Intel® 460GX Chipset System Software Developer's Manual

June 2001

Document Number: 248704-001

Contents

Figures

Tables

Introduction 1

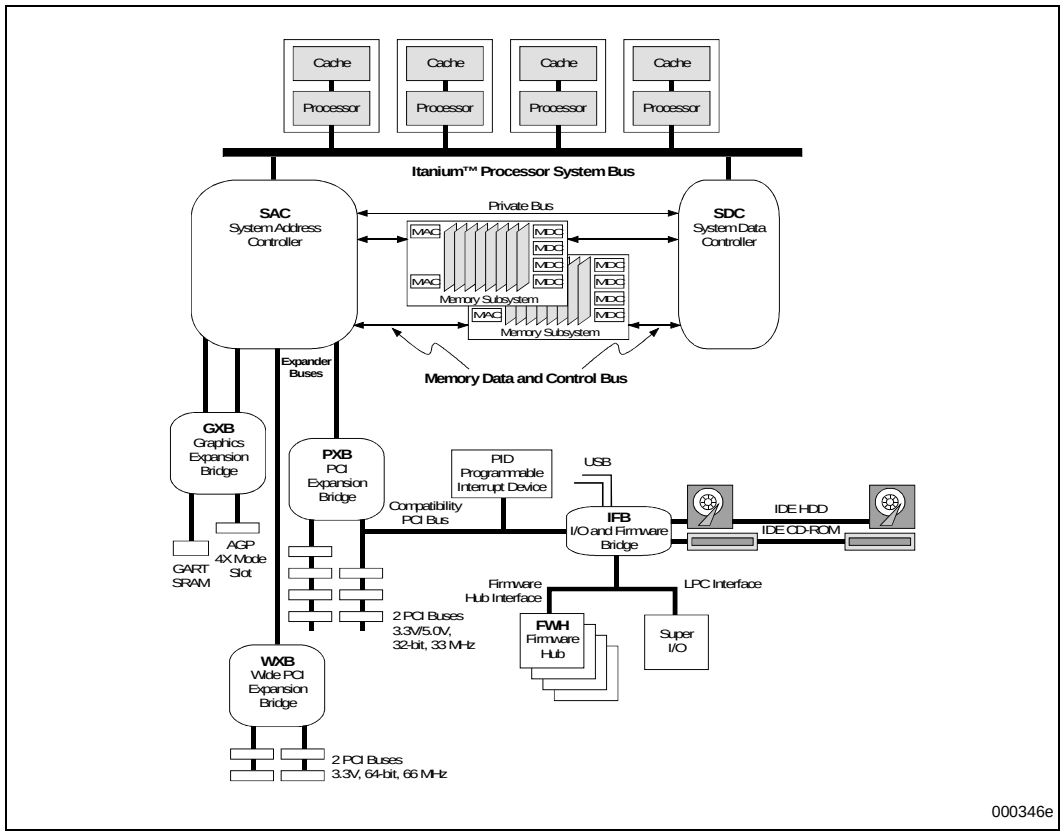
This document provides information about the Intel® 460GX chipset components. The 460GX chipset is a high performance memory and I/O chipset for the Intel Itanium™ processor, targeted for multiprocessor server and high-end workstation designs.

This document describes the software programmer's interface to the 460GX chipset. It provides a brief summary of the system architecture supported by the 460GX chipset, a list of features within the chipset and a detailed description of software or other externally visible segments.

1.1 System Overview

The Intel 460GX chipset is a high performance chipset for Intel Itanium processor-based systems, targeted for multiprocessor servers and high-performance workstations. It provides the memory controller interface and appropriate bridges to PCI, AGP 4X, and other standard I/O buses. Figure 1-1 illustrates the basic system configuration of a four-processor platform.

Figure 1-1. Diagram of a Typical Intel® 460GX Chipset-based System with AGP



1.1.1 Component Overview

Table 1-1 lists the 460GX chipset components.

Table 1-1. Intel® 460GX Chipset Components

Component	Name	Function
SAC	82461GX System Address Controller	Interfaces the address and control portion of the Itanium™ processor system bus and the memory bus. Acts as a host bridge interface to peripheral I/O devices through four Expander busses.
SDC	82462GX System Data path Controller	Interfaces the data portion of the Itanium processor system bus and the memory bus.
MAC	82463GX Memory Address Controller	Provides the SDRAM RAS/CAS/WE/CS generation as well as redriving the address to the SDRAMs. It is capable of buffering several commands from the SAC.
MDC	82464GX Memory Data path Controller	Multiplexes the data from the SDRAM to the SDC. On reads, it latches data from the SDRAM, then transfers the data to the SDC. On writes, it latches the data from the SDC, then writes the data to the SDRAM.
GXB	82465GX Graphics Expander Bridge	Provides the control and data interface for an AGP 4X graphics port. This device attaches to the SAC via two Expander busses which utilize a special configuration.
WXB	82466GX Wide and fast PCI Expander Bridge	Provides the primary control and data interface for two independent 64-bit, 66 MHz PCI interfaces. This device attaches to the SAC via an Expander bus.
PXB	82467GX PCI Expander Bridge	Provides the primary control and data interface for two independent 32-bit, 33-MHz PCI interfaces. These two 32-bit interfaces may operate together to produce a single 64-bit, 33-MHz interface via a configuration option. This device attaches to the SAC via an Expander bus.
IFB	82468GX I/O and Firmware Bridge	The IFB is a multi-function PCI device implementing a PCI-to-LPC bridge function, a PCI IDE function, a Universal Serial Bus Host/Hub function, an SMBus Interface function, Power Management function and the Firmware Hub interface.
FWH	82802AC Firmware Hub 8Mb	The FWH component interfaces to the IFB component and provides firmware storage and security features. Further FWH information can be found at http://developer.intel.com/design/chipsets/datasheets or by ordering document 290658.
PID	NEC# UPD66566S1-016 Programmable Interrupt Device	The PID is an interrupt controller that provides interrupt steering functions. The PID contains the logic required to support 8259A mode, APIC mode, and SAPIC mode interrupt controller operations. The PID interfaces include a PCI bus interface, an APIC bus interface, a serial IRQ interface, and an interrupt input interface.

1.2 Product Features

- High performance hardware based on IA-64 architecture
  - 4.2 GB/s memory bandwidth can simultaneously support both the full system bus and the full I/O bus bandwidths
  - Architectural support for 64 MB to 64 GB of SDRAM
  - Support for up to four bridge chips that interface to the 82461GX (SAC) through four Expander channels, each 30 bits wide and providing 533 MB/s peak bandwidth
  - AGP 4X compatible, via the 82465GX (GXB) and two Expander channels running at 266 MHz totaling 1 GB/s peak bandwidth
  - Support for two 64-bit, 66-MHz PCI buses using one 82466GX (WXB) component per Expander channel
  - Support for two independent 32-bit, 33-MHz PCI buses or one 64-bit, 33-MHz PCI bus via the 82467GX (PXB) per Expander channel
  - Data streaming support between Expanders and DRAM, up to 533 MB/s per Expander channel
- Extensive RAS features for mission-critical needs
  - ECC protection on the system bus data signals
  - Memory ECC with single-bit error correction, double and nibble error detection
  - Address and data flows protected by parity throughout chipset
  - ECC bits in DRAM accessible by diagnostics
  - Fault recording of multiple errors; sticky through reset
  - JTAG TAP port for debug and boundary scan capability
  - I2C slave interface for viewing and modifying specific error and configuration registers
  - Bus, memory and I/O performance counters
  - Support of ACPI/DMI functions (support is provided in the IFB)
- High bandwidth system bus for multiprocessor scalability
  - Support of the Intel® Itanium™ processor 64-bit data bus
  - Full support for 4-way multiprocessing
  - 266 MHz data bus frequency
  - Cache line size of 64 bytes
  - Enhanced defer feature for out-of-order data delivery using IDS#
  - AGTL+ bus driver technology
- Features to support flexible platform environments
  - Hardware compatible with IA-32 binaries
  - AGP address space up to 32 GB supported
  - Support for Auto Detection of SDRAM memory type and mixed memory sizes allowed between rows
  - Supports 16-, 64-, 128- and 256-Mbit DRAM devices
  - Full support for the PCI Configuration Space Enable (CSE) protocol to devices on all Expander channels
  - WXB supports 3.3 volt PCI bus operation (supports universal and 3.3 volt PCI cards) and has an Integrated Hot-Plug Controller**
  - PCI Rev. 2.2 compliant on the WXB and PXB
  - GXB supports fast writes and 1x, 2x and 4x data rates
  - 1 MB or greater of firmware storage provided by the 82802AC (FWH)
  - Interrupt controller, bus-mastering IDE and Universal Serial Bus supported by the 82468GX (IFB)
  - Support of 8259A mode, APIC mode and SAPIC mode interrupts via the UPD55566S1-016 (PID) provided by NEC*

**Based on technology licensed from Compaq Computer Corp.

1.3 Itanium™ Processor System Bus Support

- Full support for the Itanium processor system bus.
  - 64-bit data bus.
  - 266 MHz data bus frequency.
  - Cache line size of 64 bytes.
  - Supports SAPIC interrupt protocol.
- Full support for 4-way multiprocessing.
- Parity protection on address and control signals, ECC protection on the data signals.
- GTL+ bus driver technology.

1.4 DRAM Interface Support

- SDRAM 3.3 volt, 168-pin DIMM's are the only memory type supported.
- Support for 64 MB to 64 GB of DRAM.
- Minimum memory size is 64 MB using 16 MB DIMM's.
- Minimum incremental size is 64 MB using 16 MB DIMM's.
- Maximum memory size is 16 GB using 128 MB DIMM's.
- Maximum memory size is 64 GB using 1 GB DIMM's.
- Only 3.3 volt memory is supported.
- Support for Auto Detection of SDRAM Memory Type.
- Supports 16, 64, 128 and 256 Mbit DRAM devices.
- Mixed memory sizes allowed between rows.
- Staggered CAS-before-RAS refresh (standard SDRAM refresh).
- ECC with single-bit error correction, double and nibble error detection.
- Extensive processor-to-Memory and PCI-to-Memory write data buffering, thus minimizing the interference of writes on read latency.

1.5 I/O Support

- 4 Expander ports, each 30 bits wide and providing 533 MB/s peak bandwidth.
- Each Expander bus supports a single PXB or WXB. Two Expander busses can be configured to support a GXB.
- Full support for the PCI Configuration Space Enable (CSE) protocol to devices on all Expander ports.
- Data streaming support between Expanders and DRAM, up to 533 MB/s per Expander port.
- All outbound memory and I/O reads (except locked reads) are deferred.
- All outbound memory space writes are posted. Outbound I/O space writes are optionally posted (unless targeting an address with side effects, in which case they are deferred).
- All inbound memory reads are delayed.
- All inbound memory space writes are posted.
- Supports concurrent processor and I/O initiated transactions to main memory.
- Maintains coherency with processors by snooping all inbound transactions to the system bus.
- Supports non-coherent traffic (for AGP), with a direct path to memory bypassing the system bus.

1.5.1 PXB Features

- Can be configured to provide two independent 32 bit, 33 MHz PCI buses or one 64 bit, 33 MHz PCI bus.
- PCI Rev. 2.2, 5V tolerant (PXB drives 3.3 volts, but is 5.0 volt tolerant).

- Parity protection on all PCI signals.
- Data collection & write assembly.
  - — Combines back-to-back sequential processor-to-PCI memory writes to PCI burst writes.
  - — Processor to PCI write assembly of full/partial line writes.
- Two outbound read requests containing a total of two cache lines of read data for each PCI bus.
- Supports six outbound write requests containing a total of three cache lines of write data for each 32 bit PCI bus. Supports 12 outbound write requests containing a total of six cache lines of write data for a 64 bit PCI bus.
- Supports two delayed inbound read requests.
- Supports the I/O and Firmware Bridge (IFB).
- Supports either internal or external arbitration, allowing additional bus masters, on the PCI bus.

1.5.2 WXB Features

- Support for two 64 bit, 66 MHz PCI busses.
- 3.3 Volt PCI bus operation (supports Universal and 3.3 Volt PCI cards).
- PCI Specification, Revision 2.2.
- Integrated Hot-Plug controller.

1.5.3 GXB Features

- The GXB is AGP and AGP 4X mode compatible, nominal 66 MHz, 266 MHz, 1 GB/s peak bandwidth.
- The GXB supports pipelined operation or sideband signals on AGP 4X mode bus.
- AGP address space of 1 GB or 256 MB supported. Also supports 32 GB of GART window, if 4 MB pages are used.
- Supports Fast Writes and 1x, 2x and 4x data rates.

1.6 RAS Features

- ECC coverage of system data bus using the Itanium™ processor SEC/DED ECC code. Memory is protected using a SEC/DED code which also provides nibble detection capabilities of 4 bits. All control and address signals are parity protected. Local control buses are parity protected. The Expander is covered by parity.
- Data flows protected by parity throughout chipset.
- ECC bits in DRAM accessible by diagnostics.
- Fault recording of multiple errors; sticky through reset, but NOT through power-down.
- Memory scrubbing implemented in hardware.
- Boundary test capability through JTAG.
- JTAG TAP port for debug.

- I2C Slave Interface will allow viewing and modifying of specific error and configuration registers.

1.7 Other Platform Components

These 460GX devices provide access to flash space, interrupt collection and legacy features.

1.7.1 I/O & Firmware Bridge (IFB)

The 460GX chipset is designed to work with the IFB south bridge. As part of this support, the PXB includes an internal PCI arbiter as well as support for an external PCI arbiter. The IFB consists of an 8259C Interrupt controller, a bus-mastering IDE interface, and a Universal Serial Bus interface. Devices using IFB are limited to a 32 bit addressing space available for DMA, not the full 44 bits supported by the Itanium™ processor.

1.7.2 Programmable Interrupt Device (PID)

The PID is a PCI device that gathers interrupts and delivers them from the PCI bus to the system bus using the SAPIC interrupt protocol. The interrupt will be presented to one of the processors on the bus for servicing. A 460GX chipset based platform requires at least one PID located on the compatibility PCI bus. The compatibility PID will handshake with the IFB before delivering a south bridge/compatibility device interrupt. The same PID may also be used to deliver some portion of the PCI based interrupts.

The system implementor can choose how many PIDs are used in the platform. If enough interrupt lines are shared, there need be only one PID in the system; all interrupts in the system would then be routed to that PID. Each PID has enough interrupt inputs to handle dedicated interrupts from the cards on two PCI buses. Therefore, using one PID per PXB provides a high performance solution with minimum routing between PCI buses.

1.8 Reference Documents

In addition to this document, the reader should be familiar with the following reference documents:

- *Intel® 460GX Chipset Datasheet*
  (Document Number: 248703)
- *Intel® Itanium™ Processor Hardware Developer's Manual*
  (Document Number: 248701)
- *Intel® Itanium™ Processor at 800 MHz and 733 MHz Datasheet*
  (Document Number: 245481)
- *Intel® 82460GX Chipset OLGA1 Package, Manufacturing, Mechanical, and Thermal Design Guide*
- *PCI Local Bus Specification*, Rev 2.2
  (http://www.pcisig.com/)
- *Accelerated Graphics Port Interface Specification*
  (http://www.intel.com/technology/agp/agp_index.htm)

- *JTAG IEEE 1149.1 Specification*
  (http://www.ieee.com)
- *Universal Serial Bus Specification*
  (http://www.usb.org)
- *System Management Bus Specification*, Rev. 1.0
- *Low Pin Count (LPC) Interface Specification*, Rev 1.0

Note: Contact your Intel representative for the latest revision of the documents without document numbers.

1.9 Revision History

Date	Description
June 2001	Initial release.

Register Descriptions 2

The 460GX chipset has both memory mapped and PCI configuration space mapped registers. The 460GX chipset supports access mechanism #1 as defined in the PCI specification. Two 32-bit register locations (CONFIG_ADDRESS and CONFIG_DATA) are defined in the Itanium processor's I/O space; I/O accesses to these registers are translated by the 460GX chipset into appropriate PCI configuration cycles.

To access a configuration register in the 460GX chipset (or any other I/O device), software first writes a value to the CONFIG_ADDRESS location consisting of the bus number, Device Number, function number and register number. These writes are claimed and saved by the 460GX chipset. Subsequent reads or writes to the CONFIG_DATA location result in the 460GX chipset using the information stored in CONFIG_ADDRESS to deliver a PCI configuration read or write cycle to the appropriate address on the appropriate PCI bus.

Upon reset, the 460GX chipset sets its internal configuration registers to predetermined default states, representing the minimum feature set required to successfully bring up the system. It is expected that the firmware will properly determine and program the optimal configuration settings. The 460GX chipset implements a PCI-compatible configuration space for each PCI bus under the PXBs, for each AGP bus under the GXB, and for each 460GX chipset component. Each configuration space provides hardwired device identification, address range registers, operation control registers, status and error registers. This chapter describes how the configuration spaces are accessed, then provides detailed descriptions of each register.

2.1 Access Mechanism

The PCI specification defines two bus cycles to access PCI configuration space: Configuration Read and Configuration Write. While memory and I/O spaces are supported by the microprocessor, configuration space is not directly supported. The PCI specification defines two mechanisms to access configuration space, Mechanism #1 and Mechanism #2. The 460GX chipset supports only Mechanism #1.

Mechanism #1 defines two I/O-space locations: an address register (CONFIG_ADDRESS) at location 0CF8h, and a data register (CONFIG_DATA) at location 0CFCh. Dword I/O Writes to the configuration address are latched and held; they specify the PCI Bus Number, Device Number within the bus, and Register Number within the device. Subsequent I/O reads and writes to the configuration data location cause a configuration space access the register specified by the address stored in the configuration address location.

Note: The AGP bus under the GXB looks like a standard PCI bus for configuration purposes. The term xXB refers to the PXB, WXB, or GXB. In general, any reference to an access to PCI bus includes accesses to an AGP bus.

Configuration space accesses are processed as follows:

- If the SAC detects that the I/O request is a configuration access to its own configuration space, it will service that request entirely within the SAC or the other chipset components. Reads result in data being returned to the system bus.
- If the SAC detects that the I/O request is a configuration access to a xXB configuration space, it will forward the request to the appropriate xXB for servicing. The request is not forwarded

to a PCI bus. Reads result in data being returned by the xXB through the SAC to the system bus.

- Otherwise, the access is forwarded to the xXB to be placed on the PCI bus (or AGP bus) as a Configuration Read or Configuration Write cycle. Reads will result in data being returned through the xXB and SAC back to the system bus, just as in normal Outbound Read operations.

2.2 Access Restrictions

The 460GX chipset supports PCI configuration space access using the mechanism denoted as Configuration Mechanism #1 in the PCI specification.

The 460GX chipset internal registers (both I/O Mapped and Configuration registers) are accessible by the Host CPU. The registers can be accessed as Byte, Word (16-bit), or Dword (32-bit) quantities, with the exception of CONFIG_ADDRESS which can only be accessed as a Dword. All multi-byte numeric fields use “little-endian” ordering (i.e. lower addresses contain the least significant parts of the field).

2.2.1 Partitioning

Each SAC, SDC, MAC, PXB, WXB, GXB, each AGP bus below an GXB, and each PCI bus below an PXB or WXB, has an independent configuration space. None of the registers are shared between the spaces; that is, the SAC, and each PCI bus in the PXB, have separate control and status registers.

Configuration registers are accessed using an “address” comprised of the PCI Bus Number, the Device Number within the bus, and the Register Number within the Device.

Accesses to devices on Bus #0 and Bus #(CBN) are serviced by the 460GX chipset depending on their device number. Device 10h on Bus #0 is mapped to the SAC; it contains the programmable Chipset Bus Number. All other chipset devices reside on bus CBN.

The DEVNPRES register is used to determine which chipset devices are present; see Table 2-1 for mapping information.

Table 2-1. Device Mapping on Bus CBN

No.	Device	No.	Device
00h	SAC	10h	Expander 0, Bus a[a]
01h	SAC	11h	Expander 0, Bus b
02h	reserved	12h	Expander 1, Bus a
03h	reserved	13h	Expander 1, Bus b
04h	SDC	14h	Expander 2, Bus a
05h	Memory Card A	15h	Expander 2, Bus b
06h	Memory Card B	16h	Expander 3, Bus a
07h	reserved	17h	Expander 3, Bus b
08h-0Fh	reserved	18h-1Fh	reserved

a. This is the compatibility bus (where the boot vector is always directed).

Configuration registers located in the SDC are accessed over the private data bus. The SAC translates CF8/CFC accesses to SDC registers into configuration commands over the PDB. Configuration registers located on the memory boards are accessed over the I2C port. The SAC

translates CF8/CFC accesses to the MAC registers into read/write commands over the I2C port. The SAC also contains an IIADR pointer register that can be used in conjunction with a CF8/CFC access to generate I2C commands to generic I2C devices on the memory boards.

2.2.2 Register Attributes

Registers have designated “access attributes”, with the following definitions:

Read Only	Writes to this register have no effect.
Read/Write	Data may be read from and written to this register. Selected bits in the register may be designated as “hardwired” or “read-only”; such bits are not affected by data writes to the register.
Read/Clear	Data may be read from the register. A data write operates strictly as a clear: a “1”-bit in the data field clears the corresponding bit in the register, while a “0”-bit in the data field has no effect on the corresponding bit in the register. Selected bits in the register may be designated as “hardwired” or “read-only”; such bits are not affected by data writes to the register.
Sticky	Data in this register remains valid and unchanged, during and following a hard reset. Typically, these registers contain special configuration information or error logs.

2.2.3 Reserved Bits Defined in Registers

Most 460GX chipset registers described in this section contain reserved bits. The PCI specification requires that software correctly handle reserved fields, as follows. On reads, software must use appropriate masks to extract the defined bits and not rely on reserved bits being any particular value. On writes, software must ensure that the values of reserved bit positions are preserved. That is, the values of reserved bit positions must first be read, merged with the new values for other bit positions and then written back. Note the software does not need to perform read, merge, write operation for the CONFIG_ADDRESS register.

2.2.4 Reserved or Undefined Register Locations

In addition to reserved bits within a register, the 460GX chipset contains address locations in the PCI configuration space that are marked “Reserved” or are simply undefined. Several of the 460GX chipset devices are multi-function devices; all registers in the unused functions are considered “Reserved”. Reserved registers can be 8-, 16-, or 32-bit in size. The PCI specification requires that the 460GX chipset respond to accesses to these address locations by completing the host cycle. Reserved register locations must be treated by software the same as reserved fields are treated: software can not rely on reads returning any particular value, and must not attempt to change the value returned when read.

2.2.5 Default Upon Reset

Upon reset, the 460GX chipset sets its internal configuration registers to predetermined **default** states. The default state represents the minimum functionality feature set required to successfully bring up the system. Hence, it does not represent the optimal system configuration. It is the responsibility of the system initialization software (firmware) to properly determine the DRAM configurations, operating parameters and optional system features that are applicable, and to program the 460GX chipset registers accordingly.

2.2.6 Consistency

There are a number of registers that are repeated in both the SAC and xXB/PCI spaces. It is software's responsibility to insure that these registers are programmed in a consistent fashion. Failure to insure consistency can produce indeterminate results. See the Initialization Chapter for an overview on initializing all chipset components.

When the address decode ranges of 460GX chipset devices are being updated, no other bus traffic is allowed over the address ranges being affected by the update. This means that the code that updates initial configuration must be executing from a location that will not be affected by the update. Furthermore in a multiprocessor system, precautions should be taken to assure that only one CPU is accessing configuration space at a time.

2.2.7 GART Programming Region

The region starting at FE20_0000h is used for programming the GARTs. This region is accessible either by the processor or PCI. See Section 7.2.1 for GART programming details

2.3 I/O Mapped Registers

The 460GX chipset contains two registers that reside in the CPU I/O address space: the Configuration Address (CONFIG_ADDRESS) Register and the Configuration Data (CONFIG_DATA) Register. The Configuration Address Register enables/disables the configuration space and determines what portion of configuration space is visible through the Configuration Data window. The following sections define the fields within the CONFIG_ADDRESS and CONFIG_DATA registers. The 460GX chipset's device ID mapping into the CONFIG_ADDRESS definition is shown in Table 2-1.

2.3.1 CONFIG_ADDRESS: Configuration Address Register

I/O Address:	CF8h [Dword]	Size:	32 bits
Default Value:	00000000h	Attribute:	Read/Write
Sticky:	No	Locked:	No

CONFIG_ADDRESS is a 32 bit register accessed only when referenced as a Dword. A Byte or Word reference will "pass through" the Configuration Address Register onto the PCI bus as an I/O cycle. The CONFIG_ADDRESS register contains the Bus Number, Device Number, Function Number, and Register Number for which a subsequent configuration access is intended.

Bits	Description
31	**Configuration Enable(CFGE).** When this bit is set to 1 accesses to PCI configuration space are enabled. If this bit is reset to 0 accesses to PCI configuration space are disabled.
30:24	*reserved* (0)
23:16	**Bus Number.** When the Bus Number is programmed to match the Chipset Bus Number (CBN), the target of the Configuration Cycle is the 460GX chipset. If the Bus Number is not CBN, the destination and type of access is determined by the Bus Number and Subordinate Bus Number of each PCI port in each PXB. A type 0 access is generated on the appropriate PCI bus if one of the PXB port's bus number is matched. Otherwise, a type 1 configuration cycle is generated on the appropriate PCI bus below the PXB port whose

subordinate bus number is in that range. For a type 1 cycle, the Bus Number is mapped to AD[23:16] during the address phase.

15:11 **Device Number.**
This field selects one agent on the PCI bus selected by the Bus Number. Device 16 (10h) on Bus #0 is always reserved for programming the CBN. On the bus that the chipset is mapped into (determined by the CBN register), Device Numbers 0-31 are reserved for the 460GX chipset components as shown in Table 2-1. All other devices numbers are forwarded to the selected bus.

1 0:8 **Function Number.**
This field is mapped to AD[10:8] during PCI configuration cycles. This allows the configuration registers of a particular function in a multi-function device to be accessed.

7:2 **Register Number.**
This field selects one register within a particular Bus, Device, and Function as specified by the other fields in the Configuration Address Register. This field is mapped to AD[7:2] during PCI configuration cycles.

1:0 *reserved* (0)

2.3.2 CONFIG_DATA: Configuration Data Register

I/O Address:	CFCh	Size:	32 bits
Default Value:	00000000h	Attribute:	Read/Write
Sticky:	No	Locked:	No

CONFIG_DATA is a 32 bit read/write window into configuration space. The portion of configuration space that is referenced by CONFIG_DATA is determined by the contents of CONFIG_ADDRESS.

Bits **Description**

31:0 **Configuration Data Window (**CDW**).**
If bit 31 of CONFIG_ADDRESS is 1 any I/O reference that falls in the CONFIG_DATA I/O space will be mapped to configuration space using the contents of CONFIG_ADDRESS.

2.4 Error Handling Registers

2.4.1 SAC

2.4.1.1 SECTID: SEC ITID

Bus CBN, Device Number:	00h	Function:	0
Address Offset:	80h	Size:	8 bits
Default Value:	00h	Attribute:	Read Only/Write Clear, Read/Write
Sticky:	Yes	Locked:	No

This register is used to capture the ITID for a single bit memory ECC error. The ITID can then be used to determine the address of the failure. To force the ITID to be cleared and re-used, write a 1 to bit 6. This register is set anytime that the SEC bit is sent from the SDC to the SAC on a ’Retire ITID’ command.

Bits	Description
7	**Disable** This bit can be written by software. When set, the ITID is retired immediately and not captured. Therefore there can be no checking of the address. See Section 6 for the usage of this bit.
6	**Valid** If set then the ITID in bits 5:0 is valid and shows the address of a single-bit memory error. Writing a 1 to this bit will clear the ITID and reset the valid bit.
5:0	**ITID** The ITID of the SEC error. These bits are read-only.

2.4.1.2 DEDTID: DED ITID

Bus CBN, Device Number:	00h	Function:	0
Address Offset:	81h	Size:	8 bits
Default Value:	00h	Attribute:	Read Only/Write Clear, Read/Write
Sticky:	Yes	Locked:	No

This register is used to capture the ITID for a double bit memory ECC error. The ITID can then be used to determine the address of the failure. To force the ITID to be cleared and re-used, write a 1 to bit 6. This register is set anytime that the DED bit is sent from the SDC to the SAC on a ‘Retire ITID’ command.

Bits	Description
7	**Disable** This bit can be written by software. When set, the ITID is retired immediately and not captured. Therefore there can be no checking of the address. See Section 6 for the usage of this bit.
6	**Valid** If set then the ITID in bits 5:0 is valid and shows the address of a double-bit memory error. Writing a 1 to this bit will clear the ITID and reset the valid bit.
5:0	**ITID** The ITID of the double-bit error. These bits are read-only.

2.4.1.3 FSETID: FSE ITID

Bus CBN, Device Number:	00h	Function:	0
Address Offset:	82h	Size:	8 bits
Default Value:	00h	Attribute:	Read Only/Write Clear, Read/Write
Sticky:	Yes	Locked:	No

This register is used to capture the ITID for a system bus data error. The ITID can then be used to determine the address of the failure. To force the ITID to be cleared and re-used, write a 1 to bit 6. This register is set anytime that the ADE bit is asserted and both SEC and DED are deasserted on a ’Retire ITID’ command from the SDC to the SAC. NOTE: this register is set for both processor-bus errors and errors on the SAC-to-SDC data bus.

Bits	Description
7	**Disable** This bit can be written by software. When set, the ITID is retired immediately and not captured. Therefore there can be no checking of the address. See Section 6 for the usage of this bit.
6	**Valid** If set then the ITID in bits 5:0 is valid and shows the address of a system bus data error. Writing a 1 to this bit will clear the ITID and reset the valid bit.
5:0	**ITID** The ITID of the system bus error. These bits are read-only.

2.4.1.4 FERR_SAC: First Error Status Register

Bus CBN, Device Number:	00h	Function:	1
Address Offset:	40h	Size:	32 bits
Default Value:	000000h	Attribute:	Read/Write Clear
Sticky:	Yes	Locked:	No

This register records the first error condition detected in the SAC/SDC.

Bits	Description
31	**Memory Card B Error (MBE)** Set when the memory card B signals a fatal error.
30	**Memory Card A Error (MAE)** Set when the memory card A signals a fatal error.
29	**XSERR# Asserted (XSA)** Set when the SAC sees the signal XSERR# active.
28	**'Store-Write' Command Underflow, card A, Stack L (SCAL)**
27	**'Store-Write' Command Underflow, card A, Stack R (SCAR)**
26	**'Store-Write' Command Underflow, card B, Stack L (SCBL)**
25	**'Store-Write' Command Underflow, card B, Stack R (SCBR)** One of these 4 is asserted when a signal is sent from the SDC to the SAC indicating write data was sent to the MDC, and there is no outstanding write in the SAC.
24	**SDC Correctable Memory Error (SCME)** Reports correctable DRAM errors (single-bit ECC errors). This bit does not mask other bits in the FERR register from being set. It is the one exception to the rule that only one bit in FERR may be set at a time.
23	**SDC Non-Fatal Error (SNE)** Reports non-fatal errors that are uncorrectable such as double-bit ECC error, or parity errors. This also reports a single-bit correctable error on the system bus. This bit will also be set if there is a second correctable error from memory in the SDC, and the first one has not been cleared by the time the second one occurs. The first correctable memory error would have set the SCME bit, and all later correctable memory errors (until the SDC's error registers are cleared) are reported as SNE in the FERR or NERR.
22	**SDC Fatal Error (SFE)** Fatal error in SDC.
21	**'Completion' Command Underflow; MAC A, Stack L (CCAL)**
20	**'Completion' Command Underflow; MAC A, Stack R (CCAR)**
19	**'Completion' Command Underflow; MAC B, Stack L (CCBL)**

18 **'Completion' Command Underflow; MAC B, Stack R (CCBR)**
One of these 4 bits is set when the SAC receives a completion from the MAC and the SAC has no outstanding transaction.

17 **BERR# Observed (BER)**
BERR# seen on the system bus. Set whenever BERR# is observed active.

16 **IOQ Underflow/Overflow (IUE)**
Set when the IOQ is empty and the SDC sends out a signal saying it popped something from the top of the queue. Or set when the IOQ is 8 (or 1 when the IOQ depth is set to 1) and an ADS# is seen on the bus.

15 ***reserved(0)***

14 **External XBINIT# Active**. **(XBE)**
Set when XBINIT# is seen active. This signal is from an Expander port or other external agent.

13 **False Retirement (FRE)**
Retirement from SDC that doesn't match an outstanding ITID in the SAC.

12 **Address above TOM (TE)**
Asserted when an address on the system bus is above TOM and not inside the I/O gap below 4 GB.

11 **Illegal HITM# (IHS)**
HITM# on non-memory access.

10 **Unsupported ASZ[1:0]# (ASE)**
Processor access to an address above 64 GB, so that ASZ# = 10b or 11b.

9 **System Bus Address Parity Error (AE)**
Parity error on A[36:3]#.

8 **System Bus Request Parity Error (RQE)**
Parity error on REQ[4:0]#.

7 **PDB ITID Parity Error (IPE)**
Parity error on the ITID bus from SDC to SAC.

6 **Retirement Bus Parity Error (RPE)**
Parity error on the retirement bus from the SDC to the SAC.

5 **Lock# Transaction With No Resources Available (LTE)**
Set when a LOCK# transaction occurs and there are no outbound resources available in which to place the lock.

4:1 *reserved(0)*

0 **Resource Counter Overflow/Underflow (RCE)**
Set if the resource counter has an underflow or overflow.

2.4.1.5 NERR_SAC: All Error Status Register

Bus CBN, Device Number:	00h	Function:	1
Address Offset:	44h	Size:	32 bits
Default Value:	000000h	Attribute:	Read/Write Clear
Sticky:	Yes	Locked:	No

This register records all error conditions detected in the SAC/SDC.

Bits **Description**

31:0 **See FERR_SAC for bit definitions.**

2.4.1.6 SA_FERR: System Address on First Error

Bus CBN, Device Number:	00h	Function:	1
Address Offset:	60h	Size:	128 bits
Default Value:	undefined after	Attribute:	Read Only
Sticky:	Yes	Locked:	No

This register records and latches the address for the first system bus error detected.

Bits	Description
127:107	*Reserved (0)*
106	**LOCK, ’b’ phase.**
105	**ADS, ’b’ phase**.
104	**RP#, ’b’ phase**.
103:99	**REQ, ’b’ phase**.
98	**AP1; ’b’ phase.**
97	**AP0; ’b’ phase.**
96:64	**A[35:3]#, ’b’ phase**.
63:43	*Reserved (0)*
42	**LOCK#, ’a’ phase**.
41	**ADS#, ’a’ phase.**
40	**RP# for REQa#**. Parity on REQa# signals.
39:35	**REQa#.** REQa signals on error.
34:33	**AP[1:0]#, ’a’ phase**. Address parity for failing address.
32:0	**Aa[35:3]#, ’a’ phase.** System Bus - System Address of Error.

2.4.1.7 BIUITID: BIU ITID Register

Bus CBN, Device Number:	00h	Function:	1
Address Offset:	80h	Size:	8 bits
Default Value:	0	Attribute:	Read/Write
Sticky:	No	Locked:	No

A write to this register causes the SAC to update the BIUDATA register with the contents of the CAM and RAM associated with the ITID that is written into this register.

Bits	Description
7:6	*reserved (0)*
5:0	**ITID** This is the ITID that is used to address the CAM/RAM structure.

2.4.1.8 BIUDATA: BIU Data Register

Bus CBN, Device Number:	00h	Function:	1
Address Offset:	90h	Size:	128 bits
Default Value:	undefined	Attribute:	Read Only
Sticky:	No	Locked:	No

This is the contents of the CAM concatenated with the contents of the RAM associated with the ITID in BIUITID.

Bits	Description
127:116	*reserved(0)*
115:82	**Address bits [35:2].** This is the contents of the CAM with address bits [5:2] from the RAM (bit 2 is only of interest if the transaction came from an Expander bus).
81:76	*reserved(0)*
75:71	**Reqa.** Request phase a[4:0].
70:63	**DID.** The DID for the transaction.
62:55	**BE.** The byte enables for the transaction.
54	*reserved (0)*
53	**OWN.** OWN# active.
52	**DPS.** DPS# active.
51:49	**Reqb.** Request phase b bits [4:2].
48	**Lock.** The transaction.had LOCK# asserted.
47	**LockLoad.** The transaction is the first occurrence of an OB lock sequence.
46:43	**Dst.** The destination of the transaction.
42	**ORetry.** A retry due to HITO.
41:36	**CMD.** The command for the transaction.
35	**P2P.** Set for peer-to-peer transactions.
34	**FEorR.** The end-of-request bit from the Expander port.
33:30	**FRoute.** The Expander bus route.
29:22	**FLEN.** The length on the Expander bus.
21:12	**FTID.** The Expander id.
11:9	**Len.** The length of the transaction.
8	**System Bus Retry.** The transaction was retried on the bus.
7	**Dfr.** The transaction is deferred.
6	**MEM.** The target for the transaction if memory.
5	**System Bus.** The target for the transaction is the system bus.
4	**CLINE.** The transaction is for a full line.
3	**Zero.** If set then this is a 0-length transaction.
2:0	**RS.** The Response generated for the transaction by the BIU. This may not match the system bus response sent, since the BIU's response may be changed by the MIU. This may be for a HITM# or other reasons.

Note: Note: if the P2P bit is not set, then bits [34:12] and [76] are not defined, since the transaction originated on the system bus and not the Expander bus.

2.4.2 SDC

2.4.2.1 SEC0_D_FERR: Data on First Memory Card B SEC

Bus CBN, Device Number:	04h		
Address Offset:	40-47h	Size:	64 bits
Default Value:	0	Attribute:	Read Only, New Value Latched anytime appropriate FERR register bit is set

This register records and latches the data corresponding to the first SEC detected by memory interface 0 in the SDC.

Bits	Description
63:0	**DE** - System Data of Error.

2.4.2.2 SEC0_ECC_FERR: ECC on First Memory Card B SEC

Bus CBN, Device Number:	04h		
Address Offset:	48h	Size:	8 bits
Default Value:	00h	Attribute:	Read Only, New Value Latched anytime appropriate FERR register bit is set

This register records and latches the ECC checkbits corresponding to the first SEC detected by memory interface 0 in the SDC

Bits	Description
7:0	**ECC** - ECC of Error.

2.4.2.3 SEC0_TXINFO_FERR: TXINFO on First Memory Card B SEC

Bus CBN, Device Number:	04h		
Address Offset:	49-4Ah	Size:	16 bits
Default Value:	00h	Attribute:	Read Only, New Value Latched anytime appropriate FERR register bit is set

This register records the ITID and failing chunk corresponding to the first SEC detected by memory interface 0 in the SDC.

Bits	Description
15:9	reserved(0)
8:6	**DC** - Data Chunk of ITID.
5:0	**ITID** - ITID of error.

2.4.2.4 DED0_D_FERR: Data on First Memory Card B DED

Bus CBN, Device Number:	04h		
Address Offset:	50-57h	Size:	64 bits
Default Value:	0	Attribute:	Read Only, New Value Latched anytime appropriate FERR register bit is set

This register records and latches the data corresponding to the first DED detected by memory interface 0 in the SDC.

Bits	Description
63:0	**DE** - System Data of Error.

2.4.2.5 DED0_ECC_FERR: ECC on First Memory Card B DED

Bus CBN, Device Number:	04h		
Address Offset:	58h	Size:	8 bits
Default Value:	00h	Attribute:	Read Only, New Value Latched anytime appropriate FERR register bit is set

This register records and latches the ECC checkbits corresponding to the first SEC detected by memory interface 0 in the SDC

Bits	Description
7:0	**ECC** - ECC of Error.

2.4.2.6 DED0_TXINFO_FERR: TXINFO on First Memory Card B DED

Bus CBN, Device Number:	04h		
Address Offset:	59-5Ah	Size:	16 bits
Default Value:	00h	Attribute:	Read Only, New Value Latched anytime appropriate FERR register bit is set

This register records the ITID and failing chunk corresponding to the first DED detected by memory interface 0 in the SDC.

Bits	Description
15:9	*reserved(0)*
8:6	**DC** - Data Chunk of ITID.
5:0	**ITID** - ITID of error.

2.4.2.7 SEC1_D_FERR: Data on First Memory Card A SEC

Bus CBN, Device Number:	04h		
Address Offset:	60-67h	Size:	64 bits
Default Value:	0	Attribute:	Read Only, New Value Latched anytime appropriate FERR register bit is set

This register records and latches the data corresponding to the first SEC detected by memory interface 1 in the SDC.

Bits	Description
63:0	**DE** - System Data of Error.

2.4.2.8 SEC1_ECC_FERR: ECC on First Memory Card A SEC

Bus CBN, Device Number:	04h		
Address Offset:	68h	Size:	8 bits
Default Value:	00h	Attribute:	Read Only, New Value Latched anytime appropriate FERR register bit is set

This register records and latches the ECC checkbits corresponding to the first SEC detected by memory interface 1 in the SDC.

Bits	Description
7:0	**ECC** - ECC of Error.

2.4.2.9 SEC1_TXINFO_FERR: TXINFO on First Memory Card A SEC

Bus CBN, Device Number:	04h		
Address Offset:	69-6Ah	Size:	16 bits
Default Value:	00h	Attribute:	Read Only, New Value Latched anytime appropriate FERR register bit is set

This register records the ITID and failing chunk corresponding to the first SEC detected by memory interface 1 in the SDC.

Bits	Description
15:9	*reserved(0)*
8:6	**DC** - Data Chunk of error.
5:0	**ITID** - ITID of error.

2.4.2.10 DED1_D_FERR: Data on First Memory Card A DED

Bus CBN, Device Number:	04h		
Address Offset:	70-77h	Size:	64 bits
Default Value:	0	Attribute:	Read Only, New Value Latched anytime appropriate FERR register bit is set

This register records and latches the data corresponding to the first DED detected by memory interface 1 in the SDC.

Bits	Description
63:0	**DE** - System Data of Error.

2.4.2.11 DED1_ECC_FERR: ECC on First Memory Card A DED

Bus CBN, Device Number:	04h		
Address Offset:	78h	Size:	8 bits
Default Value:	00h	Attribute:	Read Only, New Value Latched anytime appropriate FERR register bit is set

This register records and latches the ECC checkbits corresponding to the first DED detected by memory interface 0 in the SDC.

Bits	Description
7:0	**ECC** - ECC of Error.

2.4.2.12 DED1_TXINFO_FERR: TXINFO on First Memory Card A DED

Bus CBN, Device Number:	04h		
Address Offset:	79-7Ah	Size:	16 bits
Default Value:	00h	Attribute:	Read Only, New Value Latched anytime appropriate FERR register bit is set

This register records the ITID and failing chunk corresponding to the first DED detected by memory interface 1 in the SDC.

Bits	Description
15:9	*reserved(0)*
8:6	**DC** - Data Chunk of ITID.
5:0	**ITID** - ITID of error.

2.4.2.13 SDC_FERR: First Error Status Register

Bus CBN, Device Number:	04h		
Address Offset:	80-83h	Size:	32 bits
Default Value:	0000h	Attribute:	Read/Write to Clear

This register records the first error condition detected in the SDC. Writing a ’1’ to this register will clear the bit in both SDC_FERR and the same bit in SDC_NERR.

Bits	Description
31	Simultaneous S/W write-one-to-clear and H/W error detected in the same cycle. This bit will only be set if another bit is also set. This implies that the ERROR>_<TYPE>_FERR data registers associated with the other asserted bit contain stale data.
30	**PDB Receive Length Error (RLE)** Private Bus receive length error
29	**DRDY# Protocol Error (FS2)** Asserted when a protocol error is found involving DRDY#, SBUSY# and DBUSY#.
28	**Write Data Protocol Error (FS1)** Asserted on write protocol errors.
27	**LEN# Protocol Error (FS0)** Asserted on mismatches of LEN# field and actual data transmitted.

26 **'Forward' Overlapping 'Forward'; Card A (FWMDI1)**
Indicates FWMDI sampled asserted while a store transaction is in progress

25 **'Load' Overlapping 'Load'; Card A (LRMDI1)**
Indicates LRMDI sampled asserted while a store transaction is in progress

24 **'Load' Overlapping 'Forward'; Card A (WrRd1)**
Memory interface 1 detected simultaneous read and write operation. Write and Read collision.

23 **'Forward' Overlapping 'Load'; Card A (RdWr1)**
Memory interface 1 detected simultaneous read and write operation. Read and write collision.

22 **'Forward' Underflow; Card A Right Stack Error (FR1)**
Memory interface 0 received Forward right Bank without corresponding Store command

21 **'Forward' Underflow; Card A Left Stack Error (FL1)**
Memory interface received Forward left Bank without corresponding Store command

20 **'Accept Underflow'; Card A (AE1)**
Memory interface 0 received data without corresponding Accept command

19 **'Forward' Overlapping 'Forward'; Card B (FWMDI0)**
Indicates FWMDI sampled asserted while a store transaction is in progress

18 **'Load' Overlapping 'Load'; Card B (LRMDI0)**
Indicates LRMDI sampled asserted while a store transaction is in progress

17 **'Load' Overlapping 'Forward'; Card B (WrRd0)**
Memory interface 1 detected simultaneous read and write operation. Write and Read collision.

16 **'Forward' Overlapping 'Load'; Card B (RdWr0)**
Memory interface 1 detected simultaneous read and write operation. Read and write collision.

15 **'Forward' Underflow; Card B Right Stack Error (FR0)**
Memory interface 0 received Forward right Bank without corresponding Store command

14 **'Forward' Underflow; Card B Left Stack Error (FL0)**
Memory interface received Forward left Bank without corresponding Store command

13 **'Accept' Underflow; Card B (AE0)**
Memory interface 0 received data without corresponding Accept command

12 **Configuration Information Parity Error (CIE)**
Data buffer detected data parity while reading config address or data from SDC RAM.

11 **Response Bus Transmission Error (RTE)**
Indicates that the SDCRSP bus detected a transmission error.

10 **PDB - ITID Parity Error (IPE)**
Look in ITID_FERR Register to isolate.

9 **PDB - Command Parity Error (CPE)**
Look in CMD_FERR Register to isolate.

8 **PDB Byte Enable Parity Error (BPE)**
Parity error on the Byte-enables from the SAC.

7 **SDC Data Buffer RAM Parity Error (RPE)**
SDC detected bad parity on good data stored in its data buffer. Indicates potential RAM cell disturbance due to alpha or cosmic hit. All four data port map to this bit.

6 **PDB - Data Parity Error** (DPE)
Parity Error Detected on transfer of Data from SAC to SDC.

5	**System Bus Double Bit Error** (DEDF) ECC Double Bit Error Detected on system bus.
4	**System Bus Single Bit Error** (SECF) ECC Single Bit Error Detected on system bus.
3	**SDC Card A Double Bit Error** (DED1) ECC Double Bit Error Detected from Memory Card A.
2	**SDC Card A Single Bit Error** (SEC1) ECC Single Bit Error Detected from Memory Card A.
1	**SDC Card B Double Bit Error** (DED0) ECC Double Bit Error Detected from Memory Card B.
0	**SDC Card B Single Bit Error** (SEC0) ECC Single Bit Error Detected from Memory Card B.

2.4.2.14 SDC_NERR: SDC Next Error Status Register

Bus CBN, Device Number:	04h		
Address Offset:	84-87h	Size:	32 bits
Default Value:	0000h	Attribute:	Read/Write to Clear

This register records the next error status within the SDC. Writing a ’1’ to this register will clear the bit in both SDC_NERR and the same bit in SDC_FERR.

Bits	Description
31:0	See SDC_FERR for bit definitions.

2.4.2.15 PCMD_FERR: Command on First PCMD Parity Error

Bus CBN, Device Number:	04h		
Address Offset:	88-8Bh	Size:	32 bits
Default Value:	00h	Attribute:	Read Only, New Value Latched anytime appropriate FERR register bit is set

This register records and latches the data associated with the first parity error detected on the PCMD bus.

Bits	Description
31:19	reserved(0)
18	If set then the error was detected on the 1st half of the double–pumped transfer. Otherwise, these fields contain the information from the 2nd half of the double-pumped transfer.
17	Parity of Error
16:0	PCMD - Private Data Command value of Error.

2.4.2.16 PITID_FERR: Data on First PITID Parity Error

Bus CBN, Device Number:	04h		
Address Offset:	8Ch	Size:	8 bits
Default Value:	0h	Attribute:	Read Only, New Value Latched anytime appropriate FERR register bit is set

This register records and latches the data associated with the first parity error detected on the PITID bus.

Bits	Description
7	If set then the error was detected on the 1st half of the double-pumped transfer. Otherwise, these fields contain the information from the 2nd half of the double-pumped transfer.
6	Parity of Error
5:0	PITID - Private ITID bus value of Error.

2.4.2.17 SDCRSP_FERR: Response on First SDCRSP Error

Bus CBN, Device Number:	04h		
Address Offset:	8Dh	Size:	8 bits
Default Value:	0h	Attribute:	Read Only, New Value Latched anytime appropriate FERR register bit is set

This register records and latches the data and inverted data associated with the first transmission error detected on the SDCRSP bus.

Bits	Description
7:4	Response Bus for 2nd half of double–pumped transfer.
3:0	Response Bus for 1st half of double–pumped transfer.

2.4.2.18 DPBRLE_FERR: Private Data Bus Receive Length Error

Bus CBN, Device Number:	04h		
Address Offset:	8Eh	Size:	8 bits
Default Value:	0h	Attribute:	Read Only, New Value Latched anytime appropriate FERR register bit is set

This register indicates that the amount of data transferred from the SAC to the SDC for a given transfer did not match the expected transfer length.

Bits	Description
7:3	*reserved(0)*
2	Data packet longer than expected (LDP)
1	Data packet shorter than expected (SDP)
0	No data packet shipped as expected (NDP)

2.4.2.19 ECCMSK0: ECC Mask Register - Card B

Bus CBN, Device Number:	04h		
Address Offset:	C8h	Size:	8 bits
Default Value:	00h	Attribute:	Read/Write

This register is used to test the ECC error detection logic in the memory subsystem for memory card 0. To test, this register is written with a masking function. All subsequent writes into memory will store a masked version of the computed ECC. Subsequent reads of memory locations written

while masked will return an invalid ECC code. To disable testing, the mask value is left at 0h (the default). The mask is a bit-wise XOR with the computed ECC.

Bits	Description
7:0	**ECC Generation Mask -** For 64 bits of data**.**

2.4.2.20 ECCMSK1: ECC Mask Register - Card A

Bus CBN, Device Number:	04h		
Address Offset:	C9h	Size:	8 bits
Default Value:	00h	Attribute:	Read/Write

This register is used to test the ECC error detection logic in the memory subsystem for memory card 1. To test, this register is written with a masking function. All subsequent writes into memory will store a masked version of the computed ECC. Subsequent reads of memory locations written while masked will return an invalid ECC code. To disable testing, the mask value is left at 0h (the default). The mask is a bit-wise XOR with the computed ECC.

Bits	Description
7:0	**ECC Generation Mask -** For 64 bits of data**.**

2.4.2.21 ECCMSKF: ECC Mask Register

Bus CBN, Device Number:	04h		
Address Offset:	CAh	Size:	8 bits
Default Value:	00h	Attribute:	Read/Write

This register is used to test the ECC error detection logic of the host processor bus. To test, this register is written with a masking function. All subsequent processor reads will received a masked version of ECC code. To disable testing, the mask value is left at 0h (the default). The mask is a bit-wise XOR with the computed ECC.

Bits	Description
7:0	**ECC Generation Mask -** For 64 bits of data**.**

2.4.2.22 ParMskP: PB Parity Mask and IB Correction Enable Register

Bus CBN, Device Number:	04h		
Address Offset:	CBh	Size:	8 bits
Default Value:	00h	Attribute:	Read/Write

The first 4 bits of this register are used to test the data parity error detection logic of the private bus. To test, bits 3:0 are written with a masking function. All subsequent private bus reads will receive a masked version of double byte parity. To disable testing, the mask value is left at 0h (the default). The mask is a bit-wise XOR with the computed parity.

Bits 7:4 are used to enable ECC or parity checking for the different busses. Note that the SDC defaults to no parity or ECC checking at power-on.

Bits	Description
7	Private Bus parity detection enable.
6	Front Side Bus ECC correction/detection enable.

5	Memory Bus A ECC correction/detection enable.
4	Memory Bus B ECC correction/detection enable.
3:0	Double byte parity mask for 128 bits of data.

2.4.2.23 PVD_D_FERR: Data on First PVD Parity Error

Bus CBN, Device Number:	04h		
Address Offset:	D0-D7h	Size:	64 bits
Default Value:	0	Attribute:	Read Only, New Value Latched anytime appropriate FERR register bit is set

This register records and latches the data associated with the first parity error detected on the PVD bus.

Bits	Description
63:0	PVD - Private Data Bus data.

2.4.2.24 PVD_PAR_FERR: Parity on First PVD Parity Error

Bus CBN, Device Number:	04h		
Address Offset:	D8h	Size:	8 bits
Default Value:	0	Attribute:	Read Only, New Value Latched anytime appropriate FERR register bit is set

This register records and latches the data associated with the first parity error detected on the PVD bus.

Bits	Description
7:4	*reserved(0)*
3:0	Double–byte parity of error

2.4.2.25 PVD_TXINFO_FERR: TXINFO on First PVD Parity Error

Bus CBN, Device Number:	04h		
Address Offset:	D9-DAh	Size:	16 bits
Default Value:	00h	Attribute:	Read Only, New Value Latched anytime appropriate FERR register bit is set

This register records the ITID and failing chunk corresponding to the first double-byte parity detected by private bus interface in the SDC.

Bits	Description
15:9	*reserved(0)*
8:6	**DC** - Data Chunk of ITID.
5:0	**ITID** - ITID of error.

2.4.2.26 SECF_D_FERR: Data on First System Bus SEC

Bus CBN, Device Number:	04h		
Address Offset:	E0-E7h	Size:	64 bits
Default Value:	0	Attribute:	Read Only, New Value Latched anytime appropriate FERR register bit is set

This register records and latches the data corresponding to the first SEC detected by system bus interface in the SDC.

Bits	Description
63:0	**DE** - System Data of Error.

2.4.2.27 SECF_ECC_FERR: ECC on First System Bus SEC

Bus CBN, Device Number:	04h		
Address Offset:	E8h	Size:	8 bits
Default Value:	00h	Attribute:	Read Only, New Value Latched anytime appropriate FERR register bit is set

This register records and latches the ECC checkbits corresponding to the first SEC detected by system bus interface in the SDC.

Bits	Description
7:0	**ECC** - ECC of Error.

2.4.2.28 SECF_TXINFO_FERR: TXINFO on First System Bus SEC

Bus CBN, Device Number:	04h		
Address Offset:	E9-EAh	Size:	16 bits
Default Value:	00h	Attribute:	Read Only, New Value Latched anytime appropriate FERR register bit is set

This register records the ITID and failing chunk corresponding to the first SEC detected by system bus interface in the SDC.

Bits	Description
15:9	*reserved(0)*
8:6	**DC** - Data Chunk of error.
5:0	**ITID** - ITID of error.

2.4.2.29 DEDF_D_FERR: Data on First System Bus DED

Bus CBN, Device Number:	04h		
Address Offset:	F0-F7h	Size:	64 bits
Default Value:	0	Attribute:	Read Only, New Value Latched anytime appropriate FERR register bit is set

This register records and latches the data corresponding to the first DED detected by system bus interface in the SDC.

Bits	Description
63:0	**DE** - System Data of Error.

2.4.2.30 DEDF_ECC_FERR: ECC on First System Bus DED

Bus CBN, Device Number:	04h		
Address Offset:	F8h	Size:	8 bits
Default Value:	00h	Attribute:	Read Only, New Value Latched anytime appropriate FERR register bit is set

This register records and latches the ECC checkbits corresponding to the first DED detected by system bus interface in the SDC.

Bits	Description
7:0	**ECC** - ECC of Error.

2.4.2.31 DEDF_TXINFO_FERR: TXINFO on First System Bus DED

Bus CBN, Device Number:	04h		
Address Offset:	F9-FAh	Size:	16 bits
Default Value:	00h	Attribute:	Read Only, New Value Latched anytime appropriate FERR register bit is set

This register records the ITID and failing chunk corresponding to the first DED detected by system bus interface in the SDC.

Bits	Description
15:9	*reserved(0)*
8:6	**DC** - Data Chunk of ITID.
5:0	**ITID** - ITID of error.

2.4.3 MAC

2.4.3.1 FERR_MAC: First Error Status Register

Bus CBN, Device Number:	05h,06h	Function Number:	00h,01h
Address Offset:	98h	Size:	8 bits
Default Value:	00h	Attribute:	Read

This register records the first error condition detected in the MAC.

Bits	Description
7:2	*reserved(0)*
1	**Que-Overflow Error** Signals that the MAC received too many commands from the SAC.

0 **Parity Error - CMND**
Parity Error Detected on SAC-MAC CMND Bus. Look in CMND_FERR Register to isolate. When the error is detected, the MAC will complete those operations which have a RAS pending, and stop. No new RAS cycles will be issued after the parity error and that card is effectively dead.

2.4.3.2 CMND_FERR: Command on First Error

Bus CBN, Device Number:	05h,06h	Function Number:	00h,01h
Address Offset:	9Ch	Size:	24 bits
Default Value:	0000h	Attribute:	Read

This register records and latches the data on the SAC-MAC Command Bus for the first error detected.

Bits	Description
23:22	reserved (0)
21:19	Row address [2:0]
18:17	Command [1:0]
16:0	MA[16:0]

2.4.4 PXB

2.4.4.1 ERRSTS: Error Status Register

Address Offset:	44h	Size:	8 bits
Default Value:	00h	Attribute:	Read/Write Clear, Sticky

This register records error conditions detected from the PCI bus (not already covered in PCISTS), from the Expander Bus, and performance monitoring events.

The register is sticky through reset; that is, the contents of the register remain unchanged during and following the assertion of X(0,1)RST#. This allows system recovery software invoked following a forced reset to examine the flags to determine the cause of an error. Once set, the flags remain set until explicitly cleared by software or a *power-good* reset.

Bits	Description
7	*reserved(0).*
6	**PERR# observed on PCI Bus** This flag is set if the PXB detects the PERR# input asserted, and the PXB was not the asserting agent. This flag may be configured to assert SERR# or PERR# in the ERRCMD register. This bit remains set until explicitly cleared by software writing a 1 to this bit.
5	**Parity Error on Received PCI Data** This flag is set if the PXB detects a parity error on data being read from the PCI bus. This flag may be configured to assert SERR# or PERR# in the ERRCMD register. This bit remains set until explicitly cleared by software writing a 1 to this bit.
4	**Parity Error on PCI Address** This flag is set if the PXB detects a parity error on the PCI address. This flag may be configured to assert SERR# in the ERRCMD register. This bit remains set until explicitly cleared by software writing a 1 to this bit.

3 **Inbound Delayed Read Time-out Flag**
Each inbound read request that is accepted and serviced as a delayed read (i.e. the PXB retries the request) will initiate a watchdog timer (2^{15} cycles, per the PCI spec). If the data has been returned and the timer expires before the requesting master initiates its repeat request, this flag will be set. This flag may be configured to assert SERR# or PERR# in the ERRCMD register. This bit remains set until explicitly cleared by software writing a 1 to this bit.

2 *reserved(0)*

1 **Performance Monitor #1 Event Flag**
This flag is set when the Performance Monitor #1 requests that an interrupt request be asserted. The PME and PMR registers (Section 2.5.3.3, Section 2.5.3.2) describe the conditions that can cause this to occur. While this bit is set, the INT(A,B)RQ# line will be asserted. This bit remains set until explicitly cleared by software writing a 1 to this bit.

0 **Performance Monitor #0 Event Flag**
This flag is set when the Performance Monitor #0 requests that an interrupt request be asserted. The PME and PMR registers (Section 2.5.3.3, Section 2.5.3.2) describe the conditions that can cause this to occur. While this bit is set, the INT(A,B)RQ# line will be asserted. This bit remains set until explicitly cleared by software writing a 1 to this bit.

2.4.4.2 ERRCMD: Error Command Register

Address Offset:	46h	Size:	8 bits
Default Value:	00h	Attribute:	Read/Write

This register provides extended control over the assertion of SERR# beyond the basic controls specified in the PCI-standard PCICMD register.

Bits	Description
7	*reserved(0)*
6	**Assert SERR# on Observed Parity Error** If set, the PXB asserts SERR# if PERR# is observed asserted, and the PXB was not the asserting agent.
5	**Assert SERR# on Received Data with Parity Error** If set, the PXB asserts SERR# upon receiving PCI data (i.e. an inbound write or outbound read) with a parity error. This occurs regardless of whether PXB asserts it's PERR# pin.
4	**Assert SERR# on Address Parity Error** If set, the PXB asserts SERR# on detecting a PCI address parity error.
3	**Assert PERR# on Data Parity Error** If set, and the PERRE bit is set in the PCICMD register, the PXB asserts PERR# upon receiving PCI data with parity errors.
2	**Assert SERR# on Inbound Delayed Read Time-out** Each inbound read request that is accepted and serviced as a delayed read (i.e. the PXB retries the request) will initiate a watchdog timer (2^{15} cycles, per the PCI spec). If this enable is set, the PXB will assert SERR# if the data has been returned and the timer expires before the requesting master initiates its repeat request. Default=0.
1	*reserved(0)*
0	**Return Hard Fail Upon Generating Master Abort** If set, the PXB will return a Hard Fail response through the SAC to the system bus after generating a master abort time-out for an outbound transaction placed on the PCI bus. If cleared, the PXB will return a normal response (with data of all 1's for a read). In either case, an error flag is set in the PCISTS register. Default=0.

2.4.5 GXB

2.4.5.1 FERR_GXB

Function Number:	BFN+1		
Address Offset:	80h	Size:	8 bits
Default Value:	00h	Attribute:	Read/Write Clear
Sticky:	Yes	Locked:	No

These registers record the first error detected by the GXB. For the order to clear this register with respect to the other GXB error registers.

Bits	Description
7:3	*reserved (0)*
2	**FERR_PCI bit asserted**
1	**FERR_AGP bit asserted**
0	**FERR_GART bit asserted**

2.4.5.2 FERR_PCI

Function Number:	BFN+1		
Address Offset:	84h	Size:	8 bits
Default Value:	00h each	Attribute:	Read/Write Clear
Sticky:	Yes	Locked:	No

These registers record the first error detected in GPI.

Bits	Description
7	**PCISTS Error Logged** This bit is asserted when an error, *except for a master abort*, has been logged in the PCI Status register.
6	**Non-Configuration Master Abort** This bit is asserted when a master abort occurs on any transaction other than a configuration read or configuration write. *reported the same as a master abort - see PCISTS register*
5	**Discard Timer Expiration** This is the 2^{15} clock timeout.
4	**SERR# Observed**
3	**PERR# Observed**
2	**PCI Inbound Read Que Data Parity Error** Parity error detected as read data is retrieved from buffer.
1	**PCI Outbound Write Que Data Parity Error** Parity error detected as write data is retrieved from buffer.
0	**Illegal OB GART Access** Access may continue or abort, results undefined.

2.4.5.3 FERR_AGP: First Error Status Register for AGP

Function Number:	BFN+1		
Address Offset:	85h	Size:	8 bits

Default Value:	00h	Attribute:	Read/Write Clear
Sticky:	Yes	Locked:	No

These registers record and latch the first error detected in the AGP interface.

Bits	Description
7:6	*reserved (0)*
5	**Lo-priority Read Data Que Parity Error** This is data returned to the graphics card out of the Low-priority buffer.
4	**Hi-priority Read Data Que Parity Error** This is data returned to the graphics card out of the Hi-priority buffer.
3	**Use of Pipe with Sideband Enabled**
2	**AGP address from graphics card [63:40] not equal to 0**
1	**AGP Request Queue Overflow** The GXB supports 16 outstanding requests. This bit is set if a new request is sent by the AGP card when the GXB already has 16 requests.
0	**Illegal AGP Command**

2.4.5.4 FERR_GART: First Error Status Register for GART

Function Number:	BFN+1		
Address Offset:	86h	Size:	8 bits
Default Value:	00h	Attribute:	Read/Write Clear
Sticky:	Yes	Locked:	No

These registers record and latch the first error detected in the AGP interface.

Bits	Description
7:4	*reserved (0)*
3	**GART Parity Error**.
2	**GART Entry Invalid**
1	**Illegal Address** (after GART translation) in range between GAPBAS and GAPTOP, or in VGA range and VGAGE is asserted, or directed by MARG to PCI instead of memory, or above TOM.
0	*reserved (0)*

2.4.5.5 NERR_AGP: Next Errors Status Register for AGP

Function Number:	BFN+1		
Address Offset:	8Dh	Size:	8 bits
Default Value:	00h	Attribute:	Read/Write Clear
Sticky:	Yes	Locked:	No

This register records all error conditions detected in the AGP interface after the first error. Errors recorded in FERR_AGP are not recorded here.

Bits	Description
7:0	See FERR_AGP for definition of these bits.

2.4.5.6 NERR_GART

Function Number:	BFN+1		
Address Offset:	8Eh	Size:	8 bits
Default Value:	00h each	Attribute:	Read/Write Clear
Sticky:	Yes	Locked:	No

This register records all error conditions detected in the GART logic after the first error. Errors recorded in FERR_GART are not recorded here.

Bits	Description
7:0	See FERR_GART for definitions of these bits.

2.4.5.7 PAC_ERR: PCI Address & Cmd First Error

Function Number:	BFN+1		
Address Offset:	A0h	Size:	64 bits
Default Value:	0000000000h each	Attribute:	Read/Write
Sticky:	Yes	Locked:	No

These registers record and latch the Address and Command information on the PCI Bus for the first error detected.

Bits	Description
63:46	*reserved(0)*
45	PCI Parity (2nd phase of DAC, not defined for non-DAC address).
44	PCI Parity (if DAC, this is the parity of the first half of the address).
43:40	**PCI Command** - Command of Error.
39:0	**PCI Address** - Address Received on Error. (possible DAC address).

2.4.5.8 PD_ERR: PCI Data First Error

Function Number:	BFN+1		
Address Offset:	A8h	Size:	64 bits
Default Value:	00h each	Attribute:	Read/Write
Sticky:	Yes	Locked:	No

These registers record and latch the Data and Byte Enable information on the PCI Bus for the first error detected.

Bits	Description
63:37	*reserved(0)*
36	PCI Parity.
35:32	**PCI Byte Enable [3:0]** - Byte Enable of Error.
31:0	**PCI Data** - Data of Error.

2.4.6 WXB

2.4.6.1 ERRSTS: Error Status Register

Address Offset:	44h	Size:	8 bits
Default Value:	00h	Attribute:	Read/Write Clear, Sticky

This register records certain error conditions detected from the PCI bus. This register is sticky through reset; that is, the contents of the register remain unchanged during and following the assertion of XRST#. This allows system recovery software invoked following a forced reset to examine the flags to determine the cause of an error. Once set, the flags remain set until explicitly cleared by software or a ***power-good*** reset.

Note: Bits 6, 4, and 2 are Reserved on side-b and records a zero value only!

Bits	Description
7	**INTRQ Asserted Flag** This flag is set if the WXB has initiated an INTRQ interrupt event. This bit remains set, and the event signaled, until explicitly cleared by software writing a 1 to this bit. Default = 0.
6	**XBINIT Asserted Flag** This flag is set if the WXB has asserted XBINIT#. This bit remains set, and the event signaled, until explicitly cleared by software writing a 1 to this bit. Default = 0.

Note: This bit is Reserved on side-b and records a zero value only!

5	**NEPCI Register Records an PCI Bus Error Flag** This flag is set when the PCI bus reports an error (e.g. data parity error) on transactions to/from other PCI bus agents. This bit remains set until explicitly cleared by software writing a 1 to this bit. This bit is only set when the error is reported through the NEPCI Next Error register, indicating that the error is ***not*** the first error occurrence since the First Error register was last cleared. Default = 0.
4	*reserved (0)*
3	**FEPCI Register Records an PCI Bus Error Flag** This flag is set when the PCI bus reports an error (e.g. data parity error) on transactions to/from other PCI bus agents. This bit remains set until explicitly cleared by software writing a 1 to this bit. This bit is only set when the error is reported through the FEPCI register, indicating that the error ***is*** the first error occurrence since the FEPCI register was last cleared. Default = 0.
2	*reserved (0)*
1	**Performance Monitor #1 Event Flag** This flag is set when the Performance Monitor #1 detects an event. The PCI_WXB_PMC1 registers describes the conditions that can cause this to occur. This bit remains set until explicitly cleared by software writing a 1 to this bit. Default = 0.
0	**Performance Monitor #0 Event Flag** This flag is set when the Performance Monitor #0 detects an event. The PCI_WXB_PMC0 registers describes the conditions that can cause this to occur. This bit remains set until explicitly cleared by software writing a 1 to this bit. Default = 0.

2.4.6.2 ERRCMD: Error Command Register

Address Offset:	45h– 46h	Size:	16 bits
Default Value:	8040h	Attribute:	Read/Write

This register provides extended control over the signalling of errors through SERR_OUT#, XBINIT#, and INTRQ#. These controls are in addition to the defined controls specified in the PCI-standard PCICMD register for SERR# assertion.

Bits	Description
15	**XBINITO: XBinit Override Enable** This bit should always be initially set to 0 by software. If set to 0, XBINIT# may be asserted by the WXB. The WXB will automatically set this bit after an XBINIT# is signaled. Default = 1

Note: Software should verify that there are no errors pending (by evaluating the ERRSTS register) before clearing this bit.

Note: This bit is Reserved on side-b and records a value of one only!

14	*reserved(0)*

Note: This bit is Reserved on side-b and records a value of zero only!

13	**IRQE: INTRQ Enable** Controls the reporting of WXB transmitted data errors. If set, the WXB will assert an INTRQ interrupt for observed PERR# (including IHPC-driven parity errors) and data parity errors detected in outbound transactions (e.g. Internal Queue Error detected during read by PCI interface). Default = 0.
12	**ASAPE: Assert SERR# on Address Parity Error** This bit should always be set to 1. When the WXB detects a PCI Address Parity Error and both SERRE and PERRE are set, SERR# (and SERR_OUT#) will be signaled. Default = 0.
11	**ASDPE: Assert SERR# on any Data Parity Error** If set, the WXB will assert SERR# (and SERR_OUT#) whenever a data parity error is detected in an inbound transaction. The SERRE bit in the PCICMD register must also be set for SERR# (and SERR_OUT#) to be signaled. Default = 0.
10	**ASDTE: Assert SERR# on Discard Timer Expiration** This bit should always be set to 1. When an inbound read Discard Timer Expiration occurs and SERRE is set, SERR# (and SERR_OUT#) will be signaled. Default = 0.
9:7	*reserved(0)*
6	*reserved(1)*
5:0	*reserved (0)*

2.4.6.3 FEPCI: PCI Bus First Error Status Register

Address Offset:	83h	Size:	8 bits
Default Value:	00h	Attribute:	Read/Write Clear, Sticky

This register records and latches the first error observed on the PCI bus. Once an error has been noted in this register, no further updates are allowed. This register is a write-1-to-clear register, meaning that software must write a 1 to the specific bit location it wishes cleared. The response to

each of these errors varies and is (generally) controlled through a combination of the PCICMD and ERRCMD registers. Refer to Section 6.12 for information on the conditional reporting of these errors via the SERR#, XBINIT#, or INTRQ# outputs.

