” on page 198, the “ECC Second Address Register - ECCSAR” on page 199, and the “ECC Attribute Register - ECCAR” on page 200 for the transaction.
- The transaction associated with the Split Completion Message is discarded.
- When the discarded transaction was a read, the read completion is aborted on the internal bus of the 4138xx.

2.7.4 Correctable Data Errors on the PCI Interface

When the 4138xx PCI interface is operating in Mode 2 and Single-Bit Correction is enabled, correctable data errors may occur on the PCI bus.

Two kinds of correctable data errors can occur on the PCI interface: errors as an initiator and errors as a target.

Errors encountered as an initiator:

— Outbound Read Request

— Outbound Write Request

— Inbound Read Completions

— Inbound Configuration Write Completion Messages

As an initiator, the ATU provides no error response for correctable data errors.

Errors encountered as a target:

— Inbound Read Request (Immediate Data Transfer)

— Inbound Write Request

— Outbound Read Completions

— Inbound Configuration Write

— Split Completion Messages

As a target, the ATU provides an error response for correctable data errors on inbound writes, outbound read completions, inbound configuration writes, and split completion messages. However, the ATU provides no error response for correctable data errors on inbound read requests.

In general, when the ATU is receiving data from the PCI bus as a target, any correctable data errors are corrected and logged. Otherwise, the ATU functions as when no error had occurred.

2.7.4.1 Inbound Read Request Correctable Data Errors

2.7.4.1.1 Immediate Data Transfer

As a target device in this scenario, no action is required and no error bits are set.

2.7.4.1.2 Split Response Termination

As a target device in this scenario, no action is required and no error bits are set.

2.7.4.2 Inbound Write Request Correctable Data Errors

As a target device, when an inbound write request correctable data error is detected, the following actions are taken:

- Error is corrected and ATU completes the transaction on the PCI bus as when no error had occurred. Then, the transaction is normally forwarded to the internal bus.
- Update "ECC Control and Status Register - ECCCSR" on page 195, "ECC First Address Register - ECCFAR" on page 198, "ECC Second Address Register - ECCSAR" on page 199, and "ECC Attribute Register - ECCAR" on page 200 for transaction.
    — When ATU Detected Correctable Error Interrupt Mask bit in the ATUIMR is clear, set the Detected Correctable Error bit in the ATUISR. When set, no action.

2.7.4.3 Outbound Read Completion Correctable Data Errors

As a target device, when an outbound read completion correctable data error is detected, the following actions are taken:

- The error is corrected and the ATU completes the transaction on the PCI bus as when no error had occurred. Then, the transaction is forwarded to the internal bus normally.
- Update the "ECC Control and Status Register - ECCCSR" on page 195, the "ECC First Address Register - ECCFAR" on page 198, the "ECC Second Address Register - ECCSAR" on page 199, and the "ECC Attribute Register - ECCAR" on page 200 for the transaction.
  — When the ATU Detected Correctable Error Interrupt Mask bit in the ATUIMR is clear, set the Detected Correctable Error bit in the ATUISR. When set, no action.

2.7.4.4 Inbound Configuration Write Request

As a target device, when an inbound configuration write request correctable data error is detected, the following actions are taken:

- The error is corrected and the ATU completes the transaction on the PCI bus as when no error had occurred. Then, the transaction is forwarded to the internal bus normally.
- Update the "ECC Control and Status Register - ECCCSR" on page 195, the "ECC First Address Register - ECCFAR" on page 198, the "ECC Second Address Register - ECCSAR" on page 199, and the "ECC Attribute Register - ECCAR" on page 200 for the transaction.
  — When the ATU Detected Correctable Error Interrupt Mask bit in the ATUIMR is clear, set the Detected Correctable Error bit in the ATUISR. When set, no action.

2.7.4.5 Split Completion Messages

As a target device, when a split completion message correctable data error is detected, the following actions are taken:

- The error is corrected and the ATU completes the transaction on the PCI bus as when no error had occurred. Then, the transaction is forwarded to the internal bus normally.
- Update the "ECC Control and Status Register - ECCCSR" on page 195, the "ECC First Address Register - ECCFAR" on page 198, the "ECC Second Address Register - ECCSAR" on page 199, and the "ECC Attribute Register - ECCAR" on page 200 for the transaction.
  — When the ATU Detected Correctable Error Interrupt Mask bit in the ATUIMR is clear, set the Detected Correctable Error bit in the ATUISR. When set, no action.

2.7.5 Master Aborts on the PCI Interface

As an initiator on the PCI bus, the ATU can encounter master abort conditions during:

- Outbound Read Request
- Outbound Write Request
- Inbound Read Completion
- Inbound Configuration Write Completion Message

As a target, the ATU PCI interface is capable of signaling a master abort case during:

- Uncorrectable Address Error (Conventional Mode)
- Inbound Read Request (PCI-X Mode)

2.7.5.1 Master Aborts for Outbound Read or Write Request

This error may be encountered in both the Conventional and the PCI-X modes. For an Outbound transaction, there are two ways in which a Master-Abort may be signaled to the ATU:

1. In the Conventional or PCI-X modes, a master abort is signaled when the target of the transaction does not assert **DEVSEL#** within 5 clocks (7 clocks when operating in the PCI-X mode) of the assertion of **FRAME#**.
2. In PCI-X mode, ATU may enqueue a Split request (Read or Write) on target-side interface of a PCI-to-PCI Bridge. When PCI-to-PCI Bridge detects a Master Abort on requester-side interface for that Split Request, master abort is signaled to ATU through a Master-Abort Split Completion Error Message (class=1h - bridge error and index=00h - Master Abort). The following actions with given constraints are performed by ATU when a master abort is detected by the PCI initiator interface or the PCI target interface receives a Master-Abort Split Completion error message:

- Set the Master Abort bit (bit 13) in the ATUSR.
- When the ATU PCI Master Abort Interrupt Mask bit in the ATUIMR is clear, set the PCI Master Abort bit in the ATUISR. When set, no action.
- When an outbound write or inbound completion, flush data and address.
- When the transaction is an MSI outbound write and the **SERR#** Enable bit in the ATUCMD is set, assert **SERR#**, otherwise no action. When the ATU asserts **SERR#,** additional actions are taken:
  - Set the **SERR#** Asserted bit in the ATUSR
  - When the ATU **SERR#** Asserted Interrupt Mask Bit in the ATUIMR is clear, set the **SERR#** Asserted bit in the ATUISR. When set, no action.
  - When the ATU **SERR#** Detected Interrupt Enable Bit in the ATUCR is set, set the **SERR#** Detected bit in the ATUISR. When clear, no action
- When operating in PCI-X mode and Master-Abort is signaled via a Split Completion Error Message, the Received Split Completion Error Message bit in PCIXSR is set. When ATU sets this bit, additional actions are taken:
  - When the ATU Received Split Completion Error Message Interrupt Mask bit in the ATUIMR is clear, set the Received Split Completion Error Message bit in the ATUISR. When set, no action.
- For an Outbound Read request, generate a split completion error message (class=1h - 4138xx Outbound Request error and index=00h - master abort) on the internal bus.
- Flush the address from the OTQ.

2.7.5.2 Inbound Read Completion or Inbound Configuration Write Completion Message

The ATU encounters this error only in the PCI-X mode.

A master abort is signaled when the target of the transaction does not assert **DEVSEL#** within 7 clocks of the assertion of **FRAME#**.

When the ATU is signaled a Master-Abort while initiating either a Split Read Completion Transaction or a Split Write Completion Message, the ATU discards the Split Completion and take no further action.

2.7.5.3 Master-Aborts Signaled by the ATU as a Target

2.7.5.3.1 Uncorrectable Address Errors

The ATU can only signal this error during an Uncorrectable Address Error in the Conventional mode.

Please see Section 2.7.1, "Uncorrectable Address and Uncorrectable Attribute Errors on the PCI Interface" on page 95 for details on the ATU response to an Uncorrectable Address Error in the Conventional mode.

2.7.5.3.2 Internal Bus Master-Abort

The ATU can only signal this error during an internal bus master abort in the PCI-X mode.

Please see Section 2.7.9.1, "Master Abort on the Internal Bus" on page 115 for details on the ATU response to an Internal Bus Master Abort in the PCI-X mode.

2.7.6 Target Aborts on the PCI Interface

As an initiator on the PCI bus, the ATU can encounter target abort conditions during:

- Outbound Read Request
- Outbound Write Request
- Inbound Read Completion
- Inbound Configuration Write Completion Message

As a target, the ATU PCI interface is capable of signaling a target abort case during:

- Inbound Read Request (PCI-X and Conventional Modes)
- Inbound Write Request to EROM memory space (PCI-X and Conventional Modes)

2.7.6.1 Target Aborts for Outbound Read Request or Outbound Write Request

This error can be encountered by the ATU in both the Conventional and PCI-X modes. For an Outbound transaction, there are two ways in which a Target-Abort may be signaled to the ATU:

1. In the Conventional or PCI-X modes, a target abort is signaled when the target of the transaction simultaneously deasserts **DEVSEL#**, deasserts **TRDY#,** and asserts **STOP#**.
2. In PCI-X mode, ATU may enqueue a Split request (Read or Write) on target-side interface of a PCI-to-PCI Bridge. When PCI-to-PCI Bridge detects a Target Abort on requester-side interface for that Split Request, target abort is signaled to ATU through a Target-Abort Split Completion Error Message (class=1h - bridge error and index=01h - Target Abort). The following actions with the given constraints are performed by the ATU when a target abort is detected by the PCI initiator interface or the PCI target interface receives a Target-Abort Split Completion error message:

- Set the Target Abort (master) bit (bit 12) in the ATUSR.
- When the ATU PCI Target Abort (master) Interrupt Mask bit in the ATUIMR is clear, set the PCI Target Abort (master) bit in the ATUISR. When set, no action.
- When the transaction is an MSI outbound write and the **SERR#** Enable bit in the ATUCMD is set, assert **SERR#**; otherwise, no action is taken. When the ATU asserts **SERR#,** additional actions are taken:
  — Set the **SERR#** Asserted bit in the ATUSR.
  — When the ATU **SERR#** Asserted Interrupt Mask Bit in the ATUIMR is clear, set the **SERR#** Asserted bit in the ATUISR. When set, no action.
  — When the ATU **SERR#** Detected Interrupt Enable Bit in the ATUCR is set, set the **SERR#** Detected bit in the ATUISR. When clear, no action.
- When operating in the PCI-X mode and the Target-Abort is signaled via a Split Completion Error Message, the Received Split Completion Error Message bit in the PCIXSR is set. When the ATU sets this bit, additional actions are taken:
  — When the ATU Received Split Completion Error Message Interrupt Mask bit in the ATUIMR is clear, set the Received Split Completion Error Message bit in the ATUISR. When set, no action.
- For an Outbound Read request, the read completion is aborted on the internal bus.
- Flush the address from the OTQ.

2.7.6.2 Inbound Read Completion or Inbound Configuration Write Completion Message

The ATU encounters this error only in the PCI-X mode.

A target abort is signaled when the target of the transaction simultaneously deasserts **DEVSEL#**, deasserts **TRDY#,** and asserts **STOP#**.

When the ATU is signaled a Target-Abort while initiating either a Split Read Completion Transaction or a Split Write Completion Message, the ATU discards the Split Completion and take no further action

2.7.6.3 Target-Aborts Signaled by the ATU as a Target

2.7.6.3.1 Internal Bus Master Abort

A target abort can be signaled by the ATU during an inbound read request where the internal bus cycle resulted in a master abort on the Internal Bus.

Please see Section 2.7.9.1, “Master Abort on the Internal Bus” on page 115 for details on the ATU response to an Internal Bus Master Abort.

2.7.6.3.2 Internal Bus Target Abort

A target abort can be signaled by the ATU during an inbound read request where the internal bus cycle resulted in a Target Abort from the memory controller due to a non-recoverable multi-bit ECC error.

Please see Section 2.7.9.2, “Target Abort on the Internal Bus” on page 117 for details on the ATU response to an Internal Bus Target Abort.

2.7.6.3.3 Inbound EROM Memory Write

Since the EROM memory window is defined to be read-only by the *PCI Local Bus Specification*, Revision 2.3, the ATU target-aborts when an inbound write transaction is claimed by the EROM memory window.

The following additional actions with the given constraints are performed by the ATU when a target abort is signaled by the PCI target interface during an inbound EROM Memory write transaction:

- Set the Target Abort (target) bit (bit 11) in the ATUSR.
  - When the ATU PCI Target Abort (target) Interrupt Mask bit in the ATUIMR is clear, set the PCI Target Abort (target) bit in the ATUISR. When set, no action.

2.7.7 Corrupted or Unexpected Split Completions

Warning: When any of the errors discussed in this section actually occur, a catastrophic system failure is likely to result from which the *PCI-X Protocol Addendum to the PCI Local Bus Specification,* Revision 2.0 provides no recovery mechanism. In these cases, the ATU may be communicating with a non-compliant target device or the system may not be configured properly.

2.7.7.1 Completer Address

The ATU only asserts **DEVSEL#** for split completion transactions where the Sequence ID (Requester ID & Tag) matches that of a currently outstanding split request in the OTQ.

Conversely, the ATU does not assert **DEVSEL#** for any split completion transaction where either the Requester ID does not match that of the ATU or the Tag does not match that of any currently outstanding split request. No further action is taken.

When the Sequence ID of a split completion transaction matches that of an outstanding request, but the Lower Address field is not valid, the ATU accepts the split completion transaction in its' entirety according to the invalid Lower Address field and set the Unexpected Split Completion bit in the PCIXSR. No further action is taken.

2.7.7.2 Completer Attributes

When the Sequence ID of a split completion transaction matches that of an outstanding request, but the Byte Count is not valid, the ATU accepts the split completion transaction in its' entirety according to the invalid byte count field and set the Unexpected Split Completion bit in the PCIXSR. In this case, the ATU discards all the data. No further action is taken.

2.7.8 SERR# Assertion and Detection

The ATU is capable of reporting error conditions through the use of the **SERR#** output.

The following conditions may result in the assertion of **SERR#** by the ATU:

- An uncorrectable address error (or an uncorrectable attribute error when operating in the PCI-X mode) is detected by the ATU PCI interface (see Section 2.7.1, "Uncorrectable Address and Uncorrectable Attribute Errors on the PCI Interface" on page 95 for details).
- A Master Data Parity Error is recorded in the ATUSR while operating in the PCI-X mode (see Section 2.7.3, "Uncorrectable Data Errors on the PCI Interface" on page 97 for details).
- An outbound MSI write transaction is either signaled a Master-Abort or a Target-Abort by the target.
- An inbound write transaction is master aborted on the internal bus (see Section 2.7.9.1, "Master Abort on the Internal Bus" on page 115 for details).
- The **SERR#** Manual Assertion bit in the ATUCR has been set and the **SERR#** Enable bit is set in the ATUCMD.

Note that the **SERR#** manual assertion bits must be cleared manually before they can be set again resulting in **SERR#** asserted. Refer to Section 2.14.40, "ATU Configuration Register - ATUCR" on page 177 for details.

The following actions with the given constraints are performed by the ATU when **SERR#** is asserted by the PCI interface:

- Set the **SERR#** Asserted bit in the ATUSR.
- When the ATU **SERR#** Asserted Interrupt Mask bit in the ATUIMR is clear, set the **SERR#** Asserted bit in the ATUISR. When set, no action.
- When **SERR#** is asserted and the ATU **SERR#** Detected interrupt enable is set in the ATUCR, set the **SERR#** Detected bit in the ATUISR. When clear, no action.

The following actions with the given constraints are performed by the ATU when **SERR#** is detected by the PCI interface:

- When **SERR#** is detected and the ATU **SERR#** Detected interrupt enable is set in the ATUCR, set the **SERR#** Detected bit in the ATUISR. When clear, no action.

Note: Whenever the ATU asserts **SERR#**, both the asserted and detected status bits may be set in the corresponding ISR. To mask an interrupt to the core when the ATU asserts **SERR#**, the **SERR#** asserted mask bit must be set and the **SERR#** detected interrupt enable bit must be clear.

2.7.9 Internal Bus Error Conditions

An internal bus error results in a bit being set in the Interrupt Status Registers at which time an interrupt is driven to the Intel XScale® processor. Unlike PCI errors, internal bus error conditions are not maskable.

The following sections detail internal bus error conditions for the ATU.

2.7.9.1 Master Abort on the Internal Bus

A master abort on the internal bus is seen by the ATU when the inbound translated address presented on the internal bus is not claimed.

2.7.9.1.1 Inbound Write Request

The following action with the given constraints are performed by the ATU when a master abort is detected by the internal master interface during an inbound write request transaction:

- Set the Internal Bus Master Abort bit (bit 7) in the ATUISR.
- When the Inbound Error **SERR#** Enable bit is set in the ATUIMR and the **SERR#** Enable bit is set in the ATUCMD, assert **SERR#** on the PCI interface. When both bits are not set, take no action. When **SERR#** is asserted, additional actions are taken:
    - Set the **SERR#** Asserted bit in the ATUSR
    - When the ATU **SERR#** Asserted Interrupt Mask bit in the ATUIMR is clear, set the **SERR#** Asserted bit in the ATUISR. When set, no action
    - When the ATU **SERR#** Detected interrupt enable is set in the ATUCR, set the **SERR#** Detected bit in the ATUISR. When clear, no action
- Flush the transaction that was master aborted from the IWQ.

The Internal Bus Master Abort bit is non-maskable and always results in an interrupt being driven to the core processor.

2.7.9.1.2 Inbound Read Request

When operating in the Conventional mode, the following actions with the given constraints are performed by the ATU when a master abort is detected by the internal initiator interface during an inbound read transaction:

- Set the Internal Bus Master Abort bit (bit 7) in the ATUISR
- Return a target abort condition to the initiating master during the delayed completion cycle on the PCI bus. No data is ever read from the internal bus and returned to the PCI bus.

The following additional actions with the given constraints are performed by the ATU when a target abort is signaled by the PCI interface during an inbound delayed read completion cycle:

- Set the Target Abort (target) bit (bit 11) in the ATUSR.
- When the ATU PCI Target Abort (target) Interrupt Mask bit in the ATUIMR is clear, set the PCI Target Abort (target) bit in the ATUISR. When set, no action.
- Flush the transaction that was master aborted from the ITQ after the target abort is delivered on the PCI interface.

When operating in the PCI-X mode, the following actions with the given constraints are performed by the ATU when a master abort is detected by the internal master interface during an inbound split read transaction:

- Set the Internal Bus Master Abort bit (bit 7) in the ATUISR.
- Generate a Split Completion Error Message (with message class=2h - completer error and message index=80h - 4138xx internal bus master abort) on the PCI bus.
- When the Initiated Split Completion Error Message Interrupt Mask in the ATUIMR is clear, set the Initiated Split Completion Error Message bit in the ATUISR. When set, no action.
- Flush the transaction that was master aborted.

Note: This split completion error message includes a device specific message index. The error handler would need to have knowledge of the device specific error messages of the 4138xx in order to fully diagnose the problem.

The Internal Bus Master Abort bit is non-maskable and always results in an interrupt being driven to the core processor.

2.7.9.2 Target Abort on the Internal Bus

Target Aborts can be seen by the internal bus requester interface during inbound read operations to the memory controller. During inbound read operations, the memory controller is capable of signalling a target abort when a multi-bit, unrecoverable ECC error is encountered. This can occur during any read operation.

Note target aborts are signalled on a Qword basis. When either Dword of a Qword target aborts, both are considered to have target aborted.

The Memory Controller is responsible for creating an interrupt to the Intel XScale® processor for any multi-bit ECC errors.

2.7.9.2.1 Conventional Mode

When operating in the Conventional PCI mode, when the data word which was target aborted on the internal bus is actually requested and delivered on the PCI Bus, and the ATU ECC Target Abort Enable bit is set in the ATUIMR, a target abort is returned to the PCI initiator on that data word. When the ATU ECC Target Abort Enable bit is cleared in the ATUIMR, a disconnect with data is returned to the PCI initiator during the data word that was target aborted on the internal bus. In both cases, the IRQ is flushed after the completion cycle is performed on the PCI bus

The following additional actions with the given constraints are performed by the ATU when a target abort is signaled by the PCI target interface during an inbound read transaction:

- Set the Target Abort (target) bit (bit 11) in the ATUSR.
- When the ATU PCI Target Abort (target) Interrupt Mask bit in the ATUIMR is clear, set the PCI Target Abort (target) bit in the ATUISR. When set, no action.

2.7.9.2.2 PCI-X Mode

When operating in the PCI-X mode, a Target-Abort of an inbound read transaction (split read request) on the Internal Bus results in the following actions.

- The ATU initiates a Split Completion Error Message (with message class=2h - completer error and message index=81h - 4138xx internal bus target abort) on the PCI bus.
- When the Initiated Split Completion Error Message Interrupt Mask in the ATUIMR is clear, set the Initiated Split Completion Error Message bit in the ATUISR. When set, no action.

Note: This split completion error message includes a device specific message index. The error handler would need to have knowledge of the device specific error messages of the 4138xx in order to fully diagnose the problem.

2.7.9.3 Parity Error on the Internal Bus

The 4138xx provides support for byte-wise parity protection on the internal bus. The internal bus consists of a 36 bit address bus and 128 bit data bus; both are protected by byte-wise parity. The internal bus parity protection is provided independent of the operating mode of the ATUs PCI interface.

When initiating transactions on the internal bus, the ATUs internal bus interface generates byte-wise parity. As a target the ATU checks byte-wise parity.

As an initiator, for outbound write transactions the ATU will forward bad data parity on the PCI bus if the ATU detected a parity error on the internal bus interface. And for outbound read transactions the ATU will forward bad data parity on the internal bus if the ATU detected a parity error on the PCI interface. Outbound read parity error will be detected and logged by the internal bus initiator. For outbound read data that has to flow through the internal bus bridge, the bridge will log the error. Refer to the internal bus bridge chapter for more details on how the parity error is handled.

As a target, for inbound read transactions the ATU will forward bad data parity on the PCI bus if the ATU detected a parity error on the internal bus interface. And for inbound write transactions the ATU will forward bad data parity on the internal bus if the ATU detected a parity error on the PCI interface. Inbound write parity error will be detected and logged by the internal bus target. For write data that has to flow through the internal bus bridge, the bridge will log the error. Refer to the internal bus bridge chapter for more details on how the parity error is handled.

2.7.9.3.1 Conventional Mode

On an inbound read transaction, when the data word where the internal bus parity error is detected is actually requested and returned to the PCI bus, a target abort is returned to the PCI initiator on that data word. The IRQ is flushed after the completion cycle is performed on the PCI bus

The following additional actions with the given constraints are performed by the ATU when a target abort is signaled by the PCI target interface during an inbound read transaction:

- Set the Target Abort (target) bit (bit 11) in the ATUSR.
- When the ATU PCI Target Abort (target) Interrupt Mask bit in the ATUIMR is clear, set the PCI Target Abort (target) bit in the ATUISR. When set, no action.

2.7.9.3.2 PCI-X Mode

An internal bus parity error of an inbound read transaction (split read request) on the Internal Bus results in the following actions.

- The ATU initiates a Split Completion Error Message (with message class=2h - completer error and message index=81h - 4138xx internal bus target abort) on the PCI bus.
- When the Initiated Split Completion Error Message Interrupt Mask in the ATUIMR is clear, set the Initiated Split Completion Error Message bit in the ATUISR. When set, no action.

Note: This split completion error message includes a device specific message index. The error handler would need to have knowledge of the device specific error messages of the 4138xx in order to fully diagnose the problem.

2.7.10 ATU Error Summary

Table 18 summarizes the ATU error reporting for PCI bus errors and Table 19 summarizes the ATU error reporting for internal bus errors. The tables assume that all error reporting is enabled through the appropriate command registers (unless otherwise noted). The ATU Status Register records PCI bus errors. Note that the **SERR#** Asserted bit in the Status Register is set only when the **SERR#** Enable bit in the Command Register is set. The ATU Interrupt Status Registers record Intel XScale® processor interrupt status information.

Table 18. ATU Error Reporting Summary - PCI Interface (Sheet 1 of 5)

Error Condition (Bus Mode[a])	Bits Set in ATU Status Register (ATUSR[b]) or PCI-X Status Register (PCIXSR[c]) and/or ECC Logging Registers[d] (ECCLOG)	Bits Set in ATU Interrupt Status Register (ATUISR)	Interrupt Mask Bit in ATUIMR or ATUCR
	PCI Bus Error Response (i.e., signal Target-Abort, signal Master-Abort etc.)		
Uncorrectable Address or Attribute Error (All)	In Conventional Mode, ignore (Master-Abort) the transaction, and then signal **SERR#**. In PCI-X Mode, claim the transaction and complete as when no error had occurred; and, signal **SERR#** upon uncorrectable error detection.		
(All)	**SERR#** Asserted - bit 14	**SERR#** Asserted - bit 10	ATUIMR bit 6
(All)	N/A	**SERR#** Detected - bit 4	ATUCR bit 9
(All)	Detected Uncorrectable Address or Attribute Error - bit 20 of the PCI Configuration and Status Register	N/A	N/A
(All)	Detected Parity Error - bit 15	Detected Parity Error - bit 9	ATUIMR bit 7
(PCI-X2)	ECCLOG Updated	N/A	N/A
Correctable Address or Attribute Error (PCI-X2)	The transaction is completed as when no error had occurred. Then the transaction is forwarded to the internal bus normally.		
(PCI-X2)	ECCLOG Updated	Detected Correctable Error - bit 14	ATUIMR bit 11
Outbound Read Request Uncorrectable Data Error (All)	Signal **PERR#** and **SERR#** (PCI-X Mode Only).		
(All)	Master Parity Error - bit 8	Master Parity Error - bit 0	ATUIMR bit 2
(PCI-X)	**SERR#** Asserted - bit 14	**SERR#** Asserted - bit 10	ATUIMR bit 6
(PCI-X)	N/A	**SERR#** Detected - bit 4	ATUCR bit 9
(All)	Detected Parity Error - bit 15	Detected Parity Error - bit 9	ATUIMR bit 7
Outbound Write Request Uncorrectable Data Error (All)	Signal **SERR#** (only for PCI-X or MSI Writes).		
(All)	Master Parity Error - bit 8	Master Parity Error - bit 0	ATUIMR bit 2
(PCI-X or MSI)	**SERR#** Asserted - bit 14	**SERR#** Asserted - bit 10	ATUIMR bit 6
(PCI-X or MSI)	N/A	**SERR#** Detected - bit 4	ATUCR bit 9
Inbound Read Completion Uncorrectable Data Error (PCI-X)	None.		

Table 18. ATU Error Reporting Summary - PCI Interface (Sheet 2 of 5)

Error Condition (Bus Mode[a])	Bits Set in ATU Status Register (ATUSR[b]) or PCI-X Status Register (PCIXSR[c]) and/or ECC Logging Registers[d] (ECCLOG)	Bits Set in ATU Interrupt Status Register (ATUISR)	Interrupt Mask Bit in ATUIMR or ATUCR
	PCI Bus Error Response (i.e., signal Target-Abort, signal Master-Abort etc.)		
Inbound Configuration Write Completion Message Uncorrectable Data Error (PCI-X)	None.		
Inbound Read Request Uncorrectable Data Error (All)	None.		
Inbound Write Request Uncorrectable Data Error (All)	Signal **PERR#**.		
(All)	Detected Parity Error - bit 15	Detected Parity Error - bit 9	ATUIMR bit 7
(PCI-X2)	ECCLOG Updated	N/A	N/A
Outbound Read Completion Uncorrectable Data Error (All)	Signal **PERR# and SERR#**.		
(PCI-X	Master Parity Error - bit 8	Master Parity Error - bit 0	ATUIMR bit 2
(PCI-X)	**SERR#** Asserted - bit 14	**SERR#** Asserted - bit 10	ATUIMR bit 6
(PCI-X)	N/A	**SERR#** Detected - bit 4	ATUCR bit 9
(PCI-X)	Detected Parity Error - bit 15	Detected Parity Error - bit 9	ATUIMR bit 7
(PCI-X2)	ECCLOG Updated	N/A	N/A
Outbound Split Write Uncorrectable Data Error Message (PCI-X)	Signal **SERR#**.		
(PCI-X)	Master Parity Error - bit 8	Master Parity Error - bit 0	ATUIMR bit 2
(PCI-X)	**SERR#** Asserted - bit 14	**SERR#** Asserted - bit 10	ATUIMR bit 6
(PCI-X)	N/A	**SERR#** Detected - bit 4	ATUCR bit 9
(PCI-X)	Received Split Completion Error Message - bit 29	Received Split Completion Error Message - bit 12	ATUIMR bit 9
Inbound Configuration Write Request Uncorrectable Data Error (All)	Signal **PERR#**. Initiate an Uncorrectable Split Write Data Error Message addressed to the Requester (PCI-X Mode Only).		
(PCI-X)	N/A	Initiated Split Completion Error Message - bit 13	ATUIMR bit 10
(All)	Detected Parity Error - bit 15	Detected Parity Error - bit 9	ATUIMR bit 7
(PCI-X2)	ECCLOG Updated	N/A	N/A
Split Completion Message Uncorrectable Data Error (PCI-X)	Signal **PERR#** and **SERR#**.		

Table 18. ATU Error Reporting Summary - PCI Interface (Sheet 3 of 5)

Error Condition (Bus Mode[a])	Bits Set in ATU Status Register (ATUSR[b]) or PCI-X Status Register (PCIXSR[c]) and/or ECC Logging Registers[d] (ECCLOG)	Bits Set in ATU Interrupt Status Register (ATUISR)	Interrupt Mask Bit in ATUIMR or ATUCR
	PCI Bus Error Response (i.e., signal Target-Abort, signal Master-Abort etc.)		
(PCI-X)	Master Parity Error - bit 8	Master Parity Error - bit 0	ATUIMR bit 2
(PCI-X)	**SERR#** Asserted - bit 14	**SERR#** Asserted - bit 10	ATUIMR bit 6
(PCI-X)	N/A	**SERR#** Detected - bit 4	ATUCR bit 9
(PCI-X and SCE[e])	Received Split Completion Error Message - bit 29	Received Split Completion Error Message - bit 12	ATUIMR bit 9
(PCI-X)	Detected Parity Error - bit 15	Detected Parity Error - bit 9	ATUIMR bit 7
(PCI-X2)	ECCLOG Updated	N/A	N/A
Outbound Read Request Correctable Data Error (PCI-X2)	None.		
Outbound Write Request Correctable Data Error (PCI-X2)	None.		
Inbound Read Completion Correctable Data Error (PCI-X)	None.		
Inbound Configuration Write Completion Message Correctable Data Error (PCI-X)	None.		
Inbound Read Request Correctable Data Error (PCI-X2)	None.		
Inbound Write Request Correctable Data Error (PCI-X2)	None.		
(PCI-X2)	ECCLOG Updated	Detected Correctable Error - bit 14	ATUIMR bit 11
Outbound Read Completion Correctable Data Error (PCI-X2)	None.		
(PCI-X2)	ECCLOG Updated	Detected Correctable Error - bit 14	ATUIMR bit 11
Inbound Configuration Write Request Correctable Data Error (PCI-X2)	None.		
(PCI-X2)	ECCLOG Updated	Detected Correctable Error - bit 14	ATUIMR bit 11

Table 18. ATU Error Reporting Summary - PCI Interface (Sheet 4 of 5)

Error Condition (Bus Mode[a])	Bits Set in ATU Status Register (ATUSR[b]) or PCI-X Status Register (PCIXSR[c]) and/or ECC Logging Registers[d] (ECCLOG)	Bits Set in ATU Interrupt Status Register (ATUISR)	Interrupt Mask Bit in ATUIMR or ATUCR
	PCI Bus Error Response (i.e., signal Target-Abort, signal Master-Abort etc.)		
Split Completion Message Correctable Data Error (PCI-X)	None.		
(PCI-X2)	ECCLOG Updated	Detected Correctable Error - bit 14	ATUIMR bit 11
Outbound Read Request Master-Abort (All)	None.		
(All)	Master Abort - bit 13	PCI Master Abort - bit 3	ATUIMR bit 5
(PCI-X and SCE)	Received Split Completion Error Message - bit 29	Received Split Completion Error Message - bit 12	ATUIMR bit 9
Outbound Write Request Master-Abort (All)	None.		
(All)	Master Abort - bit 13	PCI Master Abort - bit 3	ATUIMR bit 5
(MSI)	**SERR#** Asserted - bit 14	**SERR#** Asserted - bit 10	ATUIMR bit 6
(MSI)	N/A	**SERR#** Detected - bit 4	ATUCR bit 9
(PCI-X and SCE)	Received Split Completion Error Message - bit 29	Received Split Completion Error Message - bit 12	ATUIMR bit 9
Inbound Read Completions Master-Abort (PCI-X)	None.		
Inbound Configuration Write Completion Message Master-Abort (PCI-X)	None.		
Outbound Read Request Target-Abort (All)	None.		
(All)	Target Abort (master) - bit 12	PCI Target Abort (master) - bit 2	ATUIMR bit 4
(PCI-X and SCE)	Received Split Completion Error Message - bit 29	Received Split Completion Error Message - bit 12	ATUIMR bit 9
Outbound Write Request Target-Abort (All)	None.		
(All)	Target Abort (master) - bit 12	PCI Target Abort (master) - bit 2	ATUIMR bit 4
(MSI)	**SERR#** Asserted - bit 14	**SERR#** Asserted - bit 10	ATUIMR bit 6
(MSI)	N/A	**SERR#** Detected - bit 4	ATUCR bit 9
(PCI-X and SCE)	Received Split Completion Error Message - bit 29	Received Split Completion Error Message - bit 12	ATUIMR bit 9

Table 18. ATU Error Reporting Summary - PCI Interface (Sheet 5 of 5)

Error Condition (Bus Mode[a])	Bits Set in ATU Status Register (ATUSR[b]) or PCI-X Status Register (PCIXSR[c]) and/or ECC Logging Registers[d] (ECCLOG)	Bits Set in ATU Interrupt Status Register (ATUISR)	Interrupt Mask Bit in ATUIMR or ATUCR
	PCI Bus Error Response (i.e., signal Target-Abort, signal Master-Abort etc.)		
Inbound EROM Write Request Target-Abort (All)	Signal Target-Abort.		
(All)	Target Abort (target) - bit 11	PCI Target Abort (target) - bit 1	ATUIMR bit 3
Unexpected Split Completion (PCI-X)	In the PCI-X mode, the transaction completes normally according to the invalid lower address field or invalid byte count.		
(PCI-X)	Unexpected Split Completion - bit 19	N/A	N/A

a. Codes for bus mode in which this error response applies: PCI-X means PCI-X Mode 1 or PCI-X Mode 2, PCI-X2 means PCI-X Mode 2 only, Conventional means Conventional PCI Mode Only, and All means that the error response applies in the Conventional, PCI-X Mode 1 and PCI-X Mode 2 modes of operation. MSI stands for Message-Signaled Interrupts and refers to an Outbound Write transaction that is actually an MSI write transaction.

b. Table assumes that Parity Error Response - bit 6 of the ATUCMD register is set.

c. Table assumes that Data Parity Recovery Enable - bit 0 of the PCIXCMD is clear.

d. When a correctable or uncorrectable data error occurs in PCI-X Mode 2, the ECC Logging registers consisting of the ECC Control and Status Register - ECCCSR, the ECC First Address Register - ECCFAR, the ECC Second Address Register - ECCSAR, and the ECC Attribute Register - ECCAR are updated with additional information about the ECC error.

e. When the SCE bit (bit 30 of the Completer Attributes) and the SCM bit (bit 29 of the Completer Attributes) are set during the Attribute phase of a Split Completion Transaction, the transaction is a Split Completion Message that is an Error Message. In this case, the Received Split Completion Error Message - bit 29 of the PCIXSR is set.

Table 19. ATU Error Reporting Summary - Internal Bus Interface

Error Condition[a] (Bus Mode[b])	Bits Set in ATU Status Register (ATUSR[c])	Bits Set in ATU Interrupt Status Register (ATUISR)	Interrupt Mask Bit in ATUIMR or ATUCR
	PCI Bus Error Response (i.e., signal Target-Abort, signal Master-Abort etc.)		
Inbound Write Request Master-Abort (All)	Assert **SERR#**.		
(All)	N/A	Internal Bus Master Abort - bit 7	N/A
(All)	**SERR#** Asserted - bit 14	**SERR#** Asserted - bit 10	ATUIMR bit 6
(All)	N/A	**SERR#** Detected - bit 4	ATUCR bit 9
Inbound Read Request Master-Abort (All)	In the Conventional Mode signal Target-Abort. In the PCI-X Mode send a device specific Split Completion Error Message to the Requester.		
(All)	N/A	Internal Bus Master Abort - bit 7	N/A
(Conventional)	Target Abort (target) - bit 11	PCI Target Abort (target) - bit 1	ATUIMR bit 3
(PCI-X)	N/A	Initiated Split Completion Error Message - bit 13	ATUIMR bit 10
Inbound Read Request Target-Abort (All)	In the Conventional Mode signal Target-Abort. In the PCI-X Mode send a device specific Split Completion Error Message to the Requester.		
(Conventional)	Target Abort (target) - bit 11	PCI Target Abort (target) - bit 1	ATUIMR bit 3
(PCI-X)	N/A	Initiated Split Completion Error Message - bit 13	ATUIMR bit 10

a. There are no Inbound Write Request Target-Abort Error Conditions.
b. Codes for bus mode in which this error response applies: PCI-X means PCI-X Mode 1 or PCI-X Mode 2, PCI-X2 means PCI-X Mode 2 only, Conventional means Conventional PCI Mode Only, and All means that the error response applies in the Conventional, PCI-X Mode 1 and PCI-X Mode 2 modes of operation. MSI stands for Message-Signaled Interrupts and refers to an Outbound Write transaction that is actually an MSI write transaction.
c. Table assumes that the ATU Inbound **SERR#** Enable bit (bit 1 of the ATUIMR), the ATU ECC Target Abort Enable (bit 0 of the ATUIMR), and the **SERR#** Enable bit (bit 8 of the ATUCMD) are set.

2.8 Message-Signaled Interrupts

The Messaging Unit is responsible for the generation of all of the Outbound Interrupts from the 4138xx. These interrupts can be delivered to the Host Processor via the **P_INTA#** output pin or the Message Signaled Interrupt (MSI) mechanism.

When a host processor enables Message-Signaled Interrupts (MSI) on the 4138xx, an outbound interrupt is signaled to the host via a PCI write instead of the assertion of the **P_INTA#** output pin.

In support of MSI, the 4138xx implements the MSI capability structure. The capability structure includes the Section 4.7.20, "MSI Capability Identifier Register - Cap_ID" on page 429, the Section 4.7.21, "MSI Next Item Pointer Register - MSI_Next_Ptr" on page 430, the Section 4.7.23, "Message Address Register - Message_Address" on page 432, the Section 4.7.24, "Message Upper Address Register - Message_Upper_Address" on page 433 and theSection 4.7.25, "Message Data Register- Message_Data" on page 434.

The Message Unit generates MSIs by writing to the MSI port via the internal bus. The ATU generates a write transaction whenever the Message Unit writes to the MSI port, using the address specified in the Section 4.7.23, "Message Address Register - Message_Address" on page 432, the Section 4.7.24, "Message Upper Address Register - Message_Upper_Address" on page 433 and theSection 4.7.25, "Message Data Register- Message_Data" on page 434.

2.9 Internal Interrupts

The ATU has 3 internal interrupts that connect to the internal Interrupt Controller Unit.

- ATU Interrupt Status Register Interrupt
- ATU Configuration Write Interrupt
- ATU BIST Interrupt

2.10 Vital Product Data

Vital Product Data (VPD) provides detailed information to the system regarding the hardware, software and microcode elements of a device. This information may include Part Number, Serial Number or other detailed information. This information resides on a non-volatile storage device (i.e., Flash Memory) attached to the 4138xx. In addition VPD also provides a mechanism for storing information such as performance or failure data on the device being monitored.

Support of VPD involves the implementation of the VPD Extended Capabilities List Item in the Primary ATU. The VPD Extended capabilities header consists of five registers, the "VPD Capability Identifier Register - VPD_Cap_ID" on page 185, the "VPD Next Item Pointer Register - VPD_Next_Item_Ptr" on page 185, the "VPD Address Register - VPDAR" on page 186, and the "VPD Data Register - VPDDR" on page 186.

Scheduled by Intel XScale® processor interrupts, the 4138xx may be used to retrieve or store VPD information through the VPD extended capabilities list item.

Please consult Appendix I of the *PCI Local Bus Specification*, Revision 2.3 for the definitions of compliant VPD format.

2.10.1 Configuring Vital Product Data Operation

By default, the 4138xx VPD functionality is not configured for operation. Specifically, the VPD Extended Capabilities List Item is not discovered during a PCI bus scan and the ATUs VPD interrupt status bit in the "ATU Interrupt Status Register - ATUISR" on page 181 is masked by the "ATU Interrupt Mask Register - ATUIMR" on page 183. The following steps should be followed to properly configure the 4138xx support for VPD:

1. The 4138xx must be strapped to Retry Type 0 Configuration cycles following the deassertion of **P_RST#**. Enabling this configuration cycle retry mechanism insures that the Intel XScale® processor can make the VPD Extended Capabilities List Item visible before the system configures the 4138xx. The configuration retry mechanism is controlled through bit 2 of the "PCI Configuration and Status Register - PCSR" on page 178.
2. When the configuration retry mechanism is strapped enabled as described in step 1, typically, the 4138xx would also be strapped such that the Intel XScale® processor would immediately boot following the deassertion of **P_RST#** (bit 1 of the PCSR), though this is not required.
3. The Intel XScale® processor writes E8H to the "PCI-X Next Item Pointer Register - PCI-X_Next_Item_Ptr" on page 191. This links the PCI-X Capabilities List Item to the VPD Capabilities List Item.
4. The Intel XScale® processor clears bit 12 of the ATUIMR to enable the ATUs VPD interrupt status bit.

2.10.2 Accessing Vital Product Data

The VPD Capabilities List Item provides three fields which the system uses to access the Vital Product Data:

VPD Address	DWORD Aligned Byte address of the VPD to be accessed which is represented by VPDAR[14:0]. Note that this means that the maximum size of the VPD is 128 Kbytes. The user may pick any 128 Kbyte block of memory in the storage component for the VPD.
Flag	The flag register is used to indicate when the transfer between the VPD Data Register and the storage component is completed. The flag is in VPDAR[15] which means that the Flag is written at the same time that VPD address is written.
VPD Data	Four bytes of VPD Data can be read or written through this field which is represented by VPDDR[31:0]. The least significant byte of this register represents the byte at the VPD Address (VPDAR[14:0]). Four bytes are always transferred between this register and the VPD storage component.

2.10.2.1 Reading Vital Product Data

Using the fields defined in the VPD Capabilities List Item, the 4138xx reads Vital Product Data using the following sequence of events:

1. Host processor executes a configuration write of the VPD address to the VPDAR with the Flag cleared.
2. An interrupt to the Intel XScale® processor is triggered and bit 17 of the ATUISR is set. Meanwhile, the host processor polls the VPDAR register waiting for the Flag to be set.

Warning: When any configuration writes to either the VPDAR or the VPDDR occur prior to the Flag being set, the results of the original read operation are unpredictable.

3. Using the VPD Address, the Intel XScale® processor retrieves the Vital Product Data from the VPD storage component (i.e., Flash Memory).
4. The Intel XScale® processor then writes this data to VPD Data Register (VPDDR).
5. The Intel XScale® processor clears the VPD interrupt status bit in the ATUISR.
6. The Intel XScale® processor then sets the Flag in the VPDAR register.
7. When the host processor detects that the Flag has been set, the host processor then reads the retrieved VPD from the VPDDR.

2.10.2.2 Writing Vital Product Data

Using the fields defined in the VPD Capabilities List Item, the 4138xx writes Vital Product Data using the following sequence of events:

1. Host processor executes a configuration write of the VPD data to be written to the VPDDR.
2. Host processor executes a configuration write of the VPD address to the VPDAR with the Flag set.
3. An interrupt to the Intel XScale® processor is triggered and bit 17 of the ATUISR is set. Meanwhile, the host processor polls the VPDAR register waiting for the Flag to be cleared.

Warning: When any configuration writes to either the VPDAR or the VPDDR occur prior to the Flag being cleared, the results of the original write operation are unpredictable.

4. Using the VPD Address, the Intel XScale® processor writes the Vital Product Data from the VPDDR to the VPD storage component (i.e., Flash Memory).
5. The Intel XScale® processor clears the VPD interrupt status bit in the ATUISR.
6. The Intel XScale® processor then clears the Flag in the VPDAR register.
7. When the host processor detects that the Flag has been cleared, the host processor has been informed that the VPD write operation is complete.

2.11 Multi-Function Support

Multiple functions are not supported for 4138xx.

2.11.1 PCI-X Interface Control Parameters

The following registers are located in the configuration space header and extended space and provide control of the PCI Interface. The effect of each bit is detailed below.

Note: Table 20 is referring to only enabled functions. In root complex mode multi-function is not applicable.

Table 20. PCI-X Interface Control Parameters Usage

Register Name	Register Bits Description	Usage
ATU Command Register - ATUCMD	Bit 10 - Interrupt Disable	Each function can independently control this bit.
	Bit 9 - Fast Back-to-Back Enable	Not applicable, since this is not supported.
	Bit 8 - SERR# Enable	SERR# bit from each function is logically ORed and then fed to the PCI Interface. This implies that SERR# is globally enabled when only one of the functions enables SERR#.
	Bit 6 - Parity Error Response	Parity Error Response bit from each function is logically ORed and fed to the PCI Interface. This implies that Parity Error Response is globally enabled when only one of the functions enables Parity Error Response.
	Bit 4 - MWI Enable	MWI Enable bit from each function is logically ORed and then fed to the PCI Interface. This implies that MWI Enable is globally enabled when only one of the functions enables MWI Enable.
	Bit 2 - Bus Master Enable	Each function can independently control this bit.
	Bit 1 - Memory Enable	Each function can independently control this bit.
	Bit 0 - I/O Enable	Each function can independently control this bit.
ATU Cacheline Size Register - ATUCLSR	Entire Register	Provided by function 0. For 4138xx, the ATU provides this parameter.
ATU Latency Timer Register - ATULT	Entire Register	Each function can independently control this register.
ATU BIST Register - ATUBISTR	Entire Register	Each Function can independently control this register
ATU Minimum Grant Register - ATUMGNT	Entire Register	Each Function can independently control this register
ATU PCI-X Command Register - PCIXCMD	Bit[6:4] - Maximum Outstanding Split Transactions	The PCI Interface sums the values from each function and use the result as the parameter.
	Bit[3:2] - Maximum Memory Read Byte Count	The PCI Interface uses the least common denominator (LCD) based on the values from all the functions.
	Bit0 - Uncorrectable Error Recovery	Uncorrectable Error Recovery bit from each function is logically ANDed and then fed to the PCI Interface. This implies that Uncorrectable Error Recovery is enabled when all of the functions enable this capability.
ATU ECC Control and Status Register - ECCCSR	Bit 30 - Disable Single-Bit Error Correction	Disable Single-Bit Error Correction bit from each function is logically ANDed and then fed to the PCI Interface. This implies that Disable Single-Bit Error Correction is disabled when all of the functions disable this capability.

2.11.2 PCI-X Interface Status Reporting

The following registers are located in the configuration space header and extended space and provide status (error conditions) of the PCI Interface.

Table 21. PCI-X Host Interface Status Reporting Usage[a]

Register Name	Register Bits Description	Usage
ATU Status Register - ATUSR	Bit 15 - Detected Parity Error	Address and Attribute parity error is reported to all functions simultaneously. Data parity error is reported to only the function involved.
	Bit 14 - SERR# Asserted	SERR# Asserted is reported to only the function involved.
	Bit 13 - Master Abort	Master Abort is reported to only the function involved.
	Bit 12 - Target Abort (Master)	Target Abort (Master) is reported to only the function involved.
	Bit 11 - Target Abort (Target)	Target Abort (Target) is reported to only the function involved.
	Bit 8 - Bit Master Parity Error	Master Parity Error is reported to only the function involved.
ATU PCI-X Status Register - PCIXSR	Bit 29 - Received Split Completion Error Message	Received Split Completion Error Message is reported to only the function involved.
	Bit 19 - Unexpected Split Completion	Unexpected Split Completion is reported to only the function involved.
	Bits[15:8] - Bus Number	
	Bits[7:3] - Device Number	
	Bits[2:0]	Each function indicates its function number.
ATU ECC Control and Status Register - ECCCSR	Bits[27:2]	Address and Attribute ECC error is reported to all functions simultaneously. Data ECC error is reported to only the function involved.

a. This table is referring to only enabled functions. And in root complex mode multi-function is not applicable.

Note: Registers or bits within a register that are described in Table 20 and Table 21 exist in each function independently.

2.12 Central Resource Functionality

Central Resource is not supported on 4138xx.

2.12.1 Multi-Function Support

When operating as a central resource, the ATU behaves as a single function device and claims memory transactions only for the ATU function. This means that the BARs and control signals from other internal functions are ignored, and any PCI status updates are limited to the ATU.

2.12.2 Outbound Transactions

ATU outbound transaction support is detailed in Section 2.2.2, "Outbound Transactions-Single Address Cycle (SAC) Internal Bus Transactions". The behavior is the same in both end point and central resource modes of operation.

2.12.3 PCI Reset (P_RSTOUT#)

When the Central Resource is enabled (**PCIX_EP#** = 1), the ATU controls generation of **P_RSTOUT#** for the PCI Bus segment attached to the ATU.

The ATU controls the PCI Reset signal (**P_RSTOUT#**) for the external agents supported by the 4138xx Central Resource. When the Central Resource is enabled, **P_RSTOUT#** remains asserted following the deassertion of the fundamental reset. **P_RSTOUT#** can be deasserted in the "PCI Configuration and Status Register - PCSR" on page 178.

2.12.4 PCI Clock Outputs (P_CLKOUT, P_CLKO[3:0])

The **P_CLKOUT**/**P_CLKO[3:0]** PCI clock outputs are available when operating in central resource mode and the primary clock source is the PCI Express* reference clock (**CLK_SRC_PCIE#** = 0). In this operating mode, the **P_CLKOUT** signal is connected to the **P_CLKIN** pin and trace match to the **P_CLKO[3:0]** signals.

Warning: When the **P_CLKIN** is the primary clock source (**CLK_SRC_PCIE#** = 1), the PCI Clock outputs are disabled and cannot be used as a clock source for any device.

2.12.5 External Clock Driver (CR_FREQ[1:0])

When the internal PCI Clock outputs are not sufficient, an external clock driver can be used to supply additional PCI Clocks. To facilitate the use of an external driver, the **CR_FREQ[1:0]** pins are driven based on the settings in the PCI-X capability field (bits 19:16) in the "PCI Configuration and Status Register - PCSR". These output pins can be connected to a PCI-X clock driver to select the desired bus frequency.

Table 22. CR_FREQ[1:0] Encoding

PCSR[19:16] (PCIX Init Pattern)	PCSR[10] (P_M66EN)	CR_FREQ[1:0]	Bus Frequency	Bus Mode
1111	0	11	33 MHz	Conventional PCI
1111	1	10	66 MHz	Conventional PCI
1110	-	10	66 MHz	PCI-X
1101	-	01	100 MHz	PCI-X
1100	-	00	133 MHz	PCI-X

2.12.6 Bus Mode and Frequency Initialization

The ATUs PCI Bus interface is capable of operating at a variety of frequencies, and in either Conventional PCI mode, or in PCI-X mode. The bus mode is established when coming out of the bus segment reset sequences. When the ATUs central resource is enabled, the resultant mode and frequency is dependent upon the device capabilities reported as well as any system specific loading information.

The ATU, as the central resource is the originating device for the PCI bus and as such, sets the bus mode and frequency when exiting out of the bus reset sequence. The two key components that factor into the resultant secondary bus mode and frequency are the PCI-X standard sampling of downstream device capabilities, and the system specific physical bus loading characteristics for which the *PCI-X Protocol Addendum to the PCI Local Bus Specification,* Revision 2.0 does not provide any standard means of reporting.

Downstream device capabilities are indicated by the values of **P_M66EN**, and **P_PCIXCAP** during **P_RST#** assertion.

Table 23. Device Mode/Frequency Capability Reporting

M66EN	PCIXCAP[a]	Conventional PCI Device Frequency Capability	PCI-X Device Frequency Capability
Ground	Ground	33 MHz	Not capable
8.2K pull-up[b]	Ground	66 MHz	Not capable
Ground	10K Pull-down	33 MHz	PCI-X 66 MHz
8.2K pull-up[b]	10K Pull-down	66 MHz	PCI-X 66 MHz
Ground	NC	33 MHz	PCI-X 133 MHz
8.2K pull-up[b]	NC	66 MHz	PCI-X 133 MHz
Ground	3.16K 1% pull-down	33 MHz	PCI-X 266 MHz
8.2K pull-up[b]	3.16K 1% pull-down	66 MHz	PCI-X 266 MHz

a. Resistor values are specified in Section 2.3.4, "PCIXCAP and MODE2 Connection," in *PCI-X Electrical and Mechanical Addendum to the PCI Local Bus Specification,* Revision 2.0.
b. M66EN may be pulled high on the motherboard.

Note: Knowledge of the device capabilities alone is insufficient information to robustly select the bus frequency. In order to be sure of what the bus operating frequency should be set to, knowledge of the bus layout (e.g., number of slots), is necessary.

When, for example, a 133 MHz PCI-X capable adapter was the sole occupant of a two slot segment, then it would be necessary to slow the bus to 100 MHz, even though the card reported it could operate at 133 MHz due to the additional electrical loading imposed by the two slot board and connector layout.

The ATU provides a strapping approach for reporting system specific bus loading information that is used in determining the maximum operating frequency of the secondary bus. The ATU considers this strap along with the device capabilities reported during **P_RST#** to determine the PCI bus's mode and frequency when emerging from **P_RST#**.

This strap, entitled PCI-X Bus 100 MHz Enable, is sampled on **PCIXM1_100#**, indicating to the ATU what to limit the bus frequency to a maximum of 100MHz. The value of this field is determined by the system designer, after having assessed the characteristics of the PCI bus system/adapter implementation.

Table 24 details the PCI bus frequency initialization as a function of the PCI Bus **PCIXM1_100#** and **PCIXM2_100#** reset strap, and the sampled secondary device capabilities when operating in PCI-X mode.

Table 24. PCI Bus Frequency Initialization[a]

P_PCIXCAP	P_MODE2	P_M66EN	PCIXM1_100#	PCIXM2_100#	PCI Bus Mode	PCI Bus Frequency	PCSR[19:16]
< 0.11VCC	-[b]	Ground	-	-	PCI	33 MHz	1111
< 0.11VCC	-	Not connected	-	-	PCI	66 MHz	1111
<0.6VCC & >0.11 VCC	GND	-	-	-	PCI-X Mode 1	66 MHz	1110
<0.6VCC & >0.11VCC	VCC	-	-	GND	PCI-X Mode 2	100 MHz (PCI-X 200 MHz)	0101
<0.6VCC & >0.11VCC	VCC	-	-	VCC	PCI-X Mode 2	133 MHz (PCI-X 266 MHz)	0100
<0.89VCC & >0.6VCC	-	-	-	-	PCI-X Mode 1	66 MHz	1110
>0.89VCC	-	-	GND	-	PCI-X Mode 1	100 MHz	1101
>0.89VCC	-	-	VCC	-	PCI-X Mode 1	133 MHz	1100

a. 4138xx does not support PCI-X 533 Mhz.
b. A '-' in the table indicates the value is a don't care for computing the bus mode/frequency. All signals must still be pulled to a valid logic level.

Note: The PCI Bus PCI-X 100 MHz Enable strapping feature enables implementations to force the PCI bus of 4138xx to operate at 100 MHz even with no standard provisions in the *PCI-X Protocol Addendum to the PCI Local Bus Specification,* Revision 2.0 for reporting device capability of 100 MHz operation.

When a card is plugged into a four slot PCI bus, a **P_PCIXCAP** pull-down (R1) strapping ensures that the bus runs at no greater than 66 MHz in PCI-X mode, and grounding **P_M66EN** ensures that the bus runs at no greater than 33 MHz in PCI, regardless of the reported downstream device capabilities.

When a card is plugged into a two slot secondary bus, the **PCIXM1_100#** pull-down strapping ensures that the bus runs at no greater than 100 MHz in PCI-X mode regardless of the reported downstream device capabilities.

When a card is plugged into a single slot secondary (i.e., a segment that should be able to run at 133 MHz), by strapping the **PCIXM1_100#** to 0b (as though it were a two slot configuration), the bus operates at 100 MHz maximum[8].

8. Adapters that report 133MHz PCI-X device capability with this **PCIXM1_100#** setting is limited to 100MHz operation.

Table 25 describes the bus mode and frequency initialization pattern that the ATU signals on its secondary bus when coming out of **P_RST#**, after having evaluated the above information.

Table 25. PCI-X Initialization Pattern

PERR#	DEVSEL#	STOP#	TRDY#	Mode[a]	Clock Period (Ns)		Clock Frequency (MHz)	
					Maximum	Minimum	Minimum	Maximum
Deasserted	Deasserted	Deasserted	Deasserted	PCI 33	60	30	16	33
				PCI 66	30	15	33	66
Deasserted	Deasserted	Deasserted	Asserted	PCI-X Mode 1	20	15	50	66
Deasserted	Deasserted	Asserted	Deasserted	PCI-X Mode 1	15	10	66	100
Deasserted	Deasserted	Asserted	Asserted	PCI-X Mode 1	10	7.5	100	133
Deasserted	Asserted	Deasserted	Deasserted	PCI-X Mode 1	Reserved			
Deasserted	Asserted	Deasserted	Asserted	PCI-X Mode 1				
Deasserted	Asserted	Asserted	Deasserted	PCI-X Mode 1				
Deasserted	Asserted	Asserted	Asserted	PCI-X Mode 1				
Asserted	Deasserted	Deasserted	Deasserted	PCI-X 266 (Mode 2)				
Asserted	Deasserted	Deasserted	Asserted	PCI-X 266[b] (Mode 2)	20	15	50	66
Asserted	Deasserted	Asserted	Deasserted	PCI-X 266 (Mode 2)	15	10	66	100
Asserted	Deasserted	Asserted	Asserted	PCI-X 266 (Mode 2)	10	7.5	100	133
Asserted	Asserted	Deasserted	Deasserted	PCI-X	Reserved			
Asserted	Asserted	Deasserted	Asserted	PCI-X				
Asserted	Asserted	Asserted	Deasserted	PCI-X				
Asserted	Asserted	Asserted	Asserted	PCI-X				

a. 4138xx supports neither PCI-X 533 Mode nor ECC in Mode 1.
b. The **P_CLK[3:0]** frequency and associated initialization pattern in PCI-X 266 mode for is selected in the "PCI Configuration and Status Register - PCSR" on page 178.

When operating in Central Resource Mode (PCIX_EP# = 1), the PCI-X initialization pattern is driven directly from the PCI-X capability field (bits 19:16) in the "PCI Configuration and Status Register - PCSR". The default value of this field is determined by the **PCIX_EP#**, **PCIXM1_100#**, **PCIXM2_100#** straps, as well as the **P_MODE2**, **P_PCIXCAP**, and **P_M66EN**, signals as described in Table 24, "PCI Bus Frequency Initialization" on page 135.

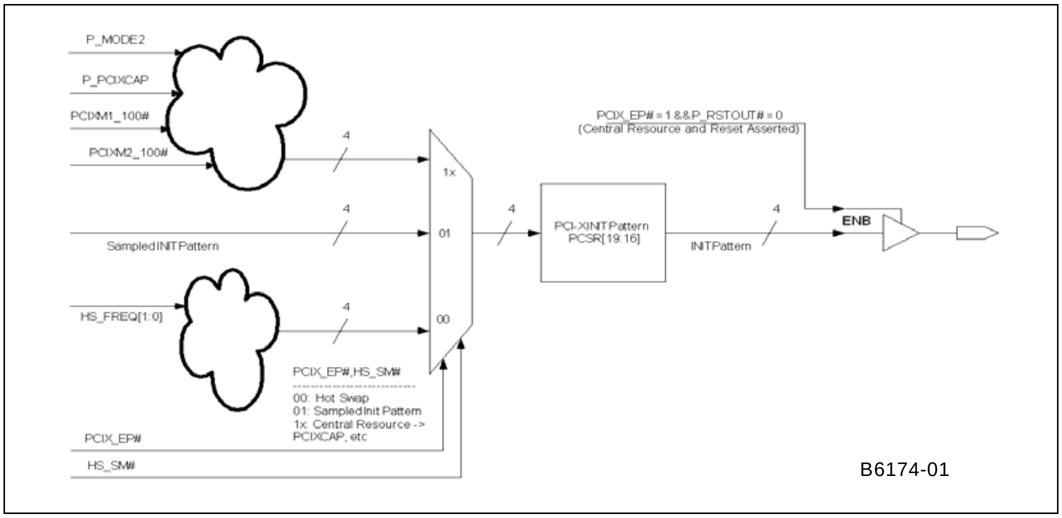
While **P_RSTOUT#** is asserted the initialization pattern is driven on the PCI bus. Software can override the default pattern by writing a new value to PCSR[19:16], before clearing the Central Resource PCI Bus Reset field (bit 21) in the "PCI Configuration and Status Register - PCSR".

When **P_RSTOUT#** is asserted any time after initial power on, software must enforce the 1ms reset assertion time (T_{rst}) required in the PCI specifications.

When operating as an endpoint (**PCIX_EP#** = 0), PCSR[19:16] contains the initialization pattern captured off the bus during **P_RST#**.

When operating as an endpoint in Hot-Swap mode (**HS_SM#** = 0), PCSR[19:16] is set based on the **HS_FREQ[1:0]** pins. For more details see Table 8, "HS_FREQ Encoding" on page 81.

Figure 14. PCI-X Initialization Pattern Setting / Drive



2.13 Embedded Bridge Functionality

Note: Not supported for 4138xx.

2.14 Register Definitions

Every PCI device implements its own separate configuration address space and configuration registers. The *PCI Local Bus Specification*, Revision 2.3 requires that configuration space be 256 bytes, and the first 64 bytes must adhere to a predefined header format.

Figure 15 defines the header format. Table 26 shows the PCI configuration registers, listed by internal bus address offset. Table 26 shows the entire ATU configuration space (including header and extended registers) and the corresponding section that describes each register. Note that all configuration read and write transactions is accepted on the internal bus as 32-bit transactions. Refer to Chapter 19.0, "Peripheral Registers".

2.14.1 PCI Configuration Registers

Figure 15. ATU Interface Configuration Header Format

ATU Device ID		Vendor ID		00H
Status		Command		04H
ATU Class Code			Revision ID	08H
ATUBISTR	Header Type	Latency Timer	Cacheline Size	0CH
Inbound ATU Base Address 0				10H
Inbound ATU Upper Base Address 0				14H
Inbound ATU Base Address 1				18H
Inbound ATU Upper Base Address 1				1CH
Inbound ATU Base Address 2				20H
Inbound ATU Upper Base Address 2				24H
Reserved				28H
ATU Subsystem ID		ATU Subsystem Vendor ID		2CH
Expansion ROM Base Address				30H
Reserved			Capabilities Pointer	34H
Reserved				38H
Maximum Latency	Minimum Grant	Interrupt Pin	Interrupt Line	3CH

The ATU is programmed via a Type 0 configuration command on the PCI interface. See Section 2.2.1.4, "Inbound Configuration Cycle Translation" on page 64. ATU configuration space is function number zero of the 4138xx single-function PCI device.

Beyond the required 64 byte header format, ATU configuration space implements extended register space in support of the units functionality. Refer to the *PCI Local Bus Specification*, Revision 2.3 for details on accessing and programming configuration register space.

The ATU unit includes six extended capability configuration spaces beginning at configuration offsets 90H, 98H, A0H, B0H, D0H, and E8H. The extended configuration spaces can be accessed by a device on the PCI interface through a mechanism defined in the *PCI Local Bus Specification*, Revision 2.3.

In the ATU Status Register (Section 2.14.6) the appropriate bit is set indicating that the Extended Capability Configuration space is supported. When this bit is read, the device can then read the Capabilities Pointer register (Section 2.14.22) to determine the configuration offset of the Extended Capabilities Configuration Header. The format of these headers are depicted in Figure 16, Figure 18, Figure 19, Figure 19 and Figure 21.

Figure 16. ATU Interface Extended Configuration Header Format (Power Management)

Power Management Capabilities	Next Item Pointer	Capability Identifier	98H
Reserved	Power Management Control/Status		9CH

B6326-01

The first byte at the Extended Configuration Offset 98H is the ATU Capability Identifier Register (Section). This identifies this Extended Configuration Header space as the type defined by the *PCI Bus Power Management Interface Specification*, Revision 1.1.

Following the Capability Identifier Register is the single byte Next Item Pointer Register (Section 2.14.49) which indicates the configuration offset of an additional Extended Capabilities Header, when supported. In the ATU, the Next Item Pointer Register is set to B0H indicating that there is an additional Extended Capabilities Headers supported in the ATUs configuration space.

To enable the *PCI Bus Power Management Interface Specification*, Revision 1.1 compliance support, the Power State Transition interrupt mask in bit 8 of the ATUIMR needs to be cleared. It is the configuration software's responsibility to properly enable and initialize the ATUs Power Management Interface before the Configuration Cycle Retry Bit in the Section 2.14.41, "PCI Configuration and Status Register - PCSR" on page 178 is cleared in order for the ATU to be *Advanced Configuration and Power Interface Specification,* Revision 2.0 compliant.

Figure 17. ATU Interface Extended Configuration Header Format (MSI-X Capability)

MSI-X Message Control	MSI-X Next Item Pointer	MSI-X Capability ID	B0H
MSI-X Table Offset		Table BIR	B4H
MSI-X PBA Offset		PBA BIR	B8H

B6327-01

Note: MSI-X Capability Registers are defined in Chapter 4.0, "Messaging Unit."

The first byte at the Extended Configuration Offset B0H is the MSI-X Capability Identifier Register (Section 4.7.26, "MSI-X Capability Identifier Register - MSI-X_Cap_ID"). This identifies this Extended Configuration Header space as the type defined by the *PCI Local Bus Specification*, Revision 2.3.

Following the Capability Identifier Register is the single byte Next Item Pointer Register (Section 4.7.27, "MSI-X Next Item Pointer Register - MSI-X_Next_Item_Ptr") which indicates the configuration offset of an additional Extended Capabilities Header, when supported. In the ATU, the Next Item Pointer Register is set to A0H indicating that there is an additional Extended Capabilities Headers supported in the ATUs configuration space.

Figure 18. ATU Interface Extended Configuration Header Format (MSI Capability)

MSI Message Control	MSI Next Item Pointer	MSI Capability ID	A0H
MSI Message Address			A4H
MSI Message Upper Address			A8H
Reserved	MSI Message Data		ACH

B6328-01

Note: MSI-X Capability Registers are defined in Chapter 4.0, "Messaging Unit."

The first byte at the Extended Configuration Offset A0H is in Section 4.7.20, "MSI Capability Identifier Register - Cap_ID". This identifies this Extended Configuration Header space as the type defined by the *PCI Local Bus Specification*, Revision 2.3.

Following the Capability Identifier Register is the single byte Section 4.7.21, "MSI Next Item Pointer Register - MSI_Next_Ptr", which indicates the configuration offset of an additional Extended Capabilities Header, when supported. In the ATU, the Next Item Pointer Register is set to D0H indicating that there is an additional Extended Capabilities Headers supported in the ATUs configuration space.

Figure 19. ATU Interface Extended Configuration Header Format (PCI-X Capability Type 1)

PCI-X Command	Next Item Pointer	PCI-X Capability ID	D0H
PCI-X Status			D4H
ECC Control and Status (Mode 2)			D8H
ECC First Address (Mode 2)			DCH
ECC Second Address (Mode 2)			E0H
ECC Attribute (Mode 2)			E4H

B6329-01

The first byte at the Extended Configuration Offset D0H is the PCI-X Capability Identifier Register (Section 2.14.53). This identifies this Extended Configuration Header space as the type defined by the *PCI-X Protocol Addendum to the PCI Local Bus Specification,* Revision 2.0. Indicated by bits 13:12 of the PCI-X Command Register, the 4138xx includes version 1 of the PCI-X Capabilities List Item indicating the 24 byte length of the item and the fact that ECC protection is provided in Mode 2 only.

Following the Capability Identifier Register is the single byte Next Item Pointer Register (Section 2.14.54) which indicates configuration offset of an additional Extended Capabilities Header, when supported. In the ATU, the Next Item Pointer Register is set to E8H indicating that there is an additional Extended Capabilities Headers supported in the ATUs configuration space.

Figure 20. ATU Extended Configuration Header Format (Compact PCI Hot-Swap Capability)

Reserved	Hot-Swap Ctrl/Status	Next Item Pointer	cPCI Capability ID	E8H

B6330-01

The first byte at the Extended Configuration Offset E8H is the Compact PCI Hot-Swap Capability Identifier Register (Section 2.14.61). This identifies this Extended Configuration Header space as the type defined by the *Compact PCI Hot-Swap Specification*, Revision 2.1.

Following the Capability Identifier Register is the single byte Next Item Pointer Register (Section 2.14.62) which indicates the configuration offset of an additional Extended Capabilities Header, when supported. In the ATU, the Next Item Pointer Register is set to 00H by default to indicate there are no additional Extended Capabilities Headers in the ATU configuration space. Software can set this pointer to 90H indicating there is an additional Extended Capabilities Headers supported in the ATUs configuration space.

Figure 21. ATU Interface Extended Configuration Header Format (VPD Capability)

VPD Address	Next Item Pointer	VPD Capability ID	90H
VPD Data			94H

B6331-01

The first byte at the Extended Configuration Offset 90H is the VPD Capability Identifier Register (Section 2.14.44). This identifies this Extended Configuration Header space as the type defined by the *PCI Local Bus Specification*, Revision 2.3.

Following the Capability Identifier Register is the single byte Next Item Pointer Register (Section 2.14.45) which indicates the configuration offset of an additional Extended Capabilities Header, when supported. In the ATU, the Next Item Pointer Register is set to 00H indicating that there are no additional Extended Capabilities Headers supported in the ATUs configuration space.

The following sections describe the ATU and Expansion ROM configuration registers. Configuration space consists of 8, 16, 24, and 32-bit registers arranged in a predefined format. Each register is described in functionality, access type (read/write, read/clear, read only) and reset default condition.

All registers adhere to the definitions found in the *PCI Local Bus Specification*, Revision 2.3 unless otherwise noted.

The Register Offset for PCI configuration registers is given in Table 26. As stated, a Type 0 configuration command on the bus, with an active **IDSEL** or a memory-mapped internal bus access required to read or write these registers. The ATU is always located at Function 0 in configuration space.

PCI Configuration command access to registers with offsets higher than 0FFH is not supported.

Note: In PCI-X Mode 2, the ATU handles all configuration command accesses to registers with offsets higher than 0FFH as Reserved.

Note: Each configuration register access type is individually defined for PCI configuration accesses. Some PCI read-only configuration registers have read/write capability from the 4138xx core CPU. See also Chapter 19.0, "Peripheral Registers".

2.14.2 Internal Bus Registers

A subset of the ATU registers are accessible through both inbound PCI configuration cycles and the 4138xx core CPU (Register offsets 000H through 0FFH). The balance of the registers are accessible only via the internal bus. Table 26, "Address Translation Unit Registers" on page 143 represents all of the ATU registers.

The location of these registers are specified as a relative offset to a 512KB aligned global PMMR offset. The default for the 512KB aligned offset is 0 FFD8 0000H defined by the PMMRBAR register. See also Chapter 19.0, "Peripheral Registers".

The Internal Bus Address Offset to PMMRBAR of any ATU Register can be derived by adding the 4 KB address aligned Internal Bus Memory Mapped Register Range Offset (Table 27, "ATU Internal Bus Memory Mapped Register Range Offsets" on page 146) to the Register Offset (Table 26, "Address Translation Unit Registers" on page 143)

For example, when **INTERFACE_SEL_PCIX#** and **CONTROLLER_ONLY#** are both asserted, the offset to PMMRBAR of the "ATU Command Register - ATUCMD" would be (4 C000H+004H) or 4 C004H.

Note: The 4 KB Address Aligned Range Offset can be different depending on two configuration straps as described in Table 27.

Table 26. Address Translation Unit Registers (Sheet 1 of 3)

Register Offset	ATU Register Section, Name, Page
000H	Section 2.14.3, "ATU Vendor ID Register - ATUVID" on page 147
002H	Section 2.14.4, "ATU Device ID Register - ATUDID" on page 147
004H	Section 2.14.5, "ATU Command Register - ATUCMD" on page 148
006H	Section 2.14.6, "ATU Status Register - ATUSR" on page 149
008H	Section 2.14.7, "ATU Revision ID Register - ATURID" on page 151
009H	Section 2.14.8, "ATU Class Code Register - ATUCCR" on page 151
00CH	Section 2.14.9, "ATU Cacheline Size Register - ATUCLSR" on page 152
00DH	Section 2.14.10, "ATU Latency Timer Register - ATULT" on page 152
00EH	Section 2.14.11, "ATU Header Type Register - ATUHTR" on page 153
00FH	Section 2.14.12, "ATU BIST Register - ATUBISTR" on page 154
010H	Section 2.14.13, "Inbound ATU Base Address Register 0 - IABAR0" on page 155
014H	Section 2.14.14, "Inbound ATU Upper Base Address Register 0 - IAUBAR0" on page 156
018H	Section 2.14.15, "Inbound ATU Base Address Register 1 - IABAR1" on page 157
01CH	Section 2.14.16, "Inbound ATU Upper Base Address Register 1 - IAUBAR1" on page 158
020H	Section 2.14.17, "Inbound ATU Base Address Register 2 - IABAR2" on page 159
024H	Section 2.14.18, "Inbound ATU Upper Base Address Register 2 - IAUBAR2" on page 160
02CH	Section 2.14.19, "ATU Subsystem Vendor ID Register - ASVIR" on page 161
02EH	Section 2.14.20, "ATU Subsystem ID Register - ASIR" on page 161
030H	Section 2.14.21, "Expansion ROM Base Address Register - ERBAR" on page 162
034H	Section 2.14.22, "ATU Capabilities Pointer Register - ATU_Cap_Ptr" on page 163
03CH	Section 2.14.24, "ATU Interrupt Line Register - ATUILR" on page 166
03DH	Section 2.14.25, "ATU Interrupt Pin Register - ATUIPR" on page 167
03EH	Section 2.14.26, "ATU Minimum Grant Register - ATUMGNT" on page 167
03FH	Section 2.14.27, "ATU Maximum Latency Register - ATUMLAT" on page 168
040H	Section 2.14.28, "Inbound ATU Limit Register 0 - IALR0" on page 169
044H	Section 2.14.29, "Inbound ATU Translate Value Register 0 - IATVR0" on page 170

Table 26. Address Translation Unit Registers (Sheet 2 of 3)

Register Offset	ATU Register Section, Name, Page
048H	Section 2.14.30, "Inbound ATU Upper Translate Value Register 0 - IAUTVR0" on page 170
04CH	Section 2.14.31, "Inbound ATU Limit Register 1 - IALR1" on page 171
050H	Section 2.14.32, "Inbound ATU Translate Value Register 1 - IATVR1" on page 172
054H	Section 2.14.33, "Inbound ATU Upper Translate Value Register 1 - IAUTVR1" on page 172
058H	Section 2.14.34, "Inbound ATU Limit Register 2 - IALR2" on page 173
05CH	Section 2.14.35, "Inbound ATU Translate Value Register 2 - IATVR2" on page 174
060H	Section 2.14.36, "Inbound ATU Upper Translate Value Register 2 - IAUTVR2" on page 174
064H	Section 2.14.37, "Expansion ROM Limit Register - ERLR" on page 175
068H	Section 2.14.38, "Expansion ROM Translate Value Register - ERTVR" on page 176
06CH	Section 2.14.39, "Expansion ROM Upper Translate Value Register - ERUTVR" on page 176
070H	Section 2.14.40, "ATU Configuration Register - ATUCR" on page 177
074H	Section 2.14.41, "PCI Configuration and Status Register - PCSR" on page 178
078H	Section 2.14.42, "ATU Interrupt Status Register - ATUISR" on page 181
07CH	Section 2.14.43, "ATU Interrupt Mask Register - ATUIMR" on page 183
080H — 08FH	Reserved
090H	Section 2.14.44, "VPD Capability Identifier Register - VPD_Cap_ID" on page 185
091H	Section 2.14.45, "VPD Next Item Pointer Register - VPD_Next_Item_Ptr" on page 185
092H	Section 2.14.46, "VPD Address Register - VPDAR" on page 186
094H	Section 2.14.47, "VPD Data Register - VPDDR" on page 186
098H	Section 2.14.48, "PM Capability Identifier Register - PM_Cap_ID" on page 187
099H	Section 2.14.49, "PM Next Item Pointer Register - PM_Next_Item_Ptr" on page 187
09AH	Section 2.14.50, "ATU Power Management Capabilities Register - APMCR" on page 188
09CH	Section 2.14.51, "ATU Power Management Control/Status Register - APMCSR" on page 189
0A0H	Section 4.7.20, "MSI Capability Identifier Register - Cap_ID" on page 429[a]
0A1H	Section 4.7.21, "MSI Next Item Pointer Register - MSI_Next_Ptr" on page 430
0A2H	Section 4.7.22, "Message Control Register - Message_Control" on page 431
0A4H	Section 4.7.23, "Message Address Register - Message_Address" on page 432
0A8H	Section 4.7.24, "Message Upper Address Register - Message_Upper_Address" on page 433
0ACH	Section 4.7.25, "Message Data Register- Message_Data" on page 434
0AEH	Reserved
0B0H	Section 4.7.26, "MSI-X Capability Identifier Register - MSI-X_Cap_ID" on page 435
0B1H	Section 4.7.27, "MSI-X Next Item Pointer Register - MSI-X_Next_Item_Ptr" on page 436
0B2H	Section 4.7.28, "MSI-X Message Control Register - MSI-X_MCR" on page 437
0B4H	Section 4.7.29, "MSI-X Table Offset Register — MSI-X_Table_Offset" on page 438
0B8H	Section 4.7.30, "MSI-X Pending Bit Array Offset Register - MSI-X_PBA_Offset" on page 439
0BCH — 0C8H	Reserved
0CCH	Section 2.14.52, "ATU Scratch Pad Register - ATUSPR" on page 190
0D0H	Section 2.14.53, "PCI-X Capability Identifier Register - PCI-X_Cap_ID" on page 190
0D1H	Section 2.14.54, "PCI-X Next Item Pointer Register - PCI-X_Next_Item_Ptr" on page 191
0D2H	Section 2.14.55, "PCI-X Command Register - PCIXCMD" on page 191
0D4H	Section 2.14.56, "PCI-X Status Register - PCIXSR" on page 193
0D8H	Section 2.14.57, "ECC Control and Status Register - ECCCSR" on page 195
0DCH	Section 2.14.58, "ECC First Address Register - ECCFAR" on page 198
0E0H	Section 2.14.59, "ECC Second Address Register - ECCSAR" on page 199
0E4H	Section 2.14.60, "ECC Attribute Register - ECCAR" on page 200

Table 26. Address Translation Unit Registers (Sheet 3 of 3)

Register Offset	ATU Register Section, Name, Page
0E8H	Section 2.14.61, "CompactPCI Hot-Swap Capability ID Register" on page 200
0E9H	Section 2.14.62, "Offset EDh: HS_NXTP - Next Item Pointer" on page 201
0EAH	Section 2.14.63, "HS_CNTRL - Hot-Swap Control/Status Register" on page 202
0EBH	Reserved
0ECH — 0FFH	Reserved
100H — 1FFH	Reserved
200H	Section 2.14.64, "Inbound ATU Base Address Register 3 - IABAR3" on page 204
204H	Section 2.14.65, "Inbound ATU Upper Base Address Register 3 - IAUBAR3" on page 205
208H	Section 2.14.66, "Inbound ATU Limit Register 3 - IALR3" on page 206
20CH	Section 2.14.67, "Inbound ATU Translate Value Register 3 - IATVR3" on page 207
210H	Section 2.14.68, "Inbound ATU Upper Translate Value Register 3 - IAUTVR3" on page 207
214H — 2FCH	Reserved
300H	Section 2.14.69, "Outbound I/O Base Address Register - OIOBAR" on page 208
304H	Section 2.14.70, "Outbound I/O Window Translate Value Register - OIOWTVR" on page 209
308H	Section 2.14.71, "Outbound Upper Memory Window Base Address Register 0 - OUMBAR0" on page 210
30CH	Section 2.14.72, "Outbound Upper 32-bit Memory Window Translate Value Register 0 - OUMWTVR0" on page 211
310H	Section 2.14.73, "Outbound Upper Memory Window Base Address Register 1 - OUMBAR1" on page 212
314H	Section 2.14.74, "Outbound Upper 32-bit Memory Window Translate Value Register 1 - OUMWTVR1" on page 213
318H	Section 2.14.75, "Outbound Upper Memory Window Base Address Register 2 - OUMBAR2" on page 214
31CH	Section 2.14.76, "Outbound Upper 32-bit Memory Window Translate Value Register 2 - OUMWTVR2" on page 215
320H	Section 2.14.77, "Outbound Upper Memory Window Base Address Register 3 - OUMBAR3" on page 216
324H	Section 2.14.78, "Outbound Upper 32-bit Memory Window Translate Value Register 3 - OUMWTVR3" on page 217
328H	Reserved
32CH	Reserved
330H	Section 2.14.79, "Outbound Configuration Cycle Address Register - OCCAR" on page 218
334H	Section 2.14.80, "Outbound Configuration Cycle Data Register - OCCDR" on page 219
338H	Section 2.14.81, "Outbound Configuration Cycle Function Number - OCCFN" on page 219
33CH — 37CH	Reserved
380H	Section 2.14.82, "PCI Interface Error Control and Status Register - PIECSR" on page 220
384H	Section 2.14.83, "PCI Interface Error Address Register - PCIEAR" on page 221
388H	Section 2.14.84, "PCI Interface Error Upper Address Register - PCIEUAR" on page 222
38CH	Section 2.14.85, "PCI Interface Error Context Address Register — PCIECAR" on page 223
394H	Section 2.14.86, "Internal Arbiter Control Register - IACR" on page 224
398H	Section 2.14.87, "Multi-Transaction Timer - MTT" on page 225
39CH	Reserved.

a. All MSI and MSI-X capability register descriptions are in Section 4.7, "Register Definitions" on page 410 of Chapter 4.0, "Messaging Unit."

Table 27. ATU Internal Bus Memory Mapped Register Range Offsets

INTERFACE_SEL_PCIX#	CONTROLLER_ONLY#	Internal Bus MMR Address Range Offset (Relative to PMMRBAR)
Asserted (0)	Deasserted (1)	+4 8000H
Asserted (0)	Asserted (0)	+4 C000H
Deasserted (1)	Asserted (0)	+4 C000H
Deasserted (1)	Deasserted (1)	+4 D000H

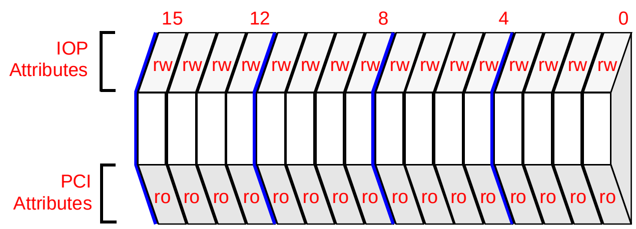
Table 28. PCI-X Pad Registers

Register Offset	Section, Register Name - Acronym (Page)
2100H	Section 2.14.88, "PCIX RCOMP Control Register — PRCR" on page 226
2104H	Section 2.14.89, "PCIX Pad ODT Drive Strength Manual Override Values Registers — PPODSMOVR" on page 227
2108H	Section 2.14.90, "PCIX PAD DRIVE STRENGTH Manual Override Values Register (3.3 V/1.5 V Switch Supply Voltage) — PPDSMOVR3.3_1.5" on page 228
210CH	Section 2.14.91, "PCIX PAD DRIVE STRENGTH Manual Override Values Register (3.3 V Dedicated Supply Voltage) — PPDSMOVR3.3" on page 229

2.14.3 ATU Vendor ID Register - ATUVID

ATU Vendor ID Register bits adhere to the definitions in the *PCI Local Bus Specification*, Revision 2.3.

Table 29. ATU Vendor ID Register - ATUVID



Register Offset
+000H

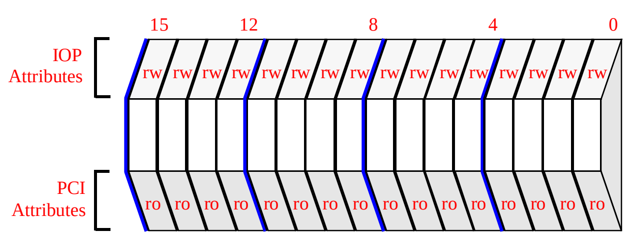
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
15:00	8086H	ATU Vendor ID - This is a 16-bit value assigned to Intel. This register, combined with the DID, uniquely identify the PCI device. Access type is Read/Write to allow the 4138xx to configure the register as a different vendor ID to simulate the interface of a standard mechanism currently used by existing application software.

2.14.4 ATU Device ID Register - ATUDID

ATU Device ID Register bits adhere to the definitions in the *PCI Local Bus Specification*, Revision 2.3.

Table 30. ATU Device ID Register - ATUDID



Register Offset
+002H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

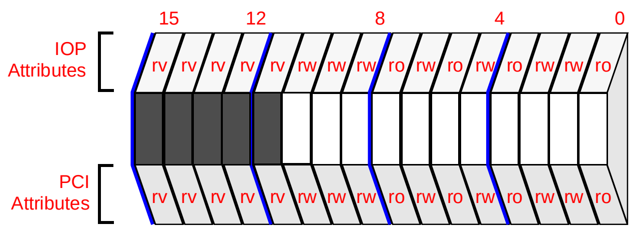
Bit	Default	Description
15:00	Product Type Dependent[a]	ATU Device ID - This is a 16-bit value assigned to the ATU. This ID, combined with the VID, uniquely identify any PCI device.

a. See Intel® 413808 and 413812 I/O Controllers in TPER Mode *Specification Update.*

2.14.5 ATU Command Register - ATUCMD

ATU Command Register bits adhere to the definitions in the *PCI Local Bus Specification*, Revision 2.3 and in most cases, affect the behavior of the PCI ATU and devices on the PCI bus.

Table 31. ATU Command Register - ATUCMD



Register Offset
+004H

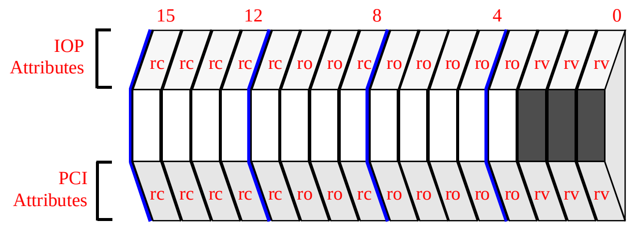
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
15:11	00000_2	Reserved
10	0_2	Interrupt Disable - Setting this bit disables the ATU function of 4138xx from asserting the **INT[A:D]#** signals. Clearing this bit enables the assertion of the **INT[A:D]#** signals by the ATU function of 4138xx.
09	0_2	Fast Back to Back Enable - When cleared, the ATU interface is not allowed to generate fast back-to-back cycles on its bus. Ignored when operating in the PCI-X mode.
08	0_2	**SERR#** Enable - When cleared, the ATU interface is not allowed to assert **SERR#** on the PCI interface.
07	0_2	Hard-wired 0
06	0_2	Parity Error Response - When set, the ATU takes normal action when a parity (or uncorrectable ECC) error is detected. When cleared, parity (or uncorrectable ECC) checking is disabled. ***Note:*** When the bit is cleared in PCI-X Mode 2, the 4138xx records all ECC errors in the ECC Control and Status, ECC First Address, ECC Second Address, and ECC Attribute registers but in all other respects treats the transaction as though it had no error. Correctable ECC errors are corrected independent of the state of this bit.
05	0_2	VGA Palette Snoop Enable - The ATU interface does not support I/O writes and therefore, does not perform VGA palette snooping.
04	0_2	Memory Write and Invalidate Enable - When set, ATU may generate MWI commands. When clear, ATU use Memory Write commands instead of MWI. Ignored when operating in the PCI-X mode.
03	0_2	Special Cycle Enable - The ATU interface does not respond to special cycle commands in any way. Not implemented and a reserved bit field.
02	0_2	Bus Master Enable - The ATU interface can act as a master on the PCI bus. When cleared, disables the device from generating PCI accesses. When set, allows the device to behave as a PCI bus master. When operating in the PCI-X mode, ATU initiates a split completion transaction regardless of the state of this bit.
01	0_2	Memory Enable - Controls the ATU interface's response to PCI memory addresses. When cleared, the ATU interface does not respond to any memory access on the PCI bus.
00	0_2	I/O Space Enable - Controls the ATU interface response to I/O transactions. The ATU does not support I/O transactions. This bit is Read Only.

2.14.6 ATU Status Register - ATUSR

The ATU Status Register bits adhere to the *PCI Local Bus Specification*, Revision 2.3 definitions. The *read/clear* bits can only be set by internal hardware and cleared by either a reset condition or by writing a 1_2 to the register.

Table 32. ATU Status Register - ATUSR (Sheet 1 of 2)

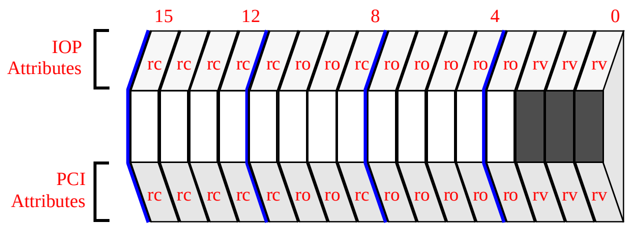


Register Offset
+006H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
15	0_2	Detected Parity Error - set when a parity error is detected in data received by the ATU on the PCI bus even when the ATUCMD register's Parity Error Response bit is cleared. Set under the following conditions: • Uncorrectable Write Data Error when the ATU is a target (inbound write). • Uncorrectable Read Data Error when the ATU is a requester (outbound read). • Any Uncorrectable Address or Attribute (PCI-X Only) Error on the Bus (including one generated by the ATU).
14	0_2	**SERR#** Asserted - set when **SERR#** is asserted on the PCI bus by the ATU.
13	0_2	Master Abort - set when a transaction initiated by the ATU PCI master interface, ends in a Master-Abort or when the ATU receives a Master Abort Split Completion Error Message in PCI-X mode.
12	0_2	Target Abort (master) - set when a transaction initiated by the ATU PCI master interface, ends in a target abort or when the ATU receives a Target Abort Split Completion Error Message in PCI-X mode.
11	0_2	Target Abort (target) - set when the ATU interface, acting as a target, terminates the transaction on the PCI bus with a target abort.
10:09	01_2	DEVSEL# Timing - These bits are read-only and define the slowest DEVSEL# timing for a target device in Conventional PCI Mode regardless of the operating mode (except configuration accesses). 00_2 = Fast 01_2 = Medium 10_2 = Slow 11_2 = Reserved The ATU interface uses Medium timing.
08	0_2	Master Parity Error - The ATU interface sets this bit under the following conditions: • The ATU asserted **PERR#** itself or the ATU observed **PERR#** asserted. • And the ATU acted as the requester for the operation in which the error occurred. • And the ATUCMD register's Parity Error Response bit is set • Or (PCI-X Mode Only) the ATU received an Uncorrectable Write Data Error Message • And the ATUCMD register's Parity Error Response bit is set
07	1_2 (Conventional mode) 0_2 (PCI-X mode)	Fast Back-to-Back - The ATU interface is capable of accepting fast back-to-back transactions in Conventional PCI mode when the transactions are not to the same target. Since fast back-to-back transactions do not exist in PCI-X mode, this bit is forced to 0 in the PCI-X mode.
06	0_2	UDF Supported - User Definable Features are not supported
05	1_2	66 MHz. Capable - 66 MHz operation is supported.
04	1_2	**Capabilities** - When set, this function implements extended capabilities.

Table 32. ATU Status Register - ATUSR (Sheet 2 of 2)



Register Offset
+006H

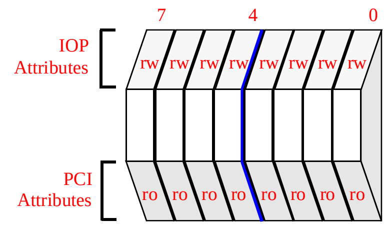
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
03	0_2	Interrupt Status - This bit reflects the state of the interrupt in the 4138xx ATU function. Only when the Interrupt Disable bit in the ATUCMD is a 0 and this Interrupt Status bit is a 1, are any of the 4138xx's **INT[A:D]#** signals asserted by the ATU function. ***Note:*** Setting the Interrupt Disable bit to a 1 in (bit 10 of ATUCMD) has no effect on the state of this bit.
02:00	000_2	Reserved

2.14.7 ATU Revision ID Register - ATURID

Revision ID Register bit definitions adhere to *PCI Local Bus Specification*, Revision 2.3.

Table 33. ATU Revision ID Register - ATURID



Register Offset
+008H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

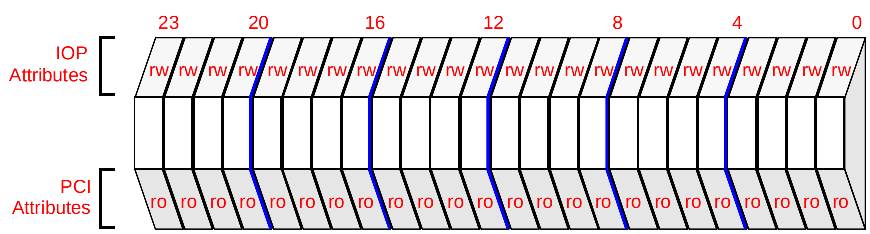
Bit	Default	Description
07:00	xxH[a]	ATU Revision - identifies the revision number.

a. See *Intel® 81348 I/O Processor Specification Update*.

2.14.8 ATU Class Code Register - ATUCCR

Class Code Register bit definitions adhere to *PCI Local Bus Specification*, Revision 2.3. Auto configuration software reads this register to determine the PCI device function.

Table 34. ATU Class Code Register - ATUCCR



Register Offset
+009H

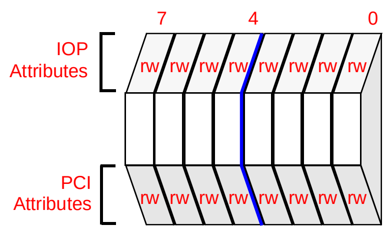
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
23:16	05H	Base Class - Memory Controller
15:08	80H	Sub Class - Other Memory Controller
07:00	00H	Programming Interface - None defined

2.14.9 ATU Cacheline Size Register - ATUCLSR

Cacheline Size Register bit definitions adhere to *PCI Local Bus Specification*, Revision 2.3. This register is programmed with the system cacheline size in DWORDs (32-bit words). Cacheline Size is restricted to either 0, 8 or 16 DWORDs; the ATU interprets any other value as "0".

Table 35. ATU Cacheline Size Register - ATUCLSR



Register Offset
+00CH

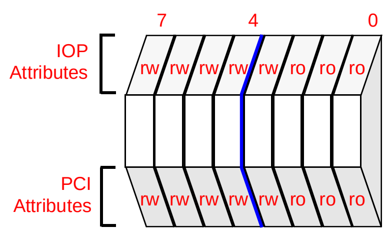
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
07:00	00H	ATU Cacheline Size - specifies the system cacheline size in DWORDs. Cacheline size is restricted to either 0, 8 or 16 DWORDs.

2.14.10 ATU Latency Timer Register - ATULT

ATU Latency Timer Register bit definitions apply to the PCI interface.

Table 36. ATU Latency Timer Register - ATULT



Register Offset
+00DH

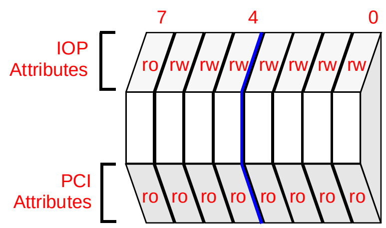
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
07:03	00000_2 (for Conventional mode) 01000_2 (for PCI-X mode)	Programmable Latency Timer - This field varies the latency timer for the interface from 0 to 248 clocks. The default value is 0 clocks for Conventional PCI mode, and 64 clocks for PCI-X mode.
02:00	000_2	Latency Timer Granularity - These Bits are read only giving a programmable granularity of 8 clocks for the latency timer.

2.14.11 ATU Header Type Register - ATUHTR

Header Type Register bit definitions adhere to *PCI Local Bus Specification*, Revision 2.3. This register indicates the layout of ATU configuration space bytes 10H to 3FH. The MSB indicates whether or not the device is multi-function.

Table 37. ATU Header Type Register - ATUHTR



Register Offset
+00EH

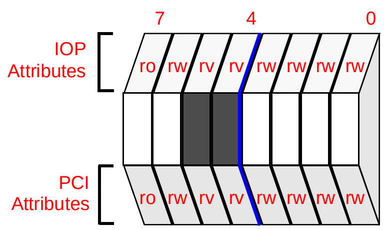
Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
07	End Point: **DF_SEL[2:0]** != "000" Central Resource: 0	Single Function/Multi-Function Device - Identifies the 4138xx as a single-function or multi-function PCI device depending on the setting of the **DF_SEL[2:0]** strap during **P_RST#** assertion. ***Note:*** The 4138xx can be configured as a single-function device (ATU only) or a multi-function device (ATU and storage controller) for split driver support. As a Central Resource, this bit is always '0'.
06:00	0000000_2	PCI Header Type - This bit field indicates the type of PCI header implemented. The ATU interface header conforms to *PCI Local Bus Specification*, Revision 2.3.

2.14.12 ATU BIST Register - ATUBISTR

The ATU BIST Register controls the functions the Intel XScale® processor performs when BIST is initiated. This register is the interface between the host processor requesting BIST functions and the 4138xx replying with the results from the software implementation of the BIST functionality.

Table 38. ATU BIST Register - ATUBISTR



Register Offset
+00FH

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
07	0_2	BIST Capable - This bit value is always equal to the ATUCR ATU BIST Interrupt Enable bit. See Section 2.14.40, "ATU Configuration Register - ATUCR" on page 177.
06	0_2	Start BIST - When the ATUCR BIST Interrupt Enable bit is set: Setting this bit generates an interrupt to the Intel XScale® processor to perform a software BIST function. The Intel XScale® processor clears this bit when the BIST software has completed with the BIST results found in ATUBISTR register bits [3:0]. When the ATUCR BIST Interrupt Enable bit is clear: Setting this bit does not generate an interrupt to the Intel XScale® processor and no BIST functions is performed. The Intel XScale® processor does not clear this bit.
05:04	00_2	Reserved
03:00	0000_2	BIST Completion Code - when the ATUCR BIST Interrupt Enable bit is set and the ATUBISTR Start BIST bit is set (bit 6): The Intel XScale® processor places the results of the software BIST in these bits. A nonzero value indicates a device-specific error.

2.14.13 Inbound ATU Base Address Register 0 - IABAR0

The Inbound ATU Base Address Register 0 (IABAR0) together with the Inbound ATU Upper Base Address Register 0 (IAUBAR0) defines the block of memory addresses where the inbound translation window 0 begins. The inbound ATU decodes and forwards the bus request to the 4138xx internal bus with a translated address to map into 4138xx local memory. The IABAR0 and IAUBAR0 define the base address and describes the required memory block size; see Section 2.14.23, "Determining Block Sizes for Base Address Registers" on page 164. Bits 31 through 12 of the IABAR0 is either read/write bits or read only with a value of 0 depending on the value located within the IALR0. This configuration allows the IABAR0 to be programmed per *PCI Local Bus Specification*, Revision 2.3.

By default the first 8 Kbytes of memory defined by the IABAR0, IAUBAR0 and the IALR0 is reserved for the Messaging Unit.

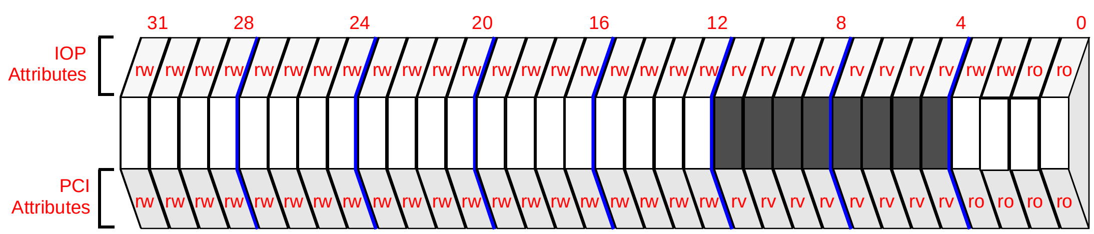
The programmed value within the base address register must comply with the PCI programming requirements for address alignment. Refer to the *PCI Local Bus Specification*, Revision 2.3 for additional information on programming base address registers.

Warning: When IALR0 is cleared prior to host configuration, the user should also clear the Prefetchable Indicator and the Type Indicator. Assuming IALR0 is not cleared:

a. Since non prefetchable memory windows can never be placed above the 4 Gbyte address boundary, when the Prefetchable Indicator is cleared prior to host configuration, the user should also set the Type Indicator for 32 bit addressability.

b. For compliance to the *PCI-X Protocol Addendum to the PCI Local Bus Specification*, Revision 2.0, when the Prefetchable Indicator is set prior to host configuration, the user should also set the Type Indicator for 64 bit addressability. This is the default for IABAR0.

Table 39. Inbound ATU Base Address Register 0 - IABAR0



Register Offset
+010H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:12	00000H	Translation Base Address 0 - These bits define the actual location the translation function is to respond to when addressed from the PCI bus.
11:04	00H	Reserved.
03	1_2	Prefetchable Indicator - When set, defines the memory space as prefetchable.
02:01	10_2	Type Indicator - Defines the width of the addressability for this memory window: 00 - Memory Window is locatable anywhere in 32 bit address space 10 - Memory Window is locatable anywhere in 64 bit address space
00	0_2	Memory Space Indicator - This bit field describes memory or I/O space base address. The ATU does not occupy I/O space, thus this bit must be zero.

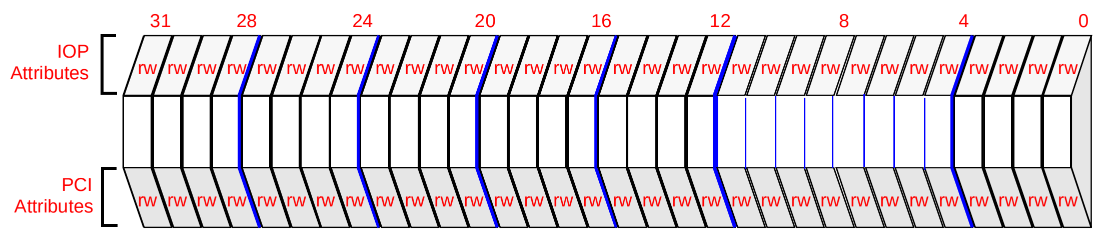
2.14.14 Inbound ATU Upper Base Address Register 0 - IAUBAR0

This register contains the upper base address when decoding PCI addresses beyond 4 GBytes. Together with the Translation Base Address this register defines the actual location the translation function is to respond to when addressed from the PCI bus for addresses > 4GBytes (for DACs).

The programmed value within the base address register must comply with the PCI programming requirements for address alignment. Refer to the *PCI Local Bus Specification*, Revision 2.3 for additional information on programming base address registers.

Note: When the Type Indicator of IABAR0 is set to indicate 32 bit addressability, the IAUBAR0 register attributes are read-only. Prior to changing the Type Indicator in the IABAR0 to support 32-bit addressability, the IAUBAR0 must be written with zero unless it already contains zero. Zero is the default value for the IAUBAR0.

Table 40. Inbound ATU Upper Base Address Register 0 - IAUBAR0



Register Offset
+014H

Attribute Legend:	RW = Read/Write
RV = Reserved	RC = Read Clear
PR = Preserved	RO = Read Only
RS = Read/Set	NA = Not Accessible

Bit	Default	Description
31:0	00000H	Translation Upper Base Address 0 - Together with the Translation Base Address 0 these bits define the actual location the translation function is to respond to when addressed from the PCI bus for addresses > 4GBytes.

2.14.15 Inbound ATU Base Address Register 1 - IABAR1

The Inbound ATU Base Address Register 1 (IABAR1) together with the Inbound ATU Upper Base Address Register 1 (IAUBAR1) defines the block of memory addresses where the inbound translation window 1 begins. The inbound ATU decodes and forwards the bus request to the 4138xx internal bus with a translated address to map into 4138xx local memory. The IABAR1 and IAUBAR1 define the base address and describes the required memory block size; see Section 2.14.23, "Determining Block Sizes for Base Address Registers" on page 164. Bits 31 through 12 of the IABAR1 is either read/write bits or read only with a value of 0 depending on the value located within the IALR1. This configuration allows the IABAR1 to be programmed per *PCI Local Bus Specification*, Revision 2.3.

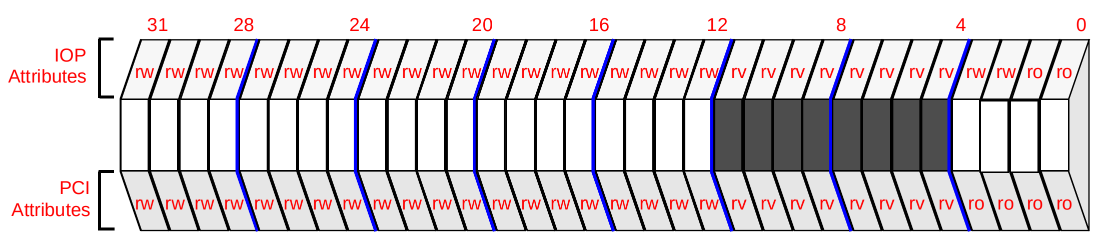
The programmed value within the base address register must comply with the PCI programming requirements for address alignment. Refer to the *PCI Local Bus Specification*, Revision 2.3 for additional information on programming base address registers.

Warning: When IALR1 is cleared prior to host configuration, the user should also clear the Prefetchable Indicator and the Type Indicator. Assuming IALR1 is not cleared:

c. Since non prefetchable memory windows can never be placed above the 4 Gbyte address boundary, when the Prefetchable Indicator is cleared prior to host configuration, the user should also set the Type Indicator for 32 bit addressability.

d. For compliance to the *PCI-X Protocol Addendum to the PCI Local Bus Specification*, Revision 2.0, when the Prefetchable Indicator is set prior to host configuration, the user should also set the Type Indicator for 64 bit addressability. This is the default for IABAR1.

Table 41. Inbound ATU Base Address Register 1 - IABAR1



Register Offset
+018H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:12	00000H	Translation Base Address 1 - These bits define the actual location of window 1 on the PCI bus.
11:04	00H	Reserved.
03	0_2	Prefetchable Indicator - When set, defines the memory space as prefetchable.
02:01	00_2	Type Indicator - Defines the width of the addressability for this memory window: 00 - Memory Window is locatable anywhere in 32 bit address space 10 - Memory Window is locatable anywhere in 64 bit address space
00	0_2	Memory Space Indicator - This bit field describes memory or I/O space base address. The ATU does not occupy I/O space, thus this bit must be zero.

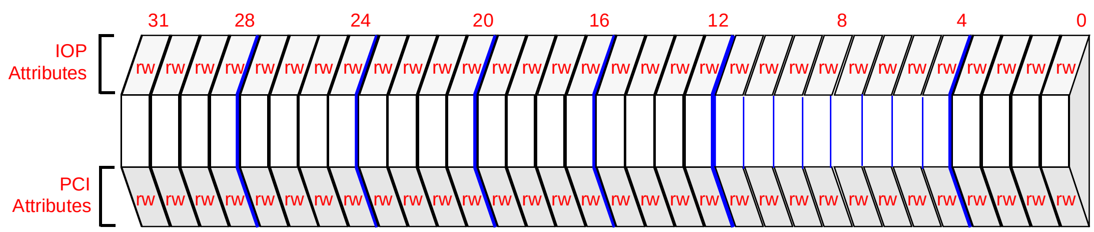
2.14.16 Inbound ATU Upper Base Address Register 1 - IAUBAR1

This register contains the upper base address when decoding PCI addresses beyond 4 GBytes. Together with the Translation Base Address this register defines the actual location the translation function is to respond to when addressed from the PCI bus for addresses > 4GBytes (for DACs).

The programmed value within the base address register must comply with the PCI programming requirements for address alignment. Refer to the *PCI Local Bus Specification*, Revision 2.3 for additional information on programming base address registers.

Note: When the Type indicator of IABAR1 is set to indicate 32 bit addressability, the IAUBAR1 register attributes are read-only. By default the IAUBAR1 register has read-only attributes. Prior to changing the Type Indicator in the IABAR1 to support 32-bit addressability, the IAUBAR1 must be written with zero unless it already contains zero. Zero is the default value for IAUBAR1.

Table 42. Inbound ATU Upper Base Address Register 1 - IAUBAR1



Register Offset
+01CH

Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
31:0	00000H	Translation Upper Base Address 1 - Together with the Translation Base Address 1 these bits define the actual location for this memory window on the PCI bus for addresses > 4GBytes.

2.14.17 Inbound ATU Base Address Register 2 - IABAR2

The Inbound ATU Base Address Register 2 (IABAR2) together with the Inbound ATU Upper Base Address Register 2 (IAUBAR2) defines the block of memory space addresses where the inbound translation window 2 begins. The inbound ATU decodes and forwards the bus request to the 4138xx internal bus with a translated address to map into 4138xx local memory. The IABAR2 and IAUBAR2 (Memory Space only) define the base address and describes the required address block size; see Section 2.14.23, "Determining Block Sizes for Base Address Registers" on page 164. Bits 31 through 8 of the IABAR2 is either read/write bits or read only with a value of 0 depending on the value located within the IALR2. This configuration allows the IABAR2 to be programmed per *PCI Local Bus Specification*, Revision 2.3.

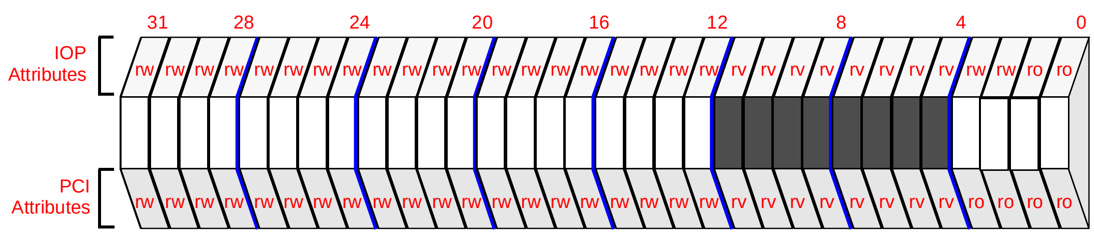
The programmed value within the base address register must comply with the PCI programming requirements for address alignment. Refer to the *PCI Local Bus Specification*, Revision 2.3 for additional information on programming base address registers.

Warning: When IALR2 is cleared prior to host configuration, the user should also clear the Prefetchable Indicator and the Type Indicator. Assuming IALR2 is not cleared:

e. Since non prefetchable memory windows can never be placed above the 4 Gbyte address boundary, when the Prefetchable Indicator is cleared prior to host configuration, the user should also set the Type Indicator for 32 bit addressability.

f. For compliance to the *PCI-X Protocol Addendum to the PCI Local Bus Specification*, Revision 2.0, when the Prefetchable Indicator is set prior to host configuration, the user should also set the Type Indicator for 64 bit addressability. This is the default for IABAR2.

Table 43. Inbound ATU Base Address Register 2 - IABAR2



Register Offset
+020H

Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
31:12	00000H	Translation Base Address 2 - These bits define the actual location the translation function is to respond to when addressed from the PCI bus.
11:04	000H	Reserved.
03	0_2	Prefetchable Indicator - When set, defines the memory space as prefetchable.
02:01	00_2	Type Indicator - Defines the width of the addressability for this memory window (Memory Space Indicator = 0): 00 - Memory Window is locatable anywhere in 32 bit address space 10 - Memory Window is locatable anywhere in 64 bit address space
00	0_2	Memory Space Indicator - This bit field describes memory or I/O space base address. The ATU does not occupy I/O space, thus this bit must be zero.

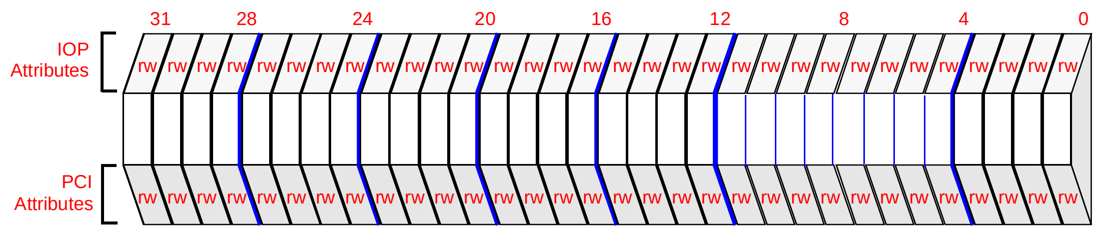
2.14.18 Inbound ATU Upper Base Address Register 2 - IAUBAR2

This register contains the upper base address when decoding PCI addresses for memory space (Memory Space Indicator in IABAR2 is clear) beyond 4 GBytes. Together with the Translation Base Address this register defines the actual location the translation function is to respond to when addressed from the PCI bus for addresses > 4GBytes (for DACs).

The programmed value within the base address register must comply with the PCI programming requirements for address alignment. Refer to the *PCI Local Bus Specification*, Revision 2.3 for additional information on programming base address registers.

Note: When the Type indicator of IABAR2 is set to indicate 32 bit addressability, the IAUBAR2 register attributes are read-only. By default the IAUBAR2 register has read-only attributes. Prior to changing the Type Indicator in the IABAR2 to support 32-bit addressability, the IAUBAR2 must be written with zero unless it already contains zero. Zero is the default value for IAUBAR2.

Table 44. Inbound ATU Upper Base Address Register 2 - IAUBAR2



Register Offset
+024H

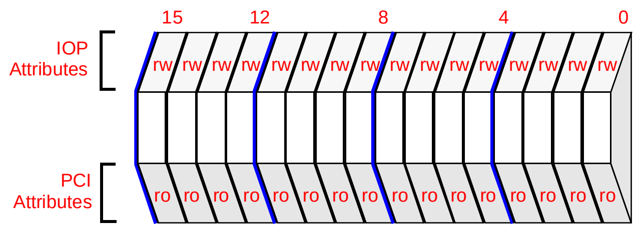
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:0	00000H	Translation Upper Base Address 2 - Together with the Translation Base Address 2 these bits define the actual location the translation function is to respond to when addressed from the PCI bus for addresses > 4GBytes.

2.14.19 ATU Subsystem Vendor ID Register - ASVIR

ATU Subsystem Vendor ID Register bit definitions adhere to *PCI Local Bus Specification*, Revision 2.3.

Table 45. ATU Subsystem Vendor ID Register - ASVIR



Register Offset
+02CH

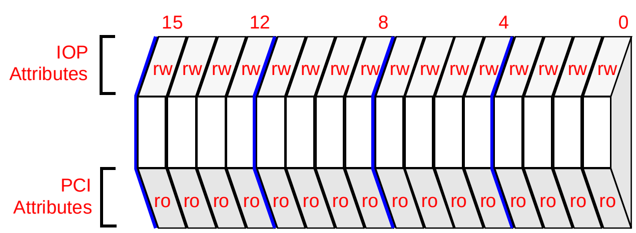
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
15:0	0000H	Subsystem Vendor ID - This register uniquely identifies the add-in board or subsystem vendor.

2.14.20 ATU Subsystem ID Register - ASIR

ATU Subsystem ID Register bit definitions adhere to *PCI Local Bus Specification*, Revision 2.3.

Table 46. ATU Subsystem ID Register - ASIR



Register Offset
+02EH

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

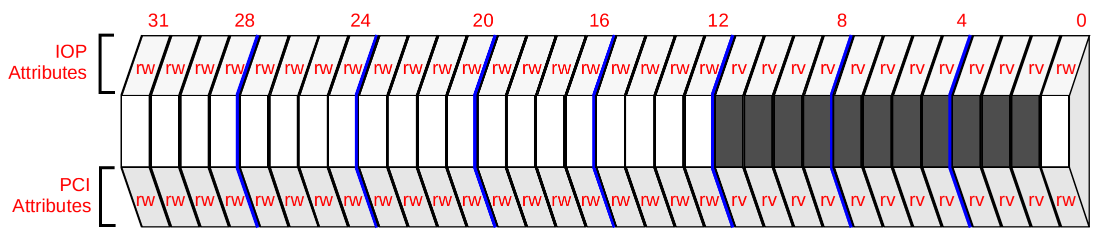
Bit	Default	Description
15:0	0000H	Subsystem ID - uniquely identifies the add-in board or subsystem.

2.14.21 Expansion ROM Base Address Register - ERBAR

The Expansion ROM Base Address Register defines the block of memory addresses used for containing the Expansion ROM. It permits the inclusion of multiple code images, allowing the device to be initialized. The code image supplied consists of either executable code or an interpreted code. Each code image must start on a 512 byte boundary and each must contain the PCI Expansion ROM header. Image placement in ROM space depends on the length of code images which precede it within ROM. ERBAR defines the base address and describes the required memory block size; see Section 2.14.23. Expansion ROM address space (limit size) can be a maximum of 16 MBytes. Bits 31 through 12 of the ERBAR is either read/write bits or read only with a value of 0 depending on the value located within the ERLR. This configuration allows the ERBAR to be programmed per *PCI Local Bus Specification*, Revision 2.3.

The Expansion ROM Base Address Register's programmed value must comply with the PCI programming requirements for address alignment. Refer to the *PCI Local Bus Specification*, Revision 2.3 for additional information on programming Expansion ROM base address registers.

Table 47. Expansion ROM Base Address Register -ERBAR



Register Offset +030H

PCI Configuration Address Offset 30H - 33H

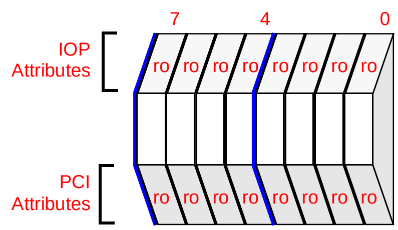
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:12	00000H	Expansion ROM Base Address - These bits define the actual location where the Expansion ROM address window resides when addressed from the PCI bus on any 4 Kbyte boundary.
11:01	000H	Reserved
00	0_2	Address Decode Enable - This bit field shows the ROM address decoder is enabled or disabled. When cleared, indicates the address decoder is disabled.

2.14.22 ATU Capabilities Pointer Register - ATU_Cap_Ptr

The Capabilities Pointer Register bits adhere to the definitions in the *PCI Local Bus Specification*, Revision 2.3. This register provides an offset in this function's PCI Configuration Space for the location of the first item in the first Capability list. In the case of the 4138xx, this is the PCI Bus Power Management extended capability as defined by the *PCI Bus Power Management Interface Specification*, Revision 1.1.

Table 48. ATU Capabilities Pointer Register - ATU_Cap_Ptr



Register Offset
+034H

Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

36 Bit	Default	Description
07:00	98H	Capability List Pointer - This provides an offset in this function's configuration space that points to the 4138xx PCI Bus Power Management extended capability.

2.14.23 Determining Block Sizes for Base Address Registers

The required address size and type can be determined by writing ones to a base address register and reading from the registers. By scanning the returned value from the least-significant bit of the base address registers upwards, the programmer can determine the required address space size. The binary-weighted value of the first non-zero bit found indicates the required amount of space. Table 49 describes the relationship between the values read back and the byte sizes the base address register requires.

Note: The use must exercise caution when re-programming the BAR, Limit, and Translate Value Registers. Since these 3 registers can not be programmed simultaneously, it is recommended that the BAR be disabled during reprogramming.

Table 49. Memory Block Size Read Response

Response After Writing all 1s to the Base Address Register	Size (in Bytes)	Response After Writing all 1s to the Base Address Register	Size (in Bytes)
FFFFFFF0H	16	FFF00000H	1 M
FFFFFFE0H	32	FFE00000H	2 M
FFFFFFC0H	64	FFC00000H	4 M
FFFFFF80H	128	FF800000H	8 M
FFFFFF00H	256	FF000000H	16 M
FFFFFE00H	512	FE000000H	32 M
FFFFFC00H	1K	FC000000H	64 M
FFFFF800H	2K	F8000000H	128 M
FFFFF000H	4K	F0000000H	256 M
FFFFE000H	8K	E0000000H	512 M
FFFFC000H	16K	C0000000H	1 G
FFFF8000H	32K	80000000H	2 G
FFFF0000H	64K	00000000H	Register not implemented, no address space required.
FFFE0000H	128K		
FFFC0000H	256K		
FFF80000H	512K		

As an example, assume that FFFF.FFFFH is written to the ATU Inbound Base Address Register 0 (IABAR0) and the value read back is FFF0.0008H. Bit zero is a zero, so the device requires memory address space. Bit three is one, so the memory does supports prefetching. Scanning upwards starting at bit four, bit twenty is the first one bit found. The binary-weighted value of this bit is 1,048,576, indicated that the device requires 1 Mbyte of memory space.

The ATU Base Address Registers and the Expansion ROM Base Address Register use their associated limit registers to enable which bits within the base address register are read/write and which bits are read only (0). This allows the programming of these registers in a manner similar to other PCI devices even though the limit is variable.

Table 50. ATU Base Registers and Associated Limit Registers

Base Address Register	Limit Register[a]	Description
Inbound ATU Base Address Register 0	Inbound ATU Limit Register 0	Defines the inbound translation window 0 from the PCI bus.
Inbound ATU Upper Base Address Register 0	N/A	Together with ATU Base Address Register 0 defines the inbound translation window 0 from the PCI bus for DACs.
Inbound ATU Base Address Register 1	Inbound ATU Limit Register 1	Defines the inbound translation window 1 from the PCI bus.
Inbound ATU Upper Base Address Register 1	N/A	Together with ATU Base Address Register 1 defines the inbound translation window 1 from the PCI bus for DACs.
Inbound ATU Base Address Register 2	Inbound ATU Limit Register 2	Defines the inbound translation window 2 from the PCI bus.
Inbound ATU Upper Base Address Register 2	N/A	Together with ATU Base Address Register 2 defines the inbound translation window 2 from the PCI bus for DACs.
Inbound ATU Base Address Register 3	Inbound ATU Limit Register 3	Defines the inbound translation window 3 from the PCI bus.
Inbound ATU Upper Base Address Register 3	N/A	Together with ATU Base Address Register 3 defines the inbound translation window 3 from the PCI bus for DACs. ***Note:*** This is a private BAR that resides outside of the standard PCI configuration header space (offsets 00H-3FH).
Expansion ROM Base Address Register	Expansion ROM Limit Register	Defines the window of addresses used by a bus master for reading from an Expansion ROM.

a. For Inbound Memory Windows 0-2, bit 0 of the limit register is a “claiming disable” bit. This feature allows the Memory Window to be used to a Memory Range for use in communication with Private PCI devices.

2.14.24 ATU Interrupt Line Register - ATUILR

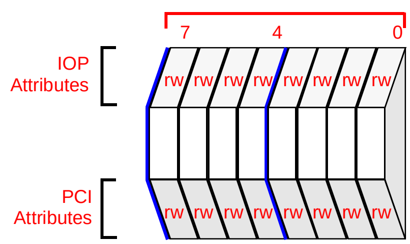
ATU Interrupt Line Register bit definitions adhere to *PCI Local Bus Specification*, Revision 2.3. This register identifies the system interrupt controller's interrupt request lines which connect to the device's PCI interrupt request lines (as specified in the interrupt pin register).

In a PC environment, for example, the register values and corresponding connections are:

- 0 (00H) through 15 (0FH) correspond to IRQ0 through IRQ15
- 16 (10H) through 254 (FEH) are reserved
- 255 (FFH) indicates "unknown" or "no connection"

The operating system or device driver can examine each device's interrupt pin and interrupt line register to determine which system interrupt request line the device uses to issue requests for service.

Table 51. ATU Interrupt Line Register - ATUILR



Register Offset
+03CH

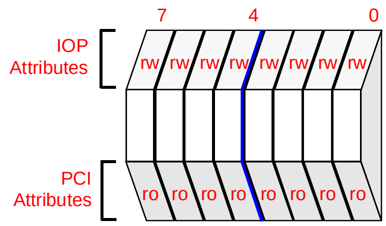
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
07:00	FFH	Interrupt Assigned - system-assigned value identifies which system interrupt controller's interrupt request line connects to the device's PCI interrupt request lines (as specified in the interrupt pin register). A value of FFH signifies "no connection" or "unknown".

2.14.25 ATU Interrupt Pin Register - ATUIPR

ATU Interrupt Pin Register bit definitions adhere to *PCI Local Bus Specification*, Revision 2.3. This register identifies the interrupt pin the ATU and Messaging Unit interface uses. The 4138xx is, a PCI single-function device and, as such, generates only one interrupt output. The interrupt output is for the Messaging Unit on **INTA#**.

Table 52. ATU Interrupt Pin Register - ATUIPR



Register Offset
+03DH

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

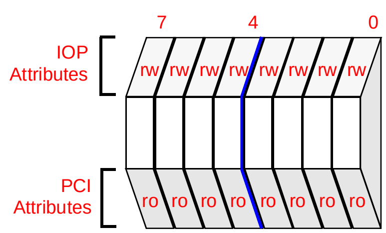
Bit	Default	Description
07:00	01H	Interrupt Used - A value of 01H signifies that the ATU interface unit uses **INTA#** as the interrupt pin.

2.14.26 ATU Minimum Grant Register - ATUMGNT

ATU Minimum Grant Register bit definitions adhere to *PCI Local Bus Specification*, Revision 2.3. This register specifies the burst period the device requires in increments of 8 PCI clocks.

This register and the ATU Maximum Latency register are information-only registers which the configuration uses to determine how often a bus master typically requires access to the PCI bus and the duration of a typical transfer when it does acquire the bus. This information is useful in determining the values to be programmed into the bus master latency timers and in programming the algorithm to be used by the PCI bus arbiter.

Table 53. ATU Minimum Grant Register - ATUMGNT



Register Offset
+003EH

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

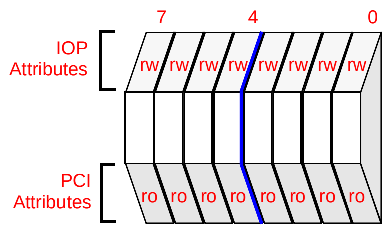
Bit	Default	Description
07:00	80H	This register specifies how long a burst period the device needs in increments of 8 PCI clocks.

2.14.27 ATU Maximum Latency Register - ATUMLAT

ATU Maximum Latency Register bit definitions adhere to *PCI Local Bus Specification*, Revision 2.3. This register specifies how often the device needs to access the PCI bus in increments of 8 PCI clocks.

This register and the Minimum Grant Register are information-only registers which the configuration uses to determine how often a bus master typically requires access to the PCI bus and the duration of a typical transfer when it does acquire the bus. This information is useful in determining the values to be programmed into the bus master latency timers and in programming the algorithm to be used by the PCI bus arbiter.

Table 54. ATU Maximum Latency Register - ATUMLAT



Register Offset
+03FH

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
07:00	00H	Specifies frequency (how often) the device needs to access the PCI bus in increments of 8 PCI clocks. A zero value indicates the device has no stringent requirement.

2.14.28 Inbound ATU Limit Register 0 - IALR0

Inbound address translation for memory window 0 occurs for data transfers occurring from the PCI bus (originated from the PCI bus) to the 4138xx internal bus. The address translation block converts PCI addresses to internal bus addresses.

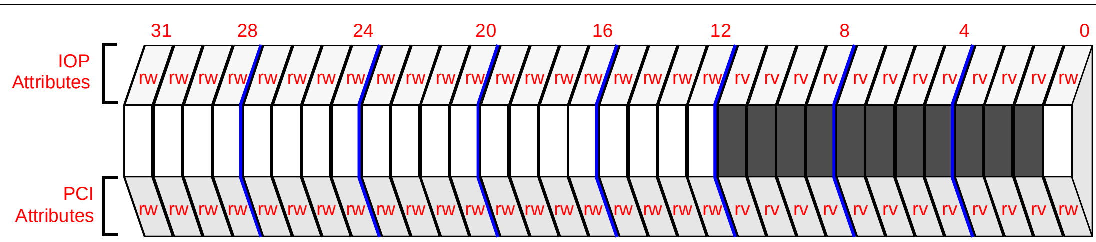
The inbound translation base address for inbound window 0 is specified in Section 2.14.13. When determining block size requirements — as described in Section 2.14.23 — the translation limit register provides the block size requirements for the base address register. The remaining registers used for performing address translation are discussed in Section 2.2.1.1.

The 4138xx value register's programmed value must be naturally aligned with the base address register's programmed value. The limit register is used as a mask; thus, the lower address bits programmed into the 4138xx value register are invalid. Refer to the *PCI Local Bus Specification*, Revision 2.3 for additional information on programming base address registers.

Bits 31 to 12 within the IALR0 have a direct effect on the IABAR0 register, bits 31 to 12, with a one to one correspondence. A value of 0 in a bit within the IALR0 makes the corresponding bit within the IABAR0 a read only bit which always returns 0. A value of 1 in a bit within the IALR0 makes the corresponding bit within the IABAR0 read/write from PCI. Note that a consequence of this programming scheme is that unless a valid value exists within the IALR0, all writes to the IABAR0 has no effect since a value of all zeros within the IALR0 makes the IABAR0 a read only register.

Note: Bit 0 can be used to disable claiming of Memory Cycles that hit Inbound Memory Window 0 even though the host processor has allocated memory of the size requested by IABAR0/IALR0[31:12].

Table 55. Inbound ATU Limit Register 0 - IALR0



Register Offset
+040H

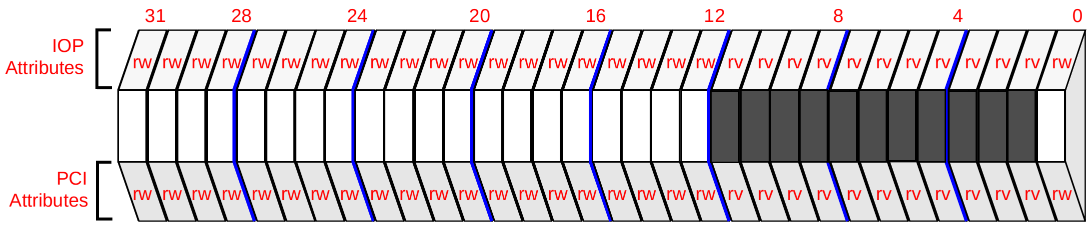
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:12	FF000H	Inbound Translation Limit 0 - This value determines the memory block size required for inbound memory window 0 of the address translation unit. This defaults to an inbound window of 16MB.
11:01	000H	Reserved
00	0_2	Memory Window 0 Claim Disable -- When this bit is set, Inbound Memory Window 0 does not claim memory cycles on the PCI Bus. When clear, Inbound Memory Window 0 claims PCI Memory Cycles Normally.

2.14.29 Inbound ATU Translate Value Register 0 - IATVR0

The Inbound ATU Translate Value Register 0 (IATVR0) in conjunction with the "Inbound ATU Upper Translate Value Register 0 - IAUTVR0" on page 170 contain bits 35 to 12 of the internal bus address used to convert PCI bus addresses. The converted address is driven on the internal bus as a result of the inbound ATU address translation.

Table 56. Inbound ATU Translate Value Register 0 - IATVR0



Register Offset +044H

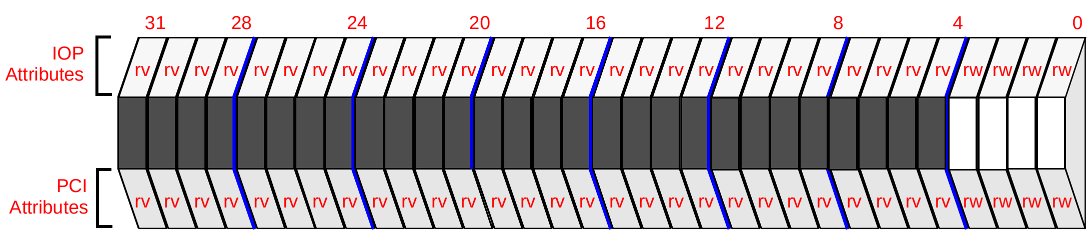
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:12	FF000H	Inbound ATU Translation Value 0 - This value represents bits 31 to 12 of the internal bus address used to convert the PCI address to internal bus addresses. This value must be naturally aligned with the IABAR0 register's programmed value (see Section 2.14.23, "Determining Block Sizes for Base Address Registers" on page 164).The default address allows the ATU to access the internal 4138xx memory-mapped registers.
11:01	000H	Reserved
00	0	Big Endian Byte Swap enable - When set the ATU performs a byte swap on all PCI read/write transactions through BAR0. When clear, no swap is performed. Refer to Section 2.3, "Big Endian Byte Swapping" on page 78 for more details.

2.14.30 Inbound ATU Upper Translate Value Register 0 - IAUTVR0

The Inbound ATU Upper Translate Value Register 0 (IAUTVR0) in conjunction with the "Inbound ATU Translate Value Register 0 - IATVR0" on page 170 contain bits 35 to12 of the internal bus address used to convert PCI bus addresses. The converted address is driven on the internal bus as a result of the inbound ATU address translation.

Table 57. Inbound ATU Upper Translate Value Register 0 - IAUTVR0



Register Offset +048H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:04	000 0000H	Reserved
3:0	0H	Inbound Upper ATU Translation Value 0 - This value represents bits 35 to 32 of the internal bus address used to convert the PCI address to internal bus addresses. The default address allows the ATU to access the internal 4138xx memory-mapped registers.

2.14.31 Inbound ATU Limit Register 1 - IALR1

Inbound address translation for memory window 1 occurs for data transfers occurring from the PCI bus (originated from the PCI bus) to the 4138xx internal bus. The address translation block converts PCI addresses to internal bus addresses.

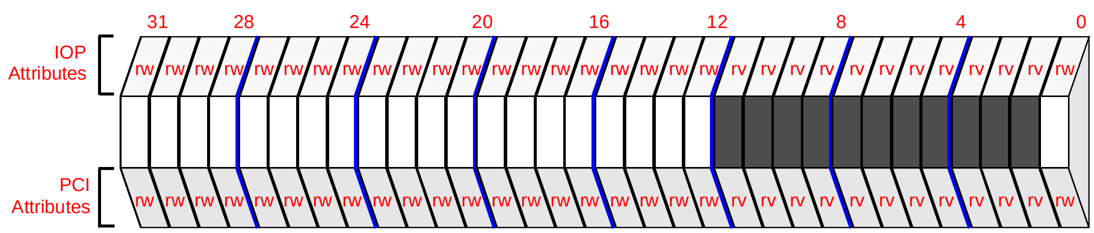
The inbound translation base address for inbound window 1 is specified in Section 2.14.15. When determining block size requirements — as described in Section 2.14.23 — the translation limit register provides the block size requirements for the base address register. The remaining registers used for performing address translation are discussed in Section 2.2.1.1.

The 4138xx value register programmed value must be naturally aligned with the base address register's programmed value. The limit register is used as a mask; thus, the lower address bits programmed into the 4138xx value register are invalid. Refer to the *PCI Local Bus Specification*, Revision 2.3 for additional information on programming base address registers.

Bits 31 to 12 within the IALR1 have a direct effect on the IABAR1 register, bits 31 to 12, with a one to one correspondence. A value of 0 in a bit within the IALR1 makes the corresponding bit within the IABAR1 a read only bit which always returns 0. A value of 1 in a bit within the IALR1 makes the corresponding bit within the IABAR1 read/write from PCI. Note that a consequence of this programming scheme is that unless a valid value exists within the IALR1, all writes to the IABAR1 has no effect since a value of all zeros within the IALR1 makes the IABAR1 a read only register.

Note: Bit 0 can be used to disable claiming of Memory Cycles that hit Inbound Memory Window 1 even though the host processor has allocated memory of the size requested by IABAR1/IALR1[31:12].

Table 58. Inbound ATU Limit Register 1 - IALR1



Register Offset
+04CH

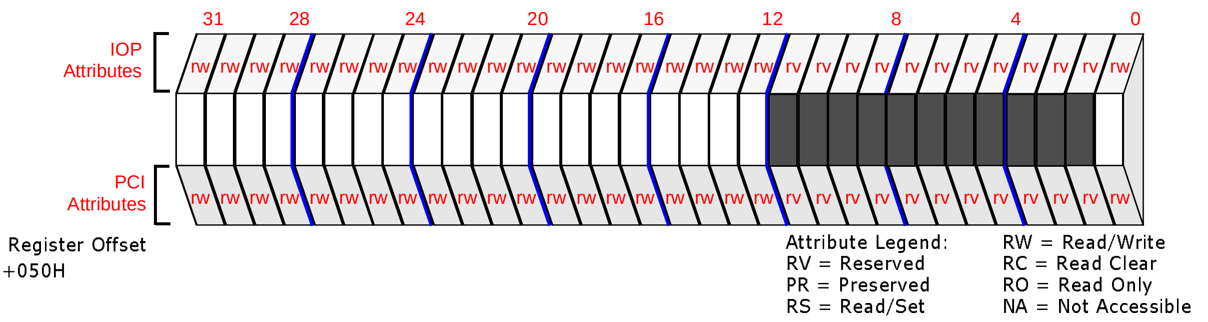
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:12	00000H	Inbound Translation Limit 1 - This value determines the memory block size required for the ATUs memory window 1.
11:01	000H	Reserved
00	0_2	Memory Window 1 Claim Disable -- When this bit is set, Inbound Memory Window 1 does not claim memory cycles on the PCI Bus. When clear, Inbound Memory Window 1 claims PCI Memory Cycles Normally.

2.14.32 Inbound ATU Translate Value Register 1 - IATVR1

The Inbound ATU Translate Value Register 1 (IATVR1) in conjunction with the "Inbound ATU Upper Translate Value Register 1 - IAUTVR1" on page 172 contain bits 35 to 12 of the internal bus address used to convert PCI bus addresses. The converted address is driven on the internal bus as a result of the Inbound ATU address translation.

Table 59. Inbound ATU Translate Value Register 1 - IATVR1

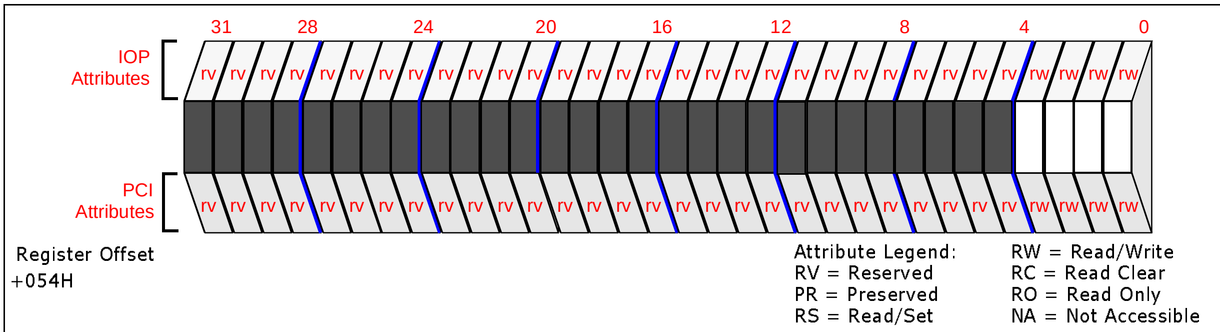


Bit	Default	Description
31:12	00000H	Inbound ATU Translation Value 1: Bits 31 to 12 of the internal bus address used to convert the PCI address to internal bus addresses. Must be naturally aligned with the IABAR1 register's programmed value (see Section 2.14.23, "Determining Block Sizes for Base Address Registers" on page 164).
11:01	000H	Reserved
00	0	Big Endian Byte Swap enable: 0 = No swap performed. See Section 2.3, "Big Endian Byte Swapping" on page 78 for more details. 1 = Performs a byte swap on all PCI read/write transactions through BAR1.

2.14.33 Inbound ATU Upper Translate Value Register 1 - IAUTVR1

The Inbound ATU Upper Translate Value Register 1 (IAUTVR1) in conjunction with the "Inbound ATU Translate Value Register 1 - IATVR1" on page 172 contain bits 35 to12 of the internal bus address used to convert PCI bus addresses. The converted address is driven on the internal bus as a result of the inbound ATU address translation.

Table 60. Inbound ATU Upper Translate Value Register 1 - IAUTVR1



Bit	Default	Description
31:04	000 0000H	Reserved
3:0	0H	Inbound Upper ATU Translation Value 1 - This value represents bits 35 to 32 of the internal bus address used to convert the PCI address to internal bus addresses.

2.14.34 Inbound ATU Limit Register 2 - IALR2

Inbound address translation for inbound window 2 occurs for data transfers occurring from the PCI bus (originated from the PCI bus) to the 4138xx internal bus. The address translation block converts PCI addresses to internal bus addresses.

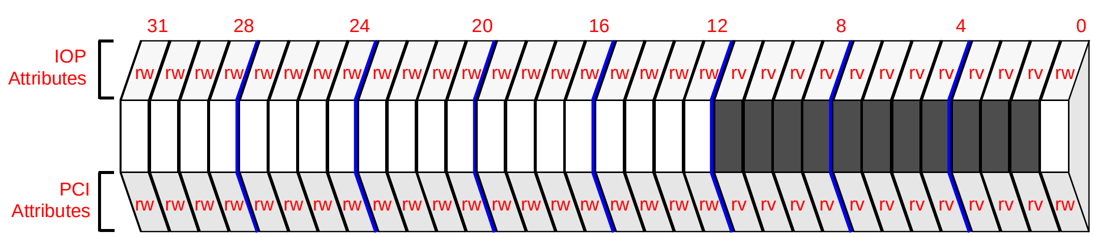
The inbound translation base address for inbound window 2 is specified in Section 2.14.17. When determining block size requirements — as described in Section 2.14.23 — the translation limit register provides the block size requirements for the base address register. The remaining registers used for performing address translation are discussed in Section 2.2.1.1.

The 4138xx value register's programmed value must be naturally aligned with the base address register's programmed value. The limit register is used as a mask; thus, the lower address bits programmed into the 4138xx value register are invalid. Refer to the *PCI Local Bus Specification*, Revision 2.3 for additional information on programming base address registers.

Bits 31 to 12 within the IALR2 have a direct effect on the IABAR2 register, bits 31 to 12, with a one to one correspondence. A value of 0 in a bit within the IALR2 makes the corresponding bit within the IABAR2 a read only bit which always returns 0. A value of 1 in a bit within the IALR2 makes the corresponding bit within the IABAR2 read/write from PCI. Note that a consequence of this programming scheme is that unless a valid value exists within the IALR2, all writes to the IABAR2 has no effect since a value of all zeros within the IALR2 makes the IABAR2 a read only register.

Note: Bit 0 can be used to disable claiming of PCI Cycles that hit Inbound Window 1 even though the host processor has allocated memory of the size requested by IABAR2/IALR2[31:12].

Table 61. Inbound ATU Limit Register 2 - IALR2



Register Offset
+058H

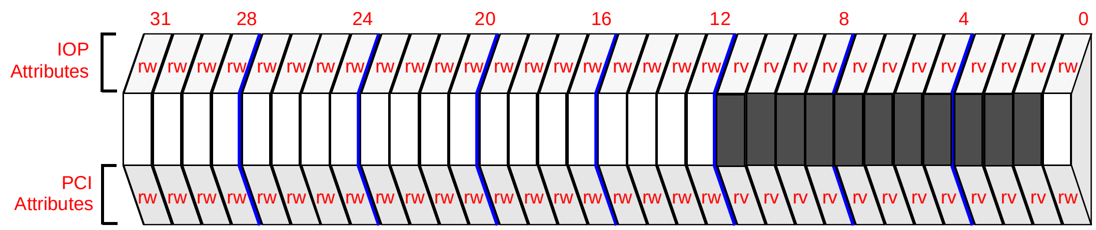
Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
31:12	00000H	Inbound Translation Limit 2 - This value determines the memory block size required for the ATUs memory window 2.
11:01	000H	Reserved
00	0_2	Window 2 Claim Disable -- When this bit is set, Inbound Window 2 does not claim cycles on the PCI Bus. When clear, Inbound Memory Window 2 claims PCI Cycles Normally.

2.14.35 Inbound ATU Translate Value Register 2 - IATVR2

The Inbound ATU Translate Value Register 2 (IATVR2) in conjunction with the "Inbound ATU Upper Translate Value Register 2 - IAUTVR2" on page 174 contain bits 35 to 12 of the internal bus address used to convert PCI bus addresses. The converted address is driven on the internal bus as a result of the Inbound ATU address translation.

Table 62. Inbound ATU Translate Value Register 2 - IATVR2



Register Offset +05CH

PCI Configuration Address Offset 5CH - 5FH

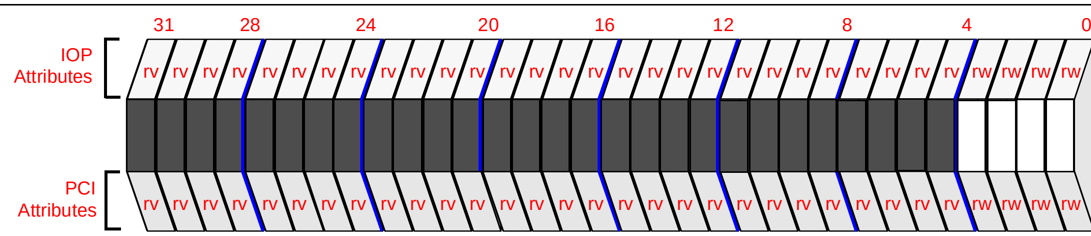
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:12	00000H	Inbound ATU Translation Value 2 - This value represents bits 31 to 12 of the internal bus address used to convert the PCI address to internal bus addresses. This value must be naturally aligned with the IABAR2 register's programmed value (see Section 2.14.23, "Determining Block Sizes for Base Address Registers" on page 164).
11:01	000H	Reserved
00	0	Big Endian Byte Swap enable - When set the ATU performs a byte swap on all PCI read/write transactions through BAR2. When clear, no swap is performed. Refer to Section 2.3, "Big Endian Byte Swapping" on page 78 for more details.

2.14.36 Inbound ATU Upper Translate Value Register 2 - IAUTVR2

The Inbound ATU Upper Translate Value Register 2 (IAUTVR2) in conjunction with the "Inbound ATU Translate Value Register 2 - IATVR2" on page 174 contain bits 35 to 8 of the internal bus address used to convert PCI bus addresses. The converted address is driven on the internal bus as a result of the inbound ATU address translation.

Table 63. Inbound ATU Upper Translate Value Register 2 - IAUTVR2



Register Offset +060H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

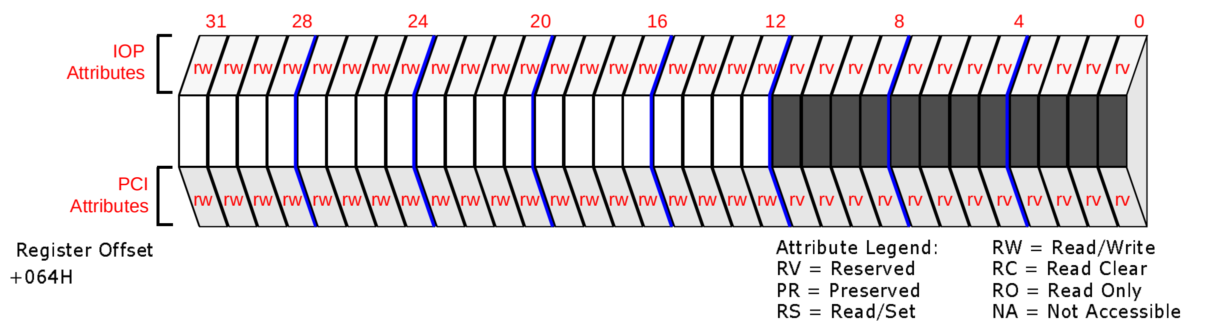
Bit	Default	Description
31:04	000 0000H	Reserved
3:0	0H	Inbound Upper ATU Translation Value 2 - This value represents bits 35 to 32 of the internal bus address used to convert the PCI address to internal bus addresses.

2.14.37 Expansion ROM Limit Register - ERLR

The Expansion ROM Limit Register (ERLR) defines the block size of addresses the ATU defines as Expansion ROM address space. Block size is programmed by writing a value into the ERLR.

Bits 31 to 12 within the ERLR have a direct effect on the ERBAR register, bits 31 to 12, with a one to one correspondence. A value of 0 in a bit within the ERLR makes the corresponding bit within the ERBAR a read only bit which always returns 0. A value of 1 in a bit within the ERLR makes the corresponding bit within the ERBAR read/write from PCI.

Table 64. Expansion ROM Limit Register - ERLR

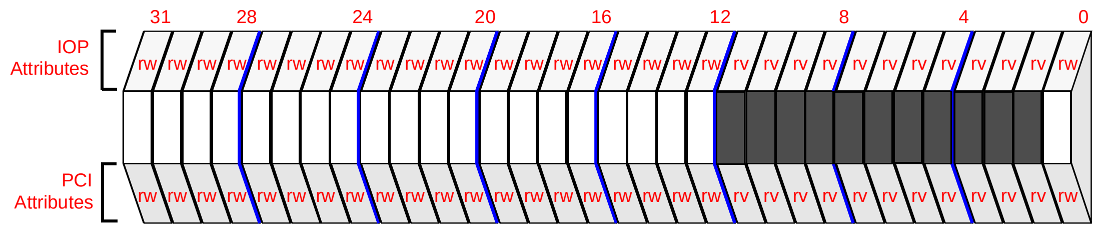


Bit	Default	Description
31:12	000000H	Expansion ROM Limit - Memory block size required for Expansion ROM translation unit. Default value 0, indicates no Expansion ROM address space and all bits within ERBAR are read only with a value of 0.
11:00	000H	Reserved

2.14.38 Expansion ROM Translate Value Register - ERTVR

The Expansion ROM Translate Value Register 0 (ERTVR) in conjunction with the "Expansion ROM Upper Translate Value Register - ERUTVR" on page 176 contain bits 35 to 12 of the internal bus address used to convert PCI bus addresses. The converted address is driven on the internal bus as a result of the Expansion ROM address translation.

Table 65. Expansion ROM Translate Value Register - ERTVR



Register Offset
+068H

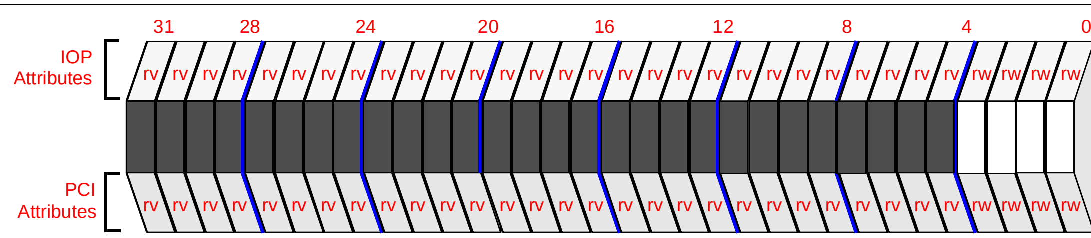
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:12	00000H	Expansion ROM Translation Value - Bits 31 to 12 of internal bus address used to convert PCI address to internal bus addresses. This value must be naturally aligned with the ERBAR register's programmed value (see Section 2.14.23, "Determining Block Sizes for Base Address Registers" on page 164).
11:01	000H	Reserved
00	0	Big Endian Byte Swap enable - When set the ATU performs a byte swap on all PCI read transactions through EROM BAR. When clear, no swap is performed. Refer to Section 2.3, "Big Endian Byte Swapping" on page 78 for more details.

2.14.39 Expansion ROM Upper Translate Value Register - ERUTVR

The Expansion ROM Upper Translate Value Register (ERUTVR) in conjunction with the "Expansion ROM Translate Value Register - ERTVR" on page 176 contain bits 35 to12 of the internal bus address used to convert PCI bus addresses. The converted address is driven on the internal bus as a result of the Expansion ROM address translation.

Table 66. Expansion ROM Upper Translate Value Register - ERUTVR



Internal Bus Address
+06CH

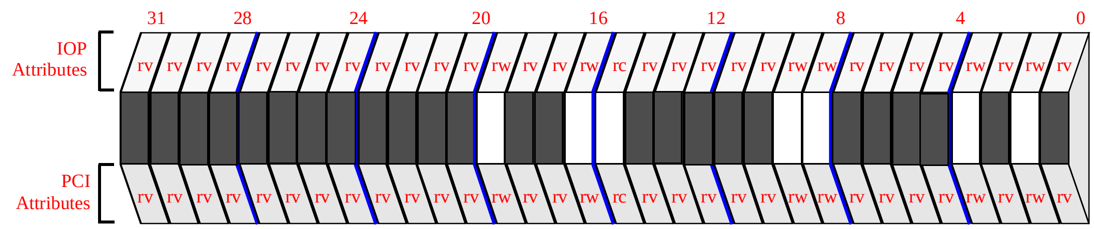
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:04	000 0000H	Reserved
3:0	0H	Expansion ROM Upper Translation Value - This value represents bits 35 to 32 of the internal bus address used to convert the PCI address to internal bus addresses.

2.14.40 ATU Configuration Register - ATUCR

The ATU Configuration Register controls the outbound address translation for address translation unit. It also contains bits for Conventional PCI Delayed Read Command (DRC) aliasing, discard timer status, **P_SERR#** manual assertion, **P_SERR#** detection interrupt masking, and ATU BIST interrupt enabling.

Table 67. ATU Configuration Register - ATUCR



Register Offset
+070H

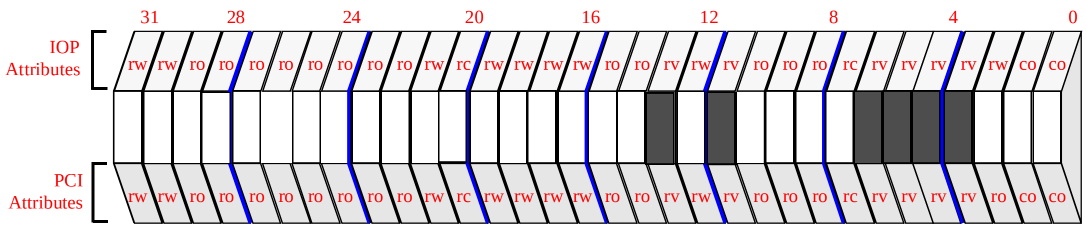
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:20	00H	Reserved
19	0_2	ATU DRC Alias - when set, the ATU does not distinguish read commands when attempting to match a current PCI read transaction with read data enqueued within the DRC buffer. When clear, a current read transaction must have the exact same read command as the DRR for the ATU to deliver DRC data. Not applicable in the PCI-X mode.
18:17	00b	Reserved
16	0_2	**P_SERR#** Manual Assertion - when set, the ATU asserts **P_SERR#** for one clock on the PCI interface. Until cleared, **P_SERR#** may not be manually asserted again. Once cleared, operation proceeds as specified.
15	0_2	ATU Discard Timer Status - when set, one of the 4 discard timers within the ATU has expired and discarded the delayed completion transaction within the queue. When clear, no timer has expired.
14:10	00_2	Reserved
09	0_2	**P_SERR#** Detected Interrupt Enable - When set, the Intel XScale® processor is signalled an **HPI#** interrupt when the ATU detects that **P_SERR#** was asserted. When clear, the Intel XScale® processor is not interrupted when **P_SERR#** is detected.
08	0_2	Halt ATU On Error Enable - When set and a Data Parity, Master-Abort, or Target-Abort error occurs on an Outbound PCI Write Request, the ATU halts operation using the following sequence: 1. Stop accepting Outbound Requests by clearing the Outbound ATU Enable (bit 1 of the ATUCR). 2. Flush any pending outbound write requests from the transaction queue. 3. Pull the Intel XScale® processor interrupt as normal for the error. Firmware should wait for the Outbound Queue Busy Bit in "PCI Configuration and Status Register - PCSR" to clear before setting the Outbound ATU Enable (ATUCR[1]). ***Note:*** Due to the pipelined nature of PERR# signaling, one additional write may have issued on the PCI bus before the error condition is detected. All remaining writes in the queue is flushed. Read Requests are allowed to proceed and complete normally though no new ones are accepted.
07:04	000_2	Reserved
03	0_2	ATU BIST Interrupt Enable - When set, enables an interrupt to the Intel XScale® processor when the start BIST bit is set in the ATUBISTR register. This bit is also reflected as BIST Capable bit 7 in the ATUBISTR register.
02	0_2	Reserved
01	0_2	Outbound ATU Enable - When set, enables the outbound address translation unit. When cleared, disables the outbound ATU.
00	0_2	Reserved

2.14.41 PCI Configuration and Status Register - PCSR

The PCI Configuration and Status Register has additional bits for controlling and monitoring various features of the PCI bus interface.

Table 68. PCI Configuration and Status Register - PCSR (Sheet 1 of 3)

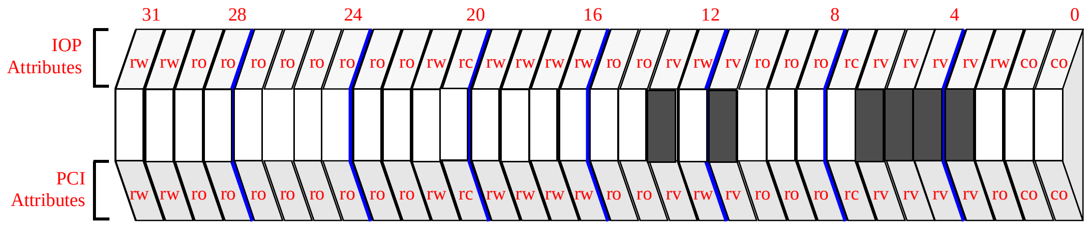


Register Offset
+074H

Attribute Legend:
RV = Reserved
CO = Clear Only
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:30	0_2	Initiate Core Processor Reset. These bits are used to place the Intel XScale® processor processors into reset. When a '1' is written to this bits the corresponding Core Processor Reset (Bit 1:0)) is set. This bit is self clearing and always reads '0'.
29	**HS_LSTAT**	Hot-Swap Latch Status: Reflects the state of the HS_LSTAT pin.
28	**HS_SM#**	Hot-Swap Start up Mode: Indicates 4138xx behavior when targeted by Configuration transactions with Configuration Retry Mode enabled (bit 2 of the PCSR). 0 = 4138xx ignores all configuration transactions targeting it thereby causing the transaction to ultimately Master Abort. 1 = 4138xx retries all configuration transactions targeting it.
27:26	**HS_FREQ[1:0]**	Hot-Swap Frequency selection These bits reflect the value on the HS_FREQ[1:0] pins when the **HS_SM#** strap is asserted. 11 PCI Mode, 33 or 66 MHz. Use P_M66EN to determine frequency 10 PCI-X Mode, 66 MHz 01 PCI-X Mode, 100 MHz 00 PCI-X Mode, 133 MHz ***Note:*** 4138xx does not support CompactPCI Hot-Swap in PCI-X Mode2.
25	**PCIX_EP#**	PCIX End Point Strap 0 = ATUX is operating in End Point mode 1 = ATUX is operating as a central resource.
24	**PCIXM1_100#**	PCIX Mode1 - 100MHz limit When all bus devices are capable of operating at greater then 66MHz, the **PCIXM1_100#** strap limits the bus to operate at a maximum frequency of 100MHz. 0 = PCI-X Bus is limited to 100MHz operation in Mode 1 1 = PCI-X Bus can operate at 133MHz. in Mode 1
23	**PCIXM2_100#**	PCIX Mode2 - 100MHz limit (200MHz data rate) When all bus devices are capable of operating at greater then 66MHz, the **PCIXM1_100#** strap limits the bus to operate at a maximum frequency of 100MHz. 0 = PCI-X Bus is limited to 100MHz operation in Mode 1 1 = PCI-X Bus can operate at 133MHz. in Mode 1
22	**EXTARB#**	External Arbiter When operating in Central Resource Mode (**PCIX_EP#** = 1), this strap enables the use of an external arbiter 0 = External Arbiter Enabled 1 = External Arbiter Disabled (Use Internal Arbiter).

Table 68. PCI Configuration and Status Register - PCSR (Sheet 2 of 3)

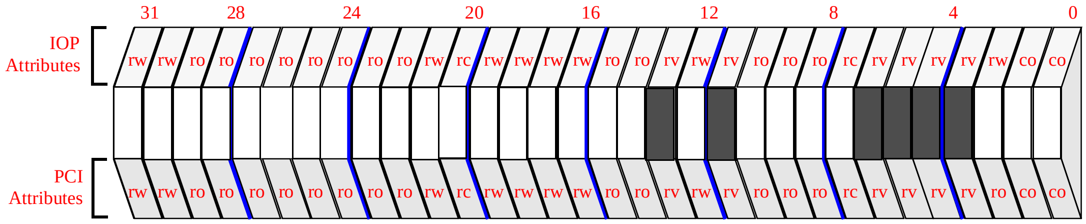


Register Offset
+074H

Attribute Legend:
RV = Reserved
CO = Clear Only
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
21	1	Central Resource PCI Bus Reset - When set, **P_RSTOUT#** is asserted. When cleared, **P_RSTOUT#** is deasserted. After clearing this bit, there is a delay of about 300 uS before hardware de-asserts the **P_RSTOUT#** signal. After this bit is cleared, the hardware waits about 150 uS to allow the PLL to warm-up and another 150 uS to allow the clocks to stabilize. Therefore, firmware has to wait about 300uS for the P_RSTOUT# signal to get de-asserted. After hardware de-asserts P_RSTOUT#, firmware has to wait before issuing the first configuration cycle in order to meet the PCI timing parameter Trhfa (about 2^{26} PCI clocks). Note that the PCI timing parameter Trhfa is dependent on the PCI bus speed selected. ***Note:*** **P_RSTOUT#** is asserted by default. This output should remain unconnected when operating as an endpoint.
20	0_2	Detected Uncorrectable Address or Attribute Error - set when an uncorrectable error is detected during either the address or attribute phase of a transaction on the PCI bus even when the ATUCMD register's Parity Error Response bit is cleared. Set under the following conditions: • Any Uncorrectable Address or Attribute (PCI-X Only) Error on the Bus (including one generated by the ATU).
19:16	See description for default value	PCI-X capability - These bits define the mode of the PCI bus (conventional or PCI-X) as well as the operating frequency in the case of PCI-X mode and are consistent with the electrical value on the PCI bus. As a Central Resource, this field controls the initialization pattern driven on the PCI bus during reset and the value driven on the **CR_FREQ[1:0]** pins. The default value of this field is dependent on the following pins/straps: **PCIXM1_100#**, **PCIXM2_100#**, **P_MODE2**, **P_PCIXCAP**, and **P_M66EN.** As an endpoint, this register reflects the value captured off the bus during reset based on the following signals: P_PERR#, P_DEVSEL#, P_STOP#, and P_TRDY#. 1111 - Conventional PCI mode(frequency depends on **P_M66EN**) 1110 - PCI-X 66 1101 - PCI-X 100 1100 - PCI-X 133 0110 - PCI-X 266 (66 MHz **P_CLK**) 0101 - PCI-X 266 (100 MHz **P_CLK**) 0100 - PCI-X 266 (133 MHz **P_CLK**) All other values are reserved. All other patterns are reserved or not supported by the 4138xx. See Section 2.12.6, "Bus Mode and Frequency Initialization" on page 134 for more details.
15	0_2	Outbound Transaction Queue Busy: 0 = Outbound Transaction Queue Empty 1 = Outbound Transaction Queue Busy
14	0_2	Inbound Transaction Queue Busy: 0 = Inbound Transaction Queue Empty 1 = Inbound Transaction Queue Busy
13	0_2	Reserved
12	0_2	Discard Timer Value - This bit controls the time-out value for the four discard timers attached to the queues holding read data. A value of 0 indicates the time-out value is 2^{15} clocks. A value of 1 indicates the time-out value is 2^{10} clocks.
11	0_2	Reserved

Table 68. PCI Configuration and Status Register - PCSR (Sheet 3 of 3)



Register Offset
+074H

Attribute Legend:
RV = Reserved
CO = Clear Only
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
10	Varies with external state of **P_M66EN** at PCI bus reset	Conventional PCI Bus Operating at 66 MHz - When set, the interface has been initialized to function at 66 MHz in Conventional PCI mode by the assertion of **P_M66EN** during bus initialization. When clear, the interface has been initialized as a 33 MHz bus. ***Note:*** When PCSR bits 19:16 are not equal to zero, then this bit is meaningless since the4138xx is operating in PCI-X mode.
09	**PMODE2**	PCI Bus Mode 2 Indicates the bus is capable of operating in PCI-X Mode 2.
08	Varies with external state of **P_REQ64#** at PCI bus reset	PCI Bus 64-Bit Capable - When clear, the PCI bus interface has been configured as 64-bit capable by the assertion of **P_REQ64#** on the rising edge of **P_RST#**. When set, the PCI interface is configured as 32-bit only. Note that this bit is valid only when acting as an endpoint.
07	Varies with external state of the **FW_TIMER_OFF#** strap	Firmware Timer Disable 0 = Firmware Timeout is disabled. 1 = Firmware Timeout is enabled When enabled, a 400mS timer is started at the trailing edge of reset. The Configuration Request Retry bit (bit 2 of this register) is cleared when it is not cleared before the expiration of the timer. For example, when firmware does not clear the Configuration Request Retry bit before the timer expires. The firmware timer bit is also automatically cleared by the firmware timer when the Configuration Request Retry bit is still set when the timer expires. After the host is allowed access, the firmware timer bit can be used to indicate how the Configuration Request Retry bit was cleared: 0 = The firmware timer expired and cleared both the Configuration Request Retry and the firmware timer bits. 1 = Firmware cleared the Configuration Request Retry bit before the timer expired. ***Note:*** When the firmware timer is disabled, firmware is responsible to clear the Configuration Request Retry bit. Otherwise, the ATU indefinitely retries all host configuration cycles.
06:03	00_2	Reserved
02	Varies with external state of **RETRY** pin at PCI bus reset	Configuration Cycle Retry - When this bit is set, the PCI interface of the 4138xx responds to all configuration cycles with a Retry condition. When clear, the 4138xx responds to the appropriate configuration cycles. The default condition for this bit is based on the external state of the **RETRY** pin at the rising edge of **P_RST#**. When the external state of the pin is high, the bit is set. When the external state of the pin is low, the bit is cleared.
01:00	Varies with external state of the **RST_MODE [1:0]#** pins at PCI bus reset	Core Processor Reset - These bits are set to their default values by the hardware when either **P_RST#** is asserted or the Reset Internal Bus bit in PCSR is set. When these bits are set, the associated Intel XScale® processors are being held in reset. Software cannot these bits. Software is required to clear these bit(s) to deassert Intel XScale® processor reset. The default condition for these bits are based on the external state of the **RST_MODE[1:0]#** pins at the rising edge of **P_RST#**. When the external state of these pins are low, the default value of these bits is set. When the external state of the pin is high, the default value of these bits is clear. ***Note:*** RST_MODE0# is associated with Intel XScale® processor 0 which is the Protocol Core while RST_MODE1# is associated with Intel XScale® processor 1 which is the Application core.

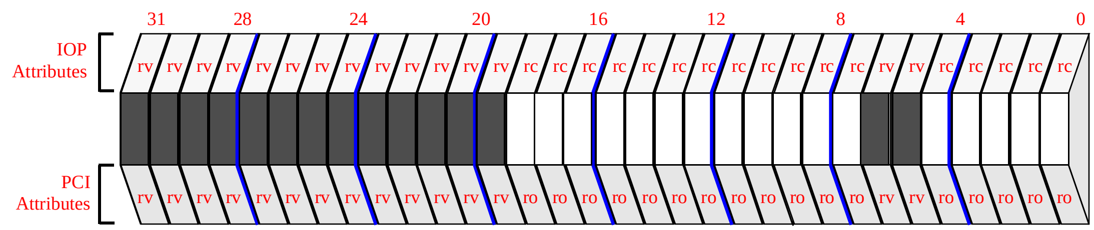
2.14.42 ATU Interrupt Status Register - ATUISR

The ATU Interrupt Status Register is used to notify the core processor of the source of an ATU interrupt. In addition, this register is written to clear the source of the interrupt to the interrupt unit of the 4138xx. All bits in this register are Read/Clear.

Bits 4:0 are a direct reflection of bits 14:11 and bit 8 (respectively) of the ATU Status Register (these bits are set at the same time by hardware but need to be cleared independently). Bit 7 is set by an error associated with the internal bus of the 4138xx. Bit 8 is for software BIST. The conditions that result in an ATU interrupt are cleared by writing a 1 to the appropriate bits in this register.

Note that bits 4:0, and bits 15 and 13:7 can result in an interrupt being driven to the Intel XScale® processor.

Table 69. ATU Interrupt Status Register - ATUISR (Sheet 1 of 2)

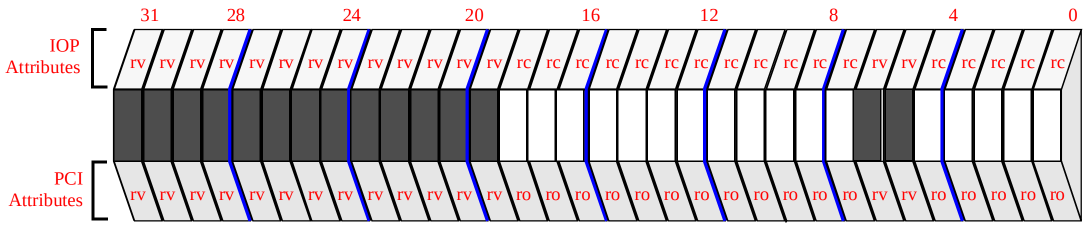


Register Offset +078H

Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
31:19	0000H	Reserved
18	0_2	PCI Interface Error - This bit is set when any PCI interface error occurs. When set, this bit results in the assertion of the ATU PCI Interface Error Interrupt.
17	0_2	VPD Address Register Updated - This bit is set when a PCI bus configuration write occurs to the VPDAR register. When set, this bit results in the assertion of the ATU Config Reg Write Interrupt.
16	0_2	Internal Bus Parity Error Detected - This bit is set when an Internal Bus Parity Error is detected.
15	0_2	ATU Configuration Write - This bit is set when a PCI bus configuration write occurs to any enabled function. When set, this bit results in the assertion of the ATU Config Reg Write Interrupt.
14	0_2	Detected Correctable Error - This bit is set in PCI-X Mode 2 only when the 4138xx detects a single bit ECC error in any phase of a PCI transaction.
13	0_2	Initiated Split Completion Error Message - This bit is set when the device initiates a Split Completion Message on the PCI Bus with the Split Completion Error attribute bit set.
12	0_2	Received Split Completion Error Message - This bit is set when the device receives a Split Completion Message from the PCI Bus with the Split Completion Error attribute bit set.
11	0_2	Power State Transition - When the Power State Field of the ATU Power Management Control/Status Register is written to transition the ATU function Power State from D0 to D3, D0 to D1, or D3 to D0 and the ATU Power State Transition Interrupt mask bit is cleared, this bit is set.
10	0_2	**P_SERR#** Asserted - set when **P_SERR#** is asserted on the PCI bus by the ATU.
09	0_2	Detected Parity Error - set when a parity error is detected on the PCI bus even when the ATUCMD register's Parity Error Response bit is cleared. Set under the following conditions: • Uncorrectable Write Data Error when the ATU is a target (inbound write). • Uncorrectable Read Data Error when the ATU is an initiator (outbound read). • Any Uncorrectable Address or Attribute (PCI-X Only) Error on the Bus.
08	0_2	ATU BIST Interrupt - When set, the host processor has set the start BIST, ATUBISTR register bit 6, and the ATU BIST interrupt enable (ATUCR register bit 3) is enabled. The Intel XScale® processor can initiate the software BIST and store the result in ATUBISTR register bits 3:0.

Table 69. ATU Interrupt Status Register - ATUISR (Sheet 2 of 2)



Register Offset
+078H

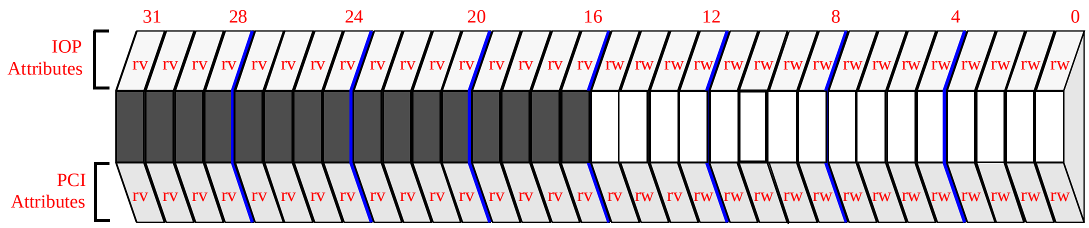
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
07	0_2	Internal Bus Master Abort - set when a transaction initiated by the ATU internal bus initiator interface ends in a Master-abort.
06:05	00_2	Reserved.
04	0_2	**P_SERR#** Detected - set when **P_SERR#** is detected on the PCI bus by the ATU.
03	0_2	PCI Master Abort - set when a transaction initiated by the ATU PCI initiator interface ends in a Master-abort.
02	0_2	PCI Target Abort (master) - set when a transaction initiated by the ATU PCI master interface ends in a Target-abort.
01	0_2	PCI Target Abort (target) - set when the ATU interface, acting as a target, terminates the transaction on the PCI bus with a target abort.
00	0_2	PCI Master Parity Error - Master Parity Error - The ATU interface sets this bit under the following conditions: • The ATU asserted **PERR#** itself or the ATU observed **PERR#** asserted. • And the ATU acted as the requester for the operation in which the error occurred. • And the ATUCMD register's Parity Error Response bit is set • Or (PCI-X Mode Only) the ATU received an Uncorrectable Write Data Error Message • And the ATUCMD register's Parity Error Response bit is set

2.14.43 ATU Interrupt Mask Register - ATUIMR

The ATU Interrupt Mask Register contains the control bit to enable and disable interrupts generated by the ATU.

Table 70. ATU Interrupt Mask Register - ATUIMR (Sheet 1 of 2)

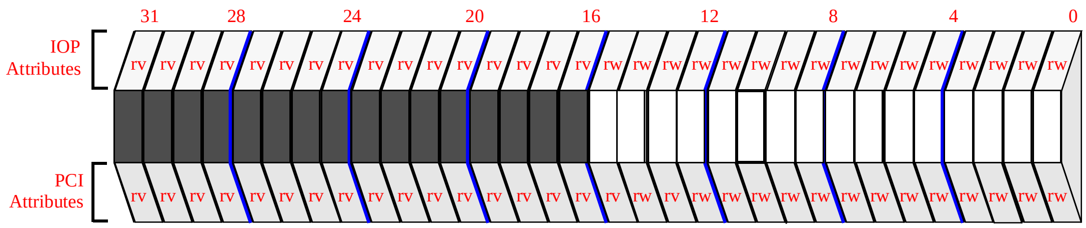


Register Offset
+07CH

Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
31:16	0 0000H	Reserved
15	0_2	PCI Interface Error Mask - Controls the setting of bit 18 of the ATUISR and generation of the PCI Interface error when any PCI bus error occurs. 0 = Not Masked 1 = Masked
14	0_2	VPD Address Register Updated Mask - Controls the setting of bit 17 of the ATUISR and generation of the ATU Configuration Register Write interrupt when a PCI bus write occurs to the VPDAR register. 0 = Not Masked 1 = Masked
13	0_2	Internal Bus Parity Error Detected Mask - Controls the setting of bit 16 of the ATUISR and generation of the ATU Error interrupt when an internal bus parity error is detected. 0 = Not Masked 1 = Masked
12	0_2	Configuration Register Write Mask - Controls the setting of bit 15 of the ATUISR and generation of the ATU Configuration Register Write interrupt when a PCI bus write occurs to any ATU configuration register except those covered by mask bit 11 and bit 14 of this register. 0 = Not Masked 1 = Masked
11	1_2	Detected Correctable Error Mask - Controls setting of bit 14 of the ATUISR and generation of the Correctable Error interrupt when in PCI-X Mode 2, a correctable error is detected in any phase of a PCI transaction. 0 = Not Masked 1 = Masked
10	0_2	Initiated Split Completion Error Message Interrupt Mask - Controls the setting of bit 13 of the ATUISR and generation of the ATU Error interrupt when the ATU initiates a Split Completion Error Message. 0 = Not Masked 1 = Masked
09	0_2	Received Split Completion Error Message Interrupt Mask- Controls setting of bit 12 of ATUISR and generation of ATU Error interrupt when a Split Completion Error Message results in bit 29 of PCIXSR being set. 0 = Not Masked 1 = Masked
08	1_2	Power State Transition Interrupt Mask - Controls the setting of bit 12 of the ATUISR and generation of the ATU Error interrupt when ATU Power Management Control/Status Register is written to transition the ATU Function Power State from D0 to D3, D0 to D1, D1 to D3 or D3 to D0. 0 = Not Masked 1 = Masked
07	0_2	ATU Detected Parity Error Interrupt Mask - Controls the setting of bit 9 of the ATUISR and generation of the ATU Error interrupt when a parity error detected on the PCI bus that sets bit 15 of the ATUSR. 0 = Not Masked 1 = Masked

Table 70. ATU Interrupt Mask Register - ATUIMR (Sheet 2 of 2)



Register Offset
+07CH

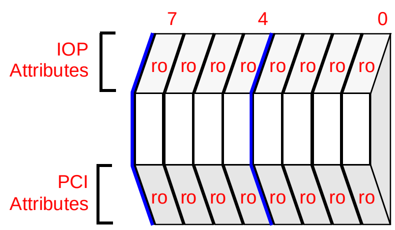
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
06	0_2	ATU **SERR#** Asserted Interrupt Mask - Controls the setting of bit 10 of the ATUISR and generation of the ATU Error interrupt when **SERR#** is asserted on the PCI interface resulting in bit 14 of the ATUSR being set. 0 = Not Masked 1 = Masked ***Note:*** This bit is specific to the ATU asserting **SERR#** and not detecting **SERR#** from another master.
05	0_2	ATU PCI Master Abort Interrupt Mask - Controls the setting of bit 3 of the ATUISR and generation of the ATU Error interrupt when a master abort error resulting in bit 13 of the ATUSR being set. 0 = Not Masked 1 = Masked
04	0_2	ATU PCI Target Abort (Master) Interrupt Mask- Controls the setting of bit 12 of the ATUISR and ATU Error generation of the interrupt when a target abort error resulting in bit 12 of the ATUSR being set 0 = Not Masked 1 = Masked
03	0_2	ATU PCI Target Abort (Target) Interrupt Mask- Controls the setting of bit 1 of the ATUISR and generation of the ATU Error interrupt when a target abort error resulting in bit 11 of the ATUSR being set. 0 = Not Masked 1 = Masked
02	0_2	ATU PCI Master Parity Error Interrupt Mask - Controls the setting of bit 0 of the ATUISR and generation of the ATU Error interrupt when a parity error resulting in bit 8 of the ATUSR being set. 0 = Not Masked 1 = Masked
01	0_2	ATU Inbound Error **SERR#** Enable - Controls when ATU asserts (when enabled through the ATUCMD) **SERR#** on the PCI interface in response to a master abort on the internal bus during an inbound write transaction. 0 = **SERR#** Not Asserted due to error 1 = **SERR#** Asserted due to error
00	0_2	ATU ECC Target Abort Enable - Controls the ATU response on the PCI interface to a target abort (ECC error) from the memory controller on the internal bus. In conventional mode, this action only occurs during an inbound read transaction where the data phase that was target aborted on the internal bus is actually requested from the inbound read queue. 0 = Disconnect with data (the data being up to 64 bits of 1's) 1 = Target Abort ***Note:*** In PCI-X Mode, The ATU initiates a Split Completion Error Message (with message class=2h - completer error and message index=81h - 4138xx internal bus target abort) on the PCI bus, independent of the setting of this bit.

2.14.44 VPD Capability Identifier Register - VPD_Cap_ID

The Capability Identifier Register bits adhere to the definitions in the *PCI Local Bus Specification*, Revision 2.3. This register in the PCI Extended Capability header identifies the type of Extended Capability contained in that header. In the case of the 4138xx, this is the VPD extended capability with an ID of 03H as defined by the *PCI Local Bus Specification*, Revision 2.3.

Table 71. VPD Capability Identifier Register - VPD_Cap_ID



Register Offset
+090H

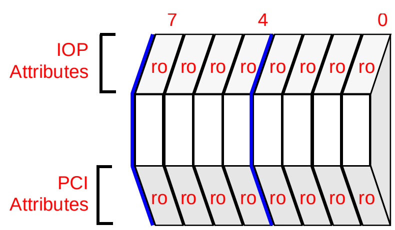
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
07:00	03H	Cap_Id - This field with its' 03H value identifies this item in the linked list of Extended Capability Headers as being the VPD capability registers.

2.14.45 VPD Next Item Pointer Register - VPD_Next_Item_Ptr

The Next Item Pointer Register bits adhere to the definitions in the *PCI Local Bus Specification*, Revision 2.3. This register describes the location of the next item in the function's capability list. For the 4138xx, this the final capability list, and hence, this register is set to 00H.

Table 72. VPD Next Item Pointer Register - VPD_Next_Item_Ptr



Register Offset
+091H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

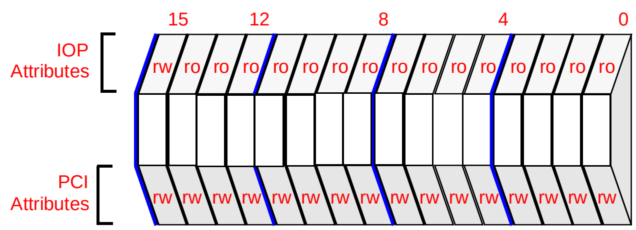
Bit	Default	Description
07:00	00H	Next_ Item_ Pointer - This field provides an offset into the function's configuration space pointing to the next item in the function's capability list. Since the VPD capabilities are the last in the linked list of extended capabilities in the 4138xx, the register is set to 00H.

2.14.46 VPD Address Register - VPDAR

The VPD Address register (VPDAR) contains the DWORD-aligned byte address of the VPD to be accessed. The register is read/write and the initial value at power-up is indeterminate.

A PCI Configuration Write to the VPDAR interrupts the Intel XScale® processor. Software can use the Flag setting to determine whether the configuration write was intended to initiate a read or write of the VPD through the VPD Data Register.

Table 73. VPD Address Register - VPDAR



Register Offset
+092H

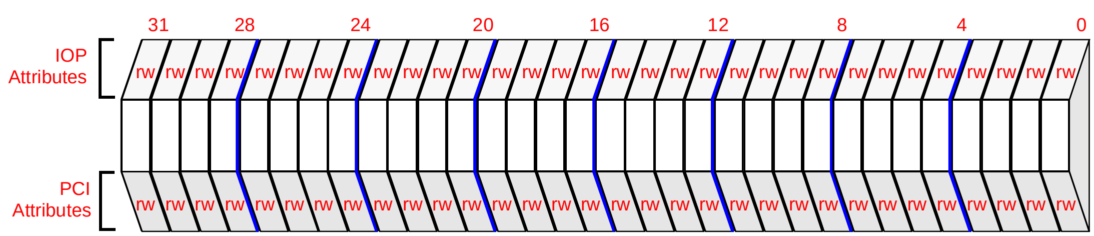
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
15	0_2	Flag - A flag is used to indicate when a transfer of data between the VPD Data Register and the storage component has completed. Please see Section 2.10, "Vital Product Data" on page 127 for more details on how the 4138xx handles the data transfer.
14:0	0000H	VPD Address - This register is written to set the DWORD-aligned byte address used to read or write Vital Product Data from the VPD storage component.

2.14.47 VPD Data Register - VPDDR

This register is used to transfer data between the 4138xx and the VPD storage component.

Table 74. VPD Data Register - VPDDR



Register Offset
+094H

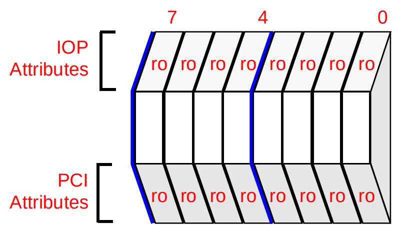
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:0	0000H	VPD Data - Four bytes are always read or written through this register to/from the VPD storage component.

2.14.48 PM Capability Identifier Register - PM_Cap_ID

The Capability Identifier Register bits adhere to the definitions in the *PCI Local Bus Specification*, Revision 2.3. This register in the PCI Extended Capability header identifies the type of Extended Capability contained in that header. In the case of the 4138xx, this is the PCI Bus Power Management extended capability with an ID of 01H as defined by the *PCI Bus Power Management Interface Specification*, Revision 1.1.

Table 75. PM_Capability Identifier Register - PM_Cap_ID



Register Offset +098H

PCI Configuration Offset 98H

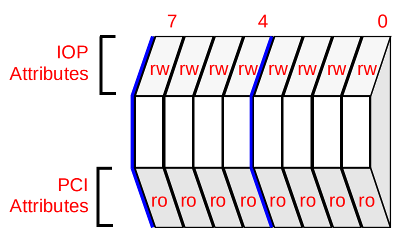
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
07:00	01H	Cap_Id - This field with its' 01H value identifies this item in the linked list of Extended Capability Headers as being the PCI Power Management Registers.

2.14.49 PM Next Item Pointer Register - PM_Next_Item_Ptr

The Next Item Pointer Register bits adhere to the definitions in the *PCI Local Bus Specification*, Revision 2.3. This register describes the location of the next item in the function's capability list. For the 4138xx, the next capability (MSI-X capability list) is located at off-set B0H. Note that the PM_Next_Item_Ptr can be written by the processor.

Table 76. PM Next Item Pointer Register - PM_Next_Item_Ptr



Register Offset +099H

PCI Configuration Offset 99H

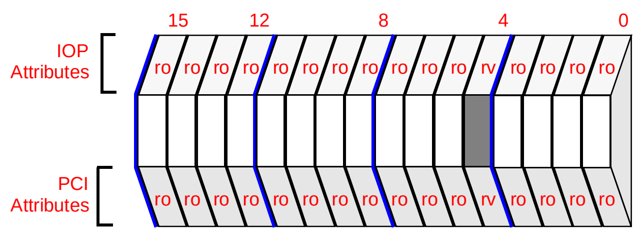
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
07:00	B0H	Next_ Item_ Pointer - This field provides an offset into the function's configuration space pointing to the next item in the function's capability list which in the 4138xx is the MSI-X extended capabilities header.

2.14.50 ATU Power Management Capabilities Register - APMCR

Power Management Capabilities bits adhere to the definitions in the *PCI Bus Power Management Interface Specification*, Revision 1.1. This register is a 16-bit read-only register which provides information on the capabilities of the ATU function related to power management.

Table 77. ATU Power Management Capabilities Register - APMCR



Register Offset
+09AH

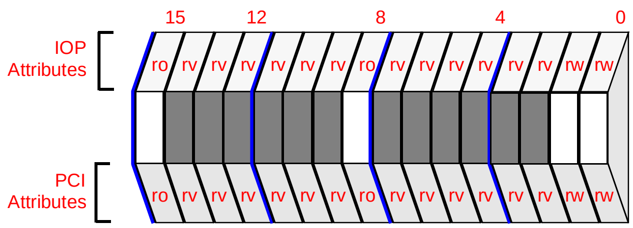
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
15:11	00000_2	PME_Support - This function is not capable of asserting the **PME#** signal in any state, since **PME#** is not supported by the 4138xx.
10	0_2	D2_Support - This bit is set to 0_2 indicating that the 4138xx does not support the D2 Power Management State
9	1_2	D1_Support - This bit is set to 1_2 indicating that the 4138xx supports the D1 Power Management State
8:6	000_2	Aux_Current - This field is set to 000_2 indicating that the 4138xx has no current requirements for the 3.3Vaux signal as defined in the *PCI Bus Power Management Interface Specification*, Revision 1.1
5	0_2	DSI - This field is set to 0_2 meaning that this function does not require a device specific initialization sequence following the transition to the D0 uninitialized state.
4	0_2	Reserved.
3	0_2	PME Clock - Since the 4138xx does not support **PME#** signal generation this bit is cleared to 0_2.
2:0	010_2	Version - Setting these bits to 010_2 means that this function complies with *PCI Bus Power Management Interface Specification*, Revision 1.1

2.14.51 ATU Power Management Control/Status Register - APMCSR

Power Management Control/Status bits adhere to the definitions in the *PCI Bus Power Management Interface Specification*, Revision 1.1. This 16-bit register is the control and status interface for the power management extended capability.

Table 78. ATU Power Management Control/Status Register - APMCSR



Register Offset
+09CH

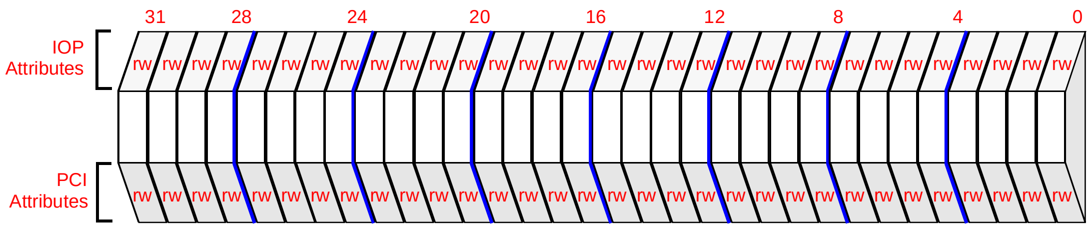
Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
15	0_2	PME_Status - This function is not capable of asserting the PME# signal in any state, since **PME##** is not supported by the 4138xx.
14:9	00H	Reserved
8	0_2	PME_En - This bit is hard-wired to read-only 0_2 since this function does not support **PME#** generation from any power state.
7:2	000000_2	Reserved
1:0	00_2	Power State - This 2-bit field is used both to determine the current power state of a function and to set the function into a new power state. The definition of the values is: 00_2 - D0 01_2 - D1 10_2 - D2 (Unsupported) 11_2 - D3$_{hot}$ The 4138xx supports only the D0 and D3$_{hot}$ states.

2.14.52 ATU Scratch Pad Register - ATUSPR

This register can be used for application specific purposes and has no direct impact on the hardware.

Table 79. Scratch Pad Register - ATUSPR



Internal Bus Address Offset
+0CCH

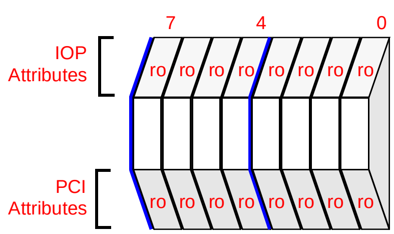
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:0	0000H	Scratch Pad Data - Entire register is available for application specific purposes.

2.14.53 PCI-X Capability Identifier Register - PCI-X_Cap_ID

The Capability Identifier Register bits adhere to the definitions in the *PCI Local Bus Specification*, Revision 2.3. This register in the PCI Extended Capability header identifies the type of Extended Capability contained in that header. In the case of the 4138xx, this is the PCI-X extended capability with an ID of 07H as defined by the *PCI-X Protocol Addendum to the PCI Local Bus Specification,* Revision 2.0.

Table 80. PCI-X_Capability Identifier Register - PCI-X_Cap_ID



Register Offset
+0D0H

PCI Configuration Offset
D0H

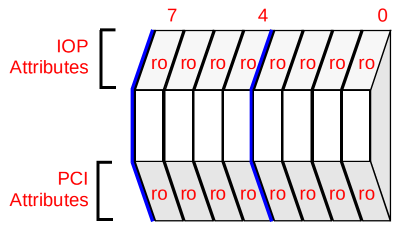
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
07:00	07H	Cap_Id - This field with its' 07H value identifies this item in the linked list of Extended Capability Headers as being the PCI-X capability registers.

2.14.54 PCI-X Next Item Pointer Register - PCI-X_Next_Item_Ptr

The Next Item Pointer Register bits adhere to the definitions in the *PCI Local Bus Specification*, Revision 2.3. This register describes the location of the next item in the function's capability list.

Table 81. PCI-X Next Item Pointer Register - PCI-X_Next_Item_Ptr



Register Offset
+0D1H

PCI Configuration Offset
D1H

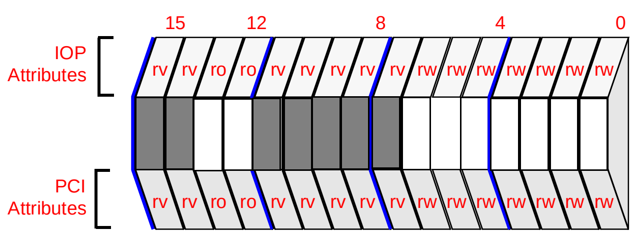
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
07:00	E8H	Next_ Item_ Pointer - This field provides an offset into the function's configuration space pointing to the next item in the function's capability list which in the 4138xx is the CompactPCI extended capabilities header.

2.14.55 PCI-X Command Register - PCIXCMD

This register controls various modes and features of ATU and Message Unit when operating in the PCI-X mode.

Table 82. PCI-X Command Register - PCIXCMD (Sheet 1 of 2)

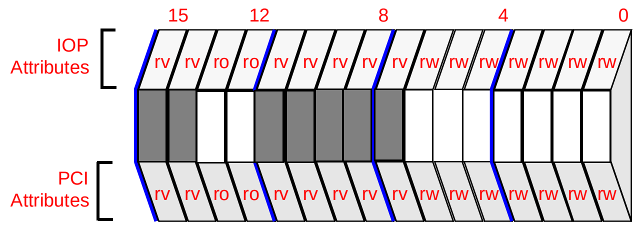


Register Offset
+0D2H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
15:14	00_2	Reserved
13:12	01_2	PCI-X Capabilities List Item Version - This field indicates that the 4138xx implements version 1 of the PCI-X Capabilities list item defined in the *PCI-X Protocol Addendum to the PCI Local Bus Specification,* Revision 2.0. Specifically, the 4138xx implements the 24 byte format of the PCI-X capabilities list item and support for ECC in Mode 2 only.
11:7	00000_2	Reserved

Table 82. PCI-X Command Register - PCIXCMD (Sheet 2 of 2)



Register Offset
+0D2H

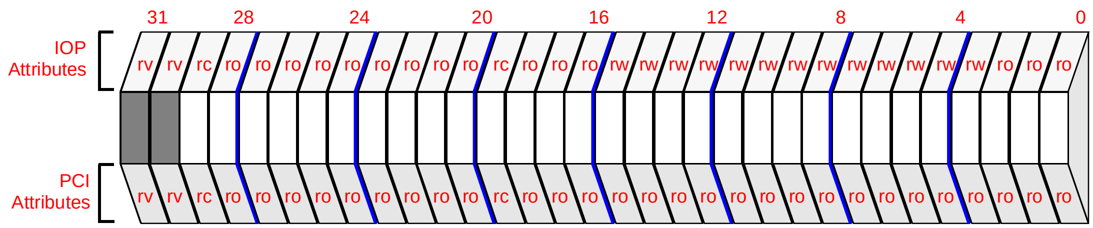
Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
6:4	011_2	Maximum Outstanding Split Transactions - This register sets the maximum number of Split Transactions the device is permitted to have outstanding at one time. Register Maximum Outstanding 0 1 1 2 2 3 3 4 4 8 5 12 6 16 7 32
3:2	00_2	Maximum Memory Read Byte Count - This register sets the maximum byte count the device uses when initiating a Sequence with one of the burst memory read commands. Register Maximum Byte Count 0 512 1 1024 2 2048 3 4096
1	1_2	Enable Relaxed Ordering - When set, the 4138xx may set the relaxed ordering bit in the Requester Attributes of Transactions.
0	0_2	Uncorrectable Data Error Recovery Enable - The device driver sets this bit to enable the device to attempt to recover from uncorrectable data errors. When this bit is 0 and the device is in PCI-X mode, the device asserts **P_SERR#** (when enabled) whenever the Master Data Parity Error bit (Status register, bit 8) is set.

2.14.56 PCI-X Status Register - PCIXSR

This register identifies the capabilities and current operating mode of ATU when operating in the PCI-X mode.

Table 83. PCI-X Status Register - PCIXSR (Sheet 1 of 2)

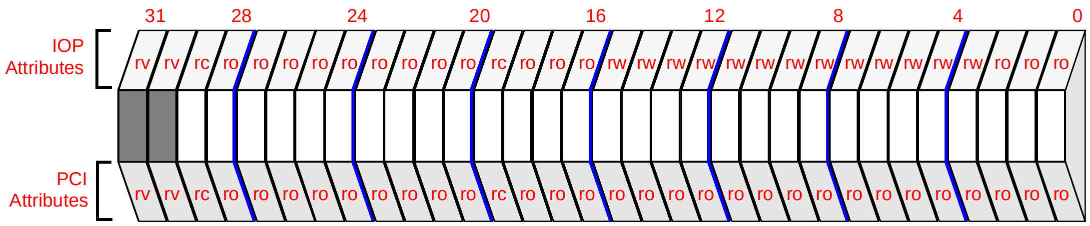


Register Offset
+0D4H

Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
31:30	00_2	Reserved
29	0_2	Received Split Completion Error Message - This bit is set when the device receives a Split Completion Message with the Split Completion Error attribute bit set. Once set, this bit remains set until software writes a 1 to this location. 0 = no Split Completion error message received. 1 = a Split Completion error message has been received.
28:26	001_2	Designed Maximum Cumulative Read Size (DMCRS) - The value of this register depends on the setting of the Maximum Memory Read Byte Count field of the PCIXCMD register: DMCRS Max ADQs *Maximum Memory Read Byte Count Register Setting* 1 16 512 (Default) 2 32 1024 2 32 2048 2 32 4096
25:23	011_2	Designed Maximum Outstanding Split Transactions - The 4138xx can have up to four outstanding split transactions.
22:21	01_2	Designed Maximum Memory Read Byte Count - The 4138xx can generate memory reads with byte counts up to 1024 bytes.
20	1_2	4138xx is a complex device.
19	0_2	Unexpected Split Completion - This bit is set when an unexpected Split Completion with this device's Requester ID is received. Once set, this bit remains set until software writes a 1 to this location. 0 = no unexpected Split Completion has been received. 1 = an unexpected Split Completion has been received.
18	0_2	Split Completion Discarded - This bit is set when the device discards a Split Completion because the requester would not accept it. See Section 5.4.4 of the *PCI-X Protocol Addendum to the PCI Local Bus Specification,* Revision 2.0 for details. Once set, this bit remains set until software writes a 1 to this location. 0 = no Split Completion has been discarded. 1 = a Split Completion has been discarded. ***Note:*** The 4138xx **never** sets this bit since there is no Inbound address responding to Inbound Read Requests with Split Responses (Memory or Register) that has "read side effects."
17	1_2	4138xx is a 133 MHz capable device.
16	**32BITPCI#**	4138xx can be configured to identify the add-in card to the system as 64-bit or 32-bit wide via a user-configurable strap (**32BITPCI#**). This strap, by default, identifies the 4138xx subsystem as 64-bit unless the user attaches the appropriate pull-down resistor to the strap. 0 = The bus is 32 bits wide. 1 = The bus is 64 bits wide.