Note, if multiple errors are observed very close in time multiple errors may be signaled as a first error.

Bits	Description
7	**PCILV: PCI Error Logs Valid** This flag is set when an error has been logged in the FEPCIAL and FEPCIDL log registers. The FEPCI status bit that lead to the logging must be cleared prior to clearing the PCILV flag. Default = 0.
6	**UMATA: Unexpected Master or Target Abort** This flag is set when an unexpected master abort or any target abort occurs. Master aborts are reported as described for the RMA bit in the PCISTS register. Target aborts through the RTA bit. Default = 0.
5	**DTE: Discard Timer Expiration** This flag is set when the 2^{15} clock timeout timer expires. This flag may be configured to assert SERR#, XBINIT#, or an INTRQ interrupt through the ERRCMD register. Default = 0.
4	**SES: System Error Signaled** This flag is set when the a PCI agent other than the WXB asserts SERR#. This flag may be configured to assert XBINIT#, or an INTRQ interrupt through the ERRCMD register. Default = 0
3	**PODT: PERR# Observed on PCI Data Transfer** This flag is set if the WXB detects the PERR# input asserted, and the WXB was not the asserting agent. This flag may be configured to assert SERR#, XBINIT#, or an INTRQ interrupt through the ERRCMD register. Default = 0.
2	*reserved (0)*
1	**PEOD: Parity Error on Received PCI Data** This flag is set if the WXB detects an parity error (i.e. calculates a parity different from what is provided with the data) on data being sent to the WXB on the PCI bus (from a master read or target write). Default = 0.
0	**PEPA: Parity Error on PCI Address** This flag is set if the WXB detects a parity error on the PCI address. Default = 0.

2.4.6.4 NEPCI: PCI Bus Next Error Status Register

Address Offset:	87h	Size:	8 bits
Default Value:	00h	Attribute:	Read/Write Clear, Sticky

This register records any PCI bus errors detected after the first error is observed and recorded in the FEPCI register. This register is a write-1-to-clear register, meaning that software must write a 1 to the specific bit location it wishes cleared. The response to each of these errors varies and is (generally) controlled through a combination of the PCICMD and ERRCMD registers. Error logging is ***not*** performed for Next Error occurrences.

Bits	Description
7:0	See the FEPCI register description for definitions. Error logging is ***not*** performed for Next Error occurrences.

2.4.6.5 FEPCIAL: PCI First Error Address/Command Log

Address Offset:	A5h–ADh	Size:	72 bits
Default Value:	000000000000000000h	Attribute:	Read/Write Clear, Sticky

These registers record and latch the address/command information, sent or received, associated with a PCI Bus error for the first error detected.

Bits	Description
71:68	*reserved (0)*
67:64	**C/BE[3:0]**
63:32	**AD[63:32]**
31:0	**AD[31:0]**

2.4.6.6 FEPCIDL: PCI First Error Data Log

Address Offset:	AFh – B3h	Size:	40 bits
Default Value:	0000000000h	Attribute:	Read/Write Clear, Sticky

These registers record and latch the PCI information, sent or received, specifically associated with the PCI bus error for the first error detected. The recorded data contains the upper or lower AD and C/BE and PAR signals.

Bits	Description
39:37	*reserved (0)*
36	**PAR**
35:32	**C/BE**
31:0	**AD**

2.5 Performance Monitor Registers

2.5.1 SAC

2.5.1.1 IT_MON_PMD_[0 to 5]: Internal Transaction Performance Monitor Data Register

Bus CBN, Device Number:	00h	Function:	2
Address Offset:	90-97h, 98-9Fh, A0-A7h, A8-AFh, B0-B7h, B8-BFh	Size:	64 bits each
Default Value:	0 each	Attribute:	Read/Write
Sticky:	No	Locked:	No

The value written to this address, loads the counter and is also saved in a reload register. Each counter can be configured to reload the data when it overflows.

The IT_MON_PMD_[0 to 5] registers hold the performance monitoring count values. 39-bits of the counter are used for event counting, the 40th-bit is used as a overflow detection bit. The 39-bit count value allows up to 70 minutes of event collection at 133 MHz. Event selection is controlled by the PMC registers (Section 2.5.1.2).

Each counter may be stopped/started independently, using the controls available in the associated PMC register.

Bits	Description
63:40	*reserved(0)*
39	**Overflow** This bit is asserted when the Event Count bit 38 carries into bit 39.
38:0	**Count Value** This register contains the Performance Monitor Data Register. You may preset the value of the performance counter by writing to this register. You may read back the value of the performance counter by reading this register.

2.5.1.2 IT_MON_PMC_[0 to 5]: Internal Transaction Performance Monitor Config. Register

Bus CBN, Device Number:	00h	Function:	2
Address Offset:	D0-D7h, D8-DFh, E0-E7h, E8-EFh, F0-F7h, F8-FFh	Size:	64 bits each
Default Value:	0h each	Attribute:	Read/Write
Sticky:	No	Locked:	No

The IT_MON_PMC_[0 to 5] Registers specify the configuration of the Internal Transaction Performance Monitors. This includes specifying Event Selection, Unit Mask, Enable & Disable Source and Reload Control.

Bits	Description
63:41	*reserved(0)*
40:33	**Length Encodings** 0000 0000b Any Length Transaction 0111 0000b 0 - 8 Bytes 0111 0001b 16 Bytes 0111 0010b 32 Bytes 0111 0111b 48 Bytes 0111 0011b 64 Bytes 0110 0000b < 32 Bytes 1000 0011b < 64 Bytes

Note: Length Encodings only used during the following Performance Monitor Setting:
Mem Read - Delayed
Mem Read - Deferred Reply
Memory Write

32:24	**DMASK Encodings** 0 0001 1010 - Configuration Space - Monitor transactions Destined for Config Block 0 0000 1100 - Memory - Monitor transactions Destined for Memory 0 0001 1110 - Broadcasts - Monitor Broadcasts 1 0000 0000 - All Destinations - Monitor all Destinations

23:15 **UMASK Encodings**
0 0000 0000 - Processor 0 - Monitor transactions originating from Processor 0
0 0000 0010 - Processor 1 - Monitor transactions originating from Processor 1
0 0000 0100 - Processor 2 - Monitor transactions originating from Processor 2
0 0000 0110 - Processor 3 - Monitor transactions originating from Processor 3
0 1000 0000 - All Processors - Monitor transactions originating from all Processors
0 0001 1010 - Configuration Space - Monitor transactions originating from Config Block
1 0000 0000 - All Initiators - Monitor all transactions

14:8 **Event Select**

Note: In the fields below bit 13 sometimes has an 'R'. This stands for retry. If 'R' is set to a 1 then the count is the number of occurrences of the event that were retried. If 'R' is set to a 0, then the count is for non-retried occurrences of the event. For example 110_0000b would be retried Interrupt Acks and 100_0000b is for non-retried Interrupt Acks.

Selects the event to be monitored.

000 0000b Monitoring Disabled.
011 1111b All Clocks.
1R0 0000b Interrupt Ack
1R1 0001b Special Transaction and Defer Reply to Write.
1R0 0010b I/O Read
1R0 0011b I/O Write - Deferred Reply
1R1 0011b I/O Write - Posted.
1R0 0101b Purge TC and reserved and Branch Trace Messages
1R1 0101b Outbound Interrupt or Hard Fail Write Completion
1R0 0110b Mem Read
1R1 0111b Memory Write
110 1000b Check Connection
1R0 1010b Cfg Read
1R0 1011b Cfg Write - Deferred Reply
1R1 1011b Cfg Write - Posted
000 1100b Inbound Interrupt
1R0 1110b Locked Read - In Order
1R1 1110b Locked Read - Delayed
1R1 1111b Mem Write-Line
1R1 0000b Normal Read Completion
1R1 0100b Memory Scrub
1R0 1100b SSBR
1R1 1101b RSBR
000 0001b Snoop events
000 0010b Snoop Stall events caused by 460GX chipset
000 0011b Snoop Stall events caused by processors
000 0100b Snoop Stall events by either CPU's or GX
000 0101b Hit events from CPU
000 0110b BNR events from GX
000 0111b BNR events from CPU
000 1000b BNR events from GX or CPU
000 1001b BPRI Clocks from GX
000 1010b BPRI Events from GX
000 1011b BPRI ADS's from GX
001 0001b Speculative reads that are not used because of HITW
001 0010b Speculative reads that are not used because of HITM
001 0011b Speculative reads that are not used because of retry
001 0100b Speculative reads that are not used because read was restarted
001 0101b Speculative reads that were successful
001 0110b Speculative reads that were not done speculatively

001 0111b Memory Read that was to be retried and received a HITM
001 1000b Memory Read with active OWN# and received a HITM
001 1001b Memory Read from a CPU that received a HITW
001 1010b Memory Read from a CPU that received a HITM
001 1011b Memory Read from PCI that received a HITW
001 1100b Memory Read from PCI that received a HITM
001 1101b Memory Write from PCI that received a HITM
010 0001b EV0 Events
010 0010b EV0 Clocks
010 0011b EV1 Events
010 0100b EV1 Clocks

7 *reserved(0).*

6:5 **Disable Source**
Selects event that will disable the performance monitor.

00b Never Disable.
01b Disable when this counter is overflowed (Note that if this setting is used, and bits [4:3] are set to 'Enable Always', then when this counter is overflowed, counting will not resume until the counter is loaded with a non-overflow value).
10b Disable on falling edge of SAC Event 0.
11b Disable on falling edge of SAC Event 1.

4:3 **Enable Source**
Selects event that will enable the performance monitor.

00b Never Enable.
01b Enable Always (Disable events overrides this setting and will disable counting).
10b Enable on rising edge of SAC Event 0.
11b Enable on rising edge of SAC Event 1.

2:0 **Reload Control**
Selects event that will control the Reloading of the performance monitor with the value written into the associated PMD2 register.

000b Never Reload.
001b Reload when counter overflows.
010b Reload on SAC Event 0 Asserted.
011b Reload on SAC Event 1 Asserted.
100b Reload on SAC Event 0 Asserting edge.
101b Reload on SAC Event 1 Asserting edge.

Note: When counting retried reads (code of 110_0110b) the logic will count reads that were to be retried, but got a HITM# and therefore the HITM# overrides the retry. For an exact count of reads that are truly retried, count retried reads (110_0110b) and subtract out the number of reads that were to be retried but got a HITM# (code of 001_0111b). This give the exact number of reads that were retried on the bus.

2.5.2 SDC

2.5.2.1 FSB_D_PMC_[1,0]: System Bus Performance Monitor Configuration Register

Bus CBN, Device Number:	04h		
Address Offset:	98-9Ah, 9C-9Eh	Size:	24 bits each
Default Value:	000000h each	Attribute:	Read/Write

The FSB_D_PMC_[1,0] Registers specify the configuration of the SDC system bus performance monitors. This includes specifying Event Selection, Unit Mask, Enable & Divisible Source & Reload Control.

Bits **Description**

23:17 *reserved(0)*.

16:15 **Mask**.
This field contains the Event Specific Mask Bits. This allows qualifying event collection by the issuing agent of the transaction.

00b reserved.
01b Monitor only if 460GX chipset initiated the transaction.
10b Monitor only if 460GX chipset did not initiate the transaction.
11b Monitor all transactions regardless of issuing agent.

14:8 **Event Select**.
Selects the event to be monitored.

000 0000b Monitoring Disabled.
000 0001b System Bus Clocks.
000 0010b DBSY# Clocks.
100 0010b DBSY# Events.
000 0011b DRDY# Clocks.
100 0011b DRDY# Events.
100 0100b DBSY# and not(DRDY#) Events.
000 0101b TRDY# Clocks.
100 0101b TRDY# Events.
000 0110b TRDY# asserted when DBUSY# asserted (clocks)
100 0110b TRDY# asserted when DBUSY# asserted (events)
000 0111b Read from SDC waiting on processor write (clocks)
100 0111b Read from SDC waiting on processor write (events)
000 1000b Read from SDC waiting on an SDC read to complete (clocks)
100 1000b Read from SDC waiting on an SDC read to complete (events)
100 1001b *reserved*
100 1010b *reserved*
000 1011bEvent Logic 0 Active Clocks
100 1011b Event Logic 0 Events
000 1100b Event Logic 1 Active Clocks
100 1100b Event Logic 1 Events

7 *reserved(0)*.

6:5 **Disable Source**.
Selects event that will disable the performance monitor. Note that the disable and enable sources are edge-triggered for Event 0 or 1.

00b Never Disable.

01b Disable when counter overflows.
10b Disable on falling edge (Deassertion) of SDC Event 0.
11b Disable on falling edge (Deassertion) of SDC Event 1.

4:3 **Enable Source**.
Selects event that will enable the performance monitor.

00b Never Enable (in this mode the counter will never count).
01b Enable Always (note that if this setting is used, the disable events in bits [6:5] will only disable counting for one clock, and then the counter resumes. When bits [4:3] are set to 'Enable Always', the only meaningful setting for bits [6:5] is 'Never Disable'.
10b Enable on rising edge (Assertion) of SDC Event 0.
11b Enable on rising edge (Assertion) of SDC Event 1.

2:0 **Reload Control**.
Selects event that will control the Reloading of the performance monitor with the value written into the associated PMD1 register.

000b Never Reload.
001b Reload when counter overflows.
010b Reload on SDC Event 0 Asserted.
011b Reload on SDC Event 1 Asserted.
100b Reload on SDC Event 0 Asserting edge.
101b Reload on SDC Event 1 Asserting edge.

2.5.2.2 FSB_D_PMD_[1,0]: System Bus Performance Monitor Data Registers

Bus CBN, Device Number:	04h		
Address Offset:	A0-A7h, A8-AFh	Size:	64 bits each
Default Value:	0 each	Attribute:	Read/Write

Two performance monitoring counters, with associated event selection and control registers, is provided in the SDC component. These counters may be configured to track system bus events. Event detection may be configured to increment a counter, affect performance monitoring pins, and issue an interrupt request on counter overflow.

The value written to this address, loads the counter and is also saved in a reload register. Each counter can be configured to reload the data when it overflows.

The FSB_D_PMD_[1,0] registers hold the performance monitoring count values. 39-bits of the counter are used for event counting, the 40th-bit is used as a overflow detection bit. The 39-bit count value allows up to 70 minutes of event collection at 133 MHz. Event selection is controlled by the PMC registers.

Each counter may be stopped/started independently, using the controls available in the associated PMD register.

Bits **Description**

63:40 *reserved(0)*

39 **Overflow**
This bit is asserted when the Event Count bit 38 carries into bit 39.

38:0 **Count Value**
This register contains the Performance Monitor Data Register. You may preset the value of the performance counter by writing to this register. You may read back the value of the performance counter by reading this register.

2.5.3 PXB

2.5.3.1 PMD[1:0]: Performance Monitoring Data Register

Address Offset:	D8-DBh, E0-E3h	Size:	32 bits each
Default Value:	0000_0000h each	Attribute:	Read/Write

Two performance monitoring counters, with associated event selection and control registers, are provided for each PCI bus in the PXB. Each counter may be configured to track PCI bus events. Event detection may be configured to increment a counter, toggle a pin on event or counter overflow, and issue an interrupt request on counter overflow.

The PMD registers hold the performance monitoring count values. These registers are 32-bit counters. Event selection is controlled by the PME registers, and the action performed on event detection is controlled by the PMR registers.

Each counter may be stopped/started independently, using the controls available in the associated PMR register.

Bits **Description**

31:0 **Count Value**

2.5.3.2 PMR[1:0]: Performance Monitoring Response

Address Offset:	DDh, E5h	Size:	8 bits each
Default Value:	0000h each	Attribute:	Read/Write

There are two PMR registers for each PCI bus, one for each PMD counter. Each PMR register specifies how the event selected by the corresponding PME register affects the associated PMD register, P(A,B)MON# pins, and the INT(A,B)RQ# pins.

Bits **Description**

7:6 **Interrupt Assertion**
Defines how selected event affects INTRQ# assertion. Whenever INTRQ# is asserted, a flag for this counter is set in the Error Status Register, so that software can determine the cause of the interrupt. This flag is reset by writing the Error Status Register.

0 Selected event does not assert INTRQ #
1 *reserved*
2 Assert INTRQ# pin when event occurs
3 Assert INTRQ# pin when counter overflows

5:4 **Performance Monitoring pin assertion**
Defines how the selected event affects the PMON# pin for this counter.

0 PMON# pin is tristated. Selected event has no effect.
1 *reserved*
2 Assert this counter's PMON# pin when event occurs
3 Assert this counter's PMON# pin when counter overflows

3:2 **Count Mode**
Selects when the counter is updated for the detected event.

0 Stop counting.
1 Count each cycle selected event is active.
2 Count on each rising edge of the selected event.
3 Trigger. Start counting on the first rising edge of the selected event, and continue counting each clock cycle.

Once configured to count, all counters in the SAC and each PXB can be (nearly) simultaneously started and stopped using a separate enable.

1:0 **Reload Mode**
Reload has priority over increment. That is, if a Reload event and a count event happen simultaneously, the count event has no effect.

0	Never Reload
1	Reload when this counter overflows.
2	Reload when the other counter overflows.
3	Reload unless the other counter increments.

2.5.3.3 PME[1:0]: Performance Monitoring Event Selection

Address Offset:	E8 - EBh	Size:	16 bits each
Default Value:	0000h each	Attribute:	Read/Write

The PME registers specify the particular event to track in the performance monitoring counters. The PXB supports tracking of PCI bus transactions (both specific and generic), and PCI bus signal assertion. Bus transactions may be qualified by the originating agent and transaction destination.

Accumulated event counts are held in the PMD registers, while the PMR registers specify the action to performed on event detection.

Bits **Description**

15 *reserved (0)*

14 **Count Data Cycles**
1: Count data cycles associated with selected event
0: Count the selected event

13:10 **Initiating Agent Selection**
This field qualifies the tracking of bus transactions by limiting event detection to those transactions issued by specific agents. That is, unless otherwise noted for the specific event selected (below), the agent initiating the bus transaction must match the selection specified here for the transaction to be tracked.

0000	Agent 0	1000	*reserved*
0001	Agent 1	1001	*reserved*
0010	Agent 2	1010	*reserved*
0011	Agent 3	1011	*reserved*
0100	Agent 4	1100	*reserved*
0101	Agent 5	1101	South bridge
0110	*reserved*	1110	460GX chipset agent (i.e. outbound)
0111	*reserved*	1111	Any agent

Note: This field is applicable only if the PCI bus is operated in internal-arbiter mode. If the bus is operated using an external arbiter, this field must be set to *Any Agent* to trigger any events.

9:8 **Transaction Destination Selection**
This field qualifies the tracking of bus transactions by limiting event detection to those transactions directed to a specific resource. That is, unless otherwise noted for the specific event selected (below), the source or destination of the data must match the selection specified here for the transaction to be tracked.

00	any	10	PCI Target
01	Main Memory	11	Parallel Segment Peer

7:6 *reserved*

5:0 **Event Selection**
This field specifies the basic PCI bus transaction or PCI bus signal to be monitored.

Individual Bus Transactions

00 0000	*reserved*	00 1000	*reserved*
00 0001	*reserved*	00 1001	*reserved*
00 0010	I/O Read	00 1010	*reserved*
00 0011	I/O Write	00 1011	*reserved*
00 0100	*reserved*	00 1100	Memory Read Multiple
00 0101	*reserved*	00 1101	Dual Address Cycle
00 0110	Memory Read	00 1110	Memory Read Line
00 0111	Memory Write	00 1111	Memory Write & Invalidate

Generic (Grouped) Bus Transactions

010 000	Any bus transaction	010 100	Any I/O transaction
010 001	Any memory transaction	010 101	Any I/O or memory transactions
010 010	Any memory read	010 110	Any I/O or memory read
010 011	Any memory write	010 111	Any I/O or memory write

Bus Signal Assertions

011 000	*reserved*	011 100	*reserved*
011 001	*reserved*	011 101	*reserved*
011 010	RETRY[1]	011 110	LOCK
011 011	*reserved*	011 111	ACK64

All other encodings are reserved.

NOTE:1. Counting data cycles is undefined for this selection.

2.5.4 GXB

2.5.4.1 AGP_PMD_0,1: AGP Performance Monitor Data Registers

Function Number:	BFN+1		
Address Offset:	50h, 58h	Size:	64 bits
Default Value:	0	Attribute:	Read/Write
Sticky:	No	Locked:	No

This counter may be configured to track AGP bus events as well as events internal to the GXB. Event detection may be configured to increment a counter, affect performance monitoring pins, and issue an interrupt request on counter overflow.

The value written to this address, loads the counter and is also saved in a reload register. Each counter can be configured to reload the data.

The AGP_G_PMD register holds the performance monitoring count value. 39-bits of the counter are used for event counting, the 40th-bit is used as a overflow detection bit. The 39-bit count value allows up to 70 minutes of event collection at 133 MHz. Event selection is controlled by the PMC registers.

Each counter may be stopped/started independently, using the controls available in the associated PMD register.

Bits	Description
63:40	reserved(0)
39	**Overflow** This bit is asserted when the Event Count bit 38 carries into bit 39.

38:0 **Count Value**
This register contains the Performance Monitor Data Register. You may preset the value of the performance counter by writing to this register. You may read back the value of the performance counter by reading this register.

2.5.4.2 PCI_PMD: PCI Performance Monitor Data Registers

Function Number:	BFN+1		
Address Offset:	60h	Size:	64 bits
Default Value:	0	Attribute:	Read/Write
Sticky:	No	Locked:	No

This counter may be configured to track PCI bus events as well as events internal to the GXB. Event detection may be configured to increment a counter, affect performance monitoring pins, and issue an interrupt request on counter overflow.

The value written to this address, loads the counter and is also saved in a reload register. Each counter can be configured to reload the data.

The PCI_PMD register holds the performance monitoring count value. 39-bits of the counter are used for event counting, the 40th-bit is used as a overflow detection bit. The 39-bit count value allows up to 70 minutes of event collection at 133 MHz. Event selection is controlled by the PMC registers.

Each counter may be stopped/started independently, using the controls available in the associated PMD register.

Bits	Description
63:40	*reserved(0)*
39	**Overflow** This bit is asserted when the Event Count bit 38 carries into bit 39.
38:0	**Count Value** This register contains the Performance Monitor Data Register. You may preset the value of the performance counter by writing to this register. You may read back the value of the performance counter by reading this register.

2.5.4.3 PERCON: Performance Monitor Control Register

Function Number:	BFN+1		
Address Offset:	E0h	Size:	8 bits
Default Value:	00h	Attribute:	Read/Write
Sticky:	No	Locked:	No

The PERCON Register allows one software write to start and stop the performance monitors. The 2 bits are fed into the Event Logic generation. Bit 0 feeds Event 0 and bit 1 feeds Event 1. Since the counters can be enabled and disabled by Events 0 or 1, then one write can start or stop all the counters together.

Bits	Description
7:2	*reserved (0)*
1	**Event 1 Input** This bit is fed as an input into Event 1 logic. This bit is OR'ed with any other logic generating Event 1, guaranteeing that if this bit is set, then Event 1 will be asserted.

0 **Event 0 Input**
This bit is fed an input into Event 0 logic. This bit is OR'ed with any other logic generating Event 0, guaranteeing that if this bit is set, then Event 0 will be asserted.

2.5.4.4 AGP_PMC_[0,1]: AGP Performance Monitor Configuration Register

Function Number:	BFN+1		
Address Offset:	ECh, F0h	Size:	32 bits
Default Value:	000000h	Attribute:	Read/Write
Sticky:	No	Locked:	No

The AGP_G_PMC Register specifies the configuration of the GXB AGP Performance Monitor. This includes specifying Event Selection, Unit Mask, Enable & Disable Source and Reload Control.

Bits **Description**

31:24 'n' for threshold Monitoring

23:20 *reserved(0)*.

19:18 **Events to monitor, when Event Select Code points to events. Data Transfer is one QW (8 bytes). Used also for code 11 0000**
00 - All events
01 - Events with 'n' Data Transfer Cycles
02 - Events with < 'n' Data Transfer Cycles
03 - Events with > 'n' Data Transfer Cycles

17:16 **Pipe or Sideband Request Mask**

00b reserved.
01b Selects Low Priority.
10b Selects High Priority.
11b Selects Both.

15:14 *reserved(0)*.

13:8 **Event Select**
Selects the event to be monitored.

00 0000b Monitoring Disabled.
00 0001b AGP Read Request Events.
00 0010b AGP Write Request Events.
00 0011b All AGP Request Events (does not include Flush or Fence requests).
00 0101b Singe Read that splits 4k page boundary.
00 0110b Single Write that splits line boundary.
01 0000b Flushes (events)
01 0001b Fences (events)
01 0010b RBF# active events.
01 0011b Events of write wait states inserted by AGP card
01 0100b Events of LPTT timeout with another request active.
01 0101b Events of MTT timeout with another request active.
01 0111b GART Aperture Misses
10 0000b AGP Clocks.
10 0100b Count AGP clocks that RBF# is stalled
10 0101b Count AGP clocks that low-priority read buffer is physically empty.
10 0110b Count AGP clocks that high-priority read buffer is physically empty.
10 0111b Count AGP clocks that both the low and high priority buffers are empty.
11 0000b Count AGP clocks that there are <=> 'n' requests queued; using bits 31:24 and 19:18.

7 **EVENT1 Count Enable**
If set, then this bit over-rides bits 13:8. If set, then AGP_PMD_0 will count the number of occurrences of EVENT1 and AGP_PMD_1 will count the number of clocks that EVENT1 is active. When this bit is not set, then the 2 counters are controlled by bits 13:8.

6:5 **Disable Source**
Selects event that will disable the performance monitor.

00b Never Disable.
01b Disable when counter overflows.
10b Disable on falling edge of GXB Event 0.
11b Disable on falling edge of GXB Event 1.

4:3 **Enable Source**
Selects event that will enable the performance monitor.

00b Never Enable.
01b Enable Always (with this setting, the only meaningful setting for bits 6:5 is 00b).
10b Enable on rising edge of GXB Event 0.
11b Enable on rising edge of GXB Event 1.

2:0 **Reload Control**.
Selects event that will control the Reloading of the performance monitor with the value written into the associated PMD register.

000b Never Reload.
001b Reload when counter overflows.
010b Reload when GXB Event 0 Asserted.
011b Reload when GXB Event 1 Asserted.
100b Reload on GXB Event 0 Asserting edge.
101b Reload on GXB Event 1 Asserting edge.

2.5.4.5 PCI_PMC: PCI Performance Monitor Configuration Register

Function Number:	BFN+1		
Address Offset:	F4h	Size:	24 bits
Default Value:	000000h	Attribute:	Read/Write
Sticky:	No	Locked:	No

The PCI_PMC Register specifies the configuration of the GXB PCI Performance Monitor. This includes specifying Event Selection, Unit Mask, Enable & Disable Source and Reload Control. NOTE: the Event Select field determines whether clocks or events are counted. In the case of events, then all events are counted. In the case of clocks, all clocks in which the selection is active are counted.

Bits **Description**

23:18 *reserved(0).*

17:16 **Initiating Agent**
00b reserved.
01b Outbound.
10b Inbound.
11b Selects Both.

15:14 *reserved(0).*

13:8 **Event Select**
Selects the event to be monitored.
00 0000b Monitoring Disabled

00 0010b All PCI Clocks
00 0100b Idle Bus Cycles
00 0111b All Disconnect Events
00 1000b Lock Asserted Clocks
00 1001b Lock Asserted Events
00 1011b I/O Reads - Events
00 1101b I/O Writes - Events
00 1111b Memory Read Events
01 0001b Memory Write Events
01 0011b SRAM Read Events
01 0101b SRAM Write Events
01 0111b Write Combining Events
01 1000b WBF# Active - Clocks
01 1001b WBF# Active - Events
01 1011b Retry Read that's not Delayed - Events
01 1101b Retry Write because no Write Slot available - Events
01 1110b Count PCI clocks waiting (Devsel and !IRDY and TRDY)
10 0000b Count PCI clocks waiting (Devsel and IRDY and !TRDY)
10 0010b Count PCI clocks data is transferring

7 *reserved(0)*.

6:5 **Disable Source**
Selects event that will disable the performance monitor.

00b Never Disable.
01b Disable when counter overflows.
10b Disable on falling edge of GXB Event 0.
11b Disable on falling edge of GXB Event 1.

4:3 **Enable Source**
Selects event that will enable the performance monitor.

00b Never Enable.
01b Enable Always (with this setting, the only meaningful setting for bits 6:5 is 00b).
10b Enable on rising edge of GXB Event 0.
11b Enable on rising edge of GXB Event 1.

2:0 **Reload Control**
Selects event that will control the Reloading of the performance monitor with the value written into the associated PMD register.

000b Never Reload.
001b Reload when counter overflows.
010b Reload on GXB Event 0 Asserted.
011b Reload on GXB Event 1 Asserted.
100b Reload on GXB Event 0 Asserting edge.
101b Reload on GXB Event 1 Asserting edge.

2.5.5 WXB

2.5.5.1 PCI_WXB_PMC0: PCI Performance Monitor Configuration Register

Address Offset:	DCh – DFh	Size:	32bits
Default Value:	00000000h	Attribute:	Read/Write

This register controls the PCI performance monitors. There are two performance monitors for each PCI bus. This register defines the events to be monitored, and when monitoring should start and stop. The selected event can be qualified by data transferred, and issuing agent.

Bits **Description**

31:24 *reserved (0)*

23:21 **Data Transfer and Transaction Qualifier**
Qualifies a selected "Packet Type" by the presence or absence of data transferred.
000bAll events
001bRetry - for any reason
010bRetry - no buffers available (inbound read or write transactions only)
011bRetry - no data available (inbound read transactions only)
101bLocked
110bDual Address Cycles

20:19 *reserved (0)*

18:17 **Issuing Agent Qualifier**
Monitor only those selected events issued by the following agent:
00breserved
01bOutbound: Issued by WXB
10bInbound: Not issued by WXB
11bAll: Issued by any agent

16:11 **Event Select**
Selects the event(s) to be monitored.

Measurement

00 0000b Monitoring Disabled

Transaction Types

01 0001bI/O Reads
01 0010bMemory Reads
01 0100bMem Read Lines
01 1000bMem Read Multiples
01 1111bAny Read
10 0001bI/O Writes
10 0010bMemory Writes
10 0100bMem Wr & Invalidates
10 1111bAny Write
11 1111bAny Transaction

10:4 *reserved (0)*

3 **Enable Source**
When this bit is set to 1, the performance monitoring logic is enabled. Default=0.

2:0 *reserved (0)*

2.5.5.2 PCI_WXB_PMC1: PCI Performance Monitor Configuration Register

Address Offset:	E8h – EBh	Size:	32bits
Default Value:	00000000h	Attribute:	Read/Write

This register controls the PCI performance monitors. There are two performance monitors for each PCI bus. This register defines the events to be monitored and when monitoring starts and stops. The selected event can be qualified by data transferred, and issuing agent.

Bits	Description
31:0	*See PCI_WXB_PMC0 definitions.*

2.6 Interrupt Related Registers

2.6.1 SAC

2.6.1.1 XTPRS: External Task Priority Registers

Bus CBN, Device Number:	00h	Function:	0
Address Offset:	C0-C7h	Size:	64 bits
Default Value:	80h	Attribute:	Read Only
Sticky:	No	Locked:	No

The XTPRS are used to support redirectable interrupts. These registers are made observable to software primarily for test and debug purposes. These registers will be updated by XTPR Update Special Cycle on the system bus. The second cycle of the XTPR Update Special Cycle's address determines the value to load into the register. Ab[27:24]# is the 4 bit XTPR value. Ab[23:20]# determines which register to update. Since the high priority agent reserves the uppermost agent ID bit, only Ab[22:20]# are used. These 3 bits decode to one of the 8 registers. Ab[31] is the enable bit for the register.

All registers default to 1000_0000b, which is the disabled state.

Each register is defined as:
Bit 7 - enable (1=disable, 0=enable)
Bits 6:4 - reserved (0)
Bits 3:0 - value of XTPR

Bits	Description
63:56	**XTPR 7** These bits represent the external task priority for symmetric agent ID 07h.
55:48	**XTPR 6** These bits represent the external task priority for symmetric agent ID 06h.
47:40	**XTPR 5** These bits represent the external task priority for symmetric agent ID 05h.
39:32	**XTPR 4** These bits represent the external task priority for symmetric agent ID 04h.
31:24	**XTPR 3** These bits represent the external task priority for symmetric agent ID 03h.
23:16	**XTPR 2** These bits represent the external task priority for symmetric agent ID 02h.

15:8 **XTPR 1**
These bits represent the external task priority for symmetric agent ID 01h.

7:0 **XTPR 0**
These bits represent the external task priority for symmetric agent ID 00h.

2.6.2 PID PCI Memory-mapped Registers

The PID uses two 32-bit memory-mapped registers to provide the indirect addressing access to its (x)APIC interrupt redirection registers as well as to its ID, version, and arbitration ID registers. The memory-mapped registers are placed at default addresses of FEC0_0000h, FEC0_0010h, and FEC0_0040h for (x)APIC compatibility.

The I/O select register is used to provide the index of the internal register being accessed. The register being accessed is determined by bits 7 through 0 of this register. The I/O window register is used to provide/receive data associated with the access.

Table 2-2 summarizes the memory-mapped registers. Detailed descriptions of each register follow.

Note: The default base address FEC00 which is mapped to A[31:12] can be changed by reprogramming the (x)APIC base address register via PCI configuration space access.

Table 2-2. Memory-Mapped Register Summary

Address	Name	Access	Default Value
FEC00000h	I/O Register Select Register	R/W	00000000h
FEC00010h	I/O Window Register	R/W	00000000h
FEC00040h	(x)APIC EOI Register	R/W	00000000h

2.6.2.1 I/O Register Select Register (FEC00000h)

The I/O register select register selects which indirect access register appears in the I/O window register where it can be manipulated by software. The selector values for the indirect access registers are listed in Section 2.6.3. Software programs bits 7 through 0 of this register to select the desired internal register. The contents of the selected 32-bit register can be manipulated via the I/O window register. The I/O register select register is read/write by software and its default is listed in Section 2.6.2.2. This starting address, where the I/O register select register and I/O window register reside, can be relocated to a different address via the APIC base address register. The format of the I/O register select register is shown in Table 2-3.

Note: This register is defined as a 32-bit register, but only the lower eight bits are used. This register must be accessed using 32-bit memory reads or writes. Bits 31 through 8 should be set to 0s.

Table 2-3. I/O Select Register Format

Register Offset: FEC00000hDefault Value: [00000000h]Attribute: Read/Write

Bit(s)	Name	Description
31:8	Reserved	These 24 bits are reserved.
7:0	REGISTER ADDRESS	These eight bits provide the address offset of the internal 32-bit register. This number is used to select consecutive 32-bit internal registers via the I/O window register. Described in Section 2.6.2.2 is 64-bit register access.

2.6.2.2 I/O Window Register (FEC00010h)

This register is mapped onto the PID's internal register that is selected by the I/O register select register. Readability/writeability by software is determined by the characteristics of the internal register that is currently selected. The format of the I/O window register is shown in Table 2-4 below. This register must be accessed using 32-bit read or write operations.

Table 2-4. I/O Window Register Format

Register Offset: FEC00010hDefault Value: [00000000h]Attribute: Read/Write

Bit(s)	Name	Description
31:0	IOWIN[31:0]	This 32-bit register contains the 32-bit write or read data value.

2.6.2.3 (x)APIC EOI Register (FEC00040h)

The (x)APIC EOI register provides a means of informing the PID that interrupt service has been completed by the processor for a given interrupt vector (mapped to 1 of 64 interrupt inputs received by the PID). The PID will compare this vector value with the vector field of each entry in the RT. When a match is found, the RIRR bit for that entry will be cleared. In the case of level-triggered interrupts, clearing the remote IRR bit will initiate a resampling of the corresponding interrupt input signal. If the interrupt input is still asserted, the PID will issue a new level-triggered interrupt message. The PID will therefore issue a single new interrupt message upon receiving an EOI write, corresponding to the still asserted interrupt input pin. The PID only uses the (x)APIC EOI register in SAPIC mode.

Note: If multiple redirection entries assign the same vector for more than one interrupt pin, each of those pins will be resampled and new interrupt messages issued for those that are still asserted. This register must be accessed using 32-bit read or write operations.

Table 2-5. (x)APIC EOI Register Format

Register Offset: FEC00040hDefault Value: [00000000h]Attribute: Read/Write

Bit(s)	Name	Description
31:8	Reserved	Reserved
7:0	SAPICEOI[7:0]	Interrupt vector

2.6.3 PID Indirect Access Registers

The PID provides several indirect access registers. These registers are accessed via the I/O select and I/O window registers described above. The indirect access registers include the (x)APIC ID, version, and arbitration ID registers, and 64 RTEs for each of the 64 interrupt inputs to the PID. Registers at offsets 03h-0Fh are reserved and will return a 00h value when read.

Table 2-6 summarizes the indirect access registers. Detailed descriptions of each register follow.

2.6.3.1 I/O (x)APIC ID Register (00h)

Table 2-6. Memory-mapped Register Summary

Offset	Name	Access	Default Value
00h	I/O (x)APIC ID Register	R/W	00000000h
01h	I/O (x)APIC Version Register	R/O	003F00vvh [a]
02h	I/O (x)APIC Arbitration ID Register	R/O	00000000h
03h-0Fh	Reserved	R/O	00000000h
10h	RTE 0	R/W	00000000_00010000h
12h	RTE 1	R/W	00000000_00010000h
14h	RTE 2	R/W	00000000_00010000h
16h	RTE 3	R/W	00000000_00010000h
18h	RTE 4	R/W	00000000_00010000h
1Ah	RTE 5	R/W	00000000_00010000h
1Ch	RTE 6	R/W	00000000_00010000h
1Eh	RTE 7	R/W	00000000_00010000h
20h	RTE 8	R/W	00000000_00010000h
22h	RTE 9	R/W	00000000_00010000h
24h	RTE 10	R/W	00000000_00010000h
26h	RTE 11	R/W	00000000_00010000h
28h	RTE 12	R/W	00000000_00010000h
2Ah	RTE 13	R/W	00000000_00010000h
2Ch	RTE 14	R/W	00000000_00010000h
2Eh	RTE 15	R/W	00000000_00010000h
30h	RTE 16	R/W	00000000_00010000h
32h	RTE 17	R/W	00000000_00010000h
34h	RTE 18	R/W	00000000_00010000h
36h	RTE 19	R/W	00000000_00010000h
38h	RTE 20	R/W	00000000_00010000h
3Ah	RTE 21	R/W	00000000_00010000h
3Ch	RTE 22	R/W	00000000_00010000h
3Eh	RTE 23	R/W	00000000_00010000h
40h	RTE 24	R/W	00000000_00010000h
42h	RTE 25	R/W	00000000_00010000h
44h	RTE 26	R/W	00000000_00010000h
46h	RTE 27	R/W	00000000_00010000h
48h	RTE 28	R/W	00000000_00010000h
4Ah	RTE 29	R/W	00000000_00010000h
4Ch	RTE 30	R/W	00000000_00010000h
4Eh	RTE 31	R/W	00000000_00010000h
50h	RTE 32	R/W	00000000_00010000h

Table 2-6. Memory-mapped Register Summary (Cont'd)

Offset	Name	Access	Default Value
52h	RTE 33	R/W	00000000_00010000h
54h	RTE 34	R/W	00000000_00010000h
56h	RTE 35	R/W	00000000_00010000h
58h	RTE 36	R/W	00000000_00010000h
5Ah	RTE 37	R/W	00000000_00010000h
5Ch	RTE 38	R/W	00000000_00010000h
5Eh	RTE 39	R/W	00000000_00010000h
60h	RTE 40	R/W	00000000_00010000h
62h	RTE 41	R/W	00000000_00010000h
64h	RTE 42	R/W	00000000_00010000h
66h	RTE 43	R/W	00000000_00010000h
68h	RTE 44	R/W	00000000_00010000h
6Ah	RTE 45	R/W	00000000_00010000h
6Ch	RTE 46	R/W	00000000_00010000h
6Eh	RTE 47	R/W	00000000_00010000h
70h	RTE 48	R/W	00000000_00010000h
72h	RTE 49	R/W	00000000_00010000h
74h	RTE 50	R/W	00000000_00010000h
76h	RTE 51	R/W	00000000_00010000h
78h	RTE 52	R/W	00000000_00010000h
7Ah	RTE 53	R/W	00000000_00010000h
7Ch	RTE 54	R/W	00000000_00010000h
7Eh	RTE 55	R/W	00000000_00010000h
80h	RTE 56	R/W	00000000_00010000h
82h	RTE 57	R/W	00000000_00010000h
84h	RTE 58	R/W	00000000_00010000h
86h	RTE 59	R/W	00000000_00010000h
88h	RTE 60	R/W	00000000_00010000h
8Ah	RTE 61	R/W	00000000_00010000h
8Ch	RTE 62	R/W	00000000_00010000h
8Eh	RTE 63	R/W	00000000_00010000h

a. vv is 13h in APIC mode of operation, 21h in SAPIC mode of operation.

The I/O (x)APIC ID register is read/write by software. On reset, this register's contents are reset to zero. This register is provided for APIC compatibility only and it does not serve any other purpose. The PID's (x)APIC ID register has a default value of 00000000h. This register should be programmed with the correct (x)APIC ID value before using the PID in APIC mode. The (x)APIC ARBID register is also written during a write to this register.

Table 2-7. I/O APIC ID Register Format

Register Offset: 00h **Default Value: [00000000h]Attribute: Read/Write**

Bit(s)	Name	Description
31:28	Reserved	These four bits are reserved.
27:24	ID[3:0]	These four bits provide the APIC ID. This field is used by the I/O APIC unit of the PID. In SAPIC or compatibility mode of operation, these bits are ignored.
23:16	Reserved	These 24 bits are reserved.
15	DT	This bit defines the delivery type. A '0' in this field indicates APIC delivery mechanism. A '1' indicates SAPIC delivery mechanism. This bit reflects the PICMODE strap pin.
14	LTS	Level deassert message support. This bit is always '0' since the PID does not support "level deassert" messages.
13:0	Reserved	Reserved.

2.6.3.2 I/O (x)APIC Version Register (01h)

The PID contains an I/O (x)APIC version register that identifies the type of I/O (x)APIC it implements. Software can use this to provide compatibility between different I/O (x)APIC implementations and their versions. The version register also contains the maximum RTE.

Table 2-8. I/O (x)APIC Version Register Format

Register Offset: 01h **Default Value: [003F00vvh]Attribute: Read-Only**

Bit(s)	Name	Description
31:24	Reserved	These eight bits are reserved
23:16	MAX REDIR	This is the entry number (0 being the lowest entry) of the highest entry in the I/O RT. It is equal to the number of interrupt input pins minus one that the PID supports. This field is hardwired and is read-only. The PID sets this field to **3FH**, indicating that it supports 64 RTEs.
15:8	Reserved	These eight bits are reserved.
7:0	VERSION	This is a version number that identifies the implementation version of the APIC or SAPIC units in the PID. This field is hardwired and is read-only. When the PID powers-up in APIC mode this field will return a value of **13H.** When the PID powers-up in SAPIC mode this field will return a value of **21H.**

2.6.3.3 I/O (x)APIC Arbitration ID Register (02h)

This register contains the APIC bus arbitration priority for the PID. This register is loaded whenever the PID's (x)APIC ID register is loaded. A rotating priority scheme is used for APIC bus arbitration. The winner of the arbitration becomes the lowest-priority agent and assumes an arbitration ID of 0. All other agents except the agent whose ARBID is 15, increment their ARBID by 1. The agent whose ARBID is 15 will take the winner's ARBID and will increment it by 1. ARBIDs are changed (incremented or assumed) only for messages that are successfully transmitted. A message transmitted successfully means that no CS error or acceptance error was reported for that message. The PID APIC ARBID is always loaded with the PID APIC ID during a "INIT-level deassert" message.

Note: Only four bits are required for the APIC ARBID. Bits 27:24 are used for this ID.

Table 2-9. I/O (x)APIC Arbitration ID Register Format

Register Offset: 02h **Default Value: [00000000h]Attribute: Read-Only**

Bit(s)	Name	Description
31:28	Reserved	These four bits are reserved.
27:24	ARBID	APIC Arbitration ID.
23:0	Reserved	These 24 bits are reserved.

2.6.3.4 I/O (x)APIC RTE (10h-8Fh)

The interrupt RT has a dedicated entry for each interrupt input pin. Software can individually choose the interrupt vector number for input pins. For each individual pin, the operating system can also specify the signal polarity (low-active or high-active), whether the interrupt is signaled as edges or levels, as well as the destination and delivery mode of the interrupt. The information in the RT is used to generate the inter-(x)APIC message upon assertion/deassertion of the interrupt pin. The PID's version register contains the number of entries in the interrupt RT. Offsets 10h through 8Fh are used for the interrupt RT. Each entry in the table is 64-bits. The format of each entry is described in Table 2-10.

Note: The interrupt RT is shared between SAPIC and APIC modes. Some of the fields have different meanings in the two modes as indicated in Columns 2 and 3. The PID implements only those bits that have valid functional fields associated with them. Reserved bits cannot be written and will return 0s when read.

Since each RTE is 64 bits wide, it must be accessed using two 32-bit memory read or write operations to consecutive dword-aligned addresses. The lower half of the entry, bits 31 through 0, is located at the even offset (such as 10h) and the upper half of the entry, bits 63 through 32, is located at the odd offset (such as 11h). While programming the RTEs, it is recommended that the lower half be programmed first, followed by the upper half.

Table 2-10. I/O (x)APIC RTE Format

Register Offset: 10-8Fh **Default Value: Undefined except mask bit is 1Attribute: Read/Write**

Bit(s)	SAPIC Mode Name	APIC Mode Name	Description
63:56	DEST ID	DEST ID	This field contains an (x)APIC ID. Local (x)APIC units receiving an interrupt message compare the DEST ID and DEST EID fields to the corresponding fields in their LID registers to determine if the interrupt is to be serviced by them.
55:48	DEST EID	Reserved	This field contains the extended (x)APIC ID. Local (x)APIC units receiving an interrupt message will use this field along with the DEST ID field to determine if the interrupt is to be accepted by them. This field is not used during APIC mode.
47:32	Reserved	Reserved	These 16 bits are reserved
31:18	Reserved	Reserved	These 14 bits are reserved
17	FLUSHEN	FLUSHEN	This bit controls the flushing of the I/O buffer on a per-interrupt basis. A 0 indicates that the buffer must be flushed before the interrupt is sent to the local (x)APIC. This setting will cause the hardware flush control signals to be used. A 1 indicates that the buffer does not need to be flushed before the interrupt is sent out to the local (x)APIC. This setting will cause the hardware flush control signals to be ignored.

Table 2-10. I/O (x)APIC RTE Format (Cont'd)

Register Offset: 10-8Fh Default Value: Undefined except mask bit is 1Attribute: Read/Write

Bit(s)	SAPIC Mode Name	APIC Mode Name	Description
16	MASK	MASK	This bit masks the (x)APIC delivery of this interrupt. A 0 indicates that delivery of this interrupt is not masked. An edge or level on an interrupt pin that is not masked results in the delivery of the interrupt to the destination. A 1 indicates that delivery of this interrupt is masked. It is the software's responsibility to deal with the case where the mask bit is set after the interrupt message has been accepted by a local (x)APIC unit but before the interrupt is dispensed to the processor. When the mask bit is 1, interrupts coming into the PID are routed to the INTIO, I2O_INT#, or I2B pins, provided they are correctly mapped. No delivery status bit latching is performed in this mode. When the mask bit is a 0, interrupts are latched in the delivery status bit. If the mask bit is set to a 1 after the interrupt is latched but before it is delivered, the latched value will be held until the interrupt is unmasked and delivered.
15	TRIGGER MODE	TRIGGER MODE	The trigger mode field indicates the type of signal on the interrupt pin that triggers an interrupt. A 0 indicates edge sensitive. A 1 indicates level sensitive.
14	RIRR	RIRR	This bit is read-only. During (x)APIC mode, this bit is used for level-triggered interrupts. Its meaning is undefined for edge-triggered interrupts. For level-triggered interrupts, this bit is set when the local (x)APIC(s) accepts the level interrupt sent by the PID. The RIRR bit is reset when an EOI message is received from the local (x)APIC.
13	POLARITY	POLARITY	This bit specifies the polarity of each interrupt signal connected to the interrupt pins of the PID. A value of 0 means the signal is high-active and a value of 1 means the signal is low-active. In the case of level, high-active means if the pin is sampled high, it is considered active. In the case of an edge, high-active means the low-to-high edge is considered active.
12	DELIVERY STATUS	DELIVERY STATUS	This bit is read-only. It holds the current status of interrupt delivery to the processor. This bit functions differently depending on the Trigger Mode bit. When the Trigger Mode bit indicates edge sensitive: The Delivery Status bit is set when an active edge is detected on the interrupt pin. The Delivery Status bit is reset when the interrupt message is successfully sent to the processor. When the Trigger Mode bit indicates level sensitive: The Delivery Status bit reflects the assertion of the pin. i.e. If the pin is at its active level, according to its Polarity bit, the Delivery Status bit is set; If the pin is at its inactive level, according to its Polarity bit, the Delivery status bit is reset. To accurately determine the delivery status of an interrupt in level mode the RIRR bit must be considered as well as the Delivery Status bit.

Table 2-10. I/O (x)APIC RTE Format (Cont'd)

Register Offset: 10-8Fh Default Value: Undefined except mask bit is 1Attribute: Read/Write

Bit(s)	SAPIC Mode Name	APIC Mode Name	Description
11	DESTINATION MODE	DESTINATION MODE	This bit determines the interpretation of the destination field. A 0 indicates physical mode. In physical APIC mode, a destination APIC is identified by its ID. Bits 56 through 59 of the destination field specify the 4-bit APIC ID. In physical SAPIC mode, the destination ID is defined by bit 63 through 48 of the destination field. A 1 indicates logical mode. In logical APIC mode, destinations are identified by matching on logical destination under the control of the destination format register and logical destination register in each local APIC. Bits 56 through 63 (8 MSB) of the destination field specify the 8-bit APIC ID. In logical SAPIC mode, the destination field bits 63 through 56 make up 8 bits of the destination ID. The remaining 8 bits are bits 55 through 48 of the destination field and must always be programmed to 0.
10:8	DELIVERY MODE	DELIVERY MODE	The delivery mode is a 3-bit field that specifies how the local (x)APIC units listed in the destination field should act upon reception of this signal. Note that certain delivery modes will only operate as intended when used in conjunction with a specific trigger mode. These restrictions are indicated in the table below for each delivery mode. Delivery mode encodings are shown below. Modes 001 and 010&Vector Field apply to APIC mode only: **00x (Fixed SAPIC Mode):** This means deliver the interrupt on the INTR signal of the processor listed in the destination. Trigger mode can be edge or level. The least significant bit of this delivery mode is a HINT bit that is communicated to the platform to allow it to redirect the interrupt to another processor on the same processor system bus as the destination. The way this redirection occurs is independent of platform implementation. Otherwise, the processor can ignore the least significant bit. **000 (Fixed APIC Mode):** This means deliver the signal on the INTR signal of all processor cores listed in the destination field. Trigger mode for "fixed" delivery mode can be edge or level. **001 (Lowest-Priority APIC Mode):** This means deliver the interrupt on the INTR signal of the processor core that is executing at the lowest-priority among all the processors listed in the specified destination. Trigger mode for lowest-priority delivery mode can be edge or level. **010 (PMI or SMI):** Delivery mode is edge only. For systems that rely on SMI semantics, the vector is ignored but must be all zeros for future upgrades. For systems that rely on PMI semantics, the vector number has meaning and is not ignored. **011 (Reserved)**. **100 (NMI):** This means deliver the interrupt on the NMI signal of the processor listed in the destination. Vector information is ignored. The NMI is always delivered as an edge-triggered interrupt. The trigger mode field is ignored. **101 (INIT):** This means deliver the interrupt on the INIT signal of the processor listed in the destination. Vector information is ignored. **110 (Reserved)**. **111 (ExtINT):** This means deliver the interrupt to the processor listed in the destination as an interrupt that originated in an externally connected (8259A-compatible) interrupt controller. The INTA cycle that corresponds to this ExtINT delivery will be routed to the external controller that is expected to supply the vector. A delivery mode of ExtINT requires an edge-triggered mode. ExtINT should be targeted for only one processor.
7:0	VECTOR	VECTOR	This is the vector number identifying the interrupt being sent.

System Architecture 3

This chapter provides an explanation of the 460GX chipset's handling of various aspects of the system architecture. It covers coherency, ordering, interrupts and related issues.

3.1 Coherency

For any computer system, data coherency between processor caches and other elements within the system is one of the main concerns of the system designer. In a Symmetric Multiprocessor (SMP) system, hardware is usually required to maintain this coherency. In some instances, though, software may assume responsibility for coherency. The use of WC (write-combining) memory requires software to manage coherency. Coherency applies to the processor caches, the I/O sub-system and graphics traffic. The programming model for a particular architecture determines precisely how the hardware must behave. A loosely coupled system may require that software guarantee that different processors which need to use the same piece of data are able to do so. For most SMP systems, the hardware in the system guarantees that a piece of data is updated to all processors if it is shared. For coherent data or code, there can exist only one valid version. There may be shared copies of this data or code, but all elements in the system have the same value. If the data is not coherent, then different elements may have different values for the identical address. For example, non-coherent AGP traffic is not kept coherent with processor caches, since non-coherent AGP addresses are not snooped on the bus. Coherency is related to, but different than cacheability.

3.1.1 Processor Coherency

Intel processors do not have a specific bit to specify coherency for each transaction. Data and code are usually considered fully coherent with respect to other processors and to each other. There are exceptions to this - such as the WC (write combining) memory type. Data that is marked as WC in the page table will not be coherent between processors. The Itanium processor uses the standard MESI (Modified/Exclusive/Shared/Invalid) protocol for its caches. As well, the Itanium processor, when in EM mode, does not guarantee in hardware that the instruction caches are coherent with data transfers. Software must flush the i-cache and reload it when self-modifying code is running.

Processors maintain coherency with each other by placing the addresses for referenced memory on the common system bus. Each of the other processors snoops this address to see if it has the data in any of its own caches. If one has the data unmodified, then it signals this using the HIT# pin, and memory, since it has the latest version of that address, provides the data. Both the responding and the requesting processors will then go to the shared state in the cache. If the snooping processor has the data modified, it asserts the HITM pin and provides the data, since it owns the most up-to-date copy of that address. The memory system will also take the data provided and update the copy in SDRAM, unless the OWN# signal is active, in which case the memory is not updated.

For WB memory space the MESI protocol requires that only one processor has a copy of modified data. Therefore if processor A is writing the data, it was required to gain ownership of the line before updating it, and as it gained ownership the other processors in the system would have gone to the invalid state for that line.

For WC memory, one processor may write to an address that is marked WC in its page table and hold the write in its own data buffer, while waiting to write to the bus. If a different processor were to read this address, the first processor does not snoop the WC holding registers and therefore would not provide the newly written data. So the 2nd processor would get the data from memory which is stale or un-updated with respect to the first processor.

Processor caches use physical addresses for indexing into the cache. The memory attributes are marked in the page tables which are based on the virtual address. If the O/S points two virtual addresses to the same physical page and provides different attributes for each, then coherency may not be maintained between processors, since one may have used the virtual address that pointed to the page as WB and the other one used a different virtual address that pointed to the same physical page, but was marked WC. This is usually considered a programming error. It may be useful in some applications, but software takes full responsibility for the results.

3.1.2 PCI Coherency

As well as processor to processor coherency, the I/O subsystem maintains coherency between processor caches and transactions initiated on PCI or other I/O devices that are directed to memory. This is done in hardware and therefore software is not required to flush caches before I/O operations (as long as that page is not marked WC).

Reads originating from the I/O sub-system are presented to the system bus for snooping. If a processor has the modified (dirty) data, then it provides it on the data bus and the SAC presents this data back to I/O. If no cache has the data modified, then the data is provided by the SDRAMs.

Writes are presented to the system bus as well. This allows a new code or data page to be brought in and the old page to be invalidated as the write is being done for each line. Writes may get a HITM# snoop which causes the processor to write back the entire line and then having the I/O write to overwrite the particular bytes it wishes to update.

3.1.3 AGP Coherency

AGP transactions originate from a graphics (or other) device residing in the one AGP slot provided when using the GXB. There are 2 different types of transactions originating from a device in this slot. The graphics card may do coherent or non-coherent transactions.

Non-coherent transactions are not required to be placed on the system bus (although they could be, with some loss of bus bandwidth). The 460GX chipset implementation does not pass non-coherent AGP traffic to the system bus. These addresses are sent directly to the memory queue. The processor could still have these addresses cached. If the application wishes to do this, then it must handle coherency itself. Software must flush the processor’s cache.

Coherent traffic from the graphics card is treated like coherent PCI traffic. Addresses are sent to the system bus before being serviced by the memory system.

3.2 Ordering

Intel processors prior to the Itanium processor have used processor consistency for their ordering model. The Itanium processor allows both a strongly ordered and a weakly ordered programming model to be implemented.

New EM code may be weakly ordered. To allow the processor to take advantage of this, the 460GX chipset defers all reads and returns the data out-of-order to the processor. By returning data in an out-of-order fashion, the DRAM's may be accessed in an optimal manner. Accesses are sent out of the memory queue to free banks of SDRAM's. Thus, if consecutive addresses are to the same bank, instead of holding up all later accesses while doing the first 2 in order, later accesses may move around the second access and allow data to move continuously from SDRAM to the system bus.

To maintain ordering in the system, the processor issues an address and must wait until that address has been accepted by the system, or in other words become globally visible. If the operation is kept in the in-order queue, then visibility occurs at the snoop phase with no defer. This means that operation is not retried and visibility has been met. A read becomes visible when no later store can change the value seen by the reading processor. A write becomes visible when all later reads will see the result of that write. With this definition, the 460GX chipset is able to guarantee visibility is met when the access is deferred, since it prevents any later access from affecting that read. Any coherent write to that line would be retried until the read is complete. Writes to memory are posted and so are immediately visible and complete from the system perspective.

There is no ordering relationship between the PCI command streams of an AGP card and its AGP command streams. The AGP spec mandates certain ordering rules within each stream that are visible by the graphics card, but the order in which the system does the transactions is not specified. Therefore, the typical producer-consumer model can not be guaranteed by doing simple reads and writes across command streams. The AGP card must first issue a "flush" command in order to guarantee the AGP low-priority stream is observable in memory before sending a flag (which indicates all the writes are visible to the processor) up the PCI stream.

3.3 Processor to PCI Traffic and PCI to PCI (Peer-to-Peer) Traffic

Due to the limited number of resources for transactions directed to PCI, a transaction may be retried if all the resources are utilized. It is possible for one processor or one PCI agent to keep taking all the resources and preventing a different processor or PCI agent from making any forward progress.

3.4 WXB Arbitration

The following topics highlight a few of the methods employed within the WXB for starvation prevention.

3.4.0.1 WXB Arbitration at the PCI Bus

Arbitration for Inbound Transactions

The WXB implements a simple two-level PCI arbitration scheme in the same vein as that of the PXB-C0. Access to inbound resources is subject to the PCI arbitration scheme and to inbound resource management algorithms. Inbound Write Request (IWR) acceptance is subject to an IWR starvation prevention mechanism while Inbound Read Request acceptance is subject to the availability of either an invalid stream slot or to space in the Delayed Transaction Reservation Buffer.

Arbitration for Outbound Transactions

The WXB relies heavily on the PCIset core and the *PCI Specification* regarding transaction ordering for dealing with starvation on outbound transactions. Once the WXB has won PCI arbitration for an outbound transaction, the WXB will initiate the request at the top of the Outbound Transaction Queue (OTQ) ***unless*** there is a read within the Outbound Read Request FIFO (ORRF). In this case, if the read at the top of the ORRF has already received some data, and the transaction at the top of the OTQ is a read, it will be moved behind the read in the ORRF. The read at the head of the ORRF will then be initiated on the PCI bus. The read at the head of the ORRF will be moved to the back if it hasn't already received some data. If, however, the transaction at the top of the OTQ is a write, then the arbitration policy will be to choose, on a round-robin basis, between the read at the head of the ORRF and the write. The write will attempt to burst to the end of a cache line; if there are no reads in the ORRF at the end of the cache line burst, the write will continue to burst to the end of the next cache line unless the MLT causes the WXB to disconnect.

The IHPC will also participate in arbitration with the purpose of idling the PCI bus. When the IHPC has won grant, and FRAME# has deasserted, there will be moments of inactivity as the IHPC alters the state of various external signal and power control registers for one or more PCI slots.

3.5 Big-endian Support

The Itanium processor supports both little-endian and big-endian accesses. The chipset does not need to know which mode the processor is in. The chipset provides data in the same manner in both cases. There is no indication on the system bus which mode the processor is using.

3.6 Indivisible Operations

3.6.1 Processor Locks

The 460GX chipset supports locks on the system bus, done by the processor. These locked transactions are either a series of atomic Read-Write or Read-Read-Write-Write transactions. They may be targeted to I/O devices or to SDRAM. See 'Transactions' Chapter for the flow for locks. During the sequence, the system bus is locked and no other traffic can occur on that bus. Traffic may be flowing throughout the rest of the system, such as AGP to memory or transactions that stay within the non-locked PCI buses.

The 460GX chipset does not support locks that cross device boundaries. In other words, if the first read in a locked sequence targets device X, then the remaining transactions in the lock (either R-W-W or W) must also target device X. The only exception to this rule is when device firmware has been "in-line shadowed" using the MAR registers. In this case the Read(s) in a locked sequence can be mapped to the compatibility PCI bus, and the Writes(s) could be mapped to memory. When a MAR has been mapped to write protect memory, a locked sequence to that MAR region is completely redirected to PCI, in order to avoid the resource allocation problems associated with crossing memory/PCI device boundaries. The 460GX chipset does not provide any special checks to detect locks that cross device boundaries outside of the MARs. If software attempts to establish such a lock, indeterminate results will occur: either the lock will appear to work, even thought the access was not performed atomically, or a deadlock will result.

AGP LOCKS

There is no LOCK signal on the AGP bus. However, legacy code that issues read-modify-write (RMW) transactions could still be converted for use with an AGP device. The GXB will attempt to establish a “pseudo-lock” to cover such an event. However, there is still a deadlock case within the AGP controller that the GXB can not address. This case is covered in a “special design considerations” section of the AGP specification.

The deadlock occurs when the device internally posts an inbound write after the first read completes in a “R-R-W-W” locked sequence. The specification requires that the AGP master resolve the problem in either software or hardware:

- Software must prevent the device driver from accessing internal registers with misaligned reads while there are posted writes in the PCI interface. This works if the device never posts writes, implements a unified interface (i.e.: no “internally posted” writes), or disallows misaligned read access (no multi-word registers).
- Hardware allows the read to proceed even in the presence of the posted write. Technically this is a violation of protocol, but the master is at liberty to insure that internal status doesn’t get updated on behalf of the “posted” write until that data actually leaves the part.

3.6.2 Inbound PCI Locks

The 460GX chipset does not support inbound locks.

3.6.3 Atomic Writes

Some system bus operations such as Write 8 bytes, Write 16 bytes and Write 32 bytes, are indivisible operations on the system bus. However, since the PCI protocol allows target device to disconnect at any point in a transfer sequence, these operations are not indivisible on the PCI bus. Furthermore, these accesses cannot be locked because PCI specification allows use of locked cycles only if the first transaction of the locked operation is a read. Therefore software must not rely upon atomicity of system bus write transactions which are greater than 32 bits or Dword misaligned once they are translated to the PCI bus.

3.6.4 Atomic Reads

The system bus memory read operations to PCI can request more than 32-bits of data (i.e. 8 byte, 16 byte and 32 byte). The problem of indivisibility of operations is very critical for this type of transaction. The PXB does NOT Lock multi-cycle reads to guarantee atomicity. Note that ATOMICITY of Host-PCI reads which are greater than 32 bits or are Dword misaligned is NOT GUARANTEED.

The PXB can accept inbound reads and writes while an outbound read or write is in a partially completed state.

3.6.5 Locks with AGP Non-coherent Traffic

AGP has a non-coherent stream that does not go over the system bus. Therefore processor locks, whether established to memory or I/O, do not prevent the non-coherent AGP accesses from occurring. If a processor reads a location in memory with a locked read, and then writes the same

location, there is no guarantee that AGP has not written the location while the lock was active on the system bus. AGP may read or write those locations or any other memory location, independent of the processor lock.

3.7 Interrupt Delivery

Interrupts may be delivered to the processors over the system bus. The interrupts may come from an I/O device or from another processor. The new system bus delivery method for interrupts to Itanium processor is referred to as SAPIC.

For SAPIC, the interrupts appear on the bus with a specific encoding on the REQa/REQb signals. The address used is 0x000FEEzzzzy. The 16 bits ZZZZ determine the target to which the interrupt is being sent. Both I/O interrupts and inter-processor interrupts are delivered in this manner. The data field contains the interrupt vector.

Devices may send their interrupts to one specific processor, or have the system choose the processor to which to deliver the interrupt. The algorithm used by the system to deliver interrupts depends on the software. The chipset will have a register for each processor. This register, called the XTPR (eXternal Task Priority Register), is programmable and may be set as software wishes. One could use this feature for lowest-priority delivery, as one example. The processor updates its XTPR when it updates its own internal priority register. An interrupt which is received by the chipset with the redirectable hint bit set, will be sent to the processor with the lowest value in the XTPR. If 2 or more processors tie for the lowest value, the processor with the lowest processor ID will be selected.

The 4 XTPR registers in the 460GX chipset are updated when the processor does a special cycle on the bus. When the special cycle is decoded, the low order 3 bits of the DID are used to determine which register to update. Each XTPR register is disabled at reset, and requires a special cycle XPTR-update to be enabled.

3.8 WXB PCI Hot-Plug Support

“Hot-Plug” is the term given to describe the capability to insert and remove PCI add-in cards into a computer while the PCI bus itself and other subsystems in the computer are fully operational. Hot-Plug logic in the WXB supports system hot-plug capability by providing the state machines and programming interface to individually power up and power down PCI slots in a controlled fashion. An Integrated Hot-Plug Controller (IHPC) comprises the hot-plug logic for one PCI bus.

Hot-Plug functionality in a complete system which includes the WXB requires three subsystems to work together: (1) the software subsystem, including firmware and drivers, (2) the hardware logic subsystem in the WXB, including hot-plug registers and memory and state machines, and (3) the external hardware subsystem, including interlock switches, shift registers, FETs, and LEDs on each PCI bus.

The PCI hot-plug logic performs three sets of operations: reset, slot enable (power-up and connect to bus), and slot disable (disconnect from bus and power-down). In normal operation, slot enable and slot disable occur only under software direction. Slot disable may occur automatically if a PCI card generates a power fault or if the user opens an interlock switch to remove a powered-up PCI card. The slot enable and disable sequences are briefly described below.

3.8.1 Slot Power-up and Enable

To power-up a PCI slot, software sets a command bit in a register. Then the hot-plug logic performs the following steps:

1. Set PWREN active to the slot and clock the parallel latch.
2. Set CLKEN# active to the slot but do not clock the parallel latch.
3. Wait 200 msec for slot power to stabilize, except in test mode.
4. Gain ownership of the PCI bus through arbitration.
5. Clock the parallel latch.
6. Release ownership of the bus after 480 nsec.
7. Set RESET# inactive to the slot but do not clock the parallel latch.
8. Wait 200 msec for slot clock to stabilize, except in test mode.
9. Gain ownership of the PCI bus through arbitration.
10. Clock the parallel latch.
11. Release ownership of the bus after 480 nsec.
12. Set BUSEN# active to the slot but do not clock the parallel latch.
13. Wait 1000 msec for PCI card initialization, except in test mode.
14. Gain ownership of the PCI bus through arbitration.
15. Clock the parallel latch.
16. Release ownership of the bus after 480 nsec.

3.8.2 Slot Power-down and Disable

To power-down a PCI slot, software sets a command bit in a register. Then the hot-plug logic performs the following steps:

1. Set BUSEN# inactive to the slot but do not clock the parallel latch.
2. Gain ownership of the PCI bus through arbitration.
3. Clock the parallel latch.
4. Release ownership of the bus after 480 nsec.
5. Set RESET# active to the slot but do not clock the parallel latch.
6. Gain ownership of the PCI bus through arbitration.
7. Clock the parallel latch.
8. Release ownership of the bus after 480 nsec.
9. Set CLKEN# inactive to the slot but do not clock the parallel latch.
10. Gain ownership of the PCI bus through arbitration.
11. Clock the parallel latch.
12. Release ownership of the bus after 480 nsec.
13. Set PWREN inactive to the slot and clock the parallel latch.

System Address Map 4

4.1 Memory Map

The Itanium™ processor supports a 44 bit address space. The 460GX chipset supports only 36 bits of the address bus for a 64 GB of physical memory and must address up to several GB of memory mapped I/O space. The 460GX chipset attaches to A#[35:3].

The memory address space is divided into four regions: the 1 MB Compatibility Area, the 15 MB Low Extended Memory region, the (4 GB minus 16 MB) Medium Extended Memory region, and the (16 TB minus 4 GB) High Extended Memory region. The first three regions are divided into multiple subregions, with dedicated purpose and semantics. The memory address map is illustrated in Figure 4-1.

4.1.1 Compatibility Region

This is the range from 0-1 MB (0_0000 to F_FFFF). Addresses here may be directed to a PCI bus (the compatibility bus) or main memory. Any DRAM located in this area that is not used as main memory is not recovered. This region is divided into four subregions, some of which are further subdivided. Regions below 1M that are mapped to memory are accessible by the processors and by any PCI bus.

Regions below 1M that are mapped to PCI are accessible by the processor, and by PCI devices on the targeted PCI bus (in the case of the regions mapped by the MAR this means the compatibility PCI bus; for the VGA region this means the PCI bus to which VGA is mapped). Parallel segment peer-to-peer accesses are not supported below 1M; a PXB will either forward the access to memory or let it be claimed/master abort on the PCI bus below it.

4.1.1.1 DOS Region

The DOS Region is the lowest 640 KB, in the address range 0h to 9_FFFFh. DOS applications execute here. The lower 512K of this region is always mapped to main memory, and is accessible by the processors and by any PCI bus. The upper 128K of this region can be mapped to either PCI or main memory, and is mapped using one of the MAR registers. The range defaults to memory.