Table 83. PCI-X Status Register - PCIXSR (Sheet 2 of 2)



Register Offset
+0D4H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

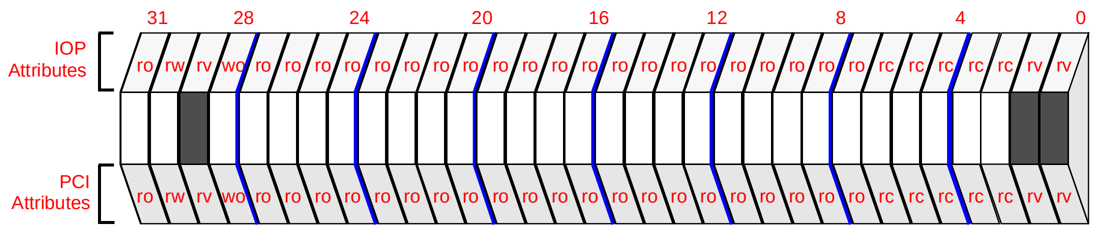
Bit	Default	Description
15:8	FFH	Bus Number - This register is read for diagnostic purposes only. It indicates the number of the bus segment for the device containing this function. The function uses this number as part of its Requester ID and Completer ID. For all devices other than the source bridge, each time the function is addressed by a Configuration Write transaction, the function must update this register with the contents of AD[7:0] of the attribute phase of the Configuration Write, regardless of which register in the function is addressed by the transaction. The function is addressed by a Configuration Write transaction when all of the following are true: 1. The transaction uses a Configuration Write command. 2. IDSEL is asserted during the address phase. 3. AD[1:0] are 00b (Type 0 configuration transaction). 4. AD[10:08] of the configuration address contain the appropriate function number.
7:3	1FH	Device Number - This register is read for diagnostic purposes only. It indicates the number of the device containing this function, i.e., the number in the Device Number field (AD[15:11]) of the address of a Type 0 configuration transaction that is assigned to the device containing this function by the connection of the system hardware. The system must assign a device number other than 00h (00h is reserved for the source bridge). The function uses this number as part of its Requester ID and Completer ID. Each time the function is addressed by a Configuration Write transaction, the device must update this register with the contents of AD[15:11] of the address phase of the Configuration Write, regardless of which register in the function is addressed by the transaction. The function is addressed by a Configuration Write transaction when all of the following are true: 1. The transaction uses a Configuration Write command. 2. IDSEL is asserted during the address phase. 3. AD[1:0] are 00b (Type 0 configuration transaction). 4. AD[10:08] of the configuration address contain the appropriate function number.
2:0	000_2	Function Number - This register is read for diagnostic purposes only. It indicates the number of this function; i.e., the number in the Function Number field (AD[10:08]) of the address of a Type 0 configuration transaction to which this function responds. The function uses this number as part of its Requester ID and Completer ID.

2.14.57 ECC Control and Status Register - ECCCSR

The ECCCSR register provides additional information about ECC errors that occurred on the PCI bus. Registers that store information from the failing transaction always store information directly from the PCI bus (uncorrected), even when correction of the error is possible.

Note: The "ECC Control and Status Register - ECCCSR", "ECC First Address Register - ECCFAR", "ECC Second Address Register - ECCSAR", and "ECC Attribute Register - ECCAR" report the actual transaction that has the error. For example, when the Split Completion of an original Outbound Read request has an error, the information regarding the Split Completion is reported.

Table 84. ECC Control and Status Register - ECCCSR (Sheet 1 of 3)

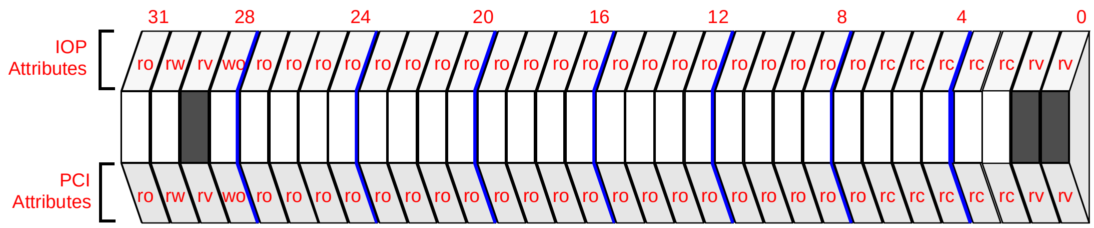


Internal Bus Address
+0D8H

Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	WO = Write Only
	NA = Not Accessible

Bit	Default	Description
31	0_2 (Mode 1 or Conventional) 1_2 (Mode 2)	ECC Mode - When this bit is 1, the 4138xx is in ECC mode. When this bit is 0, the 4138xx is in parity mode. The state of this bit after **P_RST#** is determined by the PCI-X initialization pattern described in the *PCI-X Protocol Addendum to the PCI Local Bus Specification,* Revision 2.0. In PCI-X Mode 2, this bit is a 1, otherwise this bit is a 0. ***Note:*** The 4138xx does not support ECC in PCI-X Mode 1 or in Conventional PCI mode.
30	0_2	Disable Single-Bit-Error Correction - When the 4138xx is in ECC mode and this bit is 0, correctable errors (as described in the *PCI-X Protocol Addendum to the PCI Local Bus Specification,* Revision 2.0) are corrected. When the 4138xx is in ECC mode and this bit is 1, correctable errors are not corrected and are treated as uncorrectable errors, including the setting of status bits and assertion of error indicator signals on the bus. Disabling single-bit error correction enhances the error detection capability of the ECC. In parity mode (ECC Mode bit is 0), this bit has no meaning and is ignored by the 4138xx. ***Note:*** Writes to this register do not affect this bit unless the ECC Control Update Enable bit is a 1 in the data pattern being written.
29	0_2	Reserved
28	0_2	ECC Control Update Enable - This bit always reads as a 0. When this bit is 1 in the data pattern being written, the Disable Single-Bit-Error Correction and ECC Mode bits are also updated (written). When this bit is 0 in the data pattern being written, the Disable Single-Bit-Error Correction and ECC Mode bits are not updated.
27:24	0H	Error Upper Attributes - When the ECC Error Phase register is non-zero, this register indicates the contents of the **P_C/BE[3:0]#** bus for the attribute phase of the transaction that included the error.
23:20	0H	Error Second Command - When the ECC Error Phase register is non-zero and the transaction that included the error used a dual address cycle, this register indicates the contents of the **P_C/BE[3:0]#** bus for the second address phase of the transaction that included the error.
19:16	0H	Error First (or only) Command - When the ECC Error Phase register is non-zero, this register indicates the contents of the **P_C/BE[3:0]#** bus for the first (or only) address phase of the transaction that included the error.

Table 84. ECC Control and Status Register - ECCCSR (Sheet 2 of 3)

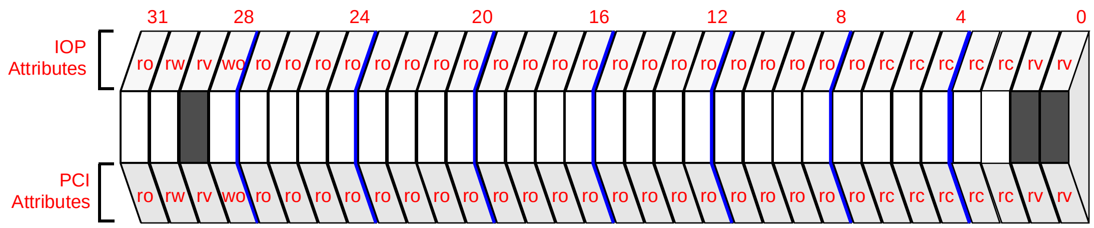


Internal Bus Address
+0D8H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
WO = Write Only
NA = Not Accessible

Bit	Default	Description
15:8	00H	Syndrome - The syndrome indicates information about the bit or bits that are in error, as described in the *PCI-X Protocol Addendum to the PCI Local Bus Specification,* Revision 2.0. Bit Syndrome 8 E0 9 E1 10 E2 11 E3 12 E4 13 E5 14 E6 15 E7 for 64-bit data, 0b for 32-bit data
7	0_2	ECC Error Corrected - When the ECC Error Phase register is non-zero, this bit indicates whether the error that was captured was corrected. Correctable ECC errors that occur while error correction is enabled (see Disable ECC Correction bit) are the only errors that are corrected. When the ECC Error Phase register is zero, this bit is undefined. 0 = The error that was captured was not corrected. 1 = The error that was captured was corrected.
6:4	000_2	ECC Error Phase - When the 4138xx detects either a correctable or uncorrectable ECC error, this register indicates in which phase of the transaction the error occurred, and for data phase errors whether it was a 32-bit data error (seven-bit ECC) or 64-bit data error (eight-bit ECC). When this register is set to 0, the 4138xx is enabled to latch information about an ECC error. When the 4138xx detects an error, it latches the phase of the error in this register, and stores status information for the error in this register and in the ECC Address, and ECC Attribute registers. Register ECC Error Phase 0 No Error 1 First 32 bits of address 2 Second 32 bits of address 3 Attribute phase 4 32 data phase 5 64 bit data phase 6 Reserved 7 Reserved ***Note:*** Writing a 1 to **any** of these bits clears this register and enables the device to capture the next error.
3	0_2	Additional Uncorrectable ECC Error - This bit is set when the 4138xx detects an uncorrectable ECC error, or a correctable ECC error while error correction is disabled, and the device is already indicating some other ECC error (i.e. the ECC Error Phase register is non-zero). 0 = No additional uncorrectable ECC error has been detected. 1 = One or more additional uncorrectable ECC errors have been detected.

Table 84. ECC Control and Status Register - ECCCSR (Sheet 3 of 3)



Internal Bus Address
+0D8H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
WO = Write Only
NA = Not Accessible

Bit	Default	Description
2	0_2	Additional Correctable ECC Error - This bit is set when the 4138xx detects a correctable ECC error while error correction is enabled and the device is already indicating some other ECC error (i.e. the ECC Error Phase register is non-zero). 0 = No additional correctable ECC error has been detected. 1 = One or more additional correctable ECC errors have been detected.
1:0	00_2	Reserved

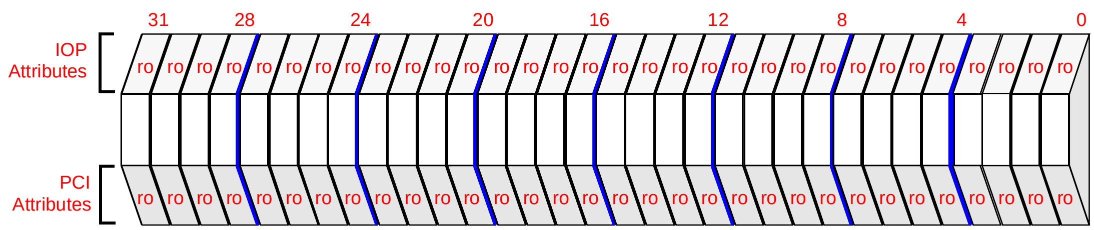
2.14.58 ECC First Address Register - ECCFAR

When the ECC Error Phase register (bits 6:4 of the ECCCSR) is non-zero (indicating that an error has been captured), the ECCFAR register indicates the contents of the **P_AD[31:0]** bus (for 64- and 32-bit buses) for the address phase of the transaction that included the error. For Dual Address Cycle (DAC) transactions, this represents the least significant 32-bits of the 64-bit address. When the ECC Error Phase register is zero, the contents of this register are undefined.

Note: Registers that store information from the failing transaction always store information directly from the bus (uncorrected), even when correction of the error is possible.

Note: The "ECC Control and Status Register - ECCCSR", "ECC First Address Register - ECCFAR", "ECC Second Address Register - ECCSAR", and "ECC Attribute Register - ECCAR" report the actual transaction that has the error. For example, when the Split Completion of an original Outbound Read request has an error, the information regarding the Split Completion is reported.

Table 85. ECC First Address Register - ECCFAR



Internal Bus Address
+0DCH

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:00	0000 0000H	ECC First Address - This register represents the 32-bit address of a failing single address cycle (SAC) transaction. Or, in the case of a failing DAC transaction, this register represents the least significant 32-bits of the address.

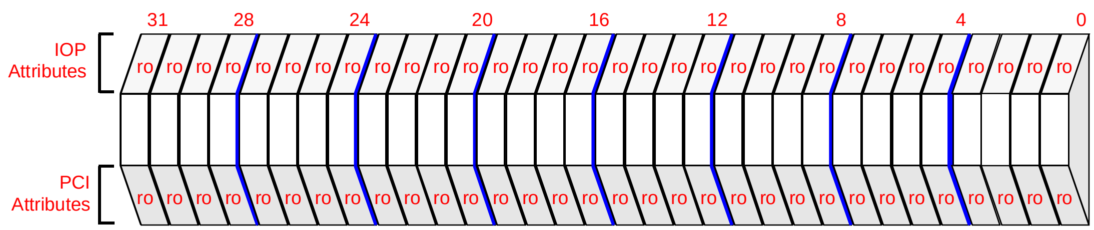
2.14.59 ECC Second Address Register - ECCSAR

When the ECC Error Phase register (bits 6:4 of the ECCCSR) is non-zero (indicating that an error has been captured) and the failing transaction included a dual address cycle (DAC), the ECCSAR register indicates the contents of the **P_AD[31:0]** bus (for 64- and 32-bit buses) for the second address phase of the transaction that included the error. When the ECC Error Phase register is zero, the contents of this register are undefined.

Note: Registers that store information from the failing transaction always store information directly from the bus (uncorrected), even when correction of the error is possible.

Note: The "ECC Control and Status Register - ECCCSR", "ECC First Address Register - ECCFAR", "ECC Second Address Register - ECCSAR", and "ECC Attribute Register - ECCAR" report the actual transaction that has the error. For example, when the Split Completion of an original Outbound Read request has an error, the information regarding the Split Completion is reported.

Table 86. ECC Second Address Register - ECCSAR



Register Offset
+0E0H

Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
31:00	0000 0000H	ECC Second Address - This register represents the most significant 32-bits of the address for a DAC transaction. ***Note:*** When the failing transaction used a single address cycle (SAC), then the contents of this register must be 0.

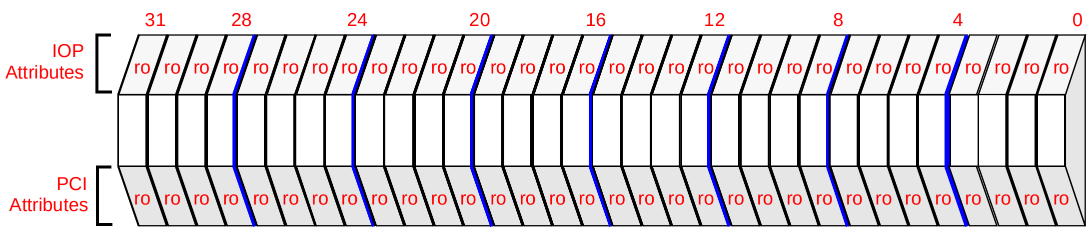
2.14.60 ECC Attribute Register - ECCAR

When the ECC Error Phase register (bits 6:4 of the ECCCSR) is non-zero (indicating that an error has been captured), the ECCAR register indicates the contents of the **P_AD[31:0]** bus (for 64- and 32-bit buses) for the attribute phase of the transaction that included the error. When the ECC Error Phase register is zero, the contents of this register are undefined.

Note: Registers that store information from the failing transaction always store information directly from the bus (uncorrected), even when correction of the error is possible.

Note: The "ECC Control and Status Register - ECCCSR", "ECC First Address Register - ECCFAR", "ECC Second Address Register - ECCSAR", and "ECC Attribute Register - ECCAR" report the actual transaction that has the error. For example, when the Split Completion of an original Outbound Read request had an error, the information regarding the Split Completion is reported.

Table 87. ECC Attribute Register - ECCAR



Register Offset
+0E4H

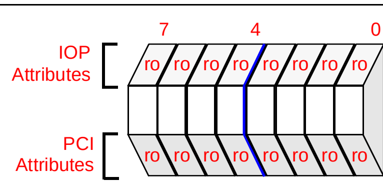
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:00	0000 0000H	ECC Attribute - This register represents contents of the **P_AD[31:0]** bus (for 64- and 32-bit buses) for the attribute phase of a failing transaction.

2.14.61 CompactPCI Hot-Swap Capability ID Register

The following register block provides support of CompactPCI* Hot-Swap functionality.

Table 88. HS_CAPID - Hot-Swap Cap ID



Register Offset
+0E8H

Attribute Legend:
RV = Reserved
RW = Read/Write
RO = Read Only
RT = Read/Toggle
RC = Read/Clear
SW = SROM Write
NA = Not Accessible

Bit	Default	Description
07:00	06h	Identifier (ID): 06h, identifying this linked list structure as being the Compact PCI Hot-Swap register block.

2.14.62 Offset EDh: HS_NXTP - Next Item Pointer

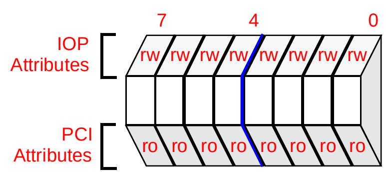
By default, the CompactPCI capability is the last capabilities list for the 4138xx, thus this register defaults to 00H.

However, this register may be written to 90H prior to host configuration to include the VPD capability located at off-set 90H.

Warning: Writing this register to any value other than 00H (default) or 90H is not supported and may produce unpredictable system behavior.

In order to insure that this register is written prior to host configuration, the 4138xx must be initialized at **P_RST#** assertion to Retry Type 0 configuration cycles (bit 2 of PCSR). Typically, the Intel XScale® processor would be enabled to boot immediately following **P_RST#** assertion in this case (bit 1 of PCSR), as well. Please see Section 2.14.41, "PCI Configuration and Status Register - PCSR" on page 178 for more details on the 4138xx initialization modes.

Table 89. HS_NXTP - Next Item Pointer



Register Offset +0E9H

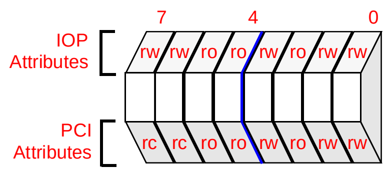
Attribute Legend:
RV = Reserved
RW = Read/Write
RO = Read Only
RT = Read/Toggle
RC = Read/Clear
SW = SROM Write
NA = Not Accessible

Bit	Default	Description
07:00	E8H	Next Capabilities Pointer (PTR): Since the cPCI capabilities are the last in the linked list of extended capabilities in the 4138xx, the register is set to 00H. However, this field may be written prior to host configuration with 90H to extend the list to include the VPD extended capabilities header.

2.14.63 HS_CNTRL - Hot-Swap Control/Status Register

The 4138xx meets the standard requirements to be considered "Hot-Swap Silicon" detailed in the *Compact PCI Hot-Swap Specification*, Revision 2.1. Refer to the *Compact PCI Hot-Swap Specification*, Revision 2.1 for more details on the insertion and extraction processes.

Table 90. HS_CNTRL - Hot-Swap Control/Status Register (Sheet 1 of 2)

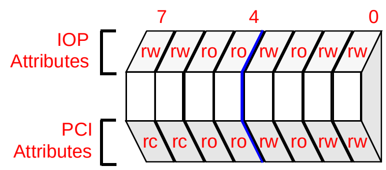


Register Offset
+0EAH

Attribute Legend:	
RV = Reserved	RT = Read/Toggle
RW = Read/Write	RC = Read/Clear
RO = Read Only	SW = SROM Write
	NA = Not Accessible

Bit	Default	Description
07	1b (This bit actually powers up to 0b, however 4138xx sets it to 1b prior to any host software access to the device)	INS: Freshly INSerted board. 4138xx sets this bit to a 1b following the de-assertion of P_RST# provided that L_STAT is sampled low indicating that the ejector handle closed. The INS bit is cleared when software writes a 1b to it. Writing 0b to this bit has no effect. 1 = 4138xx asserts P_ENUM#, (when not masked by bit(1) of this register), to indicate that the card is freshly inserted and is ready to be configured by system software. After system software has cleared this bit (by writing a 1b to it) 4138xx de-asserts P_ENUM# (when currently asserted), and is then armed for a possible future extraction event (EXT bit assertion is enabled).
06	0b	EXT: Pending EXTraction of board. 4138xx sets this bit to a 1b when: • (LOO = 0b or DHA = 0b), and • L_STAT is sampled high while P_RST# is deasserted indicating that the ejector handle is unlocked, and • The board is currently in the INSERTED state (i.e., the INS bit = 0b). The EXT bit is cleared when software writes a 1b. Writing a 0b has no effect. When 1b: 4138xx asserts P_ENUM#, (when not masked by bit(1) of this register), to indicate that the card is about to be removed.
05:04	01b	PI: Programming Interface This field is hard-wired to 01b indicating that 4138xx supports Device Hiding and PIE bit functionality.
03	0b	LOO: LED On/Off (LOO) Control. Allows software control of the LED. 0 = 4138xx drives LED_OUT low turning the external LED off. 1 = 4138xx drives LED_OUT high illuminating the external LED. ***Note:*** Additional external LED control logic must be ORed with 4138xx LED_OUT signal to ensure that the blue LED is illuminated while P_RST# is asserted or when the board is in the H0, H1, or H1F cPCI Hot-Swap defined hardware states.

Table 90. HS_CNTRL - Hot-Swap Control/Status Register (Sheet 2 of 2)



Register Offset
+0EAH

Attribute Legend:	
RV = Reserved	RT = Read/Toggle
RW = Read/Write	RC = Read/Clear
RO = Read Only	SW = SROM Write
	NA = Not Accessible

Bit	Default	Description
02	0b	PIE: Pending Insertion/Extraction This status bit is set and cleared by the 4138xx Hot-Swap state Machine. When 1b this bit indicates that either an insertion or an extraction is in progress (either INS or EXT has a value of 1b or INS is armed).
01	0b	EIM: ENUM# Interrupt Mask. When 0b: 4138xx asserts P_ENUM# when an insertion or removal event occurs as indicated by the setting of the INS or EXT bits of this register. When 1b: 4138xx does not assert P_ENUM# under any circumstances.
00	0b	DHA: Device Hiding Armed: When 1b: When **HS_SM#** = 0b, and LSTAT = 1b (Switch open) and the LOO bit = 1b, 4138xx completes any bus cycles presently in process and then cease to initiate or respond to either primary or secondary bus cycles. When LSTAT subsequently goes low, 4138xx responds normally to bus cycles. When 0: 4138xx operates normally.

2.14.64 Inbound ATU Base Address Register 3 - IABAR3

The Inbound ATU Base Address Register 3 (IABAR3) together with the Inbound ATU Upper Base Address Register 3 (IAUBAR3) defines the block of memory addresses where the inbound translation window 3 begins. The inbound ATU decodes and forwards the bus request to the 4138xx internal bus with a translated address to map into 4138xx local memory. The IABAR3 and IAUBAR3 define the base address and describe the required memory block size; see Section 2.14.23, "Determining Block Sizes for Base Address Registers" on page 164. Bits 31 through 12 of the IABAR3 is either read/write bits or read only with a value of 0 depending on the value located within the IALR3. This configuration allows the IABAR3 to be programmed per *PCI Local Bus Specification*, Revision 2.3.

The programmed value within the base address register must comply with the PCI programming requirements for address alignment. Refer to the *PCI Local Bus Specification*, Revision 2.3 for additional information on programming base address registers.

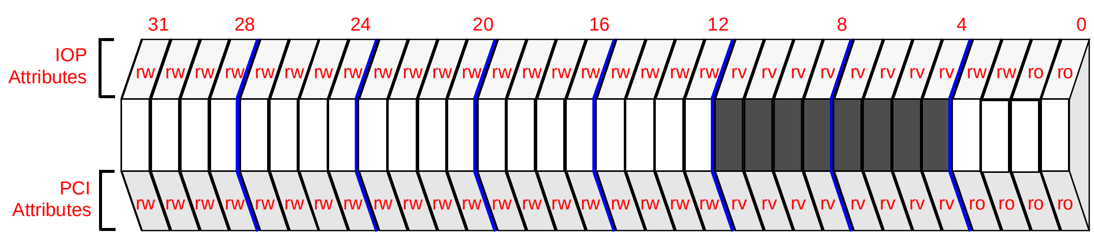
Note: Since IABAR3 does not appear in the standard PCI configuration header space (offsets 00H - 3CH), IABAR3 is not configured by the host during normal system initialization.

Warning: When a non-zero value is not written to IALR3, the user should not set either the Prefetchable Indicator or the Type Indicator for 64 bit addressability. This is the default for IABAR3. Assuming a non-zero value is written to IALR3, the user may set the Prefetchable Indicator or the Type Indicator:

a. Since non prefetchable memory windows can never be placed above the 4 Gbyte address boundary, when the Prefetchable Indicator is not set, the user should also leave the Type Indicator set for 32 bit addressability. This is the default for IABAR3.

b. For compliance to the *PCI-X Protocol Addendum to the PCI Local Bus Specification,* Revision 2.0, when the Prefetchable Indicator is set, the user should also set the Type Indicator for 64 bit addressability.

Table 91. Inbound ATU Base Address Register 3 - IABAR3



Register Offset
+200H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:12	00000H	Translation Base Address 3 - These bits define the actual location the translation function is to respond to when addressed from the PCI bus.
11:04	00H	Reserved.
03	0_2	Prefetchable Indicator - When set, defines the memory space as prefetchable.
02:01	00_2	Type Indicator - Defines the width of the addressability for this memory window: 00 - Memory Window is locatable anywhere in 32 bit address space 10 - Memory Window is locatable anywhere in 64 bit address space
00	0_2	Memory Space Indicator - This bit field describes memory or I/O space base address. The ATU does not occupy I/O space, thus this bit must be zero.

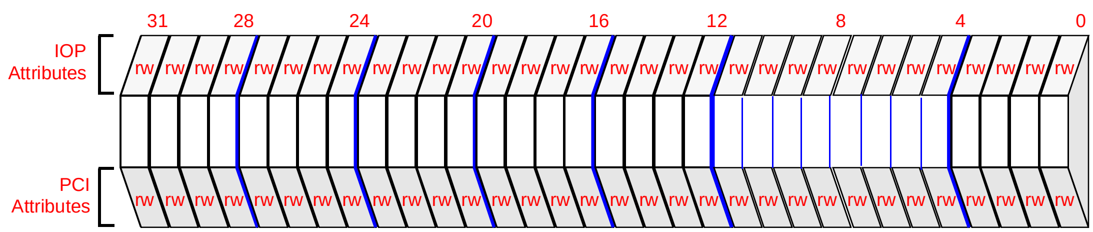
2.14.65 Inbound ATU Upper Base Address Register 3 - IAUBAR3

This register contains the upper base address when decoding PCI addresses beyond 4 GBytes. Together with the Translation Base Address this register defines the actual location the translation function is to respond to when addressed from the PCI bus for addresses > 4GBytes (for DACs).

The programmed value within the base address register must comply with the PCI programming requirements for address alignment. Refer to the *PCI Local Bus Specification*, Revision 2.3 for additional information on programming base address registers.

Note: When the Type indicator of IABAR3 is set to indicate 32 bit addressability, the IAUBAR3 register attributes are read-only. By default the IAUBAR3 register has read-only attributes. Prior to changing the Type Indicator in the IABAR3 to support 32-bit addressability, the IAUBAR3 must be written with zero unless it already contains zero. Zero is the default value for IAUBAR3.

Table 92. Inbound ATU Upper Base Address Register 3 - IAUBAR3



Register Offset
+204H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:0	00000H	Translation Upper Base Address 3 - Together with the Translation Base Address 3 these bits define the actual location the translation function is to respond to when addressed from the PCI bus for addresses > 4GBytes.

2.14.66 Inbound ATU Limit Register 3 - IALR3

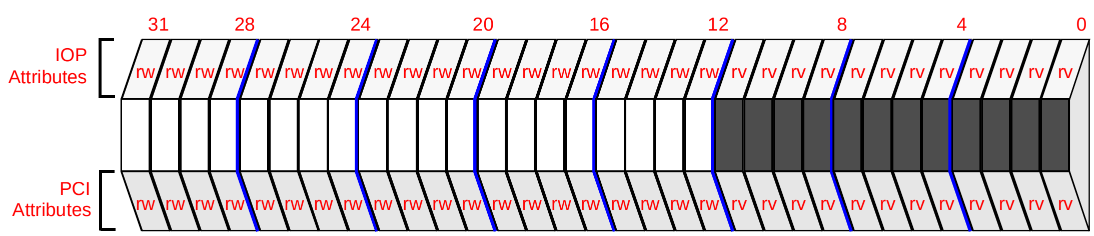
Inbound address translation for memory window 3 occurs for data transfers occurring from the PCI bus (originated from the PCI bus) to the 4138xx internal bus. The address translation block converts PCI addresses to internal bus addresses.

The inbound translation base address for inbound window 3 is specified in Section 2.14.17. When determining block size requirements — as described in Section 2.14.23 — the translation limit register provides the block size requirements for the base address register. The remaining registers used for performing address translation are discussed in Section 2.2.1.1.

The 4138xx value register's programmed value must be naturally aligned with the base address register's programmed value. The limit register is used as a mask; thus, the lower address bits programmed into the 4138xx value register are invalid. Refer to the *PCI Local Bus Specification*, Revision 2.3 for additional information on programming base address registers.

Bits 31 to 12 within the IALR3 have a direct effect on the IABAR3 register, bits 31 to 12, with a one to one correspondence. A value of 0 in a bit within the IALR3 makes the corresponding bit within the IABAR3 a read only bit which always returns 0. A value of 1 in a bit within the IALR3 makes the corresponding bit within the IABAR3 read/write from PCI. Note that a consequence of this programming scheme is that unless a valid value exists within the IALR3, all writes to the IABAR3 has no effect since a value of all zeros within the IALR3 makes the IABAR3 a read only register.

Table 93. Inbound ATU Limit Register 3 - IALR3



Register Offset
+208H

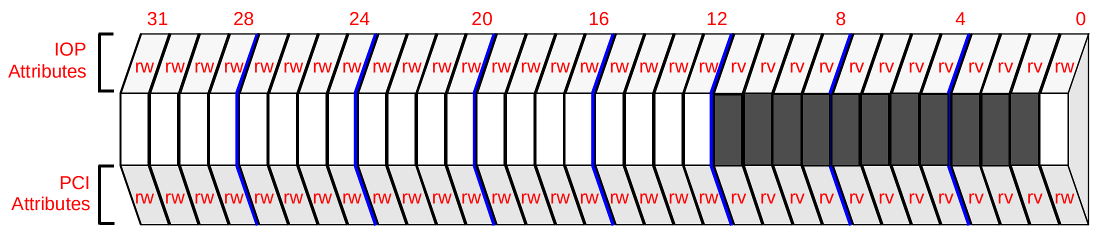
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:12	00000H	Inbound Translation Limit 3 - This value determines the memory block size required for the ATUs memory window 3.
11:00	000H	Reserved

2.14.67 Inbound ATU Translate Value Register 3 - IATVR3

The Inbound ATU Translate Value Register 3 (IATVR3) in conjunction with the "Inbound ATU Upper Translate Value Register 3 - IAUTVR3" on page 207 contain bits 35 to 12 of the internal bus address used to convert PCI bus addresses. The converted address is driven on the internal bus as a result of the inbound ATU address translation.

Table 94. Inbound ATU Translate Value Register 3 - IATVR3



Register Offset
+20CH

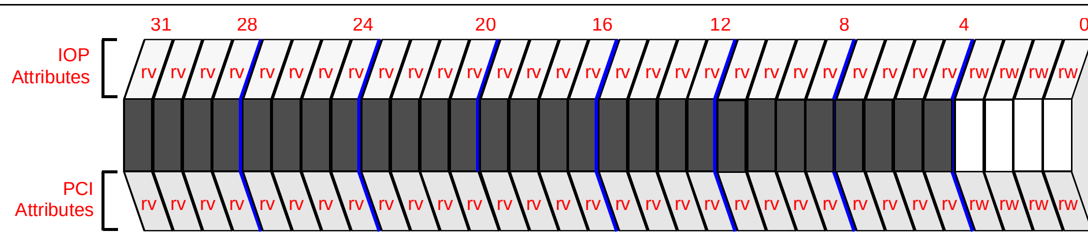
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:12	00000H	Inbound ATU Translation Value 3 - Bits 31 to 12 of internal bus address used to convert the PCI address to internal bus addresses. This value must be naturally aligned with the IABAR3 register's programmed value (see Section 2.14.23, "Determining Block Sizes for Base Address Registers" on page 164).
11:01	000H	Reserved
00	0	Big Endian Byte Swap enable - When set the ATU performs a byte swap on all PCI read/write transactions through BAR3. When clear, no swap is performed. Refer to Section 2.3, "Big Endian Byte Swapping" on page 78 for more details.

2.14.68 Inbound ATU Upper Translate Value Register 3 - IAUTVR3

The Inbound ATU Upper Translate Value Register 3 (IAUTVR3) in conjunction with the "Inbound ATU Translate Value Register 3 - IATVR3" on page 207 contain bits 35 to12 of the internal bus address used to convert PCI bus addresses. The converted address is driven on the internal bus as a result of the inbound ATU address translation.

Table 95. Inbound ATU Upper Translate Value Register 3 - IAUTVR3



Register Offset
+210H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

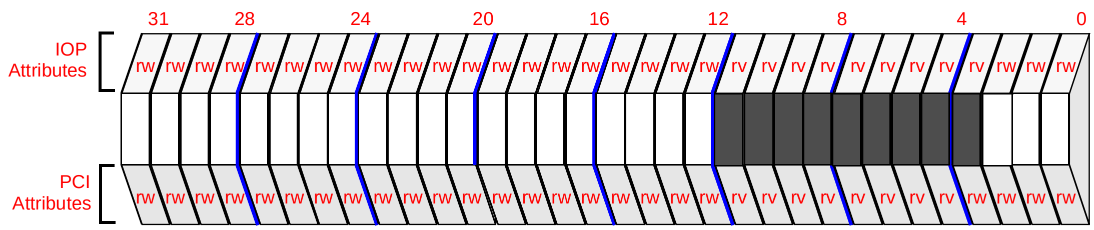
Bit	Default	Description
31:04	000 0000H	Reserved
3:0	0H	Inbound Upper ATU Translation Value 3 - This value represents bits 35 to 32 of the internal bus address used to convert the PCI address to internal bus addresses.

2.14.69 Outbound I/O Base Address Register - OIOBAR

The OIOBAR register locates the 64 KB I/O cycle address window in the 4138xx's 64 Gbyte internal address space. When A[35:16] of the internal bus address matches the value in OIOBAR, the ATU claims the transaction and forward it over to the PCI interface as an I/O cycle.

Note: In translating the internal bus address A[35:0] for the PCI bus I/O cycle, A[15:0] is forwarded over to the PCI bus unmodified while A[31:16] is set to 0000H. (see "I/O Transactions" on page 71).

Table 96. Outbound I/O Base Address Register - OIOBAR



Register Offset
+300H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

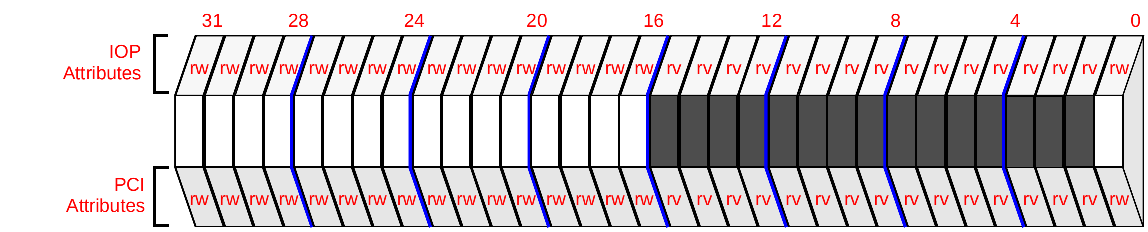
Bit	Default	Description
31:12	0 FFFBH	Outbound I/O Base Address - This value represents bits 35 to 16 of the internal bus address used to claim an outbound I/O cycle request for the ATU.
11:3	000H	Reserved
2:0	0H	Outbound I/O Function Number Mapping - The Function number in this field is used as part of the Requestor ID for outbound I/O transactions

2.14.70 Outbound I/O Window Translate Value Register - OIOWTVR

The Outbound I/O Window Translate Value Register (OIOWTVR) contains the PCI I/O address used to convert the internal bus access to a PCI address. This address is driven on the PCI bus as a result of the outbound ATU address translation. See Section 2.2.2.1, "Outbound Address Translation - Internal Bus Transactions" on page 68 for details on outbound address translation.

The I/O window is from 4138xx internal bus is set via the "Outbound I/O Base Address Register - OIOBAR" with the a fixed length of 64 Kbytes.

Table 97. Outbound I/O Window Translate Value Register - OIOWTVR



Register Offset
+304H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

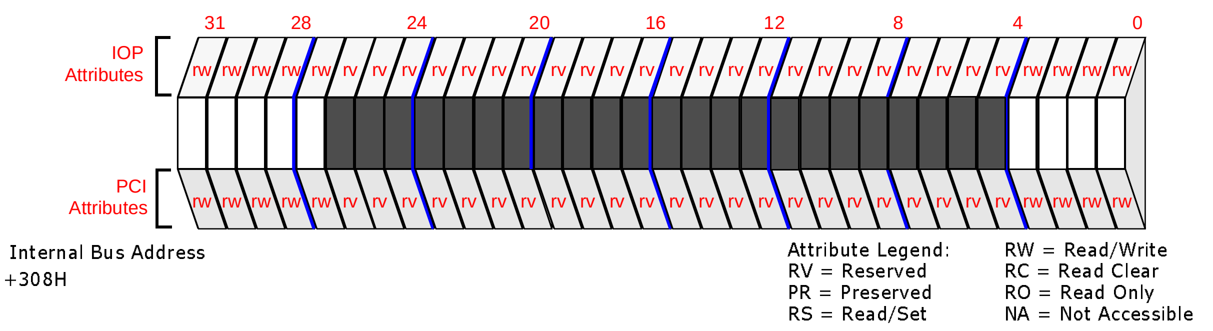
Bit	Default	Description
31:16	0000H	Outbound I/O Window Translate Value - Used to convert internal bus addresses to PCI addresses.
15:01	0000H	Reserved
00	0	Big Endian Byte Swap enable - When set the ATU performs a byte swap on all read/write transactions through the OIOBAR. When clear, no swap is performed. Refer to Section 2.3, "Big Endian Byte Swapping" on page 78 for more details.

2.14.71 Outbound Upper Memory Window Base Address Register 0 - OUMBAR0

The OUMBAR0 register locates Outbound Memory Window 0 in a 4 Gbyte Memory section in the 4138xx 64 Gbyte internal address space. When A[35:32] of the internal bus address matches the value in OUMBAR0[3:0], the ATU claims the transaction and forward it over to the PCI interface.

Note: In translating the internal bus address A[35:0], A[31:0] is forwarded over to the PCI bus unmodified. The ATU constructs a 64 bit PCI address in conjunction with the "Outbound Upper 32-bit Memory Window Translate Value Register 0 - OUMWTVR0" on page 211.

Table 98. Outbound Upper Memory Window Base Address Register 0 - OUMBAR0

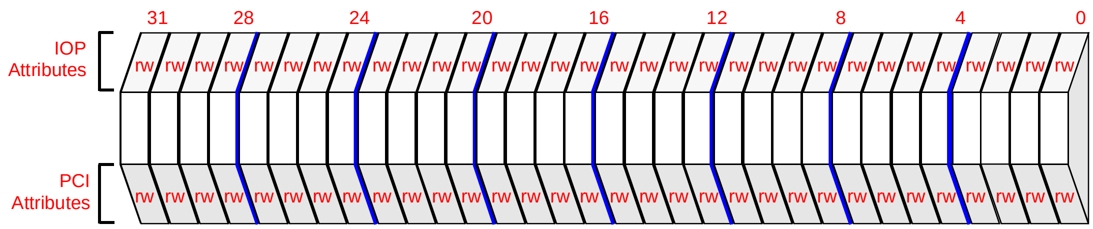


Bit	Default	Description
31	1_2	Outbound Window 0 Enable - When set, the ATU Outbound Memory Window 0 claims internal bus transactions when A[35:32] matches OUMBAR0[3:0].
30:28	000_2	Outbound Window 0 Function Number Mapping - Errors with Outbound Memory Window 0 transactions (PCI or Internal Bus) is associated with the function number contained in this field.
27	0	Big Endian Byte Swap enable - When set the ATU performs a byte swap on all read/write transactions through this BAR. When clear, no swap is performed. Refer to Section 2.3, "Big Endian Byte Swapping" on page 78 for more details.
26:04	000 0000H	Reserved
3:0	1H	Outbound Upper Memory Window Base Address 0 - This value represents bits 35 to 32 of the internal bus address used to claim an outbound memory request. ***Note:*** The ATUE has the same default value. When both ATUs exist and the outbound windows of the both ATUs are enabled, the bit field must be reprogrammed to avoid a conflict.

2.14.72 Outbound Upper 32-bit Memory Window Translate Value Register 0 - OUMWTVR0

The Outbound Upper 32-bit Memory Window Translate Value Register 0 (OUMWTVR0) defines the upper 32-bits of address used during a dual address cycle. This enables the outbound ATU to directly address anywhere within the 64-bit host address space. When this register is all-zero, then a SAC is generated on the PCI bus.

Table 99. Outbound Upper 32-bit Memory Window Translate Value Register 0- OUMWTVR0



Register Offset
+30CH

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

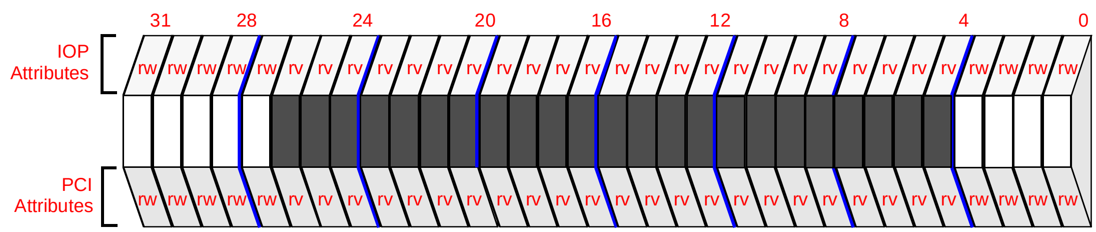
Bit	Default	Description
31:00	0000 0000H	These bits define the upper 32-bits of address driven during the dual address cycle (DAC).

2.14.73 Outbound Upper Memory Window Base Address Register 1 - OUMBAR1

The OUMBAR1 register locates Outbound Memory Window 1 in a 4 Gbyte Memory section in the 4138xx 64 Gbyte internal address space. When A[35:32] of the internal bus address matches the value in OUMBAR1[3:0], the ATU claims the transaction and forward it over to the PCI interface.

Note: In translating the internal bus address A[35:0], A[31:0] is forwarded over to the PCI bus unmodified. The ATU constructs a 64 bit PCI address in conjunction with the "Outbound Upper 32-bit Memory Window Translate Value Register 1 - OUMWTVR1" on page 213.

Table 100. Outbound Upper Memory Window Base Address Register 1 - OUMBAR1



Register Offset
+310H

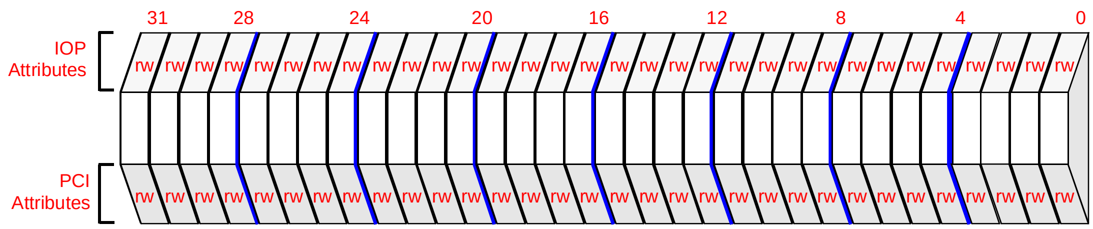
Attribute Legend:	RW = Read/Write
RV = Reserved	RC = Read Clear
PR = Preserved	RO = Read Only
RS = Read/Set	NA = Not Accessible

Bit	Default	Description
31	1_2	Outbound Window 1 Enable - When set, the ATU Outbound Memory Window 1 claims internal bus transactions when A[35:32] matches OUMBAR1[3:0].
30:28	000_2	Outbound Window 1 Function Number Mapping - Errors with Outbound Memory Window 1 transactions (PCI or Internal Bus) is associated with the function number contained in this field.
27	0	Big Endian Byte Swap enable - When set the ATU performs a byte swap on all read/write transactions through this BAR. When clear, no swap is performed. Refer to Section 2.3, "Big Endian Byte Swapping" on page 78 for more details.
26:04	000 0000H	Reserved
3:0	2H	Outbound Upper Memory Window Base Address 1 - This value represents bits 35 to 32 of the internal bus address used to claim an outbound memory request. ***Note:*** The ATUE has the same default value. When both ATUs exist and the outbound windows of the both ATUs are enabled, the bit field must be reprogrammed to avoid a conflict.

2.14.74 Outbound Upper 32-bit Memory Window Translate Value Register 1 - OUMWTVR1

The Outbound Upper 32-bit Memory Window Translate Value Register 1 (OUMWTVR1) defines the upper 32-bits of address used during a dual address cycle. This enables the outbound ATU to directly address anywhere within the 64-bit host address space. When this register is all-zero, then a SAC is generated on the PCI bus.

Table 101. Outbound Upper 32-bit Memory Window Translate Value Register 1- OUMWTVR1



Register Offset
+314H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

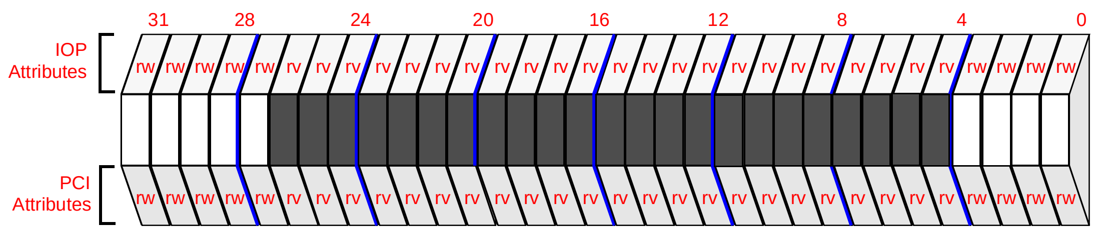
Bit	Default	Description
31:00	0000 0000H	These bits define the upper 32-bits of address driven during the dual address cycle (DAC).

2.14.75 Outbound Upper Memory Window Base Address Register 2 - OUMBAR2

The OUMBAR2 register locates Outbound Memory Window 2 in a 4 Gbyte Memory section in the 4138xx 64 Gbyte internal address space. When A[35:32] of the internal bus address matches the value in OUMBAR2[3:0], the ATU claims the transaction and forward it over to the PCI interface.

Note: In translating the internal bus address A[35:0], A[31:0] is forwarded over to the PCI bus unmodified. The ATU constructs a 64 bit PCI address in conjunction with the "Outbound Upper 32-bit Memory Window Translate Value Register 2 - OUMWTVR2" on page 215.

Table 102. Outbound Upper Memory Window Base Address Register 2- OUMBAR2



Internal Bus Address
+318H

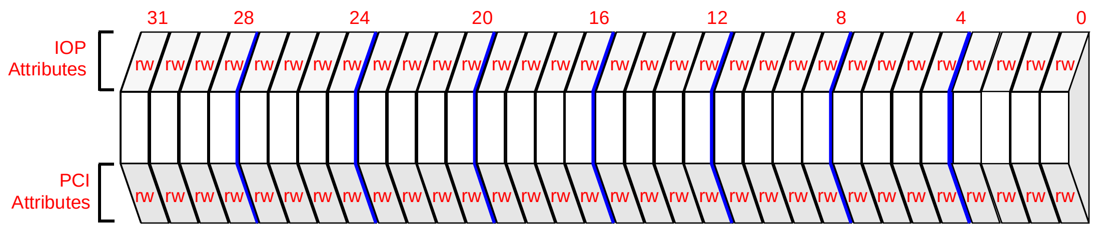
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31	0_2	Outbound Window 2 Enable - When set, the ATU Outbound Memory Window 2 claims internal bus transactions when A[35:32] matches OUMBAR2[3:0].
30:28	000_2	Outbound Window 2 Function Number Mapping - Errors with Outbound Memory Window 2 transactions (PCI or Internal Bus) is associated with the function number contained in this field.
27	0	Big Endian Byte Swap enable - When set the ATU performs a byte swap on all read/write transactions through this BAR. When clear, no swap is performed. Refer to Section 2.3, "Big Endian Byte Swapping" on page 78 for more details.
26:04	000 0000H	Reserved
3:0	3H	Outbound Upper Memory Window Base Address 2 - This value represents bits 35 to 32 of the internal bus address used to claim an outbound memory request. ***Note:*** The ATUE has the same default value. When both ATUs exist and the outbound windows of the both ATUs are enabled, the bit field must be reprogrammed to avoid a conflict.

2.14.76 Outbound Upper 32-bit Memory Window Translate Value Register 2 - OUMWTVR2

The Outbound Upper 32-bit Memory Window Translate Value Register 2 (OUMWTVR2) defines the upper 32-bits of address used during a dual address cycle. This enables the outbound ATU to directly address anywhere within the 64-bit host address space. When this register is all-zero, then a SAC is generated on the PCI bus.

Table 103. Outbound Upper 32-bit Memory Window Translate Value Register 2- OUMWTVR2



Internal Bus Address
+31CH

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

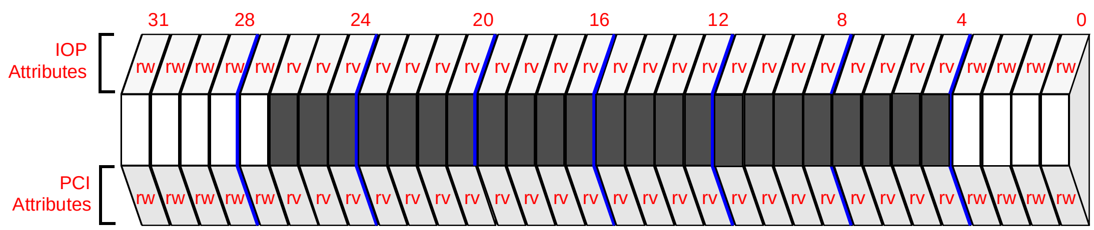
Bit	Default	Description
31:00	0000 0000H	These bits define the upper 32-bits of address driven during the dual address cycle (DAC).

2.14.77 Outbound Upper Memory Window Base Address Register 3 - OUMBAR3

The OUMBAR3 register locates Outbound Memory Window 3 in a 4 Gbyte Memory section in the 4138xx 64 Gbyte internal address space. When A[35:32] of the internal bus address matches the value in OUMBAR3[3:0], the ATU claims the transaction and forward it over to the PCI interface.

Note: In translating the internal bus address A[35:0], A[31:0] is forwarded over to the PCI bus unmodified. The ATU constructs a 64 bit PCI address in conjunction with the "Outbound Upper 32-bit Memory Window Translate Value Register 3 - OUMWTVR3" on page 217.

Table 104. Outbound Upper Memory Window Base Address Register 3 - OUMBAR3



Register Offset
+320H

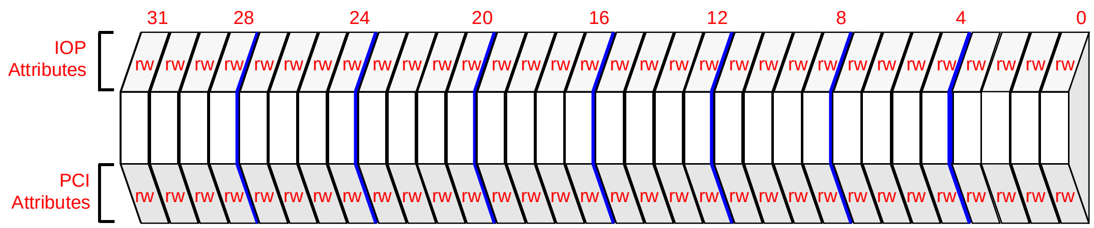
Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
31	0_2	Outbound Window 3 Enable - When set, the ATU Outbound Memory Window 3 claims internal bus transactions when A[35:32] matches OUMBAR3[3:0].
30:28	000_2	Outbound Window 3 Function Number Mapping - Errors with Outbound Memory Window 3 transactions (PCI or Internal Bus) is associated with the function number contained in this field.
27	0	Big Endian Byte Swap enable - When set the ATU performs a byte swap on all read/write transactions through this BAR. When clear, no swap is performed. Refer to Section 2.3, "Big Endian Byte Swapping" on page 78 for more details.
26:04	000 0000H	Reserved
3:0	4H	Outbound Upper Memory Window Base Address 3 - This value represents bits 35 to 32 of the internal bus address used to claim an outbound memory request. ***Note:*** The ATUE has the same default value. When both ATUs exist and the outbound windows of the both ATUs are enabled, the bit field must be reprogrammed to avoid a conflict.

2.14.78 Outbound Upper 32-bit Memory Window Translate Value Register 3 - OUMWTVR3

The Outbound Upper 32-bit Memory Window Translate Value Register 3 (OUMWTVR3) defines the upper 32-bits of address used during a dual address cycle. This enables the outbound ATU to directly address anywhere within the 64-bit host address space. When this register is all-zero, then a SAC is generated on the PCI bus.

Table 105. Outbound Upper 32-bit Memory Window Translate Value Register 3- OUMWTVR3



Register Offset
+324H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

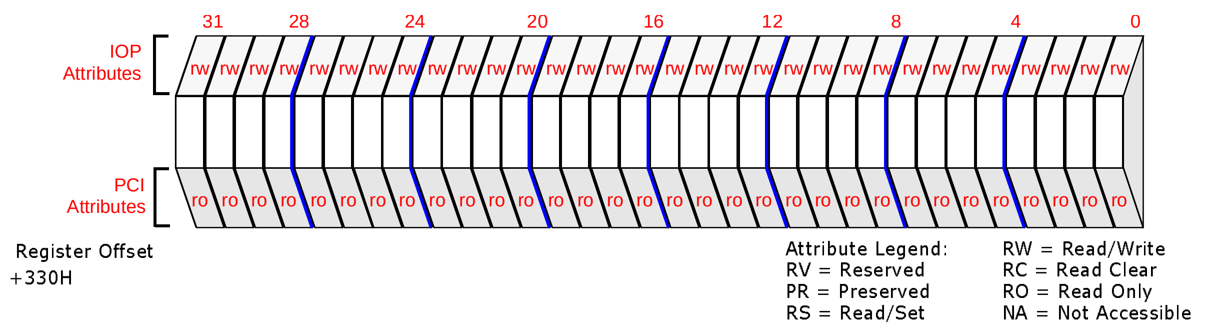
Bit	Default	Description
31:00	0000 0000H	These bits define the upper 32-bits of address driven during the dual address cycle (DAC).

2.14.79 Outbound Configuration Cycle Address Register - OCCAR

The Outbound Configuration Cycle Address Register is used to hold the 32-bit PCI configuration cycle address. The Intel XScale® processor writes the PCI configuration cycles address, which enables outbound configuration read or write. The Intel XScale® processor then performs a read or write to the Outbound Configuration Cycle Data Register to initiate the configuration cycle on the PCI bus.

Note: Bits 15:11 of the configuration cycle address for Type 0 configuration cycles are defined differently for Conventional versus PCI-X modes. When 4138xx software programs the OCCAR to initiate a Type 0 configuration cycle, the OCCAR should always be loaded based on the PCI-X definition for the Type 0 configuration cycle address. When operating in Conventional mode, the 4138xx clears bits 15:11 of the OCCAR prior to initiating an outbound Type 0 configuration cycle. See the *PCI-X Protocol Addendum to the PCI Local Bus Specification,* Revision 2.0 for details on the two formats.

Table 106. Outbound Configuration Cycle Address Register - OCCAR



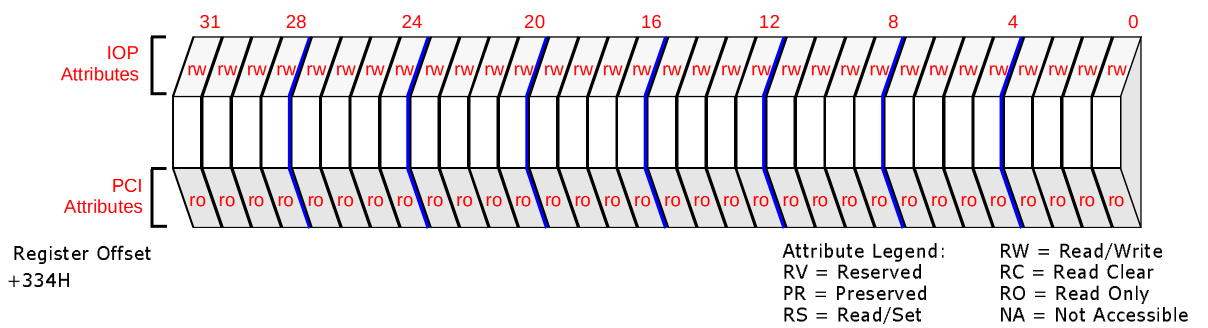
Bit	Default	Description
31:00	0000 0000H	Configuration Cycle Address - These bits define the 32-bit PCI address used during an outbound configuration read or write cycle.

2.14.80 Outbound Configuration Cycle Data Register - OCCDR

The Outbound Configuration Cycle Data Register is used to initiate a configuration read or write on the PCI bus. The register is logical rather than physical meaning that it is an address not a register. The Intel XScale® processor reads or writes the data registers memory-mapped address to initiate the configuration cycle on the PCI bus with the address found in the OCCAR. For a configuration write, the data is latched from the internal bus and forwarded directly to the OWQ. For a read, the data is returned directly from the ORQ to the Intel XScale® processor and is never actually entered into the data register (which does not physically exist).

The OCCDR is only visible from 4138xx internal bus address space and appears as a reserved value within the ATU configuration space.

Table 107. Outbound Configuration Cycle Data Register - OCCDR



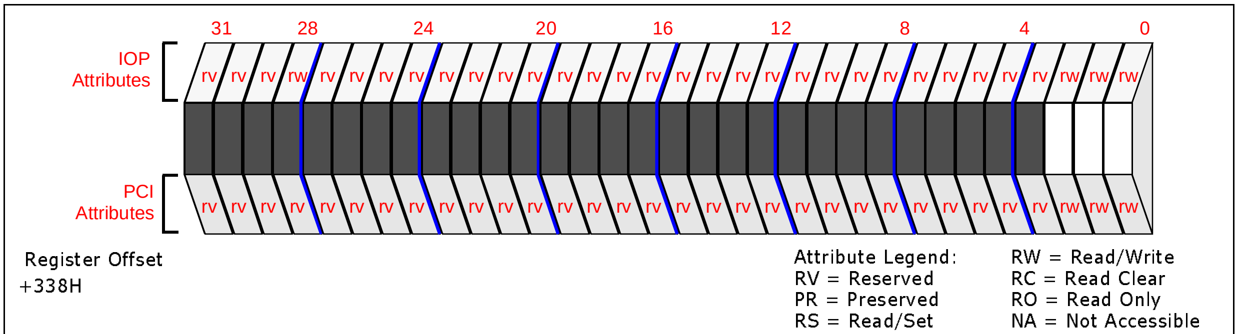
Bit	Default	Description
31:00	0000 0000H	Configuration Cycle Data - Define the data used during an outbound configuration read or write cycle.

2.14.81 Outbound Configuration Cycle Function Number - OCCFN

This register contains the Requester ID function number used for all outbound configuration requests. This field is also used to determine where errors get logged.

For 4138xx the function number should be 0 for endpoint usage and match the ATUX function number (readable in "PCI-X Status Register - PCIXSR") for Root Complex modes.

Table 108. Outbound Configuration Cycle Function Number Register - OCCFN



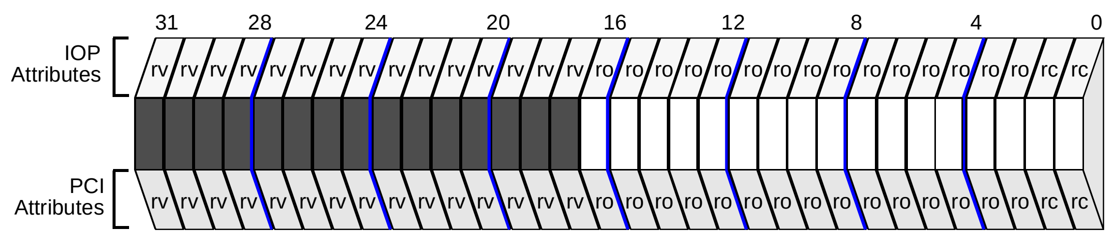
Bit	Default	Description
31:00	0000 0000H	Reserved
2:0	0H	Configuration Cycle Function Number - These field is used as part of the Requester ID for outbound configuration cycles.

2.14.82 PCI Interface Error Control and Status Register - PIECSR

This register indicates whether or not the ATU has detected and logged a PCI interface error. The register is also used to enabled the logging of additional errors. For more details, see Section 2.7, "ATU Error Conditions" on page 94.

Note: The "PCI Interface Error Control and Status Register - PIECSR", "PCI Interface Error Address Register - PCIEAR", and "PCI Interface Error Upper Address Register - PCIEUAR" report the original transaction when an error is detected on the current transaction. For example, when the Split Completion of an original Outbound Read request had an error, the information regarding the Outbound Read is reported.

Table 109. PCI Interface Error Control and Status Register - PIECSR



Register Offset
+380H

Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

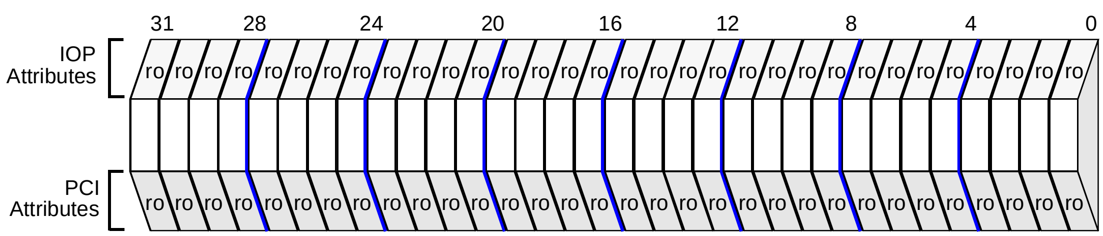
Bit	Default	Description
31:17	0000 0000H	Reserved
16:13	0H	Error Command - This field indicates the PCI command when PCIECSR bit 0 is set: This field represents the command type used during the PCI transaction other than the DAC command (when Dual Address Cycle). The 4-bit code in this field can be referenced to the actual command type via Table 2-8, "PCI-X Command Encoding" of the *PCI-X Protocol Addendum to the PCI Local Bus Specification,* Revision 2.0.
12:09	0000_2	Error Type - This field indicates the type of PCI interface error when PCIECSR bit 0 is set: 0000 Address Parity Error 0001 Data Parity Error 0010 Master Abort 0011 Target Abort 0100 Received Split Completion Error Message 0101 Unexpected Split Completion 0110 Split Completion Discarded 0111 Internal Bus Data Parity Error
08:05	0000_2	Initiator ID - When PCIECSR bit 0 is set, this Indicates the initiator associated with the detected PCI interface error. "0000" is used to denote inbound requests where the IOP was the completer of the transaction. See Internal Bus Requester IDs in the System Controller chapter for remaining details of Initiator ID.
04:02	000_2	PCI Function Number - When PCIECSR bit 0 is set, this field indicates the device function number that this error is associated with. ***Note:*** This field is undefined when the Error Type (bits 12:09) indicates an Address Parity Error.
01	0_2	Multiple PCI Interface Errors Detected - This bit is set when a PCI Interface Error has already been logged (bit 0 of the PIECSR is set), but an additional error is detected. Note that a second error is logged for a subsequent transaction.
00	0_2	PCI Interface Error Detected - This is bit is set when a PCI Interface Error is detected. When this register is cleared, the 4138xx is enabled to latch information about a PCI Interface error.

2.14.83 PCI Interface Error Address Register - PCIEAR

When PCIECSR bit 0 is set, this register represents the lower 32-bits of the address for the error detected on the PCI Bus. Note that for a DAC cycle the address may be 64-bit. This register is used in conjunction with Section 2.14.84, "PCI Interface Error Upper Address Register - PCIEUAR" on page 222 in order to interpret the entire 64-bit PCI address for the error. One error can be detected and logged. The software knows which PCI address had the error by reading this register and decoding the contents of the PCIECSR. For error details, see Section 2.7, "ATU Error Conditions" on page 94).

Note: The "PCI Interface Error Control and Status Register - PIECSR", "PCI Interface Error Address Register - PCIEAR", and "PCI Interface Error Upper Address Register - PCIEUAR" report the original transaction when an error is detected on the current transaction. For example, when the Split Completion of an original Outbound Read request had an error, the information regarding the Outbound Read is reported.

Table 110. PCI Interface Error Address Register - PCIEAR



Register Offset
+384H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

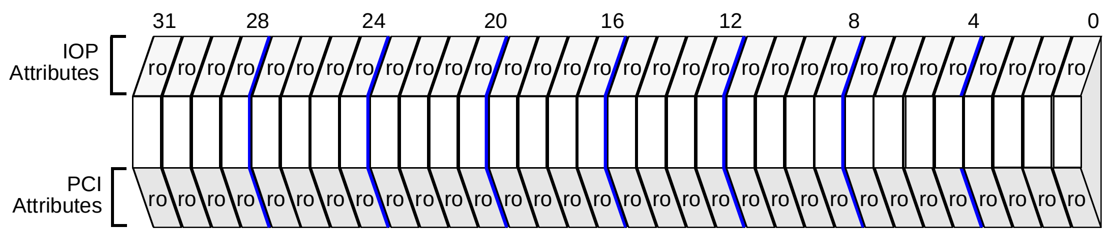
Bit	Default	Description
31:00	0000 0000H	Error Address: When PCIECSR bit 0 is set, this register represents the lower 32-bits of the PCI Address.

2.14.84 PCI Interface Error Upper Address Register - PCIEUAR

When PCIECSR bit 0 is set and the PCI error detected included a DAC cycle, this register represents the upper 32-bit address of where the error was detected on the PCI bus. This register is used in conjunction with the Section 2.14.83, “PCI Interface Error Address Register - PCIEAR” on page 221. One error can be detected and logged. The software knows which PCI address had the error by reading this register and decoding contents of the PCIECSR. For error details, see Section 2.7, “ATU Error Conditions” on page 94).

Note: The “PCI Interface Error Control and Status Register - PIECSR”, “PCI Interface Error Address Register - PCIEAR”, and “PCI Interface Error Upper Address Register - PCIEUAR” report the original transaction when an error is detected on the current transaction. For example, when the Split Completion of an original Outbound Read request had an error, the information regarding the Outbound Read is reported.

Table 111. PCI Interface Error Upper Address Register - PCIEUAR



Register Offset
+388H

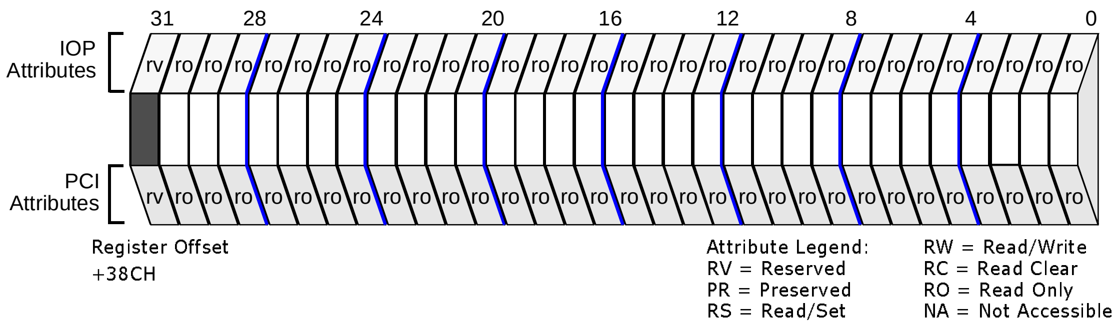
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:00	0000 000H	Upper 32-bit Address - When bit 0 of the PCIECSR is set, this register represents the upper 32 bits of the PCI Address.

2.14.85 PCI Interface Error Context Address Register — PCIECAR

When PCIECSR bit 0 is set, this register contains the DMA Channel Number and bits 30 through 5 of the address of the ADMA descriptor associated with the error detected on the PCI Bus.One error can be detected and logged. The software knows which ADMA descriptor context had the error by reading this register and decoding the contents of the PCIECSR. For error details, see Section 2.7, “ATU Error Conditions” on page 94).

Table 112. PCI Interface Error Context Address Register - PCIECAR

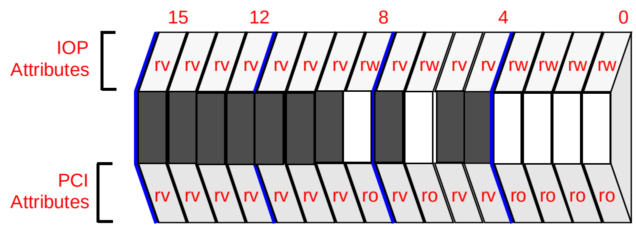


Bit	Default	Description
31	0	Reserved
30:29	00	DMA Channel Number
28:0	0H	ADMA Descriptor Address Bits 30:5

2.14.86 Internal Arbiter Control Register - IACR

The Internal Arbiter Control Register is used to control which priority ring different PCI bus requesters (including the ATU) use. In addition, the method by which the arbiter parks on masters is configurable in this register.

Table 113. Internal Arbiter Control Register - IACR



Register Offset
+394H

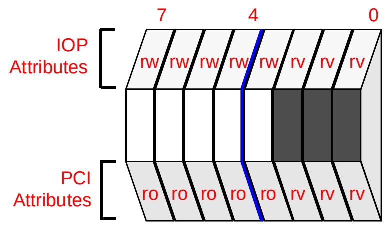
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
15:9	0_2	Reserved.
08	0_2	Bus Parking Control: Controls the bus parking behavior of the internal arbiter: 0 = Bus is parked on the last PCI agent using the bus. 1 = Bus is always parked on 4138xx.
07	0_2	Reserved
06	0_2	ATU Ring Allocation: Priority ring allocation for 4138xx ATU requests: 0 = The ATU is in the low priority ring of the internal arbiter. 1 = The ATU is in the high priority ring of the internal arbiter.
05:04:	00_2	Reserved.
03:00	0000_2	PCI Master Priority Ring Allocation: Bit 0 corresponds to REQ#[0], 1 corresponds to REQ#[1] and so on: 0 = The corresponding master is in the low priority ring of the internal arbiter. 1 = The corresponding master is in the high priority ring of the internal arbiter.

2.14.87 Multi-Transaction Timer - MTT

This register controls the amount of time that the 4138xx arbiter allows a PCI initiator to perform multiple back-to-back transactions on the PCI bus. The number of clocks programmed in the MTT represents the insured time slice (measured in PCI clocks) allotted to the current agent, after which the arbiter grants another agent that is requesting the bus.

Table 114. Multi-Transaction Timer - MTT



Register Offset
+398H

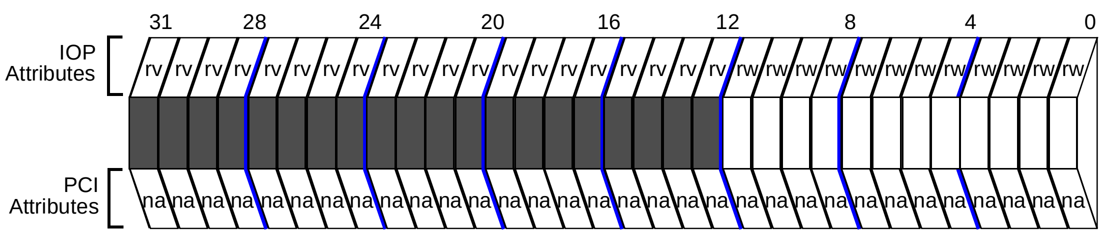
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
07:03	00H	Timer Count Value (MTC): This field specifies the amount of time that grant remains asserted to a master continuously asserting its request for multiple transfers. This field specifies the count in an 8-clock (PCI clock) granularity.
02:00	0H	Reserved

2.14.88 PCIX RCOMP Control Register — PRCR

Warning: In Central Resource mode this register can only be accessed when PCSR bit 21 is cleared. Refer to Table 68, "PCI Configuration and Status Register - PCSR" on page 178.

Table 115. PCIX RCOMP Control Register - PRCR



Intel XScale® processor Local Bus offset

+2100H

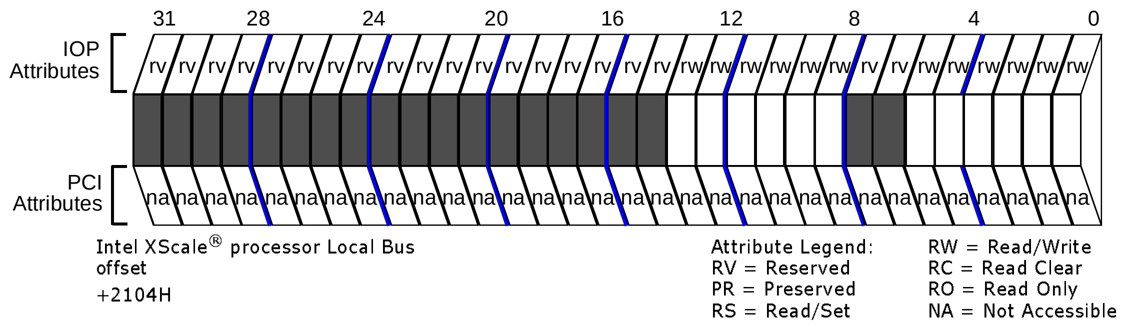
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:12	00000H	Reserved
11	0_2	When this bit is set, the slew rate manual override values are used. When this bit is cleared, the slew rate is derived from the drive strength (normal operation).
10:09	00_2	Reference select for PCIX CAP pad 2'b00 Nominal Reference Voltage 2'b01 -5% Reference 2'b10 +5% Reference
08:07	01_2	Drive strength select for ODT RCOMP pad 2'b00 103.0 ohms 2'b01 110.0 ohms 2'b10 120.0 ohms
06:04	010_2	Drive strength select for RCOMP pad dedicated 3.3V supply voltage 3'b000 18.0 ohms 3'b001 20.1 ohms 3'b010 22.0 ohms 3'b011 24.1 ohms 3'b100 26.1 ohms
03:01	001_2	Drive strength select for RCOMP pad 1.5V switch supply voltage 3'b000 18.0 ohms 3'b001 20.1 ohms 3'b010 22.0 ohms 3'b011 24.1 ohms 3'b100 26.1 ohms
00	1_2	RCOMP pads enable.

2.14.89 PCIX Pad ODT Drive Strength Manual Override Values Registers — PPODSMOVR

Warning: In Central Resource mode this register can only be accessed when PCSR bit 21 is cleared. Refer to Table 68, "PCI Configuration and Status Register - PCSR" on page 178.

Table 116. PCIX Pad ODT Drive Strength Manual Override Values Registers - PPODSMOVR

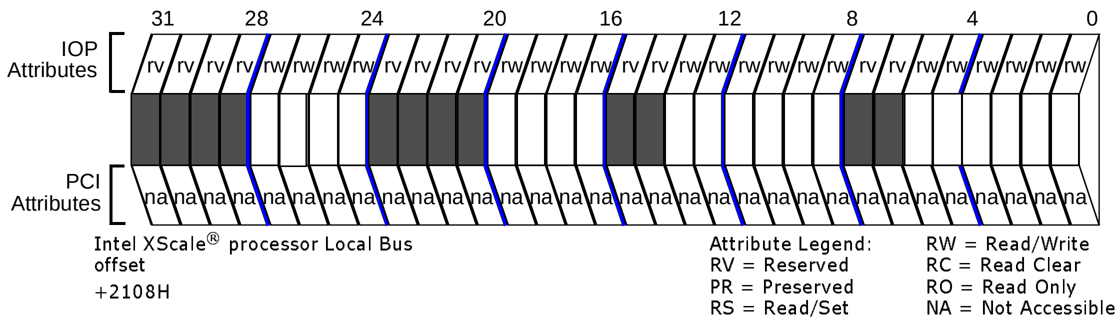


Bit	Default	Description
31:14	00000H	Reserved
13:08	100000_2	N-ODT drive strength manual override values for PCIX pad.
07:06	00_2	Reserved.
05:00	100000_2	P-ODT drive strength manual override values for PCIX pad.

2.14.90 PCIX PAD DRIVE STRENGTH Manual Override Values Register (3.3 V/1.5 V Switch Supply Voltage) — PPDSMOVR3.3_1.5

Warning: In Central Resource mode this register can only be accessed when PCSR bit 21 is cleared. Refer to Table 68, "PCI Configuration and Status Register - PCSR" on page 178.

Table 117. PCIX PAD DRIVE STRENGTH Manual Override Values Register (3.3V/1.5V Switch Supply Voltage) — PPDSMOVR3.3_1.5

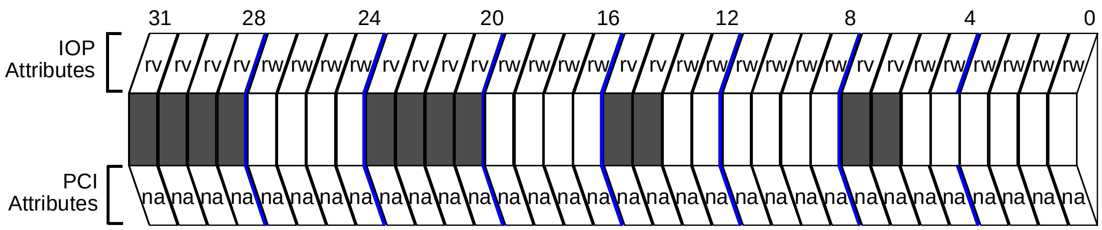


Bit	Default	Description
31:28	0H	Reserved
27:24	1000_2	N-slew rate manual override values for PCIX pad.
23:20	0000_2	Reserved.
19:16	1000_2	P-slew rate manual override values for PCIX pad.
15:14	00_2	Reserved.
13:08	100000_2	N-drive strength manual override values for PCIX pad.
07:06	00_2	Reserved.
05:00	100000_2	P-drive strength manual override values for PCIX pad.

2.14.91 PCIX PAD DRIVE STRENGTH Manual Override Values Register (3.3 V Dedicated Supply Voltage) — PPDSMOVR3.3

Warning: In Central Resource mode this register can only be accessed when PCSR bit 21 is cleared. Refer to Table 68, "PCI Configuration and Status Register - PCSR" on page 178.

Table 118. PCIX PAD DRIVE STRENGTH Manual Override Values Register (3.3 V Dedicated Supply Voltage) — PPDSMOVR3.3



Intel XScale® processor Local Bus offset
+210CH

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:28	0H	Reserved
27:24	1000_2	N-slew rate manual override values for PCIX pad.
23:20	0000_2	Reserved.
19:16	1000_2	P-slew rate manual override values for PCIX pad.
15:14	00_2	Reserved.
13:08	100000_2	N-drive strength manual override values for PCIX pad.
07:06	00_2	Reserved.
05:00	100000_2	P-drive strength manual override values for PCIX pad.

3.0 Address Translation Unit (PCI Express)

This chapter describes the operation modes, setup, and implementation of the module which interfaces between the PCI Express Link and the Intel® 413808 and 413812 I/O Controllers (4138xx) internal bus.

3.1 Overview

As indicated in Figure 22, the Address Translation Unit (ATU) — the interface between the PCI Express Link and the on-chip internal bus — consists of the Address Translation Unit (ATU) and the Expansion ROM Unit.

The ATU supports both inbound and outbound address translation. The ATU provides access between the PCI Express Link and the 4138xx internal bus.

Transactions initiated on the PCI Express Link and targeted at the 4138xx internal bus are referred to as *inbound transactions* (PCI Express to internal bus). Transactions initiated on the 4138xx internal bus and targeted at the PCI Express Link are referred to as *outbound transactions* (internal bus to PCI Express). The ATU accepts multiple inbound or outbound transactions and processes them simultaneously.

During inbound transactions, the ATU converts PCI addresses (initiated by a PCI Express Requester) to internal bus addresses and initiates the data transfer on the 4138xx internal bus. During outbound transactions, the ATU converts internal bus addresses to PCI addresses and initiates the data transfer on the PCI Express Link.

The Expansion ROM provides the PCI mechanism for downloading device/board driver code during system boot sequence. It consists of a separate inbound address range which accesses a Flash EPROM device connected through the 4138xx memory controller. Refer to the *PCI Local Bus Specification*, Revision 2.3 for details of Expansion ROM usage.

The Address Translation Unit and the Expansion ROM Translation Unit represent a single function of the multi-function 4138xx device.

The ATU supports the following PCI Express Lane widths and frequencies delivering up to 4096 Mbytes/sec of bandwidth:

— Lane Widths: x8, x4, x2, x1

— Link Frequency: 2.5Gbits/s

All PCI Express transactions are protected by link layer CRC.

On the internal interface, the ATU implements the 4138xx internal bus protocol which provides for a maximum of 4800 Mbytes/sec of bandwidth.

Address and data are protected by byte-wise parity on the internal bus.

The ATU includes four capability headers that implement Power Management capability as defined by the *PCI Bus Power Management Interface Specification*, Revision 1.1, MSI, MSI-X, and Vital Private Data (VPD) capabilities as defined by *PCI Local Bus Specification*, Revision 2.3, and PCI Express capability as defined by *PCI Express Base Specification*, Revision 1.0a.

Additionally, the ATU includes three PCI Express Extended capability headers that implement Advanced Error Handling, Device Serial Number, and Power Budgeting as defined in the *PCI Express Base Specification*, Revision 1.0a

The functionality of the ATU is described in the following sections. The ATU has a memory-mapped register interface that is visible from either the PCI interface, the internal bus interface, or both.

Figure 22. ATU Block Diagram

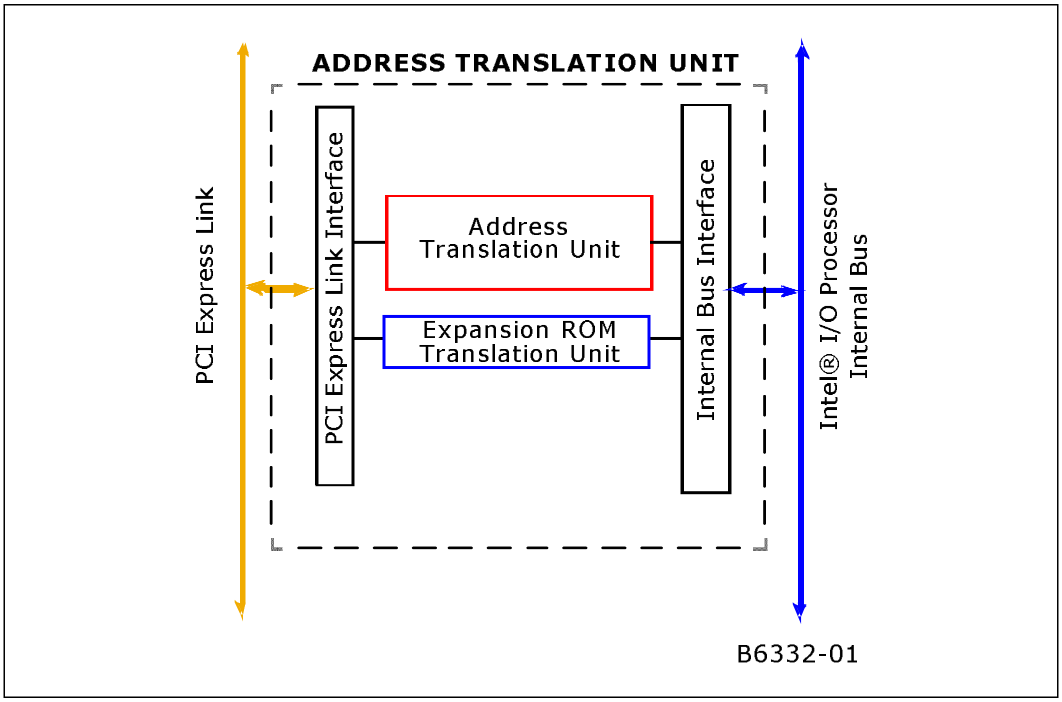
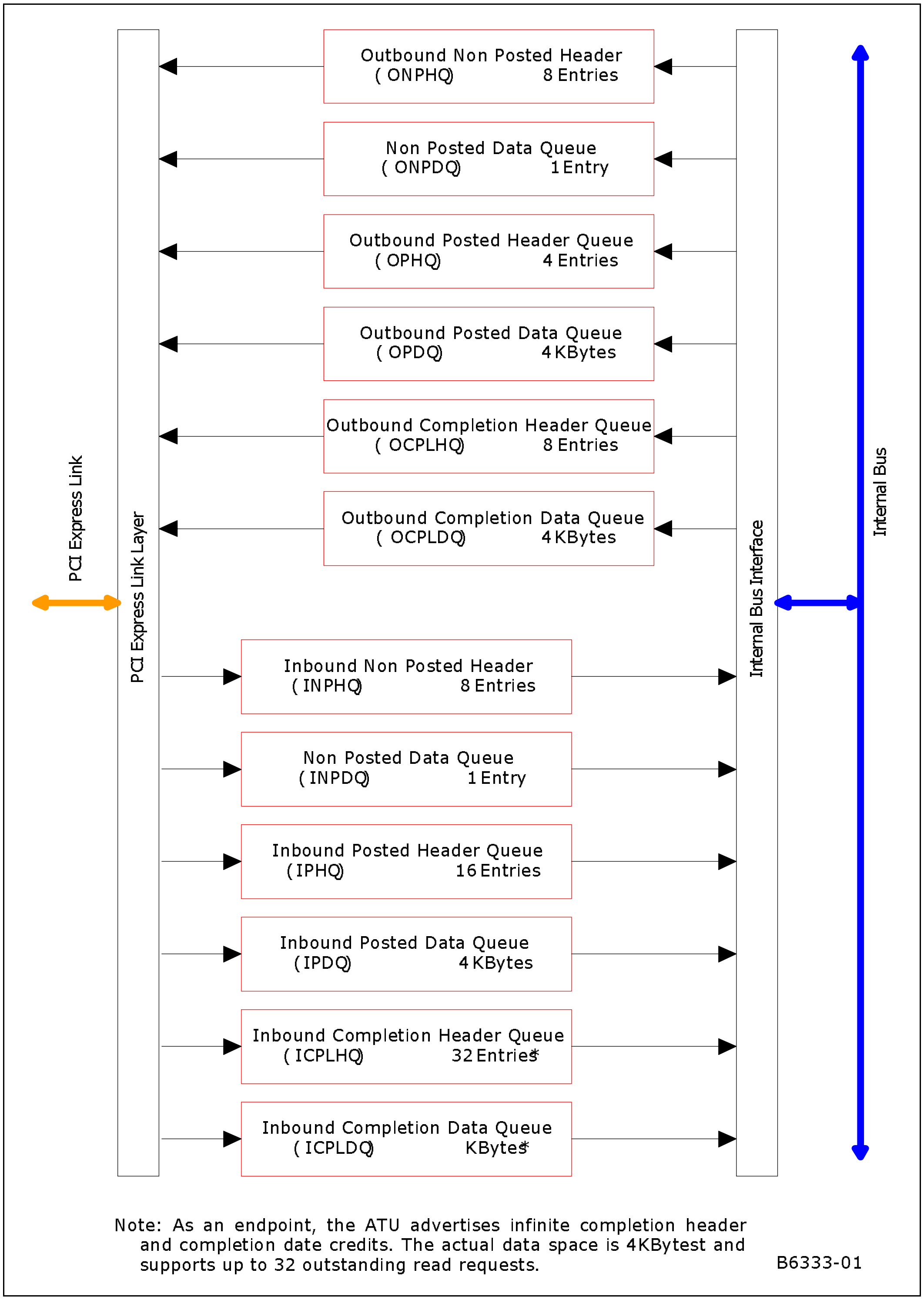


Figure 23. ATU Queue Architecture Block Diagram



3.2 PCI Express Link Characteristics

The PCI Express* port supports x8, x4, x2, and x1 operation. Lane reversal and polarity inversion automatically occur in an attempt to successfully train the link.

The PCI Express port is configured to ease adapter card implementations. The lane number and lane polarity should enable straight routing between the component and the PCI Express card edge connector. The lane reversal feature can be utilized to simplify applications where this component is connected to the upstream device in a planar fashion. For example integrating the device on a motherboard with direct connection to the I/O Hub.

Supported Lane Reversal Modes

x8	7:0, 0:7
x4	3:0, 0:3
x2	1:0, 0:1
x1	0, 7

Port bifurcation is not supported. This component supports a single x8 PCI Express port and can not be split into multiple x4 ports.

The PCI Express interface supports a maximum payload size of 512 Bytes and returns completions with minimum of 128 byte payloads.

3.3 ATU Address Translation

The ATU allows PCI Express requesters to initiate transactions to the 4138xx internal bus and allows the Intel XScale® processor to initiate transactions in the PCI Express domain.

The ATU implements an address windowing scheme to determine which addresses to claim and translate to the destination bus.

- The address windowing mechanism for inbound translation is described in Section 3.3.1.1, "Inbound Address Translation" on page 237.
- The address windowing mechanism for outbound translation is described in Section 3.3.2, "Outbound Transactions" on page 244 and Section 3.3.3, "Outbound Write Transaction" on page 251.

The ATU has the ability to accept up to eight inbound PCI Express Non Posted Read (memory read, configuration read, and I/O read) transactions, one inbound Non Posted Write (configuration write and I/O write) transaction, and16 inbound PCI Express Posted (memory write and message) transactions simultaneously.

As a PCI Express end point, the ATU must advertise infinite credits for Completion Headers and Completion Data. As a requester the ATU never requests more read data than it has room for in its Inbound Completion queues (ICPLHQ and ICPLDQ).

Also, the ATU has the ability to accept up to eight outbound Non Posted (internal bus read, configuration, and I/O) transactions and four outbound Posted (internal bus write and message) transactions simultaneously.

Of the 8 outbound Non Posted transactions, 4 is actively requesting data while the other 4 remains pending until one of the earlier active transactions is completed. Each active outbound read request may be fragmented into sub-requests based on the MAX_READ_REQUEST_SIZE parameter programmed in the "PCI Express Device Control Register - PE_DCTL" on page 344. The fragmentation may result in as many as 8 sub-requests per active read transaction. The ATU tracks a maximum of 32 outstanding non posted transactions at once. Completions for these sub-requests can return out of order on the PCI Express interface but they are returned in order on the internal bus.

Outbound memory writes and completions may be fragmented into smaller transactions based on the setting of the MAX_PAYLOAD_SIZE in the "PCI Express Device Control Register - PE_DCTL" on page 344.

Outbound completions obeys the minimum fragmentation limit of 128Bytes.

Refer to Figure 23 and Section 3.8 for details of the ATU queue architecture.

As a master on the internal bus, the ATU never requests more read data than it has room for in the Outbound Completion queues (OCPLHQ and OCPLDQ).

Inbound memory writes are fragmented on 1KB address aligned boundaries before issuing on the internal bus. Since the maximum payload size supported by the PCI Express interface is 512 Bytes each transactions are fragmented into a maximum of two internal bus transactions. Additionally, write combining does not occur.

Inbound completions are attempted on the internal bus with the same payload size as was received from PCI Express. In most instances the PCI Express completion size is 64B or 128B. The PCI express completions may be combined into larger completion transactions on the internal bus.

Inbound memory read requests are fragmented into 1KB aligned sub-requests. Completions for these sub-requests can be received out of order on the internal bus and is returned in order on the PCI Express interface.

The ATU unit allows for recognition and generation of multiple PCI Express Transaction Layer Packets (TLP) types. Table 120 shows the commands supported for both inbound and outbound ATU transactions. The type of operation seen by the ATU on inbound transactions are determined by the PCI Express requester who initiates the transaction. Claiming an inbound transaction depends on the address range programmed within the inbound translation window. The type of transaction used by the ATU on outbound transactions generated by the core processor is determined by the internal bus address and the fixed outbound windowing scheme.

ATU supports all four address spaces defined within the PCI Express architecture as both a requester and completer. These address spaces and corresponding transaction types are detailed in Table 119.

Table 119. Supported Address Spaces and Transaction Types

Address Space	Transaction Type	Basic Usage
Memory[a]	Read Write	Transfer data to/from a memory-mapped location
I/O	Read Write	Transfer data to/from and I/O-mapped location
Configuration	Read Write	Device configurations/setup
Message	Baseline Vendor-defined	From event signalling mechanism to general purpose messaging

a. ATU supports both 32-bit and 64-bit addressing for Memory space transactions.

ATU does not support Locked Requests as a Completer nor generate them as a Requestor.

Table 120. ATU Command Support

TLP Type	Fmt[1:0][a]	Type[4:0]	Supported as Completer	Generated as Requester	Valid Internal Bus Command
MRd	00 01	0 0000	Yes	Yes	Read
MRdLk	00 01	0 0001	Unsupported Request	No	N/A
MWr	10 11	0 0000	Yes	**Yes**	Write
IORd	00	0 0010	Yes	**Yes**	Read
IOWr	10	0 0010	Yes	Yes	Write
CfgRd0	00	0 0100	Yes	Yes	Read
CfgWr0	10	0 0100	**Yes**	**Yes**	Write
CfgRd1	00	0 0101	**Unsupported Request**	**Yes**	N/A
CfgWr1	10	0 0101	**Unsupported Request**	Yes	N/A
Msg	01	1 0$r_2r_1r_0$	**Yes**	Yes	Write[b]
MsgD	11	1 0$r_2r_1r_0$	**Yes**	**Yes**	Write[b]
Cpl	00	0 1010	**Yes**	**Yes**	Completion
CplD	10	0 1010	**Yes**	**Yes**	Completion
CplLk	00	0 1011	Unexpected Completion[c]	No	N/A
CplDLk	10	0 1011	Unexpected Completion[c]	No	N/A
	All encodings not shown above are reserved		Malformed Packet	No	N/A

a. Requests with two Fmt[1:0] values shown can use either 32b (the first value) or 64b (the second value) Addressing Packet formats.
b. As a completer, the ATU stores the message header and payload directly in memory mapped registers. No internal bus traffic occurs.
c. ATU does not generate lock requests and any CplLk or CplDLk would be an unexpected completion.

Inbound and outbound ATU transactions are best described by the data flows used on the PCI Express Link and the 4138xx internal bus during read and write operations. The following sections describe read and write operations for inbound ATU transactions (PCI Express to internal bus) and outbound transactions (internal bus to PCI Express). All transactions are full split and the requests and completions are described separately.

3.3.1 Inbound Transactions

Inbound transactions are received on the PCI Express receive port and forward to the 4138xx internal bus. This transactions include all requests for which the ATU is the completer as well as completions for which the ATU was the initiator.

Inbound request transactions which target the ATU are translated and executed on the 4138xx internal bus. As a PCI Express completer, the ATU is capable of accepting all memory, I/O, and configuration request. Additionally, as a PCI Express end-point, the ATU would never request more memory read data than it can hold in its Inbound Completion Data Queue.

Inbound memory write (and message) transactions have their headers entered into the inbound posted header queue (IPHQ) and data entered into the inbound posted data queue (IPDQ). The IPHQ/IPDQ pair are capable of holding up to 16 posted operations up to the size of the data queue. Inbound configuration (or I/O) write transactions use the inbound non posted header queue (INPHQ) and inbound non posted data queue (INPDQ). The INPDQ has room for one configuration or I/O write at a time. Refer to Section 3.8 for details of queue operation. Inbound read transaction (memory, configuration, and I/O) have their header entered into the inbound non posted header queue (INPHQ) and the data are returned to the PCI Express requester in the outbound completion data queue (OCPLDQ). The INPQ is capable of holding up to 8 non posted requests and any associated data.

Operation of the internal bus is defined in Section 7.0, "System Controller (SC) and Internal Bus Bridge".PCI Express has three principal mechanisms for Transaction Layer Packet (TLP) routing: address, ID, and implicit. The following sections describes how the ATU routes and translates each type.

3.3.1.1 Inbound Address Translation

PCI Express utilizes both 32-bit and 64-bit address schemes via the 3DW and 4DW headers. To prevent address aliasing, all devices must decode the entire address range. All discussions in this section refer to 64-bit addressing. When the 3DW header is used the upper 32-bits of address are assumed to be 0000_0000h.

The ATU allows external PCI Express requesters to directly access the internal bus via address routed TLPs. These PCI Express requesters can read or write 4138xx memory-mapped registers or 4138xx local memory space. The process of inbound address translation involves two steps:

1. Address Detection.
   — Verify the PCI address is within the address windows defined for the inbound ATU.
   — When the address is outside of the ATU address registers, the transaction is terminated as an unsupported request (UR).
2. Address Translation.
   — Translate the lower 32-bit PCI address to a 36-bit 4138xx internal bus address.

The ATU uses the following registers in inbound address window 0 translation:

- Section 3.17.13, "Inbound ATU Base Address Register 0 - IABAR0" on page 304
- Section 3.17.28, "Inbound ATU Limit Register 0 - IALR0" on page 318
- Section 3.17.29, "Inbound ATU Translate Value Register 0 - IATVR0" on page 319
- Section 3.17.30, "Inbound ATU Upper Translate Value Register 0 - IAUTVR0" on page 319

The ATU uses the following registers in inbound address window 1 translation:

- Section 3.17.16, "Inbound ATU Base Address Register 1 - IABAR1" on page 308
- Section 3.17.31, "Inbound ATU Limit Register 1 - IALR1" on page 320
- Section 3.17.32, "Inbound ATU Translate Value Register 1 - IATVR1" on page 321
- Section 3.17.33, "Inbound ATU Upper Translate Value Register 1 - IAUTVR1" on page 321

The ATU uses the following registers in inbound address window 2 translation:

- Section 3.17.18, "Inbound ATU Base Address Register 2 - IABAR2" on page 310
- Section 3.17.34, "Inbound ATU Limit Register 2 - IALR2" on page 322
- Section 3.17.35, "Inbound ATU Translate Value Register 2 - IATVR2" on page 323
- Section 3.17.36, "Inbound ATU Upper Translate Value Register 2 - IAUTVR2" on page 324

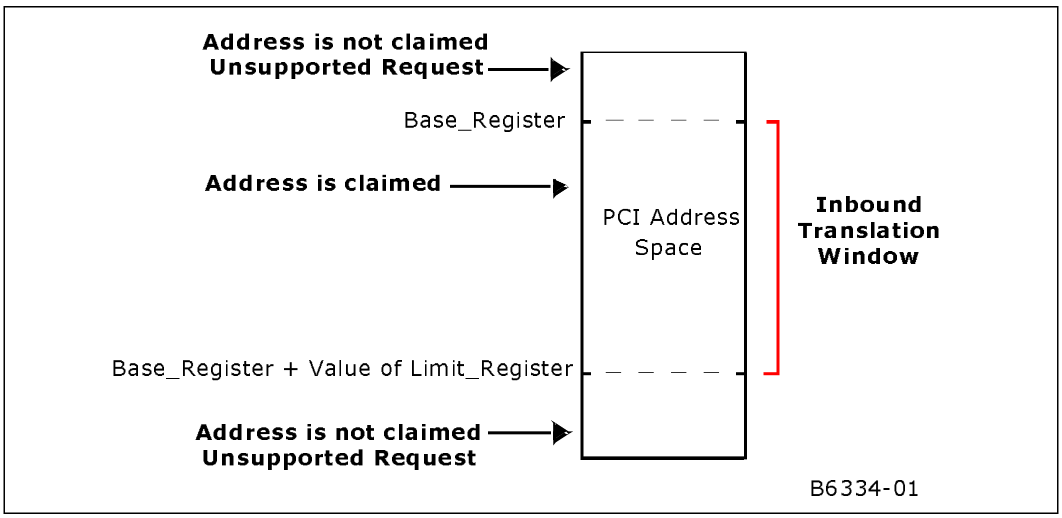
Inbound address detection is determined by comparing the 64-bit PCI address with the base address register and the limit register. In the case of 3DW headers, the upper 32-bits of the address is assumed to be 0000_0000h during address comparison. The algorithm for detection is:

Equation 8. Inbound Address Detection

```
When PCI_Address [31:0] & Limit_Register[31:0] == Base_Register[31:0] and
PCI_Address [63:32] == Base_Register[63:32] the PCI Address is translated by the Inbound ATU.
Otherwise treat as an Unsupported Request.
```

Figure 24 shows an example of inbound address detection.

Figure 24. Inbound Address Detection



The lower 32-bits of the incoming address is bitwise ANDed with the associated inbound limit register. When the result matches the base register, the inbound PCI address is detected as being within the inbound translation window and is claimed by the ATU. When the address is outside the translation window, the ATU terminates the transaction as an Unsupported Request (UR).

Note: By default, the first 8Kbytes of the ATU inbound address translation window 0 are reserved for the Messaging Unit. See Section 3.5, "Messaging Unit" on page 257.

Once the transaction is claimed, the upper 32-bits of the address is discarded and the lower 32-bits of the address must be translated from a PCI address to a 36-bit internal bus address. The algorithm is:

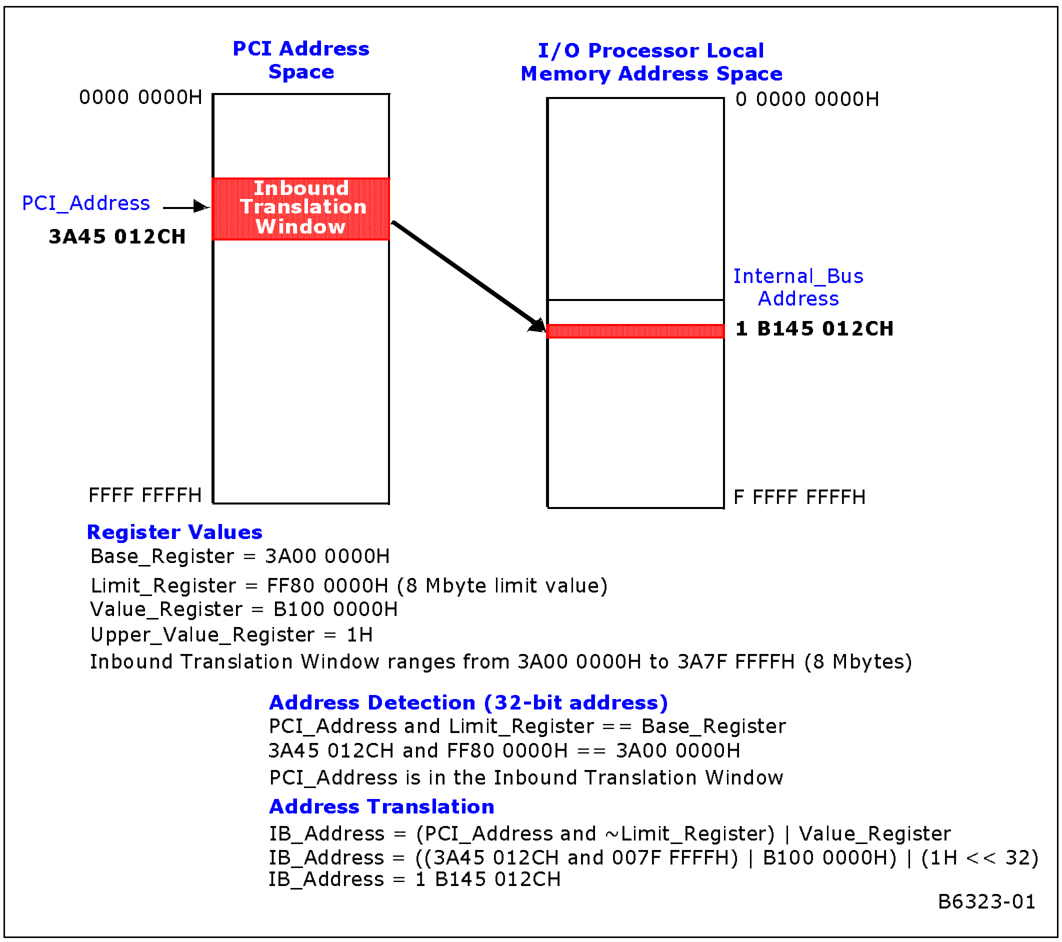
Equation 9. Inbound Translation

```
4138xx Internal Bus Address = ((PCI_Address[31:0] & ~Limit_Register[31:0]) |
ATU_Translate_Value_Register[31:0]) | (ATU_Upper_Translate Value_Register[3:0] << 32).
```

The lower 32-bits of the incoming PCI address are first bitwise ANDed with the bitwise inverse of the limit register. This result is bitwise ORed with the ATU Translate Value, which is then ORed with the 4-bit ATU Upper Translate Value left shifted by 32; the result is the 36-bit internal bus address. This translation mechanism is used for all inbound memory read and write commands excluding inbound configuration read and writes. Inbound configuration cycle translation is described in Section 3.3.1.5, "Inbound Configuration Cycle Translation (ID Routed)" on page 242.

Figure 25 shows an inbound translation example for 32-bit addressing. This example would hold true for an inbound transaction from PCI Express Link.

Figure 25. Inbound Translation Example



3.3.1.2 Inbound Memory Write Transaction

An inbound write transaction is initiated by a PCI Express requester and is targeted at either 4138xx local memory or a 4138xx memory-mapped register.

Data flow for an inbound write transaction is summarized as:

- The ATU accepts the write transaction when the PCI address is within one of the inbound translation windows defined by the ATU Inbound Base Address Register, Inbound Upper Base Address Register, and Inbound Limit Register.
- When the IPHQ is full or the IPDQ overflows, a flow control error occurred and an ERR_FATAL is returned to the root complex.

Once the inbound write packet has passed all the TLP validation checks, the ATUs internal bus interface becomes aware of the inbound write. When there are additional write transactions ahead in the IPHQ, the current transaction remains posted until ordering and priority have been satisfied (Refer to Section 3.8.3) and the transaction is attempted on the internal bus by the ATU internal master interface.

Data flow for the inbound write transaction on the internal bus is summarized as:

- The ATU internal bus master requests the internal bus when IPHQ has at least one entry.
- When the internal bus is granted, the internal bus master interface initiates the write transaction by driving the translated address onto the internal bus. For details on inbound address translation, see Section 3.3, "ATU Address Translation" on page 234.
- When an internal bus target does not claim write transaction, a master abort condition is signaled on the internal bus. The current transaction is flushed from the queue and an unsupported request (UR) message may be generated on the PCI Express interface.
- The ATU initiator interface attempts a 128-bit wide transfer on the internal bus. When the target that claims the request does not support 128-bit wide transfers, a 64-bit wide transfer is used. Transfers use internal bus byte enables to mask the bytes not written in each data phase. Write data is transferred from the IPDQ to the internal bus while data is available and the internal bus interface retains internal bus ownership. Refer to Section 7.0, "System Controller (SC) and Internal Bus Bridge" for details of internal bus operation.
- The internal bus interface stops transferring data from the current transaction to the internal bus when one of the following conditions becomes true:
  - The internal bus initiator interface loses bus ownership.
  - The data from the current transaction has completed (satisfaction of payload length). An initiator termination is performed and the bus returns to idle.
  - A Master Abort is signaled on the internal bus. Data is flushed from the IPDQ.

3.3.1.3 Inbound Memory Read Transaction

An inbound read transaction is initiated by a PCI Express requester and is targeted at either 4138xx local memory or a 4138xx memory-mapped register space. The read transaction is propagated through the inbound non posted queue (INPQ) and read data is returned through outbound completion data and header queues (OCPLHQ, OCPLDQ).

In PCI Express, all read transactions are processed as split transactions. The ATUs PCI Express interface accepts the read transaction and forwards the read request through to the internal bus and returns the read data to the PCI Express Link. Data flow for an inbound read transaction is summarized in the following statements:

- The ATU accepts the read transaction when the PCI address is within one of the inbound translation windows defined by ATU Inbound Base Address Register, Inbound Upper Base Address Register, and Inbound Limit Register.
- When the transaction crosses a 1KB aligned boundary it is fragmented into smaller requests that do not cross the aligned boundary before it is issued on the internal bus. Since PCI Express transactions cannot cross a 4KB boundary, a single read request is broken into at most 4 1KB transactions.
- When sufficient space exists in OCPLDQ, request internal bus and issue request.
- Save the completion header information in the OCLUT.
- ATUE can handle a maximum of 4 outstanding internal bus requests at one time.
- All internal bus read requests result in split completions. The completion data is queued in the Outbound Completion Data Queue.
- A zero length read (memory read request of 1 DW with no bytes enabled) has no side-effects.
- Once a completion transaction has started, it continues until one of the following is true:
  - The length is satisfied.
  - An internal bus Master Abort or Target Abort was detected. The ATU generates a Completion TLP with a Completer Abort status to inform the requester about the abnormal condition. The INPHQ for this transaction is flushed. Refer to Section 3.9.3.

The data flow for an inbound read transaction on the internal bus is summarized in the following statements:

- The ATU internal bus master interface requests the internal bus when a PCI address appears in an INPHQ and transaction ordering has been satisfied. The ATU takes advantage of the information provided by the Relaxed Ordering Attribute bit.
- Once the internal bus is granted, the internal bus master interface drives the translated address onto the bus. When a Retry is signaled, the request is repeated. When a master abort occurs, the transaction is considered complete and an unsupported request Completion is loaded into OCPLHQ for return to the PCI Express requester (request is flushed once the completion has been posted to the OCPLHQ).
- Once the translated address is on the bus and the transaction has been claimed, the internal bus target starts returning data using a split response. Read data is continuously received by the OCPLDQ until one of the following is true:
  - The full byte count requested by the ATU read request is received. The internal bus completer's initiator interface performs an initiator completion in this case.
  - A partial byte count requested by the ATU read request is received. The completer's internal bus initiator interface performs an initiator completion in this case. Also, the completer reacquires the internal bus to deliver the remaining read data byte count to the ATU.

3.3.1.4 Inbound I/O Cycle Translation

Inbound address window 2 can be configured to accept I/O Read and I/O Write transactions by setting the Memory/IO space indicator bit to 1 in the "Inbound ATU Base Address Register 2 - IABAR2" on page 310. All I/O cycles are 32-bit transactions (DWORD).

For inbound I/O reads, the INPQ is used in the same manner as inbound memory read operations and the exception cases are identical. However, the internal bus cycle that results is always be a 32-bit transaction.

For inbound I/O writes, the ATU uses the INPHQ to hold both the header and the INPDQ to hold the data. An I/O write TLP with poisoned data results in the data being dropped and a Completion with Completions status of UR is returned to the requester.

When there are no errors during the request cycle, transaction ordering and priority are satisfied. Next, the internal bus master interface requests the internal bus and deliver the write data to the target as defined in Section 3.3.1.2.

The status of the transaction on the internal bus is returned to the requestor on the PCI Express Link. When the Write Cycle Master Aborts on the Internal Bus, a Completion with status of Completer Abort is returned.

3.3.1.5 Inbound Configuration Cycle Translation (ID Routed)

The 4138xx ATU only accepts Type 0 configuration requests with a function number of zero when bit 7 of the ATUHTR (see Section 3.17.11, "ATU Header Type Register - ATUHTR" on page 302) is cleared or function numbers of zero and one when bit 7 of the ATUHTR is set.

The ATU is a native PCI Express device that supports both the standard PCI express capability structure and the PCI Express Extended capability structure. The ATU configuration space is selected by any PCI Express Type 0 configuration cycle targeting function 0. The bus number and device number is captured from all valid Type 0 configuration write TLPs that target an enabled function.

For inbound configuration reads, the INPQ is used in the same manner as inbound memory read operations.

For inbound configuration writes, the ATU uses the INPQ to hold both the header and the data. An configuration write TLP with poisoned data results in the data being dropped and an Completion with Completions status of UR is returned to the requester.

When there are no errors during the request cycle, transaction ordering and priority are satisfied. Next, the internal bus master interface requests the internal bus and deliver the write data to the target as defined in Section 3.3.1.2.

Since Master Aborts and Target Aborts cannot occur during configuration cycles on the internal bus, a Completion TLP with Successful Completion (SC) status is generated for all configuration cycles.

When the Configuration Request Retry bit is set (PCSR[2)], the Configuration Request cycle is terminated with a Configuration Request Retry Status (CRS).

3.3.1.6 Inbound Vendor_Defined Message Transactions

Inbound messages are routed to the PCI Express message unit where they are decoded and processed.

Inbound Vendor_Defined Messages (IVM) are logged in the Inbound Message Header0-3 and Inbound Message Payload registers and an interrupt is conditionally sent to the Intel XScale® processor.

Only one message can be pending in the Inbound Vendor Message registers at one time. When bit 6 of the "ATU Configuration Register - ATUCR" on page 326 is set, then subsequent IVM are dropped. This is necessary to prevent deadlock when the Intel XScale® processor has outstanding read transactions.

When bit 6 is cleared, then when a second vendor specific message transaction reaches the head of the IPHQ it stalls until the message registers are freed by clearing the Message Received bit in the ATUISR. Since messages are posted transactions, they stall all other transactions until they make progress.

When the message received interrupt mask is set in the ATUMR, then the inbound message transactions are still logged to the Inbound message register but they do not block following vendor specific message transactions.

Table 121. Inbound Vendor_Defined Message Type 0 Response.

Response for Type 0 IVM	IVM Received Interrupt Mask (ATUIMR - bit 25)	Drop subsequent IVM (ATUCR - bit 6)
Unsupported Request (UR)	1	0
UR	1	1
Return UR when firmware requests a UR response. (PEMCSR -bit 14).	0	0
Dropped Silently when interrupt pending (ATUISR - bit 25).	0	1

Note: A Type0 vendor_defined message may be discarded without a UR response when the interrupt mask is cleared and the ATU is configured to drop subsequent IVM messages when the interrupt is pending.

3.3.2 Outbound Transactions

Outbound transactions initiated by the 4138xx core processor are directed to the PCI Express interface through the ATU. As a PCI Express requester, the ATU is capable of memory, I/O, configuration, and message transactions. Outbound memory transactions with addresses below 4GB use the short address format (32-bit address). Addresses above 4GB use the long address format (64-bit).

Outbound transactions use a separate set of queues from inbound transactions. Outbound write operations have their address entered into the outbound posted header queue (OPHQ) and their data into the outbound posted data queue (OPDQ). Outbound read transactions, use the Outbound Non-Posted Queue (ONPQ) to store address, and get data returned into the Inbound Completion Data Queue (ICPLDQ). Refer to Section 3.8.2 for details of outbound queue architecture. Outbound configuration transactions use a special outbound port structure and are enqueue in the ONPQ. Refer to Section 3.3.3 for details.

For outbound write transactions, the ATU is a target on the internal bus and a requester on the PCI Express Link. For outbound read transactions, the ATU is a completer on the internal bus (initially accepts the split read as a target and then provides read data by initiating a split completion). Internal bus operation is defined in Section 7.0, "System Controller (SC) and Internal Bus Bridge".

Note: For all outbound writes, the byte enables must be contiguous. This means that write coalescing must be disabled in the Intel XScale® microarchitecture for transactions that target the outbound memory windows.

While Outbound I/O transactions are supported in all configurations, they should only be used when operating as the root complex. The PCI Express Specification states that PCI Express Endpoints must not generate I/O requests.

3.3.2.1 Outbound Address Translation - Internal Bus Transactions

In addition to providing the mechanism for inbound translation, the ATU translates Intel XScale® processor-initiated cycles to the PCI Express domain. This is known as *outbound address translation*. Outbound transactions are processor or DMA transactions targeted at the PCI Express Link. The ATU internal bus target interface claims internal bus cycles and completes the cycle on the PCI Express Link on behalf of the Intel XScale® processor or DMAs.

Figure 26 shows 4 Gbyte memory section 0 (Internal Bus Address [35:32] = 0000_2) of the 4138xx memory map with all reserved address locations highlighted. The 64KByte outbound I/O window is from 0.FFFD.0000H to 0.FFFD.FFFFH while the PMMR registers reside from 0.FFD8.0000H to 0.FFDF.FFFFH.

By default, Outbound Memory Window 0, Outbound Memory Window 1, Outbound Memory Window 2, and Outbound Memory Window 3reside in 4 Gbyte memory sections 1, 2, 3, and 4respectively, of the 64 Gbyte Internal Bus address space.

The response of the ATU to Outbound Transactions is globally controlled by the Outbound Enable bit in the ATU Configuration Register as well as the Bus Master Enable bit in each function. When the Outbound Enable bit is deasserted, outbound transaction master-abort on the internal bus and are not forwarded to the PCI Express Domain. When the Outbound Enable bit is asserted, the relevant Bus Master Enable bit for each function is used to determine the appropriate response to an outbound transaction.

The Outbound ATUs behavior for the different combinations of these control bits is described in Table 122.

Table 122. Outbound Address Translation Control

Outbound Response	Outbound Enable[a] (ATUCR[1])	Bus Master Enable[b]
Master-Abort	0	0
Master-Abort	0	1
Retry	1	0
Claim[c]	1	1

a. In addition, the outbound memory windows need to be individually enabled in order to claim the transaction. When the memory widow is disabled, it does not claim a transaction which might result in a Master-Abort. By default, Outbound Memory Windows 0 and 1 are enabled while Outbound Memory Windows 2, and 3 are disabled.
b. In a multi-function configuration, each function independently controls its own Bus Master Enable bit.
c. The ATU may respond with a Retry in this case when the Outbound Transaction Queues are full.

3.3.2.2 Outbound Address Translation Windows

Inbound translation involves a programmable inbound translation window consisting of a base and limit register and a value register for PCI to internal bus translation. The outbound address translation windows use a similar methodology except that the outbound translation window limit sizes are fixed in 4138xx internal bus address space; this removes the need for separate limit registers.

Figure 27 on page 249 illustrates the five outbound address translation windows. The ATU has four 4 Gbyte outbound memory translation windows and one 64 Kbyte outbound I/O translation window. By default, Outbound Memory Window 0 (OUMBAR0), Outbound Memory Window 1 (OUMBAR1), Outbound Memory Window 2 (OUMBAR2), and the Outbound Memory Window 3 (OUMBAR3) reside in 4 Gbyte memory sections 1, 2, 3, and 4, respectively. The default location of the 64 KByte outbound I/O window range is from 0.FFFD.0000H to 0.FFFD.FFFFH. The following registers are used to specify the five 4 Gbyte windows for claiming Outbound Memory transactions:

- Outbound Upper Memory Base Address Register 0 (OUMBAR0)
  - Default Value equal to 01H.
- Outbound Upper Memory Base Address Register 1 (OUMBAR1)
  - Default Value equal to 02H.
- Outbound Upper Memory Base Address Register 2 (OUMBAR2)
  - Default Value equal to 03H.
- Outbound Upper Memory Base Address Register 3 (OUMBAR3)
  - Default Value equal to 04H.
- Outbound I/O Base Address Register (OIOBAR)
  - Default Value equal to 0FFF D000H

An internal bus cycle with an address within one of the outbound windows initiates a read or write request on the PCI Express Link. The PCI transaction type depends on which translation window the local bus cycle "hits". The read or write decision is based on the internal bus cycle type.

ATU has windows dedicated to the following outbound transaction types:

- Memory reads and Memory writes - Memory Window
- I/O reads and writes - I/O Window

Table 123. Internal Bus-to-PCI Command Translation for Memory Windows

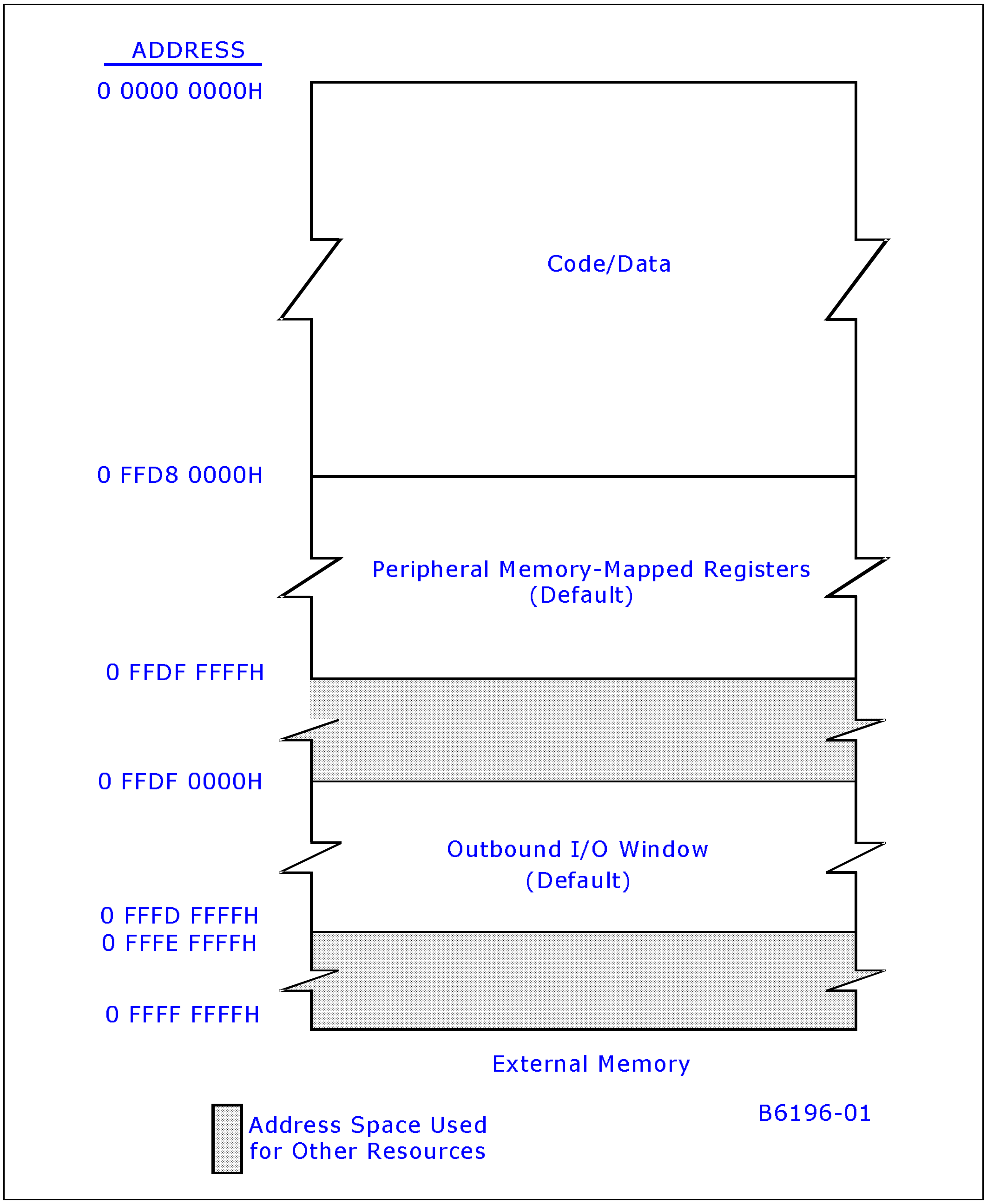
Internal Bus Command	PCI Express Transaction Type
Write	Memory Write Request
Read	Memory Read Request

Table 124. Internal Bus-to-PCI Command Translation for I/O Window

Internal Bus Command[a]	Conventional PCI Command
Write	I/O Write Request
Read	I/O Read Request

a. User should designate memory region containing I/O Window as non-cachable and non-bufferable from Intel XScale® processor. This insures all load/stores to I/O Window are of DWORD quantities. In the event that the user inadvertently issues a read to the I/O Window which crosses a DWORD address boundary, the ATU target aborts the transaction. Only bytes 3:0 is relevant dependent on the Byte Enables.

Figure 26. 4 Gbyte Section 0 of the Internal Bus Memory Map



The translation portion of outbound ATU transactions is accomplished with a value register in the same manner as inbound translations. Each outbound memory window is associated with one translation register which provides the upper translation addresses (OUMWVR0-3). When the corresponding OUMWVRx register is all-zero a 3DW header transaction is generated on the PCI Express Link. Otherwise, a 4DW header is generated on the PCI Express Link using the value in the OUMWVRx register for the upper 32-bit address. ATU uses the following registers during outbound address translation:

- Outbound Upper 32-bit Memory Window Value Register 0 (OUMWVR0)
- Outbound Upper 32-bit Memory Window Value Register 1 (OUMWVR1)
- Outbound Upper 32-bit Memory Window Value Register 2(OUMWVR2)
- Outbound Upper 32-bit Memory Window Value Register 3(OUMWVR3)
- Outbound I/O Window Value Register (OIOWVR)
- Outbound Configuration Cycle Address Register (OCCAR)

See Section 3.17 for details on outbound translation register definition and programming constraints.

The translation algorithm used, as stated, is very similar to inbound translation. For memory transactions, the algorithm is:

Equation 10. Outbound Address Translation

```
PCI Address = (Internal_Bus_Address & 0.FFFF.FFFFH) | (Upper_Window_Value_Register << 32)
```

For memory transactions, the internal bus address is bitwise ANDed with the inverse of 4 Gbytes which clears the upper 4 bits of the 36 bit address. The result is bitwise ORed with the outbound upper window value register left shifted by 32 to create the Upper 32-bits of the PCI address. When the Upper 32-bits of the PCI Address equals 0000 0000H, the ATU generates a transaction with a 3DW header on the PCI Express Link, otherwise, a 4DW header is used.

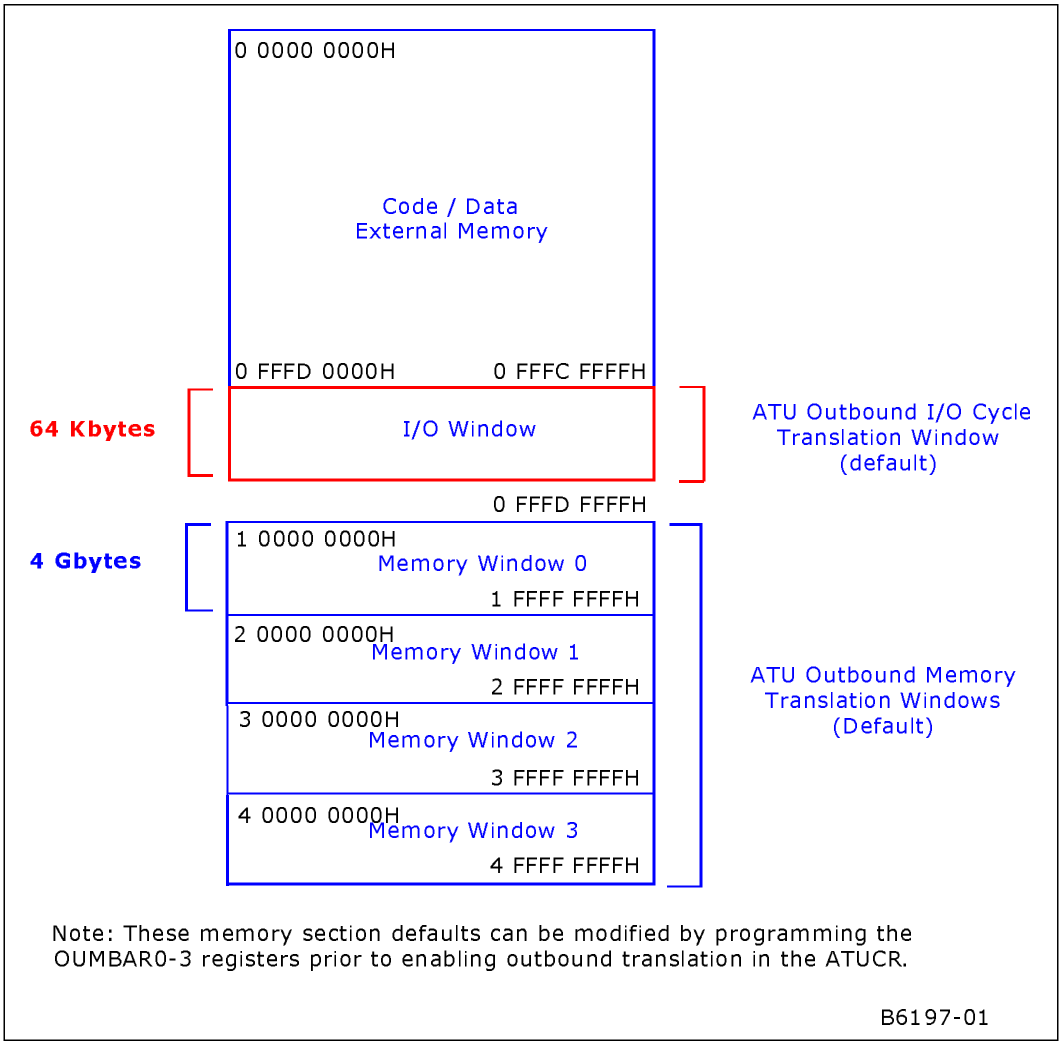
For I/O transactions, the algorithm is:

Equation 11. I/O Transactions

```
PCI Address = (Internal_Bus_Address & 0.0000.FFFFH) | Window_Value_Register
```

For I/O transactions, the internal bus address is bitwise ANDed with the inverse of 64 Kbytes which clears the upper 20 bits of address. Address aliasing is prevented by the outbound window value registers which only allow values on boundaries equivalent to the window's length.

Figure 27. Outbound Address Translation Windows



3.3.2.3 Outbound DMA Transactions

The ATU provides all ADMA channels with a transparent path through the PCI Express interface. The entire 64-bit Host I/O Interface address programmed in the DMA descriptor is passed to the PCI Express link unmodified.

3.3.2.4 Outbound Function Number

The 4138xx is a multi-function device and the ATU associates each transaction with the correct PCI Express function number according to the following algorithm:

- Outbound Configuration transactions use the function number field specified in the "Outbound Configuration Cycle Function Number - OCCFN" on page 383.
- Outbound I/O transactions use the function number field specified in the "Outbound I/O Base Address Register - OIOBAR" on page 371.
- Outbound transactions targeting the outbound memory windows utilize the "Outbound Window x Function Number Mapping" programmed into bits 30:28 of the associated OUMBARx registers
- Outbound ADMA transactions use the "Host I/O Interface Function Number" programmed in the ADMA byte count register.

3.3.3 Outbound Write Transaction

An outbound write transaction is initiated by the Intel XScale® processor[9] or by one of the DMAs and is targeted at a PCI Express domain. The outbound write address and write data are propagated from the 4138xx internal bus to a PCI Express Link through the OPHQ and OPDQ, respectively.

The ATUs internal bus target interface claims the write transaction and forwards it to the PCI Express Link. The data flow for an outbound write transaction on the internal bus is summarized in the following statements:

- ATU internal bus target interface latches the address from the internal bus into the OPHQ when that address is inside one of the outbound translate windows (see Section 3.8) and the OPHQ is not full.
- When the OPHQ is full, the target interface signals a Retry on the internal bus to the outbound cycle initiator.
- Once outbound address is latched, internal bus target interface stores write data into the OPDQ until the internal bus transaction completes or the reaches a buffer boundary.
- When the data is latched in a buffer in OPDQ, the outbound cycle is enabled for transmission on the PCI Express Link.

The PCI interface is responsible for completing the outbound write transaction with the PCI address translated from the OPHQ and the data in the OPDQ. The data flow for an outbound write transaction on the PCI Express Link is summarized in the following statements:

- Writes transactions is fragmented based on the Max_Payload_Size parameter. A write issues when the Max_Payload_Size is reached, or the write disconnects on the internal bus.
- When Posted Header and Posted Data credits are available a memory write request TLP is issued on the PCI Express Link.
- When a data parity error is detected while pulling data from the OPDQ, the TLP is poisoned.

9. For best performance, the user should designate the two Outbound Memory Windows as non-cachable and bufferable from the Intel XScale® processore. This assignment enables the Intel XScale® processor to issue multiple outstanding transactions to the Outbound Memory Windows, thereby, taking full advantage of the ATU outbound queue architecture. However, the user needs to be aware that the Outbound ATU queue architecture does not maintain strict ordering between read and write requests as described in Table 130, “ATU Outbound Data Flow Ordering Rules” on page 265. In the event that the user requires strict ordering to be maintained, the user must change the designation of this region of memory to be non-cachable/non-bufferable and enforce the requirement in software.

3.3.4 Outbound Read Transaction

An outbound read transaction is initiated by the Intel XScale® processor[10] or one of the DMAs and is targeted at a PCI slave on the PCI Express Link. The read transaction is propagated through the outbound non posted queue (ONPQ) and read data is returned through the inbound completion data queue (ICPLDQ).

The ATUs internal bus target interface claims the Memory Read transaction and forwards the read request through to the PCI Express Link and returns the read data to the internal bus.

The data flow for an outbound read transaction on the internal bus is summarized in the following statements:

- The ATU internal bus interface latches the internal bus address when the address is inside an outbound address translation window (or the direct addressing window, when enabled) and the ONPQ is not full. All read transactions are handled as split transactions. When the ONPQ is full (previous outbound transactions in progress), the internal bus interface signals a Retry to the transaction initiator.
- Read requests is fragmented into sub-requests based on the Max_Read_Request_Limit.
- When NPH credits are available, the ATU issue the read request when the head of the ONPQ has at least one entry and the ordering rules are satisfied.
- Once the request is issued, the Transaction Pending bit is set in the "PCI Express Device Status Register PE_DSTS".
- When a Completion with Completion Status of UR or CA is encountered, a flag is set and the ATU aborts the completion to the internal bus requester. The ONPQ is cleared of the transaction.
- Completions for subsequent sub-request that are already issued is marked for deletion and dropped once the completion returns.
- Once the transaction completes on the PCI Express Link, the ATU generates a completion transaction to return data to the internal bus requester.
- Once all outstanding request are satisfied, the Transaction Pending bit is cleared in the "PCI Express Device Status Register PE_DSTS"

10.For best performance, the user should designate the two Outbound Memory Windows as non-cachable and bufferable from the Intel XScale® processor. This assignment enables the Intel XScale® processor to issue multiple outstanding transactions to the Outbound Memory Windows, thereby, taking full advantage of the ATU outbound queue architecture. However, the user needs to be aware that the Outbound ATU queue architecture does not maintain strict ordering between read and write requests as described in Table 130, "ATU Outbound Data Flow Ordering Rules" on page 265. In the event that the user requires strict ordering to be maintained, the user must change the designation of this region of memory to be non-cachable/non-bufferable and enforce the requirement in software.

3.3.5 Outbound Configuration Cycle Translation

The outbound ATU provides a port programming model for outbound configuration cycles.

Performing an outbound configuration cycle to the PCI Express Link involves up to two internal bus cycles:

1. Writing Outbound Configuration Cycle Address Register (OCCAR) with the bus, device, function, and register number used during the configuration cycle. The value of this register directly maps to bytes 8-11 of the configuration transaction header. See Section 3.17.44, "PCI Express Message Control/Status Register - PEMCSR" on page 333 for information regarding configuration address cycle formats. This IB bus cycle enables the transaction.
2. Writing or reading the Outbound Configuration Cycle Data Register (OCCDR). A read causes a configuration cycle read to the PCI Express Link with the address in the outbound configuration cycle address register. Note that the Internal Bus read is executed as a split transaction. Similarly, a write initiates a configuration cycle write to PCI with the write data from the second processor cycle. Configuration cycles are non-burst and restricted to a single 32-bit word cycle[11]. This IB bus cycle executes the transaction.

When the Configuration Cycle Data Register is written, the data is latched and forwarded to the PCI Express Link. The Configuration Request TLP always uses function 0.

Note: Outbound configuration cycle data registers are not physical registers. They are a 4138xx memory mapped addresses used to initiate a transaction with the address in the associated address register. When the data register is accessed, the address is pulled from the "Outbound Configuration Cycle Address Register - OCCAR" to generate the TLP header and any write data is placed directly in the ONPDQ.

3.3.5.1 Outbound Configuration Cycle Error Conditions

When issuing configuration requests, the ATU must deal with receiving completions with Unsupported Request (UR) and Completer Abort (CA) status. When a UR or CA is received, the ATU interrupts the core by setting the Received Master Abort / Received Target Abort status in the "ATU Interrupt Status Register - ATUISR" on page 329. The read cycle is terminated with a DABORT on the internal bus.

When the completion is returned with poisoned data, the ATU sets the Detected Parity Error Interrupt status bit in the "ATU Interrupt Status Register - ATUISR" on page 329. The data is issued on the internal bus with bad parity.

3.3.5.2 Outbound Configuration Completions with Retry Status (CRS)

When issuing configuration requests, the ATU must deal with receiving a Configuration Request Retry Status (CRS). When a CRS is received, the ATU interrupts the core by setting the Received Configuration Retry Status in the "ATU Interrupt Status Register - ATUISR" on page 329. A configuration read that is completed with a CRS also results in a DABORT on the internal bus.

It is the responsibility of the software to reissue the configuration transaction.

11.The user should designate the memory region containing the OCCDR as non-cachable and non-bufferable from the Intel XScale® processor. This insures that all load/stores to the OCCDR are only of DWORD quantities. In event the user inadvertently issues a read to the OCCDR that crosses a DWORD address boundary, the ATU target aborts the transaction. All writes are terminated with a Single-Phase-Disconnect and only bytes 3:0 are relevant.

3.3.5.3 Outbound PCI Express Message Transactions

The ATU provides a port programming model to generate outbound PCI Express messages.

Generating an outbound message transaction to the PCI Express interface involves up to 5 internal bus cycles.

1. Write outbound message transaction header registers 0 - 3.
2. Write the data to the outbound message transaction payload register. This write causes the generation of the message TLP on the PCI Express interface.

Note: When the payload length is 0, a write to the payload register is still required but the written data is ignored and not sent as part of the message TLP.

3.3.5.4 Completion Timeout Mechanism

The ATU implements a completion timeout mechanism on all outbound requests that require completions. The timeout mechanism is fixed and ensures an expiration time between 16 and 32 ms.

3.4 Big Endian Byte Swapping

Each memory and I/O window has an associated byte swapping enable located in the following address translation registers:

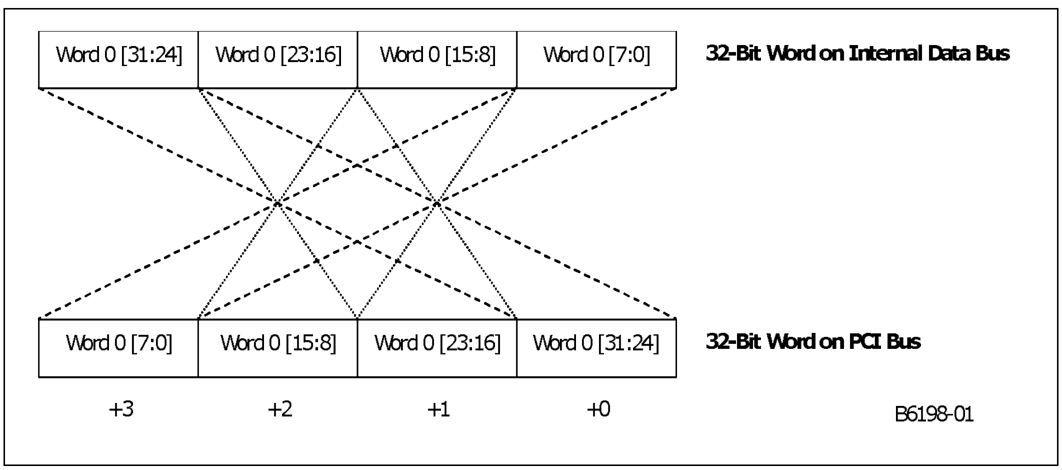
- bit 0 of Inbound Address Translate Value Register 0-2 (IATVR0-2)
- bit 0 of Inbound Expansion ROM Translate Value Register (ERTVR)
- bit 0 of Outbound I/O Window Translate Value Register (OIOWTVR)
- bit 27 of Outbound Upper Memory BAR 0-3 (OUMBAR0-3)

Note: The Messaging Unit (MU) Memory is mapped in PCI Window 0 (ATU Base Address Register 0) along with the MSI-X table structures. Byte swapping should not be enabled for BAR0 when using MSI-X.

3.4.1 Inbound Byte Swapping

When enabled, the swapping occurs as described in Figure 28, "Inbound Byte Swapping" on page 255. The bytes are swapped within a DWORD and byte swapping is performed for all transactions regardless of byte count.

Figure 28. Inbound Byte Swapping



3.4.2 Outbound Byte Swapping

When enabled, the swapping occurs as described in Figure 29, "Outbound Byte Swapping for Transaction with Byte Count of 1" on page 256, Figure 30, "Outbound Byte Swapping for Transactions with Byte Count of 2" on page 256, and Figure 31, "Outbound Byte Swapping Transaction with Byte Count of 3 or Larger" on page 256. The bytes are swapped within a 32-bit DWORD and the type of byte swapping performed is determined by the transaction byte count. For Byte Count of 3 or larger transactions, no byte swapping is performed.

Note: The byte swapping capability of the ADMA unit should be used to swap bytes within each DWORD for PCI-to-Memory Read/Write DMA transfers.

Figure 29. Outbound Byte Swapping for Transaction with Byte Count of 1

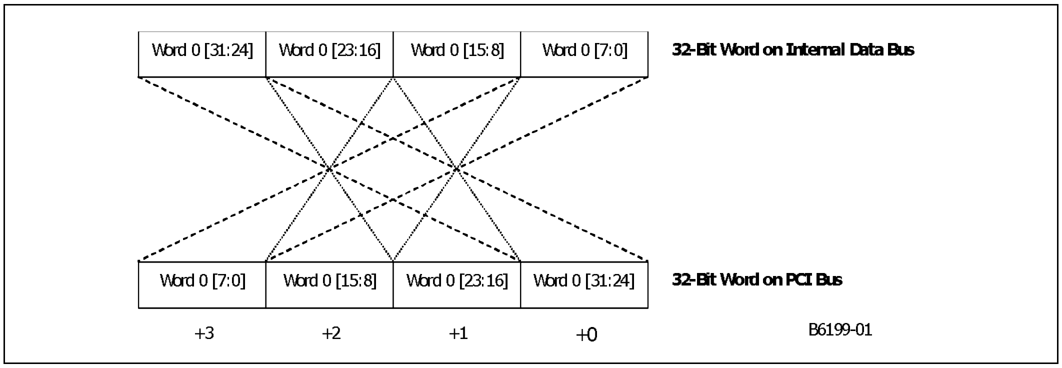


Figure 30. Outbound Byte Swapping for Transactions with Byte Count of 2

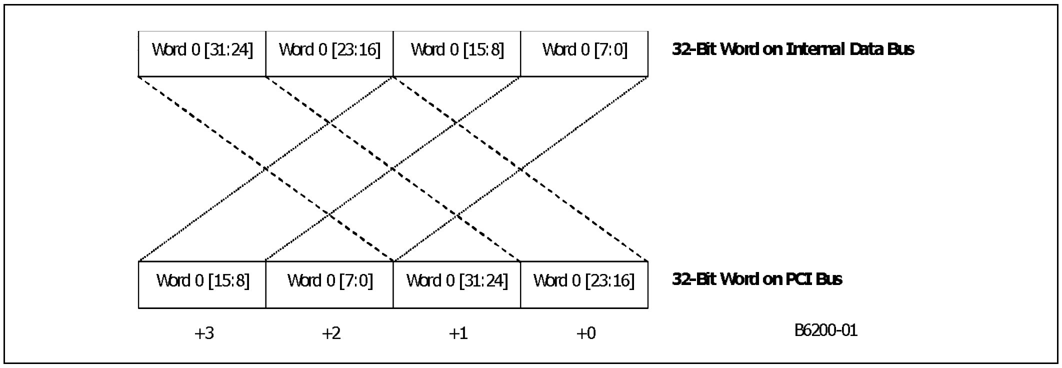
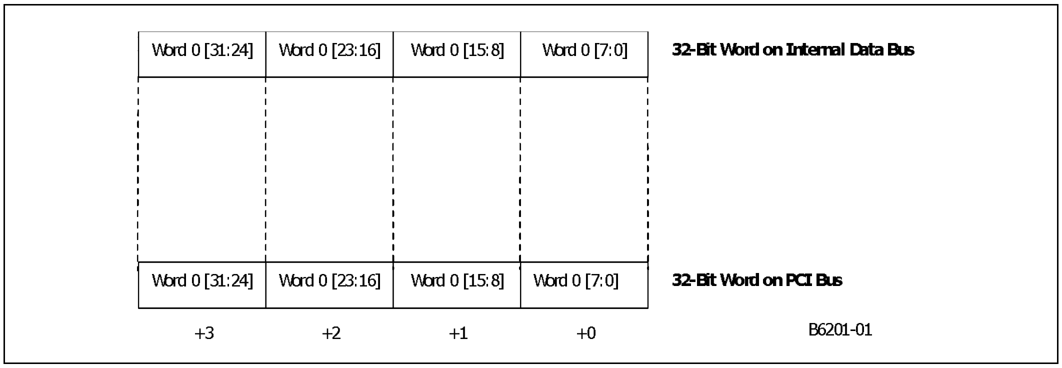


Figure 31. Outbound Byte Swapping Transaction with Byte Count of 3 or Larger



3.5 Messaging Unit

The Messaging Unit (MU) is used to transfer data between the PCI system and the 4138xx and notifies the respective system when new data arrives. The MU is located on the south internal bus of the 4138xx and is accessed via the ATU. The MU is described in Chapter 4.0, “Messaging Unit”.

The PCI window for messaging transactions is the 8 Kbytes of the inbound translation window defined by the Inbound ATU Base Address Register 0 (IABAR0) and the Inbound ATU Limit Register 0 (IALR0).

3.6 PCI Express Messages

PCI Express defines a new message mechanism that is used to communicate outside of the normal Memory, I/O, and Configuration Spaces. The messages received and initiated by 4138xx vary depending on whether the device is acting as a root complex or an endpoint. All the messages defined in the PCI Express specification and 4138xx's actions are listed in Table 125.

Table 125. Supported Message Types (Sheet 1 of 2)

<table>
<tr><th>Message</th><th>Description/ Comments</th><th>RC or End Point</th><th>Comment</th></tr>
<tr><td colspan="4">Standard Messages initiated by SRL</td></tr>
<tr><td>ERR_COR</td><td>Signal detection of a correctable error to Root Complex</td><td rowspan="3">RC, End Point</td><td rowspan="3">Set appropriate bit in ATUISR
RC: Also log in PCI Express Error Source Identification Register</td></tr>
<tr><td>ERR_NONFATAL</td><td>Signal detection of an uncorrectable error Root Complex</td></tr>
<tr><td>ERR_FATAL</td><td>Signal detection of a fatal error Root Complex</td></tr>
<tr><td>PM_Active_State_NAK</td><td>Rejection of request to enter a low power state from an End Point or Upstream Port.</td><td>RC</td><td></td></tr>
<tr><td>PME_Turn_Off[a]</td><td>Notification of pending turn-off of link clock and power</td><td>RC</td><td>Generated from PEMCSR.</td></tr>
<tr><td>PM_PME[b]</td><td>PME message conveying the ID of the PME originator</td><td>End Point</td><td></td></tr>
<tr><td>PME_TO_Ack</td><td>Acknowledge turn-off of link clock and power</td><td>End Point</td><td>Automatically generated on receipt of PME_Turn_Off message</td></tr>
<tr><td>Assert_INTx</td><td>Assert INTx virtual signal</td><td>End Point</td><td rowspan="2">Generate to reflect signals from Interrupt Controller.</td></tr>
<tr><td>Deassert_INTx</td><td>De-assert INTx virtual signal</td><td>End Point</td></tr>
<tr><td>Set_Slot_Power_Limit</td><td>Set Slot Power Limit in Upstream Port</td><td>RC</td><td>Generated due to write to Slot Capabilities register or anytime Link transitions from non-DL_Up status to a DL_Up status.</td></tr>
<tr><td>Vendor Defined Message Type 0/1</td><td>Vendor Specific Message. It is used by devices to communicate with SRL device core.</td><td>RC, End Point</td><td>Generated from Outbound Vendor Message register.</td></tr>
<tr><td>Attention_Indicator_X</td><td>These messages are issued by the root port to change the indicator on the device</td><td>RC</td><td rowspan="2">Generate due to change in indicator control bits in Slot Control Register</td></tr>
<tr><td>Power_Indicator_X</td><td>These messages are issued by the root port to change the indicator on the device</td><td>RC</td></tr>
<tr><td>Attention_Button_Pressed</td><td>Issued by endpoint when button pressed on device</td><td>End Point</td><td>Generated from PEMCSR</td></tr>
</table>

Table 125. Supported Message Types (Sheet 2 of 2)

Message	Description/ Comments	RC or End Point	Comment
Standard Messages accepted by SRL as target			
ERR_COR	Signal detection of a correctable error to Root Complex	RC	Log in PCI Express Advanced Error registers and interrupt the core via the root interrupt bit in the ATUISR
ERR_NONFATAL	Signal detection of an uncorrectable error Root Complex	RC	
ERR_FATAL	Signal detection of a fatal error Root Complex	RC	
PM_Active_State_NAK	Rejection of request to enter a low power state	End Point	
PME_Turn_Off	Notification of pending turn-off of link clock and power	End Point	Generate PME_TO_Ack message.
PM_PME	PME message conveying the ID of the PME originator	RC	Log in requester ID in the Root Status Register. Interrupt the core via the Root Complex interrupt bit in the ATUISR
PME_TO_Ack[c]	Acknowledge turn-off of link clock and power	RC	
Assert_INTx	Assert INTx virtual signal	RC	Captured and converted to signals to Interrupt Controller
Deassert_INTx	De-assert INTx virtual signal	RC	
Set_Slot_Power_Limit	Set Slot Power Limit in Upstream Port	End Point	Copy payload to Device Capabilities register and interrupt the core.
Vendor Defined Message Type 0/1	Vendor Specific Message. It is used by devices to communicate with SRL device core.	RC, End Point	Log in Inbound Vendor Message Register and interrupt the core.
Attention_Indicator_X	These messages are issued by the root port to change the indicator on the device	End Point	Log in the PEMCSR, and interrupt the core
Power_Indicator_X	These messages are issued by the root port to change the indicator on the device	End Point	Log in the PEMCSR, and interrupt the core
Attention_Button_Pressed	Issued by endpoint when button pressed on device	RC	Log in Slot Status Register and interrupt core

a. As a Root Complex, system software never requests SRL to broadcast the PME_Turn_Off message in preparation for removing the main power and clock sources. Thus, SRL doesn't generate this message as a RC. However, for reuse purpose the Message Unit is capable of generating this message.
b. As an End Point, SRL never generates a PM_PME message to the RC. It means SRL doesn't consume AUX power. However, for reuse purpose the Message Unit is capable of generating this message.
c. SRL doesn't generate PME_Turn_Off message. Thus it shouldn't receive PME_TO_Ack Message. However, for reuse purpose the Message Unit is capable of accepting this message.

3.7 Expansion ROM Translation Unit

The inbound ATU supports one address range (defined by a base/limit register pair) used for the Expansion ROM. Refer to the *PCI Local Bus Specification*, Revision 2.3 for details on Expansion ROM format and usage.

During a powerup sequence, initialization code from Expansion ROM is executed once by the host processor to initialize the associated device. The code can be discarded once executed. Expansion ROM registers are described in Section 3.17.22.

The inbound ATU supports an inbound Expansion ROM window which works like the inbound translation window. A read from the expansion ROM windows is forwarded to the internal bus. The address translation algorithm is the same as the inbound translation; see Section 3.3.1.1, "Inbound Address Translation" on page 237.

The Expansion ROM unit uses the ATU inbound transaction queue and the inbound read data queue.

Expansion ROM writes are not supported and result in a Completer Abort.

Note: Both the Memory Space enable and Expansion ROM Base Address Enable bits must be set to 1 before the ATU accepts accesses to its Expansion ROM.

3.8 ATU Queue Architecture

ATU operation and performance depends on queueing mechanism implemented between internal bus interface and PCI Express interface. Figure 23 indicates the ATU queue architecture consists of separate inbound and outbound queues. The function of each queue is described in the following sections.

3.8.1 Inbound Queues

The inbound data queues of the ATU support transactions initiated on a PCI Express Link and targeted at either 4138xx local memory or a 4138xx memory mapped register. Table 126 details the name and sizes of the ATU inbound data queues.

Table 126. Inbound Queues

Queue Mnemonic	Queue Name	Queue Size (Bytes)
IPDQ	Inbound Posted Data Queue	3.75 KBytes (240 credits)
IPHQ	Inbound Posted Header Queue	16 Headers
INPQ	Inbound Non Posted Queue	8 Headers[a]
ICPLDQ	Inbound Completion Data Queue	4 KBytes (infinite credits)[b]
ICPLHQ	Inbound Completion Header Queue	4 Headers (infinite)

a. Non Posted request with data (I/O and Configuration Writes) always use the 3DW header. The associated data is always 1 DW in size and can be stored the 4th DW of the header queue.
b. As a PCI Express endpoint the ATU must pre-allocate buffer space before issuing a read request. The ATU is required to advertise infinite credits.

3.8.1.1 Inbound Posted Queue Structure

The ATU Inbound Posted Queues consist of the inbound posted data queue (IPDQ) and the inbound posted header queue (IPHQ). The inbound posted data queue holds the data for posted (memory/message) transactions moving from a PCI Express Link to the internal bus and the header queue holds the corresponding address. The inbound posted data queue has a queue depth of 4 KBytes and moves posted transactions from the PCI Express Link to the internal bus. The corresponding header queue, IPHQ, is capable of holding 8 entries.

The following rules apply to the PCI Express Link interface and govern the acceptance of data into inbound posted queues:

- Posted transactions are drained from the head of the queue when the master interface has acquired bus ownership and transaction ordering and priority have been satisfied (see Section 3.8.3). A memory write transaction is considered drained from the queue when the entire amount of data entered on the PCI Express Link has been accepted by the internal bus target. Error conditions resulting in the cancellation of a write transaction only flush the transaction at the head of the data and address queue. All other transactions within the queues are considered still valid.

3.8.1.2 Inbound Non Posted Queue Structure

The inbound read queues are responsible for retrieving data from local memory and returning it to the PCI Express Link in response to a read transaction initiated from a PCI master. Up to 8 non posted transactions can be held in the INPQ. The read data is returned through OCPLDQ

For Configuration and I/O writes, both the header and data are placed in the INPQ. Configuration and I/O transactions always utilize the 3DW header and the data for the write transactions is always 1DW. So the data can be placed in the same queue as the header. The advantage to this is the NPD (non-posted data) credits can be advertised as infinite and the header credits prevents overrunning the INPQ.

Read requests are fragmented into 1K aligned requests before issuing to the internal bus/

3.8.1.3 Inbound Completion Queue Structure

The inbound completion queue provides insures space for all outstanding outbound read requests. This queue is 4KB in size and is used to order the completion data before returning it to the internal bus. When an outbound internal bus request is fragmented into multiple PCI Express request this queue ensures that the data is returned in order to the internal requesting agent.

3.8.1.4 Inbound Transaction Queues Command Translation Summary

Table 127. PCI to Internal Bus Command Translation for All Inbound Transactions

PCI Express TLP	Internal Bus Command
Memory Read	Read
Memory Read - Locked	none - Unsupported Request
Memory Write	Write
I/O Read	Read
I/O Write	Write
Configuration Read Type 0	Read
Configuration Write Type 0	Write
Configuration Read Type 1	none - Unsupported Request
Configuration Write Type 1	none - Unsupported Request
Message	none - Handled by express message unit
Message with Data	none - Handled by EMU
Message Advanced Switch	none - Unsupported Request
Message Advanced Switch with Data	none - Unsupported Request
Completion without Data	
Completion with Data	
Completion - Locked without Data (error condition)	none
Completion - Locked with Data	
Others (reserved encodings)	

3.8.2 Outbound Queues

The outbound queues of the ATU are used to hold read and write transactions from the core processor directed at the PCI Express Link. Each ATU outbound queue structure has a separate read queue, write queue, and address queue. Table 128 contains information about ATU outbound queues.

Table 128. Outbound Queues

Queue Mnemonic	Queue Name	Queue Size (Bytes)
OPDQ	Outbound Posted Data Queue	4 KBytes
OIPHQ	Outbound Posted Header Queue	16 Headers
ONPQ	Outbound Non Posted Queue	8 Headers[a]
OCPLDQ	Outbound Completion Data Queue	4 KBytes
OCPLHQ	Outbound Completion Header Queue	4 Headers

a. Non Posted request with data (I/O and Configuration Writes) always use the 3DW header. The associated data is always 1 DW in size and can be stored the 4th DW of the header queue.

The outbound queues are capable of holding outbound memory read, memory write, I/O read, and I/O write transactions. The type of transaction used is defined by the internal bus address and the command used on the internal bus. See Section 3.3.2 and Section 3.3.3 for details on outbound address translation.

When an internal bus agent initiates an outbound write transaction, the address is entered into the OWADQ (when not full). The data from the internal bus write is then entered into the OWQ and the transaction is forwarded to the PCI Express Link. When the write completes (or an error occurs), the address is flushed from the OWADQ. Data is flushed only for the master abort or target abort cases.

For outbound reads, the address is entered into the OTQ (when not full) and a split response termination is signaled to the requester on the internal bus. Read data is fetched and returned to the requester on the internal bus.

3.8.2.1 Relaxed Ordering and No Snoop Outbound Request Attributes

The ATU may set the Relaxed Ordering (RO) and/or the No Snoop (NS) bits for an outbound request.

For outbound Memory Read and Memory Write requests initiated by the ADMA the values of the attribute bits are controlled by the ADMA Descriptor Control Word.

For any other outbound requests, the NS and RO attribute bits is set to ‘0’.

Note: When the Enable Relaxed Ordering or Enable No Snoop bits are cleared in the “PCI Express Device Control Register - PE_DCTL” on page 344, then the ATU forces the RO and NS attributes to ‘0’ respectively for all transactions.

3.8.3 Transaction Ordering

Because the ATU can process multiple transactions, they must maintain proper ordering to avoid deadlock conditions and improve throughput. The ATU transaction ordering rules used by the 4138xx are listed in Table 129 for the inbound direction and Table 130 on page 265 for the outbound direction. The tables are based on the direction the transaction is moving, i.e. the data for inbound completion moves in the same direction as the data for an inbound write or the address/command for an inbound read.

Note: Outbound Non-Posted Writes are the result of Internal Bus Memory writes that are claimed by either the I/O translation window or the Outbound Configuration Cycle Data Register - OCCDR. Though these write requests arrive on the PCI Express Link as non-posted write requests, it is important to note that from the Intel XScale® processor's point of view, these internal bus memory write requests are posted into the Outbound ATU transaction queue. Thus, even though a write completion is returned to the ATU on the PCI Express Link for outbound non-posted write request, the write completion is not passed back through to the internal bus. Additionally, strong ordering between outbound memory (posted) write requests and outbound non-posted write requests are **not** maintained as indicated in Table 130 on page 265.

For best performance, the user should designate the two Outbound Memory Windows as non-cachable and bufferable from theIntel XScale® processor. This assignment enables the Intel XScale® processor to issue multiple outstanding transactions to the Outbound Memory Windows, thereby, taking full advantage of the ATU outbound queue architecture. However, the user needs to be aware that the Outbound ATU queue architecture does not maintain strict ordering between read and write requests as described in Table 130, "ATU Outbound Data Flow Ordering Rules" on page 265. In the event that the user requires strict ordering to be maintained, the user must change the designation of this region of memory to be non-cachable/non-bufferable and enforce the requirement in software.

Table 129. ATU Inbound Data Flow Ordering Rules

Row Pass Column?	Inbound Write or Message Request	Inbound Read Request	Inbound Configuration Write Request	Inbound Read Completion	Inbound Configuration or I/O Write Completion
Inbound Write or Message Request	No	Yes	Yes	Yes	Yes
Inbound Read Request	No	No	No	Yes	Yes
Inbound Configuration Write Request	No	No	No	Yes	Yes
Inbound Read Completion	No	Yes	Yes	Yes?	Yes?
Inbound Configuration or I/O Write Completion[a]	N/A	N/A	N/A	N/A	N/A

a. Inbound Completions for Configuration writes and I/O writes are not applicable since these transactions are never passed back to the Internal Bus Requester (Intel XScale® processor). The reason is that from the Intel XScale® processor's point of view, these write requests are posted into the Outbound ATU transaction queue.

Definitions of the terms used in Table 129 and Table 130 are as follows. PCI terms are noted in parenthesis:

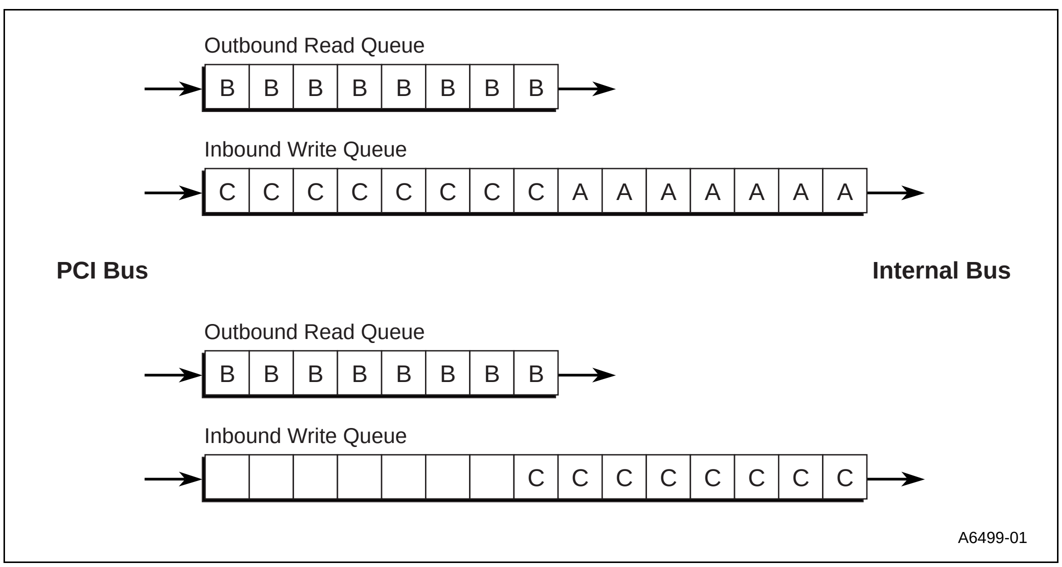
Table 130. ATU Outbound Data Flow Ordering Rules

Row Pass Column?	Outbound Write or Message Request	Outbound Read Request	Outbound Configuration Write Request	Outbound Read Completion	Outbound Configuration or I/O Write Completion
Outbound Write or Message Request	No	Yes	Yes	Yes	Yes
Outbound Read Request	No	No	No	Yes	Yes
Outbound Configuration Write Request	No	No	No	Yes	Yes
Outbound Read Completion	No	Yes	Yes	Yes	Yes
Outbound Configuration or I/O Write Completion	No	Yes	Yes	Yes	Yes

These transaction ordering rules define the way in which data moves in both directions through the ATU. In Table 129 and Table 130 a **NO** response in a box means that based on ordering rules, the current transaction (the row) can not pass the previous transaction (the column) under any circumstance. A **Yes** response in the box means that the current transaction is *allowed* to pass the previous transaction but is not required to, based on whether a consistent view of data or prevention of deadlocks are needed.

In the case of inbound write operations, multiple transactions may exist within the IPHQ and the corresponding IPDQ at any point in time. The ordering of these transactions is based on a time stamp basis. Transactions entering the queue are stamped with a relative time in relation to all other transactions moving in a similar direction.

Example 2. Inbound Queue Completion



In Example 2 on page 265, the inbound write and outbound read queues of the ATU are shown. In this example, transaction A entered the write queue at **Time 0**. Next, the ATU entered read data into the outbound read queue at **Time 1** (Transaction B). Finally, before the previous transactions could be cleared, another inbound write, Transaction C, was entered into the IWQ. The ordering in Table 129 states that nothing can pass an inbound write and therefore Transaction A must complete on the internal bus before Transaction B since an outbound read completion can not pass an inbound write. Also, Transaction A must complete before Transaction C since an inbound write can not pass another inbound write. Once Transaction A completes, Transaction C moves to the head of the IWQ. The two transactions at the head of the queues moving data in an inbound direction are now Transaction C, an inbound write, and Transaction B, an outbound read completion. Ordering states that an inbound write may pass an outbound read completion. This means that the arbitration mechanism now takes over to decide which completes. Note that ordering enforced the completion of Transaction A but arbitration dictated the completion of Transactions B and C.

The first action performed to determine which transaction is allowed to proceed (either inbound or outbound) is to apply the rules of ordering as defined in Table 129 and Table 130. Any box marked **No** must be satisfied first. For example, when an inbound read request is in ITQ and it was latched *after* the data in the IDWQ arrived (this is a configuration write), then ordering states that an Inbound Read Request may not pass an Inbound Configuration Write Request. Therefore, the Inbound Configuration Write Request must be cleared out of IDWQ before the Inbound Read Request is attempted on the internal bus. Once transaction ordering is satisfied, the boxes marked **Yes** are now resolved.

3.8.3.1 Transaction Ordering Summary

Table 131 and Table 132, define transaction ordering in relation to token assignment of the priority mechanism (see Section 3.8.3). These tables are read as follows:

1. As transaction enters the respective queue head, the question in column 2 is asked.
2. When all the answers in column 3 for a given transaction type assigns a token to the transaction at the head of the queue, a token is assigned. Otherwise, no token is assigned signifying that transaction ordering must first be satisfied. Any transaction with a token may be initiated on the bus.

Table 131. Inbound Transaction Ordering Summary

Transaction at Head of Queue	Question	Answer	Action
Inbound Posted Transaction Write in IPHQ	Is there an Inbound Posted Request with an earlier time stamp?	Yes	Do Not Assign Token Allow previous Transaction to Complete
		No	Assign Token
Inbound Non Posted Request in INPHQ	Is there an Inbound Write with an earlier time stamp?	Yes	Do Not Assign Token Allow previous Transaction to Complete
		No	Assign Token
	Is there an Inbound Non Posted Request with an earlier time stamp?	Yes	Do Not Assign Token Allow previous Transaction to Complete
		No	Assign Token
Inbound Completion in ICPLHQ	Is the relaxed order bit set in the header and the Enable Relaxed Ordering bit set in the ATUCR?	Yes	Assign Relaxed Order Token
		No	Check for earlier Posted Request
	Is there an Inbound Posted Request with an earlier time stamp?	Yes	Do Not Assign Token Allow previous Transaction to Complete
		No	Assign Token
	Is there and Inbound Completion with and earlier timestamp	Yes	Do Not Assign Token Allow previous Transaction to Complete
		No	Assign Token

Table 132. Outbound Transaction Ordering Summary

Transaction at Head of Queue	Question	Answer	Action
Outbound Posted in OPHQ	Is there an Outbound Posted Request with an earlier time stamp?	Yes	Do Not Assign Token Allow previous Transaction to Complete
		No	Assign Token
Outbound Non Posted Request in ONPHQ	Is there an Outbound Posted Request with an earlier time stamp?	Yes	Do Not Assign Token Allow previous Transaction to Complete
		No	Assign Token
	Is there an Outbound Non Posted Request with an earlier time stamp?	Yes	Do Not Assign Token Allow previous Transaction to Complete
		No	Assign Token
Outbound Completion in the OCPLHQ	Is there an Outbound Posted Request with an earlier time stamp?	Yes	Do Not Assign Token Allow previous Transaction to Complete
		No	Assign Token
	Is there an Outbound Completion with an earlier time stamp?	Yes	Do Not Assign Token and allow previous Transaction to Complete
		No	Assign Token

3.8.4 Byte Parity Checking and Generation

The ATU internal bus interface supports byte-wise parity protection on the internal bus. This includes ADDP[4:0] and DATAP[15:0] on the address bus (A[35:0]) and the data bus (D[127:0]) respectively.

For an outbound request the ATU check the address parity before claiming the request on the internal bus. When an error occurs, the transaction is not claimed. The Data parity information is captured off the internal bus and stored in the internal queues. At the PCI Express interface, the parity information is checked as the data packet is transferred on the link. When a parity error occurs, the packet is nullified by using the End Bad marker and state is updated so corrective action can be taken when the packet is replayed on the PCI Express link. Header parity errors, which can only occur in the retry buffer, results in a malformed packet that is dropped by the component on the other side of the link. Data parity errors results in the EP bit being set to notify the target of the TLP that the data is corrupt.

For an inbound write request, the ATU computes and appends address parity and data parity before placing the TLP in the inbound queues. When an ECRC violation is detected the packet is treated as when an address parity error occurred and the entire packet is dropped without forwarding it to the internal bus. When a poisoned TLP is received the parity for the entire payload is inverted so that the internal bus target detects bad parity on all bytes.

3.8.4.1 Parity Generation

Data parity signals include byte enables in the calculation. Table 133 lists data bits that are used for parity calculation. Parity bits are calculated by bit XOR-ing the data bits as shown in Table 133. As an example, the parity calculation for the lowest order byte of the data bus D[7:0] is calculated as follows:

Equation 12. DATAP0 = D[0] XOR D[1] XOR D[2] XOR D[3] XOR D[4] XOR D[5] XOR D[6] XOR D[7] XOR WBE[0]

Table 133. Parity Generation

Address/Data Parity Bit	Address/Data Bus	Address/Data Parity Bit	Address/Data Bus
ADDP4	A[35:32]	DATAP9	D[79:72], WBE[9]
ADDP3	A[31:24]	DATAP8	D[71:64], WBE[8]
ADDP2	A[23:16]	DATAP7	D[63:56], WBE[7]
ADDP1	A[15:8]	DATAP6	D[55:48], WBE[6]
ADDP0	A[7:0]	DATAP5	D[47:40], WBE[5]
DATAP15	D[127:120], WBE[15]	DATAP4	D[39:32], WBE[4]
DATAP14	D[119:112], WBE[14]	DATAP3	D[31:24], WBE[3]
DATAP13	D[111:104], WBE[13]	DATAP2	D[23:16], WBE[2]
DATAP12	D[103:96], WBE[12]	DATAP1	D[15:8], WBE[1]
DATAP11	D[95:88], WBE[11]	DATAP0	D[7:0], WBE[0]
DATAP10	D[87:80], WBE[10]		

3.8.4.2 Parity Checking

On an outbound request, address parity is checked on the address bus A[35:0]. The parity bits are checked by first bit XORing the address bits shown in Table 133 with the corresponding address parity bits, and then verifying when the result of each of the XORed operations are equal to zero. As an example, the parity calculation for the lowest order byte of the address bus A[7:0] is carried as follows:

Equation 13. PARITY_RESULT = ADDP0 XOR A[0] XOR A[1] XOR A[2] XOR A[3] XOR A[4] XOR A[5] XOR A[6] XOR A[7]

The parity logic uses the following algorithm. This algorithm logs the error when an error is detected.

```
check address parity
    if parity is good
        done
    else {error}
        create an error log
        Interrupt the core (if enabled)
```

On an outbound request with data, data parity is checked on the data bus D[127:0]. The parity bits are checked by first bit XORing the data bits shown in Table 133 with the corresponding data parity bits, and then verifying when the result of each of the XORed operations is equal to zero. As an example, the parity calculation for the lowest order byte of the data bus D[7:0] is carried as follows:

Equation 14. PARITY_RESULT = DATAP0 XOR D[0] XOR D[1] XOR D[2] XOR D[3] XOR D[4] XOR D[5] XOR D[6] XOR D[7] XOR WBE[0]

A non-zero result from the above operation indicates a parity error.

The parity logic uses the following algorithm, and this algorithm logs the error when an error is detected.

```
check data parity
    if parity is good
        done
    else {error}
        create an error log
        Interrupt the core (if enabled)
```

3.8.4.3 Parity Disabled

When software disables parity, the ATU does generate the parity inbound transactions, but does not check the parity on outbound transactions.

3.9 ATU Error Conditions

PCI Express and internal bus error conditions cause ATU to log header information and set status bits to inform error handling code of exact cause of error condition. Two sets of registers are provided to allow independent control by both the Host processor and the internal Intel XScale® microarchitecture. Error conditions and status can be found in the ATUSR. The basic flow for a PCI Express error is as follows:

- Log the Error in the PCI Express Advanced Error and the PCI Interface Error registers
- Set the bit in the ATU Status Register which corresponds to the error condition (master abort, target abort, etc.)
- Set the bit in the ATU Interrupt Status Register which corresponds to the error condition (master abort, target abort, etc.). This function is maskable for all PCI error conditions.
- The setting of the bit in the ATU Interrupt Status Register results in an interrupt being driven to the Intel XScale® processor.

Error conditions on one side of the ATU are generally propagated to the other side of the ATU and have different effects depending on the error. Error conditions and their effects are described in the following sections.

PCI Express error conditions and the action taken on the link are defined within the *PCI Express Base Specification*, Revision 1.0a. The ATU adheres to the error conditions defined within the PCI specification for both requester and completer operation. Error conditions on the internal bus are caused by an ECC error from the Memory Controller, (see Section 8.4, “ECC Interrupts/Error Conditions” on page 531 for details on memory controller error conditions), an Internal Bus Byte Parity Error, or by incorrect addressing resulting in an internal master abort or target abort. All actions on the PCI Express interface for error situations are dependent on the error control bits found in the ATU Command Register (Section 3.17.5, “ATU Command Register - ATUCMD” on page 298), the PCI Express Device Control Register (Section 3.17.59, “PCI Express Device Control Register - PE_DCTL” on page 344) and the PCI Express Advanced Error Masks (Section 3.17.71, “PCI Express Uncorrectable Error Mask - ERRUNC_MSK” on page 356 and Section 3.17.74, “PCI Express Correctable Error Mask - ERRCOR_MSK” on page 359).

The following sections detail all ATU error conditions on the PCI Express and 4138xx internal bus, action taken on these conditions, and status and control bits associated with error handling.

3.9.1 PCI Express Errors

PCI Express classifies errors as either Fatal, Uncorrectable or Correctable which allows the platform to map errors to a suitable handling mechanism. The control and mapping of errors into each of these categories are provided by the Advanced Error Handling registers. In addition to the specification defined registers, the ATU provides a duplicate set of error log in registers (see "PCI Interface Error Control and Status Register - PIE_CSR" on page 392) that allows the 4138xx component to respond to errors in an application specific fashion.

3.9.1.1 Role Based Error Reporting

In earlier versions of the PCI Express Specification, errors were reported by the agent that detected the error. The *PCI Express Base Specification*, Revision 1.1 implements a role based error reporting where the response to the errors is based on the components role in the transaction.In general, errors detected in Non Posted transactions are handled by the initial requestor and the completer may optionally send an advisory message to the root complex as an ERR_COR message. Errors in Posted transactions are still logged and reported by the target device.

Note: When the severity for the error is programmed to fatal in the PCI Express Uncorrectable Error Severity - ERRUNC_SEV register, then it is not an "Advisory Non-Fatal Error" and is signalled with an ERR_FATAL message. A fatal severity overrides all other Advisory Error control bits.

The following errors are considered "Advisory Non-Fatal Error" cases and have different handling depending based on the Transaction type.

- ECRC Check Failed
- Unsupported Request (UR)
- Completer Abort (CA)
- Unexpected Completion
- Poisoned TLP Received
- Completion Timeout

Table 134. Advisory Error Cases

Error Type	Posted	Non Posted	Completion
ECRC Check Failed	Not an Advisory Error - Send ERR_NONFATAL		
Unsupported Request	Not Advisory Error - Send ERR_NONFATAL	Advisory Error - Send ERR_COR	Signaled via device driver
Completer Abort	Not Advisory Error - Send ERR_NONFATAL	Advisory Error - Send ERR_COR	Signaled via device driver
Unexpected Completion	N/A	N/A	Advisory Error - Send ERR_COR
Poisoned TLP Received	When PIE_AEC bit 4 is set then treat as an Advisory Error - send ERR_COR. Else Not Advisory Error - Send ERR_NONFATAL	N/A	When PIE_AEC bit 5is set then treat as an Advisory Error - send ERR_COR. Else Not Advisory Error - Send ERR_NONFATAL
Completion Timeout	N/A	N/A	When PIE_AEC bit 6is set then treat as an Advisory Error - send ERR_COR. Else Not Advisory Error - Send ERR_NONFATAL

The different responses are described in detail in the following sections.

3.9.1.2 Malformed Packets

The following checks are made to detect malformed TLPs.

- Data Payload exceeds the length specified by the value in the Max_Payload_Size field of the Device Control Register.
- The value in the length field and the actual amount of data received do not match. The value in the length field applies only to data, TLP digest is not included in the length.
- A TLP with a 1b in TD field but without a TLP digest or a TLP with a TLP digest but without a 1b in TD field
- Address/Length combination which crosses a 4K boundary.
- When 4138xx is operating as Endpoint, and ATUE receives Assert_INTx/Deassert_INTx messages.
- Assert_INTx/Deassert_INTx messages do not use default Traffic Class (TC0)
- Power Management messages do not use default Traffic Class (TC0)
- Error Signalling Messages do not use default Traffic Class (TC0)
- Packets having undefined Type Field
- IO and Configuration requests are considered malformed when
  - TC[2:0] /= 000b
  - Attr[1:0] /= 00b
  - Length[9:0] /= 0000000001b
  - Last DW BE[3:0] /= 0000b
- For Read Completion, when length = 0 and the completion status /= 000, 001,010,

When a malformed packet is detected, the packet is dropped and the error is logged. No flow control information is updated for malformed packets.

3.9.1.3 ECRC Check Failed

Return ERR_NONFATAL / ERR_FATAL depending on the severity setting. ERR_COR is never generated.

This component in never an “Intermediary Receiver” so the advisory error condition does not apply.

3.9.1.4 Unsupported Request

Unsupported Requests are detected by the address decode and translation logic. A TLP is treated as unsupported in the following cases:

- the TLP fails to match any of the active Memory or I/O windows.
- a configuration TLP that targets an invalid function number
- receipt of a Vendor_Defined Type 0 message and
  - the inbound vendor message received interrupt mask is set (ATUIMR[25])
  - the inbound vendor_defined type 0 UR response bit is set (PEMCSR[14])
- a message request with an undefined or unsupported Message Code
- a poisoned I/O or Configuration request
- receipt of a Memory or I/O transaction while in a non-D0 power state
- receipt of a Memory Read Lock (MRdLk)

No checks are made to for address + length crossing a window boundary.

For posted transactions, this is **not** an Advisory Error and an ERR_NONFATAL is sent to the root complex.

For non-posted transactions, this is considered and Advisory Error. An ERR_COR is sent to the root complex and a completion with UR status is returned to the requestor.

Note: When the severity setting in "PCI Express Uncorrectable Error Severity - ERRUNC_SEV" register is fatal this is not an Advisory Error and an ERR_FATAL is sent to the root complex.

3.9.1.5 Completer Abort

Requests that target abort or master abort on the Internal Bus are treated as a Completer Abort.

These requests must have passed the Malformed TLP checks as well as the Unsupported Request checks before they are issued on the internal bus.

For posted transactions, an ERR_NONFATAL is sent to the root complex.

For non-posted transactions, an ERR_COR is sent to the root complex and a completion with CA status is returned to the requestor.

Note: When the severity setting in "PCI Express Uncorrectable Error Severity - ERRUNC_SEV" register is fatal this is not an Advisory Error and an ERR_FATAL is sent to the root complex.

3.9.1.6 Unexpected Completions

Unexpected completions occur when a completion transaction ID does not match a outstanding request. When the Requestor ID of the completion matches a valid function, the error gets logged in that function. Otherwise the error gets logged against all functions.

This an Advisory Non-Fatal Error and an ERR_COR is sent to the root complex.

Note: When the severity setting in "PCI Express Uncorrectable Error Severity - ERRUNC_SEV" register is fatal this is not an Advisory Error and an ERR_FATAL is sent to the root complex.

3.9.1.7 Poisoned TLP Received

Poisoned TLPs can be received for both Inbound Posted (Write/Message) and Inbound Completions The two TLP types can be handled differently.

Poisoned completions are passed through to the internal bus with bad parity. In addition to the Advanced Error requirements, the TLP header and DMA descriptor are logged in the PCI Interface Error Log (PIE_LOGx) registers and the Intel XScale® microarchitecture are interrupted. When bit 4 is set in the"PCI Express Advisory Error Control Register - PIE_AEC", this is treated as an "Advisory Error" and an ERR_COR is sent to the root complex; otherwise an ERR_NONFATAL is issued.

Poisoned Memory Writes are passed through to the internal bus with bad parity. In addition to the Advanced Error requirements, the TLP header is logged in the PCI Interface Error Log (PIE_LOGx) registers and the Intel XScale® microarchitecture are interrupted. When bit 5 is set in the"PCI Express Advisory Error Control Register - PIE_AEC", this is treated as an "Advisory Error" and an ERR_COR is sent to the root complex; otherwise an ERR_NONFATAL is issued.

For advisory errors, the firmware can reissue the transaction one or more (finite) times in an attempt to get valid data. When firmware decides to stop retrying the transaction it must escalate the error by setting the Generate ERR_NONFATAL bit in the "PCI Express Advisory Error Control Register - PIE_AEC".

Note: When the severity setting in "PCI Express Uncorrectable Error Severity - ERRUNC_SEV" register is fatal this is not an Advisory Error and an ERR_FATAL is sent to the root complex. Auto-recovery is discouraged as the ERR_FATAL message likely brings down the hierarchy.

3.9.1.8 Completion Timeout

When an out-bound non-posted request results in a completion timeout, the Advanced Error registers are updated and the initial request header and corresponding DMA descriptor tag are logged in the PCI Interface Error Log (PIE_LOGx) registers and the Intel XScale® microarchitecture are interrupted. When bit 6 is set in the"PCI Express Advisory Error Control Register - PIE_AEC", this is treated as an "Advisory Error" and an ERR_COR is sent to the root complex; otherwise an ERR_NONFATAL is issued.

For advisory errors, the firmware can reissue the transaction one or more (finite) times in an attempt to get valid data. When firmware decides to stop retrying the transaction it must escalate the error by setting the Generate ERR_NONFATAL bit in the "PCI Express Advisory Error Control Register - PIE_AEC".

Note: When the severity setting in "PCI Express Uncorrectable Error Severity - ERRUNC_SEV" register is fatal this is not an Advisory Error and an ERR_FATAL is sent to the root complex. Auto-recovery is discouraged as the ERR_FATAL message likely brings down the hierarchy.

3.9.2 Parity Error on the Internal Bus

The 4138xx provides support for byte-wise parity protection on the internal bus. The internal bus consists of a 36 bit address bus and 128 bit data bus; both are protected by byte-wise parity. The internal bus parity protection is provided independent of the operating mode of the ATU's PCI interface.

When initiating transactions on the internal bus, the ATU's internal bus interface generates byte-wise parity. As a target the ATU checks byte-wise parity.

3.9.3 ATU Error Summary

Table 135, "PCI Express Error Summary"summarizes the ATU error reporting for PCI Express Link errors, Table 136, "Root Complex Error Summary" summarizes the error reporting when operating as a Root Complex, and Table 137, "Internal Bus Error Summary" summarizes the ATU error reporting for internal bus errors. The tables assume that all error reporting is enabled through the appropriate command registers (unless otherwise noted).

Example: A poisoned TLP is received.

- Depending on the setting in the ERRUNC_SEV register, send ERR_FATAL/NON_FATAL to the root complex.
- Log the error in the Advanced Error registers (ADVERR_CTL, ADVERR_LOG) as well as the PCI Interface Error registers (PIE_CSR, PIE_LOG, PIE_DLOG)
- Set the Detected Parity Error, SERR# Asserted, and possible the Master Data Parity Error bits in the ATUCR.
- Set the Fatal Error Detected or Non-Fatal Error Detected in the Device Status register (PE_DSTS) depending on the error message sent
- Set the Poisoned TLP Received status bit in the ERRUNC_STS register
- Set the Poisoned TLP Received status bit in the PIE_STST register.
- Depending on the mask bits, set any or all of the following bits in the ATUISR
  — PCI Interface Error Interrupt
  — Uncorrectable Error Transmitted Interrupt
  — Detected Parity Error Interrupt
  — Master Data Parity Error Interrupt
- Possible mask bits are the SERR# Enable (ATUCMD[8]), Fatal/Non-Fatal Error Reporting Enable (PE_DCTL[2 or 1]), Interrupt Masks (ATUIMR[8,4,0]) and logging masks (ERRUNC_MSK[12] & PIE_MSK[12]).

Table 135. PCI Express Error Summary (Sheet 1 of 2)

Error Condition	Bus Protocol Action[a]	Affected Logging Register	Affected bits in Unit Status Register	Affected bits in Interrupt Status Register	Unit Interrupt Mask Bits
Transaction Layer Errors					
Poisoned TLP Received	Receiver: Send ERR_FATAL/ ERR_NONFATAL to Root Complex Log the header of the TLP that caused the error.	ADVERR_CTL ADVERR_LOGx PIE_CSR PIE_LOGx PIE_DLOG	ATUSR[15, 14, 8] PE_DSTS[2 or 1] ERRUNC_STS[12] PIE_STS[12]	ATUISR[10, 8, 4, 0]	ATUCMD[8] PE_DCTL[2 or 1] ATUIMR[8, 4, 0] ERRUNC_MSK[12] PIE_MSK[12]
ECRC Check Failed	Receiver: Send ERR_FATAL/ ERR_NONFATAL to Root Complex Log the header of the TLP that caused the error.	ADVERR_CTL ADVERR_LOGx PIE_CSR PIE_LOGx PIE_DLOG	ATUSR[14] PE_DSTS[2 or 1] ERRUNC_STS[19] PIE_STS[19]	ATUISR[10, 8]	ATUCMD[8] PE_DCTL[2 or 1] ATUIMR[8] ERRUNC_MSK[19] PIE_MSK[19]
Unsupported Request	Receiver: Send ERR_FATAL/ ERR_NONFATAL to Root Complex Log the header of the TLP that caused the error.	ADVERR_CTL ADVERR_LOGx PIE_CSR PIE_LOGx PIE_DLOG	ATUSR[14] PE_DSTS[3] PE_DSTS[2 or 1] ERRUNC_STS[20] PIE_STS[20]	ATUISR[10, 8]	ATUCMD[8] PE_DCTL[2 or 1] ATUIMR[8] ERRUNC_MSK[20] PIE_MSK[20]
Completion Timeout	Requester: Send ERR_FATAL/ERR_NON FATAL to Root Complex	PIE_CSR PIE_LOG[3:0] PIE_DLOG	ATUSR[14] PE_DSTS[2 or 1] ERRUNC_STS[14] PIE_STS[14]	ATUISR[10, 8]	ATUCMD[8] PE_DCTL[2 or 1] ATUIMR[8] ERRUNC_MSK[14] PIE_MSK[14]
Completer Abort	Completer: Send ERR_FATAL/ERR_NON FATAL to Root Complex Log the header of the TLP that caused the error.	ADVERR_CTL ADVERR_LOGx PIE_CSR PIE_LOGx PIE_DLOG	ATUSR[14, 11] PE_DSTS[2 or 1] ERRUNC_STS[15] PIE_STS[15]	ATUISR[10, 8, 5, 2]	ATUCMD[8] PE_DCTL[2 or 1] ATUIMR[8, 5, 2] ERRUNC_MSK[15] PIE_MSK[15]
Unexpected Completion	Receiver: Send ERR_FATAL/ ERR_NONFATAL to Root Complex Log the header of the TLP that caused the error.	ADVERR_CTL ADVERR_LOGx PIE_CSR PIE_LOGx PIE_DLOG	ATUSR[14] PE_DSTS[2 or 1] ERRUNC_STS[16] PIE_STS[16]	ATUISR[10, 8]	ATUCMD[8] PE_DCTL[2 or 1] ATUIMR[8] ERRUNC_MSK[16] PIE_MSK[16]
Receiver Overflow	Receiver: Send ERR_FATAL/ ERR_NONFATAL to Root Complex	PIE_CSR PIE_LOG[3:0] PIE_DLOG	ATUSR[14] PE_DSTS[2 or 1] ERRUNC_STS[17] PIE_STS[17]	ATUISR[10, 8]	ATUCMD[8] PE_DCTL[2 or 1] ATUIMR[8] ERRUNC_MSK[17] PIE_MSK[17]
Flow Control Protocol Error	Receiver: Send ERR_FATAL/ ERR_NONFATAL to Root Complex	PIE_CSR	ATUSR[14] PE_DSTS[2 or 1] ERRUNC_STS[13] PIE_STS[13]	ATUISR[10, 8]	ATUCMD[8] PE_DCTL[2 or 1] ATUIMR[8] ERRUNC_MSK[13] PIE_MSK[13]

Table 135. PCI Express Error Summary (Sheet 2 of 2)

Error Condition	Bus Protocol Action[a]	Affected Logging Register	Affected bits in Unit Status Register	Affected bits in Interrupt Status Register	Unit Interrupt Mask Bits
Malformed TLP	Receiver: Send ERR_FATAL/ ERR_NONFATAL to Root Complex Log the header of the TLP that caused the error.	ADVERR_CTL ADVERR_LOGx PIE_CSR PIE_LOGx PIE_DLOG	ATUSR[14] PE_DSTS[2 or 1] ERRUNC_STS[18] PIE_STS[18]	ATUISR[10, 8]	ATUCMD[8] PE_DCTL[2 or 1] ATUIMR[8] ERRUNC_MSK[18] PIE_MSK[18]
Physical Layer Errors					
Receiver Error	Send ERR_COR to Root Complex		PE_DSTS[0] ERRCOR_STS[0]	ATUISR[9]	PE_DCTL[0] ATUIMR[9]
Training Error	Send ERR_FATAL/ ERR_NONFATAL to Root Complex	PIE_CSR	PE_DSTS[2 or 1] ERRUNC_STS[0] PIE_STS[0]	ATUISR[8]	ATUCMD[8] PE_DCTL[2 or 1] ATUIMR[8]
Data Link Layer Errors					
Bad TLP	*Receiver*: Send ERR_COR to Root Complex		PE_DSTS[0] ERRCOR_STS[6]	ATUISR[9]	PE_DCTL[0] ERRCOR_MSK[6] ATUIMR[9]
Bad DLLP	Receiver: Send ERR_COR to Root Complex		PE_DSTS[0] ERRCOR_STS[7]	ATUISR[9]	PE_DCTL[0] ERRCOR_MSK[7] ATUIMR[9]
Replay Timeout	Transmitter: Send ERR_COR to Root Complex		PE_DSTS[0] ERRCOR_STS[12]	ATUISR[9]	PE_DCTL[0] ERRCOR_MSK[12] ATUIMR[9]
REPLAY_NUM Rollover	Transmitter: Send ERR_COR to Root Complex		PE_DSTS[0] ERRCOR_STS[8]	ATUISR[9]	PE_DCTL[0] ERRCOR_MSK[8] ATUIMR[9]
Data Link Layer Protocol Error	Send ERR_FATAL/ ERR_NONFATAL to Root Complex	PIE_CSR	PE_DSTS[2 or 1] ERRUNC_STS[4] PIE_STS[4]	ATUISR[10, 8]	ATUCMD[8] PE_DCTL[2 or 1] ATUIMR[8] ERRUNC_MSK[4] PIE_MSK[4]
Device Specific Errors					
Received Completion with UR status	None	PIE_CSR PIE_LOG[3:0] PIE_DLOG	ATUSR[13] PIE_STS[31]	ATUISR[10, 3]	ATUIMR[3] PIE_MSK[31]
Received Completion with CA status	None	PIE_CSR PIE_LOG[3:0] PIE_DLOG	ATUSR[12] PIE_STS[30]	ATUISR[10, 1]	ATUIMR[1] PIE_MSK[30]
Poisoned TLP Transmitted	None	PIE_CSR PIE_LOG[3:0] PIE_DLOG	ATUSR[8] PIE_STS[29]	ATUISR[10, 0]	ATUIMR[0] PIE_MSK[29]
Outbound Header Parity Error detected	None	PIE_CSR PIE_LOG[3:0] PIE_DLOG	PIE_STS[28]	ATUISR[10]	PIE_MSK[28]

a. ERR_FATAL / ERR_NONFATAL action is determined by the severity bits in the ERRUNC_SEV register.

Table 136. Root Complex Error Summary

Error Condition	Bus Protocol Action	Affected Logging Register	Affected bits in Unit Status Register	Affected bits in Interrupt Status Register	Unit Interrupt Mask Bits
Root Complex Errors					
Received ERR_COR	None	RERR_ID	RERR_SR[1, 0]	ATUISR[12, 11]	ATUIMR[12] RERR_CMD[0]
Received ERR_NONFATAL	None	RERR_ID	RERR_SR[5, 4, 3, 2]	ATUISR[12, 11]	ATUIMR[12] RERR_CMD[1]
Received ERR_FATAL	None	RERR_ID	RERR_SR[6, 4, 3, 2]	ATUISR[12, 11]	ATUIMR[12] RERR_CMD[2]

Table 137. Internal Bus Error Summary

Error Condition	Bus Protocol Action	Affected Logging Register	Affected bits in Unit Status Register	Affected bits in Interrupt Status Register	Unit Interrupt Mask Bits
Internal Bus Errors					
Master Abort on inbound requests	Signal Completer Abort	See Completer Abort above	See Completer Abort above	*Completer Abort and* ATUISR[5]	*Completer Abort and* ATUIMR[5]
AERR on inbound requests	Signal Completer Abort	See Completer Abort above	See Completer Abort above	*Completer Abort and* ATUISR[5]	*Completer Abort and* ATUIMR[5]
Target Abort on inbound requests	Signal Completer Abort	See Completer Abort above	See Completer Abort above	See Completer Abort above	See Completer Abort above
Data Parity Error on outbound Writes or Completions	Set EP bit in TLP Header. Posted Writes may get flushed when bit 8 of ATUCR is set.	See Poisoned TLP Transmitted above	See Poisoned TLP Transmitted above	See Poisoned TLP Transmitted above	See Poisoned TLP Transmitted above
Header Parity Error in Link Layer Retry Buffer	TLP transmitted as Malformed. Posted Writes may get flushed when bit 8 of ATUCR is set.	See Outbound Header Parity Error above			INTPND3[3]

3.10 PCI Express Hot-Plug Support

The PCI Express architecture is designed to natively support both Hot-Plug and hot remove of devices. This section defines the usage model defined for all the ATU.

ATU supports the receipt and generation of Hot-Plug messages via the Inbound/Outbound message header registers. When a Hot-Plug message is received, the core can then utilize the GPIOs to change the indicator lights, or conversely sample the switch status and generate a message as appropriate.

3.11 Reset

The PCI Express specification defines three distinct types of reset: cold, warm, and hot.

The fundamental reset that occurs following initial power on is considered a hot reset. The assertion of the PRST# or WARM_RST# pins are considered warm resets. The receipt of the inband reset training sequence is considered a hot reset.

Sticky bits are preserved under the WARM_RST# and inband hot reset conditions.

Additional reset details are specified in Chapter 17.0, "Clocking and Reset".

3.12 Message-Signaled Interrupts

The Messaging Unit is responsible for the generation of all of the Outbound Interrupts from the 4138xx. These interrupts can be delivered to the Host Processor via the legacy Assert_INTx/Deassert_INTx messages or the Message Signaled Interrupt (MSI) mechanism.

When a host processor enables Message-Signaled Interrupts (MSI) on the 4138xx, an outbound interrupt is signaled to the host via a posted write instead of the legacy Assert/Deassert messages.

In support of MSI, the 4138xx implements the MSI capability structure. The capability structure includes the Section 4.7.20, “MSI Capability Identifier Register - Cap_ID” on page 429, the Section 4.7.21, “MSI Next Item Pointer Register - MSI_Next_Ptr” on page 430, the Section 4.7.23, “Message Address Register - Message_Address” on page 432, the Section 4.7.24, “Message Upper Address Register - Message_Upper_Address” on page 433 and theSection 4.7.25, “Message Data Register- Message_Data” on page 434.

The Message Unit generates MSIs by writing to the MSI port via the internal bus. The ATU generates a write transaction whenever the Message Unit writes to the MSI port, using the address specified in the Section 4.7.23, “Message Address Register - Message_Address” on page 432 and Section 4.7.24, “Message Upper Address Register - Message_Upper_Address” on page 433 and the data provided in the Section 4.7.25, “Message Data Register- Message_Data” on page 434.

3.12.1 Legacy Interrupts

The ATU supports the generation of the legacy Assert_INTx/Deassert_INTx interrupt messages.

3.12.2 Internal Interrupts

The ATU has 4 internal interrupts that connect to the internal Interrupt Controller Unit.

ATU Error Interrupt

ATU Inbound Message Interrupt

ATU Configuration Write Interrupt

ATU BIST Interrupt

3.13 Vital Product Data

Vital Product Data (VPD) provides detailed information to the system regarding the hardware, software and microcode elements of a device. This information may include Part Number, Serial Number or other detailed information. This information resides on a non-volatile storage device (i.e., Flash Memory) attached to the 4138xx. In addition VPD also provides a mechanism for storing information such as performance or failure data on the device being monitored.

Support of VPD involves the implementation of the VPD Extended Capabilities List Item in the Primary ATU. The VPD Extended capabilities header consists of four registers, the "VPD Capability Identifier Register - VPD_Cap_ID" on page 335, the "VPD Next Item Pointer Register - VPD_Next_Item_Ptr" on page 335, the "VPD Address Register - VPDAR" on page 336, and the "VPD Data Register - VPDDR" on page 336.

Scheduled by Intel XScale® processor interrupts, the 4138xx may be used to retrieve or store VPD information through the VPD extended capabilities list item.

Please consult Appendix I of the *PCI Local Bus Specification*, Revision 2.3 for the definitions of compliant VPD format.

3.13.1 Configuring Vital Product Data Operation

By default, the 4138xx VPD functionality is not configured for operation. Specifically, the VPD Extended Capabilities List Item are not discovered during a PCI Express Link scan and the ATUs VPD interrupt status bit in the "ATU Interrupt Status Register - ATUISR" on page 329 are masked by the "ATU Interrupt Mask Register - ATUIMR" on page 332. The following steps should be followed to properly configure the 4138xx's support for VPD:

1. The 4138xx must be strapped to Retry Type 0 Configuration cycles following the deassertion of **P_RST#**. Enabling this configuration cycle retry mechanism insures that the Intel XScale® processor can make the VPD Extended Capabilities List Item visible before the system configures the 4138xx. The configuration retry mechanism is controlled through bit 2 of the "PCI Configuration and Status Register - PCSR" on page 327.
2. When the configuration retry mechanism is strapped enabled as described in step 1, typically, the 4138xx would also be strapped such that the Intel XScale® processor would immediately boot following the deassertion of **P_RST#** (bit 1 of the PCSR), though this is not required.
3. The Intel XScale® processor writes 90H to the "PCI Express Next Item Pointer Register - PCIE_NXTP" on page 341. This links the PCI Express Capabilities List Item to the VPD Capabilities List Item.
4. The Intel XScale® processor clears bit 17 of the ATUIMR to enable the ATUs VPD interrupt status bit.

3.13.2 Accessing Vital Product Data

The VPD Capabilities List Item provides three fields which the system uses to access the Vital Product Data:

VPD Address	DWORD Aligned Byte address of the VPD to be accessed which is represented by VPDAR[14:0]. Note that this means that the maximum size of the VPD is 128 Kbytes. The user may pick any 128 Kbyte block of memory in the storage component for the VPD.
Flag	The flag register is used to indicate when the transfer between the VPD Data Register and the storage component is completed. The flag is in VPDAR[15] which means that the Flag is written at the same time that VPD address is written.
VPD Data	Four bytes of VPD Data can be read or written through this field which is represented by VPDDR[31:0]. The least significant byte of this register represents the byte at the VPD Address (VPDAR[14:0]). Four bytes are always transferred between this register and the VPD storage component.

3.13.2.1 Reading Vital Product Data

Using the fields defined in the VPD Capabilities List Item, the 4138xx reads Vital Product Data using the following sequence of events:

1. Host processor executes a configuration write of the VPD address to the VPDAR with the Flag cleared.
2. An interrupt to the Intel XScale® processor is triggered and bit 17 of the ATUISR is set. Meanwhile, the host processor polls the VPDAR register waiting for the Flag to be set.

Warning: When any configuration writes to either the VPDAR or the VPDDR occur prior to the Flag being set, the results of the original read operation are unpredictable.