4.1.1.2 VGA Memory

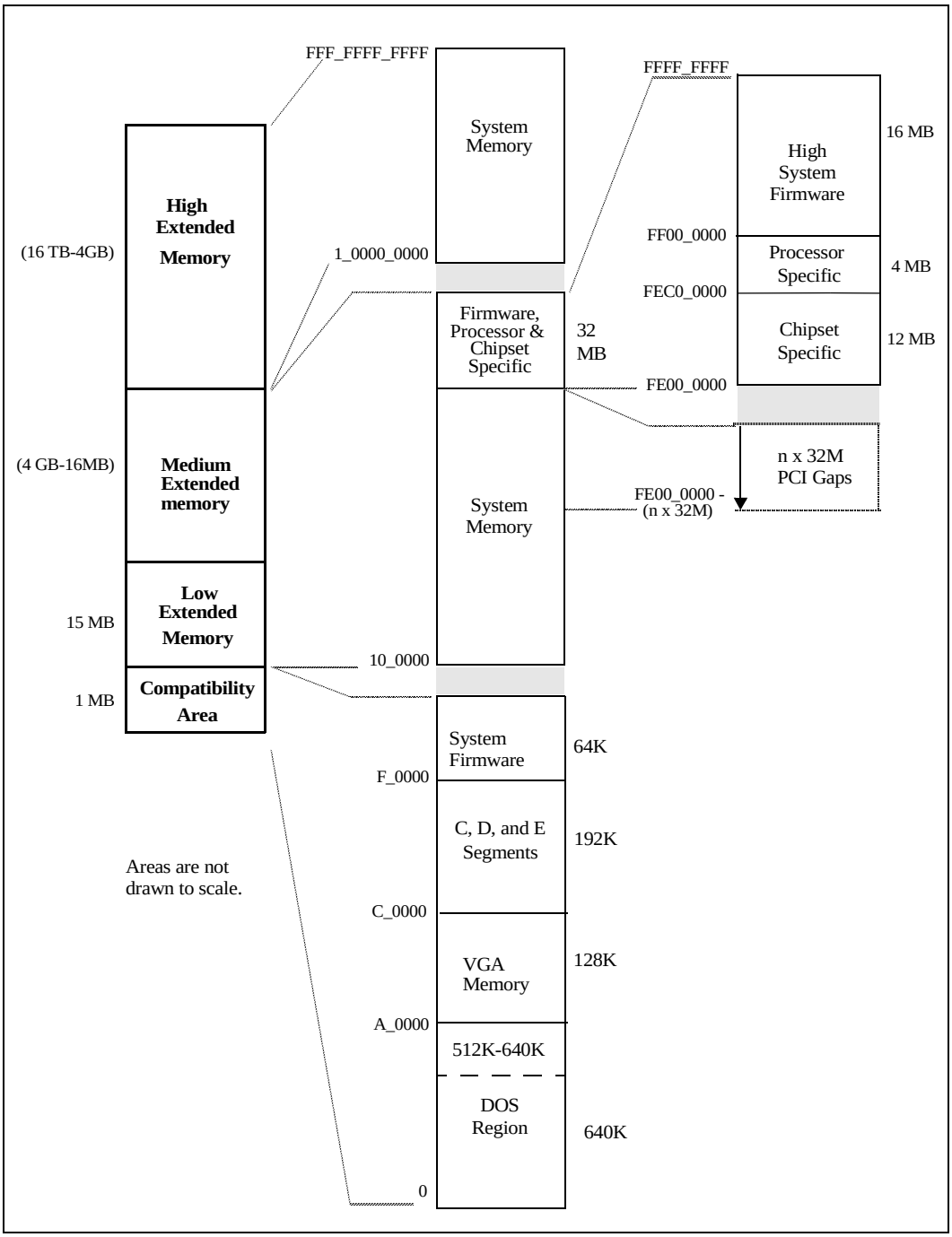
The 128 KB Video Graphics Adapter Memory subregion (A_0000h to B_FFFFh) is normally mapped to a video device on the compatibility PCI bus. Typically, this is a VGA controller. The 460GX chipset supports mapping this region to any of its logical PCI segments. At power-on this space is mapped to the compatibility PCI bus.

4.1.1.3 C, D, and E Segments

The 192 KB C, D, and E Segments are divided into smaller blocks that can be independently programmed as Mapped to Memory, Memory Write Protect, In-line Shadowed, or Mapped to PCI. These regions are used to provide memory to PCI devices requiring memory space below 1 MB.

The MAR registers determine how each range is used. The default for these segments at power-on is that they are mapped read/write to the PCI compatibility bus.

Figure 4-1. System Memory Address Space



4.1.1.4 System Firmware

The 64 KB region from F_0000h to F_FFFFh is treated as a single block. Read/Write attribute enables defined in the MAR registers may be used to direct accesses to the compatibility PCI bus or main memory. At power-on, this area is mapped by default to the PCI compatibility bus.

4.1.2 Low Extended Memory Region

The 15 MB Low Extended Memory region is always mapped to main memory. Since the 460GX chipset does not support ISA cards, there is no gap provided in this region.

4.1.3 Medium Extended Memory Region

The Medium Extended Memory region is divided into two primary regions: A fixed gap containing the firmware area along with gaps for Itanium processor and chipset specific functions, and a variable gap to support memory mapped I/O.

The fixed gap is between 4 GB and (4 GB minus 32 MB) and is always enabled. This region must not be defined as WB. DRAM supported by the 460GX chipset that is masked by this hole is remapped to an area over 4 GB. The fixed gap is further divided into three regions:

- The 4 GB to (4 GB minus 16 MB) region is reserved for system firmware. Addresses directed to this area are always directed to the compatibility PCI bus.
- The (4 GB minus 16 MB) to (4 GB minus 20 MB) region is reserved for processor specific functions. This region can be thought of as four 1 MB segments:
  - FEF0_0000 - FEFF_FFFF: This segment is used for Local APIC messages. Neither inbound nor outbound accesses to this region should be seen by the chipset. The chipset claims outbound accesses to this region, and will forward reads to PCI 0a to be master aborted and drops the writes. Inbound accesses to this region can only occur due to a programming error or address parity error; firmware programs the Expander bridges to prevent programming errors from reaching the SAC; address parity errors are protected against at each of the major bus interfaces. Therefore the SAC does not require a defined response for inbound requests that reach this area; the results are unpredictable.
  - FEE0_0000 - FEEF_FFFF: This segment is used to deliver interrupts. The chipset claims outbound accesses to this region, and will forward reads to PCI 0a to be master aborted and drops the writes. Inbound writes to this region are translated to an interrupt command encoding, forwarded to the system bus, and then claimed by the chipset. Reads to this area are illegal.
  - FED0_0000 - FEDF_FFFF: This segment is reserved. The chipset claims outbound accesses to this region, and will forward reads to PCI 0a to be master aborted and drops the writes. Inbound accesses to this region are illegal.
  - FEC0_0000 - FECF_FFFF: This segment is used for SAPIC messages. The chipset claims outbound accesses to this region and forward them to the appropriate PCI bus. The processors use this region to program the SAPIC or IOAPIC registers and for targeted EOI writes. Inbound accesses to this region are illegal.
- The (4G minus 20M) to (4G minus 32M) region is reserved for chipset specific functions. Inbound accesses to this region are expected only from the compatibility PCI bus from either a server management or a validation card. IB accesses are directed to the system bus and then forwarded to the appropriate PCI segment. This region is segmented as follows (unlisted regions are reserved):
  - FE20_0000 - FE3F_FFFF: This range is for programming the GART. Reads and writes are sent to Expander port-2 to be forwarded to the GXB. If the DEVNPRES bit for Device 14 is set (meaning that there is no xXB attached to the Expander bus), then accesses to this region will be forwarded to the compatibility bus for termination.
  - FEB0_0CB0: This address is for a memory-mapped register in the SAC.

— FEB0_0CC0: This address is used for BSP selection. It is a write once register in the SAC.

Figure 4-1 shows how the SAPIC and GART spaces are allocated. There may be up to 255 I/O SAPICs in the system. There is one region defined for the GART space.

4.1.3.1 Variable GAP

The variable gap starts at 4G-32M, and can grow downward in 32M increments. This gap is used to provide memory mapped I/O spaces to all of the logical PCI buses (this includes AGP buses) in the system. Each logical PCI bus is allowed n x 32M of contiguous space. PCI bus #0 is allowed the first a*32M below (4G-32M), PCI bus #1 is allowed the next b*32M, PCI bus #2 is allowed the next c*32M, etc. The total gap size, n, is equal to a + b + c and so on. The combined size of the variable and the fixed gaps must equal a multiple of 64 MB. Since the variable gap starts on a 32 MB boundary, the variable gap must total to an odd multiple of 32 MB. This limit is set up by firmware and is a function of the memory controller design.

4.1.4 High Extended Memory (above 4G)

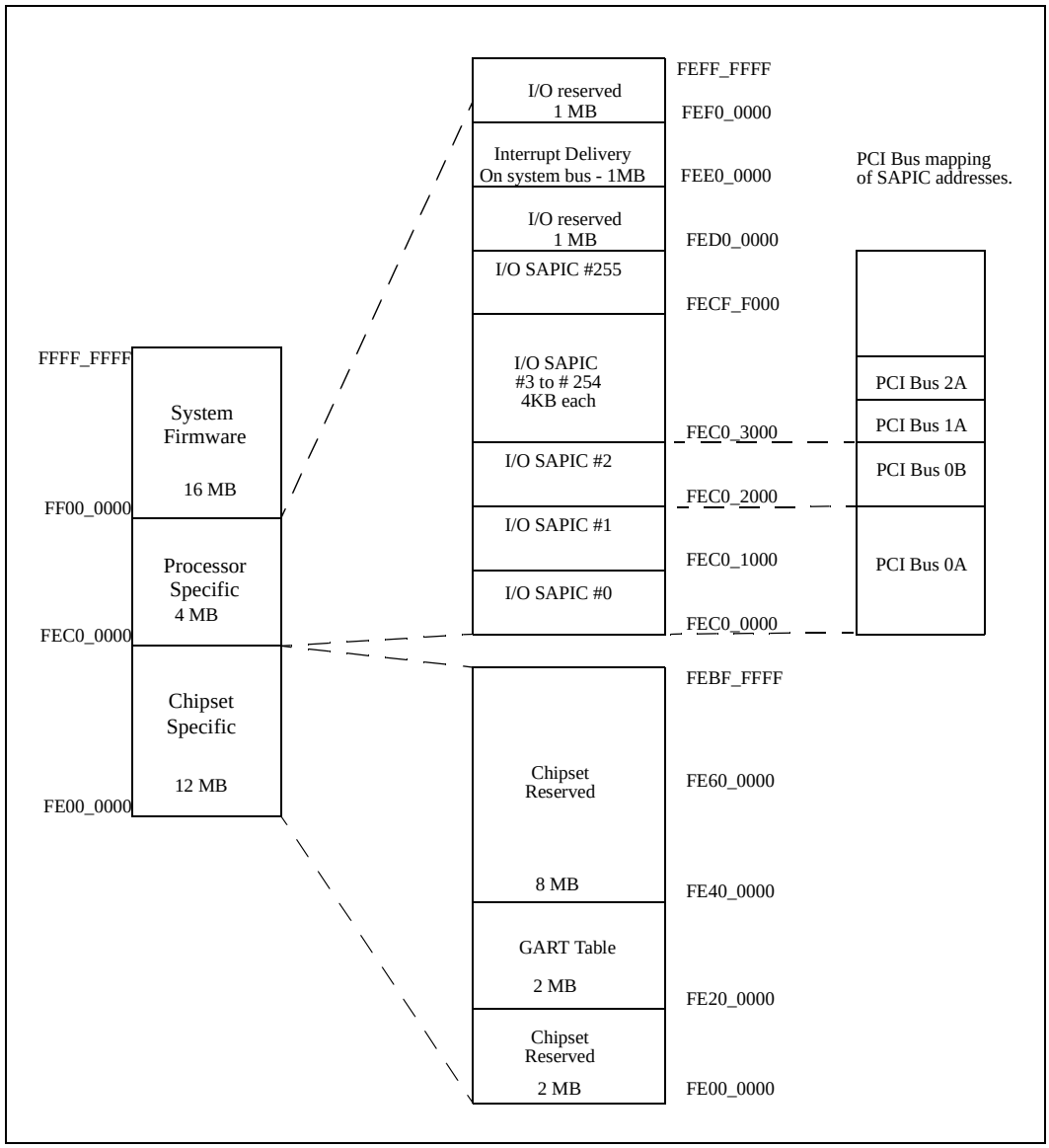
The entire address space above 4 GB is treated by the 460GX chipset as ordinary memory. The top of system memory is calculated by firmware. Processor accesses above the top of system memory are still claimed by the chipset, but are not forwarded to memory or PCI; instead they cause a BINIT#. Inbound accesses to this region can only occur due to a programming or address parity error. Firmware programs both the PXBs and GXBs with the Top of Memory value. A programming error that results in a PXB access above the Top of Memory causes the PXB to route the request as if it were to a peer PCI bus. Therefore the request goes through the SAC decoder and causes a BINIT#. A programming error that results in a GXB access above the Top of Memory (detected after GART translation) causes the GXB to force a BINIT#. Address parity errors are detected at each of the major bus interfaces.

4.1.5 Re-mapped Memory Areas

Any DRAM that lies behind an address that is mapped to PCI or is reserved in the region below 4 GB and above 1 MB is recovered. The memory that lies in the Medium Extended Memory Region is moved so that it is addressed above 4 GB. This covers all addresses in the Medium Extended Memory Region. For example, if there is 3 GB of memory and the Medium Extended Memory Region is 2 GB total (covering PCI, AGP and the reserved area), then the first 2 GB of addresses are directed to DRAM, the next 2 GB of addresses are directed to PCI or are AGP addresses or reserved, and the address range between 4 GB and 5 GB would be directed to DRAM. In other words, to access the 2 GB+1 byte of memory, the processor would use address 4 GB+1, since the physical address of 2 GB+1 is now mapped to PCI.

Any DRAM not used in the Low Extended Memory Region, i.e. the region below 1 MB is not recovered. The DRAM in this region which has its address directed to PCI is simply lost to the system.

Figure 4-2. Itanium™ Processor and Chipset-specific Memory Space



4.2 I/O Address Map

The 460GX chipset allows I/O addresses to be mapped to resources supported on the I/O buses underneath the 460GX chipset controller. This I/O space is partitioned into sixteen 4K byte segments. Each of the segments can be individually configured to any I/O bus. Segment 0 is always assigned to the compatibility I/O bus (of which there is only one per system).

There are four classes of I/O addresses that are specifically decoded by the 460GX chipset:

- All I/O addresses less than 100h: These addresses are specifically decoded as “defer-only” addresses. The SAC does not post any I/O accesses to this range, regardless of the state of the I/O posting enable bit. This is necessary because I/O accesses below 100h have historically had ordering side effects: e.g. accesses to the 8259 Interrupt Masks.

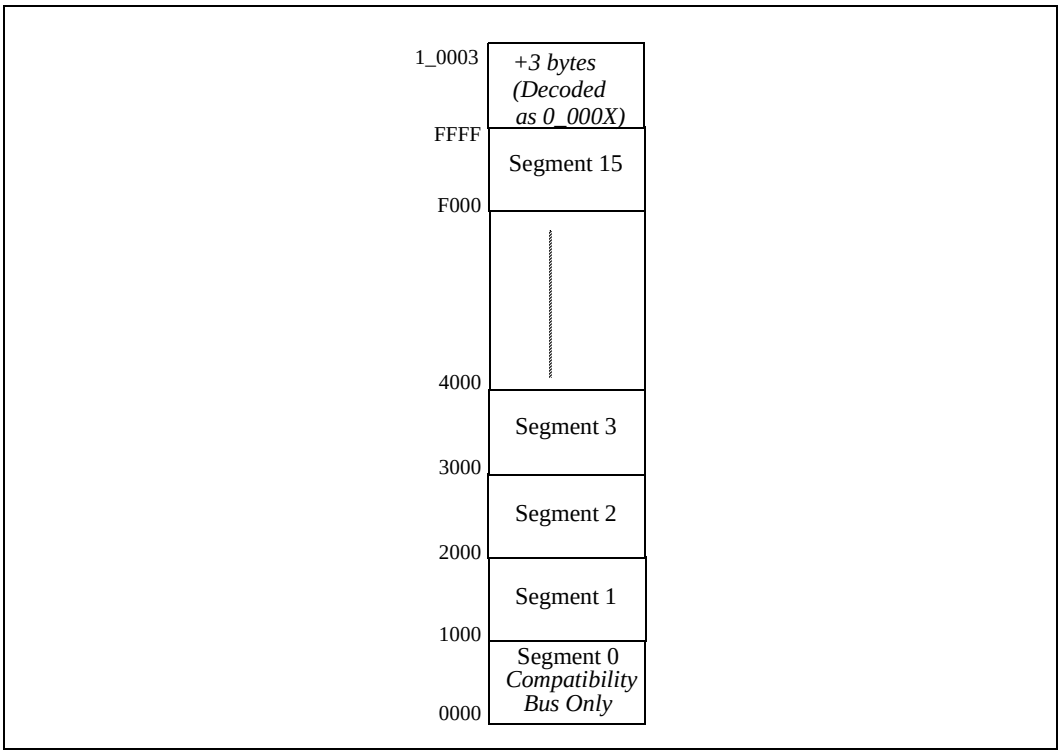
- I/O addresses used for VGA controllers: 03B0h-03BBh and 03C0h-03DFh. These addresses are specifically decoded so they can be mapped to the PCI bus specified by the VGA Space Register. An I/O access must be contained fully within the VGA I/O range to be remapped (e.g. an I/O read spanning 03BBh and 03BCh would not be remapped because it crosses the VGA I/O range). Posting of this range for writes is controlled by the state of the I/O posting enable bit in the Software-Defined Configuration Register.
- I/O addresses used for the PCI Configuration Space Enable (CSE) protocol. The I/O addresses 0CF8h and 0CFCh are specifically decoded as part of the CSE protocol. These addresses, like the I/O addresses less than 100h, are treated as “defer only” addresses.
- Posting for all other I/O addresses is controlled by the state of the I/O posting enable bit in the Software-Defined Configuration Register. If this bit is set, I/O writes are posted. If this bit is not set, all I/O writes are deferred. I/O reads are always deferred.

Note, the 460GX chipset does ***not*** support ISA expansion aliasing. The IFB supports a full I/O space decode, so the compatibility issue will be drivers that rely on the I/O aliasing behavior.

Historically, the 64k I/O space actually was 64k+3 bytes. For the extra 3 bytes, A#[16] is asserted. The 460GX chipset decodes only A#[15:3] when the request encoding indicates an I/O cycle. Therefore accesses with A#[16] asserted are decoded as if they were accesses to address 0 and will be forwarded to the compatibility bus. Since they look like accesses less than 100h, they are always deferred rather than posted. The full address is sent to the PXB and on to the compatibility PCI bus, which therefore has PCI address bit A#[16] active.

At power-on, all I/O accesses are mapped to the compatibility bus. An I/O access is never forwarded inbound by the chipset. The I/O address map is shown in Figure 4-3.

Figure 4-3. System I/O Address Space

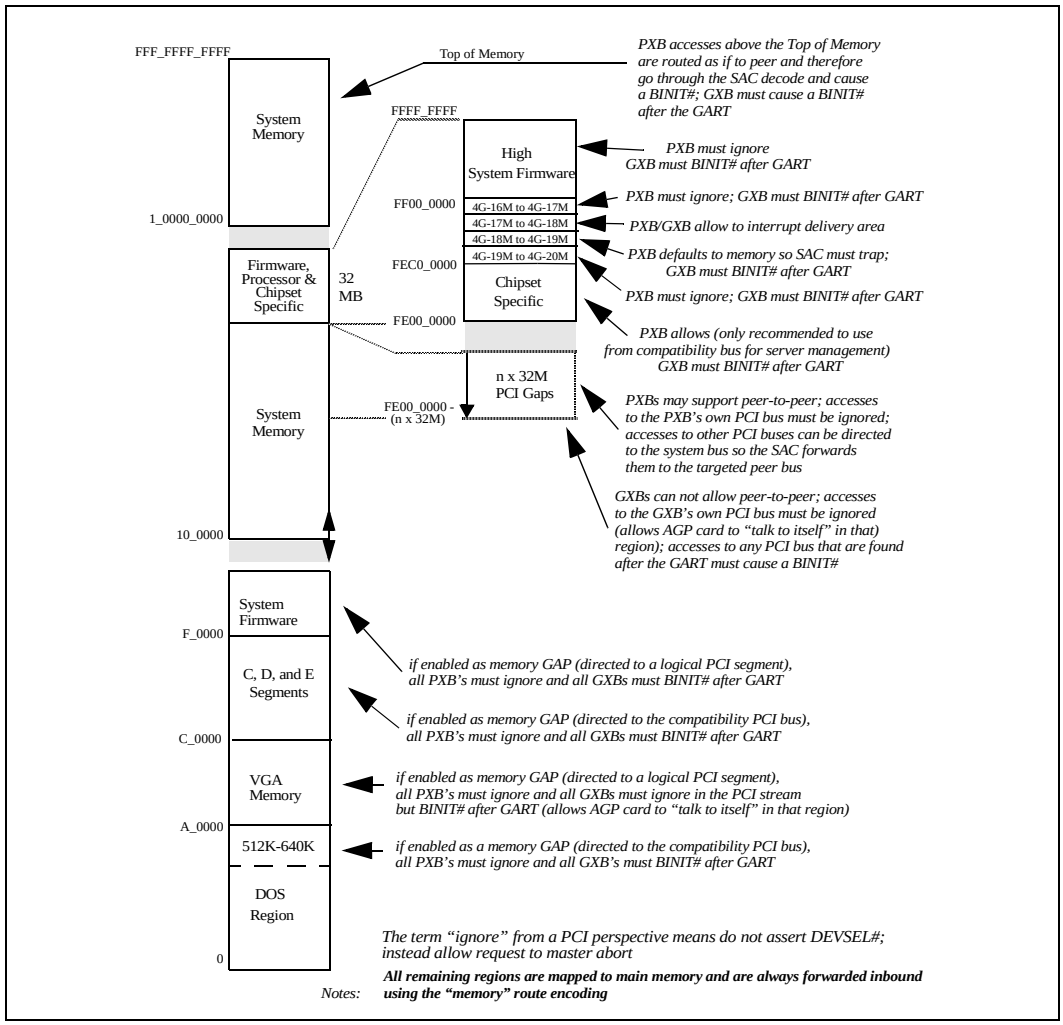


4.3 Devices View of the System Memory Map

Figure 4-1 shows an Expander Bridge device's view of system memory. The goal is to prevent invalid accesses at the expander bridge level, since different expander bridge devices are allowed to access different regions. For example, PXBs are allowed to access other logical PCI segments and GXBs are not. The SAC does not perform special checking to prevent this, and therefore the expander bridges must be set up by firmware accordingly. An exception to this rule can be made if the request is being routed to the system bus rather than directly to memory. For instance, accesses above the Top of Memory are not blocked at the PXB level, but instead cause a BINIT# in the SAC because they are guaranteed to go through the SAC's decode logic since the PXB routes these accesses to the system bus

Note: Figure 4-1 shows that parallel segment peer-to-peer accesses are only supported when initiated by a PXB (GXBs are only allowed to initiate accesses directed to system memory) and directed to one of the nx32M logical PCI segments (this segment can be anywhere except below the compatibility PCI bus). Parallel segment peer accesses are not permitted anywhere below 1 MB, not even to the VGA region. This means that if VGA is relocated below a GXB, a server management card on one of the PCI buses in the system can no longer access the VGA range.

Figure 4-4. System Memory Address Space as Viewed from an Expander Bridge (PXB/GXB)



4.4 Legal and Illegal Address Disposition

Below is the disposition of addresses done by the Bus Interface Unit (BIU).

Table 4-1. Address Disposition

Address Range	Outbound	Inbound	Dest. Decision
0-07FFFFh	DRAM	DRAM	
080000h-09FFFFh	DRAM	DRAM	MAR=11 or (Read and MAR= 01 and no system bus LOCK#) or (Write and MAR = 10)
	PCI0a	unclaimed	MAR=00 or (Read and MAR=01 and system bus LOCK#) or (Read and MAR=10) or (Write and MAR=01)
0A0000h-0BFFFFh	DRAM	DRAM	VGASE=0
	PCIx	unclaimed	VGASE=1
0C0000h-0EFFFFh (divided into 12 regions of 4k bytes)	DRAM	DRAM	MAR=11 or (Read and MAR= 01 and no LOCK#) or (Write and MAR = 10)
	PCI0a	unclaimed	MAR=00 or (Read and MAR=01 and LOCK#) or (Read and MAR=10) or (Write and MAR=01)
0F_0000h-0F_FFFFh	DRAM	DRAM	MAR=11 or (Read and MAR= 01 and no LOCK#) or (Write and MAR = 10)
	PCI0a	unclaimed	MAR=00 or (Read and MAR=01 and LOCK#) or (Read and MAR=10) or (Write and MAR=01)
10_0000h - PCIS[7]	DRAM	DRAM	PXB uses LXGB instead of PCIS[7]
PCIS[7] - FDFF_FFFFh	PCIx	PCIx	PCIS register determines target PCI bus On PCI if MMBASE<=address<=MMT, then not claimed. GXB: unclaimed
FE00_0000h-FE1F_FFFFh	undefined	undefined	This region is reserved.
FE20_0000h-FE3F_FFFFh	Expander port-2 or PCI0A	Expander port 2 or PCI0A	If DEVNPRES[14]=0, send to Expander port 2, else send to PCI-0a
FE40_0000h-FE5F_FFFFh	undefined	undefined	This region is reserved.
FE60_0000h to FEBF_FFFFh	Config unit or PCI-0a	Config unit or PCI-0a	If <=8B then: if one of defined registers read or write value, if not then return all 1's or terminate writes. If >8B, then send to PCI-0a. *NOTE: Locks to this range are forbidden and will hang the system*
FEC0_0000 to FECF_FFFFh	PCI x	unclaimed	SAR, IOABASE (PXB)
FED0_0000h to FEDF_FFFFh	PCI 0a or dropped	PXB will forward IB and mark it to memory.	Reads are sent to PCI-0a for master abort. Writes get No-Data response and are dropped
FEE0_0000h to FEEF_FFFFh	Interrupt Transaction; Reads sent to PCI 0a, writes dropped	interrupt for IB write, PCI-0a for read	Reads are sent to PCI-0a for master abort. IB writes are turned to interrupts. OB writes get No-Data response and are dropped

Table 4-1. Address Disposition (Cont'd)

Address Range	Outbound	Inbound	Dest. Decision
FEF0_0000h to FEFF_FFFFh	PCI0a	unclaimed	Reads are sent to PCI-0a for master abort. Writes get No-Data response and are dropped. PXB never claims this range
FF00_0000h to FFFF_FFFFh	PCI0a	unclaimed	Firmware region always enabled
1_0000_0000h to TOM	DRAM	DRAM	main memory (if present) above 4 GB
Above TOM	na	na	BINIT

Note: Accesses listed as "unclaimed" in the table for inbound transactions assume the PXB is programmed correctly. If an access were received up the Expander bus that hits in the listed address range, then its behavior is the same as outbound transactions to the same range.

Note: The PXB will never respond to an access that it is the master for. This means that an OB access will not be claimed by the PXB, even if that access hits a range to which the PXB would normally respond with DEVSEL#.

Note: The only ranges the PXB doesn't claim are MMBASE to MMT, FEF0_0000h to FEFF_FFFFh, and 4G-16M to 4G. If the PCI card initiates a request to any other address, it will be sent up as TPA or memory.

Memory Subsystem 5

The Intel 460GX chipset's memory subsystem consists of the SAC's DRAM controller, the SDC's buffering and datapath access, the MAC and MDC components, and the DRAMs themselves. Table 5-1 summarizes the 460GX chipset's general memory characteristics.

Table 5-1. General Memory Characteristics

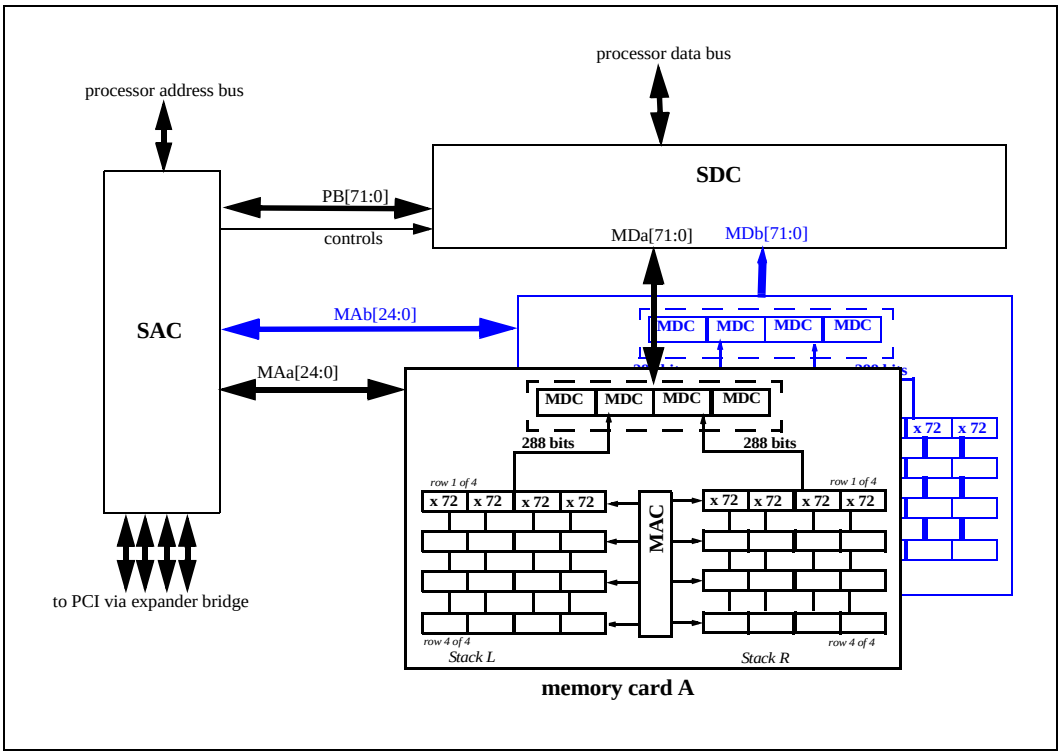
DRAM Types	Synchronous DRAM (SDRAM)
Maximum Memory Size	Up to 4 GB using 16MB DIMM's, up to 16 GB using 128 MB DIMM's, or 64 GB using 1 GB DIMM's
Minimum Memory	64 MB using 16MB DIMM's, 256 MB using 1 GB DIMM's
Memory Increment	64 MB is the smallest increment
Memory Modules	168 pin (x72) 3.3 volt DIMMs
DRAM Sizes	16 Mbit, 64 Mbit, 128Mbit, 256 Mbit
DIMMs/Row	4 DIMMs per row which must be populated as a unit
Rows/Stack	Up to 4 rows per stack; may populate any number of rows
Stacks/Card	2 stacks per card; may populate either 1 or 2 stacks
Cards/System	2 cards per system; 1 card per memory port, may have either one or two cards in the system
DRAM Types	Synchronous DRAM (SDRAM)

The SAC's DRAM controller provides addresses and commands to the 2 MAC's on each memory card. The MAC generates row and column addresses for the DRAM array and controls the data flow through the MDC. For processor-memory cycles, the address is received from the system bus, and data flows through the SDC to/from the memory cards. For PCI-DRAM cycles, the address is presented on the bus. The data moves between PCI and DRAM within the SAC/SDC, without appearing on the system bus. AGP-DRAM cycles are done non-coherently (unless coherent AGP is being used) and therefore do not have their addresses placed on the bus. The data for AGP transactions, like PCI cycles, flows to/from the DRAM within the SAC/SDC.

5.1 Organization

The 460GX chipset supports 1 or 2 memory cards. Each card supports up to 8 GB of memory using 128 MB DIMM's (32 GB with 1 GB DIMM's); 2 cards provide up to 16 GB of memory (64 GB with 1 GB DIMM's). There are 2 independent interfaces to the SAC/SDC from the memory subsystem, running simultaneously. Each memory interface supports 1 card and has a 72-bit datapath and a separate control path. Running at 266 MHz, each interface allows a 2.13 GB/s peak transfer rate, for a total of 4.27 GB/s of Bandwidth. Figure 5-1 illustrates this maximum configuration. platform.

Figure 5-1. Maximum Memory Configuration Using Two Cards



Each card is organized as 2 stacks of up to 4 rows each. A stack consists of 1 to 4 rows of DRAM which share a common data bus. A row consists of the 4 DIMM sockets which have a common address/control bus. A row is the minimum atomic unit that can be accessed. Each stack has a separate data path from the MDC to the DRAMs. Data may be transferring on both stacks simultaneously. For instance, on the same card, stack L could be doing a read while stack R is doing a write, or both could be doing a read.

Each row represents a set of memory devices simultaneously selected by a RAS signal and having a common address bus. Each row generates 288 bits (256 data, 32 ECC) of data. Each row uses 4 of the x72 (168 pin) DIMMs. There may be multiple banks per row. For instance, SDRAMs have 2, 4 or 8 banks internal to the chip. Or the DIMM may be organized such that there are 2 groups of memory on the DIMM itself (double-sided DIMMs), with 2 chips dotted together on the data pin. Since a row provides 256 bits (32 bytes) of data, the 64 byte line transfers of the processor will access a row twice to read or write the entire line. Data is interleaved by the MDCs to exchange 72 bits of data per transfer with the SDCs at the rate of one cache line every 30ns (2.13 GB/s) per interface.

There is a separate address and control bus for each memory port, with 1 card on each port. These are independent and may be driving addresses at the same time. While one bus is driving a read, the other could be driving a write from the write queue. This allows greater bandwidth and allows writes to be done without interfering with read traffic.

A system may be built with only one card, and provide only half the possible bandwidth, or have the memory on the motherboard itself. If there are 2 memory cards, then one is placed on each interface. Average memory latency is a function of how many interleaves are available and bandwidth. The present structure interleaves across cards first, then across stacks, and then rows. With 2 cards, each having 2 stacks of memory, Line 0 would be on cardB/stackR, Line 1 would be on cardA/stackR, Line 2 would be on card B/stackL, and Line 3 would be on cardA/stackL. See

Figure 5-2 for an illustration. In theory all 4 of these lines could be transferring data at the same time. It would then be muxed by the MDC to the SDC and then by the SDC to the bus. This allows data to be moved with no dead cycles on consecutive reads. Another possibility is that the data from cardA is going to the system bus, while the data from cardB is going to a PCI port. Since each memory interface can support the transfer rate of 266 MT/s, data transfers to PCI are completely overlapping transfers to the system bus such that there are no holes in the data traffic.

Table 5-2 gives a summary of the characteristics of memory configurations supported by the 460GX chipset.

Table 5-2. Minimum/Maximum Memory Size per Configuration

DRAM Technology & Configuration		DIMM Size	Double Sided? (Yes/no)	Number of Chips/ DIMM	Memory Size	
					Min	Max
16 Mb	2M x 8	2M x 72	No	9	64 MB	1 GB
	2M x 8	2Mx72x2	Yes	18	128 MB	2 GB
	4M x 4	4M x 72	No	18	128 MB	2 GB
	4M x 4	4M x 72 x 2	Yes	36	256 MB	4 GB
64 Mb	8M x 8	8M x 72	No	9	256 MB	4 GB
	8M x 8	8M x 72 x 2	Yes	18	512 MB	8 GB
	16M x 4	16M x 72	No	18	512 MB	8 GB
	16M x 4	16M x 72 x 2	Yes	36	1 GB	16 GB
128Mb	16M x 8	16M x 72	No	9	512 MB	8GB
	16M x 8	16M x 72 x 2	Yes	18	1 GB	16 GB
	32M x 4	32M x 72	No	18	1 GB	16 GB
	32M x 4	32M x 72 x 2	Yes	36	2 GB	32 GB
256 Mb	32M x 8	32M x 72	No	9	1 GB	16 GB
	32M x 8	32M x 72 x 2	Yes	18	2 GB	32 GB
	64M x 4	64Mx72	No	18	2 GB	32 GB
	64M x 4	64M x72 x 2	Yes	36	4 GB	64 GB

5.1.1 DIMM Types

The 460GX will support PC-100 DIMM's that meet the requirements defined in Table 5-3.

The only DIMMs supported are 3.3 volt 168 pin (x72) parts. These may be composed of 16Mb, 64Mb or 256Mb DRAMs. The 256Mb DRAMs must have a supply voltage of 3.3 volts.

The DIMMs used must have the serial presence detect (SPD) feature, since this is used to program the chipset configuration registers. DIMMs not having SPD will be considered as not-present in the system, since they are not visible to firmware.

DIMMs may have a buffer on the DIMM itself. The buffer can be used in a registered mode or a pass-through mode. The 460GX will support both buffered and unbuffered DIMMs. It will support the buffered DIMM in the pass-through mode, not the registered mode. Thus the timings of the state machines in the MAC will be the same for both types of DIMM. DIMMs with 36 components will be buffered. DIMMs with 9 or 18 components may be buffered.

5.2 Interleaving/Configurations

Maximum system bandwidth is obtainable in several ways. If the address patterns are well-behaved then one can use the page mode of the DRAM itself to obtain high bandwidths. Generally page hits can sustain about 5 times the bandwidth of page misses with a one-bank memory system. In systems with only several memory banks, designs tend to try and optimize the page hit rate to increase bandwidth.

A second approach is to have as many parallel operations within the memory system as possible. One can spread the addresses out across multiple DRAMs and have the data transfers in parallel. This lends itself well to designs which require a large memory system of many gigabytes.

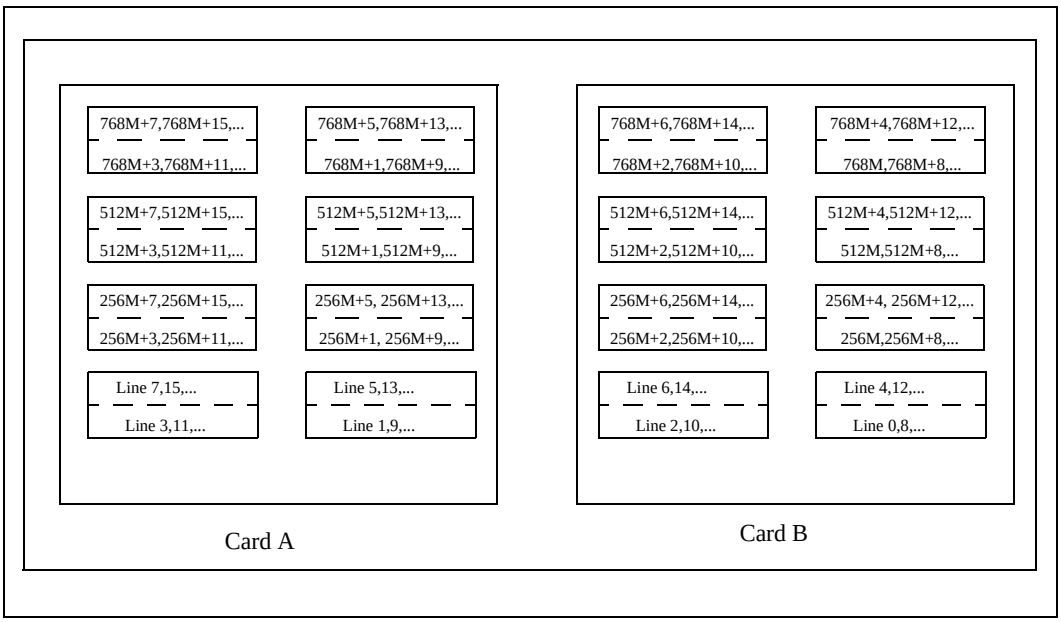
The 460GX will implement the second approach. It will attempt to increase the amount of parallelization. Addresses will be spread out across multiple rows and cards. Figure 5-2 shows the address layout. It assumes that 2Mx72 DIMMs are used, so that each row is 64 MB. With all the rows populated evenly we have 16 x 64 MB or 1 GB total memory space.

For sequential accesses, the addresses are laid out so that lines 0 and 1 can be accessed simultaneously, and can be transferred in parallel up to the final data transfer on the system bus. As 0 and 1 are being transferred, 2 and 3 can be started to the left stacks of each card and their data transfer will be done immediately following that of 0 and 1. The SDC buffers the data and sends it to the system bus with no dead cycles.

SDRAMs have at least two internal banks; 64Mb chips will generally have 4, and 256 Mb chips may have 8 internal banks. The 460GX takes advantage of these banks as well. In Figure 5-2, the rows are split into 2 halves. Since there are at least 2 banks in all SDRAMs, the system will interleave assuming all DRAMs have only 2 banks, and be split as shown. So with 16 rows, each split in 2, there is a 32 way interleaving scheme in a totally populated system.

The first 256 MB lies in the bottom 4 rows. The next 256 MB lies above that and so on up. This allows multiple processes, which may be spread throughout memory, to also be interleaved.

Figure 5-2. Address Interleaving



5.2.1 Summary of Configuration Rules

The memory system may populate any row in any order. There are preferred ways of populating the memory subsystem for performance, but all configurations will work.

The following rules summarize the way the memory system may be built up. The one hard rule is that a given row must be populated with 4 of the same DIMMs. There is no mixing allowed within a row. If the 4 DIMMs within a row are not the same, there is no guarantee as to system behavior, or that the system will even work.

- For each memory row, 4 DIMMs must be populated as a unit.
- The entire row must be populated with the exact same type of DIMM i.e. the same size, number of sides, technology (16Mb vs. 64 Mb), etc.
- Different rows may use different size DIMMs (2Mx72 in row 1 and 4Mx72 in row 2).
- Different rows may mix x4 and x8 DRAMs.
- Different rows may mix double sided and single sided DIMMs.
- Any combination of rows may be populated in any order, though performance will be affected by how the rows are populated.
- For highest performance, the total amount of memory in each stack should be the same.
- Either one or both memory cards can be populated in the system.
- Any number from 0 to 8 memory rows on a card can be populated in the system.

5.2.2 Non-uniform Memory Configurations

The example in Figure 5-2 has all the memory rows populated and all rows have the same size DIMMs. There is no requirement that memory be populated evenly. Some stacks may have fewer populated rows than others and the sizes within each stack may differ. Performance will be optimal with evenly populated rows. Knowing that users may not populate the card optimally, the 460GX will attempt to spread addresses out as best it can in an unevenly populated system.

For an easy example, use the example above, and assume that there are 4 rows of memory such that the first row of the first 3 stacks are populated with 4Mx72 DIMMs and the last stack has only 2Mx72 DIMMs; for a total of 448 MB. The addresses are broken up such that the first 256 MB are interleaved on a 4 way basis. The remaining 192 MB is interleaved on a 3 way basis; since all the addresses to the last (2Mx72 row) row have been used.

This algorithm is extended for multiple sizes and arrangements of populated rows. Each row is broken into multiple chunks and the least common denominator is found across stacks. Whatever the system configuration, there will be some level of interleaving between cards to increase parallelism.

5.3 Bandwidth

Sustained bandwidth is a function of traffic patterns as well as the system design and configuration. Each memory port can transfer 16 bytes per clock. This is a peak of 2.13 GB/s per port.

5.4 Memory Subsystem Clocking

The DIMMs are clocked at half the system bus frequency. For the Itanium processor, this means the DRAMs are clocked at 15 ns. Data is clocked out at the rate of 32B per 15 ns.

The following table lists the DRAM parameters used for the 460GX chipset.

Table 5-3. Required DRAM Parameters

Parameter	Symbol	Min. (clocks)	Max. (clocks)
Clock cycle time at CL=2	Tck	15 ns.	
Access time from CLK	Tac		6 ns.
CAS Latency	TCL		2
RAS latency			4
RAS cycle time	Trc		6
RAS to CAS delay	Trcd		2
RAS precharge	Trp		2
Data to precharge	Tdpl[a]		1

a. The sum of Tdpl and Trp are equal to Tdal as defined in the PC SDRAM Specification.

5.5 Supporting Features

5.5.1 Auto Detection

The memory controller provides capability for auto-detection of SDRAM type installed in the system during the system configuration and initialization, providing a Plug-and-Play DRAM interface to the user. This is done through the Serial Presence Detect logic on the DIMM. Firmware will read the Serial Presence Detect (SPD) for each row to determine the size of the memory in that row. Firmware will then write the size and interleaving information into the SAC and MAC through configuration cycles. Firmware will not have to go write data and see if it is there and do any addressing schemes to understand the system configuration. It will simply read a configuration register and then write a different configuration register with the chipset mapping. At the same time Firmware can calculate total system memory.

5.5.2 Removing a Bad Row

A row of memory may have a chip or DIMM fail. If an un-correctable error occurs, the system will machine-check, usually resulting in a reset. The 460GX will report which row failed. During the next re-boot or at power on, if the memory test fails, firmware may map the failing row as if it didn't exist. Since firmware goes through and reads the SPD on each row to determine its size, firmware can just set that particular row to a size of 0, as if it weren't there. No addresses will then be sent to that row. The entire row is removed, even if only one side of a double-sided DIMM were bad. But only the failing row is disabled. All other rows are still present, and the interleaving scheme will make the maximum use of the remaining rows. For example, a system with 8 rows populated that has one go bad will be restarted with 7 rows available.

5.5.3 Hardware Initialization

In order to decrease boot time of systems with large amounts of DRAM installed, hardware initialization of memory will be supported. Since multiple rows will be initialized simultaneously, the memory system will be able to initialize to 0 about 8 times faster than having the processor looping through memory with writes. The MDC will force all zeroes on the data lines, with good ECC, and the MAC will cycle through the memory addresses generating writes. The main limiter to the number of rows being simultaneously initialized is current draw on the DRAM. One row from each of the 4 stacks across the 2 cards will be initialized concurrently.

5.5.4 Memory Scrubbing

Scrubbing is the operation of walking through all installed DRAM and looking for errors. Each line is read and then written back, whether there is an error or not. Within the SAC there is an engine to generate addresses to be placed in the memory queue. These addresses are placed directly into the SAC memory queue and are not snooped on the system bus, nor are they checked for address conflicts, since the read-modify-write is treated as an atomic operation.

A scrub address is generated every 65K (65536) clocks. For a system with 32 GB of memory, this would walk through all memory every 3.2 days. The following table shows the approximate time to scrub memory based on memory size. Scrubbing may be disabled through a configuration bit.

Table 5-4. Scrubbing Time

Memory Size	Time to Scrub
64 MB	10 minutes
128 MB	20 minutes
256 MB	40 minutes
512 MB	1.2 hours
1 GB	3 hours
2 GB	5 hours
4 GB	10 hours
8 GB	20 hours
16 GB	1.6 days
32 GB	3.2 days
64 GB	6.4 days

Data Integrity and Error Handling 6

6.1 Integrity

This chapter explains the various errors in the chipset. Error handling requires catching the error, containing it, notifying the system, and recovery or system restart. Different platforms have different requirements for error handling. A server is most interested in containment. It wants bad data to be stopped before it reaches the network or the disk. On the other hand workstations with graphics may be less interested in containment. If the screen blips for one frame and the blip is gone in the next frame, the error is transient, and may not even be noticed.

The 460GX chipset will attempt to accommodate both philosophies. It will allow certain errors to be masked off, or will turn them into simple interrupts instead of fatal errors. Fatal errors are those which require a re-boot, e.g. BINIT#. Some errors will always be fatal, such as protocol errors or when the chipset has lost synchronization of queues or events. The user (OEM, O.S.) can decide the behavior for data errors. These may be considered as fatal, for maximum containment, or they may simply be reported as an interrupt while the system continues as best it can. If the data is moving to graphics, then an error may be unnoticed. It is possible that data entering memory as bad is never used, and therefore never shows up as an error to any user.

Each error will not be individually maskable. In general there are only 2 modes - aggressive and non-aggressive. In aggressive mode, every error - parity, protocol, queue management - will be considered fatal and lead to a BINIT#. In the non-aggressive mode, many errors will be reported as interrupts and not cause BINIT#. Even in non-aggressive mode, when the chipset has certain errors and doesn't know what to do with a transaction or seems out of sync across the chips, it will BINIT#.

The chipset will report errors at their use, instead of their generation. Both the processor and the chipset may 'poison' data. If the processor has an internal cache error, it may write out the data with bad ECC. If the chipset has bad parity on I/O data, it will corrupt the data as it is passed along. In both cases the data will be put in memory with bad ECC. If it isn't used, then no error is reported. If it is used, then the error is found at that point.

The 460GX chipset will isolate the error reporting as close to the error itself as possible. In some cases this can be to a failing DRAM or PCI card. In others it will be for a PCI bus or Expander port.

6.1.1 System Bus

- The 460GX chipset provides ECC generation on data delivered to the system bus, and ECC checking of data accepted from the system bus. Single-bit errors are corrected; multi-bit errors will write the data with bad ECC into the DRAM's (poisoned data) or to I/O with bad parity.
- Parity bits are generated and checked independently for the system bus address lines, the system bus request group, and the system bus response group. Errors typically result in the assertion of BINIT#.
- A variety of system bus protocol errors are also detected, and will result in assertion of BINIT#.
- The first instance of a bus error is logged with the address and error type. Additional status flags indicate subsequent errors occurred.
- For I/O accesses, good ECC is always generated for data with no parity errors. For data with bad parity, the data is poisoned with bad ECC as it's returned to the processor.

6.1.2 DRAM

- The 460GX chipset provides ECC generation on all writes into the DRAM, and ECC checking on all reads from the DRAM. Single-bit errors are corrected. Multi-bit errors will return poisoned data. Both types of errors are logged, with the address and ECC bits for the data being recorded. The row which failed, as well as the bit for single-bit errors, can be identified by software.
- The first instance of a single-bit error is logged. After the first error, additional status flags indicate subsequent errors occurred. The first multi-bit error is logged, with a status bit indicating there were more uncorrectable errors. In both cases, software can clear the error register and reset the error capture logic.
- Single-bit errors are corrected as they are received.
- To facilitate component debug and diagnostics, the ECC code generated on writes into the DRAM can be forced incorrect. A configuration bit, when set, will force the ECC bits that are written into memory to be XOR'ed with the correct value. This will allow either single or double bit failures to be generated in memory. When the data is read, the system should correct the data and report the error for single-bit errors, or report the error for double bit errors while passing bad ECC to the processor.

6.1.3 Expander Buses

- Parity bits are generated and checked independently for each Expander bus. For error behavior see Table 6-1.
- Hard Fail responses are supported.
- A mechanism for elevating fatal errors to BINIT# (XBINIT#) and non-fatal errors to BERR# (XBERR#) is provided.

6.1.4 PCI Buses

- Parity bits are generated and checked independently for each PCI bus.
- Standard PCI checking for aborts, PERR# and SERR# are also done.

6.1.5 AGP

- There is no parity on the bus between the graphics card and the GXB when using AGP protocol. Transactions using PCI protocol have parity as defined for the standard PCI bus.
- The GXB checks for illegal or unknown operations, Expander bus parity errors, or internal parity errors.
- There is parity on the GART table.

6.1.6 Private Bus between SAC and SDC

- There will be parity on the 64 bit data bus. Errors on data into the SDC will poison the data in the DB. Errors on data into the SAC will always be passed on without correction, with an option to BINIT#.
- The command bus and ITID bus are parity protected. Parity errors on this bus cause a BINIT#.

6.2 Memory ECC Routing

The ECC code used in DRAM is the same code as used in the Itanium processor, requiring 8 check bits to cover 64 bits of data. On the system bus, this code detects and corrects all single-bit errors, and detects double-bit errors.

The system designer has the option of wiring the boards such that the following is true:

- Using x4 DRAM's, multiple errors within one chip are 100% corrected
- Using x8 chips, all errors within a single chip are 100% detected

This is done by wiring the board so that each x4 DRAM has one bit in each of the 4 ECC words of a half-line. Since a half-line is 256 bits and the ECC is on 64 bits, there are 4 ECC words per half-line. For x8 chips, the bits are sliced across the 4 words, so that at most 2 bits from any one chip are in one ECC word. The ECC used on the processor will detect all 4-bit nibble errors.

6.3 Data Poisoning

When data is received that is uncorrectable, it will be passed on to the next interface as poisoned. The data may have come from memory or from the system bus with uncorrectable ECC errors. All data passes through the data buffer in the SDC. As uncorrectable data is placed in the data buffer it is marked that it was received as bad. When the data is read out of the data buffer and sent on, then the parity or ECC generated will be deliberately forced bad. Data is checked on a 'chunk' boundary, with a chunk being 64 bits of data.

Data to the system bus or to DRAM will have 2 bits of ECC corrupted for each failed chunk of data. These are bits 0 and 1 of the ECC bits, or bits 63 and 71 if looking at the entire 72 bits of data/ECC. Data passed to the private data bus will invert all the calculated parity bits associated with the failing chunk, thus passing bad parity to the private data bus.

6.4 Usage of First-error and Next-error

The first instance of an error is latched in the first-error status register (FERR). The first error does NOT set the bit in the next-error register (NERR). When an error is found, it is latched into the FERR if the FERR has no other bit set. If any bit is already set, then the appropriate bit in NERR is set.

Since the system needs to know if only one error has occurred or many, setting the FERR does not set the NERR. If there is another error of any type, including a second occurrence of the first-error, then the NERR is set. Software can read both FERR and NERR. If FERR is set but NERR is not, then only one error occurred in the system. If both are set, then multiple errors have occurred.

For the first error, as much information as possible is captured. The data, address and command information is captured if available. This allows isolation of errors and possible recovery.

In the case of 2 errors occurring in the same cycle, then 2 bits may be set in FERR. This should be a rare case. The other exception is for FERR_SAC. If there is a single-bit correctable ECC error from DRAM, then bit SCME will be set. This bit will not block other bits in FERR_SAC from being set. This allows software to poll periodically looking for single bit errors while not preventing other errors from being logged. Other than these two conditions, there should never be more than one bit set in any FERR.

Note: In the SAC if there is a single-bit error and a double-bit error reported from the SDC on the same cycle, then only the double-bit error is reported and only the double-bit error has its ITID captured in the SAC. The SDC will have its SEC bit set and so software must read and clear all the errors in the SDC after clearing the SAC.

A bit in FERR may be set that signals a minor error, such as correctable ECC error or other non-fatal error. Another error may occur before FERR is serviced, thus forcing a bit in NERR to be set. Since this next error may be fatal, both the FERR and NERR bits must be used to generate BINIT#, BERR#, INTREQ# or whatever the appropriate action is.

Both the FERR and NERR registers are write-1-to-clear registers. This means that software must write a one to the bits it wishes to clear.

For the GXB, all NERR and FERR registers must be cleared before FERR_GXB is cleared, otherwise FERR_GXB will be set again.

6.4.1 Masked Bits

Many of the errors have conditional reporting behavior. The error always sets the FERR register or NERR register. If the error is masked then a BINIT# or whatever is supposed to happen will not occur, but the bit is still set. The mask will not prevent the error from appearing in FERR/NERR. This allows software to poll, looking for errors that are not fatal to the system. If an error is masked, it will still set FERR and force all other errors to appear in NERR, thus losing logging information regarding later errors.

6.4.2 BERR#/BINIT# Generation

When an error occurs that forces BINIT#, then an enable bit in CONFIG2 is cleared as BINIT# is driven to the bus. The enable bit is automatically cleared in order to mask further BINIT# assertions. Software may also explicitly clear the enable bit to prevent BINIT# from occurring. When an error occurs, software should go out, clean up the error, clear the error status registers and only then set the enable bit so that new errors will be seen. When the bit is set, which can only be done by software writing the bit, then the SAC will assert BINIT# on an error. If a new error occurs after the first one is handled and is pending in FERR or NERR, then it will be reported when software re-enables BINIT# reporting.

The same behavior is true for BERR# as well. It is enabled by a separate bit in CONFIG2.

6.4.3 INTREQ#

The INTREQ# signal is driven by the SAC when it wishes to cause an interrupt and signal the operations system that an error has occurred that is non-fatal, but that may need to be logged. The signal is held asserted as long as the condition that caused the error exists. All the errors that cause an interrupt are OR-ed together to drive this signal. Software will reset the bit in FERR or NERR that caused the error. When the bit is reset, INTREQ# should be deasserted unless there is another error which holds the signal active.

After software clears the NERR/FERR bit that caused the error, it will do an EOI to the PID to re-enable interrupt reporting by the PID. If INTREQ# is still active after the EOI, then a new interrupt is generated.

6.4.4 XBINIT#

XBINIT# is an input to the SAC and an output from one of the xXB's or can also be generated by platform logic. XBINIT# is GTL+ level, and therefore all the outputs from the xXB's can be tied together and fed into the SAC's input. XBINIT# is held by the xXB until the xXB is reset, so if it takes multiple clocks to drive from the xXB to the SAC, there should be no problem.

6.4.5 XSERR#

XSERR# is an input to the SAC. It is generated by the system from the PCI SERR# signals or for other reasons. The system must OR all the generators together, either doing a wire-OR or adding logic for the OR'ing function.

6.5 SAC/SDC Errors

Many errors require reading both the SAC and SDC to isolate the cause. This is because most transactions involve both chips with a fair amount of handshaking between them. Refer to Table 6-1 for which register holds the status information and which ones hold error logging information. Also this table shows which errors are fatal and cause BINIT# and which are interrupts. Also some errors are maskable.

6.5.1 Data ECC or Parity Errors

The following errors are captured by the SAC and SDC together:

- SDC Non-Fatal Error (SNE). This is set for any of the following conditions: a) double-bit ECC error on memory b) double-bit ECC error from the system bus c) parity error on the private bus for data d) parity error on the private bus for byte-enables e) an internal SDC ram parity error f) a single bit correctable error on the system bus or g) the 2nd or subsequent single-bit memory ECC errors that are not recorded by the SDC as the first 1x error. On this error software must read the SDC to determine the type of error that was found. If the SDC reports only single bit errors, then the SNE bit was set for a 2nd or later single-bit memory error. Once the SDC's error registers are cleared, the first single-bit memory error is recorded as a SCME, and then later ones are SNE.
- SDC Correctable Memory Error (SCME). This is set on the first single-bit ECC error when reading DRAM. The error is always corrected before being passed on to the next interface. Since soft memory errors are not un-common, this bit has its own mask enable/disable. NOTE: if the SDC were to receive a single bit failure from both the A and B memory boards on the same clock, then FERR(SCME) and NERR(SCME) would be set. Otherwise, because only the first correctable error is reported as SCME, NERR(SCME) should not be set. This assumes that the SAC and SDC have their error registers cleared correctly. If the SDC has it's error registers cleared before the SAC and in between this time there is a new 1x memory error, then the SAC_NERR(SCME) bit would get set.
- SDC Fatal Error. This is set when the SDC detects conditions that will cause an unconditional BINIT#. The exact cause must be read from the SDC. This is set for parity errors on the control interface between the SAC and SDC or for protocol errors.

6.5.2 System Bus Errors

There are several errors that are detected by the SAC.

- System Bus Address Parity Error. Parity is checked on both address phases.
  - On A[43:24]#, detected by AP[1]#.
  - On A[23:3]#, detected by AP[0]#.
- System Bus Request Parity Error. Parity on both phases of the request bus is checked.
  - On REQ[4:0]#, detected by RP#.
- Address above TOM. Set for addresses above TOM and not in a PCI range. These addresses can't be sent to PCI0, since they may be greater than 4 GB, so are fatal. Out-bound addresses required a DAC on the PCI bus are not supported and would cause a fault at the xXB.
- Unsupported ASZ#. Since the GX only supports 36bits of address space, accesses which have ASZ# = 10b or 11b are an error.
- IOQ Underflow/Overflow. This occurs when there are no entries in the IOQ and the SDC attempts to do a Response Phase. Since the IOQ is empty, there should not be a Response phase. Or it can occur when either of the following 2 conditions are met: a) when there are 8 entries in the IOQ and a new ADS is seen b) when the IOQ depth is set to 1 for the system and the IOQ has one entry and a new ADS is seen.
- BERR# Observed#. When the GX sees BERR# active on the bus, whether driven by the processor or the SAC, it will elevate that to BINIT# if the 'BERR# to BINIT# Enable' bit in CONFIG register is set. If the enable bit is not set, then BERR# is ignored as an input. BERR# is driven active on the bus for 3 clocks. Each time a new BERR# assertion is sampled, BINIT# will be driven, unless the error is masked off.
- LOCK# Transaction with No Resources Available. Set when a LOCK# occurs and there are no outbound resources for the transaction. Since the lock can't be retried and there is no place to put the transaction, it gets dropped and lost.
- Resource Counter Overflow/Underflow. Set when the resource counter has an overflow or underflow. This occurs if there is a retirement to a counter that is empty or a transaction is not retried when the counter is full.

There is one logging register in the SAC for recording the actual error information. This is SA_FERR. It captures the system bus address and request for both the a and b phases. This register can be used to determine which bit is bad on parity errors.

6.5.3 SAC to SDC Interface Errors

The SAC will detect the following errors on the interface between the SDC and the SAC. They are all flagged in FERR_SAC.

- PDB ITID parity error. Set on a data transfer from SDC to SAC that has bad parity on the ITID sent with the data.
- Retirement Bus parity error. Set when the SDC attempts to retire an ITID and a parity error occurs.
- False retirement seen by SAC. Set when the SAC sees a retirement to an ITID that is not in use.
- 'Store-Retire' Command Underflow. Invalid write data sent from SDC to MDC. When the SDC sends data to the MDC, it signals that it did so to the SAC. This error is flagged when the 'data-sent' signal is seen by the SAC, but there are no writes to that stack pending in the SAC.

The SDC will capture the following errors on its side of the interface.

- PDB Data Parity Error. On data received from the SAC, parity is checked. If parity is bad, the data is sent to memory or the system bus with poisoned ECC.
- PDB Byte Enable Parity Error. On parity errors for the byte enables, the entire data transfer is sent to memory or the system bus with bad ECC. If the transfer is 64B then the entire line will have bad ECC.
- PDB Command Parity Error. Parity is checked over the Command bus from the SAC to SDC.
- PDB ITID Parity Error. Parity is checked on the ITID bus which accompanies the data from the SAC. The data is dropped.
- PDB Receive Length Error. Occurs when the SDC receives data and the length of the data transfer doesn't match the length indicated by the command for that transfer.
- Configuration Information Parity Error. Config accesses from the SAC to the SDC go through the data buffer in the SDC for both the address and data of the targeted register. Parity errors on reading out either the address or data from the buffer are detected.

6.5.4 SAC to MAC Interface Errors

The SAC will detect the following errors on the memory card interface.

- Completion Command Underflow. When the MAC has completed a transaction (either a read or a write), it will send a 'Complete' back to the SAC. Since each stack is kept ordered, the SAC can pop the top of the queue for the stack sending the 'Complete'. This error is flagged when the stack queue is empty and a 'Complete' is received for that stack.

The MAC will detect the following error in its interface with the SAC.

- Memory Card Error. This is set on a parity error on the command sent from the SAC. The command bus is a 23 bit bus (22 address bits and one parity bit). This error is flagged in the MAC when there is bad parity on this bus. The bus should always be driven with good parity. Parity checking is done every cycle. The MAC will complete any outstanding transactions, i.e. those with RAS already started. It will not start any new transactions for either stack until it is reset. Refreshes will continue, though no new accesses will start until the MAC is reset. The error is signaled to the system on the MAC's ERR# pin, which sets FERR_SAC [MAE or MBE].

6.5.5 SDC/Memory Card Interface Errors

The following errors are detected.

- Card x SEC. Data corrected and placed into the data buffer.
- Card x DED. Data placed into the data buffer with bad parity on each chunk that had bad ECC. Data is sent to the system bus with poisoned ECC or to the PDB with bad parity.
- 'Accept' Underflow. Set when the SDC received data from the memory card without having an 'Accept' command pending.
- 'Forward' Underflow. Set when the SDC receives a Forward signal from the MAC without a corresponding 'Store' command.
- 'Forward' Overlapping 'Forward'. Set when the SDC is doing a 'Forward' by sending data and then another 'Forward' is seen before the first finishes. The SDC does not queue up these commands and the timings would imply that two lines are being transferred at the same time.
- 'Load' overlapping 'Load'. Set when the SDC is doing a 'Load' by receiving data and a second 'Load' is seen before the first 'Load' finishes. This implies the MDC is sending data for two different lines at the same time.

- ‘Load’ Overlapping ‘Forward’. Set when the SDC is doing a ‘Forward’ by sending data to the MDC and the MDC starts to send the SDC data before the ‘Forward’ is complete.
- ‘Forward’ overlapping ‘Load’. Set when the SDC is receiving data from the MDC, and then is told to send data to the MDC at the same time it is receiving data.

6.5.6 SDC/System Bus Errors

- LEN# Protocol Error. Set when the actual write data received not matching length given by the system bus transaction. Set when the processor transfers either more or less data than is indicated by the LEN# encoding on the bus transaction.
- Write Data Protocol Error. Set when data is received from processor without a TRDY# having been given. Set when the processor places data on the bus without the correct TRDY# assertion. This may be for writes or implicit writebacks. It is also set if the processor drives a DRDY# on a zero-length write transaction.
- DRDY# Protocol Error. This error occurs and is flagged if a) DRDY# is not deasserted 1 clock after DBSY# deassertion OR b) DRDY# is not deasserted 2 clocks after SBUSY# deassertion. Either of these implies that the processor is holding onto the data bus longer than it is supposed to.

6.5.7 SDC Internal Errors

The SDC has two internal error bits that it detects.

- Data Buffer Parity Error. This is set when the data buffer detects a parity error on data that was placed into the buffer as good. Data with uncorrectable errors, such as double-bit ECC errors or data parity errors from the private bus, is placed in the buffer with good parity and has a bit set to indicate that the data is uncorrectable. If the data is read out of the buffer and parity is bad, then an internal alpha-hit or other error occurred.
- Simultaneous write-one-to-clear and hardware set. When the SDC_FERR register is set, it can only be cleared by writing a one to the set bit. If on the cycle that SDC_FERR is cleared, there is an error for that same bit, then 2 things happen a) the bit is set to a one and behaves as normal and b) the Simultaneous Clear and Set error bit is set. This bit indicates that there was not time to reset the logging registers associated with the incoming error, so that the log registers are stale and cannot be considered as valid.

  If software is writing a single one to the unique asserted bit in SDC_FERR, then the Simultaneous bit is only set if the new error is to the bit being cleared by software. If software is clearing the register by writing all 1’s to every location, then any new incoming error will also set the simultaneous bit. Software should only write to the single location (or locations) that it wishes to clear, and not write all 1’s to the entire register.

6.6 Error Determination

The status registers listed in Table 6-1 show which error occurred. Many of the errors also capture the actual error. This is listed under ‘Log Register’. Parity or ECC errors generally capture the data and the parity/ECC bits for the failing transfer. This information can be used for debug and diagnostic purposes. The log register is updated when the appropriate bit is set in the status register. Only after the status register (FERR register) is cleared will a new value be captured on subsequent errors.

Other errors capture the address associated with the failure. This is also for debug and diagnostic purposes, but also has the potential for use in system recovery. For instance, if there is an uncorrectable error on a data read, and the access can be isolated, then instead of re-starting the whole system, it might be possible to kill only the failing process and allow other users to continue running.

6.6.1 SAC Address on an Error

The SAC has several registers used to access the address for a failure. After FERR_SAC is read to determine the precise error that occurred, the address can then be determined for certain errors. The method is somewhat indirect. The SAC is the only chip that tracks the original address, so is used to get the address even when another chip may have detected the error.

The GX keeps track of all transactions using an ITID (Internal Transaction ID). All the chips use this tag to track a specific transaction through the system. When the transaction is compete the ITID is retired and a later transaction may re-use the same value. On an error the ITID for that failing transaction is captured and not retired. There are 3 registers used by the SAC to capture the failing ITID:

- SECTID - captures the ITID of the first Single-bit ECC error from memory.
- DEDTID - captures the ITID of the first Double-bit ECC error from memory.
- FSETID - captures the ITID of the first system bus or PDB data error seen by the SDC.

Each register is set as shown in Table 6-1.

SECTID, DEDTID, and FSETID all have ‘Valid’ bits in the register. The ITID is captured until the error is logged and the system is ready to clear all the error indication. Single, Double and system bus errors are recoverable, and therefore the system can clear those errors and continue running. To do so software must write a one to the ‘Valid’ bit of the register. This will cause the system to retire the ITID and that transaction is now complete.

Note that these 3 registers are sticky through reset, so that the information is preserved. After BINIT# or reset (but not power-on), the SECTID, DEDTID and FSETID registers are valid and the failing address can be retrieved from the RAM.

Multiple of these registers may be active. After the status register is read then the log register can be read to determine the ITID. The SDC signals the first of each type of error. If the first error the SDC sees is a Double-bit ECC then the FERR_SAC and SDC_FERR bits are set for that along with DEDTID in the SAC. If the SDC sees a single-bit error next, before software has cleared out the error logging, then the SDC_NERR and NERR_SAC registers are set, as well as SECTID. Anytime the SDC sends an indication of a Single, Double or system bus error, the appropriate SECTID, DEDTID, FSETID register is set. The SDC is responsible for not sending a 2nd indication of one of these errors, until it’s FERR and NERR registers are cleared.

When there is an error in the SDC and software has finished processing it, it should follow the following procedure in the order given:

- Write to the ‘Valid’ bit of SECTID, DEDTID or FSETID to release the ITID and unlock the register.
- Write ‘1’ to clear those bits that software has read as asserted in FERR_SAC and the NERR_SAC registers. Software should not just write 1’s to the entire register.
- Since any of these 3 types of errors are reported through interrupt and not BERR# or BINIT#, the interrupt must be cleared so that the next error can be visible. Write an EOI to the PID.
- Write ‘1’ to clear the SDC_FERR and SDC_NERR registers. Writing a ‘1’ to either register in the SDC will clear both the SDC_FERR and SDC_NERR at the same time.

After this the error reporting is in the clean state.

After the ITID is found, the actual address is needed. Again this is somewhat indirect. There are 2 locations in the SAC in which the address may be found. One is the Bus Interface Unit's (BIU) CAM and RAM, and the second is the MIU's RAM. The BIU's CAM contains the address for coherent transactions. The RAM contains the address, command and other information. See the register definition for BIUDATA for the bit definitions of the information captured. For the MIU, the RAM contains the address of the transaction before it has been translated into a memory address. This address may have come from the system bus for a processor or a 460GX-initiated coherent access, or it may have come from AGP or other non-coherent source. The RAM/CAM may only be used for coherent transactions. The MIU tracks both coherent and non-coherent transactions. Non-coherent transactions are those sent by an AGP card. Software can simply read the MIU for all transactions. If software reads the MIU and RAM/CAM and gets the same value for the address, then the transaction was coherent and the rest of the RAM/CAM data is valid. If the 2 addresses are different, then software should not read the rest of the RAM/CAM.

To access the system bus's address, the ITID which was read from one of the registers listed above is written into BIUITID. The write to this register causes the register BIUDATA to be updated with the RAM and CAM contents associated with the ITID written into BIUITID. BIUDATA can be read and the address determined.

To access the memory's address buffer, the procedure is slightly different. This buffer is directly readable, instead of using the indirect approach used by the BIU. To read the MIU address do:

- Read ITID from one of the registers above:
  - — If the ITID is less than or equal to 31, then do a configuration read from BUS: CBN, Device: 1, Function: 2, Address: (80h+4*ITID). This is MEMTID0 register.
  - — If the ITID is greater than 31, then do a configuration read from BUS: CBN, Device: 1, Function: 3, Address: (80h+4*(ITID-32)). This is MEMTID1 register.

6.6.1.1 Special Notes on Usage of SECTID, DEDTID, FSETID Registers

The SAC's SECTID, DEDTID and FSETID registers all define bit [7] as a Disable bit. When set, the register will not capture the ITID and the transaction is retired immediately. There is one side-effect that isn't apparent when setting the Disable bit to '1'. Not only is the ITID not captured, the FERR or NERR registers are not set either. Therefore if the Disable bit is a '1', the SAC will not indicate any error, and there will be no interrupt or signal to the system. The SDC is not affected by setting the Disable bit, so it will continue to log the errors.

- — If software sets the Disable bit in any of these 3 registers, it is responsible for polling the SDC for errors or accept that the errors will not be reported to the system. It is recommended that the Disable bit be set only for special applications or usage.

6.6.2 SDC Logging Registers

The SDC has logging registers to capture single and multiple bit errors on all the interfaces. These are:

- SEC0 - first single-bit ECC error on memory card 0.
- DED0 - first double-bit ECC error on memory card 0.
- SEC1 - first single-bit ECC error on memory card 1.
- DED1 - first double-bit ECC error on memory card 1.
- PVD - first parity error on the private data bus.
- SECF - first single-bit ECC error on the system bus.

- DEDF - first double-bit ECC error on the system bus.
- PCMD - first parity error on the command bus.
- PITID - first parity error on the ITID bus.
- SDCRSP - first failing transmission on the response bus. The response bus does not have parity, instead it sends the response in clock x and the inversion of the response in clock x+1.

All of these registers are independent. Having one of these valid doe not block any other from being valid. They record the first error of each type. Each one is locked only when the bit which is associated with the particular error in either SDC_FERR or SDC_NERR is set. In theory all the above registers could contain valid error information. There may be many cases where there are both single and double bit errors set as valid, especially if the single-bit errors are not scrubbed. If one whole line is bad both FERR and NERR get set, since each data chunk of 16B is considered as an independent unit.

Note: The error logging registers are enabled anytime the FERR and NERR registers both have the appropriate bit cleared. If an error is found and the FERR and NERR registers are cleared, then the next error will overwrite the old value in the logging register. If FERR or NERR is not cleared, then new errors will not overwrite the logging register.

6.7 Clearing Errors

Firmware or the operating system must clear out all the error registers when returning from a BINIT# or a reset. Leaving error bits set in the error registers will cause the system to flag that an error from an earlier time is still present. Firmware should read each FERR and NERR register and log any bits that are set. It should then clear those bits and continue to the next group of error registers.

6.7.1 SAC/SDC Error Clearing

In the case of a data error, both the SAC and SDC will log the event. After handling the error, software should clear the error logging in the following order

- Read the SECTID, DEDTID, or FSETID register as above
- Clear the SECTID, DEDTID or FSETID registers by writing a ‘1’ to bit 6 of the appropriate register
- Clear the SAC_NERR register if set
- Clear the SAC_FERR register
- Clear the SDC error registers only after the SAC’s registers are cleared
- Re-read the error registers to make sure they are still clear
- Re-enable interrupts or other system signaling

6.8 Multiple Errors

With the number of errors that are detected in the 460GX chipset, there are many possible multiple error cases. There is no way to specify what can happen in the case of every combination. In general the first error that is found is the important one. If there is only one error then it may be possible for the operating system to recover and continue without a full re-boot of the system. If there is more than one error, except for correctable ECC errors, then the system is probably not

capable of any recovery. The first error, especially if it is fatal, may itself have caused downstream errors to be flagged. The error that is flagged as first should be considered as correct and an indication of some real problem. The problem may have been a transient condition or a true hardware failure. Errors that are flagged in the NERR register should not be considered as caused by a hardware failure, since the first error may have actually precipitated the later errors.

6.8.1 SDC Multiple Errors

There are several important cases of multiple errors in the SDC. These are:

The data buffer in the SDC is 16B wide. Each 16B is checked for errors. Therefore if a single line has errors in multiple 16B chunks, then both the SDC_FERR and SDC_NERR registers will be set. ECC checking is done on 8B, and it is possible that a 16B chunk has both 8B chunks as bad; this case will be reported as a single error and only SDC_FERR is set. Since this is one line, there is only one error reported from the SDC to the SAC, so only FERR_SAC is set (or NERR_SAC if FERR_SAC were previously set by a different error).

Multiple lines in the SDC may have errors. For example, memory ECC errors are marked in the Data Buffer as the data is read from memory and placed in the buffer. As the data is sent from the SDC to the appropriate target, an indication is sent to the SAC as to whether there was an error or not on that transfer. If multiple lines have a 2x memory ECC error, only the first line that was retrieved by the SDC will have the SAC FERR error set. Later 2x memory errors are flagged as generic errors. For the SAC_FERR, generic errors set the SNE bit. Therefore, if SAC_NERR has the SNE bit set, this simply means that the SDC had multiple errors that were non-fatal. This may be multiple errors of the same type or different types. Even 1x correctable memory errors will set the SNE bit of SAC_NERR if there are more than one.

One transaction may have any combination of front-side bus, single-bit memory ECC error, and/or double-bit memory ECC error. The following shows what is captured for the different cases, where FSE is front-side bus, DED is 2x memory ECC error, and SEC is 1x memory ECC error:

FSE ERR	SEC ERR	DED ERR	Status
0	0	0	Store nothing, no errors
0	1	0	Store ITID in SECTID, log SCME in FERR
0	0	1	Store ITID in DEDTID, log SNE in FERR
0	1	1	Store ITID in DEDTID, log only SNE in FERR
1	0	0	Store ITID in FSETID, log SNE in FERR
1	0	1	Store ITID in DEDTID, log SNE in FERR
1	1	0	Store ITID in FSETID, log only SNE in FERR
1	1	1	Store ITID in DEDTID, log only SNE in FERR

- If there are multiple 1x Memory ECC errors, then the FERR[SCME] bit is set in the SAC. Also the FERR[SNE] bit will be set on the second 1x Memory ECC error (since SCME does not prevent FERR from being written. If there are more than 2 errors, then NERR[SNE] will also be set. Software must look at the SDC to determine that all the errors were single bit errors.

6.8.2 SAC Multiple Errors

There are several important cases of multiple errors in the SAC. Some of these are caused by the SAC and SDC not being in one chip and therefore having delays in the handshaking paths that will allow events that occur after a fatal problem to appear as errors, even when they are not real. Some of the multiple errors and the behavior for the SAC are:

- If the system has a fatal error and there is a BINIT#, then the NERR register may contain indications of other errors. For instance if there is a retirement from the SDC during the clock the BINIT# is clearing the system bus queues, then a retirement underflow may be flagged.

6.8.3 Single Errors with Multiple Reporting

There are some cases where multiple error bits are set as data is passed along.

Data moving to the Expander bus from the SDC may cause multiple error bits to be set. For instance, there is a PCI read that gets a system bus or memory ECC error. The SDC will flag the error and the expander unit will also flag that the error occurred. Either may come in first, so that FERR may have the expander bus error and NERR have SDC system bus error, or vice-versa.

A second case is for data from the expander bus to memory. The data is poisoned across the private data bus. Therefore FERR_SAC will be set for an expander bus parity error, and NERR_SAC will be set for a SNE error from the SDC.

The GXB may have an address from the graphics card that is in the aperture range, but finds the Valid bit cleared in the GART entry. This will set FERR_GART[2]. Since the address won't be translated, it is likely to be outside a valid memory range, thus an Illegal Address error is likely. Since this is one operation, both bits 2 and 1 of FERR_GART are set.

Data in the SDC is handled on an 8-byte basis, so that poisoned data causes 8 bytes to have bad parity. The expander interface in the SAC handles the data on a 4-byte basis. Therefore a single 8B data transfer that is on an 8B aligned address will cause both the FERR and NERR bits to be set, because each 4B is flagged as a separate error. If the access starts on an odd 4B boundary, then only the FERR register is set, since only 4 bytes are accessed out of the bad 8 bytes.

6.8.4 Error Anomalies

There are several cases that may have unexpected behavior. These are listed below.

Assume there is a 2x error in a certain 8B chunk of DRAM. There is a partial write which writes all 8 bytes of the bad chunk. One might expect that, since the partial write overwrote all 8 bytes of the chunk, the error would not be reported. The GX design is such that the data buffer in the SDC tracks errors on 8B chunks. The data is marked as having a 2x error when the line is read out of DRAM. Even though the entire 8B is overwritten, the chunk is still marked as having a 2x error. When the line with the merged data is written back into DRAM, the chunk that has the 2x error will be poisoned and later reads will report a 2x error on that line. NOTE: if there is a full line write, then any line in DRAM with a 2x ECC error will be overwritten, and the 2x error will never be seen.

An expander bus MWI writes an entire line into DRAM. On the system bus there is an implicit writeback. The data from the processor for the IWB has a 2x error. Even though the processor data is not used, since the expander bustransaction was a full line write, the data will be sent to DRAM as poisoned.

Take the case where processor 1 reads a line from memory and there are no errors, and then does a write into its cache. Later processor 2 does a read, getting an IWB. The SAC starts a speculative read for the line before seeing the HITM#. If the data in DRAM has a 2x error on this read the SDC data buffer will be marked as having a 2x error on this line. In this case, assuming the speculative read is done (in some cases the HITM# is seen before the memory cycle is attempted, so no speculative read is done), the IWB data that is written back into memory will be poisoned.

6.9 Data Flow Errors

Figure 6-1 and Figure 6-2 show the data flow and errors that are checked. The SDC will report DRAM errors for either single-bit (S) or double-bit (D) fails. The SDC will never signal that an error was fatal (causing BINIT#). The processor will handle data that it receives with bad ECC. The memory will have bad ECC written to it, if incoming data is bad. Data to the SAC will be handled by the SAC itself. Either the SAC will BINIT# on seeing bad data or it will pass it on to the xXB via an Expander port.

Figure 6-1. SAC Error Flow on Data

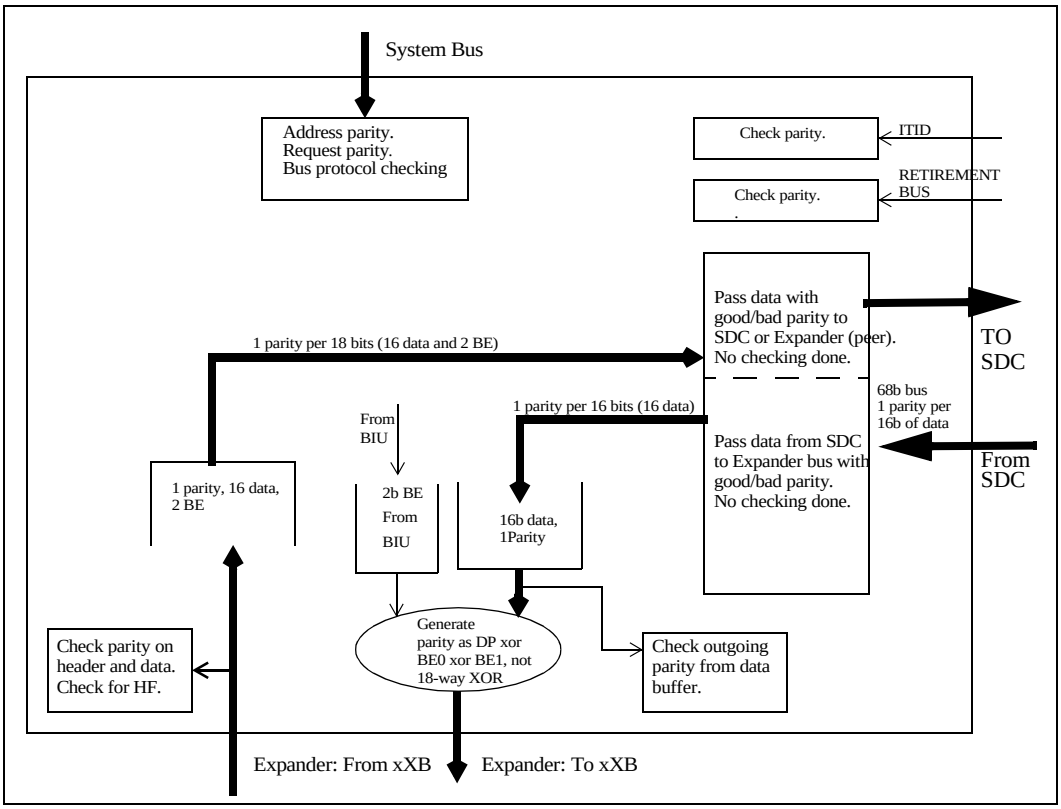
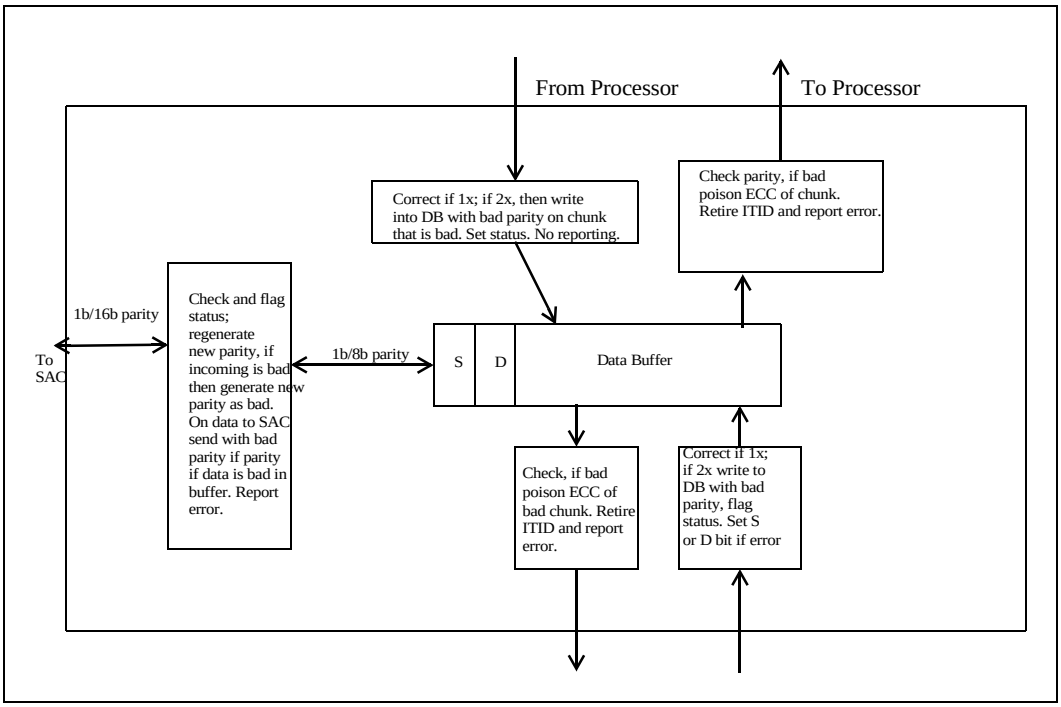


Figure 6-2. SDC Error Data Flow



6.10 Error Conditions

6.10.1 Table of Errors

Table 6-1 is a list of possible errors found in the system. The table shows the error and the system action. It also shows the information that is captured on any failure. The captured info is sticky through reset and can be read through configuration accesses. If the system action is conditional, then the qualifier column shows what the action is conditional on. Note that for the BINIT#, XBINIT#, and the interrupts there is a driver enable. If the driver is disabled, then these signals won't be active, even if it says Unconditional in the table.

Table 6-1. Error Cases

Error	Chip Detecting	System Action	Status Register	Log Register	Qualifier
System Bus 1x ECC	SDC	Correct the data and pass to bus. Conditional Interrupt.	SDC_FERR[SECF], FERR_SAC[SNE]	SECF_D_FERR, SECF_ECC_FERR, SECF_TXINFO_FERR, FSETID	SNE Enable
System Bus 2x ECC	SDC	Write into memory with bad ECC (poison data). Conditional Interrupt.	SDC_FERR[DEDF], FERR_SAC[SNE]	DEDF_D_FERR, DEDF_ECC_FERR, DEDF_TXINFO_FERR, FSETID	SNE Enable
Mem 1x ECC	SDC	Correct the data and pass to bus. Conditional Interrupt.	SDC_FERR[SECx], FERR_SAC[SCME]	SECx_D_FERR, SECx_ECC_FERR, SECx_TXINFO_FERR, SECTID	SCME Enable
Mem 2x ECC	SDC	Pass to system bus with bad ECC (poison data). Conditional Interrupt.	SDC_FERR[DEDx], FERR_SAC[SNE]	DEDx_D_FERR, DEDx_ECC_FERR, DEDx_TXINFO_FERR, DEDTID	SNE Enable
SAC to SDC Interface Errors					
PDB Command Parity Error	SDC	Unconditional BINIT#	SDC_FERR[CPE], FERR_SAC[SFE]	PCMD_FERR	
RSP Bus Transmission Error	SDC	Unconditional BINIT#	SDC_FERR[RTE], FERR_SAC[SFE]	SDCRSP_FERR	
ITID Parity Error	SDC	Unconditional BINIT#	SDC_FERR[IPE], FERR_SAC[SFE]	PITID_FERR	
PDB Receive Length Error	SDC	Unconditional BINIT#	SDC_FERR[RLE], DPBRLE_FERR, FERR_SAC[SFE]	*Nothing*	
PDB Data Parity Error	SDC	Failing Chunk of Data will be put in memory with bad ECC (poisoned), Conditional Interrupt.	SDC_FERR[DPE], FERR_SAC[SNE]	PVD_D_FERR, PVD_PAR_FERR, PVD_TXINFO_FERR	SNE Enable
PDB Byte Enables Parity Error	SDC	Entire Line of Data will be put in memory with bad ECC (poisoned), Conditional Interrupt.	SDC_FERR[BPE], FERR_SAC[SNE]	PVD_D_FERR, PVD_PAR_FERR, PVD_TXINFO_FERR	SNE Enable
Configuration Information Parity Error	SDC	Unconditional BINIT#	SDC_FERR[CIE], FERR_SAC[SFE]	*Nothing*	
False retirement seen by SAC	SAC	Unconditional BINIT#	FERR_SAC[FRE]	*Nothing*	
Retirement Bus Parity Error	SAC	Unconditional BINIT#	FERR_SAC[RPE]	*Nothing*	
PDB ITID Parity Error	SAC	Unconditional BINIT#	FERR_SAC[IPE]	*Nothing*	
'Store-Retire' Underflow	SAC	Unconditional BINIT#	FERR_SAC[SCxx]	*Nothing*	
System Bus ADD/CMND					
Address Parity Error	SAC	Conditional BINIT#	FERR_SAC[AE]	SA_FERR	Flag System Bus Parity Errors

Table 6-1. Error Cases (Cont'd)

Error	Chip Detecting	System Action	Status Register	Log Register	Qualifier
Request Parity Error	SAC	Conditional BINIT#	FERR_SAC[RQE]	SA_FERR	Flag System Bus Parity Errors
Unsupported ASZ	SAC	Conditional BINIT#	FERR_SAC[ASE]	SA_FERR	Flag System Bus Parity Errors
Illegal HITM (HITM on any Expander bus access)	SAC	Unconditional BINIT#	FERR_SAC[IHS]	*Nothing*	
Address above TOM	SAC	Unconditional BINIT#	FERR_SAC[TE]	*Nothing*	
LEN# Protocol Error	SDC	Unconditional BINIT#	SDC_FERR[FS0], FERR_SAC[SFE]	*Nothing*	
DRDY# Protocol Error	SDC	Unconditional BINIT#	SDC_FERR[FS2], FERR_SAC[SFE]	*Nothing*	
Write Data Protocol Error	SDC	Unconditional BINIT#	SDC_FERR[FS1], FERR_SAC[SFE]	*Nothing*	
BERR# Observed	SAC	Convert to BINIT#	FERR_SAC	*Nothing*	BERR# to BINIT# Enable
IOQ Overflow/ Underflow	SAC	Unconditional BINIT#	FERR_SAC[IUE]	*Nothing*	
External XBINIT# Active	SAC	Unconditional BINIT#	FERR_SAC[XBE]	*Nothing*	
External XSERR# Active	SAC	Conditional BERR#	FERR_SAC[XSA]	*Nothing*	BERR# Driver Enable
LOCK# when no resources are available	SAC	Unconditional BINIT#	FERR_SAC[LTE]	*Nothing*	
Resource Counter Overflow/ Underflow	SAC	Unconditional BINIT#	FERR_SAC[RCE]	*Nothing*	
Memory System Protocol Errors					
SAC-MAC Command Parity Error	MAC	ERR# pin asserted on MAC. Unconditional BINIT# by SAC.	FERR_SAC[MxE], FERR_MAC	CMND_FERR	
'Complete' Command Underflow	SAC	Unconditional BINIT#	FERR_SAC[CCxx]	*Nothing*	
'Forward' overlapping 'Forward'	SDC	Unconditional BINIT#	SDC_FERR[FWMDIx], FERR_SAC[SFE]	*Nothing*	
'Forward' overlapping 'Load'	SDC	Unconditional BINIT#	SDC_FERR[RdWrx], FERR_SAC[SFE]	*Nothing*	
'Load' overlapping 'Load'	SDC	Unconditional BINIT#	SDC_FERR[LRMDIx], FERR_SAC[SFE]	*Nothing*	
'Load' overlapping 'Forward'	SDC	Unconditional BINIT#	SDC_FERR[WrRdx], FERR_SAC[SFE]	*Nothing*	
'Forward' Underflow	SDC	Unconditional BINIT#	SDC_FERR[FRx, FLx], FERR_SAC[SFE]	*Nothing*	

Table 6-1. Error Cases (Cont'd)

Error	Chip Detecting	System Action	Status Register	Log Register	Qualifier
'Accept' Underflow	SDC	Unconditional BINIT#	SDC_FERR[AEx], FERR_SAC[SFE]	*Nothing*	
Internal SDC Error					
Data Buffer Ram Parity Error	SDC	Unconditional Interrupt	SDC_FERR[RPE], FERR_SAC[SNE]	*Nothing*	
GXB ERRORS					
AGP Request Queue Overflow	GXB	Unconditional XBINIT#	FERR_AGP	*Nothing*	
Use of Pipe with Sideband Enabled	GXB	Unconditional XBINIT#	FERR_AGP	*Nothing*	
AGP Address [..36] not = 0	GXB	Unconditional XBINIT#	FERR_AGP	*Nothing*	
Unsupported command using AGP semantics	GXB	Unconditional XBINIT#	FERR_AGP	*Nothing*	
PCI IB Read Que Data Parity Error	GXB	Conditional XINTR# Conditional XBINIT#	FERR_PCI	*Nothing*	TXDERR_INTE, TXDERR_BINITE
PCI OB Write Que Data Parity Error	GXB	Conditional XINTR# Conditional XBINIT#	FERR_PCI	*Nothing*	TXDERR_INTE, TXDERR_BINITE
Discard timer expiration	GXB	Unconditional XINTR# SERR# if SERRE set	FERR_PCI	*Nothing*	SERRE
SERR# Observed	GXB	Unconditional XINTR#	FERR_PCI	*Nothing*	
PERR# Observed	GXB	Unconditional XINTR#	FERR_PCI, possibly PCISTS[DPE]	*Nothing*	
PCI Parity Error on Address from Card	GXB	Let card master abort; SERR# and XINTR# if SERRE set; if SERRE not set then neither SERR# nor XINTR# driven.	PCISTS[PE], FERR_PCI, possibly PCISTS[SSE]	PAC_ERR	SERRE
PCI Parity Error on Data from Card	GXB	Data placed into queue with bad parity. Conditional PERR#	PCISTS[PE], PCISTS[DPE] if PERRE	PD_ERR, PAC_ERR	PCICM [PERRE]
Master Abort on Read by GXB	GXB	1DW: Return all 1's > 1DW: Hard Fail (HF) completion.	PCISTS [RMA], FERR_PCI	*Nothing*	
Master Abort on Write done by GXB	GXB	1 DW: normal completion >1 DW: Hard Fail (HF) completion.	PCISTS [RMA], FERR_PCI	*Nothing*	
Master Abort on Configuration Cycle	GXB	Normal completion	PCISTS [RMA] (FERR_PCI is not set)	*Nothing*	
Target Abort on Transaction Mastered by GXB	GXB	Return HF to either read or write.	PCISTS [RTA], FERR_PCI	*Nothing*	
GART Entry Invalid	GXB	Unconditional XINTR#, Conditional XBINIT#. *(NOTE: if XBINIT# is driven, then it is not required to drive XINTR#)*	FERR_GART	*Nothing*	GARTINV_BINITE

Table 6-1. Error Cases (Cont'd)

Error	Chip Detecting	System Action	Status Register	Log Register	Qualifier
GART Parity Error	GXB	Continue, use address as read from GART, Unconditional XINTR#, Conditional XBINIT#. (NOTE: if XBINIT# is driven, then it is not required to drive XINTR#)	FERR_GART	*Nothing*	GARTERR_BINITE
Illegal SMM Access	GXB	Unconditional XBINIT#	FERR_GART	*Nothing*	
Illegal Address	GXB	Unconditional XBINIT#	FERR_GART	*Nothing*	
Illegal OB GART Access	GXB	Unconditional XINTR#, Conditional XBINIT#, results undefined.	FERR_PCI	*Nothing*	GARTERR_BINITE
PXB Errors					
Detected at Expander Port					
Expander Par-err on any Header	PXB	Set Status. If outbound error handling is enabled then assert BINIT#, else then SERR#, drop request (no cmplt returned).	PCISTS [SSE], ERRSTS[2]	*Nothing*	MODES [3]
Expander Par-err on Write Data from Expander	PXB	Set Status. If outbound error handling in enabled then poison data as passed to PCI, else then SERR# and write out data as good to PCI.	PCISTS [SSE], ERRSTS[2]	*Nothing*	MODES [3]
Expander HF Read Cmplt. from SAC	PXB	PXB: Set Status. Target abort read to card. (Received in peer-to-peer)	PCISTS [STA]	*Nothing*	
Expander Par-err on IB Read Data	PXB	Set Status. If outbound error handling is enabled then poison data as passed to PCI, else then SERR# and pass read data as good to PCI.	ERRSTS[2], PCISTS[SSE]	*Nothing*	MODES [3]
Detected as PCI Master					
PCI Par-err on OB Read Data Received from Card	PXB	Set Status. Drive PERR#. Pass data with good parity to Expander. (PERR# is optionally elevated to SERR#)	ERRSTS[5], PCISTS[PE], PCISTS[DPE]	*Nothing*	
Master Abort on Read Done by PXB	PXB	Return all 1's	PCISTS [RMA]	*Nothing*	
Master Abort on Write Done by PXB	PXB	Drop data, normal CMP	PCISTS [RMA]	*Nothing*	
Target Abort received by xXB	PXB	Return HF to either read or write. If failed access is in middle of transaction then the remainder of the transaction is discarded.	PCISTS [RTA]	*Nothing*	
PERR# Asserted by Card	card	Optionally turned to SERR# by PXB.	ERRSTS[6], PCISTS[DPE]	*Nothing*	

Table 6-1. Error Cases (Cont'd)

Error	Chip Detecting	System Action	Status Register	Log Register	Qualifier
Detected as PCI Target					
PCI Par-err on Address from Card	PXB	Accept address as sent and process as if parity were good. Conditionally assert SERR#.	PCISTS[PE], PCISTS [SSE], ERRSTS[2]	*Nothing*	SERRE
PCI Par-err on Data for an IB Write	PXB	Set Status. Drive PERR#. Pass data with good parity to Expander.	ERRSTS [5]	*Nothing*	
PERR# Asserted by Card	card	Optionally turned to SERR# by PXB.	ERRSTS[6], PCISTS[DPE]	*Nothing*	

6.11 PCI Integrity

The PCI bus provides a single even-parity bit (PAR) that covers the AD[31:0] and C/BE#[3:0] lines. The agent that drives the AD[31:0] lines is responsible for driving PAR. Any undefined signals must still be driven to a valid logic level and included in the parity calculation.

6.11.1 PCI Bus Monitoring

When the PXB is not the PCI bus master, it will still check the address parity for all PCI address phases. If a parity error is detected and the Parity Error Response is enabled in the PCI Command (PCICMD) register, the event is logged in the Parity Error Detected bit in the PCI Status (PCISTS) register. If the PCICMD register's SERR# Enable bit (SERRE) is set, SERR# will also be asserted. When the PXB asserts SERR#, the PCISTS register's Signalled System Error bit (SSE) is also set.

Note that the expected settings for the GX system are: SERRE=1, PERRE=1, ERRCMD[3]=1, and ERRCMD[4]=1. ERRCMD[6:5] are to be programmed as each OEM, operating system or system user desires in order to enable SERR reporting of data parity errors. For data which has bad parity and is received from the PCI bus, the PXB will forward this up the Expander bus as if there were no parity error, so that ERRCMD[5] may need to be set for sufficient error containment, since PERR# asserted to the card will not prevent the data which had a parity error from being placed in memory

6.11.2 PXB as Master

6.11.2.1 Master Abort

If the PXB initiates a PCI transaction and no target responds, the PXB will terminate the transaction with a master-abort. The PXB will wait five PCI clocks after asserting FRAME# for a target to respond with DEVSEL#. If no target responds, the PXB will perform a master abort to terminate the cycle on the PCI bus. Special Cycle commands, which are broadcast to all PCI targets, will always be terminated with master abort. Therefore master-aborts during Special Cycle commands are not considered errors, and are never logged or reported.

When the PXB performs a master abort, if the command was not a Special cycle, it will log the event by setting the PCISTS register's Received Master Abort (RMA) bit. The PXB then has two options for generating a response.

The default option is to return a “normal” response. If the aborted transaction was a read, the PXB will return all 1’s for the data. If the aborted transaction was a write, the PXB will discard the write data. SERR# is not asserted in either case.

6.11.2.2 Received Target Disconnect

A PCI target may issue a disconnect to indicate it is unable to respond within the PCI latency guidelines. Disconnect is signalled when the target asserts both STOP# and DEVSEL#. The target controls whether another data transfer may occur by whether TRDY# is asserted when STOP# is asserted. A target disconnect is not usually issued on the first data phase of the transaction. Target disconnects are not considered errors, and are not logged or reported in any way. After a target disconnect, the PXB will wait at least two PCI clocks before re-arbitrating for the PCI bus to complete the transfer.

6.11.2.3 Received Target Retry

A PCI target may issue a retry to indicate that it is currently unable to process the transaction. Retry is signalled when the target asserts STOP# and DEVSEL# and does not assert TRDY#. Retry is actually a special case of disconnect that occurs before the first data transfer.

After receiving a retry for a transaction, the PXB will wait at least two PCI clocks before re-arbitrating for the PCI bus to retry the transaction. If the transaction is a write, the PXB will retry the transaction until it succeeds. If the transaction is a read, the PXB will retry the transaction until it succeeds, but may allow writes to pass it. Note, in all of these cases the retries are not considered errors. There is no logging or error reporting of any kind.

6.11.2.4 Received Target Abort

A PCI target may issue an abort to indicate that the current transaction should be terminated *and should not be attempted again*. This is a catastrophic failure. Target abort is signalled when STOP# is asserted and DEVSEL# is deasserted. The PXB will log the target abort by setting the PCISTS register’s Received Target Abort (RTA) bit. The PXB then returns a hard fail response to the SAC.

6.11.2.5 Data Parity Errors

When the PXB is the PCI bus master, it will check the data parity provided during read data cycles, and watch for the assertion of PERR# during write data cycles. See the earlier tables for exact details.

6.11.2.6 Other Violations

The PCI specification identifies numerous cases that are violations of the PCI protocol. Other than the cases identified above, the PXB makes no attempt to check for such violations. Response to such violations is undefined. Refer to the PCI specification for a complete description of the required PCI protocol.

6.11.3 PXB as Target

6.11.3.1 Target Disconnect

The PXB will issue a target disconnect under the following circumstances:

- After the first data transfer if the transaction is using an unrecognized addressing mode (the PXB will only support linear incrementing as a target),
- On reads, when no more data is available in the read buffers, and
- On writes, when the write crosses a 4 KB boundary.

These conditions are not treated as an error, and will not be logged or reported.

6.11.3.2 Target Retry

The PXB will issue a target retry when:

- A read request is to an address that has already been accepted as a delayed transaction (i.e. the request is already being serviced, but data has not arrived),
- A write request has insufficient buffering in the PXB to allow it to be posted (a full line is not available for an MWI), or
- The PCI interface is LOCKED from the host side.

6.11.3.3 Target Abort

The PXB will issue a target abort if a hard fail response is returned over the Expander bus. This response is limited to inbound read requests.

6.11.3.4 Data Parity Errors

When the PXB is a PCI bus target, it will check the data parity provided during write data cycles. For exact details on data errors, see the earlier tables.

6.11.3.5 Other Violations

The PCI specification identifies numerous cases that are violations of the PCI protocol. Other than the cases identified above, the PXB makes no attempt to check for such violations. Response to such violations is undefined. This includes, but is not limited to:

- MWI to a misaligned (non-cache-line-boundary) address.
- MWI to an aligned address, but with one or more byte enables not asserted.

Refer to the PCI specification for a complete description of the required PCI protocol.

Note: When multiple errors which cause an SERR# assertion occur within a few cycles of each other, there may not be a separate SERR# assertion for each error.

6.11.4 GXB Error Flow

Figure 6-3 shows a block diagram of the GXB. On the PCI side, the first error is latched in FERR_PCI. Subsequent errors are latched into NERR_PCI. Errors from PCI will cause the PCI address and data to be latched in PAC_ERR and PD_ERR.

The internal buffers of the GXB are parity protected. Data coming from the graphics card is placed in an inbound buffer with parity generated as for the Expander bus (1 parity bit over 16 bits of data and 2 byte enables). At the top of the inbound queue, the data, BE and parity is sent to the Expander bus as-is out of the queue. There is no checking done in the GXB. If there is bad parity at the SAC, the SAC will flag the error.

6.11.4.1 GXB Error Signals

The GXB has 2 error signals: XBINIT# and XINTR#.

6.11.4.1.1 GXB_XBINIT#

XBINIT# is used to signal a fatal error. All header errors are fatal, since the GXB and SAC are out of sync with each other at that point. Data parity errors may be considered fatal in some systems. For graphics, the error may be in a texture or some field that is a transient screen blip. The OEM may configure the system to BINIT# on data parity errors or not.

XBINIT# is held active until XRST# is asserted. On XRST#, GXBCTL[7] (XBINIT# enable) is reset and thus XBINIT# is deasserted. After reset, firmware will log and clear any pending error registers. It will then enable GXBCTL[7] so that future errors will be reported. Pending bits in the FERR or NERR registers that were not cleared by firmware will not cause a new BINIT# when GXBCTL[7] is enabled. Only errors that occur after GXBCTL[7] is enabled will cause the XBINIT# pin to be asserted.

6.11.4.1.2 XINTR#

XINTR# is used to signal non-fatal errors on the bus. The SERR# signal on the AGP bus will be OR'ed with internal GXB signals to create XINTR#. The system designer may take this signal and use it to force an interrupt, handle it like an NMI or SERR#, or even choose to ignore it.

XINTR# is held asserted until all internal conditions which cause INT# are cleared. So software must handle all errors and then do an EOI to the interrupt controller.

6.11.4.2 GXB Errors

The GXB will flag the following errors. See Table 6-1 for the behavior of each error.

6.11.4.2.1 PCI Interface Errors

- SERR# Observed - Set when the GXB sees SERR# that was asserted by the graphics card. This is not set if the GXB drove SERR#.
- PERR# Observed - Set when the GXB sees PERR# that was asserted by the graphics card. This is not set if the GXB drove PERR#.
- Discard Timer Expiration - Set when the 2^{15} timer expires. The timer starts when the data for a delayed read is returned to the GXB. If the card doesn't re-access the data in 2^{15} clocks, then an error is flagged.
- Non-Configuration Master Abort - Sets bits in both PCISTS and in FERR_PCI. This error occurs when the GXB is the master for a PCI transaction and there is a master abort (the card doesn't assert DEVSEL#). This only occurs on non-configuration cycles, since a master abort is not considered an error on configuration cycles. This error will not cause an interrupt or BINIT#. Reads will return 1's for DW accesses, or a Hard Fail (HF) for reads that are greater than a DW. Writes of 1 DW will return a normal write-complete. Writes of more than 1 DW will return a HF on the write-complete. If there is a master abort on the write then the rest of the data for that transaction is dropped. Writes using the Fast-write protocol may generate a master abort.
- PCISTS Error Logged - Set when any error bit, except RMA (bit 13), in PCISTS is set. This includes bits 15, 12 or 8. Setting this bit in FERR_PCI does not cause an interrupt or BINIT#.

6.11.4.2.2 GART Interface Errors

- GART Parity Error - There is one parity bit covering each GART entry. When the GART is accessed, parity is checked. If an error occurs, then this bit is set. Parity errors are only reported when the access falls within the GART aperture range. This prevents errors being reported when the GART entry was not used.
- GART Entry Invalid - Each GART entry has a valid bit associated with it. If the GART entry associated with an AGP address is not marked as valid, then this bit is set. This error is only reported when the access falls within the GART aperture range. This prevents errors being reported when the GART entry was not used.
- Illegal Address - After translation is done, the address is checked. If it is in the range between GAPBAS and GAPTOP, or in the VGA range with VGAGE asserted, or directed by the MARG's to PCI instead of memory, or above TOM; then the access is illegal and considered a fault.
- Illegal Outbound GART Access - Any programming access, read or write, to the GART that is not 4B or is not aligned on a 4B boundary. The behavior is undefined for the error.

6.11.4.2.3 AGP Interface Errors

- Use of Pipe with Sideband Enabled - The card must not mix sideband and Pipe requests. If sideband is enabled and the card attempts an access with PIPE, this error occurs.
- AGP Address[63:40] not Equal to Zero - This address is outside the legal graphics aperture and is over the TOM, therefore is an error.
- AGP Request Queue Overflow - The GXB supports 16 AGP transactions. If the card attempts to do a new transaction when there are 16 already outstanding, this error is flagged.
  Illegal AGP Command - Set whenever the GXB receives an unknown or undefined command from the graphics card.

6.11.4.2.4 Data Errors

- AGP Hi-priority Write Que Data Parity Error - AGP write data was placed in the que with good parity. If this error is set, then the write que itself was corrupted. The GXB will not report this error with either interrupt or BINIT#. The error is actually reported by the SAC or allowed to continue to memory where it will be poisoned.
- AGP Low-priority Write Que Data Parity Error - AGP write data was placed in the que with good parity. If this error is set, then the write que itself was corrupted. The GXB will not report this error with either interrupt or BINIT#. The error is actually reported by the SAC or allowed to continue to memory where it will be poisoned.
- PCI Inbound Write Que Data Parity Error - PCI write data was placed in the que with parity as received from the PCI bus. This error may have occurred because of bad parity on the PCI bus or because the que itself was corrupted. The GXB will not report this error with either interrupt or BINIT#. The error is actually reported by the SAC or allowed to continue to memory where it will be poisoned.

Note: On data transfers from the card to the GXB, there may be wait states. If there is a parity error on the AD bus when there is a wait state, then the GXB will not flag any error. It only flags an error when data is actually captured by the GXB's PCI interface.

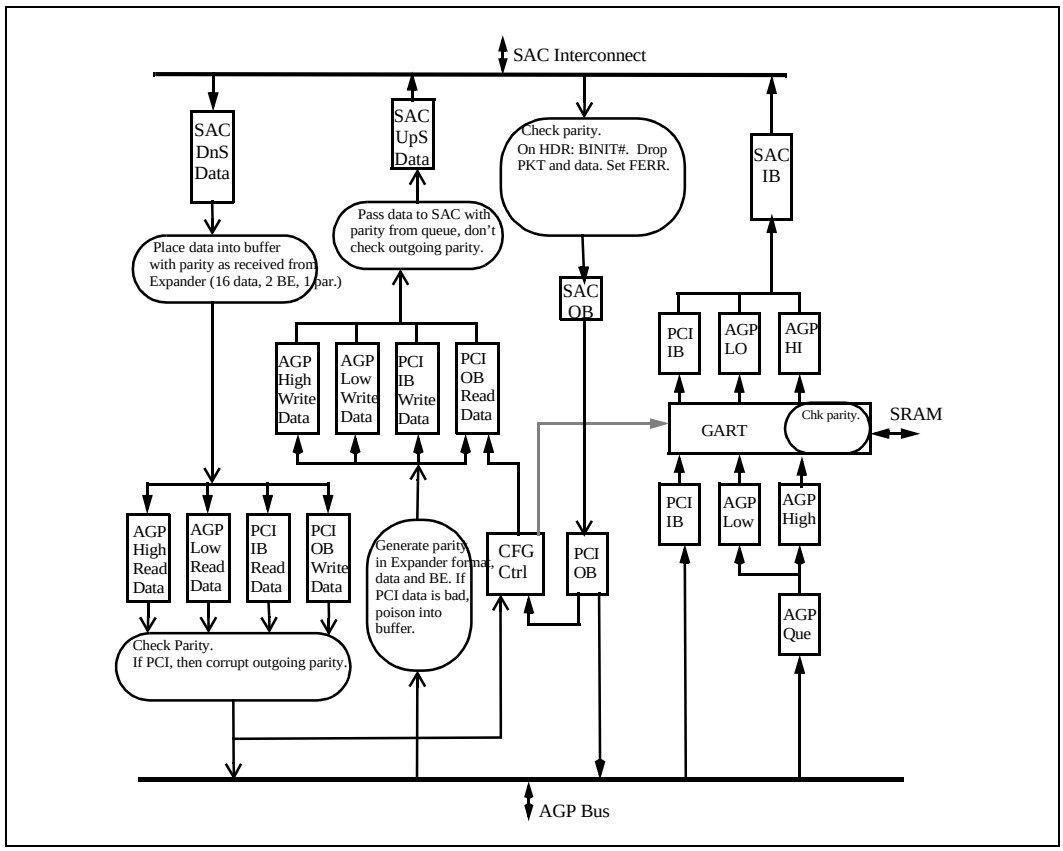
- PCI Outbound Read Que Data Parity Error - PCI read data was placed in the que with parity as received from the PCI bus. The GXB will not report this error with either interrupt or BINIT#. The error is actually reported by the SAC or allowed to continue to memory where it will be poisoned.

- PCI Outbound Write Que Data Parity Error - This error signifies that either a) data was received from the Expander bus with bad parity or b) the OB write Que was corrupted. As data is read from the queue and passed to the PCI bus, the parity is checked.
- AGP Low-priority Read Data Que Parity Error - This error signifies that either a) data was received from the Expander bus with bad parity or b) the read Que was corrupted. As data is read from the queue and passed to the AGP bus, the parity is checked.
- AGP Hi-priority Read Data Que Parity Error - This error signifies that either a) data was received from the Expander bus with bad parity or b) the read Que was corrupted. As data is read from the queue and passed to the AGP bus, the parity is checked.
- PCI Inbound Read Que Data Parity Error - This error signifies that either a) data was received from the Expander bus with bad parity or b) the read Que was corrupted. As data is read from the queue and passed to the AGP bus, the parity is checked. This is for PCI reads done by the graphics card. The parity on the PCI data will be poisoned out to the card as the data is returned.

6.11.4.3 Multiple Errors

In the case that 2 or more errors occur at the same cycle, multiple bits are set in the FERR register. This should be an extremely rare case. Software can read the register and check that only one bit is set. The data that is captured along with the error is indeterminate. Since there are multiple registers for error data, 2 errors may cause the Expander and PCI error registers to have valid data. Or, if there were 2 errors from Expander bus at the same time, then the Expander data-error register may have been set by either error.

Figure 6-3. GXB Error Flow



6.12 WXB Data Integrity and Error Handling

6.12.1 Integrity

Error handling in the context of a chipset component requires observing the error, containing it, notifying the system, and recovery or system restart. Different platforms have different requirements for error handling. A server is generally most interested in containment. It attempts to stop bad data before it reaches the network or the disk. On the other hand, workstations with graphics may be less interested in containment. If the screen blips for one frame and the blip is gone in the next frame, the error is transient and may not even be noticed.

The WXB accommodates both philosophies by allowing certain errors to be masked off completely, or by turning them into simple interrupts instead of signalling SERR# or XBINIT#. Certain errors can only be directed to cause an XBINIT#.

In general the WXB will report errors upon their observation. In the case of data parity errors on data transiting the WXB, however, the reporting can be deferred until the data is actually used. In the case of data heading inbound to memory, this could result in deferring the reporting of the error until it is read from memory or completely if the data is never used.

6.12.2 Data Parity Poisoning

When data is received by the WXB that is in error, it will be passed on to the next interface with bad (or “poisoned”) parity. Data received over the PCI bus that has bad parity will always be sent on to the chipset core with bad parity.

Note: There is no mode in the WXB to forward good parity if the data was received as bad. If the data comes in with bad parity it is always sent out with bad parity.

Note: When addressed to the IHPC, outbound writes containing data with bad parity will be “dropped on the floor.”

6.12.3 Usage of First Error and Next Error Registers

When an error is identified, and it is the first occurrence of an error, it is latched into the “First Error” register (see FEPCI register). If an error has been previously identified (and not cleared), the associated First Error bits will already be set. In such a case, subsequent errors of any type will be recorded through “Next Error” registers.

Since the system really only needs to know if just one error has occurred or many, setting a First Error bit does not affect the Next Error bits. If there is another error of any type, including a second occurrence of the first error, then the appropriate Next Error bit is set. Software can read both First Error and Next Error registers to determine a complete error status.

In general, as much information as possible is logged for the first error encountered; data, address, and/or command information are captured if available. This allows better isolation of errors and possible recovery.

Note: Multiple errors occurring in (nearly) the same cycle may result in multiple bits being set in the First Error register.

Note: Additionally, error responses such as SERR#, XBINIT# and INTRQ# are predicated on both First Error and Next Error contents since a second error may occur while the first error is in the process of being serviced by the WXB hardware. Thus, for example, the system may observe an interrupt associated with an error at almost the same instant XBINIT# is asserted for another.

For ease of isolation, the ERRSTS register records which of these registers have recorded an error.

6.12.4 Error Mask Bits

Many of the errors that the WXB handles have conditional reporting behavior. The error always sets the appropriate bit(s) within the First Error or Next Error registers. For a subset of errors additional information is also recorded in the PCI required PCISTS register. If the error is masked, then an XBINIT# (or other specified action) will not occur, but the status bit indicating the occurrence of the error is always set. This allows software to poll if it chooses, looking for errors that are not considered as fatal to the system. The error logging information (associated only with First Error) will always represent the first error identified since the register was last cleared regardless of the masking of the particular error.

Table 6-2. List of WXB Error Sources Selectively Routable to XBINIT#, SERR_OUT#, and P(A/B)INTRQ#

Abbreviation	Error
APE	PCI address parity error
CUIQ	WXB RAM-generated data parity errors in transactions directed at WXB configuration space
DTE	Inbound read discard timer expiration event
HPIQ	WXB RAM-generated data parity errors in transactions destined to the IHPC
HPSERR	IHPC-generated SERR
OPERR	WXB-observed PERR# assertion
OSERR	WXB-observed SERR# assertion
PCIDPE	PCI bus data parity error
PUIQ	WXB RAM-generated data parity errors in transactions destined to the PCI bus

6.12.5 Error Steering/Signaling

All observed errors are recorded through status bits. In most cases these errors can be (optionally) caused to signal the system that the error has occurred. Methods for signaling an error include SERR#, XBINIT# and a P(A/B)INTRQ[1] interrupt. In the case of data parity errors, the minimal response is to record the error and to simply forward the data on across the next interface with bad parity. Table 6-2 lists the WXB error sources that are routable to one or more of the WXB error signaling outputs. The error routing to SERR_OUT#, XBINIT#, and P(A/B)INTRQ as a function of both ERRCMD and PCICMD register settings is presented in Table 6-3, Table 6-4, and Table 6-5. Refer to the register definitions in Chapter 8 for more detail.

Table 6-3. Supported Error Escalation to XBINIT#[a]

XBINIT0 ERRCMD(15)	XBINIT# Escalation
0	CUDPE, CUIQ, HPIQ, HP, IT, HLM, and MXBLK
1	OFF

a. To obtain the listed escalation, the following settings are required: XBINITE='0''.

1. See Section 6.12.6 for a description of the P(A/B)INTRQ interrupt.

Table 6-4. Supported Error Escalation to SERR_OUT#[a]

ASDPE ERRCMD(11)	ASFPE ERRCMD(9)	SERRE PCICMD(8)	PERRE PCICMD(6)	SERR_OUT# Escalation
X	X	0	X	HPSERR, OSERR
0	0	1	0	HPSERR, OSERR, DTE
0	0	1	1	HPSERR, OSERR, DTE, APE
0	1	1	0	HPSERR, OSERR, DTE, OPERR, HPPERR, PUIQ
0	1	1	1	HPSERR, OSERR, DTE, APE, OPERR, HPPERR, PUIQ
1	0	1	0	HPSERR, OSERR, DTE, FUIQ, PCIDPE
1	0	1	1	HPSERR, OSERR, DTE, APE FUIQ, PCIDPE
1	1	1	0	HPSERR, OSERR, DTE, FUIQ, PCIDPE, OPERR, HPPERR, PUIQ
1	1	1	1	HPSERR, OSERR, DTE, APE FUIQ, PCIDPE, OPERR, HPPERR, PUIQ

a. To obtain the listed escalation, the following settings are required: ASAPE='1', and ASDTE='1'.

Table 6-5. Supported Error Escalation to P(A/B)INTRQ#

IRQE ERRCMD(13)	P(A/B)INTRQ#
0	OFF
1	OPERR, HPPERR, PUIQ

6.12.5.1 SERR# Generation

Most errors can be caused to steer the observation of the error to the signaling of an SERR#. The system then has a chance to respond to the event while it continues to run. Often, SERR# results in an NMI which itself often results in a system hang. If the WXB is configured to cause an SERR# the system should be configured with a usable NMI handler or wired in such a way that an SERR# is not escalated into an NMI.

Alternatively, the WXB can be configured to cause an INTRQ# interrupt or an XBINIT# instead of (or in addition to) an SERR#. This is achieved by having any unmasked errors (errors configured such that they might cause an SERR#) directed to cause the specified event, be it an INTRQ# or an XBINIT#.

SERR# signaling itself is ultimately controlled by appropriately setting or clearing the SERRE bit in the PCICMD register. If set then unmasked errors will result in SERR# being signaled. Otherwise, even unmasked errors will not cause an SERR#.

Whenever the WXB actually signals an SERR# the SSE bit in the PCISTS register will be set. If another agent has caused an SERR# then the SES bit in the FEPCI (or NEPCI) register will be set.

Note: When multiple errors, which cause an SERR# assertion, occur within a few cycles of each other, there may not be a separate SERR# assertion for each.

6.12.5.2 XBINIT# Generation

A certain subset of errors within the WXB will always result in the WXB attempting to signal an XBINIT#. Whenever an error occurs that forces an XBINIT#, an internal "override" bit is set as XBINIT# is driven to the bus. While the override bit is set, XBINIT# will no longer be reasserted, even for qualifying errors. This prevents the presence of a single error from causing XBINIT# to be signaled multiple times. Software may also set the override bit and thus prevent XBINIT# from occurring at all. When an XBINIT# is signaled, software should (1) clean up the error, (2) record and clear the error status registers, and (only) then, (3) clear the XBINIT# override bit so that new errors may be signaled. The override bit is always set on a Power Good reset.

Anytime an XBINIT# is signaled by the WXB, the "XBINIT Asserted" bit in the ERRSTS register will be set. Clearing this bit will cause the deassertion of the XBINIT# pin.

Errors that can be caused to signal an SERR# can also be caused to signal an XBINIT# (refer to the "XBINIT Escalation" bit within the ERRCMD register in Section 2).

6.12.6 INTRQ# Interrupt

The WXB has an internal mechanism to cause a level interrupt under various error conditions. The signaling of this INTRQ interrupt is directly visible through the INTRQ# pin.

An INTRQ interrupt would generally be an expected outcome anytime that a "less serious" error has occurred. The INTRQ# signal is held asserted as long as the record of the error persists. All the errors that cause such an event are wired together to drive the internal signal. Software is expected to reset the bit causing the interrupt in the First Error or Next Error register where the error is recorded. In addition, software is expected to then clear the "INTRQ Asserted" bit in the ERRSTS register. When this bit is reset, INTRQ# will be deasserted unless there is another error which holds the signal active.

It is possible to configure the WXB to cause all errors to result in an INTRQ# interrupt[1]. In this way there is some flexibility to have a soft response (i.e. interrupt) for all errors as well as a more harsh response for specific "more serious" errors.

6.12.7 Error Determination and Logging

The various error status registers identify which error(s) have occurred. In certain cases key information is logged in conjunction with the error status being set. For example, parity errors log the data and parity for the failing transfer if it is the first error occurrence. Other errors capture the address associated with the particular failure. This information can be used for debug and diagnostic purposes and ***may*** be used in system recovery. For instance, if there is an parity error on a data read, and the access can be isolated, instead of restarting the whole system it might be possible to kill only the failing process and allow other users to continue running.

The error log registers are updated at virtually the same time that the associated bit is set in the status register.

1. The single exception to this rule is Hard Fail Completion which will not initiate any sideband error signal (INTRQ#, SERR# or XBINIT#). However, an in-band PCI Target Abort will occur as a result of a Hard Fail Completion.

6.12.8 Error Conditions

6.12.8.1 WXB as Bus Master

6.12.8.1.1 Master Abort

If the WXB initiates a PCI transaction and no target responds, the WXB will terminate the transaction with a master-abort. The WXB will wait five PCI clocks after asserting FRAME# for a target to respond with DEVSEL#. If no target responds, the WXB will perform a master abort to terminate the cycle on the PCI bus. Special Cycle commands, which are broadcast to all PCI targets, will always be terminated with master abort and, therefore, are ***never*** considered an error. Master aborts during other transactions (configuration, memory map, or I/O cycles) also are generally not considered as errors. For writes, a normal completion packet will be returned to the Chipset core. Reads will additionally return all 1's as data.

When the WXB performs a master abort, it will log the event by setting the PCISTS register's Received Master Abort (RMA) bit, unless the transaction was a Special Cycle.

6.12.8.1.2 Received Target Disconnect

A PCI target may issue a disconnect to indicate it is unable to respond within the PCI latency guidelines. Disconnect is signaled when the target asserts both STOP# and DEVSEL#. The target controls whether another data transfer may occur by whether TRDY# is asserted when STOP# is asserted. A target disconnect is not usually issued on the first data phase of the transaction. Target disconnects are not considered errors, and are not logged or reported in any way. After a target disconnect, the WXB will deassert its request signal and wait at least two PCI clocks before re-arbitrating for the PCI bus to complete the transfer.

6.12.8.1.3 Received Target Retry

A PCI target may issue a retry to indicate that it is currently unable to process the transaction. Retry is signaled when the target asserts STOP# and DEVSEL# and does not assert TRDY#. Retry is actually a special case of disconnect that occurs before the first data transfer.

After receiving a retry for a transaction, the WXB will deassert its request line and wait at least two PCI clocks before re-arbitrating for the PCI bus to retry the transaction. If the transaction is a write, the WXB will retry the transaction until it succeeds. If the transaction is a read, the WXB will reattempt the transaction until it succeeds, but may allow other reads and writes to pass it. Note, in all of these cases the retries are not considered errors. There is no logging or error reporting of any kind.

6.12.8.1.4 Received Target Abort

A PCI target may issue an abort to indicate that the current transaction should be terminated *and should not be attempted again*. This is a catastrophic failure. Target abort is signaled when STOP# is asserted and DEVSEL# is deasserted. The WXB will log the target abort by setting the PCISTS register's Received Target Abort (RTA) bit. The WXB then returns a hard fail response to the Chipset core.

6.12.8.1.5 Data Parity Errors

When the WXB is the PCI bus master, it will check the data parity provided during read data cycles, and watch for the assertion of PERR# during write data cycles. If a parity error is detected, and the PCICMD registers PERRE bit is set, then the PCISTS register's Detected Parity Error

(DPE) bit is asserted. Regardless, if the transaction is a read, the PCISTS register's Parity Error (PE) bit will be set. Additionally, address, command, and data related information is logged in the FEPCIAL and FEPCIL registers if the error is the first error observed by the WXB.

6.12.8.1.6 Other Violations

The PCI specification identifies numerous cases that are violations of the PCI protocol. Other than the cases identified above, the WXB makes no attempt to check for such violations. Response to such violations is undefined. Refer to the PCI specification for a complete description of the required PCI protocol.

6.12.8.2 WXB as Target

6.12.8.2.1 Illegal PCI Request Type

The WXB will not claim transactions that use an illegal or unrecognized request type.

6.12.8.2.2 Target Disconnect

The WXB will issue a target disconnect under the following circumstances:

- After the first data transfer if the transaction is using an unrecognized addressing mode (the WXB will only support linear incrementing as a target),
- On reads, when no more data is available in the read buffers, and

These conditions are not treated as an error, and will not be logged or reported.

6.12.8.2.3 Target Retry

The WXB will issue a target retry when:

- A read request is to an address that has already been accepted as a delayed transaction (i.e. the request is already being serviced, but data has not arrived).
- A read request to this address has not yet been accepted by the WXB as a delayed transaction, there is room to enqueue a new delayed transaction, and the request is enqueued.
- A read request is to an address that has ***not yet*** been accepted by the WXB and there is no more room to enqueue a new delayed transaction.
- A write request has insufficient buffering in the WXB to allow it to be posted (e.g. a full line is not available for MWI).
- The PCI interface is LOCKED from the host side, unless the transaction is a read request and the data has already been fetched by the WXB.

6.12.8.2.4 Target Abort

The WXB will issue a target abort if a hard fail response is returned over the Expander bus. This response is limited to inbound read requests.

6.12.8.2.5 Other Violations

The PCI specification identifies numerous cases that are violations of the PCI protocol. Other than the cases identified above, the WXB makes no attempt to check for such violations. Response to such violations is undefined. This includes, but is not limited to:

- MWI to a misaligned (non-cache-line-boundary) address
- MWI to an aligned address, but with one or more byte enables not asserted

Refer to the PCI specification for a complete description of the required PCI protocol.

6.12.8.3 PCI Interface Errors

Other PCI interface errors that are handled by the WXB are:

- **System Error Signaled**

  Set within the FEPCI register when the WXB sees an SERR# asserted by another PCI agent. This is not set when the WXB drives SERR#.

- **Discard Timer Expiration**

  Set when the 2^{15} timer expires. The timer starts approximately when the data for a delayed read is requested by the WXB. If the card doesn't re-access the data in 2^{15} PCI clocks, then an error is flagged.

AGP Subsystem 7

AGP is a new port defined for graphics adapters. In the initial implementation it is a 500 MB/s port. There is also an extension called AGP 4X mode, which has a bandwidth of 1 GB/s. AGP 2X mode cards will work in an AGP 4X mode slot. The 460GX chipset is designed to work at the AGP 4X mode bandwidths. It will support 3.3V AGP 1X and 2X mode cards as well.

AGP 4X mode is a high-bandwidth port targeted to workstation graphics needs. It provides 1 GB of data bandwidth. There are several protocol modes. One mode is 66 MHz PCI. The other mode is the new AGP protocol, which comes in two flavors. The first is an extension of PCI using PIPE# instead of FRAME#. The second uses sideband signals to improve bandwidth. Only one AGP protocol may be in effect at a time. Both of the modes using AGP protocol and the mode using PCI protocol are fully supported.

This chapter explains the 460GX chipset implementation of AGP and related issues. It assumes that the reader is familiar with the AGP and AGP 4X mode specifications. This chapter explains the GX implementation of the AGP interface to the GXB and the interface between the GXB and the rest of the chipset, which is the expansion bridge chip containing the AGP port. It will not explain AGP protocol or the AGP bus itself.

7.1 Graphics Address Relocation Table (GART)

Graphics cards, especially for texture traffic, would like to see a large contiguous view of memory. This allows them to make random accesses into a large area which contains the entire texture map for the displayed image. Processors have the same view of a flat contiguous memory range in their virtual space, where a program may have many megabytes of contiguous address space to use. The memory pointed to by those virtual addresses will most likely be discontiguous in main memory. The processor, through page tables and TLBs, remaps the contiguous virtual addresses to the discontiguous physical DRAM addresses. For graphics cards, the chipset will perform the same translation. The graphics card uses, in essence, a virtual address. This address is translated by the chipset's Graphics Address Relocation Table (GART) from the virtual address used by the graphics card to the physical DRAM address the memory controller understands.

The GART table is loaded by the graphics device driver at initialization time. The table resides in the GXB, so that translations will be fast and not interfere with main memory traffic. The table will contain a new page address for each address coming in from the graphics card. It will also contain a bit to determine whether addresses to that page need to be snooped on the system bus or not. The following figures show how the GART operates. There is a base address for AGP addresses and a length for the AGP range. Addresses are received from the graphics card. If the address is in the AGP range, the page number within that range is used as a pointer into the GART and a new page address is combined with the 12 (22 for 4 MB pages) least significant bits of the address from the graphics card. This new address is used by the memory system. If the address doesn't hit inside the AGP-space, then no translation is done. Note that the AGP_BASE address bits 27:12 are always a 0 (hardwired in config register).

Figure 7-1. GART Table Usage for 4k Pages

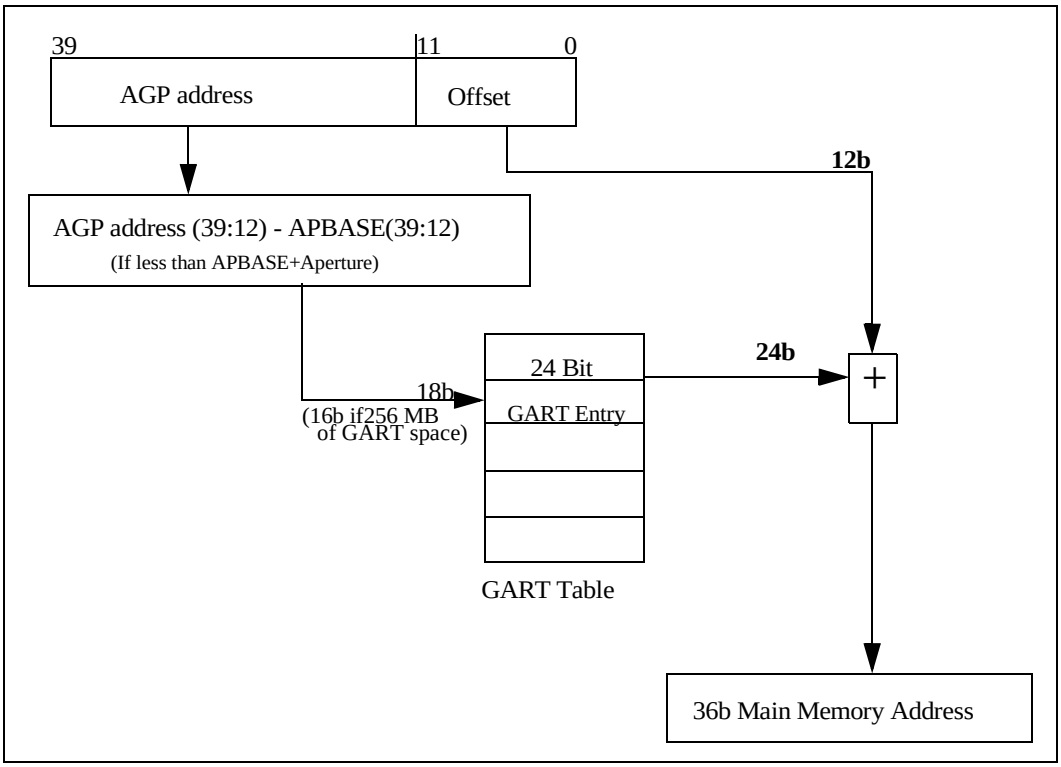
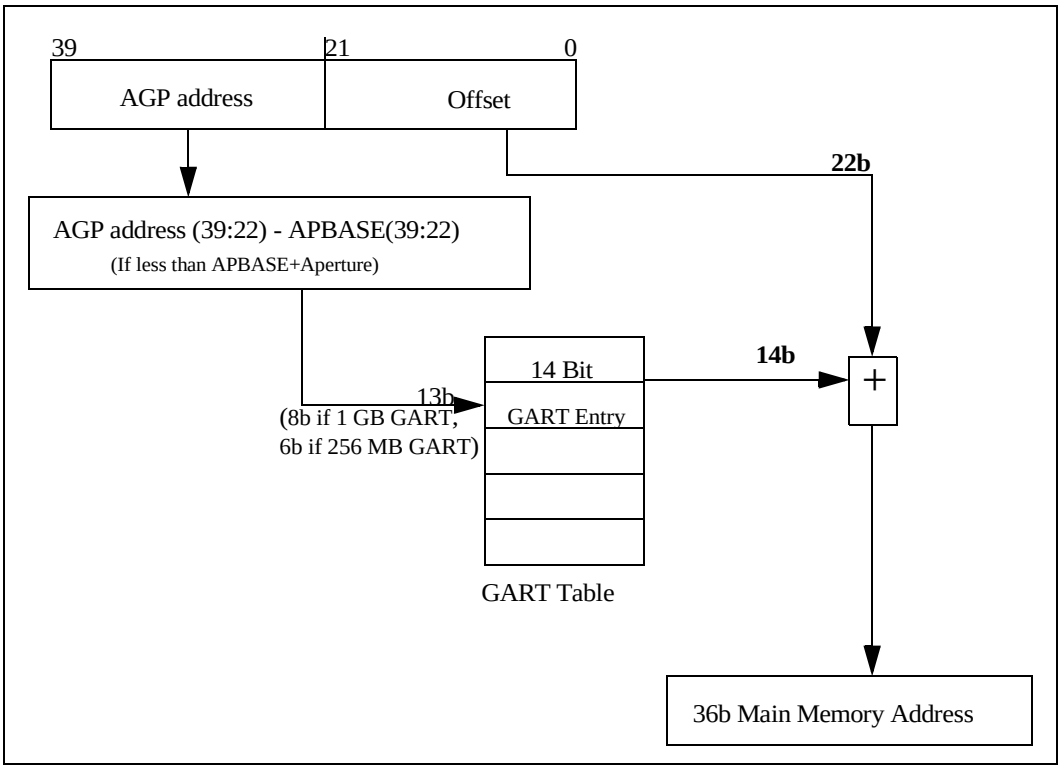


Figure 7-2. GART Table Usage for 4 MB Pages



7.1.1 GART Implementation

The GX implementation will support 256 MB, 1 GB, or 32 GB (32 GB requires 4 MB pages by the O.S.) of translation space. This limit is implementation-based not architectural. Each entry in the GART requires 24 bits for the address, 1 bit for coherency, 1 parity bit and 1 valid bit for a total of 27 bits. The width of each GART entry will be 32 bits. Each entry covers one 4kB (or 4 MB) page. One megabyte is sufficient to cover 256k entries or a total of 1 GB of translatable space for 4kB pages. Figure 7-3 and Figure 7-4 show the format of a GART entry. Accesses which hit a GART entry that does not have the valid bit set are treated as GART misses and the address is passed on untranslated. Optionally, accesses that hit an entry with the valid bit marked invalid can cause an error.

Figure 7-3. GART Entry Format for 4kB Pages

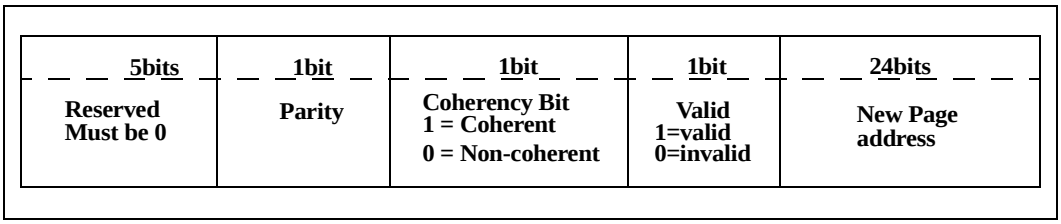
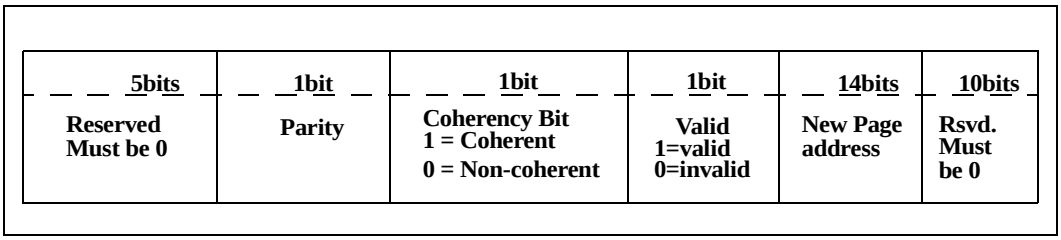


Figure 7-4. GART Entry Format for 4 MB Pages



The table itself will be kept in SRAM attached to the GXB. It is possible that the table could be in main memory and the GXB could fetch the translation from memory when it needs a new entry. Having the GART in local SRAM allows the translations to be done much faster and is a simpler implementation, since the interface is local and accesses to fetch GART entries don't compete with other traffic. A fetch from local SRAM should take 30 ns. or less instead of the roughly 300-500 ns. required to get to DRAM from the GXB.

Both 1 MB and 4 MB SRAMs will be used. To get the 1 MB required, 2 of these parts are required. For systems that don't require the full 1 GB translation space, the SRAM scales with the required space. If 256 MB of translated graphics area were sufficient, then the system could simply populate 256k of SRAM. The GX will support the following sizes of AGP space and SRAM sizes.

- 1 MB of SRAM - 1 GB of AGP space for 4kB pages, requires two 4 MB SRAMs; using 4 MB pages, the AGP space is still limited to 32 GB.
- 256kB of SRAM - 256 MB of AGP space for 4kB pages, requires two 1MB SRAMs; using 4 MB pages, the AGP space is still limited to 32 GB.

There is no requirement that there be any GART table. If the graphics card doesn't require translation, then there is no need of a GART and the SRAM need not be present at all. The GXB will fully work when there is no SRAM. Of course, in this case, there is no translation so the AGP card must put out the physical memory address itself.