3. Using the VPD Address, the Intel XScale® processor retrieves the Vital Product Data from the VPD storage component (i.e., Flash Memory).
4. The Intel XScale® processor then writes this data to VPD Data Register (VPDDR).
5. The Intel XScale® processor clears the VPD interrupt status bit in the ATUISR.
6. The Intel XScale® processor then sets the Flag in the VPDAR register.
7. When the host processor detects that the Flag has been set, the host processor then reads the retrieved VPD from the VPDDR.

3.13.2.2 Writing Vital Product Data

Using the fields defined in the VPD Capabilities List Item, the 4138xx writes Vital Product Data using the following sequence of events:

1. Host processor executes a configuration write of the VPD data to be written to the VPDDR.
2. Host processor executes a configuration write of the VPD address to the VPDAR with the Flag set.
3. An interrupt to the Intel XScale® processor is triggered and bit 17 of the ATUISR is set. Meanwhile, the host processor polls the VPDAR register waiting for the Flag to be cleared.

Warning: When any configuration writes to either the VPDAR or the VPDDR occur prior to the Flag being cleared, the results of the original write operation are unpredictable.

4. Using the VPD Address, the Intel XScale® processor writes the Vital Product Data from the VPDDR to the VPD storage component (i.e., Flash Memory).
5. The Intel XScale® processor clears the VPD interrupt status bit in the ATUISR.
6. The Intel XScale® processor then clears the Flag in the VPDAR register.
7. When the host processor detects that the Flag has been cleared, the host processor has been informed that the VPD write operation is complete.

3.14 Multi-Function Support

The 4138xx supports only one function, either ATU (TPER mode) or TPMI (IOC mode).

3.14.1 PCI Express Interface Control Parameters

The following registers are located in the configuration space header and extended space and provide control of the PCI Express Interface. The effect of each bit is detailed below.

Table 138. PCI Express Interface Control Parameters Usage (Sheet 1 of 2)[a]

Register Name	Register Bits Description	Usage
ATU Command Register - ATUCMD	Bit 8 - SERR# Enable	Each function can independently control this bit.
	Bit 6 - Parity Error Response	The Parity Error Response bit from each function is logically ORed and then fed to the PCI Interface. This implies that Parity Error Response is globally enabled when only one of the functions enables Parity Error Response.
	Bit 4 - MWI Enable	Not applicable.
	Bit 2 - Bus Master Enable	Each function can independently control this bit.
	Bit 1 - Memory Enable	Each function can independently control this bit.
	Bit 0 - I/O Enable	Each function can independently control this bit.
ATU BIST Register - ATUBISTR	Entire Register	Each Function can independently control this register
PCI Express Device Control Register - PE_DCTL	Bit[14:12] - Max_Read_Request_Size	Use smallest programmed value when functions have different values.
	Bit[11] - Enable No Snoop	Each Function can independently control this bit.
	Bit[10] - Aux Power PM Enable	Not applicable.
	Bit[9] - Phantom Functions Enable	Not applicable.
	Bit[8] - Extended Tag Field Enable	Not applicable.
	Bit[7:5] - Max_Payload_Size	Use smallest programmed value when functions have different values.
	Bit[4] - Enable Relaxed Ordering	Each Function can independently control this bit.
	Bit[3] - Unsupported Request Reporting Enable	Each function can independently control this bit.
	Bit[2] - Fatal Error Reporting Enable	Each function can independently control this bit.
	Bit[1] - Non- Fatal Error Reporting Enable	Each function can independently control this bit.
	Bit[0] - Correctable Error Reporting Enable	Each function can independently control this bit.
PCI Express Link Control Register PE_LCTL	Bit[7] - Extended Sync	The bit from each function is logically ORed and then fed to the PCI-E Interface.
	Bit[6] - Common Clock Configuration	Each function can independently control this bit.
	Bit[5] - Retrain Link	Applicable to Root Complex only.
	Bit[4] - Link Disable	Applicable to Root Complex only.
	Bit[3] - Read Completion Boundary (RCB) Control	Applicable to Root Complex only.
	Bit[1:0] - Active State PM Control	Inactive state when one of the functions is programmed as inactive.

Table 138. PCI Express Interface Control Parameters Usage (Sheet 2 of 2)[a]

Register Name	Register Bits Description	Usage
PCI Express Uncorrectable Error Mask - ERRUNC_MSK	Entire Register	Each function can independently control these bits.
PCI Express Uncorrectable Error Severity - ERRUNC_SEV	Entire Register	Each function can independently control these bit.
PCI Express Correctable Error Mask - ERRCOR_MSK	Entire Register	Each function can independently control these bit.
Advanced Error Control and Capability Register - ADVERR_CTL	Entire Register	The bits from each function is logically ORed and then fed to the PCI-E Interface.

a. This table is referring to only enabled functions. And in root complex mode multi-function is not applicable.

3.14.2 PCI Express Interface Status Reporting

The following registers are located in the configuration space header and extended space and provide status (error conditions) of the PCI Express Interface.

Table 139. PCI Express Interface Status Reporting Usage [a]

Register Name	Register Bits Description	Usage
ATU Status Register - ATUSR	Bit 15 - Detected Parity Error	Poisoned TLP received is reported in the function involved.
	Bit 14 - SERR# Asserted	SERR# Asserted is reported in the function involved.
	Bit 13 - Master Abort	Received Unsupported Request Completion is reported in the function involved.
	Bit 12 - Received Target Abort	Received Completer Abort Completion is reported in the function involved.
	Bit 11 - Signaled Target Abort	Transmitted Completer Abort Completion is reported in the function involved.
	Bit 8 - Bit Master Parity Error	Master Parity Error is reported to only the function involved.
PCI Express Device Status Register PE_DSTS	Entire Register	Reported per function.
PCI Express Link Status Register PE_LSTS	Entire Register	Reported per function.
PCI Express Uncorrectable Error Status - ERRUNC_STS	Entire Register	Reported per function
PCI Express Correctable Error Status - ERRCOR_STS	Entire Register	Reported per function
PCI Express Advanced Error Header Log - ADVERR_LOG0	Entire Register	Reported per function
PCI Express Advanced Error Header Log - ADVERR_LOG1	Entire Register	Reported per function
PCI Express Advanced Error Header Log - ADVERR_LOG2	Entire Register	Reported per function
PCI Express Advanced Error Header Log - ADVERR_LOG3	Entire Register	Reported per function

a. This table is referring to only enabled functions. And in root complex mode multi-function is not applicable.

3.15 Root Complex Functionality

The 4138xx does not support Root Complex.

3.16 Embedded Bridge Functionality

Note: Not supported on 4138xx.

3.17 Register Definitions

Every PCI device implements its own separate configuration address space and configuration registers. The PCI Express Specification extends the configuration space to 4096 bytes as compared to 256 bytes allowed by *PCI Local Bus Specification*, Revision 2.3. The ATU configuration space is divided into a PCI 2.3 compatible region consisting of the first 256 bytes and an extended PCI express configuration space region consisting of the remaining space. The PCI 2.3 compatible space provides support for legacy operating systems. The first 64 bytes must adhere to a predefined header format.

Figure 32 defines the header format. Table 141 shows the PCI configuration registers, listed by internal bus address offset. Table 141 shows the entire ATU configuration space (including header and extended registers) and the corresponding section that describes each register. Note that all configuration read and write transactions is accepted on the internal bus as 32-bit transactions. Refer to Chapter 19.0, "Peripheral Registers".

Figure 32. ATU Interface Configuration Header Format

ATU Device ID		Vendor ID		00H
Status		Command		04H
ATU Class Code			Revision ID	08H
ATUBISTR	Header Type	Latency Timer	Cacheline Size	0CH
Inbound ATU Base Address 0				10H
Inbound ATU Upper Base Address 0				14H
Inbound ATU Base Address 1				18H
Inbound ATU Upper Base Address 1				1CH
Inbound ATU Base Address 2				20H
Inbound ATU Upper Base Address 2				24H
Reserved				28H
ATU Subsystem ID		ATU Subsystem Vendor ID		2CH
Expansion ROM Base Address				30H
Reserved			Capabilities Pointer	34H
Reserved				38H
Maximum Latency	Minimum Grant	Interrupt Pin	Interrupt Line	3CH

The ATU is programmed via a Type 0 configuration command on the PCI interface. See Section 3.3.1.5, "Inbound Configuration Cycle Translation (ID Routed)" on page 242. ATU configuration space is function number zero of the 4138xx single-function PCI device.

Beyond the required 64 byte header format, ATU configuration space implements extended register space in support of the units functionality. Refer to the *PCI Express Base Specification*, Revision 1.0a for details on accessing and programming configuration register space.

3.17.1 Extended Capabilities Registers

The ATU unit includes 5 extended capability configuration spaces beginning at configuration offset 90H, 98H, A0H, B0H, and D0H. The extended configuration spaces can be accessed by a device on the PCI interface through a mechanism defined in the PCI Express Specification.

In the ATU Status Register (Section 3.17.6) the appropriate bit is set indicating that the Extended Capability Configuration space is supported. When this bit is read, the device can then read the Capabilities Pointer register (Section 3.17.23) to determine the configuration offset of the Extended Capabilities Configuration Header. The format of these headers are depicted in Figure 33, Figure 35, Figure 36and Figure 37.

Figure 33. ATU Interface Extended Configuration Header Format (Power Management)

Power Management Capabilities	Next Item Pointer	Capability Identifier	98H
Reserved	Power Management Control/Status		9CH

The first byte at the Extended Configuration Offset 98H is the ATU Capability Identifier Register for the *PCI Bus Power Management Interface Specification*, Revision 1.1.

Following the Capability Identifier Register is the single byte Next Item Pointer Register (Section 3.17.51, "PM Next Item Pointer Register - PM_Next_Item_Ptr" on page 337) which indicates the configuration offset of an additional Extended Capabilities Header, when supported. In the ATU, the Next Item Pointer Register is set to D0H indicating that there is an additional Extended Capabilities Headers supported in the ATUs configuration space.

To enable the *PCI Bus Power Management Interface Specification*, Revision 1.1 compliance support, the Power State Transition interrupt mask in bit 8 of the ATUIMR needs to be cleared. It is the configuration software's responsibility to properly enable and initialize the ATUs Power Management Interface before the Configuration Cycle Retry Bit in the Section 3.17.41, "PCI Configuration and Status Register - PCSR" on page 327 is cleared in order for the ATU to be *Advanced Configuration and Power Interface Specification,* Revision 2.0 compliant.

Figure 34. ATU Interface Extended Configuration Header Format (MSI-X Capability)

MSI-X Message Control	MSI-X Next Item Pointer	MSI-X Capability ID	B0H
MSI-X Table Offset		Table BIR	B4H
MSI-X PBA Offset		PBA BIR	B8H

Note: MSI-X Capability Registers are defined in Chapter 4.0, "Messaging Unit."

The first byte at the Extended Configuration Offset B0H is the MSI-X Capability Identifier Register (Section 4.7.26, "MSI-X Capability Identifier Register - MSI-X_Cap_ID"). This identifies this Extended Configuration Header space as the type defined by the *PCI Local Bus Specification*, Revision 2.3.

Following the Capability Identifier Register is the single byte Next Item Pointer Register (Section 4.7.27, "MSI-X Next Item Pointer Register - MSI-X_Next_Item_Ptr") which indicates the configuration offset of an additional Extended Capabilities Header, when supported. In the ATU, the Next Item Pointer Register is set to A0H indicating that there is an additional Extended Capabilities Headers supported in the ATUs configuration space.

Figure 35. ATU Interface Extended Configuration Header Format (MSI Capability)

MSI Message Control	MSI Next Item Pointer	MSI Capability ID	A0H
MSI Message Address			A4H
MSI Message Upper Address			A8H
Reserved	MSI Message Data		ACH

The first byte at the Extended Configuration Offset D0H is the Section 4.7.20, "MSI Capability Identifier Register - Cap_ID" on page 429. This identifies this Extended Configuration Header space as the type defined by the *PCI Local Bus Specification*, Revision 2.3.

Following the Capability Identifier Register is the single byte Next Item Pointer Register which indicates the configuration offset of an additional Extended Capabilities Header, when supported. In the ATU, the Next Item Pointer Register is set to D0H indicating that there is an additional Extended Capabilities Headers supported in the ATUs configuration space.

Figure 36. ATU Interface Extended Configuration Header Format (PCI Express Capability)

PCI Express Capabilities Register	Next Item Pointer	PCI Express Cap ID	D0H
PCI Express Device Capabilities			D4H
PCI Express Device Status	PCI Express Device Control		D8H
PCI Express Link Capabilities			DCH
PCI Express Link Status	PCI Express Link Control		E0H
PCI Express Slot Capabilities			E4H
PCI Express Slot Status	PCI Express Slot Control		E8H
Reserved	PCI Express Root Control		ECH
PCI Express Root Status			

The first byte at the Extended Configuration Offset D0H is the PCI Express Capability Identifier Register (Section 3.17.55). This identifies this Extended Configuration Header space as the type defined by the *PCI Express Base Specification*, Revision 1.0a.

Following the Capability Identifier Register is the single byte Next Item Pointer Register (Section 3.17.56) which indicates the configuration offset of an additional Extended Capabilities Header, when supported. In the ATU, the Next Item Pointer Register is set to 00H by default to indicate there are no additional Extended Capabilities Headers in the ATUs configuration space. Software can set this pointer to 90H indicating that there is an additional Extended Capabilities Headers supported in the ATUs configuration space.

Figure 37. ATU Interface Extended Configuration Header Format (VPD Capability)

VPD Address	Next Item Pointer	VPD Capability ID	90H
VPD Data			94H

The first byte at the Extended Configuration Offset 90H is the VPD Capability Identifier Register (Section 3.17.44). This identifies this Extended Configuration Header space as the type defined by the *PCI Local Bus Specification*, Revision 2.3.

Following the Capability Identifier Register is the single byte Next Item Pointer Register (Section 3.17.47) which indicates the configuration offset of an additional Extended Capabilities Header, when supported. In the ATU, the Next Item Pointer Register is set to 00H indicating that there are no additional Extended Capabilities Headers supported in the ATUs configuration space.

The following sections describe the ATU and Expansion ROM configuration registers. Configuration space consists of 8, 16, 24, and 32-bit registers arranged in a predefined format. Each register is described in functionality, access type (read/write, read/clear, read only) and reset default condition.

See Section 1.6, "Terminology and Conventions" on page 48 for a description of *reserved*, *read only*, and *read/clear*. All registers adhere to the definitions found in the *PCI Local Bus Specification*, Revision 2.3 unless otherwise noted.

The PCI register number for each register is given in Table 141.

Note: Each configuration register's access type is individually defined for PCI configuration accesses. Some PCI read-only configuration registers have read/write capability from the 4138xx core CPU. See also Chapter 19.0, "Peripheral Registers".

3.17.2 Internal Bus Addresses

All of the ATU registers are accessible through both inbound PCI configuration cycles and the 4138xx core CPU (Register offsets 000H through 0FFH). T.

The location of these registers are specified as a relative offset to a 512KB aligned global PMMR offset. The default for the 512KB aligned offset is 0 FFD8 0000H defined by the PMMRBAR register. See also Chapter 19.0, “Peripheral Registers”.

The Internal Bus Address Offset to PMMRBAR of any ATU Register can be derived by adding the 4 KB address aligned Internal Bus Memory Mapped Register Range Offset (Table 140, “ATU Internal Bus Memory Mapped Register Range Offsets” on page 293) to the Register Offset (Table 141, “ATU PCI Configuration Register Space” on page 294)

For example, when INTERFACE_SEL_PCIX# and CONTROLLER_ONLY# are both asserted, the offset to PMMRBAR of the “ATU Command Register - ATUCMD” would be (4 D000H+004H) or 4 D004H.

Note: The 4 KB Address Aligned Range Offset can be different depending on two configuration straps as described in Table 140.

Table 140. ATU Internal Bus Memory Mapped Register Range Offsets

INTERFACE_SEL_PCIX#	CONTROLLER_ONLY#	Internal Bus MMR Address Range Offset (Relative to PMMRBAR)
Asserted (0)	Asserted (0)	+4 D000H
Asserted (0)	Deasserted (1)	+4 D000H
Deasserted (1)	Asserted (0)	+4 D000H
Deasserted (1)	Deasserted (1)	+4 8000H

Table 141. ATU PCI Configuration Register Space (Sheet 1 of 3)

Internal Bus Address Offset	ATU PCI Configuration Register Section, Name, Page
+000H	Section 3.17.3, "ATU Vendor ID Register - ATUVID" on page 297
+002H	Section 3.17.4, "ATU Device ID Register - ATUDID" on page 297
+004H	Section 3.17.5, "ATU Command Register - ATUCMD" on page 298
+006H	Section 3.17.6, "ATU Status Register - ATUSR" on page 299
+008H	Section 3.17.7, "ATU Revision ID Register - ATURID" on page 300
+009H	Section 3.17.8, "ATU Class Code Register - ATUCCR" on page 300
+00CH	Section 3.17.9, "ATU Cacheline Size Register - ATUCLSR" on page 301
+00DH	Section 3.17.10, "ATU Latency Timer Register - ATULT" on page 301
+00EH	Section 3.17.11, "ATU Header Type Register - ATUHTR" on page 302
+00FH	Section 3.17.12, "ATU BIST Register - ATUBISTR" on page 303
+010H	Section 3.17.13, "Inbound ATU Base Address Register 0 - IABAR0" on page 304
+014H	Section 3.17.14, "Inbound ATU Upper Base Address Register 0 - IAUBAR0" on page 305
+018H	Section 3.17.16, "Inbound ATU Base Address Register 1 - IABAR1" on page 308
+01CH	Section 3.17.17, "Inbound ATU Upper Base Address Register 1 - IAUBAR1" on page 309
+020H	Section 3.17.18, "Inbound ATU Base Address Register 2 - IABAR2" on page 310
+024H	Section 3.17.19, "Inbound ATU Upper Base Address Register 2 - IAUBAR2" on page 311
+02CH	Section 3.17.20, "ATU Subsystem Vendor ID Register - ASVIR" on page 312
+02EH	Section 3.17.21, "ATU Subsystem ID Register - ASIR" on page 312
+030H	Section 3.17.22, "Expansion ROM Base Address Register - ERBAR" on page 313
+034H	Section 3.17.23, "ATU Capabilities Pointer Register - ATU_Cap_Ptr" on page 314
+03CH	Section 3.17.24, "ATU Interrupt Line Register - ATUILR" on page 315
+03DH	Section 3.17.25, "ATU Interrupt Pin Register - ATUIPR" on page 316
+03EH	Section 3.17.26, "ATU Minimum Grant Register - ATUMGNT" on page 316
+03FH	Section 3.17.27, "ATU Maximum Latency Register - ATUMLAT" on page 317
+040H	Section 3.17.28, "Inbound ATU Limit Register 0 - IALR0" on page 318
+044H	Section 3.17.29, "Inbound ATU Translate Value Register 0 - IATVR0" on page 319
+048H	Section 3.17.30, "Inbound ATU Upper Translate Value Register 0 - IAUTVR0" on page 319
+04CH	Section 3.17.31, "Inbound ATU Limit Register 1 - IALR1" on page 320
+050H	Section 3.17.32, "Inbound ATU Translate Value Register 1 - IATVR1" on page 321
+054H	Section 3.17.33, "Inbound ATU Upper Translate Value Register 1 - IAUTVR1" on page 321
+058H	Section 3.17.34, "Inbound ATU Limit Register 2 - IALR2" on page 322
+05CH	Section 3.17.35, "Inbound ATU Translate Value Register 2 - IATVR2" on page 323
+060H	Section 3.17.36, "Inbound ATU Upper Translate Value Register 2 - IAUTVR2" on page 324
+064H	Section 3.17.37, "Expansion ROM Limit Register - ERLR" on page 324
+068H	Section 3.17.38, "Expansion ROM Translate Value Register - ERTVR" on page 325
+06CH	Section 3.17.39, "Expansion ROM Upper Translate Value Register - ERUTVR" on page 325
+070H	Section 3.17.40, "ATU Configuration Register - ATUCR" on page 326
+074H	Section 3.17.41, "PCI Configuration and Status Register - PCSR" on page 327
+078H	Section 3.17.42, "ATU Interrupt Status Register - ATUISR" on page 329
+07CH	Section 3.17.43, "ATU Interrupt Mask Register - ATUIMR" on page 332
+080H	Section 3.17.44, "PCI Express Message Control/Status Register - PEMCSR" on page 333
+084H	Section 3.17.45, "PCI Express Link Control/Status Register - PELCSR" on page 334
+090H	Section 3.17.46, "VPD Capability Identifier Register - VPD_Cap_ID" on page 335
+091H	Section 3.17.47, "VPD Next Item Pointer Register - VPD_Next_Item_Ptr" on page 335
+092H	Section 3.17.48, "VPD Address Register - VPDAR" on page 336

Table 141. ATU PCI Configuration Register Space (Sheet 2 of 3)

Internal Bus Address Offset	ATU PCI Configuration Register Section, Name, Page
+094H	Section 3.17.49, "VPD Data Register - VPDDR" on page 336
+098H	Section 3.17.50, "PM Capability Identifier Register - PM_Cap_ID" on page 337
+099H	Section 3.17.51, "PM Next Item Pointer Register - PM_Next_Item_Ptr" on page 337
+09AH	Section 3.17.52, "ATU Power Management Capabilities Register - APMCR" on page 338
+09CH	Section 3.17.53, "ATU Power Management Control/Status Register - APMCSR" on page 339
+0A0H	Section 4.7.20, "MSI Capability Identifier Register - Cap_ID" on page 429[a]
+0A1H	Section 4.7.21, "MSI Next Item Pointer Register - MSI_Next_Ptr" on page 430[a]
+0A2H	Section 4.7.22, "Message Control Register - Message_Control" on page 431[a]
+0A4H	Section 4.7.23, "Message Address Register - Message_Address" on page 432[a]
+0A8H	Section 4.7.24, "Message Upper Address Register - Message_Upper_Address" on page 433[a]
+0ACH	Section 4.7.25, "Message Data Register- Message_Data" on page 434[a]
+0B0H	Section 4.7.26, "MSI-X Capability Identifier Register - MSI-X_Cap_ID" on page 435[b]
+0B1H	Section 4.7.27, "MSI-X Next Item Pointer Register - MSI-X_Next_Item_Ptr" on page 436[b]
+0B2H	Section 4.7.28, "MSI-X Message Control Register - MSI-X_MCR" on page 437[b]
+0B4H	Section 4.7.29, "MSI-X Table Offset Register — MSI-X_Table_Offset" on page 438[b]
+0B8H	Section 4.7.30, "MSI-X Pending Bit Array Offset Register - MSI-X_PBA_Offset" on page 439[b]
+0BCJ- +0C8H	Reserved
+0CCH	Section 3.17.54, "ATU Scratch Pad Register - ATUSPR" on page 340
+0D0H	Section 3.17.55, "PCI Express Capability List Register - PCIE_CAPID" on page 340
+0D1H	Section 3.17.56, "PCI Express Next Item Pointer Register - PCIE_NXTP" on page 341
+0D2H	Section 3.17.57, "PCI Express Capabilities Register - PCIE_CAP" on page 342
+0D4H	Section 3.17.58, "PCI Express Device Capabilities Register - PCIE_DCAP" on page 343
+0D8H	Section 3.17.59, "PCI Express Device Control Register - PE_DCTL" on page 344
+0DAH	Section 3.17.60, "PCI Express Device Status Register - PE_DSTS" on page 346
+0DCH	Section 3.17.61, "PCI Express Link Capabilities Register - PE_LCAP" on page 347
+0E0H	Section 3.17.62, "PCI Express Link Control Register - PE_LCTL" on page 348
+0E2H	Section 3.17.63, "PCI Express Link Status Register - PE_LSTS" on page 349
+0E4H	Section 3.17.64, "PCI Express Slot Capabilities Register - PE_SCAP" on page 350
+0E8H	Section 3.17.65, "PCI Express Slot Control Register - PE_SCR" on page 351
+0EAH	Section 3.17.66, "PCI Express Slot Status Register - PE_SSTS" on page 352
+0ECH	Section 3.17.67, "PCI Express Root Control Register - PE_RCR" on page 353
+0F0H	Section 3.17.68, "PCI Express Root Status Register - PE_RSR" on page 354
+100H	Section 3.17.69, "PCI Express Advanced Error Capability Identifier - ADVERR_CAPID" on page 354
+104H	Section 3.17.70, "PCI Express Uncorrectable Error Status - ERRUNC_STS" on page 355
+108H	Section 3.17.71, "PCI Express Uncorrectable Error Mask - ERRUNC_MSK" on page 356
+10CH	Section 3.17.72, "PCI Express Uncorrectable Error Severity - ERRUNC_SEV" on page 357
+110H	Section 3.17.73, "PCI Express Correctable Error Status - ERRCOR_STS" on page 358
+114H	Section 3.17.74, "PCI Express Correctable Error Mask - ERRCOR_MSK" on page 359
+118H	Section 3.17.75, "Advanced Error Control and Capability Register - ADVERR_CTL" on page 360
+11CH	Section 3.17.76, "PCI Express Advanced Error Header Log - ADVERR_LOG0" on page 360
+120H	Section 3.17.77, "PCI Express Advanced Error Header Log - ADVERR_LOG1" on page 361
+124H	Section 3.17.78, "PCI Express Advanced Error Header Log - ADVERR_LOG2" on page 361
+128H	Section 3.17.79, "PCI Express Advanced Error Header Log - ADVERR_LOG3" on page 362
+12CH	Section 3.17.80, "Root Error Command Register - RERR_CMD" on page 362
+130H	Section 3.17.81, "Root Error Status Register" on page 363

Table 141. ATU PCI Configuration Register Space (Sheet 3 of 3)

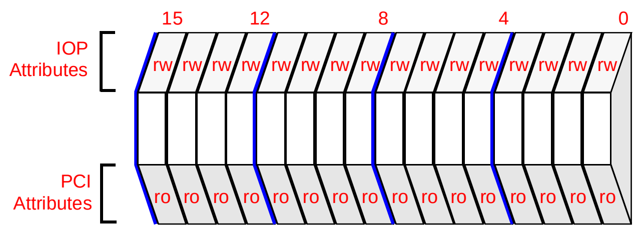
Internal Bus Address Offset	ATU PCI Configuration Register Section, Name, Page
+134H	Section 3.17.82, "Error Source Identification Register - RERR_ID" on page 364
+1E0H	Section 3.17.83, "Device Serial Number Capability - DSN_CAP" on page 364
+1E4H	Section 3.17.84, "Device Serial Number Lower DW Register - DSN_LDW" on page 365
+1E8H	Section 3.17.85, "Device Serial Number Upper DW Register - DSN_UDW" on page 365
+1ECH	Section 3.17.86, "PCI Express Advisory Error Control Register - PIE_AEC" on page 366
+1F0H	Section 3.17.87, "Power Budgeting Enhanced Capability Header - PWRBGT_CAPID" on page 367
+1F4H	Section 3.17.88, "Power Budgeting Data Select Register - PWRBGT_DSEL" on page 367
+1F8H	Section 3.17.89, "Power Budgeting Data Register - PWRBGT_DATA" on page 368
+1FCH	Section 3.17.90, "Power Budgeting Capability Register - PWRBGT_CAP" on page 369
+200H - +25FH	Section 3.17.91, "Power Budgeting Information Registers[0:23]—PWRBGT_INFO[0:23]" on page 370
+300H	Section 3.17.92, "Outbound I/O Base Address Register - OIOBAR" on page 371
+304H	Section 3.17.93, "Outbound I/O Window Translate Value Register - OIOWTVR" on page 372
+308H	Section 3.17.94, "Outbound Upper Memory Window Base Address Register 0 - OUMBAR0" on page 373
+30CH	Section 3.17.95, "Outbound Upper 32-bit Memory Window Translate Value Register 0 - OUMWTVR0" on page 374
+310H	Section 3.17.96, "Outbound Upper Memory Window Base Address Register 1 - OUMBAR1" on page 375
+314H	Section 3.17.97, "Outbound Upper 32-bit Memory Window Translate Value Register 1 - OUMWTVR1" on page 376
+318H	Section 3.17.98, "Outbound Upper Memory Window Base Address Register 2 - OUMBAR2" on page 377
+31CH	Section 3.17.99, "Outbound Upper 32-bit Memory Window Translate Value Register 2 - OUMWTVR2" on page 378
+320H	Section 3.17.100, "Outbound Upper Memory Window Base Address Register 3 - OUMBAR3" on page 379
+324H	Section 3.17.101, "Outbound Upper 32-bit Memory Window Translate Value Register 3 - OUMWTVR3" on page 380
+328H	Reserved
+32CH	Section 3.17.102, "Outbound Configuration Cycle Address Register - OCCAR" on page 381
+330H	Section 3.17.103, "Outbound Configuration Cycle Data Register - OCCDR" on page 382
+334H	Section 3.17.104, "Outbound Configuration Cycle Function Number - OCCFN" on page 383
+340H	Section 3.17.105, "Inbound Vendor Message Header Register 0 - IVMHR0" on page 384
+344H	Section 3.17.106, "Inbound Vendor Message Header Register 1 - IVMHR1" on page 385
+348H	Section 3.17.107, "Inbound Vendor Message Header Register 2 - IVMHR2" on page 386
+34CH	Section 3.17.108, "Inbound Vendor Message Header Register 3 - IVMHR3" on page 387
+350H	Section 3.17.109, "Inbound Vendor Message Payload Register - IVMPR" on page 387
+360H	Section 3.17.110, "Outbound Vendor Message Header Register 0 - OVMHR0" on page 388
+364H	Section 3.17.111, "Outbound Vendor Message Header Register 1 - OVMHR1" on page 389
+368H	Section 3.17.112, "Outbound Vendor Message Header Register 2 - OVMHR2" on page 390
+36CH	Section 3.17.113, "Outbound Vendor Message Header Register 3 - OVMHR3" on page 390
+370H	Section 3.17.114, "Outbound Vendor Message Payload Register - OVMPR" on page 391
+380H	Section 3.17.115, "PCI Interface Error Control and Status Register - PIE_CSR" on page 392
+384H	Section 3.17.116, "PCI Interface Error Status - PIE_STS" on page 393
+388H	Section 3.17.117, "PCI Interface Error Mask - PIE_MSK" on page 394
+38CH	Section 3.17.118, "PCI Interface Error Header Log - PIE_LOG0" on page 395
+390H	Section 3.17.119, "PCI Interface Error Header Log 1 - PIE_LOG1" on page 395
+394H	Section 3.17.120, "PCI Interface Error Header Log 2 - PIE_LOG2" on page 396
+398H	Section 3.17.121, "PCI Interface Error Header Log - PIE_LOG3" on page 396
+39CH	Section 3.17.122, "PCI Interface Error Descriptor Log" on page 397
+3B0H	Section 3.17.123, "ATU Reset Control Register - ATURCR" on page 397

a. Refer to the Messaging Unit Chapter for MSI Register Definitions
b. Refer to the Messaging Unit Chapter for MSI-X Register Definitions.

3.17.3 ATU Vendor ID Register - ATUVID

ATU Vendor ID Register bits adhere to the definitions in the *PCI Local Bus Specification*, Revision 2.3.

Table 142. ATU Vendor ID Register - ATUVID



Internal Bus Address Offset
+000H

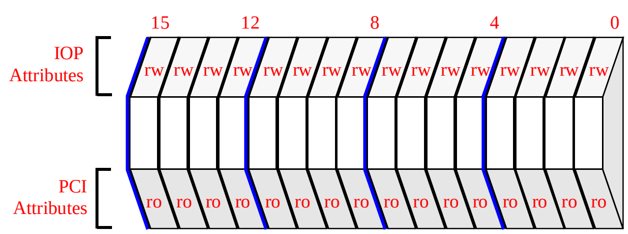
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
15:00	8086H	ATU Vendor ID - This is a 16-bit value assigned to Intel. This register, combined with the DID, uniquely identify the PCI device. Access type is Read/Write to allow the 4138xx to configure the register as a different vendor ID to simulate the interface of a standard mechanism currently used by existing application software.

3.17.4 ATU Device ID Register - ATUDID

ATU Device ID Register bits adhere to the definitions in the *PCI Local Bus Specification*, Revision 2.3.

Table 143. ATU Device ID Register - ATUDID



Internal Bus Address Offset
+002H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

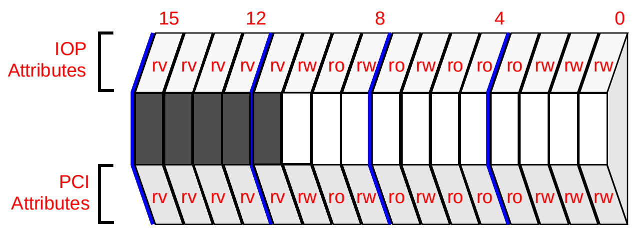
Bit	Default	Description
15:00	Product Type Dependent[a]	ATU Device ID - This is a 16-bit value assigned to the ATU. This ID, combined with the VID, uniquely identify any PCI device.

a. See *Intel® 413808 and 413812 I/O Controllers in TPER Mode Specification Update.*

3.17.5 ATU Command Register - ATUCMD

ATU Command Register bits adhere to the definitions in the *PCI Local Bus Specification*, Revision 2.3 and in most cases, affect the behavior of the PCI ATU and devices on the PCI Express Link.

Table 144. ATU Command Register - ATUCMD



Internal Bus Address Offset
+004H

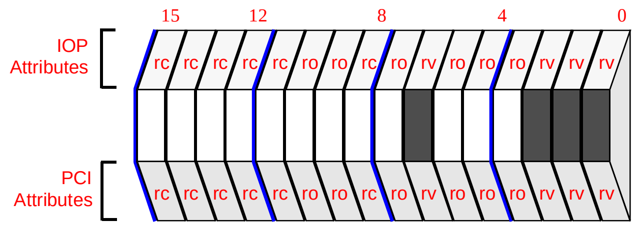
Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
15:11	00000_2	Reserved
10	0_2	Interrupt Disable - Controls the ability of the ATU to generate INTx interrupt messages. When set, the ATU is prevented from generating INTx interrupt messages and generates a Deassert_INTx message for any emulation interrupts already asserted.
09	0_2	Fast Back to Back Enable - Does not apply to PCI Express. Hard-wired to 0.
08	0_2	SERR# Enable - When set, the ATU is allowed to report non-fatal and fatal error detected by 4138xx to the Root Complex. ***Note:*** Errors are reported either through this bit or through the PCI-Express specific bits in the PCI Express Device Control Register - PE_DCTL.
07	0_2	Address/Data Stepping Control - Does not apply to PCI Express. Hard-wired to 0.
06	0_2	Parity Error Response - When set, the ATU takes normal action in response to a poisoned TLP received from PCI Express. When cleared, parity checking is disabled. ***Note:*** When the bit is cleared but the Poisoned TLP Mask is cleared in the PCI Express Uncorrectable Error Mask - ERRUNC_MSK register, the ATU still logs the error in the Advanced Error Reporting registers and generate an Uncorrectable Error message.
05	0_2	VGA Palette Snoop Enable - Does not apply to PCI Express. Hard-wired to 0.
04	0_2	Memory Write and Invalidate Enable - Does not apply to PCI Express. Hard-wired to 0.
03	0_2	Special Cycle Enable - Does not apply to PCI Express. Hard-wired to 0.
02	0_2	Bus Master Enable - When cleared, the ATU is prevented from issuing any memory or I/O read/write requests. Requests other than memory or I/O requests are not controlled by this bit. The ATU initiates a completion transaction regardless of the setting.
01	0_2	Memory Enable - Controls the ATU interface's response to memory transactions. When cleared, the ATU drops the transaction and returns an unsupported request
00	0_2	I/O Space Enable - Controls the ATU interface response to I/O transactions. When cleared, the ATU drops the transaction and returns an unsupported request.

3.17.6 ATU Status Register - ATUSR

The ATU Status Register bits adhere to the *PCI Local Bus Specification*, Revision 2.3 definitions. The *read/clear* bits can only be set by internal hardware and cleared by either a reset condition or by writing a 1_2 to the register.

Table 145. ATU Status Register - ATUSR



Internal Bus Address Offset
+006H

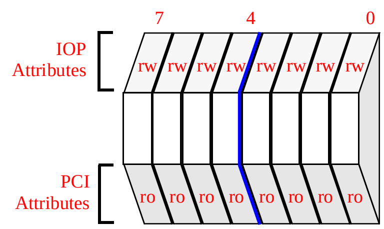
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
15	0_2	Detected Parity Error - set when the ATU receives a poisoned TLP regardless of the state of the Parity Error Response in the ATUCMD register.
14	0_2	SERR# Asserted - set when the ATU sends an ERR_FATAL or ERR_NONFATAL message, and the SERR Enable bit in the ATUCMD register is ‘1’.
13	0_2	Received Master Abort - set when the ATU receives a completion with Unsupported Request Completion Status.
12	0_2	Received Target Abort - set when the ATU receives a completion with Completer Abort Completion Status.
11	0_2	Signaled Target Abort - set when the ATU completes a Request using Completer Abort Completion Status
10:09	00_2	DEVSEL# Timing - Does not apply to PCI Express. Hard-wired to 0.
08	0_2	Master Data Parity Error - This bit is set by the ATU when its Parity Error Enable bit is set and either of the following two conditions occurs: This bit is set under the following conditions. • ATU receives a Poisoned Completion for an Outbound Read Request • ATU transmits a Poisoned TLP for an Outbound Write Request. When the Parity Error Response bit is cleared in the “ATU Command Register - ATUCMD”, this bit is never set.
07	0_2	Fast Back-to-Back - Does not apply to PCI Express. Hard-wired to 0.
06	0_2	Reserved
05	0_2	66 MHz. Capable - Does not apply to PCI Express. Hard-wired to 0
04	1_2	Capabilities List - All PCI Express devices are required to implement the PCI Express capability structure. Hard-wired to 1.
03	0_2	Interrupt Status - Indicates that an INTx interrupt message is pending internally to the device. ***Note:*** Setting the Interrupt Disable bit to a 1 in (bit 10 of ATUCMD) has no effect on the state of this bit.
02:00	000_2	Reserved

3.17.7 ATU Revision ID Register - ATURID

Revision ID Register bit definitions adhere to *PCI Local Bus Specification*, Revision 2.3.

Table 146. ATU Revision ID Register - ATURID



Internal Bus Address Offset
+008H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

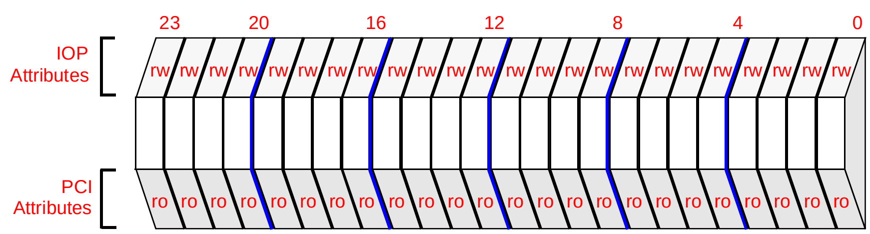
Bit	Default	Description
07:00	xxH[a]	ATU Revision - identifies the 4138xx revision number.

a. See *Intel® 81348 I/O Processor Specification Update.*

3.17.8 ATU Class Code Register - ATUCCR

Class Code Register bit definitions adhere to *PCI Local Bus Specification*, Revision 2.3. Auto configuration software reads this register to determine the PCI device function.

Table 147. ATU Class Code Register - ATUCCR



Internal Bus Address Offset
+009H

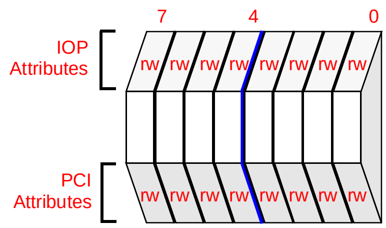
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
23:16	05H	Base Class - Memory Controller
15:08	80H	Sub Class - Other Memory Controller
07:00	00H	Programming Interface - None defined

3.17.9 ATU Cacheline Size Register - ATUCLSR

Cacheline Size Register bit definitions adhere to *PCI Local Bus Specification*, Revision 2.3. This register is programmed with the system cacheline size in DWORDs (32-bit words). Cacheline Size is restricted to either 0, 8 or 16 DWORDs; the ATU interprets any other value as "0".

Table 148. ATU Cacheline Size Register - ATUCLSR



Internal Bus Address Offset
+00CH

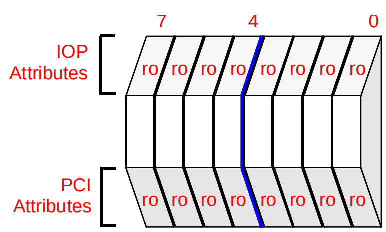
Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
07:00	00H	ATU Cacheline Size - specifies the system cacheline size in DWORDs. ***Note:*** This field is read-write for legacy compatibility purposes but has no impact on any PCI Express device functionality.

3.17.10 ATU Latency Timer Register - ATULT

ATU Latency Timer Register does not apply to PCI Express.

Table 149. ATU Latency Timer Register - ATULT



Internal Bus Address Offset
+00DH

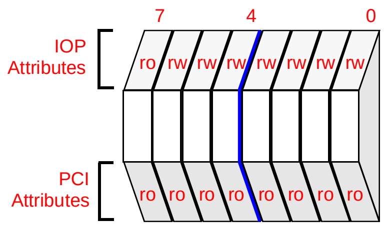
Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
07:00	00H	Programmable Latency Timer - The latency timer does not apply to PCI Express. Hard-wired 0.

3.17.11 ATU Header Type Register - ATUHTR

Header Type Register bit definitions adhere to *PCI Local Bus Specification*, Revision 2.3. This register indicates the layout of ATU configuration space bytes 10H to 3FH. The MSB indicates whether or not the device is multi-function.

Table 150. ATU Header Type Register - ATUHTR



Internal Bus Address Offset
+00EH

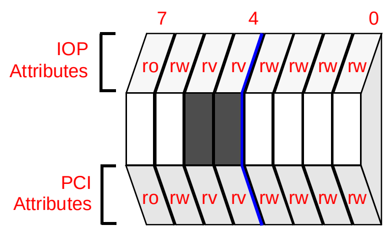
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
07	End Point: **DF_SEL[2:0]** != "000" Root Complex: 0	Single Function/Multi-Function Device - Identifies the 4138xx as a single-function or multi-function PCI device depending on the setting of the **DF_SEL[2:0]** strap during **P_RST#** assertion. ***Note:*** The 4138xx can be configured as a single-function device (ATU only) or a multi-function device (ATU and storage controller) for split driver support. As a Root Complex this bit is always '0'.
06:00	000000_2	PCI Header Type - This bit field indicates the type of PCI header implemented. The ATU interface header conforms to *PCI Local Bus Specification*, Revision 2.3.

3.17.12 ATU BIST Register - ATUBISTR

The ATU BIST Register controls the functions the Intel XScale® processor performs when BIST is initiated. This register is the interface between the host processor requesting BIST functions and the 4138xx replying with the results from the software implementation of the BIST functionality.

Table 151. ATU BIST Register - ATUBISTR



Internal Bus Address Offset
+00FH

Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
07	0_2	BIST Capable - This bit value is always equal to the ATUCR ATU BIST Interrupt Enable bit. See Section 3.17.40, "ATU Configuration Register - ATUCR" on page 326.
06	0_2	Start BIST - When the ATUCR BIST Interrupt Enable bit is set: Setting this bit generates an interrupt to the Intel XScale® processor to perform a software BIST function. The Intel XScale® processor clears this bit when the BIST software has completed with the BIST results found in ATUBISTR register bits [3:0]. When the ATUCR BIST Interrupt Enable bit is clear: Setting this bit does not generate an interrupt to the Intel XScale® processor and no BIST functions is performed. The Intel XScale® processor does not clear this bit.
05:04	00_2	Reserved
03:00	0000_2	BIST Completion Code - when the ATUCR BIST Interrupt Enable bit is set and the ATUBISTR Start BIST bit is set (bit 6): The Intel XScale® processor places the results of the software BIST in these bits. A nonzero value indicates a device-specific error.

3.17.13 Inbound ATU Base Address Register 0 - IABAR0

The Inbound ATU Base Address Register 0 (IABAR0) together with the Inbound ATU Upper Base Address Register 0 (IAUBAR0) defines the block of memory addresses where the inbound translation window 0 begins. The inbound ATU decodes and forwards the bus request to the 4138xx internal bus with a translated address to map into 4138xx local memory. The IABAR0 and IAUBAR0 define the base address and describes the required memory block size; see Section 3.17.15, “Determining Block Sizes for Base Address Registers” on page 306. Bits 31 through 12 of the IABAR0 is either read/write bits or read only with a value of 0 depending on the value located within the IALR0. This configuration allows the IABAR0 to be programmed per *PCI Local Bus Specification*, Revision 2.3.

The programmed value within the base address register must comply with the PCI programming requirements for address alignment. Refer to the *PCI Local Bus Specification*, Revision 2.3 for additional information on programming base address registers.

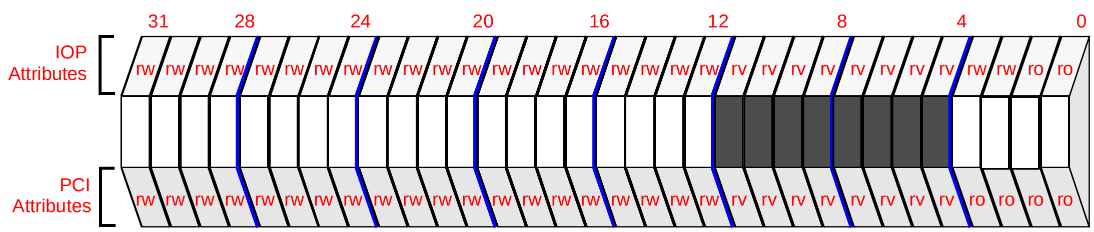
By default the first 8 Kbytes of memory defined by the IABAR0, IAUBAR0 and the IALR0 is reserved for the Messaging Unit.

Warning: When IALR0 is cleared prior to host configuration, the user should also clear the Prefetchable Indicator and the Type Indicator. Assuming IALR0 is not cleared:

a. Since non prefetchable memory windows can never be placed above the 4 Gbyte address boundary, when the Prefetchable Indicator is cleared prior to host configuration, the user should also set the Type Indicator for 32 bit addressability.

b. For compliance to the *PCI-X Protocol Addendum to the PCI Local Bus Specification,* Revision 2.0, when the Prefetchable Indicator is set prior to host configuration, the user should also set the Type Indicator for 64 bit addressability. This is the default for IABAR0.

Table 152. Inbound ATU Base Address Register 0 - IABAR0



Internal Bus Address Offset
+010H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:12	00000H	Translation Base Address 0 - These bits define the actual location the translation function is to respond to when addressed from the PCI Express Link.
11:04	00H	Reserved.
03	1_2	Prefetchable Indicator - When set, defines the memory space as prefetchable.
02:01	10_2	Type Indicator - Defines the width of the addressability for this memory window: 00 - Memory Window is locatable anywhere in 32 bit address space 10 - Memory Window is locatable anywhere in 64 bit address space
00	0_2	Memory Space Indicator - This bit field describes memory or I/O space base address. The ATU does not occupy I/O space, thus this bit must be zero.

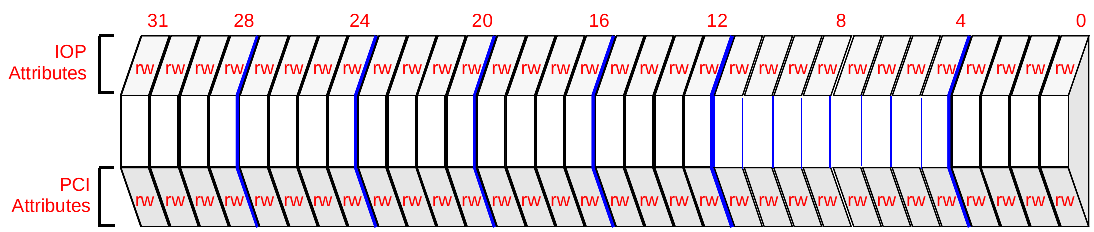
3.17.14 Inbound ATU Upper Base Address Register 0 - IAUBAR0

This register contains the upper base address when decoding PCI addresses beyond 4 GBytes. Together with the Translation Base Address this register defines the actual location the translation function is to respond to when addressed from the PCI Express Link for addresses > 4 GBytes (for DACs).

The programmed value within the base address register must comply with the PCI programming requirements for address alignment. Refer to the *PCI Local Bus Specification*, Revision 2.3 for additional information on programming base address registers.

Note: When the Type Indicator of IABAR0 is set to indicate 32 bit addressability, the IAUBAR0 register attributes are read-only. Prior to changing the Type Indicator in the IABAR0 to support 32-bit addressability, the IAUBAR0 must be written with zero unless it already contains zero. Zero is the default value for the IAUBAR0.

Table 153. Inbound ATU Upper Base Address Register 0 - IAUBAR0



Internal Bus Address Offset
+014H

Attribute Legend:	RW = Read/Write
RV = Reserved	RC = Read Clear
PR = Preserved	RO = Read Only
RS = Read/Set	NA = Not Accessible

Bit	Default	Description
31:0	00000H	Translation Upper Base Address 0 - Together with the Translation Base Address 0 these bits define the actual location the translation function is to respond to when addressed from the PCI Express Link for addresses > 4GBytes.

3.17.15 Determining Block Sizes for Base Address Registers

The required address size and type can be determined by writing ones to a base address register and reading from the registers. By scanning the returned value from the least-significant bit of the base address registers upwards, the programmer can determine the required address space size. The binary-weighted value of the first non-zero bit found indicates the required amount of space. Table 154 describes the relationship between the values read back and the byte sizes the base address register requires.

Table 154. Memory Block Size Read Response

Response After Writing all 1s to the Base Address Register	Size (in Bytes)	Response After Writing all 1s to the Base Address Register	Size (in Bytes)
FFFFFFF0H	16	FFF00000H	1 M
FFFFFFE0H	32	FFE00000H	2 M
FFFFFFC0H	64	FFC00000H	4 M
FFFFFF80H	128	FF800000H	8 M
FFFFFF00H	256	FF000000H	16 M
FFFFFE00H	512	FE000000H	32 M
FFFFFC00H	1K	FC000000H	64 M
FFFFF800H	2K	F8000000H	128 M
FFFFF000H	4K	F0000000H	256 M
FFFFE000H	8K	E0000000H	512 M
FFFFC000H	16K	C0000000H	1 G
FFFF8000H	32K	80000000H	2 G
FFFF0000H	64K	00000000H	Register not implemented, no address space required.
FFFE0000H	128K		
FFFC0000H	256K		
FFF80000H	512K		

As an example, assume that FFFF.FFFFH is written to the Inbound ATU Base Address Register 0 - IABAR0 and the value read back is FFF0.0008H. Bit zero is a zero, so the device requires memory address space. Bit three is one, so the memory does supports prefetching. Scanning upwards starting at bit four, bit twenty is the first one bit found. The binary-weighted value of this bit is 1,048,576, indicated that the device requires 1 Mbyte of memory space.

The ATU Base Address Registers and the Expansion ROM Base Address Register use their associated limit registers to enable which bits within the base address register are read/write and which bits are read only (0). This allows the programming of these registers in a manner similar to other PCI devices even though the limit is variable.

Table 155. ATU Base Registers and Associated Limit Registers

Base Address Register	Limit Register	Description
Inbound ATU Base Address Register 0	Inbound ATU Limit Register 0	Defines the inbound translation window 0 from the PCI Express Link.
Inbound ATU Upper Base Address Register 0	N/A	Together with ATU Base Address Register 0 defines the inbound translation window 0 from the PCI Express Link
Inbound ATU Base Address Register 1	Inbound ATU Limit Register 1	Defines the inbound translation window 1 from the PCI Express Link.
Inbound ATU Upper Base Address Register 1	N/A	Together with ATU Base Address Register 1 defines the inbound translation window 1 from the PCI Express Link
Inbound ATU Base Address Register 2	Inbound ATU Limit Register 2	Defines the inbound translation window 2 from the PCI Express Link.
Inbound ATU Upper Base Address Register 2	N/A	Together with ATU Base Address Register 2 defines the inbound translation window 2 from the PCI Express Link
Expansion ROM Base Address Register	Expansion ROM Limit Register	Defines the window of addresses used by a bus master for reading from an Expansion ROM.

3.17.16 Inbound ATU Base Address Register 1 - IABAR1

The Inbound ATU Base Address Register 1 (IABAR1) together with the Inbound ATU Upper Base Address Register 1 (IAUBAR1) defines the block of memory addresses where the inbound translation window 1 begins. The inbound ATU decodes and forwards the bus request to the 4138xx internal bus with a translated address to map into 4138xx local memory. The IABAR1 and IAUBAR1 define the base address and describes the required memory block size; see Section 3.17.15, "Determining Block Sizes for Base Address Registers" on page 306. Bits 31 through 12 of the IABAR1 is either read/write bits or read only with a value of 0 depending on the value located within the IALR1. This configuration allows the IABAR1 to be programmed per *PCI Local Bus Specification*, Revision 2.3.

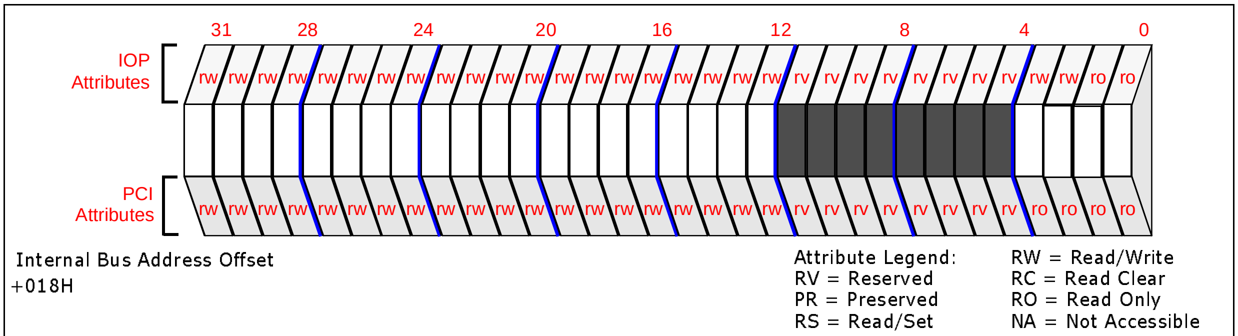
The programmed value within the base address register must comply with the PCI programming requirements for address alignment. Refer to the *PCI Local Bus Specification*, Revision 2.3 for additional information on programming base address registers.

Warning: When IALR1 is cleared prior to host configuration, the user should also clear the Prefetchable Indicator and the Type Indicator. Assuming IALR1 is not cleared:

a. Since non prefetchable memory windows can never be placed above the 4 Gbyte address boundary, when the Prefetchable Indicator is cleared prior to host configuration, the user should also set the Type Indicator for 32 bit addressability.

b. For compliance to the *PCI-X Protocol Addendum to the PCI Local Bus Specification,* Revision 2.0, when the Prefetchable Indicator is set prior to host configuration, the user should also set the Type Indicator for 64 bit addressability. This is the default for IABAR1.

Table 156. Inbound ATU Base Address Register 1 - IABAR1



Bit	Default	Description
31:12	00000H	Translation Base Address 1 - These bits define the actual location of window 1 on the PCI Express Link.
11:04	00H	Reserved.
03	0_2	Prefetchable Indicator - When set, defines the memory space as prefetchable.
02:01	00_2	Type Indicator - Defines the width of the addressability for this memory window: 00 - Memory Window is locatable anywhere in 32 bit address space 10 - Memory Window is locatable anywhere in 64 bit address space
00	0_2	Memory Space Indicator - This bit field describes memory or I/O space base address. The ATU does not occupy I/O space, thus this bit must be zero.

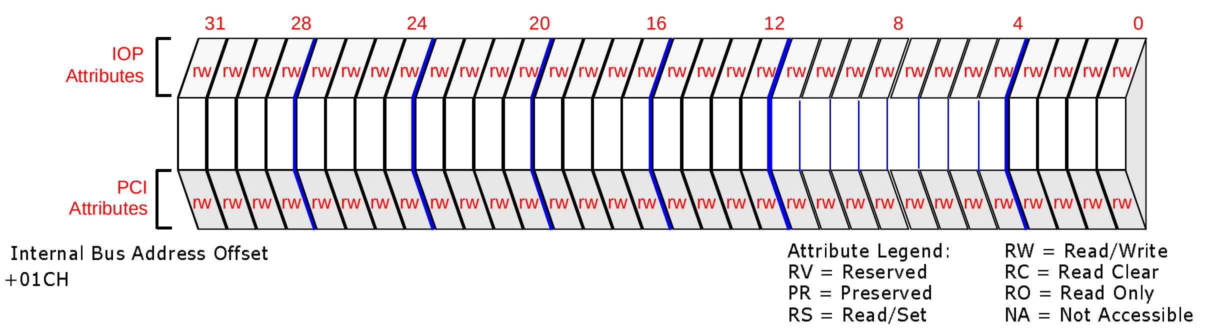
3.17.17 Inbound ATU Upper Base Address Register 1 - IAUBAR1

This register contains the upper base address when decoding PCI addresses beyond 4 GBytes. Together with the Translation Base Address this register defines the actual location the translation function is to respond to when addressed from the PCI Express Link for addresses > 4GBytes (for DACs).

The programmed value within the base address register must comply with the PCI programming requirements for address alignment. Refer to the *PCI Local Bus Specification*, Revision 2.3 for additional information on programming base address registers.

Note: When the Type indicator of IABAR1 is set to indicate 32 bit addressability, the IAUBAR1 register attributes are read-only. By default the IAUBAR1 register has read-only attributes. Prior to changing the Type Indicator in the IABAR1 to support 32-bit addressability, the IAUBAR1 must be written with zero unless it already contains zero. Zero is the default value for IAUBAR1.

Table 157. Inbound ATU Upper Base Address Register 1 - IAUBAR1



Bit	Default	Description
31:0	00000H	Translation Upper Base Address 1 - Together with the Translation Base Address 1 these bits define the actual location for this memory window on the PCI Express Link for addresses > 4GBytes.

3.17.18 Inbound ATU Base Address Register 2 - IABAR2

The Inbound ATU Base Address Register 2 (IABAR2) together with the Inbound ATU Upper Base Address Register 2 (IAUBAR2) defines the block of memory space or I/O space addresses where the inbound translation window 2 begins. The inbound ATU decodes and forwards the bus request to the 4138xx internal bus with a translated address to map into 4138xx local memory. The IABAR2 and IAUBAR2 (Memory Space only) define the base address and describes the required address block size; see Section 3.17.15, "Determining Block Sizes for Base Address Registers" on page 306. Bits 31 through 8 of the IABAR2 is either read/write bits or read only with a value of 0 depending on the value located within the IALR2. This configuration allows the IABAR2 to be programmed per *PCI Local Bus Specification*, Revision 2.3.

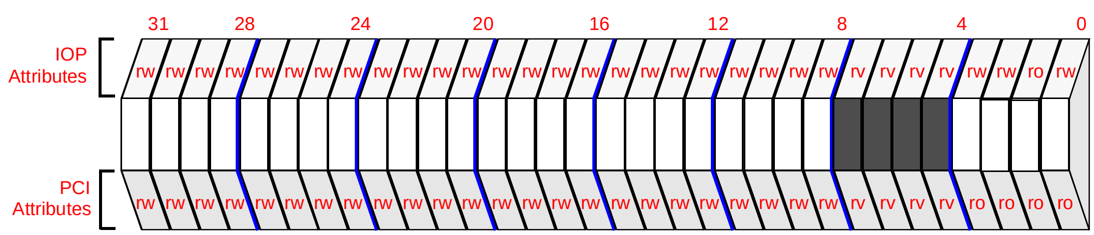
The programmed value within the base address register must comply with the PCI programming requirements for address alignment. Refer to the *PCI Local Bus Specification*, Revision 2.3 for additional information on programming base address registers.

Warning: When IALR2 is cleared prior to host configuration, the user should also clear the Prefetchable Indicator and the Type Indicator. Assuming IALR2 is not cleared:

a. Since non prefetchable memory windows can never be placed above the 4 Gbyte address boundary, when the Prefetchable Indicator is cleared prior to host configuration, the user should also set the Type Indicator for 32 bit addressability.

b. For compliance to the *PCI-X Protocol Addendum to the PCI Local Bus Specification,* Revision 2.0, when the Prefetchable Indicator is set prior to host configuration, the user should also set the Type Indicator for 64 bit addressability. This is the default for IABAR0.

Table 158. Inbound ATU Base Address Register 2 - IABAR2



Internal Bus Address Offset
+020H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:8	000000H	Translation Base Address 2 - These bits define the actual location the translation function is to respond to when addressed from the PCI Express Link.
07:04	00H	Reserved.
03	0_2	Prefetchable Indicator - When set, defines the memory space as prefetchable.
02:01	00_2	Type Indicator - Defines the width of the addressability for this memory window 00 - Memory Window is locatable anywhere in 32 bit address space 10 - Memory Window is locatable anywhere in 64 bit address space
00	0_2	Memory Space Indicator - This bit field describes memory or I/O space base address.

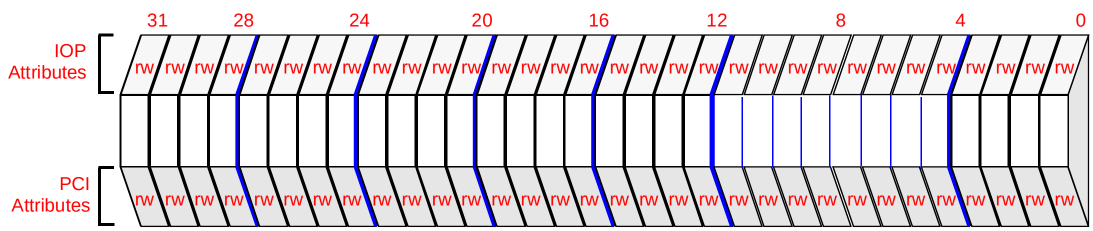
3.17.19 Inbound ATU Upper Base Address Register 2 - IAUBAR2

This register contains the upper base address when decoding PCI addresses for memory space (Memory Space Indicator in IABAR2 is clear) beyond 4 GBytes. Together with the Translation Base Address this register defines the actual location the translation function is to respond to when addressed from the PCI Express Link for addresses > 4 GBytes (for DACs).

The programmed value within the base address register must comply with the PCI programming requirements for address alignment. Refer to the *PCI Local Bus Specification*, Revision 2.3 for additional information on programming base address registers.

Note: When the Type indicator of IABAR2 is set to indicate 32 bit addressability or the Memory Space indicator of IABAR2 is set indicating I/O space, the IAUBAR2 register attributes are read-only. By default the IAUBAR2 register has read-only attributes. Prior to changing the Type/Memory Space Indicator in the IABAR2 to support 32-bit addressability, the IAUBAR2 must be written with zero unless it already contains zero. Zero is the default value for IAUBAR2.

Table 159. Inbound ATU Upper Base Address Register 2 - IAUBAR2



Internal Bus Address Offset
+024H

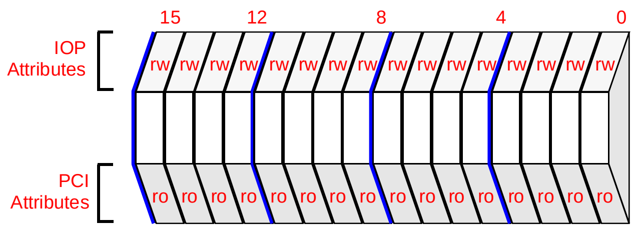
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:0	00000H	Translation Upper Base Address 2 - Together with the Translation Base Address 2 these bits define the actual location the translation function is to respond to when addressed from the PCI Express Link for addresses > 4GBytes.

3.17.20 ATU Subsystem Vendor ID Register - ASVIR

ATU Subsystem Vendor ID Register bit definitions adhere to *PCI Local Bus Specification*, Revision 2.3.

Table 160. ATU Subsystem Vendor ID Register - ASVIR



Internal Bus Address Offset
+02CH

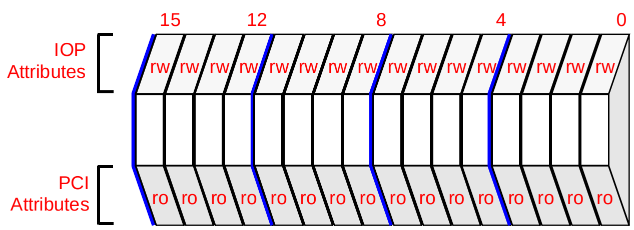
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
15:0	0000H	Subsystem Vendor ID - This register uniquely identifies the add-in board or subsystem vendor.

3.17.21 ATU Subsystem ID Register - ASIR

ATU Subsystem ID Register bit definitions adhere to *PCI Local Bus Specification*, Revision 2.3.

Table 161. ATU Subsystem ID Register - ASIR



Internal Bus Address Offset
+02EH

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
15:0	0000H	Subsystem ID - uniquely identifies the add-in board or subsystem.

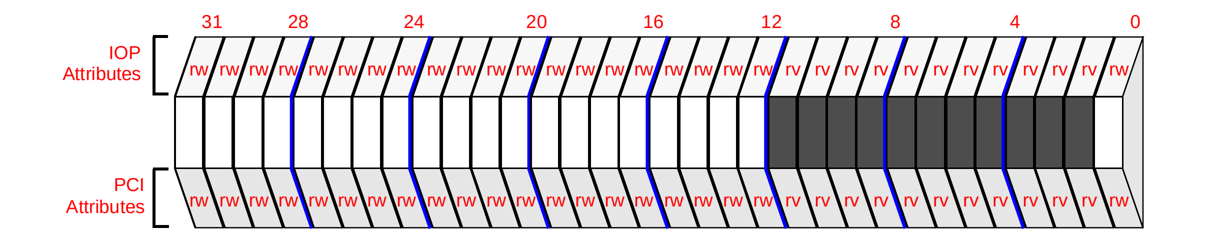
3.17.22 Expansion ROM Base Address Register - ERBAR

The Expansion ROM Base Address Register defines the block of memory addresses used for containing the Expansion ROM. It permits the inclusion of multiple code images, allowing the device to be initialized. The code image supplied consists of either executable code or an interpreted code. Each code image must start on a 512 byte boundary and each must contain the PCI Expansion ROM header. Image placement in ROM space depends on the length of code images which precede it within ROM. ERBAR defines the base address and describes the required memory block size; see Section 3.17.15. Expansion ROM address space (limit size) can be a maximum of 16 MBytes. Bits 31 through 12 of the ERBAR is either read/write bits or read only with a value of 0 depending on the value located within the ERLR. This configuration allows the ERBAR to be programmed per *PCI Local Bus Specification*, Revision 2.3.

The Expansion ROM Base Address Register's programmed value must comply with the PCI programming requirements for address alignment. Refer to the *PCI Local Bus Specification*, Revision 2.3 for additional information on programming Expansion ROM base address registers.

Note: A write targeting the Expansion ROM window terminates as an Unsupported Request

Table 162. Expansion ROM Base Address Register -ERBAR



Internal Bus Address Offset
+030H

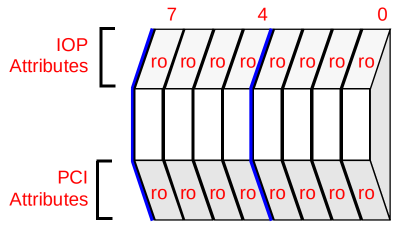
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:12	00000H	Expansion ROM Base Address - These bits define the actual location where the Expansion ROM address window resides when addressed from the PCI Express Link on any 4 Kbyte boundary.
11:01	000H	Reserved
00	0_2	Address Decode Enable - This bit field shows the ROM address decoder is enabled or disabled. When cleared, indicates the address decoder is disabled.

3.17.23 ATU Capabilities Pointer Register - ATU_Cap_Ptr

The Capabilities Pointer Register bits adhere to the definitions in the *PCI Local Bus Specification*, Revision 2.3. This register provides an offset in this function's PCI Configuration Space for the location of the first item in the first Capability list. In the case of the 4138xx, this is the PCI Express Link Power Management extended capability as defined by the *PCI Bus Power Management Interface Specification*, Revision 1.1.

Table 163. ATU Capabilities Pointer Register - ATU_Cap_Ptr



Internal Bus Address Offset
+034H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
07:00	98H	Capability List Pointer - This provides an offset in this function's configuration space that points to the 4138xx's PCI Bus Power Management extended capability.

3.17.24 ATU Interrupt Line Register - ATUILR

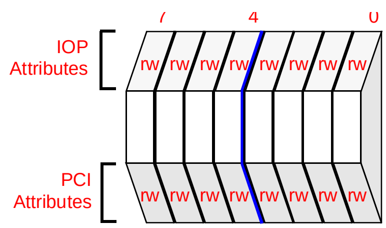
ATU Interrupt Line Register bit definitions adhere to *PCI Local Bus Specification*, Revision 2.3. This register identifies the system interrupt controller's interrupt request lines which connect to the device's PCI interrupt request lines (as specified in the interrupt pin register).

In a PC environment, for example, the register values and corresponding connections are:

- 0 (00H) through 15 (0FH) correspond to IRQ0 through IRQ15
- 16 (10H) through 254 (FEH) are reserved
- 255 (FFH) indicates "unknown" or "no connection"

Operating system or device driver can examine each device interrupt pin and interrupt line register to determine which system interrupt request line the device uses to issue requests for service.

Table 164. ATU Interrupt Line Register - ATUILR



Internal Bus Address Offset
+03CH

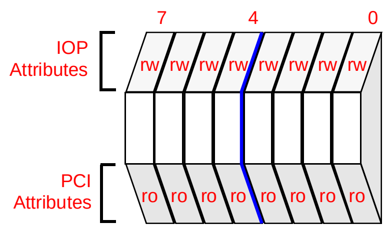
Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
07:00	FFH	Interrupt Assigned - system-assigned value identifies which system interrupt controller interrupt request line connects to the device's PCI interrupt request lines (as specified in the interrupt pin register). A value of FFH signifies "no connection" or "unknown".

3.17.25 ATU Interrupt Pin Register - ATUIPR

ATU Interrupt Pin Register bit definitions adhere to *PCI Local Bus Specification*, Revision 2.3. This register identifies the interrupt pin the ATU and Messaging Unit interface uses.

Table 165. ATU Interrupt Pin Register - ATUIPR



Internal Bus Address Offset
+03DH

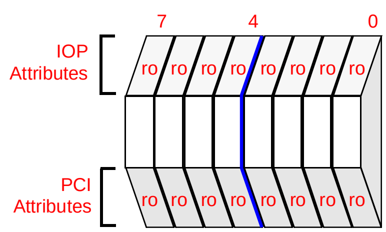
Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
07:00	01H	Interrupt Used - A value of 01H signifies that the ATU interface unit uses the INTA legacy interrupt message.

3.17.26 ATU Minimum Grant Register - ATUMGNT

This register does not apply to PCI Express.

Table 166. ATU Minimum Grant Register - ATUMGNT



Internal Bus Address Offset
+03EH

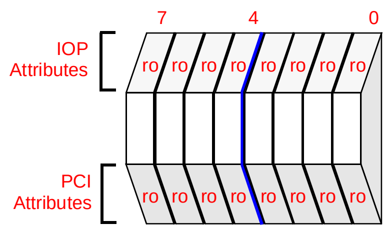
Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
07:00	00H	This register does not apply to PCI Express. Hard-wired to 0

3.17.27 ATU Maximum Latency Register - ATUMLAT

This register does not apply to PCI Express.

Table 167. ATU Maximum Latency Register - ATUMLAT



Internal Bus Address Offset
+03FH

Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
07:00	00H	This register does not apply to PCI Express. Hard-wired to 0

3.17.28 Inbound ATU Limit Register 0 - IALR0

Inbound address translation for memory window 0 occurs for requests originating in the PCI Express domain and targeting the 4138xx internal bus. The address translation block converts PCI addresses to internal bus addresses.

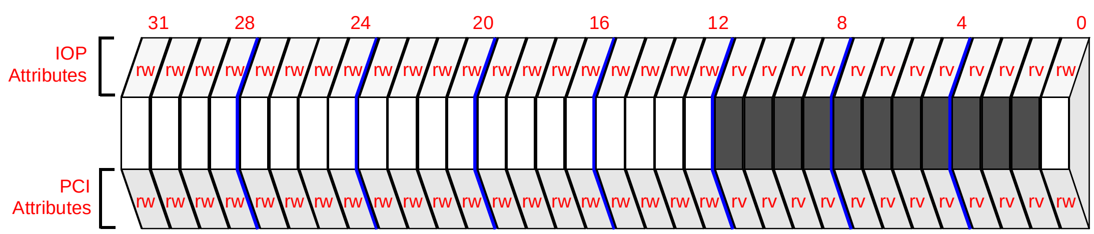
The inbound translation base address for inbound window 0 is specified in Section 3.17.13. When determining block size requirements — as described in Section 3.17.15 — the translation limit register provides the block size requirements for the base address register. The remaining registers used for performing address translation are discussed in Section 3.3.1.1.

The 4138xx value register's programmed value must be naturally aligned with the base address register's programmed value. The limit register is used as a mask; thus, the lower address bits programmed into the 4138xx value register are invalid. Refer to the *PCI Local Bus Specification*, Revision 2.3 for additional information on programming base address registers.

Bits 31 to 12 within the IALR0 have a direct effect on the IABAR0 register, bits 31 to 12, with a one to one correspondence. A value of 0 in a bit within the IALR0 makes the corresponding bit within the IABAR0 a read only bit which always returns 0. A value of 1 in a bit within the IALR0 makes the corresponding bit within the IABAR0 read/write from PCI. Note that a consequence of this programming scheme is that unless a valid value exists within the IALR0, all writes to the IABAR0 has no effect since a value of all zeros within the IALR0 makes the IABAR0 a read only register.

Note: Bit 0 can be used to disable claiming of Memory Cycles that hit Inbound Memory Window 0 even though the host processor has allocated memory of the size requested by IABAR0/IALR0[31:12].

Table 168. Inbound ATU Limit Register 0 - IALR0



Internal Bus Address Offset
+040H

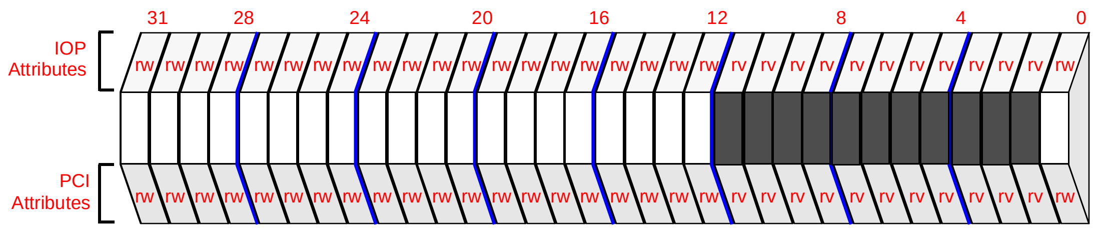
Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
31:12	FF000H	Inbound Translation Limit 0 - This readback value determines the memory block size required for inbound memory window 0 of the address translation unit. This defaults to an inbound window of 16MB.
11:01	000H	Reserved
00	0_2	Memory Window 0 Claim Disable -- When clear, Inbound Memory Window 0 claims PCI Memory Cycles Normally. When set, memory transactions targeting Inbound Memory Window 0 is terminated as an Unsupported Request (UR).

3.17.29 Inbound ATU Translate Value Register 0 - IATVR0

The Inbound ATU Translate Value Register 0 (IATVR0) in conjunction with the “Inbound ATU Upper Translate Value Register 0 - IAUTVR0” on page 319 contain bits 35 to 12 of the internal bus address used to convert PCI Express Link addresses. The converted address is driven on the internal bus as a result of the inbound ATU address translation.

Table 169. Inbound ATU Translate Value Register 0 - IATVR0



Internal Bus Address Offset
+044H

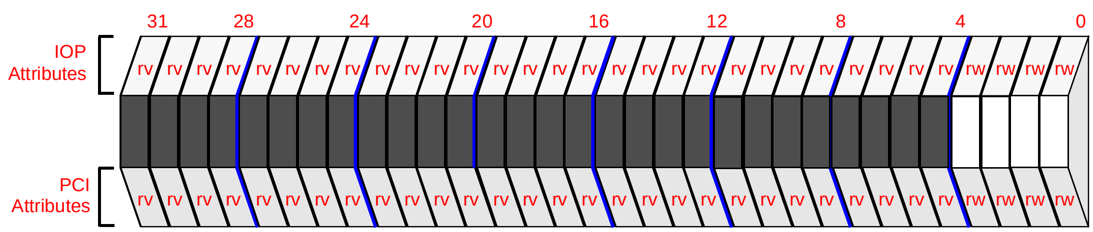
Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
31:12	FF000H	Inbound ATU Translation Value 0 - This value represents bits 31 to 12 of the internal bus address used to convert the PCI address to internal bus addresses. This value must be naturally aligned with the IABAR0 register’s programmed value (see Section 3.17.15, “Determining Block Sizes for Base Address Registers” on page 306). The default address allows the ATU to access the internal 4138xx memory-mapped registers.
11:01	000H	Reserved
00	0	Big Endian Byte Swap enable - When set the ATU performs a byte swap on all PCI read/write transactions through BAR0. When clear, no swap is performed. Refer to Section 3.4, “Big Endian Byte Swapping” on page 255 for more details.

3.17.30 Inbound ATU Upper Translate Value Register 0 - IAUTVR0

The Inbound ATU Upper Translate Value Register 0 (IAUTVR0) in conjunction with the “Inbound ATU Translate Value Register 0 - IATVR0” on page 319 contain bits 35 to12 of the internal bus address used to convert PCI Express Link addresses. The converted address is driven on the internal bus as a result of the inbound ATU address translation.

Table 170. Inbound ATU Upper Translate Value Register 0 - IAUTVR0



Internal Bus Address Offset
+048H

Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
31:04	000 0000H	Reserved
3:0	0H	Inbound Upper ATU Translation Value 0 - This value represents bits 35 to 32 of the internal bus address used to convert the PCI address to internal bus addresses. The default address allows the ATU to access the internal 4138xx memory-mapped registers.

3.17.31 Inbound ATU Limit Register 1 - IALR1

Inbound address translation for memory window 1 occurs for transactions originated in the PCI Express domain and targeting the 4138xx internal bus. The address translation block converts PCI addresses to internal bus addresses.

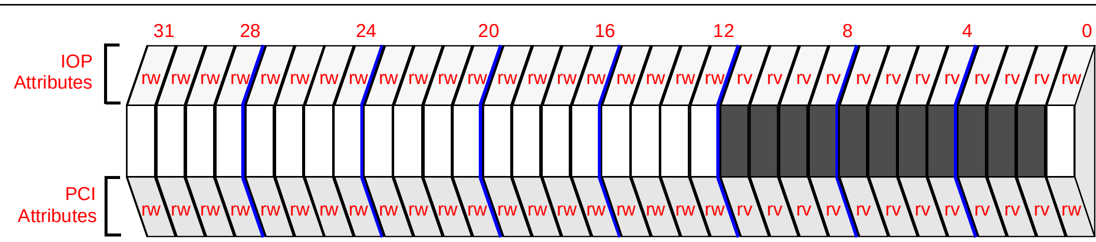
The inbound translation base address for inbound window 1 is specified in Section 3.17.16. When determining block size requirements — as described in Section 3.17.15 — the translation limit register provides the block size requirements for the base address register. The remaining registers used for performing address translation are discussed in Section 3.3.1.1.

The 4138xx value register's programmed value must be naturally aligned with the base address register's programmed value. The limit register is used as a mask; thus, the lower address bits programmed into the 4138xx value register are invalid. Refer to the *PCI Local Bus Specification*, Revision 2.3 for additional information on programming base address registers.

Bits 31 to 12 within the IALR1 have a direct effect on the IABAR1 register, bits 31 to 12, with a one to one correspondence. A value of 0 in a bit within the IALR1 makes the corresponding bit within the IABAR1 a read only bit which always returns 0. A value of 1 in a bit within the IALR1 makes the corresponding bit within the IABAR1 read/write from PCI. Note that a consequence of this programming scheme is that unless a valid value exists within the IALR1, all writes to the IABAR1 has no effect since a value of all zeros within the IALR1 makes the IABAR1 a read only register.

Note: Bit 0 can be used to disable claiming of Memory Cycles that hit Inbound Memory Window 1 even though the host processor has allocated memory of the size requested by IABAR1/IALR1[31:12].

Table 171. Inbound ATU Limit Register 1 - IALR1



Internal Bus Address Offset
+04CH

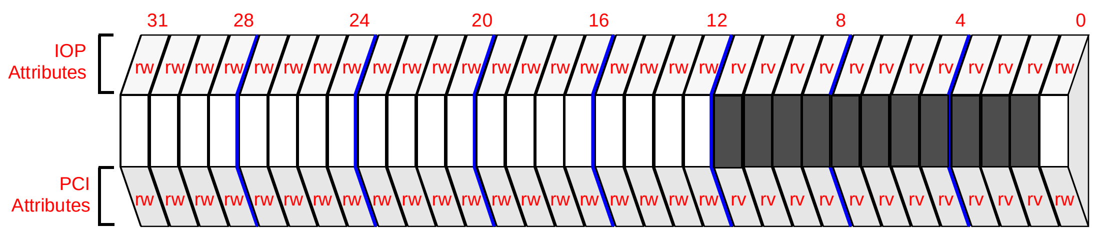
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:12	00000H	Inbound Translation Limit 1 - This readback value determines the memory block size required for the ATUs memory window 1.
11:01	000H	Reserved
00	0_2	Memory Window 1 Claim Disable -- When clear, Inbound Memory Window 1 claims PCI Memory Cycles Normally. When set, memory transactions targeting Inbound Memory Window 0 is terminated as an Unsupported Request (UR).

3.17.32 Inbound ATU Translate Value Register 1 - IATVR1

The Inbound ATU Translate Value Register 1 (IATVR1) in conjunction with the “Inbound ATU Upper Translate Value Register 1 - IAUTVR1” on page 321 contain bits 35 to 12 of the internal bus address used to convert PCI Express Link addresses. The converted address is driven on the internal bus as a result of the Inbound ATU address translation.

Table 172. Inbound ATU Translate Value Register 1 - IATVR1



Internal Bus Address Offset
+050H

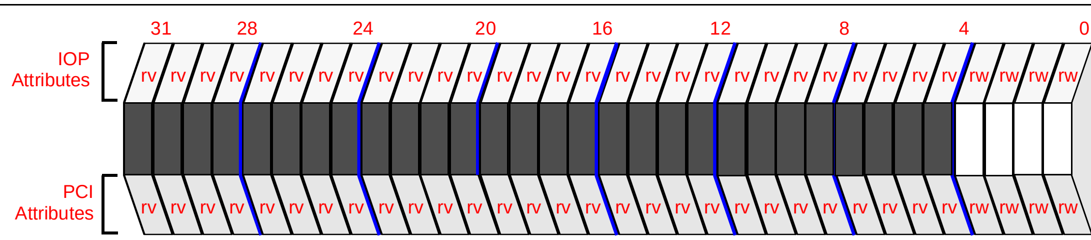
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:12	00000H	Inbound ATU Translation Value 1 - Bits 31 to 12 of the internal bus address used to convert PCI address to internal bus addresses. This value must be naturally aligned with the IABAR1 register’s programmed value (see Section 3.17.15, “Determining Block Sizes for Base Address Registers” on page 306).
11:01	000H	Reserved
00	0	Big Endian Byte Swap enable - When set the ATU performs a byte swap on all PCI read/write transactions through BAR1. When clear, no swap is performed. Refer to Section 3.4, “Big Endian Byte Swapping” on page 255 for more details.

3.17.33 Inbound ATU Upper Translate Value Register 1 - IAUTVR1

The Inbound ATU Upper Translate Value Register 1 (IAUTVR1) in conjunction with the “Inbound ATU Translate Value Register 1 - IATVR1” on page 321 contain bits 35 to12 of the internal bus address used to convert PCI Express Link addresses. The converted address is driven on the internal bus as a result of the inbound ATU address translation.

Table 173. Inbound ATU Upper Translate Value Register 1 - IAUTVR1



Internal Bus Address Offset
+054H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:04	000 0000H	Reserved
3:0	0H	Inbound Upper ATU Translation Value 1 - This value represents bits 35 to 32 of the internal bus address used to convert the PCI address to internal bus addresses.

3.17.34 Inbound ATU Limit Register 2 - IALR2

Inbound address translation for inbound window 2 occurs for transactions originated in the PCI Express domain and targeting the 4138xx internal bus. The address translation block converts PCI addresses to internal bus addresses.

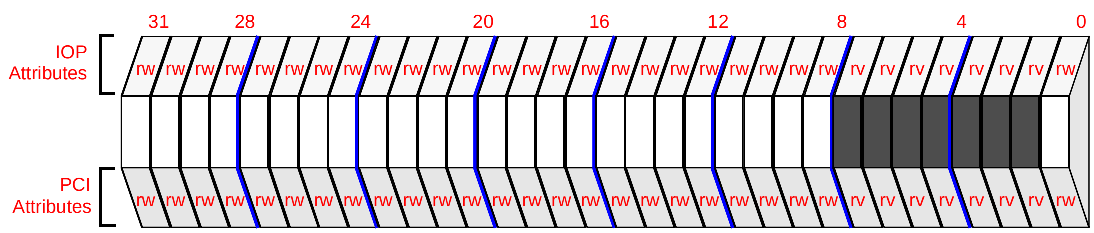
The inbound translation base address for inbound window 2 is specified in Section 3.17.18. When determining block size requirements — as described in Section 3.17.15 — the translation limit register provides the block size requirements for the base address register. The remaining registers used for performing address translation are discussed in Section 3.3.1.1.

The 4138xx value register's programmed value must be naturally aligned with the base address register's programmed value. The limit register is used as a mask; thus, the lower address bits programmed into the 4138xx value register are invalid. Refer to the *PCI Local Bus Specification*, Revision 2.3 for additional information on programming base address registers.

Bits 31 to 8 within the IALR2 have a direct effect on the IABAR2 register, bits 31 to 8, with a one to one correspondence. A value of 0 in a bit within the IALR2 makes the corresponding bit within the IABAR2 a read only bit which always returns 0. A value of 1 in a bit within the IALR2 makes the corresponding bit within the IABAR2 read/write from PCI. Note that a consequence of this programming scheme is that unless a valid value exists within the IALR2, all writes to the IABAR2 has no effect since a value of all zeros within the IALR2 makes the IABAR2 a read only register.

Note: Bit 0 can be used to disable claiming of PCI Cycles that hit Inbound Window 1 even though the host processor has allocated memory of the size requested by IABAR2/IALR2[31:8].

Table 174. Inbound ATU Limit Register 2 - IALR2



Internal Bus Address Offset
+058H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:8	00000H	Inbound Translation Limit 2 - This readback value determines the memory block size required for the ATUs memory window 2.
07:01	000H	Reserved
00	0_2	Window 2 Claim Disable -- When clear, Inbound Memory Window 2 claims PCI Memory or I/O Cycles Normally. When set, transactions targeting Inbound Memory Window 2 is terminated as an Unsupported Request (UR).

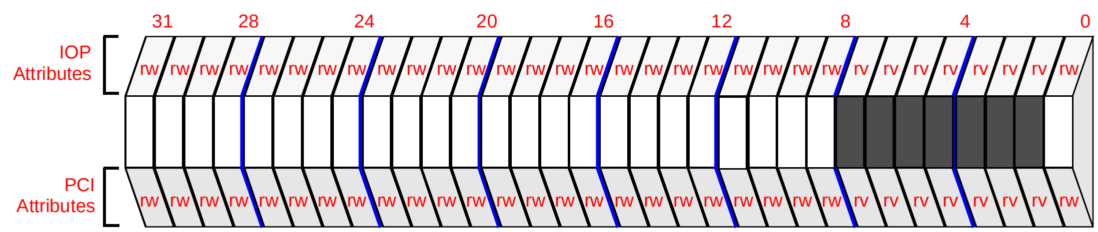
3.17.35 Inbound ATU Translate Value Register 2 - IATVR2

The Inbound ATU Translate Value Register 2 (IATVR2) in conjunction with the “Inbound ATU Upper Translate Value Register 2 - IAUTVR2” on page 324 contain bits 35 to 8 of the internal bus address used to convert PCI Express Link addresses. The converted address is driven on the internal bus as a result of the Inbound ATU address translation.

Warning: When the IABAR2 register’s Memory Space Indicator is set, inbound window 2 will be in I/O space. In this case, the *PCI Local Bus Specification*, Revision 2.3 requires that the IABAR2 request no more than 256 bytes of I/O space. Thus, IALR2 must be set to FFFF FF00H when the Memory Space Indicator in IABAR2 is set.

Warning: Although IATVR2[09:08] are programmable bits, hardware does not translate these bits for PCI windows that are defined to be less than 1-Kbyte. Hardware uses the PCI Address bits[09:08] as captured on the PCI address bus to drive the internal bus address[09:08].

Table 175. Inbound ATU Translate Value Register 2 - IATVR2



Internal Bus Address Offset
+05CH

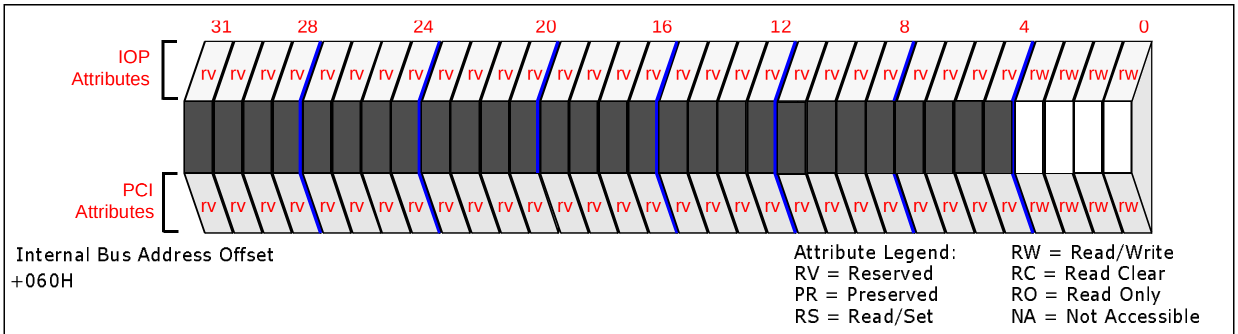
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:08	00000H	Inbound ATU Translation Value 2 - This value represents bits 31 to 8 of the internal bus address used to convert the PCI address to internal bus addresses. This value must be naturally aligned with the IABAR2 register’s programmed value (see Section 3.17.15, “Determining Block Sizes for Base Address Registers” on page 306).
07:01	000H	Reserved
00	0	Big Endian Byte Swap enable - When set the ATU performs a byte swap on all PCI read/write transactions through BAR2. When clear, no swap is performed. Refer to Section 3.4, “Big Endian Byte Swapping” on page 255 for more details.

3.17.36 Inbound ATU Upper Translate Value Register 2 - IAUTVR2

The Inbound ATU Upper Translate Value Register 2 (IAUTVR2) in conjunction with the "Inbound ATU Translate Value Register 2 - IATVR2" on page 323 contain bits 35 to 8 of the internal bus address used to convert PCI Express Link addresses. The converted address is driven on the internal bus as a result of the inbound ATU address translation.

Table 176. Inbound ATU Upper Translate Value Register 2 - IAUTVR2



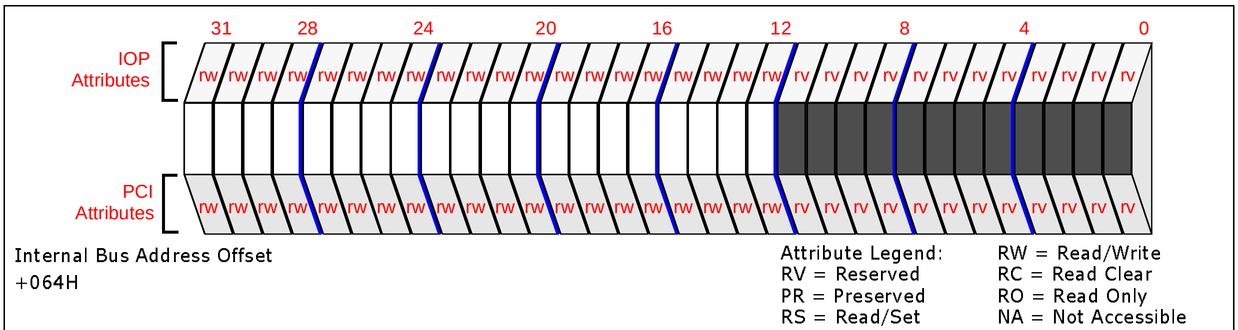
Bit	Default	Description
31:04	000 0000H	Reserved
3:0	0H	Inbound Upper ATU Translation Value 2 - This value represents bits 35 to 32 of the internal bus address used to convert the PCI address to internal bus addresses.

3.17.37 Expansion ROM Limit Register - ERLR

The Expansion ROM Limit Register (ERLR) defines the block size of addresses the ATU defines as Expansion ROM address space. Block size is programmed by writing a value into the ERLR.

Bits 31 to 12 within the ERLR have a direct effect on the ERBAR register, bits 31 to 12, with a one to one correspondence. A value of 0 in a bit within the ERLR makes the corresponding bit within the ERBAR a read only bit which always returns 0. A value of 1 in a bit within the ERLR makes the corresponding bit within the ERBAR read/write from PCI.

Table 177. Expansion ROM Limit Register - ERLR

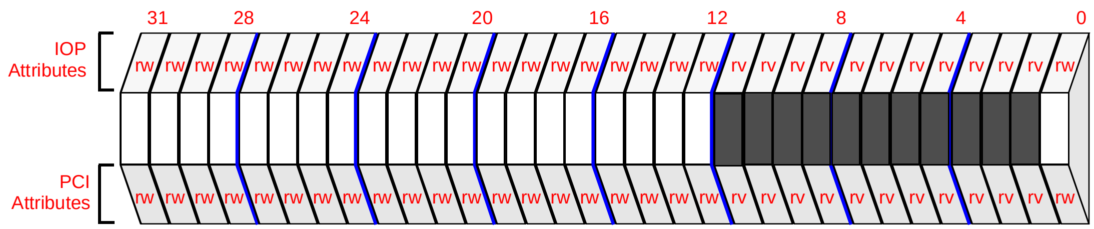


Bit	Default	Description
31:12	000000H	Expansion ROM Limit - Block size of memory required for Expansion ROM translation unit. Default value is 0, which indicates no Expansion ROM address space and all bits within the ERBAR are read only with a value of 0.
11:00	000H	Reserved

3.17.38 Expansion ROM Translate Value Register - ERTVR

The Expansion ROM Translate Value Register 0 (ERTVR) in conjunction with the "Expansion ROM Upper Translate Value Register - ERUTVR" on page 325 contain bits 35 to 12 of the internal bus address used to convert PCI Express Link addresses. The converted address is driven on the internal bus as a result of the Expansion ROM address translation.

Table 178. Expansion ROM Translate Value Register - ERTVR



Internal Bus Address Offset
+068H

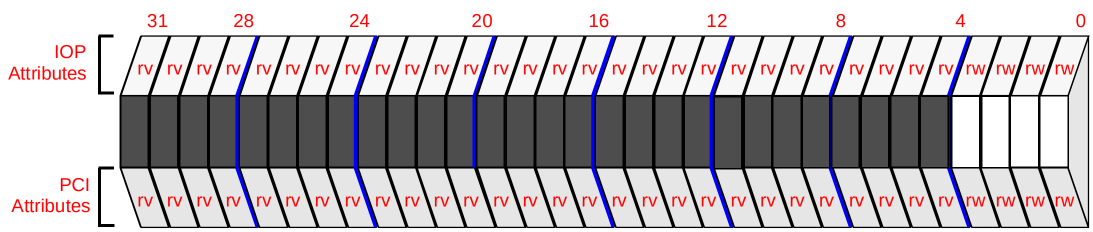
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:12	00000H	Expansion ROM Translation Value - This value represents bits 31 to 12 of the internal bus address used to convert the PCI address to internal bus addresses. This value must be naturally aligned with the ERBAR register's programmed value (see Section 3.17.15, "Determining Block Sizes for Base Address Registers" on page 306).
11:01	000H	Reserved
00	0	Big Endian Byte Swap enable - When set the ATU performs a byte swap on all PCI read transactions through ERBAR. When clear, no swap is performed. Refer to Section 3.4, "Big Endian Byte Swapping" on page 255 for more details.

3.17.39 Expansion ROM Upper Translate Value Register - ERUTVR

The Expansion ROM Upper Translate Value Register (ERUTVR) in conjunction with the "Expansion ROM Translate Value Register - ERTVR" on page 325 contain bits 35 to12 of the internal bus address used to convert PCI Express Link addresses. The converted address is driven on the internal bus as a result of the Expansion ROM address translation.

Table 179. Expansion ROM Upper Translate Value Register - ERUTVR



Internal Bus Address Offset
+06CH

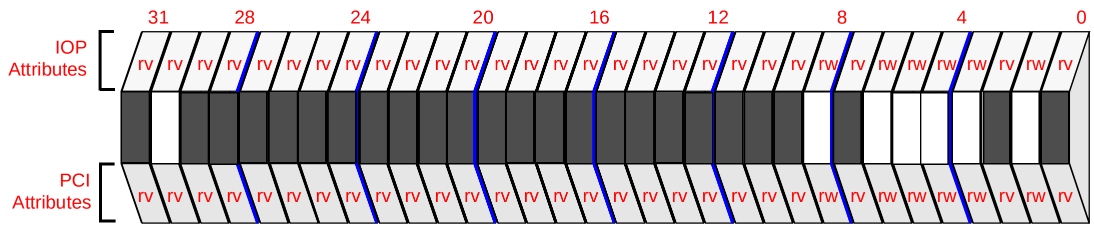
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:04	000 0000H	Reserved
3:0	0H	Expansion ROM Upper Translation Value - This value represents bits 35 to 32 of the internal bus address used to convert the PCI address to internal bus addresses.

3.17.40 ATU Configuration Register - ATUCR

The ATU Configuration Register contains some additional parameters in the ATU.

Table 180. ATU Configuration Register - ATUCR



Internal Bus Address Offset
+070H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

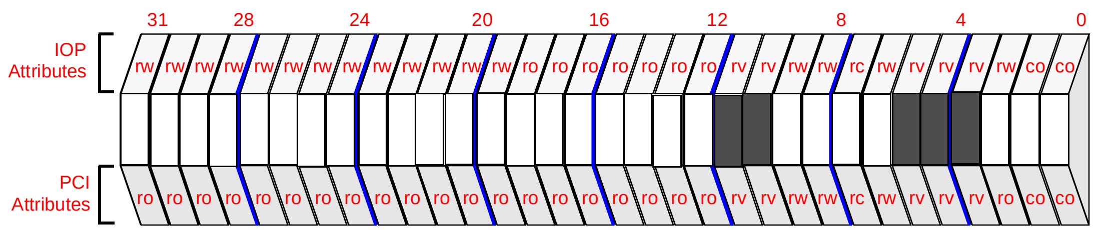
Bit	Default	Description
31	0	Reserved
30	0	Completion Timeout Disable. - When set, this bit disables the completion timeout mechanism for outbound non-posted requests. Read/Write.
29:09	0	Reserved
08	0	Halt on Error. - When enabled, an Address or Data Parity an Outbound write request results in the follow actions: • Stop accepting outbound requests by clearing the outbound enable. • Flush any pending outbound write requests from the transaction queue that are behind the one that had the error. • Assert the Halt On Error Interrupt to the Intel XScale® Processor. • Assert the appropriate error interrupt to the Intel XScale® Processor. • Read requests is allowed to proceed normally though no new ones is accepted.
07	0	Reserved
06	1	Drop subsequent inbound vendor defined messages (IVM). 0 = When cleared, subsequent IVMs remains in the inbound posted queue and blocks all other transactions. 1 = When set and the inbound vendor defined message registers are full with a pending IVM, subsequent IVMs is discarded. This is necessary to prevent a deadlock condition when the Intel XScale® processor needs to receive a read completion before it can handle the IVM interrupt.
05	0	Outbound Completion Size - This bit controls how completions are returned on the PCI Express interface. 0 = Max Payload Size setting is used to format completions. Once enough data has accumulated to reach a Max Payload address boundary, the completion TLP is formed and pushed into the completion queue. Completion TLPs has a payload size of 128, 256, or 512 bytes depending on the setting of the Max_Payload_Size field in the "PCI Express Device Control Register - PE_DCTL" on page 344. 1 = 128B Payload Size. All completions uses a maximum size of 128 bytes regardless of the setting of the Max_Payload_Size field.
04	0	Inbound Minimum Completion Size - This bit controls how completion data is returned to the internal bus. 0 = Return completion data as it is received from the PCI Express interface. This size may be as small as 64Bytes. 1 = Wait for the entire request to be satisfied before returning completion data.
03	0	ATU BIST Interrupt Enable - When set, enables an interrupt to the Intel XScale® processor when the start BIST bit is set in the ATUBISTR register. This bit is also reflected as the BIST Capable bit 7 in the ATUBISTR register.
02	0_2	Reserved
01	0_2	Outbound ATU Enable - When set, enables the outbound address translation unit. When cleared, disables the outbound ATU. ***Note:*** Transport Firmware always sets the bit to 1. Do no clear this bit.
00	0_2	Reserved

3.17.41 PCI Configuration and Status Register - PCSR

The PCI Configuration and Status Register has additional bits for controlling and monitoring various features of the PCI Express interface.

Warning: The PCI Express Bus Number and Device Number are used to form the Requestor/Completer ID and should only be changed when operating as a Root Complex. These fields are updated whenever a type 0 configuration write targets the IOP. System instability may result when the Bus/Device numbers are modified while operating as an endpoint.

Table 181. PCI Configuration and Status Register - PCSR (Sheet 1 of 2)

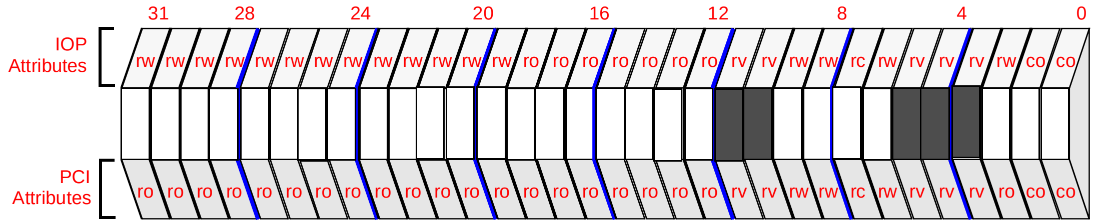


Internal Bus Address Offset
+074H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
CO = Clear Only

Bit	Default	Description
31:24	00H	PCI Express Bus Number
23:19	0_0000_2	PCI Express Device Number
18:16	000_2	PCI Express Function Number
15	0_2	Outbound Transaction Queue Busy: 0 = Outbound Transaction Queue Empty 1 = Outbound Transaction Queue Busy ***Note:*** This tracks outbound transactions and includes the Outbound Non-Posted, Outbound Posted, and Inbound Completion queues.
14	0_2	Inbound Transaction Queue Busy: 0 = Inbound Transaction Queue Empty 1 = Inbound Transaction Queue Busy ***Note:*** This tracks inbound transactions and includes the Inbound Non-Posted, Inbound Posted, and Outbound Completions queues.
13	Varies with external state of the **PCIE_RC#** strap.	PCI Express Root Complex mode 0 = PCI Express Root Complex enabled 1 = PCI Express Root Complex disabled (end point mode).
12	0	Link Layer Retry Buffer (LLRB) Busy. 0 = LLRB Empty 1 = LLRB Busy
11:10	00_2	Reserved

Table 181. PCI Configuration and Status Register - PCSR (Sheet 2 of 2)



Internal Bus Address Offset
+074H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
CO = Clear Only

Bit	Default	Description
9:8	00	Core reset. These bits are used to place the Intel XScale® microarchitecture processors into reset. When a '1' is written to these bits the corresponding Core Processor Reset bit (Bits1:0) is set. These bits are self clearing and always reads '0'.
07	Varies with external state of the **FW_TIMER_OFF#** strap	Firmware Timer Disable 0 = Firmware Timeout is disabled. 1 = Firmware Timeout is enabled When enabled, a 400mS timer is started at the trailing edge of reset. The Configuration Request Retry (bit 2 of this register) is cleared when it is not cleared before the expiration of the timer. For example, when firmware does not clear the Configuration Request Retry bit before the timer expires. The firmware timer bit is also automatically cleared by the firmware timer when the Configuration Request Retry bit is still set when the timer expires. After the host is allowed access, the firmware timer bit can be used to indicate how the Configuration Request Retry bit was cleared: 0 = The firmware timer expired and cleared both the Configuration Request Retry and the firmware timer bits. 1 = Firmware cleared the Configuration Request Retry bit before the timer expired. ***Note:*** When the firmware timer is disabled, firmware is responsible to clear the Configuration Request Retry bit. Otherwise, the ATU indefinitely retries all host configuration cycles.
05:03	00_2	Reserved
02	Varies with external state of **RETRY** pin at PCI Express Link reset	Configuration Request Retry - When this bit is set, the PCI Express interface of the 4138xx responds to all configuration cycles with a Completion Retry Status (CRS) condition. When clear, the 4138xx responds to the appropriate configuration cycles. The default condition for this bit is based on the external state of the **RETRY** pin at the rising edge of **P_RST#**. When the external state of the pin is high, the bit is set. When the external state of the pin is low, the bit is cleared. ***Note:*** This bit self-clears when the Firmware Timer expires. Bit 7 of this register is used to stop the firmware timer.
01:00	Varies with external state of **RST_MODE[1:0]#** pin at PCI Express Link reset	Core Processor Reset - This bit is set to its default value by the hardware when either **P_RST#** is asserted or the Reset Internal Bus bit in PCSR is set. When this bit is set, the Intel XScale® processor is being held in reset. Software cannot set this bit. Software is required to clear this bit to deassert Intel XScale® processor reset. The default condition for this bit is based on the external state of the **RST_MODE[1:0]#** pin at the rising edge of **P_RST#**. When the external state of the pin is low, the default value of this bit is set. When the external state of the pin is high, the default value of this bit is clear. ***Note:*** This bits are "Clear Only". A write of '0' clears the bit, a write of '1' has no affect.

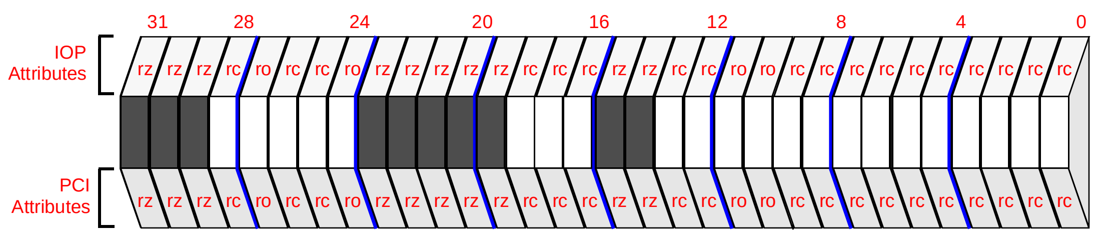
3.17.42 ATU Interrupt Status Register - ATUISR

The ATU Interrupt Status Register is used to notify the core processor of the source of an ATU interrupt. In addition, this register is written to clear the source of the interrupt to the interrupt unit of the 4138xx. All bits in this register are Read/Clear.

Bits 4:0 are a direct reflection of bits 15, 13:11, and bit 8 (respectively) of the ATU Status Register (these bits are set at the same time by hardware but need to be cleared independently). Bit 5 is set by an error associated with the internal bus of the 4138xx. Bit 24 is for software BIST. The conditions that result in an ATU interrupt are cleared by writing a 1 to the appropriate bits in this register.

Note: The interrupt status bits are not set when the corresponding mask bit is set.

Table 182. ATU Interrupt Status Register - ATUISR (Sheet 1 of 3)

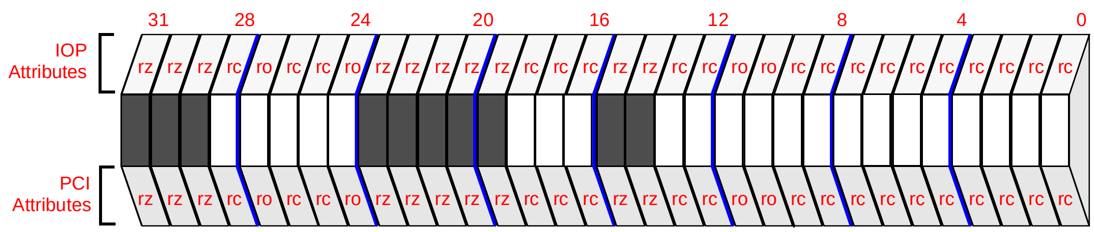


Internal Bus Address Offset
+078H

Attribute Legend:
RZ = Reserved Zero
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:29	0	Reserved Zero - Software must write 0 to these bits.
28	0	Slot Power Message Received - Indicates a Set_Slot_Power_Limit Message was received and logged in the "PCI Express Device Capabilities Register - PCIE_DCAP" on page 343. Generates the ATU Inbound Message Interrupt.
27	0	PME Interrupt - Indicates a PME message was received and logged in the "PCI Express Root Status Register PE_RSR" on page 354 **Note:** This read only bit is a copy of the start PME Status bit in the PE_RSR register. To clear this interrupt software must clear the PME Status bit. The "mask" for this interrupt is controlled by the PME interrupt enable in the "PCI Express Root Control Register - PE_RCR" on page 353. Generates the ATU Inbound Message Interrupt
26	0	Hot-Plug Message Received - This bit set when a Hot-Plug message is received that changes the value of the Attention Indicator Status or Power Indicator Status in the "PCI Express Message Control and Status Register - PEMCSR" Generates the ATU Inbound Message Interrupt
25	0	Inbound Vendor Message Received (IVM) - Note while this bit is asserted, any additional Inbound Vendor Messages may be blocked which can block all other inbound transactions. This bit must be cleared after the message has been processed. See Section 3.3.1.6, "Inbound Vendor_Defined Message Transactions" on page 243 for more details. Generates the ATU Inbound Message Interrupt
24	0	ATU BIST Interrupt - When set, the host processor has set the start BIST, ATUBISTR register bit 6, and the ATU BIST interrupt enable (ATUCR register bit 3) is enabled. The Intel XScale® processor can initiate the software BIST and store the result in ATUBISTR register bits 3:0. **Note:** This read only bit is a copy of the start BIST bit in the ATUBISTR register. To clear this interrupt software must clear the start BIST bit. The "mask" for this interrupt is controlled by the BIST interrupt enable in the ATUCR register. Generates the ATU BIST Interrupt
23:19	0	Reserved Zero - Software must write 0 to these bits.
18	0	ATU Configuration Write - This bit is set when a PCI Express configuration write occurs to any enabled function. When set, this bit results in the assertion of the ATU Config Reg Write Interrupt.

Table 182. ATU Interrupt Status Register - ATUISR (Sheet 2 of 3)

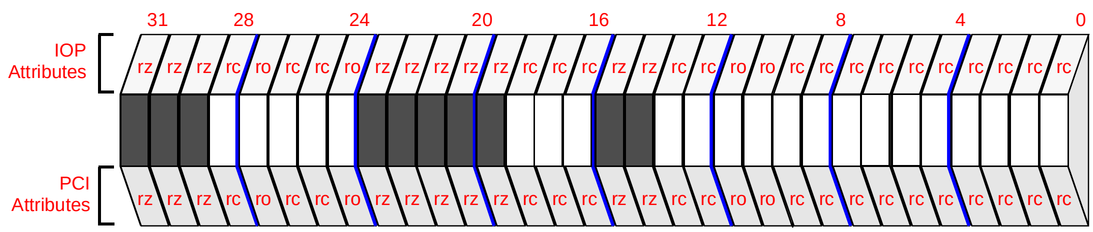


Internal Bus Address Offset
+078H

Attribute Legend:
RZ = Reserved Zero
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
17	0	VPD Address Register Updated - Set when a configuration write occurs to the VPDAR register. When set, this bit results in the assertion of the ATU Config Reg Write Interrupt.
16	0	Power State Transition - Set when the Power State Field of the ATU Power Management Control/Status Register is written to transition from D0 ->D3, D0 ->D1, D1 -> D3, or D3 ->D0. When set, this bit results in the assertion of the ATU Config Req Write Interrupt.
15:14	0	Reserved Zero - Software must write 0 to these bits
13	0	Halt on Error Interrupt - Set when an outbound address or data parity error is detected on a posted write and the Halt on Error control (bit 8) is set in the "ATU Configuration Register - ATUCR"
12	0	Root System Error Interrupt - This bit set when the System Error enable bits are set in the "PCI Express Root Control Register - PE_RCR" on page 353 and a non-masked ERR_FATAL, ERR_NONFATAL, or ERR_COR message is received. ***Note:*** A PCI Express Error message may also cause bit 11 to be set. Generates the ATU Error Interrupt
11	0	Root Error Message received - Indicates a Root Complex error message was received and logged in the "Root Error Status Register - RERR_SR" ***Note:*** This read only bit is set when any of bits 6, 5, or 0 is set in the RERR_SR registers. To clear this interrupt, these bit must be cleared in the RERR_SR register. The mask for this interrupt is controlled by the "Root Error Command Register - RERR_CMD" register. Generates the ATU Error Interrupt
10	0	PCI Interface Error - Indicates a non-masked error was logged in the "PCI Interface Error Status - PIE_STS" register. ***Note:*** This read only bit is an OR of the unmasked PIE_STS bits. Generates the ATU Error Interrupt
09	0	Correctable Error Message Transmitted - Indicates a ERR_COR message was sent to the Root Complex. A message is transmitted when a correctable error was logged and the corresponding mask is cleared in the "PCI Express Correctable Error Mask - ERRCOR_MSK" register Generates the ATU Error Interrupt
08	0	Uncorrectable Error Message Transmitted - Indicates an ERR_FATAL or ERR_NONFATAL message was sent to the root complex. A message is transmitted when an uncorrectable error was logged and the corresponding mask is cleared in the "PCI Express Uncorrectable Error Mask - ERRUNC_MSK" register Generates the ATU Error Interrupt
07	0	Received Configuration Retry Status (CRS) - This bit set whenever an outbound configuration request is completed with a CRS status. Generates the ATU Error Interrupt
06	0	Link Down Interrupt - This bit is set whenever the PCI Express link goes down.
05	0	Internal Bus Master Abort - Set when a transaction initiated by the ATU internal bus initiator interface ends in a Master-abort. ***Note:*** An internal bus master abort may also result in the setting of the Signaled Target Abort. Generates the ATU Error Interrupt
04	0	Detected Parity Error Interrupt - Set when a poisoned TLP is received by any function, regardless of the state of the Parity Error Response. Generates the ATU Error Interrupt

Table 182. ATU Interrupt Status Register - ATUISR (Sheet 3 of 3)



Internal Bus Address Offset
+078H

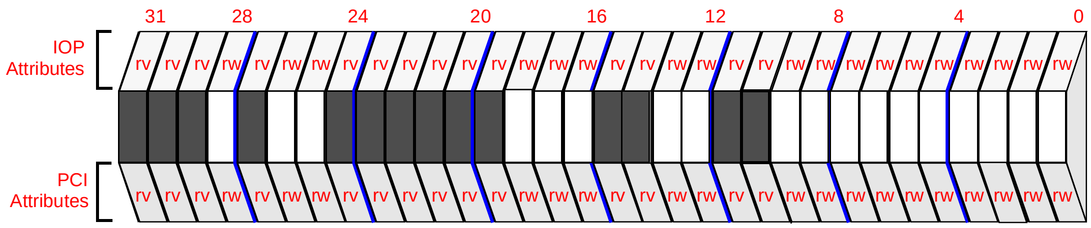
Attribute Legend:
RZ = Reserved Zero
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
03	0	Received Master Abort Interrupt - Set when an Unsupported Request (UR) completion is received by any function. Generates the ATU Error Interrupt
02	0	Signaled Target Abort Interrupt - Set when a request is completed using a Completer Abort (CA) Completion Status by any function. Generates the ATU Error Interrupt
01	0	Received Target Abort Interrupt - Set when a completion with Completer Abort Completion Status is received by any function. Generates the ATU Error Interrupt
00	0	Master Data Parity Error Interrupt - Set when the Parity Error Response is enabled and any function transmits a Poisoned Write TLP or receives a Poisoned Completion TLP. Generates the ATU Error Interrupt

3.17.43 ATU Interrupt Mask Register - ATUIMR

The ATU Interrupt Mask Register contains the control bit to enable and disable interrupts generated by the ATU.

Table 183. ATU Interrupt Mask Register - ATUIMR



Internal Bus Address Offset
+07CH

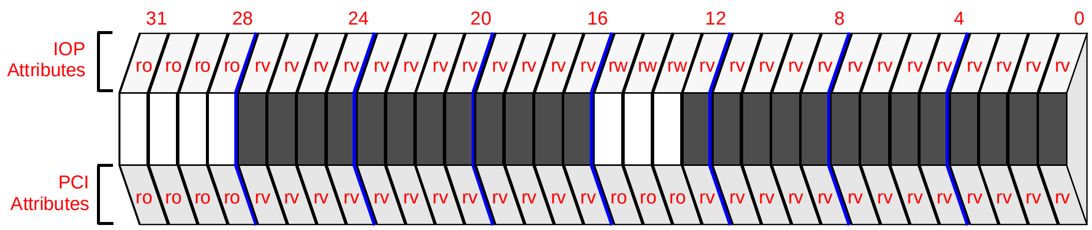
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:29	0	Reserved
28	0	Slot Power Message Received Interrupt Mask - When '1' the interrupt is masked.
27	0	Reserved ***Note:*** The PME interrupt is masked in the "PCI Express Root Control Register - PE_RCR" on page 353
26	0	Hot-Plug Message Received Interrupt Mask - When '1' the interrupt is masked
25	0	Inbound Vendor Message Received Interrupt Mask - When '1' the interrupt is masked
24	0	Reserved ***Note:*** The ATUBIST interrupt is masked by the BIST interrupt enable in the ATUCR register.
23:19	0	Reserved
18	0	ATU Configuration Write Mask - When '1' the interrupt is masked.
17	0	VPD Address Register Updated Mask- When '1' the interrupt is masked
16	0	Power State Transition Mask - When '1' the interrupt is masked
15:14	0	Reserved
13	0	Halt on Error Interrupt Mask - When '1' the interrupt is masked.
12	0	Root System Error Interrupt Mask - When '1' the interrupt is masked
11	0	Reserved
10	0	Reserved ***Note:*** ATUISR[10] is controlled by the "PCI Interface Error Mask - PIE_MSK" register.
09	0	Correctable Error Logged Interrupt Mask - When '1' the interrupt is masked
08	0	Uncorrectable Error Logged Interrupt Mask - When '1' the interrupt is masked
07	0	Received Configuration Retry Status (CRS) Mask - When '1' the interrupt is masked
06	0	Link Down Interrupt Mask - When '1' the interrupt is masked.
05	0	Internal Bus Master Abort Interrupt Mask - When '1' the interrupt is masked
04	0	Data Parity Error Interrupt Mask - When '1' the interrupt is masked
03	0	Received Master Abort Interrupt Mask - When '1' the interrupt is masked
02	0	Signaled Target Abort Interrupt Mask - When '1' the interrupt is masked
01	0	Received Target Abort Interrupt Mask - When '1' the interrupt is masked
00	0	Master Data Parity Error Interrupt Mask - When '1' the interrupt is masked

3.17.44 PCI Express Message Control/Status Register - PEMCSR

The PCI Express Message Control/Status Register controls the generation and logs the receipt of PCI Express Power Management and Hot-Plug messages.

Table 184. PCI Express Message Control and Status Register - PEMCSR



Internal Bus Address Offset
+080H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

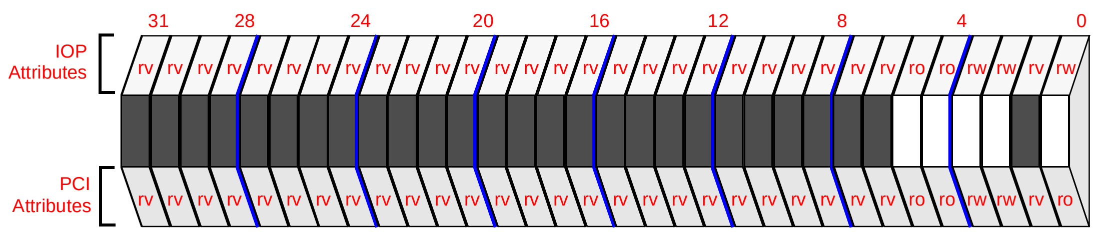
Bit	Default	Description
31:30	00_2	Attention Indicator Status As an end point, the ATU may receive Hot-Plug Attention Indicator Control Messages. When the message is received, the status is logged in these fields and the Hot-Plug interrupt bit is set in the ATUISR. 00 Reserved 01 On 10 Blink 11 Off ***Note:*** These bits are updated regardless of the state of the Hot-Plug Interrupt Mask in the ATUIMR.
29:28	00_2	Power Indicator Status As an end point, the ATU may receive Hot-Plug Power Indicator Control Messages. When the message are received, the status is logged in these fields and the Hot-Plug interrupt bit is set in the ATUISR. 00 Reserved 01 On 10 Blink 11 Off ***Note:*** These bits are updated regardless of the state of the Hot-Plug Interrupt Mask in the ATUIMR.
27:16	000H	Reserved.
15	0_2	Attention Button Pressed Control When this bit is asserted, an Attention_Button_Pressed message is generated. This bit self-clears after the message has been transmitted. Only valid as an end point.
14	0_2	Inbound Vendor_Defined Type 0 Unsupported Request response When asserted, the ATU generates an Unsupported Request response for the Message logged in the Inbound Vendor Defined Message Registers. The contents of the IVMH registers is copied to the Advanced Error Header Log and the unsupported request status is updated. When the UR response is required, software must set this bit prior to clearing the Interrupt status bit in the ATUISR. This bit is self clearing after the UR message has been delivered.
13:0	0000H	Reserved

3.17.45 PCI Express Link Control/Status Register - PELCSR

The PCI Express Link Control/Status Register controls various parameters of the Link Layer including Training Sequence.

Note: When operating as an endpoint, these bits operate as status bits and reflect the settings of the most recent TS1/TS2 training sequences. When operating as a Root Complex, these bits are control bits that are used when sending the training sequences. Training sequences can be initiated by setting the Retrain Link bit in the "PCI Express Link Control Register PE_LCTL" on page 348

Table 185. PCI Express Link Control and Status Register - PELCSR



Internal Bus Address Offset
+084H

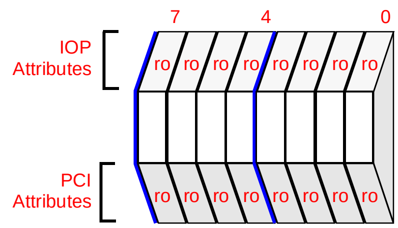
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:6	0000H	Reserved
5		Scrambling Disabled Status 0 = Scrambling Active 1 = Scrambling Disabled
4		Loopback Status 0 = Loopback Disabled 1 = Loopback Enabled
3	0	Disable Scrambling - This controls the disable scrambling bit in the TS1 Ordered Set. 0 = Scrambling Active 1 = Scrambling Disabled
2	0	Loopback - This controls the loopback bit in the TS1 Ordered Set. 0 = Loopback Disabled 1 = Loopback Enabled
1	0	Reserved ***Note:*** The Disable Link control is located in the "PCI Express Link Control Register PE_LCTL" on page 348
0	0	Hot Reset - When operating as an endpoint, this is a status bit. When operating as a Root Complex, this is a control bit. 0 = Hot Reset Disabled 1 = Hot Reset Enabled

3.17.46 VPD Capability Identifier Register - VPD_Cap_ID

The Capability Identifier Register bits adhere to the definitions in the *PCI Local Bus Specification*, Revision 2.3. This register in the PCI Extended Capability header identifies the type of Extended Capability contained in that header. In the case of the 4138xx, this is the VPD extended capability with an ID of 03H as defined by the *PCI Local Bus Specification*, Revision 2.3.

Table 186. VPD Capability Identifier Register - VPD_Cap_ID



Internal Bus Address Offset
+090H

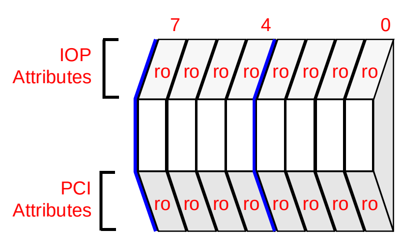
Attribute Legend:	RW = Read/Write
RV = Reserved	RC = Read Clear
PR = Preserved	RO = Read Only
RS = Read/Set	NA = Not Accessible

Bit	Default	Description
07:00	03H	Cap_Id - This field with its' 03H value identifies this item in the linked list of Extended Capability Headers as being the VPD capability registers.

3.17.47 VPD Next Item Pointer Register - VPD_Next_Item_Ptr

The Next Item Pointer Register bits adhere to the definitions in the *PCI Local Bus Specification*, Revision 2.3. This register describes the location of the next item in the function's capability list. For the 4138xx, this the final capability list, and hence, this register is set to 00H.

Table 187. VPD Next Item Pointer Register - VPD_Next_Item_Ptr



Internal Bus Address Offset
+091H

Attribute Legend:	RW = Read/Write
RV = Reserved	RC = Read Clear
PR = Preserved	RO = Read Only
RS = Read/Set	NA = Not Accessible

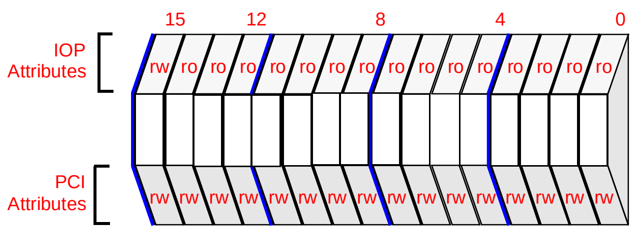
Bit	Default	Description
07:00	00H	Next_ Item_ Pointer - This field provides an offset into the function's configuration space pointing to the next item in the function's capability list. Since the VPD capabilities are the last in the linked list of extended capabilities in the 4138xx, the register is set to 00H.

3.17.48 VPD Address Register - VPDAR

The VPD Address register (VPDAR) contains the DWORD-aligned byte address of the VPD to be accessed. The register is read/write and the initial value at power-up is indeterminate.

A PCI Configuration Write to the VPDAR interrupts the Intel XScale® processor. Software can use the Flag setting to determine whether the configuration write was intended to initiate a read or write of the VPD through the VPD Data Register.

Table 188. VPD Address Register - VPDAR



Internal Bus Address Offset
+092

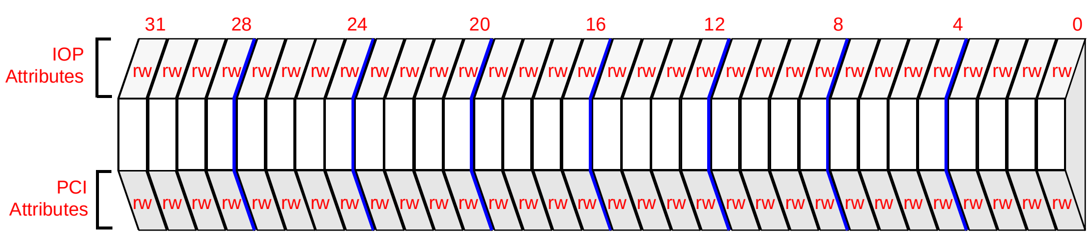
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
15	0_2	Flag - A flag is used to indicate when a transfer of data between the VPD Data Register and the storage component has completed. Please see Section 3.13, "Vital Product Data" on page 282 for more details on how the 4138xx handles the data transfer.
14:0	0000H	VPD Address - This register is written to set the DWORD-aligned byte address used to read or write Vital Product Data from the VPD storage component.

3.17.49 VPD Data Register - VPDDR

This register is used to transfer data between the 4138xx and the VPD storage component.

Table 189. VPD Data Register - VPDDR



Internal Bus Address Offset
+094H

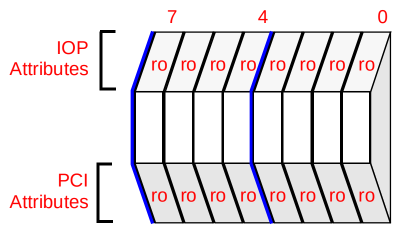
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:0	0000H	VPD Data - Four bytes are always read or written through this register to/from the VPD storage component.

3.17.50 PM Capability Identifier Register - PM_Cap_ID

The Capability Identifier Register bits adhere to the definitions in the *PCI Local Bus Specification*, Revision 2.3. This register in the PCI Extended Capability header identifies the type of Extended Capability contained in that header. In the case of the 4138xx, this is the PCI Express Link Power Management extended capability with an ID of 01H as defined by the *PCI Bus Power Management Interface Specification*, Revision 1.1.

Table 190. PM_Capability Identifier Register - PM_Cap_ID



Internal Bus Address Offset
+098H

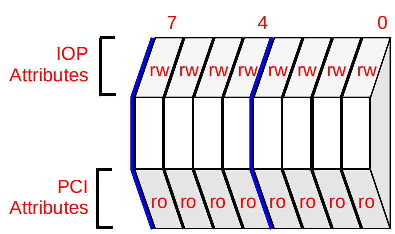
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
07:00	01H	Cap_Id - This field with its' 01H value identifies this item in the linked list of Extended Capability Headers as being the PCI Power Management Registers.

3.17.51 PM Next Item Pointer Register - PM_Next_Item_Ptr

The Next Item Pointer Register bits adhere to the definitions in the *PCI Local Bus Specification*, Revision 2.3. This register describes the location of the next item in the function's capability list. For the 4138xx, the next capability (MSI-X capability list) is located at off-set B0H. Note that the PM_Next_Item_Ptr can be written by the processor.

Table 191. PM Next Item Pointer Register - PM_Next_Item_Ptr



Internal Bus Address Offset
+099H

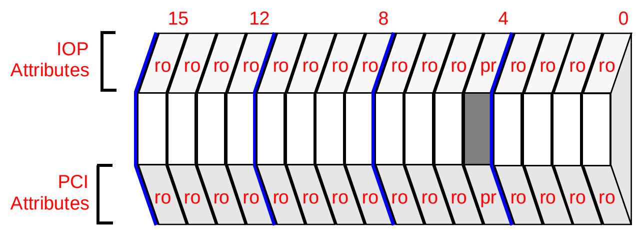
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
07:00	B0H	Next_ Item_ Pointer - Provides an offset into the function's configuration space pointing to the next item in the function's capability list which in the 4138xx is the MSI-X extended capabilities header.

3.17.52 ATU Power Management Capabilities Register - APMCR

Power Management Capabilities bits adhere to the definitions in the *PCI Bus Power Management Interface Specification*, Revision 1.1. This register is a 16-bit read-only register which provides information on the capabilities of the ATU function related to power management.

Table 192. ATU Power Management Capabilities Register - APMCR



Internal Bus Address Offset
+09AH

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

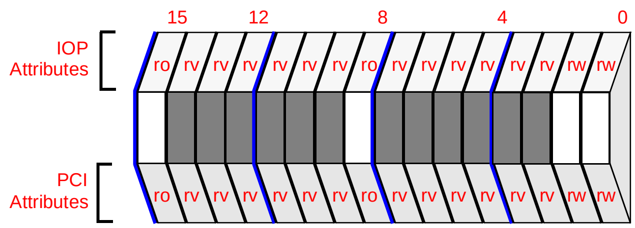
Bit	Default	Description
15:11	00000_2	PME_Support - Not capable of asserting the **PME#** signal in any state, since **PME#** is not supported by the 4138xx.
10	0_2	D2_Support - Set to 0_2 indicating the 4138xx does not support the D2 Power Management State
9	1_2	D1_Support - Set to 1_2 indicating that the 4138xx supports the D1 Power Management State
8:6	000_2	Aux_Current - This field is set to 000_2 indicating that the 4138xx has no current requirements for the 3.3Vaux signal as defined in the *PCI Bus Power Management Interface Specification*, Revision 1.1
5	0_2	DSI - Set to 0_2 meaning that this function does not require a device specific initialization sequence following the transition to the D0 uninitialized state.
4	0_2	Preserved.
3	0_2	PME Clock - Does not apply to PCI Express. Hard-wired 0
2:0	010_2	Version - Setting these bits to 010_2 means that this function complies with *PCI Bus Power Management Interface Specification*, Revision 1.1

3.17.53 ATU Power Management Control/Status Register - APMCSR

Power Management Control/Status bits adhere to the definitions in the *PCI Bus Power Management Interface Specification*, Revision 1.1. This 16-bit register is the control and status interface for the power management extended capability.

Note: Some bits in this register are sticky through reset.

Table 193. ATU Power Management Control/Status Register - APMCSR



Internal Bus Address Offset
+09CH

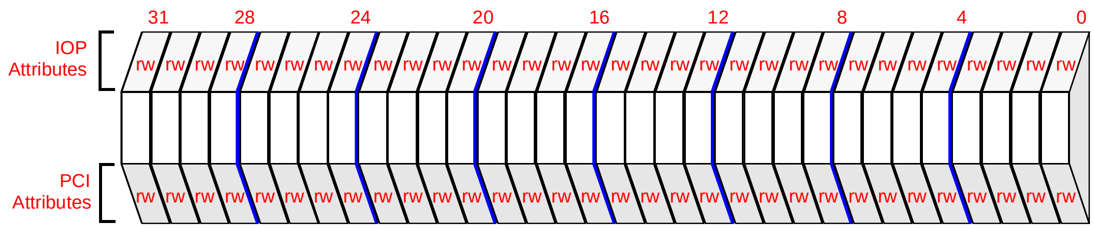
Attribute Legend:	RW = Read/Write
RV = Reserved	RC = Read Clear
PR = Preserved	RO = Read Only
RS = Read/Set	NA = Not Accessible

Bit	Default	Description
15	0_2	PME_Status - This function is not capable of asserting the PME# signal in any state, since **PME##** is not supported by the 4138xx. Hard-wired 0
14:9	00H	Reserved
8	0_2	PME_En - This bit is hard-wired to read-only 0_2 since this function does not support **PME#** generation from any power state.
7:2	000000_2	Reserved
1:0	00_2	Power State - This 2-bit field is used both to determine the current power state of a function and to set the function into a new power state. The definition of the values is: 00_2 - D0 01_2 - D1 10_2 - D2 (Unsupported) 11_2 - D3hot The 4138xx supports the D0, D1, and D3hot states. ***Note:*** A write of 10 to this field is discarded and does not change to power state. Additionally a change of state from D0->D3, D0->D1, D1->D3, or D3->D0 can result in setting bit 16 of the ATUISR.

3.17.54 ATU Scratch Pad Register - ATUSPR

This register can be used for application specific purposes and has no direct impact on the hardware.

Table 194. Scratch Pad Register - ATUSPR



Internal Bus Address Offset
+0CCH

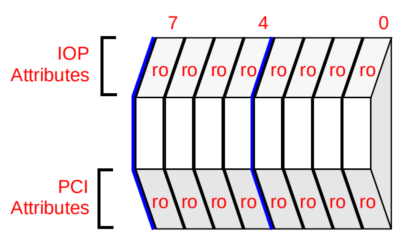
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:0	0000H	Scratch Pad Data - Entire register is available for application specific purposes.

3.17.55 PCI Express Capability List Register - PCIE_CAPID

The Capability Identifier Register bits adhere to the definitions in the *PCI Express Base Specification*, Revision 1.0a. This is the PCI Express Capability List with an ID of 10H as defined by the *PCI-X Protocol Addendum to the PCI Local Bus Specification,* Revision 2.0.

Table 195. PCI Express Capability Identifier Register - PCIE_CAPID



Internal Bus Address Offset
+0D0H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
07:00	10H	Cap_Id - This field with its' 10H value identifies this item in the linked list of Extended Capability Headers as being the PCI Express Capability List registers.

3.17.56 PCI Express Next Item Pointer Register - PCIE_NXTP

The Next Item Pointer Register bits adhere to the definitions in the *PCI Local Bus Specification*, Revision 2.3. This register describes the location of the next item in the function's capability list.

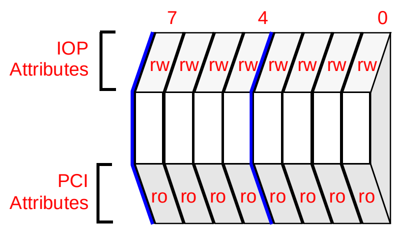
By default, the PCI Express capability is the last capabilities list for the 4138xx, thus this register defaults to 00H.

However, this register may be written to 90H prior to host configuration to include the VPD capability located at off-set 90H.

Warning: Writing this register to any value other than 00H (default) or 90H is not supported and may produce unpredictable system behavior.

In order to insure that this register is written prior to host configuration, the 4138xx must be initialized at **P_RST#** assertion to Retry Type 0 configuration cycles (bit 2 of PCSR). Typically, the Intel XScale® processor would be enabled to boot immediately following **P_RST#** assertion in this case (bit 1 of PCSR), as well. Please see Section 3.17.41, "PCI Configuration and Status Register - PCSR" on page 327 for more details on the 4138xx's initialization modes.

Table 196. PCI Express Next Item Pointer Register - PCIE_NXTP



Internal Bus Address Offset
+0D1H

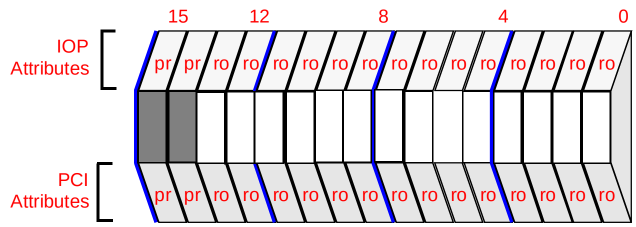
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
07:00	00H	Next_ Item_ Pointer - This field provides an offset into the function's configuration space pointing to the next item in the function's capability list. Since the PCI Express capabilities are the last in the linked list of extended capabilities in the 4138xx, the register is set to 00H. However, this field may be written prior to host configuration with 90H to extend the list to include the VPD extended capabilities header.

3.17.57 PCI Express Capabilities Register - PCIE_CAP

This register controls various modes and features of ATU and Message Unit when operating in the PCI Express mode.

Table 197. PCI Express Capabilities Register PCIE_CAP



Internal Bus Address Offset
+0D2H

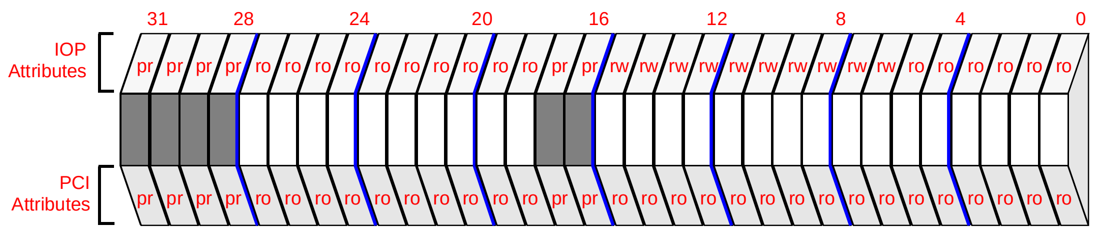
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
15:14	00_2	Preserved
13:9	00000	Interrupt Message Number - 4138xx uses a single interrupt message. This register is hardcoded to 0.
8	Endpoint: 0 Root Complex: 1	Slot Implemented - Indicates that the PCI Express Link associated with this port is connected to a slot. Only valid for root complex and switch downstream ports. Hard-wired to 0 Set to 1 for root complex this bit is initialized via strapping options.
7:4	0000 or 0100	Device/Port Type - Indicates the type of PCI Express logical device. 0000b - PCI Express Endpoint device 0100b - Root Port of PCI Express Root Complex these bits are initialized via strapping options.
3:0	1H	Capability Version - Indicates PCI-SIG defined PCI Express capability structure version number 4138xx supports version 1h.

3.17.58 PCI Express Device Capabilities Register - PCIE_DCAP

This register identifies the capabilities and current operating mode of ATU, DMAs and Message Unit when operating in the PCI Express mode.

Table 198. PCI Express Device Capabilities Register - PCIE_DCAP



Internal Bus Address Offset
+0D4H

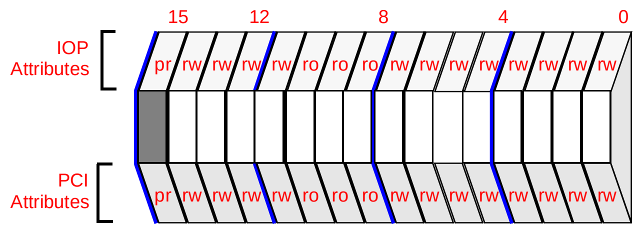
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:28	00_2	Preserved
27:26	00	Captured Slot Power Limit Scale - Specifies the scale used for the Slot Power Limit Value. This value is set when the Set_Slot_Power_Limit message is received.
25:18	00H	Captured Slot Power Limit Value - In combination with the Slot Power Limit Scale value, specifies the upper limit on power supplied by slot. This value is set when the Set_Slot_Power_Limit message is received.
17:16	00	Preserved
15	1	Role-Based Error Reporting - this bit is set to indicate that this device implements the Role Base Error Reporting defined in *PCI Express Base Specification*, Revision 1.1.
14	0	Power Indicator Present on Device - When set, indicates that a Power Indicator is implemented on the card or module.
13	0	Attention Indicator Preset on Device - When set, indicates that an Attention Indicator is implemented on the card or module
12	0	Attention Button Present on Device - When set, indicates that an Attention Button is implemented on the card or module
11:9	000	Endpoint L1 Acceptable Latency - 4138xx does not support L1 active state power management.
8:6	111	Endpoint L0 Acceptable Latency -Total acceptable latency that 4138xx can withstand due to a transition from L0s to L0 state.
5	0	Extended Tag Field Supported - Indicates the maximum supported size of the Tag field as a Requester. 4138xx does not generate 8-bit Tags but supports 8-bit Tags as a completer.
4:3	00	Phantom Functions Supported The ATU does not use phantom functions to extend the number of outstanding requests.
2:0	010	Max Payload Size Supported - Indicates that 4138xx can support a max payload of 512B

3.17.59 PCI Express Device Control Register - PE_DCTL

This register controls various modes and features of ATU and Message Unit when operating in the PCI Express mode.

Table 199. PCI Express Device Control Register - PE_DCTL (Sheet 1 of 2)

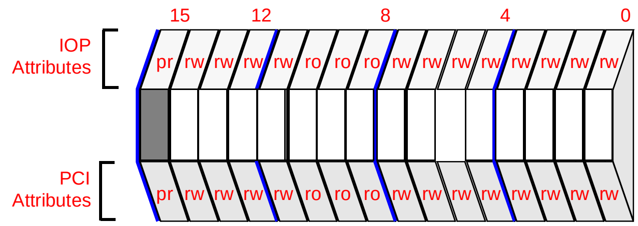


Internal Bus Address Offset
+0D8H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
15	0_2	Preserved
14:12	010_2	Max_Read_Request_Size – This field sets the maximum Read Request size for the Device as a Requester. The Device must not generate read requests with size exceeding the set value. Defined encodings for this field are: 000b 128B max read request size 001b 256B max read request size 010b 512B max read request size 011b 1024B max read request size 100b 2048B max read request size 101b 4096B max read request size 110b Reserved 111b Reserved Any reserved value is treated as 4096B. ***Note:*** In a multifunction configuration, the minimum programmed value from all functions is used when issuing requests.
11	1	Enable No Snoop
10	0	Aux Power PM Enable - The ATU does not utilize Auxiliary power. Hard-wired to 0.
9	0	Phantom Functions Enable - 4138xx does not use phantom functions. Hard-wired to 0.
8	0	Extended Tag Field Enable - 4138xx does not generate 8 bit tags. Hard-wired to 0.
7:5	000	Max_Payload_Size – This field sets maximum TLP payload size for the device. As a receiver, the device must handle TLPs as large as the set value; as transmitter, the device must not generate TLPs exceeding the set value. Defined encodings for this field are: 000b 128B max payload size 001b 256B max payload size 010b 512B max payload size 011b 1024B max payload size (Unsupported) 100b 2048B max payload size (Unsupported) 101b 4096B max payload size (Unsupported) 110b Reserved 111b Reserved Any unsupported or reserved value is treated as 128B. ***Note:*** In a multifunction configuration, the minimum programmed value from all functions is used when transmitting packets, and checking for max_payload violations.
4	1	Enable Relaxed Ordering
3	0	Unsupported Request Reporting Enable – This bit in conjunction with other bits controls the signaling of Unsupported Requests by sending Error Messages. For a multi-function device, this bit controls error reporting from the point-of-view of the respective function.

Table 199. PCI Express Device Control Register - PE_DCTL (Sheet 2 of 2)



Internal Bus Address Offset
+0D8H

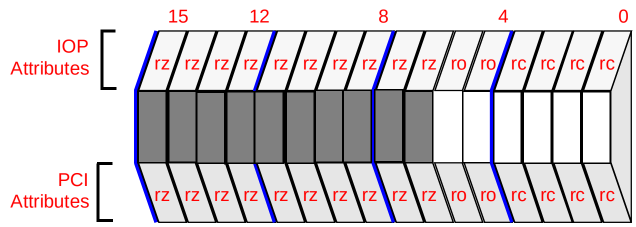
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
2	0	Fatal Error Reporting Enable – This bit in conjunction with other bits control sending ERR_FATAL messages. For a multi-function device, this bit controls error reporting for each function from the point-of-view of the respective function. For a Root Port, the reporting of fatal errors is internal to the root. No external ERR_FATAL message is generated.
1	0	Non-Fatal Error Reporting Enable – This bit in conjunction with other bits controls sending ERR_NONFATAL messages. For a multi-function device, this bit controls error reporting from the point-of-view of the respective function. For a Root Port, the reporting of non-fatal errors is internal to the root. No external ERR_NONFATAL message is generated.
0	0	Correctable Error Reporting Enable – This bit in conjunction with other bits controls sending ERR_COR messages. For a multi-function device, this bit controls error reporting from the point-of-view of the respective function. For a Root Port, the reporting of correctable errors is internal to the root. No external ERR_COR message is generated.

3.17.60 PCI Express Device Status Register - PE_DSTS

This register controls various modes and features of ATU and Message Unit when operating in the PCI Express mode.

Table 200. PCI Express Device Status Register PE_DSTS



Internal Bus Address Offset
+0DAH

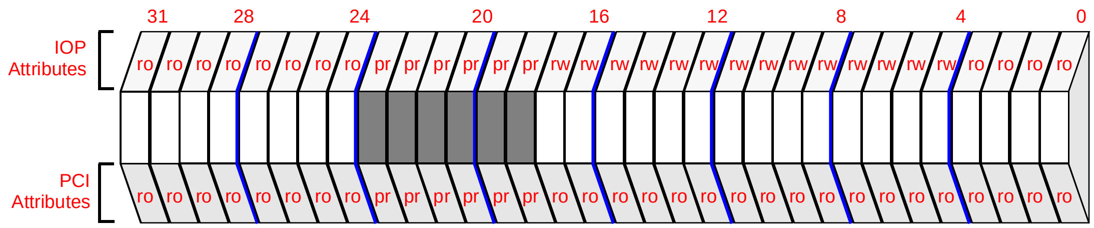
Attribute Legend:
RZ = Reserved Zero
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
15:6	00_2	Reserved Zero - Software must write 0 to these bits.
5	0	Transactions Pending – This bit when set indicates that a device has issued Non-Posted Requests which have not been completed. A device reports this bit cleared only when all Completions for any outstanding Non-Posted Requests have been received.
4	0	AUX Power Detected – ATU does not utilize AUX power. Hard-wired to 0.
3	0	Unsupported Request Detected – This bit indicates that the device received an Unsupported Request. Errors are logged in this register regardless of whether error reporting is enabled or not in the Device Control Register. For a multi-function device, each function indicates status of errors as perceived by the respective function.
2	0	Fatal Error Detected – This bit indicates status of fatal errors detected. Errors are logged in this register regardless of whether error reporting is enabled or not in the Device Control register. For devices supporting Advanced Error Handling, errors are logged in this register regardless of the settings of the correctable error mask register. For a multi-function device, each function indicates status of errors as perceived by the respective function.
1	0	Non-Fatal Error Detected – This bit indicates status of non-fatal errors detected. Errors are logged in this register regardless of whether error reporting is enabled or not in the Device Control register. For devices supporting Advanced Error Handling, errors are logged in this register regardless of the settings of the correctable error mask register. For a multi-function device, each function indicates status of errors as perceived by the respective function.
0	0	Correctable Error Detected – This bit indicates status of correctable errors detected. Errors are logged in this register regardless of whether error reporting is enabled or not in the Device Control register. For devices supporting Advanced Error Handling, errors are logged in this register regardless of the settings of the correctable error mask register. For a multi-function device, each function indicates status of errors as perceived by the respective function.

3.17.61 PCI Express Link Capabilities Register - PE_LCAP

This register identifies the capabilities and current operating mode of ATU, DMAs and Message Unit when operating in the PCI Express mode.

Table 201. PCI Express Link Capabilities Register - PE_LCAP



Internal Bus Address Offset
+0DCH

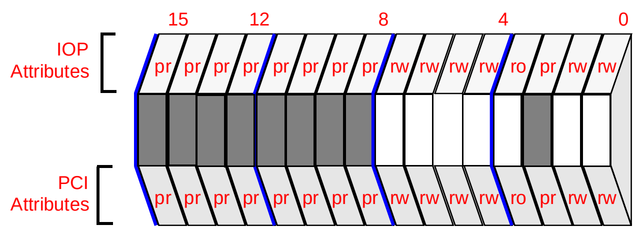
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:24	00H	Port # - PCI Express port number.
23:18	00H	Preserved
17:15	111b	L1 Exit Latency- Active State L1 Transition not supported
14:12	001b	L0s Exit Latency - 64ns - 128ns.
11:10	01b	Active State Link PM Support
9:4	08H	Maximum Link Width - This device supports a maximum width of x8.
3:0	1H	Maximum Link Speed - The PCI Express Link operates at 2.5 Gb/s.

3.17.62 PCI Express Link Control Register - PE_LCTL

This register controls various modes and features of ATU and Message Unit when operating in the PCI Express mode.

Table 202. PCI Express Link Control Register PE_LCTL



Internal Bus Address Offset
+0E0H

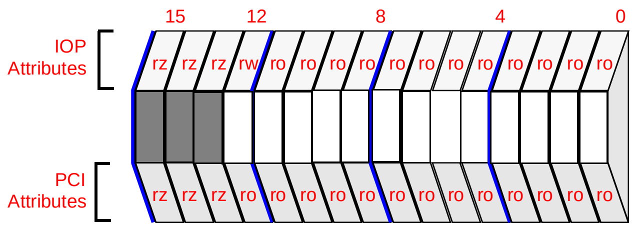
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
15:8	00_2	Preserved
7	0	Extended Synch - This bit when set forces the transmission of 4096 FTS ordered sets in the L0s state followed by a single SKP ordered set prior to entering the L0 state, and the transmission of 1024 TS1 ordered sets in the L1 state prior to entering the Recovery state. This mode provides external devices (e.g., logic analyzers) monitoring the Link time to achieve bit and Symbol lock before the Link enters the L0 or Recovery states and resumes communication.
6	0	Common Clock Configuration - When set indicates that this component and the component at the opposite end of this Link are operating with a distributed common reference clock. This bit used to report the correct L0s and L1 Exit Latencies in the PCIE_LCAP register
5	0	Retrain Link - As an end point, this bit is hard-wired to 0 As a root complex, this bit initiates Link Retraining when set. This bit is self clearing and always returns 0 when read.
4	0	Link Disable - As and endpoint, this bit is hard-wired to 0 As a root complex, this bit disables the Link when set to 1b. Writes to this bit are immediately reflected in the value read from the bit, regardless of actual Link state.
3	Endpoint: 0 Root Complex:1	Read Completion Boundary (RCB) Control - Indicates the Root Complex’s RCB. As an end point, this field is not supported and is hard-wired to 0. For Root Complex, Hard-wired to 1b indicting 128Byte RCB
2	0	Reserved
1:0	00	Active State PM Control - This field controls the level of active state PM supported on the given PCI Express Link.

3.17.63 PCI Express Link Status Register - PE_LSTS

This register controls various modes and features of ATU and Message Unit when operating in the PCI Express mode.

Table 203. PCI Express Link Status Register PE_LSTS



Internal Bus Address Offset
+0E2H

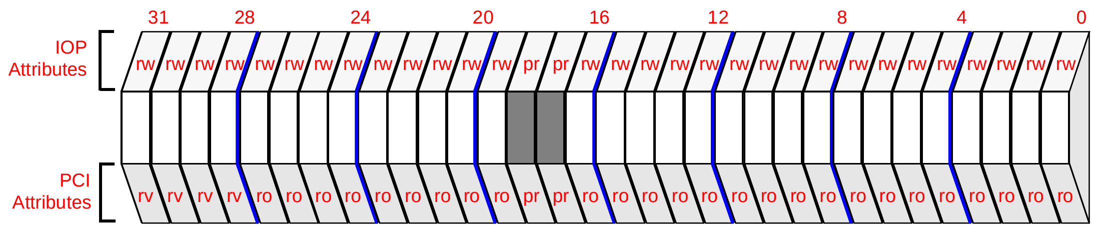
Attribute Legend:	
RZ = Reserved Zero	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
15:13	0H	Reserved
12	1	Slot Clock Configuration - Indicates that the component uses the same physical reference clock that the platform provides on the connector. This bit must be cleared when the device uses an independent reference clock.
11	0	Link Training - As an endpoint, this bit is hard-wired to 0 For root complex, this read-only bit indicates that Link training is in progress. Hardware clears this bit once Link training is complete.
10	0	Link Training Error - As an endpoint, this bit is hard-wired to 0 For root complex, this read-only bit indicates that a Link training error occurred. This bit is cleared by hardware upon successful training of the Link to L0 link state.
9:4	00H	Negotiated Link Width - Defined encodings are 01H x1 02H x2 04H x4 08H x8 12H x12 (Unsupported) 10H x16 (Unsupported) 20H x32 (Unsupported) All other encodings are reserved
3:0	1H	Link Speed - Negotiated Link Speed. 1H indicates 2.5Gb/s Link speed. All other encodings are reserved.

3.17.64 PCI Express Slot Capabilities Register - PE_SCAP

This register identifies PCI Express slot specific capabilities.

Table 204. PCI Express Slot Capabilities Register - PE_SCAP



Internal Bus Address Offset
+0E4H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

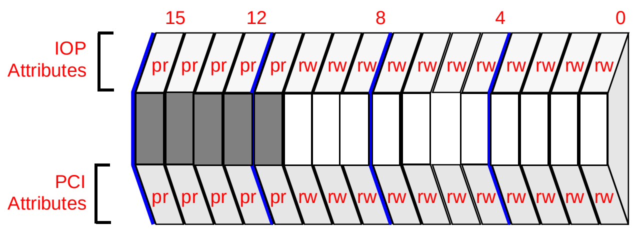
Bit	Default	Description
31:19	0000H	Physical Slot Number This filed indicates the physical slot number attached to the Port. The hardware initialized value must be assigned a slot number that is globally unique within the chassis. These registers should be initialized to 0 for ports connected to devices that are integrated on the system board.
18:17	00b	Reserved
16:15	00b	Slot Power Limit Scale
14:7	00H	Slot Power Limit Value
6	0	Hot-Plug Capable
5	0	Hot-Plug Surprise
4	0	Power Indicator Present
3	0	Attention Indicator Present
2	0	MRL Sensor Present
1	0	Power Controller Present
0	0	Attention Button Present

3.17.65 PCI Express Slot Control Register - PE_SCR

This register controls PCI Express Slot specific parameters.

4138xx does not implement Hot-Plug support for its downstream ports when operating as a root complex. This is left as R/W for the IOP in case a software solution can be implemented using the GPIO pins.

Table 205. PCI Express Slot Control Register PE_SCR



Internal Bus Address Offset
+0E8H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

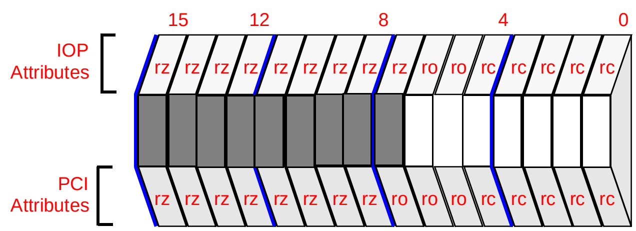
Bit	Default	Description
15:11	0H	Reserved
10	0	Power Controller Control
9:8	00	Power Indicator Control
7:6	00	Attention Indicator Control
5	0	Hot-Plug Interrupt Enable
4	0	Command Completed Interrupt Enable
3	0	Presence Detect Changed Enable
2	0	MRL Sensor Changed Enable
1	0	Power Fault Detected Enable
0	0	Attention Button Power Enable

3.17.66 PCI Express Slot Status Register - PE_SSTS

This register provides information about PCI Express Slot specific parameters.

4138xx does not implement Hot-Plug support for its downstream ports when operating as a root complex. This is left as R/W for the IOP in case a software solution can be implemented using the GPIO pins.

Table 206. PCI Express Slot Status Register PE_SSTS



Internal Bus Address Offset
+0EAH

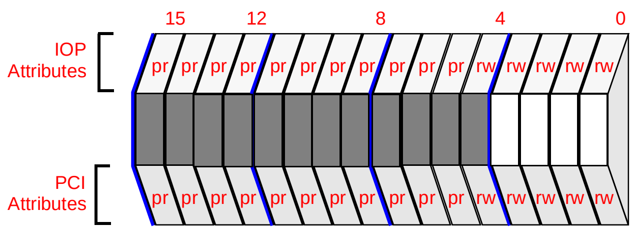
Attribute Legend:
RZ = Reserved Zero
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
15:7	0H	Reserved Zero
6	0	Presence Detect State
5	0	MRL Sensor State
4	0	Command Completed
3	0	Presence Detect Changed
2	0	MRL Sensor Changed
1	0	Power Fault Detected
0	0	Attention Button Pressed

3.17.67 PCI Express Root Control Register - PE_RCR

This register controls PCI Express Slot specific parameters.

Table 207. PCI Express Root Control Register - PE_RCR



Internal Bus Address Offset
+0ECH

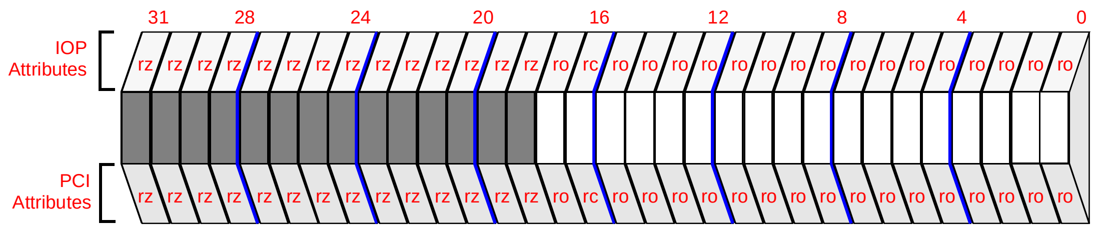
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
15:4	0H	Reserved
3	0	PME Interrupt Enable This bit when set enables interrupt generation upon receipt of a PME message as reflected in the PME Status register bit. A PME Interrupt is also generated when the PME Status register bit is set when this bit is set from a cleared state.
2	0	System Error on Fatal Error Enable When set, the ATU generates the ATU_SERR interrupt when a fatal error (ERR_FATAL) message is received or a fatal error is detected by ATU. This is only valid when operating as the root complex
1	0	System Error on Non-Fatal Error Enable When set, the ATU generates the ATU_SERR interrupt when a non-fatal error (ERR_NONFATAL) message is received or a non-fatal error is detected by ATU. This is only valid when operating as the root complex
0	0	System Error on Correctable Error Enable When set, the ATU generates the ATU_SERR interrupt when a correctable error (ERR_COR) message is received or a correctable error is detected by ATU. This is only valid when operating as the root complex

3.17.68 PCI Express Root Status Register - PE_RSR

The Root Statue Register provides information about PCI Express device specific parameters.

Table 208. PCI Express Root Status Register PE_RSR



Internal Bus Address Offset
+0F0H

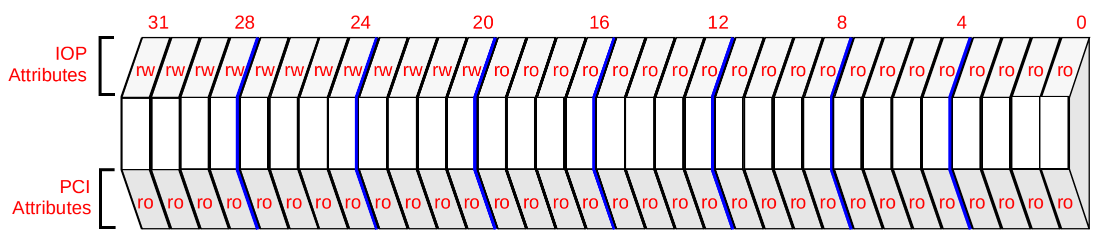
Attribute Legend:
RZ = Reserved Zero
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:18	0000H	Reserved Zero - Software must write 0 to these bits
17	0	PME Pending: The read-only bit indicates that another PME is pending when the PME Status bit is set. When the PME Status bit is cleared by software; the PME is delivered by hardware by setting the PME Status bit again and updating the Requester ID appropriately. The ATU only supports a single PME at a time. This bit hard-wired to 0.
16	0	PME Status: This bit indicate that PME was asserted by the requestor ID indicated in the PME requestor ID field. Subsequent PMEs are dropped until the status register is cleared by software by writing a 1 to this bit.
15:0	0000H	PME Requestor ID This field indicates the PCI requestor ID of the first PME requestor.

3.17.69 PCI Express Advanced Error Capability Identifier - ADVERR_CAPID

This register stores the PCI Express extended capability ID value.

Table 209. PCI Express Advanced Error Capability Identifier - ADVERR_CAPID



Internal Bus Address Offset
+100H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

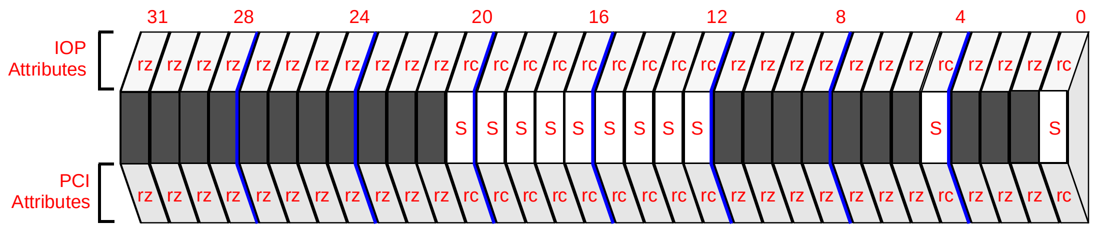
Bit	Default	Description
31:20	000H	Next PCI Express Extended Capability Pointer: This is the last capability. Program with 1E0H to point to the PCI Express Device Serial Number as the next capability, or program with 1F0H to bypass the DSN and point to Power Budgeting as the next capability.
19:16	1H	Advanced Error Capability Version Number: PCI Express Advanced Error Reporting Extended Capability Version Number.
15:0	0001H	Advanced Error Capability ID: PCI Express Extended Capability ID indicating Advanced Error Reporting Capability.

3.17.70 PCI Express Uncorrectable Error Status - ERRUNC_STS

The Uncorrectable Error Status register indicates error detection status of individual uncorrectable errors on a PCI Express device. An individual error status bit that is set to "1" indicates that a particular error was detected; software may clear an error status by writing a 1 to the respective bit.

Note: All bits in this register are sticky through reset.

Table 210. PCI Express Uncorrectable Error Status - ERRUNC_STS



Internal Bus Address Offset
+104H

Attribute Legend:
RZ = Reserved Zero
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

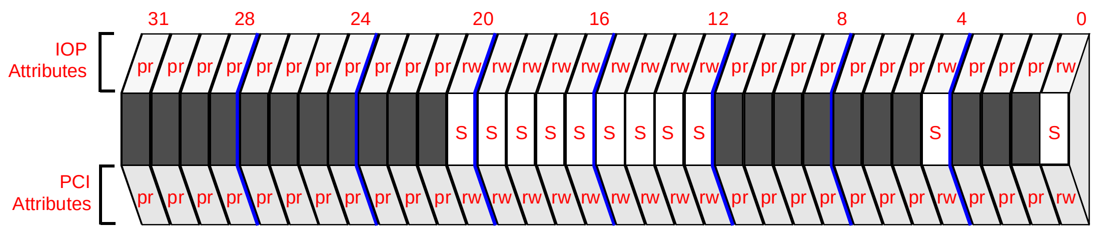
Bit	Default	Description
31:21	0	Reserved Zero - Software must write 0 to these bits.
20	0	Unsupported Request Error Status - As a receiver, Set whenever an unsupported request is detected. The Header is logged.
19	0	ECRC Check - As a receiver, set when ECRC check fails. The Header is logged.
18	0	Malformed TLP: As a receiver, set whenever a malformed TLP is detected. The Header is logged.
17	0	Receiver Overflow: Set when PCI Express receive buffers overflow.
16	0	Unexpected Completion: As a receiver, set whenever a completion is received that does not match the 4138xx's requestor ID or outstanding Tag. The Header is logged.
15	0	Completer Abort: As a completer, set whenever an internal agent signals a data abort. The header is logged.
14	0	Completion Timeout: As a requester, set whenever an outbound Non Posted Request does not receive a completion within 16-32ms.
13	0	Flow Control Protocol Error Status: Set whenever a flow control protocol error is detected.
12	0	Poisoned TLP Received: As a receiver, set whenever a poisoned TLP is received from PCI Express. The header is logged.
11:5	0	Reserved Zero - Software must write 0 to these bits.
4	0	Data Link Protocol Error: Set whenever a data link protocol error is detected.
3:1	0	Reserved Zero - Software must write 0 to these bits.
0	0	Training Error - Set whenever a training error is detected.

3.17.71 PCI Express Uncorrectable Error Mask - ERRUNC_MSK

The Uncorrectable Error Mask register controls reporting of individual errors by the device to the PCI Express Root Complex via a PCI Express error message. A masked error (respective bit set to 1b in the mask register) is not logged in the Header Log register, does not update the First Error Pointer, and is not reported to the PCI Express Root Complex by an individual device.

Note: All bits in this register are sticky through reset.

Table 211. PCI Express Uncorrectable Error Mask - ERRUNC_MSK



Internal Bus Address Offset
+108H

Attribute Legend:
RZ = Reserved Zero
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

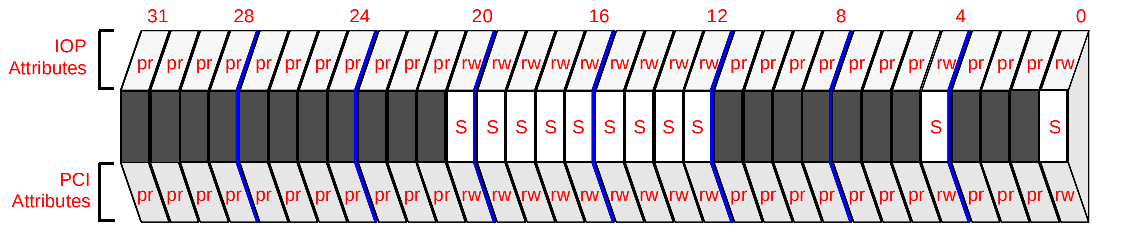
Bit	Default	Description
31:21	0	Preserved.
20	0	Unsupported Request Error Status Error Mask - When '1' error reporting is masked.
19	0	ECRC Check Error Mask - When '1' error reporting is masked.
18	0	Malformed TLP Error Mask - When '1' error reporting is masked.
17	0	Receiver Overflow Error Mask - When '1' error reporting is masked.
16	0	Unexpected Completion Error Mask - When '1' error reporting is masked.
15	0	Completer Abort Error Mask - When '1' error reporting is masked.
14	0	Completion Time Out Error Mask - When '1' error reporting is masked.:
13	0	Flow Control Protocol Error Status Error Mask - When '1' error reporting is masked.
12	0	Poisoned TLP Received Error Mask - When '1' error reporting is masked.
11:5	0	Preserved.
4	0	Data Link Protocol Error Mask - When '1' error reporting is masked.
3:1	0	Preserved.
0	0	Training Error Mask - When '1' error reporting is masked.

3.17.72 PCI Express Uncorrectable Error Severity - ERRUNC_SEV

The Uncorrectable Error Severity register controls whether an individual uncorrectable error is reported as a non-fatal or fatal error. An error is reported as fatal when the corresponding error bit in the severity register is set. When the bit is cleared, the corresponding error is considered non-fatal.

Note: All bits in this register are sticky through reset.

Table 212. PCI Express Uncorrectable Error Severity - ERRUNC_SEV



Internal Bus Address Offset
+10CH

Attribute Legend:
RZ = Reserved Zero
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

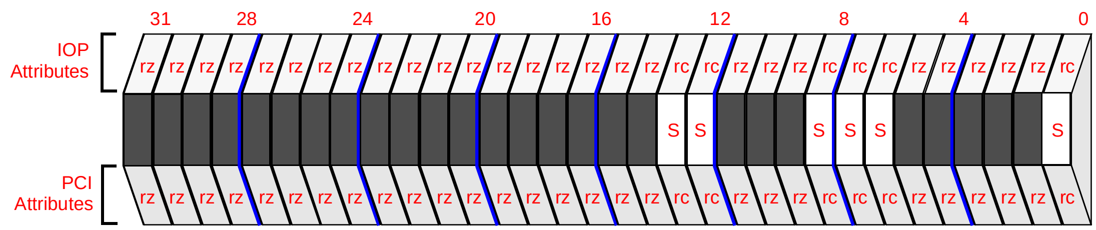
Bit	Default	Description
31:21	0	Preserved
20	0	Unsupported Request Error Status Severity
19	0	ECRC Check Severity
18	1	Malformed TLP Severity
17	1	Receiver Overflow Severity
16	0	Unexpected Completion Severity
15	0	Completer Abort Severity
14	0	Completion Time Out Severity
13	1	Flow Control Protocol Error Status Severity
12	0	Poisoned TLP Received Severity
11:5	0	Preserved
4	1	Data Link Protocol Error Severity
3:1	0	Preserved
0	1	Training Error Severity

3.17.73 PCI Express Correctable Error Status - ERRCOR_STS

The Correctable Error Status register reports error status of individual correctable error sources on a PCI Express device. When an individual error status bit is set to “1” it indicates that a particular error occurred; software may clear an error status by writing a 1 to the respective bit

Note: All bits in this register are sticky through reset.

Table 213. PCI Express Correctable Error Status - ERRCOR_STS



Internal Bus Address Offset
+110H

Attribute Legend:
RZ = Reserved Zero
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

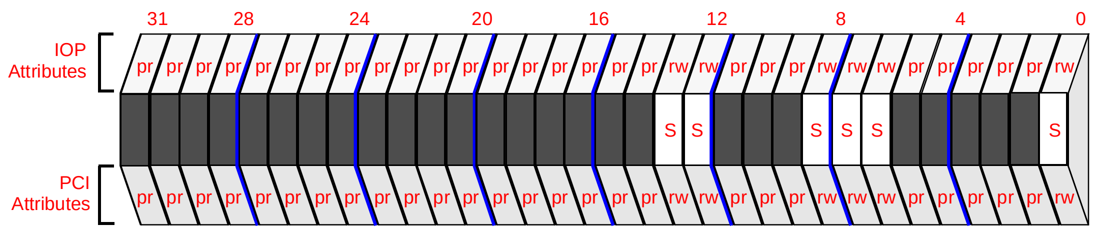
Bit	Default	Description
31:14	0	Reserved - Software must write 0 to these bits.
13	0	Advisory Non-Fatal Error Status
12	0	Replay Timer Timeout Status: Set whenever a replay timer timeout occurs.
11:9	0	Reserved - Software must write 0 to these bits.
8	0	REPLAY_NUM Rollover Status: Set whenever the replay number rolls over from 11 to 00.
7	0	Bad DLLP Status: Sets this bit on CRC errors on DLLP.
6	0	Bad TLP Status: Sets this bit on CRC errors or sequence number out of range on TLP.
5:1	0	Reserved - Software must write 0 to these bits.
0	0	Receiver Error Status: Set whenever the physical layer detects a receiver error.

3.17.74 PCI Express Correctable Error Mask - ERRCOR_MSK

The Correctable Error Mask register controls reporting of individual correctable errors via ERR_COR message. A masked error (respective bit set in mask register) is not reported to the PCI Express Root Complex. There is a mask bit per error bit in the Correctable Error Status register.

Note: All bits in this register are sticky through reset.

Table 214. PCI Express Correctable Error Mask - ERRCOR_MSK



Internal Bus Address Offset
+114H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

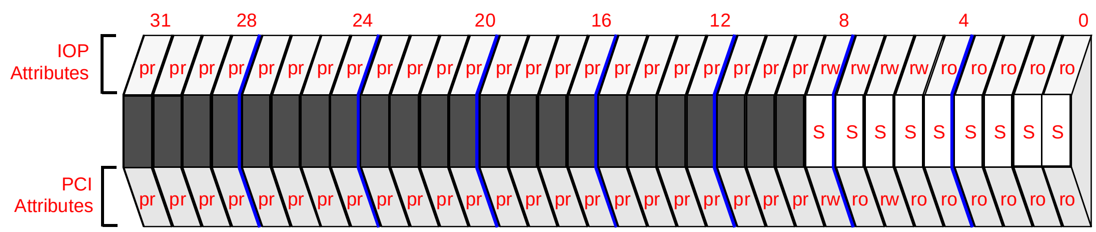
Bit	Default	Description
31:14	0	Preserved.
13	1	Advisory Non-Fatal Error Mask - this bit is set by default to enable compatibility with software that does not comprehend Role-Based Error Reporting.
12	0	Replay Timer Timeout Mask
11:9	0	Preserved.
8	0	REPLAY_NUM Rollover Mask
7	0	Bad DLLP Mask
6	0	Bad TLP Mask
5:1	0	Preserved.
0	0	Receiver Error Mask

3.17.75 Advanced Error Control and Capability Register - ADVERR_CTL

The register gives the status and control for ECRC checks and also the pointer to the first uncorrectable error that happened.

Note: All bits in this register are sticky through reset.

Table 215. Advanced Error Control and Capability Register - ADVERR_CTL



Internal Bus Address Offset
+118H

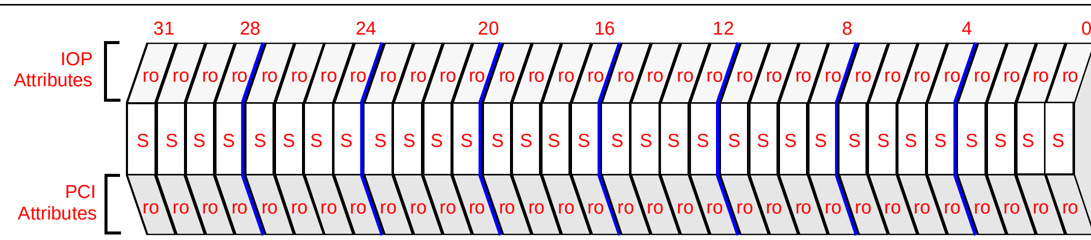
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:9	0	Preserved.
8	0	ECRC Check Enable - When set enables ECRC checking.
7	1	ECRC Check Capable - Indicates the ATU is capable of checking ECRC.
6	0	ECRC Generation Enable - When set enables ECRC generation.
5	1	ECRC Generation Capable - Indicates the ATU is capable of generating ECRC.
4:0	0 0000	The First Error Pointer - Identifies the bit position of the first error reported in the PCI Express Uncorrectable Error Status - ERRUNC_STS register. **Note:** This register does not update until all bits in the ERRUNC_STS register are cleared.

3.17.76 PCI Express Advanced Error Header Log - ADVERR_LOG0

Transaction header log for PCI Express error.

Table 216. PCI Express Advanced Error Header Log - ADVERR_LOG0



Internal Bus Address Offset
+11CH

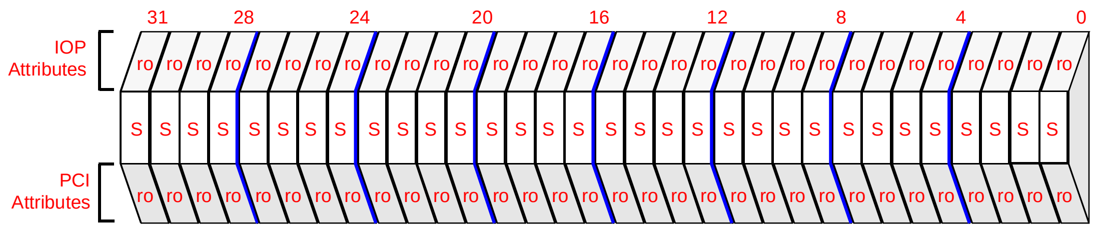
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:0	0	1st DWord of the Header for the PCI Express packet in error. Once an error is logged in this register, it remains locked for further error logging until the time the software clears the status bit that cause the header log i.e. the error pointer is rearmed to log again.

3.17.77 PCI Express Advanced Error Header Log - ADVERR_LOG1

Transaction header log for PCI Express error.

Table 217. PCI Express Advanced Error Header Log - ADVERR_LOG1



Internal Bus Address Offset
+120H

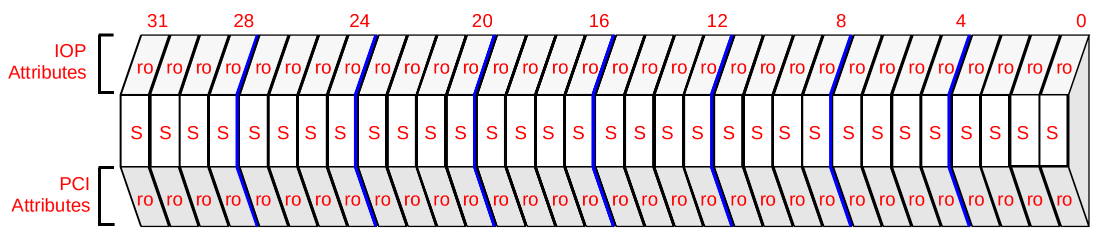
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:0	0	2nd DWord of the Header for the PCI Express packet in error. Once an error is logged in this register, it remains locked for further error logging until the time the software clears the status bit that cause the header log i.e. the error pointer is rearmed to log again.

3.17.78 PCI Express Advanced Error Header Log - ADVERR_LOG2

Transaction header log for PCI Express error.

Table 218. PCI Express Advanced Error Header Log - ADVERR_LOG2



Internal Bus Address Offset
+124H

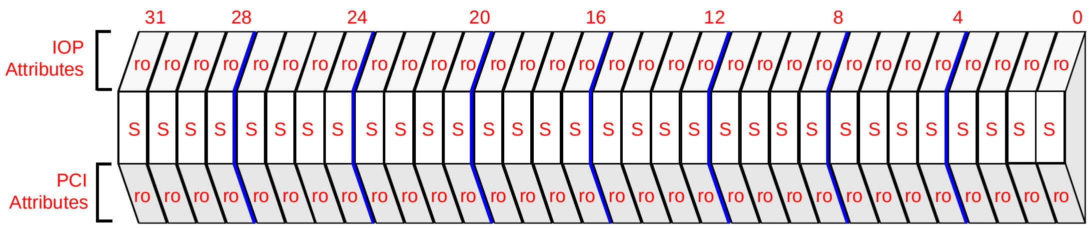
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:0	0	3rd DWord of the Header for the PCI Express packet in error. Once an error is logged in this register, it remains locked for further error logging until the time the software clears the status bit that cause the header log i.e. the error pointer is rearmed to log again.

3.17.79 PCI Express Advanced Error Header Log - ADVERR_LOG3

Transaction header log for PCI Express error.

Table 219. PCI Express Advanced Error Header Log - ADVERR_LOG3



Internal Bus Address Offset
+128H

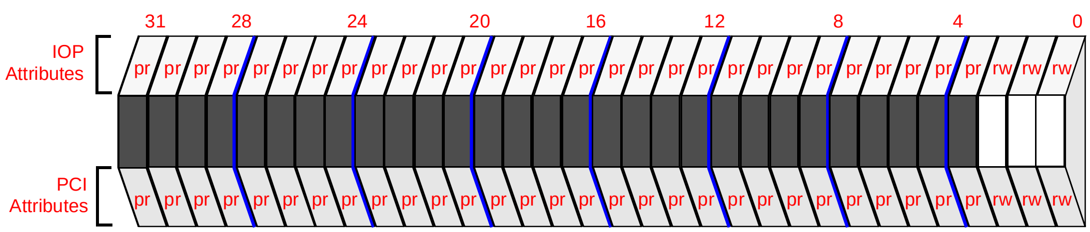
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:0	0	4th DWord of the Header for the PCI Express packet in error. Once an error is logged in this register, it remains locked for further error logging until the time the software clears the status bit that cause the header log i.e. the error pointer is rearmed to log again.

3.17.80 Root Error Command Register - RERR_CMD

The Root Error Command Register is used to notify the Intel XScale® processor in response to PCI Express Error Messages. This bits enable or disable the generation of the ATU Root Complex error interrupt.

Table 220. Root Error Command Register - RERR_CMD



Internal Bus Address Offset
+12CH

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

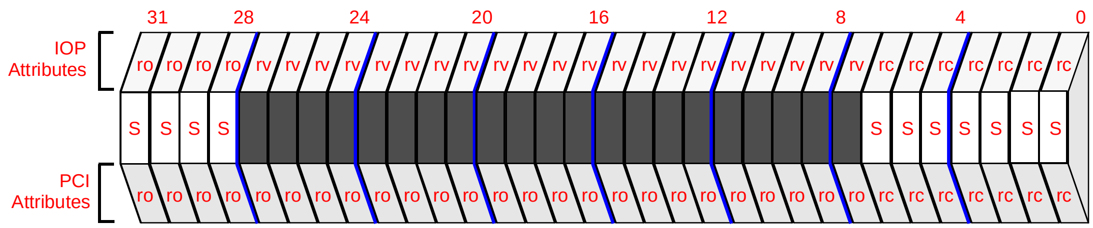
Bit	Default	Description
31:3	0000 0000H	Preserved
2	0	Fatal Error Reporting Enable When set, this bit enables the generation of the ATU Root Complex Error interrupt.
1	0	Non-Fatal Error Reporting Enable When set, this bit enables the generation of the ATU Root Complex Error interrupt
0	0	Correctable Error Reporting When set, this bit enables the generation of the ATU Root Complex Error interrupt

3.17.81 Root Error Status Register

When operating as the root complex of the PCI Express domain, the Root Error Status Register reports the status of error messages received by the ATU, and of errors detected by the ATU. Each correctable and uncorrectable (FATAL/NONFATAL) error source has a first error bit and a next error bit. When an error is received by the ATU, the respective first error bit is set and the Requester ID is logged in the Error Source Identification Register. This register is updated regardless of the settings of the Root Control register and the Root Error Command registers.

Note: All bits in this register are sticky through reset.

Table 221. Root Error Status Register - RERR_SR



Internal Bus Address Offset
+130H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

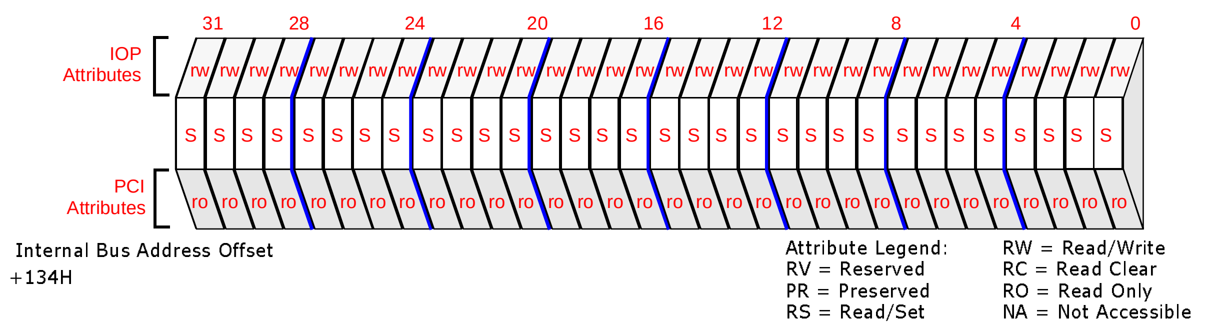
Bit	Default	Description
31:27	0	Advanced Error Interrupt Message Number The ATU signals these interrupts internally to the Intel XScale® processor. This field is hard-wired to 0.
26:7	00 0000H	Reserved
6	0	Fatal Error Message Received This bit set when one or more Fatal Uncorrectable error messages have been received
5	0	Non-Fatal Error Messages Received This bit set when one or more Non-Fatal Uncorrectable error messages have been received
4	0	First Uncorrectable Fatal This bit records the type of the first ERR_FATAL/NONFATAL message. 0 = Indicates the first Uncorrectable Error is NONFATAL 1 = Indicates the first Uncorrectable Error is FATAL ***Note:*** This bit is only valid when the ERR_FATAL/NONFATL Received bit is set.
3	0	Multiple ERR_FATAL/NONFATAL Received Set when a correctable error message is received and the ERR_FATAL/NONFATAL bit is already set.
2	0	ERR_FATAL/NONFATAL Received Set when a correctable error message is received and this bit is not already set.
1	0	Multiple ERR_COR Received Set when a correctable error message is received and the ERR_COR bit is already set.
0	0	ERR_COR Received Set when a correctable error message is received and this bit is not already set.

3.17.82 Error Source Identification Register - RERR_ID

The Root Error Command Register is used to notify the Intel XScale® processor in response to PCI Express Error Messages. This bits enable or disable the generation of the ATU Root Complex error interrupt.

Note: All bits in this register are sticky through reset.

Table 222. Error Source Identification Register RERR_ID

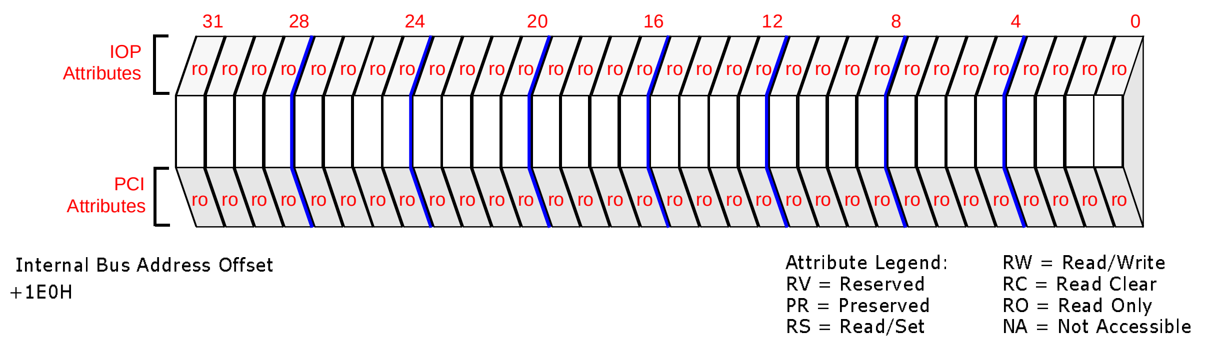


Bit	Default	Description
31:16	0000H	ERR_FATAL/NONFATAL Source Identification Register This filed loaded with the Requester ID indicated in the received ERR_COR message
15:0	0000H	ERR_COR Source Identification Register

3.17.83 Device Serial Number Capability - DSN_CAP

The Device Serial Number is a read-only 64-bit value that is unique for a given PCI Express device.

Table 223. Device Serial Number Capability - DSN_CAP

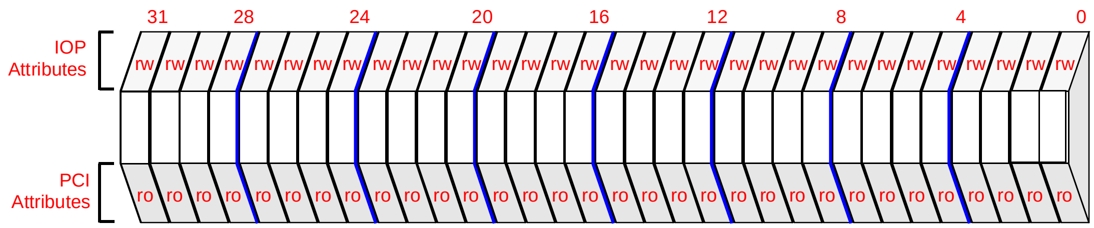


Bit	Default	Description
31:20	1F0H	Next Capability Offset - Points to the PCI Express Power Budgeting Capability as the next capability.
19:16	1H	Capability Version - This field is the PCI-SIG defined version number.
15:0	0003H	PCI Express Extended Capability ID Extended Capability ID for the Device Serial Number Capability is 0003h.

3.17.84 Device Serial Number Lower DW Register - DSN_LDW

The Serial Number register is a 64-bit field that contains the IEEE defined 64-bit extended unique identifier (EUI-64™). This identifier includes a 24-bit company id value assigned by IEEE registration authority and a 40-bit extension identifier assigned by the manufacturer.

Table 224. Device Serial Number Lower DW Register - DSN_LDW



Internal Bus Address Offset
+1E4H

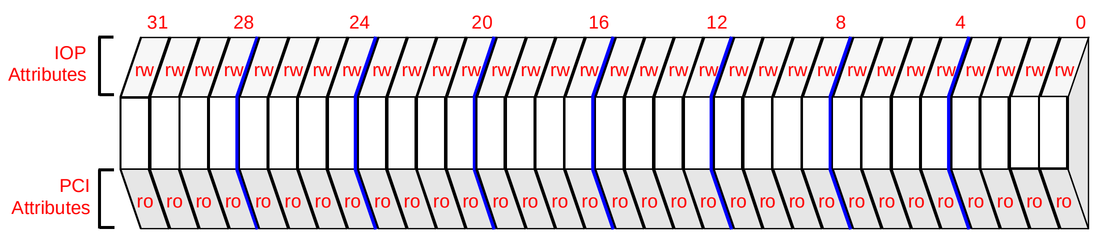
Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
31:0	0H	PCI Express Device Serial Number (Lower DW) This register represents bits 31 to 0 of the EUI-64 identifier.

3.17.85 Device Serial Number Upper DW Register - DSN_UDW

The Serial Number register is a 64-bit field that contains the IEEE defined 64-bit extended unique identifier (EUI-64). This identifier includes a 24-bit company id value assigned by IEEE registration authority and a 40-bit extension identifier assigned by the manufacturer.

Table 225. Device Serial Number Upper DW Register - DSN_UDW



Internal Bus Address Offset
+1E8H

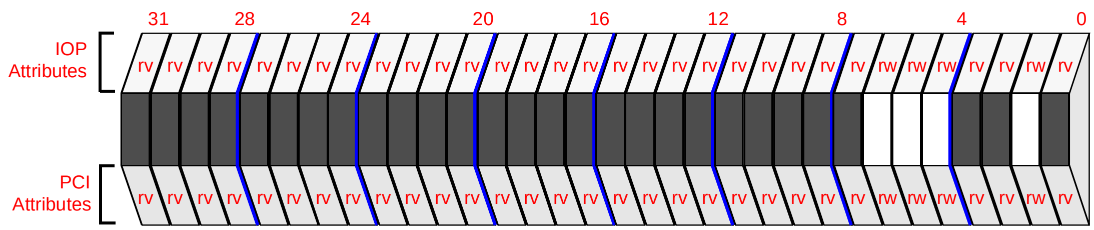
Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
31:0	0H	PCI Express Device Serial Number (Upper DW) This register represents bits 63 to 32 of the EUI-64 identifier.

3.17.86 PCI Express Advisory Error Control Register - PIE_AEC

This registers enables Advisory Error functionality for devices that attempts to recover from certain errors. Firmware can attempt to recover from these non-fatal conditions zero, once, or more (finite) times. Once the retry limit has been reached the generate ERR_NONFATAL bit should be set to alert the host to problem.

Table 226. PCI Express Advisory Error Control Register PIE_AEC



Internal Bus Address Offset
+1ECH

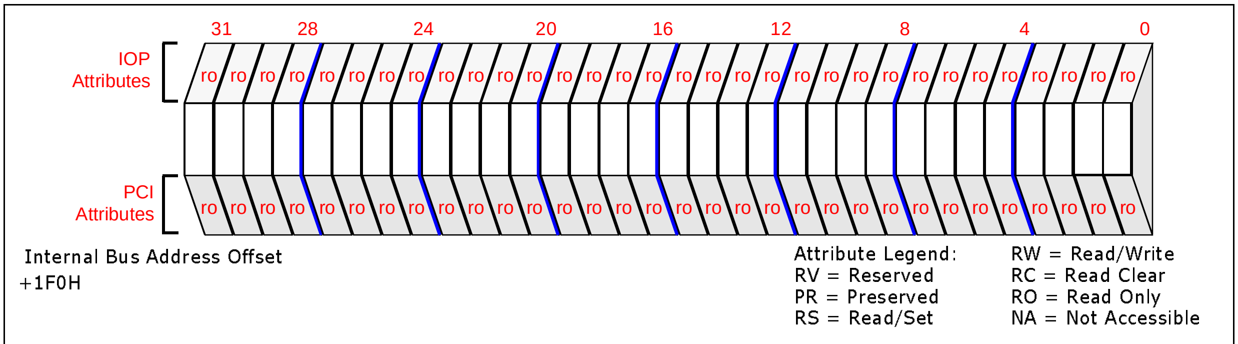
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:07	0	Reserved
6	0	Advisory Error Enable for Completion Timeout - This bit determines the behavior when detecting a completion timeout. 0 = Do not treat as an Advisory Error. Send ERR_NONFATAL to root complex when error is detected 1 = Treat as Advisory Error and send ERR_COR to root complex See Section 3.9.1.8, "Completion Timeout" on page 274 for more details.
5	0	Advisory Error Enable for Poisoned Posted TLPs - This bit determines the behavior when receiving a posted transaction that has poisoned data. 0 = Do not treat as an Advisory Error. Send ERR_NONFATAL to root complex when error is detected 1 = Treat as Advisory Error and send ERR_COR to root complex See Section 3.9.1.7, "Poisoned TLP Received" on page 274 for more details.
4	0	Advisory Error Enable for Poisoned Completions - This bit determines the behavior when receiving a completion transaction that has poisoned data. 0 = Do not treat as an advisory error. Send ERR_NONFATAL to root complex when error is detected 1 = Treat as Advisory Error and send ERR_COR to root complex See Section 3.9.1.7, "Poisoned TLP Received" on page 274 for more details.
3:2	00	Reserved
1	0	Generate ERR_NONFATAL - When asserted an ERR_NONFATAL message is sent to the root complex. This bit is self-clearing and always returns zero when read.
0	0	Reserved

3.17.87 Power Budgeting Enhanced Capability Header - PWRBGT_CAPID

This register defines the power budgeting capability identifier.

Table 227. Power Budgeting Enhanced Capability Header - PWRBGT_CAPID

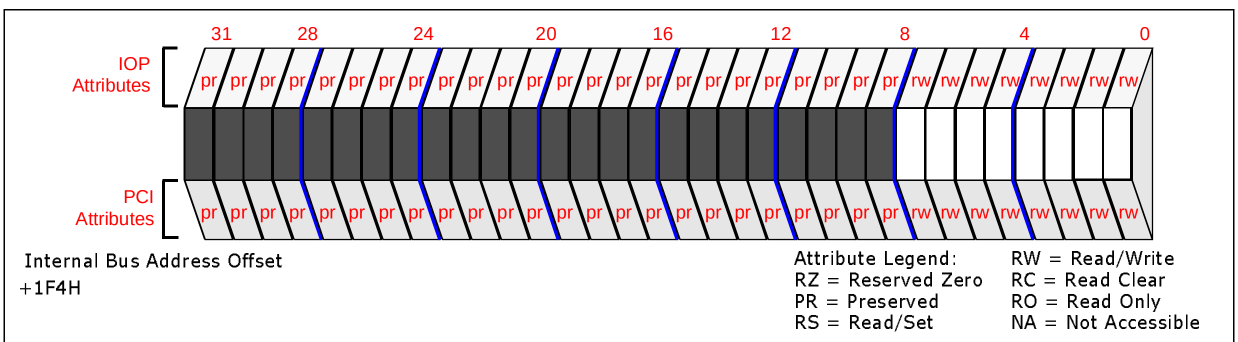


Bit	Default	Description
31:20	000H	Next PCI Express Extended Capability Pointer: This is the last capability.
19:16	1H	Power Budgeting Capability Version Number: PCI Express Power Budgeting Capability Version Number.
15:0	0004H	Power Budgeting Capability ID: PCI Express Power Budgeting Capability ID indicating Advanced Error Reporting Capability.

3.17.88 Power Budgeting Data Select Register - PWRBGT_DSEL

This register defines the power budgeting capability identifier.

Table 228. Power Budgeting Data Select Register - PWRBGT_DSEL

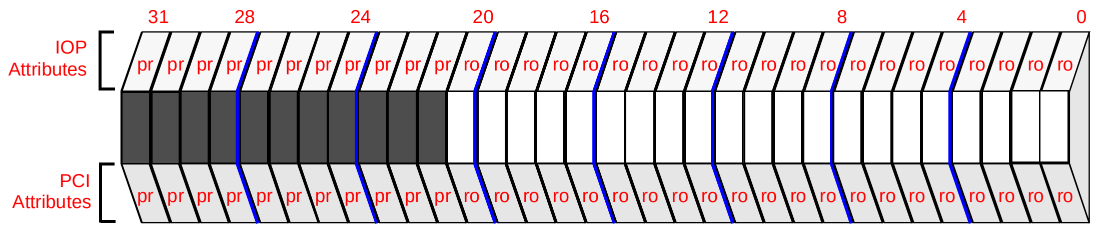


Bit	Default	Description
31:8	00_0000H	Preserved
7:0	00H	Data Select: This read-write register indexes the Power Budgeting Data reported through the Data register and selects the DWORD of Power Budgeting Data that should appear in the Data Register. Index values for this register start at 0 to select the first DWORD of Power Budgeting Data; subsequent DWORDs of Power Budgeting Data are selected by increasing index values. A value of 0 selects the DWORD data starting at address 0x314 to appear in the data register at offset 0x308, a value of 1 selects the DWORD data starting at address 0x318 to appear in the in the data registers at offset 0x308 and so on. Values greater than 23 for this register reports all zeros in the data register.

3.17.89 Power Budgeting Data Register - PWRBGT_DATA

This read-only register returns the DWORD of Power Budgeting Data selected by the Data Select Register. The values of one of the Power Budgeting Information Registers[0:23]—PWRBGT_INFO[0:23] is returned when this register is read.

Table 229. Power Budgeting Data Register - PWRBGT_DATA



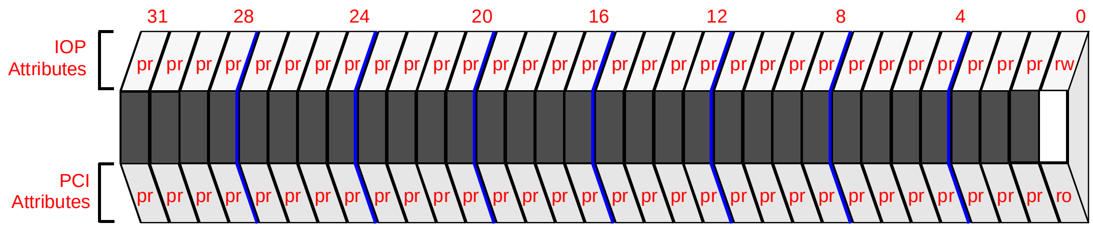
Internal Bus Address Offset
+1F8H

Attribute Legend:
RZ = Reserved Zero
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:21	000H	Preserved
20:18	000	Power Rail - Specifies the power rail of the operating condition being described. Defined encodings are: 000 Power (12V) 001 Power (3.3V) 010 Power (1.8V) 111 Thermal All other encodings are reserved.
17:15	000	Type - Specifies the type of operating condition being described. Defined encodings are: 000 PME Aux 001 Auxiliary 010 Idle 011 Sustained 111 Maximum All other encodings are reserved.
14:13	000	PM State - Specifies the power management state of the operating condition being described. 00 D0 01 D1 10 D2 11 D3 A device returns 11b in this field and Aux or PME Aux in the Type register to specify the D3-Cold PM state. An encoding of 11b with any other Type register value specifies the D3-Hot state.
12:10	000	PM Sub State - Specifies the power management sub state of the operating condition being described. 000Default Sub State 001 - 111Device Specific Sub State
9:8	00	Data Scale - Specifies the scale to apply to the Base Power Value. Defined encodings are: 00 1.0x 01 0.1x 10 0.01x 11 0.001x
7:0	00H	Base Power - Specifies in watts the base power value in the given operating condition. This value must be multiplied by the data scale to produce the actual power consumption value.

3.17.90 Power Budgeting Capability Register - PWRBGT_CAP

Table 230. Power Budgeting Capability Register - PWRBGT_CAP



Internal Bus Address Offset
+1FCH

Attribute Legend:
RZ = Reserved Zero
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:1	00_0000H	Preserved
0	0	System Allocated – This bit when set indicates that the power budget for the device is included within the system power budget. Reported Power Budgeting Data for this device should be ignored by software for power budgeting decisions when this bit is set.

3.17.91 Power Budgeting Information Registers[0:23]—PWRBGT_INFO[0:23]

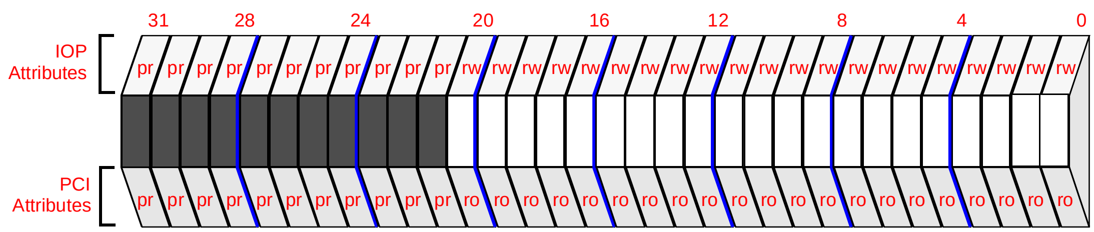
There are 24 power budgeting information registers that are used to report power consumption information for power states as defined in the *PCI Express Base Specification*, Revision 1.0a. These registers are reflected in the Power Budgeting Data Register - PWRBGT_DATA based on the setting of the Data Select field in the Power Budgeting Data Select Register - PWRBGT_DSEL.

Data Select	PWRBGT_INFOx Register	Offset
00h	0	+200H
01h	1	+204H
02h	2	+208H
03h	3	+20CH
04h	4	+210H
05h	5	+214H
06h	6	+218H
07h	7	+21CH
08h	8	+220H
09h	9	+224H
0Ah	10	+228H
0Bh	11	+22CH

Data Select	PWRBGT_INFOx Register	Offset
0Ch	12	+230H
0Dh	13	+234H
0Eh	14	+238H
0Fh	15	+23CH
10h	16	+240H
11h	17	+244H
12h	18	+248H
13h	19	+24CH
14h	20	+250H
15h	21	+254H
16h	22	+258H
17h	23	+25CH

The lower 8 bits of the offset can be determined by shifting the Data Select value left by 2 (i.e. multiply by 4). Currently the default values are all 0, this may change once we get real power/thermal numbers for 4138xx.

Table 231. Power Budgeting Information Registers[0:23]—PWRBGT_INFO[0:23]



Internal Bus Address Offset
+200 through +25FH

Attribute Legend:
RZ = Reserved Zero
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

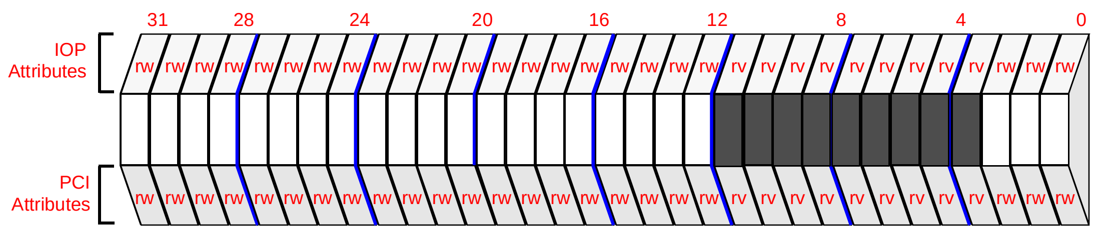
Bit	Default	Description
31:21	000H	Preserved
20:0	00000H	See "Power Budgeting Data Register - PWRBGT_DATA" on page 368 for format.

3.17.92 Outbound I/O Base Address Register - OIOBAR

The OIOBAR register locates the 64 KB I/O cycle address window in the 4138xx's 64 Gbyte internal address space. When A[35:16] of the internal bus address matches the value in OIOBAR, the ATU claims the transaction and forward it over to the PCI interface as an I/O cycle.

Note: In translating the internal bus address A[35:0] for the PCI bus I/O cycle, A[15:0] is forwarded over to the PCI bus unmodified while A[31:16] is replaced with the bits 31:16 from the "Outbound I/O Window Translate Value Register - OIOWTVR".

Table 232. Outbound I/O Base Address Register - OIOBAR



Internal Bus Address Offset
+300H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

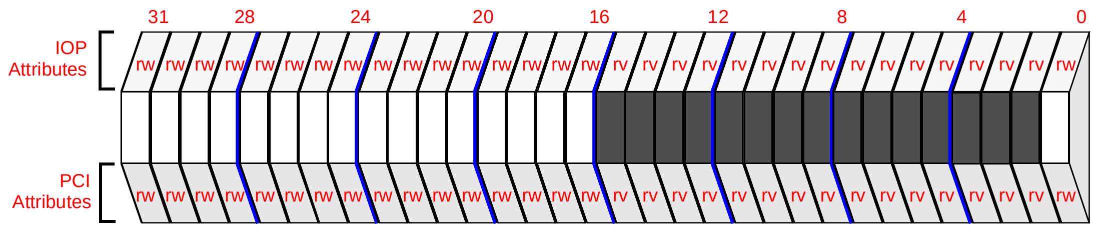
Bit	Default	Description
31:12	0 FFFDH	Outbound I/O Base Address - This value represents bits 35 to 16 of the internal bus address used to claim an outbound I/O cycle request for the ATU.
11:3	000H	Reserved
2:0	000	Outbound I/O Function Number Mapping - The Function number in this field is used as part of the Requestor ID for outbound I/O transactions.

3.17.93 Outbound I/O Window Translate Value Register - OIOWTVR

The Outbound I/O Window Translate Value Register (OIOWTVR) contains the PCI I/O address used to convert the internal bus access to a PCI address. This address is driven on the PCI Express Link as a result of the outbound ATU address translation. See Section 3.3.2.1, "Outbound Address Translation - Internal Bus Transactions" on page 245 for details on outbound address translation.

The I/O window is from 4138xx internal bus is set via the "Outbound I/O Base Address Register - OIOBAR" with the a fixed length of 64 Kbytes.

Table 233. Outbound I/O Window Translate Value Register - OIOWTVR



Internal Bus Address Offset
+304H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

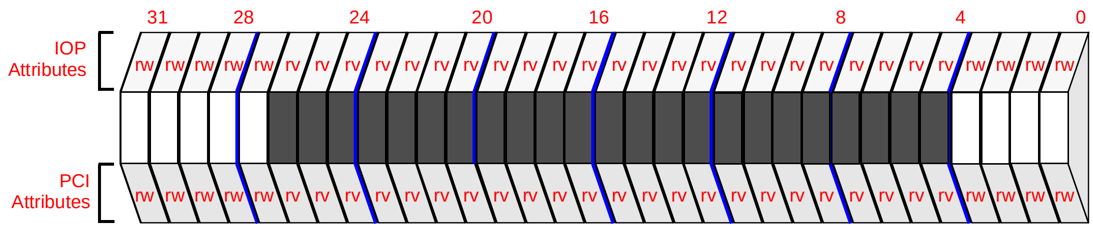
Bit	Default	Description
31:16	0000H	Outbound I/O Window Translate Value - Used to convert internal bus addresses to PCI addresses.
15:01	0000H	Reserved
00	0	Big Endian Byte Swap enable - When set the ATU performs a byte swap on all PCI read/write transactions through OIOBAR. When clear, no swap is performed. Refer to Section 3.4, "Big Endian Byte Swapping" on page 255 for more details.

3.17.94 Outbound Upper Memory Window Base Address Register 0 - OUMBAR0

The OUMBAR0 register locates Outbound Memory Window 0 in a 4 Gbyte Memory section in the 4138xx's 64 Gbyte internal address space. When A[35:32] of the internal bus address matches the value in OUMBAR[3:0], the ATU claims the transaction and forward it over to the PCI Express interface.

Note: In translating the internal bus address A[35:0], A[31:0] is forwarded over to the PCI Express Link unmodified. The ATU constructs a 64 bit PCI address in conjunction with the "Outbound Upper 32-bit Memory Window Translate Value Register 0 - OUMWTVR0" on page 374.

Table 234. Outbound Upper Memory Window Base Address Register 0 - OUMBAR0



Internal Bus Address Offset
+308H

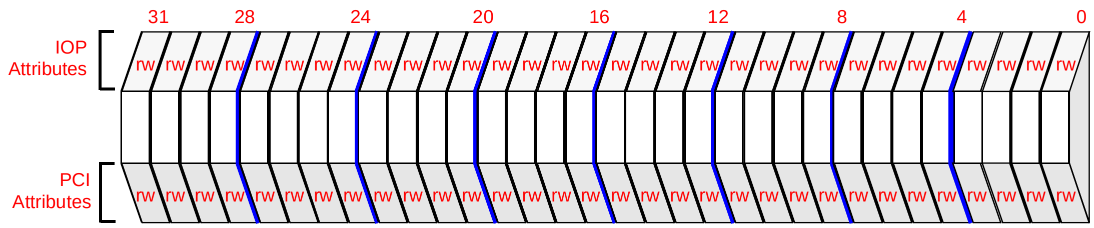
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31	1_2	Outbound Window 0 Enable - When set, the ATU Outbound Memory Window 0 claims internal bus transactions when A[35:32] matches OUMBAR0[3:0].
30:28	000_2	Outbound Window 0 Function Number Mapping - Errors with Outbound Memory Window 0 transactions (PCI or Internal Bus) is associated with the function number contained in this field.
27	0	Big Endian Byte Swap enable - When set the ATU performs a byte swap on all read/write transactions through this BAR. When clear, no swap is performed. Refer to Section 3.4, "Big Endian Byte Swapping" on page 255 for more details.
26:04	000 0000H	Reserved
3:0	1H	Outbound Upper Memory Window Base Address 0 - This value represents bits 35 to 32 of the internal bus address used to claim an outbound memory request. ***Note:*** The ATUX has the same default value. When both ATUs exist and the outbound windows of the both ATUs are enabled, the bit field must be reprogrammed to avoid a conflict.

3.17.95 Outbound Upper 32-bit Memory Window Translate Value Register 0 - OUMWTVR0

The Outbound Upper 32-bit Memory Window Translate Value Register 0 (OUMWTVR0) defines the upper 32-bits of address used during a dual address cycle. This enables the outbound ATU to directly address anywhere within the 64-bit host address space. When this register is all-zero, then a 3DW header is generated on the PCI Express Link.

Table 235. Outbound Upper 32-bit Memory Window Translate Value Register 0- OUMWTVR0



Internal Bus Address Offset
+30CH

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

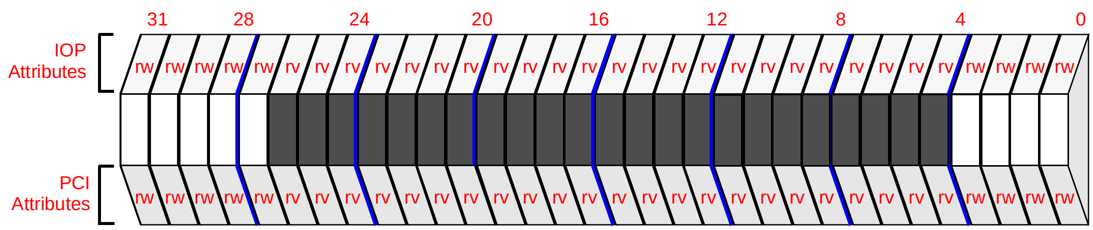
Bit	Default	Description
31:00	0000 0000H	These bits define the upper 32-bits of address driven during the dual address cycle (DAC).

3.17.96 Outbound Upper Memory Window Base Address Register 1 - OUMBAR1

The OUMBAR1 register locates Outbound Memory Window 1 in a 4 Gbyte Memory section in the 4138xx's 64 Gbyte internal address space. When internal bus address A[35:32] matches the value in OUMBAR1, the ATU claims the transaction and forward it over to the PCI interface.

Note: In translating the internal bus address A[35:0], A[31:0] is forwarded over to the PCI Express Link unmodified. The ATU constructs a 64 bit PCI address in conjunction with the "Outbound Upper 32-bit Memory Window Translate Value Register 1 - OUMWTVR1" on page 376.

Table 236. Outbound Upper Memory Window Base Address Register 1 - OUMBAR1



Internal Bus Address Offset
+310H

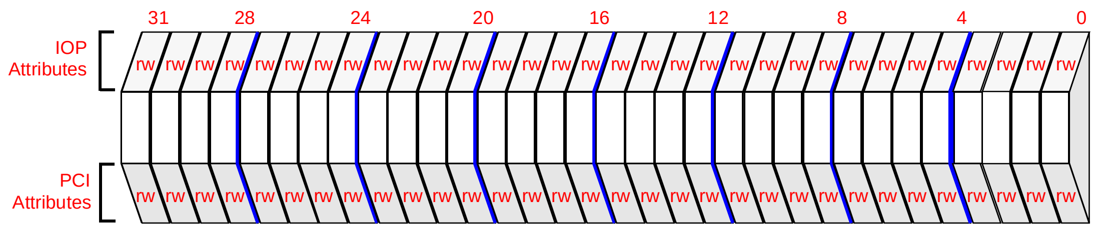
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31	1_2	Outbound Window 1 Enable - When set, the ATU Outbound Memory Window 1 claims internal bus transactions when A[35:32] matches OUMBAR1[3:0].
30:28	000_2	Outbound Window 1 Function Number Mapping - Errors with Outbound Memory Window 1 transactions (PCI or Internal Bus) is associated with the function number contained in this field.
27	0	Big Endian Byte Swap enable - When set the ATU performs a byte swap on all read/write transactions through this BAR. When clear, no swap is performed. Refer to Section 3.4, "Big Endian Byte Swapping" on page 255 for more details.
26:04	000 0000H	Reserved
3:0	2H	Outbound Upper Memory Window Base Address 1 - This value represents bits 35 to 32 of the internal bus address used to claim an outbound memory request. ***Note:*** The ATUX has the same default value. When both ATUs exist and the outbound windows of the both ATUs are enabled, the bit field must be reprogrammed to avoid a conflict.

3.17.97 Outbound Upper 32-bit Memory Window Translate Value Register 1 - OUMWTVR1

The Outbound Upper 32-bit Memory Window Translate Value Register 1 (OUMWTVR1) defines the upper 32-bits of address used during a dual address cycle. This enables the outbound ATU to directly address anywhere within the 64-bit host address space. When this register is all-zero, then a 3DW header is generated on the PCI Express Link.

Table 237. Outbound Upper 32-bit Memory Window Translate Value Register 1- OUMWTVR1



Internal Bus Address Offset
+314H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

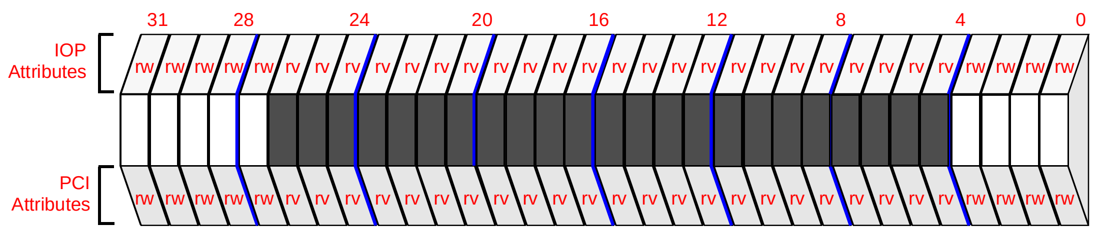
Bit	Default	Description
31:00	0000 0000H	These bits define the upper 32-bits of address driven during the dual address cycle (DAC).

3.17.98 Outbound Upper Memory Window Base Address Register 2 - OUMBAR2

The OUMBAR2 register locates Outbound Memory Window 2 in a 4 Gbyte Memory section in the 4138xx's 64 Gbyte internal address space. When A[35:32] of the internal bus address matches the value in OUMBAR2[3:0], the ATU claims the transaction and forward it over to the PCI interface.

Note: In translating the internal bus address A[35:0], A[31:0] is forwarded over to the PCI bus unmodified. The ATU constructs a 64 bit PCI address in conjunction with the "Outbound Upper 32-bit Memory Window Translate Value Register 2 - OUMWTVR2" on page 378.

Table 238. Outbound Upper Memory Window Base Address Register 2- OUMBAR2



Internal Bus Address Offset
+318H

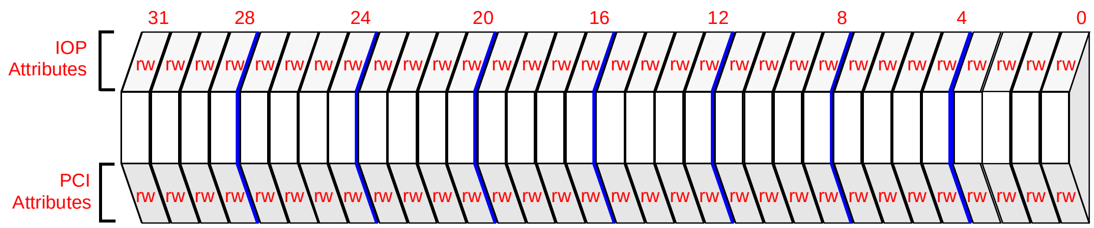
Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
31	0_2	Outbound Window 2 Enable - When set, the ATU Outbound Memory Window 2 claims internal bus transactions when A[35:32] matches OUMBAR2[3:0].
30:28	000_2	Outbound Window 2 Function Number Mapping - Errors with Outbound Memory Window 2 transactions (PCI or Internal Bus) are associated with the function number contained in this field.
27	0	Big Endian Byte Swap enable - When set the ATU performs a byte swap on all read/write transactions through this BAR. When clear, no swap is performed. Refer to Section 3.4, "Big Endian Byte Swapping" on page 255 for more details.
26:04	000 0000H	Reserved
3:0	3H	Outbound Upper Memory Window Base Address 2 - This value represents bits 35 to 32 of the internal bus address used to claim an outbound memory request. **Note:** The ATUX has the same default value. When both ATUs exist and the outbound windows of the both ATUs are enabled, the bit field must be reprogrammed to avoid a conflict.

3.17.99 Outbound Upper 32-bit Memory Window Translate Value Register 2 - OUMWTVR2

The Outbound Upper 32-bit Memory Window Translate Value Register 2 (OUMWTVR2) defines the upper 32-bits of address used during a dual address cycle. This enables the outbound ATU to directly address anywhere within the 64-bit host address space. When this register is all-zero, then a 3DW header is generated on the PCI bus.

Table 239. Outbound Upper 32-bit Memory Window Translate Value Register 2-OUMWTVR2



Internal Bus Address Offset
+31CH

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

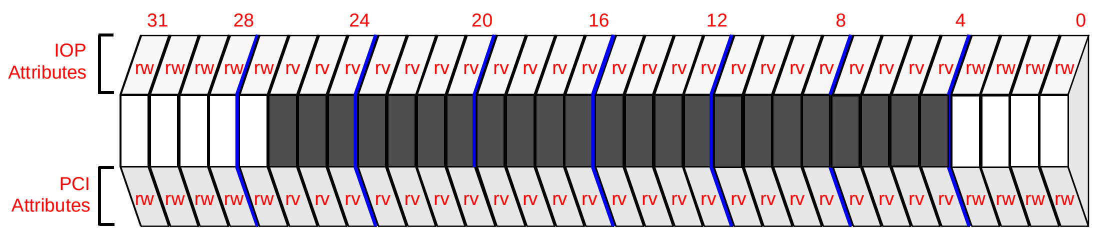
Bit	Default	Description
31:00	0000 0000H	These bits define the upper 32-bits of address.

3.17.100 Outbound Upper Memory Window Base Address Register 3 - OUMBAR3

The OUMBAR3 register locates Outbound Memory Window 3 in a 4 Gbyte Memory section in the 4138xx's 64 Gbyte internal address space. When A[35:32] of the internal bus address matches the value in OUMBAR3[3:0], the ATU claims the transaction and forward it over to the PCI interface.

Note: In translating the internal bus address A[35:0], A[31:0] is forwarded over to the PCI bus unmodified. The ATU constructs a 64 bit PCI address in conjunction with the "Outbound Upper 32-bit Memory Window Translate Value Register 3 - OUMWTVR3" on page 380.

Table 240. Outbound Upper Memory Window Base Address Register 3 - OUMBAR3



Internal Bus Address Offset
+320H

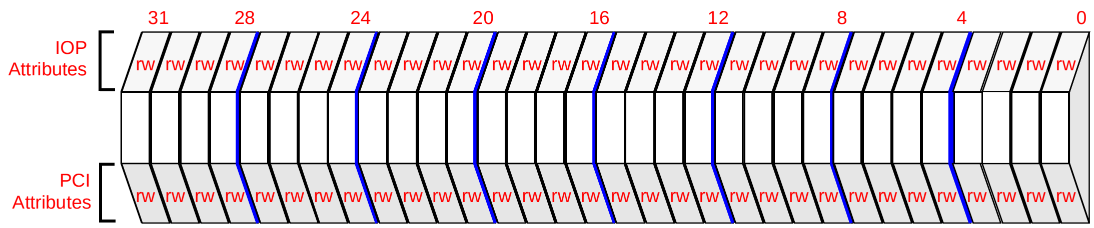
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31	0_2	Outbound Window 3 Enable - When set, the ATU Outbound Memory Window 3 claims internal bus transactions when A[35:32] matches OUMBAR3[3:0].
30:28	000_2	Outbound Window 3 Function Number Mapping - Errors with Outbound Memory Window 3 transactions (PCI or Internal Bus) are associated with the function number contained in this field.
27	0	Big Endian Byte Swap enable - When set the ATU performs a byte swap on all read/write transactions through this BAR. When clear, no swap is performed. Refer to Section 3.4, "Big Endian Byte Swapping" on page 255 for more details.
26:04	000 0000H	Reserved
3:0	4H	Outbound Upper Memory Window Base Address 3 - This value represents bits 35 to 32 of the internal bus address used to claim an outbound memory request. ***Note:*** The ATUX has the same default value. When both ATUs exist and the outbound windows of the both ATUs are enabled, the bit field must be reprogrammed to avoid a conflict.

3.17.101 Outbound Upper 32-bit Memory Window Translate Value Register 3 - OUMWTVR3

The Outbound Upper 32-bit Memory Window Translate Value Register 3 (OUMWTVR3) defines the upper 32-bits of address used during a dual address cycle. This enables the outbound ATU to directly address anywhere within the 64-bit host address space. When this register is all-zero, then a 3DW header is generated on the PCI bus.

Table 241. Outbound Upper 32-bit Memory Window Translate Value Register 3-OUMWTVR3



Internal Bus Address Offset
+324H

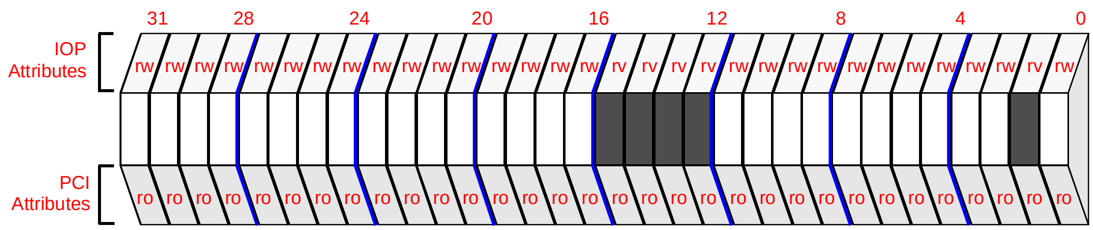
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:00	0000 0000H	These bits define the upper 32-bits of address.

3.17.102 Outbound Configuration Cycle Address Register - OCCAR

The Outbound Configuration Cycle Address Register is used to hold bytes 8-11 of the configuration transaction header. The Intel XScale® processor writes the bus, device, function, and register number which then enables the outbound configuration read or write. The Intel XScale® processor then performs a read or write to the Outbound Configuration Cycle Data Register to initiate the configuration transaction on the PCI Express Link.

Table 242. Outbound Configuration Cycle Address Register - OCCAR



Internal Bus Address Offset
+32CH

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:24	00H	Bus Number
23:19	00H	Device Number
18:16	000b	Function Number
15:12	0H	Reserved
11:8	0H	Extended Register Number
7:2	00H	Register Number
1	0H	Reserved
0	0	Configuration Type 0 = Generate Type 0 Configuration TLP 1 = Generate Type 1 Configuration TLP

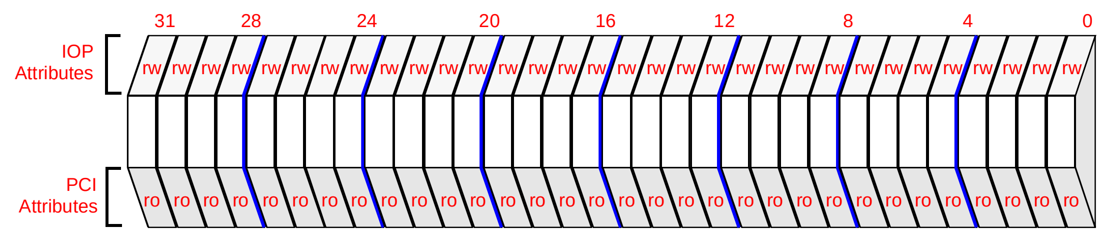
3.17.103 Outbound Configuration Cycle Data Register - OCCDR

The Outbound Configuration Cycle Data Register is used to initiate a configuration read or write transaction on the PCI Express Link. The register is logical rather than physical meaning that it is an address not a register. The Intel XScale® processor reads or writes the data registers memory-mapped address to initiate the configuration transaction on the PCI Express Link with the address found in the OCCAR. For a configuration write, the data is latched from the internal bus and forwarded directly to the ONPQ. For a read, the data is returned directly from the ICPLDQ to the Intel XScale® processor and is never actually entered into the data register (which does not physically exist).

The OCCDR is only visible from 4138xx internal bus address space and appears as a reserved value within the ATU configuration space.

Note: This register does not physically exist and reads from the PCI domain returns ‘0’.

Table 243. Outbound Configuration Cycle Data Register - OCCDR



Internal Bus Address Offset
+330H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

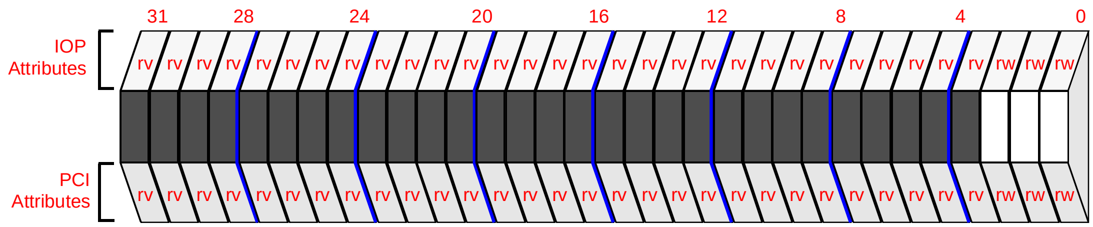
Bit	Default	Description
31:00	0000 0000H	Configuration Cycle Data - These bits define the data used during an outbound configuration read or write transaction.

3.17.104 Outbound Configuration Cycle Function Number - OCCFN

This registers contains the function number that is used as part of the Requester ID for all outbound configuration requests. This field is also used to determine where errors get logged.

Note: For 4138xx the function number should be 0 for endpoint usage and 5 for Root Complex modes.

Table 244. Outbound Configuration Cycle Function Number - OCCFN



Internal Bus Address Offset
+334H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

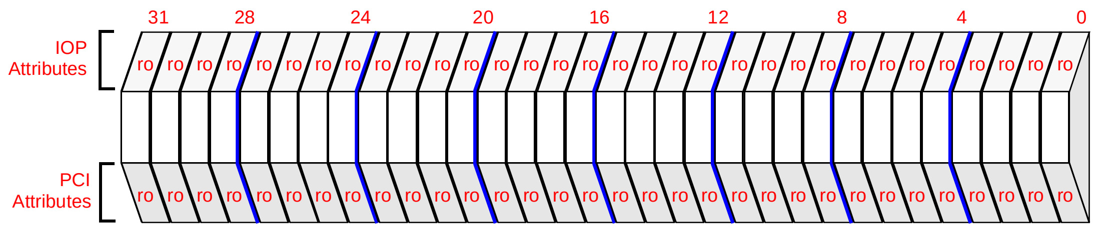
Bit	Default	Description
31:03	0000 0000H	Reserved
02:00	000	Configuration Cycle Function Number - These field is used as part of the Requester ID for outbound configuration cycles.

3.17.105 Inbound Vendor Message Header Register 0 - IVMHR0

The Inbound Vendor Message Header Registers capture the header for a vendor defined message received on the PCI Express interface. Once the inbound message has been processed, the Inbound Vendor Message Received bit is set in the ATU Interrupt Status Register - ATUISR. Subsequent inbound vendor messages are held in the inbound posted queues until the status bit is cleared or the mask bit is set in the ATU Interrupt Mask Register - ATUIMR. When the mask bit is set, then Vendor_Defined Type 0 messages are treated as unsupported requests and Vendor_Defined Type 1 messages are silently discarded.

Vendor_Defined message format is shown below in Figure 38

Table 245. Inbound Vendor Defined Message Header Register0 - IVMHR0



Internal Bus Address Offset
+340H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:24	00H	Header Byte 0
23:16	00H	Header Byte 1
15:8	00H	Header Byte 2
7:0	00H	Header Byte 3

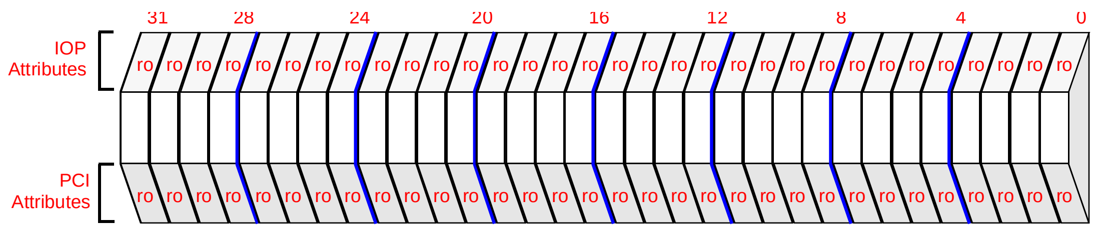
Figure 38. PCI Express Vendor_Defined Message Header

<table>
<tr><td></td><td colspan="8">+0</td><td colspan="8">+1</td><td colspan="8">+2</td><td colspan="8">+3</td></tr>
<tr><td></td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Byte 0></td><td>R</td><td colspan="2">Fmt x 1</td><td colspan="5">Type</td><td>R</td><td colspan="3">TC</td><td colspan="4">R</td><td>TD</td><td>EP</td><td colspan="2">Attr</td><td colspan="2">R</td><td colspan="10">Length</td></tr>
<tr><td>Byte 4></td><td colspan="16">Requester ID</td><td colspan="8">Tag</td><td colspan="8">Message Code - Vendor_Defined</td></tr>
<tr><td rowspan="2">Byte 8></td><td colspan="8">Bus Number</td><td colspan="5">Device Num</td><td colspan="3">Fnc No</td><td colspan="16" rowspan="2">Vendor ID</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>Byte 12></td><td colspan="32">(For Vendor Definition)</td></tr>
</table>

3.17.106 Inbound Vendor Message Header Register 1 - IVMHR1

The Inbound Vendor Message Header Registers capture the header for a vendor defined message received on the PCI Express interface. Once the inbound message has been processed, the Inbound Vendor Message Received bit is set in the ATU Interrupt Status Register - ATUISR. Subsequent inbound vendor messages are held in the inbound posted queues until the status bit is cleared or the mask bit is set in the ATU Interrupt Mask Register - ATUIMR. When the mask bit is set, then Vendor_Defined Type 0 messages are treated as unsupported requests and Vendor_Defined Type 1 messages are silently discarded.

Table 246. Inbound Vendor Defined Message Header Register 1 - IVMHR1



Internal Bus Address Offset
+344H

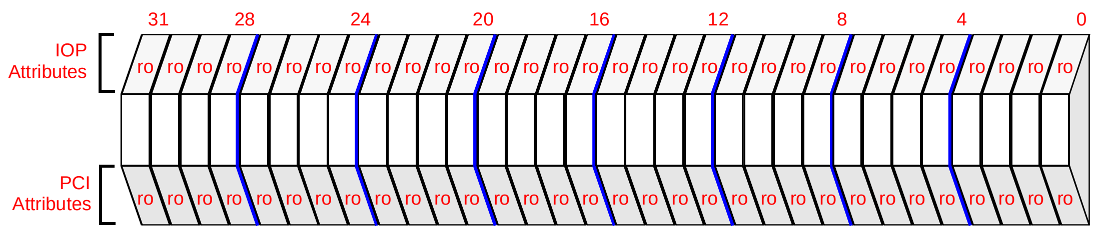
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:24	00H	Header Byte 4
23:16	00H	Header Byte 5
15:8	00H	Header Byte 6
7:0	00H	Header Byte 7

3.17.107 Inbound Vendor Message Header Register 2 - IVMHR2

The Inbound Vendor Message Header Registers capture the header for a vendor defined message received on the PCI Express interface. Once the inbound message has been processed, the Inbound Vendor Message Received bit is set in the ATU Interrupt Status Register - ATUISR. Subsequent inbound vendor messages are held in the inbound posted queues until the status bit is cleared or the mask bit is set in the ATU Interrupt Mask Register - ATUIMR. When the mask bit is set, then Vendor_Defined Type 0 messages are treated as unsupported requests and Vendor_Defined Type 1 messages are silently discarded.

Table 247. Inbound Vendor Defined Message Header Register 2 - IVMHR2



Internal Bus Address Offset
+348H

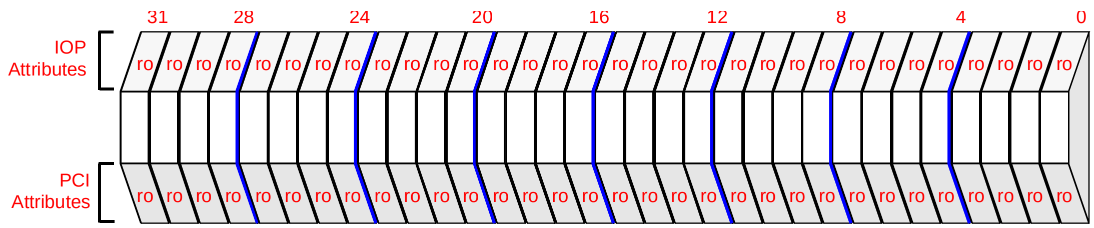
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:24	00H	Header Byte 8
23:16	00H	Header Byte 9
15:8	00H	Header Byte 10
7:0	00H	Header Byte 11

3.17.108 Inbound Vendor Message Header Register 3 - IVMHR3

Inbound Vendor Message Header Registers capture the header for a vendor defined message received on the PCI Express interface. Once the inbound message is processed, the Inbound Vendor Message Received bit is set in the ATU Interrupt Status Register - ATUISR. Subsequent inbound vendor messages are held in inbound posted queues until the status bit is cleared or mask bit is set in the ATU Interrupt Mask Register - ATUIMR. When mask bit is set, Vendor_Defined Type 0 messages are treated as unsupported requests and Vendor_Defined Type 1 messages are silently discarded.

Table 248. Inbound Vendor Defined Message Header Register 3 - IVMHR3



Internal Bus Address Offset
+34CH

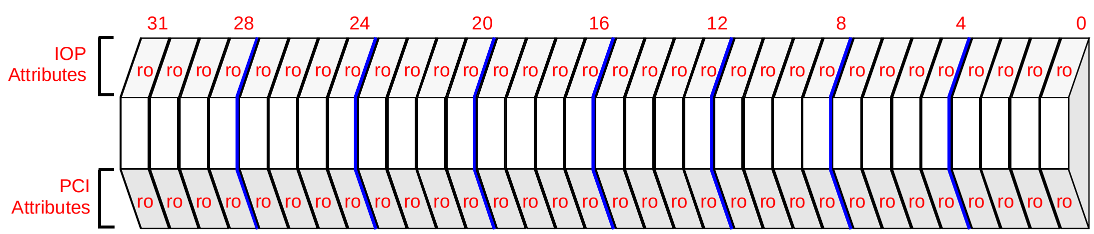
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:24	00H	Header Byte 12
23:16	00H	Header Byte 13
15:8	00H	Header Byte 14
7:0	00H	Header Byte 15

3.17.109 Inbound Vendor Message Payload Register - IVMPR

Inbound Vendor Message Payload Registers capture the payload for a vendor defined message received on the PCI Express interface. Once the inbound message has been processed, the Inbound Vendor Message Received bit is set in the ATU Interrupt Status Register - ATUISR. Subsequent inbound vendor messages are held in inbound posted queues until the status bit is cleared or mask bit is set in the ATU Interrupt Mask Register - ATUIMR. When the mask bit is set, Vendor_Defined Type 0 messages are treated as unsupported requests and Vendor_Defined Type 1 messages are silently discarded.

Table 249. Inbound Vendor Defined Message Payload Register - IVMPR



Internal Bus Address Offset
+350H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

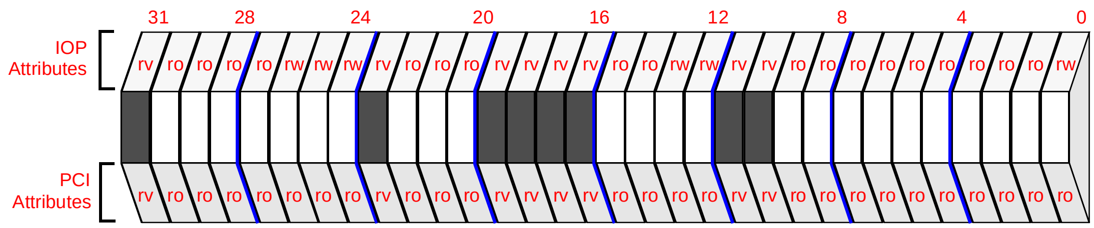
Bit	Default	Description
31:0	00H	Inbound Vendor_Defined Payload

3.17.110 Outbound Vendor Message Header Register 0 - OVMHR0

The Outbound Vendor Message Header Registers allow software to create a header that is used for a Vendor_Defined Message TLP. The OVMHR0-3 registers must be programmed prior to writing the Outbound Vendor Defined Message Payload Register - OVMPR. A write to the OVMPR initiates the Vendor_Defined Message TLP. When the payload length is 0, a write to the OVMPR is still required to initiate the TLP but the data written is ignored.

Vendor_Defined message format is shown in Figure 38

Table 250. Outbound Vendor Defined Message Header Register0 - OVMHR0



Internal Bus Address Offset
+360H

Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

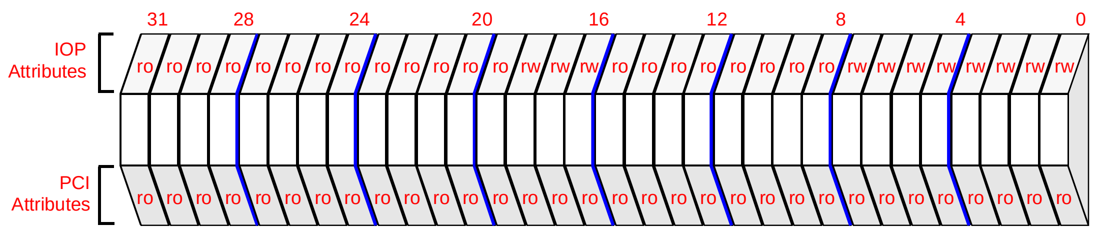
Bit	Default	Description
31	0	Reserved
30:29	01	Fmt[1:0] - Format of TLP. This field is read-only and is set based on the Payload length. Length[0] Fmt[1:0] 0 01 4DW header, no data 1 11 4DW header, with data
28:27	10	Type[4:3] - "10" indicates this is a vendor_defined message. Hard-wired to 10.
26:24	000	Type[2:0] - Message Routing. The valid encodings are specified below. 000 Routed to Root Complex 001 Routed by Address 010 Routed by ID 011 Broadcast from Root Complex 100 Local - Terminate at Receiver 101 Gathered and routed to Root Complex 110-111 Reserved - Terminate at Receiver
23	0	Reserved
22:20	000	TC[2:0] - Traffic Class - Only TC0 is supported. Hard-wired 000.
19:16	0000	Reserved
15	0	TD - TLP Digest present. Hard-wired 0.
14	0	EP - Indicates the TLP is poisoned. Hard-wired 0
13:12	00	Attr[1:0] - Attributes
11:10	00	Reserved
09:01	000H	Length[9:1] - Hardcoded 0.
00	0	Length[0] - 4138xx supports a data payload length of 1 or 0.

3.17.111 Outbound Vendor Message Header Register 1 - OVMHR1

The Outbound Vendor Message Header Registers allow software to create a header that is used for a Vendor_Defined Message TLP. The OVMHR0-3 registers must be programmed prior to writing the Outbound Vendor Defined Message Payload Register - OVMPR. A write to the OVMPR initiates the Vendor_Defined Message TLP. When the payload length is 0, a write to the OVMPR is still required to initiate the TLP but the data written is ignored.

Vendor_Defined message format is shown in Figure 38

Table 251. Outbound Vendor Defined Message Header Register 1 - OVMHR1



Internal Bus Address Offset
+364H

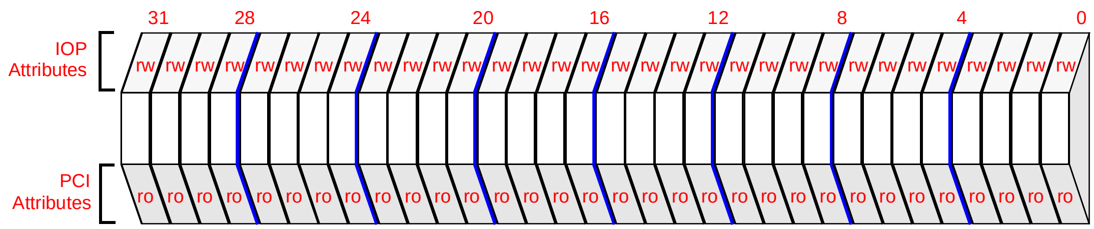
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:24	00H	Requester ID - Bus number. This value is copied from the Bus number field in the PCISR.
23:19	0_0000	Requester ID - Device Number. This value is copied from the Device number field in the PCISR
18:16	000	Requester ID - Function Number. This value is programmable.
15:8	00H	Tag - The tag field is undefined for posted requests. Hardcoded to 0.
7:0	0111_1110	Message Code[7:0] - Indicates the type of message being generated. Valid encodings are: 0111_1110 Vendor_Defined Type 0 0111_1111 Vendor_Defined Type 1 Other codes may result in malformed packets.

3.17.112 Outbound Vendor Message Header Register 2 - OVMHR2

The Outbound Vendor Message Header Registers allow software to create a header that is used for a Vendor_Defined Message TLP. The OVMHR0-3 registers must be programmed prior to writing the Outbound Vendor Defined Message Payload Register - OVMPR. A write to the OVMPR initiates the Vendor_Defined Message TLP. When the payload length is 0, a write to the OVMPR is still required to initiate the TLP but the data written is ignored. Vendor_Defined message format is shown in Figure 38

Table 252. Outbound Vendor Defined Message Header Register 2 - OVMHR2



Internal Bus Address Offset
+368H

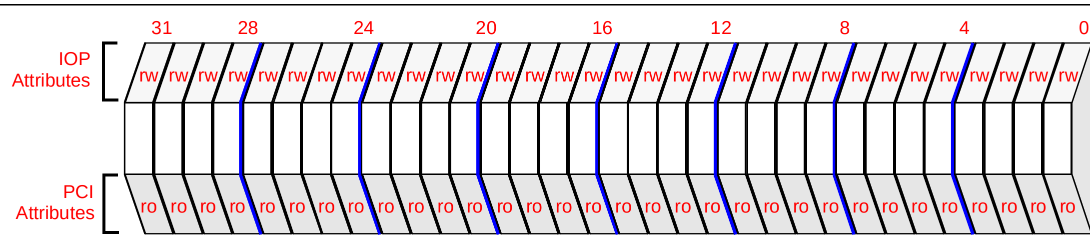
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:16	00H	Destination ID - When Type[2:0] indicates Route by ID, this field is Destination ID. Otherwise field is reserved.
16:0	00H	Vendor ID

3.17.113 Outbound Vendor Message Header Register 3 - OVMHR3

The Outbound Vendor Message Header Registers allow software to create a header that is used for a Vendor_Defined Message TLP. The OVMHR0-3 registers must be programmed prior to writing the Outbound Vendor Defined Message Payload Register - OVMPR. A write to the OVMPR initiates the Vendor_Defined Message TLP. When the payload length is 0, a write to the OVMPR is still required to initiate the TLP but the data written is ignored. Vendor_Defined message format is shown in Figure 38

Table 253. Outbound Vendor Defined Message Header Register 3 - OVMHR3



Internal Bus Address Offset
+36CH

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:0	00H	Available for Vendor Definition.

3.17.114 Outbound Vendor Message Payload Register - OVMPR

The Outbound Vendor Message Payload Register contains the payload data that is used for the Vendor_Defined Message TLP. A write to this register initiates the Vendor_Defined Message on the PCI Express Interface. When a zero length payload is desired, then a write to this register is required to initiate the transaction, but the value written is ignored.

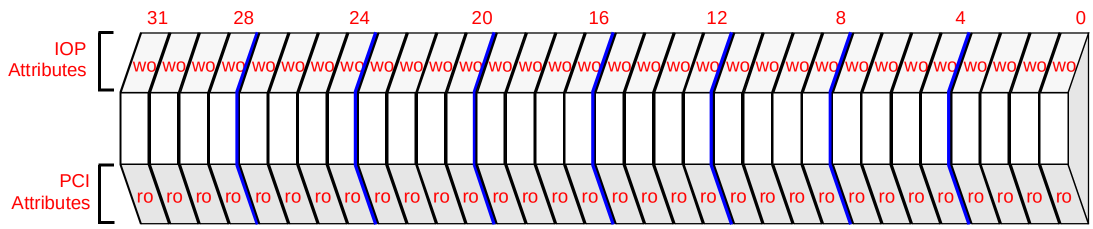
The 4138xx supports a maximum payload size of 1 DW.

The Vendor_Defined message format is shown in Figure 38

Note: This register does not physically exist. It is simply a write port. A read to this register is not claimed by the ATU and causes a data abort.

This address was chosen to put the data on lane 0 of the 4138xx internal bus.

Table 254. Outbound Vendor Defined Message Payload Register - OVMPR



Internal Bus Address Offset
+370H

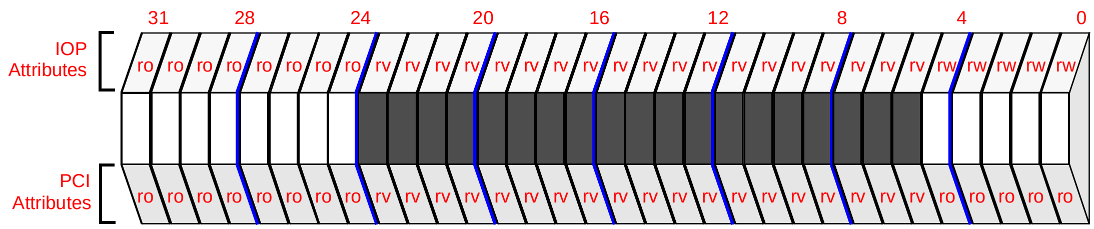
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
WO = Write Only
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:0	00H	Outbound Vendor_Defined Payload Reads of this register always return 0.

3.17.115 PCI Interface Error Control and Status Register - PIE_CSR

This register indicates whether or not the ATU has detected and logged a PCI interface error. The register is also used to enabled the logging of additional errors. For more details, see Section 3.9, “ATU Error Conditions” on page 270.

Table 255. PCI Interface Error Control and Status Register - PIE_CSR



Internal Bus Address Offset
+380H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

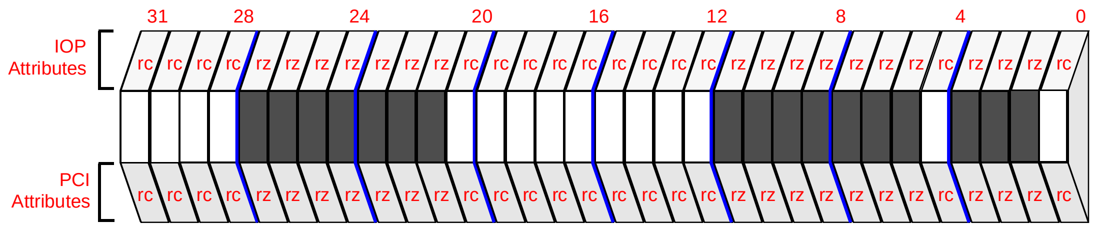
Bit	Default	Description
31:28	0000_2	Initiator ID - When the PCI Interface Error First Error Pointer (bits 4:0 of this register) is valid, this field Indicates the initiator associated with the detected PCI interface error. “0000” is used to denote inbound requests where the IOP was the completer of the transaction. See Internal Bus Requester IDs in the System Controller chapter for remaining details of Initiator ID.
27	0	General Device Error (GDE) - This bit is set when an error can not be isolated to a single function and is being logged against all function.
26:24	000_2	PCI Function Number - When PCIECSR bit 0 is set, this field indicates the 4138xx device function number that this error is associated with. ***Note:*** When the General Device Error bit is set then the value in this field is undefined and should be ignored.
23:5	0	Reserved
4:0	0 0000	PCI Interface Error First Error Pointer - Identifies the bit position of the first error reported in the PCI Interface Error Status - PIE_STS register. ***Note:*** This register does not update until all the bits in the PIE_STS register are cleared.

3.17.116 PCI Interface Error Status - PIE_STS

The PCI Interface Error Status register reports error status of individual uncorrectable error sources. An individual error status bit that is set to “1” indicates that a particular error occurred; software may clear an error status by writing a 1 to the respective bit.

Note: The status bits are set even when the reporting of the error is masked in the PIE_MSK register.

Table 256. PCI Interface Error Status - PIE_STS



Internal Bus Address Offset
+384H

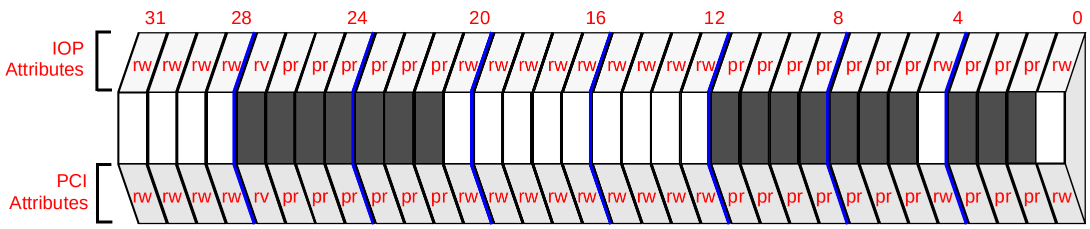
Attribute Legend:	RW = Read/Write
RZ = Reserved Zero	RC = Read Clear
PR = Preserved	RO = Read Only
RS = Read/Set	NA = Not Accessible

Bit	Default	Description
31	0	Received Completion with Unsupported Request Status
30	0	Received Completion with Completer Abort Status
29	0	Poisoned TLP Transmitted Status
28	0	Transmit: Header Parity Error detected
27:21	0	Reserved Zero - Software must write 0 to these bits.
20	0	Unsupported Request Error Status - As a receiver, Set whenever an unsupported request is detected. The Header is logged.
19	0	ECRC Check - As a receiver, set when ECRC check fails. The Header is logged.
18	0	Malformed TLP: As a receiver, set whenever a malformed TLP is detected. The Header is logged.
17	0	Receiver Overflow: Set when PCI Express receive buffers overflow.
16	0	Unexpected Completion: As a receiver, set whenever a completion is received that does not match the 4138xx’s requestor ID or outstanding Tag. The Header is logged.
15	0	Completer Abort: As a completer, set whenever an internal agent signals a data abort. The header is logged.
14	0	Completion Timeout: As a requester, set whenever an outbound Non Posted Request does not receive a completion within 16-32ms.
13	0	Flow Control Protocol Error Status: Set whenever a flow control protocol error is detected.
12	0	Poisoned TLP Received: As a receiver, set whenever a poisoned TLP is received from PCI Express. The header is logged.
11:5	0	Reserved Zero - Software must write 0 to these bits.
4	0	Data Link Protocol Error: Set whenever a data link protocol error is detected.
3:1	0	Reserved Zero - Software must write 0 to these bits.
0	0	Training Error - Set whenever a training error is detected.

3.17.117 PCI Interface Error Mask - PIE_MSK

The PCI Interface Error Mask register controls reporting of individual errors by the device to the Intel XScale® processor core via an interrupt (ATUISR bit 10). A masked error (respective bit set to 1b in the mask register) is not logged in the Header Log register, does not update the First Error Pointer, and generate an interrupt to the core.

Table 257. PCI Interface Error Mask - PIE_MSK



Internal Bus Address Offset
+388H

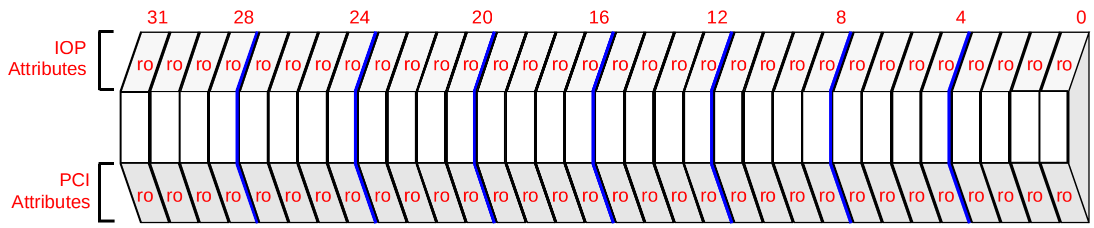
Attribute Legend:
RZ = Reserved Zero
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31	0	Received Completion with Unsupported Request Mask
30	0	Received Completion with Completer Abort Mask
29	0	Poisoned TLP Transmitted Mask
28	0	Transmit: Header Parity Error Mask
27:21	0	Reserved Zero - Software must write 0 to these bits.
20	0	Unsupported Request Error Status Error Mask - When '1' error reporting is masked.
19	0	ECRC Check Error Mask - When '1' error reporting is masked.
18	0	Malformed TLP Error Mask - When '1' error reporting is masked.
17	0	Receiver Overflow Error Mask - When '1' error reporting is masked.
16	0	Unexpected Completion Error Mask - When '1' error reporting is masked.
15	0	Completer Abort Error Mask - When '1' error reporting is masked.
14	0	Completion Time Out Error Mask - When '1' error reporting is masked.:
13	0	Flow Control Protocol Error Status Error Mask - When '1' error reporting is masked.
12	0	Poisoned TLP Received Error Mask - When '1' error reporting is masked.
11:5	0	Preserved.
4	0	Data Link Protocol Error Mask - When '1' error reporting is masked.
3:1	0	Preserved.
0	0	Training Error Mask - When '1' error reporting is masked.

3.17.118 PCI Interface Error Header Log - PIE_LOG0

Transaction header log for PCI interface errors.

Table 258. PCI Interface Error Header Log - PIE_LOG0



Internal Bus Address Offset
+38CH

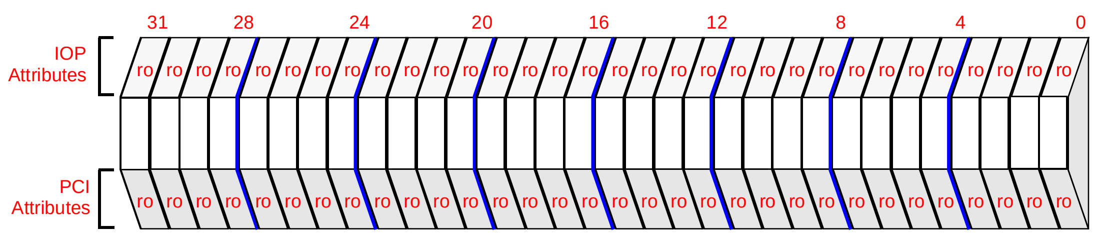
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:0	0	1st DWord of the Header for the PCI Express packet in error. Once an error is logged in this register, it remains locked for further error logging until the time the software clears the status bit that cause the header log i.e. the error pointer is rearmed to log again.

3.17.119 PCI Interface Error Header Log 1 - PIE_LOG1

Transaction header log for PCI interface errors.

Table 259. PCI Interface Error Header Log 1 - PIE_LOG1



Internal Bus Address Offset
+390H

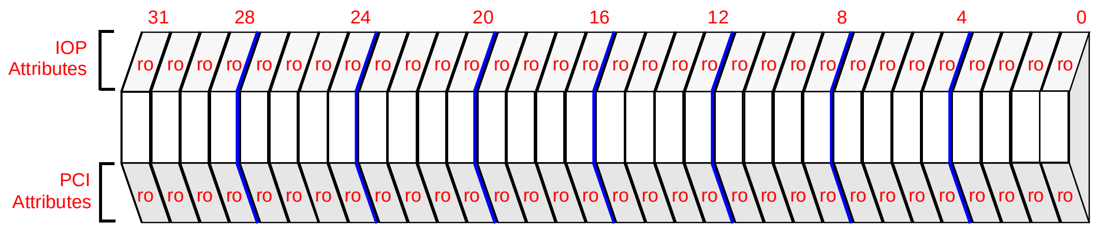
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:0	0	2nd DWord of the Header for the PCI Express packet in error. Once an error is logged in this register, it remains locked for further error logging until the time the software clears the status bit that cause the header log i.e. the error pointer is rearmed to log again.

3.17.120 PCI Interface Error Header Log 2 - PIE_LOG2

Transaction header log for PCI interface errors.

Table 260. PCI Interface Error Header Log 2 - PIE_LOG2



Internal Bus Address Offset
+394H

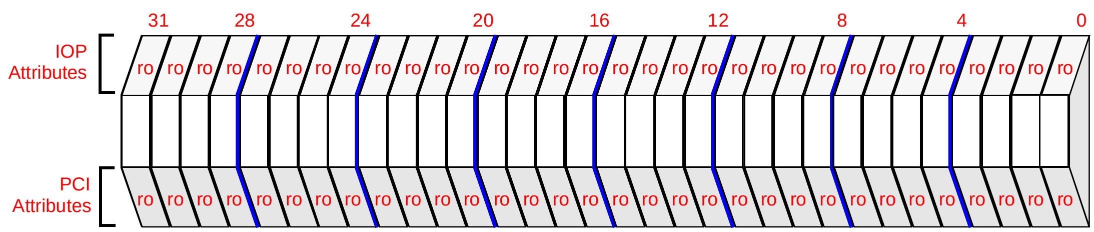
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:0	0	3rd DWord of the Header for the PCI Express packet in error. Once an error is logged in this register, it remains locked for further error logging until the time the software clears the status bit that cause the header log i.e. the error pointer is rearmed to log again.

3.17.121 PCI Interface Error Header Log - PIE_LOG3

Transaction header log for PCI interface errors.

Table 261. PCI Interface Error Header Log - PIE_LOG3



Internal Bus Address Offset
+398H

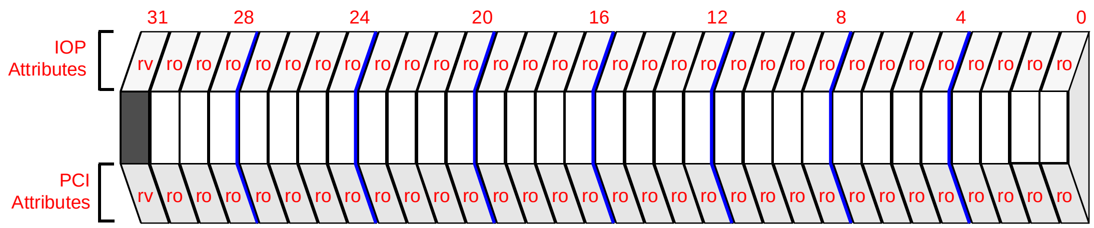
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:0	0	4th DWord of the Header for the PCI Express packet in error. Once an error is logged in this register, it remains locked for further error logging until the time the software clears the status bit that cause the header log i.e. the error pointer is rearmed to log again.

3.17.122 PCI Interface Error Descriptor Log

Descriptor log for transaction errors.

Table 262. PCI Interface Error Descriptor Log - PIE_DLOG



Internal Bus Address Offset
+39CH

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31	0	Reserved
30:29	00	DMA Channel Number
28:0	0H	ADMA Descriptor Address The DMA type can be determined by the Initiator ID in the "PCI Interface Error Control and Status Register - PIE_CSR"

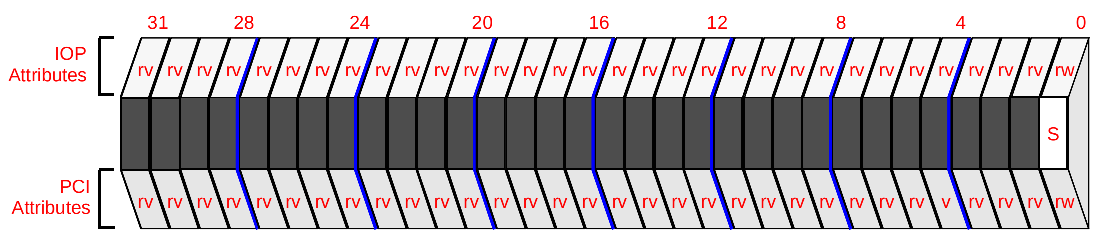
Note: In 4138xx, bits[30:0] of this register is copied directly from the USER_ATTR[65:35] sideband bus.

3.17.123 ATU Reset Control Register - ATURCR

This register controls the link down reset bypass strap (**LK_DN_RST_BYPASS#**).

Note: Some bits in this register are sticky through reset.

Table 263. ATU Reset Control Register - ATURCR



Internal Bus Address Offset
+3B0H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:01	0H	Reserved
00	0H	Link Down Reset Bypass override - When this bit is set, the Link Down Reset Bypass strap is ignored and treated as when it is deasserted.

4.0 Messaging Unit

This chapter describes the Messaging Unit (MU) of the Intel® 413808 and 413812 I/O Controllers (4138xx).

4.1 Overview

The Messaging Unit (MU) provides a mechanism for data to be transferred between the PCI bus and the Intel XScale® processor and notifying the respective system of the arrival of new data through an interrupt. The MU can be used to send and receive messages.

The MU is located on the south internal bus of the 4138xx. External PCI agents access the MU via the ATU. For example, the ATU initiates transactions on the internal bus to the MU on behalf of the external PCI agents.

The MU has two distinct messaging mechanisms. Each allows a host processor or external PCI agent and the 4138xx to communicate through message passing and interrupt generation. The two mechanisms are:

- **Message Registers** — allow the 4138xx and external PCI agents to communicate by passing messages in one of four 32-bit Message Registers. In this context, a message is any 32-bit data value. Message registers combine aspects of mailbox registers and doorbell registers. Writes to the message registers may optionally cause interrupts.
- **Doorbell Registers** — allow the 4138xx to assert the PCI interrupt signals and allow external PCI agents to generate an interrupt to the Intel XScale® processor.

Each of the above are available to the system designer at the same time. No special mode selection is needed.

4.2 Theory of Operation

The MU has two independent messaging mechanisms. The Message Registers are similar to a combination of mailbox and doorbell registers. Each holds a 32-bit value and generates an interrupt when written. The two Doorbell Registers support software interrupts. When a bit is set in a Doorbell Register, an interrupt is generated.

Interrupt status for all interrupts is recorded in the Inbound Interrupt Status Register and the Outbound Interrupt Status Register. Each interrupt generated by the Messaging Unit can be masked.

Because of read side effects, multi-word burst transactions are not supported by the Messaging Unit. The Messaging Unit must be mapped in a non-prefetchable PCI address space to avoid read side effects. Multi-word read or write transactions made to the Messaging Unit registers causes the MU to generate an address error on the internal bus of the 4138xx. Multi-word transactions made by an external PCI agent results in an error being sent to the external PCI agent. Refer to the ATU chapters for more details on how the ATUs respond to an internal bus error.

All registers needed to configure and control the Messaging Unit are memory-mapped registers.

The MU is accessed by an external PCI agent via the ATU. The MU can be mapped in any 4 Kbytes of the inbound translation window in the Address Translation Unit (ATU). The MU provides the Base Address Registers (Table 278, "MU Base Address Register - MUBAR" and "MU Upper Base Address Register - MUUBAR") which allow the MU to be relocated within the ATU translated window. This PCI address window is used for PCI transactions that access the 4138xx local memory. The PCI address of the inbound translation window is contained in the Inbound ATU Base Address Register. See Chapter 2.0, "Address Translation Unit (PCI-X)" or Chapter 3.0, "Address Translation Unit (PCI Express)" for more details on inbound ATU addressing and the ATU.

Note that since the MU is located on the internal bus of the 4138xx, any PCI transaction that is targeted for the MU is first claimed by the ATU and then the ATU issues the transaction on the internal bus of the 4138xx.

Figure 39. PCI Memory Map

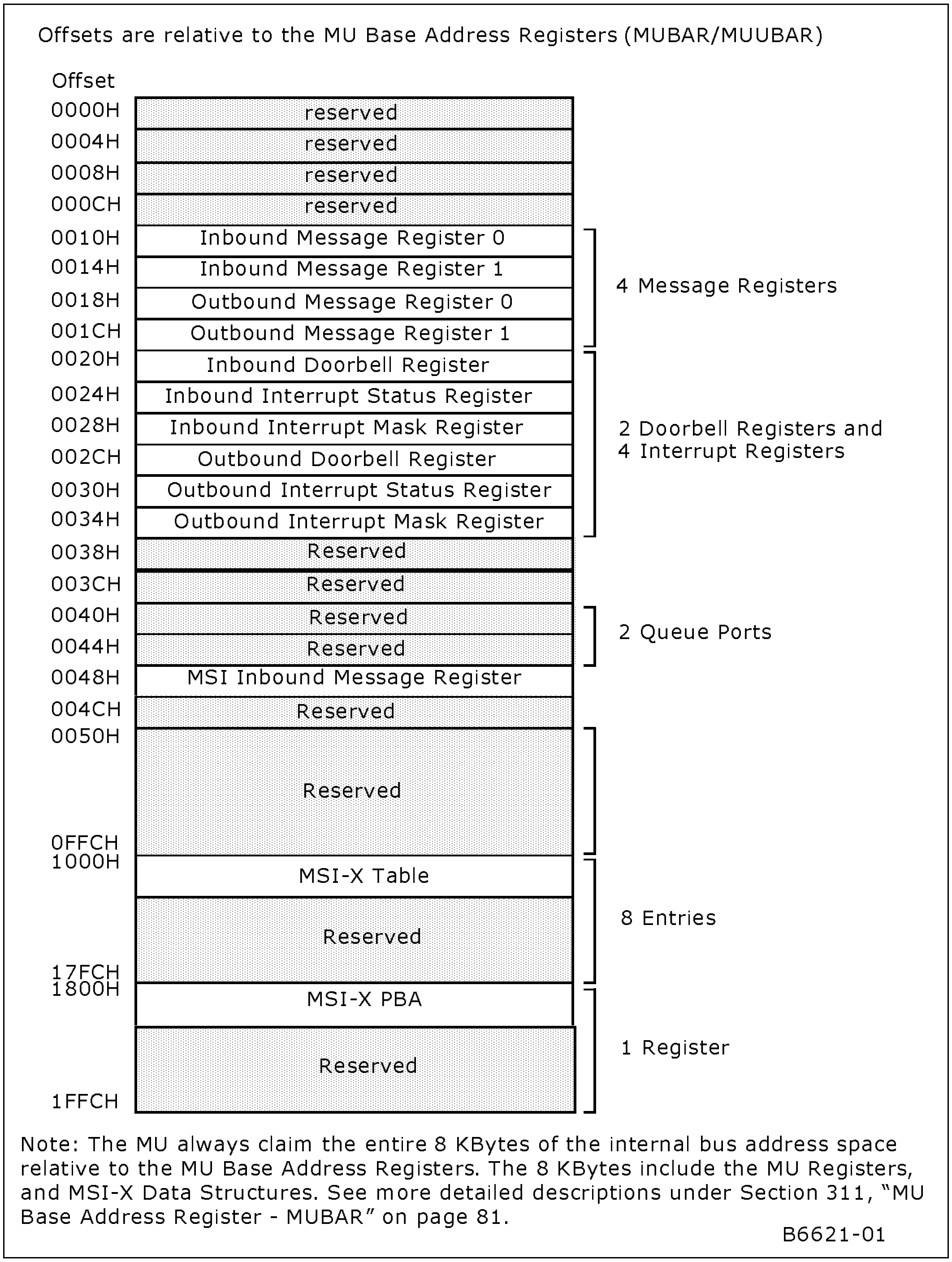
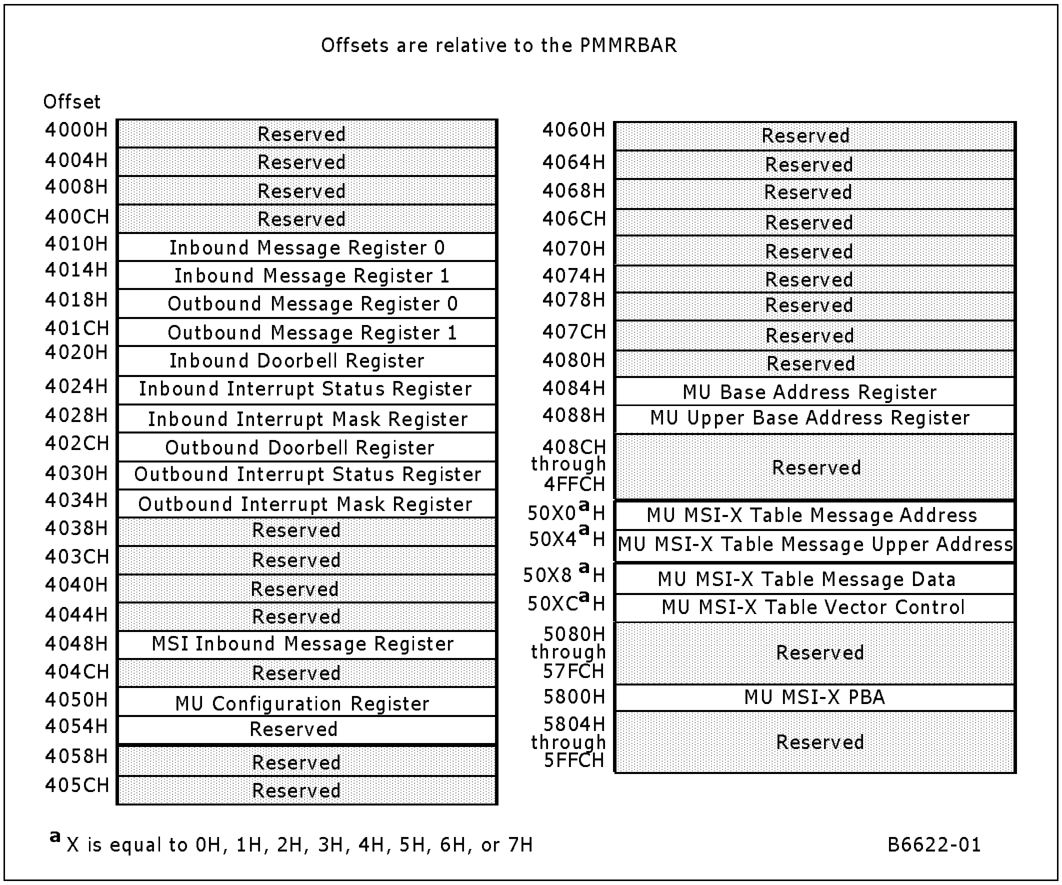


Figure 40. Internal Bus Memory Map



Note: See Table 265, "Message Unit Registers" for more details.

Table 264 provides a summary of the two messaging mechanisms used in the Messaging Unit.

Table 264. MU Summary

Mechanism	Quantity	Assert PCI Interrupt Signals?	Generate 4138xx Interrupt?
Message Registers	2 Inbound 2 Outbound	Optional	Optional
Doorbell Registers	1 Inbound 1 Outbound	Optional	Optional

4.2.1 Transaction Ordering

From a PCI standpoint, the Messaging Unit is a piece of the ATU and therefore must maintain ordering requirements against ATU transactions. Transaction ordering is achieved for the Index Registers, the Doorbell Register, and the Message Registers since these transactions are routed through the standard set of ATU read/write queues.

4.3 Message Registers

Messages can be sent and received by the 4138xx through the use of the Message Registers. When written, the message registers may cause an interrupt to be generated to either the Intel XScale® processor or the host processor. Inbound messages are sent by the host processor and received by the 4138xx. Outbound messages are sent by the 4138xx and received by the host processor.

The interrupt status for outbound messages is recorded in the Outbound Interrupt Status Register. Interrupt status for inbound messages is recorded in the Inbound Interrupt Status Register.

4.3.1 Outbound Messages

When an outbound message register is written by the Intel XScale® processor, an interrupt may be generated on the **P_INTA#** interrupt pins or a message signaled interrupt is generated when MSI is enabled.

The PCI interrupt is recorded in the Outbound Interrupt Status Register. The interrupt causes the Outbound Message Interrupt bit to be set in the Outbound Interrupt Status Register. This is a Read/Clear bit that is set by the MU hardware and cleared by software.

The interrupt is cleared when an external PCI agent writes a value of 1 to the Outbound Message Interrupt bit in the Outbound Interrupt Status Register to clear the bit.

The interrupt may be masked by the mask bits in the Outbound Interrupt Mask Register.

4.3.2 Inbound Messages

When an inbound message register is written by an external PCI agent, an interrupt may be generated to the Intel XScale® processor. The interrupt may be masked by the mask bits in the Inbound Interrupt Mask Register.

The Intel XScale® processor interrupt is recorded in the Inbound Interrupt Status Register. The interrupt causes the Inbound Message Interrupt bit to be set in the Inbound Interrupt Status Register. This is a Read/Clear bit that is set by the MU hardware and cleared by software.

The interrupt is cleared when the Intel XScale® processor writes a value of 1 to the Inbound Message Interrupt bit in the Inbound Interrupt Status Register.

4.4 Doorbell Registers

There are two Doorbell Registers: the Inbound Doorbell Register and the Outbound Doorbell Register. The Inbound Doorbell Register allows external PCI agents to generate interrupts to the Intel XScale® processor. The Outbound Doorbell Register allows the Intel XScale® processor to generate a PCI interrupt. Both Doorbell Registers may generate interrupts whenever a bit in the register is set.

4.4.1 Outbound Doorbells

When the Outbound Doorbell Register is written by the Intel XScale® processor, an interrupt may be generated on the **P_INTA#** interrupt pin or a message signaled interrupt is generated when MSI is enabled. An interrupt is generated when any of the bits in the doorbell register is written to a value of 1. Writing a value of 0 to any bit does not change the value of that bit and does not cause an interrupt to be generated. Once a bit is set in the Outbound Doorbell Register, it cannot be cleared by the Intel XScale® processor.

The interrupt is recorded in the Outbound Interrupt Status Register.

The interrupt may be masked by the mask bits in the Outbound Interrupt Mask Register. When the mask bit is set for a particular bit, no interrupt is generated for that bit. The Outbound Interrupt Mask Register affects only the generation of the interrupt and not the values written to the Outbound Doorbell Register.

The interrupt is cleared when an external PCI agent writes a value of 1 to the bits in the Outbound Doorbell Register that are set. Writing a value of 0 to any bit does not change the value of that bit and does not clear the interrupt.

In summary, the Intel XScale® processor generates an interrupt and external PCI agents clear the interrupt by setting bits in the Outbound Doorbell Register.

4.4.2 Inbound Doorbells

When the Inbound Doorbell Register is written by an external PCI agent, an interrupt may be generated to the Intel XScale® processor. An interrupt is generated when any of the bits in the doorbell register is written to a value of 1. Writing a value of 0 to any bit does not change the value of that bit and does not cause an interrupt to be generated. Once a bit is set in the Inbound Doorbell Register, it cannot be cleared by any external PCI agent. The interrupt is recorded in the Inbound Interrupt Status Register.

The interrupt may be masked by the Inbound Doorbell Interrupt mask bit in the Inbound Interrupt Mask Register. When the mask bit is set for a particular bit, no interrupt is generated for that bit. The Inbound Interrupt Mask Register affects only the generation of the normal messaging unit interrupt and not the values written to the Inbound Doorbell Register. One bit in the Inbound Doorbell Register is reserved for an Error Doorbell interrupt.

The interrupt is cleared when the Intel XScale® processor writes a value of 1 to the bits in the Inbound Doorbell Register that are set. Writing a value of 0 to any bit does not change the value of that bit and does not clear the interrupt.

4.5 Messaging Unit Error Conditions

The Messaging Unit, like the ATU, encounters error conditions on the host I/O interface as well as the internal bus interface. As a host I/O interface target, all host I/O interface errors are captured and recorded in the ATU Status Register and can be masked using the ATU mechanisms. Refer to Chapter 2.0, “Address Translation Unit (PCI-X)” or Chapter 3.0, “Address Translation Unit (PCI Express)” for further details.

4.6 Message-Signaled Interrupts

4.6.1 MSI Capability Structure

When a host processor enables Message-Signaled Interrupts (MSI) on the 4138xx ATU function, the ATU function (MU) is responsible to signal interrupt to the host via a host I/O interface write instead of the assertion of the **P_INTA#** output pin.

The *PCI-X Addendum to the PCI Local Bus Specification*, Revision 1.0a states that "PCI-X devices that generate interrupts are required to support message-signaled interrupts, as defined by the *PCI Local Bus Specification*, Revision 2.2 and must support a 64-bit message address." "Devices that require interrupts in systems that do not support message-signaled interrupts, must implement interrupt pins." Thus, the 4138xx needs to implement both wired and message-signaled interrupt delivery mechanisms.

In support of MSI, the 4138xx implements the MSI capability structure. The capability structure includes the "Message Control Register - Message_Control" on page 431, the "Message Address Register - Message_Address" on page 432, the "Message Upper Address Register - Message_Upper_Address" on page 433 and the"Message Data Register- Message_Data" on page 434.

During system initialization, the configuration software for an MSI system reads the Message Control Register to determine that the 4138xx supports a 64-bit Message Address, and that it is capable of generating two unique interrupt messages.

After gathering this data from all of the MSI capable devices in the system, the configuration software decides where to initialize the Message Address and how many unique messages each MSI capable device is allowed. Then, software writes the Message Address Registers (and the Message Upper Address Registers when Message Address is above the 4G address boundary[12]), and the Message Data Register. This system specified data is used to route the interrupt request message to the appropriate entry in a host processor Local APIC table.

Configuration of MSI completes with a write to the Message Control Register which includes an update to the Multiple Message Enable field and the MSI enable bit of each device. This informs the device how many unique messages (Local APIC table entries) have been allocated for exclusive use by that device and enable that device for MSI. Device hardware is required to handle allocation of fewer unique interrupt messages than requested by the Multiple Message Capable field.

The 4138xx is able to handle generating only one message, even though the device is capable of generating two unique messages. When two unique messages are enabled, one message is reserved for Outbound Post Queue Interrupt, the other message represents all of Outbound Doorbell and Outbound Message Interrupts. When only one message is enabled, all interrupts are represented by a single message. Interrupt handler software needs to read the 4138xx Outbound Interrupt Status Register to determine the cause of the interrupt when more than one source is represented by a single message.

To signal an Outbound Interrupt with MSI enabled, the 4138xx creates an outbound write transaction using the Message Address and the Message Data. When two unique messages are enabled, the lowest order bit of the Message Data is modified by hardware so that the host processor can distinguish between them.

12. When host software writes the Message Upper address register to a non-zero value, device hardware uses a write transaction with a Dual Address Cycle (DAC) to present the full 64-bit address to the bus.

4.6.2 MSI-X Capability and Table Structures

Similar to MSI Capability, when a host processor enables Message-Signaled Interrupts (MSI-X) on the 4138xx ATU function, the ATU function (MU) is responsible to signal interrupt to the host via a PCI write instead of the assertion of the **P_INTA#**output pin.

The *PCI-X Addendum to the PCI Local Bus Specification*, Revision 1.0a states that "PCI-X devices that generate interrupts are required to support message-signaled interrupts, as defined by the *PCI Local Bus Specification*, Revision 2.2 and must support a 64-bit message address." "Devices that require interrupts in systems that do not support message-signaled interrupts, must implement interrupt pins." Thus, the 4138xx needs to implement both wired and message-signaled interrupt delivery mechanisms.

In support of MSI-X, the 4138xx implements the MSI-X capability structure. The capability structure includes the "MSI-X Capability Identifier Register - MSI-X_Cap_ID" on page 435, the "MSI-X Next Item Pointer Register - MSI-X_Next_Item_Ptr" on page 436, the "MSI-X Message Control Register - MSI-X_MCR" on page 437, the "MSI-X Table Offset Register — MSI-X_Table_Offset" on page 438 and the"MSI-X Pending Bit Array Offset Register - MSI-X_PBA_Offset" on page 439.

During system initialization, the configuration software for an MSI-X system reads the Message Control Register to determine that the 4138xx is capable of generating eight unique interrupt messages by reading the MSI-X table size. The configuration software also reads the MSI-X Table offset Register and MSI-X Pending Bits Array Register to determine the locations these structures.

After gathering this data from all of the MSI-X capable devices in the system, the configuration software decides how to initialize the MSI-X Table by writing the Message Address Registers (and the Message Upper Address Registers when Message Address is above the 4G address boundary[13]), the Message Data Registers, and the Vector Control Registers in order to unmask a Table entry. This system specified data is used to route the interrupt request message to the appropriate entry in a host processor's Local APIC table.

Configuration of MSI-X completes with a write to the MSI-X Message Control Register which includes an update to the Multiple Message Enable field.

The MU on 4138xx is able to handle generating up to eight unique messages, representing all of Outbound Doorbell Interrupts. Each outbound interrupt on the MU has a unique MSI-X message associated to it. The MU can also be setup to generate only a single MSI-X message. A single MSI-X message is generated when the MU MSI-X Single Message Vector bit is set in the "MU MSI-X Control Register X — MMCRx" on page 440.

Note: The MU can signal an interrupt (MU MSI-X Table Write Interrupt) to the 4138xx interrupt controller when the Host processor writes any entry in the MU MSI-X Table. The MU MSI-X Table includes the following fields: the Message Address Registers, the Message Upper Address Registers, the Message Data Registers, and the Vector Control Registers. The interrupt can be enabled or disabled using the "Inbound Interrupt Mask Register - IIMR", and interrupt is posted in the "Inbound Interrupt Status Register - IISR".

13. When host software writes the Message Upper address register to a non-zero value, device hardware uses a write transaction with a Dual Address Cycle (DAC) to present the full 64-bit address to the bus.

To signal an Outbound Interrupt with MSI-X enabled, the 4138xx creates an outbound write transaction using the Message Address and the Message Data of the associated entry. On 4138xx, entry 0 of the MSI-X Table is assigned to bit OISR[0], entry 1 is assigned to bit OISR[1] and so on. For example, entry 7 of the MSI-X Table is assigned to bit OISR[7].

Note: Entry 2 is assigned to bits OSIR[2] and OISR[31]. For example, the Outbound Doorbell Register and the Firmware Interrupt bit in the Outbound Control and Status Register.

Note: When host software enables MSI, a Messaging Unit Interrupt does not result in the assertion of the **P_INTx#** output pin. However, all the **P_INT[A:D]#** pins are functional for steering of interrupts from other PCI devices that may not be MSI capable.

MSI-X Table and Pending Bits Array are mapped relative to the PCI/Host interface (Figure 41). The MU registers are located in the first 4-KByte of the 8-KByte address space claimed by the MU, whereas the MSI-X Table is located at a 4-KByte offset and the MSI-X PBA is located at a 6-KByte offset relative to the MU Base Address. Note that the MU Base Address register must be programmed such that it overlaps the address space defined by the ATU Translate and ATU Limit Registers. For example, the MU 8-KByte window must overlap onto the ATU Translation Window.

Figure 41. MSI-X Table and PBA Address Mapping Layout relative to the Host Interface

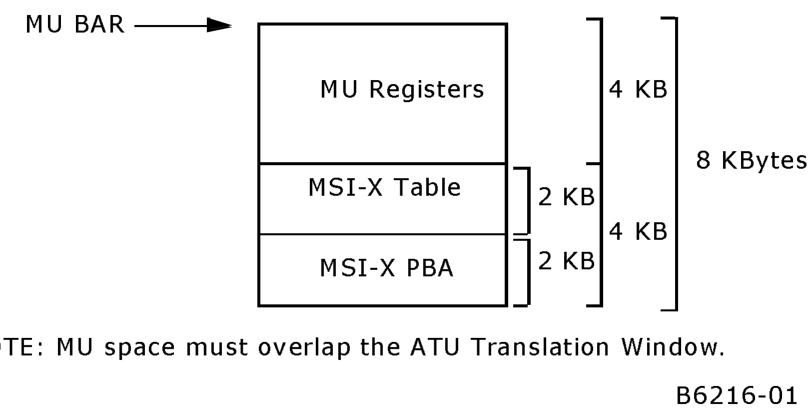
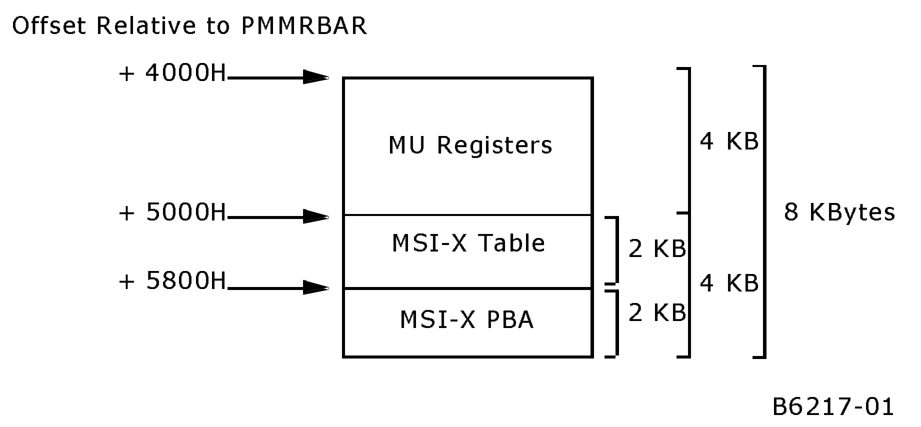


Figure 42 shows how the MSI-X Table and Pending Bits Array are mapped from the host I/O interface. The MU registers are located in the first 4-KByte of the 8-KByte address space.

Figure 42. MSI-X Table and PBA Address Mapping Layout relative to the Internal Bus



4.6.3 Level-Triggered Versus Edge-Triggered Interrupts

When MSI and MSI-X are disabled, the **P_INTA#** pin remains asserted and pended to the host when **any** of the MU interrupt sources requires service (Outbound Post Queue, Outbound Doorbell and Outbound Message). Since the PCI pin signaled interrupt is **level-triggered**, the interrupt service routine does not drop out of the service routine until the interrupt signal is deasserted. This insures that an interrupt is not missed.

MSI interrupts are inherently edge-triggered, in that an interrupt is only pended to the host as a write event when any of the MU interrupt sources requires service.

Note: Bit[10] (Interrupt Disable bit) of the ATUs ATU Command Register (ATUCMD) must be cleared for the **P_INTA#** interrupt to be generated to the Host processor. Bit[3] (Interrupt Status bit) of the ATU's ATU Status Register (ATUSR) is not affected by the state of bit[10] of the ATUCMD.

4.7 Register Definitions

The following registers are located in the Host I/O Interface address space and in the Peripheral Memory-Mapped Register (PMMR) address space. They are accessible through host I/O interface bus transactions and through Intel XScale® processor internal bus accesses. In the Host I/O Interface address space, they are mapped into the first 80 bytes of the inbound address window of the ATU.

- Inbound Message 0 Register
- Inbound Message 1 Register
- Outbound Message 0 Register
- Outbound Message 1 Register
- Inbound Doorbell Register
- Inbound Interrupt Status Register
- Inbound Interrupt Mask Register
- Outbound Doorbell Register
- Outbound Interrupt Status Register
- Outbound Interrupt Mask Register
- Inbound Reset Control and Status Register
- Outbound Reset Control and Status Register
- MSI Inbound Message Register

The following registers are located in the Peripheral Memory-Mapped Register (PMMR) address space as described in Chapter 19.0, "Peripheral Registers".

- MU Configuration Register
- MU Base Address Register
- MU Upper Base Address Register

Reading or writing a register that is reserved is undefined.

Table 265. Message Unit Registers

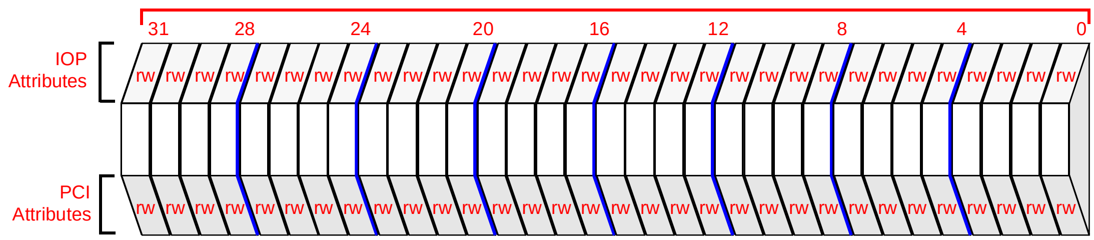
Internal Bus Address Offset	Section, Register Name - Acronym (Page)
4010H	Section 4.7.1, "Inbound Message Register - IMRx" on page 412
4014H	Section 4.7.1, "Inbound Message Register - IMRx" on page 412
4018H	Section 4.7.2, "Outbound Message Register - OMRx" on page 412
401CH	Section 4.7.2, "Outbound Message Register - OMRx" on page 412
4020H	Section 4.7.3, "Inbound Doorbell Register - IDR" on page 413
4024H	Section 4.7.4, "Inbound Interrupt Status Register - IISR" on page 414
4028H	Section 4.7.5, "Inbound Interrupt Mask Register - IIMR" on page 415
402CH	Section 4.7.6, "Outbound Doorbell Register - ODR" on page 416
4030H	Section 4.7.7, "Outbound Interrupt Status Register - OISR" on page 417
4034H	Section 4.7.8, "Outbound Interrupt Mask Register - OIMR" on page 418
4038H	Section 4.7.9, "Inbound Reset Control and Status Register - IRCSR" on page 419
403CH	Section 4.7.10, "Outbound Reset Control and Status Register - ORCSR" on page 420
4048H	Section 4.7.11, "MSI Inbound Message Register — MIMR" on page 421
4050H	Section 4.7.12, "MU Configuration Register - MUCR" on page 422
4084H	Section 4.7.13, "MU Base Address Register - MUBAR" on page 423
4088H	Section 4.7.14, "MU Upper Base Address Register - MUUBAR" on page 424
408CH - 4FFCH	Reserved,
50**X**0H[1]	Section 4.7.15, "MU MSI-X Table Message Address Registers - M_MT_MAR[0:7]" on page 425
50**X**4H[2]	Section 4.7.16, "MU MSI-X Table Message Upper Address Registers - M_MT_MUAR[0:7]" on page 426
50**X**8H[3]	Section 4.7.17, "MU MSI-X Table Message Data Registers - M_MT_MDR[0:7]" on page 427
50**X**CH[4]	Section 4.7.18, "MU MSI-X Table Message Vector Control Registers - M_MT_MVCR[0:7]" on page 428
5080H - 57FCH	Reserved.
5800H	Section 4.7.19, "MU MSI-X Pending Bits Array Register - M_MPBAR" on page 429
5804H - 5FFCH	Reserved.

Notes:
1. X is equal to 0H, 1H, 2H, 3H, 4H, 5H, 6H, or 7H.
2. X is equal to 0H, 1H, 2H, 3H, 4H, 5H, 6H, or 7H.
3. X is equal to 0H, 1H, 2H, 3H, 4H, 5H, 6H, or 7H.
4. X is equal to 0H, 1H, 2H, 3H, 4H, 5H, 6H, or 7H.

4.7.1 Inbound Message Register - IMRx

There are two Inbound Message Registers: IMR0 and IMR1. When the IMR register is written, an interrupt to the Intel XScale® processor may be generated. The interrupt is recorded in the Inbound Interrupt Status Register and may be masked by the Inbound Message Interrupt Mask bit in the Inbound Interrupt Mask Register.

Table 266. Inbound Message Register - IMRx



MU/PCI Base Address Offset
IMR0: 0010H
IMR1: 0014H

internal bus address offset
IMR0: 4010H
IMR1: 4014H

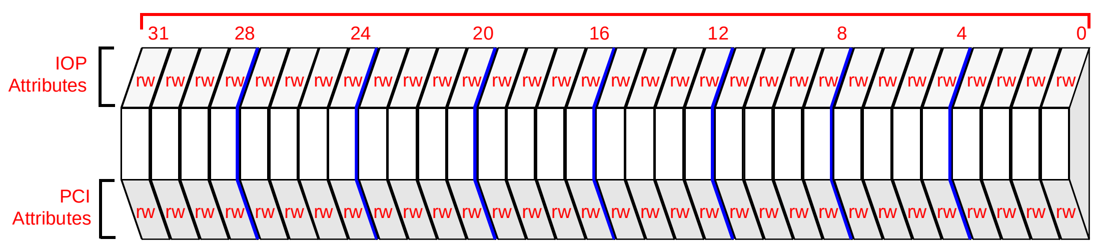
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:00	0000 0000H	Inbound Message - This is a 32-bit message written by an external Host I/O Interface agent. When written, an interrupt to the Intel XScale® processor may be generated.

4.7.2 Outbound Message Register - OMRx

There are two Outbound Message Registers: OMR0 and OMR1. When the OMR register is written, a Host I/O Interface interrupt may be generated. The interrupt is recorded in the Outbound Interrupt Status Register and may be masked by the Outbound Message Interrupt Mask bit in the Outbound Interrupt Mask Register.

Table 267. Outbound Message Register - OMRx



MU/PCI Base Address Offset
OMR0: 0018H
OMR1: 001CH

internal bus address offset
OMR0: 4018H
OMR1: 401CH

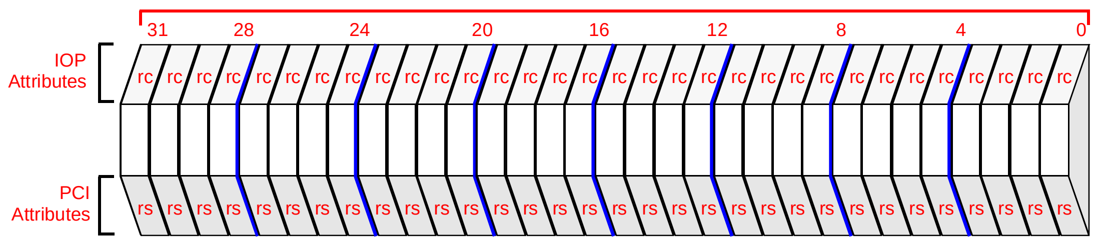
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:00	00000000H	Outbound Message - This is 32-bit message written by the Intel XScale® processor. When written, an interrupt may be generated on the PCI Interrupt pin determined by the ATU Interrupt Pin Register.

4.7.3 Inbound Doorbell Register - IDR

The Inbound Doorbell Register (IDR) is used to generate interrupts to the Intel XScale® processor. Bit 31 is reserved for generating an Error Doorbell interrupt. When bit 31 is set, an Error interrupt may be generated to the Intel XScale® processor. All other bits, when set, cause the Normal Messaging Unit interrupt line of the Intel XScale® processor to be asserted, when the interrupt is not masked by the Inbound Doorbell Interrupt Mask bit in the Inbound Interrupt Mask Register. The bits in the IDR register can only be set by an external Host I/O Interface agent and can only be cleared by the Intel XScale® processor.

Table 268. Inbound Doorbell Register - IDR



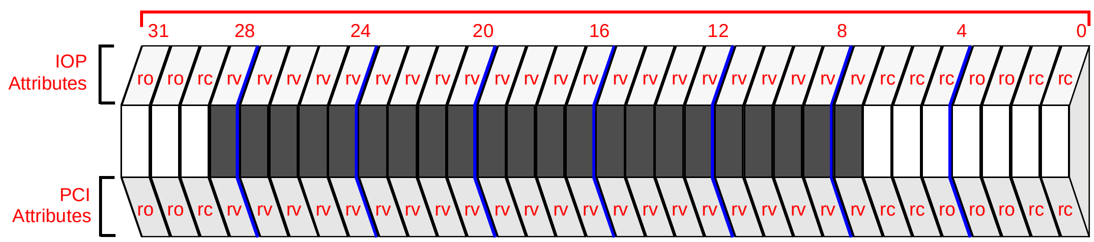
MU/PCI Base Address Offset	internal bus address offset	Attribute Legend:	
IDR: 0020H	IDR: 4020H	RV = Reserved	RW = Read/Write
		PR = Preserved	RC = Read Clear
		RS = Read/Set	RO = Read Only
			NA = Not Accessible

Bit	Default	Description
31	0_2	Error Interrupt - Generate an Error Interrupt to the Intel XScale® processor.
30:00	00000000H	Normal Interrupt - When any bit is set, generate a Normal interrupt to the Intel XScale® processor. When all bits are clear, do not generate a Normal Interrupt.

4.7.4 Inbound Interrupt Status Register - IISR

The Inbound Interrupt Status Register (IISR) contains hardware interrupt status. It records the status of Intel XScale® processor interrupts generated by the Message Registers, Doorbell Registers, and the Circular Queues. All interrupts are routed to the Normal Messaging Unit interrupt input of the Intel XScale® processor, except for the Error Doorbell Interrupt and the Outbound Free Queue Full interrupt; these two are routed to the Messaging Unit Error interrupt input. The generation of interrupts recorded in the Inbound Interrupt Status Register may be masked by setting the corresponding bit in the Inbound Interrupt Mask Register. Some of the bits in this register are Read Only. For those bits, the interrupt must be cleared through another register.

Table 269. Inbound Interrupt Status Register - IISR



MU/PCI Base Address Offset
IISR: 0024H

internal bus address offset
IISR: 4024H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

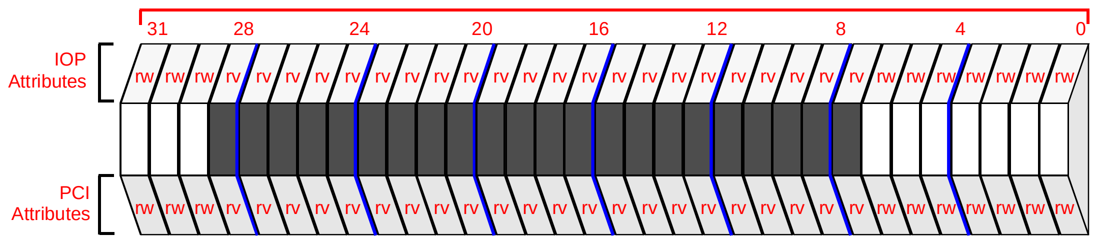
Bit	Default	Description
31	0_2	Coordinated Reset Interrupt - This bit is set when the Coordinated Reset bit (bit 1) is set in the Inbound Reset Control and Status Register.
30	0_2	Selective Reset Interrupt - This bit is set when the Selective Reset bit (bit 0) is set in the Inbound Reset Control and Status Register.
29	0_2	MU MSI-X Table Write Interrupt - This bit is set when a write occurs to any of the MU MSI-X Table entry. When set, this bit results in the assertion of the MU MSI-X Table Write Interrupt.
28:07	0000000H	Reserved
06	0_2	Index Register Interrupt - This bit is set by the MU hardware when an Index Register has been written after a Host I/O Interface transaction.
05	0_2	Outbound Free Queue Full Interrupt - This bit is set when the Outbound Free Head Pointer becomes equal to the Tail Pointer and the queue is full. An Error interrupt is generated for this condition.
04	0_2	Inbound Post Queue Interrupt - This bit is set by the MU hardware when the Inbound Post Queue has been written. Once cleared, an interrupt is NOT generated when the head and tail pointers remain unequal (i.e. queue status is Not Empty). Therefore, when software leaves any unprocessed messages in the post queue when the interrupt is cleared, software must retain the information that the Inbound Post queue status is not empty. ***Note:*** This interrupt is provided with dedicated support in the 4138xx Interrupt Controller.
03	0_2	Error Doorbell Interrupt - This bit is set when the Error Interrupt of the Inbound Doorbell Register is set. To clear this bit (and the interrupt), the Error Interrupt bit of the Inbound Doorbell Register must be clear.
02	0_2	Inbound Doorbell Interrupt - This bit is set when at least one Normal Interrupt bit in the Inbound Doorbell Register is set. To clear this bit (and the interrupt), the Normal Interrupt bits in the Inbound Doorbell Register must all be clear.
01	0_2	Inbound Message 1 Interrupt - This bit is set by the MU hardware when the Inbound Message 1 Register has been written.
00	0_2	Inbound Message 0 Interrupt - This bit is set by the MU hardware when the Inbound Message 0 Register has been written.

4.7.5 Inbound Interrupt Mask Register - IIMR

The Inbound Interrupt Mask Register (IIMR) provides the ability to mask Intel XScale® processor interrupts generated by the Messaging Unit. Each bit in the Mask register corresponds to an interrupt bit in the Inbound Interrupt Status Register.

Setting or clearing bits in this register does not affect the Inbound Interrupt Status Register. They only affect the generation of the Intel XScale® processor interrupt.

Table 270. Inbound Interrupt Mask Register - IIMR



MU/PCI Base Address Offset
IIMR: 0028H

internal bus address offset
IIMR: 4028H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

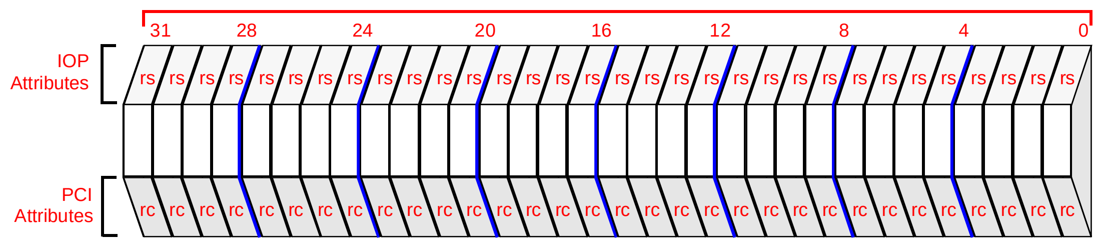
Bit	Default	Description
31	0_2	Coordinated Reset Interrupt Mask - When set, this bit masks the interrupt generated by the Coordinated Reset bit (bit 1) in the Inbound Reset Control and Status Register is set.
30	0_2	Selective Reset Interrupt Mask - When set, this bit masks the interrupt generated by the Selective Reset bit (bit 0) in the Inbound Reset Control and Status Register is set.
29	0_2	MU MSI-X Table Write Interrupt Mask - Controls the setting of bit 29 of the "Inbound Interrupt Status Register - IISR" and generation of the MU MSI-X Table Write Interrupt when a write occurs to any of the MU MSI-X Table entry. 0 = Not Masked 1 = Masked
28:07	000000H	Reserved
06	0_2	Index Register Interrupt Mask - When set, this bit masks the interrupt generated by the MU hardware when an Index Register has been written after a Host I/O Interface transaction.
05	0_2	Outbound Free Queue Full Interrupt Mask - When set, this bit masks the Error interrupt generated when the Outbound Free Head Pointer becomes equal to the Tail Pointer and the queue is full.
04	0_2	Inbound Post Queue Interrupt Mask - When set, this bit masks the interrupt generated by the MU hardware when the Inbound Post Queue has been written.
03	0_2	Error Doorbell Interrupt Mask - When set, this bit masks the Error Interrupt when the Error Interrupt bit of the Inbound Doorbell Register is set.
02	0_2	Inbound Doorbell Interrupt Mask - When set, this bit masks the interrupt generated when at least one Normal Interrupt bit in the Inbound Doorbell Register is set.
01	0_2	Inbound Message 1 Interrupt Mask - When set, this bit masks the Inbound Message 1 Interrupt generated by a write to the Inbound Message 1 Register.
00	0_2	Inbound Message 0 Interrupt Mask - When set, this bit masks the Inbound Message 0 Interrupt generated by a write to the Inbound Message 0 Register.

4.7.6 Outbound Doorbell Register - ODR

The Outbound Doorbell Register (ODR) allows software interrupt generation. It allows the Intel XScale® processor to generate Host I/O Interface interrupts to the host processor by writing to the Software Interrupt bits or to a specific Host I/O Interface interrupt bit. The generation of Host I/O Interface interrupts through the Outbound Doorbell Register may be masked by setting the Outbound Doorbell Interrupt Mask bit in the Outbound Interrupt Mask Register.

The Software Interrupt bits in this register can only be set by the Intel XScale® processor and can only be cleared by an external Host I/O Interface agent.

Table 271. Outbound Doorbell Register - ODR



MU/PCI Base Address Offset — ODR: 002CH

internal bus address offset — ODR: 402CH

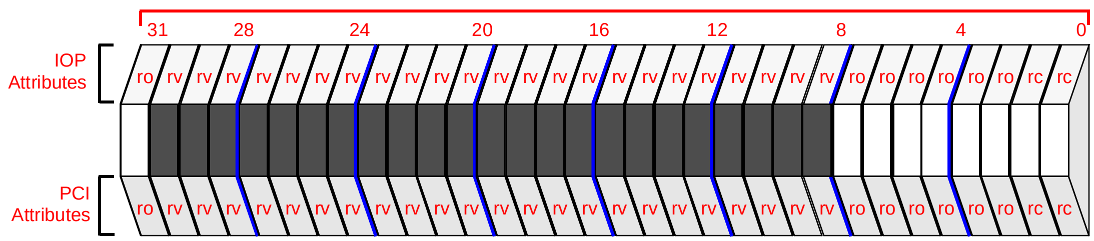
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31	0_2	PCI Interrupt D - When set, this bit causes the **P_INTD#** signal to be asserted or a Message-signaled Interrupt is generated (when enabled). When this bit is cleared, the **P_INTD#** signal is deasserted.
30	0_2	PCI Interrupt C - When set, this bit causes the **P_INTC#** signal to be asserted or a Message-signaled Interrupt is generated (when enabled). When this bit is cleared, the **P_INTC#** signal is deasserted.
29	0_2	PCI Interrupt B - When set, this bit causes the **P_INTB#** signal to be asserted or a Message-signaled Interrupt is generated (when enabled). When this bit is cleared, the **P_INTB#** signal is deasserted.
28	0_2	PCI Interrupt A - When set, this bit causes the **P_INTA#** signal to be asserted or a Message-signaled Interrupt is generated (when enabled). When this bit is cleared, the **P_INTA#** signal is deasserted.
27:00	000000H	Software Interrupt - When any bit is set, generate a PCI interrupt or a Message-signaled interrupt is generated (when enabled). The PCI interrupt pin used is determined by the ATU Interrupt Pin Register. When all bits are clear, do not generate a PCI interrupt.

4.7.7 Outbound Interrupt Status Register - OISR

The Outbound Interrupt Status Register (OISR) contains hardware interrupt status. It records the status of Host I/O Interface interrupts generated by the Message Registers, Doorbell Registers, and the Circular Queues. The generation of Host I/O Interface interrupts recorded in the Outbound Interrupt Status Register may be masked by setting the corresponding bit in the Outbound Interrupt Mask Register. Some of the bits in this register are Read Only. For those bits, the interrupt must be cleared through another register.

Table 272. Outbound Interrupt Status Register - OISR



MU/PCI Base Address Offset
OISR: 0030H

internal bus address offset
OISR: 4030H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

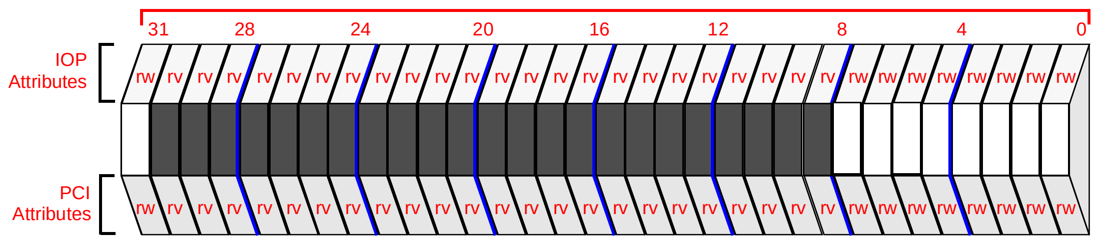
Bit	Default	Description
31	0_2	Firmware Interrupt Pending - This bit is set when the Firmware Interrupt bit is set in the Outbound Reset Control and Status Register. To clear this bit (and the interrupt), the Firmware Interrupt bit must be cleared in the Outbound Reset Control and Status Register.
30:08	000000H 000_2	Reserved
07	0_2	PCI Interrupt D - This bit is set when the PCI Interrupt D bit is set in the Outbound Doorbell Register. To clear this bit (and the interrupt), the PCI Interrupt D bit must be cleared.
06	0_2	PCI Interrupt C - This bit is set when the PCI Interrupt C bit is set in the Outbound Doorbell Register. To clear this bit (and the interrupt), the PCI Interrupt C bit must be cleared.
05	0_2	PCI Interrupt B - This bit is set when the PCI Interrupt B bit is set in the Outbound Doorbell Register. To clear this bit (and the interrupt), the PCI Interrupt B bit must be cleared.
04	0_2	PCI Interrupt A - This bit is set when the PCI Interrupt A bit is set in the Outbound Doorbell Register. To clear this bit (and the interrupt), the PCI Interrupt A bit must be cleared.
03	0_2	Outbound Post Queue Interrupt - This bit is set when data in the prefetch buffer is valid. This bit is cleared when any prefetch data has been read from the Outbound Queue Port.
02	0_2	Outbound Doorbell Interrupt - This bit is set when at least one Software Interrupt bit in the Outbound Doorbell Register is set. To clear this bit (and the interrupt), the Software Interrupt bits in the Outbound Doorbell Register must all be clear.
01	0_2	Outbound Message 1 Interrupt - This bit is set by the MU when the Outbound Message 1 Register is written. Clearing this bit clears the interrupt.
00	0_2	Outbound Message 0 Interrupt - This bit is set by the MU when the Outbound Message 0 Register is written. Clearing this bit clears the interrupt.

4.7.8 Outbound Interrupt Mask Register - OIMR

The Outbound Interrupt Mask Register (OIMR) provides the ability to mask outbound Host I/O Interface interrupts generated by the Messaging Unit. Each bit in the mask register corresponds to a hardware interrupt bit in the Outbound Interrupt Status Register. When the bit is set, the Host I/O Interface interrupt is not generated. When the bit is clear, the interrupt is allowed to be generated.

Setting or clearing bits in this register does not affect the Outbound Interrupt Status Register. They only affect the generation of the Host I/O Interface interrupt.

Table 273. Outbound Interrupt Mask Register - OIMR



MU/PCI Base Address Offset
OIMR: 0034H

internal bus address offset
OIMR: 4034H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31	0_2	Firmware Interrupt Mask - When set, this bit masks the Firmware Reset interrupt signal when the Firmware interrupt bit in the Outbound Reset Control and Status Register is set.
30:05	000000H	Reserved
07	0_2	PCI Interrupt D Mask - When set, this bit masks the PCI Interrupt D signal when the PCI Interrupt D bit in the in the Outbound Doorbell Register is set.
06	0_2	PCI Interrupt C Mask - When set, this bit masks the PCI Interrupt C signal when the PCI Interrupt C bit in the in the Outbound Doorbell Register is set.
05	0_2	PCI Interrupt B Mask - When set, this bit masks the PCI Interrupt B signal when the PCI Interrupt B bit in the in the Outbound Doorbell Register is set.
04	0_2	PCI Interrupt A Mask - When set, this bit masks the PCI Interrupt A signal when the PCI Interrupt A bit in the in the Outbound Doorbell Register is set.
03	0_2	Outbound Post Queue Interrupt Mask - When set, this bit masks the PCI interrupt generated when data in the prefetch buffer is valid.
02	0_2	Outbound Doorbell Interrupt Mask - When set, this bit masks the Software Interrupt generated by the Outbound Doorbell Register.
01	0_2	Outbound Message 1 Interrupt Mask - When set, this bit masks the Outbound Message 1 Interrupt generated by a write to the Outbound Message 1 Register.
00	0_2	Outbound Message 0 Interrupt Mask- When set, this bit masks the Outbound Message 0 Interrupt generated by a write to the Outbound Message 0 Register.

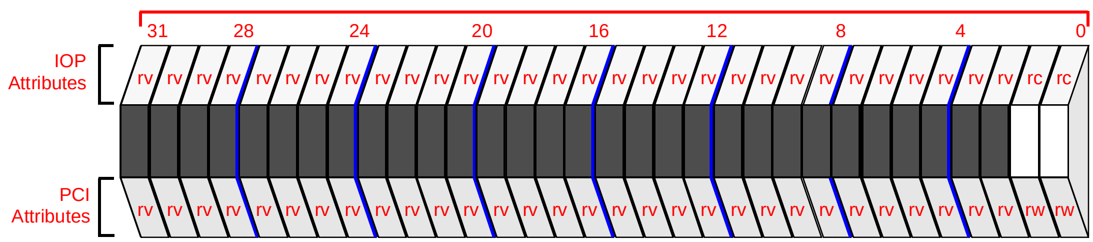
4.7.9 Inbound Reset Control and Status Register - IRCSR

The Inbound Reset Control and Status Register (IRCSR) provides the ability for the Host processor to request a Selective Reset or a Coordinate Reset. A selective reset is used to perform a soft reset. A selective reset is requested by the host processor setting the Selective Reset bit of the IRCSR, which causes an Intel XScale® processor interrupt.

A coordinated reset is used when supporting multiple PCI functions. In a multi-function scenario, before the Host driver can issue a hardware reset via one of the PCI functions, all the host drivers running must be quiesced. A coordinated reset is requested by the host processor setting the Coordinated Reset bit in the IRCSR, which causes an Intel XScale® processor interrupt. After all the host drivers have been quiesced, the host driver that initiated the coordinated reset would request a hardware reset of the 4138xx. When the host driver sets both the Selective Reset and Coordinated Reset bits simultaneously, an internal bus reset is initiated.

Note: An internal bus reset event that is caused by setting both the CR and SR bits of the IRCSR is indicated in the Reset Cause and Status Register (RCSR). For example, after an internal bus reset the cause of the internal bus reset can be identified by reading the RCSR. Refer to the Chapter 7.0, "System Controller (SC) and Internal Bus Bridge" for detailed descriptions of the RCSR register.

Table 274. Inbound Reset Control and Status Register - IRCSR



MU/PCI Base Address Offset
OIMR: 0038H

internal bus address offset
OIMR: 4038H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

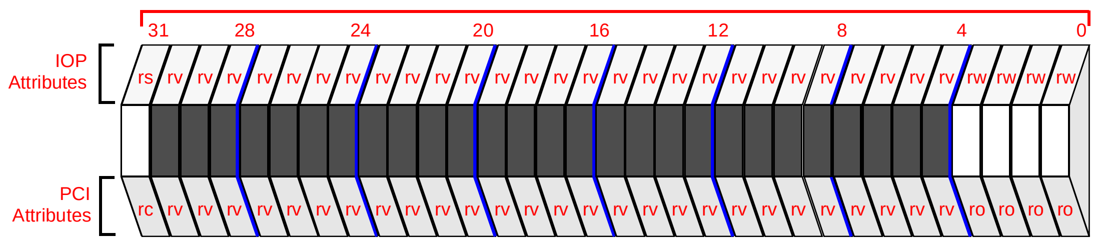
Bit	Default	Description
31:02	00000000H	Reserved.
01	0_2	Coordinated Reset (CR) - This bit is valid when the 4138xx implements Multi-Function Reset protocol, (MF=1, bit 1 in the ORCSR). A Host driver writes a one into this bit to request a Coordinated Reset. Writing a 1 to the CR bit would generate an interrupt to the Intel XScale® processor. However, writing 1's to the CR bit and the SR bit simultaneously would instead generate an internal bus reset. In a Coordinated Reset (only CR bit set), each function is notified and acknowledges the internal bus reset before it actually occurs. In an internal bus reset, the reset proceeds immediately without any notification to the any of the other Hosts.
00	0_2	Selective Reset (SR) - The Host driver sets this bit in order to request a firmware state and structures re-initialization. Setting this bit generates an interrupt to the Intel XScale® processor. However, writing 1's to the CR bit and the SR bit simultaneously would instead generate an internal bus reset. Setting the SR bit by itself does not generate an internal bus reset. Therefore, the Intel XScale® processor must be operational for the reset to be processed. Setting this bit has no impact on any other functions on 4138xx.

4.7.10 Outbound Reset Control and Status Register - ORCSR

The Outbound Reset Control and Status Register (ORCSR) provides the ability for the I/O-processor to coordinate a hardware reset with the Host processor when multi-function is being used. In a multi-function scenario, before the Host driver can issue a hardware reset via one of the functions, all the host drivers running must be quiesced.

Note: GRO and RM bits of the ORCSR are saved in the Reset Cause Status Register (RCSR) when both CR and SR bits of the IRCSR are set to cause an internal bus reset. The Reset Cause Status Register implements sticky bits. The initiator of the internal bus reset is also indicated in the RCSR. Refer to the Exception Initiator and Boot Sequence Chapter for detailed descriptions of the RCSR register.

Table 275. Outbound Reset Control and Status Register - ORCSR



MU/PCI Base Address Offset: OIMR: 003CH
internal bus address offset: OIMR: 403CH

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31	0_2	Firmware Interrupt - This bit is set by the firmware to initiate an interrupt to the Host processor and this bit is used during a coordinated reset handshaking sequence. Firmware sets this bit. The host clears this bit by writing a 1.
30:04	00000H	Reserved.
03	0_2	Global Reset Outstanding (GRO) - This bit is only used when the 4138xx implements Multi-Function Reset protocol (MF=1, bit 1 in the ORCSR). When the Host Processor request a coordinated reset in one of the Function Interfaces, and the 4138xx implements Multi-function Reset protocol, the 4138xx firmware sets the GRO bit on all function interfaces. When Multi-Function Reset protocol is not implemented, this bit is always zero. ***Note:*** The state of this bit is saved in the Reset Cause Status Register (RCSR) when an internal bus reset occurs. Refer to the Exception Initiator and Boot Sequence Chapter for detailed descriptions of the RCSR register.
02	0_2	Reset Coordination Complete (RCC) - This bit is used during a Coordinated Reset to indicate to the Reset Master that all the other Host Drivers have acknowledged the pending reset and that the Reset Master may proceed with the reset. The Coordinated Reset is only available on 4138xx that implement the Multi-function Feature (MF=1, bit 1 in the ORCSR). This bit is only set by the Core when GRO and RM are both set.
01	0_2	Multi-Function (MF) - The Intel XScale® processor sets this bit to indicate that the 4138xx supports multiple PCI function features such as multi-function reset.
00	0_2	Reset Master (RM) - This bit is set by firmware when one of the Host drivers requests a Coordinated Reset and this Interface takes on the Reset Master role during the Coordinated Reset. This feature is only available when the MF bit is set, bit 1 in the ORCSR. The Host driver requests a Coordinated Reset by writing a 1 into the CR bit, bit 1 of the IRCSR. ***Note:*** The state of this bit is saved in the Reset Cause Status Register (RCSR) when an internal bus reset occurs. Refer to the Exception Initiator and Boot Sequence Chapter for detailed descriptions of the RCSR register.

4.7.11 MSI Inbound Message Register — MIMR

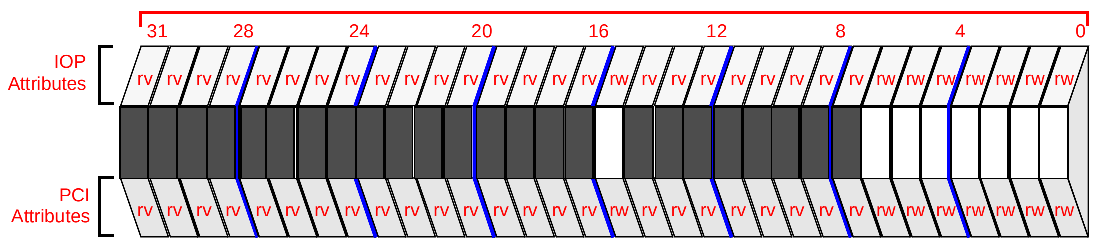
The MSI Inbound Message (MIMR) is a 16-bit data register that can be used to receive inbound MSI interrupt (Message-Signaled Interrupt) from external PCI devices. When operating as a Root Complex (ATU-E) or Central Resource (ATU-X) device, an external PCI device can signal an interrupt by writing the MSI Inbound Message Register. The MU interprets the data bits of MIMR as follows:

— Bit[15] of MIMR is used to select the Intel XScale® processors to be the target of the MSI interrupt. For example, the MSI interrupt can be targeted to either Intel XScale® processor 0 (coreID = 0) or Intel XScale® processor 1 (coreID = 1). Refer to the EI_BS chapter for more details on coreID assignments.

— Bits[14:07] of MIMR are reserved.

Bits[06:00] of MIMR are treated as a vector field which is one-hot decoded and the bits are posted in Intel XScale® processor co-processor registers. Refer to Section 4.7.32, "Inbound MSI Interrupt Pending Register x — IMIPRx" for more details on IMIPR[0:3].

Table 276. MSI Inbound Message Register - MIMR



MU/PCI Base Address Offset
MIMR: 0048H

internal bus address offset
MIMR: 4048H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

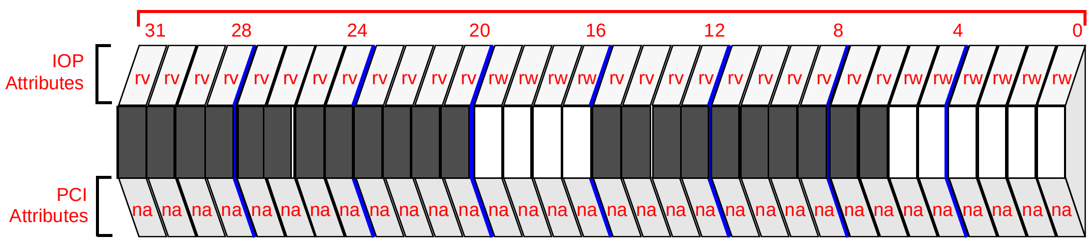
Bit	Default	Description
31:16	0000H	Reserved
15	0_2	Core Select Bit - This bit is used to select the Intel XScale® processor which is the target of the MSI Interrupt. 0 = Intel XScale® processor0 (coreID == 0) 1 = Intel XScale® processor1 (coreID == 1)
14:07	00H	Reserved
06:00	0000000_2	Interrupt Vector - This bit field is used to generate a one-hot bit which is posted in Intel XScale® processor co-processor registers (Section 4.7.32, "Inbound MSI Interrupt Pending Register x — IMIPRx"). Each interrupt vector value is associated with one bit in the co-processor registers. For example, interrupt vector value == 0 is associated with bit 0 of IMIPR0, interrupt vector value == 32 is associated with bit 0 of IMIPR1, interrupt vector value == 64 is associated with bit 0 of IMIPR2, and so on. Refer to Section 4.7.32, "Inbound MSI Interrupt Pending Register x — IMIPRx" for more details on IMIPR[0:3].

4.7.12 MU Configuration Register - MUCR

The MU Configuration Register (MUCR) contains the Circular Queue Enable bit and the size of one Circular Queue. The Circular Queue Enable bit enables or disables the Circular Queues. The Circular Queues are disabled at reset to allow the software to initialize the head and tail pointer registers before any PCI accesses to the Queue Ports. Each Circular Queue may range from 4 K entries (16 Kbytes) to 64 K entries (256 Kbytes) and there are four Circular Queues.

This register also contains the upper four bits of the 36-bit QBR address. Local memory is 36-bit addressable.

Table 277. MU Configuration Register - MUCR



internal bus address offset

MUCR 4050H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

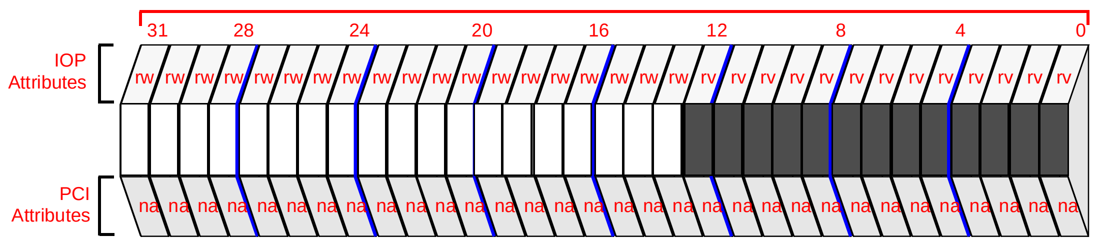
Bit	Default	Description
31:20	000H	Reserved
19:16	0H	Reserved
15:06	000000H	Reserved
05:01	00001_2	Reserved
00	0_2	Must be 0 for 4138xx.

4.7.13 MU Base Address Register - MUBAR

MU Base Address Register (MUBAR) contains lower 32-bit of the 36-bit local memory base address of the MU address space as depicted in Figure 39. For example, the MU address space as viewed from Host I/O Interface. The MU base address is required to be located on an 8-KByte boundary. The upper four-bits of the MU Base Address are located in the MU Upper Address Register (MUBAR). Refer to Section 4.7.14, "MU Upper Base Address Register - MUUBAR". The MU always claim the entire 8 KBytes of the internal bus address space relative to the MU Base Address Registers. The 8 KBytes include the MU Registers and MSI-X Data Structures.

Note: The default values of MUBAR/MUBAR are programmed to match the default values programmed in the Inbound ATU Translate Value Register 0 - IATVR0/Inbound ATU Upper Translate Value Register 0 - IAUTVR0. This allows the MU registers to be mapped in the first 8-KByte of the PCI Window 0 Address space.

Table 278. MU Base Address Register - MUBAR



internal bus address offset

MUBAR 4084H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

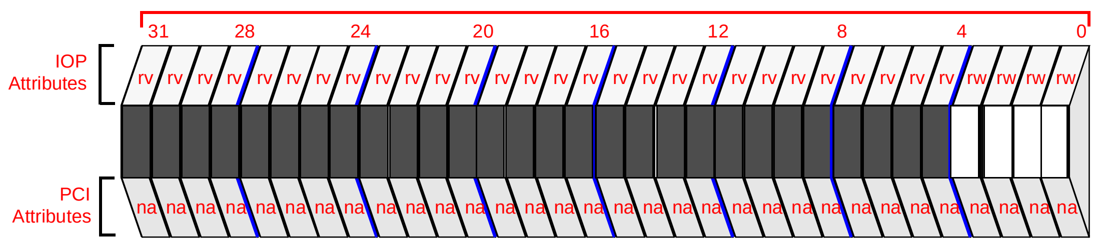
Bit	Default	Description
31:13	FF000H	MU Base Address - Local memory address of the MU (including MSI-X) registers as viewed from Host I/O Interface. The MU occupies a 8-KByte space from Host I/O Interface.
12:00	000H	Reserved

4.7.14 MU Upper Base Address Register - MUUBAR

The MU Upper Base Address Register (MUUBAR) contains the upper 4-bit of the 36-bit local memory base address of the MU address space as depicted in Figure 39. For example, the MU address space as viewed from Host I/O Interface interface. The MU base address is required to be located on a 8-KByte boundary. The lower 32-bits of the MU Base Address are located in the MU Upper Address Register (MUBAR). Refer to Section 4.7.13, "MU Base Address Register - MUBAR" on page 423.

Note: The default values of MUBAR/MUBAR are programmed to match the default values programmed in the Inbound ATU Translate Value Register 0 - IATVR0/Inbound ATU Upper Translate Value Register 0 - IAUTVR0. This allows the MU registers to be mapped in the first 8-KByte of the PCI Window 0 Address space.

Table 279. MU Upper Base Address Register - MUUBAR



internal bus address offset

MUUBAR 4088H

Attribute Legend:	RW = Read/Write
RV = Reserved	RC = Read Clear
PR = Preserved	RO = Read Only
RS = Read/Set	NA = Not Accessible

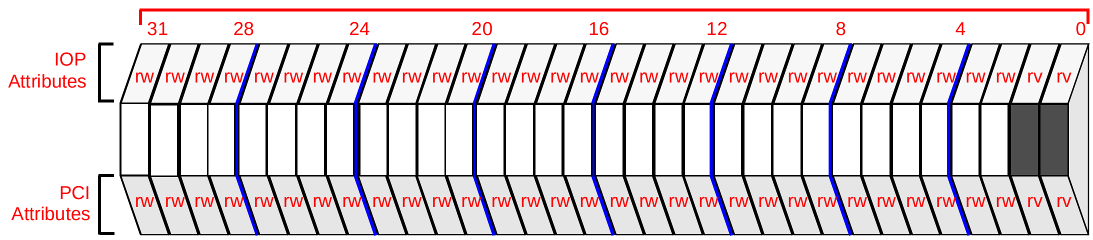
Bit	Default	Description
31:04	0000000H	Reserved
03:00	0H	MU Upper Base Address - Local memory address of the MU (including MSI-X) registers as viewed from Host I/O Interface. The MU occupies a 8-KByte space from Host I/O Interface.

4.7.15 MU MSI-X Table Message Address Registers - M_MT_MAR[0:7]

The MU MSI-X Table Message Address Register contains the lower 30 bits of the MSI-X message address. An entry in the MSI-X Table is made up of four DWORDs.

Note: The M_MT_MAR[0:7] registers are not reset with an internal bus reset.

Table 280. MU MSI-X Table Message Address Registers - M_MT_MAR [0:7]



	MU/PCI Base Address Offset		internal bus address offset
M_MT_MAR0	1000H	M_MT_MAR0	5000H
M_MT_MAR1	1010H	M_MT_MAR1	5010H
M_MT_MAR2	1020H	M_MT_MAR2	5020H
M_MT_MAR3	1030H	M_MT_MAR3	5030H
M_MT_MAR4	1040H	M_MT_MAR4	5040H
M_MT_MAR5	1050H	M_MT_MAR5	5050H
M_MT_MAR6	1060H	M_MT_MAR6	5060H
M_MT_MAR7	1070H	M_MT_MAR7	5070H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

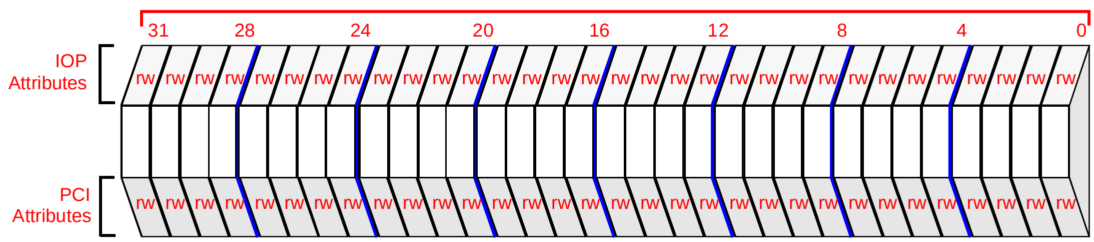
Bit	Default	Description
31:02	0000 0000H	Message Lower Address: This field contains the lower 30 bits of the MSI-X Message Address.
01:00	00_2	Reserved.

4.7.16 MU MSI-X Table Message Upper Address Registers - M_MT_MUAR[0:7]

The MU MSI-X Table Message Upper Address Register contains the upper 32 bits of the MSI-X message address. An entry in the MSI-X Table is made up of four DWORDs.

Note: The M_MT_MUAR[0:7] registers are not reset with an internal bus reset.

Table 281. MU MSI-X Table Message Upper Address Registers - M_MT_MUAR [0:7]



	MU/PCI Base Address Offset		internal bus address offset
M_MT_MAR0	1004H	M_MT_MAR0	5004H
M_MT_MAR1	1014H	M_MT_MAR1	5014H
M_MT_MAR2	1024H	M_MT_MAR2	5024H
M_MT_MAR3	1034H	M_MT_MAR3	5034H
M_MT_MAR4	1044H	M_MT_MAR4	5044H
M_MT_MAR5	1054H	M_MT_MAR5	5054H
M_MT_MAR6	1064H	M_MT_MAR6	5064H
M_MT_MAR7	1074H	M_MT_MAR7	5074H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

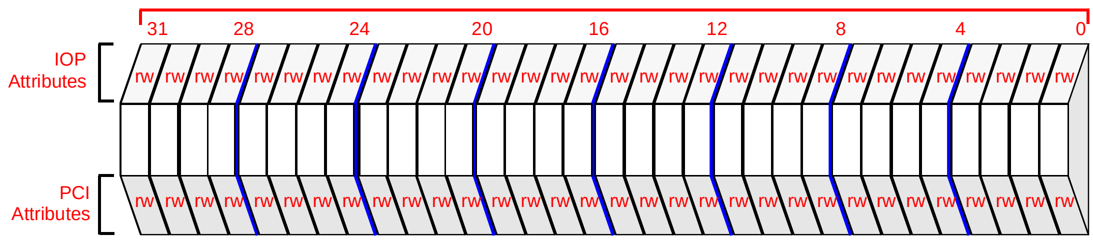
Bit	Default	Description
31:00	0000 0000H	Message Upper Address: This field contains the upper 32 bits of the MSI-X Message Address.

4.7.17 MU MSI-X Table Message Data Registers - M_MT_MDR[0:7]

The MU MSI-X Table Message Data Register contains the message data of the MSI-X message. An entry in the MSI-X Table is made up of four DWORDs.

Note: The M_MT_MDR[0:7] registers are not reset with an internal bus reset.

Table 282. MU MSI-X Table Message Upper Address Registers - M_MT_MUAR [0:7]



MU/PCI Base Address Offset		internal bus address offset	
M_MT_MAR0	1008H	M_MT_MAR0	5008H
M_MT_MAR1	1018H	M_MT_MAR1	5018H
M_MT_MAR2	1028H	M_MT_MAR2	5028H
M_MT_MAR3	1038H	M_MT_MAR3	5038H
M_MT_MAR4	1048H	M_MT_MAR4	5048H
M_MT_MAR5	1058H	M_MT_MAR5	5058H
M_MT_MAR6	1068H	M_MT_MAR6	5068H
M_MT_MAR7	1078H	M_MT_MAR7	5078H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

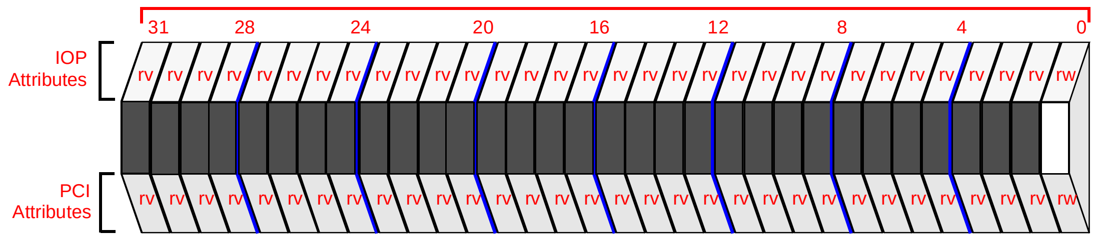
Bit	Default	Description
31:00	0000 0000H	Message Data: This field contains the message data for this Table entry.

4.7.18 MU MSI-X Table Message Vector Control Registers - M_MT_MVCR[0:7]

The MU MSI-X Table Message Vector Control Register contains the mask bit for this entry in the MSI-X Table. An entry in the MSI-X Table is made up of four DWORDs.

Note: The M_MT_MVCR[0:7] registers are not reset with an internal bus reset.

Table 283. MU MSI-X Table Message Vector Control Registers - M_MT_MVCR [0:7]



	MU/PCI Base Address Offset		internal bus address offset
M_MT_MAR0	100CH	M_MT_MAR0	500CH
M_MT_MAR1	101CH	M_MT_MAR1	501CH
M_MT_MAR2	102CH	M_MT_MAR2	502CH
M_MT_MAR3	103CH	M_MT_MAR3	503CH
M_MT_MAR4	104CH	M_MT_MAR4	504CH
M_MT_MAR5	105CH	M_MT_MAR5	505CH
M_MT_MAR6	106CH	M_MT_MAR6	506CH
M_MT_MAR7	107CH	M_MT_MAR7	507CH

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

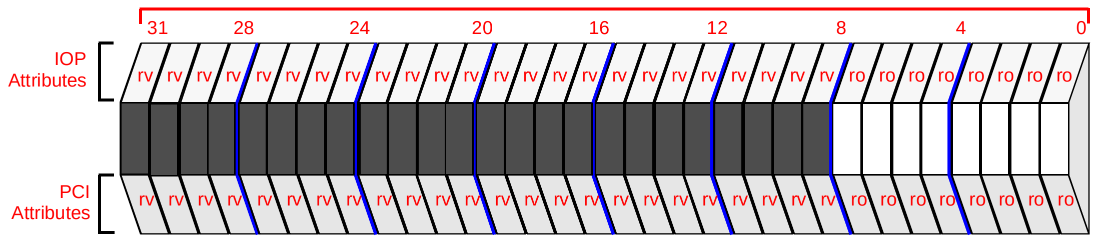
Bit	Default	Description
31:01	0000 0000H	Reserved.
00	1_2	Message Vector Control: This bit when set, prohibits the sending an MSI-X message using this entry.

4.7.19 MU MSI-X Pending Bits Array Register - M_MPBAR

The MU MSI-X Pending Bits Array Register contains the contains the pending bits for the eight MU interrupt sources. When an entry in the MSI-X table is masked in the Vector Control Register, the software may service that interrupt request by polling the pending bit.

Note: The M_MPBAR register is not reset with an internal bus reset.

Table 284. MU MSI-X Pending Bits Array Register - M_MPBAR



	MU/PCI Base Address Offset		internal bus address offset
M_MPBAR	1800H	M_MPBAR	5800H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

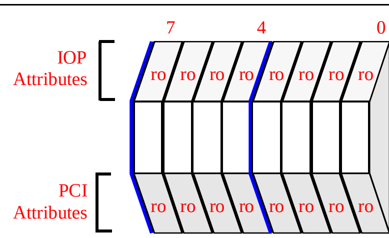
Bit	Default	Description
31:08	0000 000H	Reserved.
07:00	00000000_2	Pending Bits Array: Any bit that is set indicates that the associated MSI-X message is scheduled to be sent. When an associated entry in the MSI-X table is masked using the mask bit in the Vector Control Register, software can service the interrupt based on polling the associated pending bit.

4.7.20 MSI Capability Identifier Register - Cap_ID

The MSI Capability Identifier Register bits adhere to the definitions in the *PCI Local Bus Specification*, Revision 2.2. This register in the PCI Interface Extended Capability header identifies the type of Extended Capability contained in that header. The value of 05H in this field identifies the function as message signaled interrupt capable.

Note: Refer to the Peripheral Registers Chapter for the default internal bus address. This register is part of the configuration space of the Address Translation Unit that is setup as an endpoint.

Table 285. MSI Capability Identifier Register - MSI_Cap_ID



Internal Bus Address Offset	PCI Configuration Offset
0A0H	A0H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

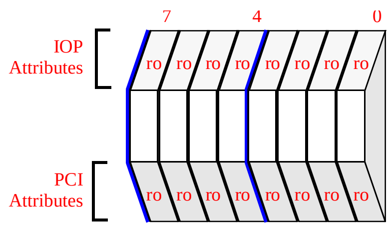
Bit	Default	Description
07:00	05H	MSI_Cap_Id - This field with its 05H value identifies this item in the linked list of Extended Capability Headers as being the message signaled interrupt capability item.

4.7.21 MSI Next Item Pointer Register - MSI_Next_Ptr

The Next Item Pointer Register bits adhere to the definitions in the *PCI Local Bus Specification*, Revision 2.2. This register describes the location of the next item in the function capability list. For the 4138xx that is the PCI-X capability header at offset E0H.

Note: Refer to the Peripheral Registers Chapter for the default internal bus address. This register is part of the configuration space of the Address Translation Unit that is setup as an endpoint.

Table 286. MSI Next Item Pointer Register - MSI_Next_Ptr



Internal Bus Address Offset
0A1H

PCI Configuration Offset
A1H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

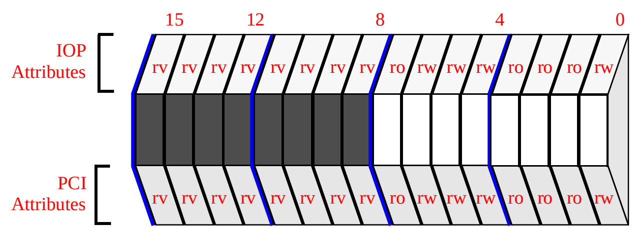
Bit	Default	Description
07:00	D0H	MSI_Next_ Pointer - This field provides an offset into the function configuration space pointing to the next item in the function capability list

4.7.22 Message Control Register - Message_Control

The Message Control Register provides system software control over MSI. After reset, MSI is disabled. System software is permitted to modify the Message Control register's read/write bits and fields while a device driver is not permitted to modify them.

Note: Refer to the Peripheral Registers Chapter for the default internal bus address. This register is part of the configuration space of the Address Translation Unit that is setup as an endpoint.

Table 287. Message Control Register - Message_Control



Internal Bus Address Offset
0A2H

PCI Configuration Offset
A2H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

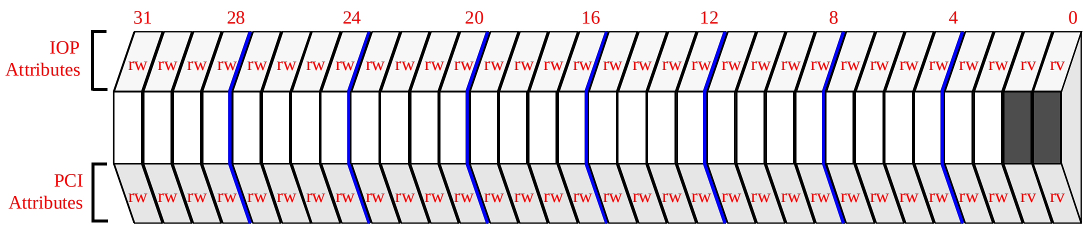
Bit	Default	Description
15:8	00H	Reserved
7	1_2	64-bit Address Support - This field is set to 1_2 indicating that the 4138xx is capable of generating a 64-bit message address.
6:4	000_2	Multiple Message Enable - System software writes to this field to indicate the number of messages allocated to the 4138xx. While, the 4138xx requests two messages, it is possible that system software only allocates one message. The device hardware is designed to handle both cases.
3:1	001_2	Multiple Message Capable - This field is set to 001_2 indicating that the 4138xx can issue up to two unique interrupt messages.
0	0_2	MSI Enable - Setting this bit enables the 4138xx MSI functionality and disables the use of the **P_INTA#** interrupt output for 4138xx interrupts.

4.7.23 Message Address Register - Message_Address

The Message address register specifies the DWORD aligned address for the MSI memory write transaction. The value is set by system software during initialization.

Note: Refer to the Peripheral Registers Chapter for the default internal bus address. This register is part of the configuration space of the Address Translation Unit that is setup as an endpoint.

Table 288. Message Address Register - Message_Address



Internal Bus Address Offset 0A4H

PCI Configuration Offset A4H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

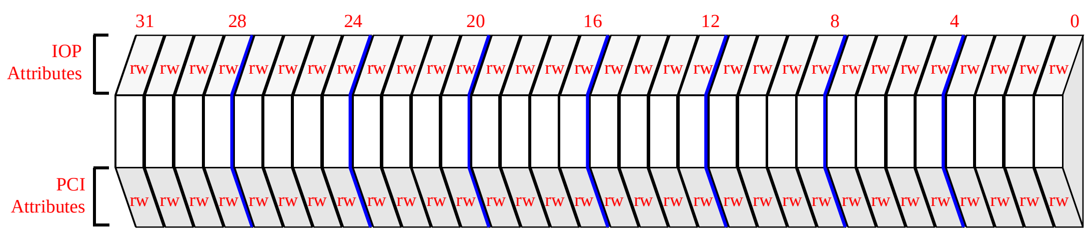
Bit	Default	Description
31:2	00000000H	Message Address - DWORD aligned Message Address. This value is set by system software.
1:0	00_2	Reserved.

4.7.24 Message Upper Address Register - Message_Upper_Address

The Message Upper Address register is set during system initialization when system software wishes to place the MSI address location above the 4G address boundary. When this register is set to a non-zero value, the 4138xx generates a dual address cycle for the MSI write command and uses the contents of this register as the upper 32-bits of that address.

Note: Refer to the Peripheral Registers Chapter for the default internal bus address. This register is part of the configuration space of the Address Translation Unit that is setup as an endpoint.

Table 289. Message Upper Address Register - Message_Upper_Address



Internal Bus Address Offset
0A8H

PCI Configuration Offset
A8H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:00	00000000H	Message Upper Address - Upper 32 bits of a 64-bit PCI address. This value is set by system software.

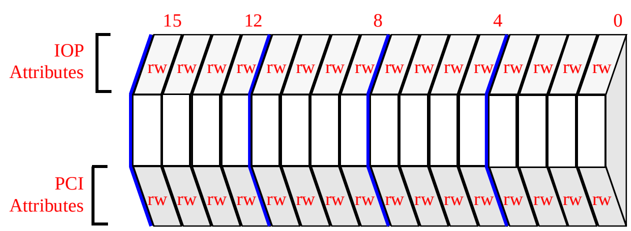
4.7.25 Message Data Register- Message_Data

The value in the Message Data Register contains the data used during an MSI write transaction. When two unique messages are enabled, one message is reserved for the Outbound Post Queue Interrupt and the other message represents all of the Outbound Doorbell and Outbound Message Interrupts. When only one message is enabled, all of these interrupts are represented by a single message. Interrupt handler software needs to read the 4138xx Outbound Interrupt Status Register to determine the cause of the interrupt when more than one source is represented by a single message.

During an MSI write data phase, the value in the Message Data Register is driven on to **AD[15:0]** while **AD[31:16]** are driven to zero. **C/BE[3:0]#** are asserted during the data phase of the memory write transaction.

Note: Refer to the Peripheral Registers Chapter for the default internal bus address. This register is part of the configuration space of the Address Translation Unit that is setup as an endpoint.

Table 290. Message Data Register - Message_Data



Internal Bus Address Offset
0ACH

PCI Configuration Offset
ACH

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

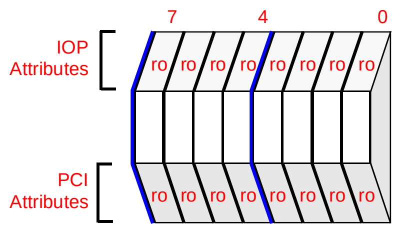
Bit	Default	Description
15:00	0000H	Message Data - System software specifies a 16-bit value to be transferred during the data phase of an MSI write transaction. During initialization, system software can allocate two messages to the 4138xx by writing "001" to the Multiple Message Enable field of the Message Control Register. In this case, the hardware shall modify bit 0 of the Message Data value before generating the MSI write transaction. The Host I/O Interface Outbound interrupts are distributed across the two messages by the setting and clearing of bit 0 of the Message Data Register's contents before the data is used for the MSI write: 0 = Outbound Post Queue Interrupt 1 = Outbound Doorbell and Outbound Message Interrupts When software leaves the Multiple Message Enable field of the Message Control Register at its' default value of "000", the 4138xx has only been allocated one message. Consequently, the value in the MSI data register is transmitted unmodified during an MSI write cycle.

4.7.26 MSI-X Capability Identifier Register - MSI-X_Cap_ID

The Capability Identifier Register bits adhere to the definitions in the *PCI Local Bus Specification*, Revision 2.3. This register in the PCI Extended Capability header identifies the type of Extended Capability contained in that header. In the case of the 4138xx, this is the MSI-X extended capability with an ID of 0DH as defined by the *PCI-X Protocol Addendum to the PCI Local Bus Specification,* Revision 2.0.

Note: Refer to the Peripheral Registers Chapter for the default internal bus address. This register is part of the configuration space of the Address Translation Unit that is setup as an endpoint.

Table 291. MSI-X_Capability Identifier Register - MSI-X_Cap_ID



Internal Bus Address Offset
0B0H

PCI Configuration Offset
B0H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

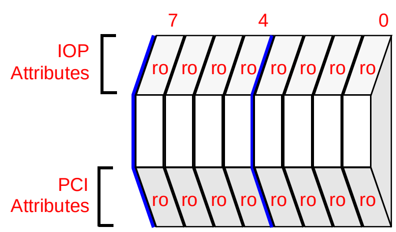
Bit	Default	Description
07:00	0DH	Cap_Id - This field with its' 0DH value identifies this item in the linked list of Extended Capability Headers as being the MSI-X capability registers.

4.7.27 MSI-X Next Item Pointer Register - MSI-X_Next_Item_Ptr

The Next Item Pointer Register bits adhere to the definitions in the *PCI Local Bus Specification*, Revision 2.3. This register describes the location of the next item in the function's capability list. For the 4138xx, the next capability (PCI-X capability list) is located at off-set E0H.

Note: Refer to the Peripheral Registers Chapter for the default internal bus address. This register is part of the configuration space of the Address Translation Unit that is setup as an endpoint.

Table 292. MSI-X Next Item Pointer Register - MSI-X_Next_Item_Ptr



Internal Bus Address Offset
480B1H

PCI Configuration Offset
B1H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

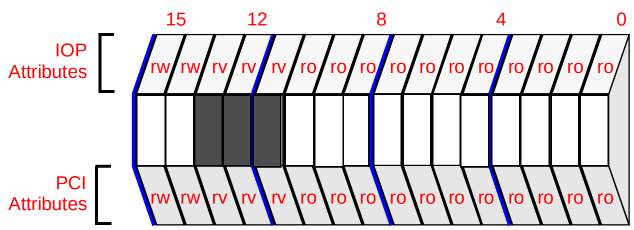
Bit	Default	Description
07:00	A0H	Next_ Item_ Pointer - This field provides an offset into the function's configuration space pointing to the next item in the function's capability list which in the 4138xx is the MSI extended capabilities header.

4.7.28 MSI-X Message Control Register - MSI-X_MCR

MSI-X Capabilities bits adhere to the definitions in the *PCI Local Bus Specification*, Revision 2.3. This register is a 16-bit read-only register which provides information on the capabilities of the ATU function related to Message Signaled Interrupts.

Note: Refer to the Peripheral Registers Chapter for the default internal bus address. This register is part of the configuration space of the Address Translation Unit that is setup as an endpoint.

Table 293. MSI-X Message Control Register - MSI-X_MCR



Internal Bus Address Offset
0B2H

PCI Configuration Offset
B2 - B3H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

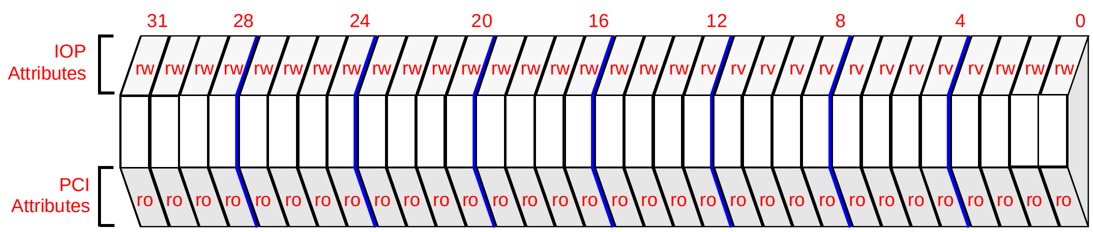
Bit	Default	Description
15	0_2	**MSI-X Enable:** When set, the 4138xx is able to use MSI-X to request service.
14	0_2	**Function Mask:** When set, all the vectors in the MSI-X Table are globally masked, regardless of the per-vector Mask Bit states in the Vector Control Register of the MSI-X Table entries.
13:11	000_2	Reserved
10:00	00000000000_2 or 00000000111_2 (See description for default value)	**MSI-X Table Size:** This field indicates the MSI-X Table size N. This field is encoded as N-1. Up to eight messages can be generated. However, when the MSI-X Single Message Vector bit is set in the "MU MSI-X Control Register X — MMCRx" on page 440, only a single MSI-X message is generated. The value of this register field is dependent on the setting of the MSI-X Single Message Vector bit. MSI-X Single Message Bit — Default Value 0 — 00000000111_2 1 — 00000000000_2

4.7.29 MSI-X Table Offset Register — MSI-X_Table_Offset

This register indicates in which PCI Memory Window the MSI-X Table is mapped. This register also provides an offset in the specified PCI Memory Window of where the MSI-X Table begins.

Note: Refer to the Peripheral Registers Chapter for the default internal bus address. This register is part of the configuration space of the Address Translation Unit that is setup as an endpoint.

Table 294. MSI-X Table Offset Register - MSI-X_Table_Offset



Internal Bus Address Offset
0B4H

PCI Configuration Offset
B4 - B7H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

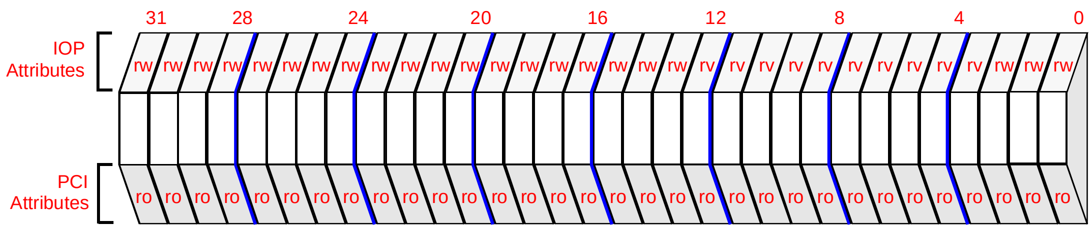
Bit	Default	Description
31:13	0	**MSI-X Table Offset:** Indicates the starting address of the MSI-X Table relative to the address in the Base Address Register indicated bits [2:0] of this register. This part of the MSI-X Table Offset field is programmable and is based on the value programmed in the "MU Base Address Register - MUBAR" on page 423 and the ATU Limit Value Register. The following equation may be used to compute MSI-X Table Offset[31:13]. Note that the Messaging Unit occupies 8-KByte of address space and must overlap the address space defined by the ATU Value and the ATU Limit registers. **Equation:** MSI-X Table Offset[31:13] = {(~ATU Limit_Register[31:0] & MU_Bar[31:0]} >> 13 ***Note:*** The default location of the MU space after reset is in the first 8-KByte of the default ATU Translation Window.
12:3	$1\_0000\_0000\_0_2$	**MSI-X Table Offset:** Indicates the starting address of the MSI-X Table relative to the address in the Base Address Register indicated bits [2:0] of this register. This part of the MSI-X Table Offset field is fixed which forces the table to offset at a 4-KByte offset relative to the "MU Base Address Register - MUBAR" on page 423.
2:0	000_2	**MSI-X Table BAR Indication Register (BIR)**: indicates which Base Address Register of the ATU function the MSI-X Table is mapped into. BIR Value Base Address Register 0 10H 1 14H 2 18H 3 1CH 4 20H 5 24H All other values are reserved.

4.7.30 MSI-X Pending Bit Array Offset Register - MSI-X_PBA_Offset

This register indicates in which PCI Memory Window the MSI-X PBA is mapped. This register also provides an offset in the specified PCI Memory Window of where the MSI-X PBA begins.

Note: Refer to the Peripheral Registers Chapter for the default internal bus address. This register is part of the configuration space of the Address Translation Unit that is setup as an endpoint.

Table 295. MSI-X Pending Bit Array Offset Register - MSI-X_PBA Offset



Internal Bus Address Offset
0B8H

PCI Configuration Offset
B8 -BBH

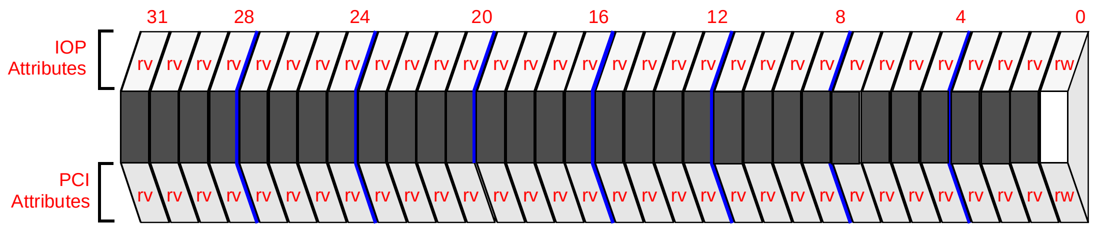
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:13	0	**MSI-X Table Offset:** Indicates the starting address of the MSI-X Table relative to the address in the Base Address Register indicated bits [2:0] of this register. This part of the MSI-X Table Offset field is programmable and is based on the value programmed in the "MU Base Address Register - MUBAR" on page 423 and the ATU Limit Value Register. The following equation may be used to compute MSI-X Table Offset[31:13]. Note that the Messaging Unit occupies 8-KByte of address space and must overlap the address space defined by the ATU Value and the ATU Limit registers. **Equation**: MSI-X Table Offset[31:13] = {(~ATU Limit_Register[31:0] & MU_Bar[31:0]} >> 13. ***Note:*** The default location of the MU space after reset is in the first 8-KByte of the default ATU Translation Window.
12:3	$1\_1000\_0000\_0_2$	**MSI-X Table Offset:** Indicates the starting address of the MSI-X Table relative to the address in the Base Address Register indicated bits [2:0] of this register. This part of the MSI-X Table Offset field is fixed which forces the table to offset at a 6-KByte offset relative to the "MU Base Address Register - MUBAR" on page 423.
2:0	000_2	**PBA BAR Indication Register (BIR)**: indicates which Base Address Register of the ATU function the Pending Bit Array is mapped into. BIR Value Base Address Register 0 10H 1 14H 2 18H 3 1CH 4 20H 5 24H All other values are reserved.

4.7.31 MU MSI-X Control Register X — MMCRx

By default, the MU can generate up to eight MSI-X messages. The MMCRx register provides a control bit that allows collapsing the eight MSI-X messages down to only a single message. When the Host processor cannot honor the eight requested MSI-X messages, the MU MSI-X Single Message Vector bit can be set to cause only a single MSI-X message to be generated.

Table 296. MU MSI-X Control Register X — MMCRx



Register Offset
+0BCH

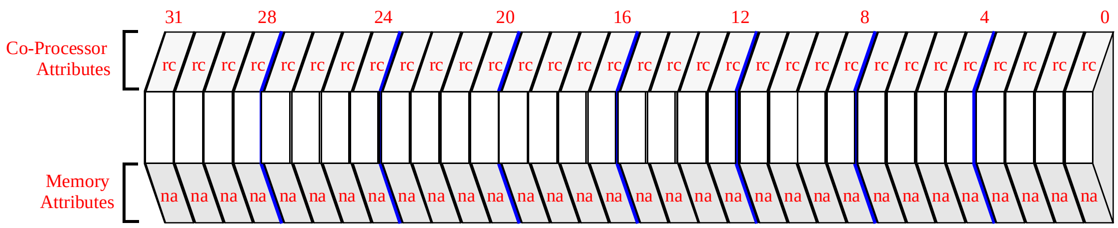
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:01	0000 0000H	Reserved
00	0_2	MU MSI-X Single Message Vector: This bit when set, causes only a single MSI-X Message to be generated when MSI-X is enabled. This bit affects the default value of the MSI-X Table Size field in the MSI-X Message Control Register - MSI-X_MCR.

4.7.32 Inbound MSI Interrupt Pending Register x — IMIPRx

The Inbound MSI Interrupt Pending register is a 32-bit register that is used to post the one-hot decoded bits that are generated by the MU (Messaging Unit) when receiving inbound MSI (Message-Signaled Interrupt). The MU can generate up to 128 interrupts. Refer to the MSI Inbound Message Register — MIMR in the MU Chapter. Any bit set in this register generates an interrupt to the Intel XScale® processor via the Inbound MSI Interrupt pending signal. Software must clear this register to clear any pending interrupt.

Table 297. Inbound MSI Interrupt Pending Registers — IMIPR [0:3]



Intel XScale® processor Local Bus Address Offset coreID0 - n/a coreID1 - n/a	Intel XScale® processor Coprocessor Address IMIPR 0: CP6, CRm 1, CRn 8 IMIPR 1: CP6, CRm 1, CRn 9 IMIPR 2: CP6, CRm 1, CRn 10 IMIPR 3: CP6, CRm 1, CRn 11	Attribute Legend: RV = Reserved PR = Preserved RS = Read/Set	RW = Read/Write RC = Read Clear RO = Read Only NA = Not Accessible

Bit	Default	Description
31:00	0000 0000H	Inbound MSI Pending Interrupts — Each bit reflects an inbound MSI interrupt that was generated by the MU (Messaging Unit). The MU receives, decodes and generates a one-hot bit for every MSI message that it receives. Any bit set generates an interrupt to the Intel XScale® processor via the Inbound MSI Interrupt Pending signal.

4.8 Power/Default Status

The software is responsible for initializing the Circular Queue Size in the MU Configuration Register and all head and tail pointer registers before setting the Circular Queue Enable bit.

5.0 SRAM DMA Unit (SDMA)

5.1 Introduction

This chapter describes the operation and control of the SRAM DMA (SDMA) Unit.

5.2 Overview

The SRAM DMA (SDMA) unit provides a means for memory to be transferred between local memory (SRAM) and host memory through the PCIe bus. The SDMA provides two separate channels (HostToLocal and LocalToHost) that operate independently from one another.

The HostToLocal channel performs DMA operations from Host Memory to Local Memory (SRAM). The LocalToHost channel performs DMA operations from Local Memory to Host Memory (SRAM).

5.3 Theory of Operation

To perform a DMA operation, the firmware writes to either the HostToLocal or LocalToHost registers, depending on the direction of transfer. One DMA is underway in each direction simultaneously.

Each SDMA channel provides a “single-shot” DMA capability, in other words, one DMA operation at a time. There is no ability to queue multiple DMA requests in a given channel.

Before programming any of the registers, firmware must ensure that the DMA channel is not active by ensuring that all previous DMA operations have completed (for example by reading the Interrupt Counter / Interrupt Acknowledge and comparing that against the last DMA operation requested). Firmware then writes the Host Addresses, Local Addresses, and the Byte Count. The firmware then sets the CHGO bit to start the DMA.

Upon completion of the DMA operation, the firmware receives an interrupt. Two interrupts are associated with the SDMA, a Normal and an Error interrupt.

In either case the firmware reads the Interrupt Counter of the channel(s) it has programmed. Note that because the HostToLocal and LocalToHost channels are independent then when both have been programmed the firmware needs to read each channel Interrupt Counter to determine which channel has completed its DMA.

After reading the Interrupt Counter, firmware writes that counter value back to the Interrupt Acknowledge field in the same register. This clears the interrupt, and another DMA on that channel is started.

Errors are indicated by flags within each channel Control/Status registers. Further information on the errors is obtained by examining the ATU or XSI System Controller error registers.

Example 3. Pseudo Code Programming Example: (RedBoot* command line prompts shown)

To perform a single shot DMA of 0x40 from local SRAM offset 0x0 to host address 0x20_0000:

INITIAL SETUP:

1. Assure the registers are enabled for access by setting bit 0 of the control/status register.
   mfill -b 0xFFD9823c -l 0x4 -p 0x1
2. Unmask SDMA error and status registers in INTCTL2 bits 12 and 13.

TO PERFORM A DMA:

1. Check to make sure that the channel is idle by reading the Int Counter/Ack register and assuring the upper and lower counts are equal.
   x -b 0xFFD98238 -l 0x4 -4
2. Set bit 30 in the byte swap register to disable byte swapping.
   mfill -b 0xFFD98200 -l 0x4 -p 0x40000000
3. Set the host destination address lower and upper.
   mfill -b 0xFFD98204 -l 0x4 -p 0x200000
   mfill -b 0xFFD98208 -l 0x4 -p 0x0
4. Set the source offset from the base of SRAM.
   mfill -b 0xFFD9820C -l 0x4 -p 0x0
5. Write the byte counts in both the upper and lower parts of the byte count register.
   mfill -b 0xFFD98218 -l 0x4 -p 0x00400040
6. Write the channel go bit (its OK to assure bit 0 is set at the same time).
   mfill -b 0xFFD9823c -l 0x4 -p 0x3
7. Read the Int Counter/Ack register.
   x -b 0xFFD98238 -l 0x4 -4
8. Assume step 7 resulted in a value of 0x0300002, now write the int counter back to the int Ack.
   mfill -b 0xFFD98238 -l 0x4 -p 0x3
9. Upon INT, check status.
   x -b 0xFFD9823c -l 0x4 -4

5.3.1 Interrupt Control for SDMA

Refer to the silicon C Spec for full register definitions, the following control the SDMA:

INTPND2:
- bit 13: SDMA Error Interrupt Pending
- bit 12: SDMA Normal Interrupt Pending

INTCTL2:
- bit 13: SDMA Error Interrupt Mask.
  - 0 = Masked
  - 1 = Not Masked
- bit 12: SDMA Normal Interrupt Mask.
  - 0 = Masked
  - 1 = Not Masked

INTSTR2
- bit 13: SDMA Error Interrupt Steering.
  - 0 = Interrupt Directed to internal IRQ
  - 1 = Interrupt Directed to internal FIQ
- bit 12: SDMA Normal Interrupt Steering.
  - 0 = Interrupt Directed to internal IRQ
  - 1 = Interrupt Directed to internal FIQ

INTSRC2
- bit 13: SDMA Error Interrupt
  - 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL2
  - 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL2
- bit 12: SDMA Normal Interrupt
  - 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL2
  - 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL2

FINTSRC2
- bit 13: SDMA Error Interrupt
  - 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL2
  - 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL2
- bit 12: SDMA Normal Interrupt
  - 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL2
  - 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL2

IPR
- 27:26: SDMA Error Interrupt Priority
- 25:24: SDMA Normal Interrupt Priority

5.4 Register Definitions

The SDMA controller contains separate LocalToHost (L2H) and HostToLocal (H2L) channels that are independent of each other. These are used simultaneously thus allowing full duplex transfer to occur.

The location of these registers are specified as a relative offset to a 512KB aligned global PMMR offset. The default for the 512KB aligned offset is 0 FFD8 0000H defined by the PMMRBAR register.

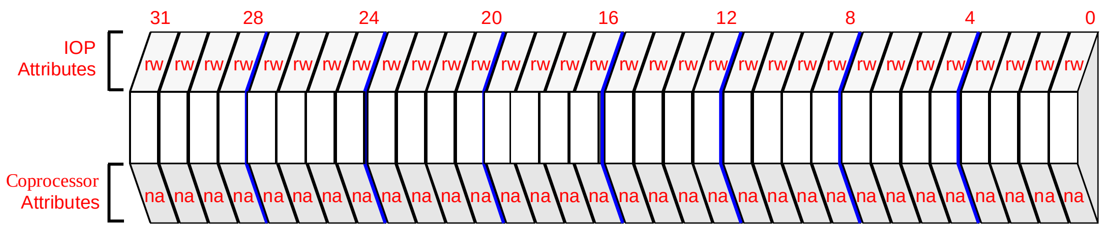
Table 298. SDMA Controller Unit Registers

Section, Register Name, Acronym, Page
Section 5.4.1, "LocalToHost Destination Lower Address Register - L2H_DLAR" on page 447
Section 5.4.2, "LocalToHost Destination Upper Address Register - L2H_DUAR" on page 447
Section 5.4.3, "LocalToHost Source Lower Address Register - L2H_SLAR" on page 448
Section 5.4.4, "LocalToHost Byte Count Register - L2H_BCR" on page 449
Section 5.4.5, "LocalToHost Interrupt Counter/Acknowledge Register L2H_ICAR" on page 450
Section 5.4.6, "LocalToHost Control/Status Register - L2H_CSR" on page 451
Section 5.4.7, "LocalToHost Byte Swap Control Register - L2H_BSCR" on page 452
Section 5.4.8, "HostToLocal Destination Lower Address Register - H2L_DLAR" on page 452
Section 5.4.9, "HostToLocal Source Upper Address Register - H2L_SUAR" on page 453
Section 5.4.10, "HostToLocal Source Lower Address Register - H2L_SLAR" on page 453
Section 5.4.11, "HostToLocal Byte Count Register - H2L_BCR" on page 454
Section 5.4.12, "HostToLocal Interrupt Counter/Acknowledge Register - H2L_ICAR" on page 455
Section 5.4.13, "HostToLocal Control/Status Register - H2L_CSR" on page 456
Section 5.4.14, "HostToLocal Byte Swap Control Register - H2L_BSCR" on page 457

5.4.1 LocalToHost Destination Lower Address Register - L2H_DLAR

The LocalToHost Destination Lower Address Registers (L2H_DLAR) represent the lower 32-bits of the destination (host) address.

Table 299. LocalToHost Destination Lower Address Register - L2H_DLAR



Intel XScale® Microarchitecture internal bus address offset
18204H

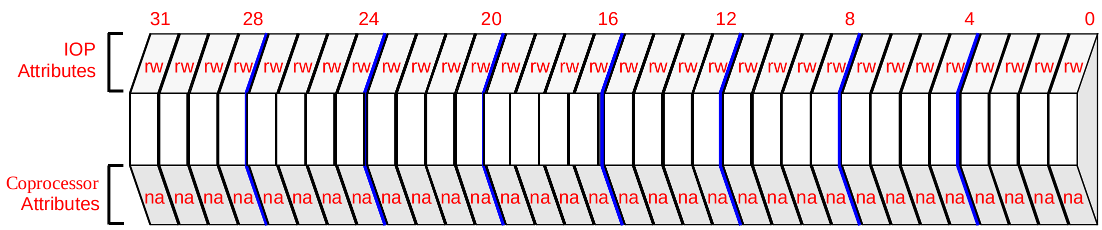
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:00	00000000H	Destination Lower Address Register (DLAR) - Read/Write This field specifies the low-order 32 bits of the destination (host) memory address. This field is CLEARED by a hardware or software reset.

5.4.2 LocalToHost Destination Upper Address Register - L2H_DUAR

The LocalToHost Destination Upper Address Register (L2H_DUAR) represents the upper 32-bits of the destination (host) address.

Table 300. LocalToHost Destination Upper Address Register - L2H_DUAR



Intel XScale® Microarchitecture internal bus address offset
18208H

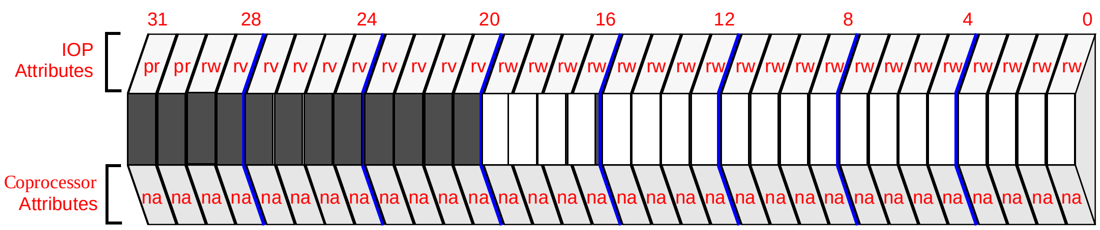
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:00	00000000H	Destination Upper Address Register (DUAR) - Read/Write This field specifies the upper-order 32 bits of the destination (host) memory address. This field is CLEARED by a hardware or software reset.

5.4.3 LocalToHost Source Lower Address Register - L2H_SLAR

The LocalToHost Source Lower Address Register (L2H_SLAR) represents the lower 32 bits of the source (local) address. The upper address bits are zero, as local memory is limited to 1 Mbyte.

Table 301. LocalToHost Source Lower Address Register - L2H_SLAR



Intel XScale® Microarchitecture internal bus address offset
1820CH

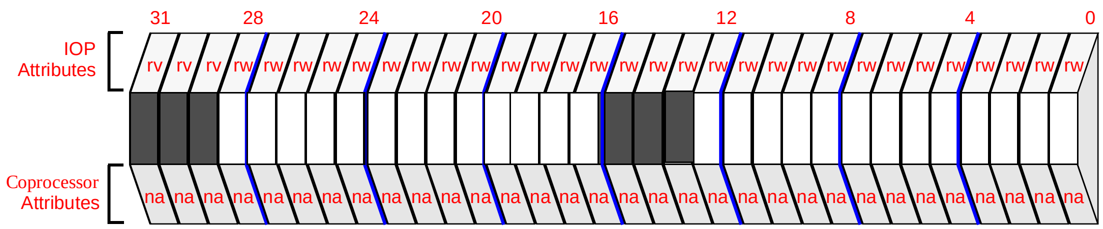
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:20	000H	Reserved. Must be written as zero.
19:00	00000H	Source Lower Address Register (SLAR) - Read/Write Specifies first source (local) memory starting byte address that SDMA Processor uses to read data. The field decodes a 1MB local memory address space that represents of the offset from the base of SRAM.

5.4.4 LocalToHost Byte Count Register - L2H_BCR

The LocalToHost Byte Count Register (L2H_BCR) represents the byte count associated with data to be moved. Note for internal architecture reasons the byte count must be entered in two locations within this register.

Table 302. LocalToHost Byte Count Register - L2H_BCR



Intel XScale® Microarchitecture internal bus address offset
18218H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

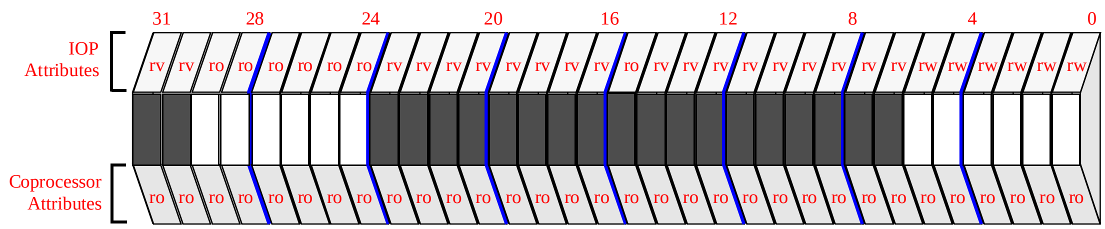
Bit	Default	Description
31:29	000_2	Reserved. Must be written as zero.
28:16	00000H	Byte Count Register (BCR) This field specifies the length in bytes that is for the data to be transferred. The maximum transfer count value is 4096 bytes (1000H). This field is CLEARED by a hardware or software reset.
15:13	000_2	Reserved. Must be written as zero.
12:00	000H	Byte Count Register (BCR) This field specifies the length in bytes that is for the data to be transferred. The maximum transfer count value is 4096 bytes (1000H). This field is CLEARED by a hardware or software reset. This must be written the same as bits 28:16, otherwise unpredictable results occur.

Note: It is required that the byte count be written to both the upper and lower parts of this register (bits 28:16 and bits 12:00). Failure to put the byte count in both locations renders unpredictable results.

5.4.5 LocalToHost Interrupt Counter/Acknowledge Register L2H_ICAR

Firmware uses the LocalToHost Interrupt Counter/Acknowledge Register (L2H_ICAR) to keep track of and acknowledge interrupts.

Table 303. LocalToHost Interrupt Counter/Acknowledge Register - L2H_ICAR



Internal bus address offset
18238H

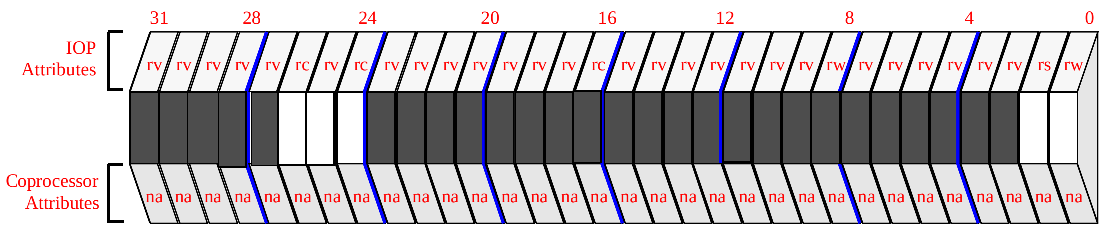
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:30	0H	Reserved
29:24	000000_2	Interrupt Counter - Read Only The Interrupt Counter is a counter of the number of interrupts on this DMA channel. It is monotonically increasing (by 1) and wraps around to zero after reaching its maximum value. When a DMA completes for this channel, the Interrupt Counter is incremented and an interrupt is asserted. Firmware reads the Interrupt Counter of each channel to determine which one raised the interrupt. Firmware then writes back the Interrupt Counter value to the Interrupt Acknowledge to clear the interrupt.
23:06	0000H	Reserved
05:00	00H	Interrupt Acknowledge - Read/Write The Interrupt Acknowledge is used to clear interrupts. Firmware reads the Interrupt Counter and then writes that value back to the Interrupt Acknowledge field.

5.4.6 LocalToHost Control/Status Register - L2H_CSR

The LocalToHost Control/Status Register (L2H_CSR) provides the control and status for the LocalToHost channel.

Table 304. LocalToHost Control/Status Register - L2H_CSR



Internal bus address offset
1823CH

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

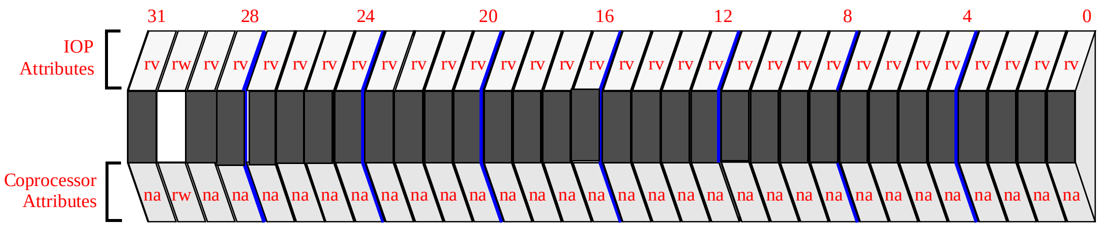
Bit	Default	Description
31:27	00000_2	**Reserved**
26:24	000_2	Error Flags - Read/Clear One or more of these bits is set to indicate an error has occurred with the DMA operation. The precise cause of the error is determined by reading the ATU or XSISC error registers.
23:02	0000H	Reserved
01	0_2	Channel Go (CHGO) - Read/Set Set this bit to begin the DMA operation. This bit must be cleared while the DMA registers are being set up. The CHGO bit is set by firmware to begin the DMA operation. When the DMA operation begins the hardware clears this bit. Note that clearing of this bit does not indicate the DMA operation is complete, rather that the hardware has started processing it.
00	0_2	Enable Register - Read/Write Must be set for proper operation. A value of zero (when the CHGO bit is set) has unpredictable results.

5.4.7 LocalToHost Byte Swap Control Register - L2H_BSCR

The LocalToHost Byte SWap Control Register (L2H_BSCR) provides the control to enable/disable byte swapping.

Note: The "Default" *enables* byte swapping.

Table 305. LocalToHost Byte Swap Control Register - L2H_BSCR



Internal bus address offset
18200H

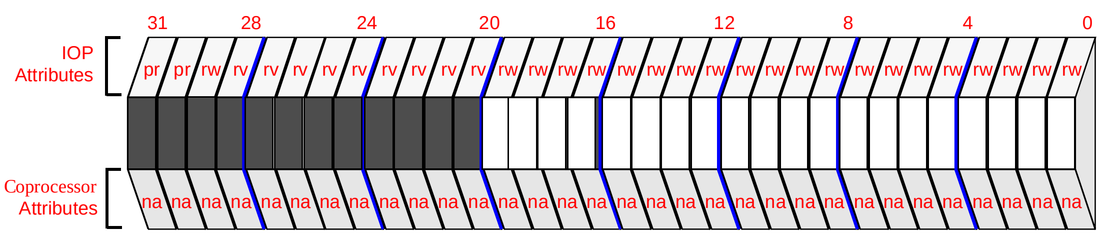
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31	00	**Reserved.**
30	00	Byte Swap Disable - Read/Write This bit must be set to prevent SDMA transfers from byte swapping.
29:0	00	**Reserved.**

5.4.8 HostToLocal Destination Lower Address Register - H2L_DLAR

The HostToLocal Destination Lower Address Register - H2L_DLAR represents the local memory address.

Table 306. HostToLocal Destination Lower Address Register - H2L_DLAR



Channel #

Intel XScale® Microarchitecture internal bus address offset
1825CH

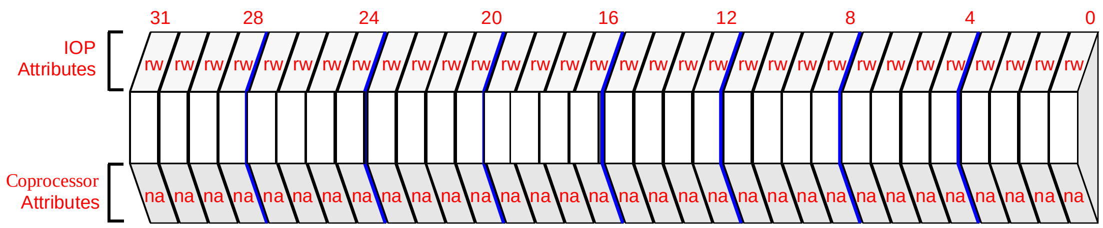
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:20	000H	Reserved
19:00	00000H	Destination Lower Address Register (DLAR) - Read/Write This field specifies the destination (local) memory starting byte address that the SDMA Processor uses to write data. The field decodes a 1MB local memory address space that represents of the offset from the base of SRAM.

5.4.9 HostToLocal Source Upper Address Register - H2L_SUAR

The HostToLocal Source Upper Address Register (H2L_SUAR) represents the upper 32-bits of the source (host) address.

Table 307. HostToLocal Source Upper Address Register - H2L_SUAR



Intel XScale® Microarchitecture internal bus address offset
18258H

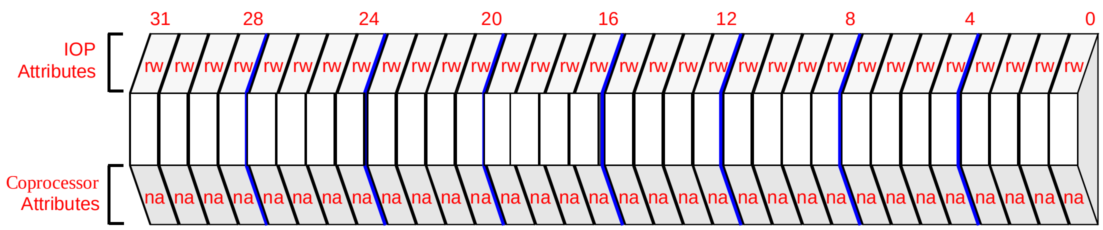
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:00	00000000H	Source Upper Address Register (SUAR) - Read/Write This field specifies the upper-order 32 bits of the source (host) memory address. This field is CLEARED by a hardware or software reset.

5.4.10 HostToLocal Source Lower Address Register - H2L_SLAR

The HostToLocal Source Lower Address Register (H2L_SLAR) represent the lower 32-bits of the source (host) address.

Table 308. HostToLocal Source Lower Address Register - H2L_SLAR



Intel XScale® Microarchitecture internal bus address offset
18254H

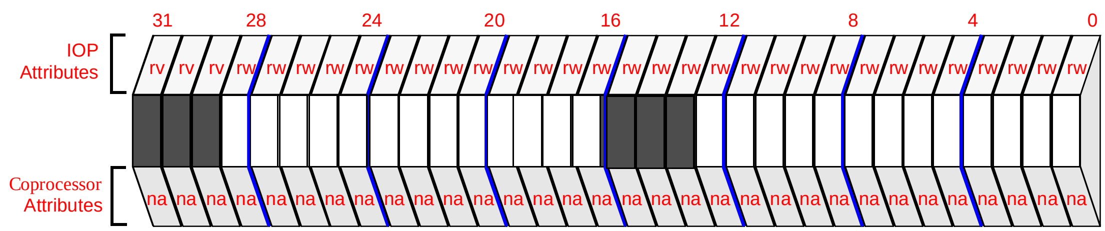
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:00	00000000H	Source Lower Address Register (SLAR) - Read/Write This field specifies the low-order 32 bits of the source (host) memory address. This field is CLEARED by a hardware or software reset.

5.4.11 HostToLocal Byte Count Register - H2L_BCR

The HostToLocal Byte Count Registers (H2L_BCR) represent the byte count of the DMA.

Table 309. HostToLocal Byte Count Register - H2L_BCR



Intel XScale® Microarchitecture internal bus address offset

18268H

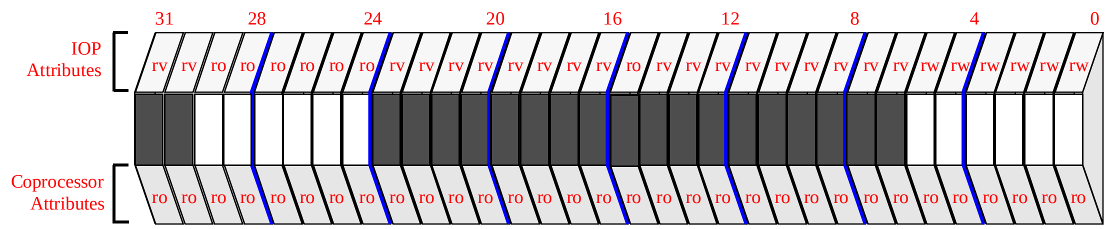
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:29	000_2	Reserved
28:16	00000H	Byte Count Register (BCR) This field specifies the length in bytes that is for the data to be transferred. The maximum transfer count value is 4096 bytes (1000H). This field is CLEARED by a hardware or software reset.
15:13	000_2	Reserved. Must be written as zero.
12:00	000H	Byte Count Register (BCR) This field specifies the length in bytes that is for the data to be transferred. The maximum transfer count value is 4096 bytes (1000H). This field is CLEARED by a hardware or software reset. This must be written the same as bits 28:16, otherwise unpredictable results occur.

5.4.12 HostToLocal Interrupt Counter/Acknowledge Register - H2L_ICAR

Firmware uses the HostToLocal Interrupt Acknowledge Register (H2L_ICAR) to keep track of and acknowledge interrupts.

Table 310. HostToLocal Interrupt Counter/Acknowledge Register - H2L_ICAR



Internal bus address offset
18288H

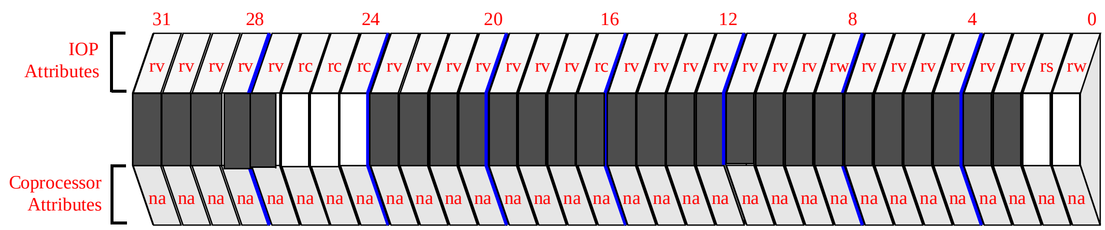
Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
31:30	0H	Reserved
29:24	000000_2	Interrupt Counter - Read Only The Interrupt Counter is a counter of the number of interrupts on this DMA channel. It is monotonically increasing (by 1) and wraps around to zero after reaching its maximum value. When a DMA completes for this channel, the Interrupt Counter is incremented and an interrupt is asserted. Firmware reads the Interrupt Counter of each channel to determine which one raised the interrupt. Firmware then writes back the Interrupt Counter value to the Interrupt Acknowledge to clear the interrupt.
23:06	0000H	Reserved
05:00	00H	Interrupt Acknowledge - Read/Write The Interrupt Acknowledge is used to clear interrupts. Firmware reads the Interrupt Counter and then writes that value back to the Interrupt Acknowledge field.

5.4.13 HostToLocal Control/Status Register - H2L_CSR

The HostToLocal Control/Status Register (H2L_CSR) provides the status and control of the HostToLocal channel.

Table 311. HostToLocal Control/Status Register - H2L_CSR



Internal bus address offset
1828CH

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

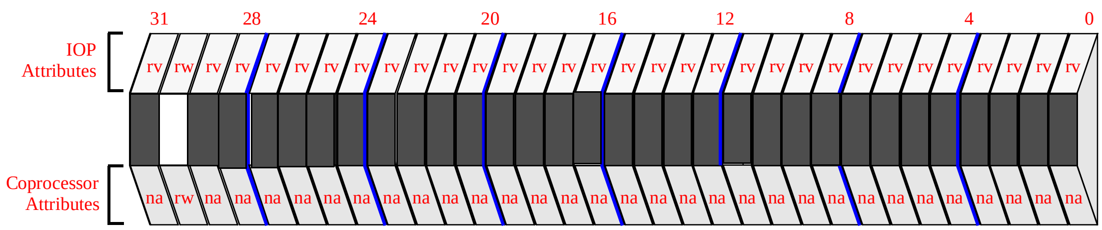
Bit	Default	Description
31:27	00000_2	Reserved
26:24	000_2	Error Flags - Read/Clear One or more of these bits is set to indicate an error has occurred with the DMA operation. The precise cause of the error is determined by reading the ATU or XSISC error registers.
23:02	0000H	Reserved
01	0_2	Channel Go (CHGO) - Read/Set Set this bit to begin the DMA operation. This bit must be cleared while the DMA registers are being set up. The CHGO bit is set by firmware to begin the DMA operation. When the DMA operation begins the hardware clears this bit. Note that clearing of this bit does not indicate the DMA operation is complete, rather that the hardware has started processing it.
00	0_2	Enable Register - Read/Write Must be set for proper operation. A value of zero (when the CHGO bit is set) has unpredictable results.

5.4.14 HostToLocal Byte Swap Control Register - H2L_BSCR

The LocalToHost Byte SWap Control Register (H2L_BSCR) provides the control to enable/disable byte swapping.

Note: The "Default" *enables* byte swapping.

Table 312. HostToLocal Byte Swap Control Register - H2L_BSCR



Internal bus address offset
18250H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31	00	**Reserved.**
30	00	Byte Swap Disable - Read/Write This bit must be set to prevent SDMA transfers from byte swapping.
29:0	00	**Reserved.**

6.0 SGPIO Unit

Note: For TPER mode the register interface defined here can be used. For Intel® 413808 and 413812 I/O Controllers (4138xx) non-TPER mode, see the SAS/SATA Command Summary for API to control the SGPIO units. Some limitations may apply when controlling via the API.

6.1 Overview

This section describes Serial General Purpose Input Output (SGPIO) interface. The 4138xx (based on Intel XScale® technology[14]) supports two SGPIO interfaces. The SGPIO is a serial bus consisting of four signals:

- SClock
- SLoad
- SDataOut
- SDataIn

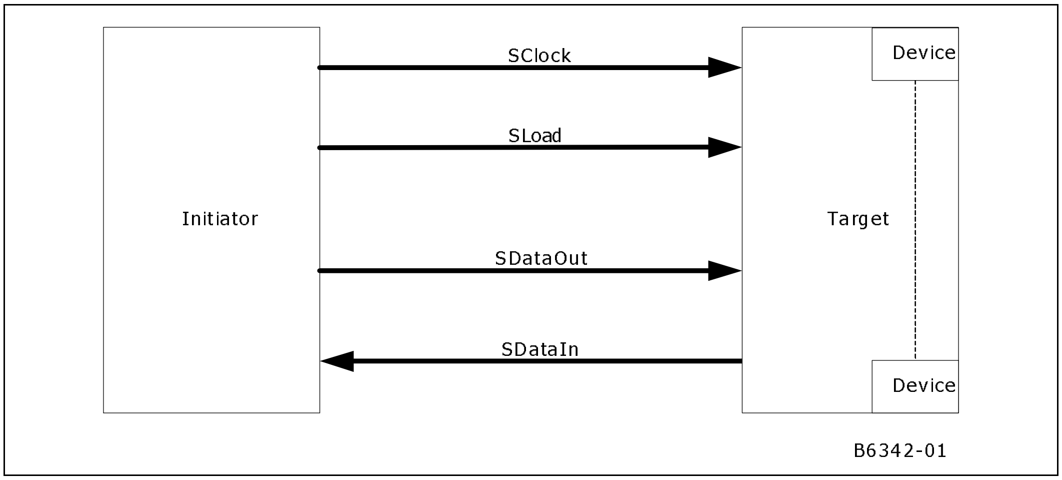
The SGPIO is used to serialize general purpose I/O signals. The SGPIO defines communication between an initiator and a target. The target typically converts output signals into multiple parallel LED signals and provides inputs frsom general purpose inputs. Figure 43 shows the SGPIO bus. A target typically consists of multiple devices, and SGPIO protocol allows each device on the target to support up to three output and three input signals.

Each SGPIO interface on 4138xx can support up to eight devices (drives) on the target end. Each device can control up to three output bits and three input bits. Therefore, each SGPIO interface on 4138xx can support up to twenty-four input signals and twenty-four output signals. Some usage models require both SGPIO units on 4138xx to be used in conjunction with each other.

For example, when using direct LED support for eight drives, both SGPIO units have to used together, as one SGPIO can only support up to four drives for direct LED.

14.ARM architecture compliant.

Figure 43. SGPIO Bus Overview

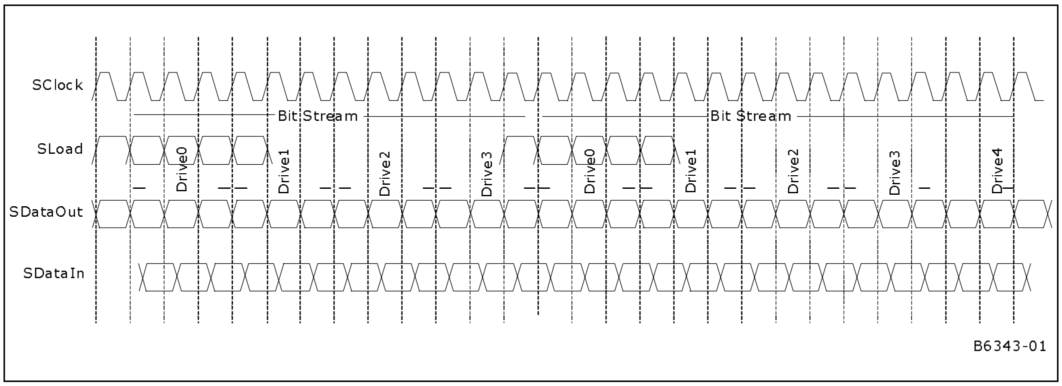


6.2 Theory of Operation

The SGPIO is used to serialize general purpose I/O signals. For example, the initiator may want to drive multiple LEDs on the target, and thus do so by sampling and serializing the parallel initiator LED signals at a fixed sampling rate dictated by the low-to-high transition of the SLoad signal. Note that SClock is a free-running clock. The receiver (initiator or target) would then take the bit samples from the bit stream and converts them into parallel LED signals.

Figure 44 shows the input and output bit streams relative to SClock and SLoad signals. Note that the SGPIO interface sends a repeating bit stream on SDataOut and receives a repeating bit stream SDataIn. The bit stream is restarted each time the SLoad signal is set high. Note that the example in Figure 44 shows four drives and five drives. The bit stream need not be the same length every time.

Figure 44. SGPIO Repeating Bit Stream



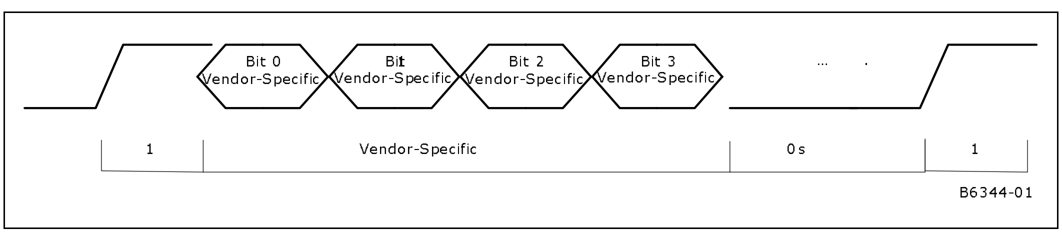
6.2.1 SGPIO SClock Output Signal

SClock is a free-running clock, running at a fixed frequency of up to 100 KHz. The rising edge of SClock is used to transmit SLoad, SDataOut, and SDataIn. The falling edge of SClock is used to latch SLoad, SDataOut, and SDataIn.

6.2.2 SGPIO SLoad Output Signal

The initiator shall repeatedly send SDataOut bits and receives SDataIn bits. The SLoad signal indicates when the bit stream is ending or being restarted. After SLoad is asserted (set to 1), the next four bits positions on SLoad contain a vendor-specific pattern. Following the vendor-specific pattern, the initiator shall set the SLoad to 0 until it wants to restart the bits stream.

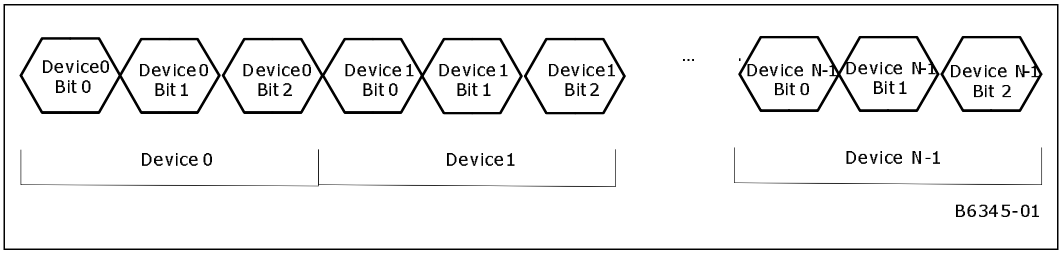
Figure 45. SLoad Signal



6.2.3 SDataOut

The SDataOut signal carries output bits associated with devices on the target. For example, on 4138xx the SGPIO can drive up to three bits per device and up to eight devices on the target, thus is able to control twenty-four outputs on the target. The SDataOut signal carries the 3-bit outputs for each device in the same order in each repeated bit stream.

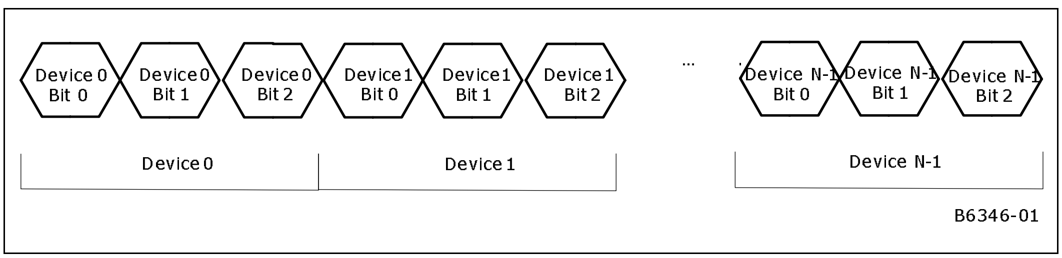
Figure 46. SDataOut Signal



6.2.4 SGPIO SDataIn Signal

The SDataIn signal carries input bits associated with devices on the target. For example, on 4138xx the SGPIO can receive up to three bits per device and up to eight devices on the target, thus is able to receive twenty-four inputs from the target. The SDataIn signal carries the 3-bit inputs for each device in the same order in each repeated bit stream.