7.1.1.1 Page Sizes

The Itanium processor supports both a 4kB and a 4 MB page size. The AGP programming model is designed using 4kB pages for GART entries. Using the larger page size would greatly reduce the number of misses and reduce the number of entries needed in the GART, thus lowering system cost.

The GXB will support 4 MB pages. Using 4 MB pages, 256kB of SRAM could provide 256 GB of AGP space. The 460GX chipset will not support more than 32 GB of translatable graphics space.

7.1.1.2 GTLB

The GXB does not implement any GTLB or hold any old translations. Each address received by the GXB will be translated through the SRAM.

7.1.1.3 Parity

There is one parity bit which covers each GART entry. The parity covers the entire GART entry, including reserved bits. The parity bit is generated by hardware. Software does not need to compute parity for the GART entry. Software should mask out the parity bit and treat it as a reserved bit on GART reads. The parity bit is returned on a read, so that it can be read by diagnostics. Writing a zero or a one to the parity bit has no affect since the hardware will compute correct parity and write that into the SRAM, which may cause the bit to be read as different from what was written. Parity will be done such that the total number of 1’s, including the parity bit, total to an even number.

7.1.2 Programming GART

The addresses to load or read the GART itself lie in the chipset specific area below 4 GB. There is a 12 MB area for the chipset to use. The GART will exist in this range.

The GART lies at address range FE20_0000h to FE3F_FFFFh. GART entries may be read or written from the processor. Accesses by the processor must be 4 byte accesses on an aligned 4B boundary. Any other type of access is considered an error and the system will fault. For the Dual-Expander bus GXB which attaches to Expander ports 2 and 3, it will be programmed through port 2 addresses only.

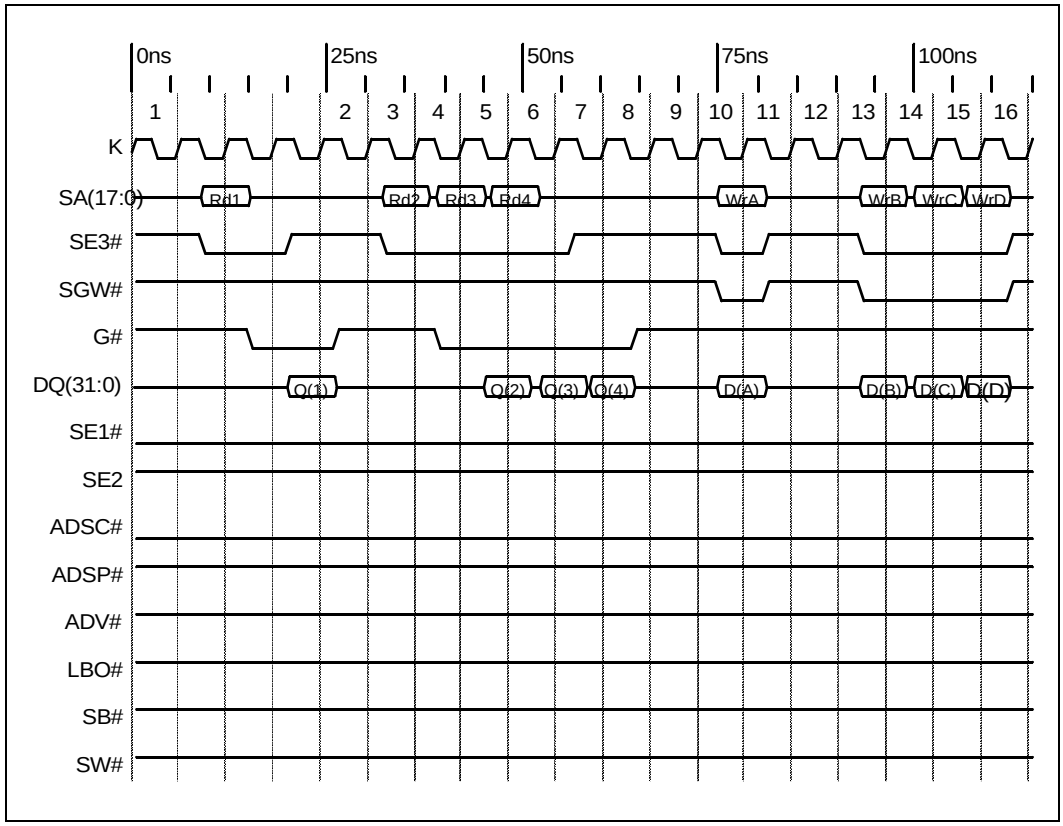
To the GXB, address bits 22 and 21 are don’t cares and shouldn’t be used in the decode for any access at FEzx_xxxxh, where z is 0xx0 (binary). Address bit 23 must be a 0 for the address to point to the GART. Address bit 20 must be a 0 also. The range FE30_0000h to FE3F_FFFFh is not decoded by the GXB as a GART access, but is present in the address map for future growth.

When programming the GART, software must read the last location written, to guarantee that all writes have taken affect. All writes to the GART table are done in order, so only the last write address needs to be read. This ensures the GART table is updated in SRAM before allowing the graphics cards to try and access translatable space. The processor must map the area from (4 GB-32 MB) to (4 GB-20 MB) as UC.

7.1.3 GART Implementation

Figure 7-5 shows the timings for the SRAM interface. Synchronous SRAM will be clocked at 7.5 ns. The SRAM will be used in the pipelined mode. This allows addresses to be presented to the SRAM every cycle. The data is valid to be latched by the GXB on the 3rd clock edge after the clock edge which drove the address. Writes require the data to be driven with the address. The entire 32 bits of data are read or written at one time.

Figure 7-5. GART SRAM Timings



NOTES:
1. LBO# is a don't care, but must be tied either high or low.
2. SE1#, SE2, ADSC#, ADSP#, ADV#, LBO#, SB# and SW# are not connected to the GXB. These signals must be tied to the required level by the board.

7.1.4 Coherency

Traffic from the graphics card may or may not want to be coherent with the system bus. For the discussion here, coherency means that addresses will appear on the system bus so that the processors may snoop their caches for that address. If the texture map or other image data is marked WC by the processor, then that data is not coherent. Addresses on the bus which hit in a processor's WC buffer are not snooped and even if there is dirty data in the buffer, no implicit writeback is done. Or the application may know that the data in memory was not used by the processor (e.g. it came from disk) and wants the graphics card to fetch the data without using address bus bandwidth, so forces the access to be non-coherent.

For all AGP-type accesses which hit in the AGP range, there is a bit per GART entry which determines whether the address is coherent. For AGP-type accesses outside the AGP range, there is a bit in a configuration register of the GXB which determines the coherency. Coherency or non-coherency applies to accesses using AGP protocol only. Accesses using PCI protocol are always done coherently, whether they hit the translation table or not.

Accesses using AGP protocol which are non-coherent are truly not coherent. There is no ordering between them and processor traffic at all. The traffic goes directly to memory regardless of processor cache state or bus locks. However accesses within the AGP stream must follow the AGP ordering rules regardless of whether or not they are coherent. Coherent low-priority writes will push non-coherent low-priority writes, as the AGP spec calls for. Coherent traffic is not a separate stream from non-coherent traffic within the GXB.

7.1.4.1 3.3V AGP 1X and 2X Mode Compatibility

The GXB will be compatible with 3.3V AGP 1X and 2X mode cards. If they do accesses of less than 32 bytes, then their performance may not be adequate. As well, some AGP cards might expect lower latencies than the 460GX chipset will guarantee. These cards must not have a watchdog timer or other time-out mechanism which requires a latency lower than the 460GX chipset is able to guarantee.

7.1.5 Interrupt Handling

Before an interrupt is delivered to the processor, the system must ensure visibility or completion of all operations from that device which were generated before that interrupt. For PCI, device drivers are required to read a status register on the card that caused the interrupt. If the card had done a write before the interrupt, then the read completion would force the write to have been seen by the system.

AGP, with its hi-priority, low-priority and PCI streams, has no concept of ordering between the streams. Therefore reading an AGP cards register will not enforce pending hi or low-priority writes to have been completed. If the graphics card needs to force all writes to be visible to the system before the interrupt is visible, then the AGP card must issue a flush operation to guarantee that all writes are complete, wait for the return of the flush acknowledgment, and then issue the interrupt.

Even using Itanium processor's SAPIC interrupt protocol, whereby a write on the PCI bus becomes an interrupt to the processor, does not guarantee that writes from other streams have become visible to the system by the time the interrupt is received.

The interrupt signals on the AGP 4X mode interface will be routed to an I/O APIC chip (PID). The GXB will not receive these signals. The GXB will have an interrupt pin that it will drive for error reporting.

7.2 AGP Traffic

7.2.1 Addresses Used by the Graphics Card

AGP introduced the concept of a contiguous virtual address range that the graphics card could use. This address range lies outside the physical memory space. It also lies outside the range of addresses mapped to I/O. This range may exist in one of two places in the 460GX chipset system map. The virtual address is limited to 40 bits for the GXB, even when in 64-bit addressing mode.

The range may lie above the top of physical memory. Or the range may be placed in one of the gaps used to map addresses to PCI, and have that gap marked as reserved and not usable for addressing PCI devices.

In the first case, the virtual range used by the graphics card may be above or below the 4 GB boundary. If it is above, then it can be placed anywhere in the 40 bit address range supported by the GXB. This requires dual-address cycles (DAC) on PCI-type accesses or for AGP-type accesses which use the sideband signals or the DAC command with PIPE#. Having the virtual range above the top of memory means that s/w can allocate the full GART space per GXB at boot time and not worry about it again. The address must be translated by the GART to a 36-bit address, since only 36 physical bits are supported after the GART. In this case, there is a special BAR that is used by the firmware to know how much space to allocate for AGP. This special BAR is the BAPBASE register in the GXB and is exactly the same as the standard PCI-defined BAR, except that it is not in the PCI-defined Header space.

In the second case, one can map out the required region by using the standard BAR configuration register in the SAC. Firmware will read the BAR on the AGP card itself and give it the address space it needs. It will then read the BAR on the GXB to see how much AGP space is being mapped. It will then use the PCIS register for Device 14h to allocate this space. The PCIS registers for Devices 15h, 16h, and 17h will all be set to that of 14h, which in effect disables 15-17h.

The graphics card can issue addresses to the non-AGP space using AGP protocol. The addresses it puts out are the physical memory addresses, the same as PCI cards or the processor does today.

The physical memory used for graphics may change during run-time. One application may need only a few megabytes of AGP space and would only get that many bytes of physical space from the o.s. It would only program the few entries in the GART that it needed. A later application might ask for hundreds of MBs of physical memory and then map more entries in the GART. When that application finished it could release the physical memory back to the O.S. In all cases the virtual addresses used are allocated at boot/config time. They do not change during runtime, only the mapping of the physical memory they point to changes.

7.2.2 Traffic Priority

The AGP specification has both hi-priority and low-priority AGP transfers as well as PCI transfers. The GXB will do hi-priority requests ahead of any low-priority or PCI requests. The SAC will not prioritize any AGP traffic over other system traffic. So, except for the GXB, the chipset has no concept of traffic priority. Since the chipset's intent is to provide the best service to all agents, trying to give one agent priority does not fit well. Since memory is highly interleaved, having the GXB move hi-priority traffic ahead of low-priority traffic should be sufficient to provide the latency needed by the traffic.

Since the processor and PCI traffic must appear on the system bus, a new address may appear every 3 clocks from any coherent agent. The other 2 clocks are used for moving non-coherent traffic to the memory controller. Thus the processor can never hold off AGP traffic.

7.2.3 Coherency, Translation and Types of AGP Traffic

Table 7-1 shows the types of traffic versus address range and the resulting behavior. AGP-space means an address that hits in the translation window. DRAM-space is an address which is a physical DRAM location. PCI-space is an address that would normally be directed to a PXB and a PCI device. Both the DRAM-space and the PCI-space lie outside the AGP-space. These areas must not overlap. The destination of the request is determined by address registers in configuration space.

Note: Accesses from an AGP card that are directed to a PCI bus are a system fault and cause a BINIT# (system reboot). The 460GX chipset does NOT support any access originating from the AGP port to another PCI bus. This is true for PCI cycles (FRAME# active) as well as AGP cycles.

PCI accesses are always disconnected at a line boundary for writes and a read will at most prefetch to the end of the line. Therefore each PCI request will have only one translation.

An AGP request may be more than a line in length and therefore could cross a page-boundary. In that case the request may be broken into 2 parts, each with its own translation and targeted location (DRAM-space or AGP-space). Each part of the request is handled individually and then the data recombined before delivering back to the graphics device.

Table 7-1. Coherency for AGP/PCI Streams

Transaction Type		System Effect		
Traffic Type	**Address Range**	**Translate**	**Coherent**	**System Action**
PCI (Frame Active)	AGP-space	Yes	Always	Translated access to memory
	DRAM-space	No	Always	Untranslated access to memory
	PCI-space	No	Not applicable	Peer-to-peer PCI access (THIS IS ILLEGAL AND CAUSES SYSTEM BINIT#)
AGP (Pipe# or SB)	AGP-space	Yes	GART Sets.	Translated access to memory
	DRAM-space	No	Depends on Config bit	Untranslated access to memory
	PCI-space	No	Not applicable	Peer-to-peer PCI access (THIS IS ILLEGAL AND CAUSES SYSTEM BINIT#)

7.2.4 Ordering Rules

The GX will obey AGP ordering rules as spelled out in the AGP specification. The coherent stream will follow the ordering rules for the type of transaction it is. Low-priority coherent traffic is treated as being in the same stream as low-priority non-coherent traffic.

7.2.5 Processor Locks and AGP Traffic

AGP non-coherent traffic is not affected by locks on the system bus that target memory or other I/O devices. The accesses continue to memory independent of whether the LOCK# bit is set on the bus. However, since legacy code may issue a locked transaction to an AGP device, the GXB must ensure there is no deadlock in the presence of a lock targeting AGP (see Section 3.6.1 for more information on processor locks). The lock flow for a single Read-Modify-Write to the AGP bus is outlined below.

1. An outbound Locked Read request is transferred across the Expander bus.
2. When the locked read request reaches the head of the outbound PCI request queue, it is issued on the AGP bus with PCI semantics.
3. Once the locked read transaction is complete the read completion is placed in the inbound PCI transaction queue. At this point the PCI interface establishes a pseudo-lock and begin retrying all inbound transactions.
4. When the Read completion reaches the head of the PCI inbound PCI transaction queue, assuming the higher priority queues are empty, it is transferred across the Expander ring. At

this point the inbound Expander logic establishes a pseudo-lock and will no-longer send coherent requests from the AGP streams. Non-coherent requests can still be issued, but anything that can block the PCI stream in the SAC's queues must be held in the GXB.

5. The outbound Locked Write request is transferred across the Expander bus.
6. An Unlock transactions is transferred across the Expander bus (locked writes are posted; the Unlock will occur once the system bus lock goes away, so this will occur even if the data has not been delivered to its destination). The Unlock request, which throughout the Expander queues has the same semantics as a posted write, must be pushed into the outbound PCI queue behind the Locked Write.
7. When the Locked Write reaches the head of the outbound PCI queue it is performed on the AGP Bus with PCI semantics.
8. When the write completes on the AGP bus, a Write Completion is pushed into the inbound PCI transaction queue.
9. The Unlock transaction reaches the head of the outbound PCI queue.
10. A write completion for the Unlock transaction is placed in the inbound PCI transaction queue. At this point the PCI interface's pseudo-lock is released.
11. Both the completion for the locked write and for the Unlock transaction are delivered over the Expander bus. When the Expander logic transmits the completion for the Unlock transaction, its pseudo-lock is released.

7.2.6 Address Alignment and Transfer Sizes

The AGP specification allows the graphics card to request reads of size 8-64 bytes in eight-byte increments or of size 32 to 256 bytes in 32-byte increments. Reads and writes are both aligned on any 8-byte boundary. There is no concept of cache-line aligned or even page aligned. The request may cross a page boundary and require 2 translations. There is no requirement that the physical addresses resulting from the translation be contiguous. There is also no requirement that the coherency be the same on these two pages.

7.2.6.1 Address Faults

See the 'ERROR' chapter for the GXB behavior on addresses that are considered illegal.

The graphics card may do AGP transfers to areas outside the GART range. There is no translation on these addresses, so the graphics card must use the physical DRAM address for the object it is attempting to access. This is perfectly acceptable, but the graphics card and device driver must be able to provide the correct physical address to be used.

7.2.7 PCI Semantics Traffic

7.2.7.1 Inbound Reads

Delayed Transactions

The GXB supports the Delayed Transaction mechanism as defined in the PCI 2.2 Specification. The process of latching all information required to complete the transaction, terminating with Retry, and completing the request without holding the master in wait-states is called a Delayed Transaction. The GXB will delay all memory space read requests (unless a delayed slot is unavailable); no other request types are delayed.

Delayed transactions are issued and serviced as follows:

1. Upon receiving a read request, the address is compared against the GXB's internal buffers. Unless the data corresponding to this request is already available in the buffers (i.e. from a previously retried request), the read cycle is immediately retried (the GXB retries the read cycle in three PCI clocks from FRAME# driven active).
2. If there is an available Delayed Transaction Request buffer (there is only one per GXB), then the GXB will latch the pertinent information and forward the read request.
3. When the data is returned, the GXB will store it in its internal buffers, and flag the buffers as "available" for a subsequent read to that address (i.e. GOTO #1).

The GXB allows only one outstanding Delayed Transaction access. After accepting a Delayed Transaction, the GXB maintains system concurrency by continuing to accept and process additional writes as buffering allows. If the Delayed Transaction Request slot is full, the GXB will retry all reads that do not match the address and byte enables of the pending Delayed Transaction. For PCI Stream AGP to DRAM writes, the GXB will post the cycle unless the posted write slots are full (in which case the write is retried).

The GXB does not differentiate between read transactions from different masters. If a different master, other than the one that initiated the Delayed Transaction, attempts to read the same DRAM location as the Delayed Transaction, the GXB will respond with the data and complete the Delayed Transaction on the AGP bus.

7.2.7.2 PCI Stream Read Prefetching

The GXB delays all inbound reads. A Memory Read targeting memory will fetch 8B, unless the transaction begins 4B from the end of a cache line, in which case the transaction only fetches 4B. A Memory Read Line or Memory Read Multiple will always prefetch up to one cache line of data. When the read data is available in the GXB, the next matching read attempt from the controller is accepted and the data is streamed to the AGP bus until the controller disconnects or the prefetched data is consumed. Any data left when the controller disconnects is discarded. If the controller disconnects half way through a line, and comes back with the next address; then a new read is sent to DRAM for the data.

7.2.7.3 Inbound Reads Directed To Memory

All PCI reads-to-memory are propagated up through the SAC and placed on the system bus to allow snooping. The snoops are performed using the Memory Read (Length=0) transaction, which does not transfer data on the system bus. Instead, the data is transferred directly from the memory to the SAC, and finally back to the GXB. This path means that PCI Stream AGP-DRAM reads need consume no host data bus bandwidth (except in the case of implicit writebacks).

Note: The GXB as a PCI target supports only a linear incrementing burst (denoted by AD[1:0]=00 during the address phase). A PCI burst request specifying any type of sequence other than linear results in a disconnect after the first data phase.

7.2.7.4 Inbound Delayed Read Matching Rules

The completion response returned to the GXB for a Delayed Read transaction is matched to the requesting PCI master by using information that was latched into the target during the initial request. This information consists of the address, command and byte enables. Table 7-2 illustrates which read request information is used for matching Delayed Read Completions to the requesting agent.

Table 7-2. Delayed Read Matching Criteria

Command	Address	BEs
Any Memory Read	Match	Match

When a DRC is valid in the GXB, a 2^{15} PCI clock timer is started as described in the PCI 2.2 Specification. When the timer expires the DRC is discarded and the associated delayed read matching registers are cleared. This condition is optionally treated as an error. See the "Error" Chapter for details.

7.2.7.5 Inbound I/O Reads

I/O reads on the PCI bus are not claimed by the GXB.

7.2.7.6 Inbound Writes

Memory write requests may be directed to the memory only. All write requests are posted. A series of buffers in both the GXB and SAC allow posted writes to be accumulated and serviced as convenient. The actual data is forwarded to buffers in the SAC, where it is held until the request can be snooped on the system bus.

Writes to Memory

The snoops on the system bus for PCI Stream AGP-DRAM writes will be initiated by using the Memory Read and Invalidate of Length=0 system bus transaction. Any implicit write back due to snoop hit to a modified line will be snarfed by the SDC. The write back data will be merged with the AGP write data within the SDC.

Writes to memory use the linear burst ordering provided on the AGP bus.

The Write and Write & Invalidate commands have small differences, as follows:

Write — Accumulates posted data until (a) a cache line boundary is reached, or (b) the master disconnects, before forwarding the request to the SAC. The SAC therefore deals with a single packet that represents up to a cache line of data. The SAC places a cache-line snoop on the bus, and the SDC handles the writing/merging of the partial data into memory. The Expander bus command is a write.

Write & Invalidate — By definition this must be an aligned cache line worth of data. It is forwarded to the SAC as a cache line of data. The SAC places a cache-line snoop on the bus for each line of data, and the SDC writes the full cache line into memory. In the event the snoop results in a HITM#, the write back data from the processor is discarded by the SDC.

Note that the actual writing of the data into memory may be delayed. However, in effect, the data can be counted as being written in the memory, since no other write to that location may pass this write.

7.2.7.7 Inbound I/O Writes

I/O writes on the PCI bus are not claimed by the GXB.

7.2.7.8 Retry/Disconnect Conditions

The GXB as a PCI target retries the initial data phase of inbound access when:

- The read request is to an address that has already been accepted as a delayed transaction (i.e. the request is already being serviced, but data has not arrived).
- A write request has insufficient buffering in the gxb to allow it to be posted. (a full line is not available for mwi).
- The pci interface is locked from the host side.
- No delayed read buffer is available.

The GXB as a target will issue a disconnect to a read when no more data is available (at the end of the cacheline), or if linear addressing is not used. The GXB as a target will issue a disconnect to a write when no more posting buffers are available, or when the write crosses a cacheline boundary.

7.2.7.9 Outbound Reads

Outbound reads are enqueued in the GXB. The GXB will hold up to four cache line reads at one time. The GXB will continuously retry an outbound read until it completes successfully on the AGP bus. Outbound posted writes must be allowed to pass the read(s) between retry attempts. PCI 2.2 ordering rules allow reads to pass each other. However, this is optional in the specification and is not required to guarantee forward progress. For the GXB, outbound reads are done in order. If a read is retried, then all other reads after it will wait until the first has completed.

7.2.7.10 Outbound Writes

Write Combining

The GXB optimizes outbound write performance by combining writes to sequential locations (if enabled) into a single write burst on the AGP bus. ***This holds true for all memory attributes, not just WC.*** The GXB only provides write combining; no collapsing or byte merging is performed. The source of the transaction is not checked. Accesses from processors could be combined with peer accesses.

The GXB will combine an access if the next data is valid at the head of the queue in time; this puts a restriction on the size of requests that can be combined (if a request is too short the GXB does not have time to look ahead). The GXB's write combining support is illustrated in Table 7-3.

Notice that even if use of the Memory Write and Invalidate command is enabled, the GXB will not terminate a burst that was started with a Memory Write command in order to switch to the Memory Write Invalidate Command when it receives a full line. However, if a Memory Write Invalidate burst has been started and the next sequential access is less than a line, the GXB must terminate and switch to using the Memory Write command. Table 7-4 shows some write combining examples.

Table 7-3. Burst Write Combining Modes

Write Command Used	Transfer Mode	Data Length	Combining Supported
Memory Write	1X	< 4 DW	Can not be combined with the next access.
		>= 4 DW	Can combine next access if it is sequential, regardless of next access size.
Memory Write	2X	< 8 DW	Can not be combined with the next access.
		>= 8 DW	Can combine next access if it is sequential, regardless of next access size.
Memory Write	4X	< 16 DW	Can not be combined with the next access.
		>= 16 DW	Can combine next access if it is sequential, regardless of next access size.
Memory Write Invalidate	All	must be Line	Can only combine sequential lines.

Table 7-4. Burst Write Combining Examples with 3 Writes in 1X Transfer Mode

1st write	2nd write	3rd write	Transferred as:
x DW (x<4)	y DW	z DW	if y < 4 x DW followed by y DW followed by z DW else x DW followed by (y+z) DW
x DW (x>=4)	y DW	z DW	if y < 4 (x+y) DW followed by z DW else (x+y+z) DW

7.2.7.11 Fast Back-to-Back Transactions

The GXB as a PCI target will accept fast back-to-back cycles from a PCI master accessing different agents during back-to-back sequence. As an initiator the GXB does not generate a fast back-to-back cycle.

7.3 Bandwidth

The bandwidth of AGP 4X mode is 1 GB peak. The sustained bandwidth will be less than this. The sustainable bandwidth obtainable with the GX is dependent on several things including the size of the requests from the graphics card. Table 7-5 shows an estimate of obtainable bandwidth. There is a fairly broad range for each of the sizes. The bandwidth will depend on the mix of traffic. Traffic consisting of all reads by the graphics card or all writes by the processor to the card will result in higher bandwidths than a 50/50 mix of reads by the graphics card and writes by the processor to the card. This table will change as the implementation is completed and should be viewed as a guideline for the graphics card designer for relative performance trade-offs.

Table 7-5. Bandwidth Estimates for Various Request Sizes

Request Size (in bytes)	Sustainable Bandwidth (MB/s)	Latency (in ns.) for First Data Returned on AGP Bus
8	100-300	500
16	250-500	500
32	450-700	530
64	750-850	600
128	850-925	750
256	800-850	1000

7.3.1 Inbound Read Prefetching

The PCI protocol has no transfer size explicitly spelled out. Reads begin and continue until the device has the data it needs. For performance, a PCI bridge could prefetch data ahead of when the PCI device requests it. AGP, as opposed to PCI, has a length field in its protocol. Therefore the graphics bridge does know how much data to fetch. Also the GXB has 1 GB/s peak into it, which is the AGP peak, so there is no extra bandwidth in case the bridge prefetched the wrong data. For this reason, the 460GX chipset will have no prefetching of graphics data. The graphics card should have enough buffering and outstanding transactions to keep its pipes full. With pipelining, the card can fetch the data it needs in advance of using it. Knowing what it wants to do, the graphics card can prefetch only the data it expects to use. This allows the best utilization of the graphics port.

7.4 Latency

Latency of reads from the graphics card is an important parameter. Unfortunately it is difficult to specify usefully. For an idle system, the latency of a graphics read will be 450-500 ns. For a moderately loaded system, the latency of a read should be 600-900 ns. Defining ‘moderately loaded’ is the difficult part. If the GXB queues are backed up servicing many small AGP requests, then the latency may be much higher.

7.5 GXB Address Map

The ‘System Address Map’ chapter contains a section describing the what the system address map looks like from the perspective of an expander bridge. This section is included to explicitly state which address ranges must be checked at the AGP interface when using PCI protocol and which ranges must be checked after GART translation.

There are two address regions which must be checked at the AGP bus interface when the card is using PCI protocol: the GXB’s nx32M logical PCI space and the VGA space. This is required since the AGP card may attempt to “talk to itself” using the PCI protocol in these regions, so the GXB must not assert DEVSEL# to these accesses. The GXB’s nx32M region is specified using its MMBASE and MMTOP registers. The VGA gap is enabled as a memory hole using the VGASE register in the GXB. All other accesses using PCI protocol will receive DEVSEL#. This includes accesses above TOM, in a MARG, etc. that may cause XBINIT# after the GART (see below).

All regions, including the two described above, must be checked after GART translation. The GXB must only allow accesses that are directed to physical memory to reach the SAC. Therefore, the GXB must force a BINIT#, by asserting its “XBINIT#” output, when it detects an access falling in any of the following regions:

- Above the TOM register value (firmware sets TOM to top of physical system memory - this may be anywhere in memory, as low as 64 MB or as high as 64 GB)
- Between GAPBASE and GAPTOP registers and not in the SAPIC interrupt delivery region - firmware must set these two registers to cover the following ranges:
  - — In High System firmware (fixed range from 4G to 4G-16M).
  - — In first megabyte of Itanium processor specific (fixed range from 4G-16M to 4G-17M).
  - — In Itanium processor specific below the Interrupt region (fixed range from 4G-18M to 4G-20M).
  - — In chipset specific (fixed range from 4G-20M to 4G-32M).
  - — In any of the nx32M PCI spaces.

Therefore GAPTOP=4G and GAPBASE is the lowest PXB’s/GXB’s MMBASE in the system. Note, the SAPIC interrupt delivery area from 4G-17M to 4G-18M must be allowed and specifically decoded so it will *not* cause a fault.

- In a MARG with DRAM accesses disabled (the MARGs cover the C, D, E, and F segments).
- In the VGA region (A_0000-BFFFF) with VGAGE enabled so that this region is directed to PCI, a PCI access in this range would not have gotten DEVSEL#.

WXB Hot-Plug 8

8.1 IHPC Configuration Registers

Each WXB supports two independent Integrated Hot-Plug[1] Controllers (IHPCs). The A-side controller (IHPA) and the B-side controller (IHPB) are configured independently. Each IHPC therefore has its own configuration space. Both configuration spaces are identical.

Each IHPC reserves a 256 byte configuration space. A list of the configuration registers specific to hot-plug operation follows. The default power-up value is included in each register description heading. All registers return to their default values upon the assertion of XRST# or PCI reset (for the particular side in question) unless otherwise noted in the description. The standard PCI Configuration registers are not described here when implemented exactly as described in the PCI Specification 2.2.

The following map shows registers specific to the IHPC.

1. WXB Integrated Hot-Plug controller derived from technology licensed from Compaq Corporation.

Table 8-1. IHPC Configuration Register Space

DID		VID		00h			80h
PCISTS		PCICMD		04h			84h
CLASS		RID		08h			88h
	HDR	MLT	CLS	0Ch			8Ch
Base Address				10h			90h
				14h			94h
				18h			98h
				1Ch			9Ch
				20h			A0h
				24h			A4h
				28h			A8h
		SID	SVID	2Ch			ACh
				30h			B0h
				34h			B4h
				38h			B8h
		Interrupt Pin	Interrupt Line	3Ch			BCh
Miscellaneous Configuration (RW)		Slot ID		40h			C0h
		Hot-Plug Features		44h			C4h
	Arbiter SERR status	Power Fault SERR status	Switch Change SERR status	48h			C8h
				4Ch			CCh
Memory Access Index				50h			D0h
Memory Mapped Register Access Port				54h			D4h
				58h			D8h
				5Ch			DCh
				60h			E0h
				64h			E4h
				68h			E8h
				6Ch			ECh
				70h			F0h
				74h			F4h
				78h			F8h
				7Ch			FCh

NOTE: The first 64 bytes are predefined in the PCI specification. All other locations are defined specifically for the component of interest.

8.1.1 Page Number List for the IHPC PCI Register Descriptions

8.1.2 VID: Vendor Identification Register

Address Offset:	00 – 01h	Size:	16 bits
Default Value:	8086h	Attributes:	Read-Only

The VID Register contains the vendor identification number, identifying the manufacturer of the device. This 16-bit register combined with the Device Identification Register uniquely identifies any PCI device. Writes to this register have no effect.

Bits	Description
15:0	**Vendor Identification Number** This is a 16-bit value assigned to Intel. Intel VID = 8086h

8.1.3 DID: Device Identification Register

Address Offset:	02 – 03h	Size:	16 bits
Default Value:	123Fh	Attributes:	Read-Only

This 16-bit register combined with the Vendor Identification register uniquely identifies any PCI device. Writes to this register have no effect.

Bits	Description
15:0	**Device Identification Number** The value 123F indicates the IHPC

8.1.4 PCICMD: PCI Command Register

Address Offset:	04h-05h	Size:	16 bits
Default Value:	0000h	Attribute:	Partial Read/Write

The PCI command register provides control over the IHPC's ability to generate and respond to PCI cycles. When a zero (0) is written to this register, the IHPC is logically disconnected from the PCI bus for all accesses except configuration. The register bit list below shows the layout of the register, explains the meanings of the different bits in the command register, gives the default value of this register upon power-up, and gives the specific implementation of individual bits in the IHPC (i.e. R/O or R/W).

Bits	Description
15:10	*reserved (0)*
9	**Fast Back-to-Back Enable** This bit is not supported. Hardwired Value = 0.
8	**SERR# Enable** This bit is an enable bit for the SERR# driver. A value of zero (0) disables the SERR# driver. A value of one (1) enables the SERR# driver. WXB configuration space PCICMD(6) must be set to 1 to permit WXB SERR# pin assertion. Since the IHPC is integrated into the WXB and is not addressable from the local PCI bus segment, the IHPC will not report address parity errors via SERR#. This bit is configurable in the IHPC with a default value of zero (0).
7	**Wait Cycle Control** This bit is not supported. Hardwired Value = 0.
6	**Parity Error Enable** This bit controls a device's response to IHPC configuration and memory write parity errors. When the bit is 1, the IHPC will assert both the Detected Parity Error status bit and the PERR# output upon detection of an error. When the bit is 0, the IHPC will assert the Detected Parity Error status on an error but will not assert PERR#. This bit must be set to zero (0) after RST#. This bit is configurable in the IHPC with the default value zero (0).
5	**VGA Palette Snoop** This bit controls how VGA-compatible devices handle accesses to their palette registers. This bit is always set to zero (0) in the IHPC (disabled).
4	**Memory Write and Invalidate Enable** This is an enable bit for using the memory write and invalidate command. This bit is always set to zero (0) in the IHPC (disabled).
3	**Special Cycle** This bit controls a device's action on special cycle operations. A value of zero (0) causes the device to ignore all special cycle operations. This bit is always set to zero (0) in the IHPC.
2	**Bus Master** This bit controls a device's ability to act as a master on the PCI bus. A value of zero (0) disables the device from generating PCI accesses. A value of one (1) allows the device to behave as a bus master. Hardwired Value = 0.
1	**Memory Space** This bit controls a device's response to memory space accesses. A value of zero (0) disables the device response. A value of one (1) allows the device to respond to memory space accesses. This bit is configurable in the IHPC with the default value of zero (0).
0	**I/O Space** This bit controls a device's response to I/O space accesses. The IHPC does not respond to I/O space accesses. Hardwired Value = 0.

8.1.5 PCISTS: PCI Status Register

Address Offset:	06h – 07h	Size:	16 bits
Default Value:	0200h	Attribute:	Partial Read/Write, Sticky

The PCI status register is used to record status information for PCI bus-related events. The definition of each of the bits is given in the register bit list below. The specific implementation of each bit in the IHPC is also given. Reads to this register behave normally. Writes are slightly different in that bits can be reset, but not set. A bit is reset whenever the register is written and the data in the corresponding bit location is a one (1). For instance, to clear bit 14 and not affect any other bits, write the value 0100_0000_0000_0000b to the register.

Bits	Description
15	**Detected Parity Error** This bit is set by the IHPC whenever it detects a parity error, even if parity error handling is disabled (as controlled by bit 6 in the command register). The default value of this bit in the IHPC is zero (0).
14	**Signaled System Error** This bit is set whenever the IHPC asserts SERR#. The default value of this bit in the IHPC is zero (0).
13	**Received Master Abort** Not supported. Hardwired Value = 0
12	**Received Target Abort** Not supported. Hardwired Value = 0
11	**Signaled Target Abort** Not supported. Hardwired Value = 0
10:9	**DEVSEL# Timing** These bits encode the timing of DEVSEL#. There are three allowable timings for assertion of DEVSEL#. These are encoded as 00b for fast, 01b for medium, and 10b for slow (11b is reserved). The value of these bits are always set to medium (01).
8	**Data Parity Error** Not supported. Hardwired Value = 0
7	**Fast Back-to-Back Capable** Not supported. Hardwired Value = 0
6:0	*reserved(0)*

8.1.6 RID: Revision Identification Register

Address Offset:	08h	Size:	8 bits
Default Value:	Stepping-Dependent	Attribute:	Read-Only

This register contains the revision number of the IHPC. These bits are read-only and writes to this register have no effect.

Bits	Description
7:0	**Revision Identification Number** This is an 8-bit value that indicates the revision identification number for the IHPC WXB A Steppings: Hardwired Value = 00h WXB B0 Step: Hardwired Value = 01h

8.1.7 CLASS: Class Register

Address Offset:	09 – 0Bh	Size:	24 bits
Default Value:	080400h	Attribute:	Read-Only

This register contains the Class Code for the IHPC, specifying the device function. Writes to this register will have no effect.

Bits	Description
23:16	**Base Class** This field indicates the general device category. The IHPC is a Base System Peripheral. Hardwired Value = 08h.
15:8	**Sub-Class** This field qualifies the Base Class, providing a more detailed specification of the device function. For the IHPC this field indicates a Generic PCI Hot-Plug Controller. Hardwired Value = 04h.
7:0	**Register-level Programming Interface** This field identifies a specific programming interface (if any), that device independent software can use to interact with the device. The Interface is not defined. Hardwired Value = 00h.

8.1.8 CLS: Cache Line Size

Address Offset:	0Ch	Size:	8 bits
Default Value:	00h	Attribute:	Read/Write

See *PCI Specification, Rev. 2.2.*

8.1.9 MLT: Master Latency Timer Register

Address Offset:	0Dh	Size:	8 bits
Default Value:	00h	Attribute:	Read/Write

See *PCI Specification, Rev. 2.2. This register is not applicable to the IHPC's within the WXB.*

8.1.10 HDR: Header Register

Address Offset:	0Eh	Size:	8 bits
Default Value:	00h	Attribute:	Read-Only

This register identifies the header layout of the configuration space. Writes to this register will have no effect.

Bits	Description
7	**Multi-function Device** Selects whether this is a multi-function device, that may have alternative configuration layouts. The IHPC is not a multifunction device. Hardwired Value = 0.
6:0	**Configuration Layout** This field identifies the format of the 10h through 3Fh space. This field specifies the "standard" or "default" PCI configuration layout. Hardwired Value = 00h

8.1.11 Base Address

Address Offset:	10h-13h	Size:	32 bits
Default Value:	00000000h	Attribute:	Read/Write, Read-Only

This is a standard PCI configuration register which provides power-up software with the ability to build a consistent address map before booting the machine.

Bits	Description
31:8	**Base Address** Read/Write.
7:4	**Indicate 256-byte Address Space Requested** Hardwired Value = 0h.
3	**Not Prefetchable** Hardwired Value = 0.
2:1	**Type:** Located anywhere in 32-bit address space. Hardwired Value = 00.
0	**Memory Space Indicator** Hardwired Value = 0.

8.1.12 SVID: Subsystem Vendor Identification

Address Offset:	2Ch –2Dh	Size:	16 bits
Default Value:	0000h	Attributes:	Write-Once (Pwr Good Reset Only)

See *PCI Specification, Revision 2.2*.

8.1.13 SID: Subsystem ID

Address Offset:	2Eh - 2Fh	Size:	16 bits
Default Value:	0000h	Attributes:	Write-Once (Pwr Good Reset Only)

See *PCI Specification, Revision 2.2*.

8.1.14 Interrupt Line

Address Offset:	3Ch	Size:	8 bits
Default Value:	FFh	Attribute:	Read-Write (Pwr Good Reset Only)

This is a standard PCI configuration register which defines which interrupt request line on the interrupt controller this function's interrupt pin (see register 3DH) is connected to. **The power-up default value is FFh**.

8.1.15 Interrupt Pin

Address Offset:	3Dh	Size:	8 bits
Default Value:	01h	Attribute:	Read Only

This is a standard PCI configuration register which defines which of the four PCI interrupt pins, INTA# through INTD#, this function is connected to. PCI Hot-Plug is connected to INTA#, making this value hard-wired to 01h.

8.1.16 Hot-Plug Slot Identifier

Address Offset:	40h-41h	Size:	16 bits
Default Value:	0000h	Attribute:	Partial Read/Write

This register indicates which slots support hot-plug. Not used by the IHPC, this register is set by the boot ROM for later reference by the hot-plug drivers. This register is also mapped to memory space. System designers are required to follow a convention whereby hot-plug slots are always implemented in sequential PCI device number order. IHPC slot A should be the lowest numbered slot (i.e. the device number in this register).

Bits	Description
15:8	*reserved(0)*
7:4	The PCI device number for the first slot that supports hot-plug.
3:0	Number of hot-plug slots controlled.

8.1.17 Miscellaneous Hot-Plug Configuration

Address Offset:	42h-43h	Size:	16 bits
Default Value:	0002h [1]	Attribute:	Read/Write, Write-Once, Read-Only

This is a hot-plug specific register used to configure many features of the IHPC.

Bits	Description
15	Change device ID. Write Once/Read Only. Changes the device ID from 123Fh to 123Eh in order to "hide" the IHPC from industry-standard hot-plug drivers and device enumerators in non-hot-plug systems.
14	Inhibit hot-plug registers. Write Once/Read Only. This bit prohibits access to the IHPC registers, effectively disabling the IHPC.
13	Enable Power Fault functions. This bit must be set to enable other power-fault related SERR signaling.
12	Enable Lock of the Auto Power-Down Disable bit. Write Once/Read Only. This bit should be written to a logic 1 (by firmware) after the Auto-Power-Down Disable bit of the Hot-Plug Miscellaneous register has been written. Once written, this bit causes the Auto-Power-Down Disable bit to be unchangeable until XRST# or PCI reset. This prevents driver software from changing the auto-power-down feature.

1. Bit 0 may be changed to a 1 after Reset in future implementation.

11:8	*reserved(0)*
7	Enable PCI Configuration Space Access to Hot-Plug Registers. Enables IHPC memory-mapped register access through the index register (configuration offset 50h) and data port (configuration offset 54h).
6:2	*reserved (0)*
1	*reserved (1)*
0	On / Off Busy (OOBS) status. Read Only. Same as bit 24 of the memory-mapped Hot-Plug Miscellaneous register.

8.1.18 Hot-Plug Features

Address Offset:	44h-45	Size:	16 bits
Default Value:	0000h	Attribute:	Read Only

Definitions of each bit within this register are expected to be constant through the industry but are, as of yet, undefined.

Bits	Description
15:0	*reserved (0)*

8.1.19 Switch Change SERR Status

Address Offset:	48h	Size:	8 bits
Default Value:	00h	Attribute:	Partial Read/Write

Bits	Description
7:6	*reserved (0)*
5:0	Switch Change SERR Status. Slot F is MSB. Slot A is LSB. Similar to the Power Fault SERR status register, but applicable to switch changes when the switch interrupt redirect bit for a slot is set and the associated interrupt mask bit is logic 0. Unlike the Power Fault SERR status register, clearing this bit will also clear the associated interrupt.

8.1.20 Power Fault SERR Status

Address Offset:	49h	Size:	8 bits
Default Value:	00h	Attribute:	Partial Read/Write

Bits	Description
7:6	*reserved.*
5:0	Power Fault SERR Status. If the power fault function enable bit and the SERR on power fault bit are both set, then these six bits will indicate (by a logic 1, one for each slot) if a power fault has occurred while a slot was connected to the bus or clock. Slot F is MSB. Slot A is LSB. These bits can be cleared by writing a logic 1 to the appropriate position. This register does not effect PCI interrupts.

8.1.21 Arbiter SERR Status

Address Offset:	4A	Size:	8 bits
Default Value:	00h	Attribute:	Partial Read/Write

Bits	Description
7:0	*reserved (0)*

8.1.22 Memory Access Index

Address Offset:	50h-53h	Size:	32 bits
Default Value:	00000000h	Attribute:	Partial Read/Write

When the “Enable PCI Config Space Access to Hot-Plug Registers” bit in the Miscellaneous Hot-Plug Configuration Register is set, this register becomes a pointer into the IHPC memory mapped register space (divided into 64, 32-bit Dwords).

Bits	Description
31:8	*reserved (0)*
7:2	Hot-Plug Memory Access Index Read/Write
1:0	*reserved (0)*

8.1.23 Memory Mapped Register Access Port

Address Offset:	54h-57h	Size:	32 bits
Default Value:	00000000h	Attribute:	Read/Write

When the “Enable PCI Config Space Access to Hot-Plug Registers” bit in the Miscellaneous Hot-Plug Configuration Register is set, this register becomes mapped into the IHPC memory mapped register space at the location pointed to by the Memory Index Register.

Bits	Description
31:0	Memory Mapped Register Access Port

8.2 IHPC Memory Mapped Registers

Each IHPC reserves 256 bytes of memory mapped registers. A list of those registers follows. Unlike other register descriptions in this document, the offset listed is always Dword aligned with bit offset provided. This nomenclature is used to be consistent with the Memory Access Index Register in IHPC configuration space. The default power-up value is included in each register description heading.

Table 8-2. IHPC Memor Mapped Register Space

Hot-Plug Miscellaneous (RW)		Slot Enable (RW)		00h	80h
LED Control (RW)				04h	84h
Hot-Plug Interrupt Input and Clear (RW)				08h	88h
Present 1 bits	Present 2 bits	Power Faults	Switches		
Hot-Plug Interrupt Mask(RW)				0Ch	8Ch
Present 1 mask	Present 2 mask	Power Fault mask	Switch Mask		
User-Defined Outputs (RW)	Reserved	Serial Input Byte Pointer (RW)	Serial Input Data (RO)	10h	90h
Hot-Plug Non-Interrupt Inputs (RW)				14h	94h
Reserved	Reserved	Reserved	M66EN		
Reserved				18h	98h
				1Ch	9Ch
				20h	A0h
				24h	A4h
			Slot ID (RW)	28h	A8h
		Slot Power (RW)	Switch Int. Redirect Enable (RW)	2Ch	ACh
Extended Hot-Plug Misc. (RW)				30h	B0h
				34h	B4h
				38h	B8h
				3Ch	BCh
				40h	C0h
				44h	C4h
				48h	C8h
				4Ch	CCh
				50h	D0h
				54h	D4h
				58h	D8h
				5Ch	DCh
				60h	E0h
				64h	E4h
				68h	E8h
				6Ch	ECh
				70h	F0h
				74h	F4h
				78h	F8h
				7Ch	FCh

NOTE: The first 64 bytes are predefined in the PCI specification. All other locations are defined specifically for the component of interest.

8.2.1 Page Number List for IHPC Memory Mapped Register Descriptions

8.2.2 Slot Enable

Address Offset:	01h	Size:	8 bits
Default Value:	00h	Attribute:	Partial Read/Write (Pwr Good Rst Only)

Used to power-on a slot and connect it to the bus (or disconnect and power-down). The SOGO bit must be set to start the output sequence. The set of usable Enable Slot bits is determined by the strapping values on the P(A,B)HSIL, P(A,B)HSOL, and P(A,B)HSOC inputs. Unsupported slots in a system do not have writeable Enable Slot bits. Writing a zero to a Enable Slot bit will clear the associated Slot Power register bit.

Bits	Description
7:6	*reserved(0)*
5	**Enable Slot F** When 1, Slot F is powered and connected to the PCI bus
4	**Enable Slot E** When 1, Slot E is powered and connected to the PCI bus
3	**Enable Slot D** When 1, Slot D is powered and connected to the PCI bus
2	**Enable Slot C** When 1, Slot C is powered and connected to the PCI bus
1	**Enable Slot B** When 1, Slot B is powered and connected to the PCI bus
0	**Enable Slot A** When 1, Slot A is powered and connected to the PCI bus

8.2.3 Hot-Plug Miscellaneous

Address Offset:	02h - 03h	Size:	16 bits
Default Value:	0040h	Attribute:	Partial Read/Write

Bits	Description
15	*reserved (0)*
14	Enable SERR on Power Fault. When set, the assertion of a slot power fault causes a SERR# to be asserted if SERR# generation is enabled in the PCI device command register, the slot is connected to the bus or PCI clock, and IHPC power fault functions are enabled (bit 10 of this register).
13	*reserved (0)*
12	Input Scan Complete. This bit is cleared at the conclusion of each input cycle. Software can use this bit to determine when fresh data is available by setting it and waiting until a logic 0 is seen.
11	66 MHz Prescaler Enable, Read Only, set according to the PCI frequency, a 1 indicates the PCI frequency is 66 MHz, a 0 indicates 33 MHz.
10	Enable Power Fault Functions. This bit is also mapped to bit 13 of the IHPC Misc. Configuration Register (configuration offset 42h).
9	Auto Power Down Disable. Controls whether or not opening a slot switch will cause a powered slot to auto-power-down.
8	On / Off Busy status. (Same as Configuration Register 42, bit 0)
7	*reserved (0)*
6	*reserved (1)*
5	*reserved (0)*
4	Dummy Cycle Enable, this feature is not supported on this stepping.
3	General-interrupt-input Interrupt Pending. Set to a logic 1 when an interrupt is generated. Cleared when the interrupt is cleared.
2	Shift Output Interrupt Pending / Clear. When read as logic 1, a hot-plug interrupt was generated by SOBS changing from 1 to 0 while the Serial Output Interrupt Enable bit was set. Writing a logic 1 clears this bit and its interrupt.
1	Shift Output Interrupt Enable. When set to a logic 1, an interrupt will be generated when SOBS changes from 1 to 0, indicating completion of a serial output sequence.
0	Shift Output Go / Busy Status. Writing a logic 1 to this bit (after it has been read as a logic 0) initiates a serial output sequence (e.g. power-down of a slot). When read as a logic 0, the previous serial output sequence has completed. Ensure that LED blinking has completed (or is forced to complete) prior to initiating a SOGO for a slot power-up or power-down.

8.2.4 LED Control

Address Offset:	04h	Size:	32 bits
Default Value:	sampled at PWRGD	Attribute:	Read/Write (Pwr Good Rst Only)

If both the MSB and LSB for an LED are logic 0, then the LED is turned off. The LSB is for Blink phase A and the MSB is for Blink phase B. Setting either phase A or phase B will program the LED to blink. Setting both the MSB and the LSB will turn the LED on. An auto-power-down sequence will turn off the LEDs for that slot. It is intended that the green LED is a power indicator and the amber LED is the attention indicator.Following chip power on, all LED bits will be cleared

for unpopulated slots and slots with open switches. The set of usable LED Control bits is determined by the strapping values on the P(A,B)HSIL, P(A,B)HSOL, and P(A,B)HSOC inputs. Unsupported slots in a system do not have writeable LED Control bits. LEDs are not affected by changes to these register bits until a SOGO is initiated following the changes.

Bits	Description
31:30	*reserved (0)*
29	Slot F Amber LED, msb
28	Slot E Amber LED, msb
27	Slot D Amber LED, msb
26	Slot C Amber LED, msb
25	Slot B Amber LED, msb
24	Slot A Amber LED, msb
23:22	*reserved (0)*
21	Slot F Amber LED, lsb
20	Slot E Amber LED, lsb
19	Slot D Amber LED, lsb
18	Slot C Amber LED, lsb
17	Slot B Amber LED, lsb
16	Slot A Amber LED, lsb
15:14	*reserved (0)*
13	Slot F Green LED, msb
12	Slot E Green LED, msb
11	Slot D Green LED, msb
10	Slot C Green LED, msb
9	Slot B Green LED, msb
8	Slot A Green LED, msb
7:6	*reserved(0)*
5	Slot F Green LED, lsb
4	Slot E Green LED, lsb
3	Slot D Green LED, lsb
2	Slot C Green LED, lsb
1	Slot B Green LED, lsb
0	Slot A Green LED, lsb

8.2.5 Hot-Plug Interrupt Input and Clear

Address Offset:	08h	Size:	32 bits
Default Value:	Varies	Attribute:	Read/Write Clear (Pwr Gd Rst Only)

These bits indicate the state of the interrupt capable inputs when no interrupt is pending. When a hot-plug general interrupt changes state (either high to low or low to high), and the bit is not masked in the Interrupt Mask register, an interrupt is generated and the state of that input is latched

into this register. Writing a logic 1 will clear the pending interrupt. If there are no other pending interrupts on the bit, the bit will clear. This register takes on a value based on the monitored status of the slots and therefore has no particular default value.

Bits	Description
31:30	*reserved (0)*
29	Slot F PRSNT(0)#, PCI Present Signal 1
28	Slot E PRSNT(0)#, PCI Present Signal 1
27	Slot D PRSNT(0)#, PCI Present Signal 1
26	Slot C PRSNT(0)#, PCI Present Signal 1
25	Slot B PRSNT(0)#, PCI Present Signal 1
24	Slot A PRSNT(0)#, PCI Present Signal 1
23:22	*reserved (0)*
21	Slot F PRSNT(1)#, PCI Present Signal 2
20	Slot E PRSNT(1)#, PCI Present Signal 2
19	Slot D PRSNT(1)#, PCI Present Signal 2
18	Slot C PRSNT(1)#, PCI Present Signal 2
17	Slot B PRSNT(1)#, PCI Present Signal 2
16	Slot A PRSNT(1)#, PCI Present Signal 2
15:14	*reserved (0)*
13	Slot F FAULT#, PCI Power Fault Signal
12	Slot E FAULT#, PCI Power Fault Signal
11	Slot D FAULT#, PCI Power Fault Signal
10	Slot C FAULT#, PCI Power Fault Signal
9	Slot B FAULT#, PCI Power Fault Signal
8	Slot A FAULT#, PCI Power Fault Signal
7:6	*reserved(0)*
5	Slot F Hot-Plug Switch, 0 = lever closed (board installed)
4	Slot E Hot-Plug Switch, 0 = lever closed (board installed)
3	Slot D Hot-Plug Switch, 0 = lever closed (board installed)
2	Slot C Hot-Plug Switch, 0 = lever closed (board installed)
1	Slot B Hot-Plug Switch, 0 = lever closed (board installed)
0	Slot A Hot-Plug Switch, 0 = lever closed (board installed)

8.2.6 Hot-Plug Interrupt Mask

Address Offset:	0Ch	Size:	32 bits
Default Value:	FFFFFFFFh	Attribute:	Read/Write

This read/write mask register is used to indicate which inputs should generate interrupts and which should not. The mask bits in this register map one-for-one with the HIICR (Interrupt Input and Clear Register) bits. If a state change occurs on an input while the mask bit for that input is set to one, then no interrupt will be generated for that state change. If the mask bit is cleared, then an interrupt will be generated on the next state change.

Bits	Description
31:30	reserved (0)
29	Slot F PRSNT(0)#, PCI Present Signal 1
28	Slot E PRSNT(0)#, PCI Present Signal 1
27	Slot D PRSNT(0)#, PCI Present Signal 1
26	Slot C PRSNT(0)#, PCI Present Signal 1
25	Slot B PRSNT(0)#, PCI Present Signal 1
24	Slot A PRSNT(0)#, PCI Present Signal 1
23:22	*reserved (0)*
21	Slot F PRSNT(1)#, PCI Present Signal 2
20	Slot E PRSNT(1)#, PCI Present Signal 2
19	Slot D PRSNT(1)#, PCI Present Signal 2
18	Slot C PRSNT(1)#, PCI Present Signal 2
17	Slot B PRSNT(1)#, PCI Present Signal 2
16	Slot A PRSNT(1)#, PCI Present Signal 2
15:14	*reserved (0)*
13	Slot F FAULT#, PCI Power Fault Signal
12	Slot E FAULT#, PCI Power Fault Signal
11	Slot D FAULT#, PCI Power Fault Signal
10	Slot C FAULT#, PCI Power Fault Signal
9	Slot B FAULT#, PCI Power Fault Signal
8	Slot A FAULT#, PCI Power Fault Signal
7:6	*reserved(0)*
5	Slot F Hot-Plug Switch, 0 = lever closed (board installed)
4	Slot E Hot-Plug Switch, 0 = lever closed (board installed)
3	Slot D Hot-Plug Switch, 0 = lever closed (board installed)
2	Slot C Hot-Plug Switch, 0 = lever closed (board installed)
1	Slot B Hot-Plug Switch, 0 = lever closed (board installed)
0	Slot A Hot-Plug Switch, 0 = lever closed (board installed)

8.2.7 Serial Input Byte Data

Address Offset:	10h	Size:	8 bits
Default Value:	00h	Attribute:	Read-Only (Pwr Good Rst Only)

After the Serial Input Busy Status is read as a logic zero, the requested byte will be in the Serial Input Data Register. The contents will be a copy of the corresponding byte in the Interrupting Input and Clear Register (least significant) or Non-Interrupting Inputs Register (most significant).

Bits	Description
7:0	Serial Input Data Register. RO

8.2.8 Serial Input Byte Pointer

Address Offset:	11h	Size:	16 bits
Default Value:	00h	Attribute:	Read/Write, Read-Only

Used to input a byte into the IHPC input registers. The byte number is written to the pointer. After the Serial Input Busy Status is read as a logic zero, the requested byte will be in the Serial Input Data Register. The contents will be a copy of the corresponding byte in the Interrupting Input and Clear Register (least significant) or Non-Interrupting Inputs Register (most significant).

Bits	Description
7	Serial Input Busy Status: RO
6:4	*reserved (0)*
3:0	Serial Input Byte Pointer. RW

8.2.9 General Purpose Output

Address Offset:	13h	Size:	8 bits
Default Value:	00h	Attribute:	Read/Write (Pwr Good Rst Only)

These bits are only driven out following SOGO/LED cycles.

Bits	Description
7:6	*reserved (0)*
5:0	General Purpose Output bits (one bit per supported slot). Slot F is MSB. Slot A is LSB.

8.2.10 Hot-Plug Non-interrupt Inputs

Address Offset:	14h	Size:	32 bits
Default Value:		Attribute:	Read Only

Bits	Description
31:6	*reserved (0)*
5	Slot F M66EN, PCI 66 MHz Clock Enable
4	Slot E M66EN, PCI 66 MHz Clock Enable
3	Slot D M66EN, PCI 66 MHz Clock Enable
2	Slot C M66EN, PCI 66 MHz Clock Enable
1	Slot B M66EN, PCI 66 MHz Clock Enable
0	Slot A M66EN, PCI 66 MHz Clock Enable

8.2.11 Hot-Plug Slot Identifier

Address Offset:	28h	Size:	16 bits
Default Value:	0000h	Attribute:	Read/Write

This register is a copy of Configuration Register 40h, byte 0.

8.2.12 Hot-Plug Switch Interrupt Redirect Enable

Address Offset:	2Ch	Size:	8 bits
Default Value:	00h	Attribute:	Read/Write

This register allows the slot switch change interrupts to be redirected to the SERR# instead of the INTA#.

Bits	Description
7:6	*reserved (0)*
5	Slot F INTR Redirect Enable
4	Slot E INTR Redirect Enable
3	Slot D INTR Redirect Enable
2	Slot C INTR Redirect Enable
1	Slot B INTR Redirect Enable
0	Slot A INTR Redirect Enable

8.2.13 Slot Power Control

Address Offset:	2Dh	Size:	8 bits
Default Value:	Sampled at PWRGD	Attribute:	Read/Write (Pwr Good Rst Only)

This register is used to power a slot without connecting it to the bus. SOGO must be set to cause the output sequence. Slots that are connected to the bus cannot be powered down through this register (the Slot Enable register is used instead). The set of usable Slot Power Control bits is determined by the strapping values on the P(A,B)HSIL, P(A,B)HSOL, and P(A,B)HSOC inputs. Unsupported slots in a system do not have writeable Slot Power Control bits.

Bits	Description
7:6	*reserved (0)*
5	Enable Power to Slot F. When 1, slot F is powered up. When 0, slot F is powered down
4	Enable Power to Slot E. When 1, slot E is powered up. When 0, slot E is powered down
3	Enable Power to Slot D. When 1, slot D is powered up. When 0, slot D is powered down
2	Enable Power to Slot C. When 1, slot C is powered up. When 0, slot C is powered down
1	Enable Power to Slot B. When 1, slot B is powered up. When 0, slot B is powered down
0	Enable Power to Slot A. When 1, slot A is powered up. When 0, slot A is powered down

8.2.14 Extended Hot-Plug Miscellaneous

Address Offset:	32h	Size:	16 bits
Default Value:	0000h	Attribute:	Partial Read/Write

Bits	Description
15:0	*reserved (0)*

IFB Register Mapping 9

The IFB internal registers are organized into four Functions–LPC/FWH interface bridge, IDE Controller, USB Host Controller, and Enhanced Power Management. Each Function has its registers divided into 1 set of PCI Configuration Registers and one or more register sets located in system I/O space.

Software should not map programmable memory or I/O address registers such that any part of the range overlaps addresses decoded by other IFB devices.

Some of the IFB registers contain reserved bits. Software must deal correctly with fields that are reserved. On reads, software must use appropriate masks to extract the defined bits and not rely on reserved bits being any particular value. On writes, software must ensure that the values of reserved bit positions are preserved. That is, the values of reserved bit positions must first be read, merged with the new values for other bit positions and then written back.

In addition to reserved bits within a register, the IFB contains address locations in the PCI configuration space that are marked “Reserved”. The IFB responds to accesses to these address locations by completing the Host cycle. Software should not write to reserved IFB configuration locations in the device-specific region (above address offset 3Fh).

During a hard reset, the IFB sets its internal registers to predetermined **default** states. The default values are indicated in the individual register descriptions.

The following notation is used to describe register access attributes:

RO **Read Only**. If a register is read only, writes have no effect.

WO **Write Only**. If a register is write only, reads have no effect.

R/W **Read/Write**. A register with this attribute can be read and written. Note that individual bits in some read/write registers may be read only.

R/WC **Read/Write Clear**. A register bit with this attribute can be read and written. However, a write of a 1 clears (sets to 0) the corresponding bit and a write of a 0 has no effect.

9.1 PCI / LPC / FWH Configuration

The IFB PCI Function 0 contains a LPC/FWH interface, interrupt controller and counter / timers, including the real time clock. The register set associated with this Functionality and associated logic is shown below with actual register descriptions given in this section.

9.1.1 PCI Configuration Registers (Function 0)

Table 9-1. PCI Configuration Registers–Function 0(PCI to LPC/FWH Interface Bridge)

Configuration Offset	Mnemonic	Register	Register Access
00–01h	VID	Vendor Identification	RO
02–03h	DID	Device Identification	RO
04–05h	PCICMD	PCI Command	R/W

Table 9-1. PCI Configuration Registers–Function 0(PCI to LPC/FWH Interface Bridge) (Cont'd)

Configuration Offset	Mnemonic	Register	Register Access
06–07h	PCISTS	PCI Device Status	R/W
08h	RID	Revision Identification	RO
09-0Bh	CLASSC	Class Code	RO
0C–0Dh	–	Reserved	–
0Eh	HEDT	Header Type	RO
0F–3Fh	–	Reserved	–
40–43h	ACPIBR	ACPI Base Address Register	R/W
44h	ACPIEN	ACPI Enable	R/W
45h	SCIRC	Reserved	R/W
46–4Bh	–	Reserved	–
4Ch	–	Reserved	R/W
4Dh	–	Reserved	–
4E–4Fh	BIOSEN	Reserved	R/W
50–5Fh	–	Reserved	–
60–63h	PIRQRC[A:D]	Reserved	R/W
64h	SERIRQC	Serial IRQ Control	R/W
65–68h	–	Reserved	–
69h	TOM	Top of Memory	R/W
6A–6Bh	MSTAT	Miscellaneous Status	R/W
6C–75h	–	Reserved	–
76–77h	–	Reserved	R/W
78–7Bh	–	Reserved	R/W
7C–7Fh	–	Reserved	–
80h	–	APIC Base Address Relocation	R/W
81h	–	Reserved	–
82h	DLC	Deterministic Latency Control	R/W
83h	–	Reserved	–
84–85h	MGPIOC	Muxed GPIO Control	R/W
86–8Fh	–	Reserved	–
90-91h	PDMACFG	PCI DMA Configuration	R/W
92-95h	DDMABASE	Distributed DMA Slave Base Pointer	R/W
96-C7h	–	Reserved	–
C8h	RTCCFG	Real Time Clock Configuration	R/W
C9–CFh	–	Reserved	–
D0–D3h	GPIOBA	GPIO Base Address Register	R/W
D4h	GPIOE	GPIO Enable	R/W
D5–DFh	–	Reserved	–
E0h	LPCCD	LPC COM Decode	R/W
E1h	LPCFD	LPC FDD/LPT Decode	R/W
E2h	LPCSD	LPC Sound Decode	R/W
E3h	FWHDE	Firmware Hub Decode Enable	–
E4–E5h	LPCGD	LPC Generic Decode Range	R/W
E6–E7h	LPCDE	LPC Decode Enables	R/W
E8h	FWHS	Firmware Hub Select	–
E9–FFh	–	Reserved	–

9.2 IDE Configuration

The IFB PCI function 1 contains an IDE Controller capable of standard Programmed I/O (PIO) transfers as well as Bus Master transfer capability. It also supports the “Ultra DMA/33” synchronous DMA mode of data transfer.

9.2.1 PCI Configuration Registers (Function 1)

Table 9-2. PCI Configuration Registers–Function 1 (IDE Interface)

Configuration Offset	Mnemonic	Register	Register Access
00–01h	VID	Vendor Identification	RO
02–03h	DID	Device Identification	RO
04–05h	PCICMD	PCI Command	R/W
06–07h	PCISTS	PCI Device Status	R/W
08h	RID	Revision Identification	RO
09-0Bh	CLASSC	Class Code	RO
0Ch	–	Reserved	–
0Dh	MLT	Master Latency Timer	R/W
0Eh	HEDT	Header Type	RO
0F–1Fh	–	Reserved	–
20–23h	BMIBA	Bus Master Interface Base Address	R/W
24–3Fh	–	Reserved	–
40–43h	IDETIM	IDE Timing	R/W
44h	SIDETIM	Slave IDE Timing	R/W
45–47h	–	Reserved	–
48h	SDMACTL	Synchronous DMA Control	R/W
49h	–	Reserved	–
4A–4Bh	SDMATIM	Synchronous DMA Timing	R/W
4C–F7h	–	Reserved	–
F8-FBh	---	Manufacturer’s ID	---
FC-FFh	---	Reserved	---

9.3 Universal Serial Bus (USB) Configuration

The IFB integrates an USB Controllers. The USB Controller is UHCI 1.1 compliant. It implements the root hub of the USB, which contains two ports.

The IFB PCI function 2 reflects both the Universal Serial Bus Host and Root Hubs, with 2 connected USB ports.

9.3.1 PCI Configuration Registers (Function 2)

Table 9-3. PCI Configuration Registers–Function 2 (USB Interface)

Configuration Offset	Mnemonic	Register	Register Access
00–01h	VID	Vendor Identification	RO
02–03h	DID	Device Identification	RO
04–05h	PCICMD	PCI Command	R/W
06–07h	PCISTS	PCI Device Status	R/W
08h	RID	Revision Identification	RO
09-0Bh	CLASSC	Class Code	RO
0Ch	–	Reserved	–
0Dh	MLT	Latency Timer	R/W
0Eh	HEDT	Header Type	RO
0F–1Fh	–	Reserved	–
20–23h	USBBA	USB I/O Space Base Address	R/W
24–3Bh	–	Reserved	–
3Ch	INTLN	Interrupt Line	R/W
3Dh	INTPN	Interrupt Pin	RO
3E–5Fh	–	Reserved.	–
60h	SBRNUM	Serial Bus Release Number	RO
61–69h	–	Reserved	–
6A–6Bh	MCR	Miscellaneous Control Register	R/W
6C–BFh	–	Reserved	–
C0–C1h	LEGSUP	USB Legacy Keyboard/Mouse Control	R/W
C2-C3h	---	Reserved	-----
C4h	USBRES	USB Resume Enable	R/W
C5-F3h	–	Reserved	–
F4-F7h	–	Reserved	–
F8-FBh	–	Manufacturer’s ID Register	–
FC-FFh	–	Reserved	–

9.4 SMBus Controller Configuration

The IFB PCI function 3 contains the SMBus Controller configuration space.

9.4.1 SMBus Configuration Registers (Function 3)

Table 9-4. PCI Configuration Registers–Function 3 (SMBus Controller Interface)

Configuration Offset	Mnemonic	Register	Register Access
00–01h	VID	Vendor Identification	RO
02–03h	DID	Device Identification	RO
04–05h	PCICMD	PCI Command	R/W
06–07h	PCISTS	PCI Device Status	R/WC
08h	RID	Revision Identification	RO
09-0Bh	CLASSC	Class Code	RO
0C-1Fh	–	Reserved	–
20-23h	BAR	Base Address Register	R/W
24-3Bh	–	Reserved	–
3Ch	IL	Interrupt Line	RW
3Dh	IP	Interrupt Pin	RO
3E-3Fh	–	Reserved	–
40h	HC	Host Configuration	RW
41h	SCOM	Slave Command Port	RW
42h	SS1	Slave Shadow Address 1	RW
43h	SS2	Slave Shadow Address 2	RW
41-F3h	–	Reserved	–
F4-F7h	–	Reserved	–
F8-FBh	–	Manufacturer’s ID Register	–
FC-FFh	–	Reserved	–

IFB Usage Considerations 10

This section talks about the normal usage for some of the features in the IFB component.

10.1 Usage of 1MIN Timer in Power Management

IFB does not support the global standby timer concept. The determination of a system inactivity can be done by using the 1MIN Timer that can be used by the SMI handler to generate an SMI# every minute. The SMM handler can check all the appropriate power management status registers in IFB to see if there is any system inactivity. After n minutes of no system activity (where n is determined by the SMM handler, time based setup option for the user to indicate system inactivity), the SMM handler can decide to put the system into a lower power state.

10.2 Usage of the SW SMI# Timer

The Software SMI# Timer provides a mechanism for the SMM code to temporarily exit SMM space. This is required when some application code is required to complete before the SMM code can complete. An example of this may be when a Soft-Off or Suspend-To-Disk is requested, and some application code must complete.

After starting the timer, the SMM code exits. When the timer expires, another SMI# is generated. At that point, the SMM code has the option to restart the timer, if needed.

If the SMM handler desires, it can halt the timer by setting its enable bit to 0. This will prevent the SMI# generation at the next time-out.

10.3 CD-ROM AUTO RUN Feature of the OS

Whenever a CD-ROM controller is detected in a system, the OS may use the auto run feature if enabled to indicate to the user whether a CD-ROM has been plugged-in into the CD-ROM controller. The OS driver issues a Test Unit Ready (TUR) command to check for the insertion of media. The firmware or the OS driver needs to ensure that the IRQ on which the CD-ROM controller is being serviced is not being monitored with the help of the enabled Wake/Break enable registers. Also the PM timers available in the IFB should be used appropriately before the firmware goes to the low power state. All these changes need to be done if the system wishes to use the auto run feature in the OS and still be able to use the power management features of the IFB.

10.4 ACPI, SMBus, GPIO Base Address Reporting to the OS

The firmware should report all of the register space that is being used by the ACPI, SMBus and GPIO functionalities to the OS by using the PNP device node entries. They should be declared as Plug and Play motherboard resources with the Device ID of PNP0C02. If the ACPI tables are used instead of the PNP Device node entries for reporting resources the change has to be incorporated

into the system firmware by the vendor. This reporting will make these register locations safe and the OS will not use these locations randomly if a PNP conflicting device is relocatable in those I/O or memory locations. These locations also got to be reported to the OS whenever the OEM sends the systems for their WHQL suite test.