Figure 47. SDataIn Signal



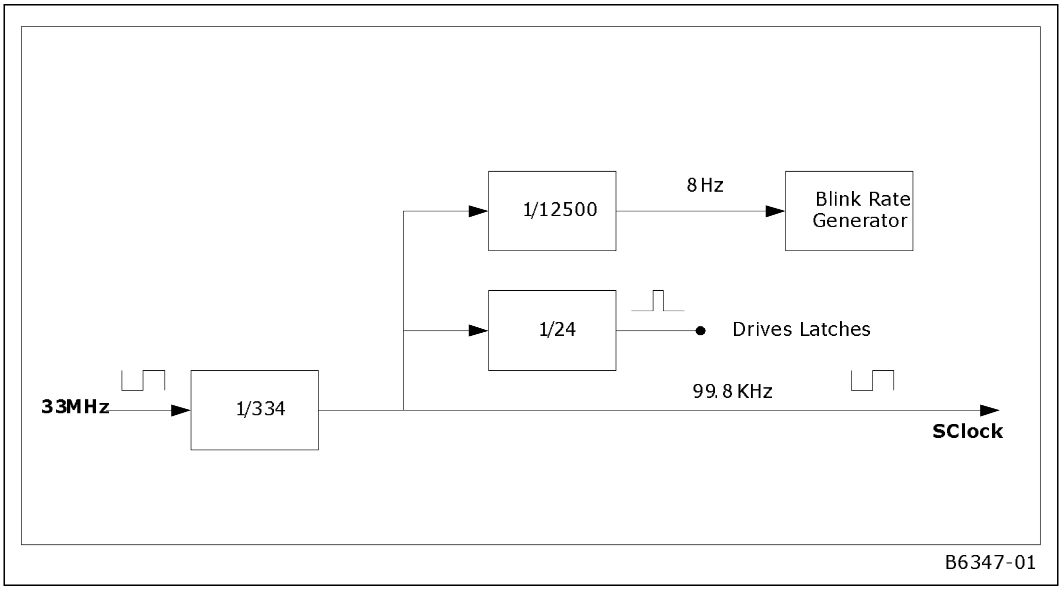
6.3 Clock Requirements

4138xx generates and drives three clock signals that are used to run the various blocks of the SGPIO units.

- SClock - is the output clock of the SGPIO interface and runs at a fixed 99.8 KHz.
- Load Clock - this clock is used internally to load the internal latches. This clock runs at 1/24 the SClock rate.
- Blink Generator Clock - this clock is used to drive the blink generator. This clock runs at 1/12500 of the SClock rate.

Figure 48 shows the clock structure.

Figure 48. Clock Structure



6.4 Output Signals

Each of the 4138xx SGPIO units can support up to eight drives, and each drive can support up to three output signals. This allows the two SGPIO units on 4138xx to be able to drive up to twenty-four output signals.

4138xx supports the following output signals:

- Fixed High
- Protocol Engine Activity, Protocol Engine Status, or Reserved
- Two programmable Blinks (A and B)

In addition the outputs can be optionally inverted.

Each output bit can be independently selected using the Table 328, "SGPIO Output Data Select Register[0:7] x - SGODSR[0:7]x" on page 484. The selected output can in turn be inverted by firmware using the Table 328, "SGPIO Output Data Select Register[0:7] x - SGODSR[0:7]x" on page 484.

Figure 49, Figure 50, and Figure 51 respectively show the three output signals supported per drive (OD0, OD1, and OD2) and the supported output signal selections.

Figure 49. SGPIO Output OD0 Signal

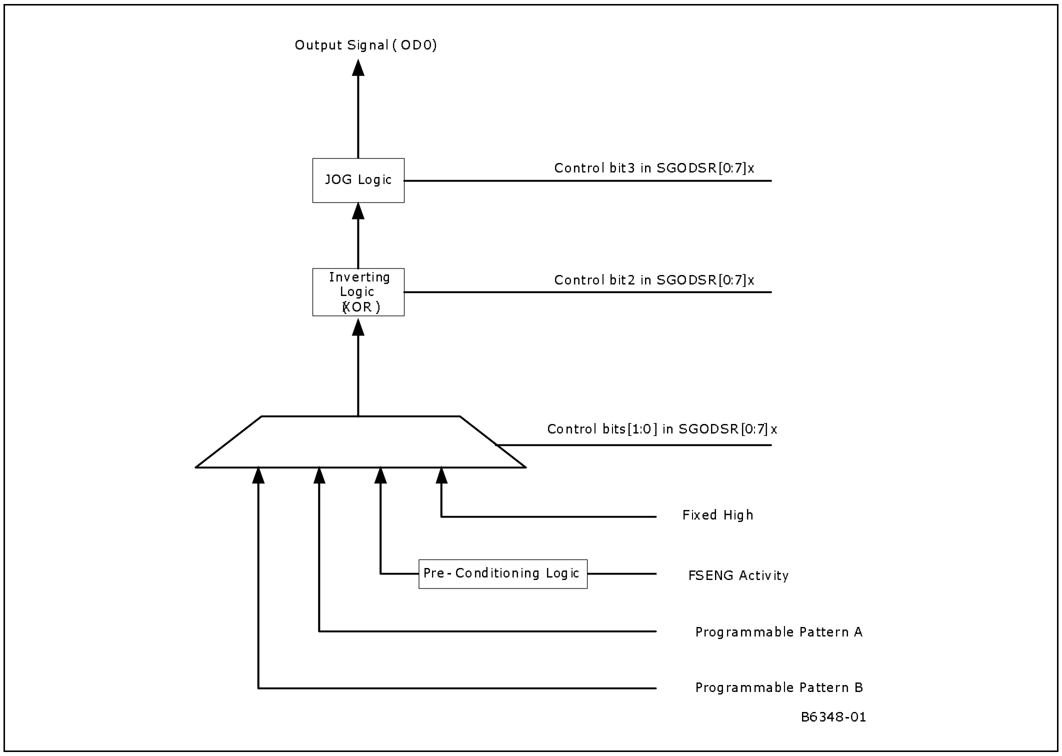


Figure 50. SGPIO Output OD1 Signal

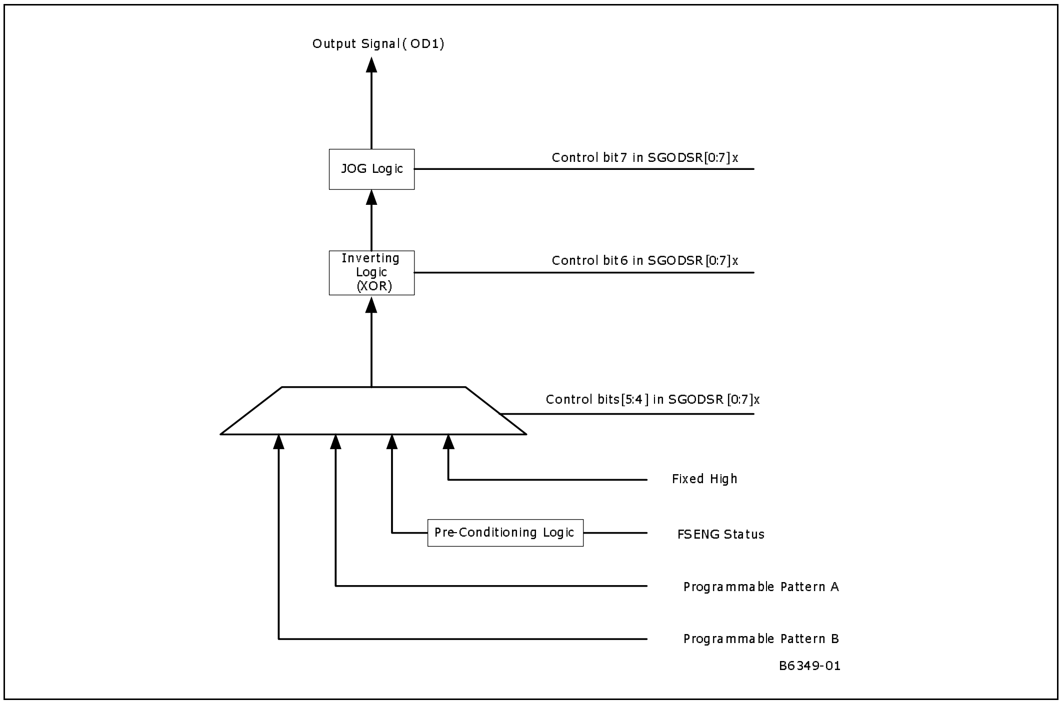
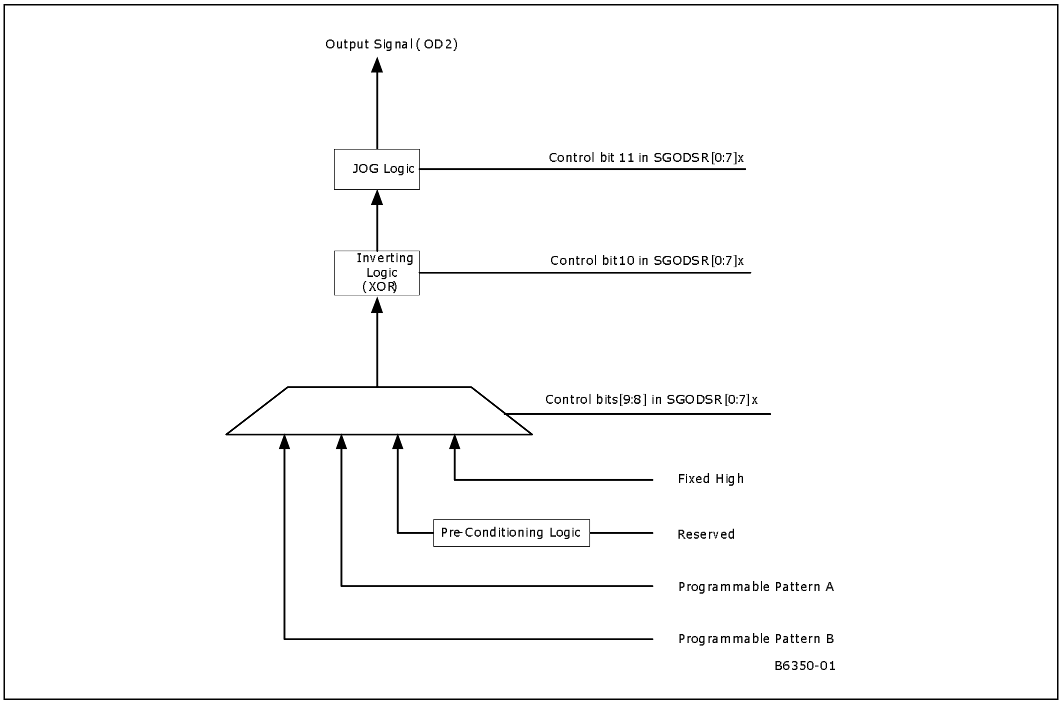


Figure 51. SGPIO Output OD2 Signal



6.4.1 Protocol Engine Input Signals

There are eight Protocol Engine activity signals (S_ACT[7:0]) and eight Protocol Engine status signals (S_STAT[7:0]) which are all input signals both SGPIO units. These Protocol Engine activity and status signals can be selected as optional output signals of the SGPIO Units that can be driven on the SDataOut pins or on the direct LED signals. Refer to Figure 49 and Figure 50 for the output selections. Table 313 shows how the input signals are mapped to the ODx inputs of the SGPIO unit.

Note: In Table 313 the Protocol Engine activity and status signal pairs are not connected to the corresponding drive numbers. This is done on 4138xx because the way the Protocol Engine ports are mapped.

Table 313. SGPIO Input Mapping (Sheet 1 of 2)

Input Signals	SGPIOx Inputs	Input Signals	SGPIOx Inputs
Fixed High	Drive0.OD0	Fixed High	Drive4.OD0
Activity [5]		Protocol Engine Activity [0]	
Programmable Pattern A		Programmable Pattern A	
Programmable Pattern B		Programmable Pattern B	
Fixed High	Drive0.OD1	Fixed High	Drive4.OD1
Protocol Engine Status [5]		Protocol Engine Status [0]	
Programmable Pattern A		Programmable Pattern A	
Programmable Pattern B		Programmable Pattern B	
Fixed High	Drive0.OD2	Fixed High	Drive4.OD2
Reserved		Reserved	
Programmable Pattern A		Programmable Pattern A	
Programmable Pattern B		Programmable Pattern B	
Fixed High	Drive1.OD0	Fixed High	Drive5.OD0
Protocol Engine Activity [7]		Protocol Engine Activity [2]	
Programmable Pattern A		Programmable Pattern A	
Programmable Pattern B		Programmable Pattern B	
Fixed High	Drive1.OD1	Fixed High	Drive5.OD1
Protocol Engine Status [7]		Protocol Engine Status [2]	
Programmable Pattern A		Programmable Pattern A	
Programmable Pattern B		Programmable Pattern B	
Fixed High	Drive1.OD2	Fixed High	Drive5.OD2
Reserved		Reserved	
Programmable Pattern A		Programmable Pattern A	
Programmable Pattern B		Programmable Pattern B	
Fixed High	Drive2.OD0	Fixed High	Drive6.OD0
Protocol Engine Activity [1]		Protocol Engine Activity [4]	
Programmable Pattern A		Programmable Pattern A	
Programmable Pattern B		Programmable Pattern B	

Table 313. SGPIO Input Mapping (Sheet 2 of 2)

Input Signals	SGPIOx Inputs	Input Signals	SGPIOx Inputs
Fixed High	Drive2.OD1	Fixed High	Drive0.OD1
Protocol Engine Status [1]		Protocol Engine Status [4]	
Programmable Pattern A		Programmable Pattern A	
Programmable Pattern B		Programmable Pattern B	
Fixed High	Drive2.OD2	Fixed High	Drive6.OD2
Reserved		Reserved	
Programmable Pattern A		Programmable Pattern A	
Programmable Pattern B		Programmable Pattern B	
Fixed High	Drive3.OD0	Fixed High	Drive7.OD0
Protocol Engine Activity [3]		Protocol Engine Activity [6]	
Programmable Pattern A		Programmable Pattern A	
Programmable Pattern B		Programmable Pattern B	
Fixed High	Drive3.OD1	Fixed High	Drive7.OD1
Protocol Engine Status [3]		Protocol Engine Status [6]	
Programmable Pattern A		Programmable Pattern A	
Programmable Pattern B		Programmable Pattern B	
Fixed High	Drive3.OD2	Fixed High	Drive7.OD2
Reserved		Reserved	
Programmable Pattern A		Programmable Pattern A	
Programmable Pattern B		Programmable Pattern B	

6.4.1.1 JOG Requirements

The jog feature is optional and is controlled by the Table 328, "SGPIO Output Data Select Register[0:7] x - SGODSR[0:7]x" on page 484. When enabled, this feature monitors the input signal and if the input signal is detected low for about four seconds it will be forced high for a 250 ms duration.

6.4.1.2 Protocol Engine Pre-Conditioning Requirements

All the Protocol Engine activity and status signals are pre-conditioned when entering the SGPIO units. The pre-conditioning logic monitors for any short pulse or any high frequency input signal and ensures that the input signal is stretched and held high for at least 125 ms.

6.4.2 Programmable Blink Patterns

Each of the SGPIO output signal supports two programmable blink patterns that can be selected using the Table 328, "SGPIO Output Data Select Register[0:7] x - SGODSR[0:7]x" on page 484. The blink rate generator is clocked using an 8 Hz clock and allows the user to program a low and a high duration time using two 4-bit fields located in the Table 322, "SGPIO Programmable Blink Register x - SGPBRx" on page 476. The shortest low/high duration time that can be program is 125 ms and the longest low/high duration time that can be programmed is 2 seconds. The shortest blink rate period is 250 ms and the longest blink rate period is 4 seconds.

6.5 SGPIO Unit Mode of Operations

Each SGPIO unit on 4138xx can be programmed to support the following modes:

- Direct LED
- SGPIO

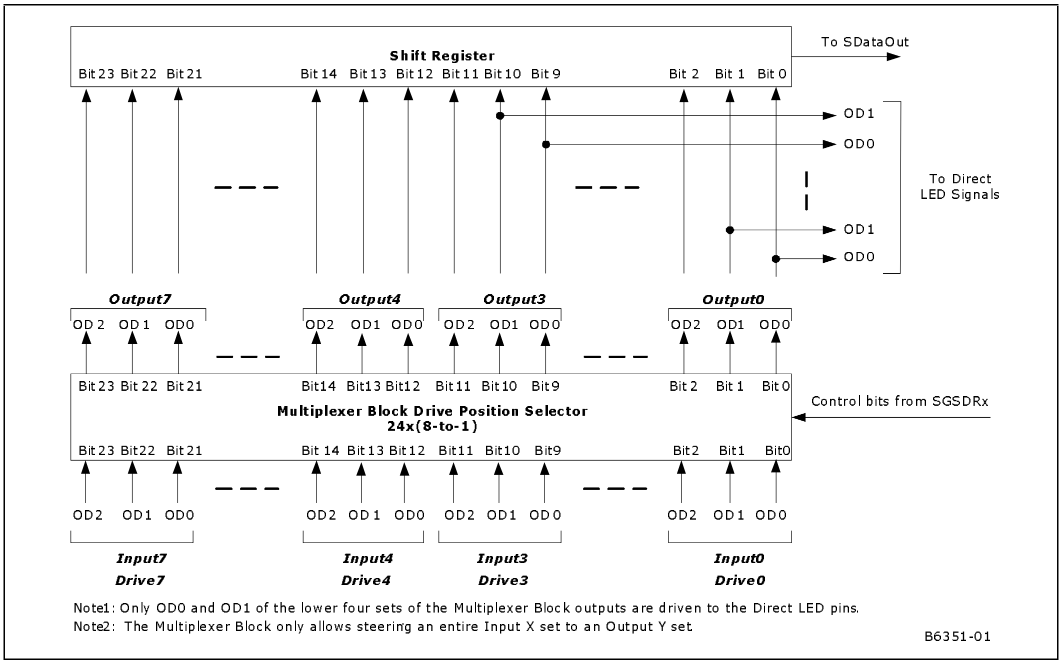
4138xx provides eight configurable pins per SGPIO unit to accommodate SGPIO mode.

When the SGPIO unit is set up to operate in Direct LED mode by clearing bit[0] of Table 321, "SGPIO Interface Control Register x - SGICRx" on page 475, all eight pins provide direct LED support. For example, all the output signals are directly driven to the corresponding pins.

When the SGPIO unit is set up to operate in SGPIO mode by setting bit[0] of the Table 321, "SGPIO Interface Control Register x - SGICRx" on page 475, four of eight pins provide the SGPIO bus.

Figure 52 shows how selected output signals are routed into a multiplexer block and how outputs of the multiplexer block are routed to the shift register and direct LED signals. The multiplexer block provides the ability to select how drive output signals are ordered before being driven to the shift register and to direct LED signals. Note, all three output signals of each drive are selected simultaneously. In direct LED mode, programming the multiplexer allows any four drive output signals to be selected and driven on direct LED pins. The programmable feature of SGPIO allows for either SGPIO unit to support any number of drive combinations. For example, the eight drives supported on 4138xx can be distributed between the two SGPIO units.

Figure 52. Output Signal Routing



Note: Only output signals OD0 and OD1 of the lower four outputs of the multiplexer block (Output[3:0]) are routed to direct LED signals. For example, each SGPIO unit can only support up to eight direct LED output signals: four Protocol Engine activity and four Protocol Engine Status signals.

Example 4. SGPIO Unit 0 is in Direct LED mode supporting Drives [4,6] and SGPIO Unit 1 is in SGPIO Mode supporting Drives[1,3,5,7]

In this example, SGPIO unit 0 is set up in Direct LED mode supporting Drive 4 and Drive 6. Bit[0] in Table 321, "SGPIO Interface Control Register x - SGICRx" on page 475 must be cleared for SGPIO unit 0, as this sets up the external pins for Direct LED mode. The drive shift order for SGPIO unit 0 is programmed as follows: Drive 4 (first drive to be shifted), Drive 6, Drive 0, Drive 1, Drive 2, Drive 3, Drive 5, and Drive 7 (last drive to be shifted). Note that the last six drives (0, 1, 3, 4, 5, and 7) are not used for SGPIO unit 0, but are simply programmed in that order in Table 324, "SGPIO Start Drive Upper Register x — SGSDURx" on page 480. Table 314 shows the outputs[7:0] of the multiplexer block for SGPIO unit 0.

SGPIO unit 1 supports four drives. Bit[0] in Table 321, "SGPIO Interface Control Register x - SGICRx" on page 475 must be set for SGPIO unit 1, as this sets up the external pins for SGPIO mode. The drive shift order for SGPIO unit 1 is programmed as follows: Drive 3 (first drive to be shifted), Drive 1, Drive 7, Drive 5, Drive 0, Drive 2, Drive 4, and Drive 6 (last drive to be shifted). Note that the last four drives (0, 2, 4, and 6) are not used for SGPIO unit 1, but are simply programmed in that order in Table 324, "SGPIO Start Drive Upper Register x — SGSDURx" on page 480. Table 315 shows the outputs[7:0] of the multiplexer block for SGPIO unit 1.

Table 314. Example 1: Multiplexer Block Outputs for SGPIO Unit 0 in Direct LED Mode

Multiplexer Block Output	Output 7	Output 6	Output 5	Output 4	Output 3	Output 2	Output 1	Output 0
Drive to Shift Register[a,b]	Drive 7	Drive 5	Drive 3	Drive 2	Drive 1	Drive 0	Drive 6	Drive 4
Drive to Direct LED Signals[a]	N/A[c]	N/A	N/A	N/A	Drive 1	Drive 0	Drive 6	Drive 4

a. The grayed cells imply that these outputs are not valid for this example, but are programmed in that manner and yield the outputs shown.
b. This entire row is shown simply to demonstrate how the drives' order would be mapped if the SGPIO Unit 0 were to be used in SGPIO mode.
c. The cells labeled "N/A" are not valid outputs for Direct LED Mode. For example, they are not connected.

Table 315. Example 2: Multiplexer Block Outputs for SGPIO Unit 1 in SGPIO Mode

Multiplexer Block Output	Output 7	Output 6	Output 5	Output 4	Output 3	Output 2	Output 1	Output 0
Drive to Shift Register[a]	Drive 6	Drive 4	Drive 2	Drive 0	Drive 5	Drive 7	Drive 1	Drive 3
Drive to Direct LED Signals[a,b]	N/A[c]	N/A	N/A	N/A	Drive 5	Drive 7	Drive 1	Drive 3

a. The grayed cells imply that these outputs are not valid for this example, but are programmed in that manner and yield the outputs shown.
b. This entire row is shown simply to demonstrate how the drives' order would be mapped if the SGPIO Unit 1 were used in Direct LED mode.
c. The cells labeled "N/A" are not valid outputs for Direct LED Mode. For example, they are not connected.

Example 5. Both SGPIO Units are used in SGPIO mode with SGPIO Unit 0 supporting Drives[0,1,3,4,5,6] and SGPIO Unit 1 supporting Drives[2,7]

In this example, both SGPIO units are used and they are set up in SGPIO mode by setting bit[0] in Table 321, "SGPIO Interface Control Register x - SGICRx" on page 475. SGPIO unit 0 supports six drives. The drive shift order for SGPIO unit 0 is programmed as follows: Drive 6 (first drive to be shifted), Drive 4, Drive 3, Drive 5, Drive 0, Drive 1, Drive 2, and Drive 7 (last drive to be shifted). Note that the last two drives (2 and 7) are not used for SGPIO unit 0, but are simply programmed in that order in Table 324, "SGPIO Start Drive Upper Register x — SGSDURx" on page 480. Table 316 shows the outputs[7:0] of the multiplexer block for SGPIO unit 0.

SGPIO unit 1 supports two drives. The drive order for SGPIO unit 1 is programmed as follows: Drive 7 (first drive to be shifted), Drive 2, Drive 0, Drive 1, Drive 3, Drive 4, Drive 5, Drive 6 (last drive to be shifted). Note that the last six drives (0, 1, 3, 4, 5, and 6) are not used for SGPIO unit 1, but are simply programmed in that order in Table 324, "SGPIO Start Drive Upper Register x — SGSDURx" on page 480. Table 317 shows the outputs[7:0] of the multiplexer block for SGPIO unit 1.

Table 316. SGPIO Unit 0 Multiplexer Block Outputs for Example 2

Multiplexer Block Output	Output 7	Output 6	Output 5	Output 4	Output 3	Output 2	Output 1	Output 0
Drive to Shift Register[a]	Drive 7	Drive 2	Drive 1	Drive 0	Drive 5	Drive 3	Drive 4	Drive 6
Drive to Direct LED Signals[a,b]	N/A[c]	N/A	N/A	N/A	Drive 5	Drive 3	Drive 4	Drive 6

a. The grayed cells imply that these outputs are not valid for this example, but are programmed in that manner and yield the outputs shown.
b. This entire row is shown simply to demonstrate how the drives' order would be mapped if the SGPIO Unit 0 were used in Direct LED mode.
c. The cells labeled "N/A" are not valid outputs for Direct LED Mode. For example, they are not connected.

Table 317. SGPIO Unit 1 Multiplexer Block Outputs for Example 2

Multiplexer Block Output	Output 7	Output 6	Output 5	Output 4	Output 3	Output 2	Output 1	Output 0
Drive to Shift Register[a]	Drive 6	Drive 5	Drive 4	Drive 3	Drive 1	Drive 0	Drive 2	Drive 7
Drive to Direct LED Signals[a,b]	N/A[c]	N/A	N/A	N/A	Drive 1	Drive 0	Drive 2	Drive 7

a. The grayed cells imply that these outputs are not valid for this example, but are programmed in that manner and yield the outputs shown.
b. This entire row is shown simply to demonstrate how the drives' order would be mapped if the SGPIO Unit 0 were used in Direct LED mode.
c. The cells labeled "N/A" are not valid outputs for Direct LED Mode. For example, they are not connected.

6.5.1 Pin Multiplexing

All S_ACT[7:0] and S_STAT[7:0] pins are multiplexed. Note that an SGPIO interface is a 4-pin interface. SGPIO unit 0 uses pins S_ACT[3:0] and S_STAT]3:0] for SGPIO signaling and direct LED controls, whereas SGPIO unit 1 uses pins S_ACT[7:4] and S_STAT[7:4] for SGPIO signaling and direct LED controls. Table 318 and Figure 322 show how the SGPIO unit 0 signals are multiplexed. Table 319 on page 473 and Figure 54 on page 473 show how the SGPIO unit 1 signals are multiplexed. Bit 0 of Table 321, "SGPIO Interface Control Register x - SGICRx" on page 475 is used to select between the SGPIO signals and the direct LED signals, and by default bit 0 selects the SGPIO signals. The direct LED signals and the TXRATE signals are selected based on product types.

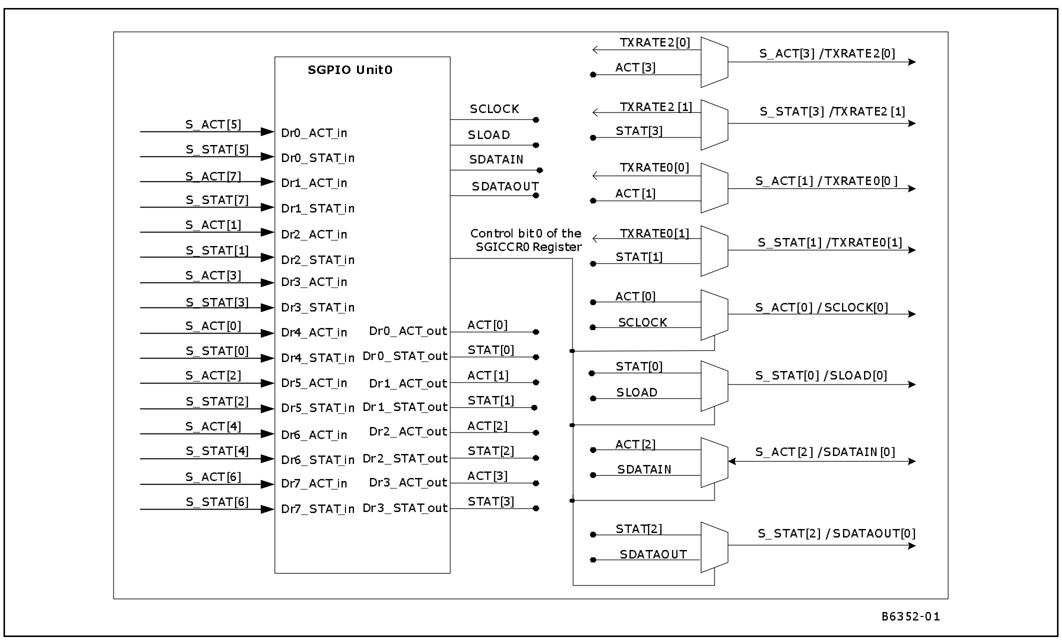
Note: TXRATEx[y] signals are valid signals only for product types that support fibre channel ports. The multiplexers are controlled per the product type. For example, a 4138xx that supports only SAS ports has the multiplexers selecting S_ACT[x] and S_STAT[x] signals.

Table 318. SGPIO Unit 0 Pin Multiplexing

Activity Pin	Shared Pin	Status Pin	Shared Pin
S_ACT[0]	SCLOCK[0]	S_STAT[0]	SLOAD[0]
S_ACT[1]	TXRATE0[0]	S_STAT[1]	TXRATE0[1]
S_ACT[2]	SDATAIN[0]	S_STAT[2]	SDATAOUT[0]
S_ACT[3]	TXRATE2[0]	S_STAT[3]	TXRATE2[1]

Note: Protocol Engine activity and status signal pairs are not connected to corresponding SGPIO unit drive numbers.

Figure 53. Intel® 413808 and 413812 I/O Controllers in TPER Mode SGPIO Unit 0 Pin Mapping



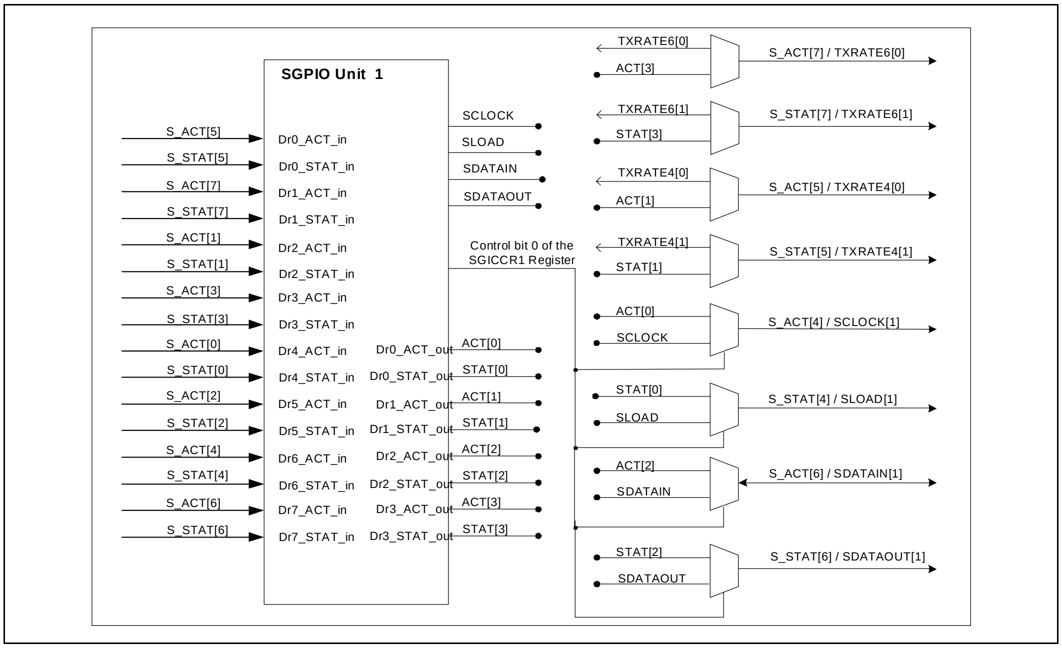
Note: TXRATEx[y] signals are valid signals only for product types that support fibre-channel ports. Multiplexers are controlled per product type. For example, a 4138xx that supports only SAS ports has the multiplexers selecting S_ACT[x] and S_STAT[x] signals.

Table 319. SGPIO Unit 1 Pin Multiplexing

Activity Pin	Shared Pin	Status Pin	Shared Pin
S_ACT[4]	SCLOCK[1]	S_STAT[4]	SLOAD[1]
S_ACT[5]	TXRATE4[0]	S_STAT[5]	TXRATE4[1]
S_ACT[6]	SDATAIN[1]	S_STAT[6]	SDATAOUT[1]
S_ACT[7]	TXRATE6[0]	S_STAT[7]	TXRATE6[1]

Note: The Protocol Engine activity and status signal pairs are not connected to the corresponding SGPIO unit drive numbers.

Figure 54. 4138xx SGPIO Unit 1 Pin Mapping



6.6 Register Definitions

The SGPIO contains memory-mapped registers for:

- selecting ODx output signals that are driven on the serial data bus and to direct LED pins,
- reading serial data from the input data bus,
- Programming Vendor Specific Code

Warning: The SGPIO Units must be programmed for proper operation. By default the SGPIO units are initialized to operate in SGPIO modes. The user must program the units to place them in the desired mode of operations. The user must also select the start drive for each SGPIO unit as the default start drive is drive number 0 for both SGPIO units.

Memory-Mapped Registers mentioned above are located as relative offsets of a 512 KB aligned global PMMR Block. Default for the 512 KB aligned PMMR Block is 0 FFD8 0000H defined by the PMMRBAR register. See also Chapter 19.0, "Peripheral Registers".

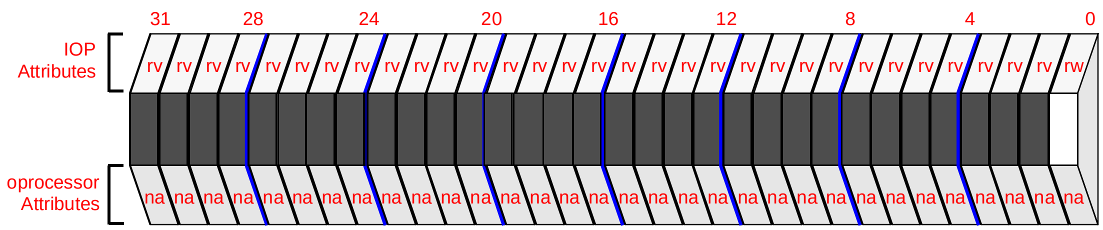
Table 320. SGPIO Memory-Mapped Rejecters

Section, Register Name, Acronym, Page	Address Offsets	
	SGPIO Unit 0	SGPIO Unit 1
Table 321, "SGPIO Interface Control Register x - SGICRx" on page 475	+2600H	+2680H
Table 322, "SGPIO Programmable Blink Register x - SGPBRx" on page 476	+2604H	+2684H
Table 323, "SGPIO Start Drive Lower Register x — SGSDLRx" on page 478	+2608H	+2688H
Table 324, "SGPIO Start Drive Upper Register x — SGSDURx" on page 480	+260CH	+268CH
Table 325, "SGPIO Serial Input Data Lower Register x - SGSIDLRx" on page 482	+2610H	+2690H
Table 326, "SGPIO Serial Input Data Upper Register x - SGSIDURx" on page 483	+2614H	+2694H
Table 327, "SGPIO Vendor Specific Code Register x - SGVSCRx" on page 483	+2618H	+2698H
Reserved	+261CH	+269CH
Table 328, "SGPIO Output Data Select Register[0:7] x - SGODSR[0:7]x" on page 484	+2620H through +263FH	+26A0H through +26BFH
Reserved	+2640H through +267FH	+26C0H through +26FFH

6.6.1 SGPIO Interface Control Register x — SGICRx

The SGPIO Interface Control Register x - SGPICRx is used to select the SGPIO unit mode of operations - SGPIO bus or direct LED interface. Each SGPIO unit can either drive eight output signals on the serial SGPIO bus or directly drive the eight output signals on eight separate pins.

Table 321. SGPIO Interface Control Register x - SGICRx



SGPIO #	Intel XScale® processor internal bus address offset
0	+2600H
1	+2680H

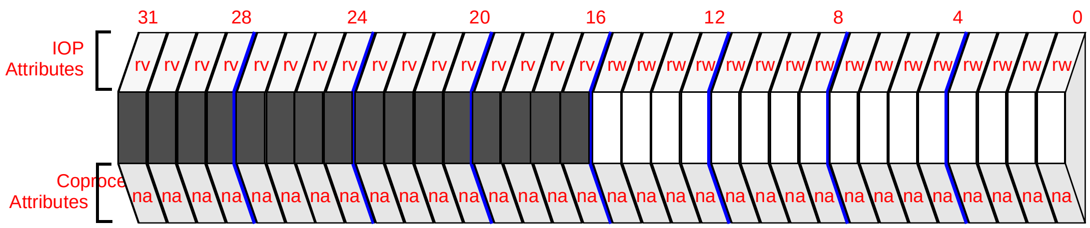
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:16	0000H	Reserved.
15:01	0000H	Reserved.
00	0_2	SGPIO x Functionality Enable: For SGPIO unit 0 - When this bit is set, the SGPIO bus pins (S_ACT[0], S_STAT[0], S_ACT[2] and S_STAT[2]) are used for SGPIO signaling. When cleared, the SGPIO pins are used for direct LED controls. For SGPIO unit 1 - When this bit is set, the SGPIO bus pins (S_ACT[4], S_STAT[4], S_ACT[6], and S_STAT[6]) are used for SGPIO signaling. When cleared, the SGPIO pins are used for direct LED controls.

6.6.2 SGPIO Programmable Blink Register x — SGPBRx

This SGPIO Programmable Blink Register x - SGPBRx is used to program the programmable blink patterns. Each output signal supports two programmable blink patterns and each pattern can be programmed using two 4-bit fields. The two 4-bit fields allow the user to program a low and a high duration time. The shortest low/high duration time that can be programmed is 125 ms, whereas the longest low/high duration time that can be programmed is two seconds.

Table 322. SGPIO Programmable Blink Register x - SGPBRx (Sheet 1 of 2)

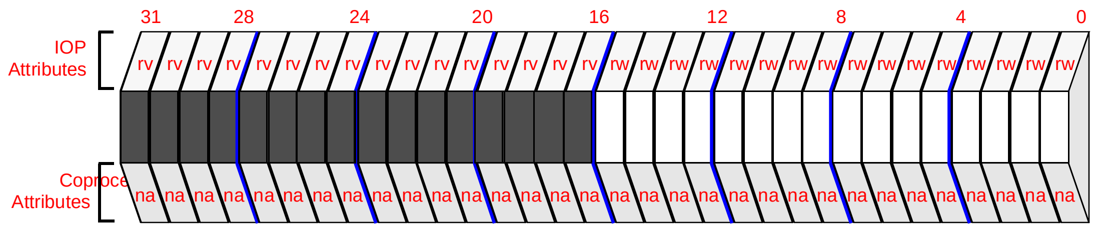


SGPIO #	Intel XScale® processor internal bus address offset
0	+2604H
1	+2684H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:16	0000H	Reserved.
15:12	0000_2	Programmable Pattern B High Duration Time - This field is used to program the high duration time in millisecond for pattern B. Bits Duration (millisecond) 0000_2 125 0001_2 250 0010_2 375 0011_2 500 0100_2 625 0101_2 750 0110_2 875 0111_2 1000 1000_2 1125 1001_2 1250 1010_2 1375 1011_2 1500 1100_2 1625 1101_2 1750 1110_2 1875 1111_2 2000
11:08	0000_2	Programmable Pattern B Low Duration Time - This field is used to program the low duration time in millisecond for pattern B. Bits Duration (millisecond) 0000_2 125 0001_2 250 0010_2 375 0011_2 500 0100_2 625 0101_2 750 0110_2 875 0111_2 1000 1000_2 1125 10001_2 1250 1010_2 1375 1011_2 1500 1100_2 1625 1101_2 1750 1110_2 1875 1111_2 2000

Table 322. SGPIO Programmable Blink Register x - SGPBRx (Sheet 2 of 2)



SGPIO #	Intel XScale® processor internal bus address offset
0	+2604H
1	+2684H

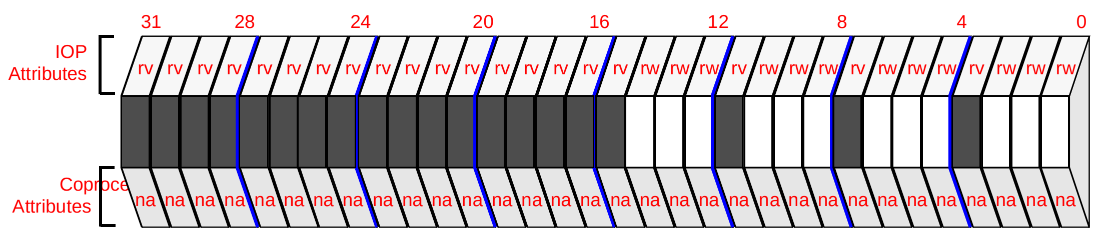
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
07:04	0000_2	Programmable Pattern A High Duration Time - This field is used to program the high duration time in millisecond for pattern A. Bits Duration (millisecond) 0000_2 125 0001_2 250 0010_2 375 0011_2 500 0100_2 625 0101_2 750 0110_2 875 0111_2 1000 1000_2 1125 10001_2 1250 1010_2 1375 1011_2 1500 1100_2 1625 1101_2 1750 1110_2 1875 1111_2 2000
03:00	0000_2	Programmable Pattern A Low Duration Time - This field is used to program the low duration time in millisecond for pattern A. Bits Duration (millisecond) 0000_2 125 0001_2 250 0010_2 375 0011_2 500 0100_2 625 0101_2 750 0110_2 875 0111_2 1000 1000_2 1125 10001_2 1250 1010_2 1375 1011_2 1500 1100_2 1625 1101_2 1750 1110_2 1875 1111_2 2000

6.6.3 SGPIO Start Drive Lower Register x — SGSDLRx

The SGPIO Start Drive Lower Register x - SGSDLRx is used to program the drive outputs order as they are shifted out on the serial bit stream. For example, after the Vendor-Specific Code bits are shifted out on the SDataout pin, the user can choose in which order each drive's outputs are shifted out. This register controls the steering of drive inputs[0:3] of the multiplexer block.

Table 323. SGPIO Start Drive Lower Register x — SGSDLRx (Sheet 1 of 2)

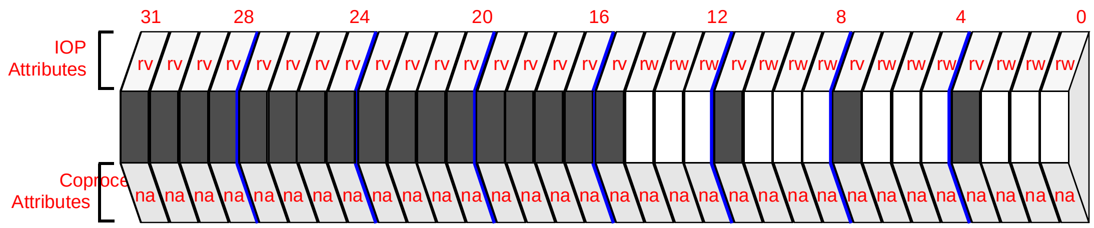


SGPIO #	Intel XScale® processor internal bus address offset
0	+2608H
1	+2688H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:16	0000H	Reserved.
15	0_2	Reserved.
14:12	011_2	Output 3 Select Bits - This bit field selects which ***Input[0:7]*** of the Multiplexer Block is selected to drive ***Output 3***. Refer to Figure 52, "Output Signal Routing" on page 469. Bits Output Number 000_2 0 001_2 1 010_2 2 011_2 3 100_2 4 101_2 5 110_2 6 111_2 7
11	0_2	Reserved.
10:08	010_2	Output 2 Select Bits - This bit field selects which ***Input[0:7]*** of the Multiplexer Block is selected to drive ***Output 2***. Refer to Figure 52, "Output Signal Routing" on page 469. Bits Output Number 000_2 0 001_2 1 010_2 2 011_2 3 100_2 4 101_2 5 110_2 6 111_2 7
07	0_2	Reserved.
06:04	001_2	Output 1 Select Bits - This bit field selects which ***Input[0:7]*** of the Multiplexer Block is selected to drive ***Output 1***. Refer to Figure 52, "Output Signal Routing" on page 469. Bits Output Number 000_2 0 001_2 1 010_2 2 011_2 3 100_2 4 101_2 5 110_2 6 111_2 7
03	0_2	Reserved.

Table 323. SGPIO Start Drive Lower Register x — SGSDLRx (Sheet 2 of 2)



SGPIO #	Intel XScale® processor internal bus address offset
0	+2608H
1	+2688H

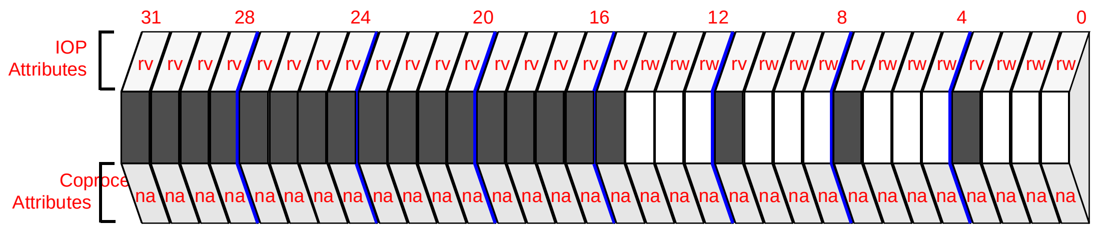
Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
02:00	000_2	Output 0 Select Bits - This bit field selects which ***Input[0:7]*** of the Multiplexer Block is selected to drive ***Output 0***. Refer to Figure 52, "Output Signal Routing" on page 469. Bits Output Number 000_2 0 001_2 1 010_2 2 011_2 3 100_2 4 101_2 5 110_2 6 111_2 7

6.6.4 SGPIO Start Drive Upper Register x — SGSDURx

The SGPIO Start Drive Upper Register x — SGSDURx is used to program the drive output order as they are shifted out on the serial bit stream. For example, after the Vendor-Specific Code bits are shifted out on the SDataout pin, the user can choose in which order each drive outputs are shifted out. This register controls the steering of drive inputs[4:7] of the multiplexer block.

Table 324. SGPIO Start Drive Upper Register x — SGSDURx (Sheet 1 of 2)



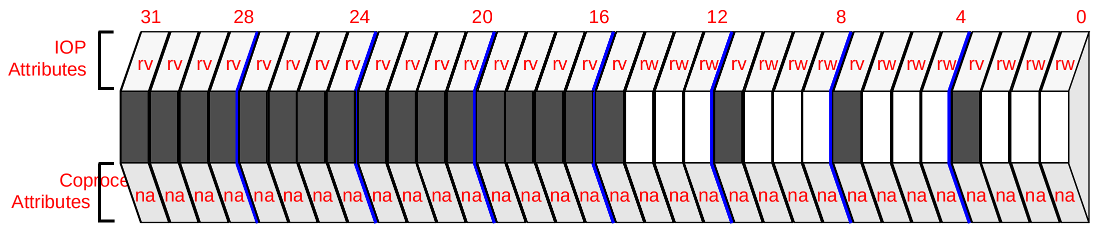
SGPIO #	Intel XScale® processor internal bus address offset
0	+260CH
1	+268CH

Attribute Legend:

RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
31:16	0000H	Reserved.
15	0_2	Reserved.
14:12	111_2	Output 7 Select Bits - This bit field selects which ***Input[0:7]*** of the Multiplexer Block is selected to drive ***Output 7***. Refer to Figure 52, "Output Signal Routing" on page 469. Bits Output Number 000_2 0 001_2 1 010_2 2 011_2 3 100_2 4 101_2 5 110_2 6 111_2 7
11	0_2	Reserved.
10:08	110_2	Output 6 Select Bits - This bit field selects which ***Input[0:7]*** of the Multiplexer Block is selected to drive ***Output 6***. Refer to Figure 52, "Output Signal Routing" on page 469. Bits Output Number 000_2 0 001_2 1 010_2 2 011_2 3 100_2 4 101_2 5 110_2 6 111_2 7
07	0_2	Reserved.
06:04	101_2	Output 5 Select Bits - This bit field selects which ***Input[0:7]*** of the Multiplexer Block is selected to drive ***Output 5***. Refer to Figure 52, "Output Signal Routing" on page 469. Bits Output Number 000_2 0 001_2 1 010_2 2 011_2 3 100_2 4 101_2 5 110_2 6 111_2 7
03	0_2	Reserved.

Table 324. SGPIO Start Drive Upper Register x — SGSDURx (Sheet 2 of 2)



SGPIO #	Intel XScale® processor internal bus address offset
0	+260CH
1	+268CH

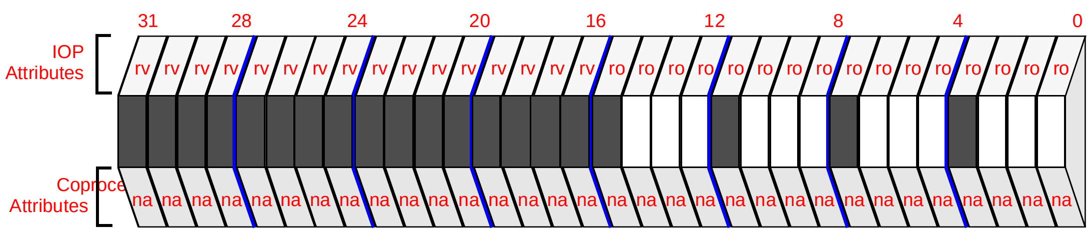
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
02:00	100_2	Output 4 Select Bits - This bit field selects which ***Input[0:7]*** of the Multiplexer Block is selected to drive ***Output 4***. Block. Refer to Figure 52, "Output Signal Routing" on page 469. Bits Output Number 000_2 0 001_2 1 010_2 2 011_2 3 100_2 4 101_2 5 110_2 6 111_2 7

6.6.5 SGPIO Serial Input Data Lower Register x — SGSIDLRx

The SGPIO Serial Input Data Lower Register x - SGIDLRx is used to read the input data bits. Each drive sends three bits. This register provides the input data bits for drives 0, 1, 2, and 3.

Table 325. SGPIO Serial Input Data Lower Register x - SGSIDLRx



SGPIO #	Intel XScale® processor internal bus address offset
0	+2610H
1	+2690H

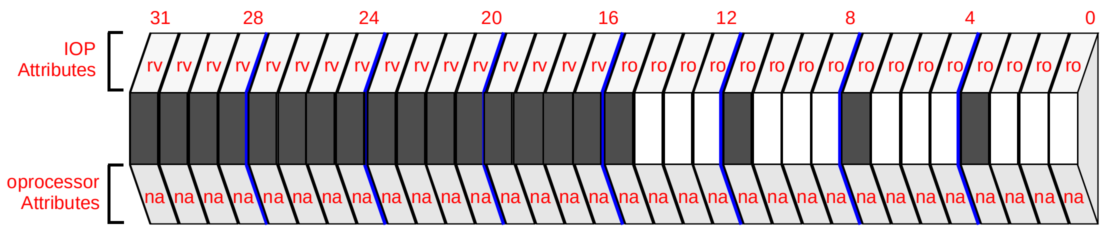
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:16	0000H	Reserved.
15	0_2	Reserved.
14:12	000_2	Drive 3 input data.
11	0_2	Reserved.
10:08	000_2	Drive 2 input data.
07	0_2	Reserved.
06:04	000_2	Drive 1 input data.
03	0_2	Reserved.
02:00	000_2	Drive 0 input data.

6.6.6 SGPIO Serial Input Data Upper Register x — SGSIDURx

The SGPIO Serial Input Data Upper Register x - SGIDURx is used to read the drive input data bits. Each drive sends three bits. This register provides the input data bits for drives 4, 5, 6, and 7.

Table 326. SGPIO Serial Input Data Upper Register x - SGSIDURx



SGPIO #	Intel XScale® processor internal bus address offset
0	+2614H
1	+2694H

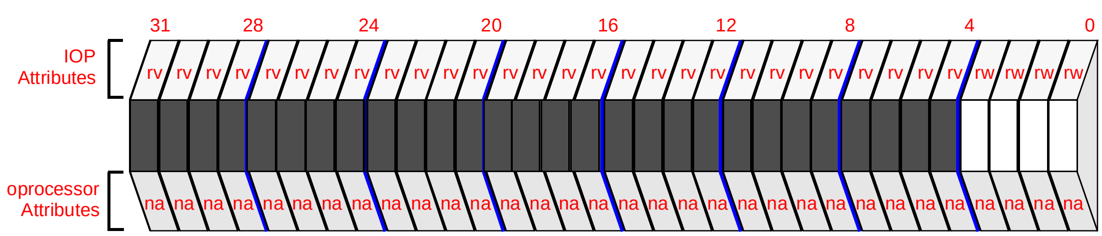
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:16	0000H	Reserved.
15	0_2	Reserved.
14:12	000_2	Drive 7 input data.
11	0_2	Reserved.
10:08	000_2	Drive 6 input data.
07	0_2	Reserved.
06:04	000_2	Drive 5 input data.
03	0_2	Reserved.
02:00	000_2	Drive 4 input data.

6.6.7 SGPIO Vendor Specific Code Register x — SGVSCRx

SGPIO Vendor Specific Code Register x is used to program vendor-specific code. The four bits vendor-specific code is the first four bits shifted on the SLoad pin after SLoad is driven high.

Table 327. SGPIO Vendor Specific Code Register x - SGVSCRx



SGPIO #	Intel XScale® processor internal bus address offset
0	+2618H
1	+2698H

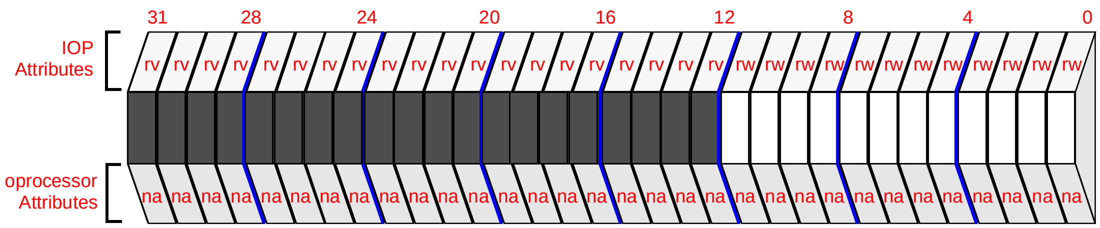
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:04	0000000H	Reserved.
03:00	0H	Vendor Specific data.

6.6.8 SGPIO Output Data Select Register[0:7] x — SGODSR[0:7]x

The SGPIO Output Data Select Register[0:7] x - SGODSR[0:7]x provides the bit fields to select the output signals. Each drive can support up to three output signals.

Table 328. SGPIO Output Data Select Register[0:7] x - SGODSR[0:7]x



SGPIO 0 — Intel XScale® processor internal bus address offset

SGPIO 1 — Intel XScale® processor internal bus address offset

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Drive # (SGPIO 0)	Offset	Drive # (SGPIO 1)	Offset
0	+2620H	0	+26A0H
1	+2624H	1	+26A4H
2	+2628H	2	+26A8H
3	+262CH	3	+26ACH
4	+2630H	4	+26B0H
5	+2634H	5	+26B4H
6	+2638H	6	+26B8H
7	+263CH	7	+26BCH

Bit	Default	Description
31:12	00000H	Reserved.
11	0_2	OD2 JOG Enable - When set this bit enables the jog mechanism to be applied on the input selected by bits[09:08]. When cleared, the selected input is not altered.
10	0_2	Invert OD2 Selected Input - When set this bit causes the input selected by bits[09:08] to be inverted. When cleared, the selected input is not altered.
09:08	00_2	OD2 Input Select - This field selects the input that drives output OD2 of Drive N, where N = 0 - 7. Bits Selection 00_2 Fixed - High 01_2 Programmable pattern A 10_2 Programmable pattern B 11_2 Reserved
07	0_2	OD1 JOG Enable - When set this bit enables the jog mechanism to be applied on the input selected by bits[05:04]. When cleared, the selected input is not altered.
06	0_2	Invert OD1 Selected Input - When set this bit causes the input selected by bits[05:04] to be inverted. When cleared, the selected input is not altered.
05:04	00_2	OD1 Input Select - This field selects the input that drives output OD1 of Drive N, where N = 0 - 7. Bits Selection 00_2 Fixed - High 01_2 Programmable pattern A 10_2 Programmable pattern B 11_2 Protocol Engine Status
03	0_2	OD0 JOG Enable - When set this bit enables the jog mechanism to be applied on the input selected by bits[01:00]. When cleared, the selected input is not altered.
02	0_2	Invert OD0 Selected Input - When set this bit causes the input selected by bits[01:00] to be inverted. When cleared, the selected input is not altered.
01:00	00_2	OD0 Input Select - This field selects the input that drives output OD0 of Drive N, where N = 0 - 7. Bits Selection 00_2 Fixed - High 01_2 Programmable pattern A 10_2 Programmable pattern B 11_2 Protocol Engine Activity

7.0 System Controller (SC) and Internal Bus Bridge

This chapter describes the System Controllers (SC) of the Intel® 413808 and 413812 I/O Controllers (4138xx). The System Controller controls the internal bus and its agents. There are two System Controllers on 4138xx since there are two internal busses.

7.1 Overview

The System Controller controls the internal bus agents arbitrating for the internal bus. The Internal Bus on 4138xx contains a separate address bus arbiter and data bus arbiter as the address and data busses are completely de-multiplexed. There are two internal busses on 4138xx and therefore there are two system controllers implemented — one for the North Internal Bus and one for the South Internal Bus.

- The north internal bus SC controls the two Intel XScale® processors, the DDR Memory Controller, the Bridge, and the SAS Interface.
- The south internal bus SC controls the ATU-E, ATU-X, the Bridge, the DDR SDRAM Memory Controller, the Application DMAs, the PBI, and the APB interface.

In addition to providing the address bus and data bus arbitration functionality, the SC also initiates address and data transactions once the bus has been granted. The SC is also the central hub which takes as inputs all the agents address and data busses, and then controlling how the address bus or data bus is routed to an agent.

The SC also provides hardware functionality that can be used to force address and data parity errors. This feature allows software to test error handling routines by forcing address or data parity error.

7.2 Theory of Operation

7.2.1 System Controller

The XSI System Controller (SC) is the arbiter for the XSI bus. There is one SC for the North XSI bus and one for the South XSI bus since there are two XSI busses on 4138xx. The XSI bus supports fully demultiplex and independent address and data paths, thus the SC performs arbitration for address requests and data requests separately. Up to 15 agents are supported by the SC.

For address request arbitration, the SC receives address bus request from each requesting agent seeking to master requests on the address bus and implements an arbitration algorithm to generate address bus grant to grant the bus to a particular requesting agent. The SC actually frames the address transactions, and therefore drives the address strobe along with the address bus grant. The assertion of address strobe indicates that a valid XSI bus command is driven the following cycle.

For data request arbitration, the SC receives data bus request from each requesting agent seeking to master a data transaction on the data bus and implements an arbitration algorithm to generate data bus grant to grant the data bus to a particular requesting agent. The SC frames the data transactions, and therefore initiates the data transaction along with the data bus grant.

The SC ORs several bus outputs from each agent and routes the ORed signals back to each agent, including the Address Qualifier Signals (AQS) and Data Qualifier Signals (DQS).

Both the address and data bus arbiters support simple round-robin algorithms. Each internal bus initiator can be individually enabled or disabled. When an initiator is disabled, any address request made by that initiator is not granted the internal bus. Each arbiter parks on the last master.

7.2.2 Internal Bus Requester IDs

Each of the initiator/requester on the 4138xx has an assigned unique ID, which helps identify an initiator when returning read data and for the purpose of logging transaction errors. Table 329 lists the encoded initiator IDs.

Table 329. Intel® 413808 and 413812 I/O Controllers in TPER Mode Initiator IDs

Internal Bus Initiator	Initiator ID
Reserved	0000_2
Intel XScale® processor 0 (coreID = 0)	0001_2
Intel XScale® processor 1 (coreID = 1)	0010_2
ATU-X	0011_2
ATU-E	0100_2
Revered	0101_2
Revered	0110_2
Messaging Unit[a]	0111_2
Revered	1000_2
SMBus	1001_2
Revered	1010_2
Reserved	1011_2 through 1111_2

a. The Messaging Unit acts as internal bus initiator only for issuing MSI or MSI-X writes to the ATUE or ATUX.

7.2.3 Parity Testing

The 4138xx supports parity protections on both the 36-bit address and 128-bit data bus south internal bus. Parity is supported on a byte-wise basis. The SC provides hardware test features that allows the user to force address or data parity errors on the internal bus. This feature allows the user to test software error handling routines by forcing an address or data parity error on the internal bus. Parity error is forced by XORing (inverting) good parity bit or bits before they are driven on the bus. The 4138xx provides separate sets of error registers for injecting address bus and data bus parity error. The 4138xx provides an address mask register and a data mask register. The user can program the mask registers to select which parity bit(s) ought to be inverted. 4138xx also provides a register that allows the programmer to select the initiator of a given transaction by providing the ID of the initiator.

This register also provides an enable bit which must be set by software and is reset by hardware. This provides a way of injecting an error only once — a one-shot process. For example, error is injected only during the address cycle for an address parity test, and during the first data cycle for a data parity test. Table 331 lists the Initiator IDs that must be programmed in the *Initiator ID* field of SIBATCR for address parity testing during address request. Table 331 lists the Initiator IDs that must be programmed in the *Initiator ID* field of SIBDTCR for data parity testing during writes. Table 331 lists the Initiator IDs that must be programmed in the *Initiator ID* field of SIBDTCR for data parity testing during read completions.

The initiator in this context is used to identify the source of the address or data. For example, when data parity error is to be injected while the DDR MCU is returning read completion data, the DDR MCU Initiator ID must be used. However, when data parity error is to be injected when the ATU is writing data to the DDR Memory, the ATU Initiator ID must be used.

Figure 55. Typical Internal Bus System Controller Block Diagram

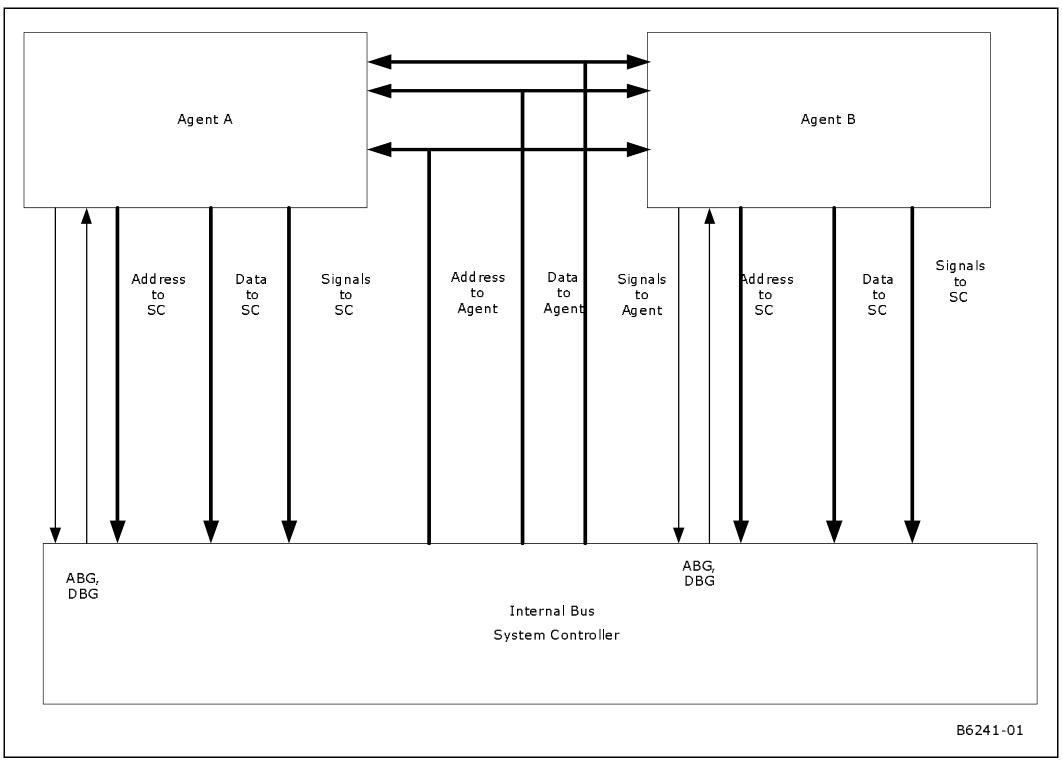


Table 330. Address and Data Parity Testing Initiator IDs

Internal Bus Initiator	Initiator ID
Reserved	0000_2
Intel XScale® processor 0 (coreID = 0)	0001_2
Intel XScale® processor 1 (coreID = 1)	0010_2
ATU-X	0011_2
ATU-E	0100_2
Reserved	0101_2
Reserved	0110_2
Messaging Unit	0111_2
Reserved	1000_2
SMBus and **PMON**	1001_2
Reserved	1010_2
Reserved	1011_2 through 1111_2

Note: This table contains the Initiator IDs for injecting address parity error when these initiators are making address requests. In addition these same Initiator IDs can be used when injecting data parity error when these initiators are pushing data during writes.

Table 331. Data Parity Testing Completer IDs

Internal Bus Initiator	Initiator ID
Reserved	0000_2
Not Applicable[a]	0001_2
Not Applicable	0010_2
ATU-X	0011_2
ATU-E	0100_2
Reserved	0101_2
Reserved	0110_2
Messaging Unit	0111_2
Reserved	1000_2
UART, I2C, GPIO, PBI, and **PMON**	1001_2
Not Applicable	1010_2
Reserved	1011_2 through 1111_2

Note: This table contains the Initiator IDs for injecting data parity error when these Initiators are returning data during read completions. Note that in this scenario the initiator of the data transaction is the actual completer.

a. Not applicable implies that the ID associated with that initiator does not return completion data.

7.3 Internal Bus Bridge

This section describes the internal bus bridge. The internal bus bridge isolates traffic on the north internal bus and the south internal bus. The internal bus bridge is a bidirectional bridge. Transactions targeting the south internal bus from the north internal bus are referred to as ***outbound transactions***. Transactions targeting the north internal bus from the south internal bus are referred to as ***inbound transactions***.

7.3.1 Theory of Operation

The bridge forwards north internal bus address requests that are not claimed on the north internal bus. For example, the bridge performs subtractive decoding on the north internal bus interface. The bridge claims and forwards a north internal bus data transaction when the ID provided with the data transaction matches the ID of a previously claimed write address request.

On the south internal bus, the XBG defines a Bridge Memory Window. Transactions on the south internal bus that target the Bridge Memory Window are claimed and forwarded to the north internal bus. The bridge claims and forwards a south internal bus data transaction when the ID provided with the data transaction matches the ID of a previously claimed write address request. The Bridge south interface also claims transactions that target the Bridge memory-mapped registers. Intel XScale® processor transactions that target the Bridge memory-mapped registers are also propagated to the south internal bus, and then claimed by the Bridge south interface.

Both the north and south internal busses on 4138xx support the same bus protocol. The internal bus operates by performing split transactions on both read and write address requests. Every address request contains an Address Transfer ID (ATID), which is the ID of the initiator. Every initiator has a unique ATID. When an initiator makes a read request, it drives its ATID along with the address request. Once the Bridge claims the request, it maintains the ATID, which it uses when returning read completion data in the form of a Data Transfer ID (DTID). The initiator of the read request claims the read completion transaction by observing that the DTID matches its ATID. For a write request, the initiator drives its ATID along with the address request. Once the Bridge claims the request, it maintains the ATID, which it uses to claim a data transaction. The initiator of the write request drives a Data Transfer ID (DTID) with the data transaction. The Bridge claims the data transaction by observing that the DTID matches the stored ATID.

Each of the initiator/requester on the 4138xx has an assigned unique ID, which helps identify an initiator when returning read data and for the purpose of logging transaction errors. Table 329 lists the encoded initiator IDs. The bridge uses the same initiator ID of a transaction it claims when forwarding the transaction on the opposite internal bus.

7.3.2 Internal Bus Commands

Table 332 lists the internal bus commands that are supported on the north and south bridge interfaces.

Table 332. Bridge supported Internal Bus Commands

Internal Bus Command Encoding	Internal Bus Command Type	Claimed on North Internal Bus	Generated on South Internal Bus	Claimed on South Internal Bus	Generated on North Internal Bus
0000	NULL	No	Yes	No	Yes
0001	Sync	No	No	No	No
0010	Special	No	No	No	No
0011	Reserved	No	No	No	No
0100	Reserved	No	No	No	No
0101	Reserved	No	No	No	No
0110	Reserved	No	No	No	No
0111	Reserved for TLBIE	No	No	No	No
1000	Read	Yes	Yes	Yes	Yes
1001	Read Line	Yes	Yes	Yes	Yes
1010	Reserved for Invalidate Line	No	No	No	No
1011	Read and Invalidate Line	Yes	Yes	Yes	Yes
1100	Write	Yes	Yes	Yes	Yes
1101	Reserved for Clean and Invalidate Line	No	No	No	No
1110	Write Line	Yes	Yes	Yes	Yes
1111	Reserved for Clean Line	No	No	No	No

7.3.3 Transaction Queues

Both the north and south interfaces of the bridge support a read queue of 8 entries and each supporting up to 32 Bytes of data buffers. Both interfaces support a write queue of 8 entries, each supporting up to 32 Bytes of data buffers.

Note: The bridge master-aborts any transaction request that tries to cross a 32-byte boundary. Since each data buffer is 32 bytes in size, the bridge can only transfer a maximum byte-count of 32 bytes of data per request and only when the address is aligned on a 32-byte boundary. In other words, for a non 32-byte aligned address, the sum of the non-aligned address and byte-count has to be less than the next 32-byte aligned address boundary for the bridge to enqueue the request.

7.3.4 Bridge Memory Window

The North Internal Bus interface of the Bridge performs subtractive decoding. For example, transactions on the north internal bus that are not claimed by other targets on the north internal bus are claimed by the Bridge North Interface.

The South Bridge Interface performs positive decoding. The Bridge provides a Bridge Memory Window defined by the Bridge Window Base Address Register — BWBAR and the Bridge Limit Register — BWLR. Transactions on the south internal bus that target this memory window are claimed and forwarded to the north internal bus. The South Bridge Interface also claims transactions that target the Bridge memory-mapped registers. The memory-mapped registers are only accessible from the south internal bus. For example, transactions to the Bridge memory-mapped registers by the Intel XScale® processors that reside on the north internal bus, are propagated from the north internal bus to the south internal bus via the Bridge, and then claimed by the Bridge on the south interface.

Note: For bridge memory window access overlapping the bridge memory-mapped register space (+1780 through +17FFH), the bridge memory window is not accessible, and the bridge memory-mapped register space is addressed.

7.3.5 Ordering and Passing Rules

Table 333 lists the ordering and passing rules requirements for the bridge. Although write requests and write data completions are completely independent transactions on the internal bus, the bridge internally combines a write request with its corresponding write data transaction when enqueuing write requests and when issuing write requests. For example, a write request made to the target side of the bridge is not considered valid in the bridge until the write request and the data for that write request have been received. Similarly, a write request mastered by the bridge is not considered done until the write request and the write data completion have been executed on the internal bus. For that reason, Table 333 shows a write request (Addr(Wr)) and a write data completion (Data(Wr)) as one entity.

— Addr(Wr) — Write Address Request

— Addr(Rd) — Read Address Request

— Data(Rd) — Read Data Completion

— Data(Wr) — Write Data Completion

Table 333. Ordering and Passing Rules for both Inbound and Outbound Transactions

Row Pass Column?	Addr(Wr)/Data(Wr) (Column 1)	Addr(Rd) (Column 2)	Data(Rd) (Column 3)
Addr(Wr)/Data(Wr) (Row 1)	No (Row 1, Column 1)	Yes[a] (Row 1, Column 2)	Yes/No (Row 1, Column 3)
Addr(Rd) (Row 2)	No (Row 2, Column 1)	No (Row 2, Column 2)	Yes/No (Row 2, Column 3)
Data(Rd) (Row 3)	No (Row 3, Column 1)	Yes (Row 3, Column 2)	Yes/No (Row 3, Column 3)
No - The row is *not* allowed to pass the column Yes - The row must be allowed to pass the column Yes/No - There are no requirements			

a. Strong Ordering Rule Requirements for Outbound Write Requests in order to maintain data coherency. Refer to Section 7.3.5.1, Strong Ordering Rule Requirements.

7.3.5.1 Strong Ordering Rule Requirements

This rule applies for Write Requests targeting only the north interface of the Bridge. For example, outbound write requests from the north internal bus to the south internal bus. Although the bridge must allow write requests to pass read requests as shown in Table 333 (Row 1, Column 2), to maintain data coherency the bridge does not allow enqueuing a write request whose cacheline address matches that of a previously enqueued read request. The write request is retried on the north internal bus until the read request has been retired. For example, the data for the read request has been returned.

Note: The south interface of the South Bridge does not follow the Strong Ordering Rule Requirements described in Section 7.3.5.1, Strong Ordering Rule Requirements.

7.3.6 Parity Support

The bridge supports parity as required by the south internal bus. The south internal bus supports both byte-wise address and data parity. Therefore, as a initiator the bridge is responsible to drive byte-wise parity on the south internal bus on both the 36-bit address bus and the 128-bit data bus. Also when completing read requests for south internal bus initiators, the bridge drives data parity. The south internal bus supports even parity. Note that the north internal bus does not support any parity. The bridge also supports byte-wise parity on the internal data buffers.

7.3.6.1 Address Parity Generation

Only the bridge south interface generates byte-wise address parity on address it initiates on the south internal bus.

7.3.6.2 Address Parity Checking

Only the south interface of the bridge verifies address parity when claiming south internal bus write transactions.

7.3.6.3 Data Parity on Outbound Transactions

For an outbound transaction (transaction flowing from the north internal bus to the south internal bus as either a read completion or write request), the bridge generates data parity as the data enters the north bridge interface. The data and its parity are stored in the internal data buffers. When the transaction is initiated on the south internal bus, the bridge simply drives the data along with the parity as stored in the data buffers. For example, the bridge does not generate parity. The receiver of the data on the south internal bus verifies the data parity.

7.3.6.4 Data Parity on Inbound Transactions

For an inbound transaction (transaction flowing from the south internal bus to the north internal bus as either a read completion or a write request), the bridge simply writes the data along with the received data parity to the internal data buffers. The bridge then checks the data parity while forwarding the transaction to the north internal bus. When the bridge detects a parity error on a write transaction, the bridge logs the error and also forwards the transaction on the north internal bus. When the bridge detects a parity error on a read completion, the bridge logs the error and assert DABORT instead of completing the transaction on the north internal bus. Refer to the following error logging registers: "Bridge Error Control and Status Register — BECSR", "Bridge Error Address Register — BERAR" and the "Bridge Error Upper Address Register — BERUAR"

7.3.7 Error Detection and Handling

The Bridge provides a set of error logging registers that are used to log any error that are encountered by the Bridge: north interface or south interface. Only one error is logged, when more errors occur when one error is already logged, the bridge would indicate that it detected more errors, but does not log these newer errors.

The bridge drives a single interrupt (South Internal Bus Bridge Error Interrupt Pending) to the Interrupt Controller Unit (ICU). An interrupt may be generated by the bridge to report any error condition detected to the Intel XScale® processor by setting bit 16 of BECSR. Whenever the bridge toggles bit 0 of BECSR from 0 to 1, an interrupt is generated to the core.

7.3.7.1 Bridge North Internal Bus Interface Error

The following conditions may be encountered by the Bridge North Interface:

- Master Abort on the North Internal Bus interface. This condition may happen when the bridge attempts an address request on the north internal bus and the request is not claimed by any target. The bridge would log the error as a Master Abort.
    - For a write request, the bridge simply logs the master abort error condition and discard the write data.
    - For a read request, the Bridge logs the Master Abort. In addition, the Bridge completes the read completion as follows: it indicates to the initiating agent on the south internal bus that it encountered an error by performing a target abort during the read completion.
- Address Request Error on the north internal bus interface. This condition may happen when a target on the north internal bus detected an error during the address request. For example, the target may indicate that the byte-count is out of range.
    - For a write request, the bridge logs the address request error condition and discard the write data.
    - For read request, the Bridge logs the address request error. In addition, the Bridge completes the read completion as follows: it indicates to the initiating agent on the south internal bus that it encountered an error by performing a target abort during the read completion.
- Target Abort on the north internal bus. This condition may occur when a target claimed a previously issued read request, but is unable to return the read completion data. The bridge would log the error as a target abort.
    - The Bridge logs the target abort. In addition, the Bridge completes the read completion as follows: it indicates to the initiating agent on the south internal bus that it encountered an error by performing a target abort during the read completion. Target Abort on the north internal bus. This condition may occur when the north interface detects a parity error during the data phase of a write request to the north internal bus, or during a read completion to the north internal bus.
    - For a read completion, the Bridge logs the target abort condition and completes the read completion as follows: it indicates to the initiating agent on the north internal bus that it encountered an error by performing a target abort during the read completion. For a write transaction, the Bridge logs the target abort condition and completes the write transaction on the north internal bus.

7.3.7.2 Bridge South Internal Bus Interface Error

The following conditions may be encountered by the Bridge South Interface:

- Master Abort on the South Internal Bus interface. This condition may happen when the bridge attempts an address request on the south internal bus and the request is not claimed by any target. The bridge would log the error as a master abort.
  - For a write request, the bridge logs the master abort error condition and discard the write data.
  - For a read request, the Bridge logs the master abort. In addition, the Bridge completes the read completion as follows: it indicates to the initiating agent on the north internal bus that it encountered an error by performing a target abort during the read completion.
- Address Request Error on the south internal bus interface. This condition may happen when a target on the south internal bus detected an error during the address request. For example, the target may indicate that the byte-count is out of range or the target detected an address parity error.
  - For a write request, the bridge logs the address request error condition and discard the write data.
  - For read request, the Bridge logs the address request error. In addition, the Bridge completes the read completion as follows: it indicates to the initiating agent on the north internal bus that it encountered an error by performing a target abort during the read completion.
- Target Abort on the south internal bus. This condition may occur when a target claimed a previously issued read request, but is unable to return the read completion data. The bridge would log the error as a target abort.
  - The Bridge logs the target abort. In addition, the Bridge completes the read completion as follows: it indicates to the initiating agent on the north internal bus that it encountered an error by performing a target abort during the read completion. Address Parity Error on the south internal bus. This condition may occur when the Bridge south interface is acting as a target and either observes AERR asserted by another target on the south internal bus, or detects an address parity error while claiming an address request.

7.4 System Controller Register Definitions

The following registers are located in the Peripheral Memory-Mapped Register (PMMR) address space. They are accessible through the south internal bus accesses. The Internal Bus Arbitration Control Register provides controls for both the North and South Internal address busses. The south internal bus address and data test registers are used to force address or data parity errors on the south internal bus respectively. Note that the north internal bus does not support address and data parity.

- Internal Bus Arbitration Control Register
- South Internal Bus Address Test Register
- South Internal Bus Data Test Register
- Peripheral Memory-Mapped Register Base Address Register

The system controller only claims the address offset range +1640H through +164FH.

7.5 Internal Bus Bridge Register Definitions

The following registers are located in the Peripheral Memory-Mapped Register (PMMR) address space. They are only accessible from the south internal bus. Accesses to the Bridge registers that originate from the north internal bus are propagated to the south internal bus, and then claimed by the Bridge on the south interface. The Bridge Error Status register indicates the type of error that was encountered by the bridge on either the north or south interfaces. The Bridge Error Address and Error Upper Address registers indicate the address of the request that encountered the error. The Bridge Window Base Address and Window Limit Registers together define a memory window for the Bridge to claim transactions on the south internal bus.

- Bridge Window Base Address Register
- Bridge Widow Upper Base Address Register
- Bridge Window Limit Register
- Bridge Error Status Register
- Bridge Error Address Register
- Bridge Error Upper Address Register

The internal bus bridge only claims the address offset range +1780H through +1797H.

7.5.1 Internal Bus Arbitration Control Register — IBACR

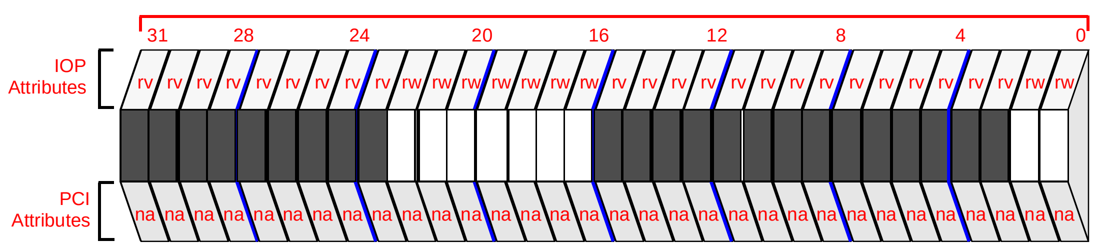
The 4138xx has two internal buses: the north internal bus and the south internal bus. Refer to the block diagram shown in Figure 2, "Intel® 413808 and 413812 I/O Controllers in TPER Mode Functional Block Diagram" on page 43. The two internal buses are identical and provide de-multiplexed address and data buses. Therefore, each internal bus consists of two independent arbiters: an ***address*** arbiter and a ***data*** arbiter. The north internal bus address arbiter controls the north internal bus ***address*** initiators, whereas the north internal data arbiter controls the north internal bus ***data*** initiators. Similarly, the south internal bus address arbiter controls the south internal bus ***address*** initiators, whereas the south internal data arbiter controls the south internal bus ***data*** initiators.

The IBACR can be used to enable or disable an internal bus ***address*** initiator from acquiring the internal address bus. Note that this register does not disable the agent. It only prevents an address initiator arbitrating for the internal address bus from acquiring the internal address bus. Bits [15:0] control the address initiators on the north internal address bus, and bits [31:16] control the address initiators on the south internal address bus.

Warning: Since the internal address arbiter parks on an agent that was last granted the internal address bus, disabling a parked agent using the IBACR does not take effect immediately. The IBACR disable bit is observed by the internal address arbiter only when an agent arbitrates for the internal address bus.

Note: The data initiators cannot be disabled.

Table 334. Internal Bus Arbitration Control Register — IBACR (Sheet 1 of 2)



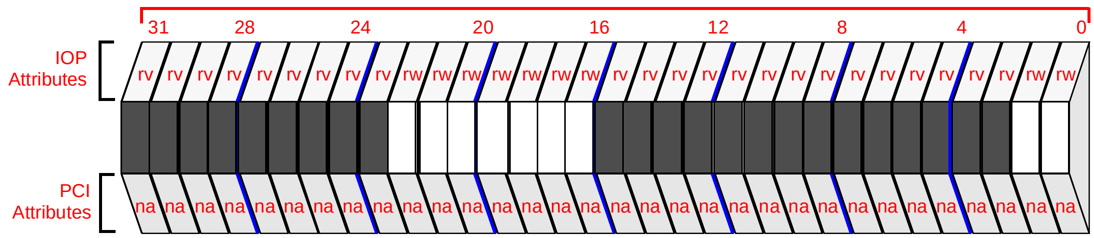
internal bus address offset

South XBG +1640H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:25	00H	Reserved.
24	0_2	Reserved.
23	0_2	Reserved.
22	0_2	Reserved.
21	0_2	DDRMCU/MU Group Control — this bit controls the MU initiator. The MU address initiator is not granted the south internal address bus when this bit is set. 0 = Enabled 1 = Disabled ***Note:*** The LMU acts as an internal bus address initiator only for issuing MSI or MSI-X writes to the ATUE or ATUX. Otherwise, the MU is a target device on the internal bus, which are accessed by the Cores or indirectly by the PCI Host via the ATUs.
20	0_2	Reserved.

Table 334. Internal Bus Arbitration Control Register — IBACR (Sheet 2 of 2)



South XBG — internal bus address offset +1640H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

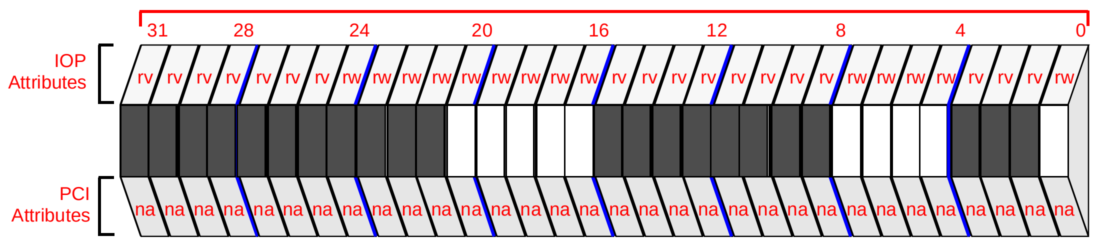
Bit	Default	Description
19	0_2	Application DMA Group Control — this bit controls the Application DMA Initiators. The ADMA address initiators are not granted the south internal address bus when this bit is set. 0 = Enabled 1 = Disabled
18	0_2	ATU-X Control — this bit controls the ATU-X. The ATU-X address initiator is not granted the south internal address bus when this bit is set. 0 = Enabled 1 = Disabled
17	0_2	ATU-E Control — this bit controls ATU-E. The ATU-E address initiator is not granted the south internal address bus when this bit is set. 0 = Enabled 1 = Disabled
16	0_2	SMBus Group Control — this bit controls the SMBus and **PMON** initiators. The SMBus and **PMON** address initiators are not granted the south internal address bus when this bit is set. 0 = Enabled 1 = Disabled
15:03	000H	Reserved.
02	0_2	Reserved.
01	0_2	XSC coreID1 control — this bit controls coreID1. The Intel XScale® processor with coreID1 address initiator is not granted the north internal address bus when this bit is set. 0 = Enabled 1 = Disabled
00	0_2	XSC coreID0 control — this bit controls coreID0. The Intel XScale® processor with coreID0 address initiator is not granted the north internal address bus when this bit is set. 0 = Enabled 1 = Disabled

7.5.2 South Internal Bus Address Test Control Register — SIBATCR

The SIBATCR can be used to inject an address parity error on the south internal address bus. The user must provide the ID of the initiator and also set the enable bit. The enable bit (when set) is used by hardware to inject an address parity error on the next address transaction provided the programmed ID matches. The parity error is injected in the address phase of the request. The enable bit is cleared by hardware in the cycle that follows the address phase. This is done to prevent recurring parity errors.

Note: In order to inject the address parity error in the desired transaction, the user must try to write this register immediately before the transaction is issued on the internal bus.

Table 335. South Internal Bus Address Test Control Register — SIBATCR



South XBG	internal bus address offset +1644H	Attribute Legend: RV = Reserved PR = Preserved RS = Read/Set RW = Read/Write RC = Read Clear RO = Read Only NA = Not Accessible

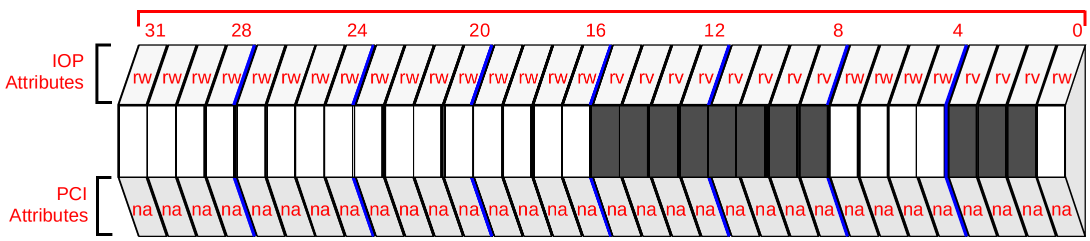
Bit	Default	Description
31:21	000H	Reserved.
20:16	00000_2	Address Parity Mask bits — Each bit of the generated address parity is XORed with the appropriate bits in this mask field before the parity bits are driven on the north internal bus. Bit16 corresponds to address parity bit 0, bit 17 corresponds to address parity bit 1, and so on.
07:04	0000_2	Initiator ID — This field specifies the initiator ID of the address (for example, the unit sourcing the address). Refer to Table 331, "Data Parity Testing Completer IDs" on page 489 for the initiator IDs.
03:01	000_2	Reserved.
00	0_2	Enable bit — When this bit is set, an address parity error is injected on the next address request made when the initiator ID of the transaction matches the specified Initiator ID in bits[07:04]. This bit is set by software and cleared by hardware.

7.5.3 South Internal Bus Data Test Control Register — SIBDTCR

The SIBDTCR can be used to inject a data parity error on the south internal data bus. The user must provide the ID of the initiator and also set the enable bit. The enable bit (when set) is used by hardware to inject a data parity error on the next data transaction provided the programmed ID matches. The parity error is injected in the first data phase. The enable bit is cleared by hardware in the cycle that follows the first data phase. This is done to prevent recurring parity errors.

Note: In order to inject the data parity error in the desired transaction, the user must try to write this register immediately before the transaction is issued on the internal bus.

Table 336. South Internal Bus Data Test Control Register — SIBDTCR



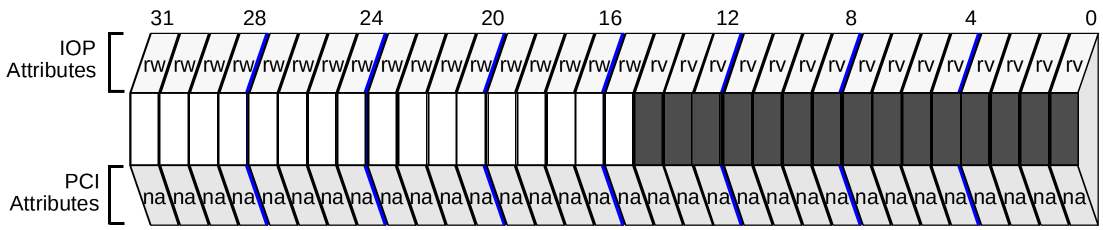
	internal bus address offset	Attribute Legend:	
South XBG	+1648H	RV = Reserved	RW = Read/Write
		PR = Preserved	RC = Read Clear
		RS = Read/Set	RO = Read Only
			NA = Not Accessible

Bit	Default	Description
31:16	0000_2	Data Parity Mask bits — Each bit of the generated data parity is XORed with the appropriate bits in this mask field before the parity bits are driven on the south internal bus. Bit16 corresponds to data parity bit 0, bit 17 corresponds to data parity bit 1, and so on.
07:04	0000_2	Initiator ID — This field specifies the initiator ID of the data (for example, the unit sourcing the data). Refer to Table 331, "Data Parity Testing Completer IDs" on page 489 for the initiator IDs.
03:01	000_2	Reserved.
00	0_2	Enable bit — When this bit is set, a data parity error is injected on the next data request made when the initiator ID of the transaction matches the specified Initiator ID in bits[07:04] of this register. This bit is set by software and cleared by hardware.

7.5.4 Peripheral Memory-Mapped Register Base Address Register — PMMRBAR

This is a 32-bit register that contains the Base or starting address of the Peripheral Memory-Mapped Register space (PMMR). The PMMR space is aligned on a 512-KByte boundary. The PMMRBAR can be used to relocate the PMMR block to any 512-KByte space of the 64 GBytes address space. Bits [31:15] of this register represent bits [35:19] of the 36-bit internal bus address. The PMMR block default starting address after reset is 0 FFD8 0000H.

Table 337. Peripheral Memory-Mapped Register Base Address Register — PMMRBAR



	Intel XScale® processor Local Bus Address	Attribute Legend:	
South XBG	F FFFF FFFCH	RV = Reserved	RW = Read/Write
		PR = Preserved	RC = Read Clear
		RS = Read/Set	RO = Read Only
			NA = Not Accessible

Bit	Default	Description
31:15	0000 1111 1111 1101 1_2	**PMMR Base Address**: These bits define the starting address of the PMMR block. The PMMR block is aligned on a 512-KByte boundary. These bits represent bits [35:19] of the 36-bit internal bus address. ***Note:*** This value cannot be changed or the Transport Firmware does not execute.
14:00	000 0000 0000 0000_2	Reserved.

7.5.5 Determining Block Sizes for Memory Windows

The memory window size can be determined by writing ones to the appropriate upper bits of the limit register. The binary-weighted value of the first non-zero bit set in the limit register indicates the size of the memory window. Table 338 describes the relationship between limit register values and the byte sizes of the memory window.

Table 338. Memory Block Size Limit Register Value

Limit Register Value[a]	Size (in Bytes)	Limit Register Value	Size (in Bytes)
FFFFF000H	4K	FF000000H	16M
FFFFE000H	8K	FE000000H	32M
FFFFC000H	16K	FC000000H	64M
FFFF8000H	32K	F8000000H	128M
FFFF0000H	64K	F0000000H	256M
FFFE0000H	128K	E0000000H	512M
FFFC0000H	256K	C0000000H	1G
FFF80000H	512K	80000000H	2G
FFF00000H	1 M	00000000H	Address Window Closed.
FFE00000H	2 M		
FFC00000H	4 M		
FF800000H	8 M		

a. The smallest window supported is 4 Kbytes.

As an example, assume that FFF0 0004H is written to the Bridge Window Limit Register (BWLR). Scanning upwards starting at bit 12, bit 20 is the first one bit found. The binary-weighted value of this bit is 1,048,576, indicating a 1 Mbyte of memory window.

When programming the Base and Limit Registers for a memory window, the Base Address always needs to be aligned the size of the memory window set in a limit register. For a 1 Mbyte memory window, only bit 20 through bit 31 of the base address from the Bridge Base Address Register (BBAR) are relevant to the Bridge when decoding Memory Window.

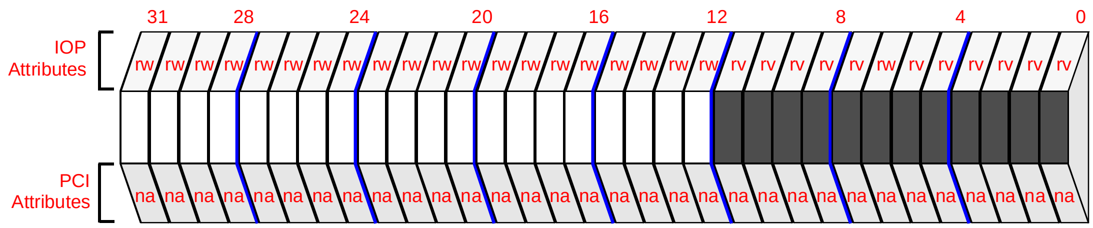
Warning: Care must be exercised when modifying the Bridge Base (BBAR) and Limit (BWLR) register pair during the time there is activity.

Warning: Since the smallest bridge memory window is 4 KBytes, care must be exercised when the bridge memory window is programmed to access memory-mapped registers that are located on the north internal bus. The 4-KByte bridge window overlaps other south internal bus memory-mapped registers. The user must not access the south internal bus memory-mapped registers that overlap the bridge memory window until the bridge memory window is re-mapped and does not overlap.

7.5.6 Bridge Window Base Address Register — BWBAR

The Bridge Base Address Register (BWBAR) defines the block of memory addresses where the Bridge Memory Window begins. The BWBAR is used in conjunction with the BWLR to form a memory window that is used by the Bridge to claim transactions on the South Internal Bus. The BWBAR defines the base address and describes the required memory block size; see Section 7.5.5, "Determining Block Sizes for Memory Windows" on page 504. The selected base address needs to be naturally aligned to the granularity of the memory block size. For instance, when a 64 Kbyte memory window size is selected, the base address needs to be 64 Kbyte address aligned (i.e., bits 15:12 of the base address are required to be 000b).

Table 339. Bridge Window Base Address Register — BWBAR



South XBG

Intel XScale® processor Local Bus Address Offset
+1780H

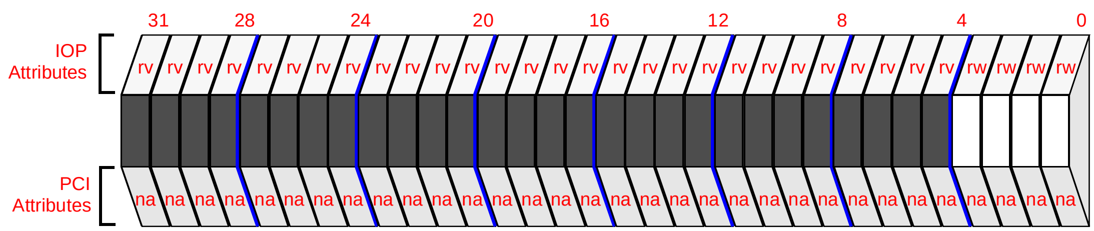
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:12	FFE0 0H	**Bridge Memory Window Base Address:** These bits define the actual location the Bridge responds to for accesses to the Bridge Memory Window on the South Internal Bus.
11:00	000H	Reserved.

7.5.7 Bridge Window Upper Base Address Register — BWUBAR

The Bridge Window Upper Base Address Register (BWUBAR) provides the upper 4 bits of the block of memory addresses where the Bridge Memory Window begins. The BWUBAR is used in conjunction with the BWBAR to form a 36-bit base address register. Refer to the Section 7.5.6, "Bridge Window Base Address Register — BWBAR" on page 505.

Table 340. Bridge Window Upper Base Address Register — BWUBAR



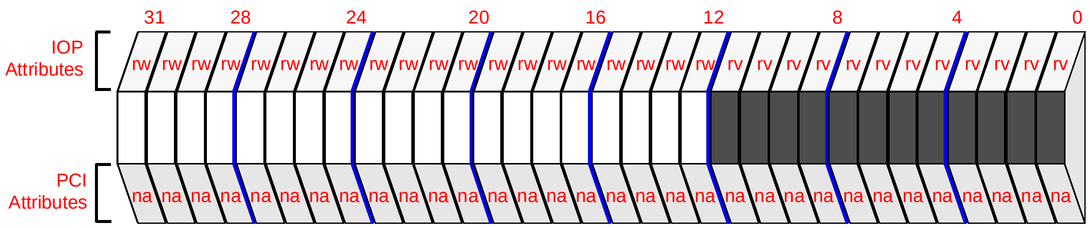
South XBG	Intel XScale® processor Local Bus Address Offset +1784H	Attribute Legend: RV = Reserved PR = Preserved RS = Read/Set	RW = Read/Write RC = Read Clear RO = Read Only NA = Not Accessible

Bit	Default	Description
31:04	0000 000H	Reserved.
03:00	0H	**Bridge Upper Memory Window Base Address:** These bits are the upper 4-bits of the 36-bit base address that defines the actual location the Bridge responds to for accesses to the Bridge Memory Window on the South Internal Bus.

7.5.8 Bridge Window Limit Register — BWLR

The 4138xx limit register's (BWLR) programmed value must be naturally aligned with the base address register's (BWBAR) programmed value. The limit register is used as a mask when the address decode for Bridge memory window is performed.

Table 341. Bridge Limit Register — BWLR



South XBG

Intel XScale® processor Local Bus Address Offset
+1788H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:12	FFF0 0H	**Bridge Memory Window Limit:** This value determines the memory range required for the Bridge Memory Window. Defaults to a 1MB Memory Window.
11:00	000H	Reserved

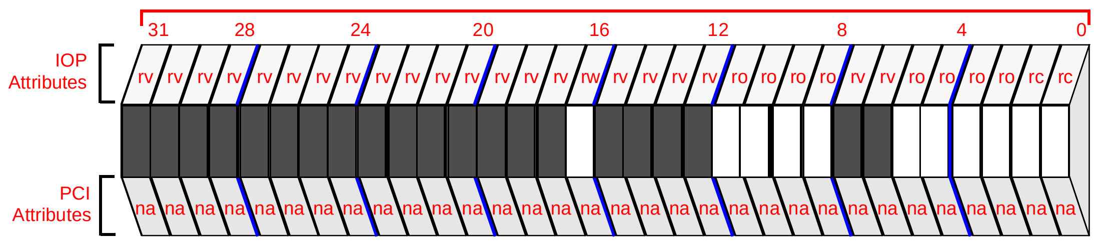
7.5.9 Bridge Error Control and Status Register — BECSR

The BECSR logs the type of error that the bridge encountered on either north or south interface. Only one error can be logged at a time.

The Bridge has two interrupt conditions: first Bridge error (BECSR[0]), and more than one Bridge error (BECSR[1]).

When the Bridge detects an error and BECSR[0] is cleared, the error is logged in BECSR, and BECSR[0] is set to 1. When BECSR[0] is not cleared, any additional Bridge errors are not logged and BECSR[1] is set.

Table 342. Bridge Error Control and Status Register — BECSR (Sheet 1 of 2)

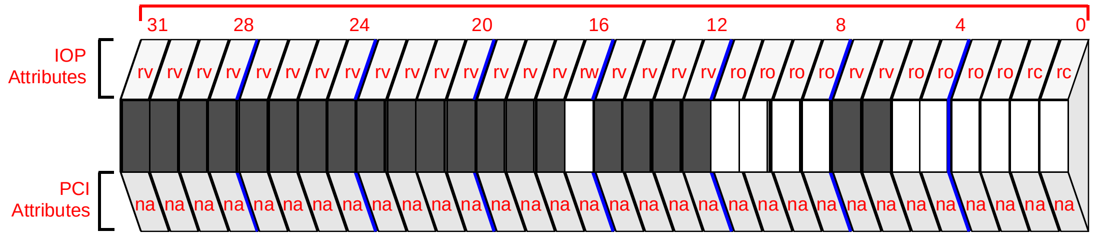


South XBG — internal bus address offset +178CH

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:17	0000H	Reserved.
16	0_2	Error Reporting Enable: This bit when set enables the internal bus bridge to signal an interrupt to the Intel XScale® processor when a South Internal Bus Error is detected. When cleared, an interrupt to the Intel XScale® processor is not signaled when a South Internal Bus Address Error is detected ***Note:*** This enable applies to all error types reported by the Internal Bus bridge through BECSR[3:2]
15:12	0H	Reserved.
11:08	0H	This field logs the Requester ID of the transaction that resulted in an error. **Requester ID** Core 0 1 Core 1 2 ATU-X 3 ATU-E 4 Reserved 5 N/A 6 LMU 7 N/A 8 SMBus 9
07:06	00_2	Reserved.
05	0_2	Interface Type: This bit indicates the Bridge interface that encountered the error condition. 0 = North Internal Bus Interface. 1 = South Internal Bus Interface
04	0_2	Command Type: Indicates the command type that encountered the error: 0 = Read. 1 = Write.

Table 342. Bridge Error Control and Status Register — BECSR (Sheet 2 of 2)



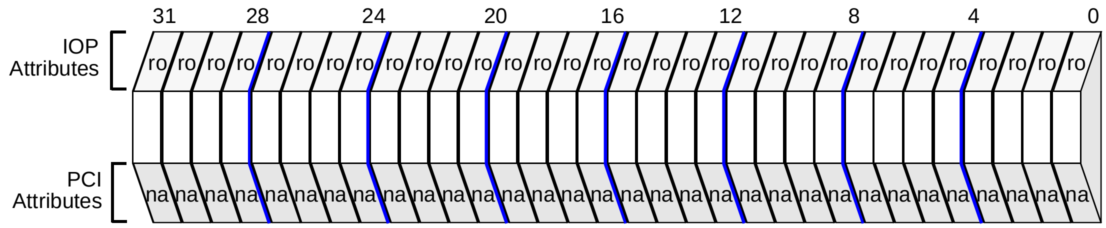
	internal bus address offset	Attribute Legend:	
South XBG	+178CH	RV = Reserved	RW = Read/Write
		PR = Preserved	RC = Read Clear
		RS = Read/Set	RO = Read Only
			NA = Not Accessible

Bit	Default	Description
03:02	00_2	Error Type: This field specifies the error type: 00 Master Abort 01 Bridge Master Address Error 10 Target Abort 11 Target Address Error ***Note:*** Bridge Master Address Error is an error condition that the Bridge detects when acting as a Master on either interface. Target Address Error is an error condition indicating that **any** target (including the bridge) on the south internal bus has detected an Address Error. ***Note:*** Since the Bridge can potentially detect two errors simultaneously — one error per interface, the Bridge only logs one of the errors as there is only one set of log registers.
01	0_2	**Error N Detected:** Indicates that the Bridge detected an error on either the north or south internal bus interface while BECSR[0] was set. 0 = No error detected 1 = Error detected
00	0_2	**Error Detected:** Indicates that the Bridge detected an error on either the north or south internal bus. 0 = No error detected 1 = Error detected and recorded in BAR and BUAR.

7.5.10 Bridge Error Address Register — BERAR

This register is responsible for logging the lower 32-bit of the 36-bit address of the bridge request that encountered the error. This register is used in conjunction with the BERUAR. Refer to Section 7.5.11, "Bridge Error Upper Address Register — BERUAR" on page 510.

Table 343. Bridge Error Address Register — BERAR



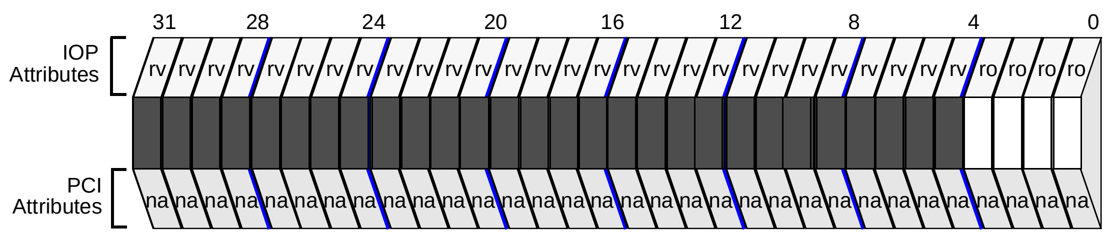
South XBG	Intel XScale® processor Local Bus Address Offset +1790H	Attribute Legend: RV = Reserved PR = Preserved RS = Read/Set	RW = Read/Write RC = Read Clear RO = Read Only NA = Not Accessible

Bit	Default	Description
31:00	0000 0000H	Error Address: Stores the lower 32-bit of the 36-bit address of the request that resulted in an error.

7.5.11 Bridge Error Upper Address Register — BERUAR

This register is responsible for logging the upper 4-bit address of the 36-bit address of the Bridge request that encountered the error. This register is used in conjunction with the BERAR (Section 7.5.10, "Bridge Error Address Register — BERAR" on page 510).

Table 344. Bridge Error Upper Address Register — BERUAR



South XBG	Intel XScale® processor Local Bus Address Offset +1794H	Attribute Legend: RV = Reserved PR = Preserved RS = Read/Set	RW = Read/Write RC = Read Clear RO = Read Only NA = Not Accessible

Bit	Default	Description
31:04	0000 000H	**Reserved.**
03:00	0000_2	Error Address: Stores the upper 4 bits of the 36-bit address of the request that resulted in an error.

8.0 SRAM Memory Controller

This chapter describes the integrated SRAM Memory Controller Unit (SMCU). The operating modes, initialization, and implementation are detailed in this chapter.

8.1 Overview

The Intel® 413808 and 413812 I/O Controllers in TPER Mode (4138xx integrates a high performance, multi-ported SRAM Memory Controller to provide access to the on-chip 1.0 MByte SRAM Memory. The SRAM Memory Controller supports:

- 1.0 MByte SRAM Memory
- Dedicated port for Intel XScale® processor to the SRAM
- Optimized core processor data processing 32-bit region.
- Single-bit error correction, multi-bit detection support (ECC)
- 256-bit wide SRAM Memory Interface
- 128-bit wide port with data parity protection
- ECC supported on 32-bit data width

The SRAM interface provides a direct connection to a high bandwidth and reliable memory subsystem. An 7-bit Error Correction Code (ECC) across every 32-bit word improves system reliability. The ECC is stored into the SRAM array along with the 32-bit data and is checked when the data is read. If the code is incorrect, the SMCU corrects the data (if possible) before reaching the initiator of the read. User-defined fault correction software is responsible for scrubbing the memory array.

- The SMCU responds to the north internal bus, memory accesses within its programmed address range and issues the memory request to the SRAM interface.
- The SMCU contains transaction queues for the north internal bus port enabling pipelining of transactions to the SRAM for maximum performance.
- ECC implemented on 32-bit data for higher core write performance core by avoiding Read-Modify-Write (RMW) operation to the SRAM.

8.2 Glossary

This section lists commonly used terms throughout this chapter:

Table 345. Commonly Used Terms

Term	Definition
Scrubbing	Once an error is detected within the memory array, the SMCU must correct the error (if possible) while delivering the data to the initiator. Correcting the memory location is referred to as "scrubbing the array." The SMCU relies on software to scrub any errors.
Syndrome	A syndrome is a value which indicates an error in the data read from the memory array. The SMCU computes the syndrome with every memory read. Decoding the syndrome indicates: the bit in error for a single-bit error, or a multi-bit error. Table 346 defines the syndrome decoding.

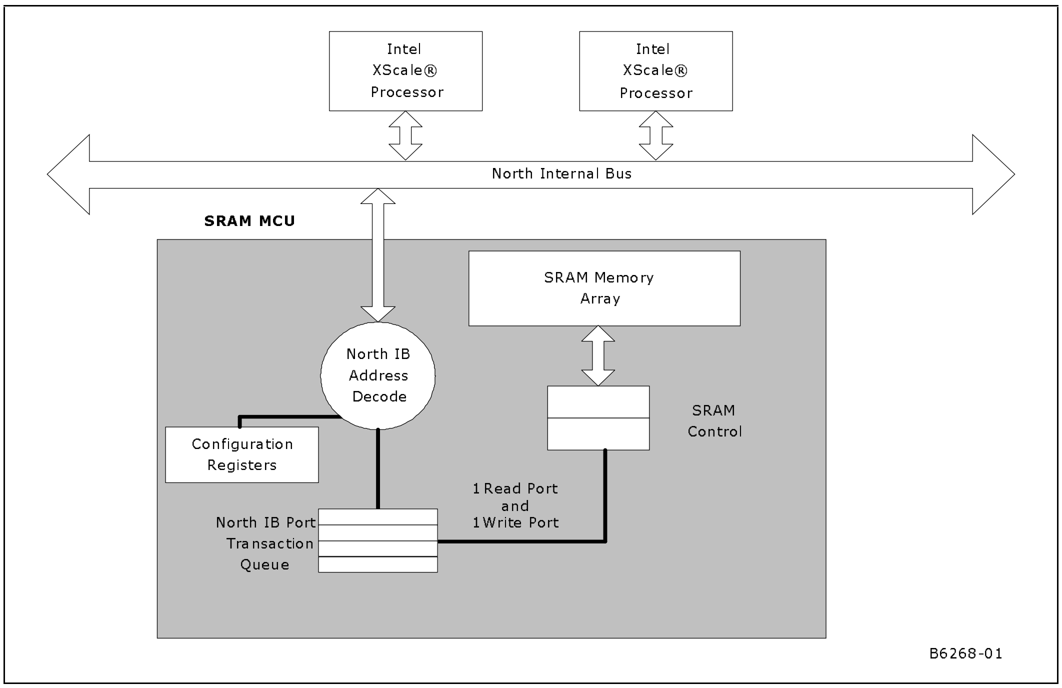
8.3 Theory of Operation

The 4138xx SRAM memory controller translates transactions from the north internal bus into the protocol supported by the SRAM memory subsystem.

8.3.1 Functional Block

The SRAM memory controller logically comprises the blocks illustrated in Figure 56. The SMCU supports a separate read and write port.

Figure 56. Intel® 413808 and 413812 I/O Controllers in TPER Mode SRAM Memory Controller Block Diagram



The SMCU provides ports for direct SRAM access. Each device connects to the SMCU using a separate read port and write port. Each port provides a 128-bit data path. The ports are described in the next sub-sections:

8.3.1.1 North Internal Bus Ports

The North Internal Bus Port provides the 4138xx processor core access to the SRAM Memory Controller. This North Internal Bus Port allows core transactions targeting the SRAM via the North Internal Bus bus to pass directly to the SRAM.

8.3.1.2 Address Decode Blocks

Address Decode is performed for transactions from input North Internal Bus port to determine if the SMCU should claim the transaction. The north internal bus port claims transactions targeting the SRAM memory array space, and transactions targeting the memory-mapped registers.

8.3.1.2.1 SRAM Memory Array Space

The SRAM Memory Array Space can be accessed from the north internal bus port and the SRAM memory space is defined with the SRAM Base Address Registers (SBAR and SBUAR).

8.3.1.2.2 Memory-Mapped Register Space

The SMCU PMMR memory space offset is +1500H to +157FH. The registers are detailed in Section 8.6, “Register Definitions” on page 535.

The Memory-mapped registers are only accessible from the north internal bus port.

8.3.1.2.3 North Internal Bus Port Address Decode

North internal bus transactions are decoded to determine if they address the SRAM Memory space or the memory-mapped registers. If the transaction addresses the SRAM Memory Space, the transaction is queued in the north internal bus port transaction queue.

8.3.1.3 Memory Transaction Queues

There are one set of transaction queues for transactions which address the SRAM Memory Space from the north internal bus.The transaction queues are located in each respective unit.

8.3.1.3.1 North Internal Bus Port Transaction Queue (NIBPTQ)

The NIBPTQ stores memory transactions from the north internal bus. The NIBPTQ supports 16 read transactions, each with up to 32 bytes buffer. The NIBPTQ also supports 16 posted write transactions up to 32 bytes each.

8.3.1.4 Configuration Registers

The Configuration Registers block contains all of the memory-mapped registers listed in Section 8.6, “Register Definitions” on page 535. These registers define the memory subsystem connected to the 4138xx. The status registers indicate the current SMCU status.

8.3.1.5 SRAM Control Block

The SRAM Control Block contains all functionality to process the SRAM data accesses per the transactions issued by the SMARB. To process a transaction the SRAM Control Block employs several sub-blocks. The sub-blocks include the SRAM State Machine and Pipeline Queues, and Error Correction Logic.

8.3.1.5.1 SRAM State Machine and Pipeline Queues

Since the SMCU generates error correction codes based on the data, the SMCU is a pipelined architecture. Pipelining also ensures acceptable AC timings to the memory interfaces. The SRAM state machine pipelines SRAM memory operations for several clocks.

8.3.1.5.2 Error Correction Logic

The Error Correction Logic generates the ECC code for SRAM reads and writes. For reads, this logic compares the ECC codes read with the locally generated ECC code. If the codes mismatch then the Error Correction Logic determines the error type. For a single-bit error, this block determines which bit is in error and corrects the error. For a single-bit or multi-bit error, the Error Correction Logic logs the error in ELOG0 and ELOG1. See Section 8.3.3, "Error Correction and Detection" on page 519 for more details.

8.3.1.6 North Internal Bus Port Transaction Ordering

Since requests from the north internal bus port are queued in the NIBPTQ, the port needs to maintain order of requests addressing the SRAM. Coherency between the north internal bus port and the other ports are maintained by the SMCU as described in "SMCU Port Coherency" below.

8.3.1.7 SMCU Port Coherency

With the queueing of SRAM transactions in multiple ports, coherency of memory must be maintained. The SMARB maintains memory coherency by ensuring that all writes to a given memory address are completed before any read to the same address is processed. This address comparison is done with a 1 KByte granularity.

The current read transaction is compared to all pending write transactions in the NIBPTQ transaction queue. If a write transaction is pending for the same memory location (1 KByte granularity), the write is allowed to complete first, before the read transaction can be processed. Also, to maintain ordering rules, all write transactions preceding the 'incoherent write' are also processed.

8.3.2 SRAM Memory Interface Support

The 4138xx memory controller supports 1.0 Mbytes of on-chip SRAM. The SMCU supports a 256-bit data bus width memory with ECC. The SMCU supports 7-bit ECC on every 32-bit data quantity, providing higher performance when the core processor is processing data by eliminating any RMW cycle required for 4-Byte store ECC generation.

The SMCU supports seamless read/write bursting of long data streams.

8.3.2.1 SRAM Initialization

Initialization software should initialize the entire SRAM memory array in order to have the correct ECC values for each ECC location. Refer to Section 8.3.3, "Error Correction and Detection" on page 519) for more details. Reading from an uninitialized memory location will result in an ECC error. By default data parity checking is disabled, firmware must enable data parity checking if required. Refer to Section 357, "SRAM Parity Control and Status Register — SPARCSR" on page 542.

8.3.2.2 SRAM Read Sequence

Read transactions require ECC codes to be calculated and compared with the ECC returned by the SRAM array. The following steps describe the read sequence.

1. Each of the SMCU inbound memory transaction ports decodes the address to determine if the transaction should be claimed.
   — If the address falls in the SRAM address range indicated by the SRAMBAR and SRAMUBAR the SMCU claims the transaction.
2. Once the SMARB selects the highest priority port transaction, it forwards the transaction to the SRAM control block.
3. Upon receipt of the data, the SRAM Control Block calculates the ECC code from the data and compares it with the ECC returned by the SRAM array. Section 8.3.3, "Error Correction and Detection" on page 519 explains the ECC algorithm in more detail.
4. Assuming the calculated ECC matches the read ECC, the SRAM Control Block drives the data back to the corresponding memory port.
   — For each burst read issued, the memory controller increments the address by sixteen.

The SMCU continues to return data to the corresponding memory port based on the byte count of the transaction.

8.3.2.3 SRAM Write Sequence

Write transactions require ECC codes to be generated and stored in the SRAM array with the data being written. The behavior is different depending on the size of the data being written. Section 8.3.3, "Error Correction and Detection" on page 519 explains the ECC algorithm in more detail.

1. Each of the SMCU inbound memory transaction ports decodes the address to determine if the transaction should be claimed.
   — If the address falls in the SRAM address range indicated by the SRAMBAR and SRAMUBAR the SMCU claims the transaction.
2. Once the SMARB selects the highest priority port transaction, it forwards the transaction to the SRAM control block.
   — The ECC logic generates the ECC code for the data to be written.
   — The SRAM Control Block drives the new data to the memory array each cycle until the transaction is completed with the byte count expiring.
   — For each burst issued, the SRAM Control Block increments the address by sixteen.
   — When the data to write is less than an aligned DWORD, the SRAM Control Block will perform a read-modify-write of the entire 4 byte aligned DWORD and incorporate the new data while regenerating ECC.

8.3.3 Error Correction and Detection

The SMCU is capable of correcting any single bit errors and detecting any double bit errors in the 4138xx SRAM memory subsystem. ECC enhances the reliability of a memory subsystem by correcting single bit errors caused by electrical noise or occasional alpha particle hits on the SRAM memory array.

Similar to parity, which simply detects single bit errors, error correction requires an additional 7-bit code word for the 32-bit datum. The 4138xx implements a 256-bit data path to the SRAM array, but a 7-bit error correction code per every 32-bit datum. For example, **SCB0[6:0]** for **DQ[31:0]**, **SCB1[6:0]** for **DQ[63:32]**, **SCB2[6:0]** for **DQ[95:64]** and so on, resulting in a 312-bit wide memory subsystem. During SRAM read cycles, the SRAM Control Block detects single bit errors and corrects the data prior to returning the data to the respective memory transaction queue. SRAM write cycles generate the ECC and sends it with the data to the memories.

Scrubbing is the process of correcting an error in the memory array. The chance of an unrecoverable multi-bit error increases if the software does not correct a single-bit error in the array. For the 4138xx, scrubbing is handled by software. If error reporting is enabled, the SMCU logs the error type in SELOG and the address in SECAR and SECUAR when an error occurs.

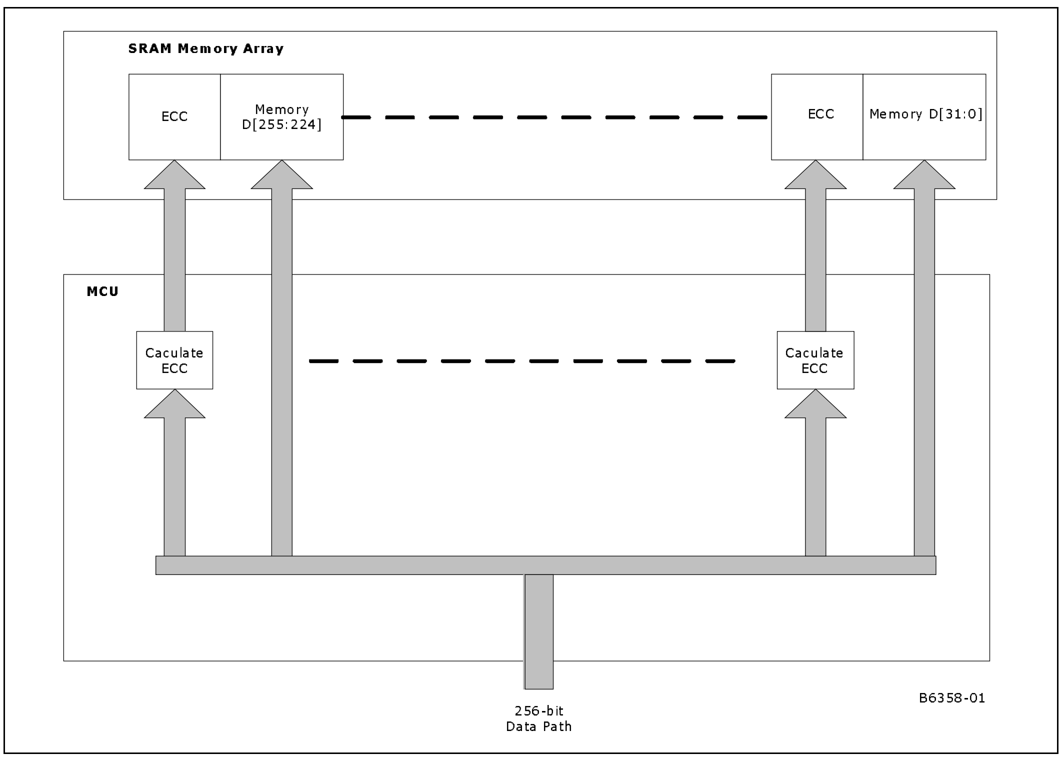
8.3.3.1 ECC Generation

For write operations, the SMCU generates the error correction code which is written along with the data. This section describes the operation of the SRAM Control Block for ECC generation on 32-bit data of the 256-bit wide memory. The SMCU will generate 7-bit wide ECC on every 32-bit data. The algorithm for a write transaction is:

```
if data to write is 32-bit wide

    Generate the ECC_with the G-matrix

    Write the new data and ECC

else {Partial Write}

    Read entire 32-bit data word from memory

    Merge the new data portion with the data from memory

    Generate the new ECC with the G-matrix

    Write new data and ECC
```

Figure 57 shows how the data logically flows through the ECC hardware for a write transaction.

Figure 57. ECC Write Flow



The G-Matrix in Figure 57 generates the ECC. The data to be written is input to the matrix and the output is the ECC code. Each row of the G-Matrix indicates which data bits' codes of **DQ[31:0]**, **DQ[63:32]**, **DQ[95:64]**, or **DQ[255:224]** needs to be XORed together to form the ECC code. The resulting ECC bits are driven on **SCB0[6:0]**, **SCB1[6:0]**, **SCB2[6:0]**, and **SCB7[6:0]** respectively.

8.3.3.2 ECC Generation for Partial Writes

Figure 58. Intel® 413808 and 413812 I/O Controllers G-Matrix (generates the ECC)

Data Bit	ECC Code	ECC Check Bits (E[6:0]							Data Bit	ECC Code	ECC Check Bits (E[6:0]						
		E6	E5	E4	E3	E2	E1	E0			E6	E5	E4	E3	E2	E1	E0
D31	2CH		X		X	X			**D15**	45H	X				X		X
D30	4AH	X			X		X		**D14**	51H	X		X				X
D29	1AH			X	X		X		**D13**	43H	X					X	X
D28	29H		X		X			X	**D12**	61H	X	X					X
D27	5EH	X		X	X	X	X		**D11**	25H		X			X		X
D26	6BH	X	X		X		X	X	**D10**	31H		X	X				X
D25	2AH		X		X		X		**D9**	13H			X			X	X
D24	3BH		X	X	X		X	X	**D8**	5BH	X		X	X		X	X
D23	64H	X	X			X			**D7**	46H	X				X	X	
D22	26H		X			X	X		**D6**	32H		X	X			X	
D21	3EH		X	X	X	X	X		**D5**	23H		X				X	X
D20	15H			X		X		X	**D4**	68H	X	X		X			
D19	34H		X	X		X			**D3**	4CH	X			X	X		
D18	54H	X		X		X			**D2**	52H	X		X			X	
D17	37H		X	X		X	X	X	**D1**	62H	X	X				X	
D16	6EH	X	X		X	X	X		**D0**	49H	X			X			X

If the memory transaction writes less than 32-bit data, then the SRAM Control Block translates the write transaction into a read-modify-write transaction. For a partial write, the SRAM Control Block calculates the ECC for the modified datum and writes it back. So, if an external unit issues a write cycle with partial data to an SMCU port, the SMCU:

1. Issues a 32-bit read.
2. Modifies the value with the new portion to be written.
3. Calculates the ECC on the modified value.
4. Writes the 32-bit value and ECC.

Note: If the SMCU detects a single-bit error during the read, it is corrected BEFORE being merged with the write data so the corrected data is written back to the array. If a multi-bit error is detected, the SMCU causes an interrupt to the core by writing to the MCISR. The memory location is overwritten by the SMCU with the error data but valid ECC, making the contents of memory invalid. For more details on how the SMCU handles error conditions, see Section 8.4, "ECC Interrupts/Error Conditions" on page 531.

8.3.3.3 ECC Checking

The ECC logic uses the following ECC read algorithm. This algorithm corrects the data before it's driven onto the internal bus. The ECC algorithm for a read transaction is:

```
Read 32-bit data and 7-bit ECC

Compute the syndrome by passing the 32-bit data through the G-Matrix and XORing the
7-bit result with the 7-bit ECC

if the syndrome <> 0 {ECC Error}

    Look up in H-matrix to determine error type

    Register the address where the error occurred

    if error is correctable {single bit}

        if single-bit error correction is enabled

            Correct data

            Send corrected data to internal bus

        if single bit error reporting is enabled

            Interrupt core for software scrubbing

    else {uncorrectable}

        if the read cycle is not part of a RMW cycle {read}

            Target-Abort the Internal Bus read transaction.

        else {write requiring RMW}

            Merge the new data portion with the read data from memory

            Generate the new ECC with the G-matrix

            Write new data and ECC

        if multi-bit error reporting is enabled

                Interrupt the core for uncorrectable error
```

When the SMCU reads the ECC from the memory subsystem, it is compared (XORed) with an ECC result that the SMCU generates from the data read from the memory. The resulting value of the XOR operation is called the syndrome. Table 346 shows how the SMCU decodes the syndrome for SRAM read cycles.

Table 346. Syndrome Decoding

Error Type	Symptom
None	The syndrome is 0000 0000.
Single-Bit	Use the H-Matrix in Figure 60 to determine which bit the SMCU will invert to fix the error.
Multi-Bit	If the Syndrome does not match an 7-bit value in the H-matrix, the error is uncorrectable

Figure 59 shows how the data flows through the ECC hardware for a read transaction.

Figure 59. ECC Read Data Flow

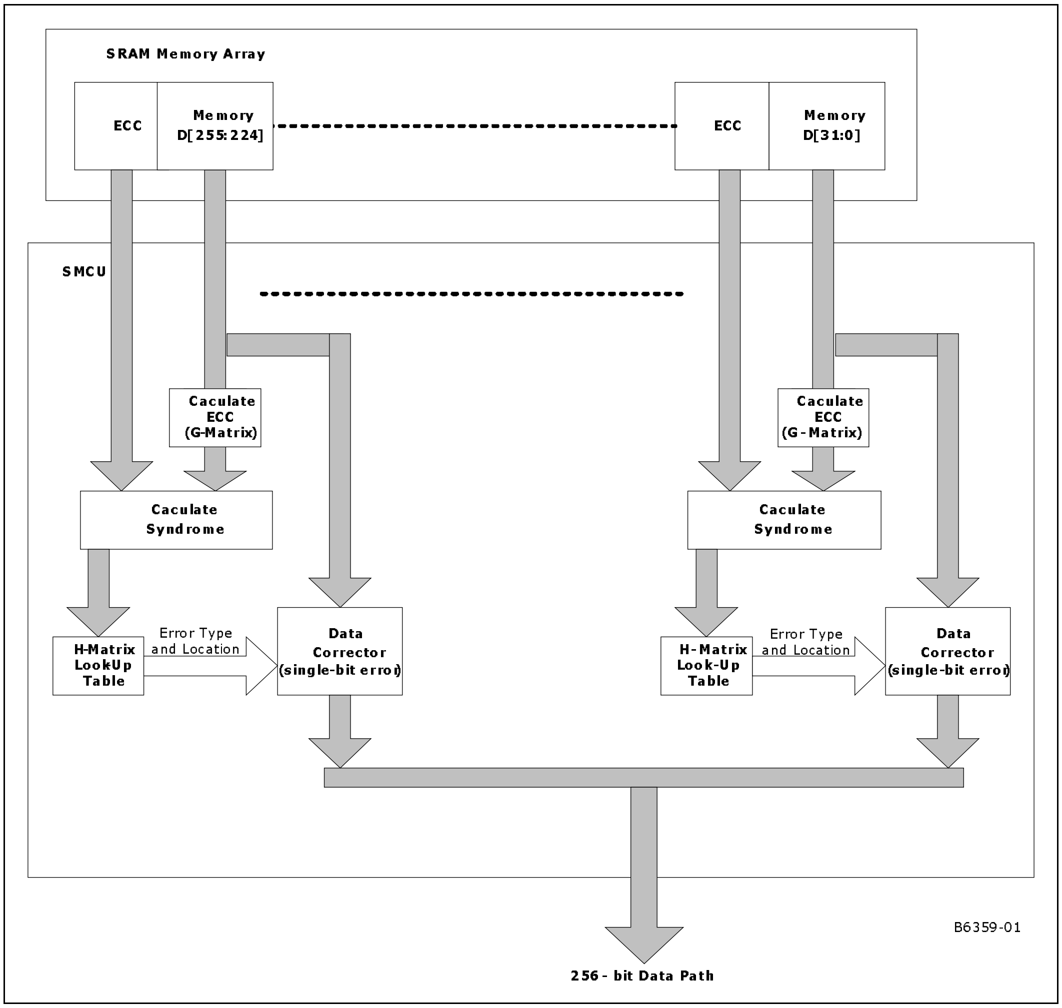


Figure 60 illustrates the H-Matrix used for decoding the syndrome. For single-bit errors, the H-Matrix indicates the bit that contains the error and consequently, which bit to fix.

Figure 60. 4138xx H-Matrix (indicates the single-bit error location)

Data Bit	Syndrome	ECC Check Bits (E[6:0]							Data Bit	Syndrome	ECC Check Bits (E[6:0]						
		E 6	E 5	E 4	E 3	E 2	E 1	E 0			E 6	E 5	E 4	E 3	E 2	E 1	E 0
E6	40H	X							**D15**	45H	X				X		X
E5	20H		X						**D14**	51H	X		X				X
E4	10H			X					**D13**	43H	X					X	X
E3	08H				X				**D12**	61H	X	X					X
E2	04H					X			**D11**	25H		X			X		X
E1	02H						X		**D10**	31H		X	X				X
E0	01H							X	**D9**	13H			X			X	X
D31	2CH		X		X	X			**D8**	5BH	X		X	X		X	X
D30	4AH	X			X		X		**D7**	46H	X				X	X	
D29	1AH			X	X		X		**D6**	32H		X	X			X	
D28	29H		X		X			X	**D5**	23H		X				X	X
D27	5EH	X		X	X	X	X		**D4**	68H	X	X		X			
D26	6BH	X	X		X		X	X	**D3**	4CH	X			X	X		
D25	2AH		X		X		X		**D2**	52H	X		X			X	
D24	3BH		X	X	X		X	X	**D1**	62H	X	X				X	
D23	64H	X	X			X			**D0**	49H	X			X			X
D22	26H		X			X	X										
D21	3EH		X	X	X	X	X										
D20	15H			X		X		X									
D19	34H		X	X		X											
D18	54H	X		X		X											
D17	37H		X	X		X	X	X									
D16	6EH	X	X		X	X	X										

Referring to Figure 59, the syndrome bits are created by XORing the ECC code bits as indicated by the appropriate row of the G-Matrix in Figure 58 with the corresponding ECC bits read from memory. For example, the SMCU derives syndrome bit 0 by XORing ECC code associated with data bits 0, 2, 5, 8…15, 17, 20, 22, 24, 26, 28 and ECC bit 0 (physically read on **SCB[0]**). The SMCU performs seven such XOR operations (one per syndrome bit).

If decoding the syndrome indicates multi-bit error (see Table 346), the transaction results in a target-abort for Internal Bus transactions, or a multi-bit error in the BIU for Core transactions. If an internal bus master detects a target-abort, the master asserts an interrupt to the core. Write cycles are posted to the memory transaction queues, and already completed to the initiating master. For write cycles with a multi-bit error and ECC Error reporting is enabled, the SMCU reports the interrupt in the MCISR and interrupts the core.

If the syndrome indicates a single-bit error and single-bit error correction is enabled, the H-Matrix is used to determine the bit in error (see Figure 60). For example, if the syndrome was 100 1001, the error is with bit 0 of **DQ[31:0]**. The SMCU inverts bit 0 before driving the data on internal bus.

If error reporting is enabled in the SECCR and the SMCU detects a single-bit or multi-bit error, the SMCU stores the address in SECAR and the syndrome in SELOG. Then, the SMCU signals an interrupt to the core. Software decides how to proceed through an interrupt handler. By registering the address in SECAR, software can identify the faulty location.

For details about the SMCU error conditions and how the MMR registers are affected, refer to Section 8.4, "ECC Interrupts/Error Conditions" on page 531.

8.3.3.4 Scrubbing

Fixing the data error in memory is called scrubbing. The 4138xx relies on Intel XScale® processor software to perform the scrubbing. When the SMCU detects an error during a read, the SMCU logs the address where the error occurred and interrupts the core. The core decides how to fix the error through an interrupt handler. Software could decide to perform the scrubbing on:

- the data location that failed
- the entire row of the data that failed
- the entire memory

For single-bit errors reported on a write transaction scrubbing is not required, as the SMCU will have scrubbed the data during the RMW operation. For single-bit errors, the error is fixed by reading the location that failed and writing back the data after the ECC hardware fixed it. The scrubbing routine should read the 32-bit data using a `ld` instruction and write the data back with a `st` instruction. Software should isolate activity on the memory location to guarantee animosity.

Note: If the scrubbing routine reads the failed location in order to fix the single-bit error, a second error will be reported. Therefore, software should disable single-bit ECC reporting (SECCR[0]) during the scrubbing routine. Multi-bit errors cannot be fixed by the H-Matrix.

8.3.3.4.1 ECC Example Using the H-Matrix

Assume the core writes 9ABC DEF0H to the SRAM memory space on **DQ[31:0]**.

Using the G-Matrix in Figure 58, the SRAM Control Block creates each check bit by XORing the appropriate bits in the row. Using 9ABC DEF0H, the ECC code generated is 11H. This code is written with the data to the SRAM memory on **SCB[7:0]**.

Assume that bit 17 was corrupted in the array. Therefore, the bit has been inverted from 0 to 1.

At some later point in time, the core wishes to read from the same address. The core issues a read transaction.to the SRAM memory. Upon the receipt of 9AB**E** DEF0H on **DQ[31:0]**, the SRAM Control Block calculates the syndrome with the G-Matrix in Figure 58. The SRAM Control Block calculates a syndrome of 37H.

Note: During a memory write, ECC code is created by XORing the appropriate data bits' ECC codes indicated by the G-Matrix. The syndrome is created during a memory read by XORing the 7-bit value generated by XORing appropriate data bits' ECC codes indicated by the G-Matrix with the check bits (**SCB[7:0]**).

Referring to Table 346, if the syndrome is non-zero and matches a value in the H-Matrix, there is a single-bit error that can be fixed. A syndrome of 37H matches a value in the H-Matrix (see Figure 60) which indicates that bit 17 has an error. The SRAM Control Block inverts bit 17 prior to returning the corrected data on the internal bus. The SMCU returns 9ABC DEF0H on the internal bus.

Assuming this was the first error the SMCU records the address where the error occurred in SECAR and error type in SELOG. If error reporting is enabled in the SECCR, the SMCU writes a 1 to SMCISR[0] which generates an interrupt to the core. A software interrupt handler scrubs the array and fixes the error in bit 17. Unless more errors occur, future reads from this location do not result in an error.

8.3.3.5 ECC Disabled

If software disables ECC, the SMCU does generate the ECC byte for writes, but does not check the ECC byte for reads.

8.3.3.6 ECC Testing

Section 8.3.3.4, "Scrubbing" on page 526 explains how software is responsible for correcting an error in the memory array once it has been detected by the ECC logic. The SMCU implements the SECTST register providing the programmer the ability to test error handling software. For write transactions, the SECTST register value is XORed with the generated ECC. This inverts the bits where the mask is set prior to writing the ECC to memory. When the SMCU reads the address later, the ECC mismatches and the error condition occurs (see Section 8.4, "ECC Interrupts/Error Conditions" on page 531).

8.3.4 Byte Parity Checking and Generation

All the direct memory ports of the SMCU supports byte-wise parity on the data bus.

For write requests made to an SMCU port, the direct memory port interface performs the following tasks when receiving data:

- checks for data parity on the incoming data (parity error is logged if detected)
- generates ECC on the data

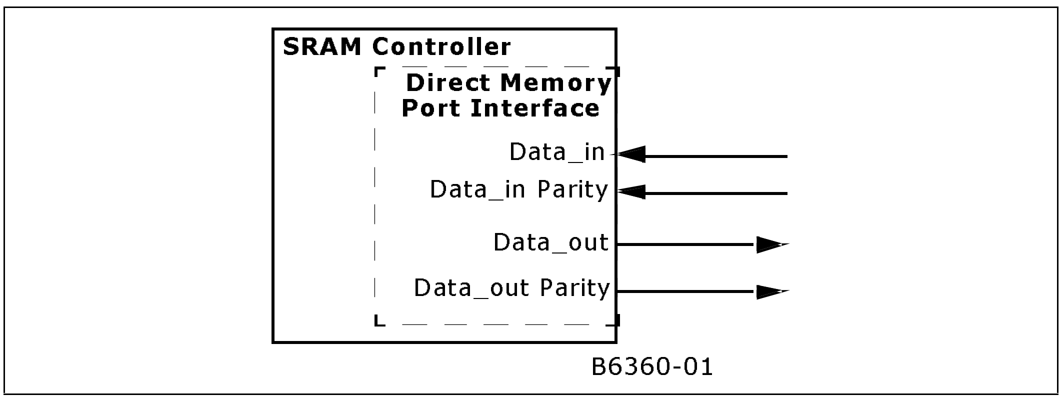
For the memory ports that are connected to the south internal bus, the write data parity is received from the south internal bus as driven by the initiator of the data on the south internal bus. For example, south internal bus initiators generate data parity. Since the north internal bus does not support parity, data parity is generated by the north internal bus port as the data enters the port and before it is written to the data queue.

For read requests made to an SMCU port, the direct memory port interface performs the following tasks before delivering data:

- checks for ECC on the data read from memory
- generates data parity on the data before delivering it on the port

For the memory ports that are connected to the south internal bus, the read data parity that are generated by the SMCU are directly driven onto the south internal bus to the requester. The data parity is then verified by the requester. Since the north internal bus does not support parity, the read data parity that are generated by the SMCU are verified by the north internal bus port when the data is read out of the data queue and before being delivered onto the north internal bus. If the north internal bus port detects a parity error while reading the data out of the data queue, the north internal bus port will return a target abort.

Figure 61. Logical Data Access Paths with Parity Protection



8.3.4.1 Parity Generation

The direct memory port interface of the SMCU only generates data parity before delivering data onto the port. After the requested data is read from memory and ECC has been verified, the direct memory port interface generates even data parity before delivering the data on the direct memory port. Table 347 lists the data bytes that are used for data parity calculation. The parity bits are calculated by bit XORing the data bits as shown in Table 347. As an example, the parity calculation for the lowest order byte of the data bus D[7:0] is calculated as follows:

Note: The direct memory port does not support address parity.

Equation 15. D_PARITY0 = D[0] XOR D[1] XOR D[2] XOR D[3] XOR D[4] XOR D[5] XOR D[6] XOR D[7] XOR BE[0]

Table 347. Data Parity Checking/Generation

Data Parity Bit	Data Byte	Byte Enable
D_PARITY15	D[127:120]	BE[15]
D_PARITY14	D[119:112]	BE[14]
D_PARITY13	D[111:104]	BE[13]
D_PARITY12	D[103:96]	BE[12]
D_PARITY11	D[95:88]	BE[11]
D_PARITY10	D[87:80]	BE[10]
D_PARITY9	D[79:72]	BE[9]
D_PARITY8	D[71:64]	BE[8]
D_PARITY7	D[63:56]	BE[7]
D_PARITY6	D[55:48]	BE[6]
D_PARITY5	D[47:40]	BE[5]
D_PARITY4	D[39:32]	BE[4]
D_PARITY3	D[31:24]	BE[3]
D_PARITY2	D[23:16]	BE[2]
D_PARITY1	D[15:8]	BE[1]
D_PARITY0	D[7:0]	BE[0]

8.3.4.2 Parity Checking

The direct memory port interface of the SMCU only checks data parity while receiving data. The direct memory port interface verifies data parity on the port interface, and then generates ECC before writing the data to memory. Table 347 lists the data bits that are used for the parity calculation. The parity bits are calculated by bit XORing the data bits as shown in Table 347. As an example, the parity calculation for the lowest order byte of the data bus D[7:0] is calculated as follows:

Note: The direct memory port does not support address parity.

Equation 16. DATA_PARITY_RESULT = D_PARITY0 XOR D[0] XOR D[1] XOR D[2] XOR D[3] XOR D[4] XOR D[5] XOR D[6] XOR D[7] XOR BE[0]

A non-zero result from the above operation indicates a parity error.

The parity logic uses the following algorithm, and this algorithm logs the error if an error is detected.

```
check data parity

    if data parity result is good

        done

    else {error}

        create an error log

        Interrupt the core (if enabled)
```

8.3.4.3 Parity Disabled

If software disables parity, the SMCU would generate data parity as explained above, but would not check and report data parity on the interface.

8.3.4.4 Parity Testing

The System Controller provides the ability for the programmer to test error handling software by forcing address or data parity error. Refer to the Chapter 6.0, “System Controller (SC) and Internal Bus Bridge” for more details.

8.4 ECC Interrupts/Error Conditions

The SMCU has two ECC conditions which require intervention from the Intel XScale® processor. If a single-bit error is detected during a read cycle, the SMCU can correct the data returned but software needs to fix the error in the memory array. If a multi-bit error is detected, the core decides how to handle the condition. For all ECC errors, the SMCU records the requester of the transaction resulting in the error in SELOG[23:16] and interrupts the core.

If the SMCU detects an ECC error during a read or write cycle, SMCISR[0] is set to 1. Whenever the SMCU toggles the SMCISR[0] bit from 0 to 1, an interrupt is generated to the core.

Table 348 shows how the SMCU responds to error conditions.

Table 348. SMCU Error Response

Error Type	SMCU Action
Single-Bit during a read or write	Fix Error (if ECC error correction enabled in the SECCR)
Multi-bit during a read	Target Abort the Internal Bus transaction
Multi-bit during a write	New ECC is generated with bad data and written to SRAM array. Data location is no longer valid.

Note: If ECC reporting is enabled with SECCR[1] or SECCR[0] and an ECC error occurs, SMCISR[0] is set and SELOG/SECAR/SECUAR logs the error in addition to Table 348 actions.

8.4.1 Single-Bit Error Detection

When enabled, the SMCU interrupts the core when the ECC logic detects a single-bit error by setting the appropriate bit in the MCISR register. The core knows the interrupt was caused by a single-bit error by polling the SELOG register. The SRAM Control Block ensures that correct data is returned but the interrupt handler is responsible for scrubbing the error in the array (refer to Section 8.3.3.4, "Scrubbing" on page 526).

An example flow for a single-bit error with error detection and reporting enabled is:

- A single-bit ECC error is detected on the data bus by the SMCU.
- SMCU fixes the error prior to returning the data.
- SMCU clears SELOG[8] indicating a single-bit error.
- SMCU records requester of transaction that resulted in an error in SELOG[23:16]
- The SMCU loads SELOG[7:0] with the syndrome that indicated the error.
- The SMCU loads SECAR[31:2] and SECUAR with address where the error occurred.
- Since the core needs to scrub the error in the array, the SMCU sets MCISR[0] to 1 (assuming it is not already set).
  — Setting any bit in the MCISR causes an interrupt to the core.
- Software polls interrupt status register. Bit 0 = 1 indicates first error has occurred.
- Software polls SELOG, SECAR and SECUAR and scrubs the error at the location specified by SECAR and SECUAR.
- Software writes a 1 to SMCISR[0] thereby clearing it.

If software does not perform error scrubbing, the probability of an unrecoverable multi-bit error increases for the memory location containing the single-bit error.

SECAR, SECUAR and SELOG remain registered until software explicitly clears them.

If a second error occurs before software clears the first by resetting SMCISR[0], the error is not logged but the SMCU carries out the action described in Table 348.

8.4.2 Multi-bit Error Detection

If a multi-bit error occurs during a read or write transaction and error reporting is enabled, the SMCU sets SMCISR[0] which asserts an interrupt to the core. Upon receiving an interrupt, the core knows the interrupt was caused by a multi-bit error by polling the SELOG registers.

When SMCU detects a multi-bit error during a read cycle and ECC calculation is enabled in the SECCR, the SMCU target aborts the transaction, indicating to the MCU port that an unrecoverable error has been detected. When the SMCU port is the north internal bus port, the north internal bus port notifies the internal bus initiator of a multi-bit error by returning a target abort. The SMCU records the error type in SELOG and the address in SECAR and SECUAR.

When SMCU detects a multi-bit error during a write[15] cycle and error reporting is enabled in the SECCR, the SMCU records the first multi-bit error by programming SELOG, SECAR and SECUAR. The SMCU generates new ECC with the data before sending it on **DQ[31:0]** so the contents of memory after the read-modify-write cycle will be corrupted with correct ECC.

If a second error occurs before software clears the first by resetting SMCISR[0], the error is not logged but the SMCU carries out the action described in Table 348.

It is the interrupt handler responsibility to decide how to handle this error condition and clear the SMCISR.

15.Any error condition during a write cycle actually occurs while performing the read portion of a read-modify-write on a partial write. See Section 8.3.3.1, "ECC Generation" on page 520 for details.

8.5 Parity Interrupts/Error Conditions

If a data parity error is detected on any of the SMCU ports and parity is enabled, the SMCU records the requesting port that detected the parity error in the SPCSR[19:16] and interrupts the core. Refer to the Section 8.6.8, "SRAM Parity Control and Status Register — SPARCSR" on page 542

When the SMCU detects a parity error, the SMCISR[8] is set to 1. Whenever the SMCU toggles the SMCUSR[8] bit from 0 to 1, an interrupt is generated to the core.

8.6 Register Definitions

A series of configuration registers control the SMCU. Software can determine the status of the SMCU by reading the status registers. Table 349 lists all of the SMCU registers which are detailed further in proceeding sections.

Note: Constant polling of SMCU MMRs can result in inducing long latencies in peripheral unit SRAM transactions, and therefore may negatively impact performance. Polling of SMCU MMRs should be avoided.

Table 349. Memory Controller Register

Section, Register Name — Acronym (Page)
Section 8.6.1, "SRAM Base Address Register — SRAMBAR" on page 536
Section 8.6.2, "SRAM Upper Base Address Register — SRAMUBAR" on page 536
Section 8.6.3, "SRAM ECC Control Register — SECR" on page 536
Section 8.6.4, "SRAM ECC Log Register — SELOGR" on page 538
Section 8.6.5, "SRAM ECC Address Register — SEAR" on page 540
Section 8.6.6, "SRAM ECC Context Address Register — SECAR" on page 540
Section 8.6.7, "SRAM ECC Test Register — SECTST" on page 541
Section 8.6.8, "SRAM Parity Control and Status Register — SPARCSR" on page 542
Section 8.6.9, "SRAM Parity Address Register — SPAR" on page 543
Section 8.6.10, "SRAM Parity Upper Address Register — SPUAR" on page 543
Section 8.6.6, "SRAM ECC Context Address Register — SECAR" on page 540
Section 8.6.11, "SRAM Memory Controller Interrupt Status Register — SMCISR" on page 544

8.6.1 SRAM Base Address Register — SRAMBAR

This register indicates the lower twelve bits of the beginning address (base address) of SRAM memory array space. The SRAM is addressed using a 36-bit address. This register is used in conjunction with the Section 8.6.2, SRAM Upper Base Address Register — SRAMUBAR. After reset the default starting address of the SRAM memory is 0 FFE0 0000H.

Note: SRAM memory space must ***never*** cross a 1 Mbyte boundary.

Table 350. SRAM Base Address Register — SRAMBAR

Intel XScale® processor Local Bus Address Offset
+ 1500H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

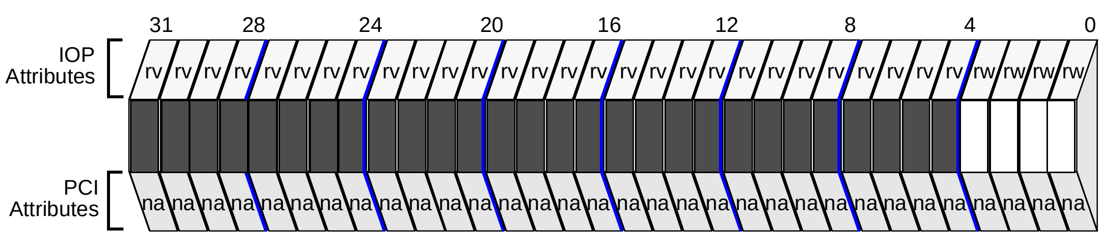
Bit	Default	Description
31:20	1111 1111 1110$_2$	**SRAM Base Address:** Provide lower twelve bits of SRAM base address. Default SRAM base address is 0 FFE0 0000H.
19:00	0 0000H	Reserved

8.6.2 SRAM Upper Base Address Register — SRAMUBAR

This register indicates the upper four bits of the beginning address (base address) of SRAM memory array space. The SRAM is addressed using a 36-bit address. This register is used in conjunction with the Section 8.6.1, SRAM Base Address Register — SRAMBAR. After reset the default starting address of the SRAM memory is 0 FFE0 0000H.

Note: SRAM memory space must ***never*** cross a 1 Mbyte boundary.

Table 351. SRAM Upper Base Address Register — SRAMUBAR



Intel XScale® processor Local Bus Address Offset
+1504H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:04	0000 000H	Reserved
03:00	0H	**SDRAM Upper Base Address:** These bits define the upper four bits of the SRAM base address. The default SRAM base address is 0 FFE0 0000H.

8.6.3 SRAM ECC Control Register — SECR

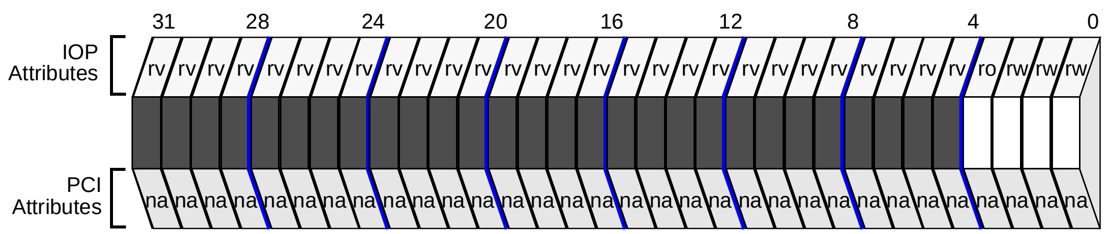
This register programs the SMCU error correction and detection capabilities. The configuration depends on the application's needs but a typical configuration is:

- ECC Mode Enabled
- Enable multi-bit error reporting

- Disable single-bit error reporting
- Enable single-bit error correcting

For more details, see Section 8.3.3, “Error Correction and Detection” on page 519 and Section 8.4, “ECC Interrupts/Error Conditions” on page 531.

Table 352. SRAM ECC Control Register — SECR



Intel XScale® processor Local Bus Address Offset
+1508H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:04	000 0000H	Reserved
03	1_2	Read-only as 1_2.
02	0_2	**Single Bit Error Correction Enable:** Enables or disables the correction of a single bit error. 0 = Disable single bit error correction 1 = Enable single bit error correction
01	0_2	**Multi-Bit Error Reporting Enable:** Enables or disables the reporting (interrupt generation) of a multi-bit error condition. 0 = Disable multi-bit error reporting 1 = Enable multi-bit error reporting
00	0_2	**Single Bit Error Reporting Enable:** Enables or disables the reporting (interrupt generation) of a single bit error condition. 0 = Disable single bit error reporting 1 = Enable single bit error reporting

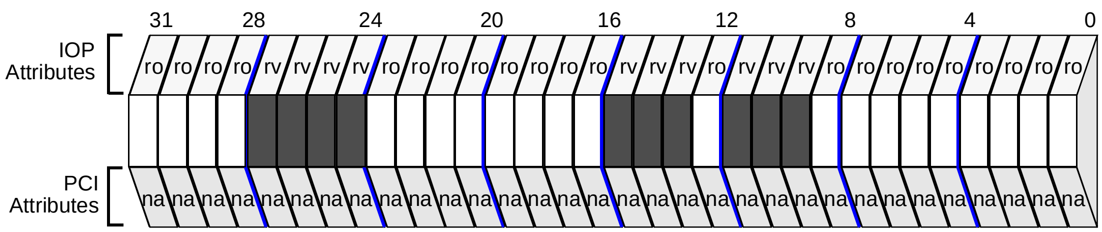
8.6.4 SRAM ECC Log Register — SELOGR

The SRAM ECC Log Register is responsible for logging the error types detected on the local memory bus. One error can be detected and logged. The error type is logged (single-bit or multi-bit) along with the syndrome that indicated the error. For a single-bit error, software can read this syndrome and determine which bit had the error in order to perform scrubbing. For a multi-bit error, the syndrome will not match an entry in the H-Matrix and thus, is not correctable (see Table 346, "Syndrome Decoding" on page 522).

The error recorded in SELOGR corresponds to the address in SECAR and SECUAR.

The SELOGR comprise read-only bits and only have meaning if SMCISR[0] or SMCISR[1] is non-zero. For more details on error handling, see Section 8.3.3, "Error Correction and Detection" on page 519.

Table 353. SRAM ECC Log Register — SELOG (Sheet 1 of 2)

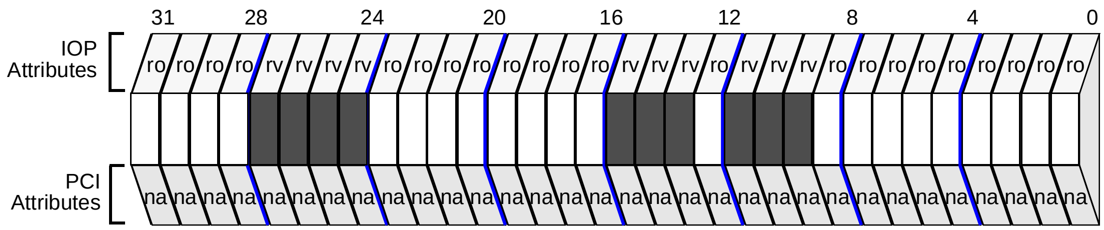


Intel XScale® processor Local Bus Address Offset
+150CH

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:28	0H	**Upper ECC Address** — The upper 4 bits of the 36-bit ECC Address is stored in this 4-bit field when an ECC error is logged. For example, the lower 32 bits are logged in the SRAM ECC Address Register (SEAR).
27:24	0000_2	Reserved
23:20	0H	**ECC Error Requester: Indicates the requester of the logged error:** **Internal Bus Requester ID** — **Requester Name** 0000_2 — Reserved 0001_2 — Intel XScale® processor 0 (coreID0) 0010_2 — Intel XScale® processor 1 (coreID1) 0011_2 — ATUX 0100_2 — ATUE 0101_2 — Application DMA 0110_2 — Reserved 0111_2 — Messaging Unit 1000_2 — Reserved 1001_2 — SMBus All other IDs are reserved. ***Note:*** This field is only valid when the Port ID field in this register (bits[19:16]) indicates the north internal bus as the memory port. ***Note:*** Not all of the Requesters will access the SRAM Memory.

Table 353. SRAM ECC Log Register — SELOG (Sheet 2 of 2)



Intel XScale® processor Local Bus Address Offset
+150CH

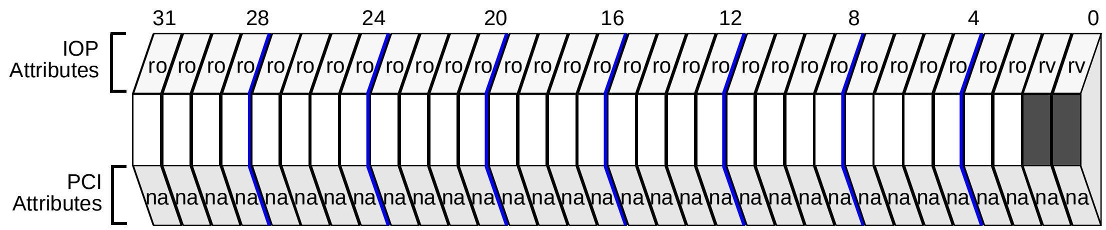
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
19:16	0000_2	**Direct Memory Port ID: Indicates the direct memory port associated with the ECC logged error.** **Port ID** — **Port Name** 0000_2 North Internal Bus Port 0001_2 Reserved 0010_2 Reserved 0011_2 Reserved 0100_2 Reserved 0101_2 Reserved 0110_2 Reserved 0111_2 Reserved 1000_2 Reserved 1001_2 Reserved 1010_2 Reserved All other IDs are reserved.
15:13	000_2	Reserved
12	0_2	**Read or Write**: Indicates if the error occurred during a read or write transaction. 0 = Read error 1 = Write Error
11:09	000_2	Reserved
08	0_2	**ECC Error Type:** Indicates the type of error that occurred at this address. 0 = Single Bit Error 1 = Multi-Bit Error
07:00	00H	**Syndrome:** Holds the syndrome value that indicated the error.

8.6.5 SRAM ECC Address Register — SEAR

This register is responsible for logging the address where the error was detected on the local memory bus. One error can be detected and logged. The software knows which SRAM address had the error by reading this register and decoding the syndrome in the log register. The upper 4 bits are captured in the SECR — refer to Section 8.6.3, SRAM ECC Control Register — SECR. For error details, see Section 8.3.3, "Error Correction and Detection" on page 519).

Table 354. SRAM ECC Address Register — SEAR



Intel XScale® processor Local Bus Address Offset
+1510H

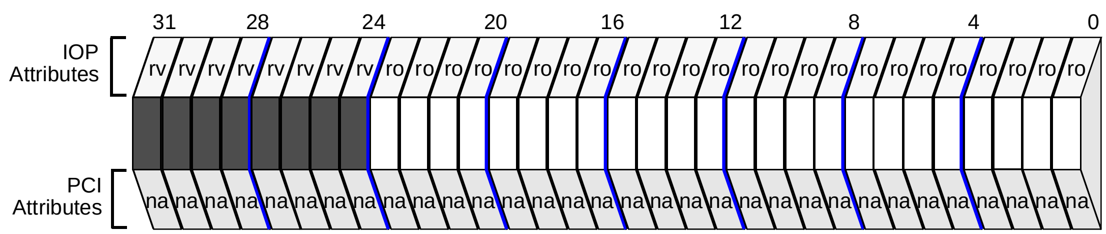
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:02	0	Error Address: Stores the lower 30 bits of the address that resulted in a single bit or multi-bit error.
01:00	00_2	Reserved

8.6.6 SRAM ECC Context Address Register — SECAR

This register is responsible for logging the descriptor tag of the descriptor while the ECC error was detected on the local memory bus. One error can be detected and logged. The software knows which descriptor was being processed by reading this register.

Table 355. SRAM ECC Context Address Register — SECAR



Intel XScale® processor Local Bus Address Offset
+1514H

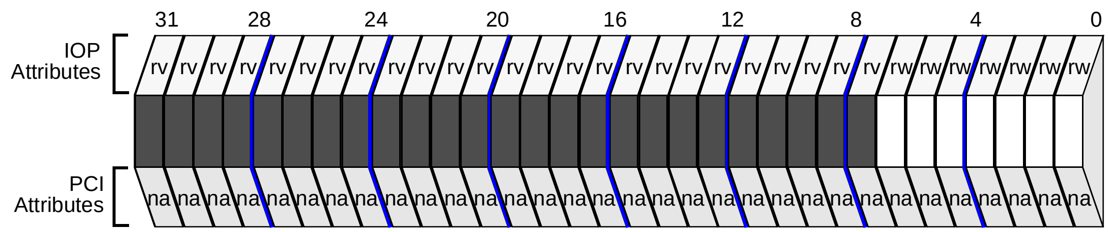
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:24	00H	Reserved
23:00	00 0000H	Reserved.

8.6.7 SRAM ECC Test Register — SECTST

This register allows testing between the SRAM ECC logic and the memory subsystem (Section 8.3.3.6, "ECC Testing" on page 527). To test error handling software, the programmer writes this register with a non-zero masking function. Any subsequent writes to memory stores a masked version of the computed ECC. Therefore, any subsequent reads to these locations result in an ECC error.

Table 356. SRAM ECC Test Register — SECTST



Intel XScale® processor Local Bus Address Offset
+1518H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

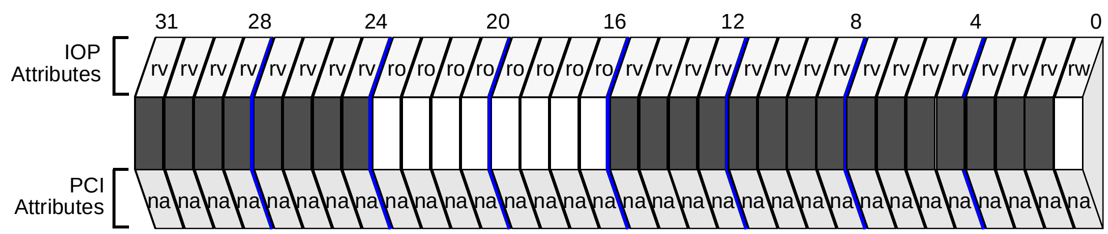
Bit	Default	Description
31:07	00 0000H	Reserved
06:00	00H	ECC Mask: 7-bit ECC mask. Each bit of the generated ECC is XORed with the appropriate bit in this mask field before the ECC is stored into memory. See Section 8.3.3.6, "ECC Testing" on page 527.

8.6.8 SRAM Parity Control and Status Register — SPARCSR

This register programs the SMCU parity checking capabilities. This register is also responsible for logging the error types detected on the SMCU memory ports. Only one error can be detected and logged. The error recorded corresponds to the addresses in (SPAR, SPUAR) and (SPCAR, SPCUAR).

The status bits are read-only bits and only have meaning if SMCISR[8] is non-zero. For more details, see Section 8.3.4, "Byte Parity Checking and Generation" on page 528 and Section 8.5, "Parity Interrupts/Error Conditions" on page 534.

Table 357. SRAM Parity Control and Status Register — SPARCSR



Intel XScale® processor Local Bus Address Offset
+151CH

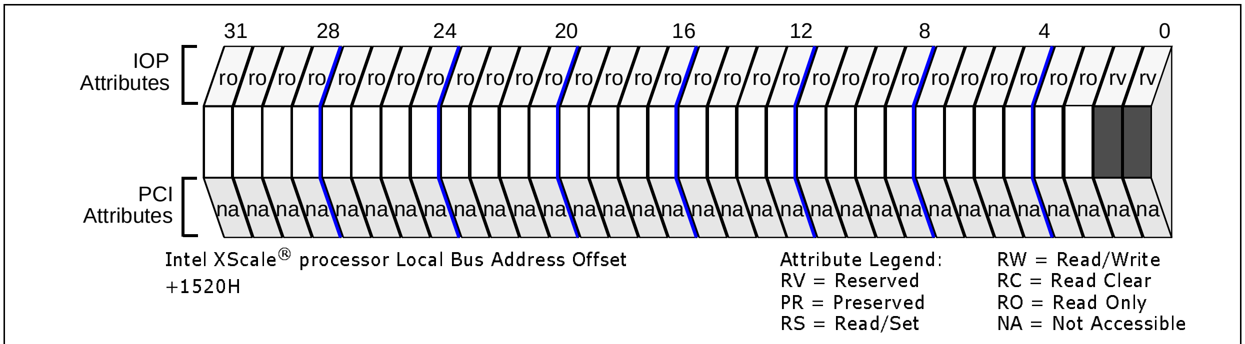
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:24	00H	Reserved
23:20	0000$_2$	**Parity Error Requester: Indicates the requester of the logged error:** **Internal Bus Requester ID** **Requester Name** 0000$_2$ Reserved 0001$_2$ Intel XScale® processor 0 (coreID0) 0010$_2$ Intel XScale® processor 1 (coreID1) 0011$_2$ ATUX 0100$_2$ ATUE 0101$_2$ Application DMAs 0110$_2$ Reserved 0111$_2$ Messaging Unit 1000$_2$ Reserved 1001$_2$ SMBus All other IDs are reserved. ***Note:*** This field is only valid when the Port ID in this register (bits[19:16]) indicates the north internal bus as the memory port. ***Note:*** Not all of the Requesters can or will access the SRAM Memory.
19:16	0000$_2$	**Direct Memory Port ID:** Indicates the direct memory port associated with the parity error: **Port ID** **Port Name** 0000$_2$ North Internal Bus 0001$_2$ Reserved 0010$_2$ Reserved 0011$_2$ Reserved 0100$_2$ Reserved 0101$_2$ Reserved 0110$_2$ Reserved 0111$_2$ Reserved 1000$_2$ Reserved 1001$_2$ Reserved 1010$_2$ Reserved All other Port IDs are reserved.
15:01	0000H	Reserved
00	0$_2$	**SMCU Parity Enabled:** Enables or disables checking, logging, and reporting (interrupt generation) of parity errors on the memory ports. 0 = Disable Parity Error 1 = Enable Parity Error

8.6.9 SRAM Parity Address Register — SPAR

This register is responsible for logging the lower 32-bit address of where the error was detected on the SMCU memory ports. Note that the address is 36-bit. This register is used in conjunction with the Section 8.6.10, "SRAM Parity Upper Address Register — SPUAR" on page 543. One error can be detected and logged. The software knows which SRAM address had the error by reading this register and decoding contents of associated log register. For error details, see Section 8.3.3, "Error Correction and Detection" on page 519).

Table 358. SRAM Parity Address Registers — SPAR

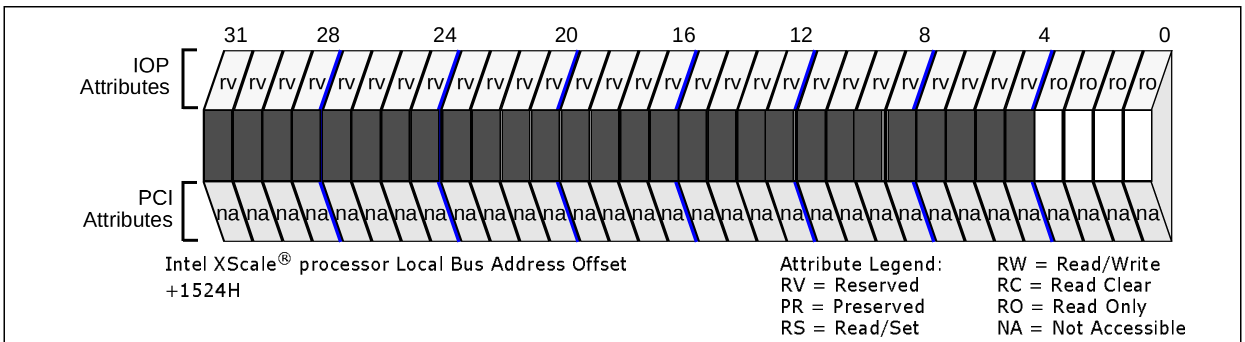


Bit	Default	Description
31:02	0000 0000H	Error Address: Stores the upper 30 bits of the address that resulted in a parity error.
01:00	00_2	Reserved

8.6.10 SRAM Parity Upper Address Register — SPUAR

This register is responsible for logging the upper 4-bit address of where the error was detected on the SMCU memory ports. Note that the address is 36-bit. This register is used in conjunction with the Section 8.6.9, "SRAM Parity Address Register — SPAR" on page 543. One error can be detected and logged. The software knows which SRAM address had the error by reading this register and decoding contents of associated log register. For error details, see Section 8.3.4, "Byte Parity Checking and Generation" on page 528).

Table 359. SRAM Parity Upper Address Register — SPUAR



Bit	Default	Description
31:04	0000 000H	**Reserved**
03:00	0000_2	Parity Error Address: Stores the upper 4 bits of the 36-bit address that resulted in the parity error.

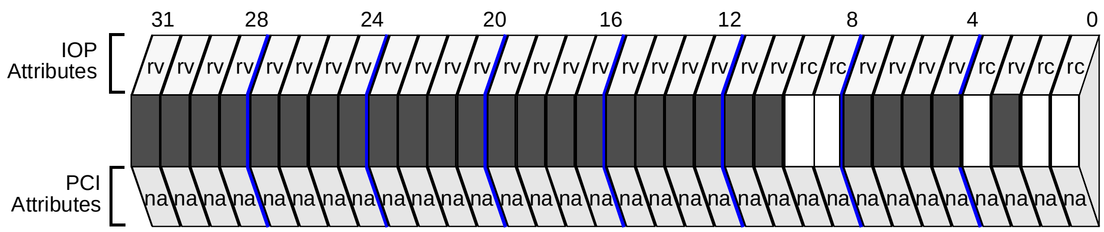
8.6.11 SRAM Memory Controller Interrupt Status Register — SMCISR

Setting the SMCISR asserts an interrupt to the core. Upon an interrupt, the Intel XScale® processor polls the interrupt status register for each unit. The interrupt status register tells the core the reason for the interrupt. The SMCU has five interrupt conditions: first ECC error (SMCISR[0]), more than one ECC error (SMCISR[1]), first parity error (SMCISR[8]), and more than one parity error (SMCISR[9]).

If the SMCU detects an ECC error and SMCISR[0] is cleared, the error is logged in SELOG and SMCISR[0] is set to 1. If SMCISR[0] is not cleared, any additional ECC errors are not logged and SMCISR[1] is set.

Similarly, if the SMCU detects a parity error and SMCISR[8] is cleared, the parity error is logged in SPCSR and SMCISR[8] is set to 1. If SMCISR[8] is not cleared, any additional parity errors are not logged and SMCISR[9] is set.

Table 360. SRAM Memory Controller Interrupt Status Register — SMCISR



Intel XScale® processor Local Bus Address Offset
+152CH

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:05	0000 000H	Reserved
09	0_2	**Parity N:** Indicates that the SMCU detected a Parity error while SMCISR[8] was set. 0 = No error detected 1 = Error detected
08	0_2	**Parity Error:** Indicates that the SMCU detected a Parity error and recorded the error in SPCSR. 0 = No error detected 1 = Error detected and recorded in SPLOG
07:04	0H	Reserved
03	0_2	Address Range Error: Indicates that a transaction to an invalid address range. For example, an address that is outside the 1.0 MBytes SRAM space. 0 = No error detected 1 = Error detected
02	0_2	Reserved
01	0_2	**ECC Error N:** Indicates that the SMCU detected an ECC error while MCISR[0] was set. 0 = No error detected 1 = Error detected
00	0_2	**ECC Error 0:** Indicates that the SMCU detected an ECC error and recorded the error in SELOG. 0 = No error detected 1 = Error detected and recorded in SELOG

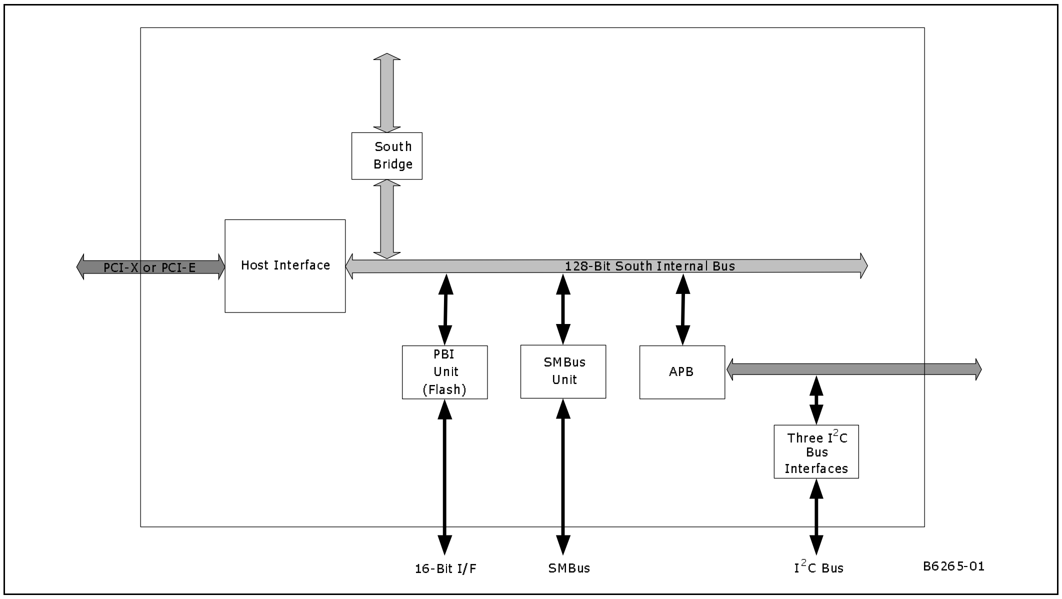
9.0 Peripheral Bus Interface Unit

This chapter describes the Peripheral Bus Interface Unit (PBI) of the Intel® 413808 and 413812 I/O Controllers in TPER Mode (4138xx). It explains the following:

- Peripheral Bus signals, which consist of address/data, control/status
- Peripheral Bus Read, and write transactions
- Peripheral Bus configuration and Flash Memory Support
- Registers

This chapter also serves as a starting point for the hardware designer when interfacing typical flash components to the 4138xx Peripheral Bus.

Figure 62. The Peripheral Bus Interface Unit



9.1 Overview

The Peripheral Bus Interface Unit (PBI) is a data communication path to the flash memory components and peripherals of a 4138xx hardware system. The PBI allows the processor to read and write data to these supported flash components and other peripherals. To perform these tasks at high bandwidth, the PBI bus features a burst read transfer capability which allows successive data transfers for multi-byte read requests. The maximum PBI bus read burst size is four data transfers, regardless of bus width. This means that some read requests made to the PBI may result in multiple PBI bus burst accesses. For example a read request for 32 bytes on a 16-bit PBI bus results in 4 bursts of four 16-bit words on the Peripheral Bus. Multi-byte read requests and multi-byte write requests are supported differently by the PBI. Write requests are limited to a maximum of 4 bytes only and must not span a DWORD boundary. The PBI signals an address error when a write request has its byte-count out of range.

The peripheral bus is controlled by the on-chip bus masters: the Intel XScale® processor 0, the Intel XScale® processor 1, the ATU-E and the ATU-X units.

The address and data paths are demultiplexed, and the bus width is programmable to 8-, and 16-bit widths. The PBI performs the necessary packing and unpacking of bytes to communicate properly across the 4138xx Internal Bus.

The PBI unit includes two chip enables. The PBI chip enables activate the appropriate peripheral device when the address falls within one of the PBIs two programmable address ranges. Both address ranges incorporate functionality that optimizes an interface for Flash Memory devices. Each chip enable can support up to 32 MBytes of addressability. For example, there are 25 address signals provided on the PBI interface.

Note: Be sure to refer to the System Software Architecture Specification and Design Guide for details on supported Flash parts, since the Transport Firmware must provide support for the Flash device in addition to PBI.

9.2 Peripheral Bus Signals

Bus signals consist of three groups:

- address
- data
- control/status

9.2.1 Address Signal Definitions

The address signal group **A[24:0]** consists of 25 lines which allows the PBI to address up to 32 MBytes per peripheral device. During and address cycle (T_A), the processor drives **A[24:0]** with the starting address of the bus access. During bursted read access the wait cycle (T_W) and the T_D cycle, **A[2:0]** address pins provide incrementing byte addresses.

9.2.2 Data Signal Definitions

The data signal group **D[15:0]** consists of 16 lines. The PBI supports either an 8-bit data bus width on **D[7:0]** or 16-bit data bus width on **D[15:0]**. During the address cycle (T_A), bits **D[1:0]** carry the SIZE of the access.

9.2.3 Control/Status Signal Definitions

The control/status signals control peripheral device enables and direction. All output control/status signals are three-state.

A peripheral read may be either non-burst or burst. A non-burst read ends after one data transfer to a single location.

When the data bus is configured for 16 bits, address bits **A[2:1]** are used to burst across up to four short-words. For an 8-bit data bus, address bits **A[1:0]** are used to burst across up to four bytes.

The Output Enable, **POE#**, is used for burst or non-burst read accesses to a peripheral device and is asserted during the $T_{A/}T_W/T_D$ states.

The Write Enable, **PWE#**, is used for non-burst write accesses to a peripheral device and is asserted during the T_W/T_D states.

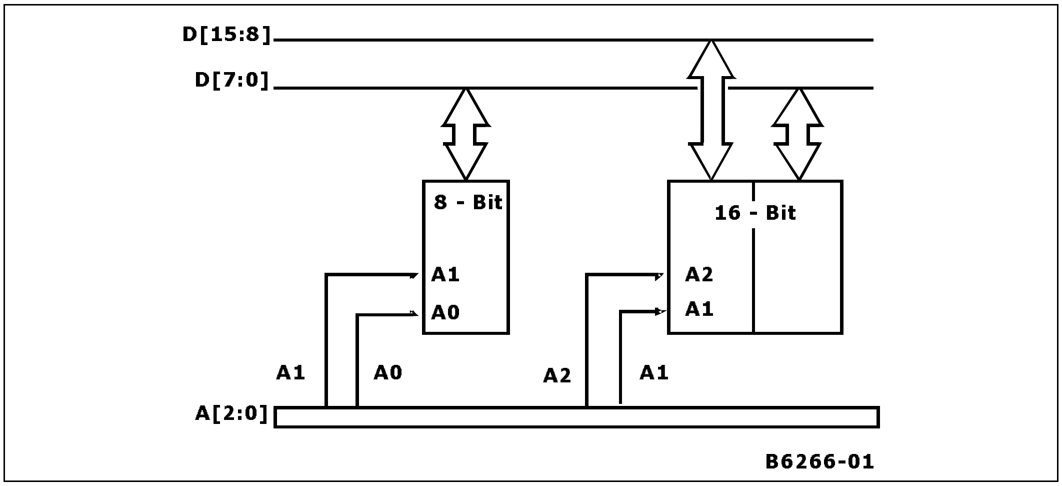
The PBI Reset, **PB_RSTOUT#**, is used to reset the peripheral device. It has the same timings as the internal bus reset signal.

Note: Burst write accesses are not supported by the PBI bus. A multi-byte write request made to the PBI translates into multiple single data write transactions on the PBI bus. For example, each write transaction on the PBI bus ends after one data transfer to a single address location. Note that the number of single data write transactions initiated on the PBI bus are dependent to the PBI bus width.

9.2.4 Bus Width

Each address range's attributes are programmed in the PBIs boundary registers. The PBI allows an 8-, or 16-bit data bus width for each range. The PBI places 8- and 16-bit data on low-order data signals, simplifying the interface to narrow bus external devices. As shown in Figure 63, 8-bit data is placed on lines **D[7:0]**; 16-bit data is placed on lines **D[15:0]**.

Figure 63. Data Width and Low Order Address Lines



The user needs to wire up the flash memories in a manner consistent with the programmed bus width:

- 8-bit region: **A[1:0]** provide the demultiplexed byte address for a read burst.
- 16-bit region: **A[2:1]** provide the demultiplexed short-word address for a read burst.

During initialization, bus width is selected for each of the two address ranges in the Peripheral Base Address Registers (PBBAR0 — PBBAR1). In addition, the PBBAR0-PBBAR1 can be used to configure these ranges as Peripheral Windows and to set a Wait state profile.

The PBI drives determinate values on all address/data signals during T_W/T_D write operation states. For an 8-bit bus, the PBI continues to drive address on unused data signals **D[15:8]**.

9.2.5 Detailed Signal Descriptions

Bus signal descriptions are detailed in Table 362.

Table 361. Bus Signal Descriptions

NAME	DESCRIPTION
D[15:0]	**DATA BUS** carries 16-bit physical addresses and 8-, or 16-bit data to and from memory. During a data (T_d) cycle, bits 0-7, or 0-15 contain read or write data, depending on the corresponding bus width. During write operations to 8-bit wide memory regions, the PBI drives unused bus pins high or low.
A[24:0]	**ADDRESS BUS 24:0** carries the 25-bit address bus which allows the PBI to address up to 32 MBytes per peripheral device. During an address (T_a) cycle, bits **A[2:0]** contains the starting address of the access. During a bursted read data (T_d) cycle, **A[2:0]** represents the current byte address in the bursted transaction. Address bits **A[24:3]** provide the upper address of the current access and is a constant during the address (T_a), wait state (T_w) and data cycles (T_d) cycles. **A[2:1]** are used for an 16-bit wide peripheral while A1:0 are used for an 8-bit wide peripheral.
POE#	**PERIPHERAL OUTPUT ENABLE** specifies, during a T_a cycle, whether the operation is a write (1) or read (0). It is latched on-chip and remains valid during T_d cycles. This signal is used as an OUTPUT ENABLE signal (OE#) for Peripheral Devices.
PCE[1:0]#	**PERIPHERAL CHIP ENABLES 1:0** specify, during a T_a cycle, which of the two Memory Address Ranges are associated with the current bus access. It remains valid during T_d cycles
PWE#	**PERIPHERAL WRITE ENABLE** indicates to a peripheral device whether or not to use the data on the D15:0 bus to write the addressed space. It is low during T_w cycles and deasserts during the T_d cycle for a write; it is high during T_a and T_w/T_d cycles for a read.
PB_RSTOUT#	**PERIPHERAL BUS RESET** can be used to reset the peripheral device. **PB_RSTOUT#** has the same timings as the internal bus reset signal.

9.2.6 Flash Memory Support

Note: Be sure to refer to the System Software Architecture Specification and Design Guide for details on supported Flash parts, since the Transport Firmware must provide support for the Flash device in addition to PBI.

PBI peripheral bus interface supports 8-, or 16- bit Flash devices.

The PBI provides programmable wait state functionality for peripheral memory windows.

Note: Potentially, programmable wait state functionality could be connected to any peripheral device that has a deterministic wait state profile. However, data valid and turn-around times would need to fit within parameters provided by programmable wait state profiles to support Flash devices.

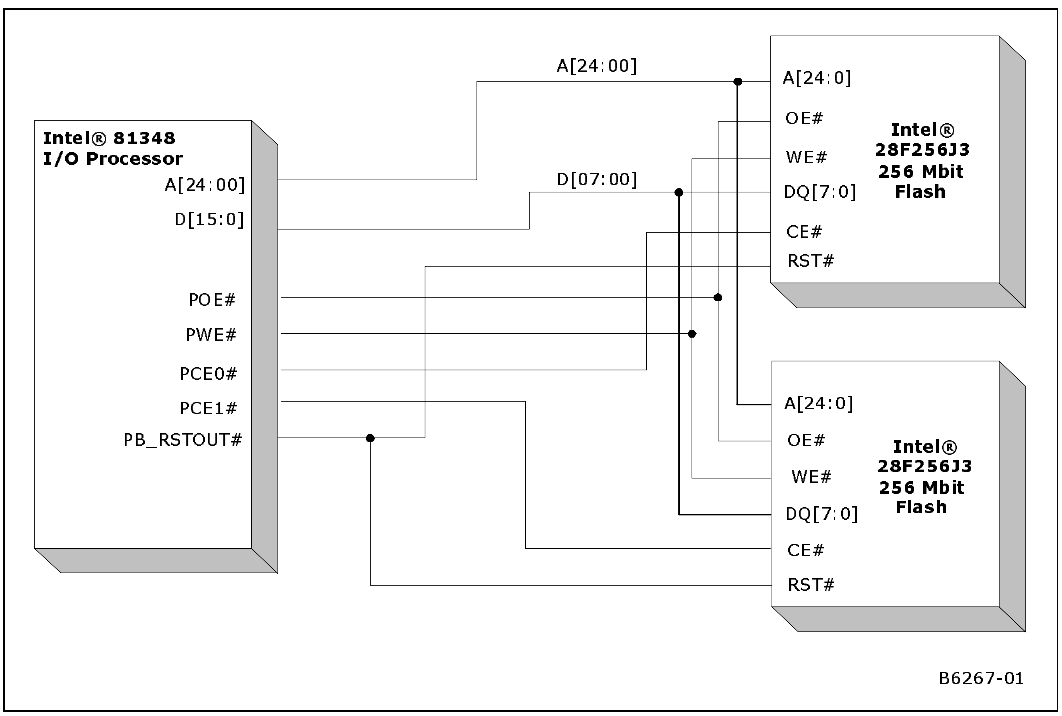
Any write transactions issued to a Flash address space window must always represent a single flash bus data cycle (**strb, strh**).

The peripheral chip enables, **PCE[1:0]#**, activate the appropriate Peripheral window when the address falls within one of the Peripheral address ranges.

Note: By default, bank 0 is enabled with the maximum number of Address-to-Data and Recovery Wait states. The width of the interface can be strapped for either 8-bit wide Flash or 16-bit wide flash. Thus, **PCE0#** is the Peripheral Bus chip enable to be used for booting purposes.

Figure 64 on page 550 illustrates how two 8-bit Flash devices would interface with an Intel XScale® processor through the PBI Interface.

Figure 64. Sixty-Four Mbyte Flash Memory System



9.2.6.1 Flash Read Cycle

Reading a Flash device involves driving the address, output enable, and chip enable. Depending on the speed of the Flash device, the data returns several cycles later.

The definition of address-to-data wait states are the number of cycles between the assertion of **PCE[1:0]#**, and the arrival of data from the Flash device on **D[7:0]** (8-bit Flash). The definition of recovery wait states are the number of cycles between the data arrival on **D[7:0]** and the address for the next Peripheral transaction.

Address-to-data and recovery wait states are programmed in PBBAR0 and PBBAR1 and are identical for reads and writes. Since the read wait state requirement is typically greater, the write wait state requirement is insured to be met.

Figure 65 illustrates a single read cycle example for a 120 ns Flash device. The number of wait states used for address-to-data is provided by the *Address-to-Data Wait States* field in the PBBARx.

Figure 65. 120 ns Flash Single Transfer Read Cycle

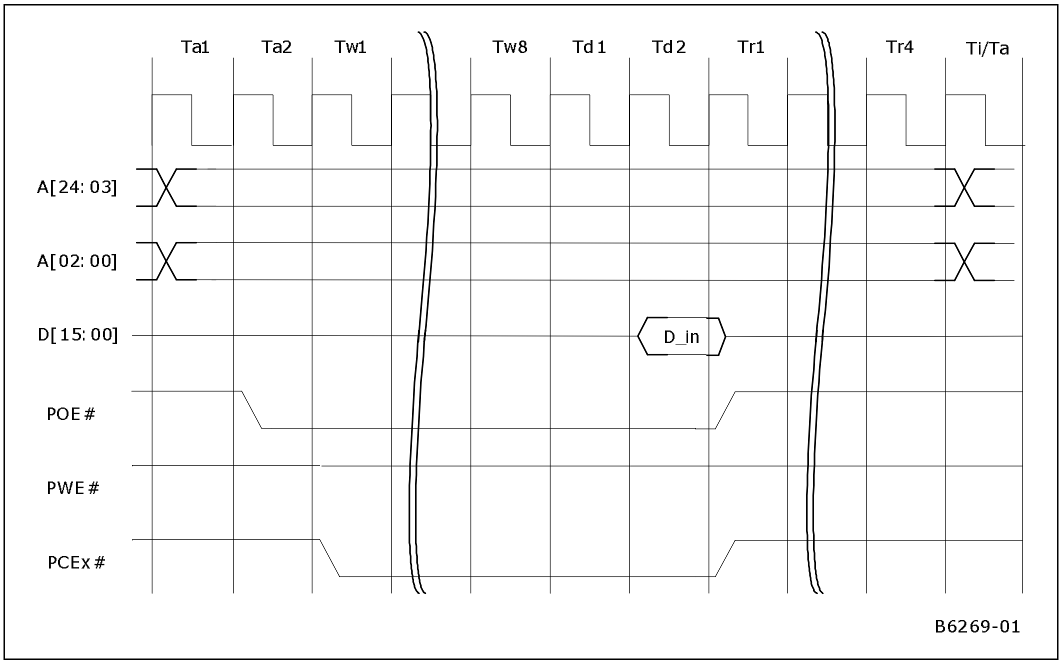
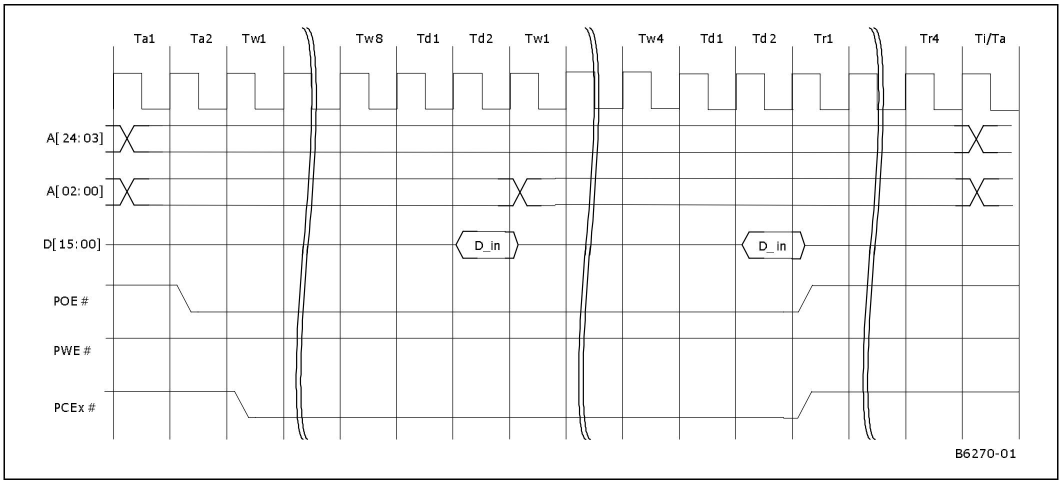


Figure 66 illustrates a burst read cycle example for a 120 ns Flash device. This example is illustrating a burst of two bytes or words. The PBI is capable of bursting up to four bytes or words. The number of wait states used for address-to-data and data-to-data are provided by the *Address-to-Data Wait States* and the *Data-to-Data Wait States* field in PBBARx respectively.

Figure 66. 120 ns Flash Burst Read Cycle



Refer to Table 362 for the programmable address-to-data, data-to-data, and recovery wait states. These numbers are based on a 66 MHz internal clock for the PBI interface.

Table 362. Flash Wait State Profile Programming[1]

Flash Speed	Address-to-Data Wait States	Data-to-Data Wait States[2]	Recovery Wait States
<= 55 ns	4	4	1
<= 120 ns	8	4	4
<= 150 ns	12	4	4

Notes:
1. Each Wait State represents a 15 ns period based on a 66 MHz clock. Refer to the appropriate Flash device datasheets for programming accurate wait state numbers.
2. Data-to-Data wait states are used for burst reads.

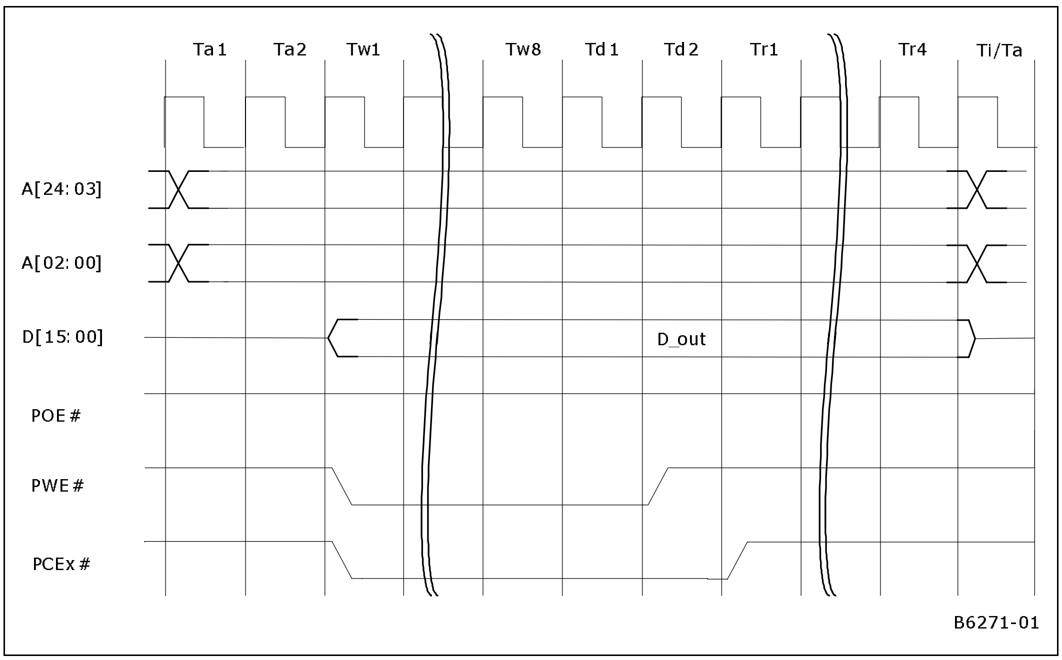
9.2.6.2 Flash Write Cycle

Address-to-data and recovery wait states for reads and writes are identical and programmed in PBBAR0 and PBBAR1. Refer to Table 362 for the programmable address-to data wait states. However, Any write transactions issued to a Peripheral address space window must always represent a single peripheral bus data cycle (**strb, strh**) depending on the bus width selected in PBBAR0 — PBBAR1.

The PBI supports multi-byte write requests from the internal bus agents. Multi-byte read and write requests are supported differently by the PBI. Write requests are limited to a maximum of four bytes only and must not span a DWORD boundary. For a write request with an address and byte-count combination that spans a DWORD boundary, the PBI signals an address error and set bit 0 of the "PBI Status Register — PBISR" on page 555. Unlike multi-byte read requests, the PBI supports multi-byte write requests by breaking the writes on the PBI bus into multiple single data write transactions. The number of single data write transactions initiated on the PBI bus are dependent to the PBI bus width. For example, an aligned DWORD write on an 8-bit PBI bus turns into four single 8-bit write transactions on the PBI bus, and an aligned DWORD write request on a 16-bit PBI bus turns into two single 16-bit write transactions on the PBI bus. Figure 67 shows the only type of write transaction supported by the PBI bus. For example, each write transaction ends after one data transfer to a single address location regardless of bus width.

Figure 67 illustrates a write cycle example for a 120 ns Flash device. Both 8- and 16-bit wide bus timings are identical.

Figure 67. 120 ns Flash Single Write Cycle[1]



Notes:

1. The PBI bus does not burst write transactions. A multi-byte write request made to the PBI is translated into multiple single data write transactions on the PBI bus. And the number of single data write transactions initiated on the PBI bus are dependent to the PBI bus width.

9.3 Register Definitions

A series of configuration registers control PBI. Software can determine PBI status by reading the status register. Table 363 lists all PBI registers which are detailed further in proceeding sections.

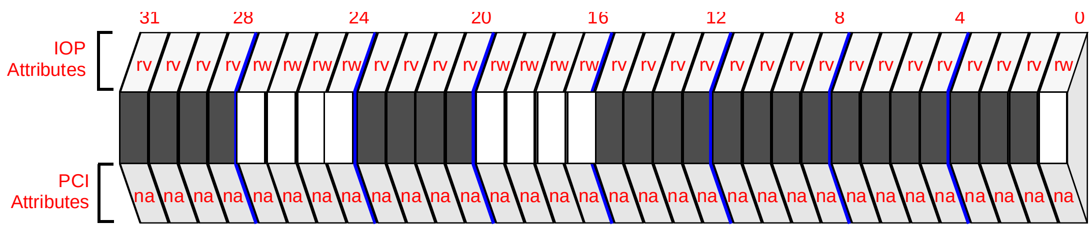
Table 363. Peripheral Bus Interface Registers

Section, Register Name — Acronym (Page)
Section 9.3.1, "PBI Control Register — PBCR" on page 555
Section 9.3.2, "PBI Status Register — PBISR" on page 555
Section 9.3.4, "PBI Base Address Register 0 — PBBAR0" on page 557
Section 9.3.5, "PBI Limit Register 0 — PBLR0" on page 558
Section 9.3.6, "PBI Base Address Register 1 — PBBAR1" on page 559
Section 9.3.7, "PBI Limit Register 1 — PBLR1" on page 560
Section 9.3.8, "PBI Drive Strength Control Register — PBDSCR" on page 561
Section 9.3.9, "Processor Frequency Register - PFR" on page 562
Reserved.
Reserved.
Section 9.3.10, "External Strap Status Register 0 — ESSTSR0" on page 563
Reserved.
Section 9.3.12, "Unique ID Register 1 — UID1" on page 564

9.3.1 PBI Control Register — PBCR

The PBI Control Register (PBCR) is responsible for enabling operation of PBI state machines.

Table 364. PBI Control Register — PBCR



Intel XScale® processor Local Bus Address Offset
+1580H

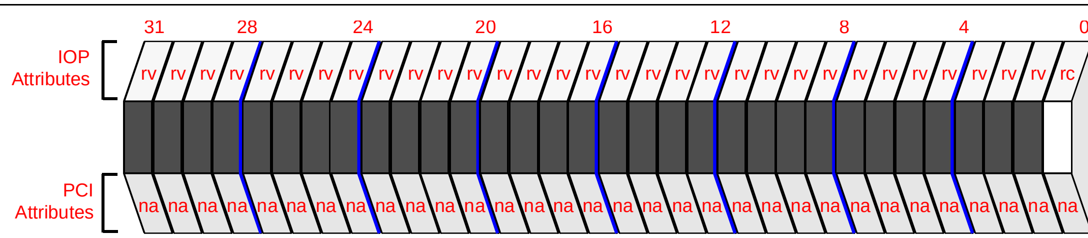
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:28	0H	Reserved.
27:24	0H	PBI Memory Window 1 Upper 4-bit address. These bits, with PBBAR1 form a 36-bit base address.
23:20	0H	Reserved.
19:16	0H	PBI Memory Window 0 Upper 4-bit address. These bits, with PBBAR0 form a 36-bit base address.
15:04	000H	Reserved.
03	0_2	Reserved.
02:01	00	Reserved (bit[2] reads as 1).
00	1_2	PBI Enable: When set, this bit enables the PBI unit bus interface 0 = Peripheral Bus Disabled 1 = Peripheral Bus Enabled.

9.3.2 PBI Status Register — PBISR

The PBI Status Register (PBISR) allows software to determine the cause of any PBI interrupts.

Table 365. PBI Status Register — PBISR



Intel XScale® processor Local Bus Address Offset
+1584H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:01	0	Reserved
00	0_2	Byte Count Out of Range Error — Indicates that the address and byte-count combination of a write request claimed by the PBI was not legal. Write requests made to the PBI is only permitted to be within a DWORD boundary. For example, when the address and byte-count combination spans a DWORD boundary, the PBI signals an address error condition and set this bit.

9.3.3 Determining Block Sizes for Memory Windows

The memory window size can be determined by writing ones to the appropriate upper bits of the limit register. The binary-weighted value of the first non-zero bit set in the limit register indicates the size of the memory window. Table 366 describes the relationship between limit register values and the byte sizes of the memory window.

Table 366. Memory Block Size Limit Register Values

Limit Register Value[a]	Size (in Bytes)	Limit Register Value	Size (in Bytes)
FFFFF000H	4K	FF800000H	8 M
FFFFE000H	8K	FF000000H	16 M
FFFFC000H	16K	FE000000H	32 M
FFFF8000H	32K	FC000000H	Address Window Closed.
FFFF0000H	64K	F8000000H	
FFFE0000H	128K	F0000000H	
FFFC0000H	256K	E0000000H	
FFF80000H	512K	C0000000H	
FFF00000H	1 M	80000000H	
FFE00000H	2 M	00000000H	
FFC00000H	4 M		

a. Smallest Limit Register Value is 4 KBytes.

As an example, assume that FFF0 0000H is written to the PBI Limit Register 0 (PBLR0). Scanning upwards starting at bit 12, bit 20 is the first one bit found. The binary-weighted value of this bit is 1,048,576, indicating a 1 Mbyte of memory window.

When programming the Base and Limit Registers for a memory window, the Base Address always needs to be aligned the size of the memory window set in a limit register. For a 1 Mbyte memory window, only bit 20 through bit 31 of the base address from the PBI Base Address Register 0 (PBBAR0) are relevant to the PBI when decoding Memory Window 0.

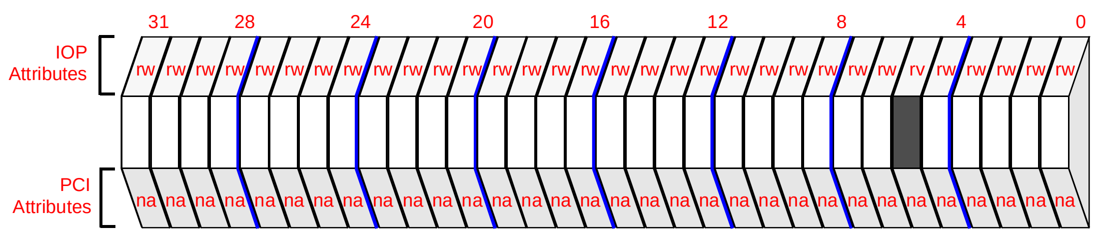
Warning: A given PBI Base (PBBAR0-PBBAR1) and Limit (PBLR0-PBLR1) register pair should not be modified during the time there is activity on the peripheral bus associated with that particular peripheral memory window. For instance, following boot-up, code executing from Peripheral Memory Window 0 may be used to modify the PBI Base and Limit registers for Peripheral Memory Window 1, but **not** for Peripheral Memory Window 0.

9.3.4 PBI Base Address Register 0 — PBBAR0

The PBI Base Address Register 0 (PBBAR0) defines the block of memory addresses where PBI Memory Window 0 begins. The PBBAR0 defines the base address and describes the required memory block size; see Section 9.3.3, "Determining Block Sizes for Memory Windows" on page 556. The selected base address needs to be naturally aligned to the granularity of the memory block size. For instance, when a 64 Kbyte memory window size is selected, the base address needs to be 64 Kbyte address aligned (i.e., bits 15:12 of the base address are required to be 000b).

Bits 1:0 define the PBI bus width for PBI Memory Window 0.

Table 367. PBI Base Address Register 0 — PBBAR0



Intel XScale® processor Local Bus Address Offset
+1588H

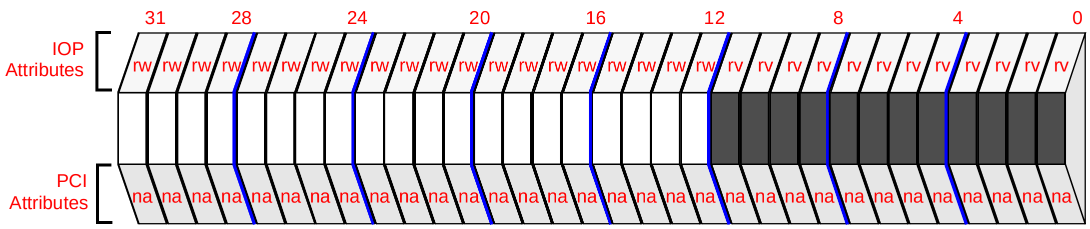
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:12	00000H	Memory Window 0 Base Address: These bits define the actual location the PBI responds to for accesses to Memory Window 0.
11:09	001_2	Data-to-Data Wait States: Defines the number of data-to-data wait states for the Peripheral window during a burst read transaction. 00X - Wait states are equal to the programmed value in the Address-to-Data Wait States field. 010 - 4 Data-to-Data wait states 011 - 8 Data-to-Data wait states 100 - 12 Data-to-Data wait states 101 - 16 Data-to-Data wait states Others - 20 Data-to-Data wait states ***Note:*** By default the data-to-data wait states are 20, since it is the same as the address-to-data wait states.
08:06	111_2	Recovery Cycle Wait States: Defines the number of recovery cycle wait states for the Peripheral window. 000 - 1 Recovery wait state 001 - 4 Recovery wait states 010 - 8 Recovery wait states 011 - 12 Recovery wait states 100 - 16 Recovery wait states Others (Default) - 20 Recovery wait states
05	0_2	Reserved
04:02	111_2	Address-to-Data Wait States: Defines the number of address-to-data wait states for the Peripheral window during a read or write transaction. 000 - 4 Address-to-Data wait states 001 - 8 Address-to-Data wait states 010 - 12 Address-to-Data wait states 011 - 16 Address-to-Data wait states Others (Default) - 20 Address-to-Data wait states
01:00	Varies with external state of **BOOT_WIDTH_8#,** during a Internal Bus reset	Bus Width: Defines the bus width for this Memory Window: 00 - 8 bits wide 01 - 16 bits wide 10 - Reserved 11 - Reserved ***Note:*** When **BOOT_WIDTH_8#** is asserted, the default bus width is 8 bits wide (00), else the default bus width is 16 bits wide (01).

9.3.5 PBI Limit Register 0 — PBLR0

The 4138xx limit register's (PBLR0) programmed value must be naturally aligned with the base address register's (PBBAR0) programmed value. The limit register is used as a mask when the address decode for memory window 0 is performed.

Table 368. PBI Limit Register 0 — PBLR0



Intel XScale® processor Local Bus Address Offset
+158CH

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

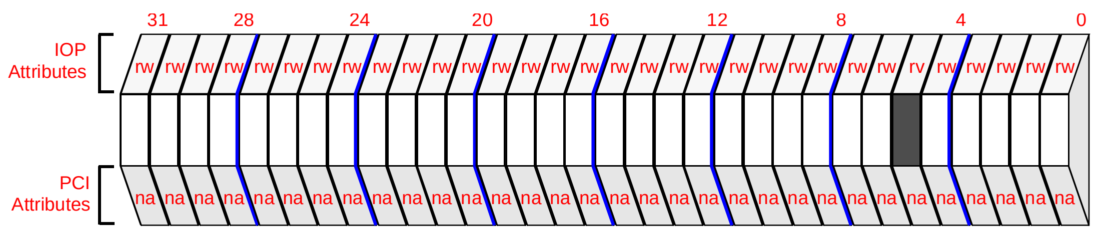
Bit	Default	Description
31:12	FE000H	Memory Window 0 Limit: This value determines the memory block size required for the Memory Window 0. Defaults to an 32MB Peripheral Window
11:00	000H	Reserved

9.3.6 PBI Base Address Register 1 — PBBAR1

The PBI Base Address Register 1 (PBBAR1) defines the block of memory addresses where PBI Memory Window 1 begins. The PBBAR1 defines the base address and describes the required memory block size; see Section 9.3.3, "Determining Block Sizes for Memory Windows" on page 556. The selected base address needs to be naturally aligned to the granularity of the memory block size. For instance, when a 64 Kbyte memory window size is selected, the base address needs to be 64 Kbyte address aligned (i.e., bits 15:12 of the base address are required to be 000b).

Bits 1:0 define the PBI bus width for PBI Memory Window 1.

Table 369. PBI Base Address Register 1 — PBBAR1



Intel XScale® processor Local Bus Address Offset
+1590H

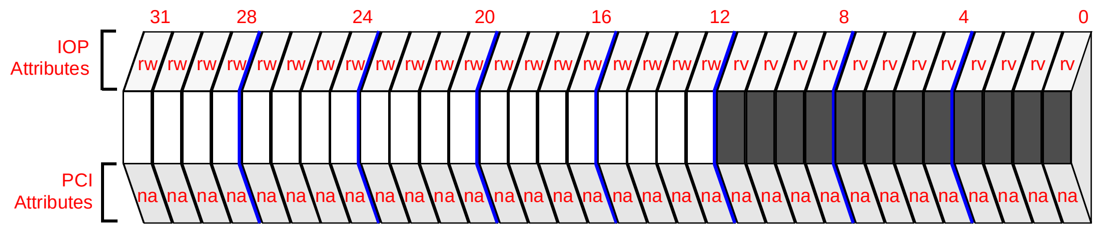
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:12	00000H	Memory Window 1 Base Address: These bits define the actual location the PBI responds to for accesses to Memory Window 1.
11:09	001_2	Data-to-Data Wait States: Defines the number of data-to-data wait states for the Peripheral window during a burst read transaction. 00X - Wait states are equal to the programmed value in the Address-to-Data Wait States field. 010 - 4 Data-to-Data wait states 011 - 8 Data-to-Data wait states 100 - 12 Data-to-Data wait states 101 - 16 Data-to-Data wait states Others - 20 Data-to-Data wait states ***Note:*** By default, data-to-data wait states are 20, since it is the same as address-to-data wait states.
08:06	111_2	Recovery Cycle Wait States: Defines the number of recovery cycle wait states for the Peripheral window. 000 - 1 Recovery wait state 001 - 4 Recovery wait states 010 - 8 Recovery wait states 011 - 12 Recovery wait states 100 - 16 Recovery wait states Others (Default) - 20 Recovery wait states
05	0_2	Reserved
04:02	111_2	Address-to-Data Wait States: Defines the number of address-to-data wait states for the Peripheral window during a read or write transaction. 000 - 4 Address-to-Data wait states 001 - 8 Address-to-Data wait states 010 - 12 Address-to-Data wait states 011 - 16 Address-to-Data wait states Others (Default) - 20 Address-to-Data wait states
01:00	00_2	Bus Width: Defines the bus width for this Memory Window: 00 - 8 bits wide 01 - 16 bits wide 10 - Reserved 11 - Reserved

9.3.7 PBI Limit Register 1 — PBLR1

The 4138xx limit register (PBLR1) and base address register (PBBAR1) programmed values must be naturally aligned. The limit register is used as a mask when the address decode for memory window 1 is performed.

Table 370. PBI Limit Register 1 — PBLR1



Intel XScale® processor Local Bus Address Offset
+1594H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

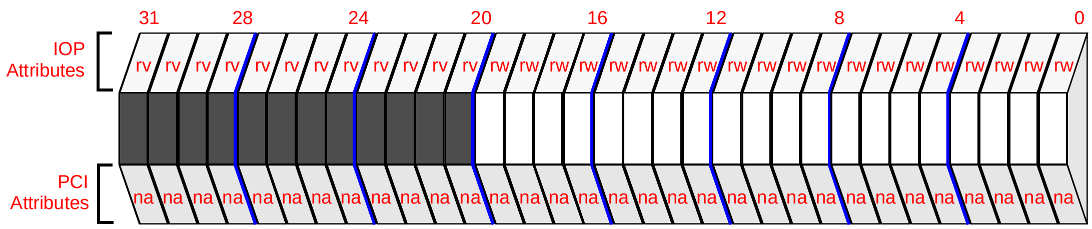
Bit	Default	Description
31:12	00000H	Memory Window 1 Limit: Determines the memory block size required for the Memory Window 1.
11:00	000H	Reserved

9.3.8 PBI Drive Strength Control Register — PBDSCR

This register is used to manually control slew rate and drive strength for all of 4138xx pins with the exception of high-speed serial interfaces, the SDRAM interface, and the PCI-X interface.

Note: By default, this register is **not** required to program. This register should not be programmed to a different value without consulting the 4138xx where the appropriate values are specified.

Table 371. PBI Drive Strength Control Register — PBDSCR



Intel XScale® processor Local Bus Address offset +2080H

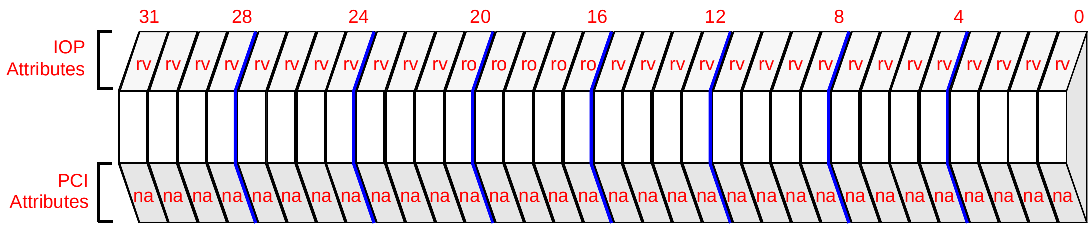
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:20	000H	Reserved
19:16	0011_2	Pull-Down Slew Rate Control (NSLW[3:0]): Tunes the slew rate of the n-drivers of all the pins with the exception of the high speed serial interfaces, the SDRAM interface and the PCI-X interface.
15:12	0011_2	Pull-Up Slew Rate Control (PSLW[3:0]): Tunes the slew rate of the p-drivers of all the pins with the exception of the high speed serial interfaces, the SDRAM interface and the PCI-X interface.
11:06	001100_2	Pull-Down Drive Strength (NDRV[5:0]): Programs the strength of the n-drivers of all the pins with the exception of the high speed serial interfaces, the SDRAM interface and the PCI-X interface.
05:00	010110_2	Pull-Up Drive Strength (PDRV[5:0]): This field programs the strength of the p-drivers of all the pins with the exception of the high speed serial interfaces, the SDRAM interface and the PCI-X interface.

9.3.9 Processor Frequency Register - PFR

This register indicates the clock frequencies of the Intel XScale® processor and the internal buses of the 4138xx. For products with two enabled Intel XScale® processors, the cores run at the same clock frequency.

Table 372. Processor Frequency Register - PFR



Intel XScale® processor Local Bus Address offset
+2180H

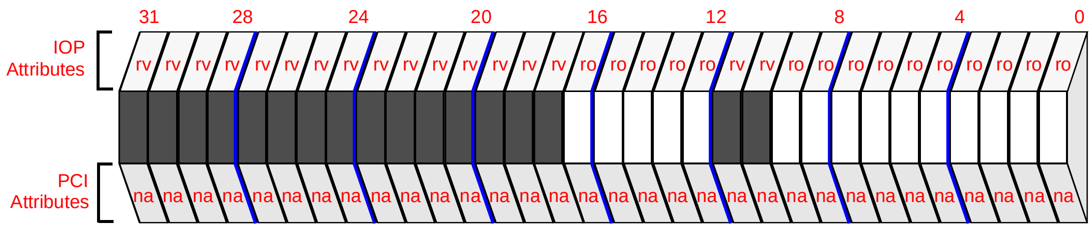
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:21	xH	Reserved. Values read are undefined.
20:19	Dependent on the Product Type.	XSI Bus Frequency Status. Note that the internal bus frequency is set at manufacturing time and is product dependent, which means that not all the frequencies as listed below are supported. Bits Ratio 00_2 2:1 - Indicates that the Internal bus is running at the Core Frequency divided by 2. 01_2 3:1 - Indicates that the Internal bus is running at the Core Frequency divided by 3. 10_2 4:1 - Indicates that the Internal bus is running at the Core Frequency divided by 4. 11_2 Reserved
18:16	Dependent on the Product Type.	Intel XScale® processor Frequency Status. Note that the Intel XScale® processor frequency is set at manufacturing time and is product dependent, which means that not all the frequencies as listed below are supported. Bits Core Frequency 000_2 600 MHz 001_2 667 MHz 010_2 800 MHz 011_2 933 MHz 100_2 1000 MHz 101_2 1200 MHz 110_2 Reserved 111_2 Reserved
15:00	xH	Reserved. Values read are undefined.

9.3.10 External Strap Status Register 0 — ESSTSR0

The External Strap Status Register 0 provides software a way to read the states of the current settings of the straps. Refer to the Clock and Reset Chapter for external strap descriptions.

Table 373. External Strap Status Register 0 — ESSTS0



Intel XScale® processor Local Bus Address offset
+2188H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

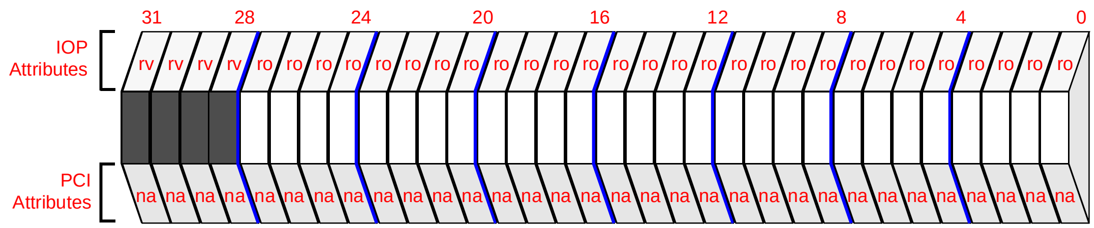
Bit	Default[a]	Description
31:17	0000H	Reserved.
16	x_2	CLK_SRC_PCIE#
15	x_2	INTERFACE_SEL_PCIX#
14	x_2	CONTROLLER_ONLY#
13	x_2	LK_DN_RST_BYPASS#
12	x_2	PCIX_PULLUP#
11:10	xx_2	Reserved.
09:07	xxx_2	DEFSEL[2:0]
06	x_2	BOOT_WIDTH_8#. ***Note:*** This bit reflects the inverted value of the **BOOT_WIDTH_8#** strap.
05:04	xx_2	MEM_FREQ[1:0]
03:00	$xxxx_2$	SMBus Address Straps: Register Bit SMBus Address Bit 0 SMB_A1 1 SMB_A2 2 SMB_A3 3 SMB_A5

a. Default values are product and feature dependent. See bit descriptions for default values.

9.3.11 Unique ID Register 0 — UID0

The Unique ID register 0 represents a 28-bit unique value.

Table 374. Unique ID Register 0 — UID0



Intel XScale® processor Local Bus Address offset
+2194H

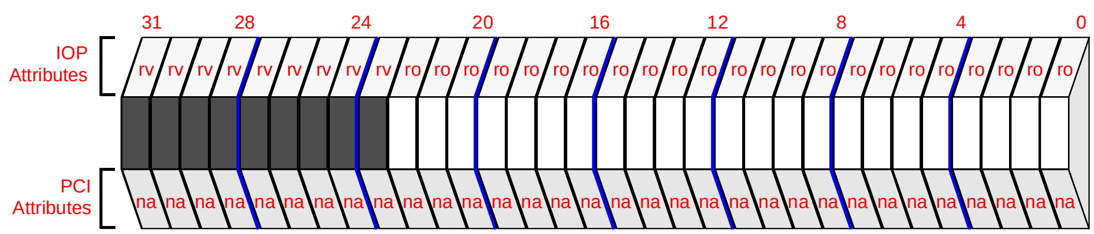
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:28	0H	Reserved.
27:00	xxxx.xxxx.xxxx.xxxx.xxxx.xxxx.xxxx$_2$	Unique ID 0.

9.3.12 Unique ID Register 1 — UID1

The Unique ID register 1 represents a 23-bit unique value.

Table 375. Unique ID Register 1 — UID1



Intel XScale® processor Local Bus Address offset
+2198H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:23	00H	Reserved.
22:00	xxx.xxxx.xxxx.xxxx.xxxx$_2$	Unique ID 1.

10.0 Interrupt Controller Unit

This chapter describes the Intel® 413808 and 413812 I/O Controllers in TPER Mode(4138xx) Interrupt Controller Unit. Operation modes, setup, external memory interface, and implementation of interrupts are described in this chapter.

infrastructure

10.1 Overview

The interrupt control unit manages interrupt routing and interrupt sources to the Intel XScale® processor.

interrupts are events causing a temporary break in program execution so the processor can handle another task. Interrupts commonly request I/O services or synchronize the processor with some external hardware activity. For interrupt handler portability across Intel XScale® microarchitecture family (ARM* architecture compliant), the architecture defines a consistent exception handling mechanism. To manage exceptions which include interrupt requests in parallel with processor execution, the 4138xx provides an on-chip programmable interrupt controller.

Requests for interrupt service come from many sources and are prioritized such that instruction execution is redirected only when an exception interrupt request is of higher priority than that of the executing task. On the 4138xx, interrupt requests may originate from external hardware sources, internal peripherals or software. The 4138xx contains a number of integrated peripherals which may generate interrupts, including the following:

- SRAM DMA Unit
- UART 0 and 1
- Peripheral Bus Interface Unit
- Performance Monitoring Unit
- ATU-E
- TPMI Unit 0
- SRAM DMA
- I^2C Bus Interface Units 0, 1 and 2
- ATU-X
- Messaging Unit
- SRAM Memory Controller Unit
- Timer 0, Timer 1 and Watch Dog Timer[a]

a. Per Intel XScale® processor

The interrupt controller unit can also forward interrupts to the PCI interrupt pins (**P_INT[D:A]#**) when the PCI-X interface is being used as an endpoint. Interrupts are detected with the chip 8-bit interrupt port, an 8-bit GPIO port, and with a dedicated High-Priority Interrupt (**HPI#**) input. Interrupt requests originate from software by the **SWI** instruction. Ultimately, all interrupt sources that are steered to the Intel XScale® processor processor are combined into one of two internal interrupt exceptions: IRQ and FIQ.

10.2 Theory of Operation

10.2.1 Interrupt Controller Unit

The 4138xx Interrupt Controller Unit (ICU) provides the ability to generate interrupts to both the Intel XScale® processor and the PCI interrupt pins.

In addition to the internal peripherals, external devices may also generate interrupts to the Intel XScale® processor. External devices can generate interrupts via the **XINT[15:0]#** pin and the **HPI#** pin. The Interrupt Controller Unit provides the ability to direct PCI interrupts as outputs (**P_INT[D:A]#**) for the PCI-X interface when acting as an endpoint.

The Interrupt Controller Unit has two functions:

- Internal Peripheral Interrupt Control
- External Interrupt Generation

The internal peripheral interrupt control mechanism consolidates a number of interrupt sources for a given peripheral into a single interrupt driven to the Intel XScale® processor. High performance data movement associated interrupts are fully demultiplexed into the ICU, however. In order to provide the executing software with the knowledge of interrupt source, coprocessor mapped status registers describe the source of the active interrupts and the vectors to interrupt handlers for the highest priority active sources. All of the peripheral interrupts are individually enabled from the respective peripheral control registers.

In addition to the interrupts supported by the ICU, the 4138xx provides one non-maskable interrupt per Intel XScale® processor. Non-Maskable Interrupt 0 (**NMI0#**) — is driven to Intel XScale® processor 0, and Non-Maskable Interrupt 1 (**NMI1#**) — is driven to Intel XScale® processor 1. These pins are falling edge triggered input signals. **NMI0#** or **NMI1#** causes an imprecise data abort which is the second highest priority exception (higher than even an FIQ exception). Refer to Table 376, "Exception Priorities And Vectors" on page 568. Since the data abort is imprecise it could potentially occur while in the middle of an abort handler, making it impossible to resume normal operation. When this an error needs to be recoverable the system should route these errors to either IRQ or FIQ by using the external interrupt input pins **XINT[0:15]#**.

10.3 The Intel XScale® Processor Exceptions Architecture

The Intel XScale® processor supports five types of exceptions[16], and a privileged processing mode for each type.

- IRQ and FIQ internal interrupt exceptions (normal and fast interrupts, respectively)
- memory aborts (used to implement memory protection or virtual memory)
- attempted execution of an undefined instruction
- software interrupts (SWIs) (used to make a call to an Operating System)

When an exception occurs, some of the standard registers are replaced with registers specific to the exception mode. All exceptions have replacement (or banked) registers for R14 and R13, and one interrupt mode has more registers for fast interrupt processing.

After an exception, R14 holds the return address for exception processing, which is used both to return after the exception is processed and to address the instruction that caused the exception.

R13 is banked across exception modes to provide each exception handler with a private stack pointer (SP). The fast interrupt mode also banks R8 to R12, so that interrupt processing can begin without the need to save or restore these registers. There is a seventh processing mode, System Mode, that does not have any banked registers (it uses the User mode registers), which is used to run normal (non-exception) tasks that require a privileged processor mode.

10.3.1 CPSR and SPSR

All other processor state is held in status registers. The current operating processor status is in the Current Program Status Register or CPSR. The CPSR holds:

- Four condition code flags (Negative, Zero, Carry and Overflow)
- Two interrupt disable bits (one for each type of interrupt)
- Five bits which encode the current processor mode

All five exception modes also have a Saved Program Status Register (SPSR) which holds the CPSR of the task immediately before the exception occurred. Both the CPSR and SPSR are accessed with special instructions.

10.3.2 The Exception Process

Note: Refer to the System/Software Architecture Specification for details on the exception process for the 4138xx in TPER mode during the common boot phase.

When an exception occurs, the Intel XScale® processor halts execution after the current instruction and begins execution at a fixed address in low memory, known as the exception vectors. There is a separate vector location for each exception (and two for memory aborts to distinguish between data and instruction accesses).

An operating system installs a handler on every exception at initialization. Privileged operating system tasks normally run in System mode to allow exceptions to occur within the operating system without state loss (exceptions overwrite their R14 when an exception occurs, and System mode is the only privileged mode that cannot be entered by an exception).

16.Exception Description from the ARM Architecture Reference Manual, p. 1-3, Copyright Advanced RISC Machines Ltd. (ARM) 1996

10.3.3 Exception Priorities and Vectors

It is important to note that fast interrupt (FIQ) is higher priority than the normal interrupt (IRQ). In addition, while an FIQ exception is executing, the IRQ exception is masked out.

When an exception is taken by the processor, the Program Counter (PC) is loaded with the vector associated with that exception as specified by Table 376.

Generally, the instruction at this location is required to be a branch instruction to the associated exception handler. However, in the case of an FIQ, this is not necessary since the vector location is at the very bottom of all the defined exception vectors, thus the entire FIQ exception handler can be placed at that vector location.

Table 376. Exception Priorities And Vectors

Exception	Priority	Vector[a]
Reset	1 (Highest)	0000 0000H
Data Abort	2	0000 0010H
FIQ	3	0000 001CH
IRQ	4	0000 0018H
Prefetch Abort	5	0000 000CH
Undefined Instructions	6 (Lowest)	0000 0004H
Software Interrupt (SWI)[b]	6 (Lowest)	0000 0008H

a. By enabling the Exception Vector Relocation mode (bit 13, CP15, Register 1), the Vectors (except Reset Vector) can be relocated to be based at FFFF 0000H rather than 0000 0000H. (i.e., FIQ Vector located at FFFF 001CH)
b. Undefined Instruction and SWI can not occur at the same time since SWI is a particular instruction decoding.

10.3.4 Software Requirements For Exception Handling

To use the processor's exception handling facilities, user software must provide the following items in memory:

- Exception Handler Routines
- Software handler to nest certain exceptions (i.e., FIQ and IRQ)

These items are established in memory as part of the initialization procedure.

10.3.4.1 Nesting FIQ and IRQ Exceptions

Hardware does not provide support for nesting of any particular exception, including the FIQ and IRQ exceptions.

In order to provide support for nested interrupts, a software handler must be provided to save the Link Register (R14) and the SPSR (Saved Program Status Register) before reenabling the FIQ or IRQ exception.

10.4 Intel® 413808 and 413812 I/O Controllers in TPER Mode External Interrupt Interface

The interrupt controller attached to the Intel XScale® processor has the facilities necessary to handle all core processor and peripheral internal interrupts as well as the sixteen external interrupts (**XINT[15:0]#**) and a high priority interrupt (**HPI#**).

The 4138xx Primary PCI local bus interface includes four interrupt output signals (**XINT[3:0]#**/**P_INT[D:A]#**). Interrupts from the Messaging Unit and Address Translation Units are routed to these interrupt output signals.

10.4.1 Interrupt Inputs

The 17 external interrupt input pins of the 4138xx have the following definitions:

XINT[3:0]# **External Interrupt (Input)** — These pins (**XINT[3:0]#**) cause interrupts to be requested. Each pin is a level-detect input only. These pins are internally synchronized. These pins act as interrupt inputs and must be unmasked in the INTCTL[3:0] registers. These pins also act as interrupt outputs (**P_INT[D:A]#**) for the PCI-X interface when configured as an endpoint. These pins can also function as general purpose inputs/outputs (**GPIO[11:8]**) when not used as interrupt pins.

XINT[7:4]# **External Interrupt (Input)** — These pins (**XINT[7:4]#**) cause interrupts to be requested. Each pin is a level-detect input only. These pins are internally synchronized. These pins only act as interrupt inputs when they are unmasked in the INTCTL[3:0] registers. These pins can also function as general purpose inputs/outputs (**GPIO[15:12]**) when not used as interrupt pins.

XINT[15:8]# **External Interrupt (Input)** — These pins (**XINT[15:8]#**) cause interrupts to be requested. Each pin is a level-detect input only. These pins are internally synchronized. These pins only act as interrupt inputs when they are configured as general purpose inputs and are unmasked in the INTCTL[3:0] registers. These pins can also function as general purpose inputs/outputs (**GPIO[7:0]**) when not used as interrupt pins.

HPI# **High-Priority Interrupt (Input)** — Causes a high priority interrupt event to occur. The external **HPI#** input requires a level input and is maskable by the INTCTL[3:0] registers. This pin is internally synchronized.

NMI0# **Non-Maskable Interrupt 0 (Input)** — Causes a non-maskable imprecise data abort exception to the Intel XScale® processor 0. The external **NMI0#** input requires a falling edge triggered input. This pin is internally synchronized.

NMI1# **Non-Maskable Interrupt 1 (Input)** — Causes a non-maskable imprecise data abort exception to the Intel XScale® processor 1. The external **NMI1#** input requires a falling edge triggered input. This pin is internally synchronized.

Note: **NMI0#** and **NMI1#** are not implemented as part of the Interrupt Controller Units. They are external pins which are falling edge triggered, internally synchronized and then directly driven to the respectively Intel XScale® processors.

The external interrupt input interface for the 4138xx consists of the pins shown in Table 377.

Table 377. Interrupt Input Pin Descriptions

Signal	Description
P_INTA#/XINT0# /GPIO[8]	This is a bi-directional pin. When 4138xx is setup as an endpoint with the PCI-X interface, this pin acts as an output pin (**P_INTA#**). This pin can act as an input (**XINT0#**) and drive the **XINT0#** input of the Interrupt Controller. The Interrupt Controller Unit input **XINT0#** can be steered to either the FIQ or the IRQ internal interrupt input of the Intel XScale® processor. To enable a given pin as an interrupt into the ICU, it needs to be unmasked in the INTCTL[3:0] register. This pin can also function as a general purpose input/output pin (**GPIO[8]**) when not used as an interrupt pin.
P_INTB#/XINT1# /GPIO[9]	This is a bi-directional pin. When 4138xx is setup as an endpoint with the PCI-X interface, this pin acts as an output pin (**P_INTB#**). This pin can act as an input (**XINT1#**) and drive the **XINT1#** input of the Interrupt Controller. The Interrupt Controller Unit input **XINT1#** can be steered to either the FIQ or the IRQ internal interrupt input of the Intel XScale® processor. To enable a given pin as an interrupt into the ICU, it needs to be unmasked in the INTCTL[3:0] register. This pin can also function as a general purpose input/output pin (**GPIO[9]**) when not used as an interrupt pin.
P_INTC#/XINT2# /GPIO[10]	This is a bi-directional pin. When 4138xx is setup as an endpoint with the PCI-X interface, this pin acts as an output pin (**P_INTC#**). This pin can act as an input (**XINT2#**) and drive the **XINT2#** input of the Interrupt Controller. The Interrupt Controller Unit input **XINT2#** can be steered to either the FIQ or the IRQ internal interrupt input of the Intel XScale® processor. To enable a given pin as an interrupt into the ICU, it needs to be unmasked in the INTCTL[3:0] register. This pin can also function as a general purpose input/output pin (**GPIO[10]**) when not used as an interrupt pin.
P_INTD#/XINT3# /GPIO[11]	This is a bi-directional pin. When 4138xx is setup as an endpoint with the PCI-X interface, this pin acts as an output pin (**P_INTD#**). This pin can act as an input (**XINT3#**) and drive the **XINT3#** input of the Interrupt Controller. The Interrupt Controller Unit input **XINT3#** can be steered to either the FIQ or the IRQ internal interrupt input of the Intel XScale® processor. To enable a given pin as an interrupt into the ICU, it needs to be unmasked in the INTCTL[3:0] register. This pin can also function as a general purpose input/output pin (**GPIO[11]**) when not used as an interrupt pin.
XINT4#/GPIO[12]	This is a bi-directional pin. This pin can act as an input (**XINT4#**) and drive the **XINT4#** input of the Interrupt Controller. The Interrupt Controller Unit input **XINT4#** can be steered to either the FIQ or the FIQ internal interrupt input of the Intel XScale® processor. To enable a given pin as an interrupt into the ICU, it needs to be unmasked in the INTCTL[3:0] register. This pin can also function as a general purpose input/output pin (**GPIO[12]**) when not used as an interrupt pin.
XINT5#/GPIO[13]	This is a bi-directional pin. This pin can act as an input (**XINT5#**) and drive the **XINT5#** input of the Interrupt Controller. The Interrupt Controller Unit input **XINT5#** can be steered to either the FIQ or the IRQ internal interrupt input of the Intel XScale® processor. To enable a given pin as an interrupt into the ICU, it needs to be unmasked in the INTCTL[3:0] register. This pin can also function as a general purpose input/output pin (**GPIO[13]**) when not used as an interrupt pin.
XINT6#/GPIO[14]	This is a bi-directional pin. This pin can act as an input (**XINT6#**) and drives the **XINT6#** input of the Interrupt Controller. The Interrupt Controller Unit input **XINT6#** can be steered to either the FIQ or the IRQ internal interrupt input of the Intel XScale® processor. To enable a given pin as an interrupt into the ICU, it needs to be unmasked in the INTCTL1 register. This pin can also function as a general purpose input/output pin (**GPIO[14]**) when not used as an interrupt pin.
XINT7#/GPIO[15]	This is a bi-directional pin. This pin can act as an input (**XINT7#**) and drives the **XINT7#** input of the Interrupt Controller. The Interrupt Controller Unit input **XINT7#** can be steered to either the FIQ or the IRQ internal interrupt input of the Intel XScale® processor. To enable a given pin as an interrupt into the ICU, it needs to be unmasked in the INTCTL[3:0] register. This pin can also function as a general purpose input/output pin (**GPIO[15]**) when not used as an interrupt pin.
XINT[15:8]#/GPIO[7:0]	These are bi-directional pins. These pins can act as inputs (**XINT[15:8]#**) and drive the **XINT[15:8]#** input of the Interrupt Controller. The Interrupt Controller Unit input **XINT[15:8]#** can be steered to either the FIQ or the IRQ internal interrupt input of the Intel XScale® processor. To enable a given pin as an interrupt into the ICU, it needs to be unmasked in the INTCTL[3:0] register. These pins can also function as general purpose input/output pins (**GPIO[7:0]**) when not used as an interrupt pin.
HPI#	**HPI#** input can be enabled/disabled by the INTCTL[3:0] register, but can be steered to either the FIQ or the IRQ internal interrupt.
NMI0#	**NMI0#** input is a non-maskable falling edge triggered signal and is internally synchronized and directly driven to Intel XScale® processor 0.
NMI1#	**NMI1#** input is a non-maskable falling edge triggered signal and is internally synchronized and directly driven to Intel XScale® processor 1.

10.4.2 Outbound Interrupts

When 4138xx is setup as an endpoint device with the PCI-X interface, the **XINT[3:0]#** pins act as output pins (**P_INT[D:A]#**) respectively. The Messaging Unit (MU) and the TPMI functions have the capability of generating interrupts on the PCI interrupt output pins. The MU has two distinct messaging mechanisms. Each allows a host processor or external PCI agent and the 4138xx to communicate through message passing and interrupt generation. The two mechanisms are:

- **Message Registers** — allow the 4138xx and external PCI agents to communicate by passing messages in one of four 32-bit Message Registers. In this context, a message is any 32-bit data value. Message registers combine aspects of mailbox registers and doorbell registers. Writes to the message registers may optionally cause interrupts.
- **Doorbell Registers** — allow the 4138xx to assert the PCI interrupt signals and allow external PCI agents to generate an interrupt to the Intel XScale® processor.

Both mechanisms can result in Outbound Interrupts to a host processor.

Refer to the Host Interface and TPMI Chapter for detailed descriptions of the TPMI Outbound Interrupts.

The external interrupt output interface for 4138xx consists of the pins shown in Table 378.

Table 378. Interrupt Output Pin Descriptions

Signal	Description
P_INTA#	Primary PCI Interrupt output of 4138xx source from the MU.
P_INTB#	Primary PCI Interrupt output of 4138xx source from the MU.
P_INTC#	Primary PCI Interrupt output of 4138xx source from the MU.
P_INTD#	Primary PCI Interrupt output of 4138xx source from the MU.

10.5 The Intel® 413808 and 413812 I/O Controllers in TPER Mode Interrupt Controller Unit

The 4138xx Interrupt Controller Unit (ICU) provides a flexible, low-latency means for requesting interrupts and minimizing the core's interrupt handling burden.

All interrupt sources are combined into one of the two internal interrupt exceptions: IRQ and FIQ.

The interrupt controller provides the following features for managing hardware-requested interrupts:

- Flexibility to direct interrupt sources to either the FIQ or IRQ internal interrupt exception
- 17 external interrupt pins.
  - One high-priority interrupt pin, **HPI#**.
  - Sixteen Maskable Inputs, **XINT[15:0]#**. Note that when the 4138xx acts as an endpoint with the PCI-X interface, only twelve interrupt inputs (**XINT[15:4]#**) are available instead of sixteen, as the remaining four become outputs (**P_INT[D:A]#**).
- Two internal timers sources.
- Peripheral interrupt sources.
- Two Intel XScale® processor interrupts.

All interrupts are *level sensitive*: interrupt sources must keep asserting the interrupt signal until software causes the source to deassert it.

All interrupt sources are individually maskable with the ICUs Interrupt Control registers.

Additionally, all interrupts may be quickly disabled by altering the F and I bits in the CPSR as specified in the *ARM Architecture Reference Manual*.

When software running on the 4138xx is vectored to an Interrupt Service Routine (ISR), it reads the ICUs IRQ Interrupt Vector Register (IINTVEC) or FIQ Interrupt Vector register (FINTVEC) to quickly retrieve the address for the interrupt handler of the highest priority active interrupt source.

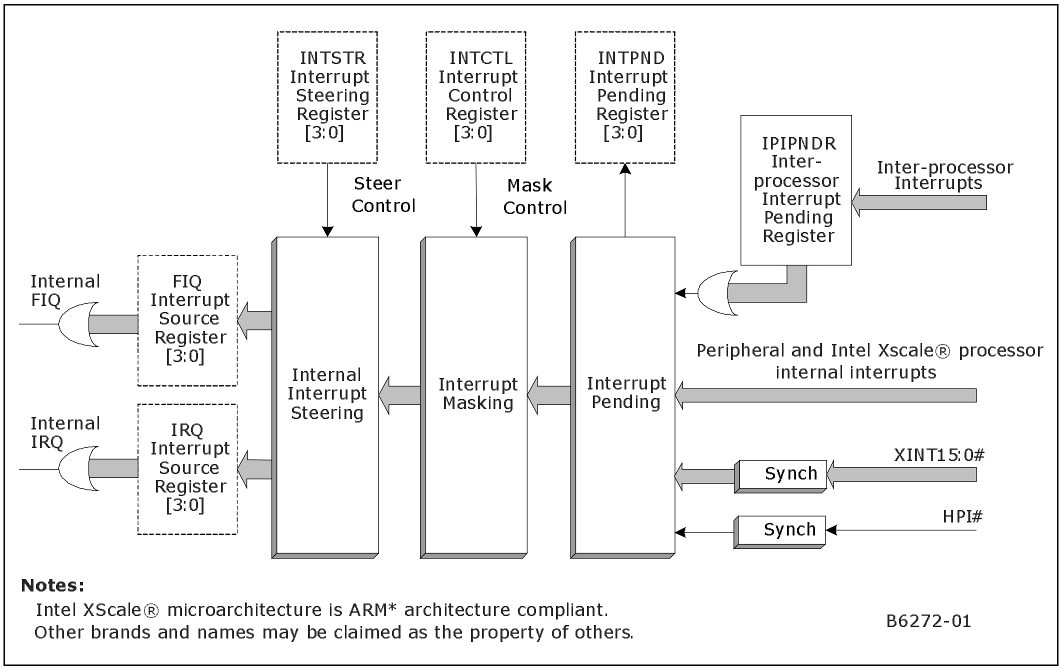
10.5.1 Programmer Model

Software has access to 15 registers in the ICU. These registers control, masking, prioritization, and vector generation for all interrupt sources.

10.5.1.1 Active Interrupt Source Control and Status

The INTCTL[3:0] are used to enable or disable (mask) individual interrupts. As mentioned, masking of all interrupts may still be accomplished via the CPSR register in the core. INTSTR[3:0] are used to direct internal interrupts to either FIQ or IRQ. IINTSRC[3:0] and FINTSRC[3:0] are read-only registers that record all currently active and unmasked interrupt sources; the architecture for the interrupt source registers and FIQ/IRQ generation is illustrated in Figure 68.

Figure 68. Interrupt Controller Block Diagram (Active Interrupt Source Registers)



10.5.1.2 Prioritization and Vector Generation for Active Interrupt Sources

IPR[7:0] registers reserve two bits for each source to assign one of four priority levels.

00_2 – High Priority
01_2 – Medium/High Priority
10_2 – Medium/Low Priority
11_2 – Low Priority

When interrupt vector generation is enabled and there are multiple requests pending either in the FINTSRC[3:0] or the IINTSRC[3:0] registers, the prioritization selects a highest priority active source for each source register.

Note: When multiple interrupts at the same priority level are pending for either FIQ or IRQ, the highest priority active source is selected according to a fixed priority based on bit location. Highest order bit is first.

The INTBASE and INTSIZE registers are used to establish a contiguous Interrupt Service Routine (ISR) memory range for all of 128 possible sources. The architecture provides for an ISR ranging from 4 bytes to 64 Kbytes per source.

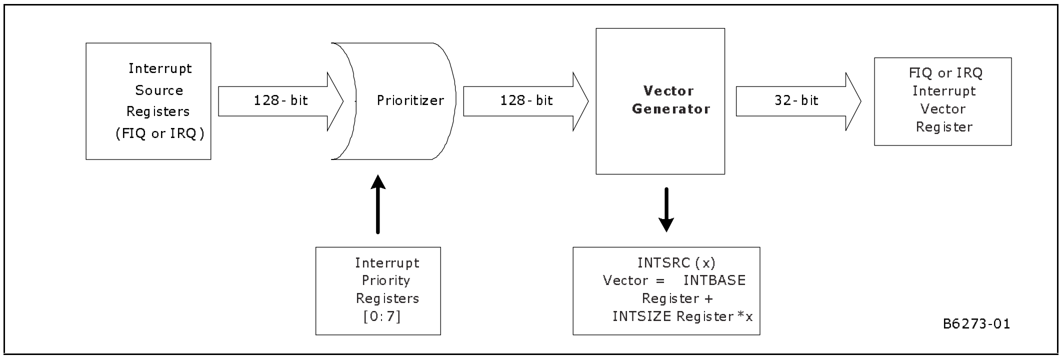
The actual vector value is a function of the INTBASE and the INTSIZE registers and is based on a fixed order of all 128 possible interrupt sources. The vectors begin at INTBASE with source 0 (i.e., IINTSRC0 bit 0), and end at INTBASE + INTSIZE (per source) × 127 with source 127 (that is, IINTSRC3 bit 31).

Example 6. Determining the Location of the Interrupt Handler for Source 25

```
INTBASE = 0x81400000 ; 4 Mbyte Aligned Base Address

INTSIZE = 0xE ; 32 Kbytes per source (ISR Memory Range of 4 Mbytes)

ISR Address(25) = 0x81400000 + conv_hex(2^15*25) = 0x814C8000
```

Based on IINTSRC[3:0], FINTSRC[3:0], IPR[7:0], INTBASE, and INTSIZE, the interrupt controller generates the values provided by the IINTVEC and FINTVEC registers as illustrated in Figure 69. The IINTVEC and FINTVEC registers present the vector for the active interrupt source with the highest priority to the IRQ and FIQ exception handlers, respectively.

Figure 69. Interrupt Controller Block Diagram (FIQ/IRQ Interrupt Vector Generation)



Note: The 4138xx does not use all 128 possible sources.

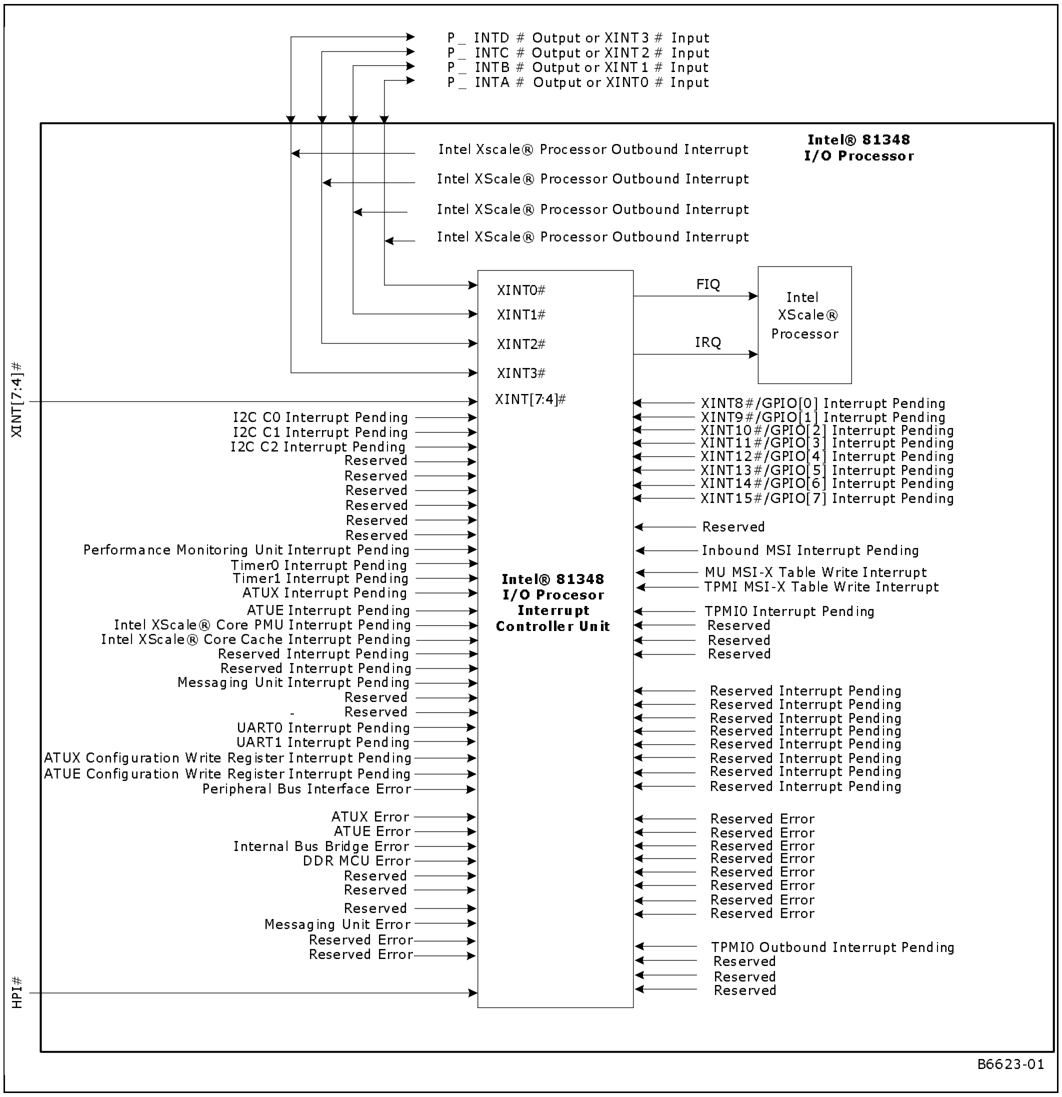
ICU registers reside in Coprocessor 6 (CP6). They may be accessed/manipulated with the MCR, MRC, STC, and LDC instructions. The instruction *CRn* field denotes the accessed register number. The instruction *opcode_1*, *opcode_2*, and *CRm* fields should be zero. Most systems restrict access to CP6 to privileged processes. To control access to CP6, use Coprocessor Access Register.

An instruction that modifies an ICU register is insured to take effect before the next instruction executes. For example, when an instruction masks an interrupt source, subsequent instructions execute in an environment in which the masked interrupt does not occur.

10.5.2 Operational Blocks

The ICU provides the connections to the Intel XScale® processor. These connections are shown in Figure 70.

Figure 70. Intel® 413808 and 413812 I/O Controllers in TPER Mode Interrupt Controller Connections



10.5.3 Intel® 413808 and 413812 I/O Controllers in TPER Mode: Internal Peripheral Interrupt

The 4138xx Interrupt Controller receives inputs from multiple internal interrupt sources. All pending interrupts required during normal operation of the various peripheral units are available in either the IINTSRC[3:0] or FINTSRC[3:0] registers depending on the value in INTSTR[3:0]. To provide the best latency for high performance event driven activities, the Application DMAs interrupts are fully demultiplexed into the interrupt source registers for FIQ and IRQ so that software does not need to access these peripheral units to diagnose the exact source and cause of the interrupt. The IINTSRC[3:0] and the FINTSRC[3:0] registers also include pending interrupts that indicate that an error has occurred in one of the peripheral units. For the interrupts that indicate errors, more detail about the exact cause of the interrupt can be determined by reading the status register of the respective peripheral unit.

10.5.3.1 Normal Interrupt Sources

The 4138xx Interrupt Controller receives normal interrupts from the Application DMA channels, Performance Monitoring Unit, the I^2C Bus Interface Unit, the ATUE, the ATUX, the Programmable Timers, the Messaging Unit and the UARTs. The Application DMA channel interrupts for End of Transfer interrupt or End of Chain interrupt are demultiplexed into the interrupt controller.

A valid interrupt from any of these sources outputs a *level-sensitive* interrupt to the 4138xx Interrupt Controller input. The corresponding IRQ or FIQ interrupt source register bit in the interrupt controller should remain active as long as the interrupt is pending in the peripheral unit. The appropriate interrupt source bit is cleared by clearing the source of the interrupt at the internal peripheral.

The normal interrupt sources which drive the inputs to the 4138xx Interrupt Controller are detailed in Table 379.

Note: The UART and I^2C Bus Interface Unit interrupt sources are combined as a single interrupt, and include both normal and error conditions within the respective units.

Table 379. Normal Interrupt Sources

Unit	Register	Interrupt Condition
Intel XScale® processor	Overflow Flag Status Register (FLAG)	Counter Overflow
ATU -X & -E	ATU -X & -E Interrupt Status Register	ATU BIST Start
	Configure Reg Write	Any of the ATU Configuration registers written by an inbound Configuration Write cycle
I^2C Bus Interface Unit 2-0	I^2C Status Register 2-0	Receive Buffer Full
		Transmit Buffer Empty
		Slave Address Detect (General Call Address Detect)
		STOP Detected
		Bus Error Detected
		Arbitration Lost Detected
Messaging Unit	Inbound Interrupt Status Register	Inbound Doorbell Interrupt
		Inbound Message 1 Interrupt
		Inbound Message 0 Interrupt
		MU MSI-X Table Write Interrupt
		Selective Reset Interrupt
		Coordinated Reset Interrupt
Timer 1 & 0	Timer Mode Register 1 & 0	Timer 0 has decremented to 0 interrupt.
UART Unit 1 & 0	UART 1 & 0 Interrupt ID Register	Received Line Status
		Received Data is Available
		Character Time-out Indications
		Transmit FIFO Data Request
		Autobaud Lock Indication

10.5.3.2 Error Interrupt Sources

The 4138xx Interrupt Controller receives error interrupts from the ATUs, the Messaging Unit. Each of these interrupts represent an error condition in the peripheral unit. Refer to the appropriate units for more details.

A valid interrupt from any of these sources, outputs a *level-sensitive* interrupt to the 4138xx Interrupt Controller input. The corresponding FIQ or IRQ interrupt source register bit in the interrupt controller remains active as long as the interrupt is pending in the peripheral unit. The appropriate FIQ or IRQ interrupt source bit is cleared by clearing the source of the interrupt at the internal peripheral.

Table 380. Error Interrupt Sources

Unit	Register	Error Condition
Intel XScale® processor	L2 Cache/BIU Error Logging Register (ERRLOG)	L2 cache single bit ECC error.
ATU -X & -E	ATU -X & -E Interrupt Status Register	ATU Vital Product Data Address Updated
		ATU Inbound Memory Window 1 Base Updated
		Initiated Split Completion Error Message
		Received Split Completion Error Message
		Power State Transition
		P_SERR# Asserted
		PCI Detected Parity Error
		ATU BIST Interrupt
		IB Master Abort
		P_SERR# Detected
		PCI Master Abort
		PCI Target Abort (master)
		PCI Target Abort (target)
		PCI Master Parity Error
Messaging Unit	Inbound Interrupt Status Register	Error Doorbell Interrupt
Watch Dog Time	Timer Interrupt Status Register	Timer Expiration

10.5.4 High-Priority Interrupt (HPI#)

The **HPI#** pin generates an interrupt for implementation of critical interrupt routines.

10.5.5 Timer Interrupts

Each of the two timer units has an associated interrupt. Timer interrupts are connected directly to the 4138xx interrupt controller and are posted in either the IINTSRC[3:0] or FINTSRC[3:0] registers. These interrupts are set up through the timer control registers described in Chapter 11.0, "Timers."

10.5.6 Inter-Processor Interrupts

Note: IPIs are not supported on 4138xx.

10.5.7 Intel XScale® Processor Interrupts

The Intel XScale® processor can generate two type of interrupts that are routed from the core as outputs and into the 4138xx ICU. This mechanism allows these two core interrupts to be handled like any other peripheral interrupts by the ICU. For example, these interrupts can be masked when desired using the INTCTLx registers and steered to either IRQ or FIQ using the INTSTRx registers. The Intel XScale® processor PMU interrupt is generated when the Intel XScale® processor PMU detects an overflow of one of its counters. The Intel XScale® processor Cache interrupt is generated when the Intel XScale® processor L2 cache detects a single bit ECC error. This interrupt can be used by software wishing to scrub memory when a single-bit ECC error is detected. Refer to the Intel XScale® processor External Architecture Specification for more detailed descriptions on these interrupts.

10.5.8 Software Interrupts

The application program may use the **SWI** instruction to request interrupt service.

10.6 Default Status

The interrupt logic is reset by the PCI reset signal or through software. Table 381 shows the power-up and reset values.

Table 381. Default Interrupt Routing and Status Values

Register	Default Value	Description
INTCTL0	0000 0000H	All interrupts 31:0 masked.
INTCTL1	0000 0000H	All interrupts 63:32 masked.
INTCTL2	0000 0000H	All interrupts 95:64 masked.
INTCTL3	0000 0000H	All interrupts 127:96 masked.
INTSTR0	0000 0000H	All interrupts 31:0 steered to IRQ.
INTSTR1	0000 0000H	All interrupts 63:32 steered to IRQ.
INTSTR2	0000 0000H	All interrupts 95:64 steered to IRQ.
INTSTR3	0000 0000H	All interrupts 127:96 steered to IRQ.
IINTSRC0	0000 0000H	All IRQ interrupts 31:0 inactive.
IINTSRC1	0000 0000H	All IRQ interrupts 63:32 inactive.
IINTSRC2	0000 0000H	All IRQ interrupts 95:64 inactive.
IINTSRC3	0000 0000H	All IRQ interrupts 127:96 inactive.
FINTSRC0	0000 0000H	All FIQ interrupts 31:0 inactive.
FINTSRC1	0000 0000H	All FIQ interrupts 63:32 inactive.
FINTSRC2	0000 0000H	All FIQ interrupts 95:64 inactive.
FINTSRC3	0000 0000H	All FIQ interrupts 127:96 inactive.
IPR0	0000 0000H	All interrupts 15:0 at Priority 0.
IPR1	0000 0000H	All interrupts 32:16 at Priority 0.
IPR2	0000 0000H	All interrupts 47:33 at Priority 0.
IPR3	0000 0000H	All interrupts 63:48 at Priority 0.
IPR4	0000 0000H	All interrupts 79:64 at Priority 0.
IPR5	0000 0000H	All interrupts 95:80 at Priority 0.
IPR6	0000 0000H	All interrupts 111:96 at Priority 0.
IPR7	0000 0000H	All interrupts 127:112 at Priority 0.

Note: For the default value of a register that is not listed in this table refer to the corresponding register section.

10.7 Interrupt Control Unit Registers

All Interrupt Controller registers are visible as 4138xx memory mapped registers and can be accessed through the internal memory bus. Each is a 32-bit register and is memory-mapped in the Intel XScale® processor memory space. The programmer interface to the interrupt controller is through the coprocessor registers. Table 382 describes these registers.

The coprocessor registers may be accessed/manipulated with the MCR, MRC, STC, and LDC instructions. The *CRn* field of the instruction denotes the register number to be accessed. The *opcode_1*, *opcode_2*, and *CRm* fields of the instruction should be zero. Most systems restrict access to CP6 to privileged processes. To control access to CP6, use the Coprocessor Access Register.

Table 382. Interrupt Controller Co-Processor Register Addresses (Sheet 1 of 2)

Register Name	Description	Coprocessor CP6 (CR_m Field)	Register (CR_n Field) or MMR Address
INTBASE	Interrupt Base Register	2	Register 0
Reserved	Reserved		Register 1
INTSIZE	Interrupt Size Register		Register 2
IINTVEC	IRQ Interrupt Vector Register		Register 3
FINTVEC	FIQ Interrupt Vector Register		Register 4
IPIPNDR	Reserved		Register 8
Reserved	Reserved		Register 9
INTPND0	Interrupt Pending Register 0	3	Register 0
INTPND1	Interrupt Pending Register 1		Register 1
INTPND2	Interrupt Pending Register 2		Register 2
INTPND3	Interrupt Pending Register 3		Register 3
INTCTL0	Interrupt Control Register 0	4	Register 0
INTCTL1	Interrupt Control Register 1		Register 1
INTCTL2	Interrupt Control Register 2		Register 2
INTCTL3	Interrupt Control Register 3		Register 3
INTSTR0	Interrupt Steering Register 0	5	Register 0
INTSTR1	Interrupt Steering Register 1		Register 1
INTSTR2	Interrupt Steering Register 2		Register 2
INTSTR3	Interrupt Steering Register 3		Register 3
IINTSRC0	IRQ Interrupt Source Register 0	6	Register 0
IINTSRC1	IRQ Interrupt Source Register 1		Register 1
IINTSRC2	IRQ Interrupt Source Register 2		Register 2
IINTSRC3	IRQ Interrupt Source Register 3		Register 3
FINTSRC0	FIQ Interrupt Source Register 0	7	Register 0
FINTSRC1	FIQ Interrupt Source Register 1		Register 1
FINTSRC2	FIQ Interrupt Source Register 2		Register 2
FINTSRC3	FIQ Interrupt Source Register 3		Register 3

Table 382. Interrupt Controller Co-Processor Register Addresses (Sheet 2 of 2)

Register Name	Description	Coprocessor CP6 (CR_m Field)	Register (CR_n Field) or MMR Address
IPR0	Interrupt Priority Register 0	8	Register 0
IPR1	Interrupt Priority Register 1		Register 1
IPR2	Interrupt Priority Register 2		Register 2
IPR3	Interrupt Priority Register 3		Register 3
IPR4	Interrupt Priority Register 4		Register 4
IPR5	Interrupt Priority Register 5		Register 5
IPR6	Interrupt Priority Register 6		Register 6
IPR7	Interrupt Priority Register 7		Register 7
		9[a]	Register 0:15

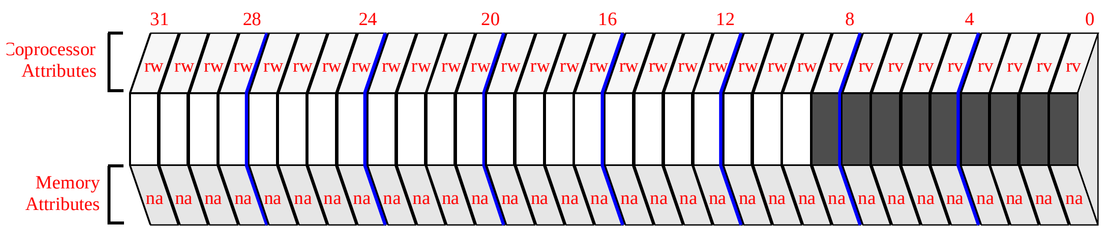
a. CP6 CRm = 9H is used by the Timers.

10.7.1 Interrupt Base Register — INTBASE

The Interrupt Base Register indicates the beginning of the Interrupt Service Routine (ISR) memory range that contains the interrupt service routines for up to 128 sources. The starting address must be on a boundary equal to the granularity of the ISR memory range as specified by the INTSIZE registers.

For instance, the upper 23 bits are used for a 512 Byte ISR memory range, and the upper 16 bits for a 64 KByte ISR memory range., etc.

Table 383. Interrupt Base Register — INTBASE



Intel XScale® processor Coprocessor address
CP6, Page 2, Register 0

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

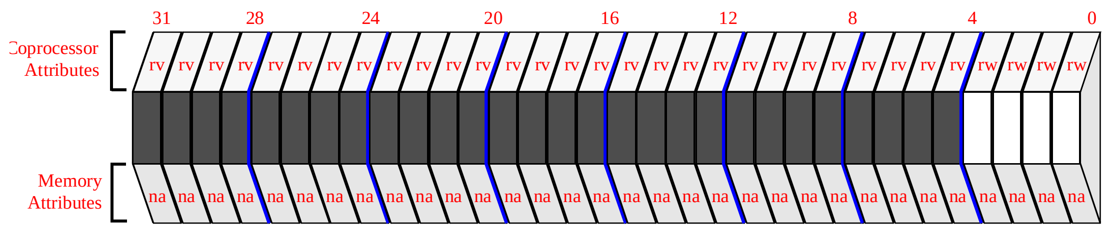
Bit	Default	Description
31:09	0000 00H	Interrupt Base — These bits define the upper 23 bits of the base address for the ISR memory range.
08:00	00H	Reserved

10.7.2 Interrupt Size Register — INTSIZE

The Interrupt Size Register indicates the size of the Interrupt Service Routine (ISR) memory range that contains the interrupt service routines for up to 128 sources. The INTSIZE register can allocate from 4 bytes to 64 Kbytes of memory address space for the ISR per source. This means that the INTSIZE register can allocate a total ISR memory space that ranges in size from 512 bytes to 8 Mbytes.

Along with the starting address defined in the INTBASE register, the INTSIZE register fully specifies the ISR memory range.

Table 384. Interrupt Size Register — INTSIZE



Intel XScale® processor Coprocessor address
CP6, Page 2, Register 2

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:04	0000000H	Reserved
03:00	0H	ISR Memory Range Size These bits define the size of the ISR memory range: INTSIZE / ISR Range Size / ISR Size (per Source) 0 (Disabled) 1 512 bytes 4 bytes 2 1 Kbytes 8 bytes 3 2 Kbytes 16 bytes 4 4 Kbytes 32 bytes 5 8 Kbytes 64 bytes 6 16 Kbytes 128 bytes 7 32 Kbytes 256 bytes 8 64 Kbytes 512 bytes 9 128 Kbytes 1 Kbytes 10 256 Kbytes 2 Kbytes 11 512 Kbytes 4 Kbytes 12 1 Mbytes 8 Kbytes 13 2 Mbytes 16 Kbytes 14 4 Mbytes 32 Kbytes 15 8 Mbytes 64 Kbytes

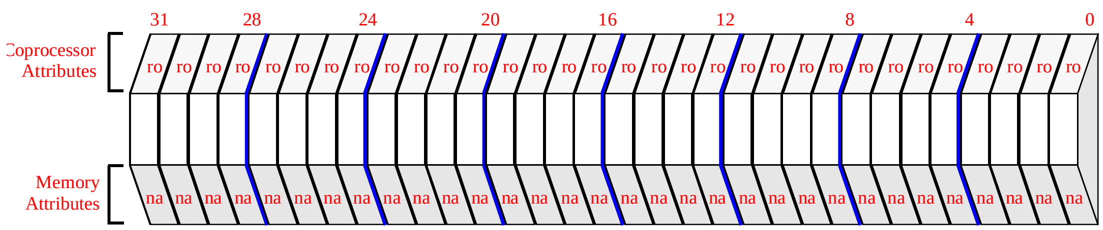
10.7.3 IRQ Interrupt Vector Register — IINTVEC

The IRQ Interrupt Vector Register is a 32-bit Coprocessor 6 control register. Following an IRQ exception, the IRQ interrupt service routine reads the 32-bit vector to the ISR for the active IRQ source with the highest priority.

The actual vector value is a function of the INTBASE and the INTSIZE registers and is based on a fixed order of all 128 possible interrupt sources. The vectors begin at INTBASE with source 0 (i.e., IINTSRC0 bit 0), and end at INTBASE + INTSIZE (per source)*127 with source 127 (that is, IINTSRC3 bit 31).

Before returning to User Mode from Interrupt Mode, the software reads the IINTVEC register and process any lower priority IRQ sources that are active. When there are no longer any active IRQ sources, a read from the IINTVEC register returns FFFF FFFFH.

Table 385. IRQ Interrupt Vector Register- IINTVEC



Intel XScale® processor Coprocessor address
CP6, Page 2, Register 3

Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
31:00	00000000H	IRQ Interrupt Vector—Vector to the highest priority active IRQ source. This register reads FFFF FFFFH when there are no active IRQ sources.

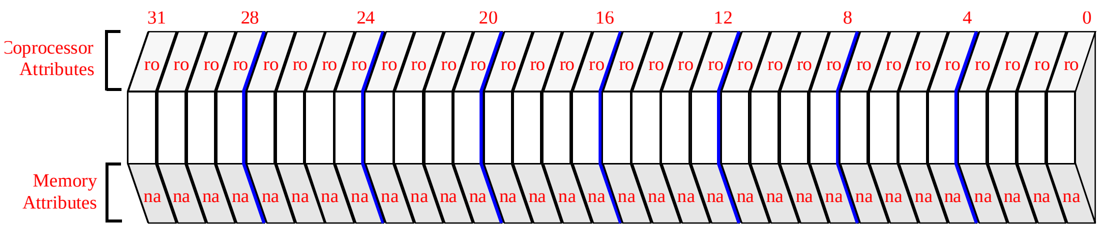
10.7.4 FIQ Interrupt Vector Register — FINTVEC

The FIQ Interrupt Vector Register is a 32-bit Coprocessor 6 control register. Following an FIQ exception, the FIQ interrupt service routine reads the 32-bit vector to the ISR for the active FIQ source with the highest priority.

The actual vector value is a function of the INTBASE and the INTSIZE registers and is based on a fixed order of all 128 possible interrupt sources. The vectors begin at INTBASE with source 0 (i.e., FINTSRC0 bit 0), and end at INTBASE + INTSIZE (per source)*127 with source 127 (that is, FINTSRC3 bit 31).

Before returning to User Mode from Interrupt Mode, the software reads the FINTVEC register and process any lower priority FIQ sources that are active. When there are no longer any active FIQ sources, a read from the FINTVEC register returns FFFF FFFFH.

Table 386. FIQ Interrupt Vector Register- FINTVEC



Intel XScale® processor Coprocessor address
CP6, Page 2, Register 4

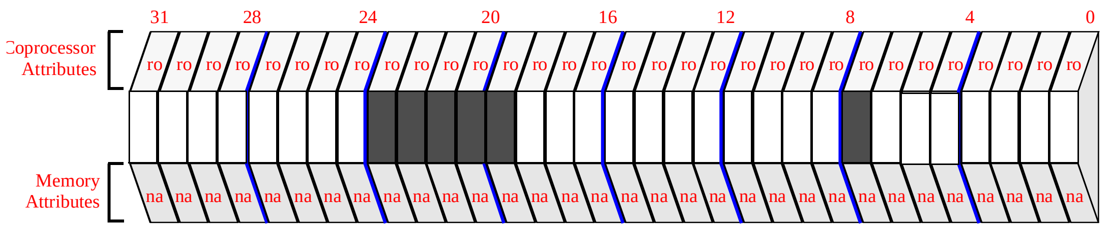
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:00	00000000H	FIQ Interrupt Vector—Vector to the highest priority active FIQ source. This register reads FFFF FFFFH when there are no active FIQ sources.

10.7.5 Interrupt Pending Register 0 — INTPND0

The Interrupt Pending register 0 is a 32-bit Coprocessor 6 control register that can be used to verify pending interrupts. Software can use this registers to poll interrupts as this register is located before the INTCTL0 mask Register.

Table 387. Interrupt Pending Register 0 — INTPND0



Intel XScale® processor Coprocessor address
CP6, Page 3, Register 0

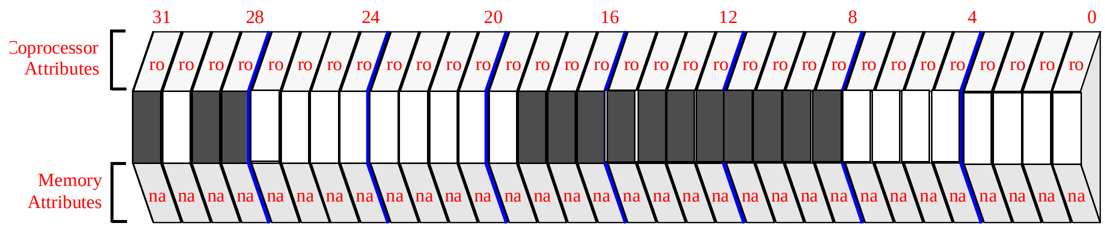
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31	0_2	**XINT7#** Interrupt Pending.
30	0_2	**XINT6#** Interrupt Pending.
29	0_2	**XINT5#** Interrupt Pending.
28	0_2	**XINT4#** Interrupt Pending.
27	0_2	**XINT3#** Interrupt Pending.
26	0_2	**XINT2#** Interrupt Pending.
25	0_2	**XINT1#** Interrupt Pending.
24	0_2	**XINT0#** Interrupt Pending.
23:19	00H	Reserved.
18	0_2	Intel XScale® Processor Cache Interrupt Pending.
17	0_2	Intel XScale® Processor PMU Interrupt Pending.
16	0_2	Peripheral Performance Monitor Interrupt Pending.
15	0_2	ATU-X/Start BIST Interrupt Pending.
14	0_2	ATU-E Inbound Message Interrupt.
13	0_2	Reserved.
12	0_2	Messaging Unit Interrupt Pending.
11	0_2	I^2C Bus Interface 1 Interrupt Pending.
10	0_2	I^2C Bus Interface 0 Interrupt Pending.
9	0_2	Timer 1 Interrupt Pending.
8	0_2	Timer 0 Interrupt Pending.
7	0_2	Reserved.
6	0_2	Watch Dog Timer Interrupt Pending.
5	0_2	Reserved.
4	0_2	Reserved.
3	0_2	Reserved.
2	0_2	Reserved.
1	0_2	Reserved.
0	0_2	Reserved.

10.7.6 Interrupt Pending Register 1 — INTPND1

The Interrupt Pending register 0 is a 32-bit Coprocessor 6 control register that can be used to verify pending interrupts. Software can use this registers to poll interrupts as this register is located before the INTCTL1 mask Register.

Table 388. Interrupt Pending Register 1 — INTPND1



Intel XScale® processor Coprocessor address
CP6, Page 3, Register 1

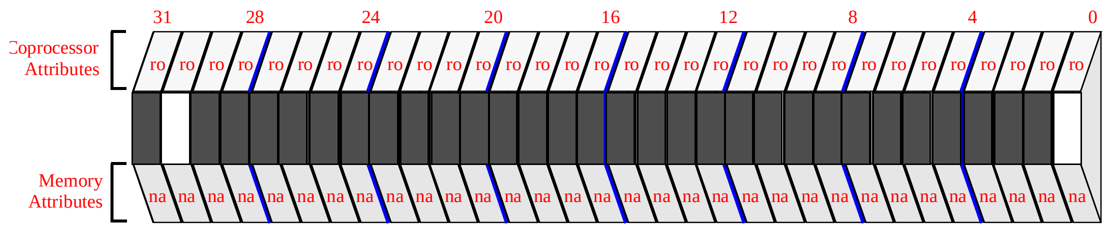
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31	0_2	Reserved.
30	0_2	Messaging Unit Error Interrupt Pending.
29	0_2	Reserved.
28	0_2	Reserved.
27	0_2	Reserved.
26	0_2	Reserved.
25	0_2	Reserved.
24	0_2	Reserved.
23	0_2	ATU-X Error Interrupt Pending.
22	0_2	ATU-X Configuration Register Write Interrupt Pending.
21	0_2	Peripheral Bus Unit Error Interrupt Pending.
20	0_2	UART 1 Interrupt Pending.
19	0_2	UART 0 Interrupt Pending.
18:08	0000H	Reserved.
07	0_2	**XINT15#** Interrupt Pending. Source of this interrupt is the **GPIO[7]** pin.
06	0_2	**XINT14#** Interrupt Pending. Source of this interrupt is the **GPIO[6]** pin.
05	0_2	**XINT13#** Interrupt Pending. Source of this interrupt is the **GPIO[5]** pin.
04	0_2	**XINT12#** Interrupt Pending. Source of this interrupt is the **GPIO[4]** pin.
03	0_2	**XINT11#** Interrupt Pending. Source of this interrupt is the **GPIO[3]** pin.
02	0_2	**XINT10#** Interrupt Pending. Source of this interrupt is the **GPIO[2]** pin.
01	0_2	**XINT9#** Interrupt Pending. Source of this interrupt is the **GPIO[1]** pin.
00	0_2	**XINT8#** Interrupt Pending. Source of this interrupt is the **GPIO[0]** pin.

10.7.7 Interrupt Pending Register 2 — INTPND2

The Interrupt Pending register 0 is a 32-bit Coprocessor 6 control register that can be used to verify pending interrupts. Software can use this registers to poll interrupts as this register is located before the INTCTL2 mask Register.

Table 389. Interrupt Pending Register 2 — INTPND2



Intel XScale® processor Coprocessor address
CP6, Page 3, Register 2

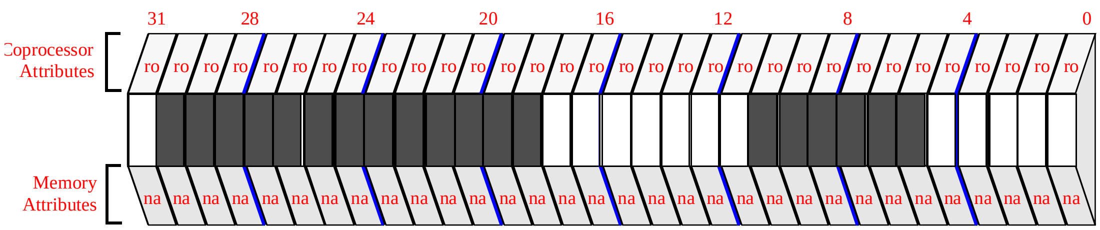
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31	0_2	SRAM Memory Controller Unit Error Interrupt Pending.
30	0_2	South Internal Bus Bridge Error Interrupt Pending.
29	0_2	Reserved.
28	0_2	Reserved.
27	0_2	Reserved.
26	0_2	Reserved.
25	0_2	Reserved.
24	0_2	Reserved.
23	0_2	Reserved.
22	0_2	Reserved.
21	0_2	Reserved.
20	0_2	Reserved.
19	0_2	Reserved.
18	0_2	Reserved.
17	0_2	Reserved.
16	0_2	Reserved.
15	0_2	Reserved.
14	0_2	Reserved.
13	0_2	SRAM DMA Error Interrupt Pending.
12	0_2	SRAM DMA Normal Interrupt Pending.
11	0_2	Reserved.
10	0_2	Reserved.
09	0_2	Reserved.
08	0_2	Reserved.
07	0_2	Reserved.
06	0_2	Reserved.
05	0_2	Reserved.
04	0_2	Reserved.
03	0_2	Reserved.
02	0_2	Reserved.
01	0_2	Reserved.
00	0_2	Reserved.

10.7.8 Interrupt Pending Register 3 — INTPND3

The Interrupt Pending register 3 is a 32-bit Coprocessor 6 control register that can be used to verify pending interrupts. Software can use this registers to poll interrupts as this register is located before the INTCTL3 mask Register.

Table 390. Interrupt Pending Register 3 — INTPND3



Intel XScale® processor Coprocessor address
CP6, Page 3, Register 3

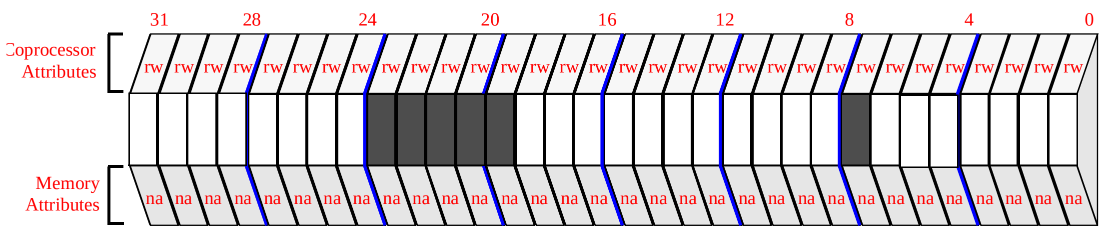
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31	0_2	HPI Interrupt Pending
30:18	0000H	Reserved.
17	0_2	Inbound MSI Interrupt Pending.
16	0_2	Reserved.
15	0_2	MU MSI-X Table Write Interrupt Pending
14	0_2	ATUE Interrupt Message D Pending
13	0_2	ATUE Interrupt Message C Pending
12	0_2	ATUE Interrupt Message B Pending
11	0_2	ATUE Interrupt Message A Pending
10:08	0_2	Reserved.
07	0_2	TPMI 0 Outbound Interrupt Pending.
06:05	0_2	Reserved.
04	0_2	Reserved.
03	0_2	ATU-E Error Interrupt Pending.
02	0_2	ATU-E Configuration Register Write Interrupt Pending.
01	0_2	ATU-E/Start BIST Interrupt Pending.
00	0_2	I²C Bus Interface 2 Interrupt Pending.

10.7.9 Interrupt Control Register 0 — INTCTL0

The Interrupt Control register 0 is a 32-bit Coprocessor 6 control register used to specify which of 32 interrupts are masked.

Table 391. Interrupt Control Register 0 — INTCTL0 (Sheet 1 of 2)

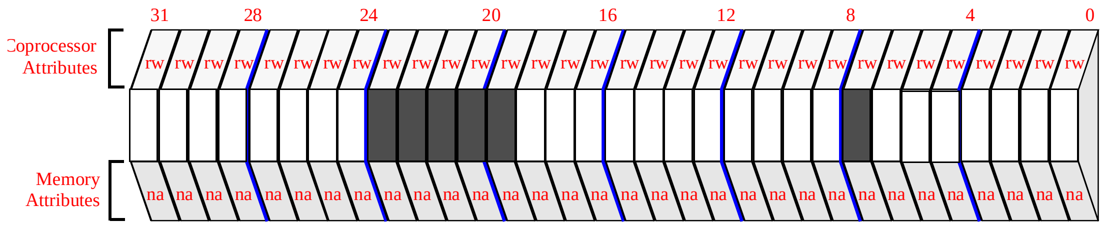


Intel XScale® processor Coprocessor address
CP6, Page 4, Register 0

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31	0_2	**XINT7#** Interrupt Mask 0 = Masked 1 = Not Masked
30	0_2	Reserved.
29	0_2	**XINT5#** Interrupt Mask 0 = Masked 1 = Not Masked
28	0_2	**XINT4#** Interrupt Mask 0 = Masked 1 = Not Masked
27	0_2	**XINT3#** Interrupt Mask 0 = Masked 1 = Not Masked
26	0_2	**XINT2#** Interrupt Mask 0 = Masked 1 = Not Masked
25	0_2	**XINT1#** Interrupt Mask 0 = Masked 1 = Not Masked
24	0_2	**XINT0#** Interrupt Mask 0 = Masked 1 = Not Masked
23:19	0_2	Reserved.
18	0_2	Intel XScale® Processor Cache Interrupt Mask 0 = Masked 1 = Not Masked
17	0_2	Intel XScale® Processor PMU Interrupt Mask 0 = Masked 1 = Not Masked
16	0_2	Peripheral Performance Monitor Interrupt Mask 0 = Masked 1 = Not Masked
15	0_2	ATU/Start BIST Interrupt Mask 0 = Masked 1 = Not Masked

Table 391. Interrupt Control Register 0 — INTCTL0 (Sheet 2 of 2)



Intel XScale® processor Coprocessor address
CP6, Page 4, Register 0

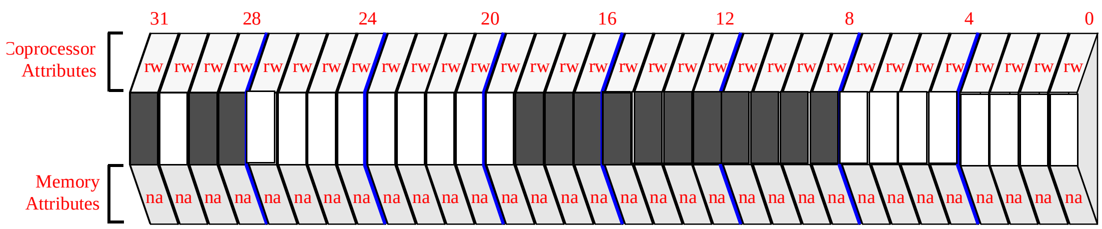
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
14	0_2	ATU-E Inbound Message Interrupt Mask 0 = Masked 1 = Not Masked
13	0_2	Messaging Unit Inbound Post Queue Interrupt Mask 0 = Masked 1 = Not Masked
12	0_2	Messaging Unit Interrupt Mask 0 = Masked 1 = Not Masked 1 =
11	0_2	I²C Bus Interface 1 Interrupt Mask 0 = Masked 1 = Not Masked
10	0_2	I²C Bus Interface 0 Interrupt Mask 0 = Masked 1 = Not Masked
9	0_2	Timer 1 Interrupt Mask 0 = Masked 1 = Not Masked
8	0_2	Timer 0 Interrupt Mask 0 = Masked 1 = Not Masked
7	0_2	Reserved.
6	0_2	Watch Dog Timer Interrupt Mask 0 = Masked 1 = Not Masked
5	0_2	Reserved.
4	0_2	Reserved.
3	0_2	Reserved.
2	0_2	Reserved.
1	0_2	Reserved.
0	0_2	Reserved.

10.7.10 Interrupt Control Register 1 — INTCTL1

The Interrupt Control register 1 is a 32-bit Coprocessor 6 control register used to specify which of 32 interrupts are masked.

Table 392. Interrupt Control Register 1 — INTCTL1 (Sheet 1 of 2)

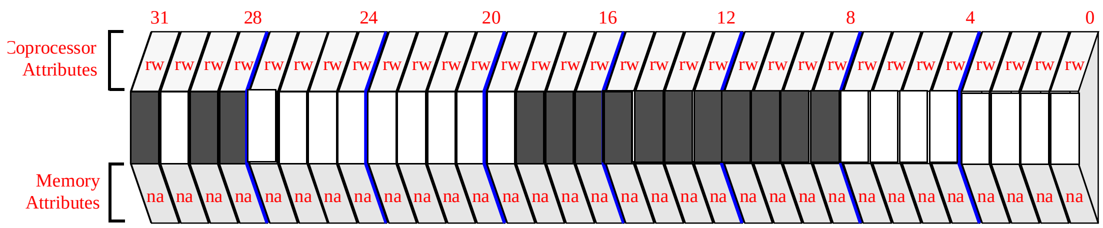


Intel XScale® processor Coprocessor address
CP6, Page 4, Register 1

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31	0_2	Reserved.
30	0_2	Messaging Unit Error Interrupt Mask 0 = Masked 1 = Not Masked
29:28	0_2	Reserved.
27	0_2	Reserved.
26	0_2	Reserved.
25	0_2	Reserved.
24	0_2	Memory Controller Unit Error Interrupt Mask 0 = Masked 1 = Not Masked
23	0_2	ATU Error Interrupt Mask 0 = Masked 1 = Not Masked
22	0_2	ATU Configuration Register Write Interrupt Mask 0 = Masked 1 = Not Masked
21	0_2	Peripheral Bus Unit Error Interrupt Mask 0 = Masked 1 = Not Masked
20	0_2	UART 1 Interrupt Mask 0 = Masked 1 = Not Masked
19	0_2	UART 0 Interrupt Mask 0 = Masked 1 = Not Masked
18:08	000H	Reserved.
07	0_2	**XINT15#** Interrupt Mask. Source of this interrupt is the **GPIO[7]** pin. 0 = Masked 1 = Not Masked
06	0_2	**XINT14#** Interrupt Mask. Source of this interrupt is the **GPIO[6]** pin. 0 = Masked 1 = Not Masked

Table 392. Interrupt Control Register 1 — INTCTL1 (Sheet 2 of 2)



Intel XScale® processor Coprocessor address
CP6, Page 4, Register 1

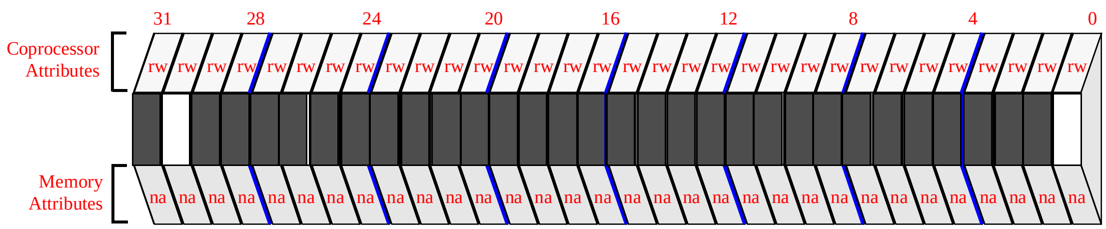
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
05	0_2	**XINT13#** Interrupt Mask. Source of this interrupt is the **GPIO[5]** pin. 0 = Masked 1 = Not Masked
04	0_2	**XINT12#** Interrupt Mask. Source of this interrupt is the **GPIO[4]** pin. 0 = Masked 1 = Not Masked
03	0_2	**XINT11#** Interrupt Mask. Source of this interrupt is the **GPIO[3]** pin. 0 = Masked 1 = Not Masked
02	0_2	**XINT10#** Interrupt Mask. Source of this interrupt is the **GPIO[2]** pin. 0 = Masked 1 = Not Masked
01	0_2	**XINT9#** Interrupt Mask. Source of this interrupt is the **GPIO[1]** pin. 0 = Masked 1 = Not Masked
00	0_2	**XINT8#** Interrupt Mask. Source of this interrupt is the **GPIO[0]** pin. 0 = Masked 1 = Not Masked

10.7.11 Interrupt Control Register 2 — INTCTL2

The Interrupt Control register 2 is a 32-bit Coprocessor 6 control register used to specify which of 32 interrupts are masked.

Table 393. Interrupt Control Register 2 — INTCTL2



Intel XScale® processor Coprocessor address
CP6, Page 4, Register 2

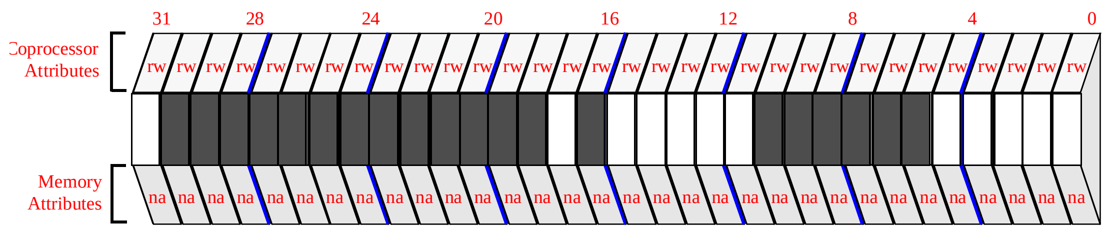
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31	0_2	SRAM Memory Controller Unit Error Interrupt Mask. 0 = Masked 1 = Not Masked
30	0_2	South Internal Bus Bridge Error Interrupt Mask. 0 = Masked 1 = Not Masked
29:14	0_2	Reserved.
13	0_2	SRAM DMA Error Interrupt Mask. 0 = Masked 1 = Not Masked
12	0_2	SRAM DMA Normal Interrupt Mask. 0 = Masked 1 = Not Masked
11:03	0_2	Reserved.
02	0_2	Reserved.
01	0_2	Reserved.
00	0_2	Reserved.

10.7.12 Interrupt Control Register 3 — INTCTL3

The Interrupt Control register 3 is a 32-bit Coprocessor 6 control register used to specify which of 32 interrupts are masked.

Table 394. Interrupt Control Register 3 — INTCTL3 (Sheet 1 of 2)

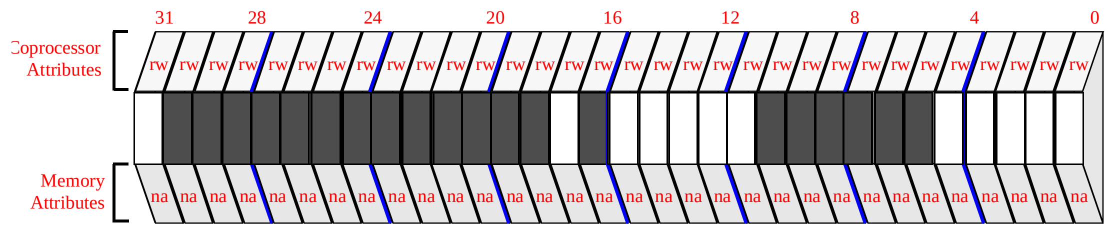


Intel XScale® processor Coprocessor address
CP6, Page 4, Register 3

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31	1_2	HPI Interrupt Pending 0 = Masked 1 = Not Masked
30:18	0000H	Reserved.
17	0_2	Inbound MSI Interrupt Pending 0 = Masked 1 = Not Masked
16	0_2	Reserved.
15	0_2	MU MSI-X Table Write Interrupt Pending 0 = Masked 1 = Not Masked
14	0_2	ATUE Interrupt Message D Pending. 0 = Masked 1 = Not Masked
13	0_2	ATUE Interrupt Message C Pending. 0 = Masked 1 = Not Masked
12	0_2	ATUE Interrupt Message B Pending. 0 = Masked 1 = Not Masked
11	0_2	ATUE Interrupt Message A Pending. 0 = Masked 1 = Not Masked
10:08	0_2	Reserved.
07	0_2	TPMI 0 Outbound Interrupt Pending.
06:05	0_2	Reserved.

Table 394. Interrupt Control Register 3 — INTCTL3 (Sheet 2 of 2)



Intel XScale® processor Coprocessor address
CP6, Page 4, Register 3

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

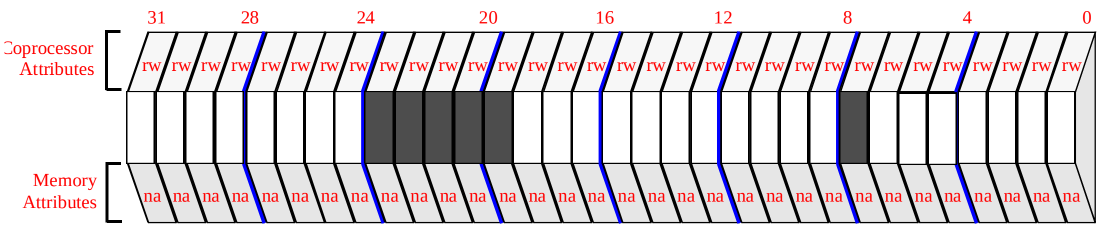
Bit	Default	Description
04	0_2	IMU Interrupt Pending. 0 = Masked 1 = Not Masked
03	0_2	ATU-E Error Interrupt Pending. 0 = Masked 1 = Not Masked
02	0_2	ATU-E Configuration Register Write Interrupt Pending. 0 = Masked 1 = Not Masked
01	0_2	ATU-E/Start BIST Interrupt Pending. 0 = Masked 1 = Not Masked
00	0_2	I²C Bus Interface 2 Interrupt Pending. 0 = Masked 1 = Not Masked

10.7.13 Interrupt Steering Register 0 — INTSTR0

The Interrupt Steering Register 0 allows system designers to direct any of 32 internal or external interrupt sources to either one of the two internal interrupt exceptions, FIQ and IRQ.

When an interrupt is enabled with the INTCTL0 register, this register steers the interrupt to an internal interrupt exception.

Table 395. Interrupt Steering Register 0 — INTSTR0 (Sheet 1 of 2)

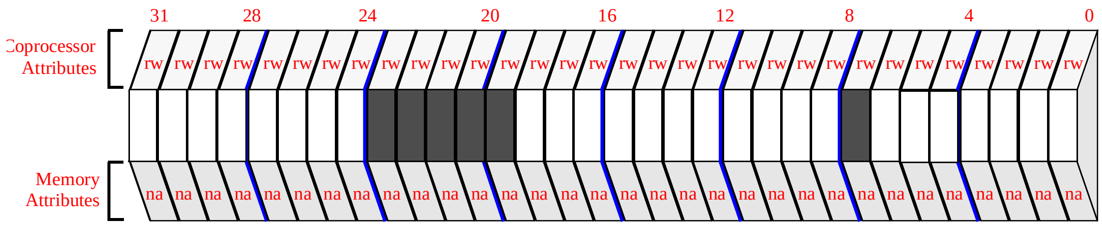


Intel XScale® processor Coprocessor address
CP6, Page 5, Register 0

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31	0_2	**XINT7#** Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
30	0_2	**XINT6#** Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
29	0_2	**XINT5#** Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
28	0_2	**XINT4#** Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
27	0_2	**XINT3#** Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
26	0_2	**XINT2#** Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
25	0_2	**XINT1#** Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
24	0_2	**XINT0#** Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
23:19	0_2	Reserved.
18	0_2	Intel XScale® Processor Cache Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
17	0_2	Intel XScale® Processor PMU Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
16	0_2	Peripheral Performance Monitor Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ

Table 395. Interrupt Steering Register 0 — INTSTR0 (Sheet 2 of 2)



Intel XScale® processor Coprocessor address

CP6, Page 5, Register 0

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set

RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

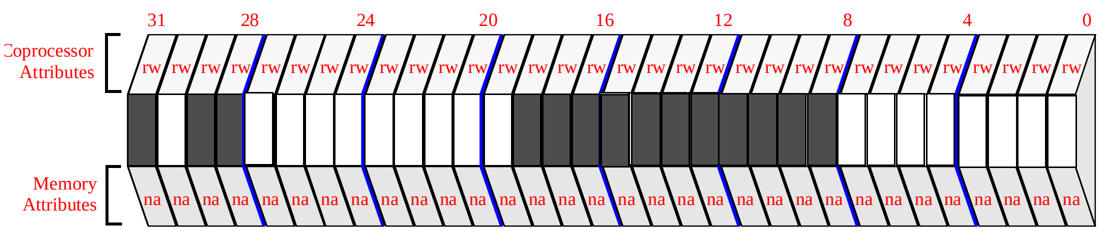
Bit	Default	Description
15	0_2	ATU/Start BIST Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
14	0_2	ATU-E Inbound Message Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
13	0_2	Reserved.
12	0_2	Messaging Unit Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
11	0_2	I^2C Bus Interface 1 Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
10	0_2	I^2C Bus Interface 0 Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
9	0_2	Timer 1 Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
8	0_2	Timer 0 Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
7	0_2	Reserved.
6	0_2	Watch Dog Timer Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
5	0_2	Reserved.
4	0_2	Reserved.
3	0_2	Reserved.
2	0_2	Reserved.
1	0_2	Reserved.
0	0_2	Reserved.

10.7.14 Interrupt Steering Register 1 — INTSTR1

The Interrupt Steering Register 1 allows system designers to direct any of 32 internal or external interrupt sources to either one of the two internal interrupt exceptions, FIQ and IRQ.

When an interrupt is enabled with the INTCTL1 register, this register steers the interrupt to an internal interrupt exception.

Table 396. Interrupt Steering Register 1 — INTSTR1 (Sheet 1 of 2)



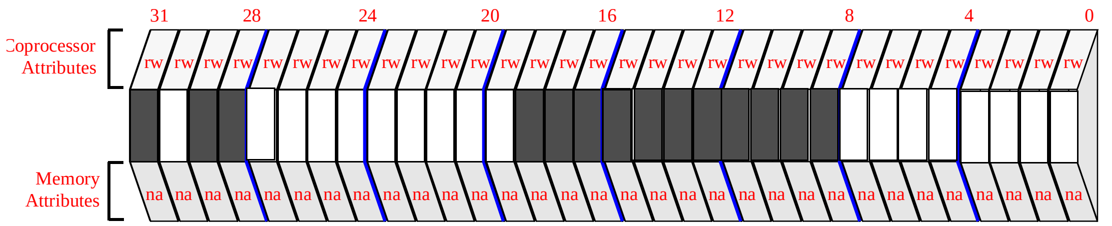
Intel XScale® processor Coprocessor address

CP6, Page 5, Register 1

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31	0_2	Reserved.
30	0_2	Messaging Unit Error Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
29:28	0_2	Reserved.
27	0_2	Reserved.
26	0_2	Reserved.
25	0_2	Reserved.
24	0_2	Reserved.
23	0_2	ATU Error Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
22	0_2	ATU Configuration Register Update Interrupt 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
21	0_2	Peripheral Bus Interface Unit Error Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
20	0_2	UART 1 Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
19	0_2	UART 0 Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
18:08	0000H	Reserved.
07	0_2	**XINT15#** Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
06	0_2	**XINT14#** Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ

Table 396. Interrupt Steering Register 1 — INTSTR1 (Sheet 2 of 2)



Intel XScale® processor Coprocessor address

CP6, Page 5, Register 1

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

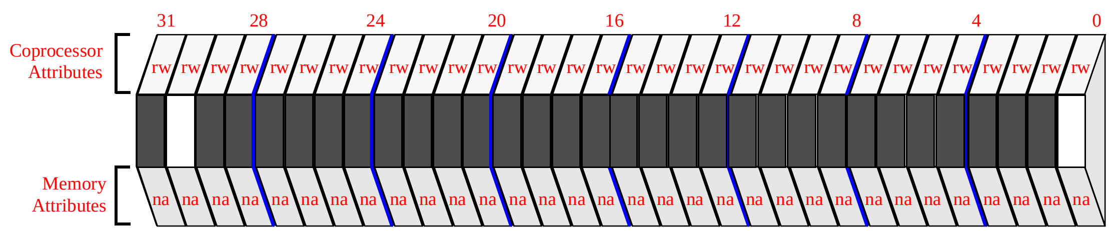
Bit	Default	Description
05	0_2	**XINT13#** Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
04	0_2	**XINT12#** Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
03	0_2	**XINT11#** Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
02	0_2	**XINT10#** Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
01	0_2	**XINT9#** Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
00	0_2	**XINT8#** Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ

10.7.15 Interrupt Steering Register 2 — INTSTR2

The Interrupt Steering Register 2 allows system designers to direct any of 32 internal or external interrupt sources to either one of the two internal interrupt exceptions, FIQ and IRQ.

When an interrupt is enabled with the INTCTL2 register, this register steers the interrupt to an internal interrupt exception.

Table 397. Interrupt Steering Register 2 — INTSTR2



Intel XScale® processor Coprocessor address

CP6, Page 5, Register 2

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

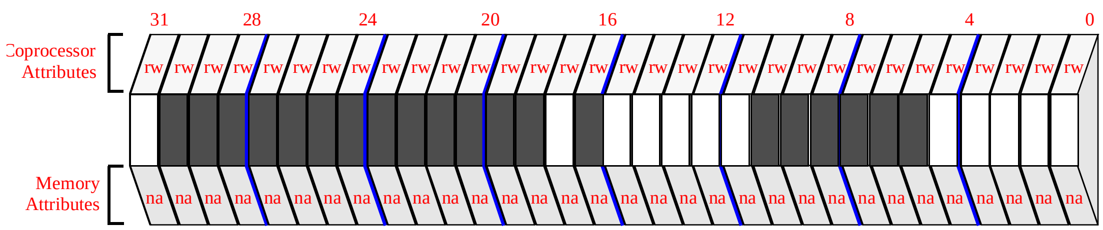
Bit	Default	Description
31	0_2	SRAM Memory Controller Unit Error Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
30	0_2	South Internal Bus Bridge Error Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
29:14	0_2	Reserved.
13	0_2	SRAM DMA Error Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
12	0_2	SRAM DMA Normal Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
11:03	0_2	Reserved.
02	0_2	Reserved.
01	0_2	Reserved.
00	0_2	Reserved.

10.7.16 Interrupt Steering Register 3 — INTSTR3

The Interrupt Steering Register 3 allows system designers to direct any of 32 internal or external interrupt sources to either one of the two internal interrupt exceptions, FIQ and IRQ.

When an interrupt is enabled with the INTCTL3 register, this register steers the interrupt to an internal interrupt exception.

Table 398. Interrupt Steering Register 3 — INTSTR3 (Sheet 1 of 2)



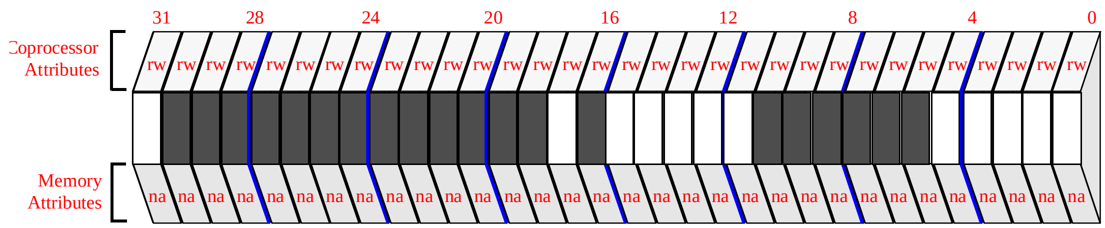
Intel XScale® processor Coprocessor address

CP6, Page 5, Register 3

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31	0_2	HPI Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
30:18	0_2	Reserved.
17	0_2	Inbound MSI Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
16	0_2	Reserved.
15	0_2	MU MSI-X Table Write Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
14	0_2	ATUE Interrupt Message D Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
13	0_2	ATUE Interrupt Message C Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
12	0_2	ATUE Interrupt Message B Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
11	0_2	ATUE Interrupt Message A Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
10:08	0_2	Reserved.
07	0_2	TPMI 0 Outbound Interrupt Steering. 0 = 0 = Interrupt Directed to Internal IRQ 1 = 1 = Interrupt Directed to Internal FIQ
06:05	0_2	Reserved.

Table 398. Interrupt Steering Register 3 — INTSTR3 (Sheet 2 of 2)



Intel XScale® processor Coprocessor address

CP6, Page 5, Register 3

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

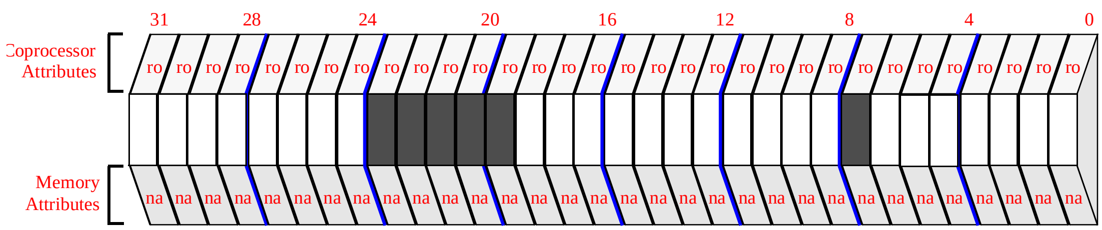
Bit	Default	Description
04	0_2	IMU Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
03	0_2	ATU-E Error Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
02	0_2	ATU-E Configuration Register Write Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
01	0_2	ATU-E/Start BIST Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ
00	0_2	I^2C Bus Interface 2 Interrupt Steering 0 = Interrupt Directed to Internal IRQ 1 = Interrupt Directed to Internal FIQ

10.7.17 IRQ Interrupt Source Register 0 — IINTSRC0

The IRQ Interrupt Source register is a 32-bit Coprocessor 6 control register used to specify which of 32 interrupts that are steered to the internal IRQ exception are unmasked by the INTCTL0 register and active. The INTSTR0 control register is used to steer individual interrupts to the IRQ exception.

The IINTSRC0 register may be used by an Interrupt Service Routine (ISR) to determine quickly the source of an IRQ interrupt.

Table 399. IRQ Interrupt Source Register 0 — IINTSRC0 (Sheet 1 of 2)

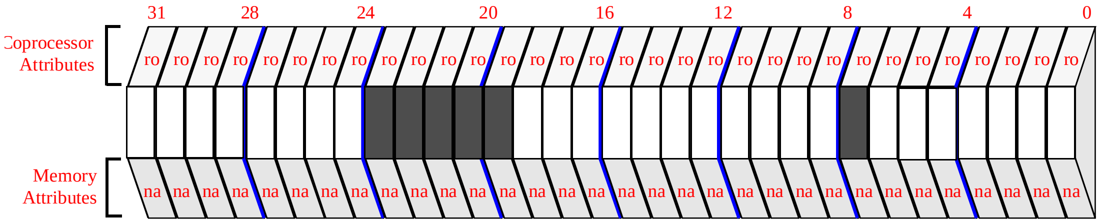


Intel XScale® processor Coprocessor address
CP6, page 6, Register 0

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:	0_2	**XINT7#** Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL0 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL0
30:	0_2	**XINT6#** Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL0 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL0
29:	0_2	**XINT5#** Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL0 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL0
28:	0_2	**XINT4#** Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL0 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL0
27	0_2	**XINT3#** Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL0 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL0
26	0_2	**XINT2#** Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL0 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL0
25	0_2	**XINT1#** Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL0 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL0
24	0_2	**XINT0#** Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL0 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL0
23:19	0_2	Reserved.
18	0_2	Intel XScale® Processor Cache Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL0 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL0
17	0_2	Intel XScale® Processor PMU Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL0 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL0

Table 399. IRQ Interrupt Source Register 0 — IINTSRC0 (Sheet 2 of 2)



Intel XScale® processor Coprocessor address
CP6, page 6, Register 0

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

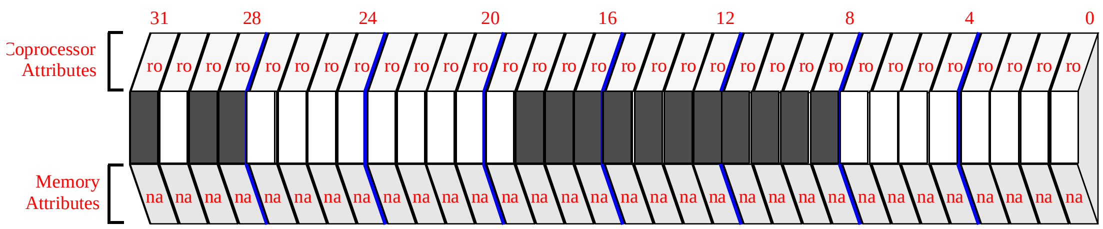
Bit	Default	Description
16	0_2	Peripheral Performance Monitor Interrupt — when set, at least one of the programmable event counters and/or the Global Time Stamp Counter contains an overflow condition. Application software identifies the counter by reading the Event Monitoring Interrupt Status register (EMISR). 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL0 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL0
15	0_2	ATU/Start BIST Interrupt — when set, the host processor has set the start BIST request in the ATUBISTR register. 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL0 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL0
14	0_2	ATU-E Inbound Message Interrupt — when set, the ATU has set the Inbound Vendor Message Received bit in the ATUISR register. 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL0 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL0
13	0_2	Reserved.
12	0_2	Messaging Unit Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL0 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL0
11	0_2	I^2C Bus Interface 1 Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL0 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL0
10	0_2	I^2C Bus Interface 0 Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL0 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL0
9	0_2	Timer 1 Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL0 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL0
8	0_2	Timer 0 Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL0 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL0
7	0_2	Reserved.
6	0_2	Watch Dog Timer Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL0 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL0
5	0_2	Reserved.
4	0_2	Reserved.
3	0_2	Reserved.
2	0_2	Reserved.
1	0_2	Reserved.
0	0_2	Reserved.

10.7.18 IRQ Interrupt Source Register 1 — IINTSRC1

The IRQ Interrupt Source register is a 32-bit Coprocessor 6 control register used to specify which of 32 interrupts that are steered to the internal IRQ exception are unmasked by the INTCTL1 register and active. The INTSTR1 control register is used to steer individual interrupts to the IRQ exception.

The IINTSRC1 register may be used by an Interrupt Service Routine (ISR) to determine quickly the source of an IRQ interrupt.

Table 400. IRQ Interrupt Source Register 1 — IINTSRC1 (Sheet 1 of 2)

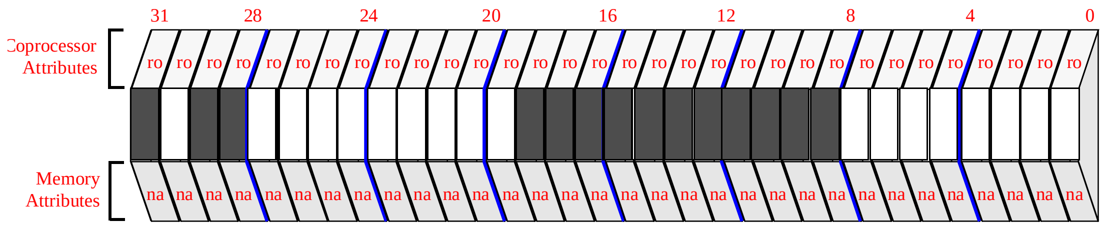


Intel XScale® processor Coprocessor address
CP6, Page 6, Register 1

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31	0_2	Reserved.
30	0_2	Messaging Unit Error Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL1 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL1
29:28	0_2	Reserved.
27	0_2	Reserved.
26	0_2	Reserved.
25	0_2	Reserved.
24	0_2	Memory Controller Unit Error Interrupt — when set, an error condition exists within the MCU. The bit indicates one of the following conditions: • A single-bit correctable or uncorrectable ECC error. • A multi-bit correctable or uncorrectable ECC error. 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL1 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL1
23	0_2	ATU Error Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL1 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL1
22	0_2	ATU Configuration Register Write Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL1 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL1
21	0_2	Peripheral Bus Interface Unit Error Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL1 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL1
20	0_2	UART 1 Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL1 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL1
19	0_2	UART 0 Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL1 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL1
18:08	0000H	Reserved.

Table 400. IRQ Interrupt Source Register 1 — IINTSRC1 (Sheet 2 of 2)



Intel XScale® processor Coprocessor address
CP6, Page 6, Register 1

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

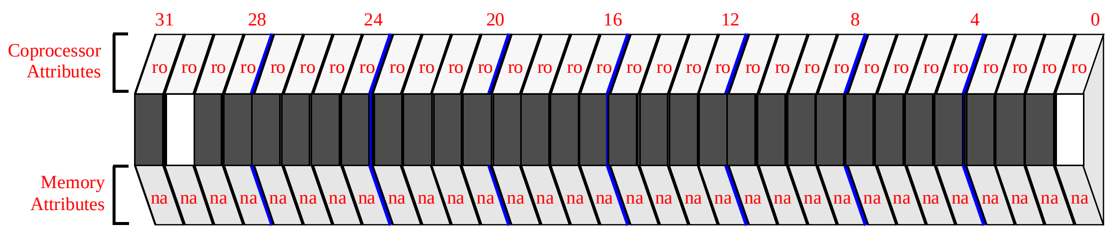
Bit	Default	Description
7	0_2	XINT15# Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL1 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL1
6	0_2	XINT14# Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL1 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL1
5	0_2	XINT13# Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL1 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL1
4	0_2	XINT12# Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL1 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL1
3	0_2	XINT11# Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL1 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL1
2	0_2	XINT10# Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL1 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL1
1	0_2	XINT9# Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL1 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL1
0	0_2	XINT8# Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL1 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL1

10.7.19 IRQ Interrupt Source Register 2 — IINTSRC2

The IRQ Interrupt Source register is a 32-bit Coprocessor 6 control register used to specify which of 32 interrupts that are steered to the internal IRQ exception are unmasked by the INTCTL2 register and active. The INTSTR2 control register is used to steer individual interrupts to the IRQ exception.

The IINTSRC2 register may be used by an Interrupt Service Routine (ISR) to determine quickly the source of an IRQ interrupt.

Table 401. IRQ Interrupt Source Register 2 — IINTSRC2



Intel XScale® processor Coprocessor address
CP6, Page 6, Register 2

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

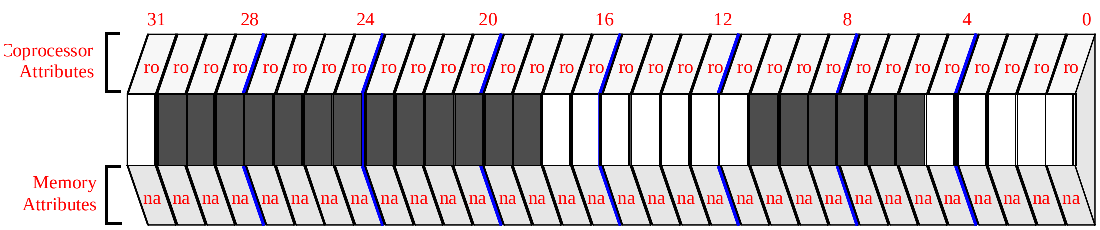
Bit	Default	Description
31	0_2	SRAM Memory Controller Unit Error Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL2 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL2
30	0_2	South Internal Bus Bridge Error Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL2 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL2
29:14	0_2	Reserved.
13	0_2	SRAM DMA Error 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL2 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL2
12	0_2	SRAM DMA Normal Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL2 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL2
11:03	0_2	Reserved.
02	0_2	Reserved.
01	0_2	Reserved.
00	0_2	Reserved.

10.7.20 IRQ Interrupt Source Register 3 — IINTSRC3

The IRQ Interrupt Source register is a 32-bit Coprocessor 6 control register used to specify which of 32 interrupts that are steered to the internal IRQ exception are unmasked by the INTCTL3 register and active. The INTSTR3 control register is used to steer individual interrupts to the IRQ exception.

The IINTSRC3 register may be used by an Interrupt Service Routine (ISR) to determine quickly the source of an IRQ interrupt.

Table 402. IRQ Interrupt Source Register 3 — IINTSRC3 (Sheet 1 of 2)

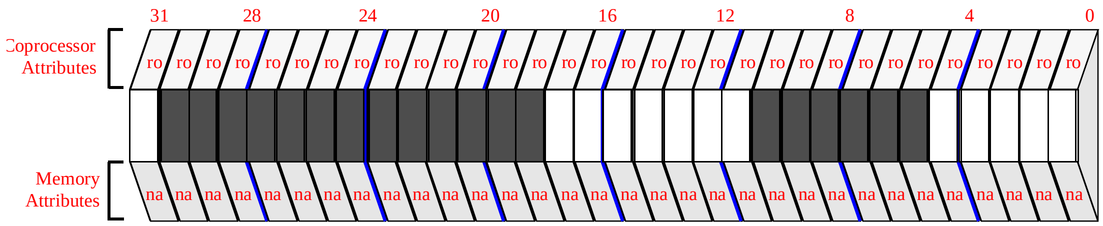


Intel XScale® processor Coprocessor address
CP6, Page 6, Register 3

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31	0_2	HPI Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL3 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL3
30:18	0000 0H	Reserved.
17	0_2	Inbound MSI Interrupt 0 = Not interrupting or not steered to internal IRQ exception or masked by INTCTL3 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL3
16	0_2	Reserved.
15	0_2	MU MSI-X Table Write Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL3 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL3
14	0_2	ATUE Interrupt Message D 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL3 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL3
13	0_2	ATUE Interrupt Message C 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL3 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL3
12	0_2	ATUE Interrupt Message B 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL3 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL3
11	0_2	ATUE Interrupt Message A 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL3 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL3
10:08	0_2	Reserved.
07	0_2	TPMI 0 Outbound Interrupt. 0 = 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL3 1 = 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL3
06:05	0_2	Reserved.

Table 402. IRQ Interrupt Source Register 3 — IINTSRC3 (Sheet 2 of 2)



Intel XScale® processor Coprocessor address
CP6, Page 6, Register 3

Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

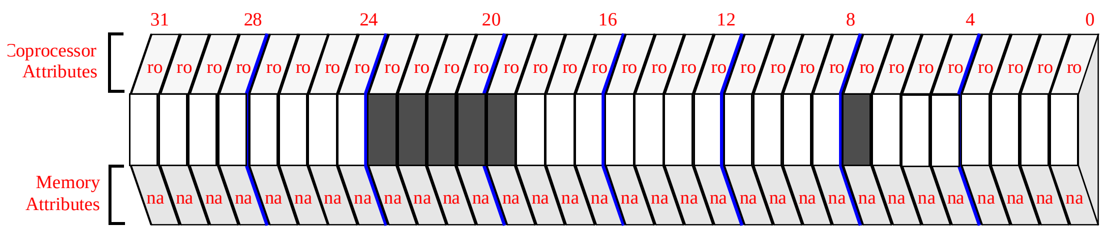
Bit	Default	Description
04	0_2	IMU Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL3 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL3
03	0_2	ATU-E Error Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL3 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL3
02	0_2	ATU-E Configuration Register Write Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL3 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL3
01	0_2	ATU-E/Start BIST Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL3 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL3
00	0_2	I^2C Bus Interface 2 Interrupt 0 = Not Interrupting or Not steered to internal IRQ exception or masked by INTCTL3 1 = Interrupting and steered to internal IRQ exception and unmasked by INTCTL3

10.7.21 FIQ Interrupt Source Register 0 — FINTSRC0

The FIQ Interrupt Source register 0 is a 32-bit Coprocessor 6 control register used to specify which interrupts that are steered to the internal FIQ exception are unmasked by the INTCTL0 register and active. The INTSTR0 control register is used to steer individual interrupts to the FIQ exception.

The FINTSRC0 register may be used by an Interrupt Service Routine (ISR) to determine quickly the source of an FIQ interrupt.

Table 403. FIQ Interrupt Source Register 0 — FINTSRC0 (Sheet 1 of 2)

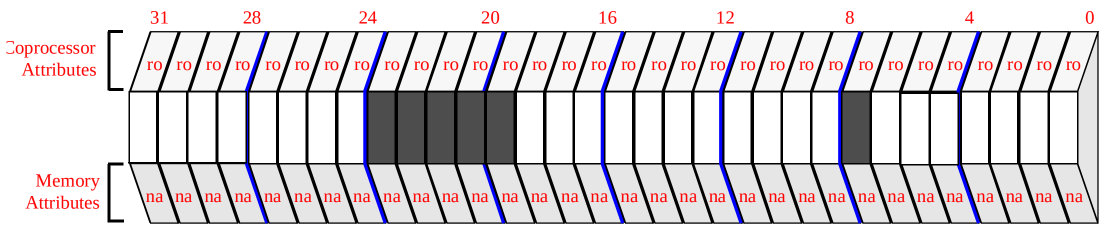


Intel XScale® processor Coprocessor address
CP6, Page 7, Register 0

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31	0_2	**XINT7#** Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL0 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL0
30:	0_2	**XINT6#** Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL0 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL0
29:	0_2	**XINT5#** Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL0 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL0
28:	0_2	**XINT4#** Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL0 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL0
27	0_2	**XINT3#** Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL0 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL0
26	0_2	**XINT2#** Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL0 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL0
25	0_2	**XINT1#** Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL0 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL0
24	0_2	**XINT0#** Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL0 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL0
23:19	0_2	Reserved.
18	0_2	Intel XScale® Processor Cache Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL0 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL0
17	0_2	Intel XScale® Processor PMU Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL0 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL0

Table 403. FIQ Interrupt Source Register 0 — FINTSRC0 (Sheet 2 of 2)



Intel XScale® processor Coprocessor address
CP6, Page 7, Register 0

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

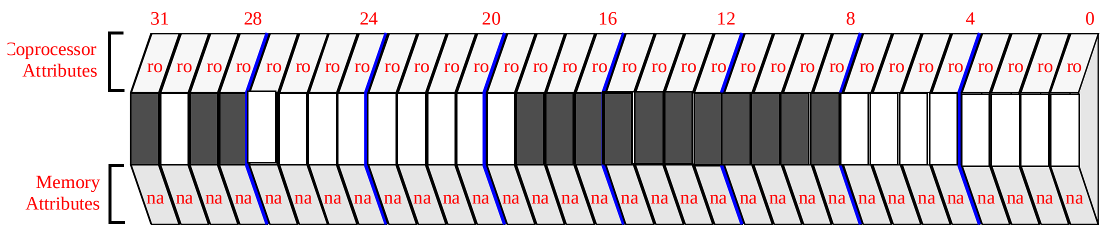
Bit	Default	Description
16	0_2	Peripheral Performance Monitor Interrupt — when set, at least one of the programmable event counters and/or the Global Time Stamp Counter contains an overflow condition. Application software identifies the counter by reading the Event Monitoring Interrupt Status register (EMISR). 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL0 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL0
15	0_2	ATU/Start BIST Interrupt — when set, the host processor has set the start BIST request in the ATUBISTR register. 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL0 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL0
14	0_2	ATU-E Inbound Message Interrupt — when set, the ATU has set the Inbound Vendor Message Received bit in the ATUISR register. 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL0 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL0
13	0_2	Reserved.
12	0_2	Messaging Unit Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL0 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL0
11	0_2	I^2C Bus Interface 1 Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL0 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL0
10	0_2	I^2C Bus Interface 0 Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL0 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL0
9	0_2	Timer 1 Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL0 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL0
8	0_2	Timer 0 Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL0 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL0
7	0_2	Reserved.
6	0_2	Watch Dog Timer Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL0 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL0
5	0_2	Reserved.
4	0_2	Reserved.
3	0_2	Reserved.
2	0_2	Reserved.
1	0_2	Reserved.
0	0_2	Reserved.

10.7.22 FIQ Interrupt Source Register 1 — FINTSRC1

The FIQ Interrupt Source register 1 is a 32-bit Coprocessor 6 control register used to specify which interrupts that are steered to the internal FIQ exception are unmasked by the INTCTL1 register and active. The INTSTR1 control register is used to steer individual interrupts to the FIQ exception.

The FINTSRC1 register may be used by an Interrupt Service Routine (ISR) to determine quickly the source of an FIQ interrupt.

Table 404. FIQ Interrupt Source Register 1 — FINTSRC1 (Sheet 1 of 2)

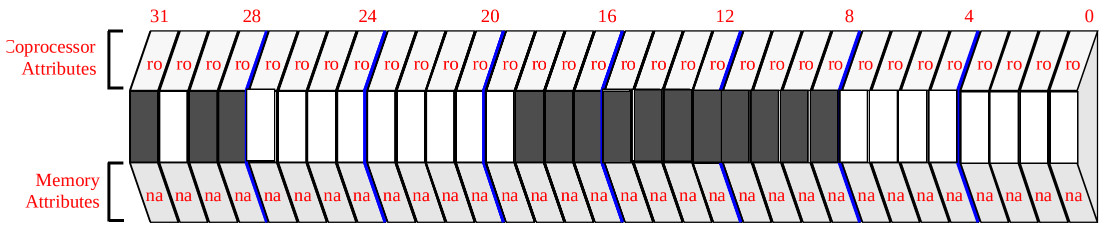


Intel XScale® processor Coprocessor address
CP6, Page 7, Register 1

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31	0_2	Reserved.
30	0_2	Messaging Unit Error Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL1 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL1
29:28	0_2	Reserved.
27	0_2	Reserved.
26	0_2	Reserved.
25	0_2	Reserved.
24	0_2	Reserved.
23	0_2	ATU Error Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL1 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL1
22	0_2	ATU Configuration Register Write Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL1 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL1
21	0_2	Peripheral Bus Interface Unit Error Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL1 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL1
20	0_2	UART 1 Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL1 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL1
19	0_2	UART 0 Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL1 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL1
18:08	0000H	Reserved.
7	0_2	XINT15# Interrupt Mask 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL1 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL1
6	0_2	XINT14# Interrupt Mask 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL1 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL1

Table 404. FIQ Interrupt Source Register 1 — FINTSRC1 (Sheet 2 of 2)



Intel XScale® processor Coprocessor address
CP6, Page 7, Register 1

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

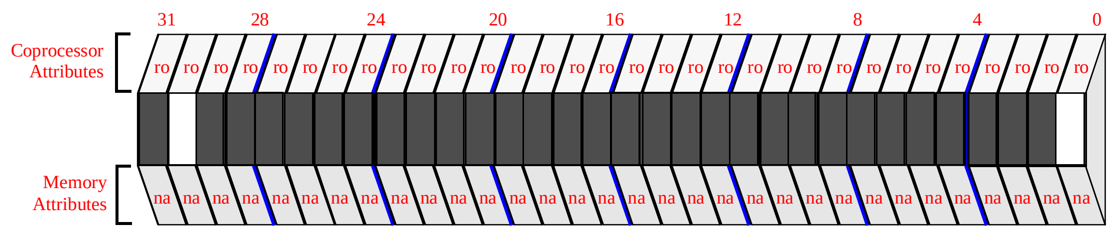
Bit	Default	Description
5	0_2	XINT13# Interrupt Mask 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL1 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL1
4	0_2	XINT12# Interrupt Mask 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL1 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL1
3	0_2	XINT11# Interrupt Mask 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL1 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL1
2	0_2	XINT10# Interrupt Mask 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL1 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL1
1	0_2	XINT9# Interrupt Mask 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL1 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL1
0	0_2	XINT8# Interrupt Mask 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL1 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL1

10.7.23 FIQ Interrupt Source Register 2 — FINTSRC2

The FIQ Interrupt Source register 2 is a 32-bit Coprocessor 6 control register used to specify which interrupts that are steered to the internal FIQ exception are unmasked by the INTCTL2 register and active. The INTSTR2 control register is used to steer individual interrupts to the FIQ exception.

The FINTSRC2 register may be used by an Interrupt Service Routine (ISR) to determine quickly the source of an FIQ interrupt.

Table 405. FIQ Interrupt Source Register 2 — FINTSRC2



Intel XScale® processor Coprocessor address

CP6, Page 7, Register 2

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

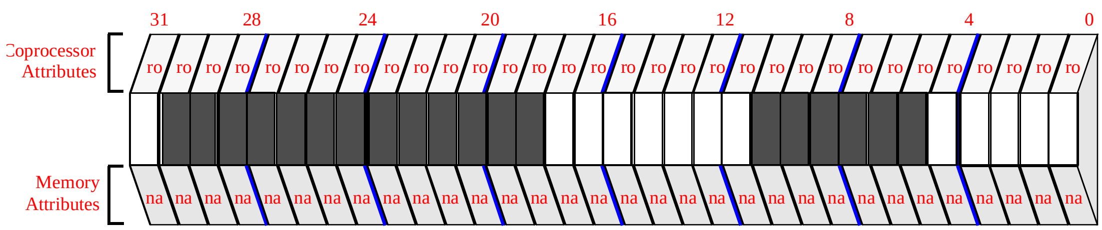
Bit	Default	Description
31	0_2	Reserved.
30	0_2	South Internal Bus Bridge Error Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL2 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL2
29:14	0_2	Reserved.
13	0_2	SRAM DMA Error 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL2 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL2
12	0_2	SRAM DMA Normal Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL2 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL2
11:03	0_2	Reserved.
02	0_2	Reserved.
01	0_2	Reserved.
00	0_2	Reserved.

10.7.24 FIQ Interrupt Source Register 3 — FINTSRC3

The FIQ Interrupt Source register 3 is a 32-bit Coprocessor 6 control register used to specify which interrupts that are steered to the internal FIQ exception are unmasked by the INTCTL3 register and active. The INTSTR3 control register is used to steer individual interrupts to the FIQ exception.

The FINTSRC3 register may be used by an Interrupt Service Routine (ISR) to determine quickly the source of an FIQ interrupt.

Table 406. FIQ Interrupt Source Register 3 — FINTSRC3 (Sheet 1 of 2)



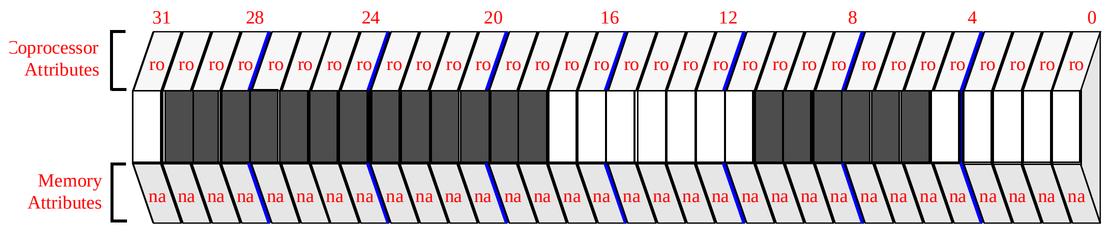
Intel XScale® processor Coprocessor address

CP6, Page 7, Register 3

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31	0_2	HPI Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL3 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL3
30:18	0_2	Reserved.
17	0_2	Inbound MSI Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL3 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL3
16	0_2	Reserved.
15	0_2	MU MSI-X Table Write Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL3 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL3
14	0_2	ATUE Interrupt Message D 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL3 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL3
13	0_2	ATUE Interrupt Message C 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL3 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL3
12	0_2	ATUE Interrupt Message B 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL3 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL3
11	0_2	ATUE Interrupt Message A 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL3 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL3
10:08	0_2	Reserved.
07	0_2	TPMI 0 Outbound Interrupt. 0 = 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL3 1 = 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL3
06:05	0_2	Reserved.

Table 406. FIQ Interrupt Source Register 3 — FINTSRC3 (Sheet 2 of 2)



Intel XScale® processor Coprocessor address
CP6, Page 7, Register 3

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
04	0_2	IMU Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL3 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL3
03	0_2	ATU-E Error Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL3 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL3
02	0_2	ATU-E Configuration Register Write Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL3 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL3
01	0_2	ATU-E/Start BIST Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL3 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL3
00	0_2	I^2C Bus Interface 2 Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL3 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL3

10.7.25 Interrupt Priority Register 0 — IPR0

The Interrupt Priority Register 0 is a 32-bit Coprocessor 6 control register used to assign a priority level to interrupt sources 15 down to 0. The IPR0 control register is used to assign one of 4 priority levels to each interrupt source independent of the INTSTR[3:0] registers:

00_2 – High Priority

01_2 – Medium/High Priority

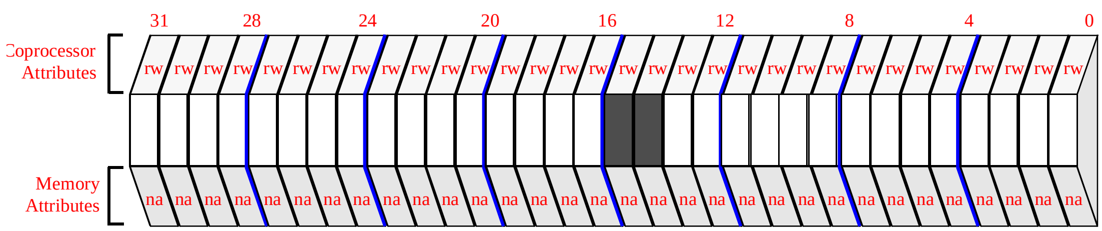
10_2 – Medium/Low Priority

11_2 – Low Priority

When interrupt vector generation is enabled and there are multiple requests pending either in the FINTSRC[3:0] or the IINTSRC[3:0] registers, the highest priority vectors pending for either FIQ or IRQ are presented in the FINTVEC or IINTVEC respectively.

Note: When multiple interrupts at the same priority level are pending for either FIQ or IRQ, the vector is selected according to a fixed priority based on bit location. Highest order bit is first.

Table 407. Interrupt Priority Register 0 — IPR0



Intel XScale® processor Coprocessor address
CP6, page 8, Register 0

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:30	00_2	ATU/Start BIST Interrupt Priority
29:28	00_2	ATU-E Inbound Message Interrupt Priority
27:26	00_2	Reserved.
25:24	00_2	Messaging Unit Interrupt Priority
23:22	00_2	I²C Bus Interface 1 Interrupt Priority
21:20	00_2	I²C Bus Interface 0 Interrupt Priority
19:18	00_2	Timer 1 Interrupt Priority
17:16	00_2	Timer 0 Interrupt Priority
15:14	00_2	Reserved.
13:12	00_2	Watch Dog Timer Interrupt Priority
11:10	00_2	Reserved.
09:08	00_2	Reserved.
07:06	00_2	Reserved.
05:04	00_2	Reserved.
03:02	00_2	Reserved.
01:00	00_2	Reserved.

10.7.26 Interrupt Priority Register 1 — IPR1

The Interrupt Priority Register 1 is a 32-bit Coprocessor 6 control register used to assign a priority level to interrupt sources 31 down to 15. The IPR1 control register is used to assign one of 4 priority levels to each interrupt source independent of the INTSTR[3:0] registers:

00_2 _ High Priority

01_2 _ Medium/High Priority

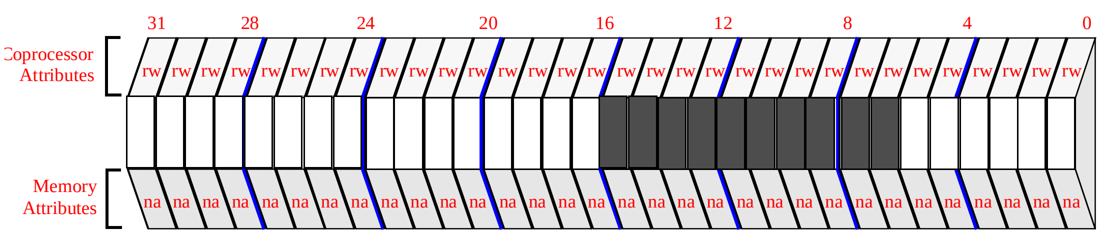
10_2 _ Medium/Low Priority

11_2 _ Low Priority

When interrupt vector generation is enabled and there are multiple requests pending either in the FINTSRC[3:0] or the IINTSRC[3:0] registers, the highest priority vectors pending for either FIQ or IRQ are presented in the FINTVEC or IINTVEC respectively.

Note: When multiple interrupts at the same priority level are pending for either FIQ or IRQ, the vector is selected according to a fixed priority based on bit location. Highest order bit is first.

Table 408. Interrupt Priority Register 1 — IPR1



Intel XScale® processor Coprocessor address
CP6, Page 8, Register 1

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:30	00_2	**XINT7#** Interrupt Priority
29:28	00_2	**XINT6#** Interrupt Priority
27:26	00_2	**XINT5#** Interrupt Priority
25:24	00_2	**XINT4#** Interrupt Priority
23:22	00_2	**XINT3#** Interrupt Priority
21:20	00_2	**XINT2#** Interrupt Priority
19:18	00_2	**XINT1#** Interrupt Priority
17:16	00_2	**XINT0#** Interrupt Priority
15:06	0000H	Reserved.
05:04	00_2	Intel XScale® Processor Cache Interrupt Priority
03:02	00_2	Intel XScale® Processor PMU Interrupt Priority
01:00	00_2	Peripheral Performance Monitor Interrupt Priority

10.7.27 Interrupt Priority Register 2 — IPR2

The Interrupt Priority Register 2 is a 32-bit Coprocessor 6 control register used to assign a priority level to interrupt sources 47 down to 32. The IPR2 control register is used to assign one of 4 priority levels to each interrupt source independent of the INTSTR[3:0] registers:

00_2 — High Priority

01_2 — Medium/High Priority

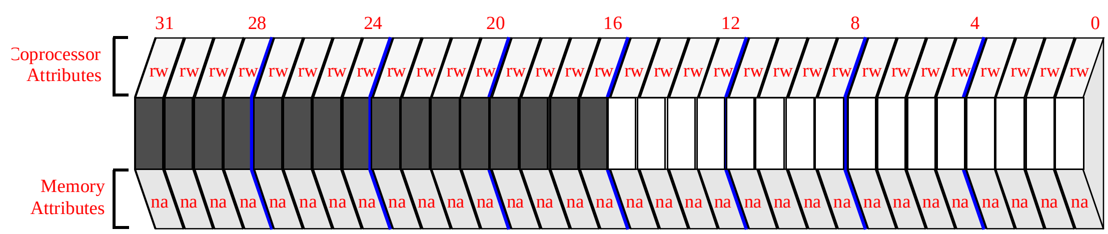
10_2 — Medium/Low Priority

11_2 — Low Priority

When interrupt vector generation is enabled and there are multiple requests pending either in the FINTSRC[3:0] or the IINTSRC[3:0] registers, the highest priority vectors pending for either FIQ or IRQ are presented in the FINTVEC or IINTVEC respectively.

Note: When multiple interrupts at the same priority level are pending for either FIQ or IRQ, the vector is selected according to a fixed priority based on bit location. Highest order bit is first.

Table 409. Interrupt Priority Register 2 — IPR2



Intel XScale® processor Coprocessor address
CP6, Page 8, Register 2

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:16	00H	Reserved.
15:14	00_2	XINT15# Interrupt Priority
13:12	00_2	XINT14# Interrupt Priority
11:10	00_2	XINT13# Interrupt Priority
09:08	00_2	XINT12# Interrupt Priority
07:06	00_2	XINT11# Interrupt Priority
05:04	00_2	XINT10# Interrupt Priority
03:02	00_2	XINT9# Interrupt Priority
01:00	00_2	XINT8# Interrupt Priority

10.7.28 Interrupt Priority Register 3 — IPR3

The Interrupt Priority Register 3 is a 32-bit Coprocessor 6 control register used to assign a priority level to interrupt sources 63 down to 48. The IPR3 control register is used to assign one of 4 priority levels to each interrupt source independent of the INTSTR[3:0] registers:

00_2 _ High Priority

01_2 _ Medium/High Priority

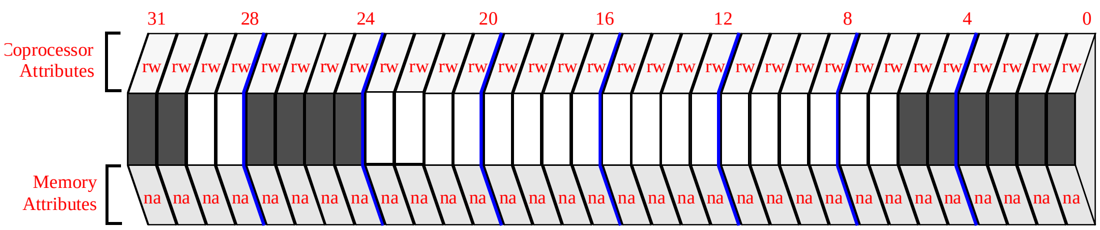
10_2 _ Medium/Low Priority

11_2 _ Low Priority

When interrupt vector generation is enabled and there are multiple requests pending either in the FINTSRC[3:0] or the IINTSRC[3:0] registers, the highest priority vectors pending for either FIQ or IRQ are presented in the FINTVEC or IINTVEC respectively.

Note: When multiple interrupts at the same priority level are pending for either FIQ or IRQ, the vector is selected according to a fixed priority based on bit location. Highest order bit is first.

Table 410. Interrupt Priority Register 3 — IPR3



Intel XScale® processor Coprocessor address
CP6, Page 8, Register 3

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:30	00_2	Reserved.
29:28	00_2	Messaging Unit Error Interrupt Priority
27:24	00_2	Reserved.
23:22	00_2	Reserved.
21:20	00_2	Reserved.
19:18	00_2	Reserved.
17:16	00_2	Memory Controller Unit Error Interrupt Priority
15:14	00_2	ATU Error Interrupt Priority
13:12	00_2	ATU Configuration Register Write Interrupt Priority
11:10	00_2	Peripheral Bus Interface Unit Error Interrupt Priority
9:8	00_2	UART 1 Interrupt Priority
7:6	00_2	UART 0 Interrupt Priority
5:0	$00\ 0000_2$	Reserved.

10.7.29 Interrupt Priority Register 4 — IPR4

The Interrupt Priority Register 4 is a 32-bit Coprocessor 6 control register used to assign a priority level to interrupt sources 79 down to 64. The IPR4 control register is used to assign one of 4 priority levels to each interrupt source independent of the INTSTR[3:0] registers:

00_2 — High Priority

01_2 — Medium/High Priority

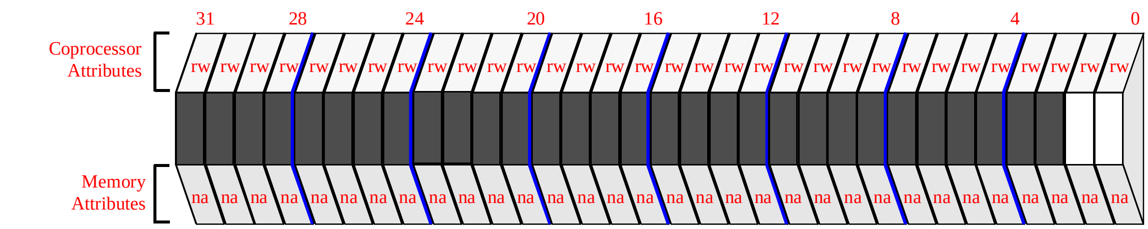
10_2 — Medium/Low Priority

11_2 — Low Priority

When interrupt vector generation is enabled and there are multiple requests pending either in the FINTSRC[3:0] or the IINTSRC[3:0] registers, the highest priority vectors pending for either FIQ or IRQ are presented in the FINTVEC or IINTVEC respectively.

Note: When multiple interrupts at the same priority level are pending for either FIQ or IRQ, the vector is selected according to a fixed priority based on bit location. Highest order bit is first.

Table 411. Interrupt Priority Register 4 — IPR4



Intel XScale® processor Coprocessor address
CP6, Page 8, Register 4

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:28	00_2	Reserved.
27:26	00_2	SRAM DMA Error Interrupt Priority
25:24	00_2	SRAM DMA Normal Interrupt Priority
23:06	00_2	Reserved.
05:04	00_2	TPMI 0 Error Interrupt Priority
03:02	00_2	TPMI 0 Normal Interrupt Priority
01:00	00_2	Reserved.

10.7.30 Interrupt Priority Register 5 — IPR5

The Interrupt Priority Register 5 is a 32-bit Coprocessor 6 control register used to assign a priority level to interrupt sources 95 down to 80. The IPR5 control register is used to assign one of 4 priority levels to each interrupt source independent of the INTSTR[3:0] registers:

00_2 _ High Priority

01_2 _ Medium/High Priority

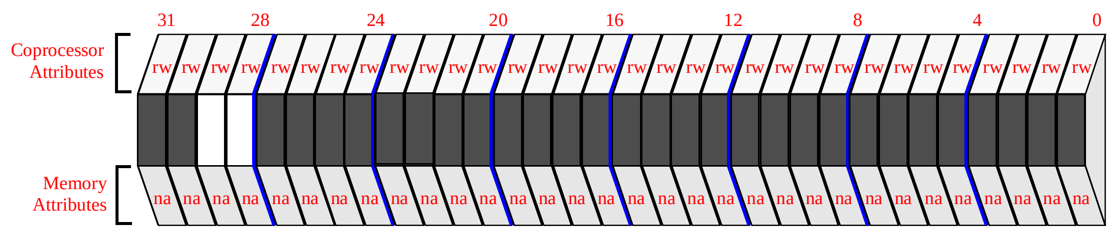
10_2 _ Medium/Low Priority

11_2 _ Low Priority

When interrupt vector generation is enabled and there are multiple requests pending either in the FINTSRC[3:0] or the IINTSRC[3:0] registers, the highest priority vectors pending for either FIQ or IRQ are presented in the FINTVEC or IINTVEC respectively.

Note: When multiple interrupts at the same priority level are pending for either FIQ or IRQ, the vector is selected according to a fixed priority based on bit location. Highest order bit is first.

Table 412. Interrupt Priority Register 5 — IPR5



Intel XScale® processor Coprocessor address
CP6, Page 8, Register 5

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:30	00_2	SRAM Memory Controller Unit Error Interrupt Priority.
29:28	00_2	South Internal Bus Bridge Error Interrupt Priority.
27:00	00_2	Reserved.

10.7.31 Interrupt Priority Register 6 — IPR6

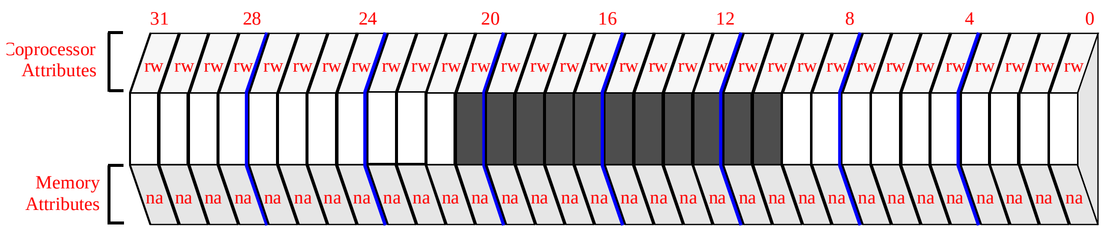
The Interrupt Priority Register 6 is a 32-bit Coprocessor 6 control register used to assign a priority level to interrupt sources 111 down to 96. The IPR6 control register is used to assign one of 4 priority levels to each interrupt source independent of the INTSTR[3:0] registers:

- 00_2 — High Priority
- 01_2 — Medium/High Priority
- 10_2 — Medium/Low Priority
- 11_2 — Low Priority

When interrupt vector generation is enabled and there are multiple requests pending either in the FINTSRC[3:0] or the IINTSRC[3:0] registers, the highest priority vectors pending for either FIQ or IRQ are presented in the FINTVEC or IINTVEC respectively.

Note: When multiple interrupts at the same priority level are pending for either FIQ or IRQ, the vector is selected according to a fixed priority based on bit location. Highest order bit is first.

Table 413. Interrupt Priority Register 6 — IPR6



Intel XScale® processor Coprocessor address
CP6, Page 8, Register 6

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:30	00_2	MU MSI-X Table Write Interrupt Priority.
29:28	00_2	ATUE Interrupt Message D Priority.
27:26	00_2	ATUE Interrupt Message C Priority.
25:24	00_2	ATUE Interrupt Message B Priority.
23:22	00_2	ATUE Interrupt Message A Priority.
21:10	00_2	Reserved.
09:08	00_2	IMU Interrupt Priority.
07:06	00_2	ATU-E Error Interrupt Priority.
05:04	00_2	ATU-E Configuration Register Write Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL3 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL3
03:02	00_2	ATU-E/Start BIST Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL3 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL3
01:00	00_2	I^2C Bus Interface 2 Interrupt 0 = Not Interrupting or Not steered to internal FIQ exception or masked by INTCTL3 1 = Interrupting and steered to internal FIQ exception and unmasked by INTCTL3

10.7.32 Interrupt Priority Register 7 — IPR7

The Interrupt Priority Register 7 is a 32-bit Coprocessor 6 control register used to assign a priority level to interrupt sources 127 down to 112. The IPR7 control register is used to assign one of 4 priority levels to each interrupt source independent of the INTSTR[3:0] registers:

00_2 _ High Priority

01_2 _ Medium/High Priority

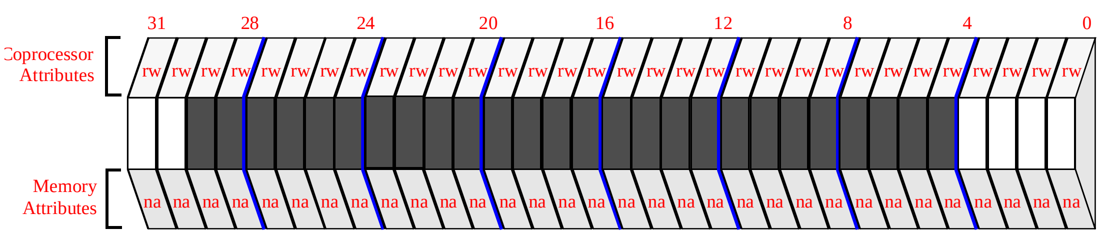
10_2 _ Medium/Low Priority

11_2 _ Low Priority

When interrupt vector generation is enabled and there are multiple requests pending either in the FINTSRC[3:0] or the IINTSRC[3:0] registers, the highest priority vectors pending for either FIQ or IRQ are presented in the FINTVEC or IINTVEC respectively.

Note: When multiple interrupts at the same priority level are pending for either FIQ or IRQ, the vector is selected according to a fixed priority based on bit location. Highest order bit is first.

Table 414. Interrupt Priority Register 7 — IPR7



Intel XScale® processor Coprocessor address
CP6, Page 8, Register 7

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:30	00_2	HPI Interrupt Priority
29:04	0000 0000H	Reserved.
03:02	00_2	Inbound MSI Interrupt Priority.
01:00	00_2	TPMI MSI-X Table Write Interrupt Priority.

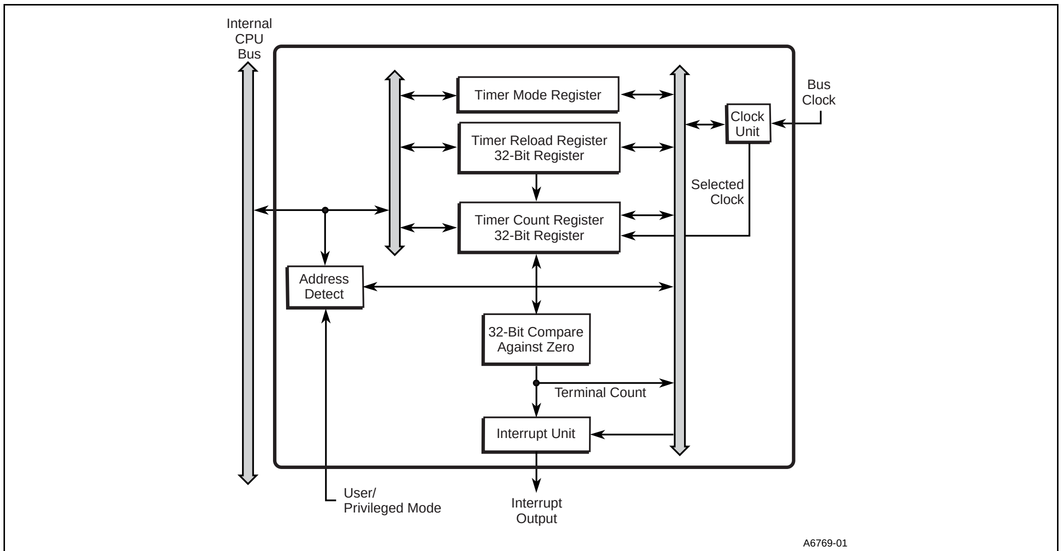
11.0 Timers

This chapter describes the Intel XScale® processor dual-programmable 32-bit timers and Watch Dog Timer. Topics include timer registers (TMRx, TCRx and TRRx), timer operation, timer interrupts, and timer register values at initialization.

Each timer is programmed by the timer registers. These registers are mapped into Intel XScale® processor Coprocessor 6, registers 0 to 8. They may be accessed/manipulated with the MCR, MRC, STC, and LDC instructions. The *CRn* field of the instruction denotes the register number to be accessed. The *opcode_1* and *opcode_2* fields of the instruction should be zero. The *CRm* field of the instruction should be nine. Most systems restrict access to CP6 to privileged processes. To control access to CP6, use the Coprocessor Access Register.

Figure 71 shows a diagram of the timer functions. See also Figure 72 for the Programmable Timer state diagram.

Figure 71. Programmable Timer Functional Diagram



When enabled, a timer decrements the user-defined count value with each Timer Clock (TCLOCK) cycle. The countdown rate is also user-configurable to be equal to the internal bus frequency, or the internal bus clock rate divided by 4, 8 or 16. The timers can be programmed to either stop when the count value reaches zero (single-shot mode) or run continuously (auto-reload mode). When a timer's count reaches zero, the timer's interrupt unit signals the processor's interrupt controller.

Table 415. Timer Performance Ranges

Internal Bus Frequency (MHz)	Max Resolution (ns)	Max Range (mins)
400	2.5	2.86

11.1 Timer Operation

This section summarizes the programmable timer and Watch Dog Timer operation and describes load/store access latency for the timer registers.

11.1.1 Basic Programmable Timer Operation

Each timer has a programmable enable bit in its control register (TMRx.enable) to start and stop counting. This allows the programmer to prevent user mode tasks from enabling or disabling the timer. Once the timer is enabled, the value stored in the Timer Count Register (TCRx) decrements every Timer Clock (TCLOCK) cycle. TCLOCK is determined by the Timer Input Clock Select (TMRx.csel) bit setting. The countdown rate can be set to equal the internal bus clock frequency, or the internal bus clock rate divided by 4, 8 or 16. Setting TCLOCK to a slower rate lets the user specify a longer count period with the same 32-bit TCRx value.

Software can read or write the TCRx value whether the timer is running or stopped. This lets the user monitor the count without using hardware interrupts.

When the TCRx value decrements to zero, the unit's interrupt request signals the processor's interrupt controller. See Section 11.2, "Timer Interrupts" on page 631 for more information. The timer checks the value of the timer reload bit (TMRx.reload) setting. When TMRx.reload. = 1, the processor:

- Automatically reloads TCRx with the value in the Timer Reload Register (TRRx).
- Decrements TCRx until it equals 0 again.

This process repeats until software clears TMRx.reload or TMR.enable.

When TMRx.reload = 0, the timer stops running and sets the terminal count bit (TMRx.tc). This bit remains set until user software reads or writes the TMRx register. Either access type clears the bit. The timer ignores any value specified for TMRx.tc in a write request.

Table 416. Timer Mode Register Control Bit Summary

TRRx	TCRx	Bit 2 (TMRx.reload)	Bit 1 (TMRx.enable)	Action
X	X	X	0	Timer disabled.

Note: X = don't care
N = a number between 1H and FFFF FFFFH

11.1.2 Watch Dog Timer Operation

The Watch Dog Timer (WDT) is a 32-bit down counter that can be used to reset the Internal Bus and the Intel XScale® processor or generate an interrupt when software gets stuck in an infinite loop. Refer to Section 426, "Watch Dog Timer Setup Register — WDTSR" on page 639 for setting up the Watchdog timer. A reset of the Internal Bus also results in the **M_RST#** output to be asserted which can be used to reset the system or as an external indicator of the WDT expiration.

Following **P_RST#** assertion, the WDT is disabled.

The software can enable the WDT by using coprocessor instructions (that is, MCR or LDC) to write the value 1E1E 1E1EH followed by the value E1E1 E1E1H to the WDT Control register. When enabled, the WDT is initialized with FFFF FFFFH and begin to decrement towards 0000 0000H.

The software is required periodically to write the WDT initialization sequence (the value 1E1E 1E1EH followed by the value E1E1 E1E1H) to the WDT Control register in order to reset the timer value to FFFF FFFFH. For a 300 MHz internal bus, this means that the sequence must be written approximately every fourteen seconds.

Note: The WDT always runs at Intel XScale® processor speed without any prescaling.

When the software fails to reinitialize the WDT prior to the timer value transitioning to zero, an Internal Bus Reset is generated. This reinitializes all Internal Bus peripherals and the Intel XScale® processor.

Once enabled, the WDT can be disabled by writing the value 1F1F 1F1FH followed by the value F1F1 F1F1H to the WDT Control register. The WDT can be enabled again by writing the value 1E1E 1E1EH followed by the value E1E1 E1E1H to the WDT Control register.

11.1.3 Load/Store Access Latency for Timer Registers

As with all other load accesses from internal memory-mapped registers, a load instruction that accesses a timer register has a latency of one internal processor cycle. With one exception, a store access to a timer register completes and all state changes take effect before the next instruction begins execution. The exception to this is when disabling a timer. Latency associated with the disabling action is such that a timer interrupt may be posted immediately after the disabling instruction completes. This can occur when the timer is near zero as the store to TMRx occurs. In this case, the timer interrupt is posted immediately after the store to TMRx completes and before the next instruction can execute. Table 417 summarizes the timer access and response timings. Refer also to the individual register descriptions for details.

Note that the processor may delay the actual issuing of the load or store operation due to previous instruction activity and resource availability of processor functional units.

The processor ensures that the TMRx.tc bit is cleared within one internal bus clock after a load or store instruction accesses TMRx.

Table 417. Timer Responses to Register Bit Settings

Name	Status	Action
(TMRx.tc) Terminal Count Bit 0	READ	Timer clears this bit when user software accesses TMRx. This bit can be set 1 internal bus clock later. The timer sets this bit within 1 internal bus clock of TCRx reaching zero when TMRx.reload=0.
	WRITE	Timer clears this bit within 1 internal bus clock after the software accesses TMRx. The timer ignores any value specified for TMRx.tc in a write request.
(TMRx.enable) Timer Enable Bit 1	READ	Bit is available 1 internal bus clock after executing a read instruction from TMRx.
	WRITE	Writing a '1' enables the internal bus clock to decrement TCRx within 1 internal bus clock after executing a store instruction to TMRx.
(TMRx.reload) Timer Auto Reload Enable Bit 2	READ	Bit is available 1 internal bus clock after executing a read instruction from TMRx.
	WRITE	Writing a '1' enables the reload capability within 1 internal bus clock after the store instruction to TMRx has executed. The timer loads TRRx data into TCRx and decrements this value during the next internal bus clock cycle.
(TMRx.csel1:0) Timer Input Clock Select Bits 4-5	READ	Bits are available 1 internal bus clock after executing a read instruction from TMRx.csel1:0 bit(s).
	WRITE	The timer re-synchronizes the clock cycle used to decrement TCRx within one internal bus clock cycle after executing a store instruction to TMRx.csel1:0 bit(s).
(TCRx.d31:0) Timer Count Register	READ	The current TCRx count value is available within 1 internal bus clock cycle after executing a read instruction from TCRx. When the timer is running, the pre-decremented value is returned as the current value. When the timer is transferring the TRRx count into TCRx in the current count cycle, the timer returns the new TCRx count value to the executing read instruction.
	WRITE	The value written to TCRx becomes the active value within 1 internal bus clock cycle. When the timer is running, the value written is decremented in the current clock cycle.
(TRRx.d31:0) Timer Reload Register	READ	The current TRRx count value is available within 1 internal bus clock after executing a read instruction from TRRx.
	WRITE	The value written to TRRx becomes the active value stored in TRRx within 1 internal bus clock cycle. When the timer is transferring the TRRx value into the TCRx, data written to TRRx is also transferred into TCRx.

11.2 Timer Interrupts

Each timer is the source for one interrupt. When a timer detects a zero count in its TCRx, the timer generates an internal level-detected Timer Interrupt signal (TINTx) to the interrupt controller, and the interrupt source (INTSRC[1:0]) bit is set in the interrupt controller. Each timer interrupt can be selectively masked in the Interrupt Control (INTCTL[1:0]) registers. Refer to the Interrupt Controller Unit Chapter for a description of interrupt controller operation.

After servicing the timer interrupt, the interrupt service routine clears the pending request by writing a '1' to the appropriate bit of the Timer Interrupt Status Register (TISR).

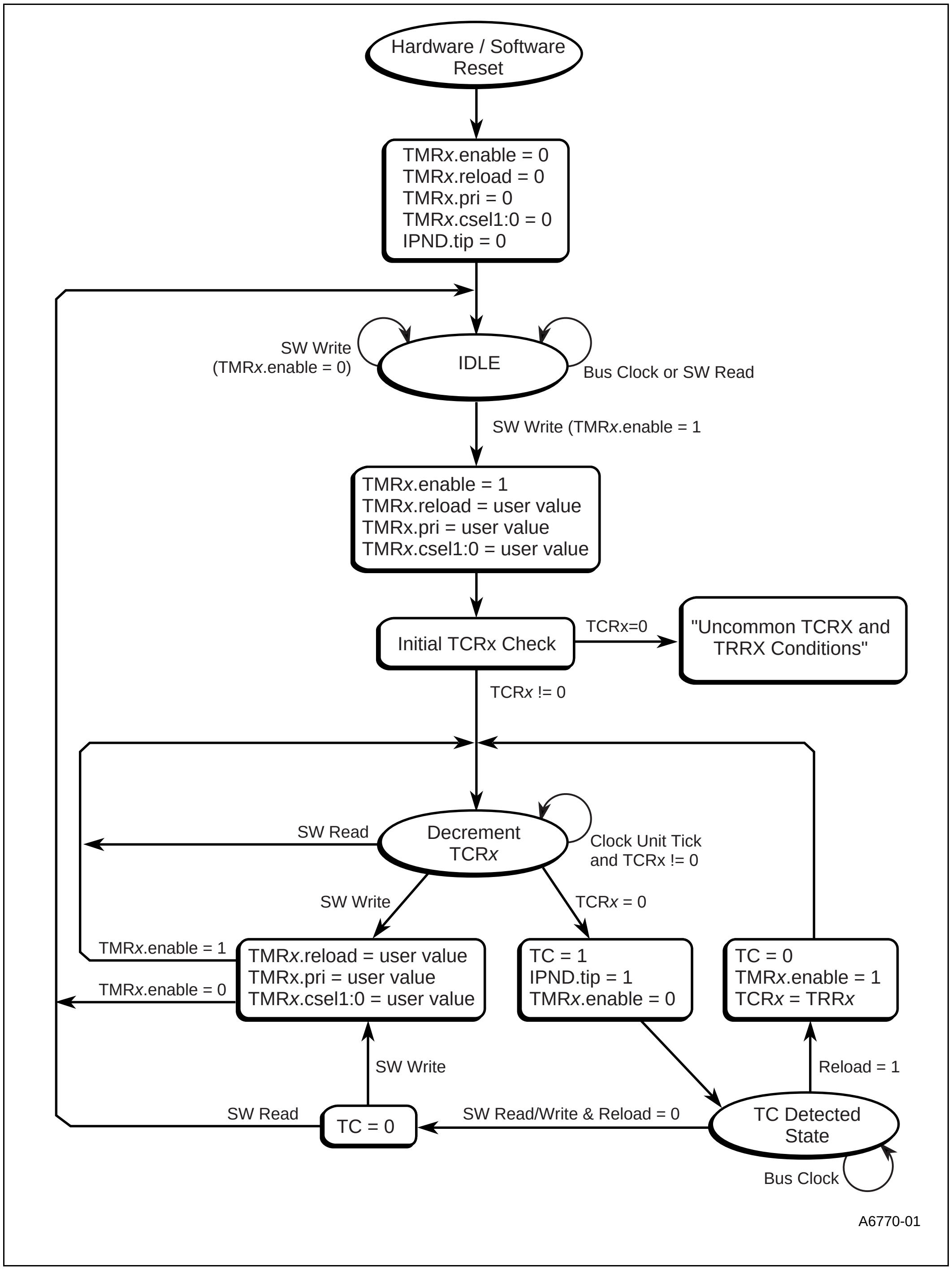
When a timer generates a second interrupt request before the CPU services the first interrupt request, the second request may be lost.

When auto-reload is enabled for a timer, the timer continues to decrement the value in TCRx even after entry into the timer interrupt handler.

11.3 Timer State Diagram

Figure 72 shows the common states of the Timer Unit. For uncommon conditions see Section 11.5, "Uncommon TCRX and TRRX Conditions" on page 640.

Figure 72. Timer Unit State Diagram



11.4 Timer Registers

As shown in Table 418, each timer has three co-processor registers:

- Timer Mode Register — programs the specific mode of operation or indicates the current programmed status of the timer. This register is described in Section 11.4.2, “Timer Mode Registers – TMR0:1” on page 634.
- Timer Count Register — contains the timer’s current count. See Section 11.4.3, “Timer Count Register – TCR0:1” on page 637.
- Timer Reload Register — contains the timer’s reload count. See Section 11.4.4, “Timer Reload Register – TRR0:1” on page 637.

Table 418. Timer Registers

Timer Unit	Register Acronym	Register Name
Timer 0	TMR0	Timer Mode Register 0
	TCR0	Timer Count Register 0
	TRR0	Timer Reload Register 0
Timer 1	TMR1	Timer Mode Register 1
	TCR1	Timer Count Register 1
	TRR1	Timer Reload Register 1

11.4.1 Power Up/Reset Initialization

Upon assertion of P_RST#, the timer registers are initialized to the values shown in Table 419.

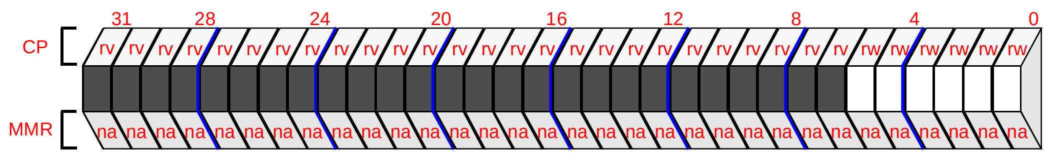
Table 419. Timer Power Up Mode Settings

Mode/Control Bit	Notes
TMRx.tc = 0	No terminal count
TMRx.enable = 0	Prevents counting and assertion of TINTx
TMRx.reload = 0	Single terminal count mode
TMRx.pri = 0	Privileged Mode and User Mode Writes Allowed
TMRx.csel1:0 = 0	Timer Clock = internal bus clock
TCRx.d31:0 = 0	Undefined
TRRx.d31:0 = 0	Undefined
TINTx output	Deasserted

11.4.2 Timer Mode Registers – TMR0:1

The Timer Mode Register (TMRx) lets the user program the mode of operation and determine the current status of the timer. TMRx bits are described in the subsections following Table 420 and are summarized in Table 416.

Table 420. Timer Mode Register – TMRx



Intel XScale® processor Coprocessor address

TMR0: CP6, Page 9, Register 0
TMR1: CP6, Page 9, Register 1

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

31:06	0000 000H	Reserved. Initialize to 0.
05:04	00_2	Timer Input Clock Selects — TMRx.csel1:0 (00) 1:1 Timer Clock = internal bus clock (01) 4:1 Timer Clock = internal bus clock / 4 (10) 8:1 Timer Clock = internal bus clock / 8 (11) 16:1 Timer Clock = internal bus clock / 16
03	0_2	Timer Register Privileged Write Control — TMRx.pri (0) Privileged and User Mode Write Enabled (1) Privileged Mode Only Write Enabled
02	0_2	Timer Auto Reload Enable — TMRx.reload (0) Auto Reload Disabled (1) Auto Reload Enabled
01	0_2	Timer Enable — TMRx.enable (0) Disabled (1) Enabled
00	0_2	Terminal Count Status — TMRx.tc (0) No Terminal Count (1) Terminal Count

11.4.2.1 Bit 0 — Terminal Count Status Bit (TMRx.tc)

The TMRx.tc bit is set when the Timer Count Register (TCRx) decrements to 0 and bit 2 (TMRx.reload) is not set for a timer. The TMRx.tc bit allows applications to monitor timer status through software instead of interrupts. TMRx.tc remains set until software accesses (reads or writes) TMRx. The access clears TMRx.tc. The timer ignores any value specified for TMRx.tc in a write request.

When auto-reload is selected for a timer and the timer is enabled, the TMRx.tc bit status is unpredictable. Software should not rely on the value of the TMRx.tc bit when auto-reload is enabled.

The processor also clears the TMRx.tc bit upon hardware or software reset. Refer to Section 17.2, "Reset Overview" on page 770.

11.4.2.2 Bit 1 — Timer Enable (TMRx.enable)

TMRx.enable bit allows user software to control the timer's RUN/STOP status. When:

TMRx.enable = 1	The Timer Count Register (TCRx) value decrements every Timer Clock (TCLOCK) cycle. TCLOCK is determined by the Timer Input Clock Select (TMRx.csel bits 0-1). See Section 11.4.2.5. When TMRx.reload=0, the timer automatically clears TMRx.enable when the count reaches zero. When TMRx.reload=1, the bit remains set. See Section 11.4.2.3.
TMRx.enable = 0	The timer is disabled and ignores all input transitions.

User software sets this bit. Once started, the timer continues to run, regardless of other processor activity. Three events can stop the timer:

- User software explicitly clearing this bit (i.e., TMRx.enable = 0).
- TCRx value decrements to 0, and Timer Auto Reload Enable (TMRx.reload) bit = 0.
- Hardware or software reset. Refer to Section 17.2, "Reset Overview" on page 770.

11.4.2.3 Bit 2 — Timer Auto Reload Enable (TMRx.reload)

The TMRx.reload bit determines whether the timer runs continuously or in single-shot mode. When TCRx = 0 and TMRx.enable = 1 and:

TMRx.reload = 1 The timer runs continuously. The processor:

1. Automatically loads TCRx with the value in the Timer Reload Register (TRRx), when TCRx value decrements to 0.
2. Decrements TCRx until it equals 0 again.

Steps 1 and 2 repeat until software clears TMRx bits 1 or 2.

TMRx.reload = 0 The timer runs until the Timer Count Register = 0. TRRx has no effect on the timer.

User software sets this bit. When TMRx.enable and TMRx.reload are set and TRRx does not equal 0, the timer continues to run in auto-reload mode, regardless of other processor activity. Two events can stop the timer:

- User software explicitly clearing either TMRx.enable or TMRx.reload.
- Hardware or software reset.

The processor clears this bit upon hardware or software reset.

11.4.2.4 Bit 3 — Timer Register Privileged Read/Write Control (TMRx.pri)

The TMRx.pri bit enables or disables user mode writes to the timer registers (TMRx, TCRx, TRRx). Privileged mode writes are allowed regardless of this bit's condition. Software can read these registers from either mode. Note that TMR1.pri also controls write access to the "Watch Dog Timer Control Register — WDTCR" on page 639 and the "Watch Dog Timer Setup Register — WDTSR" on page 639.

When:

TMRx.pri = 1	The timer ignores the user mode write to the timer registers; however, writes from the privileged modes are allowed.
TMRx.pri = 0	The timer registers can be written from either the user mode or the privileged modes.

The processor clears TMRx.pri upon hardware or software reset.

11.4.2.5 Bits 4, 5 — Timer Input Clock Select (TMRx.csel1:0)

User software programs the TMRx.csel bits to select the Timer Clock (TCLOCK) frequency. See Table 421. As shown in Figure 71, the internal bus clock is an input to the timer clock unit. These bits allow the application to specify whether TCLOCK runs at or slower than the internal bus clock frequency.

Table 421. Timer Input Clock (TCLOCK) Frequency Selection

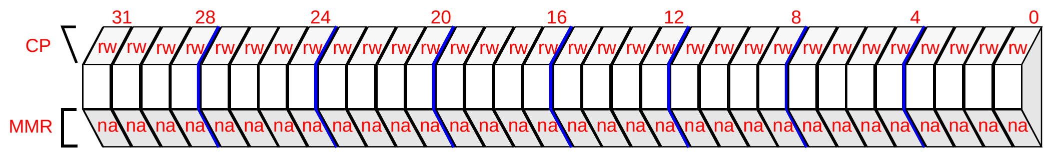
Bit 5 TMRx.csel1	Bit 4 TMRx.csel0	Timer Clock (TCLOCK)
0	0	Timer Clock = internal bus clock
0	1	Timer Clock = internal bus clock / 4
1	0	Timer Clock = internal bus clock / 8
1	1	Timer Clock = internal bus clock / 16

The processor clears these bits upon hardware or software reset (TCLOCK = Core Clock).

11.4.3 Timer Count Register – TCR0:1

The Timer Count Register (TCRx) is a 32-bit register that contains the timer's current count. The register value decrements with each timer clock tick. When this register value decrements to zero (terminal count), a timer interrupt is generated. When TMRx.reload is not set for the timer, the status bit in the timer mode register (TMRx.tc) is set and remains set until the TMRx register is accessed. Table 422 shows the timer count register.

Table 422. Timer Count Register – TCRx



Intel XScale® processor
Coprocessor address
TCR0: CP6, Page 9, Register 2
TCR1: CP6, Page 9, Register 3

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set

RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

31:00	0000 0000H	Timer Count Value — TCRx.d31:0

The valid programmable range is from 1H to FFFF FFFFH. Avoid programming TCRx to 0 as it has varying results as described in Section 11.5, "Uncommon TCRX and TRRX Conditions" on page 640. User software can read or write TCRx whether the timer is running or stopped. Bit 3 of TMRx determines user read/write control (Section 11.4.2.5). The TCRx value is undefined after hardware or software reset.

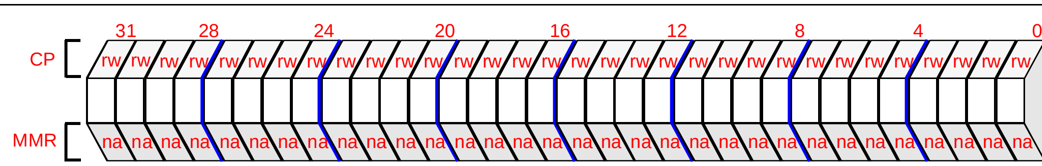
11.4.4 Timer Reload Register – TRR0:1

The Timer Reload Register (TRRx; Table 423) is a 32-bit register that contains the timer's reload count. The timer loads the reload count value into TCRx when TMRx.reload is set (1), TMRx.enable is set (1) and TCRx equals zero.

As with TCRx, the valid programmable range is from 1H to FFFF FFFFH. Avoid programming a value of 0, as it may prevent TINTx from asserting continuously. (See Section 11.5, "Uncommon TCRX and TRRX Conditions" on page 640 for more information.)

User software can access TRRx whether the timer is running or stopped. Bit 3 of TMRx determines read/write control (Section 11.4.2.5, "Bits 4, 5 — Timer Input Clock Select (TMRx.csel1:0)" on page 636). TRRx value is undefined after hardware or software reset.

Table 423. Timer Reload Register – TRRx



Intel XScale® processor
Coprocessor address
TRR0: CP6, Page 9, Register 4
TRR1: CP6, Page 9, Register 5

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set

RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

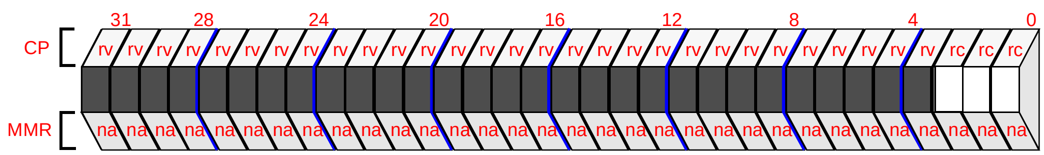
31:00	0000 0000H	Timer Auto-Reload Value — TRRx.d31:0

11.4.5 Timer Interrupt Status Register – TISR

The Timer Interrupt Status Register (TISR; Table 424) is a three-bit register that contains the timer's pending interrupt status and the Watchdog pending interrupt status (when enabled). The setting of these status bits represents the assertion of a "level-sensitive" interrupt request to the Interrupt Controller Unit. After the interrupt service routine completes processing of the interrupt request, it needs to write a '1' to the appropriate bit in the TISR to clear the pending request.

TISR interrupt requests are cleared after hardware or software reset.

Table 424. Timer Interrupt Status Register — TISR



Intel XScale® processor Coprocessor address
TISR: CP6, Page 9, Register 6

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

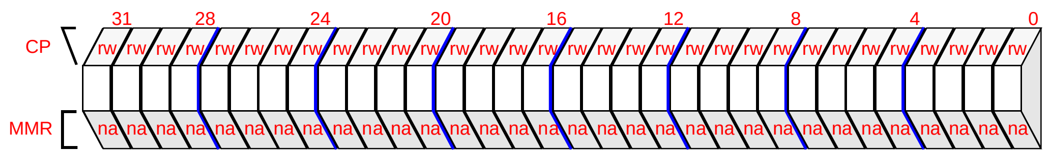
31:03	0000 0000H	Reserved.
02	0_2	Watchdog Timer Interrupt Pending — When set, there is an interrupt pending from the Watchdog timer. This occurs when Watchdog Timer detects a zero count in WDT. After servicing the interrupt, SW needs to write a '1' to this bit to clear the pending request. Note that the Watchdog timer must be setup to generate an interrupt. Refer to Section 426, "Watch Dog Timer Setup Register — WDTSR" on page 639.
01	0_2	Timer 1 Interrupt Pending — When set, there is an interrupt pending from Timer 1. This occurs when Timer 1 detects a zero count in TCR1. After servicing the interrupt, SW needs to write a '1' to this bit to clear the pending request.
00	0_2	Timer 0 Interrupt Pending — When set, there is an interrupt pending from Timer 0. This occurs when Timer 0 detects a zero count in TCR0. After servicing the interrupt, SW needs to write a '1' to this bit to clear the pending request.

11.4.6 Watch Dog Timer Control Register – WDTCR

The Watch Dog Timer Control Register (WDTCR) is a 32-bit register that software can use to enable the WDT or read the current WDT count value. The register value decrements with each internal bus clock tick. When this register value decrements to zero (terminal count), an Internal Bus Reset or an interrupt is generated. Refer to Section 426, “Watch Dog Timer Setup Register — WDTSR” on page 639.The timer can be enabled and/or reinitialized by writing 1E1E 1E1EH immediately followed by E1E1 E1E1H to the WDTCR. Once enabled, the WDT can be disabled by writing 1F1F 1F1FH immediately followed by F1F1 F1F1H to the WDTCR.

Note: This register is also controlled by the TMR1.pri (TMR1[3]) bit. For example, it can be written only by privileged processes.

Table 425. Watch Dog Timer Control Register — WDTCR



Intel XScale® processor Coprocessor address
WDTCR: CP6, Page 9, Register 7

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

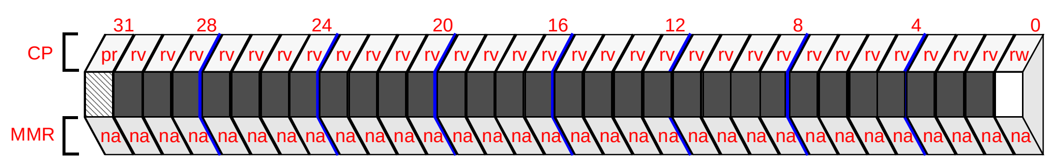
31:00	0000 0000H	Watch Dog Timer Count Value — By writing 1E1E 1E1EH followed by E1E1 E1E1H to this register, software can enable the WDT and reset the count value to FFFF FFFFH. When read, this register returns the current value contained in the WDT. Once enabled, the WDT can be disabled by writing 1F1F 1F1FH immediately followed by F1F1 F1F1H to the WDTCR

11.4.7 Watch Dog Timer Setup Register – WDTSR

The Watch Dog Timer Setup Register (WDTSR) is a 32-bit register that software can use to select the action taken when the WDT register value decrements to zero (terminal count) — either an interrupt or an internal bus reset can be generated.

Note: This register is also controlled by the TMR1.pri (TMR1[3]) bit. For example, it can be written only by privileged processes.

Table 426. Watch Dog Timer Setup Register — WDTSR



Intel XScale® processor Coprocessor address
WDTSR: CP6, Page 9, Register 8

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

31	0_2	Preserved.
30:01	0000 000H	Reserved.
00	0_2	Watch Dog Timer Event Selector bit — This bit selects whether the WDT generates an interrupt or an internal bus reset when the WDT register value decrements to zero (terminal count). When an interrupt is chosen, the interrupt is posted in the Timer Interrupt Status Register (TISR). Refer to Section 424, “Timer Interrupt Status Register – TISR” on page 638. 0 = An interrupt is generated 1 = A reset is generated

11.5 Uncommon TCRx and TRRx Conditions

Table 416 summarizes the most common settings for programming the timer registers. Under certain conditions, however, it may be useful to set the Timer Count Register or the Timer Reload Register to zero before enabling the timer. Table 427 details the conditions and results when these conditions are set.

Table 427. Uncommon TMRx Control Bit Settings

TRRx	TCRx	Bit 2 (TMRx.reload)	Bit 1 (TMRx.enable)	Action
X	0	0	1	TMRx.tc and TINTx set, TMR.enable cleared
0	0	1	1	Timer and auto reload enabled, TINTx not generated and timer enable remains set.
0	N	1	1	Timer and auto reload enabled. TINTx set when TCRx=0. The timer remains enabled but further TINTx's are not generated.

Note: X = don't care
N = a number between 1H and FFFF FFFFH

12.0 SMBus Interface Unit

This chapter describes the SMBus (System Management Bus) interface unit, including the operation modes and setup. Throughout this manual, this peripheral is referred to as the SMBus unit.

12.1 Overview

The SMBus Interface Units allows the Intel® 413808 and 413812 I/O Controllers in TPER Mode (4138xx) to serve as a slave device residing on the SMBus. The SMBus is a two-pin interface. **SMBDAT** is the data pin for input and output functions and **SMBCLK** is the clock pin for reference and control of the SMBus.

The SMBus allows the system to interface to 4138xx for system management functions. The serial bus requires a minimum of hardware for an economical system to relay status and reliability information of the 4138xx to the system.

The SMBus Interface Unit is a peripheral device that resides on a 4138xx internal bus. Data is transmitted to and received from the SMBus via a buffered interface. Control and status information is relayed through a set of registers. Refer to the SMBus Specification for complete details on SMBus operation.

12.2 SMBus Interface

SMBus provides for full access to registers in 4138xx including configuration and memory-mapped registers. Systems so configured can use the SMBus to access the registers. 4138xx supports a slave-only SMBus mode.

- *System Management Bus Specification*, Revision 2.0 (SMBus) Compliant.
- Slave mode operation only.
- Full read/write access to configuration and memory-mapped register spaces in 4138xx.

Table 428. SMBus Interface Pins

Signal	Pad Type
SMBCLK	SMBus Clock: Provides synchronous operation of the SMBus.
SMBDAT	SMBus Data: Used for data transfer and arbitration of the SMBus.
Total	2

12.3 System Management Bus Interface

This interface has no configuration registers associated with it. The SMBus address is set upon P_RST# by sampling the Peripheral Bus Interface Reset Strap inputs A[16:13]. When the pins are sampled, the resulting 4138xx address is stored in the Reset Strap Status Register and assigned as follows:

Bit	Value
7	1
6	1
5	A[16]
4	0
3	A[15]
2	A[14]
1	A[13]

The SMBus controller has access to all internal registers. It can perform reads and writes from all registers through the particular interface configuration space.

12.3.1 SMBus Controller

The 4138xx SMBus slave port interfaces to the configuration spaces of each ATU function, and also interfaces to the memory-mapped registers. This gives SM (server management) visibility into configuration space registers in the 4138xx ATUs 4138xx.

12.3.1.1 SMBus Commands

The 4138xx supports six SMBus commands:

- Block Write
- Block Read
- Word Write
- Word Read
- Byte Write
- Bytes Read

Sequencing these commands initiates internal accesses to 4138xx configuration and memory-mapped registers. For high reliability, 4138xx also supports the optional Packet Error Checking feature (CRC-8) and is enabled or disabled with each transaction.

Every configuration and memory read or write first consists of an SMBus write sequence which initializes the Bus Number, Device, function number, memory address offset etc. The term sequence is used since these variables can be initialized by the SMBus master with a single block write or multiple word or byte writes. The last write in the sequence that completes the initialization performs the internal configuration/memory read or write. The SMBus master can then initiate a read sequence which returns the status of the internal read or write command and also the data in case of a read.

Each SMBus transaction has an 8-bit command driven by the master. The command encodes the following information:

Table 429. SMBus Command Encoding

Bit	Description
7	Begin: The Begin bit when set indicates the first transaction of the read or write sequence.
6	End: The End bit when set indicates the last transaction of the read or write sequence.
5	Memory/Configure: Indicate whether memory or configuration space is being accesses in this SMBus sequence. Value of '1' indicates memory and a value of '0' indicate configuration.
4	PEC Enable: Indicates that PEC is enabled when set. When set, each transaction in the sequence ends with an extra CRC byte. 4138xx would check for CRC on writes and generate CRC on reads.
3:2	Internal Command: 00 — Read DWord 01 — Write Byte 10 — Write Word 11 — Write Dword All access are naturally aligned to the access width. This field specifies the internal command to be issued by the SMBus slave logic to the 4138xx
1:0	SMBus command: 00 — Byte 01 — Word 10 — Block 11 — Reserved This field specifies the SMBus command to be issued on the SMBus. This field is used as an indication of the length of transfer so that the slave knows when to expect the PEC packet (when enabled).

12.3.1.2 Initialization Sequence

All Configuration and memory reads and writes are accomplished through an SMBus write(s) and later followed by an SMBus read (for a read command). The SMBus write sequence is used to initialize the following for the configuration access:

- Bus Number (Bus Number is ignored on 4138xx
- Device/Function (Device Number is ignored on 4138xx
- 12-bit Register Number (in 2 separate bytes on SMBus)

Each of the parameters above is sent on SMBus in separate bytes. The register number parameter is initialized with two bytes and 4138xx ignores the most significant 4 bits of the second byte that initializes the register number.

For memory reads and writes, the write sequence initializes:

- a 19-bit memory address offset (in 3 separate bytes on SMBus)

On 4138xx memory transactions allow access to the Peripheral Memory-Mapped Registers only. All the Peripheral Memory-Mapped Registers on 4138xx are located in a 512-KByte contiguous memory space block that is relative to the PMMR Base Address Register (PMMRBAR). Refer to the Peripheral Registers Chapter for more details on the PMMRBAR register description. The 4138xx ignores the upper 5 bits of ADDR2 and the entire ADDR3 fields when memory transaction is selected. For example, an internal bus address is formed by concatenating the valid bits in the PMMRBAR and the 19 bits obtained from the SMBus in the ADDR2, ADDR1, and ADDR0 fields. Refer to Table 431, “SMBus Interface Registers for Memory Space Access” for the ADDRx fields.

The initialization of the information can be accomplished through any combination of the supported SMBus write commands (Block, Word or Byte). The Internal Command field for each write should specify the same internal command every time (read or write). After all the information is set up, the last write (End bit is set) initiates an internal read or write command. On an internal read when the data is not available before the slave interface acknowledges this last write command (ACK), the slave does a “clock stretch” until the data returns to the SMBus interface unit. On a internal write, when the write is not complete before the slave interface acknowledges this last write command (ACK), the slave “clock stretches” until the write completes internally. When an error occurs (address error, target or master abort on the internal bus) during the internal access, the last write command receives a NACK.

12.3.2 SMBus Signaling

12.3.2.1 Overview

The SMBus interface includes a pair of signals: **SMBCLK** (clock) and **SMBDAT** (serial data). **SMBCLK** provides the timing mechanism for data transfers. The SMBus master always drives **SMBCLK**. The 4138xx may optionally extend **SMBCLK** low time by driving it low to meet setup timings on the SMBus.

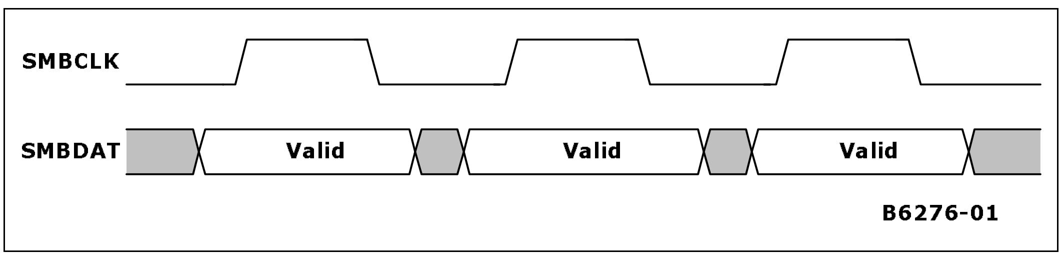
An initiator starts a transfer over the SMBus when it is free. Details of how initiators arbitrate are not described here. The current initiator communicates to the desired target through a unique 7-bit address to the target, sent MSb to LSb. All devices monitor the generated address after detecting the start condition. Once seven address bits are received, all targets compare the received address with their own and the target slave finds a match.

The next data bit from the initiator indicates the transfer direction. A value of '1' indicates that the target needs to transfer data to the initiator (read). Data transfers over SMBus are performed in 8-bit chunks. Data is transferred from MSb to LSb.

12.3.2.2 Waveforms

The timing relationship between **SMBDAT** and **SMBCLK** is defined such as the **SMBDAT** value must be valid through the duration of **SMBCLK** being in High state. The following diagram illustrates data transfer:

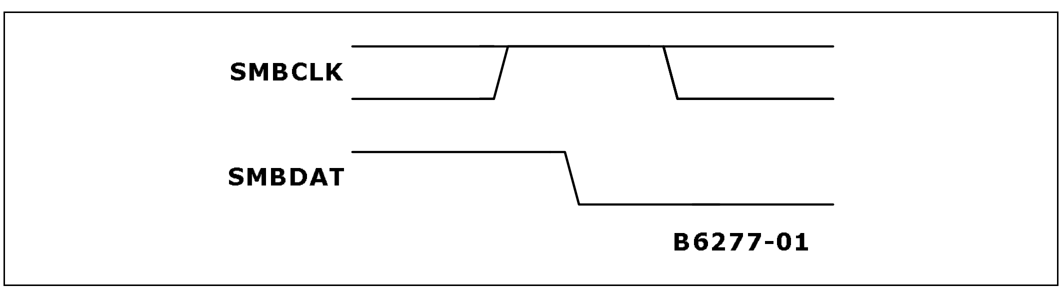
Figure 73. Basic SMBus Transfer Waveform



12.3.2.2.1 Start Phase

A start condition is generated when SMBus is idle to indicate that its state is changing to busy. The start condition occurs when **SMBDAT** transitions from High to Low while **SMBCLK** remains High. The SMBus protocol also allows a master to "Repeat Start", meaning that a new transfer is started by the same master, without a stop condition.

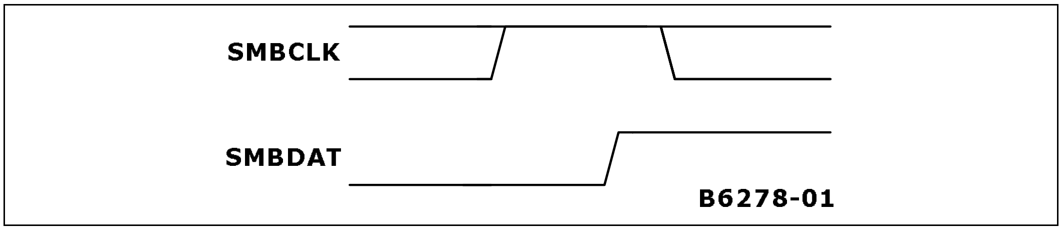
Figure 74. Start (S) / Repeat Start (Sr) Signaling



12.3.2.2.2 Stop Phase

A stop condition is generated when SMBus is busy to indicate that its state is changing to idle. The Stop condition occurs when **SMBDAT** transitions from Low to High while **SMBCLK** remains High.

Figure 75. Stop (P) Signaling



A stop bit can occur at any point in a data stream. It is not insured to occur after an ACK from a target (as later waveforms show). The 4138xx must be able to accept a stop condition at any time and clean up.

12.3.2.2.3 ACK/NACK

For every 8 bits of data transfer (including address and direction), the receiving agent must respond with ACK or NACK. An ACK is requires **SMBDAT** = 0 during **SMBCLK** = 1. A NACK requires **SMBDAT** = 1 during **SMBCLK** = 1 as shown below.

Figure 76. ACK (A) Signaling

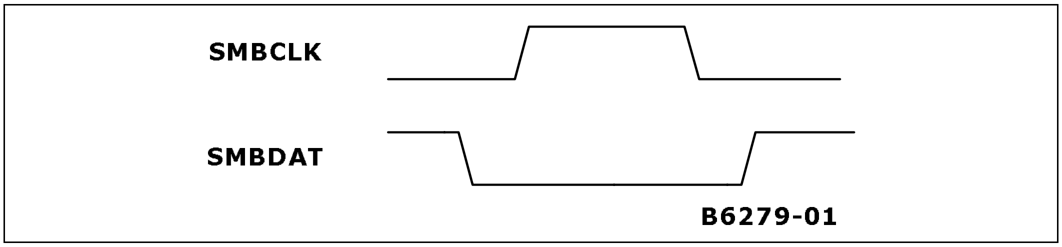
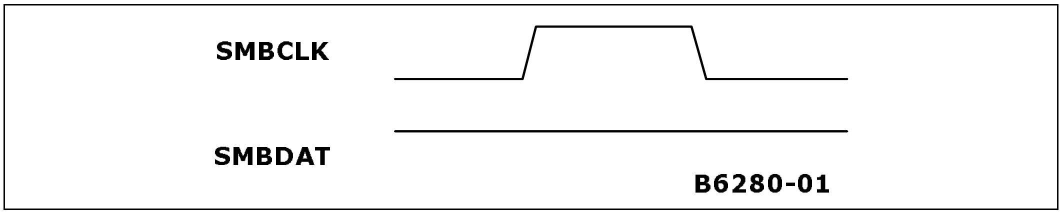


Figure 77. NACK (N) Signaling



During a write cycle, the 4138xx must drive an ACK after the address/direction phase, and after the data phase. During a read cycle, the 4138xx must drive an ACK/NACK after the address/direction phase, and (when ACKed) the initiator must drive an ACK/NACK after the 4138xx returns its 8 bits of data.

12.3.2.2.4 Wait States

The receiver (initiator or target) can add wait states, after driving ACK for receiving the last byte, by driving the **SMBCLK** line low. Further data transfers are delayed until the receiver stops driving **SMBCLK** low. It is expected the 4138xx drives the **SMBCLK** line low after receiving data on writes until the write is complete, and after receiving the direction bit on reads until the read data is ready.

12.3.3 Architecture

The 4138xx SMBus register interface consists of a set of registers that are only accessible from the SMBus interface only and are shown in Table 430 and Table 431. These registers are used to issue commands for reading and writing configuration registers and memory locations on 4138xx Table 430 shows that register format for accessing configuration space. Table 431 shows the register format for accessing memory space, The command register provides a bit that indicates whether configuration or memory space ought to be accessed.

Table 430. SMBus Interface Registers for Configuration Space Access

Register	Name and Function
CMD	Command
BYTCNT	Byte Count
ADDR3	Bus Number[a]
ADDR2	Device/Function Number.[b]
ADDR1	Extended Register Number — only bits [3:0]. The extended register allows access to 4-KByte configuration space.
ADDR0	Register Number — offset into function configuration space
DATA3	Data[31:24] — fourth byte of data.
DATA2	Data[23:16] — third byte of data.
DATA1	Data[15:8] — second byte of data.
DATA0	Data[7:0] — first byte of data.
STS	Status, only for reads.

a. The ADDR3 field is ignored as Bus Number is not applicable for 4138xx.
b. Only ADDR2[2:0] are used to select a Function Number. ADDR2[7:3] are ignored as Device Number is not applicable to 4138xx.

Table 431. SMBus Interface Registers for Memory Space Access

Register	Name and Function
CMD	Command
BYTCNT	Byte Count
ADDR3	Destination Memory.[a]
ADDR2	Address Offset — bits[23:16].[b]
ADDR1	Address Offset — bits[15:8].
ADDR0	Address Offset — bits[7:0].
DATA3	Data[31:24] — fourth byte of data.
DATA2	Data[23:16] — third byte of data.
DATA1	Data[15:8] — second byte of data.
DATA0	Data[7:0] — first byte of data.
STS	Status, only for reads.

a. The ADDR3 field is ignored on 4138xx as only the Memory-Mapped Register block are accessible as a memory space.
b. Only ADDR2[2:0] are used on 4138xx. ADDR2[7:3] are ignored as the Memory-Mapped Register Block only occupies 512 KBytes of memory space.

All SMBus accesses to internal register space are initiated via a write to the command register (CMD). The command register indicates the access type (read or write) and whether the command is targeting the configuration spaces or memory-mapped registers. Any register writes received by the 4138xx while a command is already in progress receive a NAK to prevent spurious operation. The master is no longer expected to poll the CMD register to prevent overwriting the current command in progress prior to issuing further writes. The SMBus access is delayed by stretching the clock until such time that the data is delivered. Note that per the SMBus specification, this cannot be longer than 25 ms. To set up an internal access, the command register write is followed by four ADDR byte writes. Depending on the type of access, these four bytes indicate either the Bus number, Device, Function, Extended Register Offset, and Register Offset, or the Memory-mapped region selected and the address within the region. The configuration type access utilizes the traditional bus number, device, function, and register offset; but in addition, also uses an extended register offset which expands the addressable register space from 256 bytes to 4 Kbytes. The memory-mapped type access redefines these bytes to be a memory-mapped region selection byte, a filler byte which today is all zeroes, and then the memory address within the region. Refer to the earlier tables, which display this information. Note that the filler byte is today not utilized but enforces that both types of accesses have the same number of address bytes, and allows for future expansion.

The Command Register (CMD) indicates the type and size of transfer. All configuration accesses from the SMBus port are initiated by writing to the Command register. While a command is in progress, all future writes or reads are NACKed by the 4138xx to avoid having registers overwritten while in use. There are two command size fields to allow for more flexibility on how the data payload is transferred, both internally and externally. Refer to the Command Register for more details on the size supported. The command register also provides a begin bit and an end bit. These two bits support the breaking of the SMBus transactions up into smaller transfers, by defining the start and finish of an overall transfer.

The 4138xx SMBus interface supports byte, word, or block transfer sizes. A byte size indicates that the data payload is transferred one byte at a time per SMBus transaction. A word size indicates that the data payload is transferred one word (2 bytes) at a time in a single SMBus transaction. A block size allows a variable size data payload to be transferred in a single SMBus transaction. When a block transfer is to be performed an 8-bit byte-count field follows the command register. This byte-count field provides the 4138xx target with the expected number of bytes to anticipate from the SMBus Master. For a block read transaction, the 4138xx returns a byte-count value to the master when returning read data. This provides the master with a byte-count to anticipate. Note that on 4138xx, the block transfer size is limited to only a maximum of four bytes.

12.3.3.1 Data Transfer Examples

For Figure 78 through Figure 85, the following terminology is used:

S	Start Bit
Sr	Start Repeat Bit
W	Write Command
R	Read Command
A	Acknowledge
N	Retry / not Acknowledge
P	Stop Bit

Clear boxes indicate phases of the cycle driven by the initiator, and shaded boxes indicate phases of the cycle driven by the target.

12.3.3.2 Configuration and Memory Reads

4138xx supports only read dword to internal register space. All Configuration and memory reads are accomplished through an SMBus write(s) and later followed by an SMBus read to read the status and the read data. For SMBus read transactions, the last byte of data (or the PEC byte when enabled), is NACKed by the master to indicate the end of the transaction. The SMBus memory read command returns the status of the previous internal command and the data associated previous internal read command. The status field encoding is:

Table 432. SMBus Status Byte Encoding

Bit	Description
7	Reserved
6	Internal Address Error
5	Internal Master Abort
4	Internal Target Abort
3:1	Reserved
0	Successful

Examples of configuration and memory reads are shown in Figure 78 through Figure 83. For the definition of the diagram conventions below, refer to the *System Management Bus Specification*, Revision 2.0.