10.5 Ultra DMA Configuration

The following registers are programmed in systems that contain devices that implement the Ultra DMA Protocol. These registers allow Ultra DMA to be used when PCI Bus Master IDE operation is initiated by the device driver.

10.5.1 UDMAC–Ultra DMA Control Register (IFB Function 1 PCI Configuration Offset 48h)

7	6	5	4	3	2	1	0
Reserved				Secondary Drive 1 Ultra DMA Mode Enable (SSDE1)	Secondary Drive 0 Ultra DMA Mode Enable (SSDE0)	Primary Drive 1 Ultra DMA Mode Enable (PSDE1)	Primary Drive 0 Ultra DMA Mode Enable (PSDE0)
				0: Disabled 1: Enabled	0: Disabled 1: Enabled	0: Disabled 1: Enabled	0: Disabled 1: Enabled

10.5.2 UDMATIM–Ultra DMA Timing Register (IFB Function 1 PCI Configuration Offsets 4A-4Bh)

15	14	13	12	11	10	9	8
Reserved		Secondary Drive 1 Ultra DMA Cycle Time (SCT1)		Reserved		Secondary Drive 0 Ultra DMA Cycle Time (SCT1)	
		00: CT=4 clks, RP=6 clks 01: CT=3 clks, RP=5 clks 10: CT=2 clks, RP=4 clks 11: Reserved				00: CT=4 clks, RP=6 clks 01: CT=3 clks, RP=5 clks 10: CT=2 clks, RP=4 clks 11: Reserved	

7	6	5	4	3	2	1	0
Reserved		Primary Drive 1 Ultra DMA Cycle Time (PCT1)		Reserved		Primary Drive 0 Ultra DMA Cycle Time (PCT1)	
		00: CT=4 clks, RP=6 clks 01: CT=3 clks, RP=5 clks 10: CT=2 clks, RP=4 clks 11: Reserved				00: CT=4 clks, RP=6 clks 01: CT=3 clks, RP=5 clks 10: CT=2 clks, RP=4 clks 11: Reserved	

NOTES:

- ***The Ultra DMA*** Enable bit specifies the current Ultra DMA enabled status:
  - — Disabled by default: This field needs to be enabled in order to take advantage of the IFB Ultra DMA timings. When this field is disabled, the *IFB Ultra DMA Timing Register is disabled*.
- ***The Ultra DMA Cycle Time Field*** specifies the current Ultra DMA timing mode.
  Note that this field only applies if the corresponding Ultra DMA Enable field is set.

10.5.3 Determining a Drive's Transfer Rate Capabilities

10.5.3.1 Overview

The following section provides the information that allows a drive's capabilities to be determined.

The ATA IDENTIFY_DRIVE (0xEC) and ATAPI IDENTIFY_DEVICE(0x1A) can be used to determine the capability. Each of these commands returns a 256 word buffer with fields that can allow the drive's capabilities to be determined.

Refer to the ATA Specification for more information regarding the IDENTIFY_DEVICE command.

Table 10-1. Identify Device Information Used for Determining Drive Capabilities

Capability	Word Offset	Bits	Field
Device Type			Fields that Indicate Device Type
Device Type	0	15	General Configuration: 0: ATA Device 1: ATAPI Device
Ultra DMA			**Fields that Indicate Ultra DMA Drive Capabilities**
Ultra DMA	53	2	Field Validity 0: the fields reported in word 88 are not valid 1: the fields reported in word 88 are valid
Ultra DMA	88	10:8	Ultra DMA Modes Active* bit 10: 1: Ultra DMA Mode 2 is active 0: Ultra DMA Mode 2 is not active bit 9: 1: Ultra DMA Mode 1 is active 0: Ultra DMA Mode 1 is not active bit 8: 1: Ultra DMA Mode 0 is active 0: Ultra DMA Mode 0 is not active
Ultra DMA	88	2:0	Ultra DMA Modes Supported* bit 2: 1: Ultra DMA Mode 2 is supported 0: Ultra DMA Mode 2 is not supported bit 1: 1: Ultra DMA Mode 1 is supported 0: Ultra DMA Mode 1 is not supported bit 0: 1: Ultra DMA Mode 0 is supported 0: Ultra DMA Mode 0 is not supported
PIO, DMA	53	1	Field Validity 0: the fields reported in words 64-70 are not valid 1: the fields reported in word 64-70 are valid

Table 10-1. Identify Device Information Used for Determining Drive Capabilities (Cont'd)

Capability	Word Offset	Bits	Field
DMA	62	10:8	Single Word DMA Modes Active bit 10: 1: Single Word DMA Mode 2 is active 0: Single Word DMA Mode 2 is not active bit 9: 1: Single Word DMA Mode 1 is active 0: Single Word DMA Mode 1 is not active bit 8: 1: Single Word DMA Mode 0 is active 0: Single Word DMA Mode 0 is not active
DMA	62	2:0	Single Word DMA Modes Supported bit 2: 1: Single Word DMA Mode 2 is supported 0: Single Word DMA Mode 2 is not supported bit 1: 1: Single Word DMA Mode 1 is supported 0: Single Word DMA Mode 1 is not supported bit 0: 1: Single Word DMA Mode 0 is supported 0: Single Word DMA Mode 0 is not supported
DMA	63	10:8	Multi Word DMA Modes Active bit 10: 1: Multi Word DMA Mode 2 is active 0: Multi Word DMA Mode 2 is not active bit 9: 1: Multi Word DMA Mode 1 is active 0: Multi Word DMA Mode 1 is not active bit 8: 1: Multi Word DMA Mode 0 is active 0: Multi Word DMA Mode 0 is not active
DMA	63	2:0	Multi Word DMA Modes Supported bit 2: 1: Multi Word DMA Mode 2 is supported 0: Multi Word DMA Mode 2 is not supported bit 1: 1: Multi Word DMA Mode 1 is supported 0: Multi Word DMA Mode 1 is not supported bit 0: 1: Multi Word DMA Mode 0 is supported 0: Multi Word DMA Mode 0 is not supported
DMA	65	15:0	Minimum Multi Word DMA Transfer Cycle Time Per Word
PIO	51	15:8	PIO Data Transfer Cycle Timing Mode Supported 00h: PIO0 01h: PIO1 02h: PIO2
PIO, DMA	53	1	Field Validity 0: the fields reported in words 64-70 are not valid 1: the fields reported in word 64-70 are valid
PIO	64	7:0	Advanced Flow Control PIO Transfer Modes Supported bit 0: PIO3 (w/IORDY Flow Control) bit 1: PIO4 (w/IORDY Flow Control)
PIO	68	15:0	Minimum PIO Transfer Cycle Time with IORDY Flow Control

For IFB IDE Timing Configuration, each of the following things must be determined:

- Drive Type: ATAPI or ATA (non-ATAPI)
- Best DMA Capability
- Best Ultra DMA Capability OR
- Best Multi Word DMA Capability (if Ultra DMA not supported) OR
- Best Single Word DMA Capability (if neither Ultra DMA nor Multi Word DMA supported)
- Best PIO Capability

10.5.4 Determining a Drive's Best Ultra DMA Capability

The drive's ultra DMA mode capability and current configuration are specified in the IDENTIFY_DRIVE buffer, Word 88. Software must first check to see that the Word 88 is valid before determining the Ultra DMA drive capability.

Table 10-2. Identify Device Information Used for Determining Ultra DMA Drive Capabilities

Capability	Word Offset	Bits	Field
Ultra DMA	53	2	Field Validity 0: the fields reported in word 88 are not valid 1: the fields reported in word 88 are valid
Ultra DMA	88	10:8	Ultra DMA Modes Active* bit 10: 1: Ultra DMA Mode 2 is active 0: Ultra DMA Mode 2 is not active bit 9: 1: Ultra DMA Mode 1 is active 0: Ultra DMA Mode 1 is not active bit 8: 1: Ultra DMA Mode 0 is active 0: Ultra DMA Mode 0 is not active
Ultra DMA	88	2:0	Ultra DMA Modes Supported* bit 2: 1: Ultra DMA Mode 2 is supported 0: Ultra DMA Mode 2 is not supported bit 1: 1: Ultra DMA Mode 1 is supported 0: Ultra DMA Mode 1 is not supported bit 0: 1: Ultra DMA Mode 0 is supported 0: Ultra DMA Mode 0 is not supported

The following Ultra DMA drive capabilities are supported by the IFB, from fastest to slowest:

- Ultra DMA Mode 2
- Ultra DMA Mode 1
- Ultra DMA Mode 0
- Disabled (Drive does not support any of the above Ultra DMA Modes.)

10.5.5 Determining a Drive's Best Multi Word DMA/Single Word DMA (Non-ultra DMA) Capability

This section describes how to determine a drive's multi word DMA and single word DMA capabilities.

The following DMA drive capabilities are supported by the IFB, from fastest to slowest:

- Multi Word DMA Mode 2
- Multi Word DMA Mode 1
- Single Word DMA Mode 2
- Disabled (Drive does not support any of the above Multi/Single DMA Modes.)

Software at this stage needs to determine if at least one of the above modes is supported by the drive. Software should initially determine a drive's best Mult Word DMA capability initially. If a drive doesn't support multi word DMA Modes 0 or 1, then software should check if single word DMA Mode 2 is supported.

The drive's multi word DMA mode capability and current configuration are specified in the IDENTIFY_DRIVE buffer, Words 63 and 65 Software must first check to see that the Words 64-70 are valid before determining the drive's multi word DMA drive capability.

The drive's single word DMA mode capability and current configuration are specified in the IDENTIFY_DRIVE buffer, Word 62.

Table 10-3. Identify Device Information Used for Determining Multi/Single Word DMA Drive Capabilities

Capability	Word Offset	Bits	Field
PIO, DMA	53	1	Field Validity 0: the fields reported in words 64-70 are not valid 1: the fields reported in word 64-70 are valid
DMA	62	10:8	Single Word DMA Modes Active bit 10: 1: Single Word DMA Mode 2 is active 0: Single Word DMA Mode 2 is not active bit 9: 1: Single Word DMA Mode 1 is active 0: Single Word DMA Mode 1 is not active bit 8: 1: Single Word DMA Mode 0 is active 0: Single Word DMA Mode 0 is not active
DMA	62	2:0	Single Word DMA Modes Supported bit 2: 1: Single Word DMA Mode 2 is supported 0: Single Word DMA Mode 2 is not supported bit 1: 1: Single Word DMA Mode 1 is supported 0: Single Word DMA Mode 1 is not supported bit 0: 1: Single Word DMA Mode 0 is supported 0: Single Word DMA Mode 0 is not supported
DMA	63	10:8	Multi Word DMA Modes Active bit 10: 1: Multi Word DMA Mode 2 is active 0: Multi Word DMA Mode 2 is not active bit 9: 1: Multi Word DMA Mode 1 is active 0: Multi Word DMA Mode 1 is not active bit 8: 1: Multi Word DMA Mode 0 is active 0: Multi Word DMA Mode 0 is not active
DMA	63	2:0	Multi Word DMA Modes Supported bit 2: 1: Multi Word DMA Mode 2 is supported 0: Multi Word DMA Mode 2 is not supported bit 1: 1: Multi Word DMA Mode 1 is supported 0: Multi Word DMA Mode 1 is not supported bit 0: 1: Multi Word DMA Mode 0 is supported 0: Multi Word DMA Mode 0 is not supported
DMA	65	15:0	Minimum Multi Word DMA Transfer Cycle Time per Word

To determine the drive's best DMA capability ensure that the drive reports the capability and is able to transfer at the target cycle time, as shown in Table 10-4.

Table 10-4. Drive Multi Word DMA/Single Word DMA Capability as a Function of Cycle Time

Drive's Reported DMA Mode Setting	Drives Reported DMA Cycle Time[1]	Drive's Best DMA Mode
Multi Word DMA Mode 2	t <= 120ns	Multi Word DMA Mode 2
Multi Word DMA Mode 2	120 < t <= 180ns	Multi Word DMA Mode 1
Multi Word DMA Mode 2	180 < t <= 240ns	Single Word DMA Mode 2
Multi Word DMA Mode 2	t > 240ns	N/A (Disabled)
Multi Word DMA Mode 1	t <= 180ns	Multi Word DMA Mode 1
Multi Word DMA Mode 1	180 < t <= 240ns	Single Word DMA Mode 2
Multi Word DMA Mode 1	t > 240ns	N/A (Disabled)
Multi Word DMA Mode 0	N/A	N/A (Disabled)
Single Word DMA Mode 2	t <= 240ns	Single Word DMA Mode 2
Single Word DMA Mode 2	t > 114ns	N/A (Disabled)
Single Word DMA Mode 1	N/A	N/A (Disabled)
Single Word DMA Mode 0	N/A	N/A (Disabled)
Drive does not support DMA transfers.	N/A	N/A

NOTE: Timing cycle times are defined by the ATA specification. A device that reports a given DMA mode capability must be capable of supporting the minimum DMA cycle time.

A drive's Multi Word DMA or Single Word DMA speed is the fastest Multi Word DMA speed, if supported, that is supported for the drive. If Multi Word DMA is NOT supported for the reported timing cycle time then the drive's best non Ultra DMA speed is based on its Single Word DMA 2 capability.

If N/A is used, the drive is configured for PIO-only. The PIIX mode follows the PIO mode only. If a drive does not report a DMA cycle time that is consistent with the Target DMA Cycle Time, a slower speed should be chosen.

The cycle times associated with the various timing modes are defined by the ATA Specification. A drive that reports a given Multi/Single Word DMA capability must be capable of supporting the minimum cycle time for that mode.

10.5.5.1 Determining a Drive's Best PIO Capability

This section describes how to determine a drive's PIO Capabilities.

The following PIO drive capabilities are supported by the IFB, from fastest to slowest:

- PIO4 w/IORDY
- PIO3 w/IORDY
- PIO2 w/IORDY
- PIO2 (without IORDY)
- Compatible (Drive does not support any of the above PIO Modes.)

Software at this stage needs to determine if at least one of the above modes is supported by the drive. Software should initially determine a drive's best PIO w/IORDY capability (PIO4 w/IORDY or PIO3 w/IORDY) initially. If these PIO w/IORDY modes are not supported, the drive should determine the PIO2 mode support with IORDY or PIO2 mode support without IORDY. Otherwise, Compatible timings should be applied to the drive.

The drive's PIO w/IORDY mode capability is specified in the IDENTIFY_DRIVE buffer, Words 64 and 68. Software must first check to see that the Words 64-70 are valid before determining the drive's PIO w/IORDY drive capability.

The drive's PIO2 mode capability and current configuration are specified in the IDENTIFY_DRIVE buffer, Word 51.

Table 10-5. Identify Device Information Used for Determining PIO Drive Capabilities

Capability	Word Offset	Bits	Field
PIO	51	15:8	PIO Data Transfer Cycle Timing Mode Supported 00h: PIO0 01h: PIO1 02h: PIO2
PIO, DMA	53	1	Field Validity 0: the fields reported in words 64-70 are not valid 1: the fields reported in word 64-70 are valid
PIO	64	7:0	Advanced Flow Control PIO Transfer Modes Supported bit 0: PIO3 (w/IORDY Flow Control) bit 1: PIO4 (w/IORDY Flow Control)
PIO	68	15:0	Minimum PIO Transfer Cycle Time with IORDY Flow Control

To determine the drive's best PIO capability ensure that the drive reports the capability and is able to transfer at the target cycle time:

Table 10-6. Drive PIO Capability as a Function of Cycle Time

Drive's Reported PIO Mode Setting	Drives Reported PIO Cycle Time	Drive's Best PIO Mode
PIO4	t <= 120ns	PIO4
PIO4	120 < t <= 180ns	PIO3
PIO4	180 < t <= 240ns	PIO2
PIO4	t > 240ns	PIO0/Compatible
PIO3	t <= 180ns	PIO3
PIO3	180 < t <= 240ns	PIO2
PIO3	t > 240ns	PIO0/Compatible
PIO2	N/A (drive must support t<= 240ns)	PIO2
PIO1	N/A	PIO0/Compatible
PIO0/Compatible	N/A	PIO0/Compatible

Note: The cycle times associated with the various timing modes are defined by the ATA Specification. A drive that reports a given PIO capability must be capable of supporting the minimum cycle time for that mode.

Note: If a drive does not report a PIO cycle time that is consistent with the Target PIO Cycle Time, a slower speed should be chosen.

10.5.6 IFB Timing Settings

10.5.6.1 DMA/PIO Timing Settings

In Table 10-7, ‘x’=depends on the type of drive installed, ‘1’=enabled, and ‘0’=disabled.

Ultra DMA mode settings are completely independent of the following timings.

Table 10-7. IFB Drive Mode Based on DMA/PIO Capabilities

Drive’s Best DMA Capability	Drive’s Best PIO Capability	IFB Timing Mode	Fast PIO Supported? Best PIO Mode >= Best DMA Mode	Non Ultra DMA Supported? Best DMA Mode is {SW2, MW1, MW2}	DMA Timing Enable Only Select [a]	Prefetch and Posting Enable Select	IORDY Sample Point Enable Select	Fast Timing Bank Drive Select
N/A (DMA not supported)	PIO0/1/ Compatible	Mode 0	No	No	Disabled	Enabled (if fixed disk)	Disabled	Disabled
N/A (DMA not supported)	PIO2	Mode 2	Yes	No	Disabled	Enabled (if fixed disk)	Depends on Drive	Enabled
N/A (DMA not supported)	PIO3 (w/IORDY)	Mode 3	Yes	No	Disabled	Enabled (if fixed disk)	Enabled	Enabled
N/A (DMA not supported)	PIO4 (w/IORDY)	Mode 4	Yes	No	Disabled	Enabled (if fixed disk)	Enabled	Enabled
Single Word DMA Mode 2	PIO0/1/ Compatible	Mode 2	No (special config. needed)	Yes	Enabled	Enabled (if fixed disk)	Depends on Drive	Enabled
Single Word DMA Mode 2	PIO2	Mode 2	Yes	No	Disabled	Enabled (if fixed disk)	Depends on Drive	Enabled
Single Word DMA Mode 2	PIO3 (w/IORDY)	Mode 2	Yes	No	Disabled	Enabled (if fixed disk)	Depends on Drive	Enabled
Single Word DMA Mode 2	PIO4 (w/IORDY)	Mode 2	Yes	No	Disabled	Enabled (if fixed disk)	Depends on Drive	Enabled
Multi Word DMA Mode 1	PIO0/1/ Compatible/ PIO2	Mode 3	No (special config. needed)	Yes	Enabled	Enabled (if fixed disk)	Enabled	Enabled
Multi Word DMA Mode 1	PIO3 (w/IORDY)/ PIO4 (w/IORDY)	Mode 3	Yes	No	Disabled	Enabled (if fixed disk)	Enabled	Enabled
Multi Word DMA Mode 2	PIO0/1/ Compatible/ PIO2	Mode 4	No (special config. needed)	Yes	Enabled	Enabled (if fixed disk)	Enabled	Enabled
Multi Word DMA Mode 2	PIO3 (w/IORDY)	Mode 3	Yes	No	Disabled	Enabled (if fixed disk)	Enabled	Enabled
Multi Word DMA Mode 2	PIO4 (w/IORDY)	Mode 4	Yes	No	Disabled	Enabled (if fixed disk)	Enabled	Enabled

a. Configurations where a drive reports a PIO speed much slower than its reported DMA speed require the DMA Timing Enable Only Select bit to be Enabled.

Configurations where a drive reports a PIO speed much slower than its reported DMA speed require the *DMA Timing Enable Only Select* bit to be Enabled.

Table 10-8. IDE Mode/Drive Feature Settings for Optimal DMA/PIO Operation

IFB IDE Drive 0 Mode Settings	IFB IDE Drive 1 Mode Settings	DMA Timing Enable Only Select 0[a]	Pre-Fetch and Posting Enable Select 0	IORDY Sample Point Enable Select 0[b]	Fast Timing Bank Drive Select 0	DMA Timing Enable Only Select 1	Pre-Fetch and Posting Enable Select 1	IORDY Sample Point Enable Select 1	Fast Timing Bank Drive Select 1	IDE Timing Register Value bits 15:8 (hex)[c]	Slave IDE Timing Register Value (hex) bits 3:0 (Primary) OR bits 7:4 (Second-ary)	IDE Timing Register Value bits 7:0 (binary)
										All speeds	All speeds	
Mode 4	Mode 0	Depends on Drive	Enabled (if fixed disk)	Enabled	Enabled	Disabled	Enabled (if fixed disk)	Disabled	Disabled	A3	0	0x00xx11
Mode 4	Not Present	Depends on Drive	Enabled (if fixed disk)	Enabled	Enabled	Disabled	Disabled	Disabled	Disabled	A3	0	0000xx11
Mode 4	Mode 2	Depends on Drive	Enabled (if fixed disk)	Enabled	Enabled	Depends on Drive	Enabled (if fixed disk)	Depends on Drive	Enabled	E3	4	xxx1xx11
Mode 4	Mode 3	Depends on Drive	Enabled (if fixed disk)	Enabled	Enabled	Depends on Drive	Enabled (if fixed disk)	Enabled	Enabled	E3	9	xx11xx11
Mode 4	Mode 4	Depends on Drive	Enabled (if fixed disk)	Enabled	Enabled	Depends on Drive	Enabled (if fixed disk)	Enabled	Enabled	E3	B	xx11xx11
Mode 3	Mode 0	Depends on Drive	Enabled (if fixed disk)	Enabled	Enabled	Disabled	Enabled (if fixed disk)	Disabled	Disabled	A1	0	0x00xx11
Mode 3	Not Present	Depends on Drive	Enabled (if fixed disk)	Enabled	Enabled	Disabled	Disabled	Disabled	Disabled	A1	0	0000xx11
Mode 3	Mode 2	Depends on Drive	Enabled (if fixed disk)	Enabled	Enabled	Depends on Drive	Enabled (if fixed disk)	Depends on Drive	Enabled	E1	4	xxx1xx11
Mode 3	Mode 3	Depends on Drive	Enabled (if fixed disk)	Enabled	Enabled	Depends on Drive	Enabled (if fixed disk)	Enabled	Enabled	E1	9	xx11xx11
Mode 3	Mode 4	Depends on Drive	Enabled (if fixed disk)	Enabled	Enabled	Depends on Drive	Enabled (if fixed disk)	Enabled	Enabled	E1	B	xx11xx11
Mode 2	Mode 0	Depends on Drive	Enabled (if fixed disk)	Depends on Drive	Enabled	Disabled	Enabled (if fixed disk)	Disabled	Disabled	90	0	0x00xxx1
Mode 2	Not Present	Depends on Drive	Enabled (if fixed disk)	Depends on Drive	Enabled	Disabled	Disabled	Disabled	Disabled	90	0	0000xxx1
Mode 2	Mode 2	Depends on Drive	Enabled (if fixed disk)	Depends on Drive	Enabled	Depends on Drive	Enabled (if fixed disk)	Depends on Drive	Enabled	D0	4	xxx1xxx1
Mode 2	Mode 3	Depends on Drive	Enabled (if fixed disk)	Depends on Drive	Enabled	Depends on Drive	Enabled (if fixed disk)	Enabled	Enabled	D0	9	xx11xxx1
Mode 2	Mode 4	Depends on Drive	Enabled (if fixed disk)	Depends on Drive	Enabled	Depends on Drive	Enabled (if fixed disk)	Enabled	Enabled	D0	B	xx11xxx1
Mode 0	Mode 0	Disabled	Enabled (if fixed disk)	Disabled	Disabled	Disabled	Enabled (if fixed disk)	Disabled	Disabled	80	0	00000000
Mode 0	Not Present	Disabled	Enabled (if fixed disk)	Disabled	Disabled	Disabled	Disabled	Disabled	Disabled	80	0	00000000
Mode 0	Mode 2	Disabled	Enabled (if fixed disk)	Disabled	Disabled	Depends on Drive	Enabled (if fixed disk)	Depends on Drive	Enabled	C0	4	xxx10000
Mode 0	Mode 3	Disabled	Enabled (if fixed disk)	Disabled	Disabled	Depends on Drive	Enabled (if fixed disk)	Enabled	Enabled	C0	9	xx110000
Mode 0	Mode 4	Disabled	Enabled (if ifxed disk)	Disabled	Disabled	Depends on Drive	Enabled (if fixed disk)	Enabled	Enabled	C0	B	xx110000

NOTES:
1. DMA Timing Enable Only field is in general Disabled. It is only Enabled in certain cases if the DMA Mode capability of the drive is much greater than the PIO Mode capability of the drive.
2. The IORDY Sample Point field must be Enabled for PIO Modes 3 and 4. It is Enabled on PIO2 drives if and only if IORDY capability is supported in the drive.
3. The above recommendations assume that if the attached slave drive is Mode 0 or not present, SITRE bit is '0'.

Table 10-9. DMA/PIO Timing Values Based on PIIX Cable Mode/System Speed

IFB Drive Mode	IORDY Sample Point (ISP) bits 1:0	Recovery Time (RCT) bits 1:0	IDETIMx Value Drive 0 (Master) if Slave Attached bits 15:8	IDETIMx Value Drive 0 (Master) if no Slave Attached or Slave is Mode 0 bits 15:8	SIDETIM Value Drive 1 (Slave) bits 3:0 (Primary) or bits 7:4 (Secondary)	Resultant Cycle Time (Total Clocks Base Operating Freq)
PIO0/ Compatible	Default	Default	C0h	80h	0	30 MHz: 900ns 33 MHz: 900ns
PIO2/SW2	4 clocks	4 clocks	D0h	90h	4	30 MHz: 256ns 33 MHz: 240ns
PIO3/MW1	3 clocks	3 clocks	E1h	A1h	9	30 MHz: 198ns 33 MHz: 180ns
PIO4/MW2	3 clocks	1 clock	E3h	A3h	B	30 MHz: 132ns 33 MHz: 120ns

10.5.6.2 Ultra DMA Timing Settings

The following settings apply to Ultra DMA Mode Settings only.

Table 10-10. Ultra DMA Timing Value Based on Drive Mode

IFB Drive Mode	DMA Speed Used on DMA Based Data Transfer Commands	SDMAC Value Ultra DMA Mode Enable x: Drive 0: bit 0 Drive 1: bit 1 Drive 2: bit 2 Drive 3: bit 3	SDMATIMx Value Ultra DMA Cycle Time x: Drive 0: bits 1:0 Drive 1: bits 5:4 Drive 2: bits 9:8 Drive 3: bits 13:12
N/A (Ultra DMA Not Supported)	Non-ultra DMA if supported	Disabled	Default
Ultra DMA Mode 0	Ultra DMA Mode 0	Enabled	00b: CT=4 clks, RP=6 clks
Ultra DMA Mode 1	Ultra DMA Mode 1	Enabled	01b: CT=3 clks, RP=5 clks
Ultra DMA Mode 2	Ultra DMA Mode 2	Enabled	10b: CT=2 clks, RP=4 clks

10.5.7 Drive Configuration for Selected Timings

Once the IFB Timing Modes for DMA, PIO and Ultra DMA have been selected, the Set Features Command (0 x EF) with Set Transfer Mode (subcommand 0 x 03) can be issued to set the drives on the system to the optimal speeds:

Table 10-11. Ultra DMA/Multi Word DMA/Single Word Transfer/Mode Values

Drive's Selected Ultra DMA Capability	Drive's Selected Non-ultra DMA Capability	Selected Speed	ATA SET_FEATURES - Command Set Transfer Mode Sub Command Parameter for Selected Speed
Ultra DMA Mode 2	any	Ultra DMA Mode 2	42h
Ultra DMA Mode 1	any	Ultra DMA Mode 1	41h
Ultra DMA Mode 0	any	Ultra DMA Mode 0	40h
N/A	Multi Word DMA Mode 2	Multi Word DMA Mode 2	22h
N/A	Multi Word DMA Mode 1	Multi Word DMA Mode 1	21h
N/A	Single Word DMA Mode 2	Single Word DMA Mode 2	12h
N/A	N/A	Disabled	N/A

Refer to the Set Features Command description in the ATA Specification for more information.

Table 10-12. PIO Transfer/Mode Values

Drive's Selected PIO Speed Capability	ATA SET_FEATURES -Command Set Transfer Mode Sub Command Parameter for Selected Speed
PIO0/PIO1/PIO2/Compatible	N/A
PIO3 w/IORDY Flow Control	C3
PIO4 w/IORDY Flow Control	C4

A drive may only be enabled for a Single DMA capability. In general, if a drive supports a supported Ultra DMA speed, then Ultra DMA is configured for the drive. If a drive does not support Ultra DMA, then it should be configured with it's selected DMA speed, if it exists. If a drive supports only PIO (does not have support for either Ultra DMA or DMA speeds), then the drive shall only be accessed in a PIO mode only.

If a drive is configured for an Ultra DMA speed or a DMA speed, its corresponding DMA-capable bit in the PCI Bus Master I/O Status Register (Primary: PCI Bus Master IDE I/O Offset + 02h; Secondary - PCI Bus Master I/O Offset + 0Ah) MUST be set. This will allow PCI Bus Master IDE capable device drivers to recognize the fact that this drive has been identified and configured by the firmware for PCI Bus Master IDE operation.

10.5.7.1 BMIS1 - Bus Master IDE Status Register 1 (Primary: Bus Master IDE Base I/O Address + Offset 02h)

10.5.7.2 BMIS2 - Bus Master IDE Status Register 2 (Secondary: Bus Master IDE Base I/O Address + Offset 0Ah)

7	6	5	4	3	2	1	0
Reserved	Drive 1 DMA Capable (DMACAP1)	Drive 0 DMA Capable (DMACAP0)	Reserved		IDE Interrupt Status (IDEINTS)	IDE DMA Error	Bus Master IDE Active (BMIDEA)
	0: Drive is PIO only. 1: Drive is capable and configured for DMA transfers.	0: Drive is PIO only. 1: Drive is capable and configured for DMA transfers.			0: R/W/C Software writes 1' to clear.	0: R/W/C Software writes 1' to clear.	0: RO

- The Drive 0 DMA Capable bit shall be:
  - — Set to '1' when the Drive 0 (Master) has been identified and configured for DMA transfers (Ultra DMA, Multi Word DMA or Single Word DMA).
  - — Set to '0' if Drive 0 is PIO only and/or not configured for DMA operation.
- The Drive 1 DMA Capable bit shall be:
  - — Set to '1' when the Drive 1 (Slave) has been identified and configured for DMA transfers.
  - — Set to '0' if Drive 1 is PIO only and/or not configured for DMA operation.

It is the responsibility of initialization software to ensure that these DMA capable bits are set so that a PCI Bus Master IDE device driver can determine which drives have been configured for DMA operation.

10.5.8 Settings Checklist

The following checklists can be used in determining drive modes. Refer to the "Determining a Drive's Transfer Rate Capabilities" and "IFB Timing Settings" sections for more information.

Table 10-13. Drive Capabilities Checklist

Drive	Type (ATA Fixed Disk or ATAPI)	Position	Best Ultra DMA Mode (Ultra DMA Mode 0, 1, 2, or N/A)	Best DMA Mode (Single Word 2, Multi Word 1, 2, or N/A)	Best PIO Mode (Fast PIO Mode 2, 3, 4 or Compatible)	IFB Ultra DMA Mode	IFB Mode	Non Ultra DMA Supported? Best DMA Mode is {SW2, MW1, MW2}	Fast PIO Supported? Best PIO Mode >= Best DMA Mode
Drive 0		Primary Master							
Drive 1		Primary Slave							
Drive 2		Secondary Master							
Drive 3		Secondary Slave							

Table 10-14. IFB Settings Checklist

Register	Type	Offset	Value	Comments
PCI Command Register	PCI	04h	0005h	Ensure that bits 0 and 2 are ‘1’
PCI Master Latency Timer	PCI	0Dh		
PCI Bus Master IDE Base I/O Address	PCI	20-23h		Ensure that bit 0 (of register value) is ‘1’
IDE Timing Register 1	PCI	40-41h		
IDE Timing Register 2	PCI	42-43h		
Secondary IDE Timing Register	PCI	44h		
Ultra DMA Control Register	PCI	48h		
Ultra DMA Timing Register	PCI	4A-4Bh		

10.5.9 Example Configurations

This section provides examples of drive configurations on a IFB-based system.

10.5.9.1 Example #1: Ultra DMA/33 Configuration

Drive	Type	Position	Best Ultra DMA Mode	Best DMA Mode	Best PIO Mode	IFB Ultra DMA Mode	IFB Mode	Non Ultra DMA Supported? Best DMA Mode is {SW2, MW1, MW2}	Fast PIO Supported? Best PIO Mode >= Best DMA Mode
Drive 0	Fixed Disk	Primary Single	Ultra DMA Mode 2	Multi Word DMA Mode 2	PIO4	Ultra DMA Mode 2	Mode 4	yes	yes
Drive 2	ATAPI CDROM	Secondary Single	Ultra DMA Mode 1	Multi Word DMA Mode 1	PIO3	Ultra DMA Mode 1	Mode 3	yes	yes

In the above configuration, since both drives support Ultra DMA, Ultra DMA will be enabled on each of the drives: Ultra DMA Mode 2 for Drive 0 and Ultra DMA Mode 1 for Drive 1. Non-ultra DMA and Fast PIO support will be enabled on each drive as well.

Register	Type	Offset	Value	Comments
PCI Command Register	PCI	04h	0005h	Ensure that bits 0 and 2 are ‘1’.
PCI Master Latency Timer	PCI	0Dh	system dependent	
PCI Bus Master IDE Base I/O Address	PCI	20-23h	system dependent	Ensure that bit 0 (of register value) is ‘1’.
IDE Timing Register 1	PCI	40-41h	A307h	mode config. for Primary
IDE Timing Register 2	PCI	42-43h	A303h	mode config. for Secondary
Secondary IDE Timing Register	PCI	44h	00h	
Ultra DMA Control Register	PCI	48h	05h	Drive 0 and 2 are Ultra DMA capable.
Ultra DMA Timing Register	PCI	4A-4Bh	0102h	Ultra DMA mode config.

10.5.9.2 Example #2: Mixed Ultra DMA/33 and Non-ultra DMA/33 Configuration

Drive	Type	Position	Best Ultra DMA Mode	Best DMA Mode	Best PIO Mode	IFB Ultra DMA Mode	IFB Mode	Non Ultra DMA Supported? Best DMA Mode is {SW2, MW1, MW2}	Fast PIO Supported? Best PIO Mode >= Best DMA Mode
Drive 0	Fixed Disk	Primary Master	Ultra DMA Mode 2	Multi Word DMA Mode 2	PIO4	Ultra DMA Mode 2	Mode 4	yes	yes
Drive 1	Fixed Disk	Primary Slave	N/A	Multi Word DMA Mode 2	PIO4	N/A (Ultra DMA disabled)	Mode 4	yes	yes
Drive 2	ATAPI CDROM	Secondary Single	N/A	Multi Word DMA Mode 1	PIO3	N/A (Ultra DMA disabled)	Mode 3	yes	yes

In the above configuration, Ultra DMA Mode 2 will only be enabled on Drive 0. Non-ultra DMA and Fast PIO support will be enabled on each drive as well.

Register	Type	Offset	Value	Comments
PCI Command Register	PCI	04h	0005h	Ensure that bits 0 and 2 are ‘1’.
PCI Master Latency Timer	PCI	0Dh	System dependent	
PCI Bus Master IDE Base I/O Address	PCI	20-23h	System dependent	Ensure that bit 0 (of register value) is ‘1’.
IDE Timing Register 1	PCI	40-41h	E377h	Mode config. for Primary
IDE Timing Register 2	PCI	42-43h	A103h	Mode config. for Secondary
Secondary IDE Timing Register	PCI	44h	0Bh	
Ultra DMA Control Register	PCI	48h	01h	Drive 0 is Ultra DMA capable.
Ultra DMA Timing Register	PCI	4A-4Bh	0002h	Ultra DMA mode config.

10.5.9.3 Example #3: Non Ultra DMA/33 Drive Configuration

Drive	Type	Position	Best Ultra DMA Mode	Best DMA Mode	Best PIO Mode	IFB Ultra DMA Mode	IFB Mode	Non Ultra DMA Supported? Best DMA Mode is {SW2, MW1, MW2}	Fast PIO Supported? Best PIO Mode >= Best DMA Mode
Drive 0	Fixed Disk	Primary Master	N/A	Multi Word DMA Mode 2	PIO4	N/A (Ultra DMA disabled)	Mode 4	yes	yes
Drive 1	Fixed Disk	Primary Slave	N/A	Multi Word DMA Mode 2	PIO4	N/A (Ultra DMA disabled)	Mode 4	yes	yes
Drive 2	ATAPI CDROM	Secondary Single	N/A	Multi Word DMA Mode 1	PIO3	N/A (Ultra DMA disabled)	Mode 3	yes	yes

In the above configuration, none of the drives supports Ultra DMA. Only Non-ultra DMA and Fast PIO support will be enabled on each drive.

Register	Type	Offset	Value	Comments
PCI Command Register	PCI	04h	0005h	Ensure that bits 0 and 2 are '1'.
PCI Master Latency Timer	PCI	0Dh	System dependent	
PCI Bus Master IDE Base I/O Address	PCI	20-23h	System dependent	Ensure that bit 0 (of register value) is '1'.
IDE Timing Register 1	PCI	40-41h	E377h	Mode config. for Primary
IDE Timing Register 2	PCI	42-43h	A103h	Mode config. for Secondary
Secondary IDE Timing Register	PCI	44h	0Bh	
Ultra DMA Control Register	PCI	48h	00h	Ultra DMA is disabled for all drives.
Ultra DMA Timing Register	PCI	4A-4Bh	0000h	

10.5.10 Ultra DMA System Software Considerations

This section outlines some of the key system considerations for systems where Ultra DMA operation is enabled.

The following components shall comprise a Ultra DMA/33 System:

1. Ultra DMA/33-capable Host IDE Controller.
2. Ultra DMA/33-capable ATA/ATAPI Devices.
3. Ultra DMA/33 Aware firmware and /or System Initialization Software.
4. Ultra DMA/33 Aware Device Driver.

Two of the aforementioned components are software based with specific requirements: Item #3, Ultra DMA/33 Aware firmware and/or System Initialization Software; and Item #4, Ultra DMA/33 Aware Device Driver .

Ultra DMA/33 Aware firmware and/or System Initialization Software shall:

1. Identify Ultra DMA/33 capable devices and host controllers.
2. Configure Ultra DMA/33 operation for all Ultra DMA/33 capable devices and host controllers.
3. Preserve Ultra DMA/33 configuration across reset states, restoring Ultra DMA/33 operation (described in Item #2) as necessary.

Ultra DMA/33 Aware Device Drivers shall:

I. Provide support for PCI Bus Master IDE Operation (SFF8038i).

   A. Identify system configured for PCI Bus Master IDE operation.

      1. Identify PCI Bus Master IDE and Ultra DMA/33 capable devices and host controllers.
      2. Utilize PCI Bus Master IDE when it determines that Host Controller and ATA/ATAPI devices have fulfilled device driver-specific, configuration requirements for PCI Bus Master IDE operation.

   B. Perform data transfer commands with PCI Bus Master IDE on devices, host controllers that support PCI Bus Master IDE.

II. Provide recovery for data transfers that fail as the result of Ultra DMA/33 Interface CRC Errors:

   A. Determine that the data transfer command's error source is Ultra DMA/33 Interface CRC error.

   B. Retry data transfer command when Ultra DMA/33 Interface CRC is the source of error.

III. Ensure that the Ultra DMA/33 configuration of the devices and host controller is restored when events that clear the Ultra DMA/33 enabled status are encountered.

- Ensure that Hard Resets are never issued to the device during a power cycle -OR-
- Provide path to Ultra DMA/33 Aware firmware and/or System Initialization Software in the case of a Hard/Power-On reset.

When enabled on the host controller and devices, Ultra DMA/33 operation shall be used for all data transfer commands issued by the Ultra DMA/33 Aware Device Driver with PCI Bus Master IDE, DMA operation. PIO or Multi Word DMA shall be the mode of access used with devices and host controllers that do not support Ultra DMA/33.

10.5.11 Additional Ultra DMA/PCI Bus Master IDE Device Driver Considerations

This section provides information regarding Terminating Transfers performed with Ultra DMA/ PCI Bus Master IDE device drivers or system software.

In normal bus master operations, at the end of a data transfer, the IDE device signals an interrupt. In response to the interrupt, software verifies that the bus is idle and then writes the Stop Bus Master Command. It then reads the controller status register to determine if the transfer completed successfully. For a detailed description of the Bus Master IDE Status Register refer to the last section of this document called *Bus Master IDE Command and Status Registers*.

If the IDE device does not signal the interrupt, the last bus master transfer did not complete. In this case, it is necessary to read the Bus Master IDE Status Register to check if the bus is idle or active. If the bus is active it is necessary to send the Stop Bus Master Command and reset the IDE controller and drives connected to the IDE cable prior to sending out the next ATA/ATAPI drive command to the cable. This is necessary because the cable (Primary or Secondary IDE) may not be ready to receive new commands unless the Bus Master state machine has stopped. All the drives on the cable should be reset.

In general, a prematurely terminated command on the IDE bus implies that some of the state machines in the drive and/or in the IFB are still in an "active" condition. By performing a drive reset immediately following the burst "stop", the IFB will be in a state such that the IDE DMA engines can be programmed to perform the transfer once again.

10.5.11.1 Bus Master IDE Command and Status Register

10.5.11.2 BMICX–Bus Master IDE Command Register (I/O)

Address Offset:	Primary Channel–Base + 00h; Secondary Channel–Base + 08h
Default Value:	00h
Attribute:	Read/Write

This register enables/disables bus master capability for the IDE function and provides direction control for the IDE DMA transfers. This register also provides bits that software uses to indicate DMA capability of the IDE device.

Bit	Description
7:4	Reserved.
3	**Bus Master Read/Write Control (RWCON).** 0=Reads; 1=Writes. This bit must NOT be changed when the bus master function is active. While a Ultra DMA transfer is in progress, this bit will be READ ONLY. The bit will return to read/write once the synchronous DMA transfer has been completed or halted.
2:1	**Reserved**.
0	**Start/Stop Bus Master (SSBM).** 1=Start; 0=Stop. When this bit is set to 1, bus master operation starts. The controller transfers data between the IDE device and memory only while this bit is set. Master operation can be stopped by writing a 0 to this bit. This results in all state information being lost (i.e. master mode operation cannot be stopped and then resumed). If this bit is set to 0 while bus master operation is still active (i.e. Bit 0=1 in the Bus Master IDE Status Register for that IDE channel) and the drive has not yet finished its data transfer (bit 2=0 in the channel's Bus Master IDE Status Register), the bus master command is aborted and data transferred from the drive may be discarded by IFB rather than being written to system memory. This bit is intended to be set to 0 after the data transfer is completed, as indicated by either bit 0 or bit 2 being set in the IDE Channel's Bus Master IDE Status Register.

10.5.11.3 BMISX–Bus Master IDE Status Register (I/O)

Address Offset: Primary Channel–Base + 02h; Secondary Channel–Base + 0Ah
Default Value: 00h
Attribute: Read/Write Clear

This register provides status information about the IDE device and state of the IDE DMA transfer.

Bit	Description
7	**Reserved.** This bit is hardwired to 0.
6	**Drive 1 DMA Capable (DMA1CAP)–R/W.** 1=Drive 1 is capable of DMA transfers. This bit is a software controlled status bit that indicates IDE DMA device capability and does not affect hardware operation.
5	**Drive 0 DMA Capable (DMA0CAP)–R/W.** 1=Drive 0 is capable of DMA transfers. This bit is a software controlled status bit that indicates IDE DMA device capability and does not affect hardware operation.
4:3	Reserved.
2	**IDE Interrupt Status (IDEINTS)–R/WC.** This bit, when set to a 1, indicates when an IDE device has asserted its interrupt line. When bit 2=1, all read data from the IDE device has been transferred to main memory and all write data has been transferred to the IDE device. Software sets this bit to a 0 by writing a 1 to it. IRQ14 is used for the primary channel and IRQ15 is used for the secondary channel. Note that, if the interrupt status bit is set to a 0 by writing a 1 to this bit while the interrupt line is still at the active level, this bit remains 0 until another assertion edge is detected on the interrupt line.
1	**IDE DMA Error–R/WC.** This bit is set to 1 when IFB encounters a target abort or master abort while transferring data on the PCI Bus. Software sets this bit to a 0 by writing a 1 to it.
0	**Bus Master IDE Active (BMIDEA)–RO.** IFB sets this bit to 1 when bit 0 in the BMICx Register is set to 1. IFB sets this bit to 0 when the last transfer for a region is performed (where EOT for that region is set in the region descriptor). IFB also sets this bit to 0 when bit 0 of the BMICx Register is set to 0. When this bit is read as a zero, all data transferred from the drive during the previous bus master command is visible in system memory, unless the bus master command was aborted.

Interrupt/Activity Status Combinations

Bit 2	Bit 0	Description
0	1	**DMA transfer is in progress.** No interrupt has been generated by the IDE device.
1	0	**The IDE device generated an interrupt and the Physical Region Descriptors exhausted.** This is normal completion where the size of the physical memory regions is equal to the IDE device transfer size.
1	1	**The IDE device generated an interrupt.** The controller has not reached the end of the physical memory regions. This is a valid completion case when the size of the physical memory regions is larger than the IDE device transfer size.
0	0	**Error condition.** If the IDE DMA Error bit is 1, there is a problem transferring data to/ from memory. Specifics of the error have to be determined using bus-specific information. If the Error bit is 0, the PRD specified a smaller buffer size than the programmed IDE transfer size.

10.6 USB Resume Enable Bit

Two bits have been added to the USB Host controller functionality in function 2 of IFB (PCI Register configuration space at Offset C4h). This register is in the resume well of this function.

USB Resume Enable: (IFB Function 2 Config Register)

Address Offset: C4h
Attribute: Read/Write
Default Value: 00h
Size: 8 bits

Bit	Type	Description
7:2	RO	Reserved
1	RW	**PORT1EN:** Enable port 1 of the USB controller to look at wakeup events. When set, the USB controller will monitor port 1 for remote wakeup and connect/disconnect events. When cleared, the USB controller will not look at this port for a wakeup event. For function 2, this bit applies to port 1.
0	RW	**PORT0EN:** Enable port 0 of the USB controller to look at wakeup events. When set, the USB controller will monitor port 0 for remote wakeup and connect/disconnect events. When cleared, the USB controller will not look at this port for a wakeup event. For function 2, this bit applies to port 0.

For performing legacy power management, the firmware has to set these two bits in each of the functions, if it wants the USB Host controller to monitor these ports.

LPC/FWH Interface Configuration 11

The IFB PCI Function 0 contains a LPC/FWH interface, interrupt controller and counter / timers, including the real time clock. The register set associated with this Functionality and associated logic is shown below with actual register descriptions given in this section.

11.1 PCI to LPC/FWH Interface Configuration Space Registers (PCI Function 0)

11.1.1 VID–Vendor Identification Register (Function 0)

Address Offset: 00–01h
Default Value: 8086h
Attribute: Read Only

The VID Register contains the vendor identification number. This register, along with the Device Identification Register, uniquely identifies any PCI device. Writes to this register have no effect.

Bit	Description
15:0	**Vendor Identification Number.** This is a 16-bit value assigned to Intel.

11.1.2 DID–Device Identification Register (Function 0)

Address Offset: 02–03h
Default Value: 7600h
Attribute: Read Only

The DID Register contains the device identification number. This register, along with the VID Register, defines the IFB. Writes to this register have no effect.

Bit	Description
15:0	**Device Identification Number.** This is a 16-bit value assigned to the IFB.

11.1.3 PCICMD–PCI Command Register (Function 0)

Address Offset: 04–05h
Default Value: 0007h
Attribute: Read/Write

This 16-bit register provides basic control over the IFB's ability to respond to PCI cycles.

Bit	Description
15:10	Reserved.
9	**Fast Back-to-Back Enable (Not Implemented).** This bit is hardwired to 0.
8	**SERR# Enable (SERRE).** 1=Enable. 0=Disable. When enabled (and DLC Register, bit 3=1), a delayed **transaction** time-out causes the IFB to assert the SERR# signal. The PCISTS register reports the status of the SERR# signal.
7:5	Reserved.
4	**Postable Memory Write Enable (Not Implemented).** This bit is hardwired to 0.
3	**Special Cycle Enable (SCE)**. 1=Enable, the IFB recognizes Shutdown special cycle. 0=Disable, the IFB ignores all PCI **Special** Cycles.
2	**Bus Master Enable (Not Implemented).** The IFB does not support disabling its Function 0 bus master capability. This bit is hardwired to 1.
1	**Memory Access (Not Implemented).** The IFB does not support disabling Function 0 access to memory. This bit is hardwired to 1.
0	**I/O Space Access** Enable **(Not Implemented).** The IFB does not support disabling its Function 0 response to PCI I/O cycles. This bit is hardwired to 1.

11.1.4 PCISTS–PCI Device Status Register (Function 0)

Address Offset: 06–07h
Default Value: 0280h
Attribute: Read/Write

The PCISTS Register reports the occurrence of a PCI master-abort by the IFB or a PCI target-abort when the IFB is a master. The register also indicates the IFB DEVSEL# signal timing.

Bit	Description
15	**Detected Parity Error (Not Implemented).** Read as 0.
14	**Signaled SERR# Status (SERRS)–R/WC.** When the IFB asserts the SERR# signal, this bit is set to 1. Software clears this bit by **writing** a 1 to it.
13	**Master-Abort Status (MAS)–R/WC.** When the IFB, as a master (for Function 0), generates a master-abort, MAS is set to a 1. Software sets MAS to 0 by writing a 1 to this bit location.
12	**Received Target-Abort Status (RTA)–R/WC.** When the IFB is a master on the PCI Bus (for Function 0) and receives a target-abort, this bit is set to a 1. Software sets RTA to 0 by writing a 1 to this bit location.
11	**Signaled Target-Abort** Status **(STA)–R/WC.** This bit is set when the IFB LPC bridge Function is targeted with a transaction that the IFB terminates with a target abort. Software sets STA to 0 by writing a 1 to this bit location.
10:9	**DEVSEL# Timing Status (**DEVT**)–RO.** The IFB always generates DEVSEL# with medium timing for Function 0 I/O cycles. Thus, DEVT=01. This DEVSEL# timing does not include Configuration cycles.

Bit	Description
8	**PERR# Response (Not** Implemented**).** Read as 0.
7	**Fast Back to Back–RO.** This bit indicates to the PCI Master that IFB as a target is capable of accepting fast back-to-back transactions. This bit is hardwired to 1.
6:0	Reserved.

11.1.5 RID–Revision Identification Register (Function 0)

Address Offset: 08h
Default Value: Stepping Dependent
Attribute: Read Only

This 8 bit register contains device stepping information. Writes to this register have no effect.

Bit	Description
7:0	**Revision ID Byte.**

11.1.6 CLASSC–Class Code Register (Function 0)

Address Offset: 09h-0Bh
Default Value: 060100h
Attribute: Read Only

This register identifies the Base Class Code, Sub-class Code, and Device Programming interface for the IFB PCI Function 0.

Bit	Description
23:16	**Base Class Code (**BASEC**).** 06h=Bridge device.
15:8	**Sub-Class Code (SCC)**. 01h=PCI-to-ISA Bridge. ISA is not supported, IFB forwards cycles to the LPC interface.
7:0	**Programming Interface (PI).** 00h=No register level programming interface defined.

11.1.7 HEDT–Header Type Register (Function 0)

Address Offset: 0Eh
Default Value: 80h
Attribute: Read Only

The HEDT Register identifies the IFB as a multi-Function device.

Bit	Description
7:0	**Device Type (DEVICET).** 80h=multi-Function device.

11.1.8 ACPI Base Address (Function 0)

Address: 40-43h
Default Value: 00000001h
Attribute: Read/Write

Bit	Description
31:16	Reserved.
15:6	**Base Address:** Provides the 64 bytes of I/O space.
5:1	Reserved.
0	**Resource Indicator:** Tied to 1 to indicate I/O space

11.1.9 ACPI Enable (Function 0)

Address: 44h
Default Value 00
Attribute: Read/Write

Bit	Description
7:1	Reserved.
0	**ACPI Enable:** When this bit is set to '1', decode of the I/O range pointed to by the ACPI base register is enabled, and the ACPI power management Function is enabled. Note that the APM power management ranges (B2/B3h) are always enabled and are not affected by this bit.

11.1.10 SCI IRQ Routing Control

Address: 45h
Default Value: 00h
Attributes: Read/Write

Bit	Description	
7:3	Reserved.	
2:0	**SCI IRQ Map:** Specifies on which pin the SCI will appear on internally. If not using the APIC, software must program this register to "000". This interrupt is not sharable. When this interrupt is chosen, the corresponding interrupt pin is blocked and the SERIRQ frame entry is blocked. This interrupt can be shared with a PCI interrupt.	
	Bits	SCI Map
	000	IRQ9
	001	SCI
	010	FEMPTY#
	011	IRQ8
	100	IRQ0
	101	IRQ10
	110	IRQ11
	111	Reserved

11.1.11 BIOSEN–BIOS Enable Register (FUNCTION 0)

Address Offset: 4E-4Fh
Default Value: 07C1h
Attribute: Read/Write

This register is used to implement protections to writes to firmware (BIOS) ranges.

Bit	Description
15	**BLE - BIOS Lock Enable:** When the bit is a “1”, setting BIOS_WEN bit will cause SMIs. When this bit is a “0”, setting BIOS_WEN will not cause SMIs. Once set, this bit can only be cleared by a PCIRST#.
14:11	**Reserved.**
10:3	**Reserved.** Software must preserve these register values.
2	**BIOS_WEN - BIOS Write Enable.** When this bit is set to a “1”, writes to firmware (BIOS) ranges are allowed. When this bit is a “0”, writes to firmware (BIOS) ranges are not allowed and write cycles will be master-aborted on PCI. There is no protection on writes to FWH register space. When this bit is written from a “0” to a “1”, and bit 15 (BIOS Lock Enable) is set, an SMI# is generated.
1:0	**Reserved.** Software must preserve these register values.

11.1.12 PIRQRC[A:D]–PIRQx Route Control Registers (Function 0)

Address Offset: 60h (PIRQRCA#)–63h (PIRQRCD#)
Default Value: 80h
Attribute: R/W

These registers control the routing of the PIRQ[A:D]# signals to the IRQ inputs of the interrupt controller. Each PIRQx# can be independently routed to any one of 11 interrupts. All four PIRQx# lines can be routed to the same IRQx input. Note that the IRQ that is selected through bits [3:0] must be set to level sensitive mode in the corresponding ELCR Register. When a PIRQ signal is routed to an interrupt controller IRQ, the IFB masks the corresponding IRQ signal.

Bit	Description
7	**Interrupt Routing Enable.** When this bit is a “0”, the corresponding PIRQ is routed to one of the compatibility mode interrupts specified in bits[3:0]. When this bit is a “1”, the PIRQ is not routed.
6:4	**Reserved.**
3:0	**Interrupt Routing.** When bit 7=0, this field selects the routing of the PIRQx to one of the interrupt controller interrupt inputs.

Bits[3:0]	IRQ Routing	Bits[3:0]	IRQ Routing	Bits[3:0]	IRQ Routing
0000	Reserved	0110	IRQ6	1011	IRQ11
0001	Reserved	0111	IRQ7	1100	IRQ12
0010	Reserved	1000	Reserved	1101	Reserved
0011	IRQ3	1001	IRQ9	1110	IRQ14
0100	IRW4	1010	IRQ10	1111	IRQ15
0101	IRQ5				

11.1.13 SerIRQC–Serial IRQ Control Register (Function 0)

Address Offset: 64h
Default Value: 10h
Attribute: R/W

This register controls the Start Frame Pulse Width generated on the Serial Interrupt signal (SERIRQ).

Bit	Description
7	**Serial IRQ Enable.** 1=Serial Interrupts are enabled. 0=Serial Interrupts disabled.
6	**Serial IRQ Mode Select.** When this bit is a “1”, the serial IRQ machine will be in continuous mode. When this bit is a “0”, the serial IRQ machine will be in quiet mode. When setting the SIRQEN bit, this bit must also be written as a one so that the first action of the serial IRQ machine will be a start frame.
5:2	**Serial IRQ Frame Size.** These bits select the frame size used by the Serial IRQ logic. The default is 0100b indicating a frame size of 21 (17+4). These bits are readable and writeable, however the only programmed value supported by the IFB is 0100b. All other frame sizes are unsupported.
1:0	**Start Frame Pulse Width.** These bits define the Start Frame pulse width generated by the Serial Interrupt control logic. **Bits[1:0] Pulse Width (PCI Clocks)** 00 4 Clocks 01 6 Clocks 10 8 Clocks 11 Reserved

11.1.14 TOM–Top of Memory Register (Function 0)

Address Offset: 69h
Default Value: 02h
Attribute: Read/Write

This register enables the forwarding of DMA memory cycles to the PCI Bus and sets the top of main memory accessible by DMA devices. In addition, this register controls the forwarding of DMA accesses to the lower firmware region (E0000–EFFFFh) and the 512–640 KByte main memory region (80000–9FFFFh).

Bit	Description
7:4	**Top Of Memory.** The top of memory can be assigned in 1 Mbyte increments from 1–16 Mbytes. DMA accesses within this region, and not in the memory hole region, are forwarded to PCI. **Bits[7:4] / Top of Memory / Bits[7:4] / Top of Memory / Bits[7:4] / Top of Memory** 0000 1 Mbyte 0110 7 Mbyte 1011 12 Mbyte 0001 2 Mbyte 0111 8 Mbyte 1100 13 Mbyte 0010 3 Mbyte 1000 9 Mbyte 1101 14 Mbyte 0011 4 Mbyte 1001 10 Mbyte 1110 15 Mbyte 0100 5 Mbyte 1010 11 Mbyte 1111 16 Mbyte 0101 6 Mbyte Note that the IFB only supports a main memory hole at the top of 16 Mbytes. Thus, If a 1 Mbyte memory hole is created for the Host-to-PCI Bridge DRAM controller between 15 and 16 Mbytes, the IFB Top Of Memory should be set at 15 Mbytes.
3	**Reserved.** Must be set to ‘0’.
2	**Reserved.** Must be set to ‘0’.
1	**Reserved.** Must be set to ‘0’.
0	Reserved.

11.1.15 MSTAT–Miscellaneous Status Register (Function 0)

Address Offset: 6A–6Bh
Default Value: 0000h
Attribute: Read/Write

This register provides miscellaneous status and control Functions.

Bit	Description
15	**SERR# Generation Due To Delayed Transaction Time-out–R/WC.** This status bit is set whenever the IFB times out a cycle it was running as a delayed transaction, bit 3 (Delayed Transaction SERR# enable) of offset 82h is set, and bit 8 (SERR# enable) of offset 4h is set. When a master does not return for the data within 1 ms of the cycle's completion, the IFB asserts SERR#, clears the delayed transaction, and sets this bit. If either bit 3 of 82h or bit 8 of 4h is not set then this bit will not get set and SERR# will not be generated. The IFB will still discard the delayed transaction cycle. The bit can be cleared by writing "1" to it via software.
14:1	Reserved.
0	**ECC SERR# Enable:** When this bit is a "1", it enables a one clock pulse on ECCINT# to instead be routed to generate SERR#. This allows ECC scrubbing through SMI (greater than one clock) or NMI (one clock).

11.1.16 Deterministic Latency Control Register (Function 0)

Address Offset: 82h
Default Value: 00h
Attribute: Read/Write

This register enables and disables the Delayed Transaction and Passive Release Functions. When enabled, these Functions make the IFB PCI revision 2.1 compliant.

The PCI specification requires much tighter controls on target and master latency. Targets must respond with TRDY# or STOP# within 16 clocks of FRAME#, and masters must assert IRDY# within 8 PCI clocks for any data phase. PCI cycles to or from LPC typically take longer than this. The IFB provides a programmable delayed completion mechanism described in the PCI specification to meet the required target latencies.

Bit	Description
7:4	Reserved.
3	**SERR# Generation Due To Delayed Transaction Time-out Enable.** When this bit is set, and bit 8 (SERR# enable) of the Command Register (offset 4h) is set, the IFB will generate SERR# when a delayed transaction cycle times out.
2	**USB Passive Release Enable (USBPR).** When this bit is set, the IFB enables the passive release mechanism for USB. Normally, if the IFB gets retried under USB, it will not remove PHOLD#. This bit is needed due to posting of USWC cycles, which means a retry might be due to a USWC flush.
1	**Passive Release Enable.** When this bit is a "1", it enables the Passive Release mechanism encoded on the PHOLD# signal when IFB is a PCI Master. When this bit is a "0", Passive Release is disabled.
0	**Delayed Transaction Enable.** When this bit is a "1", it enables the Delayed Transaction mechanism when the IFB is the target of a PCI transaction. When this bit is a "0", the Delayed Transaction mechanism is disabled.

11.1.17 MGPIOC–Muxed GPIO Control (Function 0)

Offset: 84-85h
Default Value: 0500h
Attribute: Read/Write

Bit	Description
16:13	Reserved.
12	**Reserved.** Must be set to ‘1’.
11	Reserved.
10	**Reserved.** Must be set to ‘1’.
9	Reserved.
8	**Reserved.** Must be set to ‘1’.
7	**Reserved.** Must be set to ‘1’.
6	**Reserved.** Must be set to ‘1’.
5	**Reserved.** Must be set to ‘1’.
4	**Reserved.** Must be set to ‘1’.
3:0	Reserved.

11.1.18 PDMACFG–PCI DMA Configuration Resister (Function O)

Address Offset: 90-91h
Default Value: 0000h
Attribute: Read/Write

Bits	Description
15:14	**Reserved**. Must be set to ‘11’.
13:12	**Reserved**. Must be set to ‘11’.
11:10	**Reserved**. Must be set to ‘11’.
9:8	Reserved.
7:6	**Reserved**. Must be set to ‘11’.
5:4	**Reserved**. Must be set to ‘11’.
3:2	**Reserved**. Must be set to ‘11’.
1:0	**Reserved**. Must be set to ‘11’.

11.1.19 DDMABP–Distributed DMA Slave Base Pointer Registers (Function 0)

Address Offset: 92-93h (CH0-3), 94-95h (CH5-7)
Default Value: 0000h
Attribute: Read/Write

These registers provide the base address for distributed DMA slave channel registers, one for each DMA controller. Bits 5:0 are reserved to provide access to a 64 byte I/O space (16 bytes per channel). The channels are accessed using offset from base address as follows (Note that Channel 4 is reserved and is not accessible).

Base Offset	Channel
00 - 0Fh	0,4
10 - 1Fh	1,5
20 - 2Fh	2,6
30 - 3Fh	3,7

Bits	Description
15:6	**Base Pointer: I/O** Address pointer to DMA Slave Channel registers. Corresponds to PCI address AD[15:6].
5:0	Reserved.

11.1.20 RTCCFG–Real Time Clock Configuration Register (Function 0)

Address Offset: C8h
Default Value: 00h
Attribute: Read/Write

This register is used to configure the internal Real Time Clock.

Bits	Description
7:5	Reserved.
4	**Lock Upper RAM Bytes:** 0 = Upper RAM data bytes 38h-3Fh in the extended bank are readable and write-able (default). 1 = Upper RAM data bytes 38h-3Fh in the extended bank are neither readable nor write-able. This is used to lock bytes 38h-3Fh in the upper 128-byte bank of RAM. Write cycles will have no effect and read cycles will not return an expected value. **Warning:** This is a write-once register that can only be reset by a hardware reset. No software means is possible to reset this bit.
3	**Lock Lower RAM Bytes:** 0 = Lower RAM data bytes 38h-3Fh in the standard bank are readable and writeable (default). 1 = Lower RAM data bytes 38h-3Fh in the standard bank are neither readable nor writeable. This is used to lock bytes 38h-3Fh in the lower 128-byte bank of RAM. Write cycles will have no effect and read cycles will not return an expected value. **Warning:** This is a write-once register that can only be reset by a hardware reset. No software means is possible to reset this bit.
2	**Upper RAM Enable:** 0 = Accesses to RTC Upper 128 byte extended bank at I/O address 72-73h is disabled. 1 = Accesses to 72-73h are forwarded to RTC Upper 128 byte extended bank.
1:0	Reserved.

11.1.21 GPIO Base Address (FUNCTION 0)

Address: D0-D3h
Default Value: 00000001h
Attributes: Read/Write

Bit	Description
31:16	Reserved.
15:6	**Base Address:** Provides the 64 bytes of I/O space.
5:1	Reserved.
0	**Resource Indicator:** Tied to 1 to indicate I/O space

11.1.22 GPIO Enable (FUNCTION 0)

Address: D4h
Default Value 00h
Attributes: Read/Write

Bit	Description
7:1	Reserved.
0	**GPIO Enable:** When this bit is set to '1', decode of the I/O range pointed to by the GPIO base register is enabled, and the GPIO Function is enabled.

11.1.23 LPC COM Decode Ranges (Function 0)

Address: E0h
Default Value: 00h
Attributes: Read/Write

Bit	Description
7	Reserved.
6:4	**Decode Range:** The following table describes which range to decode for the COMB Port

Bits	Decode Range
000	3F8 - 3FF (COM 1)
001	2F8 - 2FF (COM 2)
010	220 - 227
011	228 - 22F
100	238 - 23F
101	2E8 - 2EF (COM 4)
110	338 - 33F
111	3E8 - 3EF (COM 3)

Bit	Description
3	Reserved.
2:0	**Decode Range:** The following table describes which range to decode for the COMA Port

Bits	Decode Range
000	3F8 - 3FF (COM 1)
001	2F8 - 2FF (COM 2)
010	220 - 227
011	228 - 22F
100	238 - 23F
101	2E8 - 2EF (COM 4)
110	338 - 33F
111	3E8 - 3EF (COM 3)

11.1.24 LPC FDD/LPT Decode Ranges (Function 0)

Address: E1h
Default Value: 00h
Attributes: Read/Write

Bit	Description
7:5	Reserved.
4	**Decode Range:** The following table describes which range to decode for the FDD Port

Bits	Decode Range
0	3F0 - 3F5, 3F7 (Primary)
1	370 - 375, 377 (Secondary)

3:2	Reserved.
1:0	**Decode Range:** The following table describes which range to decode for the LPT Port

Bits	Decode Range
00	378 - 37F, 778-77F
01	278 - 27F (port 279 is read only), 678-67F
10	3BC - 3BE, 7BC-7BE
11	Reserved.

11.1.25 LPC Sound Decode Ranges (Function 0)

Address: E2h
Default Value: 00h
Attributes: Read/Write

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>7:6</td><td>Reserved.</td></tr>
<tr><td>5:4</td><td>Microsoft Sound System Decode Range: The following table describes which range to decode for the MSS port:
<table>
<tr><th>Bits</th><th>Decode Range</th></tr>
<tr><td>00</td><td>530 - 537</td></tr>
<tr><td>01</td><td>604 - 60B</td></tr>
<tr><td>10</td><td>E80 - E87</td></tr>
<tr><td>11</td><td>F40 - F47</td></tr>
</table></td></tr>
<tr><td>3</td><td>MIDI Decode Range: The following table describes which range to decode for the Midi Port
<table>
<tr><th>Bits</th><th>Decode Range</th></tr>
<tr><td>0</td><td>330 - 331</td></tr>
<tr><td>1</td><td>300 - 301</td></tr>
</table></td></tr>
<tr><td>2</td><td>Reserved.</td></tr>
<tr><td>1:0</td><td>SB16 Decode Range: The following table describes which range to decode for the Sound Blaster 16 Port
<table>
<tr><th>Bits</th><th>Decode Range</th></tr>
<tr><td>00</td><td>220 - 233</td></tr>
<tr><td>01</td><td>240 - 253</td></tr>
<tr><td>10</td><td>260 - 273</td></tr>
<tr><td>11</td><td>280 - 293</td></tr>
</table></td></tr>
</table>

11.1.26 LPC Generic Decode Range (Function 0)

Address: E4-E5h
Default Value: 0000h
Attributes: Read/Write

Bit	Description
15:9	**Base Address:** Base Address for the generic decode range. This address is aligned on a 512 byte boundary, and being I/O, must have address lines 31:16 as “0”.
8:1	Reserved.
0	**Enable:** When set, the range defined by the Base Address register is enabled for decode.

11.1.27 LPC Enables (Function 0)

Address: E6-E7h
Default Value: 0000h
Attributes: Read/Write

Bit	Description
15	**Reserved.** This bit must be a “0”.
14:13	Reserved.
12	**Secondary Configuration Enable:** Enables I/O locations 4Eh and 4Fh to be sent to the LPC bus. Super I/Os use these addresses as an alternate index/data register pair for Super I/O configuration.
11	**Configuration Enable:** Enables I/O locations 2Eh and 2Fh to be sent to the LPC bus. Super I/Os use these addresses as an index/data register pair for Super I/O configuration.
10	**ACPI µController Enable:** This enables decoding of the ports 62h and 66h to the LPC Bus.
9	**MSS Enable:** This enables decoding of the Microsoft Sound System range to the LPC Bus.
8	**Keyboard Enable:** This enables decoding of the keyboard ports at 60h and 64h to the LPC Bus.
7	**Game Port Enable:** This enables decoding of the Game Port range at 200h - 20Fh to the LPC Bus.
6	**ADLIB Enable:** This enables decoding of the ADLIB range at 388h - 38Bh to the LPC Bus.
5	**MIDI Enable:** This enables decoding of the MIDI range to the LPC Bus.
4	**SB16 Enable:** This enables decoding of the SB16 range to the LPC Bus.
3	**FDD Enable:** This enables decoding of the FDD range to the LPC Bus.
2	**LPT Enable:** This enables decoding of the LPT range to the LPC Bus.
1	**COM B Enable:** This enables decoding of the COMB range to the LPC Bus.
0	**COM A Enable:** This enables decoding of the COMA range to the LPC Bus.

11.1.27.1 Firmware Hub (FWH) Decode Enable Register

Address: E3H
Default Value: 00H[1]
Attributes: Read/Write

Bit	Description
7	**FWH_F8_EN:** This enables decoding of 512 KB of the FWH memory range starting at 4 GB – 512 KB (FFF80000H) to the top 4 GB (FFFFFFFFH). In addition, the upper 128 KB of this range is shadowed at the top of 1MB (000E0000H – 000FFFFFH). Additionally, this enables decoding of 512K of register space starting at (4 GB – 4 MB) - 512KB (FFB80000h) to the top 4 GB – 4 MB (FFBFFFFFh).
6	**FWH_F0_EN:** This enables decoding 512 KB of the FWH memory range starting at 4 GB – 1 MB (FFF00000H) to 4 GB – 512 KB (FFF7FFFFH). Additionally, this enables decoding of 512K of register space starting at (4 GB – 4 MB) - 1MB (FFB00000h) to (4 GB – 4 MB) - 512KB (FFB7FFFFh).

1. LFRAME is defined to be tri-stated at reset. This pin is sampled on PWROK. If the pin is sampled as a logic ‘1’, the firmware does not exist, and all the registers defined by E3H are cleared to ‘0’. If the pin is sampled as a logic ‘0’, then the firmware does exist, and all the bits in E3H are set to ‘1’.

Bit	Description
5	**FWH_E8_EN:** This enables decoding 512KB of the FWH memory range starting at 4 GB – 1.5 MB (FFE80000H) to 4 GB – 1 MB (FFEFFFFFH). Additionally, this enables decoding of 512K of register space starting at (4 GB – 4 MB) - 1.5MB (FFA80000h) to (4 GB – 4 MB) - 1MB (FFAFFFFFh).
4	**FWH_E0_EN:** This enables decoding 512 KB of the FWH memory range starting at 4 GB – 2 MB (FFE00000H) to 4 GB - 1.5 MB (FFE7FFFFH). Additionally, this enables decoding of 512K of register space starting at (4 GB – 4 MB) – 2MB (FFA00000h) to (4 GB – 4 MB) – 1.5MB (FFA7FFFFh).
3	**FWH_D8_EN:** This enables decoding 512 KB of the FWH memory range starting at 4 GB - 2.5 MB (FFD80000H) to 4 GB - 2 MB (FFDFFFFFH). Additionally, this enables decoding of 512K of register space starting at (4 GB – 4 MB) - 2.5MB (FF980000h) to (4 GB – 4 MB) - 2 MB (FF9FFFFFh).
2	**FWH_D0_EN:** This enables decoding 512 KB of the FWH memory range starting at 4 GB - 3 MB (FFD00000H) to 4 GB - 2.5 MB (FFD7FFFFH). Additionally, this enables decoding of 512K of register space starting at (4 GB – 4 MB) - 3.0MB (FF900000h) to 4 GB – 4 MB) - 2.5 MB (FF97FFFFh)[8].
1	**FWH_C8_EN:** This enables decoding 512 KB of the FWH memory range starting at 4 GB - 3.5 MB (FFC80000H) to 4 GB - 3 MB (FFCFFFFFH). Additionally, this enables decoding of 512K of register space starting at (4 GB – 4 MB) - 3.5MB (FF880000h) to (4 GB – 4 MB) - 3MB (FF8FFFFFh)[8].
0	**FWH_C0_EN:** This enables decoding 512 KB of the FWH memory range starting at 4 GB – 4 MB (FFC00000H) to 4 GB - 3.5 MB (FFC7FFFFH). Additionally, this enables decoding of 512K of register space starting at (4 GB – 4 MB) – 4MB (FF800000h) to (4 GB – 4 MB) - 3.5MB (FF87FFFFh).

11.1.27.2 Firmware Hub (FWH) Select Register

Address: E8H
Default Value: 00112233H
Attributes: Read/Write

Bit	Description
31:28	**FWH_F8_IDSEL**: This dictates the IDSEL of 512 KB of the FWH memory range starting at 4 GB - 512 KB (FFF80000H) to the top 4 GB (FFFFFFFFH) as well as register space starting at (4 GB-4MB) - 512KB (FFB80000h) to the top 4 GB - 4MB (FFBFFFFFh). In addition, the upper 128 KB of this range is shadowed at the top of 1 MB (000E0000H - 000FFFFFH). The enable for this range is controlled through bit 7 of the FWH Decode Enable Register at E3H.
27:24	**FWH_F0_IDSEL:** This dictates the IDSEL of 512 KB of the FWH memory range starting at 4 GB - 1 MB (FFF00000H) to 4 GB - 512 KB (FFF7FFFFH) as well as register space starting at (4 GB-4MB) - 1MB (FFB00000h) to (4 GB-4MB) - 512KB (FFB7FFFFh). The enable for this range is controlled through bit 6 of the FWH Decode Enable Register at E3H.
23:20	**FWH_E8_IDSEL**: This dictates the IDSEL of 512 KB of the FWH memory range starting at 4 GB - 1.5 MB (FFE80000H) to 4 GB - 1 MB (FFEFFFFFH) as well as register space starting at (4 GB-4MB) - 1.5MB (FFA80000h) to (4 GB-4MB) - 1MB (FFAFFFFFh). The enable for this range is controlled through bit 5 of the FWH Decode Enable Register at E3H.
19:16	**FWH_E0_IDSEL:** This dictates the IDSEL of 512 KB of the FWH memory range starting at 4 GB - 2 MB (FFE00000H) to 4 GB - 1.5 MB (FFE7FFFFH) as well as register space starting at (4 GB-4MB) - 2MB (FFA00000h) to (4 GB-4MB) - 1.5MB (FFA7FFFFh). The enable for this range is controlled through bit 4 of the FWH Decode Enable Register at E3H.
15:12	**FWH_D8_IDSEL**: This dictates the IDSEL of 512 KB of the FWH memory range starting at 4 GB - 2.5 MB (FFD80000H) to 4 GB - 2 MB (FFDFFFFFH) as well as register space starting at (4 GB-4MB) - 2.5MB (FF980000h) to (4 GB-4MB) - 2MB (FF9FFFFFh). The enable for this range is controlled through bit 3 of the FWH Decode Enable Register at E3H.
11:8	**FWH_D0_IDSEL:** This dictates the IDSEL of 512 KB of the FWH memory range starting at 4 GB - 3 MB (FFD00000H) to 4 GB - 2.5 MB (FFD7FFFFH) as well as register space starting at (4 GB-4MB) - 3.0MB (FF900000h) to (4 GB-4MB) - 2.5MB (FF97FFFFh). The enable for this range is controlled through bit 2 of the FWH Decode Enable Register at E3H.

Bit	Description
7:4	**FWH_C8_IDSEL**: This dictates the IDSEL of 512 KB of the FWH memory range starting at 4 GB - 3.5 MB (FFC80000H) to 4 GB - 3 MB (FFCFFFFFH) as well as register space starting at (4 GB-4MB) - 3.5MB (FF880000h) to (4 GB-4MB) - 3MB (FF8FFFFFh). The enable for this range is controlled through bit 1 of the FWH Decode Enable Register at E3H.
3:0	**FWH_C0_IDSEL**: This dictates the IDSEL of 512 KB of the FWH memory range starting at 4 GB - 4 MB (FFC00000H) to 4 GB - 3.5 MB (FFC7FFFFH) as well as register space starting at (4 GB-4MB) – 4MB (FF800000h) to (4 GB-4MB) - 3.5MB (FF87FFFFh). The enable for this range is controlled through bit 0 of the FWH Decode Enable Register at E3H.

11.1.27.3 Test Mode Register

Address: FC-FFh
Default Value: 00000000h
Attributes: Read/Write

Bit	Description
31:1	Reserved.
0	**Alternate Access Mode Enable:** When set, the part enters alternate access mode. This allows reads to certain write-only registers and writes to certain read-only registers. Read to Port 70h will return the NMI mask value. See Section 10.3 of the *RS-IFB (I/O & Firmware Bridge) External Design Specification*, Rev 2.0, for more detail.

11.2 PCI to LPC I/O Space Registers

11.2.1 DMA Registers

The IFB contains DMA circuitry that incorporates the functionality of two 82C37 DMA controllers (DMA1 and DMA2). The DMA registers control the operation of the DMA controllers and are all accessible from the Host CPU via the PCI Bus interface.

11.2.1.1 Dcom–Dma Command Register (I/O)

I/O Address: Channels 0-3–08h; Channels 4-7–0D0h
Default Value: 00h (CPURST or Master Clear)
Attribute: Write Only

This 8-bit register controls the configuration of the DMA. Note that disabling channels 4-7 also disables channels 0-3, since channels 0-3 are cascaded onto Channel 4.

Bit	Description
7	**Reserved.** Must be 0.
6	**Reserved.** Must be 0.
5	**Reserved.** Must be 0.
4	**DMA Group Arbitration Priority.** 1=Rotating priority; 0=Fixed priority
3	**Reserved.** Must be 0
2	**DMA Channel Group Enable.** 1=Disable; 0 = Enable.
1:0	**Reserved.** Must be 0.

11.2.1.2 Dcm–Dma Channel Mode Register (I/O)

I/O Address: Channels 0-3=0Bh; Channels 4-7=0D6h
Default Value: Bits[7:2]=0; Bits[1:0]=undefined (CPURST or Master Clear)
Attribute: Write Only

Each channel has a 6-bit DMA Channel Mode Register. The Channel Mode Registers provide control over DMA transfer type, transfer mode, address increment/decrement, and auto-initialization.

Bit	Description
7:6	**DMA Transfer Mode.** Each DMA channel can be programmed in one of four different modes: **Bits[7:6] Transfer Mode** 00 Demand Mode 01 Single Mode 10 Block Mode 11 I Cascade Mode
5	**Address Increment/Decrement Select.** 0=Increment; 1=Decrement.
4	**Auto-initialize Enable.** 1=Enable; 0=Disable.
3:2	**DMA Transfer Type.** When Bits [7:6]=11, the transfer type bits are irrelevant. **Bits[3:2] Transfer Type** 00 Verify transfer 01 Write transfer 10 Read transfer 11 I Illegal
1:0	**DMA Channel Select.** Bits [1:0] select the DMA Channel Mode Register written to by bits [7:2]. **Bits[1:0] Channel** 00 Channel 0 (4) 01 Channel 1 (5) 10 Channel 2 (6) 11 Channel 3 (7)

11.2.1.3 Dr–Dma Request Register (I/O)

I/O Address: Channels 0-3–09h; Channels 4-7–0D2h
Default Value: Bits[1:0]=undefined; Bits[7:2]=0 (CPURST or Master Clear)
Attribute: Write Only

The Request Register is used by software to initiate a DMA request. The DMA responds to the software request as though DREQx is asserted. These requests are non-maskable and subject to prioritization by the priority encoder network. For a software request, the channel must be in Block Mode. The Request Register status for DMA1 and DMA2 is output on bits [7:4] of a Status Register read.

Bit	Description
7:3	**Reserved**. Must be 0
2	**DMA Channel Service Request.** 0=Resets the individual software DMA channel request bit. 1=Sets the request bit. Generation of a TC also sets this bit to 0.
1:0	**DMA Channel Select.** Bits [1:0] select the DMA channel mode register to program with bit 2. **Bits[1:0] Channel** 00 Channel 0 01 Channel 1 (5) 10 Channel 2 (6) 11 Channel 3 (7)

11.2.1.4 WSMB–Write Single Mask Bit (I/O)

I/O Address: Channels 0-3–0Ah; Channels 4-7–0D4h
Default Value: Bits[1:0]=undefined; Bit 2=1; Bits[7:3]=0 (CPURST or a Master Clear)
Attribute: Write Only

A channel's mask bit is automatically set when the Current Byte/Word Count Register reaches terminal count (unless the channel is programmed for auto-initialization). Setting the entire register disables all DMA requests until a clear mask register instruction allows them to occur. This instruction format is similar to the format used with the DMA Request Register. Masking DMA channel 4 (DMA controller 2, channel 0) also masks DMA channels [3:0].

Bit	Description
7:3	**Reserved**. Must be 0.
2	**Channel Mask Select.** 1=Disable DREQ for the selected channel. 0=Enable DREQ for the selected channel.
1:0	**DMA Channel Select.** Bits [1:0] select the DMA Channel Mode Register for bit 2. Bits[1:0] Channel 00 Channel 0 (4) 01 Channel 1 (5) 10 Channel 2 (6) 11 Channel 3 (7)

11.2.1.5 RWAMB–Read / Write All Mask Bits (I/O)

I/O Address: Channels 0-3–0Fh; Channels 4-7–0DEh
Default Value: Bit[3:0]=1111; Bit[7:4]=0000 (CPURST or Master Clear)
Attribute: Read/Write

A channel's mask bit is automatically set to 1 when the Current Byte/Word Count Register reaches terminal count (unless the channel is programmed for auto-initialization). Setting bits [3:0] to 1 disables all DMA requests until a clear mask register instruction enables the requests. Note that, masking DMA channel 4 (DMA controller 2, channel 0), masks DMA channels [3:0]. Also note that, Masking DMA controller 2 with a write to port 0DEh also masks DREQ assertions from DMA controller 1.

Bit	Description
7:4	**Reserved**. Must be 0.
3:0	**Channel Mask Bits.** 1=Disable the corresponding DREQ(s); 0=Enable the corresponding DREQ(s). Bit Channel 0 0 (4) 1 1 (5) 2 2 (6) 3 3 (7)

11.2.1.6 Ds–Dma Status Register (I/O)

I/O Address: Channels 0-3–08h; Channels 4-7–0D0h
Default Value: 00h
Attribute: Read Only

Each DMA controller has a read-only DMA Status Register that indicates which channels have reached terminal count and which channels have a pending DMA request.

Bit	Description
7:4	**Channel Request Status.** When a valid DMA request is pending for a channel (on its DREQ signal line), the corresponding bit is set to 1. When a DMA request is not pending for a particular channel, the corresponding bit is set to 0. The source of the DREQ may be hardware or a software request. Note that channel 4 does not have DREQ or DACK lines, so the response for a read of DMA2 status for channel 4 is irrelevant. Bit Channel 4 0 5 1 (5) 6 2 (6) 7 3 (7)
3:0	**Channel Terminal Count Status.** 1=TC is reached; 0=TC is not reached. Bit Channel 0 0 1 1 (5) 2 2 (6) 3 3 (7)

11.2.1.7 DBADDR–DMA Base and Current Address Registers (I/O)

I/O Address: DMA Channel 0–000h, DMA Channel 1–002h, DMA Channel 2–004h, DMA Channel 3–006h, DMA Channel 4–0C0h, DMA Channel 5–0C4h, DMA Channel 6–0C8h, DMA Channel 7–0CCh
Default Value: Undefined (CPURST or Master Clear)
Attribute: Read/Write

This Register works in conjunction with the Low Page Register. After an auto-initialization, this register retains the original programmed value. Auto-initialize takes place after a TC. The address register is automatically incremented or decrement after each transfer. This register is read/written in successive 8-bit bytes. The programmer must issue the Clear Byte Pointer Flip-Flop command to reset the internal byte pointer and correctly align the write prior to programming the Current Address Register. Auto-initialize takes place only after a TC.

Bit	Description
15:0	**Base and Current Address [15:0].** These bits represent address bits [15:0] used when forming the 24-bit address for DMA transfers.

11.2.1.8 DBCNT–Dma Base and Current Count Registers (I/O)

I/O Address:	DMA Channel 0–001h	DMA Channel 4–0C2h
	DMA Channel 1–003h	DMA Channel 5–0C6h
	DMA Channel 2–005h	DMA Channel 6–0CAh
	DMA Channel 3–007h	DMA Channel 7–0CEh
Default Value:	Undefined (CPURST or Master Clear)	
Attribute:	Read/Write	

This register determines the number of transfers to be performed. The actual number of transfers is one more than the number programmed in the Current Byte/Word Count Register When the value in the register is decremented from zero to FFFFh, a TC is generated. Auto-initialize can only occur when a TC occurs. If it is not auto-initialized, this register has a count of FFFFh after TC.

For transfers to/from an 8-bit I/O, the Byte/Word count indicates the number of bytes to be transferred. This applies to DMA channels 0-3. For transfers to/from a 16-bit I/O, with shifted address, the Byte/Word count indicates the number of 16-bit words to be transferred. This applies to DMA channels 5-7.

Bit	Description
15:0	**Base and Current Byte/ Word Count.** These bits represent the 16 byte/word count bits used when counting down a DMA transfer.

11.2.1.9 DLPAGE–DMA Low Page Registers (I/O)

I/O Address:	DMA Channel 0–087h	DMA Channel 5–08Bh
	DMA Channel 1–083h	DMA Channel 6–089h
	DMA Channel 2–081h	DMA Channel 7–08Ah
	DMA Channel 3–082h	
Default Value:	Undefined (CPURST or Master Clear)	
Attribute:	Read/Write	

This register works in conjunction with the Current Address Register. After an auto-initialization, this register retains the original programmed value. Auto-initialize takes place after a TC.

Bit	Description
7:0	**DMA Low Page [23:16].** These bits represent address bits [23:16] of the 24-bit DMA address.

11.2.1.10 DCBP–Dma Clear Byte Pointer Register (I/O)

I/O Address:	Channels 0-3–00Ch; Channels 4-7–0D8h
Default Value:	All bits undefined
Attribute:	Write Only

Writing to this register executes the Clear Byte Pointer Command. This command is executed prior to reading/writing a new address or word count to the DMA. The command initializes the byte pointer flip-flop to a known state so that subsequent accesses to register contents address upper and lower bytes in the correct sequence. The Clear Byte Pointer Command (or CPURST or the Master Clear Command) clears the internal latch used to address the upper or lower byte of the 16-bit Address and Word Count Registers.

Bit	Description
7:0	**Clear Byte Pointer.** No specific pattern. Command enabled with a write to the I/O port address.

11.2.1.11 Dmc–Dma Master Clear Register (I/O)

I/O Address: Channel 0-3–00Dh; Channel 4-7–0DAh
Default Value: All bits undefined
Attribute: Write Only

This software instruction has the same effect as the hardware Reset.

Bit	Description
7:0	**Master Clear.** No specific pattern. Command enabled with a write to the I/O port address

11.2.1.12 Dclm–Dma Clear Mask Register (I/O)

I/O Address: Channel 0-3–00Eh; Channel 4-7–0DCh
Default Value: All bits undefined
Attribute: Write Only

This command clears the mask bits of all four channels, enabling them to accept DMA requests.

Bit	Description
7:0	**Clear Mask Register.** No specific pattern. Command enabled with a write to the I/O port address.

11.2.2 Interrupt Controller Registers

The IFB contains an interrupt controller that incorporates the Functionality of two 82C59 interrupt controllers. The interrupt registers control the operation of the interrupt controller.

11.2.2.1 Icw1–Initialization Command Word 1 Register (I/O)

I/O Address: INT CNTRL-1–020h; INT CNTRL-2–0A0h
Default Value: All bits undefined
Attribute: Write Only

A write to Initialization Command Word 1 starts the interrupt controller initialization sequence. Addresses 020h and 0A0h are referred to as the base addresses of CNTRL-1 and CNTRL-2, respectively. An I/O write to the CNTRL-1 or CNTRL-2 base address with bit 4 equal to 1 is interpreted as ICW1. For IFB-based systems, three I/O writes to “base address + 1" must follow the ICW1. The first write to “base address + 1" performs ICW2, the second write performs ICW3, and the third write performs ICW4.

ICW1 starts the initialization sequence during which the following automatically occur:

1. The Interrupt Mask register is cleared.
2. IRQ7 input is assigned priority 7.
3. The slave mode address is set to 7.
4. Special Mask Mode is cleared and Status Read is set to IRR.

Bit	Description
7:5	**ICW/OCW select.** These bits should be 000 when programming the IFB.
4	**ICW/OCW select.** Bit 4 must be a 1 to select ICW1. After the fixed initialization sequence to ICW1, ICW2, ICW3, and ICW4, the controller base address is used to write to OCW2 and OCW3. Bit 4 is a 0 on writes to these registers. A 1 on this bit at any time will force the interrupt controller to interpret the write as an ICW1. The controller will then expect to see ICW2, ICW3, and ICW4.
3	**Edge/Level Bank Select (LTIM).** This bit is disabled. Its Function is replaced by the Edge/ Level Triggered Control (ELCR) Registers.
2	**ADI.** Ignored for the IFB. This bit should be programmed to '0'.
1	**Single or Cascade (SNGL).** This bit must be programmed to a 0.
0	**ICW4 Write Required (IC4).** This bit must be set to a 1.

11.2.2.2 Icw2–Initialization Command Word 2 Register (I/O)

I/O Address: INT CNTRL-1–021h; INT CNTRL-2–0A1h
Default Value: All bits undefined
Attribute: Write Only

ICW2 is used to initialize the interrupt controller with the five most significant bits of the interrupt vector address.

Bit	Description
7:3	**Interrupt Vector Base Address.** Bits [7:3] define the base address in the interrupt vector table for the interrupt routines associated with each interrupt request level input.
2:0	**Interrupt Request Level.** Must be programmed to all 0s.

11.2.2.3 Icw3–Initialization Command Word 3 Register (I/O)

I/O Address: INT CNTRL-1–021h
Default Value: All bits undefined
Attribute: Write Only

The meaning of ICW3 differs between CNTRL-1 and CNTRL-2. On CNTRL-1, the master controller, ICW3 indicates which CNTRL-1 IRQ line physically connects the INTR output of CNTRL-2 to CNTRL-1.

Bit	Description
7:3	**Reserved.** Must be programmed to all 0s.
2	**Cascaded Mode Enable.** This bit must be programmed to 1 selecting cascade mode.
1:0	**Reserved.** Must be programmed to all 0s.

11.2.2.4 Icw3–Initialization Command Word 3 Register (I/O)

I/O Address: INT CNTRL-2–0A1h
Default Value: All bits undefined
Attribute: Write Only

On CNTRL-2 (the slave controller), ICW3 is the slave identification code broadcast by CNTRL-1.

Bit	Description
7:3	**Reserved.** Must be programmed to all 0s.
2:0	**Slave Identification Code.** Must be programmed to 02h.

11.2.2.5 Icw4–Initialization Command Word 4 Register (I/O)

I/O Address: INT CNTRL-1–021h; INT CNTRL-2–0A1h
Default Value: 01h
Attribute: Write Only

Both IFB interrupt controllers must have ICW4 programmed as part of their initialization sequence.

Bit	Description
7:5	**Reserved.** Must be programmed to all 0s.
4	**Special Fully Nested Mode (SFNM).** Bit 4, SFNM, should normally be disabled by writing a 0 to this bit. If SFNM=1, the special fully nested mode is programmed.
3	**Buffered** mode **(BUF).** Must be programmed to 0 selecting non-buffered mode.
2	**Master/Slave in Buffered Mode.** Should always be programmed to 0. Bit not used.
1	**AEOI (Automatic End of Interrupt).** This bit should normally be programmed to 0. This is the normal end of interrupt. If this bit is 1, the automatic end of interrupt mode is programmed.
0	**Microprocessor Mode.** Must be programmed to 1 indicating an Intel Architecture-based system.

11.2.2.6 Ocw1–Operational Control Word 1 Register (I/O)

I/O Address: INT CNTRL-1–021h; INT CNTRL-2–0A1h
Default Value: 00h
Attribute: Read/Write

OCW1 sets and clears the mask bits in the Interrupt Mask Register (IMR). Each interrupt request line may be selectively masked or unmasked any time after initialization. The IMR stores the interrupt line mask bits. The IMR operates on the IRR. Masking of a higher priority input does not affect the interrupt request lines of lower priority. Unlike status reads of the ISR and IRR, for reading the IMR, no OCW3 is needed. The output data bus contains the IMR when an I/O read is active and the I/O address is 021h or 0A1h (OCW1). All writes to OCW1 must occur following the ICW1-ICW4 initialization sequence, since the same I/O ports are used for OCW1, ICW2, ICW3 and ICW4.

Bit	Description
7:0	**Interrupt Request Mask (Mask [7:0])**. When a 1 is written to any bit in this register, the corresponding IRQx line is masked. For example, if bit 4 is set to a 1, then IRQ4 is masked. Interrupt requests on IRQ4 do not set channel 4's interrupt request register (IRR) bit as long is the channel is masked. When a 0 is written to any bit in this register, the corresponding IRQx is unmasked. Note that masking IRQ2 on CNTRL-1 also masks the interrupt requests from CNTRL-2, which is physically cascaded to IRQ2.

11.2.2.7 Ocw2–Operational Control Word 2 Register (I/O)

I/O Address: INT CNTRL-1–020h; INT CNTRL-2–0A0h
Default Value: Bit[4:0]=undefined; Bit[7:5]=001
Attribute: Write Only

OCW2 controls both the Rotate Mode and the End of Interrupt Mode. Following a CPURST or ICW initialization, the controller enters the fully nested mode of operation. Both rotation mode and specific EOI mode are disabled following initialization.

Bit	Description
7:5	**Rotate and EOI Codes.** R, SL, EOI - These three bits control the Rotate and End of Interrupt modes and combinations of the two. Bits[7:5] Function Bits[7:5] Function 001 Non-specific EOI Cmd 000 Rotate in Auto EOI Mode (Clear) 011 Specific EOI Cmd 111 *Rotate on Specific EOI Cmd 101 Rotate on Non-Spec EOI Cmd 110 *Set Priority Cmd 100 Rotate in Auto EOI Mode (Set) 010 No Operation * L0 - L2 Are Used
4:3	**OCW2 Select.** Must be programmed to 00 selecting OCW2.
2:0	**Interrupt Level Select (L2, L1, L0).** L2, L1, and L0 determine the interrupt level acted upon when the SL bit is active (bit 6). When the SL bit is inactive, bits [2:0] do not have a defined Function; programming L2, L1 and L0 to 0 is sufficient in this case. Bit[2:0] Interrupt Level Bit[2:0] Interrupt Level 000 IRQ 0(8) 100 IRQ 4(12) 001 IRQ 1(9) 101 IRQ 5(13) 010 IRQ 2(10) 110 IRQ 6(14) 011 IRQ 3(11) 111 IRQ 7(15)

11.2.2.8 Ocw3–Operational Control Word 3 Register (I/O)

I/O Address: INT CNTRL-1–020h; INT CNTRL-2–0A0h
Default Value: Bit[6,0]=0; Bit[7,4:2]=Undefined; Bit[5,1]=1
Attribute: Read/Write

OCW3 serves three important Functions–Enable Special Mask Mode, Poll Mode control, and IRR/ ISR register read control.

Bit	Description
7	**Reserved**. Must be 0.
6	**Special Mask Mode (SMM).** If ESMM=1 and SMM=1, the interrupt controller enters Special Mask Mode. If ESMM=1 and SMM=0, the interrupt controller is in normal mask mode. When ESMM=0, SMM has no effect.

Bit	Description
5	**Enable Special Mask Mode (ESMM).** 1=Enable SMM bit; 0=Disable SMM bit.
4:3	**OCW3 Select.** Must be programmed to 01 selecting OCW3.
2	**Poll Mode Command.** 0=Disable Poll Mode Command. When bit 2=1, the next I/O read to the interrupt controller is treated as an interrupt acknowledge cycle indicating highest priority request.
1:0	**Register Read Command.** Bits [1:0] provide control for reading the In-Service Register (ISR) and the Interrupt Request Register (IRR). When bit 1=0, bit 0 does not affect the register read selection. When bit 1=1, bit 0 selects the register status returned following an OCW3 read. If bit 0=0, the IRR will be read. If bit 0=1, the ISR will be read. Following ICW initialization, the default OCW3 port address read will be “read IRR”. To retain the current selection (read ISR or read IRR), always write a 0 to bit 1 when programming this register. The selected register can be read repeatedly without reprogramming OCW3. To select a new status register, OCW3 must be reprogrammed prior to attempting the read. Bit[1:0] Function 00 No Action 01 No Action 10 Read IRQ Register 11 Read IS Register

11.2.2.9 Elcr1–Edge/Level Control Register (I/O)

I/O Address: INT CNTRL-1–4D0h
Default Value: 00h
Attribute: Read/Write

ELCR1 register allows IRQ3 - IRQ7 to be edge or level programmable on an interrupt by interrupt basis. IRQ0, IRQ1 and IRQ2 are not programmable and are always edge sensitive. When level triggered, the interrupt is signaled active when input IRQ signal is high.

Bit	Description
7	**IRQ7 ECL.** 0 = Edge Triggered mode; 1 = Level Triggered mode.
6	**IRQ6 ECL.** 0 = Edge Triggered mode; 1 = Level Triggered mode.
5	**IRQ5 ECL.** 0 = Edge Triggered mode; 1 = Level Triggered mode.
4	**IRQ4 ECL.** 0 = Edge Triggered mode; 1 = Level Triggered mode.
3	**IRQ3 ECL.** 0 = Edge Triggered mode; 1 = Level Triggered mode.
2:0	**Reserved.** Must be 0.

11.2.2.10 Elcr2–Edge/Level Control Register (I/O)

I/O Address: INT CNTRL-2–4D1h
Default Value: 00h
Attribute: Read/Write

ELCR2 register allows IRQ[15,14,12:9] to be edge or level programmable on an interrupt by interrupt basis. Note that, IRQ[13,8#] are not programmable and are always edge sensitive. When level triggered, the interrupt is signaled active when input IRQ signal is high.

Bit	Description
7	**IRQ15 ECL.** 0 = Edge Triggered mode; 1 = Level Triggered mode.
6	**IRQ14 ECL.** 0 = Edge Triggered mode; 1 = Level Triggered mode.
5	**Reserved.** Must be 0.
4	**IRQ12 ECL.** 0 = Edge Triggered mode; 1 = Level Triggered mode.
3	**IRQ11 ECL.** 0 = Edge Triggered mode; 1 = Level Triggered mode.
2	**IRQ10 ECL.** 0 = Edge Triggered mode; 1 = Level Triggered mode.
1	**IRQ9 ECL.** 0 = Edge Triggered mode; 1 = Level Triggered mode.
0	**Reserved.** Must be 0.

11.2.3 Counter/Timer Registers

11.2.3.1 Tcw–Timer Control Word Register (I/O)

I/O Address: 043h
Default Value: All bits undefined
Attribute: Write Only

Bit	Decription
7:6	**Counter Select.** The Read Back Command is selected when bits[7:6] are both 1. Bit[7:6] Function — Bit[7:6] Function 00 Counter 0 select — 10 Counter 2 select 01 Counter 1 select — 11 Read Back Command
5:4	**Read/Write Select.** The Counter Latch Command is selected when bits[5:4] are both 0. Bit[5:4] Function — Bit[5:4] Function 00 Counter Latch Command — 10 R/W Most Significant Byte 01 R/W Least Significant Byte — 11 R/W LSB then MSB
3:1	**Counter Mode Selection.** Bits [3:1] select one of six possible counter modes. Bit[3:1] Mode Function 000 0 Out signal on end of count (=0) 001 1 Hardware re-triggerable one-shot X10 2 Rate generator (divide by n counter) X11 3 Square wave output 100 4 Software triggered strobe 101 5 Hardware triggered strobe
0	**Binary/BCD Countdown Select.** 0=Binary countdown. The largest possible binary count is 2^{16}. 1=Binary coded decimal (BCD) count is used. The largest BCD count allowed is 10^4.

The Timer Control Word Register specifies the counter selection, the operating mode, the counter byte programming order and size of the count value, and whether the counter counts down in a 16-bit or binary-coded decimal (BCD) format. After writing the control word, a new count can be written at any time. The new value takes effect according to the programmed mode.

Read Back Command

The Read Back Command is used to determine the count value, programmed mode, and current states of the OUT pin and Null count flag of the selected counter or counters. The Read Back Command is written to the Timer Control Word Register which latches the current states of the above mentioned variables. The value of the counter and its status may then be read by I/O access to the counter address. Note that The Timer Counter Register bit definitions are different during the Read Back Command than for a normal Timer Counter Register write.

Bit	Description
7:6	**Read Back Command.** When bits[7:6]=11, the Read Back Command is selected during a write to the Timer Control Word Register. Following the Read Back Command, I/O reads from the selected counter's I/O addresses produce the current latch status, the current latched count, or both if bits 4 and 5 are both 0.
5	**Latch Count of Selected Counters.** When bit 5=0, the current count value of the selected counters will be latched. When bit 5=1, the count will not be latched.
4	**Latch Status of Selected Counters.** When bit 4=0, the status of the selected counters will be latched. When bit 4=1, the status will not be latched.
3	**Counter 2 Select.** When bit 3=1, Counter 2 is selected for the latch command selected with bits 4 and 5. When bit 3=0, status and/or count will not be latched.
2	**Counter 1 Select.** When bit 2=1, Counter 1 is selected for the latch command selected with bits 4 and 5. When bit 2=0, status and/or count will not be latched.
1	**Counter 0 Select.** When bit 1=1, Counter 0 is selected for the latch command selected with bits 4 and 5. When bit 1=0, status and/or count will not be latched.
0	**Reserved**. Must be 0.

Counter Latch Command

Bit	Description
7:6	**Counter Selection.** Bits 6 and 7 are used to select the counter for latching. Bit[7:6] Function 00 latch counter 0 select 01 latch counter 1 select 10 latch counter 2 select 11 Read Back Command select
5:4	**Counter Latch Command.** When bits[5:4]=00, the Counter Latch Command is selected during a write to the Timer Control Word Register. Following the Counter Latch Command, I/O reads from the selected counter's I/O addresses produce the current latched count.
3:0	**Reserved**. Must be 0.

The Counter Latch Command latches the current count value at the time the command is received. If a Counter is latched once and then, some time later, latched again before the count is read, the second Counter Latch Command is ignored. The count read will be the count at the time the first Counter Latch Command was issued. If the counter is programmed for two byte counts, two bytes must be read. The two bytes do not have to be read successively (read, write, or programming operations for other counters may be inserted between the reads). Note that the Timer Counter

Register bit definitions are different during the Counter Latch Command than for a normal Timer Counter Register write. Note that, If a counter is programmed to read/write two-byte counts, a program must not transfer control between reading the first and second byte to another routine that also reads from that same counter. Otherwise, an incorrect count will be read.

11.2.3.2 TMRSTS–Timer Status Registers (I/O)

I/O Address: Counter 0–040h; Counter 1–041h; Counter 2–042h
Default Value: Bits[6:0]=X; Bit 7=0
Attribute: Read Only

Each counter's status byte can be read following an Interval Timer Read Back Command. If latch status is chosen (bit 4=0, Read Back Command) as a read back option for a given counter, the next read from the counter's Counter Access Ports Register returns the status byte.

Bit	Description
7	**Counter OUT Pin State.** 1=Pin is 1; 0=Pin is 0.
6	**Count Register Status.** This bit indicates when the last count written to the Count Register (CR) has been loaded into the counting element (CE). 0=Count has been transferred from CR to CE and is available for reading. 1=Count has not been transferred from CR to CE and is not yet available for reading.
5:4	**Read/Write Selection Status.** Bits[5:4] reflect the read/write selection made through bits[5:4] of the Control Register. Bit[5:4] Function 00 Counter Latch Command 01 R/W Least Significant Byte (LSB) 10 R/W Most Significant Byte (MSB) 11 R/W LSB then MSB
3:1	**Mode Selection Status.** Bits[3:1] return the counter mode programming. Bit[3:1] Mode Selected Bit[3:1] Mode Selected 000 0 X11 3 001 1 100 4 X10 2 101 5
0	**Countdown Type Status.** 0=Binary countdown; 1=Binary coded decimal (BCD) countdown.

11.2.3.3 TMRCNT–Timer Count Registers (I/O)

I/O Address: Counter 0–040h; Counter 1–041h; Counter 2–042h
Default Value: All bits undefined
Attribute: Read/Write

Each of these I/O ports is used for writing count values to the Count Registers; reading the current count value from the counter by either an I/O read, after a counter-latch command, or after a Read Back Command; and reading the status byte following a Read Back Command.

Bit	Description
7:0	**Counter Port Bits]**. Each counter I/O port address is used to program the 16-bit Count Register. The order of programming, either LSB only, MSB only, or LSB then MSB, is defined with the Interval Counter Control Register. The counter I/O port is also used to read the current count from the Count Register and return counter programming status following a Read Back Command.

11.2.4 NMI Registers

The NMI logic incorporates two different 8-bit registers. The CPU reads the NMISC Register to determine the NMI source (bits set to a 1). After the NMI interrupt routine processes the interrupt, software clears the NMI status bits by setting the corresponding enable/disable bit to a 1. The NMI Enable and Real-Time Clock Register can mask the NMI signal and disable/enable all NMI sources.

To ensure that all NMI requests are serviced, the NMI service routine software flow should be as follows:

1. NMI is detected by the processor on the rising edge of the NMI input.
2. The processor will read the status stored in port 061h to determine what sources caused the NMI. The processor may then set to 0 the register bits controlling the sources that it has determined to be active. Between the time the processor reads the NMI sources and sets them to a 0, an NMI may have been generated by another source. The level of NMI will then remain active. This new NMI source will not be recognized by the processor because there was no edge on NMI.
3. The processor must then disable all NMIs by setting bit 7 of port 070H to a 1 and then enable all NMIs by setting bit 7 of port 070H to a 0. This will cause the NMI output to transition low then high if there are any pending NMI sources. The CPU's NMI input logic will then register a new NMI.

11.2.4.1 Nmisc–Nmi Status and Control Register (I/O)

I/O Address: 061h
Default Value: 00h
Attribute: Read/Write

This register reports the status of different system components, controls the output of the speaker counter (Counter 2), and gates the counter output that drives the SPKR signal.

Bit	Description
7	**SERR# NMI Source Status–RO.** Bit 7 is set if a system board agent (PCI devices or main memory) detects a system board error and pulses the PCI SERR# line. This interrupt source is enabled by setting bit 2 to 0. To reset the interrupt, set bit 2 to 0 and then set it to 1. When writing to port 061h, bit 7 must be 0.
6	*reserved (0)*
5	**Timer Counter 2 OUT Status–RO.** The Counter 2 OUT signal state is reflected in bit 5. The value on this bit following a read is the current state of the Counter 2 OUT signal. Counter 2 must be programmed following a CPURST for this bit to have a determinate value. When writing to port 061h, bit 5 must be a 0.
4	**Refresh Cycle Toggle–RO.** The Refresh Cycle Toggle signal toggles from either 0 to 1 or 1 to 0 following every refresh cycle. When writing to port 061h, bit 4 must be a 0.
3	Must be programmed to '1'.
2	**PCI SERR# Enable–R/W.** 1=Clear and Disable; 0=Enable. For the IFB, the SERR# signal can be for a special protocol between the host-to-PCI bridge and the IFB (see MSTAT Register description, 6Ah-6Bh, Function 0).
1	**Speaker Data Enable–R/W.** 0=SPKR output is 0; 1= the SPKR output is the Counter 2 OUT signal value.
0	**Timer Counter 2 Enable–R/W.** 0=Disable; 1=Enable.

11.2.4.2 NmiEN–Nmi Enable Register (Shared with Real-time Clock Index Register) (I/O)

I/O Address: 070h
Default Value: Bit[6:0]=undefined; Bit 7=1
Attribute: Write Only

This port is shared with the real-time clock. Do not modify the contents of this register without considering the effects on the state of the other bits.

Bit	Description
7	**NMI Enable.** 1=Disable generation of NMI; 0=Enable generation of NMI.
6:0	**Real Time Clock Address.** Used by the Real Time Clock to address memory locations. Not used for NMI enabling/disabling. See description in Section 11.2.5.1.

11.2.5 Real Time Clock Registers

11.2.5.1 RTCI–Real-time Clock Index Register (Shared with NMI Enable Register) (I/O)

I/O Address: 070h
Default Value: Bit[6:0]=Undefined; Bit 7=1
Attribute: Write Only

This port is shared with the NMI enable. Do not modify the contents of this register without considering the effects on the state of the other bits.

Bit	Description
7	**NMI Enable.** Used by IFB NMI logic.
6:0	**Real Time Clock Address.** Latched by the Real Time Clock to address memory locations within the standard RAM bank accessed via the Real Time Clock Data Register (071h).

11.2.5.2 RTCD–Real-time Clock Data Register (I/O)

I/O Address: 071h
Default Value: Undefined
Attribute: Read/Write

The data port for accesses to the RTC standard RAM bank.

Bit	Description
7:0	**Standard RAM Data Port.** Data written to standard RAM bank address selected via RTC Index Register (070h).

11.2.5.3 RTCEI–Real-time Clock Extended Index Register (I/O)

I/O Address: 072h
Default Value: Unknown
Attribute: Write Only

The index port for accesses to the RTC extended RAM bank.

Bit	Description
7	Reserved.
6:0	**Real Time Clock Extended Address.** Latched by the Real Time Clock to address memory locations within the extended RAM bank accessed via the Real Time Clock Extended Data Register (073h).

11.2.5.4 RTCED–Real-time Clock Extended Data Register (I/O)

I/O Address: 073h
Default Value: Unknown
Attribute: Read/Write

The data port for accesses to the RTC extended RAM bank.

Bit	Description
7:0	**Extended RAM Data Port.** Data written to standard RAM bank address selected via RTC Extended Index Register (072h).

11.2.6 Advanced Power Management (APM) Registers

This section describes two power management registers–APMC and APMS Registers. These registers are located in normal I/O space and must be accessed (via the PCI Bus) with 8 bit accesses.

11.2.6.1 APMC–Advanced Power Management Control Port (I/O)

I/O Address: 0B2h
Default Value: 00h
Attribute: Read/Write

This register passes data (APM Commands) between the OS and the SMI handler. In addition, writes can generate an SMI. The IFB operation is not effected by the data in this register.

Bit	Description
7:0	**APM Control Port (APMC).** Writes to this register store data in the APMC Register and reads return the last data written. In addition, writes generate an SMI, if the APMC_EN bit (PCI Function 3, offset58h, bit 25) is set to 1. Reads do not generate an SMI.

11.2.6.2 APMS–Advanced Power Management Status Port (I/O)

I/O Address: 0B3h
Default Value: 00h
Attribute: Read/Write

This register passes status information between the OS and the SMI handler. The IFB operation is not effected by the data in this register.

Bit	Description
7:0	**APM Status Port (APMS).** Writes store data in this register and reads return the last data written.

11.2.7 ACPI Registers

The ACPI registers are I/O mapped. The base address is set via PCI Function 0 configuration register 40h. The registers are defined to be compliant with the ACPI 1.0 specification, and use the same bit names. All reserved bits will always return 0 when read, and will have no effect when written.

11.2.7.1 Power Management 1 Status

Address Offset: 00-01h
Attributes: Read/Write
Default Value: Bit 11:Undefined, All other bits ‘0’
Size: 16 bits

Bit	Description
15	**WAK_STS:** This bit is set when the system is in one of the Sleep states (via the SLP_EN bit) and an enabled IFB wake/break event occurs. Upon setting this bit, IFB will transition the system to the ON state. This bit can only be set by hardware and can only be cleared by writing a one to this bit position. This bit is not affected by a hard reset caused by a CF9 write.
14:12	Reserved.
11	**PWRBTNOR_STS:** This bit is set any time a Power Button Override Event occurs. The override event occurs when the power button is pressed for 4 consecutive seconds. The power button override will cause an unconditional transition to the S5 state, as well as set the AFTERG3 bit. The firmware or SCI handler can clear this bit by writing a 1 to it. This bit is not affected by a hard reset caused by a CF9 write. Upon reset, this bit is undefined.
10	**RTC_STS:** This bit is set when the RTC generates an alarm (assertion of the IRQ8# signal). Additionally if the RTC_EN bit is set then the setting of the RTC_STS bit will generate wake event. This bit is only set by hardware and can only be reset by writing a one to this bit position. This bit is not affected by a hard reset caused by a CF9 write.
9	Reserved.
8	**PWRBTN_STS:** This bit is set by the hardware when the power button is pressed. The firmware or SCI handler should clear this bit by writing a 1 to it. This bit is not affected by a hard reset caused by a CF9 write.
7:6	Reserved.
5	**GBL_STS:** This bit is set when an SCI is generated due to the firmware wanting the attention of the SCI handler. Firmware has a corresponding bit, BIOS_RLS, which will cause an SCI and set this bit. The SCI handler should then clear this bit by writing a 1 to it.
4:1	Reserved.

Bit	Description
0	**TMROF_STS:** This is the timer overflow status bit. This bit gets set anytime the 22nd bit of the 24 bit timer goes from high to low (bits are counted from 0 to 23). This will occur every 2.3435 seconds. When the TMROF_EN bit is set, the setting of the TMROF_STS bit will generate an SCI or SMI. SMI will be generated if ACPI_TMR_EN, SMI_EN and TMROF_EN are set, and SCI_EN is not set. SCI will be generated if SCI_EN and TMROF_EN are set. The SCI or SMI handler clears this bit by writing a 1 to it.

11.2.7.2 Power Management 1 Enable

Address Offset: 02-03h
Attributes: Read/Write
Default Value: Bit 10: Undefined, All other bits ‘0’
Size: 16 bits

Bit	Description
15:11	Reserved.
10	**RTC_EN:** This bit is used to enable the setting of the RTC_STS bit to generate a wake event. The RTC_STS bit is set anytime the RTC generates an alarm (asserts the IRQ8# signal active). The enable bit does not have to be set to enable the setting of the RTC_STS bit by the assertion of the RTC alarm. The value of this bit must be maintained, even through a G3 state. The IFB will not resume, from RTC, after power failure (RSMRST# low) even if this bit is set. This bit is automatically cleared by a power button override. The IFB can resume from RTC if only PWROK goes low with RSMRST# high. Upon reset, this bit is undefined.
9	Reserved.
8	**PWRBTN_EN:** This bit is set to 1 to enable the setting of the PWRBTN_STS bit to also generate an SCI or SMI. Even if the PWRBTN_EN bit is set to 0, the power button can always generate a Wake event (if in a n S1-S5 state), and the power button override can cause an unconditional transition to the S5 state.
7:6	Reserved.
5	**GBL_EN:** Global enable bit. When both GBL_EN and GBL_STS are set, an SCI event is generated.
4:1	Reserved.
0	**TMROF_EN:** This is the timer overflow interrupt enable bit. When this bit is set then an SCI or SMI is generated anytime the TMOF_STS bit is also set. When this bit is reset, then no interrupt is generated when the TMROF_STS bit is set.

11.2.7.3 Power Management 1 Control

Address Offset: 04-05h
Type: Read/Write
Default Value: Bits 12:10 Undefined, All other bits ‘0’
Size: 16 bits

Bit	Description
15:14	Reserved.
13	**SLP_EN:** This is a write-only bit and reads to it always return a zero. Setting this bit causes the system to sequence into the Sleep state defined by the SLP_TYP field.
12:10	**SLP_TYP:** This 3-bit field defines the type of Sleep the system should enter when the SLP_EN bit is set to 1. IFB doesn’t directly support any modes other than those listed in this table. Other Sleep states can be supported in external logic.

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Bits</th><th>Mode</th></tr>
<tr><td>000</td><td>ON</td></tr>
<tr><td>001</td><td>Typically mapped to S1 state. STPCLK# active. CPU in Stop-Grant state. Equivalent to Level 2.</td></tr>
<tr><td>010</td><td>Typically mapped to S1 state. Both STPCLK# and SLP# signals active. CPU in Sleep state. Equivalent to Level 3.</td></tr>
<tr><td>011</td><td>S3 state. This is also known as Suspend-To-RAM (STR). The SUSB signal will go active.</td></tr>
<tr><td>100</td><td>S4/S5 state. The S4 state is also known as Suspend-To-Disk (STD). The S5 state is also known as Soft-Off.</td></tr>
</table>
All other combinations are reserved. If a value is written other than those shown above, the IFB will ignore the value and stay in the ON (000) state. However, the last written value will be readable. For example, if software writes 111 (reserved value), IFB will stay in the ON state, but the next read to the SLP_TYP field will return 111, not 000. Upon reset, this bit is undefined.</td></tr>
<tr><td>9:3</td><td>Reserved.</td></tr>
<tr><td>2</td><td>GBL_RLS: This bit is used by the ACPI software to generate an SMI to the firmware. Firmware has corresponding enable and status bits to control its ability to receive ACPI events.</td></tr>
<tr><td>1</td><td>Reserved.</td></tr>
<tr><td>0</td><td>SCI_EN: Selects the SCI interrupt for the THRM_STS and Timer. When this bit is 1, then these events will generate an SCI interrupt. When this bit is 0, these events will generate an SMI#.</td></tr>
</table>

11.2.7.4 Power Management 1 Timer

Address Offset: 08-0Bh
Attributes: Read Only
Default Value: 00000000h
Size: 32 bits

Bit	Description
31:24	Reserved.
23:0	**TMR_VAL:** This read-only field returns the running count of the power management timer. This is a 24-bit counter that runs off a 3.579545 MHz clock. The timer is reset to an initial value of zero during a PCI reset, starts running immediately after PCI reset, and then continues counting until the 14.31818 MHz input to the chip is stopped (which can only occur in STR/STD and the power is removed to the IFB). Anytime the 22nd bit of the timer goes HIGH to LOW (bits referenced from 0 to 23), the TMROF_STS bit is set. If the TMROF_EN bit is set, an SCI or SMI is also generated.

11.2.7.5 General Purpose 0 Status

Address Offset: 0C-0Dh
Attributes: Read/Write
Default Value: 0800h
Size: 16 bits

When any bit is set in this register, and the corresponding bit is enabled in the General Purpose 0 Enable register, an SCI and a wake event will be generated.

Bit	Description
15:12	Reserved.
11	**PWR_FAIL:** This bit will be set to 1 when a power failure occurs. This is defined as either PWROK or RSMRST# going inactive unexpectedly. This bit is only set by hardware and can only be reset by writing a one to this bit position. This bit is not affected by a hard reset caused by a CF9 write. Upon power up, this bit is set to '1'.
10	**RI_STS:** This bit will be set by 1 to hardware when the RI# input signal goes active. This bit can be reset only by writing a one to this bit position. This bit is not affected by a hard reset caused by a CF9 write.
9	Reserved.
8	**USB1_STS:** This bit is set when the USB controller for ports 0 and 1 needs to cause a wake/break event. Additionally if the USB1_EN bit is set, the setting of the USB1_STS bit will generate a wake/break event. This bit is only set by hardware and can only be reset by writing a one to this bit position. This bit is not affected by a hard reset caused by a CF9 write.
7	**THRMOR_STS:** This is the thermal interrupt over-ride status. This bit is 1 anytime a thermal over-ride condition occurs and starts throttling the CPU's clock at the THRM_DTY ratio. This bit is set by hardware and can only be cleared by writing a one to this bit position.
6:2	Reserved.
1	**NMI_STS:** This indicates that an external device generated SERR# or IOCHK#. If the NMI_EN bit is set in the General Purpose 0 Enable Register, then the setting of this bit will generate an SCI. This bit is set by hardware and can only be cleared by writing a '1' to this bit position.
0	**THRM_STS:** This is the thermal interrupt status bit. This bit gets set anytime the THRM# signal is driven active as defined by the THRM_POL bit. Additionally if the THRM_EN bit is set then the setting of the THRM_STS bit will additionally generate a power management event (SCI or SMI). This bit is only set through hardware (the THRM# signal being driven active), and is cleared by software writing a one to this bit position.

11.2.7.6 General Purpose 0 Enable

Address Offset: OE-0Fh
Attributes: Read/Write
Default Value: 0000h
Size: 16 bits

Bit	Description
15:13	Reserved.
12	**AFTERG3:** If set to 0, the IFB will boot the system after the power is returned after a power failure. If set to 1, the IFB will not boot the system after a power failure and will wait for a wake event (such as the Power Button being pressed). This bit is automatically set to 1 if a Power Button Override occurs. Upon power up, this bit is undefined.
11	Reserved.
10	**RI_EN:** When RI_EN and RI_STS are both set, a Wake event will occur. If RI_EN is not set, then when RI_STS is set, no Wake event will occur. This bit must be maintained even in the G3 state. This bit is automatically set to 0 if a Power Button Override occurs. Upon power up, this bit is undefined.
9	Reserved.
8	**USB1_EN:** This bit is used to enable the setting of the USB1_STS bit to generate a wake/break event. The USB1_STS bit is set anytime the USB controller for ports 0 and 1 signals a wake/break event. The value of this bit must be maintained, even through a G3 state. The IFB will not resume, from USB, after power failure (RSMRST# low). This bit is automatically set to 0 if a Power Button Override occurs. Upon power up, this bit is undefined.
7:2	Reserved.

Bit	Description
1	**NMI_EN:** This bit enables an SCI to be generated on a NMI event. Upon power up, this bit is set to ‘0’.
0	**THRM_EN:** This is the thermal enable bit. When this bit is set an active level assertion of the THRM# signal (as defined by the THRM_POL bit) will set the THRM_STS bit and generate a power management event (an SCI or SMI). Upon power up, this bit is set to ‘0’.

11.2.7.7 General Purpose 1 Status

Address Offset: 16-17h
Attributes: Read/Write
Default Value: 0000h
Size: 16 bits

When any bit is set in this register, and the corresponding bit is enabled in the General Purpose 1 Enable register, an SCI and a wake event will be generated.

Bit	Description
15:9	Reserved.
8:0	**GPIO_STS:** Each bit corresponds to a single GPIO bit (Ex: Bit 8 refers to GPIO[8]. This bit is set when the data bit in the GPIO data register is set to a ‘1’, and cleared by software writing a ‘1’ to this bit location. Additionally, if the GPIO_EN bit in the General Purpose 1 Enable Register is set, then the setting of this bit will generate an SCI or wake event.

11.2.7.8 General Purpose 1 Enable

Address Offset: 18-19h
Attributes: Read/Write
Default Value: 0000h
Size: 16 bits

Bit	Description
15:9	Reserved.
8:0	**GPIO_EN:** Each bit corresponds to a single GPIO bit (Ex: Bit 8 refers to GPIO8). When set, the setting of the corresponding bit in the General Purpose 1 Status Register will generate an SCI or wake event. These bits are not cleared on a CF9 reset.

11.2.8 SMI Registers

In addition to the ACPI I/O registers, there are some generic registers that are defined for legacy based SMI# logic. These I/O registers are added to the end of the I/O register space defined by the ACPI block. All reserved bits will always return 0 when read, and will have no effect when written.

11.2.8.1 Global Control and Enable

Address Offset: 1A-1Bh
Attributes: Read/Write
Default Value: Bits 8 Undefined, Bit 3 ‘1’, All other bits ‘0’
Size: 16 bits

Bit	Description
15:13	Reserved.
12	**ACPI_TMR_EN:** If not using ACPI (SCI) mode, as indicated by SCI_EN not set, then the ACPI_TMR_EN bit can be set to cause an SMI#. SMI will be generated if this bit, SMI_EN and TMROF_EN are set, and SCI_EN is not set.
11	**Reserved.**
10	**APMC_EN:** Software sets this bit to 1 to enable SMIs based upon accesses to the APM control port at B2h in I/O space.
9	**ECC_EN:** This bit is used to enable the generation of SMI when the ECC_STS bit in the Global Status Register is set.
8	**EXTSMI_EN:** This bit is used to enable the setting of the EXTSMI_STS bit to generate an SMI event (SMI# or Wake/Break). The EXTSMI_STS bit is set anytime a GPIO is programmed as an input, is routed to generate an SMI#, and has its register bit set. Upon power up, this bit is undefined.
7	**SWSMI_TMR_EN:** Software sets this bit to a 1 to start the Software SMI# Timer. When the 128 ms timer expires (±4.57 ms), it will generate an SMI# and set the SWSMI_TMR_STS bit. The SWSMI_TMR_EN bit will remain at 1 until software sets it back to 0. Once the timer has been started, if SWSMI_TMR_EN bit is set to 0 before the timer expires, the timer will not expire, and the SMI# will not be generated. The default for this bit is 0.
6	**1MIN_EN:** Enables the 1 minute timer (±1.17s) to count. When it reaches its timeout, it will generate an SMI#.
5	Reserved.
4	**BIOS_EN:** Enables the generation of SMI# when ACPI software writes a 1 to the GBL_RLS bit.
3	**EOS:** End of SMI. This bit controls the arbitration of the SMI signal to the processor. When this bit is set, SMI# will be deasserted. Also this bit will be automatically cleared once IFB asserts SMI# low. In the SMI handler, the processor should clear all pending SMIs (by servicing them and then clearing their respective status bits) and then setting this bit and resuming. This will allow the SMI arbiter to re-assert SMI upon detection of an SMI event and the setting of a SMI status bit.
2	**THRM_POL:** This bit controls the polarity of the THRM# pin needed to set the THRM_STS bit. When the THRM_POL bit is LOW then a HIGH value on the THRM# signal will set the THRM_STS bit.
1	**BIOS_RLS:** Enables the generation of an SCI interrupt for ACPI software when a one is written to this bit position by firmware software. This bit always reads a zero.
0	**SMI_EN:** When set, this bit enables the generation of SMIs in the system upon any enabled SMI event. The exception is when the BIOS Write Enable bit of the BOISEN register is set, an SMI is generated regardless of the state of the SMI_EN bit. This bit is reset by a PCI reset event.

11.2.8.2 Global Status Register

Address Offset: 1Ch-1Dh
Attributes: Read/Write
Default Value: 0000h
Size: 16 bits

Bit	Description
15:11	Reserved.
10	**ECC_STS:** This bit is set when ECCINT# asserted for more than 1 PCI clock with the ECC SERR# bit in configuration space set, or for one or more clocks if the ECC SERR# bit is not set. The setting of this bit will cause an SMI if the ECC_EN bit in the Global Control and Enable register (offset 1A-1Bh) is set. This bit is only set by hardware and can only be cleared by writing a 1 to this bit position.
9	**BIOSWEN_STS:** This bit is set when the BIOSWEN bit in configuration register 4Eh of Function 0 is set. The setting of this bit will cause an SMI. This bit is only set by hardware and can only be reset by writing a one to this bit position.
8	**EXTSMI_STS:** This bit is set when a GPIO is programmed as an input, is routed to generate an SMI#, and its register bit is set. Additionally if the EXTSMI_EN bit is set, the setting of the EXTSMI_STS bit will cause the following, depending on the current state: If in the S0/C0, S0/C1 or S0/C2 states: An SMI event will be generated. If in an S1 state, then a Wake event will be generated, and SMI# will also be generated. If in an S3-S5 state, a Wake event will be generated, but no SMI# will be generated. This bit is only set by hardware and can only be reset by writing a one.
7	**SWSMI_TMR_STS:** This bit will be set to 1 by the hardware when the Software SMI# Timer expires. This bit will remain 1 until software writes a 1 to this bit. The default of this bit is 0.
6	**1MIN_STS:** This hardware will set this bit to 1 when the 1-minute timer expires. This will also cause an SMI#. This bit will remain a 1 until the software writes a 1 to this bit. The default of this bit is 0.
5	**GP_REG_STS:** This bit is an OR of the bits in the ACPI General Purpose Register (offset 0Ch).
4	**PM1_REG_STS:** This bit is an OR of the Power Management 1 Status Register (offset 00h) bits.
3	**APM_STS:** SMI# was generated by a write access to the APM control register. An SMI is generated when the APMC_EN bit is set. This bit is cleared by writing a 1.
2	Reserved.
1	**LEGACY_USB_STS:** This bit is an OR of the SMI status bits in the USB Legacy Keyboard Register ANDed with the enable bits.
0	**BIOS_STS:** SMI# was generated due to ACPI software requesting attention (writing a 1 to the GBL_RLS bit with the BIOS_EN bit set). This bit is set by hardware and cleared by software writing a 1.

11.2.9 General Purpose I/O Registers

For the following GPIO Registers, bits 28:16 refer to the Muxed GPIO signals, and bits 8:0 refer to the dedicated GPIO signal.

Bit	GPIO	Bit	GPIO
28	GPIO[22]	8	GPIO[8]
27	GPIO[21]	7	GPIO[7]
26	GPIO[20]	6	GPIO[6]

Bit	GPIO	Bit	GPIO
25	GPIO[19]	5	GPIO[5]
24	GPIO[18]	4	GPIO[4]
23	Reserved	3	GPIO[3]
22	Reserved	2	GPIO[2]
21	Reserved	1	GPIO[1]
20	Reserved	0	GPIO[0]
19	GPIO[13]		
18	GPIO[12]		
17	GPIO[11]		
16	GPIO[10]		

11.2.9.1 GP Output

Offset: 00-03h
Attribute: Read/Write
Default Value: 00000000h
Size: 32 bits

Bit	Description
31:29	Reserved.
28:24	**Mux Output:** When set to a '0', the muxed GPIO pin is programmed as an input. When set to '1', the muxed GPIO pin is programmed as an output. In the GPO mode, this bit cannot be changed once the GP Lock bit is set. The setting of this bit only has effect if the muxed GPIO is programmed to be a GPIO.
23:20	Reserved.
19:16	**Mux Output:** When set to a '0', the muxed GPIO pin is programmed as an input. When set to '1', the muxed GPIO pin is programmed as an output. In the GPO mode, this bit cannot be changed once the GP Lock bit is set. The setting of this bit only has effect if the muxed GPIO is programmed to be a GPIO.
15:9	Reserved.
8:0	**Output:** When set to a '0', the GPIO pin is programmed as an input. When set to '1', the GPIO pin is programmed as an output. This bit cannot be changed once the GP Lock bit is set.

11.2.9.2 GP Data

Offset: 04-07h
Attribute: Read/Write
Default Value: 00000000h
Size: 32 bits

Bit	Description
31:29	Reserved.
28:24	**Muxed Data:** If a data bit is programmed to be an output, then this bit can be updated by software to drive a value on the output pin. If the data bit is programmed as an input, then this bit reflects the state of the input pin and cannot be updated by software. This bit cannot be changed once the GP Lock bit is set. The value of this bit only has meaning if the muxed GPIO is enabled as a GPIO.
23:20	Reserved.

Bit	Description
19:16	**Muxed Data:** If a data bit is programmed to be an output, then this bit can be updated by software to drive a value on the output pin. If the data bit is programmed as an input, then this bit reflects the state of the input pin and cannot be updated by software. This bit cannot be changed once the GP Lock bit is set. The value of this bit only has meaning if the muxed GPIO is enabled as a GPIO.
15:9	Reserved.
8:0	**Data:** If a data bit is programmed to be an output, then this bit can be updated by software to drive a value on the output pin. If the data bit is programmed as an input, then this bit reflects the state of the input pin and cannot be updated by software. This bit cannot be changed once the GP Lock bit is set.

11.2.9.3 GP TTL

Offset: 08-0Bh
Attribute: Read/Write
Default Value: 00000000h
Size: 32 bits

Bit	Description
31:29	Reserved.
28:24	**Muxed TTL:** When set to a ‘1’, and the data bit is programmed as an output, the pin will be driven with the value of the pin. When set to a ‘0’, and the data bit is programmed as an output, the pin will be tri-stated when the pin is to be driven to a ‘1’, and driven when the pin is to be driven to a ‘0’. The setting of this bit has no effect if the pin is programmed as an input. This bit cannot be changed once the GP Lock bit is set. The value of this bit only has meaning if the muxed GPIO is enabled as a GPIO.
23:20	Reserved.
19:16	**Muxed TTL:** When set to a ‘1’, and the data bit is programmed as an output, the pin will be driven with the value of the pin. When set to a ‘0’, and the data bit is programmed as an output, the pin will be tri-stated when the pin is to be driven to a ‘1’, and driven when the pin is to be driven to a ‘0’. The setting of this bit has no effect if the pin is programmed as an input. This bit cannot be changed once the GP Lock bit is set. The value of this bit only has meaning if the muxed GPIO is enabled as a GPIO.
15:9	Reserved.
8:0	**TTL:** When set to a ‘1’, and the data bit is programmed as an output, the pin will be driven with the value of the pin. When set to a ‘0’, and the data bit is programmed as an output, the pin will be tri-stated when the pin is to be driven to a ‘1’, and driven when the pin is to be driven to a ‘0’. The value of the pin is determined by XORing the data bit with the invert bit. The setting of this bit has no effect if the pin is programmed as an input. This bit cannot be changed once the GP Lock bit is set.

11.2.9.4 GP Blink

Offset: 0C-0Fh
Attribute: Read/Write
Default Value: 00000000h
Size: 32 bits

Bit	Description
31:9	Reserved.
8:0	**Blink:** When set to a '1', and the GP pin is programmed as an output, it will blink at a rate of once per second. The value of the data bit remains unchanged during the blink process. If it was set, it remains set. The setting of this bit has no effect if the pin is programmed as an input. This bit cannot be changed once the GP Lock bit is set.

11.2.9.5 GP Lock

Offset: 10-13h
Attribute: Read/Write
Default Value: 00000000h
Size: 32 bits

Bit	Description
31:29	Reserved.
28:24	**Muxed Lock:** When set, and the pin is programmed as an output, the data bit cannot be changed. Once this bit is set, it can only be cleared by a PCIRST#. Once this bit is set, all other register bits at this bit location cannot be changed.
23:20	Reserved.
19:16	**Muxed Lock:** When set, and the pin is programmed as an output, the data bit cannot be changed. Once this bit is set, it can only be cleared by a PCIRST#. Once this bit is set, all other register bits at this bit location cannot be changed.
15:9	Reserved.
8:0	**Lock:** When set, and the pin is programmed as an output, the data bit cannot be changed. Once this bit is set, it can only be cleared by a PCIRST#. Once this bit is set, all other register bits at this bit location cannot be changed. This is applicable to GPO mode only.

11.2.9.6 GP Invert

Offset: 14-17h
Attribute: Read/Write
Default Value: 00000000h
Size: 32 bits

Bit	Description
31:9	Reserved.
8:0	**Invert:** When set to a '0', the GPIO pin is not inverted. When set to '1', the GPIO pin is inverted. When set to an output, the data bit is inverted before it is driven on the pin. When set to an input, the data bit is inverted before entering the data register. This bit cannot be changed once the GP Lock bit is set.

11.2.9.7 GP SMI

Offset: 1C-1Fh
Attribute: Read/Write
Default Value: 00000000h
Size: 32 bits

Bit	Description
31:9	Reserved.
8:0	**SMI Rout:** When set to a '1', and the corresponding data bit is set to an input, a '1' in the data bit register will be routed to an SMI. If the data bit is set to an output, this value of this bit has no effect. When cleared, no routing is performed. This bit cannot be changed once the GP Lock bit is set.

11.2.9.8 GP Pulse

Offset: 20-23h
Attribute: Read/Write
Default Value: 00000000h
Size: 32 bits

Bit	Description
31:9	Reserved.
8:0	**Pulse:** When set to a '1', and the data bit (after the invert bit) is programmed as an input, a '0' to '1' transition that is longer than 2 RTC clocks will cause the data bit to be set. A '1' to '0' transition will not clear the bit. Only a write of '1' to the data bit can clear the data bit. If the data bit is not set to an output, this value of this bit has no effect. When cleared, edge triggering is not performed. This bit cannot be changed once the GP Lock bit is set.

11.2.9.9 GP Core

Offset: 24-27h
Attribute: Read/Write
Default Value: 00000000h
Size: 32 bits

Bit	Description
31:9	Reserved.
8:0	**Core:** When set to a '1', and the data bit is programmed as an output, a '0' will be driven on the pin when core power loss is detected (PWROK low). This allows the pin, while in the resume well, to be connected to a device in the core well. When cleared, this check is not performed. This bit cannot be changed once the GP Lock bit is set.

11.2.9.10 GP Pull-up

Offset: 28-2Bh
Attribute: Read/Write
Default Value: 03FFh
Size: 32 bits

Bit	Description
31:10	Reserved.
9:0	**Pull-up:** When set, an internal pull-up will be enabled on the pin. When disabled, the pull-up is disabled. This bit cannot be changed once the GP Lock bit is set.

IDE Configuration 12

The IFB PCI Function 1 contains an IDE Controller capable of Programmed I/O (PIO) transfers as well as Bus Master transfer capability. It also supports the “Ultra DMA/33” synchronous DMA mode of data transfer. The register set associated with IDE Controller is shown below.

12.1 PCI Configuration Registers (Function 1)

Table 12-1. PCI Configuration Registers–Function 1 (IDE Interface)

Configuration Offset	Mnemonic	Register	Register Access
00–01h	VID	Vendor Identification	RO
02–03h	DID	Device Identification	RO
04–05h	PCICMD	PCI Command	R/W
06–07h	PCISTS	PCI Device Status	R/W
08h	RID	Revision Identification	RO
09-0Bh	CLASSC	Class Code	RO
0Ch	–	Reserved	–
0Dh	MLT	Master Latency Timer	R/W
0Eh	HEDT	Header Type	RO
0F–1Fh	–	Reserved	–
20–23h	BMIBA	Bus Master Interface Base Address	R/W
24–3Bh	–	Reserved	–
2C–2Dh	SVID	Subsystem Vendor ID	RO
2E–2Fh	SID	Subsystem ID	RO
30–3Fh	–	Reserved	–
40–43h	IDETIM	IDE Timing	R/W
44h	SIDETIM	Slave IDE Timing	R/W
45–47h	–	Reserved	–
48h	SDMACTL	Synchronous DMA Control	R/W
49h	–	Reserved	–
4A–4Bh	SDMATIM	Synchronous DMA Timing	R/W
4C–FFh	–	Reserved	–

12.2 IDE Controller Register Descriptions (PCI Function 1)

This section describes in detail the registers associated with the IFB IDE Controller Function. This includes Programmed I/O (PIO), Bus Master, and “Ultra DMA/33” synchronous DMA Functionality.

12.2.1 VID–Vendor Identification Register (Function 1)

Address Offset: 00–01h
Default Value: 8086h
Attribute: Read only

The VID Register contains the vendor identification number. This register, along with the Device Identification Register, uniquely identifies any PCI device. Writes to this register have no effect.

Bit	Description
15:0	**Vendor Identification Number.** This is a 16-bit value assigned to Intel

12.2.2 DID–Device Identification Register (Function 1)

Address Offset: 02–03h
Default Value: 7601h
Attribute: Read only

The DID Register contains the device identification number. This register, along with the VID Register, defines the IFB Function. Writes to this register have no effect.

Bit	Description
15:0	**Device Identification Number.** This is a 16-bit value assigned to the IFB IDE Controller Function.

12.2.3 PCICMD–PCI Command Register (Function 1)

Address Offset: 04–05h
Default Value: 0000h
Attribute: Read/Write

The PCICMD Register controls access to the I/O space registers.

Bit	Description
15:10	Reserved.
9	**Fast Back to Back Enable (FBE).** This bit is hardwired to 0.
8	**SERR# Enable.** This bit is hardwired to 0.
7	**Wait Cycle Control.** This bit is hardwired to 0.
6	**Parity Error Response.** This bit is hardwired to 0.
5	**VGA Palette Snoop.** This bit is hardwired to 0.
4	**Postable Memory Write and Invalidate Enable.** This bit is hardwired to 0.
3	**Special Cycle Enable.** This bit is hardwired to 0.
2	**Bus Master Function Enable (BME).** 1=Enable. 0=Disable.
1	**Memory Space Enable.** This bit is hardwired to 0.
0	**I/O Space Enable (IOSE).** This bit controls access to the I/O space registers. When IOSE=1, access to the Legacy IDE ports (both primary and secondary) and the PCI Bus Master IDE I/O Registers is enabled. The Base Address Register for the PCI Bus Master IDE I/O Registers should be programmed before this bit is set to 1.

12.2.4 PCISTS–PCI Device Status Register (Function 1)

Address Offset: 06–07h
Default Value: 0280h
Attribute: Read/Write

PCISTS is a 16-bit status register for the IDE interface Function. The register also indicates the IFB's DEVSEL# signal timing.