Figure 78. DWORD Configuration Read Protocol (SMBus Block Write/Block Read, PEC Enabled)

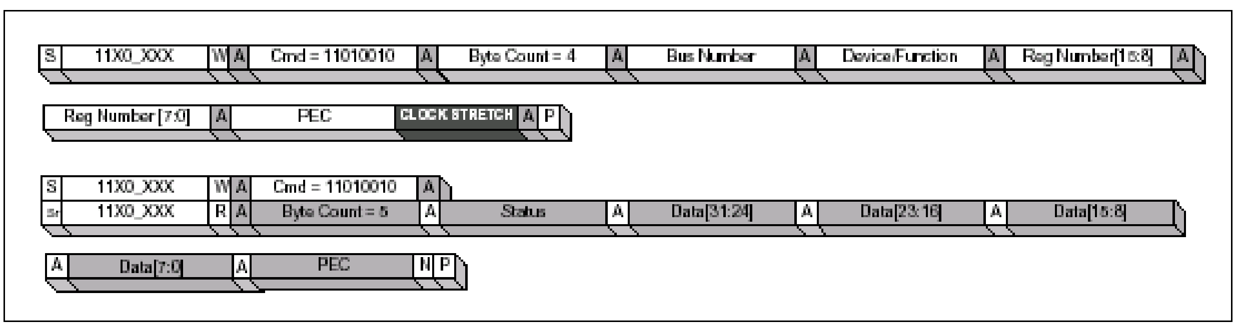


Figure 79. DWORD Memory Read Protocol (SMBus Block Write/Block Read, PEC Enabled)

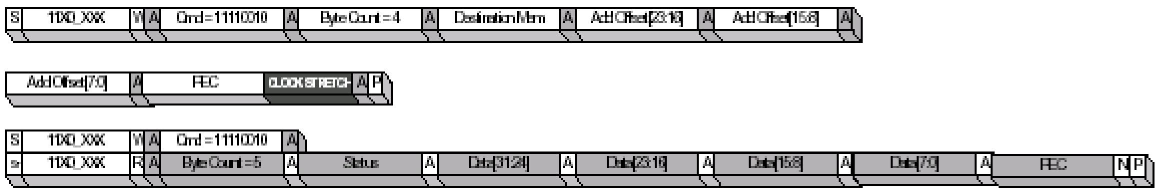


Figure 80. DWORD Configuration Read Protocol (SMBus Word Write/Word Read, PEC Enabled)

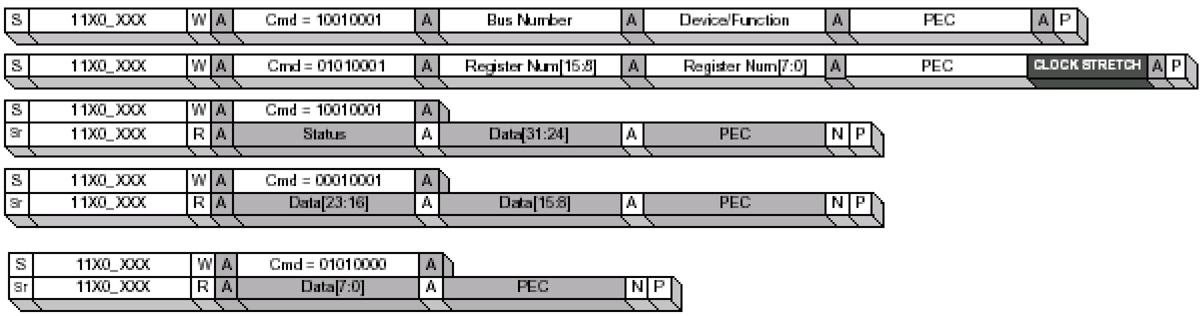


Figure 81. DWORD Configuration Read Protocol (SMBus Block Write/Block Read, PEC Disabled)

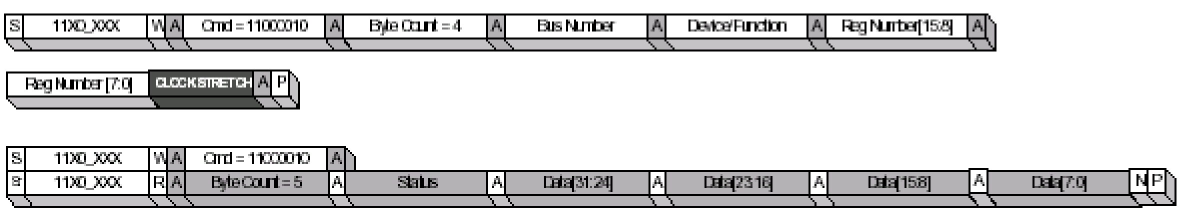


Figure 82. DWORD Memory Read Protocol (SMBus Block Write/Block Read, PEC Disabled)

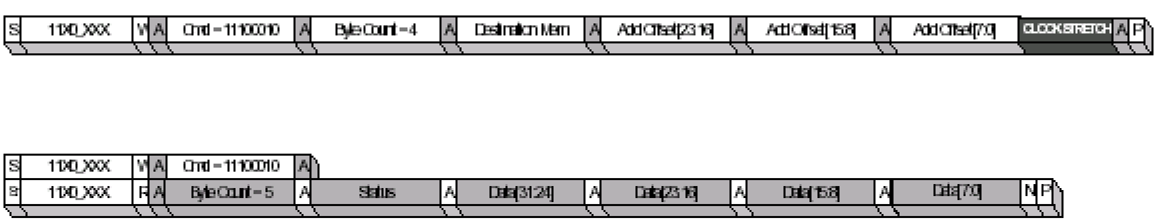


Figure 83. DWORD Configuration Read Protocol (SMBus Word Write/Word Read, PEC Disabled)

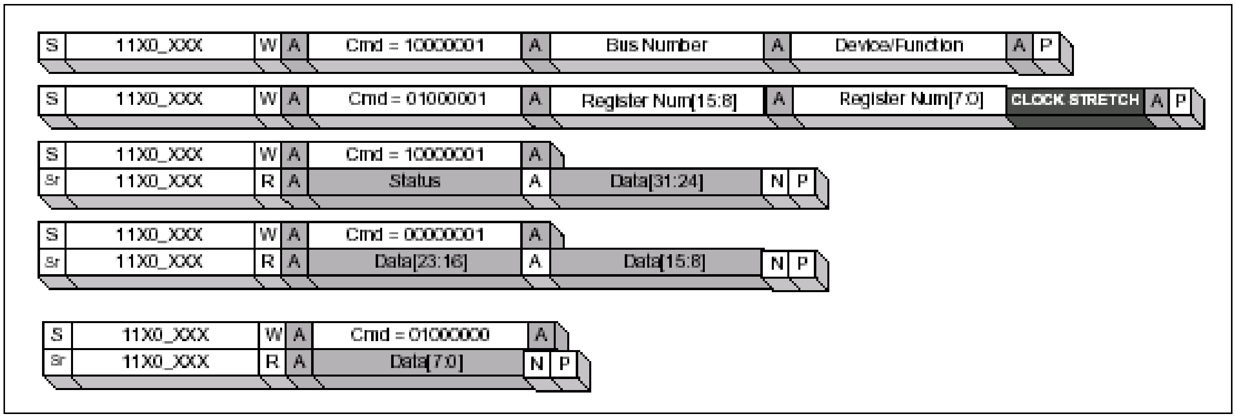


Figure 84. DWORD Memory Read Protocol (SMBus Word Write/(Word, Byte) Read, PEC Enabled)

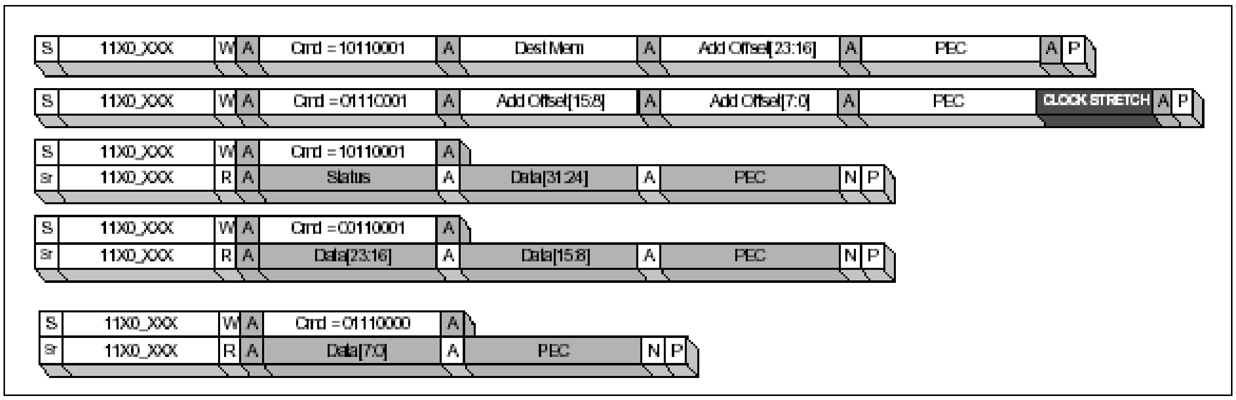
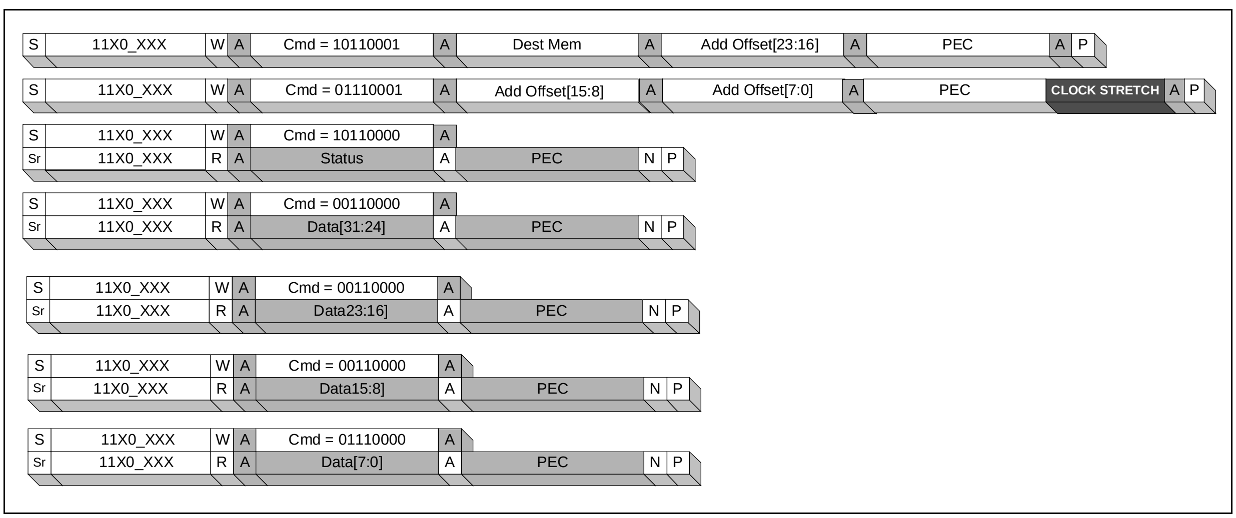


Figure 85. DWORD Memory Read Protocol (SMBus Word Write/Byte Read, PEC Enabled)



12.3.3.3 Configuration and Memory Writes

Configuration and memory writes are accomplished through a series of SMBus writes. As with reads, a write sequence is first used to initialize the Bus Number, Device, Function, and Register Number for the configuration access and the destination memory, address offset for the memory write. The writing of this information can be accomplished through any combination of the supported SMBus write commands (Block, Word or Byte).

Note: On SMBus, there is no concept of byte enables. Therefore, the Register Number written to the slave is assumed to be aligned to the length of the Internal Command. In other words, for a Write Byte internal command, the Register Number specifies the byte address. For a Write DWord internal command, the two least-significant bits of the Register Number are ignored. This is different from PCI where the byte enables are used to indicate the byte of interest.

After all the information is set up, the SMBus master initiates one or more writes which sets up the data to be written. The final write (End bit is set) initiates an internal configuration or memory write. The slave interface could potentially clock stretch the last data write until the write completes without error. When an error occurred, the SMBus interface NACKs the last write operation just before the stop bit. Examples of configuration writes are illustrated below. All the figures are with PEC Enabled. When PEC is disabled, there is no PEC byte in any of the sequences and the PEC enable bit in the command field is 0.

For the definition of the diagram conventions below, refer to the *System Management Bus Specification*, Revision 2.0.

Figure 86. DWORD Configuration Write Protocol (SMBus Block Write, PEC Enabled)

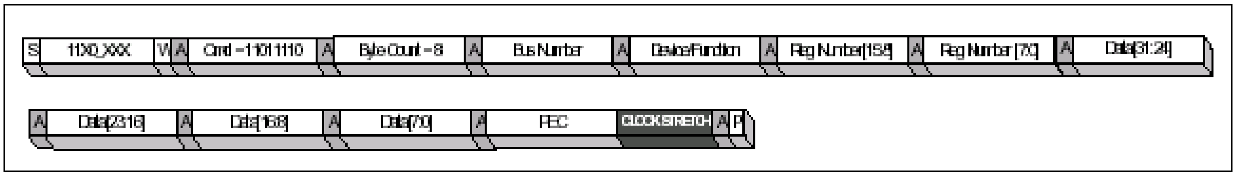


Figure 87. DWORD Memory Write Protocol (SMBus Word Write, PEC Enabled)

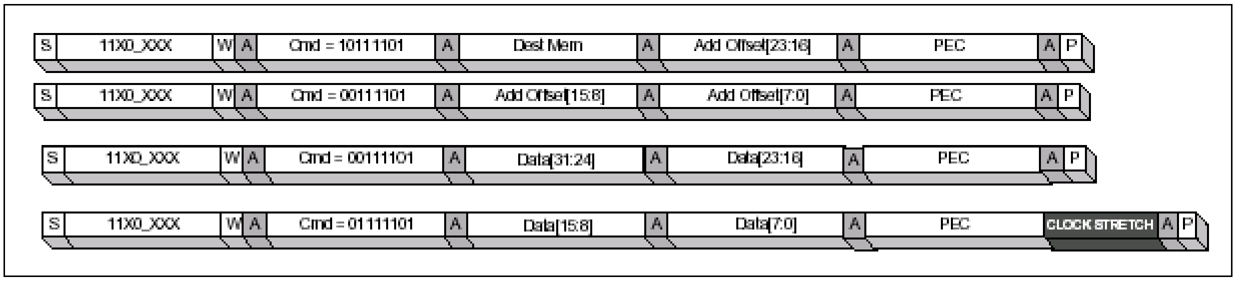
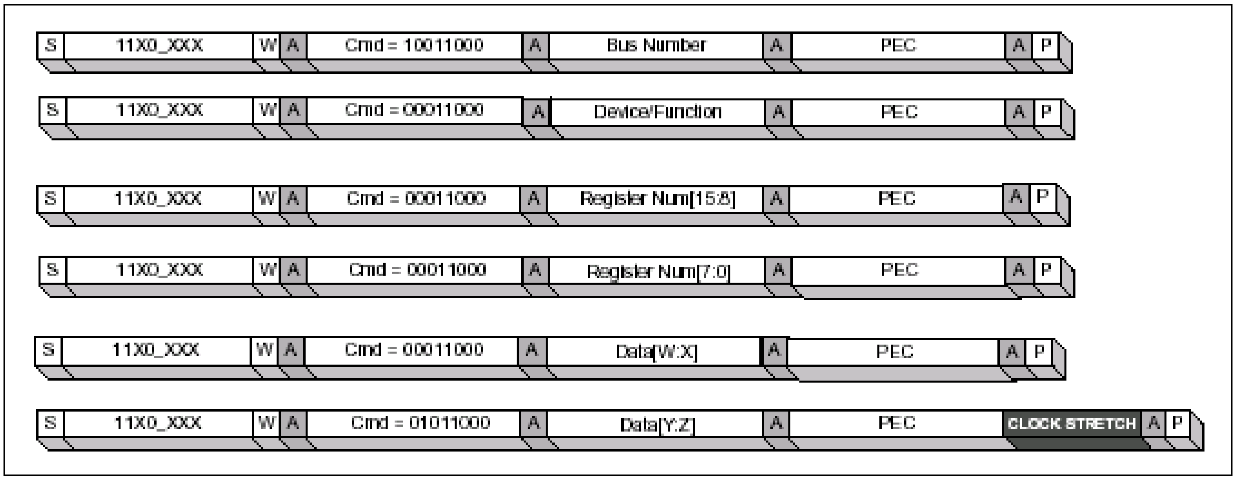


Figure 88. DWORD Configuration Write Protocol (SMBus Byte Write, PEC Enabled)



12.3.4 Error Handling

The SMBus slave interface handles two types of errors: internal address error and PEC. Internal address errors can occur, for example, when the SMBus request on the 4138xx internal bus fails due to an address parity error. This error manifests itself as a Not-Acknowledge (NACK) for the read or write command (End bit is set). Other internal errors include the read or write command receiving a master or target abort on the internal interface. When the master receives a NACK, the entire transaction should be reattempted.

When the master supports packet error checking (PEC) and the PEC enable bit in the command is set, then the PEC byte is checked in the slave interface. When the check indicates a failure, then the slave NACKs the PEC packet and does not issue the command on the internal interface.

An SMBus master must either do PEC on all transactions in a sequence or not do it at all; that is, it cannot turn on PEC in the middle of a sequence.

A PEC error in the middle of a sequence must be re-started from the beginning of the sequence; that is, the begin bit set.

12.3.5 SMBus Interface Reset

The master in two ways can reset the slave interface state machine in 4138xx:

The master holds **SMBCLK** low for 25 ms cumulative. Cumulative in this case means that all the “low time” for **SMBCLK** is counted between the Start and Stop bit. When this totals 25 ms before reaching the Stop bit, the interface is reset.

The master holds **SMBCLK** continuously high for 50 ms.

Besides these, the SMBus interface in 4138xx is also reset on a P_RST#, WARM_RST# or an in-band warm reset from PCI Express*.

12.4 Register Definitions

This section provides a summary descriptions of all the SMbus registers. The SMBus Interface Unit has eight registers which are accessible from the SMBus interface only. Firmware cannot access these registers on 4138xx.

Table 433. SMBus Register Summary

Section, Register Name, Acronym, Page
Section 434, "SMBus Controller Command Register — SM_CMD" on page 655
Section 435, "SMBus Controller Byte Count Register — SM_BC" on page 656
Section 436, "SMBus Controller ADDR3 Register — SM_ADDR3" on page 656
Section 437, "SMBus Controller ADDR2 Register — SM_ADDR2" on page 656
Section 438, "SMBus Controller ADDR1 Register Number — SM_ADDR1" on page 657
Section 439, "SMBus Controller ADDR0 Register Number — SM_ADDR0" on page 657
Section 440, "SMBus Controller Data Register — SM_DATA" on page 658
Section 441, "SMBus Controller Status Register — SM_STS" on page 658

12.4.1 SMBus Controller Command Register — SM_CMD

This is the Command Register. All accesses from the SMBus port are initiated by writing to this register. While a command is in progress, all future writes or reads are NACKed by the 4138xx to avoid having registers overwritten.

Table 434. SMBus Controller Command Register — SM_CMD

Bit	Reset	Description
07	0	Begin: The Begin bit when set indicates the first transaction of the read or write sequence.
06	0	End: The End bit when set indicates the last transaction of the read or write sequence.
05	0	Memory/Configure: Indicate whether memory or configuration space is being accesses in this SMBus sequence. Value of '1' indicates memory and a value of '0' indicate configuration.
04	0	PEC Enable: Indicates that PEC is enabled when set. When set, each transaction in the sequence ends with an extra CRC byte. 4138xx would check for CRC on writes and generate CRC on reads.
03:02	00	Internal Command: 00 — Read DWord 01 — Write Byte 10 — Write Word 11 — Write Dword All access are naturally aligned to the access width. This field specifies the internal command to be issued by the SMBus slave logic to the 4138xx
01:00	00	SMBus command: 00 — Byte 01 — Word 10 — Block 11 — Reserved This field specifies the SMBus command to be issued on the SMBus. This field is used as an indication of the length of transfer so that the slave knows when to expect the PEC packet (when enabled).

12.4.2 SMBus Controller Byte Count Register — SM_BC

The SM_BC register indicates the number of bytes following the command field when performing a write or when setting up for a read. The byte count is also used when returning the data to indicate the following bytes including the status byte which is returned prior to the data. Note that the byte count is only transmitted for block type accesses on SMBus. SMBus word or byte accesses do not use the byte count register.

Table 435. SMBus Controller Byte Count Register — SM_BC

Bit	Reset	Description
07:00	00H	Byte Count: Indicates the number of bytes to anticipate for block size transfers. Not used for byte and word size transfers.

12.4.3 SMBus Controller ADDR3 Register — SM_ADDR3

The SM_ADDR3 register should be programmed with the Bus Number of the desired configuration register. The Status Register should be checked to make sure that there is not a command currently in progress, before writing to this register. Writing to this register when the 'Busy' bit in the Status Register is asserted has indeterminate effects. When accessing memory, the SM_ADDR3 register is ignored by 4138xx.

Table 436. SMBus Controller ADDR3 Register — SM_ADDR3

Bit	Reset	Description
07:00	00H	ADDR3: Indicates the bus number to access when accessing configuration registers. Not used with memory access.

12.4.4 SMBus Controller ADDR2 Register — SM_ADDR2

This register should be programmed with the Device Number and Function Number of the desired configuration register. The Status Register should be checked to make sure that there is not a command currently in progress, before writing to this register. Writing to this register when the 'Busy' bit in the Status Register is asserted has indeterminate effects. When accessing memory, bits 0, 1 and 2 of the SM_ADDR2 register provides address bits [16,17,18] of the memory address offset. The upper 5 bits of SM_ADDR2 register are ignored by 4138xx.

Table 437. SMBus Controller ADDR2 Register — SM_ADDR2

Bit	Reset	Description
07:03	00H	Device Number (DEV): Device number of device to access.
02:00	000	Function Number (FNC): Function number of device to access. For memory access, bit 0 represents bit 16 of the address offset.

12.4.5 SMBus Controller ADDR1 Register Number — SM_ADDR1

This register should be programmed with the upper address bits (bits [11:8]) of the Register Number of the desired configuration register for 4-KByte configuration space. 4138xx ignores bit [7:4] of this register. The Status Register should be checked to make sure that there is not a command currently in progress, before writing to this register. Writing to this register when the 'Busy' bit in the Status Register is asserted has indeterminate effects. When accessing memory space, ADDR1 provides bits [15:8] of the memory address offset.

Table 438. SMBus Controller ADDR1 Register Number — SM_ADDR1

Bit	Reset	Description
07:00	00H	Upper Number (UNUM): Indicates the upper 4 bits (bits [11:8] of the register number to access for 4-KByte configuration space. Bits [7:4] of this registers is ignored by 4138xx. For memory access, this register provides bits [15:8] of the memory address offset.

12.4.6 SMBus Controller ADDR0 Register Number — SM_ADDR0

This register should be programmed with the lower address bits (bits [7:0]) of the Register Number of the desired configuration register for 4-KByte configuration space. The Status Register should be checked to make sure that there is not a command currently in progress, before writing to this register. Writing to this register when the 'Busy' bit in the Status Register is asserted has indeterminate effects. When accessing memory space, SM_ADDR0 provides bits [7:0] of the memory address offset.

Table 439. SMBus Controller ADDR0 Register Number — SM_ADDR0

Bit	Reset	Description
07:00	00H	Lower Number (LNUM): Indicates the lower 8 bits (bits [7:0] of the register number to access for 4-KByte configuration space. For memory access, this register provides bits [7:0] of the memory address offset.

12.4.7 SMBus Controller Data Register — SM_DATA

This register is used to read or write data to the desired Configuration Register.

At the completion of a Read command, this register contains the data from the selected configuration register. For reads the data register always returns 32 bits and is always aligned on a DWORD boundary.

Before issuing a write command this register should be written with the desired write data. For a byte, only the D[7:0] data is written to the desired configuration register. For a word write, only the D[15:0] data is written to the desired configuration register. The register number must be word aligned for word writes. For a DWORD write, all 32 bits of data are used. The register number must be DWORD aligned.

The Status Register should be checked to make sure that there is not a command currently in progress, before writing to this register. Writing to this register when the Busy bit in the Status Register is asserted, has indeterminate effects.

Table 440. SMBus Controller Data Register — SM_DATA

Bit	Reset	Description
31:24	00H	Byte 3 (B3): Data bits [31:24].
23:16	00H	Byte 2 (B2): Data bits [23:16].
15:08	00H	Byte 1 (B1): Data bits [15:8].
07:00	00H	Byte 0 (B0): Data bits [7:0].

12.4.8 SMBus Controller Status Register — SM_STS

The SM_STS Register provides the status of the internal transaction. For an SMBus read transaction, the data is preceded by a byte of status.

Table 441. SMBus Controller Status Register — SM_STS

Bit	Reset	Description
07	0	Reserved
06	0	Internal Address Error
05	0	Internal Master Abort
04	000	Internal Target Abort
03:01	0	Reserved
00	000	Successful

Notes:
1. An Address Error is signalled when an address parity is detected. The error is logged in the System Controller.

13.0 UARTs

Note: UART0 is owned by the Transport Core. See the System/Software Architecture Specification for details on how to change this.

This chapter describes the Universal Asynchronous Receiver/Transmitter (UART) serial ports. The Intel® 413808 and 413812 I/O Controllers in TPER Mode (4138xx) UARTs are controlled via programmed I/O through memory-mapped registers.

13.1 Overview

Each asynchronous serial port supports all the functions of 16550 UART. Each UART performs serial-to-parallel conversion on data characters received from a peripheral device or a modem and parallel-to-serial conversion on data characters received from the processor. The processor can read the complete status of a UART at any time during the functional operation. Available status information includes the type and condition of the transfer operations being performed by a UART and any error conditions (parity, overrun, framing, or break interrupt).

Each serial port can operate in either FIFO or non-FIFO mode. In FIFO mode, a 64-byte transmit FIFO holds data from the processor to be transmitted on the serial link and a 64-byte Receive FIFO buffers data from the serial link until read by the processor.

Each UART includes a programmable baud rate generator which is capable of dividing the input clock by divisors of 1 to (2^{16}–1) and producing a 16X clock to drive the internal transmitter and receiver logic. Interrupts can be programmed to the user's requirements, minimizing the computing required to handle the communications link. Each UART operates in a polled or an interrupt driven environment which is selected by software.

The UART hardware is responsible for executing serial protocol communication and for providing the programming interface. The UART features include:

- Registers are compatible with the 16550 and 16750
- Adds or deletes standard asynchronous communications bits (start, stop, and parity) to or from the serial data
- Independently controlled transmit, receive, line status and data set interrupts
- Baud-rate generator allows division of clock by 1 to ($2^{16}-1$) and generates an internal 16X clock; baud-rate can be manually or automatically programmed via auto-baud-rate detection circuitry
- Modem control functions (CTS#, RTS#)
- Autoflow capability controls data I/O without generating Interrupts:
  - RTS# (output) controlled by UART Receiver FIFO
  - CTS# (input) from modem controls UART transmitter
- Fully programmable serial-interface characteristics:
  - 5, 6, 7 or 8-bit characters
  - Even, odd, or no parity detection
  - 1, 1-1/2, or 2 stop bit generation
  - Baud rate generation (up to 115kbps)
- False start bit detection
- 64-byte Transmit FIFO
- 64-byte Receive FIFO with programmable threshold
- Complete status reporting capability
- Break generation and detection
- Internal diagnostic capabilities include:
  - Loopback controls for communications link fault isolation
  - Break, parity, overrun, and framing error simulation
- Fully prioritized interrupt system controls

13.1.1 Compatibility with 16550 and 16750

The UARTs can be programmed to be functionally compatible with industry standard 16550 and 16750. Each UART supports most of the 16550 and 16750 functions and has additional features, as listed below.

- DMA requests for transmit and receive data services
- NRZ encoding/decoding function
- 64 byte Transmit/Receive FIFO buffers
- Programmable Receive FIFO threshold
- Auto baud-rate detection
- Auto flow

13.2 Signal Descriptions

The name and description of external signals connected to a UART module are shown in Table 442.

Table 442. UART Signal Descriptions

Name*	Type	Description
Ux_RXD	Input	**SERIAL INPUT:** Serial data input from device pin to the receive shift register.
Ux_TXD	Output	**SERIAL OUTPUT:** Composite serial data output to the communications link-peripheral, modem, or data set. The TXD signal is set to the MARKING (logic 1) state upon a Reset operation.
Ux_CTS#	Input	**CLEAR TO SEND:** When low, this pin indicates that the receiving UART is ready to receive data. When the receiving UART deasserts **CTS#** high, the transmitting UART should stop transmission to prevent overflow of the receiving UARTs buffer. The **CTS#** signal is a modem-status input whose condition can be tested by the host processor or by the UART when in Autoflow mode as described below: **Non-Autoflow Mode:** When not in Autoflow mode, bit 4 (CTS) of the Modem Status register (MSR) indicates the state of **CTS#**. Bit 4 is the complement of the **CTS#** signal. Bit 0 (DCTS) of the Modem Status register indicates whether the **CTS#** input has changed state since the previous reading of the Modem Status register. **CTS#** has no effect on the transmitter. The user can program the UART to interrupt the processor when DCTS changes state. The programmer can then stall the outgoing data stream by starving the transmit FIFO or disabling the UART with the IER register. ***Note:*** When UART transmission is stalled by disabling the UART, the user does not receive an MSR interrupt when **CTS#** reasserts. This is because disabling the UART also disables interrupts. To get around this, the user can use Auto CTS in Autoflow Mode, or program the **CTS#** pin to interrupt. **Autoflow Mode:** In Autoflow mode, the UART Transmit circuity checks the state of **CTS#** before transmitting each byte. When **CTS#** is high, no data is transmitted. See Section 13.4.7, UART x Modem Control Register for more information on Auto CTS mode.
Ux_RTS#	Output	**REQUEST TO SEND:** When low, this informs the remote device that the UART is ready to receive data. A reset operation sets this signal to its Inactive (high) state. LOOP mode operation holds this signal in its Inactive state. **Non-Autoflow Mode:** The **RTS#** output signal can be asserted by setting bit 1 (RTS) of the Modem Control register to a 1. The RTS bit is the complement of the **RTS#** signal. **Autoflow Mode: RTS#** is automatically asserted by the autoflow circuitry when the Receive buffer exceeds its programmed threshold. It is deasserted when enough bytes are removed from the buffer to lower the data level back to the threshold. See Section 13.4.7, UART x Modem Control Register for more information on Auto RTS mode.

Note: * "x" in signal name replaced with either "0" or "1" for UART-0 or UART-1 respectively.

13.3 Theory of Operation

The format of a UART data frame is shown in Figure 89.

Figure 89. Example UART Data Frame

	Start Bit	Data<0>	Data<1>	Data<2>	Data<3>	Data<4>	Data<5>	Data<6>	Data<7>	Parity Bit	Stop Bit 1	Stop Bit 2	
		LSB							MSB				

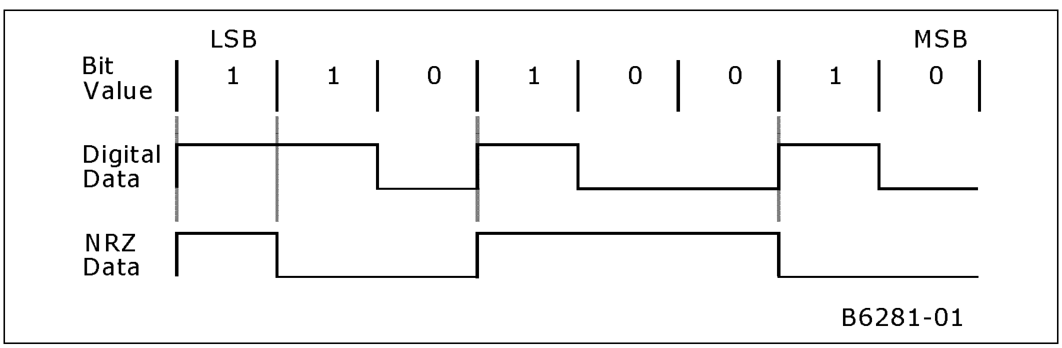
TXD or RXD pin

Shaded bits are optional and can be programmed by user. —>

Each data frame is between 7 bits and 12 bits long, depending on the size of data programmed and when parity and stop bits are enabled. The frame begins with a start bit that is represented by a high-to-low transition. Next, five to eight bits of data are transmitted, beginning with the least significant bit. An optional parity bit follows, which is set when even parity is enabled and an odd number of ones exist within the data byte; or when odd parity is enabled and the data byte contains an even number of ones. The data frame ends with one, one-and-one-half or two stop bits (as programmed by the user), which is represented by one or two successive bit periods of a logic one.

NRZ coding can be used by the UART to represent individual bit values. NRZ coding is enabled when Interrupt Enable Register (IER) bit-5 is set to high. A one is represented by a line transition and a zero is represented by no line transition. Figure 90 shows the NRZ coding of the data byte 8b 0100 1011. Note that the byte's LSB is transmitted first.

Figure 90. NRZ Bit Encoding Example – (0100 1011)



The unit is disabled upon reset, and users need to enable the unit by setting the UART Unit Enable bit (UUE, bit-6) of Interrupt Enable Register. When the unit is enabled, the receiver starts looking for the start bit of a frame; the transmitter sends data to the transmit data pin when there is data available in the transmit FIFO. Transmit data can be written to the FIFO before the unit is enabled. When the UART is disabled, the transmitter/receiver finishes the current byte being transmitted/received (when it is in the middle of transmitting/receiving a byte), and stops transmitting/receiving more data. Disabling the UART with the UUE bit does not clear transmission/reception with the original data.

Each UART has a Transmit FIFO and a Receive FIFO each holding 64 characters of data. There are two methods for moving data into/out of the FIFOs: **Interrupts and Polling**.

13.3.1 FIFO Interrupt Mode Operation

13.3.1.1 Receiver Interrupt

When the Receive FIFO and receiver interrupts are enabled (FCR[0]=1 and IER[0]=1), receiver interrupts occur as follows:

- The Receive Data Available Interrupt is asserted when the FIFO has reached its programmed trigger level. The interrupt is cleared when the FIFO drops below the programmed trigger level.
- The IIR Receive Data Available indication also occurs when the FIFO trigger level is reached, and like the interrupt, the bits are cleared when the FIFO drops below the trigger level.
- The Data Ready bit (DR in LSR register) is set to 1 as soon as a character is transferred from the shift register to the Receive FIFO. This bit is reset to 0 when the FIFO is empty.

13.3.1.2 Transmit Interrupt

When the transmitter FIFO and transmitter interrupt are enabled (FCR[0]=1, IER[1]=1), transmit interrupts occur as follows:

- When the Flow Control Register Transmitter Interrupt Level (TIL) bit (FCR[3]) is clear (0), The Transmit Data Request interrupt occurs when the transmit FIFO is half empty or more than half empty. The interrupt is cleared when the data level exceeds the half-empty mark. The interrupt is cleared as soon as the Transmit Holding Register is written or the IIR is read. 1 to 32 characters may be written to the transmit FIFO while servicing the interrupt when TIL=0.
- When the Flow Control Register Transmitter Interrupt Level (TIL) bit is set (1), The Transmit Data Request Interrupt occurs when the Transmit FIFO is empty. The interrupt is cleared as soon as the Transmit Holding Register is written or the IIR is read. 1 to 64 characters may be written to the Transmit FIFO while servicing the interrupt when TIL = 1.

Users could cause the UART Transmit FIFO to overflow when too many characters are written. FIFO underflow does not cause an error as the UART waits for the Transmit FIFO to be serviced.

13.3.2 Removing Trailing Bytes In Interrupt Mode

When the number of entries in the Receive FIFO is less than its trigger level, and no additional data is received, the remaining bytes are called trailing bytes. When the receive FIFO is being serviced by processor interrupts, trailing bytes need to be removed via the processor using the 16550 compliant character time-out interrupt: Time Out Detected (TOD) bit of Interrupt Identification Register. To enter this mode, users need to insure that the character time-out interrupt is enabled via IER[4].

To remove trailing bytes in Interrupt mode, the user must wait for the character time-out interrupt and then read all remaining bytes as indicated in the FIFO Occupancy Register (FOR), or read one byte at a time until the FIFO is empty. This can be determined by polling the Line Status Register bit 0 through programmed I/O.

13.3.2.1 Character Time-out Interrupt

When the Receiver FIFO and Receiver Time-out Interrupt are enabled, a character time-out interrupt (TOD) occurs to signal the presence of trailing bytes. The Interrupt is cleared and the timer is reset when a character is read from the Receiver FIFO. When a time-out Interrupt has not occurred, the time-out timer is reset after a new character is received or after the processor reads the Receiver FIFO.

When enabled via IER[4], a character time-out occurs under the following conditions:

- At least one character is in the FIFO.
- A character has not been received for the amount of time it takes to receive four or more characters at the current baud rate.
- The FIFO has not been read for the amount of time it takes to receive four or more characters

13.3.3 FIFO Polled Mode Operation

With the FIFOs enabled (TRFIFOE bit of FCR set to 1), clearing IER[7] and IER[4:0] puts the serial port in the FIFO polled mode of operation. Since the receiver and the transmitter are controlled separately, either one or both can be in the Polled Operation mode. In this mode, software checks Receiver and Transmitter status via the LSR. The processor polls the following bits for Receive and Transmit Data Service.

13.3.3.1 Receive Data Service

- Processor should check *Data Ready bit of LSR* which is set when 1 or more bytes remains in the Receive FIFO or Receive Buffer register (RBR).

13.3.3.2 Transmit Data Service

- Processor should check *Transmit Data Request bit of LSR* which is set when transmitter needs data.
- • Processor can also check *Transmitter Empty bit of LSR*, which is set when the Transmit FIFO or Holding register is empty.

13.3.4 Autoflow Control

Autoflow Control uses the Clear-to-Send (**CTS#**) and Request-to-Send (**RTS#**) signals to automatically control the flow of data between the UART and external modem. When autoflow is enabled, the remote device is not allowed to send data unless the UART asserts nRTS low. When the UART deasserts **RTS#** while the remote device is sending data, the remote device is allowed to send one additional byte after **RTS#** is deasserted. An overflow could occur when the remote device violates this rule. Likewise, the UART is not allowed to transmit data unless the remote device asserts **CTS#** low. This feature increases system efficiency and eliminates the possibility of a Receive FIFO Overflow error due to long Interrupt latency.

Autoflow mode can be used in two ways: **Full autoflow,** automating both **CTS#** and **RTS#**, and **half autoflow,** automating only **CTS#**. Full Autoflow is enabled by writing a 1 to bits 1 and 5 of the Modem Control register (MCR). Auto-CTS-Only mode is enabled by writing a 1 to bit 5 and a 0 to bit 1 of the MCR register.

13.3.4.1 RTS Autoflow

When in full autoflow mode, **RTS#** is asserted when the UART FIFO is ready to receive data from the remote transmitter. This occurs when the amount of data in the Receive FIFO is below the programmable threshold value. When the amount of data in the Receive FIFO reaches the programmable threshold, **RTS#** is deasserted. It is asserted once again when enough bytes are removed from the FIFO to lower the data level below the threshold.

13.3.4.2 CTS Autoflow

When in Full or Half-Autoflow mode, **CTS#** is asserted by the remote receiver when the receiver is ready to receive data from the UART. The UART checks **CTS#** before sending the next byte of data and does not transmit the byte until **CTS#** is low. When **CTS#** goes high while the transfer of a byte is in progress, the transmitter completes this byte.

Note: Autoflow mode can be used only in conjunction with FIFO mode.

13.3.5 Auto-Baud-Rate Detection

Each UART supports auto-baud-rate detection. When enabled, UART counts the number of 33.334 MHz clock cycles within the start-bit pulse. This number is then written into the Auto-Baud-Count register (ACR) and used to calculate the baud rate. When ACR is written, an Auto-Baud-Lock Interrupt is generated (when enabled), and the UART automatically programs the Divisor Latch registers with the appropriate baud rate. When preferred, the processor can read the Auto-Baud- Count register using this information to program the Divisor-Latch registers with a baud rate calculated by the processor. After the baud rate has been programmed, the processor is responsible to verify that the predetermined characters (usually AT or at) are being received correctly.

When the UART programs Divisor Latch registers, users can choose between two auto-baud calculation methods: table- and formula-based. The method is selected via bit ABT of the Auto-Baud Control register (ABR). When the formula method is used, any baudrate allowed in Equation 17 can be programmed by the UART. This method works well for higher baud rates, but could possibly fail below 28.8 kbps when the remote transmitter's actual baud rate differs by more than one percent of its target.

Equation 17. Baud Rate Formula

$$\text{BaudRate} = \frac{33.334\text{MHz}}{(16\text{XDivisor})}$$

The table method is more immune to such errors since the table rejects uncommon baud rates and rounds to the common ones. The table method allows any baud rate in Equation 17 above 28.8 kbps. Below 28.8 kbps only baud rates of 19200, 14400, 9600, 4800, 1200, and 300 baud can be programmed by UART. Some typical values for Divisor and corresponding baud rates are provided in Table 443. Baud rates above 3600 baud require only Divisor Latch Low Register to be programmed, since Divisor Latch High Register would be 0.

Table 443. Divisor Values for Typical Baud Rates

Baud Rate	Divisor	UART Rate	Error
115.2K	18	115.74K	0.47%
57.6K	36	57.87K	0.47%
38.4K	54	38.58K	0.47%
33.6K	62	33.60K	0.01%
28.8K	72	28.94K	0.47%
19.2K	109	19.29K	0.47%
14.4K	145	14.47K	0.47%
9600	217	9645	0.47%
4800	434	4800	0.01%
3600	579	3600	0.01%
2400	868	2400	0.01%
1200	1736	1200	0.01%
600	3472	600	0.01%
300	6944	300	0.01%

When the baud rate is detected, auto-baud circuitry disarms itself by clearing bit ABE of the Auto-Baud Control register (ABR). To rearm the circuitry, ABE bit must be rewritten.

Note: For the auto-baud-rate detection circuit to work correctly, the first data bit transmitted after the start bit must be a logic '1'. When a logic '0' is transmitted instead, the autobaud circuit counts the zero as part of the start bit, with an incorrect baud rate being programmed into DLL and DLH registers.

13.3.6 Manual Baud Rate Selection

Each UART contains a programmable Baud Rate Generator that is capable of taking the fixed input clock of 33.334 MHz and dividing it by any divisor from 1 to (2^{16}–1). The baud-rate generator output frequency is 16 times the baud rate. Two 8-bit registers store the divisor in a 16-bit binary format. These Divisor Registers must be loaded during initialization to ensure proper operation. When both Divisor Latches are loaded with 0, the 16X output clock is stopped. Access to the Divisor latch can be done with a word write. Equation 17 or Table 443 are used by the programmer to select the Divisor Latch value for the desired baud rate.

13.4 Register Descriptions

There are 15 registers in each UART. The registers are all 32 bit registers, but only lower 8 bits have valid data. The 12 UART registers share eight address locations in the MMR address space. Table 444 shows the registers and their addresses as offsets of a base address. The base address for each UART is 32 bits and is internal bus address offset 2300H for UART 0, and 2340H for UART 1. Note that the state of the Divisor Latch Bit (DLAB), which is the MOST significant bit of the Serial Line Control Register, affects the selection of certain of the UART registers. The DLAB bit must be set high by the system software to access the Baud Rate Generator Divisor Latches.

Table 444. UART Register Addresses as Offsets of a Base

UART Register Addresses	DLAB Bit Value	Name	Register Accessed
Base	0	UxRBR	UART x Receive BUFFER (read only)
Base	0	UxTHR	UART x Transmit BUFFER (write only)
Base + 04H	0	UxIER	UART x Interrupt Enable (R/W)
Base + 08H	X	UxIIR	UART x Interrupt I.D. (read only)
Base + 08H	X	UxFCR	UART x FIFO Control (write only)
Base + 0CH	X	UxLCR	UART x Line Control (R/W)
Base + 10H	X	UxMCR	UART x Modem Control (R/W)
Base + 14H	X	UxLSR	UART x Line Status (Read only)
Base + 18H	X	UxMSR	UART x Modem Status (Read only)
Base + 1CH	X	UxSPR	UART x Scratch Pad (R/W)
Base	1	UxDLL	UART x Divisor Latch (Low Byte, R/W)
Base + 04H	1	UxDLH	UART x Divisor Latch (High Byte, R/W)
Base + 24H	X	UxFOR	UART x FIFO Occupancy Register (R/W)
Base + 28H	X	UxABR	UART x Autobaud Control Register (R/W)
Base + 2CH	X	UxACR	UART x Autobaud Count Register (read only)

Table 445. UART Unit Registers

Section, Register Name, Acronym, page
Section 13.4.1, “UART x Receive Buffer Register” on page 670
Section 13.4.2, “UART x Transmit Holding Register” on page 670
Section 13.4.3, “UART x Interrupt Enable Register” on page 671
Section 13.4.4, “UART x Interrupt Identification Register” on page 672
Section 13.4.5, “UART x FIFO Control Register” on page 674
Section 13.4.6, “UART x Line Control Register” on page 676
Section 13.4.7, “UART x Modem Control Register” on page 678
Section 13.4.8, “UART x Line Status Register” on page 680
Section 13.4.9, “UART x Scratchpad Register” on page 684
Section 13.4.10, “Divisor Latch Registers” on page 685
Section 13.4.11, “UART x FIFO Occupancy Register” on page 686
Section 13.4.12, “UART x Auto-Baud Control Register” on page 687
Section 13.4.13, “UART x Auto-Baud Count Register” on page 688

Table 446. UART Register MMR Addresses

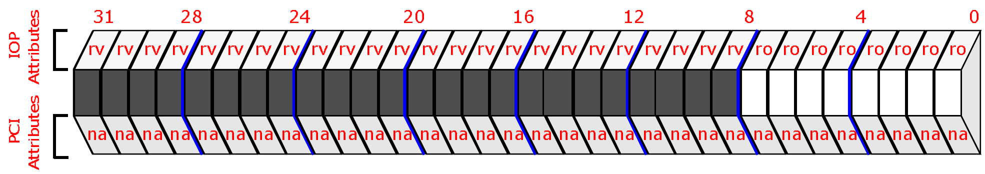
UART Register Addresses	DLAB Bit Value	Name	Register Accessed
+2300H	0	U0RBR	UART 0 Receive BUFFER (read only)
	0	U0THR	UART 0 Transmit BUFFER (write only)
+2304H	0	U0IER	UART 0 Interrupt Enable (R/W)
+2308H	X	U0IIR	UART 0 Interrupt I.D. (read only)
	X	U0FCR	UART 0 FIFO Control (write only)
+230CH	X	U0LCR	UART 0 Line Control (R/W)
+2310H	X	U0MCR	UART 0 Modem Control (R/W)
+2314H	X	U0LSR	UART 0 Line Status (Read only)
+2318H	X	U0MSR	UART 0 Modem Status (Read only)
+231CH	X	U0SPR	UART 0 Scratch Pad (R/W)
+2300H	1	U0DLL	UART 0 Divisor Latch (Low Byte, R/W)
+2304H	1	U0DLH	UART 0 Divisor Latch (High Byte, R/W)
+2324H	X	U0FOR	UART 0 FIFO Occupancy Register (R/W)
+2328H	X	U0ABR	UART 0 Autobaud Control Register (R/W)
+232CH	X	U0ACR	UART 0 Autobaud Count Register (read only)
+2340H	0	U1RBR	UART 1 Receive BUFFER (read only)
	0	U1THR	UART 1 Transmit BUFFER (write only)
+2344H	0	U1IER	UART 1 Interrupt Enable (R/W)
+2348H	X	U1IIR	UART 1 Interrupt I.D. (read only)
	X	U1FCR	UART 1 FIFO Control (write only)
+234CH	X	U1LCR	UART 1 Line Control (R/W)
+2350H	X	U1MCR	UART 1 Modem Control (R/W)
+2354H	X	U1LSR	UART 1 Line Status (Read only)
+2358H	X	U1MSR	UART 1 Modem Status (Read only)
+235CH	X	U1SPR	UART 1 Scratch Pad (R/W)
+2340H	1	U1DLL	UART 1 Divisor Latch (Low Byte, R/W)
+2344H	1	U1DLH	UART 1 Divisor Latch (High Byte, R/W)
+2364H	X	U1FOR	UART 1 FIFO Occupancy Register (R/W)
+2368H	X	U1ABR	UART 1 Autobaud Control Register (R/W)
+236CH	X	U1ACR	UART 1 Autobaud Count Register (read only)

13.4.1 UART x Receive Buffer Register

In non-FIFO mode, this register holds the character(s) received by the UART Receive Shift register. When it receives fewer than eight bits, the bits are right-justified and the leading bits are zeroed. Reading the register empties the register and resets the *data ready (DR)* bit in the Line Status register to 0. Other (error) bits in the Line Status register are not cleared. In FIFO mode, this register latches the value of the data byte(s) at the bottom of the FIFO.

When the UART is in eight-bit Peripheral Bus mode, the 24 most significant bits must be ignored and not used. Reading these bits returns unpredictable results.

Table 447. UART x Receive Buffer Register - (UxRBR)



Unit #	Intel XScale® Core internal bus address
0	+2300H (DLAB=0)
1	+2340H (DLAB=0)

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

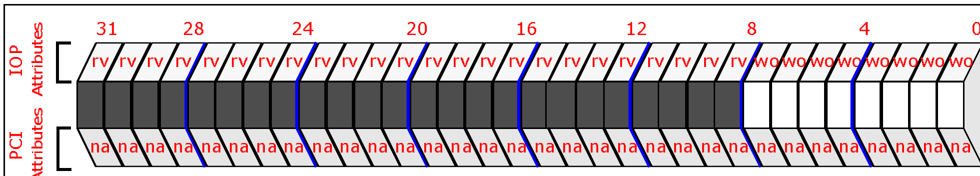
Bit	Default	Description
31:8	00h	Reserved
7:0	00h	Data Byte

13.4.2 UART x Transmit Holding Register

This register holds the next data byte(s) to be transmitted. When the Transmit Shift register becomes empty, the contents of the Transmit Holding register are loaded into the Shift register and the *Transmit Data Request (TDRQ)* bit in the Line Status register is set to one (see Table 451, "Interrupt Identification Register Decode" on page 673).

In FIFO mode, writing to THR puts data to the top of the FIFO. The data at the bottom of the FIFO is loaded to the Shift register when it is empty. In eight-bit Peripheral mode, the 24 most significant bits are ignored and is not transmitted.

Table 448. UART x Transmit Holding Register - (UxTHR)



Unit #	Intel XScale® Core internal bus address
0	+2300H (DLAB=0)
1	+2340H (DLAB=0)

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
WO = Write Only
NA = Not Accessible

Bit	Default	Description
31:8	00h	Reserved
7:0	00h	Data Byte

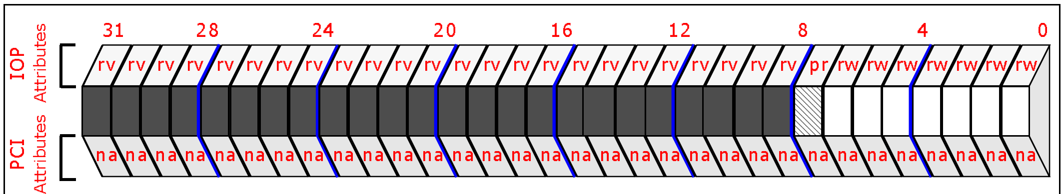
13.4.3 UART x Interrupt Enable Register

This register enables six types of interrupts which set a value in the Interrupt Identification register. Each of the six interrupt types can be disabled by clearing the appropriate bit of the IER register. Similarly, by setting the appropriate bits, selected interrupts can be enabled.

This register also has the control bits of the unit enable and NRZ coding enable. The use of bit 7 to bit 4 is different from the register definition of standard 16550.

Note: A global interrupt enable/disable exists in the Modem Control Register bit 3 (IE). After reset, this bit must be set or no interrupts occurs, regardless of the state of the IER bits. See Section 13.4.7, "UART x Modem Control Register" on page 678.

Table 449. UART x Interrupt Enable Register - (UxIER)



Unit # | Intel XScale® Core internal bus address
0 | +2304H (DLAB=x)
1 | +2344H (DLAB=x)

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:8	00 0000h	Reserved
7	0_2	Preserved
6	0_2	UART Unit Enable (UUE): 0 = the unit is disabled 1 = the unit is enabled
5	0_2	**NRZ coding Enable (NRZE):** 0 = NRZ coding disabled 1 = NRZ coding enabled
4	0_2	Receiver Time Out Interrupt Enable: (RTOIE) 0 = Receiver data Time out Interrupt disabled 1 = Receiver data Time out Interrupt enabled
3	0_2	**Modem Interrupt Enable (MIE):** 0 = Modem Status interrupt disabled 1 = Modem Status interrupt enabled
2	0_2	**Receiver Line Status Interrupt Enable (RLSE):** 0 = Receiver Line Status interrupt disabled 1 = Receiver Line Status interrupt enabled
1	0_2	**Transmit Data request Interrupt Enable (TIE):** 0 = Transmit FIFO Data Request interrupt disabled 1 = Transmit FIFO Data Request interrupt enabled
0	0_2	**Receiver Data Available Interrupt Enable (RAVIE):** 0 = Receiver Data Available (Trigger level reached) interrupt disabled 1 = Receiver Data Available (Trigger level reached) interrupt enabled

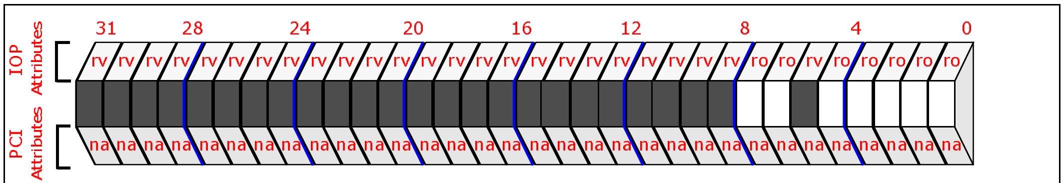
13.4.4 UART x Interrupt Identification Register

The IIR register is read to determine the type and source of UART interrupts. To be 16550 compatible, the lower 4 bits (0-3) of the IIR register are priority encoded as shown in Table 451, "Interrupt Identification Register Decode" on page 673. When two or more interrupts represented by bits (0-3) occur, only the interrupt with the highest priority is displayed. The upper 4 bits, (4-7) are not priority encoded. These bits asserts/deasserts independently of the lower 4 bits.

Bit 0 (nIP) is used to indicate the existence of an interrupt in the priority encoded bits (0-3) of the IIR register. A low signal on this bit indicates an encoded interrupt is pending. When this bit is high, no encoded interrupt is pending, regardless of the state of the other 3 bits. IP# has no effect or association with the upper bits four bits (4-7) which assert/deassert independently of IP#.

In order to minimize software overhead during data character transfers, the UART prioritizes interrupts into four levels (listed in Table 451, "Interrupt Identification Register Decode" on page 673) and records these in the Interrupt Identification register. The Interrupt Identification register (IIR) stores information indicating that a prioritized interrupt is pending and the source of that interrupt.

Table 450. UART x Interrupt Identification Register - (UxIIR)



Unit #	Intel XScale® Core internal bus address
0	+2308H (DLAB=x)
1	+2348H (DLAB=x)

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:8	00 0000h	Reserved
7:6	00_2	**FIFO Mode Enable Status (FIFOES[1:0]):** 00 = Non-FIFO mode is selected 01 = Reserved 10 = Reserved 11 = FIFO mode is selected (TRFIFOE = 1)
5	0_2	Reserved
4	0_2	**Autobaud Lock (ABL)** 0 = Autobaud circuitry has not programmed Divisor Latch registers (DLL/DLH) 1 = Divisor Latch registers (DLL/DLH) programmed by autobaud circuitry
3	0_2	**Time Out Detected (TOD):** 0 = No time out interrupt is pending 1 = Time out interrupt is pending. (FIFO mode only)
2:1	0_2	**Interrupt Source Encoded (IID[1:0]):** indicates a Modem Status Interrupt when the IP# bit is low. When IP# bit is high, there is no Interrupt. 00 = Modem Status (CTS, DSR, RI, DCD modem signals changed state) 01 = Transmit FIFO requests data 10 = Received Data Available 11 = Receive error (Overrun, parity, framing, break, FIFO error)
0	1_2	**Interrupt Pending (IP#):** 0 = Interrupt is pending. (Active low) 1 = No interrupt is pending

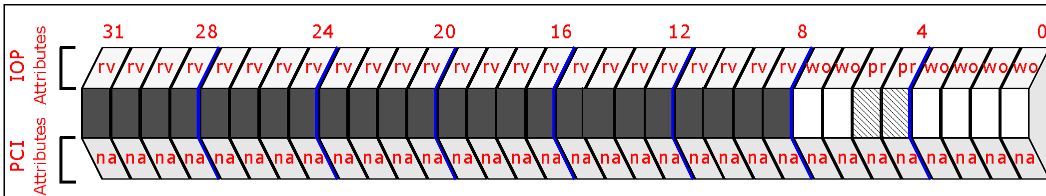
Table 451. Interrupt Identification Register Decode

	Interrupt ID bits				Interrupt SET/RESET Function			
	3	2	1	0	Priority	Type	Source	RESET Control
IP#	0	0	0	1	-	None	No Interrupt is pending.	-
IID[11]	0	1	1	0	Highest	Receiver Line Status	Overrun Error, Parity Error, Framing Error, Break Interrupt.	Reading the Line Status Register.
IID[10]	0	1	0	0	Second Highest	Received Data Available.	Non-FIFO mode: Receive Buffer is full. FIFO mode: Trigger level was reached.	Non-FIFO mode: Reading the Receiver Buffer Register. FIFO mode: Reading bytes until Receiver FIFO drops below trigger level or setting RESETRF bit in FCR register.
TOD	1	1	0	0	Second Highest	Character Timeout indication.	FIFO Mode only: At least 1 character is in receiver FIFO and there was no activity for a time period.	Reading the Receiver FIFO or setting RESETRF bit in FCR register.
IID[01]	0	0	1	0	Third Highest	Transmit FIFO Data Request	Non-FIFO mode: Transmit Holding Register Empty FIFO mode: Transmit FIFO has half or less than half data.	Reading the IIR Register (when the source of the interrupt) or writing into the Transmit Holding Register. Reading the IIR Register (when the source of the interrupt) or writing to the Transmitter FIFO.
IID[00]	0	0	0	0	Fourth Highest	Modem Status	Clear to Send, Data Set Ready, Ring Indicator, Received Line Signal Detect	Reading the Modem Status Register.
Non Prioritized Interrupts								
ABL	4				None	Autobaud Lock Indication	Autobaud circuitry has locked onto the baud rate.	Reading the IIR Register

13.4.5 UART x FIFO Control Register

FCR is a write only register that is located at the same address as the IIR (IIR is a read only register). FCR enables/disables the transmitter/receiver FIFOs, clears the transmitter/receiver FIFOs, and sets the receiver FIFO trigger level.

Table 452. UART x FIFO Control Register - (UxFCR) (Sheet 1 of 2)

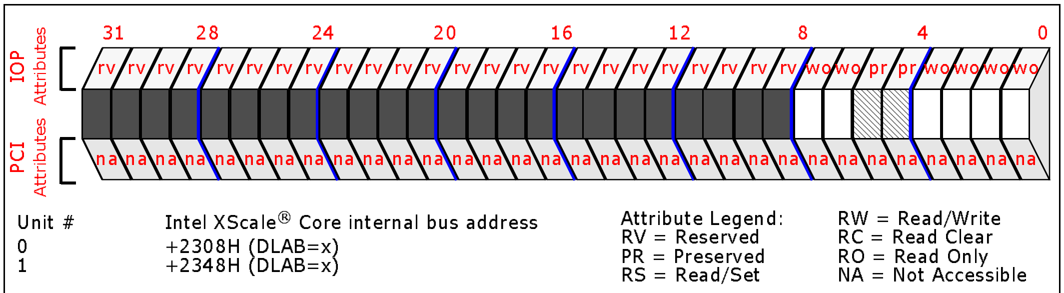


Unit # | Intel XScale® Core internal bus address
0 | +2308H (DLAB=x)
1 | +2348H (DLAB=x)

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:8	00 0000h	Reserved
7:6	00_2	**Interrupt Trigger Level (ITL):** When the number of bytes in the receiver FIFO equals the interrupt trigger level programmed into this field and the Received Data Available Interrupt is enabled (via IER), an interrupt is generated and appropriate bits are set in the IIR. 00 = 1 byte or more in FIFO causes interrupt 01 = 8 bytes or more in FIFO causes interrupt 10 = 16 bytes or more in FIFO causes interrupt 11 = 32 bytes or more in FIFO causes interrupt
5:4	0_2	Preserved
3	0_2	**Transmitter Interrupt Level (TIL):** Setting TIL causes Transmitter Interrupts to occur when the Transmit FIFO is empty. Clearing TIL causes Transmitter Interrupts to occur when the Transmit FIFO is half empty. 0 = Interrupt when FIFO is half empty 1 = Interrupt when FIFO is empty

Table 452. UART x FIFO Control Register - (UxFCR) (Sheet 2 of 2)

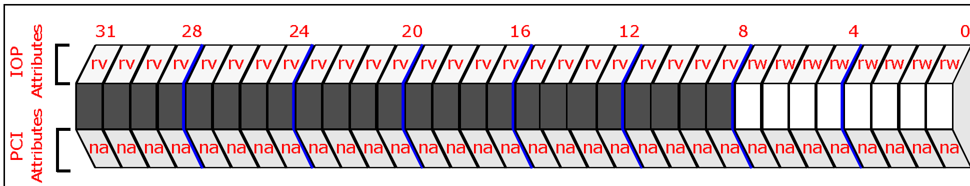


Bit	Default	Description
2	0_2	**Reset Transmitter FIFO (RESETTF):** When set, the Transmitter FIFO counter is reset to clear all the bytes in the FIFO. The *TDRQ* bit of LSR is set generating a Transmitter Requests Data Interrupt IID field of IIR when the *TIE* bit in the IER register is set. The Transmitter Shift register is not reset; it completes the current transmission. Any transmit FIFO Service-Request Interrupts are cleared. Note: After the FIFO is cleared, RESETTF is automatically reset to 0. 0 = no effect 1 = The transmitter FIFO is cleared (FIFO counter set to 0). After clearing, bit is automatically reset to 0
1	0_2	**Reset Receiver FIFO (RESETRF):** When set, the receiver FIFO counter is reset to clear all the bytes in the FIFO. The DR bit in LSR is reset to 0. All the error bits in the FIFO and the FIFOE bit in LSR are cleared. Any error bits (OE, PE, FE or BI), that had been set in LSR are still set. The receiver shift register is not cleared. Any Receive FIFO Service Request Interrupts are cleared. Note: After the FIFO is cleared, RESETRF is automatically reset to 0. 0 = no effect 1 = The receiver FIFO is cleared (FIFO counter set to 0). After clearing, bit is automatically reset to 0
0	0_2	**Transmit and Receive FIFO Enable (TRFIFOE):** Enables/disables the transmitter and receiver FIFOs. When TRFIFOE = 1, both FIFOs are enabled (FIFO Mode). When TRFIFOE = 0, the FIFOs are both disabled (non-FIFO Mode). Writing a 0 to this bit clears all bytes in both FIFOs. When changing from FIFO mode to non-FIFO mode and vice versa, data is automatically cleared from the FIFOs. Any FIFO Service Request Interrupts are cleared when TRFIFOE is cleared. Note: This bit must be 1 when other bits in this register are written, or the other bits are not programmed. 0 = FIFOs are disabled 1 = FIFOs are enabled

13.4.6 UART x Line Control Register

In the Line Control Register, the system programmer specifies the format of the asynchronous data communications exchange. The serial data format consists of a start bit (logic 0), five to eight data bits, an optional parity bit, and one or two stop bits (logic 1). The LCR has bits for accessing the Divisor Latch registers and causing a Break condition. The programmer can also read the contents of the Line Control Register. The read capability simplifies system programming and eliminates the need for separate storage in system memory.

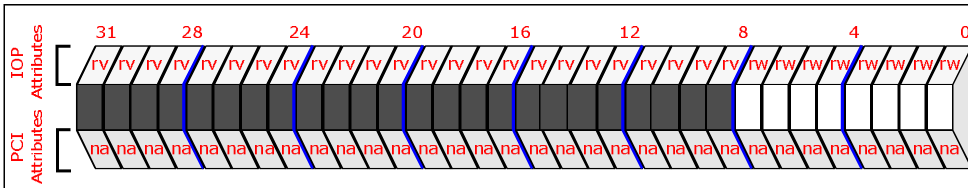
Table 453. UART x Line Control Register - (UxLCR) (Sheet 1 of 2)



Unit #	Intel XScale® Core internal bus address	Attribute Legend:	
0	+230CH (DLAB=x)	RV = Reserved	RW = Read/Write
1	+234CH (DLAB=x)	PR = Preserved	RC = Read Clear
		RS = Read/Set	RO = Read Only
			NA = Not Accessible

Bit	Default	Description
31:8	00 0000h	Reserved
7	0_2	**Divisor Latch register Access Bit (DLAB):** This bit must be set (1) to access the Divisor Latches of the Baud Rate Generator during a READ or WRITE operation. It must be clear (0) to access the Receiver Buffer, the Transmit-Holding Register, or the Interrupt-Enable Register. This bit does not have to be set when using autobaud. 0 = access Transmit Holding register (THR), Receive Buffer Register (RBR) and Interrupt Enable Register. 1 = access Divisor Latch Registers (DLL and DLH).
6	0_2	**Set break (SB):** Causes a Break condition to the receiving UART. When SB is set (1), the serial output (TXD) is forced to the spacing (logic 0) state and remains there until SB is clear (0). This bit acts only on the TXD pin and has no effect on the transmitter logic. In FIFO mode, wait for the transmitter to be idle (TEMT=1) to set and clear the break bit. 0 = no effect on TXD output. 1 = forces TXD output to 0 (space).
5	0_2	**Sticky Parity (STKYP):** Can be used in multiprocessor communications. When PEN and STKYP are set (1), the bit that is transmitted in the parity bit location (the bit just before the stop bit) is the complement of the EPS bit. When EPS is 0, then the bit at the parity bit location is transmitted as a 1. In the receiver, when STKYP and PEN are 1, then the receiver compares the bit that is received in the parity bit location with the complement of the EPS bit. When the values being compared are not equal, the receiver sets the Parity Error bit in LSR and causes an error interrupt when line status interrupts were enabled. For example, when EPS is 0, the receiver expects the bit received at the parity bit location to be 1. When it is not, then the parity error bit is set. By forcing the bit value at the parity bit location, rather than calculating a parity value, a system with a master transmitter and multiple receivers can identify some transmitted characters as receiver addresses and the rest of the characters as data. When PEN = 0, STKYP is ignored. 0 = no effect on parity bit. 1 = Forces parity bit to be opposite of EPS bit value.

Table 453. UART x Line Control Register - (UxLCR) (Sheet 2 of 2)



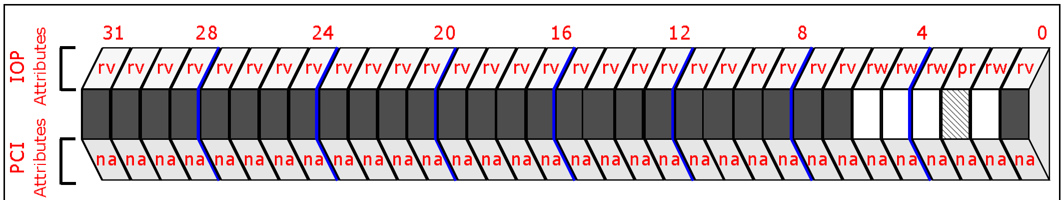
Unit #	Intel XScale® Core internal bus address	Attribute Legend:	
0	+230CH (DLAB=x)	RV = Reserved	RW = Read/Write
1	+234CH (DLAB=x)	PR = Preserved	RC = Read Clear
		RS = Read/Set	RO = Read Only
			NA = Not Accessible

Bit	Default	Description
4	0_2	**Even Parity Select (EPS):** When PEN is set (1) and EPS is clear (0), an odd number of logic ones is transmitted or checked in the data word bits and the parity bit. When PEN is set (1) and EPS is also set (1), an even number of logic ones is transmitted or checked in the data word bits and parity bit. When PEN = 0, EPS is ignored. 0 = sends or checks for odd parity. 1 = sends or checks for even parity.
3	0_2	**Parity Enable (PEN):** When set (1), a parity bit is generated (transmit data) or checked (receive data) between the last data word bit and Stop bit of the serial data. (The parity bit is used to produce an even or odd number of ones when the data word bits and the parity bit are summed.) 0 = no parity function. 1 = allows parity generation and checking.
2	0_2	**Stop bits (STB):** This bit specifies the number of stop bits transmitted and received in each serial character. When STB is clear (0), one stop bit is generated in the transmitted data. When STB is set (1) when a 5-bit word length is selected via WLS[1:0], then 1 and one half stop bits are generated. When STB is set (1) when either a 6, 7, or 8-bit word is selected, then two stop bits are generated. The receiver checks the first stop bit only, regardless of the number of stop bits selected. 0 = 1 stop bit 1 = 2 stop bits, except for 5-bit character then 1-1/2 bits
1:0	00_2	**Word Length Select (WLS[1:0]):** Specifies the number of data bits in each transmitted or received serial character. 00 = 5-bit character (default) 01 = 6-bit character 10 = 7-bit character 11 = 8-bit character

13.4.7 UART x Modem Control Register

This register controls the interface with the modem or data set (or a peripheral device emulating a modem). The contents of the Modem Control register are described below:

Table 454. UART x Modem Control Register - (UxMCR) (Sheet 1 of 2)

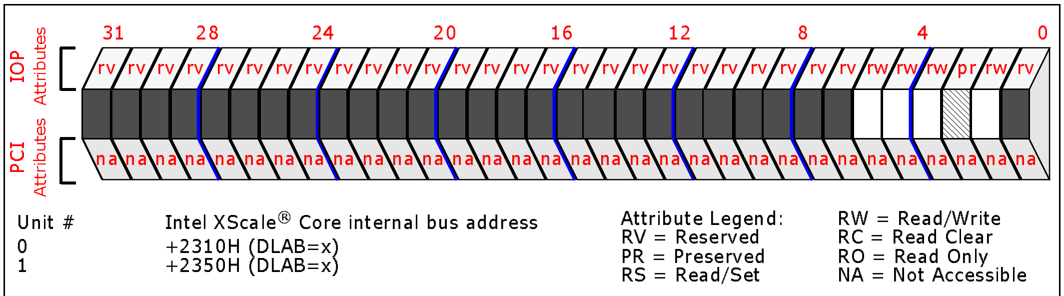


Unit # | Intel XScale® Core internal bus address
0 | +2310H (DLAB=x)
1 | +2350H (DLAB=x)

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:6	000 0000h	Reserved
5	0_2	**Autoflow Control Enable (AFE):** When set, autoflow control is enabled. Only auto-CTS is enabled when *RTS* is cleared in MCR while AFE is set. Both auto-CTS and auto-RTS are enabled when *AFE* and *RTS in MCR* are set. Auto-RTS is not enabled when *AFE* is not set regardless of the state of *RTS*. Autoflow automates the flow of data between the UART and the remote device. See Section 13.3.4, “Autoflow Control” on page 665 for more details. 0 = Auto-RTS and Auto-CTS are disabled 1 = Auto-CTS is enabled. IF RTS in MCR is also set, both auto-CTS and auto-RTS is enabled
4	0_2	**Loop back test mode (LOOP):** This bit provides a local Loopback feature for diagnostic testing of the UART. In the Diagnostic mode, data that is transmitted is immediately received. This feature allows the processor to verify the UART Transmit and Receive data paths. The Transmit, Receive and Modem Control Interrupts are operational. The modem-control input **CTS#** is activated by MCR bit 1 instead of the modem-control input. A Break signal can also be transferred from the transmitter section to the receiver section in Loop-Back mode. When LOOP is set (1), the following occurs: • The TXD (transmitter output) pin is set to a logic-1 state. • The RXD (receiver input) pin is disconnected. • The output of the Transmitter Shift register is “looped back” into the Receiver-Shift register input. • The modem-control input **CTS#** is disconnected from the pins and the modem-control output pin **RTS#** is forced to the inactive state. The RTS bit of the Modem Control register is connected to bits of the Modem Status register bits. Flow control can be tested; when autoflow is enabled the RTS bit of the Modem Control Register has no effect on the **CTS#** input as **RTS#** is asserted by the autoflow logic.: • RTS = 1 forces CTS to a 1 ***Note:*** Note: Coming out of the Loop-Back Test mode may result in unpredictable activation of the delta bit (bit 0) in the Modem Status register (MSR). It is recommended that MSR is read once to clear the delta bits in the MSR. 0 = Normal UART operation 1 = Test mode UART operation

Table 454. UART x Modem Control Register - (UxMCR) (Sheet 2 of 2)



Bit	Default	Description
3	0_2	**Interrupt Enable (IE):** Global control all UART interrupts. 0 = interrupts disabled. 1 = interrupts enabled. **NOTE:** This bit is not valid when in Loopback mode.
2	0_2	Preserved.
1	0_2	**Request to Send (RTS):** **Non-Autoflow mode:** When not in Autoflow mode (AFE bit of MCR is clear), this bit controls the Request-to-Send (**RTS#**) output pin. 0 = RTS# pin is 1 1 = RTS# pin is 0 **Autoflow mode:** When in Autoflow mode (AFE bit of MCR is set), auto-RTS is enabled. **RTS#** behaves as follows: • Auto-RTS disabled. Autoflow works only with auto-CTS. • Auto-RTS enabled. Autoflow works with both auto-CTS and auto-RTS.
0	0_2	Reserved

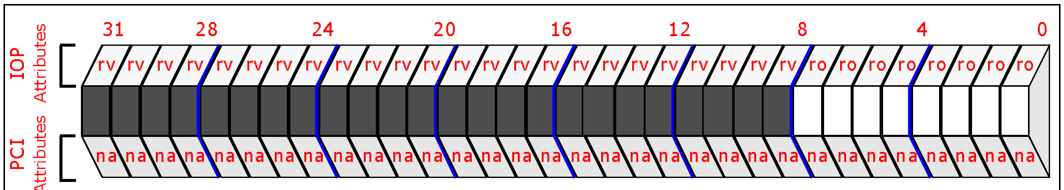
13.4.8 UART x Line Status Register

This register provides status information to the processor concerning the data transfers. Bits 5 and 6 show information about the transmitter section. The remainder of the bits contain information about the receiver.

In non-FIFO mode, three of the LSR register bits, parity error, framing error, and break interrupt, show the error status of the character that has just been received. In FIFO mode, these three status bits are stored with each received character in the FIFO. LSR shows the status bits of the character at the bottom of the FIFO. When the character at the bottom of the FIFO has errors, the LSR error bits are set and are not cleared until software reads LSR, even when the character in the FIFO is read and a new character is now at the bottom of the FIFO.

Bits 1 through 4 are the error conditions that produce a Receiver Line Status Interrupt when any of the corresponding conditions are detected and the interrupt is enabled. These bits are not cleared by reading the erroneous byte from the FIFO or receive buffer. They are cleared only by reading LSR. In FIFO mode, the Line Status Interrupt occurs only when the erroneous byte is at the bottom of the FIFO. When the erroneous byte being received is not at the bottom of the FIFO, an interrupt is generated only after the previous bytes are read and the erroneous byte is moved to the bottom of the FIFO.

Table 455. UART x Line Status Register - (UxLSR) (Sheet 1 of 3)

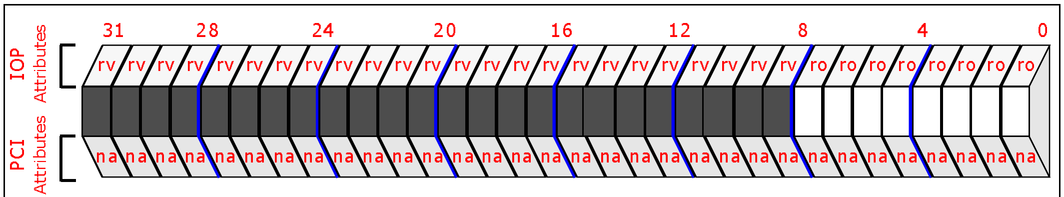


Unit # — Intel XScale® Core internal bus address
0 — +2314H (DLAB=x)
1 — +2354H (DLAB=x)

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:8	00 0000h	Reserved
7	0_2	**FIFO Error Status (FIFOE):** In non-FIFO mode, this bit is clear (0). In FIFO Mode, this bit is set (1) when there is at least one parity error, framing error, or break indication for any of the characters in the FIFO. A processor read to the Line Status register does not reset this bit. FIFOE is reset when all error bytes have been read from the FIFO. FIFOE being set to 1 does NOT generate an interrupt. 0 = No errors exit in the receive FIFO 1 = At least one character in receiver FIFO has errors
6	1_2	**Transmitter Empty (TEMT):** Set (1) when the Transmit Holding register and the Transmitter Shift register are both empty. It is reset to zero (0) when either the Transmit Holding register or the Transmitter Shift register contains a data character. In FIFO mode, TEMT is set to 1 when the transmitter FIFO and the Transmit Shift register are both empty. 0 = There is data in the Transmit Shift register, the Transmit Holding register, or the FIFO 1 = All the data in the transmitter has been shifted out

Table 455. UART x Line Status Register - (UxLSR) (Sheet 2 of 3)

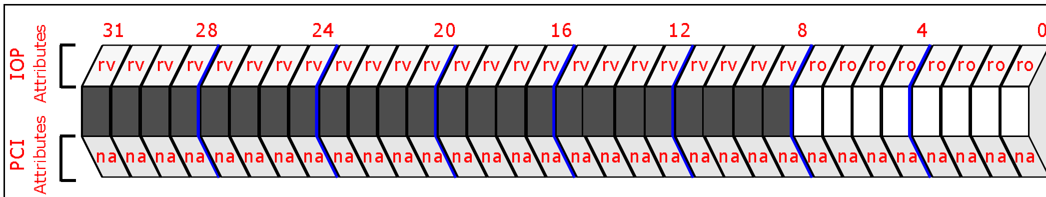


Unit #	Intel XScale® Core internal bus address
0	+2314H (DLAB=x)
1	+2354H (DLAB=x)

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
5	1_2	**Transmit Data Request (TDRQ):** Indicates that the UART is ready to accept data for transmission. The assertion of this bit causes the UART to issue an interrupt when the Transmit Data Request Interrupt Enable is set. In non-FIFO mode, the TDRQ bit is set (1) when a character is transferred from the Transmit-Holding register. The bit is cleared (0) concurrently with the loading of the Transmit Holding register by the processor. In FIFO mode, TDRQ is set (1) when the FIFO is less than half full. It is cleared when the FIFO is more than half full. When more than 64 characters are loaded into the FIFO, the excess characters are lost. 0 = The UART is NOT ready to receive data for transmission. 1 = The UART is ready to receive data for transmission.
4	0_2	**Break Indicator (BI):** Set (1) when the received data input is held in the Spacing (logic 0) state for longer than a full character transmission time (that is, the total time of *start* bit + *data* bits + *parity* bit + *stop* bits). The Break Indicator is reset (cleared to 0) when the processor reads the Line-Status register. In FIFO mode, only one Break character (equal to 0x00), is loaded into the FIFO regardless of the length of the Break condition. *BI* shows the Break condition for the character at the bottom of the FIFO, not the most recent character received. 0 = No break signal has been received. 1 = Break signal has been received.
3	0_2	**Framing Error (FE):** Indicates that the received character did not have a valid stop bit. FE is set (1) when the bit following the last data bit or parity bit is detected as a logic 0 bit (spacing level). When the Line Control register had been set for two stop bits, the receiver does not check for a valid second stop bit. The FE indicator is reset when the processor reads the Line Status Register. The UART resynchronizes after a framing error by assuming that the framing error was due to the next start bit. Therefore it samples this start bit twice and then takes in the data. In FIFO mode, FE shows a framing error for the character at the bottom of the FIFO, not for the most recently received character. 0 = No Framing error. 1 = Invalid stop bit has been detected.

Table 455. UART x Line Status Register - (UxLSR) (Sheet 3 of 3)



Unit #	Intel XScale® Core internal bus address	Attribute Legend:	
0	+2314H (DLAB=x)	RV = Reserved	RW = Read/Write
1	+2354H (DLAB=x)	PR = Preserved	RC = Read Clear
		RS = Read/Set	RO = Read Only
			NA = Not Accessible

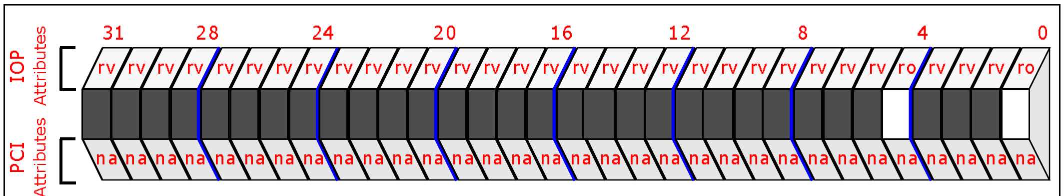
Bit	Default	Description
2	0_2	**Parity Error (PE):** Indicates that the received data character does not have the correct even or odd parity, as selected by the even parity select bit. The PE is set (1) upon detection of a parity error and is cleared (0) when the processor reads the Line Status register. In FIFO mode, PE shows a parity error for the character at the bottom of the FIFO, not the most recently received character. 0 = No Parity error. 1 = Parity error has occurred.
1	0_2	**Overflow Error (OE):** In non-FIFO mode, OE indicates that data in the Receiver Buffer register was not read before the next character was received, the new character is lost. In FIFO mode, OE indicates that all 64 bytes of the FIFO are full and the most recently received byte has been discarded. The OE indicator is set (1) upon detection of an overflow condition and reset when the processor reads the Line Status register. 0 = No overflow error. Data has not been lost. 1 = Overflow error. Receive data has been lost.
0	0_2	**Data Ready (DR):** Set to a logic 1 when a complete incoming character has been received and transferred into the receiver buffer register or the FIFO. In non-FIFO mode, DR is reset to 0 when the receive buffer is read. In FIFO mode, DR is reset to a logic 0 when the FIFO is empty (last character has been read from RBR) or the RESETRF bit is set in FCR. 0 = No data has been received. 1 = Data is available in RBR or the FIFO.

Table 456. UART x Modem Status Register

This register provides the current state of the control lines from the modem or data set (or a peripheral device emulating a modem). In addition to this current state information, the Modem Status register also provides change information. The change bit is set to a logic 1 when the control input from the Modem changes state. The change bit is reset to a logic 0 when the processor reads the Modem Status register.

Note: When the change bit (bit 0) is set to logic 1, a Modem Status interrupt is generated when bit 3 of the Interrupt Enable Register is set.

Table 457. UART x Modem Status Register - (UxMSR)



Unit #	Intel XScale® Core internal bus address
0	+2318H (DLAB=x)
1	+2358H (DLAB=x)

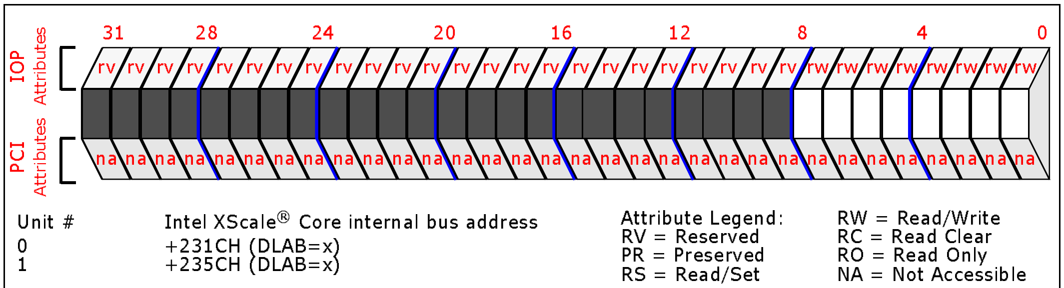
Attribute Legend: RV = Reserved, PR = Preserved, RS = Read/Set, RW = Read/Write, RC = Read Clear, RO = Read Only, NA = Not Accessible

Bit	Default	Description
31:5	000 0000h	Reserved
4	0_2	**Clear to Send (CTS):** This bit is the complement of the Clear to Send (**CTS#**) input. This bit is equivalent to bit RTS of the Modem Control register when LOOP in the MCR is set to 1. 0 = **CTS#** pin is 1 1 = **CTS#** pin is 0
3:1	000_2	Reserved
0	0_2	**Delta Clear To Send (DCTS):** 0 = No change in **CTS#** pin since last read of MSR 1 = **CTS#** pin has changed state

13.4.9 UART x Scratchpad Register

This read/write register has no effect on the UART. It is intended as a scratchpad register for use by programmers.

Table 458. UART x Scratchpad Register - (UxSCR)



Bit	Default	Description
31:8	00 0000h	Reserved
7:0	00h	No effect on UART functionality

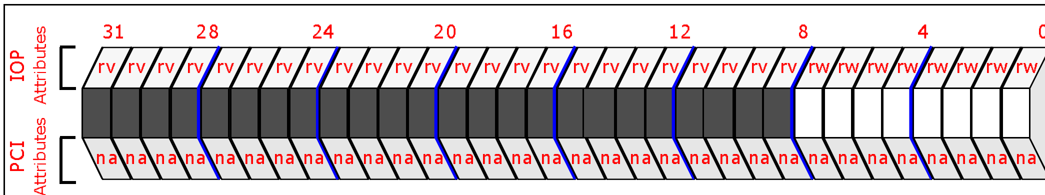
13.4.10 Divisor Latch Registers

The description of use for the Divisor Latch Registers are provided in Section 13.3.5, Auto-Baud-Rate Detection and Section 13.3.6, Manual Baud Rate Selection. Refer to those sections for details on how to program these registers.

Bit DLAB in the LCR register must be set high before the Divisor Latch registers can be accessed.

A Divisor value of 0 in the Divisor Latch Register is not allowed. A value of 0 has the affect of disabling the UART. The reset value of the divisor is 02.

Table 459. UART x Divisor Latch Low Register - (UxDLL)

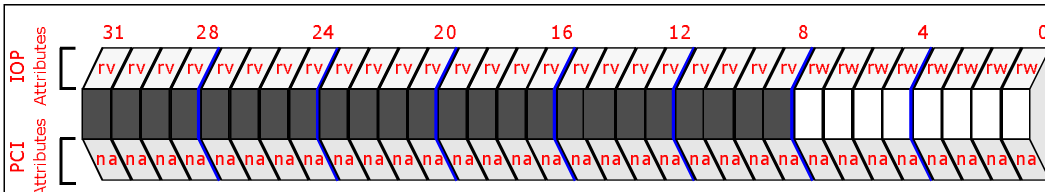


Unit # | Intel XScale® Core internal bus address
0 | +2300H (DLAB=1)
1 | +2340H (DLAB=1)

Attribute Legend: RV = Reserved, PR = Preserved, RS = Read/Set, RW = Read/Write, RC = Read Clear, RO = Read Only, NA = Not Accessible

Bit	Default	Description
31:8	00 0000h	Reserved
7:0	02h	Low byte compare value to generate baud rate

Table 460. UART x Divisor Latch High Register - (UxDLH)



Unit # | Intel XScale® Core internal bus address
0 | +2304H (DLAB=1)
1 | +2344H (DLAB=1)

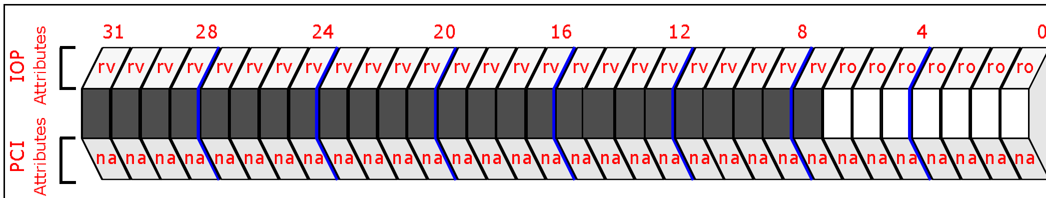
Attribute Legend: RV = Reserved, PR = Preserved, RS = Read/Set, RW = Read/Write, RC = Read Clear, RO = Read Only, NA = Not Accessible

Bit	Default	Description
31:8	00 0000h	Reserved
7:0	00h	High byte compare value to generate baud rate

13.4.11 UART x FIFO Occupancy Register

This register shows the number of bytes currently remaining the Receive FIFO. It can be used by the processor to determine the number of trailing bytes to remove from the receive FIFO when the Character Time-out Interrupt is detected. Refer to Section 13.3.2, "Removing Trailing Bytes In Interrupt Mode" on page 664. The FOR register is incremented once for each byte of data written to the Receive FIFO and decremented once for each byte read.

Table 461. UART x FIFO Occupancy Register - (UxFOR)



Unit # | Intel XScale® Core internal bus address
0 | +2324H (DLAB=x)
1 | +2364H (DLAB=x)

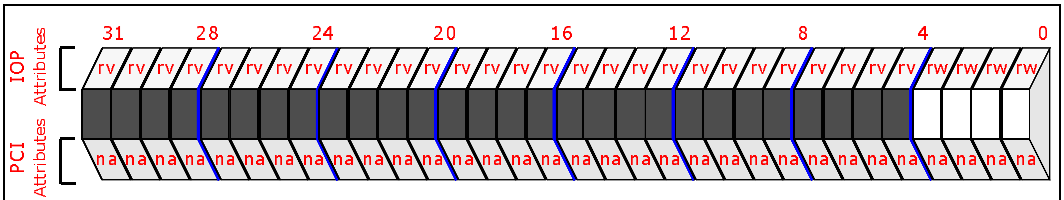
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:7	000 0000h	Reserved
6:0	000 0000_2	FOR[6:0]: Number of bytes (0-63) remaining in the Receiver FIFO

13.4.12 UART x Auto-Baud Control Register

This read/write register has no effect on the UART. It is intended as a scratchpad register for use by programmers.

Table 462. UART x Auto-Baud Control Register - (UxABR)



Unit # | Intel XScale® Core internal bus address
0 | +2328H (DLAB=x)
1 | +2368H (DLAB=x)

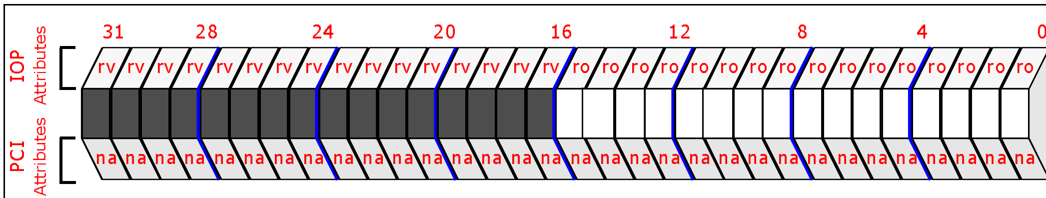
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:4	000 0000h	Reserved
3	0_2	**Auto-Baud Table (ABT):** Directs the auto-baud circuitry within the UART to use a table when selecting the final baud rate programmed into the Divisor Latch registers and written into the Auto-Baud Count register (ACR). When a table is not used, any value allowed in Equation 17 on page 666 can be selected by the UART. See Section 13.3.5, “Auto-Baud-Rate Detection” on page 666 for more information on auto-baud. 0 = Formula used to calculate baud rates allowing all possible baud rates to be chosen by UART Equation 17 on page 666. 1 = Table used to calculate baud rates which limits UART to choosing common baud rates.
2	0_2	**Auto-Baud UART Program (ABUP):** Allows the UART to automatically program the Divisor Latch registers (DLL and DLH) when the auto-baud circuit has detected the baud rate, regardless of the state of the *divisor-latch access* bit (DLAB). For this to occur, the *auto-baud enable (ABE)* bit must be set. Clearing this bit allows the processor to read the Auto-Baud Count register (ACR) and determine the baud rate using its own algorithm rather than using the UARTs. 0 = Software programs Divisor Latch Registers. 1 = UART Programs Divisor Latch Registers
1	0_2	**Auto-Baud Lock Interrupt Enable (ABLIE):** Enables the ABL Interrupt which occurs when the auto-baud circuit has detected the baud rate and written the value into the Auto-Baud Count register (ACR). The Divisor Latch registers can then be programmed by the processor or auto-baud circuitry as dictated by the state of the ABPP. 0 = Autobaud Lock Interrupt Disabled 1 = Autobaud Lock Interrupt Enabled
0	0_2	**Auto-Baud Enable (ABE):** Enables the auto-baud circuitry within the UART. The circuitry counts the number of clocks in the *start* bit and writes this count into the Autobaud Count register (ACR). It then interrupts the processor when the *auto-baud lock interrupt-enable (ABLIE)* bit is set. It also automatically programs the Divisor Latch registers (DLL and DLH) when the *auto-baud UART program (ABUP)* bit is set. 0 = Autobaud Disabled 1 = Autobaud Enabled

13.4.13 UART x Auto-Baud Count Register

The Auto-Baud Count register stores the number of 33.334 MHZ clock cycles within a *start* bit pulse. This value is then used by the processor or auto-baud circuitry within the UART to calculate the baud rate. When Auto-Baud mode and Auto-Baud Interrupts are enabled, the UART interrupt the processor with the Auto-Baud Lock Interrupt after it has written the count value into the ACR. The value is written regardless of the state of the *auto-baud UART program* bit.

Table 463. UART x Auto-Baud Count Register - (UxACR)



Unit # Intel XScale® Core internal bus address
0 +232CH (DLAB=x)
1 +236CH (DLAB=x)

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:16	0000h	Reserved
15:0	0000h	ACR[15:0]: Number of 33.334 MHz clock cycles within a start bit pulse.

14.0 I^2C Bus Interface Units

Note: I2C0 is owned by the Ttransport Core. See the System/Software Architecture Specification for details on how to change this.

This chapter describes the three I^2C (Inter-Integrated Circuit) bus interface units, including the operation modes and setup. Throughout this manual, these peripherals are referred to as the I^2C units.

14.1 Overview

The three I^2C Bus Interface Units allows the Intel® 413808 and 413812 I/O Controllers in TPER Mode (4138xx) to serve as a master and slave device residing on the I^2C bus. The I^2C bus is a serial bus developed by Philips* Corporation consisting of a two-pin interface. **SDA** is the data pin for input and output functions and **SCL** is the clock pin for reference and control of the I^2C bus.

The I^2C bus allows the 4138xx to interface to other I^2C peripherals and microcontrollers for system management functions. The serial bus requires a minimum of hardware for an economical system to relay status and reliability information on the 4138xx subsystem to an external device.

The I^2C Bus Interface Unit is a peripheral device that resides on a 4138xx internal bus. Data is transmitted to and received from the I^2C bus via a buffered interface. Control and status information is relayed through a set of memory-mapped registers. Refer to the I^2C Bus Specification for complete details on I^2C bus operation.

14.2 Theory of Operation

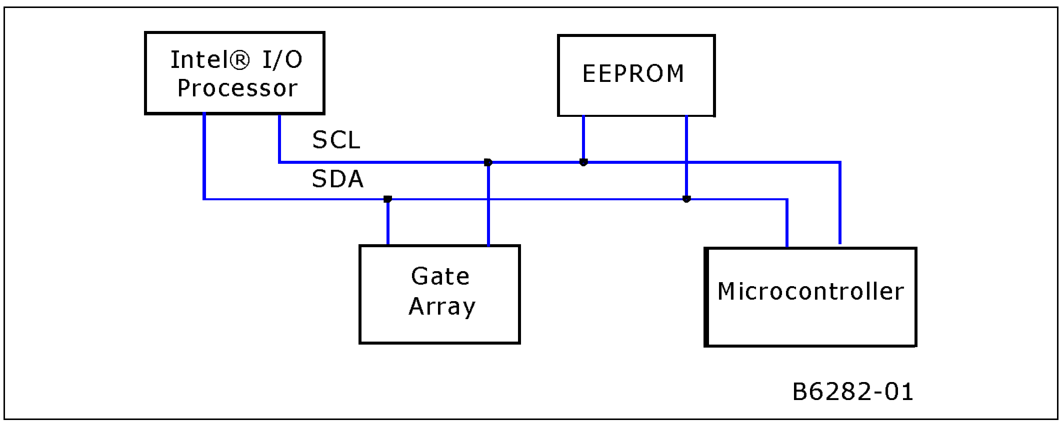
The I²C bus defines a serial protocol for passing information between agents on the I²C bus, using only a two pin interface. The interface consists of a Serial Data/Address (**SDA**) line and a Serial Clock Line (**SCL**). Each device on the I²C bus is recognized by a unique 7-bit address and can operate as a transmitter or as a receiver. In addition to transmitter and receiver, the I²C bus uses the concept of master and slave. Table 464 lists the I²C device types.

Table 464. I²C Bus Definitions

I²C Device	Definition
Transmitter	Sends data to the I²C bus.
Receiver	Receives data from the I²C bus.
Master	Initiates a transfer, generates the clock signal, and terminates the transactions.
Slave	The device addressed by a master.
Multi-master	More than one master can attempt to control the bus at the same time without corrupting the message.
Arbitration	Procedure to ensure that, when more than one master simultaneously tries to control the bus, only one is allowed. This procedure ensures that messages are not corrupted.

As an example of I²C bus operation, consider the case of the 4138xx acting as a master on the bus (Figure 91). The 4138xx, as a master, addresses an EEPROM as a slave to receive data. The 4138xx is a master-transmitter and the EEPROM is a slave-receiver. When the 4138xx reads data, the 4138xx is a master-receiver and the EEPROM is a slave-transmitter. In both cases, the master generates the clock, initiates the transaction and terminates it.

Figure 91. I²C Bus Configuration Example



The I^2C bus allows for a multi-master system, which means more than one device can initiate data transfers at the same time. To support this feature, the I^2C bus arbitration relies on the wired-AND connection of all I^2C interfaces to the I^2C bus. Two masters can drive the bus simultaneously provided they are driving identical data. The first master to drive **SDA** high while another master drives **SDA** low loses the arbitration. The **SCL** line consists of a synchronized combination of clocks generated by the masters using the wired-AND connection to the **SCL** line.

The I^2C bus serial operation uses an open-drain wired-AND bus structure, which allows multiple devices to drive the bus lines and to communicate status about events such as arbitration, wait states, error conditions and so on. For example, when a master drives the clock (**SCL**) line during a data transfer, it transfers a bit on every instance that the clock is high. When the slave is unable to accept or drive data at the rate that the master is requesting, the slave can hold the clock line low between the high states to insert a wait interval. The master's clock can only be altered by a slow slave peripheral keeping the clock line low or by another master during arbitration.

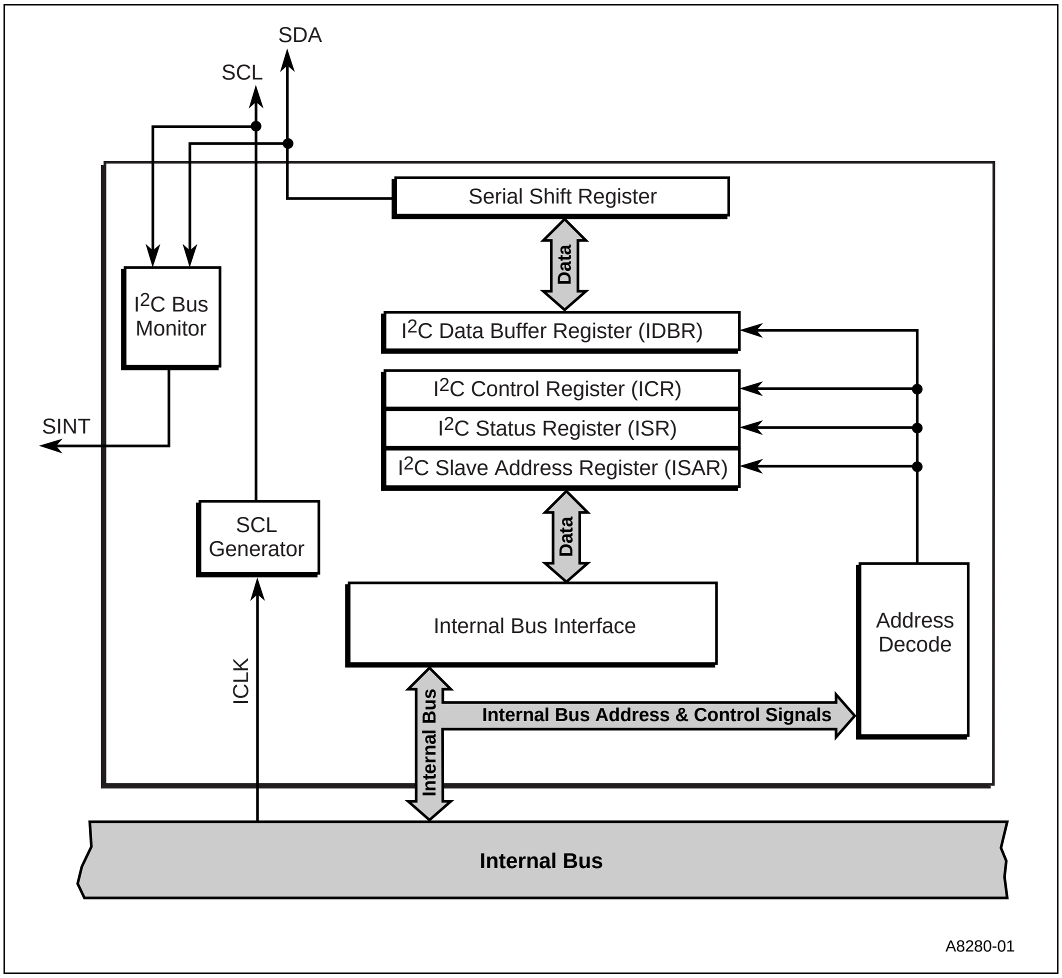
I^2C transactions are either initiated by the 4138xx as a master or are received by the processor as a slave. Both conditions may result in the processor doing reads, writes, or both to the I^2C bus.

14.2.1 Operational Blocks

The I^{2}C Bus Interface Unit is a slave peripheral device that is connected to the internal bus. The 4138xx interrupt mechanism can be used for notifying the 4138xx that there is activity on the I^{2}C bus. Polling can be also be used instead of interrupts, although it would be very cumbersome. Figure 92 shows a block diagram of the I^{2}C Bus Interface Unit and its interface to the internal bus.

The I^{2}C Bus Interface Unit consists of the two wire interface to the I^{2}C bus, an 8-bit buffer for passing data to and from the 4138xx, a set of control and status registers, and a shift register for parallel/serial conversions.

Figure 92. I^{2}C Bus Interface Unit Block Diagram



The I^2C interrupts are signalled through a single pin which provides a level sensitive interrupt to the 4138xx interrupt control unit. The I^2C Bus Interface Unit can cause and interrupt when a buffer is full, buffer empty, slave address detected, arbitration lost, or bus error condition occurs. All interrupt conditions must be cleared explicitly by software. See Section 14.8.2, “I2C Status Register x — ISRx” on page 717 for details.

The I^2C Data Buffer Register (IDBR) is an 8-bit data buffer that receives a byte of data from the shift register interface of the I^2C bus on one side and parallel data from the 4138xx internal bus on the other side. The serial shift register is not user accessible.

The control and status registers are located in the I^2C memory-mapped address space which ar eat offset (+2140H to +21BFH). The registers and their function are defined in Section 14.8.

The I^2C Bus Interface Unit supports fast mode operation of 400 Kbits/sec. Fast mode logic levels, formats, and capacitive loading, and protocols are exactly the same as the 100 Kbits/sec standard mode. Because the data setup and hold times differ between the fast and standard mode, the I^2C is designed to meet the slower, standard mode requirements for these two specifications. Refer to the I^2C Bus Specification for details.

14.2.2 I^{2}C Bus Interface Modes

The I^{2}C Bus Interface Unit can be in different modes of operation to accomplish a transfer. Table 465 summarizes the different modes.

Table 465. Modes of Operation

Mode	Definition
Master — Transmit	• I^{2}C Bus Interface Unit acts as a master. • Used for a write operation. • I^{2}C Bus Interface Unit sends the data. • I^{2}C Bus Interface Unit is responsible for clocking. • Slave device is in slave-receive mode
Master — Receive	• I^{2}C Bus Interface Unit acts as a master. • Used for a read operation. • I^{2}C Bus Interface Unit receives the data. • I^{2}C Bus Interface Unit is responsible for clocking. • Slave device is in slave-transmit mode
Slave — Transmit	• I^{2}C Bus Interface Unit acts as a slave. • Used for a read (master) operation. • I^{2}C Bus Interface Unit sends the data. • Master device is in master-receive mode.
Slave — Receive (default)	• I^{2}C Bus Interface Unit acts as a slave. • Used for a write (master) operation. • I^{2}C Bus Interface Unit receives the data. • Master device is in master-transmit mode.

While the I^{2}C Bus Interface Unit is in idle mode (neither receiving or transmitting serial data), the unit defaults to Slave-Receive mode. This allows the interface to monitor the bus and receive any slave addresses that might be intended for the 4138xx.

When the I^{2}C Bus Interface Unit receives an address that matches the 7-bit address found in the I^{2}C Slave Address Register (ISAR) or the General Call Address (00H), the interface either remains in Slave-Receive mode or transitions to Slave-Transmit mode. This is determined by the Read/Write (R/W#) bit (the least significant bit of the byte containing the slave address). When the R/W# bit is low, the master initiating the transaction intends to do a write and the I^{2}C Bus Interface Unit remains in Slave-Receive mode. When the R/W# is high, the initiating master wants to read data and the slave transitions to Slave-Transmit mode. Slave operation is further defined in Section 14.3.5, "Slave Operations" on page 705.

When the 4138xx wants to initiate a read or write on the I^{2}C bus, the I^{2}C Bus Interface Unit transitions from the default Slave-Receive mode to Master-Transmit mode. When the 4138xx wants to write data, the interface remains in Master-Transmit mode after the address transfer has completed. (see Section 14.2.3.1, "START Condition" on page 696) for START information). When the 4138xx wants to read data, the I^{2}C Bus Interface Unit transmits the start address, then transition to Master-Receive mode. Master operation is further defined in Section 14.3.4, "Master Operations" on page 702.

14.2.3 Start and Stop Bus States

The I^2C bus defines a transaction START and a transaction STOP bus state that are used at the beginning and end of the transfer of one to an unlimited number of bytes on the bus.

The 4138xx uses the START and STOP bits in the I^2C Control Register (ICR) to:

- initiate an additional byte transfer
- initiate a START condition on the I^2C bus
- enable Data Chaining (repeated START)
- initiate a STOP condition on the I^2C bus

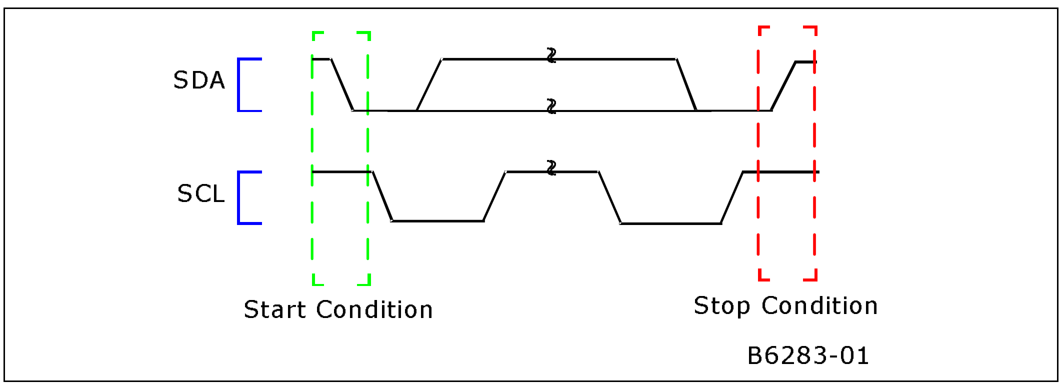
Table 466 summarizes the definition of the START and STOP bits in the ICR.

Table 466. START and STOP Bit Definitions

STOP bit	START bit	Condition	Notes
0	0	No START or STOP	• No START or STOP condition is sent by the I^2C Bus Interface Unit. This is used when multiple data bytes need to be transferred.
0	1	START Condition and Repeated START	• The I^2C Bus Interface Unit sends a START condition and transmit the contents of the 8 bit IDBR after the START. The IDBR must contain the 7-bit address and the R/W# bit before a START is initiated. • For a repeated start, the IDBR contents contains the target slave address and the R/W# bit. This enables multiple transfers to different slaves without giving up the bus. • The interface stays in Master-Transmit mode when a write is used or transition to master-receive mode when a read is requested.
1	X	STOP Condition	• In Master-Transmit mode, the I^2C Bus Interface Unit transmits the 8-bit IDBR and then send a STOP on the I^2C bus. • In Master-Receive mode, the Ack/Nack Control bit in the ICR must be changed to a negative Ack (see Section 14.3.2). The I^2C Bus Interface Unit writes the Nack bit (Ack/Nack Control bit must be 1), receive the data byte in the IDBR, then send a STOP on the I^2C bus.

Figure 93 shows the relationship between the **SDA** and **SCL** lines for a START and STOP condition.

Figure 93. Start and Stop Conditions



14.2.3.1 START Condition

The START condition (bits 1:0 of the ICR set to 01_2) initiates a master transaction or repeated START. Software must load the target slave address and the R/W# bit in the IDBR (see Section 14.8.4, “I2C Data Buffer Register x — IDBRx” on page 720) before setting the START ICR bit. The START and the IDBR contents are transmitted on the I^2C bus when the ICR Transfer Byte bit is set. The I^2C bus stays in master-transmit mode when a write is requested or enters master-receive mode when a read is requested. For a repeated start (a change in read or write or a change in the target slave address), the IDBR contains the updated target slave address and the R/W# bit. A repeated start enables multiple transfers to different slaves without giving up the bus.

The START condition is not cleared by the I^2C unit. When arbitration is lost while initiating a START, the I^2C unit may re-attempt the START when the bus becomes free. See Section 14.3.3, “Arbitration” on page 700 for details on how the I^2C unit functions under those circumstances.

14.2.3.2 No START or STOP Condition

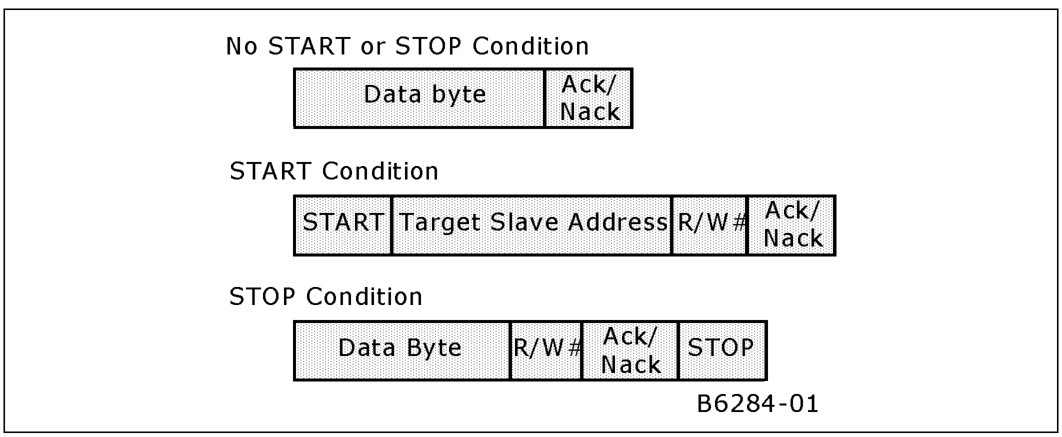
No START or STOP condition (bits 1:0 of the ICR set to 00_2) is used in master-transmit mode while the 4138xx is transmitting multiple data bytes (see Figure 93). Software writes the data byte, sets the IDBR Transmit Empty bit in the ISR (and interrupt when enabled), and clears the Transfer Byte bit in the ICR. The software then writes a new byte to the IDBR and sets the Transfer Byte ICR bit, which initiates the new byte transmission. This continues until the software sets the START or STOP bit. The START and STOP bits in the ICR are not automatically cleared by the I^2C unit after the transmission of a START, STOP or repeated START.

After each byte transfer (including the Ack/Nack bit) the I^2C unit holds the **SCL** line low (inserting wait states) until the Transfer Byte bit in the ICR is set. This action notifies the I^2C unit to release the **SCL** line and allow the next information transfer to proceed.

14.2.3.3 STOP Condition

The STOP condition (bits 1:0 of the ICR set to 10_2) terminates a data transfer. In master-transmit mode, the STOP bit and the Transfer Byte bit in the ICR must be set to initiate the last byte transfer (see Figure 93). In master-receive mode, to initiate the last transfer the 4138xx must set the Ack/Nack bit, the STOP bit, and the Transfer Byte bit in the ICR. Software must clear the STOP condition after it is transmitted.

Figure 94. START and STOP Conditions



14.3 I^2C Bus Operation

The I^2C Bus Interface Unit transfers in 1 byte increments. A data transfer on the I^2C bus always follows the sequence:

1. START.
2. 7-bit Slave Address.
3. R/W# Bit.
4. Acknowledge Pulse.
5. 8 Bits of Data.
6. Ack/Nack Pulse.
7. Repeat of Step 5 and 6 for Required Number of Bytes.
8. Repeated START (Repeat Step 1) or STOP.
9. Serial Clock Line (**SCL**) Generation.

The 4138xx's I^2C unit is required to generate the I^2C clock output when in master mode (either receive or transmit). **SCL** clock generation is accomplished through the use of the Fast Mode Enable bit, which is programmed at initialization. The following equation is used to determine the **SCL** transition period:

Equation 18. SCL Transition Period

```
SCL Transition Period = (30 ns) * (167 - (Fast Mode Enable * 120))
```

14.3.1 Data and Addressing Management

Data and slave addressing is managed via the I^2C Data Buffer Register (IDBR) and I^2C Slave Address Register (ISAR). IDBR (see Section 14.8.4, "I2C Data Buffer Register x — IDBRx") contains data or a slave address and R/W# bit. ISAR contains the 4138xx programmable slave address. Data coming into the I^2C unit is received into IDBR after a full byte is received and acknowledged. To transmit data, the processor writes to IDBR, and the I^2C unit passes this onto the serial bus when the Transfer Byte bit in the ICR is set. See Section 14.8.1, "I2C Control Register x — ICRx".

When the I^2C unit is in transmit mode (master or slave):

1. Software writes data to the IDBR over the internal bus. This initiates a master transaction or sends the next data byte, after the IDBR Transmit Empty bit is sent.
2. I^2C unit transmits data from IDBR when the Transmit Empty bit in the ICR is set.
3. When enabled, an IDBR Transmit Empty interrupt is signalled when a byte is transferred on the I^2C bus and the acknowledge cycle is complete.
4. When the I^2C bus is ready to transfer the next byte before the processor has written the IDBR (and a STOP condition is not in place), the I^2C unit inserts wait states until the processor writes a new value into the IDBR and sets the ICR Transfer Byte bit.

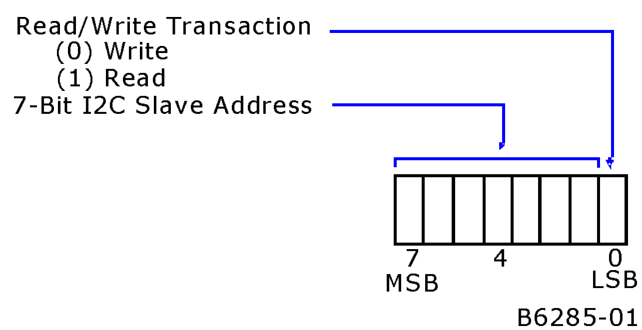
When the I^2C unit is in receive mode (master or slave):

1. The processor reads the IDBR data over the internal bus after the IDBR Receive Full interrupt is signalled.
2. I^2C unit transfers data from shift register to IDBR after the Ack cycle completes.
3. The I^2C unit inserts wait states until the IDBR is read. Refer to Section 14.3.2, "I2C Acknowledge" on page 699 for acknowledge pulse information in receiver mode.
4. After processor reads IDBR, the I^2C unit writes the ICRs Ack/Nack Control bit and the Transfer Byte bit, allowing the next byte transfer to proceed.

14.3.1.1 Addressing a Slave Device

As a master device, the I^{2}C unit must compose and send the first byte of a transaction. This byte consists of the slave address for the intended device and a R/W# bit for transaction definition. The slave address and the R/W# bit are written to the IDBR (see Figure 95).

Figure 95. Data Format of First Byte in Master Transaction



The first byte transmission must be followed by an Ack pulse from the addressed slave. When the transaction is a write, the I^{2}C unit remains in master-transmit mode and the addressed slave device stays in slave-receive mode. When the transaction is a read, the I^{2}C unit transitions to master-receive mode immediately following the Ack and the addressed slave device transitions to slave-transmit mode. When a Nack is returned, the I^{2}C unit aborts the transaction by automatically sending a STOP and setting the ISR bus error bit.

When the I^{2}C unit is enabled and idle (no bus activity), it stays in slave-receive mode and monitors the I^{2}C bus for a START signal. Upon detecting a START pulse, the I^{2}C unit reads the first seven bits and compares them to those in the I^{2}C Slave Address Register (ISAR) and the general call address (00H). When the bits match those of the ISAR register, the I^{2}C unit reads the eighth bit (R/W# bit) and transmits an Ack pulse. The I^{2}C unit either remains in slave-receive mode (R/W# = 0) or transitions to slave-transmit mode (R/W# = 1). See Section 14.3.6, “General Call Address” on page 707 for actions when a general call address is detected.

14.3.2 I²C Acknowledge

Every I²C byte transfer must be accompanied by an acknowledge pulse, which is always generated by the receiver (master or slave). The transmitter must release the **SDA** line for the receiver to transmit the acknowledge pulse (see Figure 96).

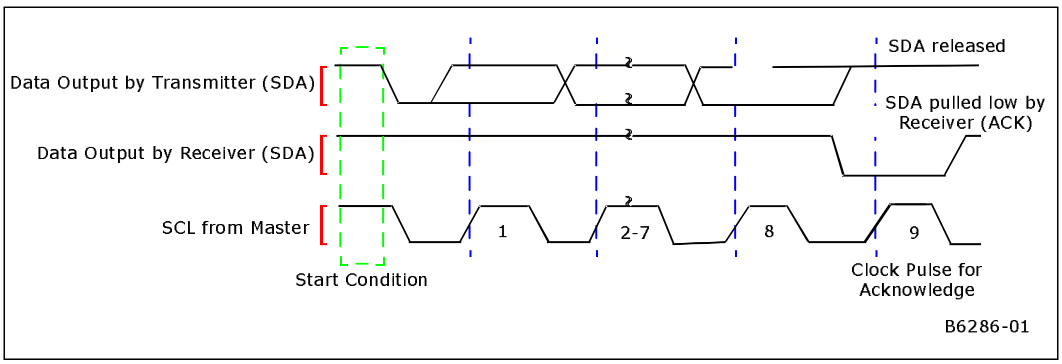
In master-transmit mode, when the target slave receiver device cannot generate the acknowledge pulse, the **SDA** line remains high. This lack of acknowledge (Nack) causes the I²C unit to set the bus error detected bit in the ISR and generate the associated interrupt (when enabled). The I²C unit aborts the transaction by generating a STOP automatically.

In master-receive mode, the I²C unit signals the slave-transmitter to stop sending data by using the negative acknowledge (Nack). The Ack/Nack bit value driven by the I²C bus is controlled by the Ack/Nack bit in the ICR. The bus error detected bit in the ISR is not set for a master-receive mode Nack (as required by the I²C bus protocol). The I²C unit automatically transmits the Ack pulse, based on the Ack/Nack ICR bit, after receiving each byte from the serial bus. Before receiving the last byte, software must set the Ack/Nack Control bit to Nack. Nack is then sent after the next byte is received to indicate the last byte.

In slave mode, the I²C unit automatically acknowledges its own slave address, independent of the Ack/Nack bit setting in the ICR. As a slave-receiver, an Ack response is automatically given to a data byte, independent of the Ack/Nack bit setting in the ICR. The I²C unit sends the Ack value after receiving the eighth data bit of the byte.

In slave-transmit mode, receiving a Nack from the master indicates the last byte is transferred. The master then sends either a STOP or repeated START. The ISR's unit busy bit (2) remains set until a STOP or repeated START is received.

Figure 96. Acknowledge on the I²C Bus



14.3.3 Arbitration

Arbitration on the I²C bus is required due to the multi-master capabilities of the I²C bus. Arbitration is used when two or more masters simultaneously generate a START condition within the minimum I²C hold time of the START condition.

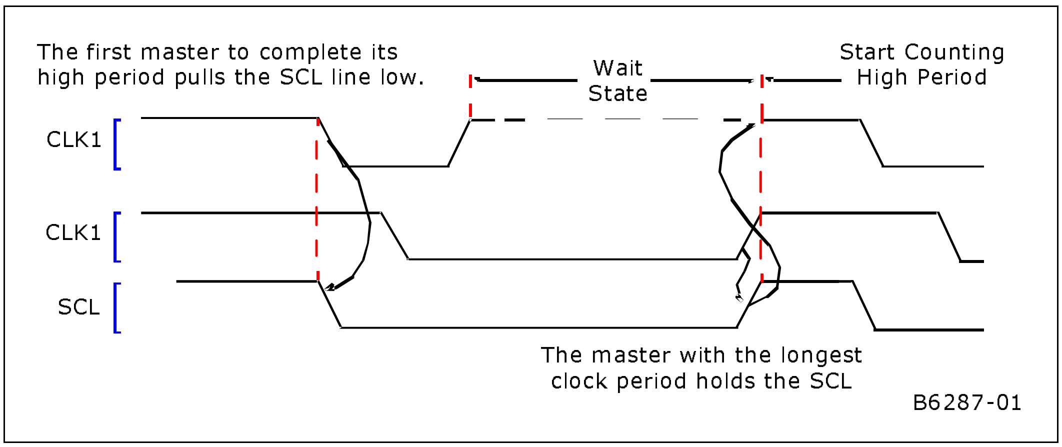
Arbitration can continue for a long period. When the address bit and the R/W# are the same, the arbitration moves to the data. Due to the wired-AND nature of the I²C bus, no data is lost when both (or all) masters are outputting the same bus states. When the address, the R/W# bit, or the data are different, the master which outputted the high state (master's data is different from **SDA**) loses arbitration and shut its data drivers off. When losing arbitration, the I²C Bus Interface Unit shuts off the **SDA** or **SCL** drivers for the remainder of the byte transfer, set the Arbitration Loss Detected bit, then return to idle (Slave-Receive) mode.

14.3.3.1 SCL Arbitration

Each master on the I²C bus generates its own clock on the **SCL** line for data transfers. With masters generating their own clocks, clocks with different frequencies may be connected to the **SCL** line. Since data is valid when the clock is in the high period, a defined clock synchronization procedure is needed during bit-by-bit arbitration.

Clock synchronization is accomplished by using the wired-AND connection of the I²C interfaces to the **SCL** line. When a master's clock transitions from high to low, this causes the master to hold down the **SCL** line for its associated period (see Figure 97). The low to high transition of the clock may not change when another master has not completed its period. Therefore, the master with the longest low period holds down the **SCL** line. Masters with shorter periods are held in a high wait-state during this time. Once the master with the longest period completes, the **SCL** line transitions to the high state, masters with the shorter periods can continue the data cycle.

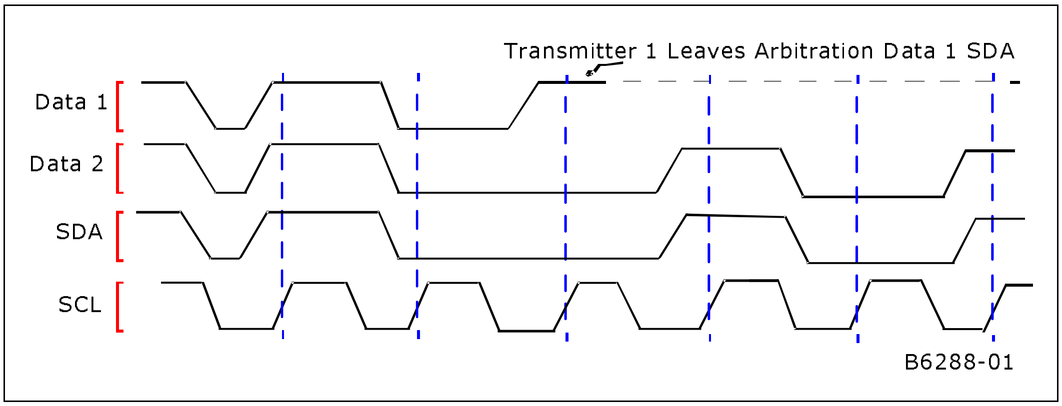
Figure 97. Clock Synchronization During the Arbitration Procedure



14.3.3.2 SDA Arbitration

Arbitration on the **SDA** line can continue for a long period, starting with address and R/W# bits and continuing with data bits. Figure 98 shows the arbitration procedure for two masters (more than two may be involved depending on how many masters are connected to the bus). When the address and R/W# are the same, arbitration moves to the data. Due to the wired-AND nature of the I^2C bus, no data is lost when both (or all) masters are outputting the same bus states. When address, R/W#, or data is different, the master that output the first low data bit loses arbitration and shuts its data drivers off. When the I^2C unit loses arbitration, it shuts off the **SDA** or **SCL** drivers for remainder of byte transfer, sets arbitration loss detected ISR bit, then returns to idle (Slave-Receive) mode.

Figure 98. Arbitration Procedure of Two Masters



When the I^2C unit loses arbitration during transmission of the seven address bits and the 4138xx is not being addressed as a slave device, the I^2C unit re-sends the address when the I^2C bus becomes free. This is possible because the IDBR and ICR registers are not overwritten when arbitration is lost.

When the arbitration loss is to due to another bus master addressing the 4138xx as a slave device, the I^2C unit switches to slave-receive mode and the original data in the I^2C data buffer register is overwritten. Software is responsible for clearing the start and re-initiating the master transaction at a later time.

Note: Software must not allow the I^2C unit to write to its own slave address. This can cause the I^2C bus to enter an indeterminate state.

Boundary conditions exist for arbitration when an arbitration process is in progress and a repeated START or STOP condition is transmitted on the I^2C bus. To prevent errors, the I^2C unit, acting as a master, provides for the following sequences:

- No arbitration takes place between a repeated START condition and a data bit
- No arbitration takes place between a data bit and a STOP condition
- No arbitration takes place between a repeated START condition and a STOP condition

These situations arise only when different masters write the same data to the same target slave simultaneously and arbitration is not resolved after the first data byte transfer.

Note: Typically, software is responsible for ensuring arbitration is lost soon after the transaction begins. For example, the protocol might insist that all masters transmit their I^2C address as the first data byte of any transaction ensuring arbitration is ended. A restart is then sent to begin a valid data transfer (slave can discard master address).

14.3.4 Master Operations

When software initiates a read or write on the I^{2}C bus, the I^{2}C unit transitions from the default slave-receive mode to master-transmit mode. The start pulse is sent followed by the 7-bit slave address and the R/W# bit. After the master receives an acknowledge, the I^{2}C unit has the option of two master modes:

- Master-Transmit — The 4138xx writes data
- Master-Receive — The 4138xx reads data

The 4138xx initiates a master transaction by writing to the ICR register. Data is read and written from the I^{2}C unit through the memory-mapped registers.

Table 467 describes the I^{2}C Bus Interface Unit responsibilities as a master device.

Table 467. Master Transactions (Sheet 1 of 2)

I^{2}C Master Action	Mode of Operation	Definition
Generate clock output	Master-transmit Master-receive	• The master always drives the **SCL** line. • The **SCL** Enable bit must be set. • The Unit Enable bit must be set.
Write target slave address to IDBR	Master-transmit Master-receive	• The Intel XScale® processor writes to IDBR bits 7-1 before a START condition is enabled. • First 7 bits sent on bus after START. • See Section 14.2.3.
Write R/W# Bit to IDBR	Master-transmit Master-receive	• The Intel XScale® processor writes to the least significant IDBR bit with the target slave address. • When low, the master remains a master-transmitter. When high, the master transitions to a master-receiver. • See Section 14.3.1.
Signal START Condition	Master-transmit Master-receive	• See "Generate clock output" above. • Performed after the target slave address and the R/W# bit are in the IDBR. • Intel XScale® processor sets the START bit. • Intel XScale® processor sets the Transfer Byte bit which initiates the start condition. • See Section 14.2.3.
Initiate first data byte transfer	Master-transmit Master-receive	• Intel XScale® processor writes byte to IDBR • I^{2}C Bus Interface Unit transmits the byte when the Transfer Byte bit is set. • I^{2}C Bus Interface Unit clears the Transfer Byte bit and sets the IDBR Transmit Empty bit when the transfer is complete.
Arbitrate for I^{2}C Bus	Master-transmit Master-receive	• When two or more masters signal a start within the same clock period, arbitration must occur. • The I^{2}C Bus Interface Unit arbitrates for as long as necessary. Arbitration takes place during slave address, R/W# bit, and data transmission and continues until all but one master loses the bus. No data is lost during arbitration. • When the I^{2}C Bus Interface Unit loses arbitration, it sets the Arbitration Loss Detect ISR bit after byte transfer is complete and transition to slave-receive (default) mode. • When I^{2}C Bus Interface Unit loses arbitration while attempting to send the target address byte, the I^{2}C Bus Interface Unit attempts to resend it when the bus becomes free. • The system designer must ensure the boundary conditions described in Section 14.3 do not occur.
Write one data byte to the IDBR	Master-transmit only	• Data transmit mode of I^{2}C master operation. • Occurs when the IDBR Transmit Empty ISR bit is set and the Transfer Byte bit is clear. When enabled, the IDBR Transmit Empty Interrupt is signalled to the Intel XScale® processor. • Intel XScale® processor writes 1 data byte to the IDBR, set the appropriate START/STOP bit combination, and then set the Transfer Byte bit to send the data. Eight bits are written on the serial bus followed by a STOP when requested.

Table 467. Master Transactions (Sheet 2 of 2)

I^2C Master Action	Mode of Operation	Definition
Wait for Acknowledge from slave-receiver	Master-transmit only	• As a master-transmitter, the I^2C Bus Interface Unit generates the clock for the acknowledge pulse. The I^2C Bus Interface Unit is responsible for releasing the **SDA** line to allow slave-receiver Ack transmission. • See Section 14.3.2.
Read one byte of I^2C Data from the IDBR	Master-receive only	• Data receive mode of I^2C master operation. • Eight bits are read from the serial bus, collected in the shift register then transferred to the IDBR after the Ack/Nack bit is read. • The Intel XScale® processor reads the IDBR when the IDBR Receive Full bit is set and the Transfer Byte bit is clear. When enabled, a IDBR Receive Full Interrupt is signalled to the Intel XScale® microarchitecture. • When the IDBR is read, when the Ack/Nack Status is clear (indicating Ack), the Intel XScale® processor writes the Ack/Nack Control bit and set the Transfer Byte bit to initiate the next byte read. • When the Ack/Nack Status bit is set (indicating Nack), Transfer Byte bit is clear, STOP bit in the ICR is set, and Unit Busy bit in the ISR is set, then the last data byte has been read into the IDBR and the I^2C Bus Interface Unit is sending the STOP. • When the Ack/Nack Status bit is set (indicating Nack), Transfer Byte bit is clear, but the STOP bit is clear, then the Intel XScale® processor has two options: 1. set the START bit, write a new target address to the IDBR, and set the Transfer Byte bit which sends a repeated start condition, 2. set the Master Abort bit and leave the Transfer Byte clear which sends a STOP only.
Transmit Acknowledge to slave-transmitter	Master-receive only	• As a master-receiver, the I^2C Bus Interface Unit generates the clock for the acknowledge pulse. The I^2C Bus Interface Unit is also responsible for driving the **SDA** line during the Ack cycle. • When the next data byte is to be the last transaction, the Intel XScale® processor sets the Ack/Nack Control bit for Nack generation. • See Section 14.3.2.
Generate a Repeated START to chain I^2C transactions	Master-transmit Master-receive	• When data chaining is desired, a repeated START condition is used instead of a STOP condition. • This occurs after the last data byte of a transaction has been written to the bus. • The Intel XScale® processor writes the next target slave address and the R/W# bit to the IDBR, set the START bit, and set the Transfer Byte bit. • See Section 14.2.3.
Generate a STOP	Master-transmit Master-receive	• Generated after the Intel XScale® processor writes the last data byte on the bus. • Intel XScale® processor generates a STOP condition by setting the STOP bit in the ICR. • See Section 14.2.3.

When the 4138xx needs to read data, the I^{2}C unit transitions from slave-receive mode to master-transmit mode to transmit the start address and immediately following the ACK pulse transitions to master-receive mode to wait for the reception of the read data from the slave device (see Figure 99). It is also possible to have multiple transactions during an I^{2}C operation such as transitioning from master-receive to master-transmit through a repeated start or Data Chaining (see Figure 100). Figure 101 shows the wave forms of **SDA** and **SCL** for a complete data transfer.

Figure 99. Master-Receiver Read from Slave-Transmitter

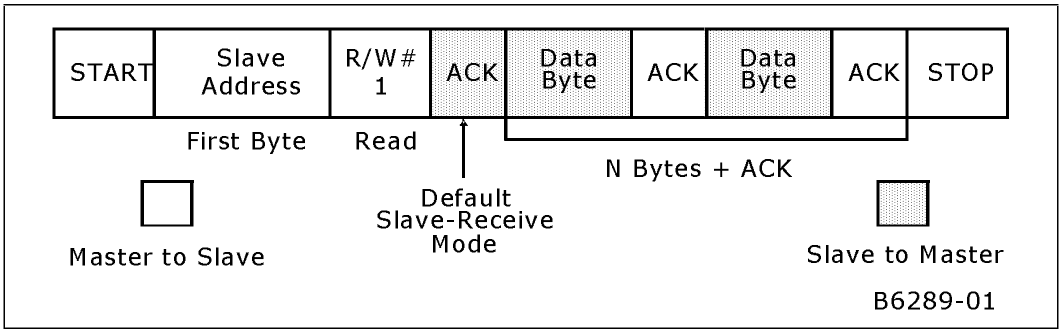


Figure 100. Master-Receiver Read from Slave-Transmitter / Repeated Start /Master-Transmitter Write to Slave-Receiver

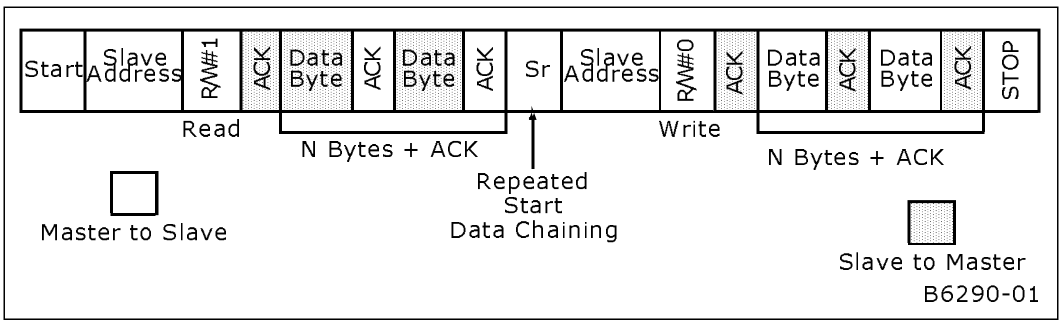
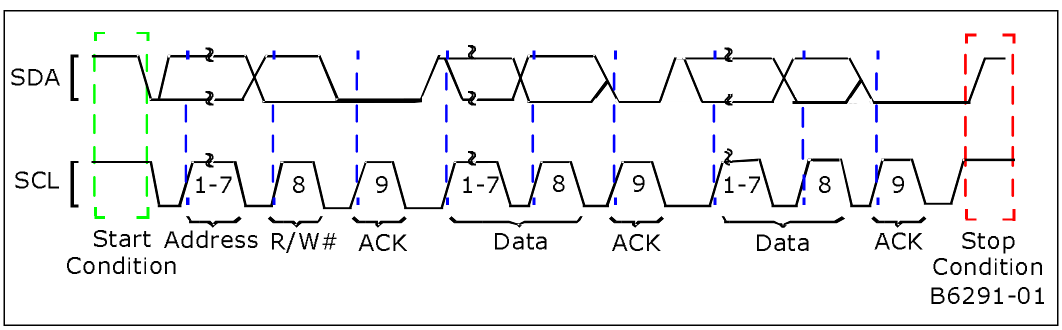


Figure 101. A Complete Data Transfer



14.3.5 Slave Operations

Table 468 describes the I^2C Bus Interface Unit's responsibilities as a slave device.

Table 468. Slave Transactions

I^2C Slave Action	Mode of Operation	Definition
Slave-receive (default mode)	Slave-receive only	• I^2C Bus Interface Unit monitors all slave address transactions. • The I^2C Bus Interface Unit Enable bit must be set. • I^2C Bus Interface Unit monitors bus for START conditions. When a START is detected, the interface reads the first 8 bits and compares the most significant 7 bits with the 7 bit I^2C Slave Address Register and the General Call address (00H). When there is a match, the I^2C Bus Interface Unit sends an Ack. • When the first 8 bits are all zeros, this is a general call address. When the General Call Disable bit is clear, both the General Call Address Detected bit and the Slave Mode Operation bit in the ISR are set. See Section 14.3.6. • When the 8th bit of the first byte (R/W# bit) is low, the I^2C Bus Interface Unit stays in slave-receive mode and the Slave Mode Operation bit is cleared. When the R/W# bit is high, the I^2C Bus Interface Unit transitions to slave-transmit mode and the Slave Mode Operation bit is set.
Setting the Slave Address Detected bit	Slave-receive Slave-transmit	• Indicates the interface has detected an I^2C operation that addresses the 4138xx (this includes general call address). The Intel XScale® processor can distinguish an ISAR match from a General Call by reading the General Call Address Detected bit. • An interrupt is signalled (when enabled) after the matching slave address is received and acknowledged.
Read one byte of I^2C Data from the IDBR	Slave-receive only	• Data receive mode of I^2C slave operation. • Eight bits are read from the serial bus into the shift register. When a full byte has been received and the Ack/Nack bit has completed, the byte is transferred from the shift register to the IDBR. • Occurs when the IDBR Receive Full bit in the ISR is set and the Transfer Byte bit is clear. When enabled, the IDBR Receive Full Interrupt is signalled to the Intel XScale® processor. • Intel XScale® processor reads 1 data byte from the IDBR. When the IDBR is read, the Intel XScale® processor writes the desired Ack/Nack Control bit and set the Transfer Byte bit. This causes the I^2C Bus Interface Unit to stop inserting wait states and let the master transmitter write the next piece of information.
Transmit Acknowledge to master-transmitter	Slave-receive only	• As a slave-receiver, the I^2C Bus Interface Unit is responsible for pulling the **SDA** line low to generate the Ack pulse during the high **SCL** period. • The Ack/Nack Control bit controls the Ack data the I^2C Bus Interface Unit drives. See Section 14.3.2.
Write one byte of I^2C data to the IDBR	Slave-transmit only	• Data transmit mode of I^2C slave operation. • Occurs when the IDBR Transmit Empty bit is set and the Transfer Byte bit is clear. When enabled, the IDBR Transmit Empty Interrupt is signalled to the Intel XScale® processor. • Intel XScale® processor writes a data byte to the IDBR and set the Transfer Byte bit to initiate the transfer.
Wait for Acknowledge from master-receiver	Slave-transmit only	• As a slave-transmitter, the I^2C Bus Interface Unit is responsible for releasing the **SDA** line to allow the master-receiver to pull the line low for the Ack. • See Section 14.3.2.

Figure 102 through Figure 104 are examples of I^2C transactions. These show the relationships between master and slave devices.

Figure 102. Master-Transmitter Write to Slave-Receiver

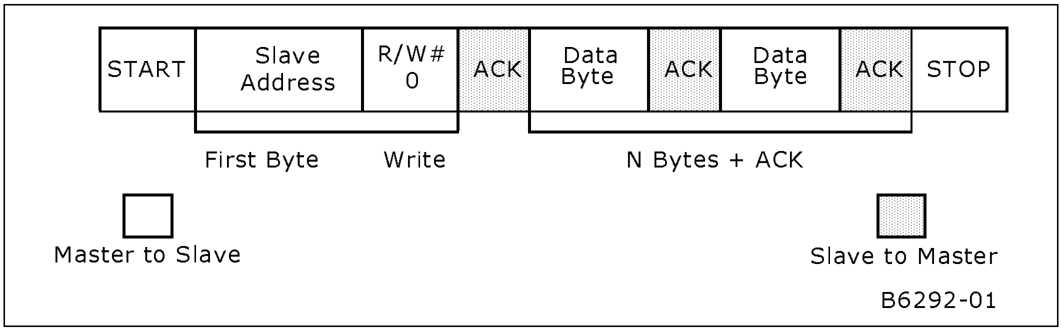


Figure 103. Master-Receiver Read to Slave-Transmitter

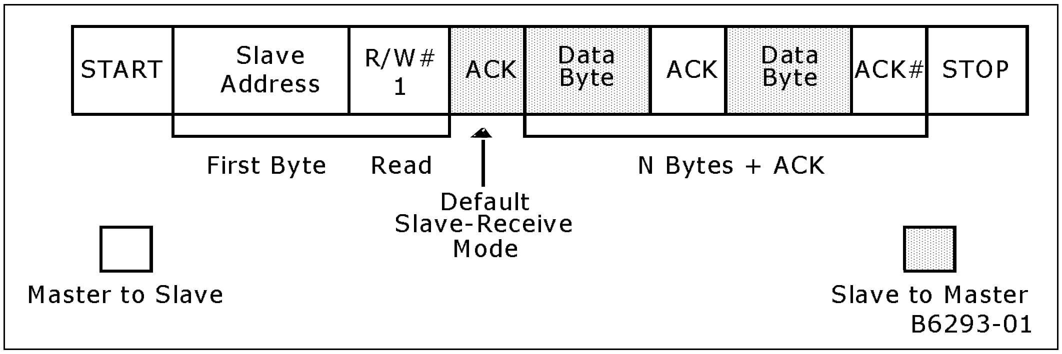
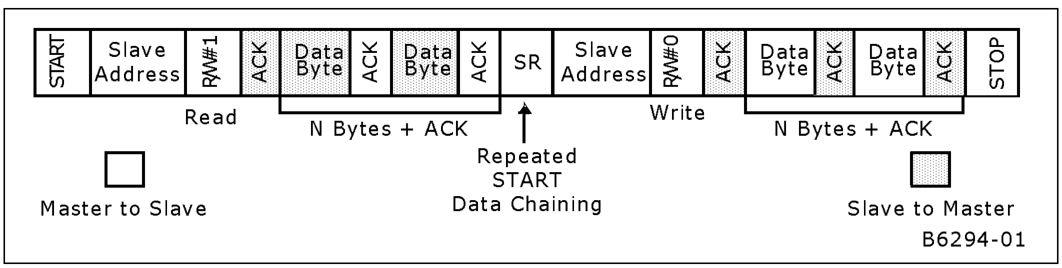


Figure 104. Master-Receiver Read to Slave-Transmitter, Repeated START, Master-Transmitter Write to Slave-Receiver



14.3.6 General Call Address

The I²C unit supports both sending and receiving general call address transfers on the I²C bus. When sending a general call message from the I²C unit, software must set the General Call Disable bit in the ICR to keep the I²C unit from responding as a slave. Failure to set this bit causes the I²C Bus to enter an indeterminate state.

A general call address is defined as a transaction with a slave address of 00H. When a device requires the data from a general call address, it acknowledges the transaction and stays in slave-receiver mode. Otherwise, the device can ignore the general call address. The second and following bytes of a general call transaction are acknowledged by every device using it on the bus. Any device not using these bytes must not Ack. The meaning of a general call address is defined in the second byte sent by the master-transmitter. Figure 105 shows a general call address transaction. The least significant bit (B) of the second byte defines the transaction. Table 469, "General Call Address Second Byte Definitions" on page 707 shows the valid values and definitions when B=0.

When the 4138xx is acting as a slave, and the I²C unit receives a general call address and the ICR General Call Disable bit is clear the I²C unit:

- Sets the ISR general call address detected bit
- Sets the ISR slave address detected bit
- Interrupts (when enabled) the 4138xx

When the I²C unit receives a general call address and the ICR General Call Disable bit is set, the I²C unit ignores the general call address.

Figure 105. General Call Address

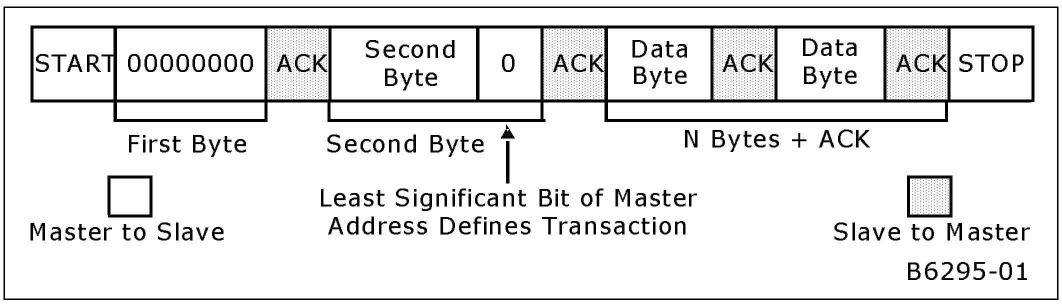


Table 469. General Call Address Second Byte Definitions

Least Significant Bit of Second Byte (B)	Second Byte Value	Definition
0	06H	2-byte transaction where the second byte tells the slave to reset and then store this value in the programmable part of their slave address.
0	04H	2-byte transaction where the second byte tells the slave to store this value in the programmable part of their slave address. No reset.
0	00H	Not allowed as a second byte

When directed to reset, the I²C Bus Interface Unit returns to its default reset condition with the exception of the ISAR. The 4138xx is responsible for ensuring this occurs, not the I²C Bus Interface Unit hardware.

When B=1, the sequence is used as a hardware general call by hardware masters only they cannot transmit a slave address, only their own address. The I²C Bus Interface Unit does not support this mode of operation.

I²C 10-bit addressing and CBUS compatibility are not supported.

14.4 Slave Mode Programming Examples

14.4.1 Initialize Unit

1. Write ISAR: Set slave address
2. Write ICR: Enable all interrupts, set Unit Enable

14.4.2 Write 1 Byte as a Slave

1. Wait for Slave Address Detected interrupt.
   Read ISR: Slave Address Detected (set), Unit Busy (set), R/W# bit (1), Ack/Nack (Clear - Ack)
2. Write IDBR: Load data byte to transfer
3. Write ICR: Set Transfer Byte bit
4. Wait for IDBR Transmit Empty interrupt.
   Read ISR: IDBR Transmit Empty (set), Ack/Nack (set - indicates last byte write), R/W# bit (0)
5. Clear interrupt by clearing the IDBR Transmit Empty Interrupt bit.
6. Wait for interrupt.
   Read ISR: Unit Busy (clear), Slave STOP Detected (set)
7. Clear interrupt by clearing Slave STOP Detected Interrupt bit.

14.4.3 Read 2 Bytes as a Slave

1. Wait for Slave Address Detected interrupt.
   Read ISR: Slave Address Detected (set), Unit busy (set), R/W# bit (0)
2. Read byte 1 on I^2C bus
   Write ICR: Set Transfer Byte bit to initiate the transfer
3. Wait for interrupt.
   Read ISR: IDBR Receive Full (set), Ack/Nack (clear), R/W# bit (0)
   Clear interrupt by clearing IDBR Receive Full bit.
   Read IDBR: To get the data.
4. Read byte 2 on I^2C bus
   Write ICR: Set Transfer Byte bit to initiate the transfer
5. Wait for interrupt.
   Read ISR: IDBR Receive Full (set), Ack/Nack (clear), R/W# bit (0)
   Clear interrupt by clearing IDBR Receive Full bit.
   Read IDBR: To get the data.
   Write ICR: Set Transfer Byte bit (to release I^2C bus allowing next transfer)
6. Wait for interrupt.
   Read ISR: Unit busy (clear), Slave STOP Detected (set)
   Clear interrupt by clearing Slave STOP Detected bit.

14.5 Master Programming Examples

14.5.1 Initialize Unit

1. Write ISAR: Set slave address
2. Write ICR: Enable all interrupts (except Arb Loss), set **SCL** Enable, set Unit Enable

14.5.2 Write 1 Byte as a Master

1. Write IDBR: Target slave address and R/W# bit (0 for write)
2. Write ICR: Set START bit, Clear STOP bit, Set Transfer Byte bit to initiate the access
3. Wait for IDBR Transmit Empty interrupt. When interrupt arrives:
   Read status register: IDBR Transmit Empty (set), Unit Busy (set), R/W# bit (clear)
   Clear IDBR Transmit Empty Interrupt bit to clear the interrupt.

Note: Arbitration Loss Detected bit may be set. When arbitration was lost, because Arb Loss interrupt was disabled, an address retry occurs when bus becomes free. Clear Arbitration Loss Detected bit when set.

4. Send byte with STOP
   Write IDBR: With data byte to send
   Write ICR: Clear START bit, Set STOP bit, Enable Arb Loss interrupt, Set Transfer Byte bit to initiate the access
5. Wait for Buffer empty interrupt. When interrupt arrives (Note: Unit is sending STOP):
   Read status register: IDBR Transmit Empty (set), Unit busy (set - maybe), R/W# bit (clear)
   Clear IDBR Transmit Empty Interrupt bit to clear the interrupt.
   Clear ICR STOP bit (optional)
   Wait until Unit busy is clear before clearing the ICR SCL Enable bit.

14.5.3 Read 1 Byte as a Master

1. Write IDBR: Target slave address and R/W# bit (1 for read)
2. Write ICR: Set START bit, Clear STOP bit, Disable Arb loss interrupt, Set Transfer Byte bit to initiate the access
3. Wait for IDBR Transmit Empty interrupt. When interrupt arrives:
   Read status register: IDBR Transmit Empty (set), Unit busy (set), R/W# bit (set)
   Clear IDBR Transmit Empty bit to clear the interrupt.
4. Read byte with STOP
   Write ICR: Clear START bit, Set STOP bit, Enable arb loss interrupt, Set Ack/Nack bit (Nack), Set Transfer Byte bit to initiate the access
5. Wait for Buffer full interrupt. When interrupt arrives (Note: Unit is sending STOP):
   Read status register: IDBR Receive Full (set), Unit Busy (set - maybe), R/W# bit (Set), Ack/Nack bit (Set)
   Clear IDBR Receive Full bit to clear the interrupt.
   Read IDBR data.
   Clear ICR STOP bit (optional), Clear ICR Ack/Nack Control bit (optional)
   Wait until Unit busy is clear before clearing the ICR SCL Enable bit. (optional)

14.5.4 Write 2 Bytes and Repeated Start Read 1 Byte as a Master

1. Write IDBR: Target slave address and R/W# bit (0 for write)
2. Write ICR: Set START bit, Clear STOP bit, Set Transfer Byte bit to initiate the access
3. Wait for IDBR Transmit Empty interrupt. When interrupt arrives:
   Read status register: IDBR Transmit Empty (set), Unit busy (set), R/W# bit (clear)
   Clear IDBR Transmit Empty bit to clear the interrupt.
4. Send byte 1
   Write IDBR: With data byte to send
   Write ICR: Clear START bit, Clear STOP bit, Enable Arb Loss interrupt, Set Transfer Byte bit to initiate the access
5. Wait for Buffer empty interrupt.
   Read status register: IDBR Transmit Empty (set), Unit busy (set), R/W# bit (clear)
   Clear IDBR Transmit Empty bit to clear the interrupt.
6. Send byte 2
   Write IDBR: With data byte to send
   Write ICR: Clear START bit, Clear STOP bit, Set Transfer Byte bit to initiate the access
7. Wait for Buffer empty interrupt.
   Read status register: IDBR Transmit Empty (set), Unit busy (set), R/W# bit (clear)
   Clear IDBR Transmit Empty bit to clear the interrupt.
8. Send repeated start as a master
   Write IDBR: Target slave address and R/W# bit (1 for read)
   Write ICR: Set START bit, Clear STOP bit, Disable Arb Loss interrupt, Set Transfer Byte bit the initiate the access
9. Wait for IDBR Transmit Empty interrupt. When interrupt comes.
   Read status register: IDBR Transmit Empty (set), Unit busy (set), R/W# bit (set)
   Clear IDBR Transmit Empty bit to clear the interrupt.
10. Read byte with STOP
    Write ICR: Clear START bit, Set STOP bit, Enable arb loss interrupt, Set Ack/Nack bit (Nack), Set Transfer Byte bit to initiate the access
11. Wait for Buffer full interrupt. When interrupt comes (Note: Unit is sending STOP).
    Read status register: IDBR Receive Full (set), Unit busy (set - maybe), R/W# bit (Set), Ack/Nack bit (Set)
    Clear IDBR Receive Full bit to clear the interrupt.
    Read IDBR data.
    Clear ICR STOP bit (optional), Clear ICR Ack/Nack Control bit (optional)
    Wait until Unit busy is clear before clearing the ICR SCL Enable bit. (optional)

14.5.5 Read 2 Bytes as a Master — Send STOP Using the Abort

1. Write IDBR: Target slave address and R/W# bit (1 for read)
2. Write ICR: Set START bit, Clear STOP bit, Disable Arb loss interrupt, Set Transfer Byte bit to initiate the access
3. Wait for IDBR Transmit Empty interrupt. When interrupt comes.
   Read status register: IDBR Transmit Empty (set), Unit Busy (set), R/W# bit (set)
   Clear IDBR Transmit Empty bit to clear the interrupt.
4. Read byte 1
   Write ICR: Clear START bit, Clear STOP bit, Enable Arb Loss interrupt, Clear Ack/Nack bit (Ack), Set Transfer Byte bit to initiate the access
5. Wait for Buffer full interrupt.
   Read status register: IDBR Receive Full (set), Unit busy (set), R/W# bit (Set), Ack/Nack bit (Clear)
   Clear IDBR Receive Full bit to clear the interrupt.
   Read IDBR data.
6. Read byte 2 with Nack (STOP is not set because STOP or Repeated START are decided on the byte read)
   Write ICR: Clear START bit, Clear STOP bit, Enable Arb Loss interrupt, Set Ack/Nack bit (Nack), Set Transfer Byte bit to initiate the access
7. Wait for Buffer full interrupt.
   Read status register: IDBR Receive Full (set), Unit Busy (set), R/W# bit (Set), Ack/Nack bit (Set)
   Clear IDBR Receive Full bit to clear the interrupt.
   Read IDBR data.

There are now two options based on the byte read:

- Send a repeated START
- Send a STOP only

Here, a STOP abort is sent.

Note: Had a NACK not been sent, the next transaction *must* involve another data byte read.

8. Send STOP abort condition. (STOP with no data transfer.)
   Write ICR: Set Master abort.

14.6 Glitch Suppression Logic

The I^2C Bus Interface Unit has built-in glitch suppression logic. Glitches are suppressed according to: 2 * I^2C clock period. For example, with the 33 MHz (30. ns period) I^2C clock glitches of 60ns or less are suppressed. This is within the 50 ns glitch suppression specified.

14.7 Reset Conditions

The I^2C unit is reset with internal bus reset. Software is responsible for ensuring the I^2C unit is not busy (ISR[3]) before asserting reset. Software is also responsible for ensuring the I^2C bus is idle when the unit is enabled after reset. When directed to reset, the I^2C unit returns to its default reset condition with the exception of the ISAR. ISAR is not affected by a reset.

When the Unit Reset bit in the ICRx is set, only the 4138xx I^2C unit resets, the associated I^2C MMRs remain intact. When resetting the I^2C unit with the ICRx unit reset, use the following guidelines:

1. In the ICRx register, set the reset bit and clear the remainder of the register.
2. Clear the ISRx register.
3. Clear reset in the ICRx.

14.8 Register Definitions

The following registers are associated with the I²C Bus Interface Units. Each I²C Bus Interface Unit has five memory-mapped control registers for independent operation. In register titles, x is 0 or 1 for unit 0 or 1, respectively.

They are all located within the peripheral memory- mapped address space of the 4138xx.

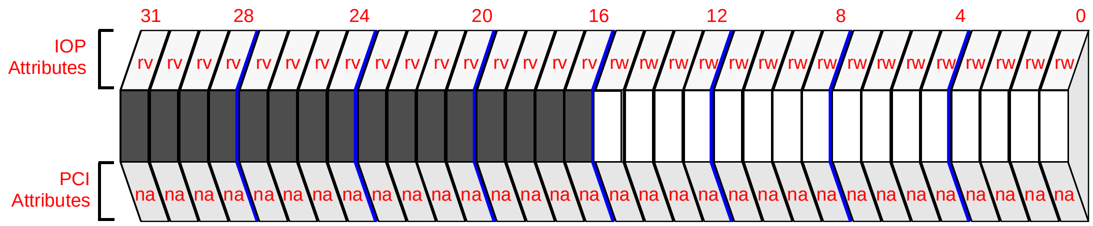
Table 470. I²C Register Summary

Section, Register Name, Acronym, Page
Section 14.8.1, "I2C Control Register x — ICRx" on page 715
Section 14.8.2, "I2C Status Register x — ISRx" on page 717
Section 14.8.3, "I2C Slave Address Register x — ISARx" on page 719
Section 14.8.4, "I2C Data Buffer Register x — IDBRx" on page 720
Section 14.8.5, "I2C Bus Monitor Register x — IBMRx" on page 721
Section 14.8.6, "I2C Manual Bus Control Register x — IMBCRx" on page 722

14.8.1 I^2C Control Register x — ICRx

The 4138xx uses the bits in the I^2C Control Register (ICRx) to control the I^2C unit.

Table 471. I^2C Control Register x — ICRx (Sheet 1 of 2)

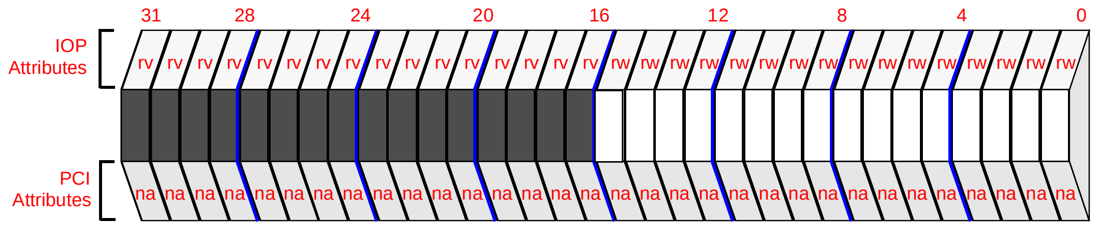


Unit #	Intel XScale® processor internal bus address offset
0	+2500H
1	+2520H
2	+2540H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:16	0000H	Reserved
15	0	Fast Mode: 0 = 100 KBit/sec operation 1 = 400 KBit/sec operation
14	0_2	**Unit Reset**: 0 = No reset. 1 = Reset the I^2C unit only.
13	0_2	**Slave Address Detected Interrupt Enable**: 0 = Disable interrupt. 1 = Enables I^2C unit to interrupt the 4138xx upon detecting a slave address match or a general call address.
12	0_2	**Arbitration Loss Detected Interrupt Enable**: 0 = Disable interrupt. 1 = Enables the I^2C unit to interrupt the 4138xx upon losing arbitration while in master mode.
11	0_2	**Slave STOP Detected Interrupt Enable**: 0 = Disable interrupt. 1 = Enables I^2C unit to interrupt the 4138xx when it detects a STOP condition in slave mode.
10	0_2	**Bus Error Interrupt Enable**: 0 = Disable interrupt. 1 = Enables the I^2C unit to interrupt the 4138xx for the following I^2C bus errors: • As a master transmitter, no Ack was detected after a byte was sent. • As a slave receiver, the I^2C unit generated a Nack pulse. ***Note:*** Software is responsible for insuring that misplaced START and STOP conditions do not occur. See Section 14.6, "Glitch Suppression Logic" on page 712.
09	0_2	**IDBR Receive Full Interrupt Enable**: 0 = Disable interrupt. 1 = Enables I^2C unit to interrupt the 4138xx when IDBR has received a data byte from the I^2C bus.
08	0_2	**IDBR Transmit Empty Interrupt Enable**: 0 = Disable interrupt. 1 = Enables the I^2C unit to interrupt the 4138xx after transmitting a byte onto the I^2C bus.
07	0_2	**General Call Disable**: 0 = Enables the I^2C unit to respond to general call messages. 1 = Disables I^2C unit response to general call messages as a slave. This bit must be set when sending a master mode general call message from the I^2C unit.
06	0_2	I^2C **Unit Enable**: 0 = Disables the unit and does not master any transactions or respond to any slave transactions. 1 = Enables the I^2C unit (defaults to slave-receive mode). Software must insure the I^2C bus is idle before setting this bit.
05	0_2	**SCL Enable**: 0 = Disables the I^2C unit from driving the **SCL** line. 1 = Enables the I^2C clock output for master mode operation.

Table 471. I²C Control Register x — ICRx (Sheet 2 of 2)



Unit #	Intel XScale® processor internal bus address offset
0	+2500H
1	+2520H
2	+2540H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
04	0_2	**Master Abort**: used by the I²C unit when in master mode to generate a STOP without transmitting another data byte. 0 = The I²C unit transmits STOP using the STOP ICR bit only. 1 = The I²C unit sends STOP without data transmission. When in Master transmit mode, after transmitting a data byte, the ICR's Transfer Byte bit is clear and IDBR Transmit Empty bit is set. When no more data bytes need to be sent, setting master abort bit sends the STOP. The Transfer Byte bit (03) must remain clear. In master-receive mode, when a Nack is sent without a STOP (STOP ICR bit was not set) and the 4138xx does not send a repeated START, setting this bit sends the STOP. Once again, the Transfer Byte bit (03) must remain clear.
03	0_2	**Transfer Byte**: used to send/receive a byte on the I²C bus. 0 = Cleared by I²C unit when the byte is sent/received. 1 = Send/receive a byte. The 4138xx can monitor this bit to determine when the byte transfer has completed. In master or slave mode, after each byte transfer including Ack/Nack bit, the I²C unit holds the **SCL** line low (inserting wait states) until the Transfer Byte bit is set.
02	0_2	**Ack/Nack Control**: defines the type of Ack pulse sent by the I²C unit when in master receive mode. 0 = The I²C unit sends an Ack pulse after receiving a data byte. 1 = The I²C unit sends a negative Ack (Nack) after receiving a data byte. The I²C unit automatically sends an Ack pulse when responding to its slave address or when responding in slave-receive mode, independent of the Ack/Nack control bit setting.
01	0_2	**STOP**: used to initiate a STOP condition after transferring the next data byte on the I²C bus when in master mode. In master-receive mode, the Ack/Nack control bit must be set in conjunction with this bit. See Section 14.2.3.3, "STOP Condition" on page 696 for more details on the STOP state. 0 = Do not send a STOP. 1 = Send a STOP.
00	0_2	**START**: used to initiate a START condition to the I²C unit when in master mode. See Section 14.2.3.1, "START Condition" on page 696 for more details on the START state. 0 = Do not send a START. 1 = Send a START.

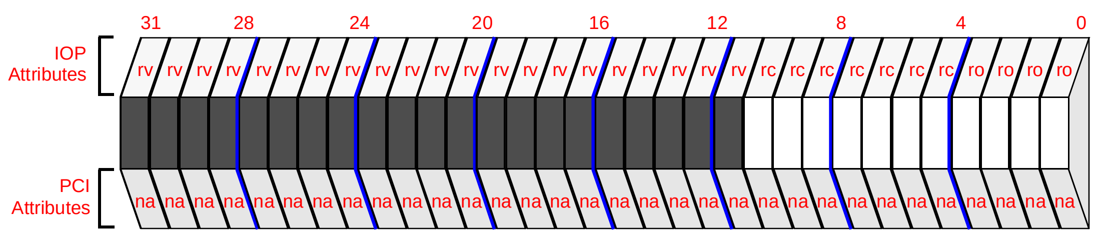
14.8.2 I^2C Status Register x — ISRx

I^2C interrupts are signalled to the 4138xx interrupt controller by the I^2C Interrupt Status Register (ISRx). Software uses the ISR bits to check the status of the I^2C unit and bus. ISRx bits (bits 9-5) are updated after the Ack/Nack bit has completed on the I^2C bus.

The ISRx is also used to clear interrupts signalled from the I^2C Bus Interface Unit. These are:

- IDBRx Receive Full
- IDBRx Transmit Empty
- Slave Address Detected
- Bus Error Detected
- STOP Condition Detect
- Arbitration Lost

Table 472. I^2C Status Register x — ISRx (Sheet 1 of 2)

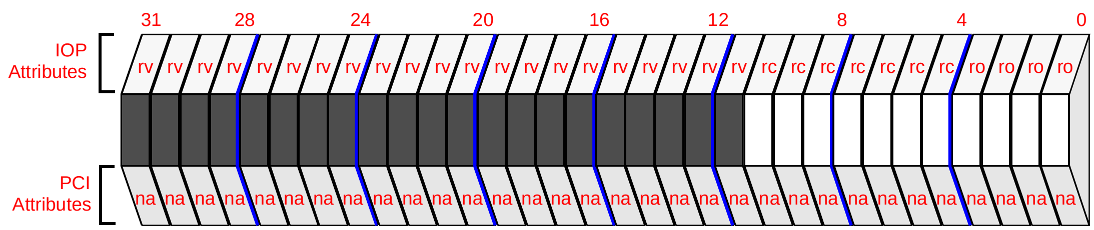


Unit #	Intel XScale® processor internal bus address offset
0	+2504H
1	+2524H
2	+2544H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:11	000000H	Reserved
10	0_2	**Bus Error Detected**: 0 = No error detected. 1 = The I^2C unit sets this bit when it detects one of the following error conditions: • As a master transmitter, no Ack was detected on the interface after a byte was sent. • As a slave receiver, the I^2C unit generates a Nack pulse. ***Note:*** When an error occurs, I^2C bus transactions continue. Software must insure that misplaced START and STOP conditions do not occur. See Section 14.3.3, "Arbitration" on page 700.
09	0_2	**Slave Address Detected**: 0 = No slave address detected. 1 = I^2C unit detected a 7-bit address that matches the general call address or ISAR. An interrupt is signalled when enabled in the ICR.
08	0_2	**General Call Address Detected**: 0 = No general call address received. 1 = I^2C unit received a general call address.
07	0_2	**IDBR Receive Full**: 0 = The IDBR has not received a new data byte or the I^2C unit is idle. 1 = The IDBR register received a new data byte from the I^2C bus. An interrupt is signalled when enabled in the ICR.
06	0_2	**IDBR Transmit Empty**: 0 = The data byte is still being transmitted. 1 = The I^2C unit has finished transmitting a data byte on the I^2C bus. An interrupt is signalled when enabled in the ICR.

Table 472. I^{2}C Status Register x — ISRx (Sheet 2 of 2)



Unit #	Intel XScale® processor internal bus address offset
0	+2504H
1	+2524H
2	+2544H

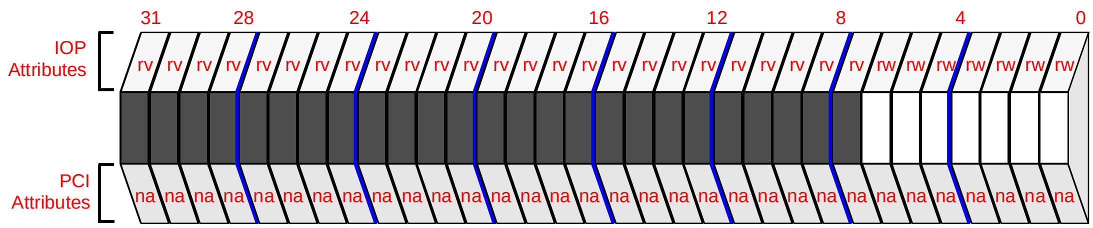
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
05	0_2	**Arbitration Loss Detected**: used during multi-master operation. 0 = Cleared when arbitration is won or never took place. 1 = Set when the I^{2}C unit loses arbitration.
04	0_2	**Slave STOP Detected**: 0 = No STOP detected. 1 = Set when the I^{2}C unit detects a STOP while in slave-receive or slave-transmit mode.
03	0_2	I^{2}C **Bus Busy**: 0 = I^{2}C bus is idle or the I^{2}C unit is using the bus (i.e., unit busy). 1 = Set when the I^{2}C bus is busy but the 4138xx I^{2}C unit is not involved in the transaction.
02	0_2	**Unit Busy**: 0 = I^{2}C unit not busy. 1 = Set when the 4138xx I^{2}C unit is busy. This is defined as the time between the first START and STOP.
01	0_2	**Ack/Nack Status**: 0 = The I^{2}C unit received or sent an Ack on the bus. 1 = The I^{2}C unit received or sent a Nack. This bit is used in slave transmit mode to determine when the byte transferred is the last one. This bit is updated after each byte and Ack/Nack information is received.
00	0_2	**Read/Write Mode**: 0 = The I^{2}C unit is in master-transmit or slave-receive mode. 1 = The I^{2}C unit is in master-receive or slave-transmit mode. This is the R/W# bit of the slave address. It is automatically cleared by hardware after a stop state.

14.8.3 I^{2}C Slave Address Register x — ISARx

The I^{2}C Slave Address Register (ISARx) (see Table 473) defines the I^{2}C unit 7-bit slave address to which the 4138xx responds when in slave-receive mode. This register is written by the 4138xx before enabling I^{2}C operations. The register is fully programmable (no address is assigned to the I^{2}C unit) so it can be set to a value other than those of hard-wired I^{2}C slave peripherals that might exist in the system. The ISAR is not affected by the 4138xx being reset. The ISAR register default value is 0000000_2.

Table 473. I^{2}C Slave Address Register x — ISARx



Unit #	Intel XScale® processor internal bus address offset
0	+2508H
1	+2528H
2	+2548H

Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
31:07	000000H	Reserved
06:00	00H	I^{2}C **Slave Address**: The 7-bit address to which the I^{2}C unit responds when in slave-receive mode.

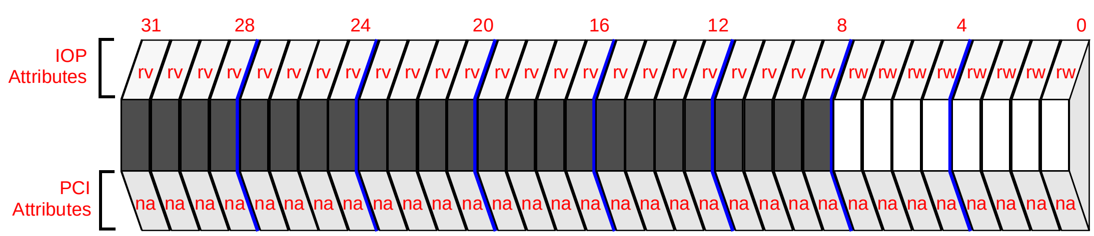
14.8.4 I²C Data Buffer Register x — IDBRx

The I²C Data Buffer Register (IDBRx) is used by the 4138xx to transmit and receive data from the I²C bus. The accesses the IDBRx by the 4138xx on one side and by the I²C shift register on the other. Data coming into the I²C Bus Interface Unit is received into the IDBRx after a full byte has been received and acknowledged. Data going out of the I²C Bus Interface Unit is written to the IDBRx by the Intel XScale® processor and sent to the serial bus.

When the I²C Bus Interface Unit is in transmit mode (master or slave), the 4138xx writes data to the IDBRx over the internal bus. This occurs when a master transaction is initiated or when the IDBRx Transmit Empty Interrupt is signalled. Data is moved from the IDBRx to the shift register when the Transfer Byte bit is set. The IDBR Transmit Empty Interrupt is signalled (when enabled) when a byte has been transferred on the I²C bus and the acknowledge cycle is complete. When the IDBRx is not written by the 4138xx (and a STOP condition was not in place) before the I²C bus is ready to transfer the next byte packet, the I²C Bus Interface Unit inserts wait states until the Intel XScale® processor writes the IDBRx and sets the Transfer Byte bit.

When the I²C Bus Interface Unit is in receive mode (master or slave), the processor reads IDBRx data over the internal bus. This occurs when the IDBRx Receive Full Interrupt is signalled. The data is moved from the shift register to the IDBRx when the Ack cycle is complete. The I²C Bus Interface Unit inserts wait states until the IDBR has been read. Refer to Section 14.3.2, "I2C Acknowledge" on page 699 for acknowledge pulse information in receiver mode. After the 4138xx reads the IDBRx, the Ack/Nack Control bit is written and the Transfer Byte bit is written, allowing the next byte transfer to proceed on the I²C Bus. The IDBRx register is 00H after reset.

Table 474. I²C Data Buffer Register x — IDBRx



Unit #	Intel XScale® processor internal bus address offset
0	+250CH
1	+252CH
2	+254CH

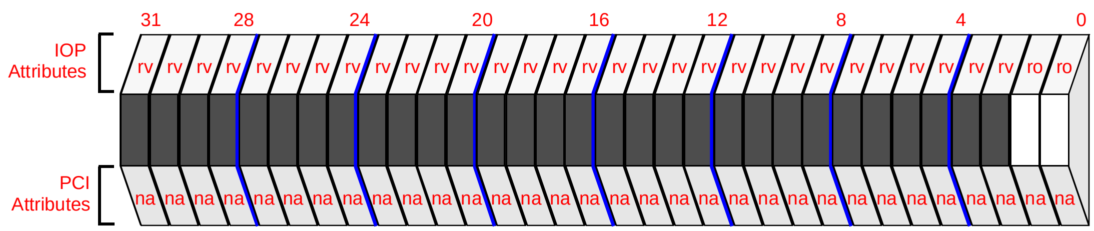
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:08	000000H	Reserved
07:00	00H	I²C **Data Buffer**: Buffer for I²C bus send/receive data.

14.8.5 I^2C Bus Monitor Register x — IBMRx

The I^2C Bus Monitor Register (IBMRx) tracks the status of the **SCL** and **SDA** pins. The values of these pins are recorded in this read-only register so that software may determine if the I^2C bus is hung and the I^2C unit must be reset.

Table 475. I^2C Bus Monitor Register x — IBMRx



Unit #	Intel XScale® processor internal bus address offset
0	+2514H
1	+2534H
2	+2554H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

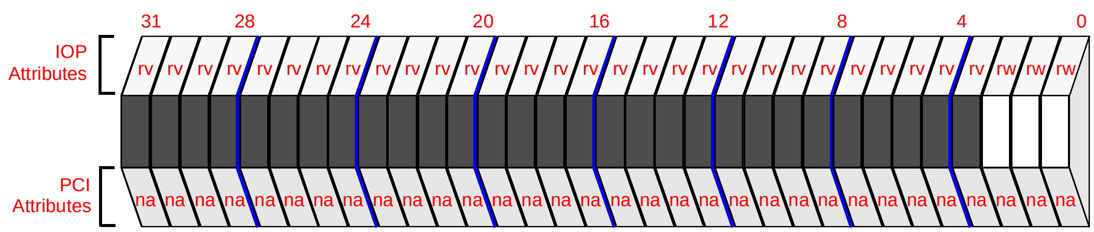
Bit	Default	Description
31:02	0	Reserved
01	1	**SCL** Status: This bit continuously reflects the value of the **SCL** pin.
00	1	**SDA** Status: This bit continuously reflects the value of the **SDA** pin.

14.8.6 I²C Manual Bus Control Register x — IMBCRx

The I²C Manual Bus Control Register (IMBCRx) can be used to manually release or pull down the **SCL** and **SDA** pins. The values of these pins are controlled by the IMBCRx bits 2:1 when bit 0 of the IMBCRx is set. When software determines that the I²C bus is hung using the "I2C Bus Monitor Register x — IBMRx" on page 721, this register may be used to force the I²C bus out of the hung state.

Note: When the I²C bus is hung, the I²C unit should also be reset using bit 14 of the "I2C Control Register x — ICRx" on page 715.

Table 476. I²C Manual Bus Control Register x — IMBCRx



Unit #	Intel XScale® processor internal bus address offset
0	+2518H
1	+2538H
2	+2558H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:03	0	Reserved
02	0	**SDA** Control: When bit 0 of the IMBCRx is set, this bit controls the **SDA** pin. 0 = Pull Down the **SDA** pin 1 = Do Not Pull Down the **SDA** pin
01	0	**SCL** Control: When bit 0 of the IMBCRx is set, this bit controls the **SCL** pin. 0 = Pull Down the **SCL** pin 1 = Do Not Pull Down the **SCL** pin
00	0	Manual I²C Pin Control Enable: When set, the **SCL** and **SDA** pins are controlled by bits 1 and 2 of the IMBCRx register. Otherwise, these pins are controlled by the I²C unit's internal state machine.

15.0 General Purpose I/O Unit

Note: For TPER mode the register interface defined here is used. For 4138xx non-TPER mode, see the SAS/SATA Command Summary for API to control the GPIO units. Some limitations apply when controlling via the API.

This chapter describes the Intel® 413808 and 413812 I/O Controllers in TPER Mode (4138xx) General Purpose I/O Unit; operation modes, setup, external memory interface, and implementation of General Purpose I/Os (GPIOs).

15.1 General Purpose Input Output Support

Twelve pins are provided as General Purpose Input Output (GPIO) pins. The Twelve pins are **GPIO[15:0]**. These pins can be used by the Intel XScale® processors to control or monitor external devices in the I/O subsystem.

15.1.1 General Purpose Inputs

The current state of the twelve GPIO pins can be read in **HPI#**.

Note: When configured as GPIOs, the twelve GPIO pins can be used as (up to) 12 additional external interrupt inputs dedicated to the Intel XScale® processor. This feature is available on a per pin basis simply by programming the INTCTL[3:0] registers.

15.1.2 General Purpose Outputs

The output function of the GPIO pins is controlled by two registers, as stated in Section 15.2.3, "GPIO Output Data Register — GPOD" on page 728) and Section 15.2.1, "GPIO Output Enable Register — GPOE" on page 725).

The output enables are mapped on a per bit basis to each of the data bits in the GPIO Output Data Register. When a bit of the GPIO Output Enable Register is cleared, the corresponding data bit value in the GPIO Output Data Register is actively driven on the appropriate GPIO pin.

15.1.3 Reset Initialization of General Purpose I/O Function

- GPIO Input Data Register is initialized to the state of **GPIO[15:0]** bus upon assertion of **P_RST#**. Note that **GPIO[3:0]** pins are multiplexed with the PCI-X interrupts **P_INT[D:A]#** and, therefore, are initialized as output pins when the PCI-X interface is used as an endpoint.
- GPIO Output Data Register is initialized to all zeros upon assertion of **P_RST#**.
- GPIO Output Enable Register is initialized to FFFFH upon assertion of **P_RST#**. This means that **GPIO[15:0]** are initialize as inputs.
- **GPIO[15:0]** pins are tristated during **P_RST#** assertion.

15.2 Register Definitions

All GPIO are visible as 4138xx memory mapped registers and can be accessed through the internal memory bus. Each is a 32-bit register and is memory-mapped in the Intel XScale® processor memory space.

The programmer interface to the General Purpose I/O is through memory-mapped control registers. Table 477 describes these registers.

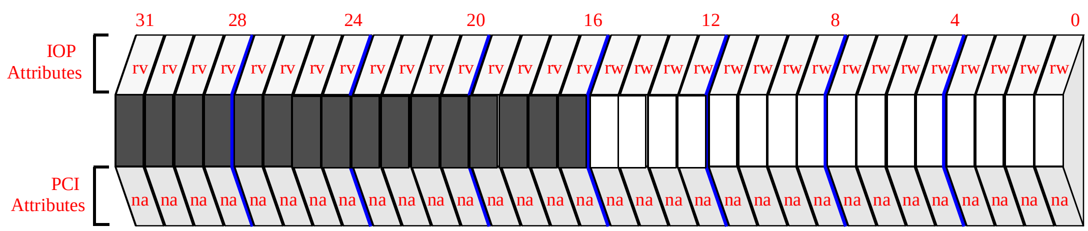
Table 477. General Purpose I/O Registers Addresses

Register Name	Description	MMR Address Offset
GPOE	GPIO Output Enable Register	2480H
GPID	GPIO Input Data Register	2484H
GPOD	GPIO Output Data Register	2488H

15.2.1 GPIO Output Enable Register — GPOE

The GPIO Output Enable Register, on a per pin basis, enables the output value contained in the GPIO Output Data Register, onto the appropriate pin. The GPIO Output Enable Register is initialized to FFFFH such that all of **GPIO[15:0]** are inputs. In order to enable a particular GPIO pin to operate as an output following the deassertion of **P_RST#**, the user needs to write a 0 into the appropriate GPOE bit.

Table 478. GPIO Output Enable Register — GPOE



Intel XScale® processor Local Bus Address offset
+2480H

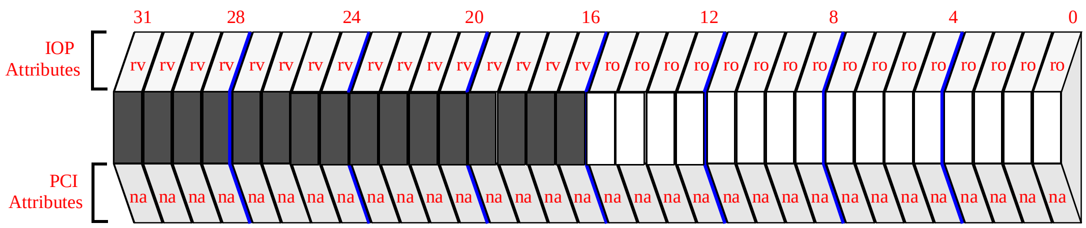
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:16	0000H	Reserved.
15	1_2	GPIO15 Output Enable — When clear, bit[15] of the GPIO Output Data Register is enabled onto the **GPIO[15]** pin.
14	1_2	GPIO14 Output Enable — When clear, bit[14] of the GPIO Output Data Register is enabled onto the **GPIO[14]** pin.
13	1_2	GPIO13 Output Enable — When clear, bit[13] of the GPIO Output Data Register is enabled onto the **GPIO[13]** pin.
12	1_2	GPIO12 Output Enable — When clear, bit[12] of the GPIO Output Data Register is enabled onto the **GPIO[12]** pin.
11	1_2	GPIO11 Output Enable — When clear, bit[11] of the GPIO Output Data Register is enabled onto the **GPIO[11]** pin.
10	1_2	GPIO10 Output Enable — When clear, bit[10] of the GPIO Output Data Register is enabled onto the **GPIO[10]** pin.
09	1_2	GPIO9 Output Enable — When clear, bit[9] of the GPIO Output Data Register is enabled onto the **GPIO[9]** pin.
08	1_2	GPIO8 Output Enable — When clear, bit[8] of the GPIO Output Data Register is enabled onto the **GPIO[8]** pin.
07	1_2	GPIO7 Output Enable — When clear, bit[7] of the GPIO Output Data Register is enabled onto the **GPIO[7]** pin.
06	1_2	GPIO6 Output Enable — When clear, bit[6] of the GPIO Output Data Register is enabled onto the **GPIO[6]** pin.
05	1_2	GPIO5 Output Enable — When clear, bit[5] of the GPIO Output Data Register is enabled onto the **GPIO[5]** pin.
04	1_2	GPIO4 Output Enable — When clear, bit[4] of the GPIO Output Data Register is enabled onto the **GPIO[4]** pin.
03	1_2	GPIO3 Output Enable — When clear, bit[3] of the GPIO Output Data Register is enabled onto the **GPIO[3]** pin.
02	1_2	GPIO2 Output Enable — When clear, bit[2] of the GPIO Output Data Register is enabled onto the **GPIO[2]** pin.
01	1_2	GPIO1 Output Enable — When clear, bit[1] of the GPIO Output Data Register is enabled onto the **GPIO[1]** pin.
00	1_2	GPIO0 Output Enable — When clear, bit[0] of the GPIO Output Data Register is enabled onto the **GPIO[0]** pin.

15.2.2 GPIO Input Data Register — GPID

The GPIO Input Data Register reflects the state of the appropriate **GPIO** bus pins following the deassertion of **P_RST#**.

Table 479. GPIO Input Data Register — GPID (Sheet 1 of 2)

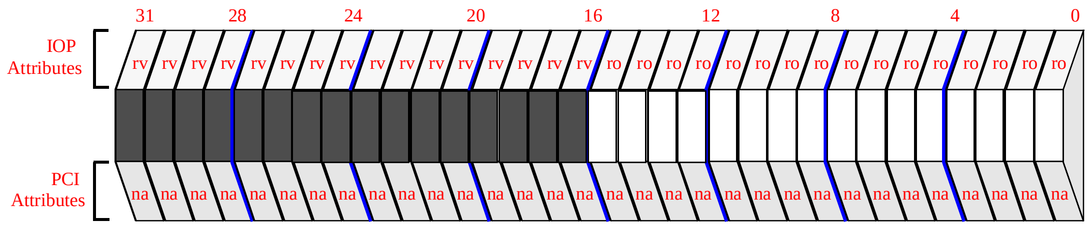


Intel XScale® processor Local Bus Address offset
+2484H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:16	0000H	Reserved
15	**GPIO[15]** during **P_RST#** assertion	GPIO15 Input Data — This bit reflects the state of the **GPIO[15]** pin.
14	**GPIO[14]** during **P_RST#** assertion	GPIO14 Input Data — This bit reflects the state of the **GPIO[14]** pin.
13	**GPIO[13]** during **P_RST#** assertion	GPIO13 Input Data — This bit reflects the state of the **GPIO[13]** pin.
12	**GPIO[12]** during **P_RST#** assertion	GPIO12 Input Data — This bit reflects the state of the **GPIO[12]** pin.
11	**GPIO[11]** during **P_RST#** assertion	GPIO11 Input Data — This bit reflects the state of the **GPIO[11]** pin.
10	**GPIO[10]** during **P_RST#** assertion	GPIO10 Input Data — This bit reflects the state of the **GPIO[10]** pin.
09	**GPIO[9]** during **P_RST#** assertion	GPIO9 Input Data — This bit reflects the state of the **GPIO[9]** pin.
08	**GPIO[8]** during **P_RST#** assertion	GPIO8 Input Data — This bit reflects the state of the **GPIO[8]** pin.
07	**GPIO[7]** during **P_RST#** assertion	GPIO7 Input Data — This bit reflects the state of the **GPIO[7]** pin.
06	**GPIO[6]** during **P_RST#** assertion	GPIO6 Input Data — This bit reflects the state of the **GPIO[6]** pin.

Table 479. GPIO Input Data Register — GPID (Sheet 2 of 2)



Intel XScale® processor Local Bus Address offset
+2484H

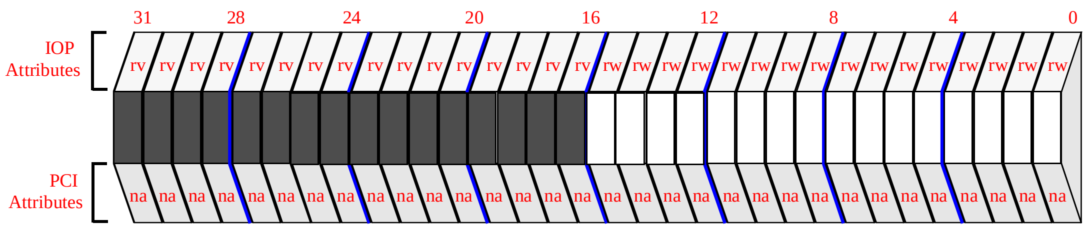
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
05	**GPIO[5]** during **P_RST#** assertion	GPIO5 Input Data — This bit reflects the state of the **GPIO[5]** pin.
04	**GPIO[4]** during **P_RST#** assertion	GPIO4 Input Data — This bit reflects the state of the **GPIO[4]** pin.
03	**GPIO[3]** during **P_RST#** assertion	GPIO3 Input Data — This bit reflects the state of the **GPIO[3]** pin.
02	**GPIO[2]** during **P_RST#** assertion	GPIO2 Input Data — This bit reflects the state of the **GPIO[2]** pin.
01	**GPIO[1]** during **P_RST#** assertion	GPIO1 Input Data — This bit reflects the state of the **GPIO[1]** pin.
00	**GPIO[0]** during **P_RST#** assertion	GPIO0 Input Data — This bit reflects the state of the **GPIO[0]** pin.

15.2.3 GPIO Output Data Register — GPOD

The GPIO Output Data Register is driven on a per bit basis on the appropriate **GPIO** bus pins following the deassertion of **P_RST#** when the corresponding bit in the GPOE register is cleared.

Table 480. GPIO Output Data Register — GPOD



Intel XScale® processor Local Bus Address offset +2488H

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:16	0000H	Reserved.
15	0_2	GPIO15 Output Data — This bit value is driven on the **GPIO[15]** pin when bit[15] of the GPOE register is cleared.
14	0_2	GPIO14 Output Data — This bit value is driven on the **GPIO[14]** pin when bit[14] of the GPOE register is cleared.
13	0_2	GPIO13 Output Data — This bit value is driven on the **GPIO[13]** pin when bit[13] of the GPOE register is cleared.
12	0_2	GPIO12 Output Data — This bit value is driven on the **GPIO[12]** pin when bit[12] of the GPOE register is cleared.
11	0_2	GPIO11 Output Data — This bit value is driven on the **GPIO[11]** pin when bit[11] of the GPOE register is cleared.
10	0_2	GPIO10 Output Data — This bit value is driven on the **GPIO[10]** pin when bit[10] of the GPOE register is cleared.
09	0_2	GPIO9 Output Data — This bit value is driven on the **GPIO[9]** pin when bit[9] of the GPOE register is cleared.
08	0_2	GPIO8 Output Data — This bit value is driven on the **GPIO[8]** pin when bit[8] of the GPOE register is cleared.
07	0_2	GPIO7 Output Data — This bit value is driven on the **GPIO[7]** pin when bit[7] of the GPOE register is cleared.
06	0_2	GPIO6 Output Data — This bit value is driven on the **GPIO[6]** pin when bit[6] of the GPOE register is cleared.
05	0_2	GPIO5 Output Data — This bit value is driven on the **GPIO[5]** pin when bit[5] of the GPOE register is cleared.
04	0_2	GPIO4 Output Data — This bit value is driven on the **GPIO[4]** pin when bit[4] of the GPOE register is cleared.
03	0_2	GPIO3 Output Data — This bit value is driven on the **GPIO[3]** pin when bit[3] of the GPOE register is cleared.
02	0_2	GPIO2 Output Data — This bit value is driven on the **GPIO[2]** pin when bit[2] of the GPOE register is cleared.
01	0_2	GPIO1 Output Data — This bit value is driven on the **GPIO[1]** pin when bit[1] of the GPOE register is cleared.
00	0_2	GPIO0 Output Data — This bit value is driven on the **GPIO[0]** pin when bit[0] of the GPOE register is cleared.

16.0 PMON Unit

16.1 PMON Counters

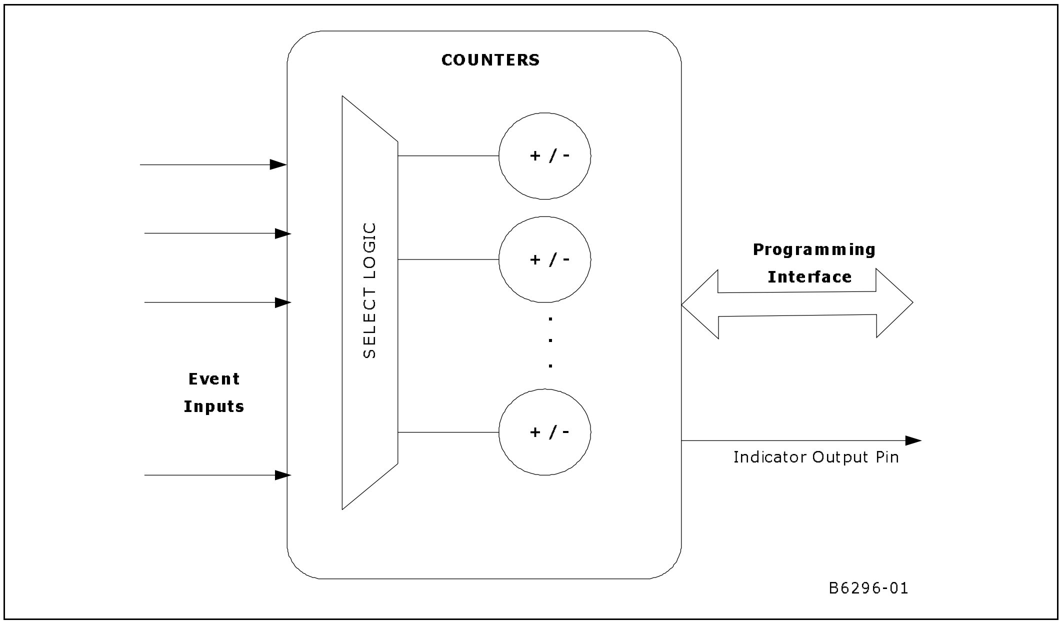
The Performance Monitoring (**PMON**) counters enable performance monitoring and gathering statistics of internal hardware events in real-time. This implementation provides users with direct event counting and timing for performance monitoring and system debugging purposes. It provides enough visibility into the internal architecture to perform utilization studies, workload characterization, and application tuning.

16.2 Overview

At the heart of the **PMON** functionality are counters with associated registers. Each counter has a corresponding command, event, status, and data register. The **PMON** unit implements eight counters.

Signals representing events from throughout the chip are routed to the **PMON** unit. Software can select which events are recorded during a measurement session. The starting, stopping, and sampling of the counters can be controlled by either software or hardware. This can be done in a time-based (polling) or event-based fashion. Each counter can be incremented or decremented by different events. In addition to simple counting of events the unit can provide data for histograms, queue analysis, and conditional event counting (example: How many times did event A happen before the first event B took place).

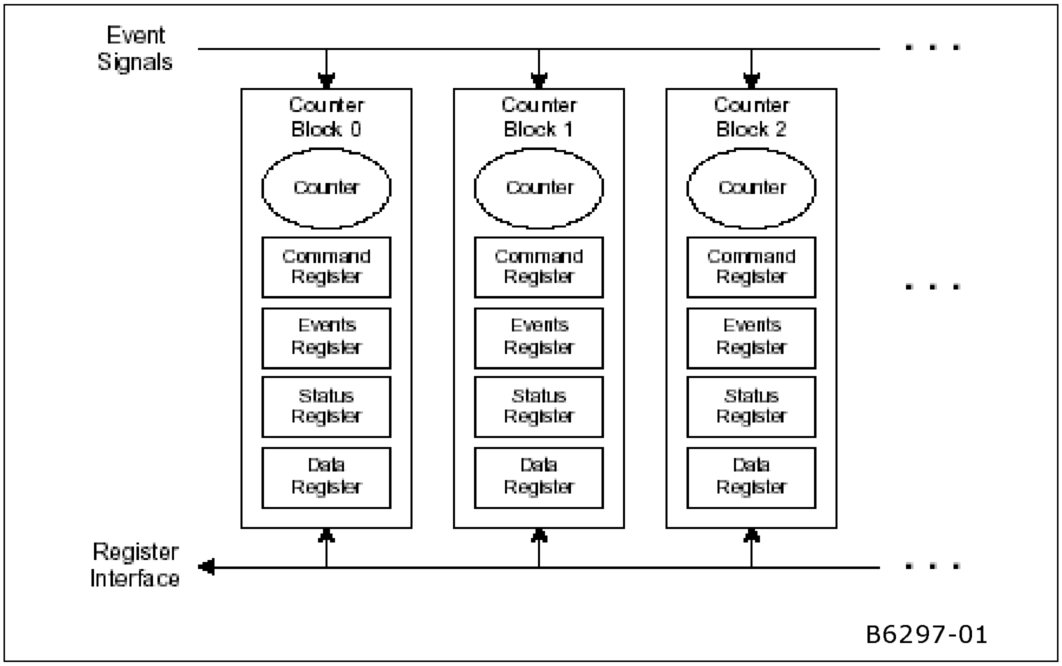
Figure 106. Example Block Diagram of Component with Counter



16.2.1 Clock Counter Control

When a counter is sampled, the current value of the counter is latched into the corresponding data register. The command, event, status, and data registers are accessible via memory mapped registers in order to facilitate high-speed sampling.

Figure 107. Conceptual Diagram of Counter Array



16.3 Definitions

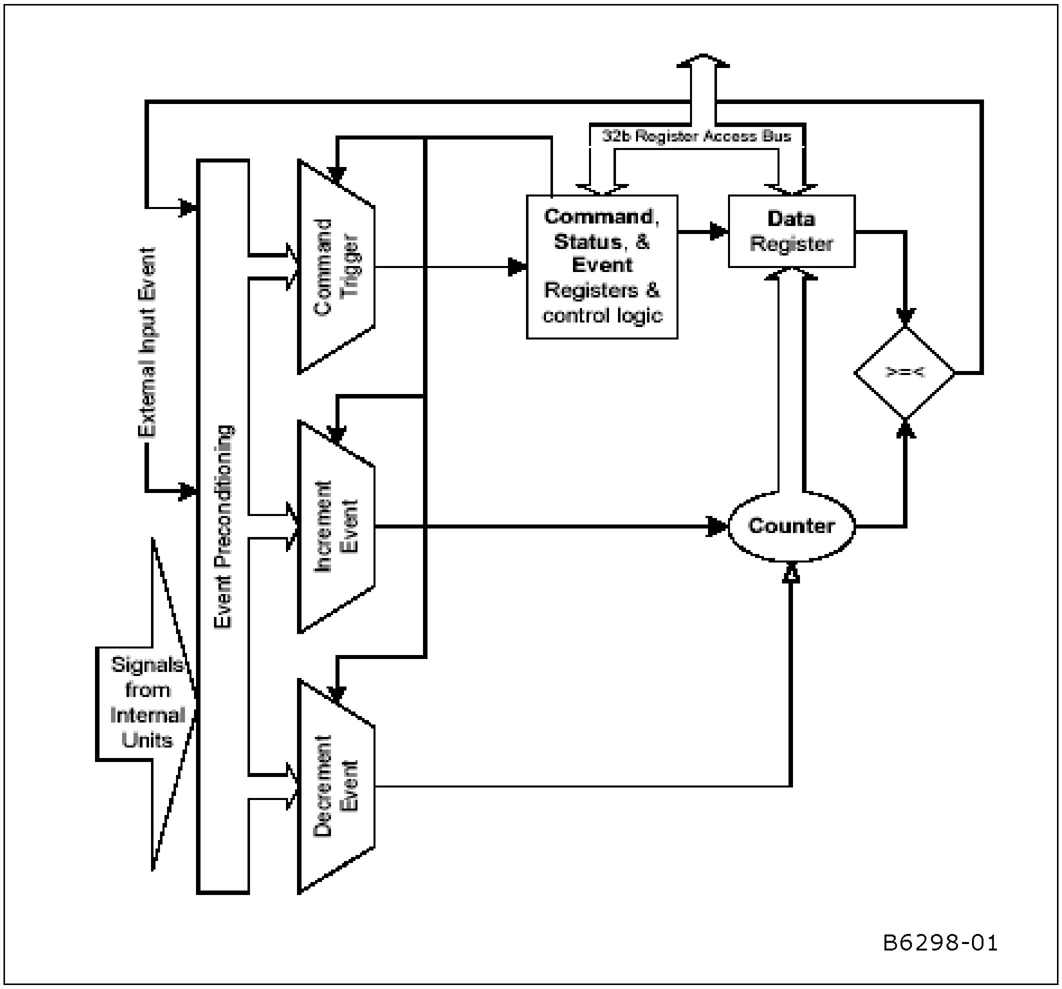
- Duration Count - The counter is incremented for each clock for which the event signal is asserted logic high.
- Occurrence Count - The counter is incremented each time a rising edge of the event signal is detected.
- Preconditioning - Altering a signal that represents an event before it is presented to be counted by the **PMON** unit. This includes clock crossing logic.

Two optional external pins allow for external visibility and control of the counters. The output pin signals that one of the following conditions generated an interrupt from any one of the counters:

- a programmable threshold condition was true,
- a command was triggered to begin
- a counter overflow or underflow occurred.

The figure below represents a single counter block. The muxes, registers, and all other logic is repeated for each counter that is present. There is a threshold event from each counter block that feeds into each mux.

Figure 108. Example Block Diagram of Single PMON Counter



16.4 Data Collection

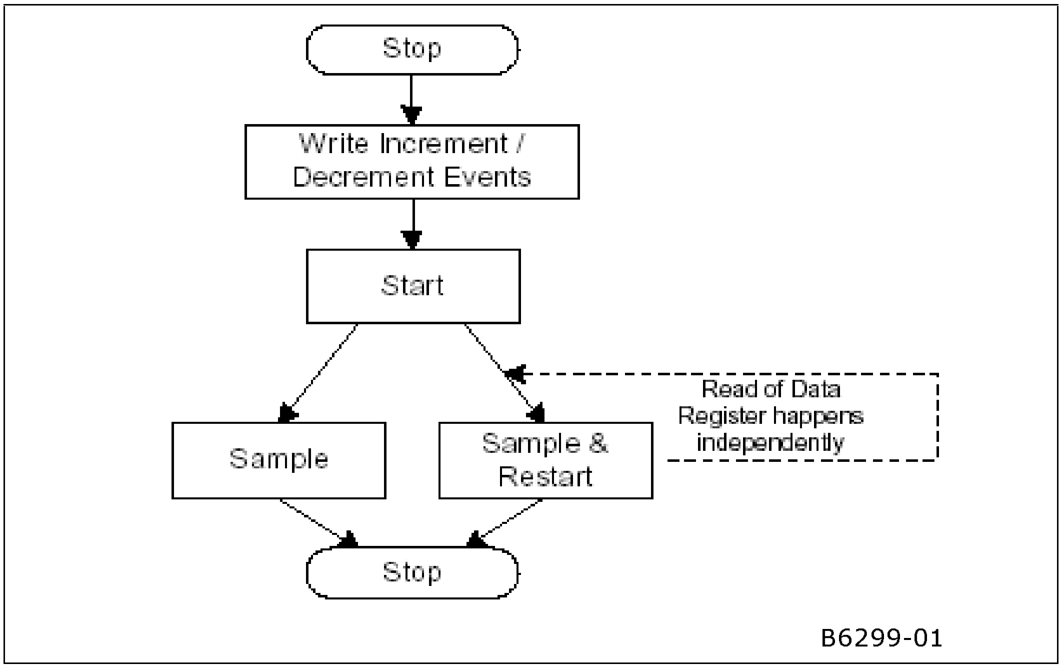
The following sections provide some insight into the intended use of the **PMON** counters at a very low level. The examples and accompanying explanation should prove especially valuable in the creation of test cases for both hardware and software.

The hardware face to the **PMON** counters is not intended to be any wider than 32 bits. This means that all registers are accessed one at a time.

All of the following examples assume starting with an idle system (all counters stopped and all registers set to default values). The waveforms in the examples do not explicitly always show when each **PMON** command was executed, but the reader should be able to deduce when each command would have or could have executed. For the purposes of these examples, the waveforms treat the counters as being level sensitive to the appropriate events.

16.4.1 Time Based Sampling

Figure 109. Flowchart of Example Commands Sequence



Example 7 has been simplified by using 12 clocks as the sampling period. In a real system the sampling would more likely be something like 1 ms. There is a certain amount of overhead associated with writing and reading to any **PMON** registers. The more frequent the interaction between the **PMON** counters and any software, the larger the margin of error that is injected into the final results.

The sampling period can be controlled by a CPU counter or another **PMON** counter periodically interrupting the system and allowing the software to read data registers or do whatever else may be desired.

Example 7. Simple Counting

This example demonstrates how to measure the number of times event A occurs during 12 clocks.

Table 481. Simple Time Based Counting of Events Example

Opcode	Target Counter	Increment Event	Decrement Event	Trigger Event
Write Event Register	0	Event A	None (000h)	
Start	0			Immed (000h)
* Sample & Restart	0			Immed (000h)
Read Data Register	0			
-- Wait 12 clocks before returning to * --				

An alternative way to represent the data in the preceding table is as follow:

Write Event Register 0 (Increment = Event A)

Start Counter 0 immediately

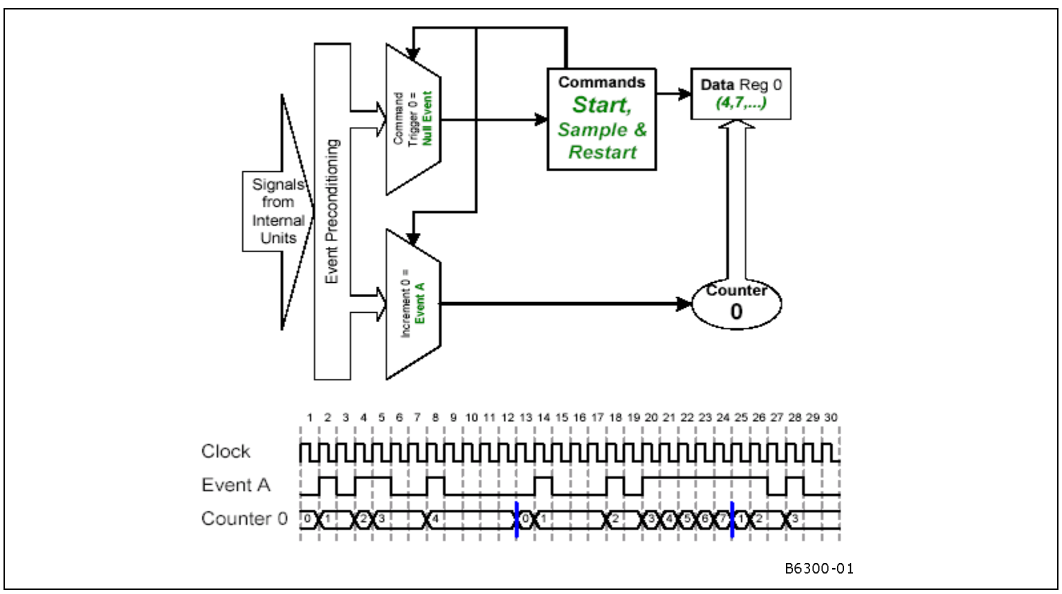
Repeat every 12 clocks

Sample & Restart Counter 0 (Threshold Condition Code is NA)

Read Data Register 0

End Repeat

Figure 110. Block Diagram and Waveforms of Time Based Sampling Example



16.4.2 Hardware Event Based Control

Hardware event based control allows a hardware event to control when another command is executed. An example of this is controlling when a sample (snapshot) is taken of the active counter(s). This is required to facilitate, among other things, hardware **data queue analysis**. No command is executed until the *command trigger mux* detects the event in the command trigger field of the command register. This allows start, stop, sample, and other commands to be executed as a result of other events happening.

Command Triggers refers to the ability of a command to be issued to the **PMON** unit, and have it not be executed until the desired event, as programmed when the pending command was issued, is detected.

The **PMON** unit has no ability to queue commands so programming must be written so as not to overwrite pending commands unless that is the desired effect.

For example, one could count how many memory reads happen before the first cache hit occurs. This would be accomplished by programming the desired **PMON** counters to start counting memory read events (Event A in Example 8 on page 735). Immediately following that command the **PMON** controller is programmed to stop counting the memory read events.

However, execution of this command is conditional upon a cache hit occurring (Event B in Example 8). Thus, the cache hit command becomes the “Command Trigger” that must occur before the memory read events would no longer be counted.

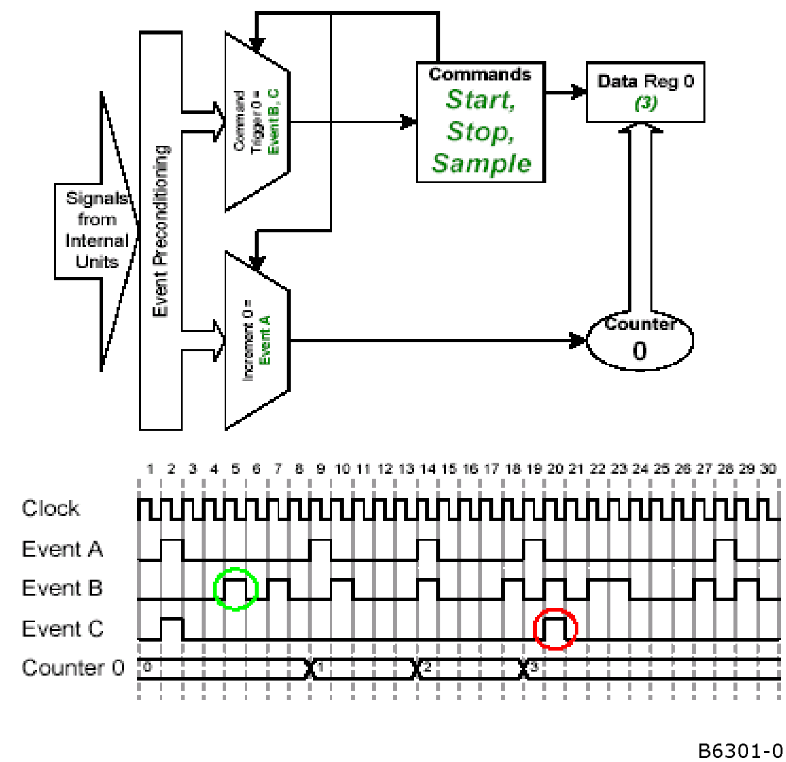
Example 8. How many Event A’s happen before the first Event B is detected

This example demonstrates how to measure the number of times event A occurs before the first occurrence of event B.

Table 482. Hardware Event Based Event Counting Example

Opcode	Target Counter	Increment Event	Decrement Event	Trigger Event
Write Event Register	0	Event A	None (000h)	
Start	0			Immed (000h)
Stop	0			Event B
Sample	0			Immed (000h)
Read Data Register	0			

Figure 111. Block Diagram and Waveforms of Time Based Sampling Example

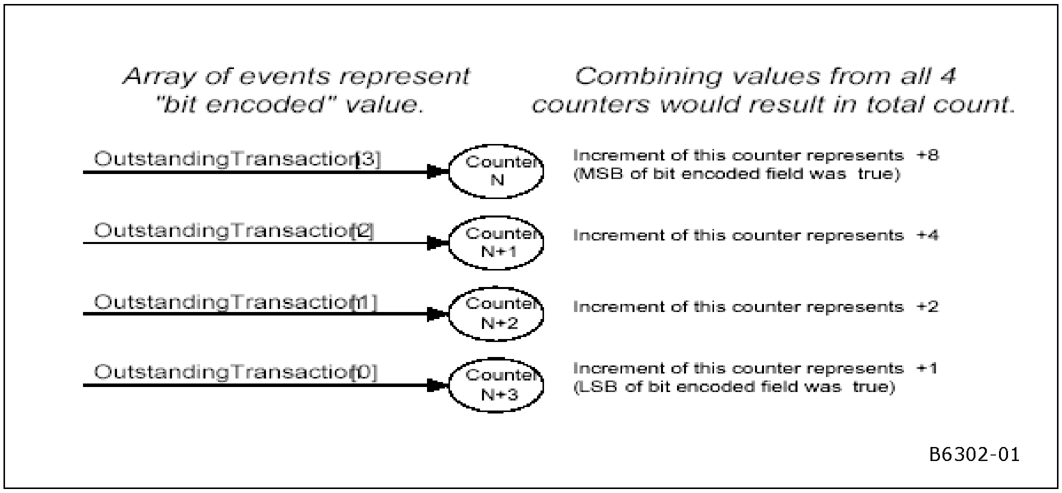


16.4.3 Incrementing By More Than 1

For scenarios where it is desired to increment a count by more than one on a single clock tick, the **PMON** unit can be sent a "bit encoded" value. This allows the counters to track an increment/decrement of up to 2^{N-1}each clock tick, where N is the number of counters available. For example, with 8 counters we could track an increment/decrement of up to 128 (2^7) each clock.

This would be done by assigning an array of events to represent a particular value to count. The counters would be coordinated to mimic a single counter which by asserting the appropriate event signals cause the set of counters to be incremented and/or decremented by the in such a way that post processing of the counter values.

Figure 112. Block Diagram & Waveforms of Time Based Sampling Example



16.4.4 Queue Analysis

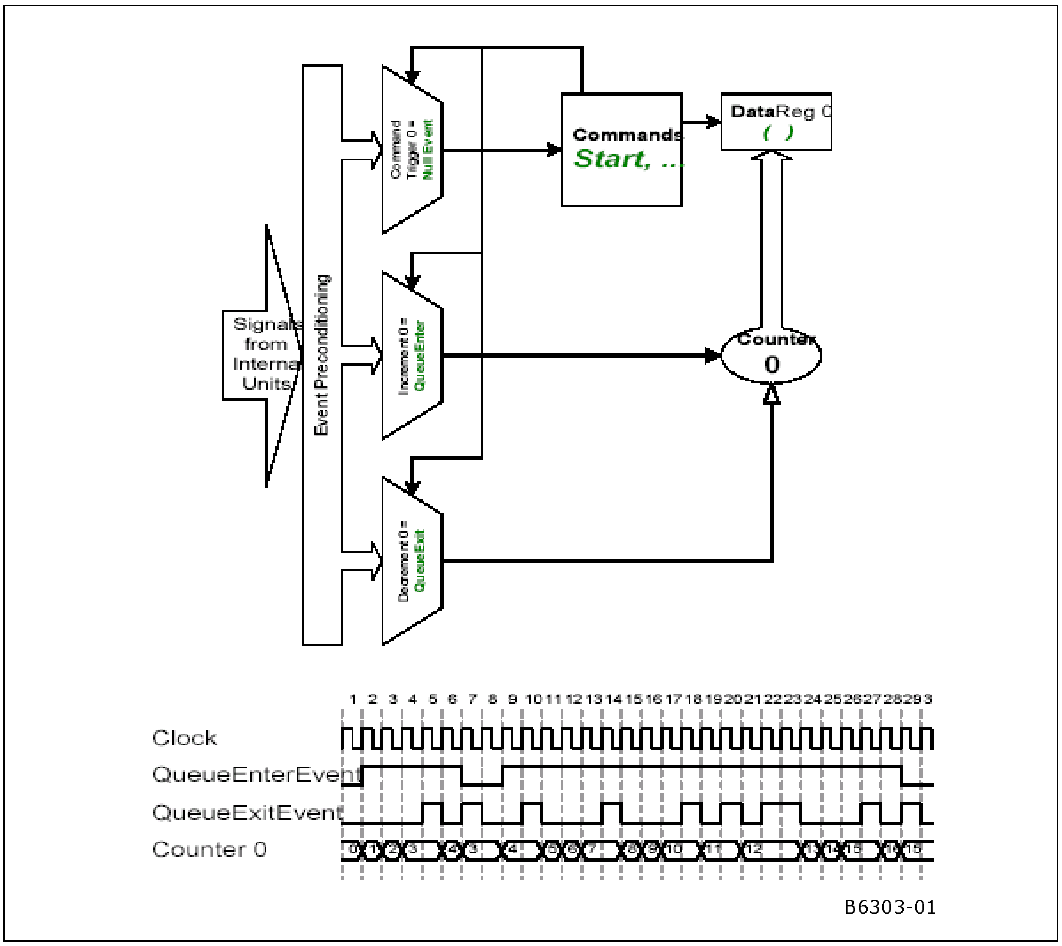
Example 9. Queue Depth Example

This example demonstrates how to measure the current fill level (depth) of a particular queue.

Table 483. Hardware Event Based Event Counting Example

Opcode	Target Counter	Increment Event	Decrement Event	Trigger Event
Write Event Register	0	Queue enter	Queue Exit	
Start (CC is =)	0			Immed (000h)
Sample	0			Immed (000h)
Read Data Register	0			

Figure 113. Block Diagram & Waveforms of Time Based Sampling Example



Gathering queue depth information, as in Example 9, is probably more useful when it is presented in a histogram. Example 10 on page 738 outlines the command sequences required to generate such a histogram. See the Head of Queue Histogram example in another section for more details on how to create histograms.

Example 10. Queue Depth Histogram Example

This example demonstrates how to measure the current fill level (depth) of a particular queue.

Table 484. Queue Depth Histogram Example

Opcode	Target Counter	Increment Event	Decrement Event	Trigger Event
Write Threshold (bucket size) in Data Register	0			
Write Event Register	0	Queue Enter	Queue Exit	
Write Event Register	1	Threshold Event 0		
Start (CC is =)	0			Queue Empty
Start	1			Immed (000h)
-- Run Workload --				
Sample	1			Immed (000h)
Read Data Register	0			
-- Repeat entire sequence with new bucket size --				

No block diagram for this example.

Example 11. Head of Queue Histogram Example

A histogram can be created using two counters working together. The following applies to all of the histogram examples:

- Start with idle system (reset counters).
- Only one slice of a histogram is measured per experiment, therefore histograms can only be generated for repeatable workloads.
- Repeat these steps, changing threshold value and using an unchanging repeatable workload to generate data for histogram.

This example demonstrates how often has a pre-amount of time elapsed when a certain event takes place.

Table 485. Head of Queue Histogram Example

Opcode	Target Counter	Increment Event	Decrement Event	Trigger Event
Write Threshold (bucket size) in Data Register	0			
Write Event Register	0	Queue Not Empty		
Write Event Register	1	Threshold Event 0		
Start	0			Queue Empty
Restart (CC is =)	0			Queue Exit
Start	1			Immed (000h)
-- Run Workload --				
Sample	1			Immed (000h)
Read Data Register	0			
-- Repeat entire sequence with new bucket size --				

An alternative way to represent the data in the preceding table is as follow:

```
For X=1 to 3
        Write Threshold (bucket size = X) into Data Register 0
        Write Event Register 0 (Increment = QueueNotEmpty)
        Write Event Register 1 (Increment = Threshold Event 0)
        Start Counter 0 immediately
        Restart Counter 0 (with Threshold Condition Code set to =)
                whenever QueueExitEvent triggers it
        Start Counter 1 immediately
        Run workload
        Sample Counter 1
        Read Data Register 1
End For
```

Figure 114. Block Diagram of HOQ Histogram Example

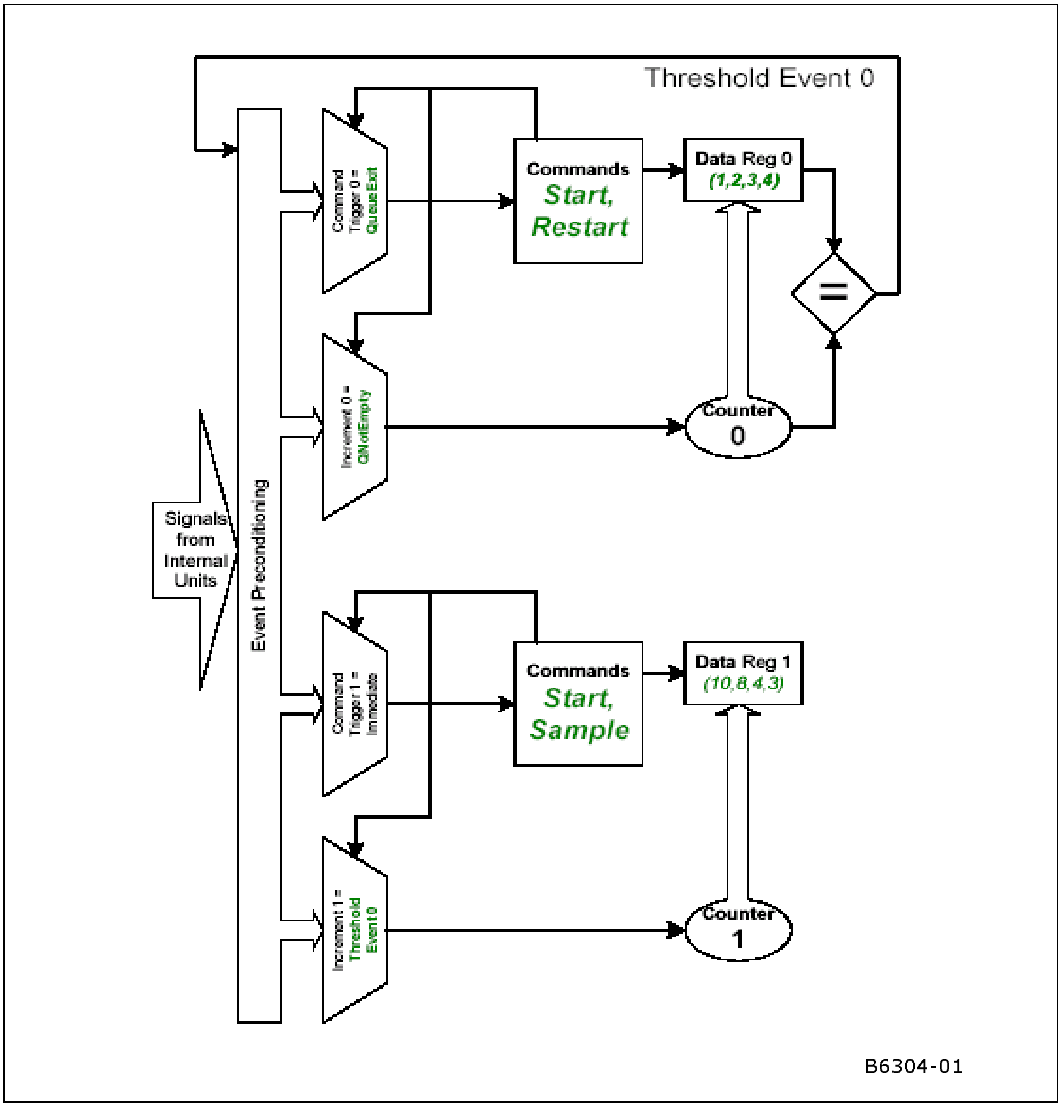
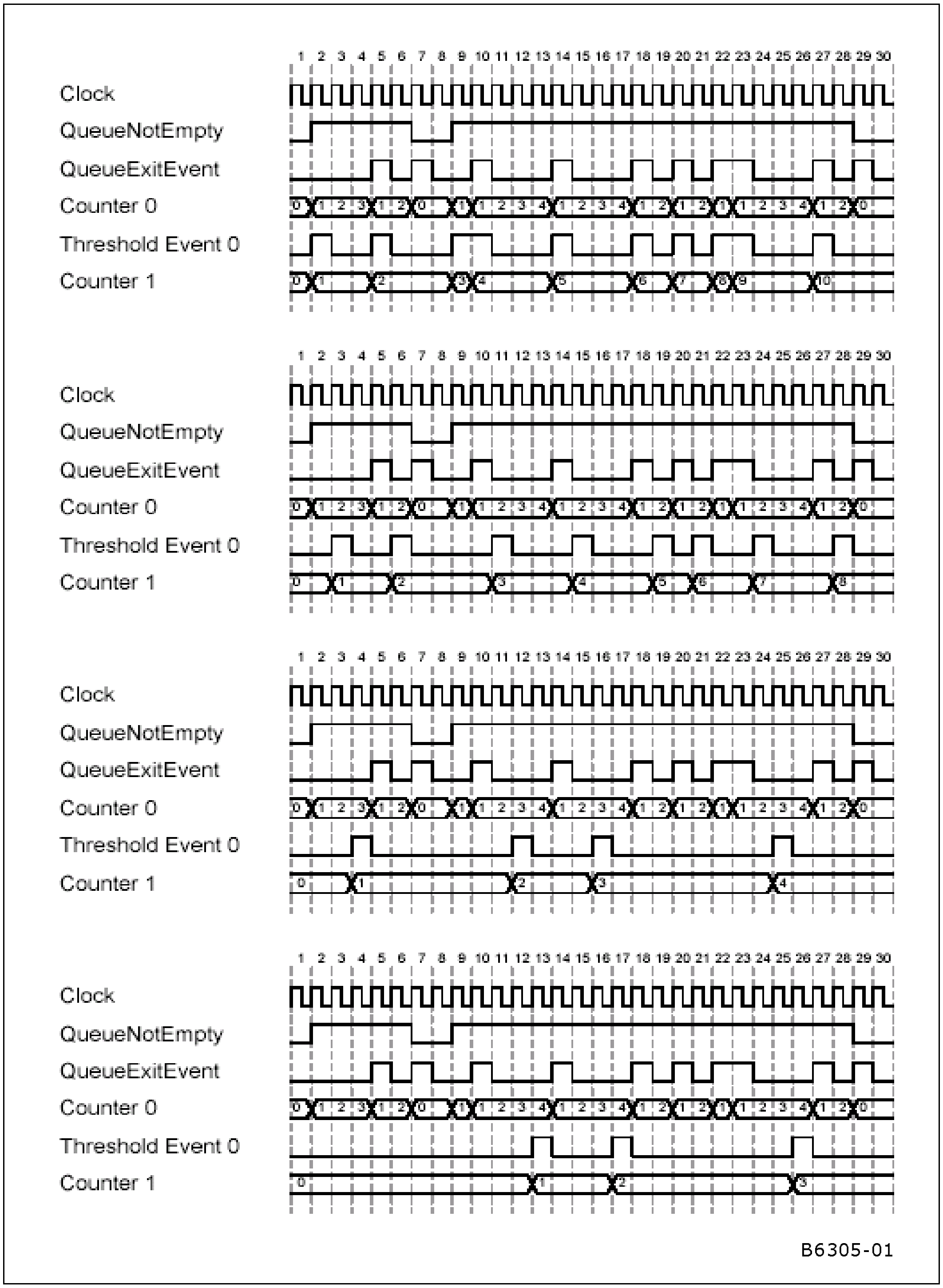


Figure 115. Waveforms of HOQ Histogram Example



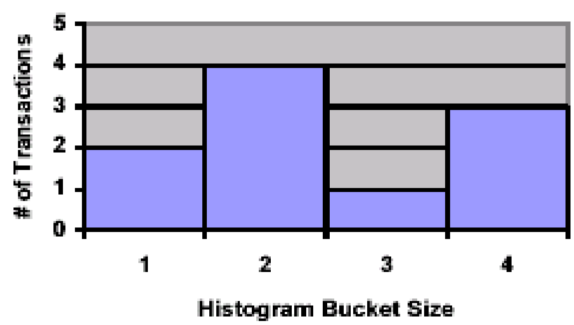
In each of the 4 iterations above, the exact same repeatable workload produces identical queue behavior, but the Counter 1 value (called Condition Code Matches below) differs due to having a different threshold value each time. The condition code match values (10, 8, 4, 3) from the 4 iterations can be used to compute the values of a histogram as shown in the following two figures.

Figure 116. Processing of HOQ Histogram Example

Threshold Value (Bucket Size)	Condition Code Matches		Cumulative Transactions		# of Transactions
4:	3	-	0	=	3
3:	4	-	3	=	1
2:	8	-	4	=	4
1:	10	-	8	=	2

B6318-01

Figure 117. Output from HOQ Histogram Example



B6319-01

16.5 Non-Register-Based Interfaces

This section describes the interfaces to the **PMON** unit that are not part of the register scheme already documented.

16.5.1 Events Input Port

Signals representing internal events are sent to the event preconditioning block where they are conditioned when required. The most common preconditioning is likely to be clock synchronization. No programmable Boolean operations can be performed on these events, but appropriate Boolean operations can be performed by the hardware.

Duration type events continually assert their signal high ('1'). The event pre-conditioning block 'ANDs' the duration type signal with a clock to produce the correct count.

Any events from different frequency domains must be preconditioned to assure count accuracy measured. For these clock domain crossing signals, 95% accuracy is sufficient over a 1 us or larger sampling window. It would be very difficult to assure accuracy of very small windows (< 1 us) without requiring a great amount of hardware to verify.

16.5.2 Output Signals

There are three potential sources of internal indicators from each counter. Each source can independently generate an indication. These are:

- a programmable threshold condition was true,
- a command was triggered to begin
- a counter overflow or underflow occurred.

Each of these conditions always sets a corresponding status bit. Indicator enable bits control which of these indicator sources drive the top level indicator. The indicators are OR'd together as shown in Figure 118, “Indicator Tree” on page 744. These internal signals are further controlled to generate either an Interrupt or an Indicator Output.

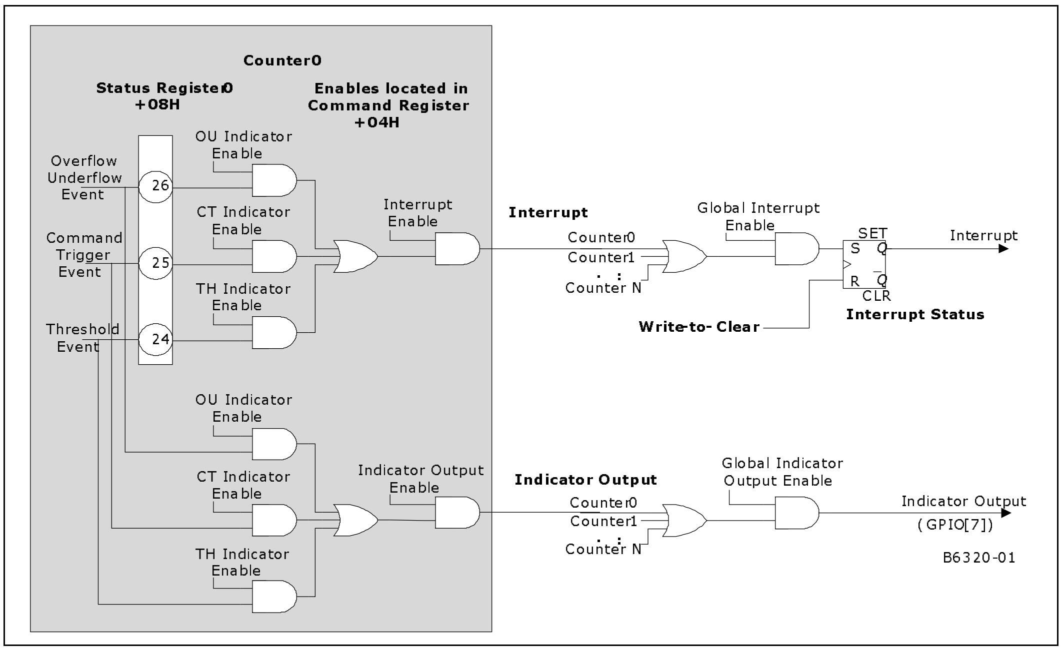
16.5.2.1 Indicator Output

The **PMON**OUT pin is a shared indicator output pin. The resulting signal allows external elements (OS, drivers, logic analyzer, etc.) to be aware of indicators without having to rely on interrupts.

For 4138xx, the **PMON**OUT function is multiplexed onto the GPIO7 pin. When the **PMON** Indicator Output Enable bit is set in the "PMON Feature Enable Register - PMONEN", the **PMON** output function overrides the GPIO7 setting in the Section 15.2.1, "GPIO Output Enable Register — GPOE" on page 725 and Section 15.2.3, "GPIO Output Data Register — GPOD" on page 728.

Warning: Since GPIO7 could be used for other purposes, care must be taken when enabling the **PMON** Indicator Output.

Figure 118. Indicator Tree



Note: Indicator and Interrupt Enables are located in the "PMON Command Register 0-7 - PMON_CMD[0:7]" on page 749
Global Indicator and Interrupt Enables are located in the "PMON Feature Enable Register - PMONEN" on page 746

16.5.2.2 Interrupt Output

An internal interrupt is delivered to the Interrupt Control Unit when:

- Interrupts are enabled in the "PMON Feature Enable Register - PMONEN" (PMONEN[0]=1).
- One or more of the interrupt sources are enabled within the individual counter via the indicator enable bits.
- The interrupt enable bit is set within the individual counter
- One or more of the enabled indicators is true

This sets the internal interrupt generated bit in the **PMON**STAT register and asserts the **PMON** interrupt input to the Interrupt control unit.

16.5.3 Internal Bus Addresses

The Internal Bus Address Offset to PMMRBAR of any **PMON** Register can be derived by adding the 4 KB address aligned Internal Bus Memory Mapped Register Range Offset (Table 486, "PMON Internal Bus Memory Mapped Register Range Offsets" on page 745) to the Register Offset (Table 487, "PMON Register Summaries" on page 745)

For example the offset to PMMRBAR of the "PMON Status Register - PMONSTAT" would be (4 E000H+044H) or 4 E044H.

Table 486. PMON Internal Bus Memory Mapped Register Range Offsets

Internal Bus MMR Address Range Offset (Relative to PMMRBAR)
+4 E000H

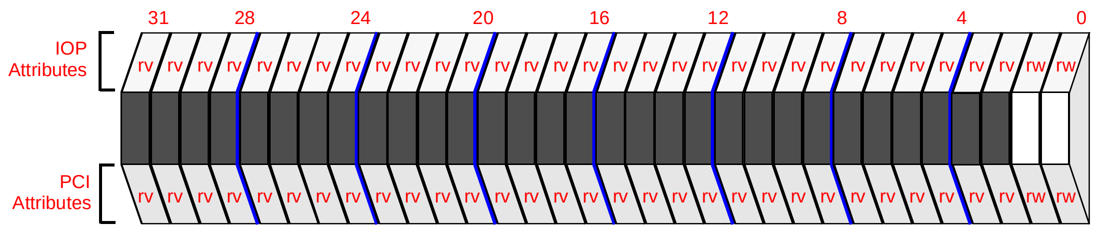
Table 487. PMON Register Summaries

Register Offset	Register Name
+040h	PMON Feature Enable Register - PMONEN
+044h	PMON Status Register - PMONSTAT

16.5.4 PMON Feature Enable Register - PMONEN

Contains control bits for **PMON** unit.

Table 488. PMON Feature Enable Register - PMONEN



Internal Bus Address Offset
+040h

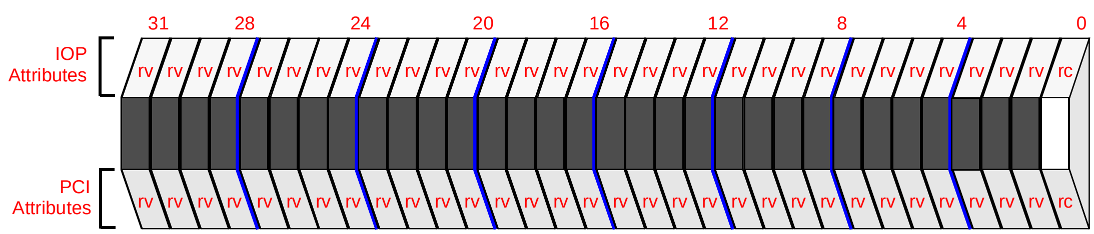
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:2	0000 0000h	Reserved
1	0b	**PMON Indicator Output Enable.** Specifies that the **PMON** unit indicator should be allowed to generate a signal on the **GPIO[7]** output pin. The conditions that can cause this signal to be toggle are documented elsewhere in this document, this bit simply provides a global mask. 0 = Do NOT generate **PMON** Indicator Output (default). 1 = DO generate **PMON** Indicator Output which goes to the optional output pin (**GPIO[7]**). ***Note:*** The **PMON** Indicator Output Enable overrides the **GPIO[7]** setting in Section 15.2.1, "GPIO Output Enable Register — GPOE" on page 725
0	0b	Interrupt Enable: Enables 4138xx **PMON** unit generated interrupts.

16.5.5 PMON Status Register - PMONSTAT

Contains status bits for **PMON** unit.

Table 489. PMON Status Register - PMONSTAT



Internal Bus Address Offset
+044h

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:1	0000 0000h	Reserved
0	0b	Interrupt Status: Indicates a pending interrupt from **PMON** unit. This bit can only be set when the Interrupt Enable bit in the **PMON**EN register is set. 0 = **PMON** Interrupt is NOT pending 1 = **PMON** Interrupt IS pending

16.5.6 PMON Memory Mapped Registers

The memory mapped registers of **PMON** unit are accessible by the Intel XScale® core.

The first set of registers provide the control of the **PMON** unit for selecting events to monitor and for data sampling. Each counter has one command, one events, one status, and one data register associated with it. These registers are numbered 0 through 7.

The location of these registers are specified as a relative offset to a 512KB aligned global PMMR offset. The default for the 512KB aligned offset is 0 FFD8 0000H defined by the PMMRBAR register. See also Chapter 19.0, “Peripheral Registers”.

The Internal Bus Address Offset to PMMRBAR of any **PMON** Register can be derived by adding the 8 KB address aligned Memory Mapped Register Range Offset (Table 490, “PMON Internal Bus Memory Mapped Register Range Offsets” on page 747) to the Register Offset (Table 491, “PMON Register Summaries” on page 748)

For example the offset to PMMRBAR of the “**PMON** Command Register 0” would be (1 A000H+004H) or 1 A004H.

Table 490. PMON Internal Bus Memory Mapped Register Range Offsets

PMON Memory Mapped Address Range Offset (Relative to PMMRBAR)
+1 A000H

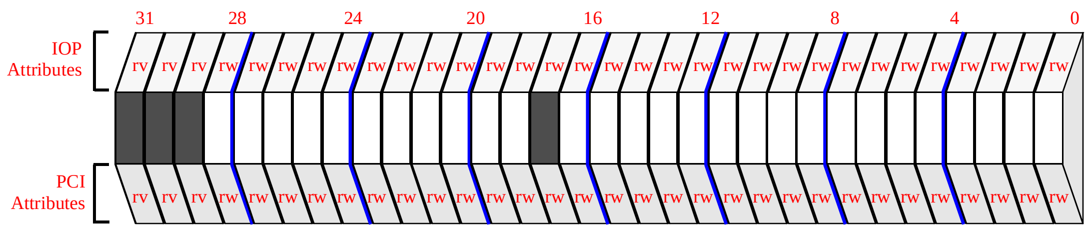
Table 491. PMON Register Summaries

Register Offset	Register Name
+000h	**PMON** Command Register 0 - **PMON**_CMD0
+004h	**PMON** Event Register 0 - **PMON**_EVR0
+008h	**PMON** Status Register 0 - **PMON**_STS0
+00Ch	**PMON** Data Register 0 - **PMON**_DATA0
+010h	**PMON** Command Register 1- **PMON**_CMD1
+014h	**PMON** Event Register 1- **PMON**_EVR1
+018h	**PMON** Status Register 1- **PMON**_STS1
+01Ch	**PMON** Data Register 1 - **PMON**_DATA1
+020h	**PMON** Command Register 2- **PMON**_CMD2
+024h	**PMON** Event Register 2- **PMON**_EVR2
+028h	**PMON** Status Register 2- **PMON**_STS2
+02Ch	**PMON** Data Register 2 - **PMON**_DATA2
+030h	**PMON** Command Register 3 - **PMON**_CMD3
+034h	**PMON** Event Register 3- **PMON**_EVR3
+038h	**PMON** Status Register 3- **PMON**_STS3
+03Ch	**PMON** Data Register 3 - **PMON**_DATA3
+040h	**PMON** Command Register 4 - **PMON**_CMD4
+044h	**PMON** Event Register 4 - **PMON**_EVR4
+048h	**PMON** Status Register 4 - **PMON**_STS4
+04Ch	**PMON** Data Register 4 - **PMON**_DATA4
+050h	**PMON** Command Register 5 - **PMON**_CMD5
+054h	**PMON** Event Register 5 - **PMON**_EVR5
+058h	**PMON** Status Register 5 - **PMON**_STS5
+05Ch	**PMON** Data Register 5 - **PMON**_DATA5
+060h	**PMON** Command Register 6 - **PMON**_CMD6
+064h	**PMON** Event Register 6 - **PMON**_EVR6
+068h	**PMON** Status Register 6- **PMON**_STS6
+06Ch	**PMON** Data Register 6 - **PMON**_DATA6
+070h	**PMON** Command Register 7 - **PMON**_CMD7
+074h	**PMON** Event Register 7- **PMON**_EVR7
+078h	**PMON** Status Register 7- **PMON**_STS7
+07Ch	**PMON** Data Register 7- **PMON**_DATA7
+080h through +FFFh	Reserved

16.5.6.1 PMON Command Register 0-7 - PMON_CMD[0:7]

This 32-bit register allows control of the **PMON** counter. When this register is written, the previous register contents are overwritten. All 32 bits must be programmed each time the register is written. When the register contained a command that was still waiting to be triggered, it would be flushed without ever being executed. The currently executing command continues to be executed only until the newly programmed command is triggered to execute.

Table 492. PMON Command Register 0-7 - PMON_CMD[0:7] (Sheet 1 of 4)



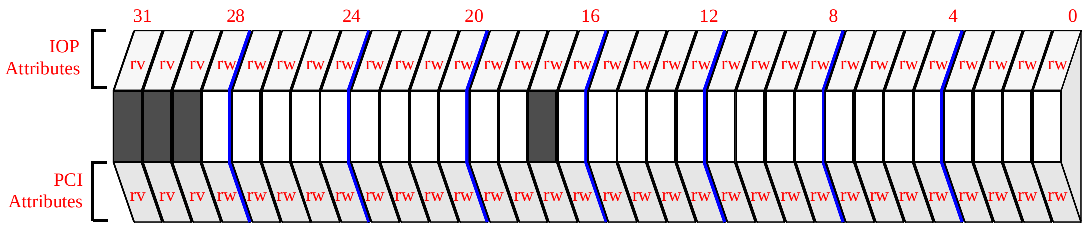
Register Offset

PMON_CMD0 +000h
PMON_CMD1 +010h
PMON_CMD2 +020h
PMON_CMD3 +030h
PMON_CMD4 +040h
PMON_CMD5 +050h
PMON_CMD6 +060h
PMON_CMD7 +070h

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
31:29	00b	Reserved
28	0b	Interrupt Enable This bit causes the counter to set the global interrupt generated bit in the **PMON**STAT register. 0 = An interrupt is not generated from this counter 1 = The global interrupt generated bit is set when one or more of the Overflow/Underflow, Command Trigger, and Threshold status bits are asserted and the individual indicator enable(s) are set. An interrupt is delivered when there is not an existing interrupt being processed from **PMON**. Upon servicing the Interrupt, the status registers should be cleared by software so that another Interrupt is not sent.
27	0b	Indicator Output Enable: 0 = **PMON**OUT is not asserted when the indicators are valid. 1 = The **PMON**OUT signal is asserted when the conditions enabled via the indicator enable bits are true.
26	0b	Overflow/Underflow Indicator Enable (OUIE) 0 = No indication provided when a counter overflow or underflow occurs except for setting the Overflow/Underflow Indicator (OUI) status bit. 1 = When the overflow/underflow condition is true, the counter sets the status bit and possibly assert the **PMON**OUT pin or enable the generation of an interrupt. This is controlled by the **PMON**OUT Enable and Interrupt Enable respectively. Interrupts are driven directly off of the status bits, whereas the **PMON**OUT pin is driven directly from the condition.
25	0b	Command Trigger Indicator Enable (CTIE) 0 = No indication provided when a command is triggered except for setting the Command Trigger Indicator (CTI) status bit. 1 = When the command trigger condition is true, the counter sets the status bit and possibly assert the **PMON**OUT pin or enable the generation of an interrupt. This is controlled by the **PMON**OUT Enable and Interrupt Enable respectively. Interrupts are driven directly off of the status bits, whereas the **PMON**OUT pin is driven directly from the condition. ***Note:*** The null event does not set the Command Trigger Indicator in the **PMON**_STS register, nor does it cause a Command Trigger that is reflected by the **PMON**OUT signal.

Table 492. PMON Command Register 0-7 - PMON_CMD[0:7] (Sheet 2 of 4)



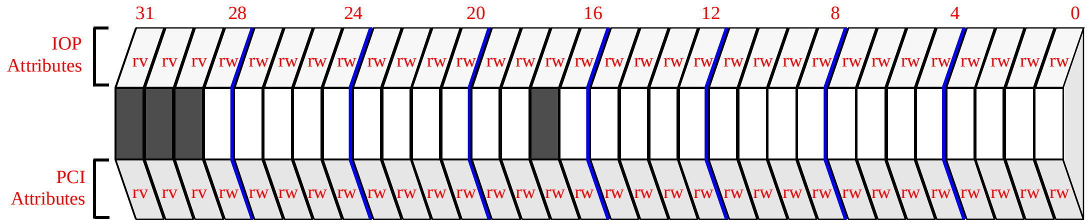
Register Offset

PMON_CMD0	+000h
PMON_CMD1	+010h
PMON_CMD2	+020h
PMON_CMD3	+030h
PMON_CMD4	+040h
PMON_CMD5	+050h
PMON_CMD6	+060h
PMON_CMD7	+070h

Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
24	0b	Threshold Indicator Enable (THIE) 0 = No indication provided when threshold condition is true except for setting the Threshold Indicator (TI) status bit. 1 = When the threshold condition is true, the counter sets the status bit and possibly assert the **PMON**OUT pin or enable the generation of an interrupt. This is controlled by the **PMON**OUT Enable and Interrupt Enable respectively. Interrupts are driven directly off of the status bits, whereas the **PMON**OUT pin is driven directly from the condition.
23:21	000b	Condition Code (CC) This field contains the code that indicates what type of threshold compare is done between the counter and the data register. For all non-0 values of this field, the counter's data register contains the threshold value. The outcome of this compare generates a threshold event and potentially an interrupt when that capability is enabled. • Bit 23 is for less than (<) • Bit 22 is for equal (=) • Bit 21 is for greater than (>) Select the proper bit mask for desired threshold condition: • 000 = False (no threshold compare) • 001 = Greater Than • 010 = Equal • 011 = Greater Than or Equal • 100 = Less Than • 101 = Not Equal • 110 = Less Than or Equal • 111 = True (always generate threshold event)
20	0b	Select ALL Counters (SAC). This bit controls how the opcode in bits 19:16 is applied to all counters. The rest of the **PMON**_CMD register is not affected by the setting of this bit. 0 = The opcode is applied only to the counter associated with this command register. 1 = The opcode is applied to ALL counters. This means that every command register is written to with the same value that was written to this particular command register. This is particularly useful for resetting all counters with a single command or starting or stopping all counters simultaneously. ***Note:*** This bit is only valid in Command Register 0 and has no effect in non-0 command registers. "Globally" executed commands (by setting this bit in Command Register 0) always override "locally" executed commands.

Table 492. PMON Command Register 0-7 - PMON_CMD[0:7] (Sheet 3 of 4)



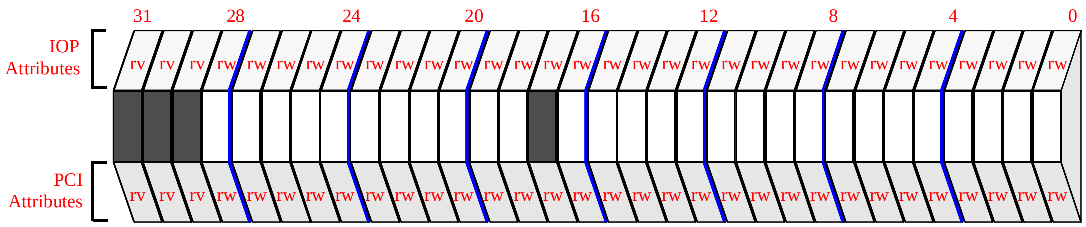
Register Offset

PMON_CMD0 +000h
PMON_CMD1 +010h
PMON_CMD2 +020h
PMON_CMD3 +030h
PMON_CMD4 +040h
PMON_CMD5 +050h
PMON_CMD6 +060h
PMON_CMD7 +070h

Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
19:16	0h	Opcode • **0000 = Stop**. The corresponding counter does not count. • **0001 = Start**. The corresponding counter begins counting. Each counter increments by one when the corresponding increment event occurs or decrements by one when the corresponding decrement event occurs. All duration type events toggle every **PMON** unit clock tick that the event is true. The desired increment and decrement events must be selected before this command executes. • **0010 = Sample**. The corresponding counter value is latched into the corresponding data register, which can then be read by reading the appropriate data register. The counter continues to count without being reset. When the Condition Code is NOT False then the Data Register is not written when a sample takes place. In other words, the Data Register is only ever updated with counter value when the Condition Code is False. • **0100 = Reset**. The corresponding counter is reset to 0000 0000h and stops counting. The 32 bit wide counter allows 4 billion clock ticks or occurrences to be counted between sample commands. When the counter rolls over, the overflow status bit is set in the corresponding status register. • **0101 = Restart**. The corresponding counter resets, then starts counting again. This is essentially a Reset & Start command. This functionality facilitates generating histograms by allowing an event to trigger to clear the counter and resume counting with no further intervention. • **0110 = Sample & Restart**. The Sample command happens and is followed immediately by the Restart command. • **1111 = Preload**. The corresponding counter is set to the value that is located in the associated data register. This facilitates rollover and overflow validation. The counter remains in the same state when preloaded. When the counter was counting before the preload was executed it continues to count after the preload. It is software' s responsibility to ensure that the counter is in the desired state (example: execute stop command) prior to issuing a preload command. All others reserved.
15	0b	Reserved

Table 492. PMON Command Register 0-7 - PMON_CMD[0:7] (Sheet 4 of 4)



Register Offset	
PMON_CMD0	+000h
PMON_CMD1	+010h
PMON_CMD2	+020h
PMON_CMD3	+030h
PMON_CMD4	+040h
PMON_CMD5	+050h
PMON_CMD6	+060h
PMON_CMD7	+070h

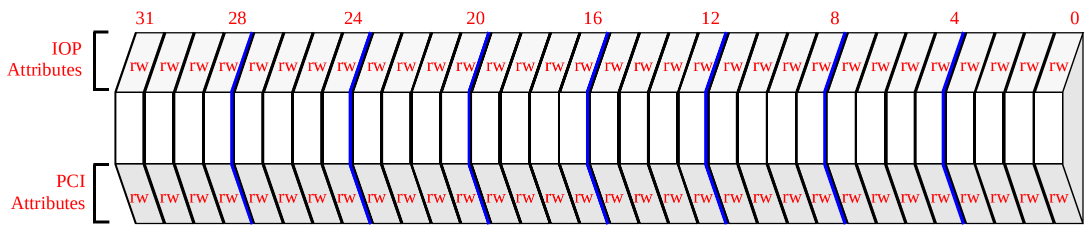
Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
14:12	000b	Command Trigger Source Select (CTSS) This field expands the Event Selection Codes in the Command trigger field to 1 of 8 ports, which share the same codes. Normally driven to 000, unless another source of the same type is to be selected. For 4138xx, there are 8 different sources for SAS/SATA events, so setting these bits as "000" choose an event from SAS/SATA Port-0, and setting these bits as "001" choose an event from SAS/SATA Port-1, etc.
11:0	000h	Command Trigger (CT) This field contains the Event Selection Code (ESC) that the unit is required to detect before executing the opcode. The previously programmed opcode continues to execute until this command trigger is detected. The ESC of 000h (the default value) is a special case, This special ESC causes the command to be triggered immediately upon being written to the command register. The special command trigger also causes that command to be executed (triggered) once and only once. All other (non-0) ESCs cause the command to be re-executed every time the trigger is detected. Refer to the Event tables for valid values. For the Stop, Start, and Reset opcodes, re-executing every time the trigger is detected is meaningless. For other opcodes, care must be taken to prevent the CT from producing undesirable behavior. The only opcodes that are anticipated as being useful with continual triggering include the Restart functionality which behavior is necessary to facilitate gathering of histogram data at hardware speeds. Only one trigger can occur per core clock. When the trigger source clock is running faster than the core clock and more than one trigger event occurs per core clock, these additional triggers are dropped. See the Event Table for valid values.

16.5.6.2 PMON Event Register 0-7 - PMON_EVR[0:7]

This 32-bit register contains the events that control the incrementing and decrementing of the **PMON** counter selected. When this register is written, the previous register contents are overwritten (the event fields are unbuffered) and the associated counter is immediately affected by the change. This register should only be programmed when the counter is idle and before the command register receives an opcode that is associated with the events in this register. When both an increment and a decrement event are detected on the same clock cycle, the counter value does not change.

Table 493. PMON Event Register 0-7 - PMON_EVR[0:7]



Register Offset

PMON_EVR0	+004h
PMON_EVR1	+014h
PMON_EVR2	+024h
PMON_EVR3	+034h
PMON_EVR4	+044h
PMON_EVR5	+054h
PMON_EVR6	+064h
PMON_EVR7	+074h

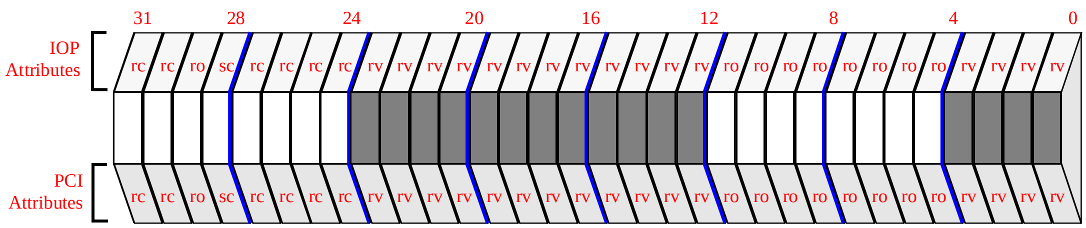
Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
31	0b	Decrement Occurrence Count Enable (DOCE) 0 = Decrement Duration Count: the counter is decremented for each clock for which the decrement event signal is asserted logic high 1 = Decrement Occurrence Count: the counter is decremented each time a rising edge of the decrement event signal is detected.
30:28	0h	Decrement Event Source Select (DESS) This field expands the Event Selection Codes in the Decrement Event field to 1 of 8 ports, which share the same codes. Normally driven to 000, unless another source of the same type is to be selected. For 4138xx, there are 8 different sources for SAS/SATA events, so setting these bits as “000” choose an event from SAS/SATA Port-0, and setting these bits as “001” choose an event from SAS/SATA Port-1, etc.
27:16	000h	Decrement Event (DE) This field contains Event Selection Code (ESC) that the unit is required to detect before decrementing the associated counter. This field is only applicable for opcodes putting the counter in a counting state. Refer to the Event tables for valid values.
15	0b	Increment Occurrence Count Enable (IOCE) 0 = Increment **Duration Count**: the counter is incremented for each clock for which the increment event signal is asserted logic high 1 = Increment **Occurrence Count**: the counter is incremented each time a rising edge of the increment event signal is detected.
14:12	000b	Increment Event Source Select (IESS) This field expands the Event Selection Codes in the Increment Event field to 1 of 8 ports, which share the same codes. Normally driven to 000, unless another source of the same type is to be selected. For 4138xx, there are 8 different sources for SAS/SATA events, so setting these bits as “000” choose an event from SAS/SATA Port-0, and setting these bits as “001” choose an event from SAS/SATA Port-1, etc.
11:0	000h	Increment Event (IE) This field contains the Event Selection Code (ESC) that the unit is required to detect before incrementing the associated counter. Refer to the Event tables for valid values.

16.5.6.3 PMON Status Register 0-7 - PMON_STS[0:7]

This 32-bit register reports the current status of the **PMON**x counter.

Table 494. PMON Status Register 0-7 - PMON_STS[0:7] (Sheet 1 of 2)

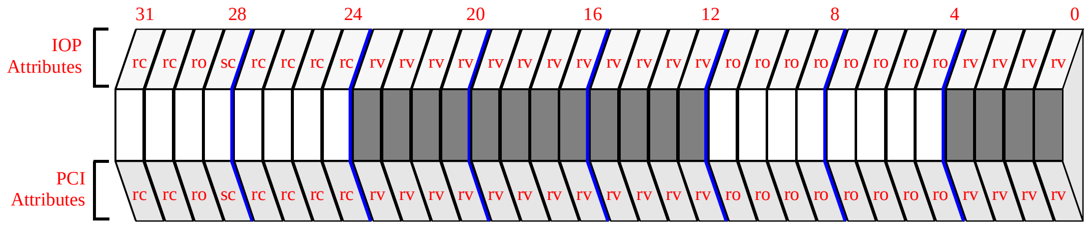


Register Offset	
PMON_STS0	+008h
PMON_STS1	+018h
PMON_STS2	+028h
PMON_STS3	+038h
PMON_STS4	+048h
PMON_STS5	+058h
PMON_STS6	+068h
PMON_STS7	+078h

Attribute Legend:	
RV = Reserved	SC = Read Set/Write Clear
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
RW = Read/Write	NA = Not Accessible

Bit	Default	Description
31	0b	Decrement Event Indicator (IEI) 0 = The selected decrement event was NOT detected. 1 = **1** = The selected decrement event WAS detected. This bit is updated every clock, whether or not the counter is counting.
30	0b	Increment Event Indicator (IEI) 0 = The selected increment event was NOT detected. 1 = **1** = The selected increment event WAS detected. This bit is updated every clock, whether or not the counter is counting.
29	0b	Counter Active Indicator (CAI) 0 = The associated counter is in a state that does NOT allow it to be incremented or decremented when the appropriate event(s) are detected. 1 = The associated counter is in a state that allows it to be incremented or decremented when the appropriate event(s) are detected. This bit is updated every clock.
28	0b	In Use Software Mutex bit • After a full component RESET, a read to this bit returns a 0 • After the first read, subsequent reads return a 1 • A write of a 1 to this bit resets the next read value to 0 • Writing a 0 to this bit has no effect Software can poll this bit until it reads a 0, and then owns the usage of the corresponding counter. This bit has no other effect on any **PMON** Counter registers, and is only used as a semaphore among various independent software threads that may need to utilize this performance counter. Software that reads this register, but does not intend to claim exclusive access, must write a 1 to this bit (when it reads a 0), in order to allow other software threads to claim it. This bit is particularly useful when supporting multi-threaded environments. Attribute is Read Set/Write Clear (RS/WC)
27	0b	Unsupported Event Indicator (UEI) 0 = No unsupported events have been selected. 1 = An unsupported Event Selection Code (ESC) was written into in one of the event selection fields (Command Trigger, Increment Event, or Decrement Event).
26	0b	Overflow/Underflow Indicator (OUI) 0 = The associated 32 bit counter has NOT rolled over since the last time it was cleared. 1 = The associated 32 bit counter HAS rolled over since the last time it was cleared.

Table 494. PMON Status Register 0-7 - PMON_STS[0:7] (Sheet 2 of 2)



Register Offset

PMON_STS0	+008h
PMON_STS1	+018h
PMON_STS2	+028h
PMON_STS3	+038h
PMON_STS4	+048h
PMON_STS5	+058h
PMON_STS6	+068h
PMON_STS7	+078h

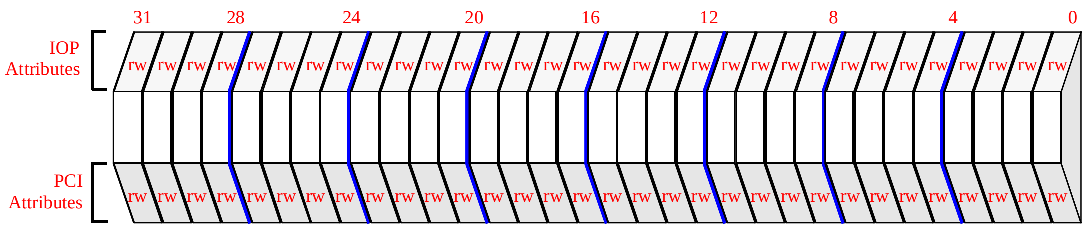
Attribute Legend:
RV = Reserved
PR = Preserved
RS = Read/Set
RW = Read/Write
SC = Read Set/Write Clear
RC = Read Clear
RO = Read Only
NA = Not Accessible

Bit	Default	Description
25	0b	Command Trigger Indicator (CTI) 0 = NO commands have been triggered since the last time this bit was cleared. 1 = A command WAS triggered since the last time this bit was cleared. Software can use this bit to know that a command that was pending earlier has been triggered. Once a command has been triggered, another command can be triggered to execute. The null event does not cause this Command Trigger Indicator to be asserted.
24	0b	Threshold Indicator (THI) 0 = No threshold event has been generated since the last time this bit was cleared. 1 = This counter generated a threshold event due to a true threshold condition compare since the last time this bit was cleared.
23:12	000h	Reserved
11:4	TBD	Clock Period (CP) This fixed point field is 5.3 format which allows representing clock periods from 0.125 ns (8 GHz) to 31.875 ns (a little over 31 MHz) in 125 ps increments. Example: 100 MHz = 10.000 ns period = 01010.000b = 50h Example: 133 MHz = 7.500 ns period = 00111.100b = 3Ch Example: 167 MHz = 6.000 ns period = 00110.000b = 30h Example: 200 MHz = 5.000 ns period = 00101.000b = 28h
3:0	0h	Reserved

16.5.6.4 PMON Data Register 0-7 - PMON_DATA[0:7]

This 32-bit register allows for reading of the sampled value from **PMON** event counter X and contains the threshold value that is compared to the value in the event counter when a threshold condition (non-0 condition code) is in effect.

Table 495. PMON DATA Register 7-0 - PMON_DATA[7:0]



Register Offset	
PMON_DATA0	+00Ch
PMON_DATA1	+01Ch
PMON_DATA2	+02Ch
PMON_DATA3	+03Ch
PMON_DATA4	+04Ch
PMON_DATA5	+05Ch
PMON_DATA6	+06Ch
PMON_DATA7	+07Ch

Attribute Legend:	
RV = Reserved	RW = Read/Write
PR = Preserved	RC = Read Clear
RS = Read/Set	RO = Read Only
	NA = Not Accessible

Bit	Default	Description
31:0	0000 0000h	**Counter X Value:** Contains either duration (number of clock ticks) or occurrences contained in **PMON** event counter n at time of sampling. The register is programmed to contain the threshold value that is compared to the value in the event counter when non-0 condition codes have been selected.

16.5.7 PMON Events

PMON events can be selected for any of the counters. Each event is defined by a unique Event Selection Code (ESC) as defined below. For events which are duplicated within 4138xx with duplicate units (ADMA, SAS ports, etc.) source select field is used to select the unit the corresponding event is associated with. Event types are designated by Type, Occurrence only (O), Duration only (D) or events that can be selected as either (OD). Events must be used according to the defined type for proper operation. Use of a Duration only event, as an Occurrence event, results in unpredictable behavior.

Table 496. Event Selection Code Summary

ESC Range	Description
000-0FF	**PMON** Counters Internal Events • 000 = Null (False) Event • 001 - 00F = Clock Domains • 010 - 020 = Threshold Events • 021 - 0FF = Reserved for **PMON** internal events
100-3FF	Reserved
400-4FF	DDR-II SDRAM Memory Controller
500-5FF	Reserved
600-67F	Reserved
680-6FF	PCI-X ATU
700-7FF	PCI Express ATU
800-87F	North Internal XSI Bus
880-8FF	South Internal XSI Bus
900-FFF	Reserved

16.5.7.1 Null Event

The Null Event is not an actual event. When used as an increment or decrement event, no action takes place. When used as Command Trigger, it causes command to be triggered immediately after command register is written to by software. The Command Trigger Indicator status bit is NOT set when the Null Event is the Command Trigger. Also called False Event.

Table 497. Intel® 413808 and 413812 I/O Controllers in TPER Mode PMON Clock Events

Event Selection Code (Hex)	Event	Type	Comment
000	Null Event	N/A	Not an actual event

16.5.7.2 Clock Events

One of the clock events matches the frequency of the **PMON** unit clock. Software must comprehend that the frequency of this **PMON** unit clock may be different from one component to another. One piece of functionality this enables is to allow for event(s) to be counted during a fixed period of time. The Clock Event that matched the **PMON** Counter frequency can be used to trigger any command to execute. This essentially allows the command trigger capability to provide a programmable timer.

For example, when needing to know how many times event AAAh occurs during 1 millisecond, start counting event AAAh at the same time as when starting to count the **PMON** Counter frequency event. With a 100 MHz clock, event 001h would be started as an EQUAL Threshold Compare against the value 000186A0h (number of 100 MHz clock ticks in 1 ms). When the threshold compare of the counted clock events is true and therefore generates a threshold event, the threshold event triggers another counter to stop counting event AAAh.

Table 498. Intel® 413808 and 413812 I/O Controllers PMON Clock Events

Event Selection Code (Hex)	Event	Type	Comment
001	XSI clock	D	up to 400MHz, SKU dependent
Others	Reserved		

16.5.7.3 Threshold Events

These are the outputs of the threshold comparators. When the value in a data register is compared to its corresponding counter value and the condition is true, a threshold event is generated. This results in:

- Pulse on the signal lines that are routed to the events input port (one signal line from each comparator).
- Pulse on the signal line that is routed to the **PMON**OUT output pin (one signal that is an OR of all signal lines from each comparator).

One piece of functionality this enables is to allow for **PMON** commands to be completed only when a Threshold Event occurs. In other words, a Threshold Event can be used as a Command Trigger to control the execution of any **PMON** command (start, stop, sample, etc.).

Table 499. Intel® 413808 and 413812 I/O Controllers PMON Threshold Events

Event Selection Code (Hex)	Event	Type	Comment
010	Counter0 Threshold	OD	
011	Counter1 Threshold	OD	
012	Counter2 Threshold	OD	
013	Counter3 Threshold	OD	
014	Counter4 Threshold	OD	
015	Counter5 Threshold	OD	
016	Counter6 Threshold	OD	
017	Counter7 Threshold	OD	

16.5.7.4 PCI Interface Events

The PCI Interface Events apply to the ATU-X unit.

Table 500. PCI Interface Events

Event Selection Code (Hex)	Event	SRC	Type	Comment
680	PCI Inbound Data Transferred	N	D	Count equals number of 8_byte data cycles (actual data transferred can be 1 to 8 bytes). Data transferred for all cases where I/O Processor is target of transactions. Sum of events 681 and 682.
681	PCI Inbound Rd Data Transferred	N	D	Count equals number of 8_byte data cycles (actual data transferred can be 1 to 8 bytes). Data transferred for an Inbound Read where I/O Processor is target of read. Read could be satisfied with immediate data in PCI mode or with a Split Completion in PCI-X
682	PCI Inbound Data Wr Transferred	N	D	Count equals number of 8_byte data cycles (actual data transferred can be 1 to 8 bytes)
683-687	reserved			
688	PCI Out bound Data Transferred	N	D	Count equals number of 8_byte data cycles (actual data transferred can be 1 to 8 bytes). Data transferred for all cases where I/O Processor is initiator of transactions. Sum of events 689 and 68A.
689	PCI Outbound Rd Data Transferred	N	D	Count equals number of 8_byte data cycles (actual data transferred can be 1 to 8 bytes). Data transferred for an Outbound Read where I/O Processor is initiator of the read. Read could be satisfied with immediate data in PCI mode or with a Split Completion in PCI-X
68A	PCI Outbound Data Wr Transferred	N	D	Count equals number of 8_byte data cycles (actual data transferred can be 1 to 8 bytes).
68B-68F	reserved			
690	PCI Inbound Memory Reads	N	O	
691	PCI Inbound Memory Writes	N	O	
692	PCI Inbound Split Completions	N	O	
693 - 695	Reserved			
696	PCI Inbound Config Reads	N	O	
697	PCI Inbound Config Writes	N	O	
598 - 6AF	Reserved			
6B0	PCI Outbound Memory Reads	N	O	
6B1	PCI Outbound Memory Writes	N	O	
6B2	PCI Outbound Split Completions	N	O	
6B3-6EF	reserved			
6F0	PCI Inbound Read Retries	N	O	Retry due to full queue
6F1	Reserved			
6F2	PCI Inbound Write Retries	N	O	Retry due to full queue
6F3 - 6FF	Reserved			

16.5.7.5 PCI Express Interface Events

The PCI Interface Events apply to the ATU-E unit.

Table 501. PCI Express Interface Summary

Event Selection Code	Event	SRC	Type	Comment
700-70F	Reserved			
710	Total TLPs Transmitted	N	O	Transaction Layer Packets
711	Total TLPs Received	N	O	The link layer put a TLP into the upstream transaction layer queues. This includes TLPs that fail CRC, or are malformed packets that eventually get dropped.
712	Total DLLPs Transmitted	N	O	Data Link Layer Packets Transmitted, including Flow Control Updates
713	Total DLLPs Received	N	O	Data Link Layer Packets Received including Flow Control Updates
714 - 71F	Reserved			
720	Inbound Data Transferred	N	OD	
721	Inbound Mem Read Request	N	O	Number of commands, not amount of data
722	Inbound Mem Write Request	N	O	Number of commands, not amount of data
723 - 73F	Reserved			
740	Outbound Data Transferred	N	OD	
741	Outbound Read Requests	N	O	Includes I/O, memory and config outbound reads
742	Outbound Write Requests	N	O	Includes I/O, memory and config outbound writes
743 -74F	Reserved			
750	Correctable Error Message Received	N	O	
751	Non-Fatal Error Message Received	N	O	
752	Fatal Error Message Received	N	O	
753 - 78F	Reserved			
790	Correctable Error Detected	N	O	
791	Non-Fatal Error Detected	N	O	
792	Fatal Error Detected	N	O	
793 - 7FF	Reserved			

16.5.7.6 North Internal Bus Events

The North Internal Bus has multiple initiators. Some events apply to each requester unit and the following table represents the Source Select Field values for each unit.

Table 502. North Internal Bus Source Select Summary

Source Select Value	Port
0	Intel XScale® core 0
1	Intel XScale® core 1
2	Internal Bus Bridge
3:7	Reserved

The events and corresponding codes for the North Internal Bus are defined in the following table. These codes are unique to the IOP programming model of the **PMON** unit.

Table 503. North Internal Bus Initiator Events

Event Selection Code	Event	SRC	Type	Comment
800	NIB Addr Acq	Y	D	Address Acquisition Duration
801	NIB Addr Gnt	Y	O	Address Grants Received
802	NIB Data Acq	Y	D	Data Acquisition Duration
803	NIB Data Gnt	Y	O	Data Grants Received
804-807	Reserved			
808	NIB Snoop Retry	Y	O	# Transactions which receive a Snoop Retry
809	NIB Coherent Requests	Y	O	# Requests in Coherent Memory (XSI Bridge only)
80A-80F	Reserved			
810	NIB Reads	Y	O	# Read Transactions
811	NIB Read Data	Y	D	# Read Data Cycles (in 16-Bytes)
812 - 81F	Reserved			
820	NIB Writes	Y	O	# Write Transactions
821	NIB Write Data	Y	D	# Write Data Cycles (in 16-Bytes)
822 - 87F	Reserved			

16.5.7.7 South Internal Bus Events

The South Internal Bus has multiple initiators. Some events apply to each requester unit and the following table represents the Source Select Field values for each unit.

Table 504. South Internal Bus Source Select Summary

Source Select Value	Port
0	ATU-E
1	ATU-X
2	Internal Bus Bridge
3	Reserved
4	Reserved
5:7	Reserved

The events and corresponding codes for the South Internal Bus are defined in the following table. These codes are unique to the IOP programming model of the **PMON** unit.

Table 505. South Internal Bus Initiator Events

Event Selection Code	Event	SRC	Type	Comment
880	SIB Addr Acq	Y	D	Address Acquisition Duration
881	SIB Addr Gnt	Y	O	Address Grants Received
882	SIB Data Acq	Y	D	Data Acquisition Duration
883	SIB Data Gnt	Y	O	Data Grants Received
884-887	Reserved			
888	SIB Snoop Retry	Y	O	# Transactions which receive a Snoop Retry
889	SIB Coherent Requests	Y	O	# Requests in Coherent Memory (Internal Bus Bridge only)
88A-88F	Reserved			
890	SIB Reads	Y	O	# Read Transactions
891	SIB Read Data	Y	D	# Read Data Cycles (in 16-Bytes)
892 - 89F	Reserved			
8A0	SIB Writes	Y	O	# Write Transactions
8A1	SIB Write Data	Y	D	# Write Data Cycles (in 16-Bytes)
8A2 - 8FF	Reserved			

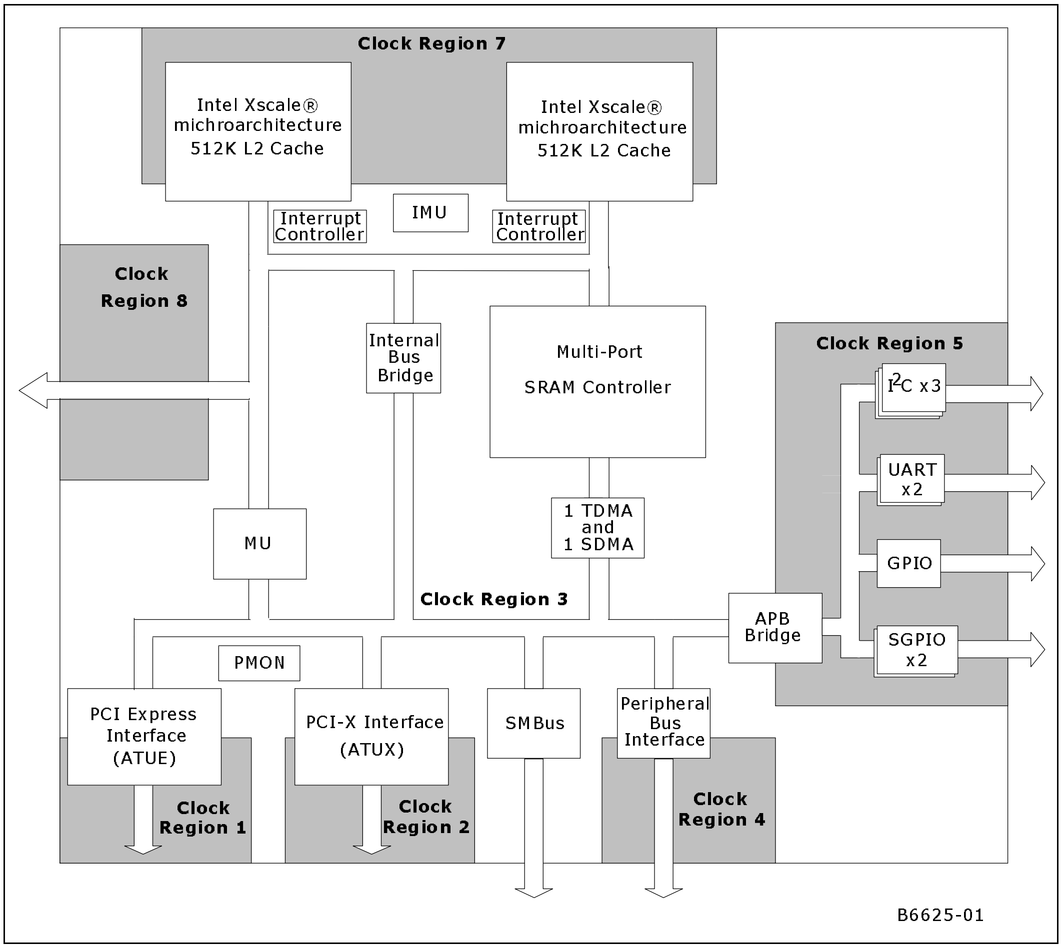
17.0 Clocking and Reset

This chapter describes the clocking and reset function of the Intel® 413808 and 413812 I/O Controllers in TPER Mode (4138xx).

17.1 Clocking Overview

The 4138xxcontains various internal clocking boundaries. PLLs are used to generate the clocks. One PLL for the PCI Express interface, one for PCI-X interface, one for Memory Interface and one for everything else. Clock regions 3, 4, and 6 are driven off the core PLL and are pseudo-synchronous to each other. There are asynchronous boundaries between regions 1/3, regions 2/3, 6/3, and between regions 7/3.

Figure 119. Intel® 413808 and 413812 I/O Controllers in TPER Mode Clocking Regions Diagram



17.1.1 Clocking Theory of Operation

Each region within the 81348 contains different clocking requirements. These requirements are summarized in the following sections.

17.1.1.1 Clocking Region 1 (PCI Express)

Region 1 obtains its input clock from the PCI Express reference clock The **REFCLK+/-** differential input supplies a 100 MHz clock for normal operation on the PCI Express interface. The analog front end for this region generates the 2.5 GHz clock used for the serial PCI Express interface as well as a 250 MHz clock used by the transaction layer. In addition to the locally generated clocks, region 1 utilizes the internal bus clock generated by the core PLL to interface to the internal bus (region 3).

Also contains a 25 MHz clock used by the power management logic.

17.1.1.2 Clocking Region 2 (PCI)

Region 2 obtains its input clock from the PCI bus clock connected to the input pin **P_CLKIN**. 81348 supports an input frequency of 33 MHz, 66 MHz, 100 MHz, and 133 MHz for normal operation on the PCI interface. Additionally region 2 utilizes the internal bus clock generated by the core PLL to interface to the internal bus (region 3).

17.1.1.2.1 Central Resource Mode (PCIX_EP# = ‘1’)

When operating as the Central Resource (**PCIX_EP#** = 1), the bus PCI bus operating frequency and default value in ATUX PCSR[19:16] is determined based on the sampling of **PCIXCAP**, **P_MODE2**, **M66EN**, **PCIXM1_100#**, and **PCIXM2_100#**.

Table 506. PCI Bus Frequency Initialization[a]

P_PCIXCAP	P_MODE2	P_M66EN	PCIXM1_100#	PCIXM2_100#	PCI Bus Mode	PCI Bus Frequency	ATUX PCSR[19:16]
< 0.11VCC	–[b]	Ground	–	–	PCI	33 MHz	1111
< 0.11VCC	–	Not connected	–	–	PCI	66 MHz	1111
< 0.6VCC & > 0.11 VCC	GND	–	–	–	PCI-X Mode 1	66 MHz	1110
< 0.6VCC & > 0.11VCC	VCC	–	–	GND	PCI-X Mode 2	100 MHz (PCI-X 200 MHz)	0101
< 0.6VCC & > 0.11VCC	VCC	–	–	VCC	PCI-X Mode 2	133 MHz (PCI-X 266 MHz)	0100
< 0.89VCC & > 0.6VCC	–	–	–	–	PCI-X Mode 1	66 MHz	1110
> 0.89VCC	–	–	GND	–	PCI-X Mode 1	100 MHz	1101
> 0.89VCC	–	–	VCC	–	PCI-X Mode 1	133 MHz	1100

a. 81348 does not support PCI-X 533Mhz.
b. A ‘–’ in table indicates value is a ‘don’t care’ for computing bus mode/frequency. All signals must still be pulled to a valid logic level.

When PCI Express* REFCLK is the primary clock source, the **CR_FREQ[1:0]** pins can be used to control the operating frequency of an external clock source for the PCI bus. The **CR_FREQ[1:0]** value can be changed by setting bits[19:16] in the ATUX PCSR register (offset x074).

Table 507. CR_FREQ[1:0] Encoding

ATUX PCSR[19:16] (PCIX Init Pattern)	ATUX PCSR[10] (P_M66EN)	CR_FREQ[1:0]	Bus Frequency	Bus Mode
1111	0	11	33 MHz	Conventional PCI
1111	1	10	66 MHz	Conventional PCI
1110	–	10	66 MHz	PCI-X
1101	–	01	100 MHz	PCI-X
1100	–	00	133 MHz	PCI-X

17.1.1.2.2 cPCI Hot-Swap Mode (PCIX_EP# = '0' and HS_SM# = '0')

When operating in a Compact PCI Hot-Swap environment (**HS_SM#**=0, **PCIX_EP#**=0), the **HS_FREQ[1:0]** and **P_M66EN** pins are used to determine PCI Bus operating frequency and set default value in ATUX PCSR[19:16]. Inputs are sampled at trailing edge of reset and used to set dividers on region 2 PLL.

Table 508. HS_FREQ Encoding [a]

HS_FREQ[1:0]	P_M66EN	Operating Mode	Bus Frequency	PCSR[19:16]
11	0	PCI	33 MHz	1111
11	1	PCI	66 MHz	1111
10	–	PCI-X (Mode 1)	66 MHz	1110
01	–	PCI-X (Mode 1)	100 MHz	1101
00	–	PCI-X (Mode 1)	133 MHz	1100

a. Hot-Swap is not supported in PCI-X Mode 2.

17.1.1.2.3 End Point Mode (PCIX_EP# = 0 and HS_SM# = 1)

When operating in end point mode, the setting of PCSR[19:16] is determined by the initialization pattern sampled off the PCI bus.

Table 509. PCI-X Initialization Pattern[1]

PERR#	DEVSEL#	STOP#	TRDY#	ATUX PCSR[19:16]	Mode	Clock Frequency (MHz)	
						Minimum	Maximum
Deasserted	Deasserted	Deasserted	Deasserted	1111	PCI 33	16	33
				1111	PCI 66	33	66
Deasserted	Deasserted	Deasserted	Asserted	1110	PCI-X 66 Mode 1	50	66
Deasserted	Deasserted	Asserted	Deasserted	1101	PCI-X100 Mode 1	66	100
Deasserted	Deasserted	Asserted	Asserted	1100	PCI-X 133 Mode 1	100	133
Deasserted	Asserted	Deasserted	Deasserted	Reserved	PCI-X Mode 1	Reserved	
Deasserted	Asserted	Deasserted	Asserted		PCI-X Mode 1		
Deasserted	Asserted	Asserted	Deasserted		PCI-X Mode 1		
Deasserted	Asserted	Asserted	Asserted		PCI-X Mode 1		
Asserted	Deasserted	Deasserted	Deasserted		PCI-X (Mode 2)		
Asserted	Deasserted	Deasserted	Asserted	0110	PCI-X 66 (Mode 2)	50	66
Asserted	Deasserted	Asserted	Deasserted	0101	PCI-X 100 (Mode 2)	66	100
Asserted	Deasserted	Asserted	Asserted	0100	PCI-X 133 (Mode 2)	100	133
Asserted	Asserted	Deasserted	Deasserted	Reserved	PCI-X	Reserved	
Asserted	Asserted	Deasserted	Asserted		PCI-X		
Asserted	Asserted	Asserted	Deasserted		PCI-X		
Asserted	Asserted	Asserted	Asserted		PCI-X		

1. 81348 supports neither PCI-X 533 Mode nor ECC in Mode 1.

17.1.1.2.4 Secondary Clock Outputs

This component has the ability to provide four PCI bus clocks (**P_CLKO[3:0]**) to drive external components as well as a dedicated feedback clock (**P_CLKOUT**) to drive the ATUX PCI interface. These clock outputs are can only be used when the PCI Express reference clock (**REFCLK+/-**) is used as the primary chip clock and the ATUX is enabled and configured to operate as a Central Resource.

Table 510. Secondary Clock Output Control

PCIX_EP#	CLK_SRC_PCIE#	INTERFACE_SEL_PCIX#	PCIX/PCIe Interfaces Active	Clock Outputs Enabled
0 (End Point)	–	–	–	No
1 (Central Resource)	1 (PCLKIN)	–	–	No
1 (Central Resource)	0 (REFCLK+/-)	1 (PCIe)	–	No
1 (Central Resource)	0 (REFCLK+/-)	0 (PCIX)	–	Yes
1 (Central Resource)	0 (REFCLK+/-)	–	Dual Interface	Yes

17.1.1.3 Clocking Region 3 (Internal Bus)

Region 3 covers the 81348 internal bus and obtains is input clock from either **REFCLK+/-**or **P_CLKIN** defending on the setting of the **CLK_SRC_PCIE#** strap. Region 3 operates at a frequency up to 400 MHz. All units interface to or reside in this region. The units which interface to this region include the Intel XScale® processors, the Memory Controller Unit, ATU and bridge to the AHB bus. Units which wholly reside within region 3 include the Application DMA, XSI Bridge, SMBus interface, and Peripheral Bus interface.

Region 3 contains an open-drain bi-directional clock (SMBCLK) used by the SMBus interface. The SMBCLK operates at a maximum clock frequency of 100 KHz.

17.1.1.4 Clocking Region 4 (Peripheral Bus Interface)

Region 4 obtains its input clock from the clocking unit specified in region 3. This region operates at a fixed 66 MHz and depending on the frequency of region 3 may be an asynchronous boundary.

17.1.1.5 Clocking Region 5

Region 5 obtains its input clock from the clocking unit specified in region 3. This region is used for low-speed peripheral units. Currently, this includes:

- I^2C bus interface
- General-Purpose I/O unit
- UART serial bus interface
- Serial General-Purpose I/O unit

Region 5 contains an output clock (SCL) used for the I^2C bus interface. The SCL clock frequency is 100 KHz or 400 KHz. SCL is generated from the internal bus clock with its frequency determined by the Fast Mode bit in the I2C Control Register x — ICRx. The UART input clock is driven at 16.67 MHz, and divided within the unit for the serial interface baud rate.

17.1.1.6 Clocking Region 7 (Intel XScale® Processor)

Region 7 obtains its input clock from either **REFCLK+/-**or **P_CLKIN** defending on the setting of the **CLK_SRC_PCIE#** strap. This region is the Intel XScale® processor (ARM* architecture compliant). It supports clock frequencies up to a maximum of 1200 MHz operation. The region 7 clock is an integer multiple of the Internal Bus clock (region 3).

17.1.2 Clocking Region Summary

Table 511 summarizes all of the input clock pins, output clock pins, and clock strapping option pins used in 81348.

Table 511. Clock Pin Summary

Pin	Input/Output	Description
P_CLKIN	Input	**PCI Bus Input Clock:** Provides timing for all PCI transactions
REFCLKP **REFCLKN**	Input	**PCI Express Input Clock:** Differential 100 MHz Clock
P_M66EN	Input	PCI 66 MHz enable.
TCK	Input	**Test Clock:** provides clock input for IEEE 1149.1 Boundary Scan Testing (JTAG).
HS_FREQ[1:0]/ CR_FREQ[1:0]	Strap/Output	Hot-Swap **Frequency:** While in Hot-Swap mode (**HS_SM#** = 0 and **PCIX_EP#** = 0), these pins are inputs which determine the PCI bus mode and frequency. See Section 17.1.1.2, "Clocking Region 2 (PCI)" on page 764 for more details. **Central Resource Frequency:** While operating in Central Resource Mode (**PCIX_EP#** = 1), these pins are outputs which control an external PCI-X clock generator.
P_CLKOUT	Output	**PCI Bus Output Clock:** When **REFCLKN**/**REFCLKP** are used, the IO Processor can generate the PCI output clocks. This pin would then be connected to **P_CLKIN** and trace length matched to **P_CLKO[3:0]**. The **P_CLKOUT** and **P_CLKO[3:0]** outputs are enabled when the PCI-X Interface is operating as a central resource (**PCIX_EP#** = 1) and the PCI Express input clock is used as the primary clock input (**CLK_SRC_PCIE#** = 0). See Section 17.1.1.2.4, "Secondary Clock Outputs" on page 767 for more details.
P_CLKO[3:0]	Output	**PCI Bus Output Clocks:** When **REFCLKN**/**REFCLKP** are used, the I/O processor can generate the PCI output clocks. These pins then provide the PCI clocks to devices on the PCI bus. The **P_CLKOUT** and **P_CLKO[3:0]** outputs are enabled when the PCI-X Interface is operating as a central resource (**PCIX_EP#** = 1) and the PCI Express input clock is used as the primary clock input (**CLK_SRC_PCIE#** = 0). See Section 17.1.1.2.4, "Secondary Clock Outputs" on page 767 for more details.
SCL[2:0]	Input/Output	**I^{2}C Clock:** These are bi-directional Open Drain clocks that provides for synchronous operation of the I^{2}C buses.
SMBCLK	Input/Output	**SMBus Output Clock:** Bi-directional open-drain clock that provides for synchronous operation of the SMBus.
CLK_SRC_PCIE#	Strap	Selects the clock source used to drive the internal logic and Intel XScale® processor. 0 = Source clock is the PCI Express reference clock (**REFCLKN**/**REFCLKP**). Note that if this selection is made, the **REFCLKN**/**REFCLKP** must be provided regardless of the PCI Express interface mode: Endpoint or Root Complex. 1 = Source clock is the PCI input clock (**P_CLKIN**). Note that if this selection is made, the **P_CLKIN** must be provided regardless of the PCI-X interface mode: Endpoint or Central Resource. ***Note:*** **CLK_SRC_PCIE#** is not sampled during a WARM Reset.
MEM_FREQ[1:0]	Strap	**Memory Frequency:** Determines the frequency of the DDR2 SDRAM memory subsystem. MEM_FREQ[1:0] 00 Reserved 01 Reserved 10 DDR-II SDRAM @ 533 MHz 11 DDR-II SDRAM @ 400 MHz (default)
PCIXM1_100#	Strap	**PCI-X Mode 1 100 MHz Enable:** When operating as the Central Resource (**PCIX_EP#** = 1) this strap limits the PCI-X bus to 100 MHz when operating in Mode 1.
PCIXM2_100#	Strap	**PCI-X Mode 2 100 MHz Enable:** When operating as the Central Resource (**PCIX_EP#** = 1) this strap limits the PCI-X bus to 100 MHz when operating in Mode 2.

17.2 Reset Overview

17.2.1 Fundamental Reset

There are four fundamental (hardware) resets for the 81348. The main power on reset is controlled through the PCI reset signal (**P_RST#**). When this signal is asserted, the entire I/O Processor is placed in a reset state. The other resets have differing behavior based on strapping options and chip mode. The reset straps are sampled at the exit of all fundamental resets.

- **P_RST#** — This is an asynchronous input pin which resets the entire chip. All reset straps are sampled at the rising edge of P_RST#.
- **WARM_RST#** — This is an asynchronous input pin which resets the entire chip with the exception of 'sticky' bits in the PMMR registers. All reset straps are sampled at the rising edge of WARM_RST#.
- **PCI Express Hot Reset —** When operating as an endpoint and the Hot Reset sequence is received in the TS1 ordered set, the entire chip is reset with the exception of the PCI Express physical layer and 'sticky' bits in the PMMR registers. This reset is not applicable when the PCI Express interface is disabled. A subset of straps are sampled as described in Section 17.5, "Reset Strapping Options"
- **PCI Express Loopback** — When operating as an endpoint and the Loopback sequence is received in the TS1 ordered set, the entire chip is reset with the exception of the PCI Express physical layer and "sticky" bits in the PMMR registers. This reset is not applicable when the PCI Express interface is disabled. A subset of straps are sampled as described in Section 17.5, "Reset Strapping Options"
- **PCI Express Disable Link —** When operating as an endpoint and the Disable Link sequence is received in the TS1 ordered set, the entire chip is reset with the exception of the PCI Express physical layer and "sticky" bits in the PMMR registers. This reset is not applicable when the PCI Express interface is disabled. A subset of straps are sampled as described in Section 17.5, "Reset Strapping Options"
- **PCI Express Link Down** — When operating as an endpoint and the PCI Express Link transitions to the link down state, the entire chip, including the PCI Express physically layer, is reset with the exception of "sticky" bits in the PMMR registers. This reset is not applicable when the PCI Express interface is disabled. A subset of straps are sampled as described in Section 17.5, "Reset Strapping Options"

17.2.2 Software Reset

In addition to the fundamental resets, 81348 provides software control to reset the internal bus, and Intel XScale® processor. Reset straps are not re-sampled due to these resets.

- **Internal Bus Reset Bit** — This reset can be initiated in two ways. The first is by writing to the coordinated reset bits in the MU Section 4.7.4, “Inbound Interrupt Status Register - IISR” on page 414. The second is via the watchdog timer as described in Section 11.1.2, “Watch Dog Timer Operation” on page 629. This reset is specific to the integrated I/O processor and the associated peripheral units. PCI Configuration Registers are preserved through this reset. See Section 17.2.8, “Internal Bus Reset” for more details.
- Intel XScale® Processor **Reset Bit** — This reset is initiated through the **HOLD_X0_IN_RST#** /**HOLD_X1_IN_RST#** straps, the ATUX Section 2.14.41, “PCI Configuration and Status Register - PCSR” on page 178, or the ATUE Section 3.17.41, “PCI Configuration and Status Register - PCSR” on page 327. Once invoked, the Intel XScale® processors are held in reset until released by software. See Section 17.2.7, “Intel XScale® Processor Reset Mechanism” for more details.
- **Targeted Core Reset** — The Targeted Core reset can be used by an Intel XScale® processor to imitate a reset to another core in the system, including itself.

17.2.3 Secondary Bus Reset

When operating as a root complex or central resource, the following ‘secondary bus’ resets apply. Reset straps are not re-sampled due to these resets.

- **PCI Express Hot Reset** — When operating as a root complex, the ATUE can generate the PCI Express Hot Reset sequence in order to reset the downstream components on the PCI Express interface. This is accomplished by setting bit 0 of the “PCI Express Link Control/Status Register - PELCSR” on page 334 in the ATUE.(Section 3.17.45)
- **PCI Bus Reset** — When operating as the central resource on the PCI/X bus, the P_RSTOUT# output can be used to reset the downstream PCI bus. This is accomplished by writing to bit 21 of the “PCI Configuration and Status Register - PCSR” on page 178 in the ATUX (Section 2.14.41). Bit 21 of PCSR defaults to a ‘1’. It is the responsibility of the firmware to clear the Central Resource PCI Bus Reset bit. After firmware clears bit 21 of PCSR hardware keeps the P_RSTOUT# signal asserted (low) for about 300uS - the hardware waits about 150uS to allow the PLL to warm-up and another 150uS to allow the clocks to stabilize. Therefore, firmware has to wait about 300 uS after clearing bit 21 of PCSR. After the 300uS has elapsed, hardware de-asserts the P_RSTOUT# signal. After P_RSTOUT# de-asserts, firmware has to wait before issuing the first configuration cycle in order to meet the PCI timing parameter Trhfa (about 2^{26} PCI clocks). Note that the PCI timing parameter Trhfa is dependent on the PCI bus speed selected.

17.2.4 PCI Reset

This is the primary reset input for both the PCI Express and PCI/X interfaces. The **P_RST#** reset clears all internal state machines and logic, and initialize all registers, including sticky bits, to their default states. The assertion and deassertion of the PCI reset signal **P_RST#** is asynchronous with respect to **P_CLKIN**/**REFCLK+/-**. The rising edge of the **P_RST#** signal must be monotonic through the input switching range and must meet the minimum slew rate. The PCI local bus specification defines the assertion of **P_RST#** for a period of 1 ms after power is stable.

Upon the assertion of **P_RST#**, all units within the component are reset.

Upon the deassertion of **P_RST#**, the strapping pins are sampled to set configuration modes (refer to Section 17.5, "Reset Strapping Options" on page 779).

17.2.5 PCI Express Hot Reset

The PCI Express* specification defines an in-band reset sequence that is used to reset the link and downstream components. The Root Complex communicates the fact that it is entering and coming out of a reset using these messages and the downstream devices respond by also going through a reset. This incoming message by nature of the PCI Express protocol is asynchronous to the reference clock.

As and End Point, the PCI Express Hot Reset clears all internal state machines and logic, and initialize all registers to their default states except 'sticky' error bits which are persistent through this reset.

As a root complex, the 4138xx can generate the reset message. In this case, the Hot Reset does not cause a full chip reset and does not affect the internal logic.

17.2.6 WARM_RST# Reset Mechanism

The **WARM_RST#** reset clear all internal state machines and logic, and initialize all registers to their default states except 'sticky' error bits which are persistent through reset. To eliminate potential system reliability problems, all devices are also required to either tristate their outputs or to drive them to safe levels during such a power on reset.

17.2.7 Intel XScale® Processor Reset Mechanism

This reset is initiated through:

- the **HOLD_X0_IN_RST#** /**HOLD_X1_IN_RST#** straps
- the ATUX — Section 2.14.41, "PCI Configuration and Status Register - PCSR" on page 178
- the ATUE — Section 3.17.41, "PCI Configuration and Status Register - PCSR" on page 327.

Once invoked, the Intel XScale® processors are held in reset until released by software

The **HOLD_X0_IN_RST#** /**HOLD_X1_IN_RST#** are sampled at the trailing edge of the fundamental resets and control the default value of the Core Processor Reset bits in function 0. When invoked via the strap, software should clear the Core Processor Reset bits in function 0 to bring the Intel XScale® processor out of reset.

After the initial boot sequence, the host driver can place the Intel XScale® processor in reset by asserting the Initiate Core Reset bits in any of the enabled functions which causes the corresponding Core Processor Reset bits to be set. Software must clear the Core Processor Reset bits in the same function in order to bring the Intel XScale® processor out of reset.

Table 512. Core Reset Control Bit Locations

Unit	Core Processor Reset	Initiate Core Reset
ATUE	PCSR[1:0]	PCSR[9:8]
ATUX	PCSR[1:0]	PCSR[31:30]

17.2.8 Internal Bus Reset

This reset can be initiated through:

- The coordinated reset bits in the MU Section 4.7.4, "Inbound Interrupt Status Register - IISR" on page 414.
- The watchdog timer as described in Section 11.1.2, "Watch Dog Timer Operation" on page 629.

This function resets the Intel XScale® processor and all units on the internal bus, while preserving the PCI Configuration Registers.

Software must quiesce all PCI bus traffic before initiating the Internal Bus Reset.

1. Disable the ATU from either claiming or initiating new transactions by clearing the *Bus Master Enable* and the *Memory Enable* in the ATU Command Register.
2. Monitor the *Inbound Read Transaction Queue Status,* the *Outbound Read Transaction Queue Status,* and in PCI Express mode, the *Link Layer Retry Buffer Status* in the PCSR.
3. When the Inbound Read Transaction queue, the Outbound Read Transaction queue, and the Link Layer Retry Buffer are empty, software writes to the Coordinated Reset bit to initiate the Internal Bus Reset.

The Intel XScale® processor may or may not be held in reset, depending on the default value of the Core Processor Reset bit as described in Section 17.2.7, "Intel XScale® Processor Reset Mechanism".

When the reset internal bus bit in the PCI Configuration and Status Register is set, there are sideband signals notifying the ATUX and MCU that a reset is coming. Table 514, "Internal Bus Reset Summary" describes the operation of each unit:

Table 513. Internal Bus Reset Control Bit Locations

Unit	Coordinated Reset	Selective Reset
MU	IRCSR[1]	IRSCR[0]

Table 514. Internal Bus Reset Summary (Sheet 1 of 2)

Unit	Preparation for Reset	Reset Status
ATUX	The affect on the ATUX depends on the **PCIX_EP#** strap. End Point Mode (**PCIX_EP#** = 0): When the ATUX is informed that an internal bus reset is coming it does the following: PCI Interface Outbound Transaction: • When the ATUX has already asserted its PCI request signal, and not yet started a transaction, the ATUX deasserts its request and not start its transaction. • When the ATUX has not yet requested the PCI bus, the ATUX never asserts its request for the PCI bus. • When the ATUX is in the middle of a transaction, the ATUX performs the existing transaction. This means that an inbound write, the ATUX transfers as much data as available in the queue. When an outbound read, the ATUX reads the data until the transaction stops naturally (meaning that the target has ended the transaction or the ATUX has read all of the data it has been requested to read). Once terminated by the ATUX or the target, the ATUX no longer requests the PCI bus. • In PCI-X mode, the ATUX allows any outstanding split completions due to prior outstanding Split Requests to Master-Abort on the PCI Bus. Since, the IOP is only accessing Prefetchable Memory on the host, no error condition is created in the system. PCI Interface Inbound Transaction: • The ATUX no longer claims any new transactions on the PCI bus. This results in a master abort to the initiating master. • In PCI-X mode, data from an outstanding split request may not be returned to the host. It is the responsibility of the host software to handle this condition as a consequence of writing to the Internal Bus Reset bit. Internal Bus Interface: Inbound Transaction: • The ATUX goes ahead and assert their IB request signals, and try to continue any pending transactions as normal. There are no special actions taken on the internal bus for inbound transactions. Internal Bus Interface: Outbound Transaction: • For all ATUX outbound transactions, there are no special requirements since the internal units are reset. Upon meeting both the outbound and inbound transaction requirements, the ATUX asserts the sideband signal to the reset unit notifying it is ready-for-reset. Central Resource Mode (**PCIX_EP#** = 1) No special requirements, the ATUX can be reset at anytime. This reset includes the configuration space.	End Point Mode: Clear all ATUX queues and state machines. All ATUX Configuration Registers retain their current values. Central Resource Mode: Entire ATUX is reset, including Configuration Registers.

Table 514. Internal Bus Reset Summary (Sheet 2 of 2)

Unit	Preparation for Reset	Reset Status
ATUE	The ATUE does not participate in the IB Reset handshake. However the affect on the ATUE varies depending on the **PCIE_RC#** strap. End Point Mode (**PCIE_RC#** = 1): An Internal Bus Reset does not reset the ATUE when operating as an endpoint. Root Complex Mode (**PCIE_RC#** = 0) No special requirements, the ATUE can be reset at anytime. This reset includes the configuration space. ***Note:*** While the ATUX must take special requirements to prevent corrupting the PCI bus, the PCI Express link is a point-to-point interface so no precautions need to be taken before resetting the link.	End Point Mode: No affect on ATUE logic. All ATUX Configuration Registers retain their current values. Root Complex Mode: Entire ATUX is reset, including Configuration Registers. However, sticky bits in the Configuration Space are not reset.
MCU	MCU cannot be reset until the powerfail sequence completes.	Clear all interrupts and reset MMRs to default value
MU	MU can be reset at any time	PCI Configuration registers and MSI-X memory registers are preserved. All other MMRs and logic are reset to their default values.
All Other Units	No special requirements	Clear all interrupts and reset MMRs and logic to default value

17.3 Reset Pins

Table 515. Reset Pin Summary

Pin	Input/Output	Description
P_RST#	Input	Primary chip reset. Should be connected to PCI RST# or PCI Express PERST# depending on the mode of operation.
WARM_RST#	Input	Warm Reset is the same as a cold reset, except sticky configuration bits are not reset. This pin should only be used when the sticky bit functionality is required. In this scenario, the WARM_RST# pin must be tied to the system reset PCI_RST# signal while the P_RST# pin can be tied to the system power good signal. When the sticky bit functionality is not required, the WARM_RST# pin should not be used and must be tied to Vcc. When the PCI Express interface is used as an endpoint, the PCI Express inband Hot Reset Mechanism can also be used to provide the sticky bit functionality. ***Note:*** Driving WARM_RST# using any other methods than suggested above may result in unpredictable behavior of the device.
P_RSTOUT#	Output	Secondary Reset. This is used as the PCI bus reset when operating as the Central Resource
PB_RSTOUT#	Output	Peripheral Bus Reset: This signal is asserted whenever the internal logic is reset.
PCI Express Hot Reset	n/a	This is an inband reset message that can be received as an endpoint, and generated as a root complex.

17.4 Device Function Select

In all cases, the **INTERFACE_SEL_PCIX#** strap affects whether the part operates as a PCI Express or PCI-X device.

Table 516. TPER Mode Per Function Storage Port Allocation (CONTROLLER_ONLY#=1)

DF_SEL[2:0]	Intel® 413808 and 413812 I/O Controllers in TPER Mode	
	Function 0 ATU	Function 1 TPMI1
000	8	n/a
001	Reserved	
010		
011		
100		
101		
110		
111		

Table 517. Non-TPER Mode Per Function Storage Port Allocation (CONTROLLER_ONLY#=0)

DF_SEL[2:0]	Intel® 413808 and 413812 I/O Controllers in TPER Mode	
	Function 0 TPMI0	Function 1 TPMI1
000[a]	8	n/a
001	Reserved	
010		
011		
100[b]	4	4
101	Reserved	
110		
111		

a. Function 1 is disabled and not visible in configuration space when **DF_SEL[2:0]** = 000.
b. Only valid for 4138xx mode.

17.5 Reset Strapping Options

Note: See Datasheet and/or Design Guide for details on how to configure reset straps.

Table 518, "Reset Strap Signals" on page 780 details the reset strapping options that are available to configure the component during reset. These straps are sampled and the component operating mode is determined at the deassertion of the fundamental reset. All the straps are sampled at the trailing edge of **P_RST#** and **WARM_RST#**; however, a subset of straps are sampled for other resets.

- **P_RST#** and **WARM_RST#**
  Sample all straps at the deassertion of both **P_RST#** and **WARM_RST#**.
- PCI Express Hot Reset, Loopback, Disable Link, and Link Down
  When operating as a PCI Express endpoint, the following straps are re-sampled at the deassertion of the reset condition.
  — DFSEL[2:0]
  — CONTROLLER_ONLY#
  — CFG_CYCLE_EN#
  — HOLD_X0_IN_RST#
  — HOLD_X1_IN_RST#
- Software Reset (Internal Bus reset, Core reset, and so on)
  Software Resets do not initiate a re-sampling of the reset straps and do not change the mode of operation of the component.

Table 518. Reset Strap Signals (Sheet 1 of 2)

Name	Description
BOOT_WIDTH_8#	**PBI Boot Bus Width: I**ndicates default bus width for the PBI Memory Boot window. 0 = 8 bits wide (Requires pull-down resistor) 1 = 16 bits wide (Default mode)
DF_SEL[2:0]	Device Function Select: See Section 17.4, “Device Function Select” for additional details.
CONTROLLER_ONLY#	Controller only enable 0 = Controller Only. Non-TPER Mode. 1 = TPER mode.
CFG_CYCLE_EN#	Configuration Cycle Enable: Determines when the PCI interface retries configuration cycles until the Configuration Cycle Retry bit is cleared in the ATU (PCSR[2]). 0 = Configuration Cycles enabled (Requires pull-down resistor) 1 = Configuration Retry enabled (Default mode) PCI-X Interface Configuration cycles are claimed and terminated with a retry status. PCI Express Interface Configuration requests result in a completion TLP with Configuration Retry Status (CRS).
HOLD_X0_IN_RST#	Core 0 Processor Reset Mode: This strap is latched at the trailing edge of reset and reflected in Core 0 Processor Reset bit in function 0. See Section 17.2.7, “Intel XScale® Processor Reset Mechanism” for more details. When asserted, the associated Intel XScale® processor is held in reset until software clears the Core 0 Processor Reset bit. 0 = Hold in reset (Requires pull-down resistor) 1 = Don’t hold in reset (Default mode).
HOLD_X1_IN_RST#	Core 1 Processor Reset Mode: This strap is latched at the trailing edge of reset and reflected in Core 1 Processor Reset bit in function 0. See Section 17.2.7, “Intel XScale® Processor Reset Mechanism” for more details. When asserted, the associated Intel XScale® processor is held in reset until software clears the Core 1 Processor Reset bit. 0 = Hold in reset (Requires pull-down resistor) 1 = Don’t hold in reset (Default mode).
INTERFACE_SEL_PCIX#	Selects the active interface and determines the address map for the PMMR registers. See the MMR chapter for details. 0 = PCI-X is active 1 = PCI Express is active (default mode) When both interfaces are active, this strap selects the ATU that is function 0 in the internal address map ***Note:*** For dual interface designs, INTERFACE_SEL_PCIX# must be set consistent with **PCIE_RC#** / **PCIX_EP#**. When operating with one interface as an endpoint and the other interface as a root complex. INTERFACE_SEL_PCIX# must correspond to the end point interface.
PCIE_RC#	PCI Express Root Complex: determines when the PCI Express interface operates as an endpoint or root complex. 1 = Endpoint (Default Mode).
PCIX_EP#	PCI-X End Point: determines when the PCI-X interface operates as an endpoint or central resource. 0 = Endpoint (Requires pull-down resistor)
PCIXM1_100#	PCI Bus Mode 1 100MHz Enable: limits the maximum PCI-X mode operating frequency to 100MHz while in mode1. Only used when ATU is acting as the central resource for the PCI domain. 0 = Limit maximum frequency to 100MHz.(Requires pull-down resistor) 1 = 133MHz enabled (Default mode)
PCIXM2_100#	PCI-X Mode 2 133MHz Enable: limits the maximum PCI-X mode 2 operating frequency to 100MHz while operating in mode 2. 0 = Limit maximum Frequency to 100MHz (200MHz data. 1 = 133MHz (266MHz data) enabled (Default mode)

Table 518. Reset Strap Signals (Sheet 2 of 2)

Name	Description
EXT_ARB#	External Arbiter: Determines wether the PCI interface enables the integrated arbiter, or uses an external arbiter. 0 = External Arbiter enabled (Requires pull-down resistor) 1 = Internal Arbiter enabled (Default mode)
PCIX_32BIT#	32-Bit PCI-X Bus: Sets the PCI-X bus width in the PCI-X Status Register. When this external strap is asserted, the Sunrise lake only uses the lower 32-bit of the data bus to operate and respond to PCI-X transactions. When the strap is deasserted the 81348 is able to operate and respond to both 32- and 64-bit PCI-X transactions. This external strap also controls how the REQ64# signal is driven when the ATU-X is used as central resource. As a central resource, REQ64# is driven accordingly based on the PCIX_32BIT# strap setting during the assertion of P_RSTOUT#. When PCIX_32BIT# is asserted REQ64# is driven high, and when PCIX_32BIT# is deasserted REQ64# is driven low. 0 = 32 Bit PCI-X Bus (Requires pull-down resistor) 1 = 64 Bit PCI-X Bus (Default mode)
HS_SM#	Hot-Swap Startup Mode 0 = Hot-Swap Mode Enabled (Requires pull-down resistor) 1 = Hot-Swap Mode Disabled (default mode)
SMB_A5 **SMB_A3** **SMB_A2** **SMB_A1**	SMBUS Address maps to address bits 5, 3, 2, and 1 of the SMBus Slave address. 0 = Address is low (Requires Pull-down resistor) 1 = Address is high (Default mode).
PCIX_PULLUP#	PCI-X Pull Up: determines when the PCI interface has On Die Pull Ups enabled. 0 = enable PCI pull up resistors 1 = disable PCI pull up resistors (Default mode).
FW_TIMER_OFF#	Firmware Timer Off: Disables the 400mS firmware timer. When enabled to timer automatically clears the Configuration Cycle Retry condition when the timer expires. 0 = firmware timer disabled 1 = firmware timer enabled (default mode)
CLK_SRC_PCIE#	Clock Source PCI-Express: selects the PCI-Express REFCLK pair as the input clock to the PLLs that control internal logic. 0 = source clock is REFCLKP/REFCKLN 1 = source clock is PCLK (default mode).
LK_DN_RST_BYPASS#	Link Down Reset Bypass: Disables the full chip reset that would normally be caused by a PCI Express Link Down or PCI Express Hot Reset. Refer to Section 17.2.1, "Fundamental Reset" on page 770 for reset descriptions. 0 = Do not reset on Link Down 1 = Reset on Link Down (default mode)

18.0 Test Logic Unit and Testability

18.1 Overview

This chapter summarizes testability and configuration features incorporated in the Intel® 413808 and 413812 I/O Controllers in TPER Mode (4138xx).

The 4138xx test and control logic is based on the IEEE 1149.1-2001 Standard Test Access Port and Boundary-Scan Architecture document (available from the IEEE). The TAP controller supports on-chip test logic such as Built-In Self Test and boundary-scan.

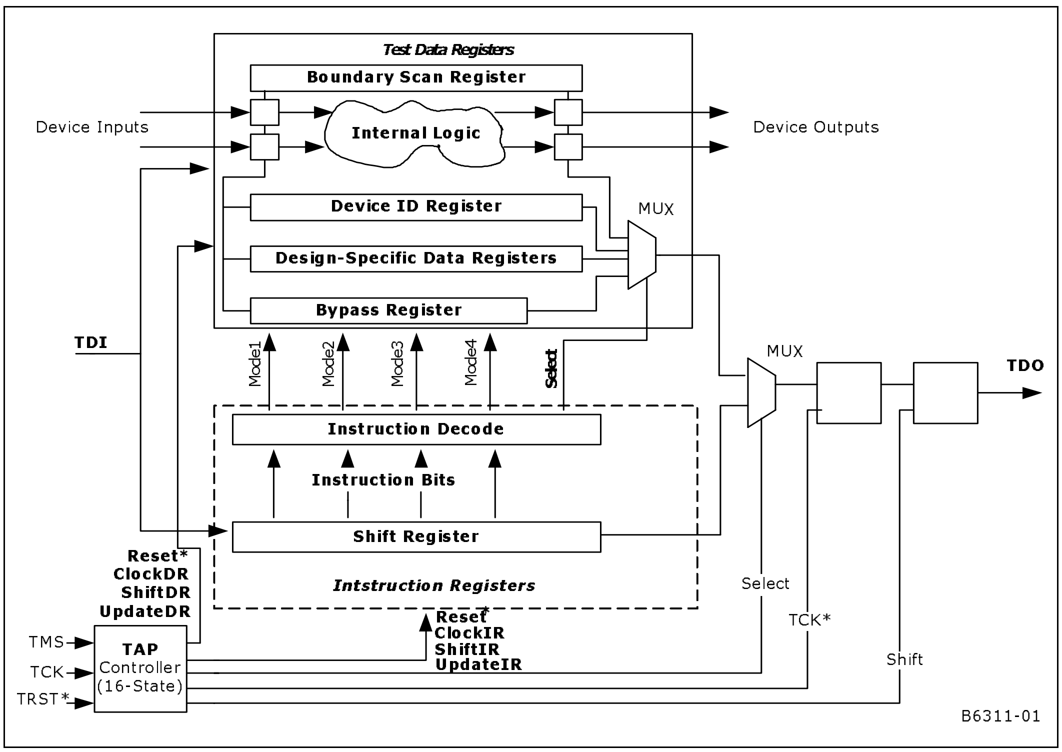
18.2 IEEE 1149.1 Standard Test Access Port (TAP)

The I/O processor contains test logic that is compatible with the IEEE Standard 1149.1-2001 Test Access Port (TAP) and Boundary Scan Architecture. Logic that conforms to this standard contains:

- Test Access Port (TAP):
  - four inputs (**TDI**, **TMS**, **TRST#** and **TCLK**)
  - single output (**TDO**).
- TAP controller
- Instruction register
- Group of test data registers

Each of these is described in more detail below. Figure 120 shows a generic diagram for logic conforming to the IEEE 1149.1 test standard.

Figure 120. IEEE 1149.1 Std. Block Diagram



18.2.1 TAP Pin Description

The internal test logic is accessed through the TAP pins. The following sections describe some of the rules and permissions of the IEEE 1149.1a Standard for the TAP pins.

18.2.1.1 Test Clock (TCK)

This is the clock input for the test logic defined by this standard, i.e. the TAP controller and associated registers. The TLU is a fully static design, thus all registers retain their states indefinitely when **TCK** is stopped at "0" or "1".

18.2.1.2 Test Mode Select (TMS)

This pin is used to control the operation of the TAP controller. The signal received at **TMS** is decoded by the TAP controller to control test operations. The state of **TMS** is sampled on the rising edge of **TCK**. Internally, there is a weak pull-up on this pin to provide a logic high when not driven, per standard definition.

18.2.1.3 Test Data Input (TDI)

This pin is used to provide serial input data to the instruction and test data registers. Data at **TDI** is sampled on the rising edge of **TCK**. Data shifted from **TDI** through a register to **TDO** appears non-inverted at **TDO** after a number of rising and falling edges of **TCK** determined by the length of the instruction or test data register selected. Internally, there is a weak pull-up on this pin to provide a logic high when not driven, per standard definition.

18.2.1.4 Test Data Output (TDO)

This is the serial data output pin. Changes in the state of **TDO** occur only following the falling edge of **TCK** while performing a shift operation. This pin is only driven while scanning (in SHDR or SHIR states) otherwise it is in inactive (high Z) state. The non-shift inactive state is provided to support parallel connection of **TDO** outputs at the board or module level.

18.2.1.5 Asynchronous Reset (TRST#)

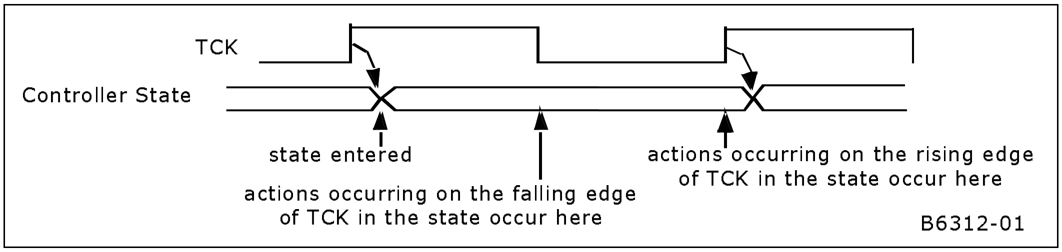
The **TRST#** signal is used to asynchronously reset the TAP controller and boundary-scan registers. The TAP controller is not initialized by any other system input, including system reset. The TAP controller initializes asynchronously on the falling edge of **TRST#** to the Test-Logic_Reset (initial) state. The TAP controller is initialized at power-up by cirrostrati built into the test logic. Upon reset, the TAP instruction register initializes to the IDCODE instruction. Internally, there is a weak pull-up on this pin to provide a logic high when not driven.

18.2.2 TAP Controller

The TAP controller, shown in Figure 122, is a sixteen-state synchronous finite state machine that changes state on the rising edge of **TCK**. The controller's next state is controlled by the state present at the **TMS** input. The TAP controller generates control signals, which together with **TCK** and control signals decoded from the instruction active in the instruction register, determine the operation of the test circuitry as defined by the IEEE Standard.

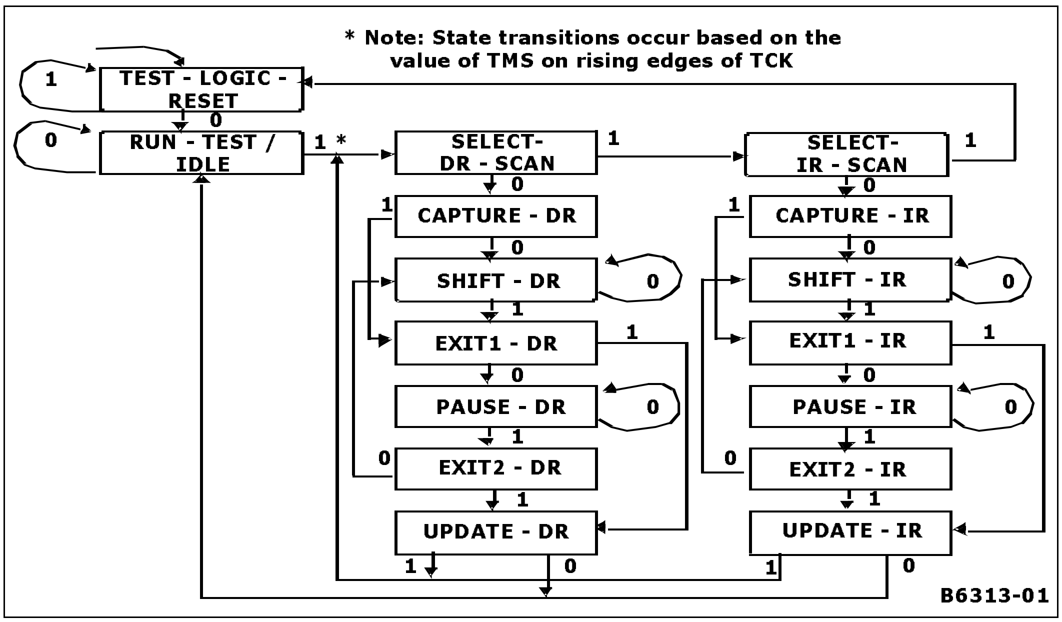
All state transitions occur based on values of TMS on the rising edge of **TCK** Actions of the test logic (instruction register, data registers, etc.) occur on either rising or falling edge of **TCK**, as show in Figure 121.. See the description of each state to learn which.

Figure 121. Timing of Actions in a TAP Controller State



For greater detail on the state machine and the public instructions, refer to IEEE 1149.1a *Standard Test Access Port and Boundary-Scan Architecture Specification*.

Figure 122. TAP Controller State Diagram



18.2.2.1 Test-Logic-Reset State

In this state, test logic is disabled to allow normal operation of the Intel XScale® processor. This is achieved by loading the instruction register with the IDCODE instruction. No matter what he state of the controller, it enters Test-Logic-Reset state when the **TMS** input is held high for at least five rising edges of **TCK**. The controller remains in this state while **TMS** is high. The TAP controller is also forced to enter this state by enabling **TRST#**.

When the controller exits the Test-Logic-Reset controller state as a result of an erroneous low signal on the **TMS** line at the time of a rising edge on **TCK** (for example, a glitch due to external interference), it returns to the Test-Logic-Reset state following three rising edges of **TCK** with the **TMS** line at the intended high logic level. Test logic operation is such that no disturbance is caused to on-chip system logic operation as the result of such an error.

Transition to next state: On the rising edge of **TCK**, when **TMS** is low move to Run-Test/Idle, else **TMS** remains high so stay in Reset.

18.2.2.2 Run-Test/Idle State

This state is a controller state between scan operations. The controller remains in this state as long as TMS is held low. In the Run-Test/Idle state, activity in selected test logic occurs only when certain instructions are present. For example, the RUNBIST instruction causes on-chip self-tests to execute in this state. Instructions that do not cause functions to execute generate no activity in the test logic while the controller is in this state.

The instruction register and all test data registers retain their current value in this state.

Transition to next state: When **TMS** is high on the rising edge of **TCK**, move to Select-DR-Scan, else remain in Idle.

18.2.2.3 Select-DR-Scan State

This is a temporary controller state. Here the decision is made to enter the Capture-DR column and initiate a scan sequence for the selected test data register.

All test data registers selected by the current instruction retain their previous value in this state.

Transition to next state: When **TMS** is low on the rising edge of **TCK**, move to Capture-DR, else move to Select-IR.

18.2.2.4 Capture-DR State

When the controller is in this state data is parallel-loaded into test data registers selected by the current instruction on the rising edge of **TCK**. Test data registers that do not have parallel inputs are not changed. Also when capturing is not required for the selected instruction, the register retains its previous state.

The instruction does not change while the TAP controller is in this state.

Transition to next state: When **TMS** is low on the rising edge of **TCK**, move to Shift-DR, else move to Exit1-DR.

18.2.2.5 Shift-DR State

In this controller state, the test data register, which is connected between **TDI** and **TDO** as a result of the current instruction, shifts data one bit position nearer to its serial output on each rising edge of **TCK**. Test data registers that the current instruction select but do not place in the serial path, retain their previous value during this state.

The instruction does not change while the TAP controller is in this state.

Transition to next state: When **TMS** is high on the rising edge of **TCK**, move to Exit1-DR, else remain at Shift-DR.

18.2.2.6 Exit1-DR State

This is a temporary controller state.

All test data registers selected by the current instruction retain their previous value during this state. The instruction does not change while the TAP controller is in this state.

Transition to next state: When **TMS** is held high on the rising edge of **TCK**, the controller enters the **Update-DR** state and the scanning process terminates. When **TMS** is held low on the rising edge of **TCK**, the controller enters the **Pause-DR** state.

18.2.2.7 Pause-DR State

The **Pause-DR** state allows the test controller to temporarily halt the shifting of data through the test data register in the serial path between **TDI** and **TDO**.

All test data registers selected by the current instruction retain their previous value during this state. The instruction does not change while the TAP controller is in this state.

Transition to next state: When **TMS** is high on the rising edge of **TCK**, move to the Exit2-DR, else remain at Pause-DR.

18.2.2.8 Exit2-DR State

This is a temporary state.

All test data registers selected by the current instruction retain their previous value during this state. The instruction does not change while the TAP controller is in this state.

Transition to next state: When **TMS** is high on the rising edge of **TCK**, the controller enters the Update-DR state and the scanning process terminates. When TMS is held low on the rising edge of **TCK**, the controller re-enters the Shift-DR state

18.2.2.9 Update-DR State

Data is latched into the parallel output of shift registers from the shift register path, on the falling edge of **TCK**.

All of the test data register's shift-register bit positions selected by the current instruction retain their previous values. The instruction does not change while the TAP controller is in this state.

Transition to next state: When **TMS** remains high on the rising edge of **TCK**, then the controller moves to the Select-DR state, else the controller moves to the Run-Test/Idle state.

18.2.2.10 Select-IR-Scan State

This is a temporary controller state. Here the decision is made to enter the Capture-IR column and initiate a scan sequence for the instruction register or to return to Test-Logic-Reset.

All test data registers selected by the current instruction retain their previous value during this state.

Transition to next state: When **TMS** is low on the rising edge of **TCK**, move to Capture-IR, else move to Test-Logic-Reset.

18.2.2.11 Capture-IR State

In this state, the shift register contained in the instruction register loads the fixed value 0000001_2 on the rising edge of **TCK**.

All test data registers selected by the current instruction retain their previous value during this state. The instruction does not change while the TAP controller is in this state.

Transition to next state: When **TMS** is low on the rising edge of **TCK**, move to Shift-IR, else move to Exit1-IR.

18.2.2.12 Shift-IR State

In this state, the shift register contained in the instruction register is connected between **TDI** and **TDO** and shifts data one bit position nearer to its serial output on each rising edge of **TCK**.

All test data registers selected by the current instruction retain their previous value during this state. The instruction does not change while the TAP controller is in this state.

Transition to next state: When **TMS** is high on the rising edge of **TCK**, move to Exit1-IR, else remain at Shift-IR.

18.2.2.13 Exit1-IR State

This is a temporary state.

All test data registers selected by the current instruction retain their previous value during this state. The instruction does not change while the TAP controller is in this state.

Transition to next state: When **TMS** is high during the next rising edge of **TCK**, the controller enters the Update-IR state and the scanning process terminates. When **TMS** is held low during the next rising edge of **TCK**, the controller enters the Pause-IR state.

18.2.2.14 Pause-IR State

This state allows the TAP controller to temporarily halt the shifting of data through the instruction register.

All test data registers selected by the current instruction retain their previous value during this state. The instruction does not change while the TAP controller is in this state.

Transition to next state: When **TMS** is high on the rising edge of **TCK**, move to Exit2-IR, else remain in the Pause-IR state.

18.2.2.15 Exit2-IR State

This is a temporary state.

All test data registers selected by the current instruction retain their previous value during this state. The instruction does not change while the TAP controller is in this state.

Transition to next state: When **TMS** is held high during the next rising edge of **TCK**, the controller enters the Update-IR state and the scanning process terminates. When **TMS** is held low during the next rising edge of **TCK**, the controller re-enters the Shift-IR state

18.2.2.16 Update-IR State

The instruction shifted into the instruction register is latched onto the parallel output from the shift-register path on the falling edge of **TCK**. Once latched, the new instruction becomes the current instruction.

All test data registers selected by the current instruction retain their previous values in this state.

Transition to next state: When **TMS** remains high on the rising edge of **TCK**, then the controller moves to the Select-DR state, else the controller moves to the Run-Test/Idle state.

18.2.3 TAP Controller Registers

The IEEE 1149.1 architecture specifies a minimum or two test data registers: bypass and boundary scan. The IOP TAP controller extends this number to allow access to test features within the device. Some registers are for public use, others are private. Per the standard, each test data register has a fixed length and can be accessed using one or more instructions. The 81348 design implements the following:

- Instruction Register
- Boundary Scan Register
- Bypass Register
- Device Identification Register

18.2.3.1 Instruction Register

Each of the TAP controllers in the design contain an instruction register (IR). Each IR is a 7-bit, master/slave-configured, parallel-loadable, serial-shift register with latched outputs. They are used in each unit to select the test data register to be accessed, the test to be performed or both.

When the TAP controller is in the **Shift-IR** state, instructions are loaded serially via **TDI** clocked by the rising edge of **TCK**. The shifted-in instruction becomes active upon latching from the serial-stages to the parallel-stages in the **Update-IR** state. At that time the IR outputs, along with the TAP finite state machine outputs, are decoded to select and control the test data register selected by that instruction. Upon latching, all actions caused by any previous instructions must terminate.

On activation of **TRST#**, the latched instruction asynchronously changes to the **IDCODE** instruction. Additionally, upon entering the **Test-Logic-Reset** state, the **ID**CODE instruction is latched and becomes active on the falling edge of **TCK.**

Because the TAP controllers are connected in series, each unit must receive its own 7-bit instruction during the Shift-IR state. The bypass instruction should be loaded into the units that are not to be accessed. When the Intel XScale® Processor Core 1 TAP controller is not MUXed into the TDI-TDO stream, the resulting instruction stream is 14-bits long. When the Intel XScale® Processor Core 1 TAP controller is inserted in the stream, the instruction stream is 21-bits long.

For example, to load the TEST_REG_READ instruction into the TLUs IR when the Intel XScale® Processor Core 1 TAP controller is not muxed into the TDI-TDO stream, the data shifted into TDI during the Shift-IR state is:

0 1 0 0 0 0 1	1 1 1 1 1 1 1	- - - - - - >	TDI
TEST_REG_READ into TLU	BYPASS into X-Scale core 0		

As another example, to load the TEST_REG_READ instruction into the TLUs IR when the Intel XScale® Processor Core 1 TAP controller is muxed into the TDI-TDO stream, the data shifted into TDI during the Shift-IR state is:

0 1 0 0 0 0 1	1 1 1 1 1 1 1	1 1 1 1 1 1 1	- - - - - - >	TDI
TEST_REG_READ into TLU	BYPASS into X-Scale core 0	BYPASS into X-Scale core 1		

18.2.3.2 Instructions

Each of the TAP controller instruction sets is composed of both public and private instructions. Public instructions are intended for use by purchasers of the part. Since the instruction set for each TAP controllers is independent of one another, each is listed separately below

Table 519. TLU TAP Controller Instruction Set

Instruction	Opcode	Public/Private	Description
EXTEST	0000000_2	Public	Intended for supporting the boundary-scan feature for testing device interconnects at the board/system level, the EXTEST instruction connects only the boundary-scan register between TDI and TDO in the Shift-DR state. When EXTEST is selected, all output signal pin values are driven by values shifted into the boundary-scan register and may change only on the falling-edge of **TCK** in the Update_DR state. Also, when extest is selected, all system input pin states must be loaded into the boundary-scan register on the rising-edge of **TCK** in the Capture_DR state. ***Note:*** Data would typically be loaded onto the latched parallel outputs of boundary-scan shift registers using the SAMPLE/PRELOAD instruction prior to selection of the EXTEST instruction.
SAMPLE/ PRELOAD	0000001_2	Public	SAMPLE/PRELOAD performs two functions: • When the TAP controller is in the Capture-DR state, the sample instruction occurs on the rising edge of **TCK** and provides a snapshot of the component's normal operation without interfering with that normal operation. The instruction causes Boundary-Scan register cells associated with outputs to sample the value being driven by or to the processor. • When the TAP controller is in the Update-DR state, the preload instruction occurs on the falling edge of **TCK**. This instruction causes the transfer of data held in the boundary-scan cells to the slave register cells. Typically the slave latched data is then applied to the system outputs by means of the extest instruction.
HIGHZ	0101110_2	Public	HIGHZ puts all output pins into a tri-state mode. When this instruction is active, the bypass register is connected between **TDI** and **TDO**.
CLAMP	0101111_2	Public	Once the clamp instruction is loaded into the TAP controller instruction register, the output pins are driven by the parallel output of the boundary-scan chain. The bypass register is selected as the serial path between **TDI** and **TDO** when the TAP controller passes through the Shift-DR state.
IDCODE	1111110_2	Public	IDCODE is used in conjunction with the device identification register. When selected, IDCODE parallel-loads the hard-wired identification code (32 bits) into the identification register on the rising edge of **TCK** following entry into the Capture-DR state. The instruction selects only the identification register for connection between **TDI** and **TDO** in the Shift-DR state for serial access. ***Note:*** The device identification register is not altered by data being shifted in on **TDI**.
BYPASS	1111111_2	Public	BYPASS selects the bypass register between TDI and TDO while in Shift-DR state, effectively bypassing the processor's test logic. 0 is captured in the Capture-DR state. While this instruction is in effect, all other test data registers have no effect on the operation of the system.

18.2.3.3 Boundary-Scan Register

Boundary-Scan Register is a set of serial-shiftable register cells, connected between each system pin and on-chip system logic. (Power, ground and TAP pins excluded.) This forms a single shift register between TDI and TDO of all the system pins.

This is the most extensive, complex register in the test circuitry. This register allows testing of circuitry external to the component (e.g. board interconnect) in addition to device system logic. It permits the system signals (into and out of the system logic) to be sampled and examined without causing interference with the normal operation of the system logic. Further definition, rules, and specifics of the Boundary-Scan Register can be found in the **IEEE Std, 1149.1-2001, Chapter 10**.

18.2.3.4 Bypass Register

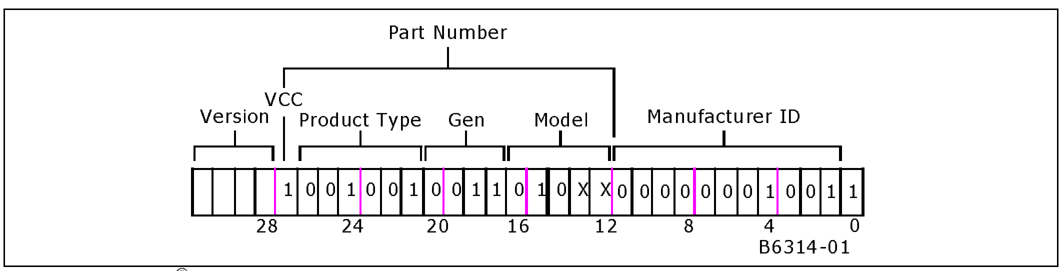
The Bypass Register is a single-bit, serial-shift register connecting **TDI** and **TDO** when the Bypass instruction is in effect. This allows rapid movement of test data to and from other board components, since this register provides the shortest path between **TDI** and **TDO**. This path can be selected when no test operation is being performed. While the Bypass Register is selected, data is transferred from **TDI** to **TDO** without inversion.

18.2.3.5 Device Identification Register

The Device Identification (ID) Register is a 32-bit register used for storing the manufacturer identification, part number, and the version of the processor. It is a dedicated part of the test logic and is not usable in system functionality.

The identification register is selected only by the **IDCODE** instruction. When the **Test-Logic-Reset** state of the TAP controller is entered, the IDCODE instruction is automatically loaded into the instruction register. The generic format of the register is discussed in chapter 11 of the IEEE 1149.1 Standard.

Figure 123. IOP Device ID Register



Note: See Intel® 413808 and 413812 I/O Controllers

Table 520. IOP Device ID Register Field Definitions

Field	Value	Definition
Version	0000 = A-0 Step	Indicates stepping changes.
Manufacturer ID	0000 0001 0011 (Indicates Intel)	Manufacturer ID assigned by IEEE.

Table 521. IOP Device ID Register Settings

Description	Device ID	
	Complete ID (Hex)	Complete ID (Binary)
4138xx A-0 (ATU-E)	0x3361	0011001101100001b
4138xx A-0 (ATU-XE)	0x3369	0011001101101001b

18.3 Definition of Terms

High-Z:	An instruction defined by the IEEE 1149.1 Standard. The requirement for this instruction are 1) all system logic outputs are high impedance; 2) the TAP controller continues to operate with the bypass register connected between TDI and TDO. The part may have other settings, as long as they do not interfere with these two requirements. Following the use of High-Z the part may be in an indeterminate state and require a reset.
ONCE:	A test mode used to test static-Icc and pin leakage. The requirements for this mode are 1) all system logic outputs are high impedance, including TDO; 2) all pull-ups are disables, including **TMS**, **TRST#**, and **TDI**; 3) all clocks are stopped. The TAP controller cannot be operational in this mode.

19.0 Peripheral Registers

This chapter summarizes the registers for the integrated peripherals. Each register is defined in detail in the corresponding unit chapter.

19.1 Overview

Intel® 413808 and 413812 I/O Controllers in TPER Mode (4138xx) Peripheral Registers can be accessed via three methods:

- **PCI Configuration Register interface:** Is supported by the PCI interface and PCI Configuration Cycle transaction type.
- **Peripheral Memory-Mapped Register (PMMR) interface:** Gives software ability to read and modify internal control registers. These registers are accessed as a memory-mapped 32-bit register with unique memory address. Access is accomplished through regular Intel XScale® processor memory-format instruction.
- **High-performance** Intel XScale® processor **(ARM* architecture compliant) Coprocessor Register interface (CCR):** Gives software ability to read and modify internal control registers at very low latency as compared to PMMR interface.

These registers are specific to the 4138xx only. They support the:

- I/O Level Control
- Internal Bus Bridge Unit
- SRAM Memory Controller
- I^2C Bus Interface Units
- Interrupt Controller Unit
- Messaging Unit
- System Controllers
- UART Units
- Address Translation Unit (PCI-X and PCI-E)
- SRAM DMA Controller Unit
- Peripheral Performance Monitoring Unit
- General Purpose I/O Unit
- Peripheral Bus Interface Unit
- Intel XScale® processor Bus Interface Unit

Each of these peripherals fully describe the independent functionality of the registers, control and usage.

Control and status registers for the Intel XScale® processor use the CCR interface. Accesses to coprocessor registers do not generate external bus cycles. See the *Intel® 80200 Processor based on Intel® XScale™ Microarchitecture Developer's Manual* (Order Number: 273411) for a full description of the usage of the coprocessor register space. For completeness, these registers are included in the tables of registers for the CCR interface. These registers can only be accessed using the coprocessor instructions.

The PMMR interface provides full accessibility from the ATU, and the Intel XScale® processor. The PMMR block can be mapped to any 512-KByte aligned address boundary using the PMMR Base Address registers. The default starting address of the PMMR block is 0 FFD8 0000H and cannot be changed.

19.2 Accessing Peripheral Memory-Mapped Registers

The PMMR interface is a slave device connected to the 4138xx internal bus. This interface accepts data transactions which appear on the internal bus from the ATU and the Intel XScale® processors.

The PMMR interface allows these devices to perform read, write, or read-modify-write transactions. The specific actions taken when modifying any value in the PMMR space is independently defined within each chapter which describes the functionality of the register.

Note: The PMMR interface does not support multi-word burst accesses from any internal bus master.

All PMMR transactions are allowed from the Intel XScale® processors operating in user or supervisor mode. In addition, the PMMR does not provide any access exception to the Intel XScale® processor.

19.3 Accessing Peripheral Registers Using the Core Coprocessor Register Interface

Registers may be accessed/manipulated through the CCR interface with the MCR, MRC, STC, and LDC instructions. The *CRn* field of the instruction denotes the register number to be accessed. The *opcode_1*,and *opcode_2* fields of the instruction should be zero. The *CRm* field must be set to 0 for the Interrupt Controller Unit and to 1 for the Programmable Timers. Most systems restrict access to coprocessor registers to privileged processes. To control access to a coprocessor register, use the Coprocessor Access Register as described in the *ARM Architecture Reference Manual*.

19.4 Architecturally Reserved Memory Space

The 4138xx provides 64 GBytes of address space. Portions of this address space is architecturally reserved and users are restricted as to their function. Figure 124 shows the reserved address space.

Addresses 0 0000 0000H through 0 0000 001FH are reserved for the exception vectors of the Intel XScale® processor.

Addresses 0 FFFF 0000H through 0 FFFF 001FH are reserved for the relocated exception vectors of the Intel XScale® processor.

Addresses 0 FFD8 0000H through 0 FFDF FFFFH are allocated to the PMMR interface by default. These registers cannot be relocated for the 4138xx in any mode.

19.5 Default Memory Space Setup

Figure 124 shows the Intel XScale® processor address space and addresses available to the applications. Figure 124 also shows the default locations of certain resources after reset.

The PMMR Block occupies 512 KBytes of space and is located at addresses 0 FFD8 0000H through 0 FFDF FFFFH. Do not change the default address of the PMMR block.

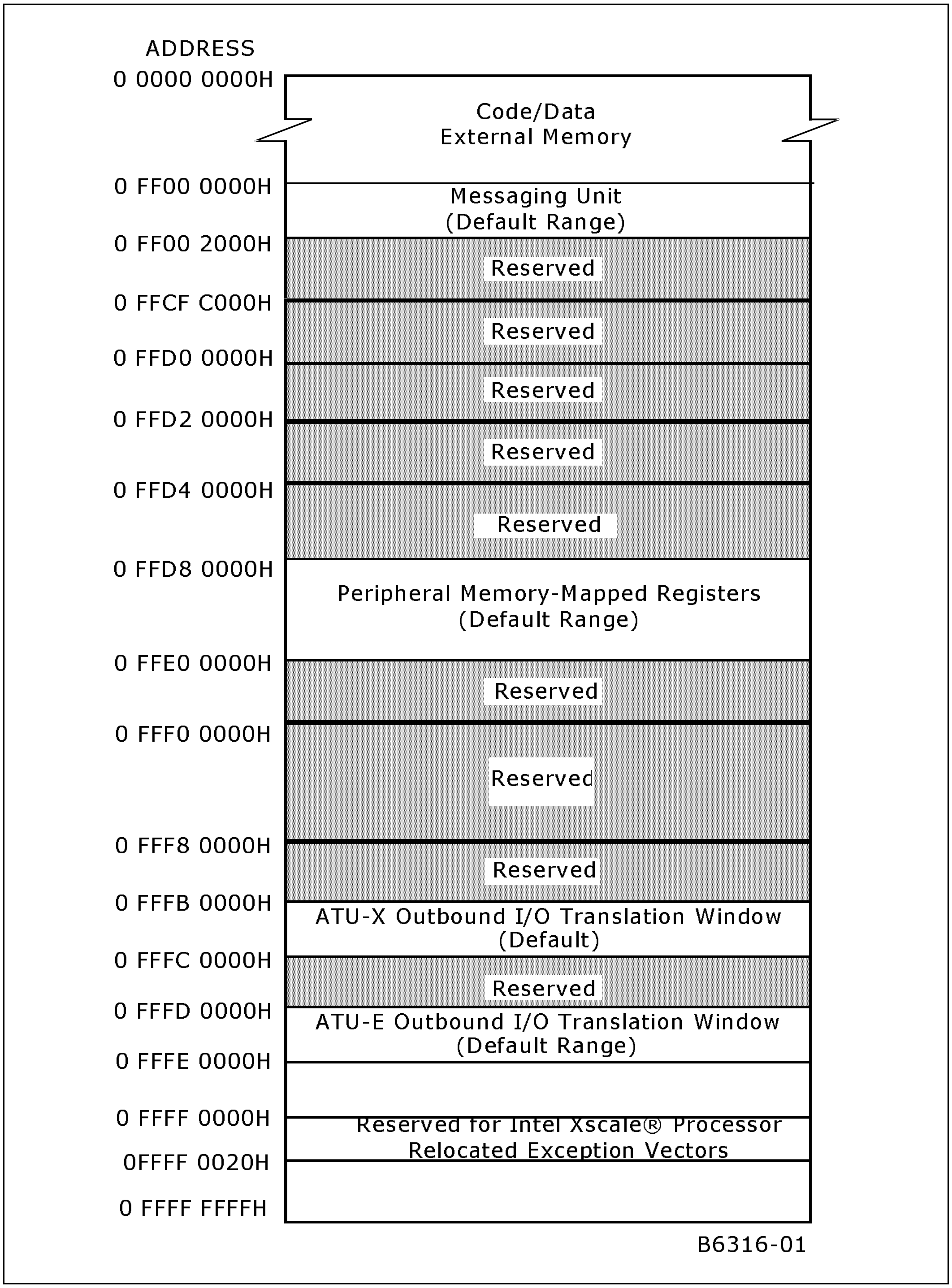
The Messaging Unit occupies 8 KBytes of space and is located at addresses 0 FF00 0000H through 0 FF00 1FFFH.

The ATU-X Outbound I/O Translation Window occupies 64 KBytes of space and is located at addresses 0 FFFB 0000H through 0 FFFB FFFFH.

The ATU-E Outbound I/O Translation Window occupies 64 KBytes of space and is located at addresses 0 FFFD 0000H through 0 FFFD FFFFH.

For a full description of the address space for the Intel XScale® processor Reset and Exception Vectors, refer to the *ARM Architecture Reference Manual*.

Figure 124. Intel® 413808 and 413812 I/O Controllers in TPER Mode Memory Address Space



19.6 Peripheral Memory-Mapped Register Address Space

The PMMR address space is divided to support the integrated peripherals on the 4138xx. Table 523 shows all of the 4138xx integrated peripheral memory-mapped registers and their internal bus address offsets. The starting address of the memory-mapped registers is programmable using the PMMRBAR register. The PMMRBAR register is located at a fixed location which is at F_FFFF_FFFCH. The default starting address of the PMMR register block is at 0_FFD8_0000H.

Table 523 lists the peripherals and the relative address offsets from the PMMRBAR content.

Warning: Care must be exercised when updating memory-mapped registers, because some of the registers can have side-effects on other registers and resources in the 4138xx architecture. As an example, updating a Base Address Register (such as the PMMRBAR) relocates other registers and resources in the 4138xx architecture, and software must not access these related registers and resources until software can ensure that the Base Address Register has truly been updated. Refer to the Intel XScale® Microarchitecture *Developer's Manual*, which describes various methods of fencing memory accesses.

Table 522. PMMR Base Address Register (PMMRBAR) Default Value

Register	Absolute Address
PMMRBAR	F_FFFF_FFFCH

Table 523. Local Addresses for Integrated Peripherals (Sheet 1 of 3)

Integrated Peripheral	Internal Address Offset (Relative to PMMRBAR)	Space Allocated
Reserved.	+0000H through 01FFH	512 Bytes
Reserved.	+0200H through 03FFH	512 Bytes
Reserved.	+0400H through 05FFH	512 Bytes
Reserved.	+0600H through 07FFH	512 Bytes
Not Claimed by any Unit[a]	+0800H through 09FFH	x
Reserved.	+0A00H through 0CFFH	768 Bytes
Not Claimed by any Unit	+0D00H through 14FFH	x
Reserved.	+1500H through 157FH	128 Bytes
"Peripheral Bus Interface Unit"	+1580H through 15FFH	128 Bytes
Not Claimed by any Unit	+1600H through 163FH	x
"System Controller Unit"	+1640H through 164FH	16 Bytes
Not Claimed by any Unit	+1650H through 167FH	32 Bytes
Not Claimed by any Unit	+1680H through 16FFH	x
Not Claimed by any Unit	+1700H through 177FH	x
"Internal Bus Bridge"	+1780H through 1797H	24 Bytes
Not Claimed by any Unit	+1798H through 177FH	104 Bytes
Reserved.	+1800H through 19FFH	512 Bytes
Not Claimed by any Unit	+1A00H through 1FFFH	x
"I/O Pad Control Unit"	+2000H through 21FFH	512 Bytes
Reserved	+2200H through 221FH	32 Bytes
Not Claimed by any Unit	+2220H through 22FFH	x
"UART" 0	+2300H through 233FH	64 Bytes
"UART" 1	+2340H through 237FH	64 Bytes
Not Claimed by any Unit	+2380H through 247FH	x

Table 523. Local Addresses for Integrated Peripherals (Sheet 2 of 3)

Integrated Peripheral	Internal Address Offset (Relative to PMMRBAR)	Space Allocated
"GPIO"	+2480H through 24BFH	64 Bytes
Not Claimed by any Unit	+24C0H through 24FFH	x
"I2C Unit" 0	+2500H through 251FH	32 Bytes
"I2C Unit" 1	+2520H through 253FH	32 Bytes
"I2C Unit" 2	+2540H through 255FH	32 Bytes
Reserved.	+2560H through 257FH	32 Bytes
Not Claimed by any Unit	+2580H through 25FFH	x
Reserved	+2600H through 267FH	128 Bytes
Reserved	+2680H through 26FFH	128 Bytes
Not Claimed by any Unit	+2700H through 3FFFH	x
"Messaging Unit"	+4000H through 5FFFH	8 KBytes
Not Claimed	+6000H through 7FFFH	x
Reserved.	+1 8000H through 1 81FFH	512 Bytes
Reserved.	+1 8200H through 1 83FFH	512 Bytes
Not Claimed by any Unit	+1 8400H through 1 9FFFH	x
"PMON Unit"	+1 A000H through 1 BFFFH	8 KBytes
Not Claimed by any Unit	+1 C000H through 3 FFFFH	x
PCI Function 0 Configuration Registers (PCI Attributes)	+4 0000H through 4 0FFFH	4 KBytes
PCI Function 1 Configuration Registers (PCI Attributes) Reserved.	+4 1000H through 4 1FFFH	4 KBytes
PCI Function 2 Configuration Registers (PCI Attributes) Reserved.	+4 2000H through 4 2FFFH	4 KBytes
PCI Function 3 Configuration Registers (PCI Attributes) Reserved.	+4 3000H through 4 3FFFH	4 KBytes
PCI Function 4 Configuration Registers (PCI Attributes) Reserved.	+4 4000H through 4 4FFFH	4 KBytes
PCI Function 5 Configuration Registers (PCI Attributes) Reserved.	+4 5000H through 4 5FFFH	4 KBytes
PCI Function 6 Configuration Registers (PCI Attributes) Reserved	+4 6000H through 4 6FFFH	4 KBytes
Not Claimed by any Unit	+4 7000H through 4 7FFFH	4 KBytes
PCI Function 0 Configuration Registers (Local Attributes)	+4 8000H through 4 8FFFH	4 KBytes
PCI Function 1 Configuration Registers (Local Attributes) Reserved.	+4 9000H through 4 9FFFH	4 KBytes
PCI Function 2 Configuration Registers (Local Attributes) Reserved.	+4 A000H through 4 AFFFH	4 KBytes
PCI Function 3 Configuration Registers (Local Attributes) Reserved.	+4 B000H through 4 BFFFH	4 KBytes
PCI Function 4 Configuration Registers (Local Attributes) Reserved.	+4 C000H through 4 CFFFH	4 KBytes
PCI Function 5 Configuration Registers (Local Attributes) Reserved.	+4 D000H through 4 DFFFH	4 KBytes
PCI Function 6 Configuration Registers (Local Attributes) Reserved	+4 E000H through 4 EFFFH	4 KBytes
Not Claimed by any Unit	+4 F000H through 4 FFFFH	4 KBytes
Reserved	+5 0000H through 5 03FFH	1 KByte
Reserved	+5 0400H through 5 07FFH	1 KByte
Reserved	+5 0800H through 5 0BFFH	1 KByte
Reserved	+5 0C00H through 5 0FFFH	1 KByte
Not Claimed by any Unit	+5 1000H through 5 13FFH	1 KByte
Not Claimed by any Unit	+5 1400H through 5 17FFH	1 KByte
Not Claimed by any Unit	+5 1800H through 5 1BFFH	1 KByte
Not Claimed by any Unit	+5 1C00H through 5 1FFFH	1 KByte
Reserved	+5 2000H through 5 23FFH	1 KByte
Reserved	+5 2400H through 5 27FFH	1 KByte
Reserved	+5 2800H through 5 2BFFH	1 KByte
Reserved	+5 2C00H through 5 2FFFH	1 KByte
Not Claimed by any Unit	+5 3000H through 5 33FFH	1 KByte
Not Claimed by any Unit	+5 3400H through 5 37FFH	1 KByte

Table 523. Local Addresses for Integrated Peripherals (Sheet 3 of 3)

Integrated Peripheral	Internal Address Offset (Relative to PMMRBAR)	Space Allocated
Not Claimed by any Unit	+5 3800H through 5 3BFFH	1 KByte
Not Claimed by any Unit	+5 3C00H through 5 3FFFH	1 KByte
Reserved.	+6 0000H through 6 1FFFH	8 KBytes
Reserved.	+6 2000H through 6 3FFFH	8 KBytes
Reserved.	+6 4000H through 6 5FFFH	8 KBytes
Reserved.	+6 6000H through 6 7FFFH	8 KBytes
Not Claimed by any Unit	+6 8000H through 6 9FFFH	8 KBytes
Not Claimed by any Unit	+6 A000H through 6 BFFFH	8 KBytes
Not Claimed by any Unit	+6 C000H through 6 DFFFH	8 KBytes
Not Claimed by any Unit	+6 E000H through 6 FFFFH	8 KBytes
Reserved.	+7 0000H through 7 1FFFH	8 KBytes
Reserved.	+7 2000H through 7 3FFFH	8 KBytes
Reserved.	+7 4000H through 7 5FFFH	8 KBytes
Reserved.	+7 6000H through 7 7FFFH	8 KBytes
Not Claimed by any Unit	+7 8000H through 7 9FFFH	8 KBytes
Not Claimed by any Unit	+7 A000H through 7 BFFFH	8 KBytes
Not Claimed by any Unit	+7 C000H through 7 DFFFH	8 KBytes
Not Claimed by any Unit	+7 E000H through 7 FFFFH	8 KBytes
Not Claimed by any Unit	+8 0000H through 8 FFFFH	x

a. Address Ranges that are defined as "Not Claimed by any Unit" Master Aborts when accessed.

Memory-mapped registers also accessible via PCI configuration transactions are:

- Address Translation Unit (PCI-X)
- Address Translation Unit (PCI-E)
- Messaging Unit (MU)

Registers which must have address translation logic configured to translate PCI addresses into the Intel XScale® processor address space, to access the memory-mapped registers from the PCI Express interface are:

- Application SRAM Controller
- Address Translation Unit (PCI-E)
- SRAM Memory Controller
- I^2C Bus Interface Unit
- Messaging Unit
- System Controllers
- Peripheral Bus Interface Unit
- Performance Monitoring Unit
- Interrupt Controller Unit
- General Purpose I/O Unit
- Interface Pad Control Registers
- Third Party Messaging Unit
- Inter-Processor Messaging Unit
- UARTs

The following sections describe the register map for the internal units of the 4138xx. See the relevant unit chapters for detailed description of register function.

19.6.1 Internal Units

19.6.1.1 Peripheral Bus Interface Unit

The Peripheral Bus Interface Unit (PBI) is allocated 128 Bytes of PMMR registers space and is always located at offset +1580H relative to the PMMRBAR.

Use the following equation to calculate the actual register address:

Internal Bus Address = PMMRBAR + PBI Base Address Offset + Register Offset.

Note: Additionally, GPIO[8:0] I/O pad control registers are located in the "I/O Pad Control Unit" registers block.

Table 524. PBI Base Address Offset.

Unit	PBI Base Address Offset (Relative to PMMRBAR)
PBI	+1580H

Table 525. Peripheral Bus Interface Unit

Register Description (Name)	Register Size in Bits	Internal Bus Address Offset (Relative to PBI Base Address Offset)
PBI Control Register — PBCR	32	+00H
PBI Status Register — PBISR	32	+04H
PBI Base Address Register 0 — PBBAR0	32	+08H
PBI Limit Register 0 — PBLR0	32	+0CH
PBI Base Address Register 1 — PBBAR1	32	+10H
PBI Limit Register 1 — PBLR1	32	+14H
Reserved	x	+18H through +7FH

19.6.1.2 System Controller

The System Controller Unit (SC) is allocated 16 Bytes of PMMR register space and is always located at offset +1640H relative to the PMMRBAR.

Use the following equation to calculate the actual register address:

Internal Bus Address = PMMRBAR + SC Base Address Offset + Register Offset.

Table 526. SC Base Address Offset.

Unit	SC Base Address Offset (Relative to PMMRBAR)
System Controller	+1640H

Table 527. System Controller Unit

Register Description (Name)	Register Size in Bits	Internal Bus Address Offset (Relative to SC Base Address Offset)
Internal Bus Arbitration Control Register — IBACR	32	+00H
South Internal Bus Address Test Register — SIBATCR	32	+04H
South Internal Bus Data Test Register — SIBDTCR	32	+08H
Reserved	x	+0CH through 0FH
Peripheral Memory-Mapped Register Base Address Register (PMMRBAR)	32	F FFFF FFFCH (absolute address)

19.6.1.3 Internal Bus Bridge

The Internal Bus Bridge is allocated 24 Bytes of PMMR register space and is always located at offset +1780H relative to the PMMRBAR.

Use the following equation to calculate the actual register address:

Internal Bus Address = PMMRBAR + Internal Bus Bridge Address Offset + Register Offset.

Table 528. Internal Bus Bridge Base Address Offset.

Unit	Internal Bus Bridge Base Address Offset (Relative to PMMRBAR)
Internal Bus Bridge	+1780H

Table 529. Internal Bus Bridge

Register Description (Name)	Register Size in Bits	Internal Bus Address Offset (Relative to Internal Bus Bridge Base Address Offset)
Bridge Window Base Address Register — BWBAR	32	+00H
Bridge Window Upper Base Address Register — BWUBAR	32	+04H
Bridge Window Limit Register — BWLR	32	+08H
Bridge Error Status Register — BECSR	32	+0CH
Bridge Error Address Register — BERAR	32	+10H
Bridge Error Upper Address Register — BERUAR	32	+14H

19.6.1.4 I/O Pad Control

The I/O Pad Control is allocated 512 Bytes of PMMR registers space and is always located at offset +2000H relative to the PMMRBAR.

Use the following equation to calculate the actual register address:

Internal Bus Address = PMMRBAR + I/O Pad Control Base Address Offset + Register Offset.

Note: The PBI drive strength register is described in the Peripheral Bus Interface chapter.

Table 530. I/O Pad Control Base Address Offset.

Unit	Associated Unit Interface	I/O Pad Control Base Address Offset (Relative to PMMRBAR)
I/O Pad Control	Reserved	+2000H
	Peripheral Bus Interface	+2080H
	PCI Interface	+2100H
	Other Units	+2180H

Table 531. I/O Pad Control Unit

Unit	Register Description (Name)	Register Size in Bits	Internal Bus Address Offset (Relative to I/O Pad Control Base Address Offset)
Peripheral Bus Interface	PBI Drive Strength Control Register — PBDSCR	32	+00H
	Reserved	x	+04H through +7FH
PCI Interface	PCIX RCOMP Control Register — PRCR	32	+00H
	PCIX Pad ODT Drive Strength Manual Override Values Registers — PPODSMOVR	32	+04H
	PCIX PAD DRIVE STRENGTH manual override values register (3.3V/1.5V switch supply voltage) — PPDSMOVR3.3_1.5	32	+08H
	PCIX PAD DRIVE STRENGTH manual override values register(3.3V dedicated supply voltage) — PPDSMOVR3.3	32	+0CH
	Reserved	x	+10H through +7FH
Other Units[1]	Reserved	32	+00H
	Reserved	32	+04H
	Reserved	32	+08H
	Reserved	32	+0CH
	Reserved	32	+10H
	Unique ID Register 0 — UID0	32	+14H
	Unique ID Register 1 — UID1	32	+18H
	Reserved	x	+1CH through +7FH

Notes:
1. Registers that belong in this group are documented in the Peripheral Bus Interface Unit chapter.

19.6.1.5 UART 0-1

The 4138xx contains two instances of the UART. Each UART is allocated 64Bytes of PMMR registers space that is located at the offset specified in Table 532 which is relative to the PMMRBAR.

Use the following equation to calculate the actual register address:

Internal Bus Address = PMMRBAR + UART Base Address Offset + Register Offset.

Table 532. UART 0-1 Offset.

Unit	UARTx Base Address Offset (Relative to PMMRBAR)
UART 0	+2300H
UART 1	+2340H

Table 533. UART

Register Description (Name)	Register Size in Bits	Internal Bus Address Offset (Relative to UARTx Base Address Offset)
UART x Receive Buffer Register (Read Only) (DLAB=0)	32	+00H
UART x Transmit Holding Register (Write Only) (DLAB=0)	32	
UART x baud Divisor Latch Low byte (DLAB=1)	8	
UART x Interrupt Enable Register (DLAB=0)	8	+04H
UART x baud Divisor Latch High byte (DLAB=1)	8	
UART x Interrupt ID Register (Read Only)	8	+08H
UART x FIFO Control Register (Write Only)	8	
UART x Line Control Register	8	+0CH
UART x Modem Control Register	8	+10H
UART x Line Status Register	8	+14H
UART x Modem Status Register	8	+18H
UART x Scratch Pad Register	8	+1CH
Reserved	32	+20H
UART x FIFO Occupancy Register	8	+24H
UART x Autobaud Control Register	8	+28H
UART x Autobaud Count Register	16	+2CH
Reserved		+30H through +3FH

19.6.1.6 GPIO

The GPIO block is allocated 64Bytes of PMMR registers space that is located at the offset specified in Table 535 which is relative to the PMMRBAR.

Use the following equation to calculate the actual register address:

Internal Bus Address = PMMRBAR + GPIO Base Address Offset + Register Offset.

Table 534. GPIO Offset.

Unit	GPIO Base Address Offset Relative to PMMRBAR)
GPIO	+2480H

Table 535. GPIO

Register Description (Name)	Register Size in Bits	Internal Bus Address Offset (Relative to GPIO Base Address Offset)
GPIO Output Enable Register — GPOE	32	+00H
GPIO Input Data Register — GPID	32	+04H
GPIO Output Data Register — GPOD	32	+08H
Reserved	x	+0CH–3FH

19.6.1.7 I^{2}C Bus Interface Unit 0-2

The 4138xx contains three instances of the I^{2}C Unit which are each allocated 32 Bytes of PMMR registers space located at the offset specified in Table 537.

Use the following equation to calculate the actual register address:

Internal Bus Address = PMMRBAR + I^{2}C Base Address Offset + Register Offset.

Table 536. I^{2}C 0-2 Offset.

Unit	I^{2}C Base Address Offset (Relative to PMMRBAR)
I^{2}C 0	+2500H
I^{2}C 1	+2520H
I^{2}C 2	+2540H

Table 537. I^{2}C Unit

Register Description (Name)	Register Size in Bits	Internal Bus Address Offset (Relative to I^{2}C Base Address Offset)
I^{2}C Control Register x — ICRx	32	+00H
I^{2}C Status Register x — ISRx	32	+04H
I^{2}C Slave Address Register x — ISARx	32	+08H
I^{2}C Data Buffer Register x — IBDRx	32	+0CH
Reserved	32	+10H
I^{2}C Bus Monitor Register x — IBMRx	32	+14H
I^{2}C Manual Bus Control Register x — IMBCRx	32	+18H
Reserved	32	+1CH

19.6.1.8 Messaging Unit

The Messaging Unit (MU) is allocated 8 KBytes of PMMR registers space that is located at the offset specified in Table 538 which is relative to the PMMRBAR.

Use the following equation to calculate the actual register address:

Internal Bus Address = PMMRBAR + MU Base Address Offset + Register Offset.

Table 538. Messaging Unit Offset.

Unit	MU Base Address Offset (Relative to PMMRBAR)
Messaging Unit	+4000H

Table 539. Messaging Unit (Sheet 1 of 2)

Register Description (Name)	Bits	Internal Bus Address Offset[a]
Reserved	x	+000H — +00C
Inbound Message Register — IMR0	32	+010H
Inbound Message Register — IMR1	32	+014H
Outbound Message Register — OMR0	32	+018H
Outbound Message Register — OMR1	32	+01CH
Inbound Doorbell Register — IDR	32	+020H
Inbound Interrupt Status Register — IISR	32	+024H
Inbound Interrupt Mask Register — IIMR	32	+028H
Outbound Doorbell Register — ODR	32	+02CH
Outbound Interrupt Status Register — OISR	32	+030H
Outbound Interrupt Mask Register — OIMR	32	+034H
Inbound Reset Control and Status Register — IRCSR	32	+038H
Outbound Reset Control and Status Register — ORCSR	32	+03CH
Reserved	x	+040H — +047H
MSI Inbound Message Register	32	+048H
Reserved	x	+04CH
MU Configuration Register — MUCR	32	+050H
Reserved	32	+054H
Reserved	32	+058H
Reserved	32	+05CH
Reserved	32	+060H
Reserved	32	+064H
Reserved	32	+068H
Reserved	32	+06CH
Reserved	32	+070H
Reserved	32	+074H
Reserved	32	+078H
Reserved	32	+07CH
Reserved	32	+080H
MU Base Address Register — MUBAR	32	+084H
MU Upper Base Address Register — MUUBAR	32	+088H
Reserved	x	+08CH — +0FFH

Table 539. Messaging Unit (Sheet 2 of 2)

Register Description (Name)	Bits	Internal Bus Address Offset[a]
MU MSI-X Table Message Address Register 0 — M_MT_MAR0	32	+1000
MU MSI-X Table Message Upper Address Register 0 — M_MT_MUAR0	32	+1004
MU MSI-X Table Message Data Register 0 — M_MT_MDR0	32	+1008
MU MSI-X Table Message Vector Control Register 0 — M_MT_MVCR0	32	+100C
MU MSI-X Table Message Address Register 1 — M_MT_MAR1	32	+1010
MU MSI-X Table Message Upper Address Register 1 — M_MT_MUAR1	32	+1014
MU MSI-X Table Message Data Register 1 — M_MT_MDR1	32	+1018
MU MSI-X Table Message Vector Control Register 1 — M_MT_MVCR1	32	+101C
MU MSI-X Table Message Address Register 2 — M_MT_MAR2	32	+1020
MU MSI-X Table Message Upper Address Register 2 — M_MT_MUAR2	32	+1024
MU MSI-X Table Message Data Register 2 — M_MT_MDR2	32	+1028
MU MSI-X Table Message Vector Control Register 2 — M_MT_MVCR2	32	+102C
MU MSI-X Table Message Address Register 3 — M_MT_MAR3	32	+1030
MU MSI-X Table Message Upper Address Register 3 — M_MT_MUAR3	32	+1034
MU MSI-X Table Message Data Register 3 — M_MT_MDR3	32	+1038
MU MSI-X Table Message Vector Control Register 3 — M_MT_MVCR3	32	+103C
MU MSI-X Table Message Address Register 4 — M_MT_MAR4	32	+1040
MU MSI-X Table Message Upper Address Register 4 — M_MT_MUAR4	32	+1044
MU MSI-X Table Message Data Register 4 — M_MT_MDR4	32	+1048
MU MSI-X Table Message Vector Control Register 4 — M_MT_MVCR4	32	+104C
MU MSI-X Table Message Address Register 5 — M_MT_MAR5	32	+1050
MU MSI-X Table Message Upper Address Register 5 — M_MT_MUAR5	32	+1054
MU MSI-X Table Message Data Register 5 — M_MT_MDR5	32	+1058
MU MSI-X Table Message Vector Control Register 5 — M_MT_MVCR5	32	+105C
MU MSI-X Table Message Address Register 6 — M_MT_MAR6	32	+1060
MU MSI-X Table Message Upper Address Register 6 — M_MT_MUAR6	32	+1064
MU MSI-X Table Message Data Register 6 — M_MT_MDR6	32	+1068
MU MSI-X Table Message Vector Control Register 6 — M_MT_MVCR6	32	+106C
MU MSI-X Table Message Address Register 7 — M_MT_MAR7	32	+1070
MU MSI-X Table Message Upper Address Register 7 — M_MT_MUAR7	32	+1074
MU MSI-X Table Message Data Register 7 — M_MT_MDR7	32	+1078
MU MSI-X Table Message Vector Control Register 7 — M_MT_MVCR7	32	+107C
Reserved	x	+1080 — +17FF
MU MSI-X Pending Bits Array Register	32	+1800
Reserved	x	+1804 — +1FFF

a. Relative to MU Base Address Offset.

19.6.1.9 PMON Unit

The **PMON** Unit (**PMON**) is allocated 8 KBytes of PMMR registers space that is located at the offset specified in Table 540 which is relative to the PMMRBAR. Use the following equation to calculate the actual register address:

Internal Bus Address = PMMRBAR + **PMON** Base Address Offset + Register Offset.

Table 540. PMON Unit Base Address Offset.

Unit	PMON Base Address Offset (Relative to PMMRBAR)
PMON Unit	+1_A000H

Table 541. PMON Unit

Register Description (Name)	Register Size (Bits)	Internal Bus Address Offset (Relative to PMON Base Address Offset)
PMON Command Register 0 — **PMON**_CMD0	32	+00H
PMON Event Register 0 — **PMON**_EVR0	32	+04H
PMON Status Register 0 — **PMON**_STS0	32	+08H
PMON DATA Register 0 — **PMON**_DATA0	32	+0CH
PMON Command Register 1 — **PMON**_CMD1	32	+10H
PMON Event Register 1 — **PMON**_EVR1	32	+14H
PMON Status Register 1 — **PMON**_STS1	32	+18H
PMON DATA Register 1 — **PMON**_DATA1	32	+1CH
PMON Command Register 2 — **PMON**_CMD2	32	+20H
PMON Event Register 2 — **PMON**_EVR2	32	+24H
PMON Status Register 2 — **PMON**_STS2	32	+28H
PMON DATA Register 2 — **PMON**_DATA2	32	+2CH
PMON Command Register 3 — **PMON**_CMD3	32	+30H
PMON Event Register 3 — **PMON**_EVR3	32	+34H
PMON Status Register 3 — **PMON**_STS3	32	+38H
PMON DATA Register 3 — **PMON**_DATA3	32	+3CH
PMON Command Register 4 — **PMON**_CMD4	32	+40H
PMON Event Register 4 — **PMON**_EVR4	32	+44H
PMON Status Register 4 — **PMON**_STS4	32	+48H
PMON DATA Register 4 — **PMON**_DATA4	32	+4CH
PMON Command Register 5 — **PMON**_CMD5	32	+50H
PMON Event Register 5 — **PMON**_EVR5	32	+54H
PMON Status Register 5 — **PMON**_STS5	32	+58H
PMON DATA Register 5 — **PMON**_DATA5	32	+5CH
PMON Command Register 6 — **PMON**_CMD6	32	+60H
PMON Event Register 6 — **PMON**_EVR6	32	+64H
PMON Status Register 6 — **PMON**_STS6	32	+68H
PMON DATA Register 6 — **PMON**_DATA6	32	+6CH
PMON Command Register 7 — **PMON**_CMD7	32	+70H
PMON Event Register 7 — **PMON**_EVR7	32	+74H
PMON Status Register 7 — **PMON**_STS7	32	+78H
PMON DATA Register 7 — **PMON**_DATA7	32	+7CH

19.6.2 Host Interface Units

This section describes the register layout of the units that are visible as PCI functions. These units include the TPMI0-3, ATUE and ATUX. The PCI Function number for each of these units vary based on strapping options that are sampled during reset.

The PCI Function number associated with each unit and its Base Address Offset are detailed in Table 542, "PCI Function MMR Locations"

Table 542. PCI Function MMR Locations

PCI Function Number	CONTROLLER_ONLY# = 0 (has priority over INTERFACE_SEL_PCIX#)	INTERFACE_SEL_PCIX# =		Internal Bus Address Offset (Relative to PMMRBAR)	
		0	1	PCI Attributes	Local Attributes
0	Reserved	ATUX	ATUE	+4 0000H	+4 8000H
1	Reserved	Reserved	Reserved	+4 1000H	+4 9000H
2	Reserved	Reserved	Reserved	+4 2000H	+4 A000H
3	Reserved	Reserved	Reserved	+4 3000H	+4 B000H
4	ATUX	Reserved	Reserved	+4 4000H	+4 C000H
5	ATUE	ATUE	ATUX	+4 5000H	+4 D000H
6	Reserved	Reserved	Reserved	+4 6000H	+4 E000H
7	Reserved	Reserved	Reserved	+4 7000H	+4 F000H

19.6.2.1 Address Translation Unit (PCI-X)

A subset of the ATU registers are accessible through both inbound PCI configuration cycles and the 4138xx core CPU (Register offsets 000H through 0FFH). The balance of the registers are accessible only via the internal bus.

The Internal Bus Address Offset to PMMRBAR of any ATU Register can be derived by adding the 4 KB address aligned Internal Bus Memory Mapped Register Range Offset (Table 543, "Intel® 413808 and 413812 I/O Controllers ATUX Configuration Space Base Address Offset" on page 810) to the Register Offset (Table 544, "Address Translation Unit Registers — ATUX" on page 811)

For example, when INTERFACE_SEL_PCIX# and CONTROLLER_ONLY# are bothasserted, the offset to PMMRBAR of the ATU Command Register would be (4 C000H+004H) or 4 C004H.

Note: The 4 KB Address Aligned Range Offset can be different depending on two configuration straps as described in Table 543.

Table 543. Intel® 413808 and 413812 I/O Controllers ATUX Configuration Space Base Address Offset

CONTROLLER_ONLY#	INTERFACE_SEL_PCIX#	ATUX Base Address Offset (Relative to PMMRBAR)
PCI Attributes		
Deasserted	Asserted	+4 0000H
Asserted	Deasserted	+4 4000H
Asserted	Asserted	+4 4000H
Deasserted	Deasserted	+4 5000H
Local Attributes		
Deasserted	Asserted	+4 8000H
Asserted	Deasserted	+4 C000H
Asserted	Asserted	+4 C000H
Deasserted	Deasserted	+4 D000H

Table 544. Address Translation Unit Registers — ATUX (Sheet 1 of 3)

Register Description (Name)	Register Size in Bits	Internal Bus Address Offset (Relative to ATUX Base Address Offset)
ATU Vendor ID Register — ATUVID	16	+000H
ATU Device ID Register — ATUDID	16	+002H
ATU Command Register — ATUCMD	16	+004H
ATU Status Register — ATUSR	16	+006H
ATU Revision ID Register — ATURID	8	+008H
ATU Class Code Register — ATUCCR	24	+009H
ATU Cacheline Size Register — ATUCLSR	8	+00CH
ATU Latency Timer Register — ATULT	8	+00DH
ATU Header Type Register — ATUHTR	8	+00EH
ATU BIST Register — ATUBISTR	8	+00FH
Inbound ATU Base Address Register 0 — IABAR0	32	+010H
Inbound ATU Upper Base Address Register 0 — IAUBAR0	32	+014H
Inbound ATU Base Address Register 1 — IABAR1	32	+018H
Inbound ATU Upper Base Address Register 1 — IAUBAR1	32	+01CH
Inbound ATU Base Address Register 2 — IABAR2	32	+020H
Inbound ATU Upper Base Address Register 2 — IAUBAR2	32	+024H
Reserved	32	+028H
ATU Subsystem Vendor ID Register — ASVIR	16	+02CH
ATU Subsystem ID Register — ASIR	16	+02EH
Expansion ROM Base Address Register — ERBAR	32	+030H
ATU Capabilities Pointer Register — ATU_Cap_Ptr	8	+034H
Reserved	24	+035H
Reserved	32	+038H
ATU Interrupt Line Register — ATUILR	8	+03CH
ATU Interrupt Pin Register — ATUIPR	8	+03DH
ATU Minimum Grant Register — ATUMGNT	8	+03EH
ATU Maximum Latency Register — ATUMLAT	8	+03FH
Inbound ATU Limit Register 0 — IALR0	32	+040H
Inbound ATU Translate Value Register 0 — IATVR0	32	+044H
Inbound ATU Upper Translate Value Register 0 — IAUTVR0	32	+048H
Inbound ATU Limit Register 1 — IALR1	32	+04CH
Inbound ATU Translate Value Register 1 — IATVR1	32	+050H
Inbound ATU Upper Translate Value Register 1 — IAUTVR1	32	+054H
Inbound ATU Limit Register 2 — IALR2	32	+058H
Inbound ATU Translate Value Register 2 — IATVR2	32	+05CH
Inbound ATU Upper Translate Value Register 2 — IAUTVR2	32	+060H
Expansion ROM Limit Register — ERLR	32	+064H
Expansion ROM Translate Value Register — ERTVR	32	+068H
Expansion ROM Upper Translate Value Register — ERUTVR	32	+06CH
ATU Configuration Register — ATUCR	32	+070H
PCI Configuration and Status Register — PCSR	32	+074H
ATU Interrupt Status Register — ATUISR	32	+078H
ATU Interrupt Mask Register — ATUIMR	32	+07CH
Reserved	32	+080H

Notes:
1. MSI and MSI-X Capability Registers are documented in the Messaging Unit Chapter.

Table 544. Address Translation Unit Registers — ATUX (Sheet 2 of 3)

Register Description (Name)	Register Size in Bits	Internal Bus Address Offset (Relative to ATUX Base Address Offset)
Reserved	32	+084H
Reserved	32	+088H
Reserved	32	+08CH
VPD Capability Identifier Register — VPD_Cap_ID	8	+090H
VPD Next Item Pointer Register — VPD_Next_Item_Ptr	8	+091H
VPD Address Register — VPDAR	16	+092H
VPD Data Register — VPDDR	32	+094H
PM Capability Identifier Register — PM_Cap_ID	8	+098H
PM Next Item Pointer Register — PM_Next_Item_Ptr	8	+099H
ATU Power Management Capabilities Register — APMCR	16	+09AH
ATU Power Management Control/Status Register — APMCSR	16	+09CH
MSI_Capability Identifier Register — MSI_Cap_ID[1]	8	+0A0H
MSI Next Item Pointer Register — MSI_Next_Item_Ptr	8	+0A1H
MSI Message Control Register — MSI_MCR	16	+0A2H
MSI Address Register — MSI_ADDR	32	+0A4H
MSI Message Upper Address Register — MSI_MUAR	32	+0A8H
MSI Message Data Register — MSI_MD	16	+0ACH
Reserved	16	+0AEH
MSI-X Capability Identifier Register — MSI-X_Cap_ID	8	+0B0H
MSI-X Next Item Pointer Register — MSI-X_Next_Item_Ptr	8	+0B1H
MSI-X Message Control Register — MSI-X_MCR	16	+0B2H
MSI-X Table Offset Register — MSI-X_Table_Offset	32	+0B4H
MSI-X Pending Bit Array Offset Register — MSI-X_PBA_Offset	32	+0B8H
MU MSI-X Control Register x — MMCRx	32	+0BCH
Reserved	x	+0C0H through 0CFH
PCI-X Capability Identifier Register — PCI-X_Cap_ID	8	+0D0H
PCI-X Next Item Pointer Register — PCI-X_Next_Item_Ptr	8	+0D1H
PCI-X Command Register — PCIXCMD	16	+0D2H
PCI-X Status Register — PCIXSR	32	+0D4H
ECC Control and Status Register — ECCCSR	32	+0D8H
ECC First Address Register — ECCFAR	32	+0DCH
ECC Second Address Register — ECCSAR	32	+0E0H
ECC Attribute Register — ECCAR	32	+0E4H
CompactPCI* Hot-Swap Capability ID Register	8	+0E8H
Offset EDh: HS_NXTP — Next Item Pointer	8	+0E9H
HS_CNTRL — Hot-Swap Control/Status Register	8	+0EAH
Reserved	x	+0EBH through 1FFH
Inbound ATU Base Address Register 3 — IABAR3	32	+200H
Inbound ATU Upper Base Address Register 3 — IAUBAR3	32	+204H
Inbound ATU Limit Register 3 — IALR3	32	+208H
Inbound ATU Translate Value Register 3 — IATVR3	32	+20CH
Inbound ATU Upper Translate Value Register 3 — IAUTVR3	32	+210H
Reserved	x	+214H through 2FFH
Outbound I/O Base Address Register — OIOBAR	32	+300H

Notes:
1. MSI and MSI-X Capability Registers are documented in the Messaging Unit Chapter.

Table 544. Address Translation Unit Registers — ATUX (Sheet 3 of 3)

Register Description (Name)	Register Size in Bits	Internal Bus Address Offset (Relative to ATUX Base Address Offset)
Outbound I/O Window Translate Value Register — OIOWTVR	32	+304H
Outbound Upper Memory Window Base Address Register 0 - OUMBAR0	32	+308H
Outbound Upper 32-bit Memory Window Translate Value Register 0 — OUMWTVR0	32	+30CH
Outbound Upper Memory Window Base Address Register 1 - OUMBAR1	32	+310H
Outbound Upper 32-bit Memory Window Translate Value Register 1 — OUMWTVR1	32	+314H
Outbound Upper Memory Window Base Address Register 2 - OUMBAR2	32	+318H
Outbound Upper 32-bit Memory Window Translate Value Register 2 — OUMWTVR2	32	+31CH
Outbound Upper Memory Window Base Address Register 3 - OUMBAR3	32	+320H
Outbound Upper 32-bit Memory Window Translate Value Register 3 — OUMWTVR3	32	+324H
Reserved.	32	+328H
Reserved.	32	+32CH
Outbound Configuration Cycle Address Register — OCCAR	32	+330H
Outbound Configuration Cycle Data Register — OCCDR	32	+334H
Outbound Configuration Cycle Function Number — OCCFN	32	+338H
Reserved	x	+33CH through +37FH
PCI Interface Error Control and Status Register — PIECSR	32	+380H
PCI Interface Error Address Register — PCIEAR	32	+384H
PCI Interface Error Upper Address Register — PCIEUAR	32	+388H
PCI Interface Error Context Address Register — PCIECAR	32	+38CH
Reserved	x	+390H
Internal Arbiter Control Register — IACR	16	+394H
Reserved	x	+396H
Multi-Transaction Timer — MTT	8	+398H
Reserved	x	+39CH through +FFFH

Notes:
1. MSI and MSI-X Capability Registers are documented in the Messaging Unit Chapter.

19.6.2.2 Address Translation Unit (PCI-E)

All of the ATU registers are accessible through both inbound PCI configuration cycles and the 4138xx core CPU (Register offsets 000H through FFFH).

The Internal Bus Address Offset to PMMRBAR of any ATU Register can be derived by adding the 4 KB address aligned ATUE Base Address Offset (Table 545, “Intel® 413808 and 413812 I/O Controllers ATUE Configuration Space Base Address Offset” on page 814) to the Register Offset (Table 546, “Address Translation Unit Registers — ATUE” on page 815).

For example, when INTERFACE_SEL_PCIX# and CONTROLLER_ONLY# are both asserted, the offset to PMMRBAR of the ATU Command Register would be (4 D000H+004H) or 4 D004H.

Note: The 4 KB Address Aligned Range Offset can be different depending on two configuration straps as described in Table 545.

Table 545. Intel® 413808 and 413812 I/O Controllers ATUE Configuration Space Base Address Offset

CONTROLLER_ONLY#	INTERFACE_SEL_PCIX#	ATUE Base Address Offset (Relative to PMMRBAR)
PCI Attributes		
Deasserted	Deasserted	+4 0000H
Asserted	Deasserted	+4 5000H
Asserted	Asserted	+4 5000H
Deasserted	Asserted	+4 5000H
Local Attributes		
Deasserted	Deasserted	+4 8000H
Asserted	Deasserted	+4 D000H
Asserted	Asserted	+4 D000H
Deasserted	Asserted	+4 D000H

Table 546. Address Translation Unit Registers — ATUE (Sheet 1 of 4)

Register Description (Name)	Register Size in Bits	Internal Bus Address Offset (Relative to ATUE Base Address Offset)
ATU Vendor ID Register — ATUVID	16	+000H
ATU Device ID Register — ATUDID	16	+002H
ATU Command Register — ATUCMD	16	+004H
ATU Status Register — ATUSR	16	+006H
ATU Revision ID Register — ATURID	8	+008H
ATU Class Code Register — ATUCCR	24	+009H
ATU Cacheline Size Register — ATUCLSR	8	+00CH
ATU Latency Timer Register — ATULT	8	+00DH
ATU Header Type Register — ATUHTR	8	+00EH
ATU BIST Register — ATUBISTR	8	+00FH
Inbound ATU Base Address Register 0 — IABAR0	32	+010H
Inbound ATU Upper Base Address Register 0 — IAUBAR0	32	+014H
Inbound ATU Base Address Register 1 — IABAR1	32	+018H
Inbound ATU Upper Base Address Register 1 — IAUBAR1	32	+01CH
Inbound ATU Base Address Register 2 — IABAR2	32	+020H
Inbound ATU Upper Base Address Register 2 — IAUBAR2	32	+024H
Reserved	32	+028H
ATU Subsystem Vendor ID Register — ASVIR	16	+02CH
ATU Subsystem ID Register — ASIR	16	+02EH
Expansion ROM Base Address Register -ERBAR	32	+030H
ATU Capabilities Pointer Register — ATU_Cap_Ptr	8	+034H
Reserved	24	+035H
Reserved	32	+038H
ATU Interrupt Line Register — ATUILR	8	+03CH
ATU Interrupt Pin Register — ATUIPR	8	+03DH
ATU Minimum Grant Register — ATUMGNT	8	+03EH
ATU Maximum Latency Register — ATUMLAT	8	+03FH
Inbound ATU Limit Register 0 — IALR0	32	+040H
Inbound ATU Translate Value Register 0 — IATVR0	32	+044H
Inbound ATU Upper Translate Value Register 0 — IAUTVR0	32	+048H
Inbound ATU Limit Register 1 — IALR1	32	+04CH
Inbound ATU Translate Value Register 1 — IATVR1	32	+050H
Inbound ATU Upper Translate Value Register 1 — IAUTVR1	32	+054H
Inbound ATU Limit Register 2 — IALR2	32	+058H
Inbound ATU Translate Value Register 2 — IATVR2	32	+05CH
Inbound ATU Upper Translate Value Register 2 — IAUTVR2	32	+060H
Expansion ROM Limit Register — ERLR	32	+064H
Expansion ROM Translate Value Register — ERTVR	32	+068H
Expansion ROM Upper Translate Value Register — ERUTVR	32	+06CH
ATU Configuration Register — ATUCR	32	+070H
PCI Configuration and Status Register — PCSR	32	+074H
ATU Interrupt Status Register — ATUISR	32	+078H
ATU Interrupt Mask Register — ATUIMR	32	+07CH
PCI Express Message Control/Status Register — PEMCSR	32	+080H

Notes:
1. MSI and MSI-X Capability Registers are documented in the Messaging Unit Chapter.

Table 546. Address Translation Unit Registers — ATUE (Sheet 2 of 4)

Register Description (Name)	Register Size in Bits	Internal Bus Address Offset (Relative to ATUE Base Address Offset)
PCI Express Link Control/Status Register — PELCSR	32	+084H
Reserved	x	+088H through +08FH
VPD Capability Identifier Register — VPD_Cap_ID	8	+090H
VPD Next Item Pointer Register — VPD_Next_Item_Ptr	8	+091H
VPD Address Register — VPDAR	16	+092H
VPD Data Register — VPDDR	32	+094H
PM_Capability Identifier Register — PM_Cap_ID	8	+098H
PM Next Item Pointer Register — PM_Next_Item_Ptr	8	+099H
ATU Power Management Capabilities Register — APMCR	16	+09AH
ATU Power Management Control/Status Register — APMCSR	16	+09CH
Reserved	16	+09EH
MSI_Capability Identifier Register — MSI_Cap_ID[1]	8	+0A0H
MSI Next Item Pointer Register — MSI_Next_Item_Ptr	8	+0A1H
MSI Message Control Register — MSI_MCR	16	+0A2H
MSI Address Register — MSI_ADDR	32	+0A4H
MSI Message Upper Address Register — MSI_MUAR	32	+0A8H
MSI Message Data Register — MSI_MD	16	+0ACH
Reserved	16	+0AEH
MSI-X Capability Identifier Register — MSI-X_Cap_ID	8	+0B0H
MSI-X Next Item Pointer Register — MSI-X_Next_Item_Ptr	8	+0B1H
MSI-X Message Control Register — MSI-X_MCR	16	+0B2H
MSI-X Table Offset Register — MSI-X_Table_Offset	32	+0B4H
MSI-X Pending Bit Array Offset Register — MSI-X_PBA_Offset	32	+0B8H
MU MSI-X Control Register x — MMCRx	32	+0BCH
Reserved	x	+0C0H through +0CFH
PCI Express Capability Identifier Register — PCIE_CAPID	8	+0D0H
PCI Express Next Item Pointer Register — PCIE_NXTP	8	+0D1H
PCI Express Capabilities Register PCIE_CAP	16	+0D2H
PCI Express Device Capabilities Register — PCIE_DCAP	32	+0D4H
PCI Express Device Control Register — PE_DCTL	16	+0D8H
PCI Express Device Status Register PE_DSTS	16	+0DAH
PCI Express Link Capabilities Register — PE_LCAP	32	+0DCH
PCI Express Link Control Register PE_LCTL	16	+0E0H
PCI Express Link Status Register PE_LSTS	16	+0E2H
PCI Express Slot Capabilities Register — PE_SCAP	32	+0E4H
PCI Express Slot Control Register PE_SCR	16	+0E8H
PCI Express Slot Status Register PE_SSTS	16	+0EAH
PCI Express Root Control Register — PE_RCR	16	+0ECH
Reserved	16	+0EEH
PCI Express Root Status Register PE_RSR	32	+0F0H
Reserved	96	+0F4H through +0FFH
PCI Express Advanced Error Capability Identifier — ADVERR_CAPID	32	+100H
PCI Express Uncorrectable Error Status — ERRUNC_STS	32	+104H
PCI Express Uncorrectable Error Mask — ERRUNC_MSK	32	+108H

Notes:
1. MSI and MSI-X Capability Registers are documented in the Messaging Unit Chapter.

Table 546. Address Translation Unit Registers — ATUE (Sheet 3 of 4)

Register Description (Name)	Register Size in Bits	Internal Bus Address Offset (Relative to ATUE Base Address Offset)
PCI Express Uncorrectable Error Severity — ERRUNC_SEV	32	+10CH
PCI Express Correctable Error Status — ERRCOR_STS	32	+110H
PCI Express Correctable Error Mask — ERRCOR_MSK	32	+114H
Advanced Error Control and Capability Register — ADVERR_CTL	32	+118H
PCI Express Advanced Error Header Log — ADVERR_LOG0	32	+11CH
PCI Express Advanced Error Header Log — ADVERR_LOG1	32	+120H
PCI Express Advanced Error Header Log — ADVERR_LOG2	32	+124H
PCI Express Advanced Error Header Log — ADVERR_LOG3	32	+128H
Root Error Command Register — RERR_CMD	32	+12CH
Root Error Status Register — RERR_SR	32	+130H
Error Source Identification Register RERR_ID	32	+134H
Reserved.	32	+140H
Reserved.	32	+144H
Reserved.	32	+148H
Reserved.	32	+14CH
Reserved.	x	+150H through +1DFH
Device Serial Number Capability — DSN_CAP	32	+1E0H
Device Serial Number Lower DW Register — DSN_LDW	32	+1E4H
Device Serial Number Upper DW Register — DSN_UDW	32	+1E8H
Power Budgeting Enhanced Capability Header — PWRBGT_CAPID	32	+1F0H
Power Budgeting Data Select Register — PWRBGT_DSEL	32	+1F4H
Power Budgeting Data Register — PWRBGT_DATA	32	+1F8H
Power Budgeting Capability Register — PWRBGT_CAP	32	+1FCH
Power Budgeting Information Registers[0:23] — PWRBGT_INFO[0:23]	32 x24	+200H through +25CH
Reserved.	x	+260H through +2FFH
Outbound I/O Base Address Register — OIOBAR	32	+300H
Outbound I/O Window Translate Value Register — OIOWTVR	32	+304H
Outbound Upper Memory Window Base Address Register 0 — OUMBAR0	32	+308H
Outbound Upper 32-bit Memory Window Translate Value Register 0- OUMWTVR0	32	+30CH
Outbound Upper Memory Window Base Address Register 1 — OUMBAR1	32	+310H
Outbound Upper 32-bit Memory Window Translate Value Register 1- OUMWTVR1	32	+314H
Outbound Upper Memory Window Base Address Register 2- OUMBAR2	32	+318H
Outbound Upper 32-bit Memory Window Translate Value Register 2- OUMWTVR2	32	+31CH
Outbound Upper Memory Window Base Address Register 3 — OUMBAR3	32	+320H
Outbound Upper 32-bit Memory Window Translate Value Register 3- OUMWTVR3	32	+324H
Reserved.	32	+328H
Outbound Configuration Cycle Address Register — OCCAR	32	+32CH
Outbound Configuration Cycle Data Register — OCCDR	32	+330H
Outbound Configuration Cycle Function Number — OCCFN	32	+334H
Reserved.	x	+338H through +33FH
Inbound Vendor Defined Message Header Register0 — IVMHR0	32	+340H
Inbound Vendor Defined Message Header Register 1 — IVMHR1	32	+344H
Inbound Vendor Defined Message Header Register 2 — IVMHR2	32	+348H
Inbound Vendor Defined Message Header Register 3 — IVMHR3	32	+34CH

Notes:
1. MSI and MSI-X Capability Registers are documented in the Messaging Unit Chapter.

Table 546. Address Translation Unit Registers — ATUE (Sheet 4 of 4)

Register Description (Name)	Register Size in Bits	Internal Bus Address Offset (Relative to ATUE Base Address Offset)
Inbound Vendor Defined Message Payload Register — IVMPR	32	+350H
Reserved.	x	+354H through +35FH
Outbound Vendor Defined Message Header Register0 — OVMHR0	32	+360H
Outbound Vendor Defined Message Header Register 1 — OVMHR1	32	+364H
Outbound Vendor Defined Message Header Register 2 — OVMHR2	32	+368H
Outbound Vendor Defined Message Header Register 3 — OVMHR3	32	+36CH
Outbound Vendor Defined Message Payload Register — OVMPR	32	+370H
Reserved.	x	+374H through +37FH
PCI Interface Error Control and Status Register — PIE_CSR	32	+380H
PCI Interface Error Status — PIE_STS	32	+384H
PCI Interface Error Mask — PIE_MSK	32	+388H
PCI Interface Error Header Log — PIE_LOG0	32	+38CH
PCI Interface Error Header Log 1 — PIE_LOG1	32	+390H
PCI Interface Error Header Log 2 — PIE_LOG2	32	+394H
PCI Interface Error Header Log — PIE_LOG3	32	+398H
PCI Interface Error Header Log — PIE_DLOG	32	+39CH
Reserved.	x	+3A0H through +FFFH

Notes:
1. MSI and MSI-X Capability Registers are documented in the Messaging Unit Chapter.

19.7 PCI Configuration Space

The PCI Configuration space of the 4138xx varies depending on the DFSEL, INTERFACE_SEL_PCIX#, and CONTROLLER_ONLY# straps. The PCI Functions that are visible in via configuration transactions are details in Table 547, "Intel® 413808 and 413812 I/O Controllers in TPER Mode PCI Function Visibility".

Configurations cycles that target 4138xx are translated into memory cycles on the internal bus and access the PCI Attributes section of the PMMR region.

Table 547. Intel® 413808 and 413812 I/O Controllers in TPER Mode PCI Function Visibility

	PCI Function Number							
DFSEL	0	1	2	3	4	5	6	7
000	ATUX[a]; ATUE[b];	Reserved						
001 through 111	ATUX; ATUE	Reserved	Reserved					

a. ATUX is visible in function 0 when the INTERFACE_SEL_PCIX# strap is sampled as 0 and CONTROLLER_ONLY# = 1.
b. ATUE is visible in function 0 when the INTERFACE_SEL_PCIX# strap is sampled as 1 and CONTROLLER_ONLY# = 1.

19.8 Coprocessor Register Space

The CCR address space is assigned to support the integrated peripherals on the 4138xx that require low latency register access. Table 548 shows all of the 4138xx integrated coprocessor registers and assigned coprocessor space. The *ARM Architecture Reference Manual* provides for a total of 16 coprocessors each of which can contain up to 256 32 bit registers. For completeness, the coprocessor space reserved by the *ARM Architecture Reference Manual* is shown.

Note: All accesses to CP6 unimplemented coprocessor registers complete and return 0s when read and show as "undefined". The same rule applies to unimplemented 4138xx. co-processors.

Table 548. Coprocessor Registers Assigned to Integrated Peripherals

Integrated Peripheral	Coprocessor
Inter-Processor Communication Unit	CP6
Interrupt Control Unit	CP6
Programmable Timers	CP6
Bus Interface Unit	CP7
Core Performance Monitoring Unit	CP14
System Control[a]	CP15

a. Reserved by the *ARM Architecture Reference Manual*.

Table 549. Coprocessor Register Locations (Sheet 1 of 4)

Peripheral	Register Description (Name)	Coprocessor	Field CR_m	Coprocessor Register (Field CR_n)
Inter-Processor Communication	Core Identification Register — CIDR	CP6	0	Register 0
	Reset Cause Status Register — RCSR		1	Register 0
	Software Interrupt Generation Register — SINTGENR			Register 1
	Targeted Reset Register — TARRSTR			Register 2
	Undefined			Register 3 through Register 7
	Inbound MSI Interrupt Pending Register 0			Register 8
	Inbound MSI Interrupt Pending Register 1			Register 9
	Inbound MSI Interrupt Pending Register 2			Register 10
	Inbound MSI Interrupt Pending Register 3			Register 11
	Undefined			Register 12 through Register 15
Interrupt Control Unit	Interrupt Base Register	CP6	2	Register 0
	Undefined			Register 1
	Interrupt Size Register			Register 2
	IRQ Interrupt Vector Register			Register 3
	FIQ Interrupt Vector Register			Register 4
	Undefined			Register 4–7
	Inter-Processor Interrupt Pending Register			Register 8
	Undefined			Register 9–15
	Interrupt Pending Register 0		3	Register 0
	Interrupt Pending Register 1			Register 1
	Interrupt Pending Register 2			Register 2
	Interrupt Pending Register 3			Register 3
	Undefined			Register 4–15
	Interrupt Control Register 0		4	Register 0
	Interrupt Control Register 1			Register 1
	Interrupt Control Register 2			Register 2
	Interrupt Control Register 3			Register 3
	Undefined			Register 4–15

Table 549. Coprocessor Register Locations (Sheet 2 of 4)

Peripheral	Register Description (Name)	Coprocessor	Field CR_m	Coprocessor Register (Field CR_n)
	Interrupt Steering Register 0		5	Register 0
	Interrupt Steering Register 1			Register 1
	Interrupt Steering Register 2			Register 2
	Interrupt Steering Register 3			Register 3
	Undefined			Register 4–15
Interrupt Control Unit	IRQ Interrupt Source Register 0	CP6	6	Register 0
	IRQ Interrupt Source Register 1			Register 1
	IRQ Interrupt Source Register 2			Register 2
	IRQ Interrupt Source Register 3			Register 3
	Undefined			Register 4–15
	FIQ Interrupt Source Register 0		7	Register 0
	FIQ Interrupt Source Register 1			Register 1
	FIQ Interrupt Source Register 2			Register 2
	FIQ Interrupt Source Register 3			Register 3
	Undefined			Register 4–15
	Interrupt Priority Register 0		8	Register 0
	Interrupt Priority Register 1			Register 1
	Interrupt Priority Register 2			Register 2
	Interrupt Priority Register 3			Register 3
	Interrupt Priority Register 4			Register 4
	Interrupt Priority Register 5			Register 5
	Interrupt Priority Register 6			Register 6
	Interrupt Priority Register 7			Register 7
	Undefined			Register 8–15

Table 549. Coprocessor Register Locations (Sheet 3 of 4)

Peripheral	Register Description (Name)	Coprocessor	Field CR_m	Coprocessor Register (Field CR_n)
Programmable Timers Unit	Timer Mode Register 0	CP6	9	Register 0
	Timer Mode Register 1			Register 1
	Timer Count Register 0			Register 2
	Timer Count Register 1			Register 3
	Timer Reload Register 0			Register 4
	Timer Reload Register 1			Register 5
	Timer Interrupt Status Register			Register 6
	Watch Dog Timer Control Register			Register 7
	Watch Dog Timer Setup Register			Register 8
	Undefined			Register 9–15
Bus Interface Unit	L2 Cache and BIU Error Logging Register	CP7	2	Register 0
	L2 Cache and BIU Error Lower Address Register			Register 1
	L2 Cache and BIU Error Upper Address Register			Register 2
	Undefined			Register 3–15
Clock and Power Management Unit	Undefined	CP14	0	Register 0–5
	Core Clock Configuration Register			Register 6
	Power Mode Register			Register 7
	Transmit Register			Register 8
	Receive Register			Register 9
	Debug Control and Status Register			Register 10
	Trace Buffer Register			Register 11
	Checkpoint 0 Register			Register 12
	Checkpoint 1 Register			Register 13
	Transmit/Receive Control Register			Register 14
	Undefined			Register 15

Table 549. Coprocessor Register Locations (Sheet 4 of 4)

Peripheral	Register Description (Name)	Coprocessor	Field CR_m	Coprocessor Register (Field CR_n)
Core Performance Monitoring Unit	Performance Monitor Control Register	CP14	1	Register 0
	Clock Count Register			Register 1
	Undefined			Register 2
	Undefined			Register 3
	Interrupt Enable Register			Register 4
	Overflow Flag Register			Register 5
	Undefined			Register 6
	Undefined			Register 7
	Event Selection Register			Register 8
	Undefined			Register 9–15
	Performance Count Register 0	CP14	2	Register 0
	Performance Count Register 1			Register 1
	Performance Count Register 2			Register 2
	Performance Count Register 3			Register 3
	Undefined			Register 4–15
System Controls	ID & Cache Type Registers	CP15	CP 15 Function[a]	Register 0
	Control and Auxiliary Control Registers			Register 1
	Translation Table Base Register			Register 2
	Domain Access Control Register			Register 3
	Undefined			Register 4
	Fault Status Register			Register 5
	Fault Address Register			Register 6
	Cache Functions Register			Register 7
	TLB Operations Register			Register 8
	Cache Lock Down			Register 9
	TLB Lock Down			Register 10
	Reserved			Register 11–12
	Process ID Register			Register 13
	Breakpoint Registers			Register 14
	CP Access (PID) Coprocessor Access Control Register			Register 15

a. Some of the CP15 registers are differentiated by the Opcode_2 field. For CP15, the CR_m is used in some cases to denote different control functions for a given coprocessor register rather than a distinct register decode. Please refer to the *Intel® 80200 Processor based on Intel® XScale™ Microarchitecture Developer's Manual* (Order Number: 273411), for more details on the operation of CP15.