Bit	Description
15	**Detected Parity Error.** Read as 0.
14	**SERR# Status.** Read as 0.
13	**Master-Abort Status (MAS)–R/WC.** When the Bus Master IDE interface Function, as a master, generates a master abort, MAS is set to a 1. Software sets MAS to 0 by writing a 1 to this bit.
12	**Received Target-Abort Status (RTA)–R/WC.** When the Bus Master IDE interface Function is a master on the PCI Bus and receives a target abort, this bit is set to a 1. Software sets RTA to 0 by writing a 1 to this bit.
11	**Signaled Target Abort Status (STA)–R/WC.** This bit is set when the IFB IDE interface Function is targeted with a transaction that the IFB terminates with a target abort. Software resets STA to 0 by writing a 1 to this bit.
10:9	**DEVSEL# Timing Status (DEVT)–RO.** For the IFB, DEVT=01 indicating medium timing for DEVSEL# assertion when performing a positive decode. DEVSEL# timing does not include configuration cycles.
8	**Data Parity Detected (DPD).** Read as 0.
7	**Fast Back to back Capable (FBC)–RO.** Hardwired to a 1.
6:0	Reserved.

12.2.5 CLASSC–Class Code Register (Function 1)

Address Offset: 09-0Bh
Default Value: 010180h
Attribute: Read only

This register identifies the Base Class Code, Sub-Class Code, and Device Programming interface for IFB PCI Function 1.

Bit	Description
23:16	**Base Class Code (BASEC).** 01h=Mass storage device.
15:8	**Sub-Class Code (SCC)**. 01h=IDE controller.
7:0	**Programming Interface (PI).** 80h=Capable of IDE bus master operation.

12.2.6 MLT–Master Latency Timer Register (Function 1)

Address Offset: 0Dh
Default Value: 00h
Attribute: Read/Write

MLT controls the amount of time IFB, as a bus master, can burst data on the PCI Bus. The count value is an 8 bit quantity. However, MLT[3:0] are reserved and 0 when determining the count value. The Master Latency Timer is cleared and suspended when IFB is not asserting FRAME#. When IFB asserts FRAME#, the counter begins counting. If the IFB finishes its transaction before the count expires, the MLT count is ignored. If the count expires before the transaction completes (count = # of clocks programmed in MLT), IFB initiates a transaction termination as soon as its PHLDA# is removed. The number of clocks programmed in the MLT represents the time slice (measured in PCI clocks) allotted to IFB. The default value of MLT is 00h or 0 PCI clocks.

Bit	Description
7:4	**Master Latency Timer Count Value (MLTC).** IFB-initiated PCI burst cycles can last indefinitely, as long as PHLDA# remains active. However, if PHLDA# is negated after the burst cycle is initiated, IFB limits the burst cycle to the number of PCI Bus clocks specified by this field.
3:0	Reserved.

12.2.7 BMIBA–Bus Master Interface Base Address Register (Function 1)

Address Offset: 20–23h
Default Value: 00000001h
Attribute: Read/Write

This register selects the base address of a 16 byte I/O space to provide a software interface to the Bus Master Functions. Only 12 bytes are actually used (6 bytes for primary and 6 bytes for secondary).

This register selects the base address of a 16 byte I/O space to provide a software interface to the Bus Master Functions. Only 12 bytes are actually used (6 bytes for primary and 6 bytes for secondary).

Bit	Description
31:16	Reserved.
15:4	**Bus Master Interface Base Address (BMIBA).** These bits provide the base address for the Bus Master interface registers and correspond to AD[15:4].
3:1	Reserved.
0	**Resource Type Indicator (RTE)–RO.** This bit is hardwired to 1 indicating that the base address field in this register maps to I/O space.

12.2.8 SVID–Subsystem Vendor ID (Function 1)

Address: 2C-2Dh
Default Value: 0000h
Attribute: Read only

Bit	Description
15:0	**Subsystem Vendor ID.**

12.2.9 SID–Subsystem ID (Function 1)

Address: 2E-2Fh
Default Value: 0000h
Attribute: Read only

Bit	Description
15:0	Subsystem ID.

12.2.10 IDETIM–IDE Timing Register (Function 1)

Address Offset: 40–41h = Primary Channel 42–43h = Secondary Channel
Default Value: 0000h
Attribute: Read/Write only

This register controls the IFB's IDE interface and selects the timing characteristics of the PCI Local Bus IDE cycle for PIO and Bus Master transfers. Note that primary and secondary denotations distinguish between the cables and the 0/1 denotations distinguish between master (0) and slave (1).

Bit	Description
15	**IDE Decode Enable (IDE).** 1=Enable. 0=Disable. When enabled, I/O transactions on PCI targeting the IDE ATA register blocks (command block and control block) are positively decoded on PCI and driven on the IDE interface. When disabled, these accesses are subtractively decoded to LPC.
14	Slave IDE Timing Register Enable (SITRE). 0 Use bits 13:12, 9:8 for both drive 0 and drive 1. 1 Use bits 13:12, 9:8 for drive 0, Slave IDE timing register for drive 1.
13:12	**IORDY Sample Point (ISP).** This field selects the number of PCI clocks between IOR#/IOW# assertion and the first IORDY sample point. Bits[13:12] Number of Clocks 00 5 01 4 10 3 11 2
11:10	Reserved.

Bit	Description
9:8	**Recovery Time (RTC).** This field selects the minimum number of PCI clocks between the last IORDY# sample point and the DIOx# strobe of the next cycle. Bits[9:8] Number of Clocks 00 4 01 3 10 2 11 1
7	**DMA Timing Enable Only (DTE1).** When DTE1=1, fast timing mode is enabled for DMA data transfers for drive 1. PIO transfers to the IDE data port will run in compatible timing. When DTE1 = 0, both DMA and PIO data transfers for drive 1 will use the fast timing mode.
6	**Prefetch and Posting Enable (PPE1).** When PPE1=1, prefetch and posting to the IDE data port is enabled for drive 1. When PPE1 = 0, prefetch and posting is disabled for drive 1.
5	**IORDY Sample Point Enable Drive Select 1 (IE1).** When IE1=0, IORDY sampling is disabled for Drive 1. When the internal IORDY signal is forced asserted, IORDY is sampled asserted at the first sample point as specified by the ISP field in this register. When IE1=1 and the currently selected drive (via a copy of bit 4 of 1x6h) is Drive 1, all accesses to the enabled I/O address range sample IORDY. The IORDY sample point is specified by the ISP field in this register.
4	**Fast Timing Bank Drive Select 1 (TIME1).** When cleared, accesses to the data port will use compatible timings for this drive. When set and bit 14 cleared, accesses to the data port will use bits 13:12 for the IORDY sample point, and bits 9:8 for the recovery time. When set and bit 14 set, accesses to the data port will use the IORDY sample point and recover time specified in the slave IDE timing register.
3	**DMA Timing Enable Only (DTE0).** When DTE0=1, fast timing mode is enabled for DMA data transfers for drive 0. PIO transfers to the IDE data port will run in compatible timing. When DTE1 = 0, both DMA and PIO data transfers for drive 0 will use the fast timing mode.
2	**Prefetch and Posting Enable (PPE0).** When PPE0 = 1, prefetch and posting to the IDE data port is enabled for drive 0. When PPE0 = 0, prefetch and posting is disabled for drive 0.
1	**IORDY Sample Point Enable Drive Select 0 (IE0).** When IE0=0, IORDY sampling is disabled for Drive 0. When the internal IORDY signal is forced asserted, IORDY is sampled asserted at the first sample point as specified by the ISP field in this register. When IE0=1 and the currently selected drive (via a copy of bit 4 of 1x6h) is Drive 0, all accesses to the enabled I/O address range sample IORDY. The IORDY sample point is specified by the ISP field in this register.
0	**Fast Timing Bank Drive Select 0 (TIME0).** When TIME0=0, accesses to the data port of the enabled I/O address range uses the 16 bit compatible timing. When TIME0=1 and the currently selected drive (via a copy of bit 4 of 1x6h) is Drive 0, accesses to the data port of the enabled I/O address range use fast timings. PIO accesses to the data port use fast timing only if bit 3 of this register (DTE0) is 0. Accesses to all non-data ports of the enabled I/O address range always use the 8 bit compatible timings.

12.2.11 SIDETIM–Slave IDE Timing Register (Function 1)

Address Offset: 44h
Default Value: 00h
Attribute: Read/Write only

This register controls the IFB's IDE interface and selects the timing characteristics for the slave drives on each IDE channel. This allows for programming of independent operating modes for each IDE agent. This register has no affect unless the SITRE bit is enabled in the IDETIM Register.

Bit	Description
7:6	**Secondary Drive 1 IORDY Sample Point (SISP1).** This field selects the number of PCI clocks between SDIOx# assertion and the first SIORDY sample point for the slave drive on the secondary channel. Bits[7:6] Number of Clocks 00 5 01 4 10 3 11 2
5:4	**Secondary Drive 1 Recovery Time (SRTC1).** This field selects the minimum number of PCI clocks between the last SIORDY# sample point and the SDIOx# strobe of the next cycle for the slave drive on the secondary channel. Bits[5:4] Number of Clocks 00 4 01 3 10 2 11 1
3:2	**Primary Drive 1 IORDY Sample Point (PISP1).** This field selects the number of PCI clocks between PDIOx# assertion and the first PIORDY sample point for the slave drive on the primary channel. Bits[3:2] Number of Clocks 00 5 01 4 10 3 11 2
1:0	**Primary Drive 1 Recovery Time (PRTC1).** This field selects the minimum number of PCI clocks between the last PIORDY# sample point and the PDIOx# strobe of the next cycle for the slave drive on the primary channel. Bits[1:0] Number of Clocks 00 4 01 3 10 2 11 1

12.2.12 DMACTL–Synchronous DMA Control Register (Function 1)

Address Offset: 48h
Default Value: 00h
Attribute: Read/Write

This register enables each individual channel and drive for Synchronous DMA transfers. For non-synchronous DMA operation, this register should be left programmed to its default value.

Bit	Description
7:4	Reserved.
3	**Secondary Drive 1 SDMA Enable (SSDE1).** 1 = Enable Synchronous DMA mode for secondary channel drive 1. 0 = Disable (default).
2	**Secondary Drive 0 SDMA Enable (SSDE0).** 1 = Enable Synchronous DMA mode for secondary channel drive 0. 0 = Disable (default).
1	**Primary Drive 1 SDMA Enable (PSDE1).** 1 = Enable Synchronous DMA mode for primary channel drive 1. 0 = Disable (default).
0	**Primary Drive 0 SDMA Enable (PSDE0).** 1 = Enable Synchronous DMA mode for primary channel drive 0. 0 = Disable (default).

12.2.13 SDMATIM–Synchronous DMA Timing Register (Function 1)

Address Offset: 4A-4Bh
Default Value: 0000h
Attribute: Read/Write only

This register controls the timings used by each Synchronous DMA enabled device. For non-synchronous DMA operation, this register should be left programmed to its default value.

Bit	Description
15:14	Reserved.
13:12	Secondary Drive 1 Cycle Time (SCT1): These bit settings determine the minimum data write strobe Cycle Time (CT) and minimum Ready to Pause time (RP). 00: CT = 4 PCICLK, RP = 6 PCICLK 01: CT = 3 PCICLK, RP = 5 PCICLK 10: CT = 2 PCICLK, RP = 4 PCICLK 11: Reserved
11:10	Reserved.
9:8	**Secondary Drive 0 Cycle Time (SCT0):** These bit settings determine the minimum data write strobe Cycle Time (CT) and minimum Ready to Pause time (RP). 00: CT = 4 PCICLK, RP = 6 PCICLK 01: CT = 3 PCICLK, RP = 5 PCICLK 10: CT = 2 PCICLK, RP = 4 PCICLK 11: Reserved
7:6	Reserved.
5:4	**Primary Drive 1 Cycle Time (PCT1):** These bit settings determine the minimum data write strobe Cycle Time (CT) and minimum Ready to Pause time (RP). 00: CT = 4 PCICLK, RP = 6 PCICLK 01: CT = 3 PCICLK, RP = 5 PCICLK 10: CT = 2 PCICLK, RP = 4 PCICLK 11: Reserved
3:2	Reserved.
1:0	**Primary Drive 0 Cycle Time (PCT0):** These bit settings determine the minimum data write strobe Cycle Time (CT) and minimum Ready to Pause time (RP). 00: CT = 4 PCICLK, RP = 6 PCICLK 01: CT = 3 PCICLK, RP = 5 PCICLK 10: CT = 2 PCICLK, RP = 4 PCICLK 11: Reserved

Table 12-2. Ultra DMA/33 Timing Mode Settings

Cycle Time Bit Settings	Ultra DMA/33 Timing Modes		
	Mode 0 (120 ns)	Mode 1 (90 ns)	Mode 2 (60 ns)
	00	01	10

Table 12-3. DMA/PIO Timing Values Based on IFB Cable Mode and System Speed

IFB Drive Mode	IORDY Sample Point (ISP)	Recovery Time (RCT)	IDETIM [15:8] Drive 0 (Master) If Slave Attached	IDETIM [15:8] Drive 0 (Master) If no Slave Attached or Slave is Mode 0[1]	SIDETIM Pri [3:0] Sec [7:4] Drive 1 (Slave)	Resultant Cycle Time Base Operating Frequency and Cycle Time[2]
PIO0/ Compatible	6 clocks (Default)	1 clocks (Default)	C0h	80h	0	30 MHz: 660ns 33 MHz: 600ns
PIO2/SW2	4 clocks	4 clocks	D0h	90h	4	30 MHz: 256ns 33 MHz: 240ns
PIO3/MW1	3 clocks	3 clocks	E1h	A1h	9	30 MHz: 198ns 33 MHz: 180ns
PIO4/MW2	3 clocks	1 clock	E3h	A3h	B	30 MHz: 132ns 33 MHz: 120ns

NOTES:
1. Table 12-3 assumes that if the attached slave drive is Mode 0 or not present, the SITRE bit is '0'.
2. Table 12-3 assumes that 25 MHz is not supported as a target PCI system speed. If the DMA Timing Enable Only (DTE) bit has been enabled for that drive, this resultant cycle time applies to data transfers performed with DMA only.

12.3 IDE Controller I/O Space Registers

The PCI IDE Function uses 16 bytes of I/O space, allocated via the BMIBA register. All bus master IDE I/O space registers can be accessed as byte, word, or DWord quantities. The description of the 16 bytes of I/O registers follows.

12.3.1 BMICx–Bus Master IDE Command Register (I/O)

Address Offset: Primary Channel–Base + 00h; Secondary Channel–Base + 08h
Default Value: 00h
Attribute: Read/Write

This register enables/disables bus master capability for the IDE Function and provides direction control for the IDE DMA transfers. This register also provides bits that software uses to indicate DMA capability of the IDE device.

Bit	Description
7:4	Reserved.
3	Bus Master Read/Write Control (RWCON). 0=Reads; 1=Writes. This bit must NOT be changed when the bus master Function is active. While a synchronous DMA transfer is in progress, this bit will be READ ONLY. The bit will return to read/write once the synchronous DMA transfer has been completed or halted.
2:1	Reserved.
0	**Start/Stop Bus Master (SSBM).** 1=Start; 0=Stop. When this bit is set to 1, bus master operation starts. The controller transfers data between the IDE device and memory only while this bit is set. Master operation can be stopped by writing a 0 to this bit. This results in all state information being lost (i.e. master mode operation cannot be stopped and then resumed). If this bit is set to 0 while bus master operation is still active (i.e. Bit 0=1 in the Bus Master IDE Status Register for that IDE channel) and the drive has not yet finished its data transfer (bit 2=0 in the channel's Bus Master IDE Status Register), the bus master command is aborted and data transferred from the drive may be discarded by the IFB rather than being written to system memory. This bit is intended to be set to 0 after the data transfer is completed, as indicated by either bit 0 or bit 2 being set in the IDE Channel's Bus Master IDE Status Register.

12.3.2 BMISx–Bus Master IDE Status Register (I/O)

Address Offset: Primary Channel–Base + 02h; Secondary Channel–Base + 0Ah
Default Value: 00h
Attribute: Read/Write Clear

This register provides status information about the IDE device and state of the IDE DMA transfer.

Bit	Description
7	**Reserved.** This bit is hardwired to 0.
6	**Drive 1 DMA Capable (DMA1CAP)–R/W.** 1=Drive 1 is capable of DMA transfers. This bit is a software controlled status bit that indicates IDE DMA device capability and does not affect hardware operation.
5	**Drive 0 DMA Capable (DMA0CAP)–R/W.** 1=Drive 0 is capable of DMA transfers. This bit is a software controlled status bit that indicates IDE DMA device capability and does not affect hardware operation.
4:3	Reserved.
2	**IDE Interrupt Status (IDEINTS)–R/WC.** This bit, when set to a 1, indicates when an IDE device has asserted its interrupt line. When bit 2=1, all read data from the IDE device has been transferred to main memory and all write data has been transferred to the IDE device. Software sets this bit to a 0 by writing a 1 to it. IRQ14 is used for the primary channel and IRQ15 is used for the secondary channel. Note that, if the interrupt status bit is set to a 0 by writing a 1 to this bit while the interrupt line is still at the active level, this bit remains 0 until another assertion edge is detected on the interrupt line.
1	**IDE DMA Error–R/WC.** This bit is set to 1 when the IFB encounters a target abort or master abort while transferring data on the PCI Bus. Software sets this bit to a 0 by writing a 1 to it.
0	**Bus Master IDE Active (BMIDEA)–RO.** The IFB sets this bit to 1 when bit 0 in the BMICx Register is set to 1. The IFB sets this bit to 0 when the last transfer for a region is performed (where EOT for that region is set in the region descriptor). The IFB also sets this bit to 0 when bit 0 of the BMICx Register is set to 0. When this bit is read as a zero, all data transferred from the drive during the previous bus master command is visible in system memory, unless the bus master command was aborted.

Table 12-4. Interrupt/Activity Status Combinations

Bit 2	Bit 0	Description
0	1	**DMA transfer is in progress.** No interrupt has been generated by the IDE device.
1	0	**The IDE device generated an interrupt and the Physical Region Descriptors exhausted.** This is normal completion where the size of the physical memory regions is equal to the IDE device transfer size.
1	1	**The IDE device generated an interrupt.** The controller has not reached the end of the physical memory regions. This is a valid completion case when the size of the physical memory regions is larger than the IDE device transfer size.
0	0	**Error condition.** If the IDE DMA Error bit is 1, there is a problem transferring data to/ from memory. Specifics of the error have to be determined using bus-specific information. If the Error bit is 0, the PRD specified a smaller buffer size than the programmed IDE transfer size.

12.3.3 BMIDTPx–Bus Master IDE Descriptor Table Pointer Register (I/O)

Address Offset: Primary Channel–Base + 04h; Secondary Channel–Base + 0Ch
Default Value: 00000000h
Attribute: Read/Write

This register provides the base memory address of the Descriptor Table. The Descriptor Table must be DWord aligned and not cross a 4 KByte boundary in memory.

Bit	Description
31:2	**Descriptor Table Base Address (DTBA).** Bits [31:2] correspond to A[31:2].
1:0	Reserved.

Universal Serial Bus (USB) Configuration 13

The IFB integrates one USB Controller. The USB Controller is UHCI 1.1 compliant and implements the root hub of the USB, which contains two ports.

The IFB PCI Function 2 reflects the USB Host and Root Hubs, with 2 connected USB ports. The register set associated with USB Host Controller is shown below with actual register descriptions given in Section 13.2 and Section 13.3.

13.1 PCI Configuration Registers (Function 2)

Table 13-1. PCI Configuration Registers–Function 2

Configuration Offset	Mnemonic	Register	Register Access
00–01h	VID	Vendor Identification	RO
02–03h	DID	Device Identification	RO
04–05h	PCICMD	PCI Command	R/W
06–07h	PCISTS	PCI Device Status	R/W
08h	RID	Revision Identification	RO
09-0Bh	CLASSC	Class Code	RO
0Ch	–	Reserved	–
0Dh	MLT	Latency Timer	R/W
0Eh	HEDT	Header Type	RO
0F–1Fh	–	Reserved	–
20–23h	USBBA	USB I/O Space Base Address	R/W
24–3Bh	–	Reserved	–
2C–2Dh	SVID	Subsystem Vendor ID	RO
2E–2Fh	SID	Subsystem ID	RO
30–3Fh	–	Reserved	–
3Ch	INTLN	Interrupt Line	R/W
3Dh	INTPN	Interrupt Pin	RO
3E–5Fh	–	Reserved	–
60h	SBRNUM	Serial Bus Release Number	RO
61–69h	–	Reserved	–
6A–6Bh	MCR	Miscellaneous Control Register	R/W
6C–BFh	–	Reserved	–
C0–C1h	LEGSUP	Legacy Support	R/W
C2-C3h	---	Reserved	---
C4h	USBREN	USB Resume Enable	R/W
C5-FF	---	Reserved	---

13.2 USB Host Controller Register Descriptions (PCI Function 2)

This section describes in detail the registers associated with the IFB USB Host Controller Functions. This includes UHCI compatible registers and Legacy Keyboard registers.

13.2.1 VID–Vendor Identification Register (Function 2)

Address Offset: 00–01h
Default Value: 8086h
Attribute: Read only

The VID Register contains the vendor identification number. This register, along with the Device Identification Register, uniquely identifies any PCI device. Writes to this register have no effect.

Bit	Description
15:0	**Vendor Identification Number.** This is a 16-bit value assigned to Intel.

13.2.2 DID–Device Identification Register (Function 2)

Address Offset: 02-03h
Default Value: 7602h
Attribute: Read only

The DID Register contains the device identification number. This register, along with the VID Register, defines the IFB USB Host Controller. Writes to this register have no effect.

Bit	Description
15:0	**Device Identification Number.** This is a 16-bit value assigned to the IFB USB Host Controller.

13.2.3 PCICMD–PCI Command Register (Function 2)

Address Offset: 04-05h
Default Value: 00h
Attribute: Read/Write

This register controls access to the I/O space registers.

Bit	Description
15:10	**Reserved.** Read 0.
9	**Fast Back to Back Enable** (**Not Implemented).** This bit is hardwired to 0.
8:5	**Reserved.** Read as 0.
4	**Memory Write and Invalidate Enable (Not Implemented).** This bit is hardwired to 0.
3	**Special Cycle Enable (Not Implemented).** This bit is hardwired to 0.
2	**Bus Master Enable (BME).** This bit controls the IFB's ability to act as a master on the PCI bus for the host controller transfers. A value of 0 disables the device from generating PCI accesses. A value of 1 allows the device to behave as a USB host controller bus master. This bit must be set to 1 before USB transactions can start.

Bit	Description
1	**Memory Space Enable (Not Implemented).** This bit is hardwired to 0.
0	**I/O Space Enable (IOSE).** 1=Enable. 0=Disable. This bit controls the access to the I/O space registers. If this bit is set, access to the host controller I/O registers is enabled. The base register for the I/O registers must be programmed before this bit is set.

13.2.4 PCISTS–PCI Device Status Register (Function 2)

Address Offset: 06-07h
Default Value: 0280h
Attribute: Read/Write

DSR is a 16-bit status register that reports the occurrence of a PCI master-abort by the USB HC module or a PCI target-abort when the Serial Bus module is a master. The register also indicates the USB HC module DEVSEL# signal timing that is hardwired in the USB HC module.

Bit	Description
15	**Detected Parity (Not Implemented).** Read as 0.
14	**SERR# Status (Not Implemented).** Read as 0.
13	**Master-Abort Status (MAS)–R/WC.** When the Serial Bus module receives a master-abort from a PCI transaction, MAS is set to a 1. Software sets MAS to 0 by writing a 1 to this bit.
12	**Received Target-Abort Status (RTA)–R/WC.** When the Serial Bus module is a master on the PCI Bus and receives a target-abort, this bit is set to a 1. Software resets RTA to 0 by writing a 1 to this bit.
11	**Signaled Target-Abort Status (STA)–R/WC.** This bit is set when the Serial Bus module Function is targeted with a transaction that the Serial Bus module terminates with a target abort. Software resets STA to 0 by writing a 1 to this bit.
10:9	**DEVSEL# Timing Status (DEVT)–RO.** This 2-bit field defines the timing for DEVSEL# assertion. These read only bits indicate the IFB's DEVSEL# timing when performing a positive decode. Since the IFB always generate the DEVSEL# with medium timing, DEVT=01. This DEVSEL# timing does not include Configuration cycles.
8	**Data Parity Detected (Not Implemented).** Read as 0.
7	**Fast Back to Back Capable (FBC)–RO.** Hardwired to 1. This bit indicates to the PCI Master that Serial Bus module as a target is capable of accepting fast back-to-back transactions.
6:0	**Reserved.** Read as 0's.

13.2.5 RID–Revision Identification Register (Function 2)

Address Offset: 08h
Default Value: Initial Stepping = 00h. Refer to IFB Specification Updates for other values programmed here.
Attribute: Read only

This 8 bit register contains device stepping information. Writes to this register have no effect.

Bit	Description
7:0	**Revision ID Byte.** The register is hardwired to the default value.

13.2.6 CLASSC–Class Code Register (Function 2)

Address Offset: 0A-0Bh
Default Value: 0C03h
Attribute: Read only

This register identifies the Base Class Code, Sub-Class Code, and Device Programming interface for IFB PCI Function 2.

Bit	Description
23:16	**Base Class Code (BASEC).** 0Ch=Serial Bus controller.
15:8	**Sub-Class Code (SCC)**. 03h=USB Host Controller..
7:0	**Programming Interface (PI).** 00h=Universal Host Controller Interface.

13.2.7 MLT–Master Latency Timer Register (Function 2)

Address Offset: 0Dh
Default Value: 00h
Attribute: Read/Write

MLT is an 8-bit register that controls the amount of time (in terms of PCI clocks) the USB module can do transactions on the PCI bus. The count value is an 8 bit quantity, however MLT[3:0] are reserved and assumed to be 0 when determining the count value. MLT is used when the USB module becomes the PCI bus master and is cleared and suspended when IFB is not asserting FRAME#. When IFB asserts FRAME#, the counter is enabled and begins counting. If the serial bus module finishes its transaction before count is expired, the MLT value is ignored. If the count expires before the transaction completes, IFB initiates a transaction termination as soon as the current transaction is completed.. The number of clocks programmed in the MLT represents the time slice (measured in PCI clocks) allotted to IFB, after which it must surrender the bus as soon as the current transaction is completed.

Bit	Description
7:4	**Master Latency Counter Value.** IFB initiated PCI cycles (including multiple transactions) can last indefinitely as long as PHLDA# remains active. However, if PHLDA# is negated after a transaction is initiated, IFB limits the duration of the transactions to the number of PCI bus clocks specified by this field.
3:0	Reserved.

13.2.8 HEDT–Header Type Register (Function 2)

Address Offset: 0Eh
Default Value: 00h
Attribute: Read only

This register identifies the Serial Bus module as a single Function device.

Bit	Description
7:0	**Device Type (DEVICET).** 00. Multi-Function device capability for IFB is defined by the HEDT register in Function 0.

13.2.9 USBBA–USB I/O Space Base Address (Function 2)

Address Offset: 20-23h
Default Value: 00000001h
Attribute: Read/Write

This register contains the base address of the USB I/O Registers.

Bit	Description
31:16	**Reserved.** Hardwired to 0s. Must be written as 0S.
15:5	**Index Register Base Address.** Bits [15:5] correspond to I/O address signals AD [15:5], respectively.
4:1	**Reserved.** Read as 0.
0	**Resource Type Indicator (RTE)–RO.** This bit is hardwired to 1 indicating that the base address field in this register maps to I/O space.

13.2.10 SVID–Subsystem Vendor ID (Function 2)

Address: 2C-2Dh
Default Value: 0000h
Attribute: Read only

Bit	Description
15:0	**Subsystem Vendor ID.**

13.2.11 SID–Subsystem ID (Function 2)

Address: 2E-2Fh
Default Value: 0000h
Attribute: Read only

Bit	Description
15:0	Subsystem ID.

13.2.12 INTLN–Interrupt Line Register (Function 2)

Address Offset: 3Ch
Default Value: 00h
Attribute: Read/Write

Software programs this register with interrupt information concerning the USB.

Bit	Description
7:0	**Interrupt Line.** The value in this register has no affect on IFB hardware operations.

13.2.13 INTPN–Interrupt Pin (Function 2)

Address Offset: 3Dh
Default Value: 04h
Attribute: Read only

This register indicates which PCI interrupt pin is used for the USB module interrupt. The USB interrupt is internally ORed to the interrupt controller with the PIRQD# signal.

Bit	Description
7:3	**Reserved**.
2:0	**Serial Bus Module Interrupt Routing.** The value of 04h in Function 2 indicates that the IFB will drive PIRQD# as its interrupt line for the USB controller.

13.2.14 Miscellaneous Control (Function 2)

Address Offset: 6A-6Bh
Default Value: 0000h
Attribute: Read/Write

Bit	Description
15:2	Reserved.
1	**Low Speed PreSOF Disable.** This bit should be set to '1' to disable.
0	Reserved.

13.2.15 SBRNUM–Serial Bus Release Number (Function 2)

Address Offset: 60h
Default Value: 10h
Attribute: Read only

This register contains the release of the USB Specification with which this USB Host Controller module is compliant.

Bit	Description
7:0	**Serial Bus Specification Release Number.** All other combinations are reserved. Bits[7:0] Release Number 00h Pre-release 1.0 10h Release 1.0

13.2.16 LEGSUP–Legacy Support Register (Function 2)

PCI Address Offset: C0-C1h
Default: 2000h
Attribute: Read/Write Clear

This register provides control and status capability for the legacy keyboard and mouse Functions.

Bit	Description
15	**End OF A20GATE Pass Through Status (A20PTS)–R/WC.** This bit is set to 1 to indicate that the A20GATE pass-through sequence has ended. This bit will only be set if bit 7 of this register is also set. Software must use the enable bits to determine the exact cause of an SMI#. Software clears this bit by writing a 1 to it.
14	Reserved.
13	**USB PIRQ Enable (USBPIRQDEN)–R/W.** 1 (default) = USB interrupt is routed to PIRQD. 0 = USB interrupt does not route to PIRQD. This bit prevents the USB controller from generating an interrupt. Note that it will probably be configured to generate an SMI using bit 4 of this register. Default to 1 for compatibility with older USB software.
12	**USB IRQ Status (USBIRQS)–RO.** This bit is set to 1 to indicate that the USB IRQ is active. Software must use the enable bits to determine the exact cause of an SMI#. Writing a 1 to this bit will have no effect. Software must clear the IRQ via the USB controller.
11	**Trap By 64h Write Status (TBY64W)–R/WC.** This bit is set to 1 to indicate that a write to port 64h occurred. Software must use the enable bits to determine the exact cause of an SMI#. Software clears this bit by writing a 1 to it.
10	**Trap By 64h Read Status (TBY64R)–R/WC.** This bit is set to 1 to indicate that a read to port 64h occurred. Software must use the enable bits to determine the exact cause of an SMI#. Software clears this bit by writing a 1 to it.
9	**Trap By 60h Write Status (TBY60W)–R/WC.** This bit is set to 1 to indicate that a write to port 60h occurred. Software must use the enable bits to determine the exact cause of an SMI#. Software clears this bit by writing a 1 to it.
8	**Trap By 60h Read Status (TBY60R)–R/WC.** This bit is set to 1 to indicate that a read to port 60h occurred. Software must use the enable bits to determine the exact cause of an SMI#. Software clears this bit by writing a 1 to it.
7	**SMI At End Of Pass Through Enable (SMIEPTE)–R/W.** 1=Enable the generation of an SMI when the A20GATE pass-through sequence has ended. 0 (default) = Disable. This may be required if an SMI is generated by a USB interrupt in the middle of an A20GATE pass through sequence and needs to be serviced later.
6	**Pass Through Status (PSS)–RO.** 1 =A20GATE pass through sequence is currently in progress. 0 (default) = Not currently executing the A20GATE pass-through sequence. This bit indicates that the host controller is executing the A20GATE pass-through sequence. If software needs to reset this bit, it should set Bit 5 to 0 causing the host controller to immediately end the A20GATE pass through sequence.
5	**A20Gate Pass Through Enable (A20PTEN)–R/W.** 1=Enable A20GATE pass through sequence. 0 (default) = Disable. When enabled, the logic will pass through the following A20GATE command sequence: Cycle Address Data Write 64h D1h (1 or more) (Starts the Sequence) Write 60h xxh Read 64h N/A (0 or more) Write 64h FFh (End of A20GATE Pass Through Sequence) Any deviation seen in the above sequence will cause the host controller to immediately exit the sequence and return to typical operation, performing an I/O trap and generating an SMI# if appropriate enable bits are set. When enabled, SMI# will not be generated during the sequence, even if the various enable bits are set. Note that during a Pass-through sequence, the above status bits will not be set for the I/O accesses that are part of the sequence.
4	**Trap/SMI ON IRQ Enable (USBSMIEN)–R/W.** 1 = Enable SMI# generation on USB IRQ. 0 (default) = Disable.
3	**Trap/SMI On 64h Write Enable (64WEN)–R/W.** 1 = Enable I/O Trap and SMI# generation on port 64h write. 0 (default) = Disable.
2	**Trap/SMI On 64h Read Enable (64REN)–R/W.** 1 = Enable I/O Trap and SMI# generation on port 64h read. 0 (default) = Disable.
1	**Trap/SMI On 60h Write Enable (60WEN)–R/W.** 1 = Enable I/O Trap and SMI# generation on port 60h write. 0 (default) = Disable.
0	**Trap/SMI On 60h Read Enable (60REN)–R/W.** 1 = Enable I/O Trap and SMI# generation on port 60h read. 0 (default) = Disable.

13.2.17 USBREN–USB Resume Enable

Address Offset: C4h
Default Value: 00h
Attribute: Read/Write

Bit	Description
7:2	Reserved.
1	**PORT1EN:** Enable port 1 of the USB controller to look at wakeup events. When set, the USB controller will monitor port 1 for a connect/disconnect, which is a resume event for USB. When cleared, the USB controller will not look at this port for a wakeup event. This bit applies to port 1.
0	**PORT0EN:** Enable port 0 of the USB controller to look at wakeup events. When set, the USB controller will monitor port 0 for a connect/disconnect, which is a resume event for USB. When cleared, the USB controller will not look at this port for a wakeup event. This bit applies to port 0.

13.3 USB Host Controller I/O Space Registers

13.3.1 USBCMD–USB Command Register (I/O)

I/O Address: Base + (00-01h)
Default Value: 0000h
Attribute: Read/Write (WORD write-able only)

The Command Register indicates the command to be executed by the serial bus host controller. Writing to the register causes a command to be executed. The table following the bit description provides additional information on the operation of the Run/Stop and Debug bits.

Bit	Description
15:9	**Reserved.**
8	**Loop Back Test Mode:** When set, the IFB is in loop back test mode. When both ports are connected together, a write to one port will be seen on the other port and the data will be store in I/O offset 18h. When clear, this will not occur.
7	**Max Packet (MAXP).** 1=64 bytes. 0=32 bytes. This bit selects the maximum packet size that can be used for full speed bandwidth reclamation at the end of a frame. This value is used by the Host Controller to determine whether it should initiate another transaction based on the time remaining in the SOF counter. Use of reclamation packets larger than the programmed size will cause a Babble error if executed during the critical window at frame end. The Babble error results in the offending endpoint being stalled. Software is responsible for ensuring that any packet which could be executed under bandwidth reclamation be within this size limit.
6	**Configure Flag (CF).** HCD software sets this bit as the last action in its process of configuring the Host Controller. This bit has no effect on the hardware. It is provided only as a semaphore service for software.
5	**Software Debug (SWDBG).** 1=Debug mode. 0=Normal Mode. In SW Debug mode, the Host Controller clears the Run/Stop bit after the completion of each USB transaction. The next transaction is executed when software sets the Run/Stop bit back to 1. The SWDBG bit must only be manipulated when the controller is in the stopped state. This can be determined by checking the HCHalted bit in the USBSTS register.
4	**Force Global Resume (FGR).** 1=Host Controller sends the Global Resume signal on the USB. Software sets this bit to 0 after 20 ms have elapsed to stop sending the Global Resume signal. At that time all USB devices should be ready for bus activity. The Host Controller sets this bit to 1 when a resume event (connect, disconnect, or K-state) is detected while in global suspend mode. Software resets this bit to 0 to end Global Resume signaling. The 1 to 0 transition causes the port to send a low speed EOP signal. This bit will remain a 1 until the EOP has completed.

Bit	Description
3	**Enter Global Suspend Mode (EGSM).** 1=Host Controller enters the Global Suspend mode. No USB transactions occur during this time. The Host Controller is able to receive resume signals from USB and interrupt the system. Software resets this bit to 0 to come out of Global Suspend mode. Software writes this bit to 0 at the same time that Force Global Resume (bit 4) is written to 0 or after writing bit 4 to 0. Software must also ensure that the Run/Stop bit (bit 0) is cleared prior to setting this bit.
2	**Global Reset (GRESET).** When this bit is set, the Host Controller sends the global reset signal on the USB and then resets all its logic, including the internal hub registers. The hub registers are reset to their power on state. This bit is reset by the software after a minimum of 10 ms has elapsed as specified in Chapter 7 of the USB Specification. Note: Chip Hardware Reset has the same effect as Global Reset (bit 2), except that the Host Controller does not send the Global Reset on USB.
1	**Host Controller Reset (HCRESET).** When this bit is set, the Host Controller module resets its internal timers, counters, state machines, etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. This bit is reset by the Host Controller when the reset process is complete. The HCReset effects on Hub registers are slightly different from Chip Hardware Reset and Global USB Reset. The HCReset affects bits [8,3:0] of the Port Status and Control Register (PORTSC) of each port. HCReset resets the state machines of the Host Controller including the Connect/ Disconnect state machine (one for each port). When the Connect/Disconnect state machine is reset, the output that signals connect/disconnect are negated to 0, effectively signaling a disconnect, even if a device is attached to the port. This virtual disconnect causes the port to be disabled. This disconnect and disabling of the port causes bit 1 (connect status change) and bit 3 (port enable/ disable change) of the PORTSC to get set. The disconnect also causes bit 8 of PORTSC to reset. About 64 bit times after HCReset goes to 0, the connect and low-speed detect will take place and bits 0 and 8 of the PORTSC will change accordingly.
0	**Run/Stop (RS).** 1=Run. 0=Stop. When set to a 1, the Host Controller proceeds with execution of the schedule. The Host Controller continues execution as long as this bit is set. When this bit is set to 0, the Host Controller completes the current transaction on the USB and then halts. The HC Halted bit in the status register indicates when the Host Controller has finished the transaction and has entered the stopped state. The Host Controller clears this bit when the following fatal errors occur: consistency check failure, PCI Bus errors.

Table 13-2. Run/Stop, Debug Bit Interaction

SWDBG (Bit 5)	Run/Stop (Bit 0)	Operation
0	0	If executing a command, the Host Controller completes the command and then stops. The 1.0 ms frame counter is reset and command list execution resumes from start of frame using the frame list pointer selected by the current value in the FRNUM register. (While Run/ Stop=0, the FRNUM register can be reprogrammed).
0	1	Execution of the command list resumes from Start Of Frame using the frame list pointer selected by the current value in the FRNUM register. The Host Controller remains running until the Run/Stop bit is cleared (by Software or Hardware).
1	0	If executing a command, the Host Controller completes the command and then stops and the 1.0 ms frame counter is frozen at its current value. All status is preserved. The Host Controller begins execution of the command list from where it left off when the Run/Stop bit is set.
1	1	Execution of the command list resumes from where the previous execution stopped. The Run/Stop bit is set to 0 by the Host Controller when a TD is being fetched. This causes the Host Controller to stop again after the execution of the TD (single step). When the Host Controller has completed execution, the HC Halted bit in the Status Register is set.

13.3.2 USBSTS–USB Status Register (I/O)

I/O Address: Base + (02-03h)
Default Value: 0000h
Attribute: Read/Write Clear

This register indicates pending interrupts and various states of the Host Controller. The status resulting from a transaction on the serial bus is not indicated in this register. Software sets a bit to 0 in this register by writing a 1 to it.

Bit	Description
15:6	Reserved.
5	**HCHalted.** The Host Controller sets this bit to 1 after it has stopped executing as a result of the Run/Stop bit being set to 0, either by software or by the Host Controller hardware (debug mode or an internal error).
4	**Host Controller Process Error.** The Host Controller sets this bit to 1 when it detects a fatal error and indicates that the Host Controller suffered a consistency check failure while processing a Transfer Descriptor. An example of a consistency check failure would be finding an illegal PID field while processing the packet header portion of the TD. When this error occurs, the Host Controller clears the Run/Stop bit in the Command register to prevent further schedule execution. A hardware interrupt is generated to the system.
3	**Host System Error.** The Host Controller sets this bit to 1 when a serious error occurs during a host system access involving the Host Controller module. In a PCI system, conditions that set this bit to 1 include PCI Parity error, PCI Master Abort, and PCI Target Abort. When this error occurs, the Host Controller clears the Run/Stop bit in the Command register to prevent further execution of the scheduled TDs. A hardware interrupt is generated to the system.
2	**Resume Detect.** The Host Controller sets this bit to 1 when it receives a “RESUME” signal from a USB device. This is only valid if the Host Controller is in a global suspend state (bit 3 of Command register = 1).
1	**USB Error Interrupt.** The Host Controller sets this bit to 1 when completion of a USB transaction results in an error condition (e.g. error counter underflow). If the TD on which the error interrupt occurred also had its IOC bit set, both this bit and Bit 0 are set.
0	**USB Interrupt (USBINT).** The Host Controller sets this bit to 1 when the cause of an interrupt is a completion of a USB transaction whose Transfer Descriptor had its IOC bit set. The Host Controller also sets this bit to 1 when a short packet is detected (actual length field in TD is less than maximum length field in TD), and short packet detection is enabled in that TD.

13.3.3 USBINTR–USB Interrupt Enable Register (I/O)

I/O Address: Base + (04-05h)
Default Value: 0000h
Attribute: Read/Write

This register enables and disables reporting of the corresponding interrupt to the software. When a bit is set and the corresponding interrupt is active, an interrupt is generated to the host. Fatal errors (Host Controller Processor Error- bit 4, USBSTS Register) cannot be disabled by the host controller. Interrupt sources that are disabled in this register still appear in the Status Register to allow the software to poll for events.

Bit	Description
15:4	**Reserved.**
3	**Short Packet Interrupt Enable.** 1=Enabled. 0=Disabled.
2	**Interrupt On Complete (IOC) Enable.** 1= Enabled. 0=Disabled.
1	**Resume Interrupt Enable.** 1= Enabled. 0=Disabled.
0	**Time-out/CRC Interrupt Enable.** 1= Enabled. 0=Disabled.

13.3.4 FRNUM–Frame Number Register (I/O)

I/O Address: Base + (06-07h)
Default Value: 0000h
Attribute: Read/Write (Writes must be Word Writes)

Bits [10:0] of this register contain the current frame number which is included in the frame SOF packet. This register reflects the count value of the internal frame number counter. Bits [9:0] are used to select a particular entry in the Frame List during schedule execution. This register is updated at the end of each frame time.

This register must be written as a word. Byte writes are not supported. This register cannot be written unless the Host Controller is in the STOPPED state as indicated by the HCHalted bit (USBSTS register). A write to this register while the Run/Stop bit is set (USBCMD register) is ignored.

Bit	Description
15:11	**Reserved**.
10:0	**Frame List Current Index/Frame Number.** Bits [10:0] provide the frame number in the SOF Frame. The value in this register increments at the end of each time frame (approximately every 1 ms). In addition, bits [9:0] are used for the Frame List current index and correspond to memory address signals [11:2].

13.3.5 FLBASEADD–Frame List Base Address Register (I/O)

I/O Address: Base + (08-0Bh)
Default Value: Undefined
Attribute: Read/Write

This 32-bit register contains the beginning address of the Frame List in the system memory. HCD loads this register prior to starting the schedule execution by the Host Controller. When written, only the upper 20 bits are used. The lower 12 bits are written as zero (4-Kbyte alignment). The contents of this register are combined with the frame number counter to enable the Host Controller to step through the Frame List in sequence. The two least significant bits are always 00. This requires Dword alignment for all list entries. This configuration supports 1024 Frame List entries.

Bit	Description
31:12	**Base Address.** These bits correspond to memory address signals [31:12], respectively.
11:0	**Reserved.** Must be written as 0s.

13.3.6 SOFMOD–Start of Frame (SOF) Modify Register (I/O)

I/O Address: Base + (0Ch)
Default Value: 40h
Attribute: Read/Write

This 1-byte register is used to modify the value used in the generation of SOF timing on the USB. Only the 7 least significant bits are used. When a new value is written into the these 7 bits, the SOF timing of the next frame will be adjusted. This feature can be used to adjust out any offset from the clock source that generates the clock that drives the SOF counter. This register can also be used to maintain real time synchronization with the rest of the system so that all devices have the same sense of real time. Using this register, the frame length can be adjusted across the full range

required by the USB specification. It's initial programmed value is system dependent based on the accuracy of hardware USB clock and is initialized by system BIOS. It may be reprogrammed by USB system software at any time. Its value will take effect from the beginning of the next frame. This register is reset upon a Host Controller Reset or Global Reset. Software must maintain a copy of its value for reprogramming if necessary.

Bit	Description
7	**Reserved.**
6:0	**SOF Timing Value.** Guidelines for the modification of frame time are contained in Chapter 7 of the USB Specification. The SOF cycle time (number of SOF counter clock periods to generate a SOF frame length) is equal to 11936 + value in this field. The default value is decimal 64 which gives a SOF cycle time of 12000. For a 12 MHz SOF counter clock input, this produces a 1 ms Frame period. The following table indicates what SOF Timing Value to program into this field for a certain frame period. Frame Length (# 12 MHz Clocks) SOF Reg. Value (decimal) (decimal) 11936 0 11937 1 . . . . 11999 63 12000 64 12001 65 . . . . 12062 126 12063 127

13.3.7 PORTSC–Port Status and Control Register (I/O)

I/O Address: Base + (10-11h)–Port 0
Base + (12-13h)–Port 1
Default: 0080h
Access: Read/Write (WORD writeable only)

After a Power-up reset, Global reset, or Host Controller reset, the initial conditions of a port are: No device connected, Port disabled, and the bus line status is 00 (single-ended zero). Note: If a device is attached, the port state will transition to the attached state and system software will process this as with any status change notification. It may take up to 64 USB bit times for the port transition to occur. If the Host Controller is in global suspend mode, then, if any of bits [6,3,1] gets set, the Host Controller will signal a global resume. Refer to Chapter 11 of the USB Specification for details on hub operation.

Bit	Description
15:13	**Reserved.** Must written as 0s when writing this register.

Bit	Description
12	**Suspend–R/W.** 1=Port in suspend state. 0=Port not in suspend state. This bit should not be written to a 1 if global suspend is active (bit 3=1 in the USBCMD register). Bit 2 and bit 12 of this register define the hub states as follows: Bits [12,2] Hub Port State x0 Disable 01 Enable 11 Suspend When in suspend state, downstream propagation of data is blocked on this port, except for single-ended 0 resets (global reset and port reset). The blocking occurs at the end of the current transaction, if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB.
11	**Over-current Indicator Change–R/WC.** 1=A change from 1 to 0 has been detected on the Over-current (OC[X]#) pin for this port. 0=No change has been detected. Software sets this bit to 0 by writing a 1 to it.
10	**Over-current Indicator–RO.** 1=Overcurrent pin (OC[X]#) for this port is at logic 0 indicating over-current condition. 0=Overcurrent pin for this port is at logic 1 indicating a normal condition. If asserted, the corresponding port is disabled.
9	**Port Reset–R/W**. 1=Port is in Reset. 0=Port is not in Reset. When in the Reset State, the port is disabled and sends the USB Reset signaling. Note that host software must ensure that the RESET signaling is active for the proper amount of time as specified in the USB Specification.
8	**Low Speed Device Attached–RO.** 1=Low speed device is attached to this port. 0=Full speed device. Writes have no effect.
7	**Reserved–RO.** Always read as 1.
6	**Resume Detect–R/W.** 1= Resume detected/driven on port. 0=No resume (K-state) detected/ driven on port. Software sets this bit to a 1 to drive resume signaling. The Host Controller sets this bit to a 1 if a J-to-K transition is detected while the port is in the Suspend state. Note that when this bit is 1, a K-state is driven on the port as long as this bit remains 1 and the port is still in suspend state. Writing a 0 (from 1) causes the port to send a low speed EOP. This bit will remain a 1 until the EOP has completed.
5:4	**Line Status–RO.** These bits reflect the D+ (bit 4) and D- (bit 5) signals lines' logical levels. These bits are used for fault detect and recovery as well as for USB diagnostics. This field is updated at EOF2 time (See Chapter 11 of the USB Specification).
3	**Port Enable/Disable Change–R/WC.** 1=Port enabled/disabled status has changed. 0=No change. For the root hub, this bit gets set only when a port is disabled due to disconnect on the that port or due to the appropriate conditions existing at the EOF2 point (See Chapter 11 of the USB Specification). Software clears this bit by writing a 1 to it.
2	**Port Enabled/Disabled–R/W.** 1=Enable. 0=Disable. Ports can be enabled by host software only. Ports can be disabled by either a fault condition (disconnect event, overcurrent condition, or other fault condition) or by host software. Note that the bit status does not change until the port state actually changes and that there may be a delay in disabling or enabling a port if there is a transaction currently in progress on the USB.
1	**Connect Status Change–R/WC.** 1=Change in Current Connect Status. 0=No change. Indicates a change has occurred in the port's Current Connect Status (see bit 0). The hub device sets this bit for any changes to the port device connect status, even if system software has not cleared a connect status change. If, for example, the insertion status changes twice before system software has cleared the changed condition, hub hardware will be "setting" an already-set bit (i.e. the bit will remain set). However, the hub transfers the change bit only once when the Host Controller requests a data transfer to the Status Change endpoint. System software is responsible for determining state change history in such a case. Software sets this bit to 0 by writing a 1 to it.
0	**Current Connect Status–RO.** 1=Device is present on port. 0=No device is present. This value reflects the current state of the port, and may not correspond directly to the event that caused the Connect Status Change bit (Bit 1) to be set.

SM Bus Controller Configuration 14

The IFB PCI Function 3 contains the SMBus Controller configuration space.

14.1 SM Bus Configuration Registers (Function 3)

Configuration Offset	Mnemonic	Register	Register Access
00–01h	VID	Vendor Identification	RO
02–03h	DID	Device Identification	RO
04–05h	PCICMD	PCI Command	R/W
06–07h	PCISTS	PCI Device Status	R/WC
08h	RID	Revision Identification	RO
09-0Bh	CLASSC	Class Code	RO
0C-1Fh	–	Reserved	–
20-23h	BAR	Base Address Register	R/W
24–3Bh	–	Reserved	–
2C–2Dh	SVID	Subsystem Vendor ID	RO
2E–2Fh	SID	Subsystem ID	RO
30–3Fh	–	Reserved	–
3Ch	IL	Interrupt Line	RW
3Dh	IP	Interrupt Pin	RO
3E-3Fh	–	Reserved	–
40h	HC	Host Configuration	RW
41h	SCOM	Slave Command Port	RW
42h	SS1	Slave Shadow Address 1	RW
43h	SS2	Slave Shadow Address 2	RW
41-FFh	–	Reserved	–

14.2 System Management Register Descriptions

This section describes in detail the registers associated with the IFB System Management Function.

14.2.1 VID–Vendor Identification Register (Function 3)

Address Offset: 00–01h
Default Value: 8086h
Attribute: Read only

The VID Register contains the vendor identification number. This register, along with the Device Identification Register, uniquely identifies any PCI device. Writes to this register have no effect.

Bit	Description
15:0	**Vendor Identification Number.** This is a 16-bit value assigned to Intel.

14.2.2 DID–Device Identification Register (Function 3)

Address Offset: 02-03h
Default Value: 7603h
Attribute: Read only

The DID Register contains the device identification number. This register, along with the VID Register, defines the IFB Power Management Controller. Writes to this register have no effect.

Bit	Description
15:0	**Device Identification Number.** This is a 16-bit value assigned to the IFB System Management Controller.

14.2.3 PCICMD–PCI Command Register (Function 3)

Address Offset: 04-05h
Default Value: 00h
Attribute: Read/Write

This register controls access to the I/O space registers.

Bit	Description
15:10	**Reserved.** Read 0.
9	**Fast Back to Back Enable (Not Implemented).** This bit is hardwired to 0.
8:5	**Reserved.** Read as 0.
4	**Memory Write and Invalidate Enable (Not Implemented).** This bit is hardwired to 0.
3	**Special Cycle Enable (SCE).** 1 = Enable, the IFB recognizes the Stop Grant special cycle. 0=Disable. The SCE bit in Function 1 PCI Command register controls IFB response to the Shutdown special cycle.
2	**Bus Master Enable (Not Implemented).** This bit is hardwired to 0.

Bit	Description
1	**Memory Space Enable (Not Implemented).** 1=Enable. 0=Disable. This bit controls the access to memory space. If this bit is set, access to the memory space by power management logic is enabled.
0	**I/O Space Enable (IOSE).** 1=Enable. 0=Disable. This bit controls the access to the I/O space registers. If this bit is set, access to the power management I/O registers is enabled. The base register for the I/O registers must be programmed before this bit is set.

14.2.4 PCISTS–PCI Device Status Register (Function 3)

Address Offset: 06-07h
Default Value: 0280h
Attribute: Read/Write

DSR is a 16-bit status register that reports the occurrence of a PCI target-abort when the System Management Function is a target device. The register also indicates the System Management DEVSEL# signal timing that is hardwired in the module. The DSR fields are shown in the table below.

Bit	Description
15	**Detected Parity (Not Implemented).** This bit is hardwired to 0.
14	**SERR# Status (Not Implemented).** This bit is hardwired to 0.
13	**Master-Abort Status (Not Implemented).** This bit is hardwired to 0.
12	**Received Target-Abort Status (Not Implemented).** This bit is hardwired to 0.
11	**Signaled Target-Abort Status (STA)–R/WC.** This bit is set when the Power Management Function is targeted with a transaction that the it terminates with a target abort. Software resets STA to 0 by writing a 1 to this bit.
10:9	**DEVSEL# Timing Status (DEVT)–RO.** This 2-bit field defines the timing for DEVSEL# assertion. These read only bits indicate the IFB's DEVSEL# timing when performing a positive decode. Since the IFB always generate the DEVSEL# with medium timing, DEVT=01. This DEVSEL# timing does not include Configuration cycles.
8	**Data Parity Detected (Not Implemented).** This bit is hardwired to 0.
7	**Fast Back to Back Capable (FBC)–RO.** Hardwired to 1. This bit indicates to the PCI Master that Power Management as a target is capable of accepting fast back-to-back transactions.
6:0	**Reserved.** Read as 0's.

14.2.5 RID–Revision Identification Register (Function 3)

Address Offset: 08h
Default Value: Initial Stepping = 00h. Refer to IFB Specification Updates for other values programmed here.
Attribute: Read only

This 8 bit register contains device stepping information. Writes to this register have no effect.

Bit	Description
7:0	**Revision ID Byte.** The register is hardwired to the default value.

14.2.6 CLASSC–Class Code Register (Function 3)

Address Offset: 09-0Bh
Default Value: 0C0500h
Attribute: Read only

This register identifies the Base Class Code, Sub-Class Code, and Device Programming interface for IFB PCI Function 3.

Bit	Description
23:16	**Base Class Code (BASEC).** 0Ch = Serial Bus Controller.
15:8	**Sub-Class Code (SCC)**. 05h = System Management Bus (SMBus) Controller.
7:0	**Programming Interface (PI).** 00h = No specific register level programming defined.

14.2.7 SMBBA–SMBus Base Address (Function 3)

Address Offset: 20-23h
Default Value: 00000001h
Attribute: Read/Write

This register contains the base address of the SMBus I/O Registers.

Bit	Description
31:16	**Reserved.** Hardwired to 0s. Must be written as 0s.
15:4	**Index Register Base Address.** Bits [15:4] correspond to I/O address signals AD [15:4], respectively.
3:1	**Reserved.** Read as 0.
0	**Resource Type Indicator (RTE)–RO.** This bit is hardwired to 1 indicating that the base address field in this register maps to I/O space.

14.2.8 SVID–Subsystem Vendor ID (Function 3)

Address: 2C-2Dh
Default Value: 0000h
Attribute: Read only

Bit	Description
15:0	**Subsystem Vendor ID.**

14.2.9 SID–Subsystem ID (Function 3)

Address: 2E-2Fh
Default Value: 0000h
Attribute: Read only

Bit	Description
15:0	Subsystem ID.

14.2.10 INTLN–Interrupt Line Register (Function 3)

Address Offset: 3Ch
Default Value: 00h
Attribute: Read/Write

Software programs this register with interrupt information concerning the Power Management module.

Bit	Description
7:0	**Interrupt Line.** The value in this register has no affect on IFB hardware operations.

14.2.11 INTPN–Interrupt Pin (Function 3)

Address Offset: 3Dh
Default Value: 02h
Attribute: Read only

This register indicates that PCI interrupt PIRQB# is used for the Power Management module.

Bit	Description
7:3	**Reserved.**
2:0	**Serial Bus Module Interrupt Routing.** This field is hardwired to 02h to indicate that PCI interrupt pin PIRQB# is used.

14.2.12 Host Configuration

Address Offset: 40h
Default Value: 00h
Attribute: Read/Write

Bit	Description
7:2	Reserved.
1	**SMI_EN:** When this bit is set, any source of an SMB interrupt will instead be routed to generate an SMI#. This bit will only take effect if the INTREN bit is set in I/O space.
0	**HST_EN**: When set, the SMB Host Controller interface is enabled to execute commands. The HST_INT_EN bit needs to be enabled in order for the SMB Host Controller to interrupt or SMI#. Additionally, the SMB Host Controller will not respond to any new requests until all interrupt requests have been. The HST_EN bit does not affect the SMB Slave Port.

14.2.13 smbslvc–SMBus Slave Command (Function 3)

Address Offset: 41h
Default Value: 00h
Attribute: Read/Write

Bit	Description
7:0	**SMBus Host Slave Command (SMBCMD)–R/W.** Specifies the command values to be matched for SMBus master accesses to the SMBus controller host slave interface (SMBus port 10h).

14.2.14 smbshdw1–SMBus Slave Shadow Port 1 (Function 3)

Address Offset: 42h
Default Value: 00h
Attribute: Read/Write

Bit	Description
7:0	**SHDW1_ADD:** Slave shadow address 1. When an SMB master generates an access to the port defined by this register and the SHDW1_EN bit is set in I/O space, then the SHDW1_STS bit is set and an interrupt or resume event is generated.

14.2.15 smbshdw2–SMBus Slave Shadow Port 2 (Function 3)

Address Offset: 43h
Default Value: 00h
Attribute: Read/Write

Bit	Description
7:0	**SHDW2_ADD:** Slave shadow address 2. When an SMB master generates an access to the port defined by this register and the SHDW2_EN bit is set in I/O space, then the SHDW2_STS bit is set and an interrupt or resume event is generated.

14.3 SMBus I/O Space Registers

The “Base” address is programmed in the IFB PCI Configuration Space for Function 3, Offset 20h-23h.

14.3.1 smbhststs–SMBus Host Status Register (I/O)

I/O Address: Base + (00h)
Default Value: 00h
Attribute: Read/Write

This register provides status information concerning the SMBus controller host interface.

Bit	Description
7:5	Reserved.
4	**Failed (FAILED)–R/WC.** 1 = Indicates that the source of SMBus interrupt was a failed bus transaction, set when KILL bit is set (SMBHSTCNT register). 0 = SMBus interrupt not caused by KILL bit. This bit is only set by hardware and can only be reset by writing a 1 to this bit position.
3	**BUS COLLISION(BUS_ERR)–R/WC.** 1 = Indicates that the source of SMBus interrupt was a transaction collision. 0 = SMBus interrupt not caused by transaction collision. This bit is only set by hardware and can only be reset by writing a 1 to this bit position.
2	**Device Error (DEV_ERR)–R/WC.** 1 = Indicates that the source of SMBus interrupt was the generation of an SMBus transaction error. 0 = SMBus interrupt not caused by transaction error. This bit is only set by hardware and can only be reset by writing a 1 to this bit position. Transaction errors are caused by: Illegal Command Field Unclaimed Cycle (host initiated) Host Device Time-out
1	**SMBus Interrupt (INTER)–R/WC.** 1 = Indicates that the source of SMBus interrupt was the completion of the last host command. 0 = SMBus interrupt not caused by host command completion. This bit is only set by hardware and can only be reset by writing a 1 to this bit position.
0	**Host Busy (HOST_BUSY)–RO.** 1 = Indicates that the SMBus controller host interface is in the process of completing a command. 0 = SMBus controller host interface is not processing a command. None of the other registers should be accessed if this bit is set.

14.3.2 smbslvsts–SMBus Slave Status Register (I/O)

I/O Address: Base + (01h)
Default Value: 00h
Attribute: Read/Write

This register provides status information concerning the SMBus controller slave interface.

Bit	Description
7:5	Reserved.
4	**Shadow2 Status (SHDW2_STS)–R/WC.** 1 = Indicates that the source of SMBus interrupt or resume event was a slave cycle address match of the SMBSHDW2 port. 0 = SMBus interrupt not caused by address match to SMBSHDW2 port. This bit is only set by hardware and can only be reset by writing a 1 to this bit position.
3	**Shadow1 Status (SHDW1_STS)–R/WC.** 1 = Indicates that the source of SMBus interrupt or resume event was a slave cycle address match of the SMBSHDW1 port. 0 = SMBus interrupt not caused by address match to SMBSHDW1 port. This bit is only set by hardware and can only be reset by writing a 1 to this bit position.

Bit	Description
2	**Slave Status (SLV_STS)–R/WC.** 1 = Indicates that the source of SMBus interrupt or resume event was a slave cycle event match of the SMBSLVC (command match) and SMBSLVEVT (data event match). 0 = SMBus interrupt not caused by slave event match. This bit is only set by hardware and can only be reset by writing a 1 to this bit position.
1	Reserved.
0	**Slave Busy (SLV_BSY)–RO.** 1 = Indicates that the SMBus controller slave interface is in the process of receiving data. 0 = SMBus controller slave interface is not processing data. None of the other registers should be accessed if this bit is set.

14.3.3 smbhstcnt–SMBus Host Control Register (I/O)

I/O Address: Base + (02h)
Default Value: 00h
Attribute: Read/Write

The control register is used to enable SMBus controller host interface Functions. Reads to this register clears the host interface's index pointer to the block data storage array.

Bit	Description
7	Reserved.
6	**Start (START)–R/W.** 1 = Writing a 1 to this bit initiates the SMBus controller host interface to execute the command programmed in the SMB_CMD_PROT field. All necessary registers should be setup prior to writing a 1 to this bit position. 0 = Writing a zero has no effect. This bit always reads zero. The HOST_BUSY bit can be used to identify when the SMBus host controller has finished executing the command.
5	**SMB_IDX_CLR**: Any read to this register clears the slave interface's internal index pointer to the block SRAM array. This bit always reads zero, but exists as a reminder that reads to this register clear the pointer index to the block SRAM array.
4:2	**SMBus Command Protocol (SMB_CMD_PROT)–R/W.** Selects the type of command the SMBus controller host interface will execute. Reads or writes are determined by bit 0 of SMBHSTADD register. This field is decoded as follows: [4:2] Protocol 0,0,0 Quick Read or Write 0,0,1 Byte Read or Write 0,1,0 Byte Data Read or Write 0,1,1 Word Data Read or Write 1,0,0 Block Read or Write 1,0,1 Reserved 1,1,0 Reserved 1,1,1 Reserved
1	**Kill (KILL)–R/W.** 1 = Stop the current in process SMBus controller host transaction. This sets the FAILED status bit and asserts the interrupt selected by the SMB_INTRSEL field. 0 = Allows the SMBus controller host interface to Function normally.
0	**Interrupt Enable (INTEREN)–R/W.** 1 = Enable the generation of interrupts upon the completion of the current host transaction. 0 = Disable.

14.3.4 smbhstcmd–SMBus Host Command Register (I/O)

I/O Address: Base + (03h)
Default Value: 00h
Attribute: Read/Write

This register is transmitted by the SMBus controller host interface in the command field of the SMBus protocol.

Bit	Description
7:0	**SMBus Host Command (HST_CMD)–R/W**. This field contains the data transmitted in the command field of SMBus host transaction.

14.3.5 smbhstadd–SMBus Host Address Register (I/O)

I/O Address: Base + (04h)
Default Value: 00h
Attribute: Read/Write

This register is transmitted by the SMBus controller host interface in the slave address field of the SMBus protocol.

Bit	Description
7:1	**SMBus Address (SMB_ADDRESS)–R/W.** This field contains the 7-bit address of the targeted slave device.
0	**SMBus Read or Write (SMB_RW)–R/W.** 1 = Execute a READ command. 0 = Execute a WRITE command.

14.3.6 smbhstdat0–SMBus Host Data 0 Register (I/O)

I/O Address: Base + (05h)
Default Value: 00h
Attribute: Read/Write

This register is transmitted by the SMBus controller host interface in the Data0 field of the SMBus protocol.

Bit	Description
7:0	**SMBus Data 0 (SMBD0)–R/W.** This register should be programmed with the value to be transmitted in the Data0 field of an SMBus host interface transaction. For a block write command, the count of the memory block should be stored in this field. The value of this register is loaded into the block transfer count field. This register must be programmed to a value between 1 and 32 for block command counts. A count of 0 or a count above 32 will result in unpredictable behavior. For block reads, the count received from the SMBus device is stored here.

14.3.7 smbhstdat1–SMBus Host Data 1 Register (I/O)

I/O Address: Base + (06h)
Default Value: 00h
Attribute: Read/Write

This register is transmitted by the SMBus controller host interface in the Data1 field of the SMBus protocol.

Bit	Description
7:0	**SMBus Data 1 (SMBD1)–R/W.** This register should be programmed with the value to be transmitted in the Data1 field of an SMBus host interface transaction.

14.3.8 smbblkdat–SMBus Block Data Register (I/O)

I/O Address: Base + (07h)
Default Value: 00h
Attribute: Read/Write

Reads and writes to this register are used to access the 32 byte block data storage array. An internal index pointer is used to address the array. It is reset to 0 by reading the SMBHSTCNT register. The index pointer then increments automatically upon each access to this register. The transfer of block data into (read) or out of (write) this storage array during an SMBus transaction always starts at index address 0.

Bit	Description
7:0	**SMBus Block Data (BLK_DAT)–R/W.** This register is used to transfer data into or out of the block data storage array.

14.3.9 smbslvcnt–SMBus Slave Control Register (I/O)

I/O Address: Base + (08h)
Default Value: 00h
Attribute: Read/Write

The control register is used to enable SMBus controller slave interface Functions.

Bit	Description
7	**SLV_INT_EN:** When set to a ‘1’, the generation of a slave interrupt or SMI# based on a master SMB device generating an access to the host controller’s slave port is enabled. The slave port will set the SLV_STS bit in the Slave Status register (offset 01h). The data will placed in the Slave Data register (offset 0Ah).
6:3	Reserved.
2	**SMBus Shadow Port 2 Enable(SHDW2_EN)–R/W. SLV_INT_EN and SHDW2_EN =**1 will enable the generation of an interrupt or resume event upon an external SMBus master generating a transaction with an address that matches the SMBSHDW2 register. 0 = Disable.
1	**SMBus Shadow Port 1 Enable(SHDW1_EN)–R/W. SLV_INT_EN and SHDW1_EN =**1 will enable the generation of an interrupt or resume event upon an external SMBus master generating a transaction with an address that matches the SMBSHDW1 register. 0 = Disable.

Bit	Description
0	**Slave Enable (SLV_EN)–R/W.** 1 = Enable the generation of an interrupt or resume event upon an external SMBus master generating a transaction with an address that matches the host controller slave port of 10h, a command field which matches the SMBSLVC register, and a match of one of the corresponding enabled events in the SMBSLVEVT register. 0 = Disable.

14.3.9.1 10.3.10.smbshdwcmd–SMBus Shadow Command Register (I/O)

I/O Address: Base + (09h)
Default Value: 00h
Attribute: Read only

This register is used to store command values for external SMBus master accesses to the host slave and slave shadow ports.

Bit	Description
7:0	**Shadow Command (SHDW_CMD)–RO.** This field contains the command value which was received during an external SMBus master access whose address field matched the host slave address (10h) or one of the slave shadow port addresses.

14.3.9.2 10.3.11.smbslvevt–SMBus Slave Event Register (I/O)

I/O Address: Base + (0Ah)
Default Value: 0000h
Attribute: Read/Write

This register is used to enable generation of interrupt or resume events for accesses to the host controller's slave port.

Bit	Description
15:0	**SM BUS Slave Event (SMB_SLV_EVT)–R/W.** This field contains data bits used to compare against incoming data to the SMBSLVDAT register. When a bit in this register is a 1 and the corresponding bit in the SMBSLVDAT register is set, then an interrupt or resume event will be generated if the command value matches the value in the SMBSLVC register and the access was to SMBus host address 10h.

14.3.10 smbslvdat–SMBus Slave Data Register (I/O)

I/O Address: Base + (0Ch)
Default Value: 0000h
Attribute: Read only

This register is used to store data values for external SMBus master accesses to the shadow ports or the SMBus host controller's slave port.

Bit	Description
15:0	**SLAVE DATA (SMB_SLV_DATA)–RO.** This field contains the data value which was transmitted during an external SMBus master access whose address field matched one of the slave shadow port addresses or the SMBus host controller slave port address of 10h.

PCI/LPC Bridge Description 15

15.1 PCI Interface

The IFB incorporates a fully PCI Bus compatible master and slave interface. As a PCI master, the IFB runs cycles on behalf of DMA, Bus Master IDE, and USB. The IFB implements an internal arbiter to request the PCI bus IDE and USB for these master Functions.

All memory cycles run by the IFB master interface target system DRAM.

As a PCI slave, the IFB responds to memory cycles destined for the firmware and I/O cycles to the integrated legacy Functions (8237, 8254, 8259), the integrated IDE controller, and the relocate-able I/O spaces for ACPI, IDE, USB, and SM Bus.

15.1.1 Transaction Termination

The IFB supports the PCI cycle termination as described in the PCI Local Bus specification. **IFB As Master–Master-initiated Termination:** The IFB supports three forms of master-initiated termination: 1.) Normal termination of a completed transaction, 2.) Normal termination of an incomplete transaction due to time-out (applies to line buffer operations-IDE Bus Master, 3.) Abnormal termination due to the slave not responding to the transaction (Abort) The **IFB As a Master–Response to Target-initiated Termination:** As a master, the IFB responds correctly to the target-termination– Target-Abort, Retry, or Disconnect.

IFB as a Target–Target-initiated Termination: The IFB supports three forms of Target-initiated Termination– Disconnect, Retry, and Target Abort.

15.1.2 Parity Support

As a master, the IFB generates address parity for read/write cycles and data parity when the IFB is providing the data. As a slave, the IFB generates data parity for read cycles. The IFB does not check parity and does not generate SERR# due to an address parity error. However, the IFB does generate an NMI when another PCI device asserts SERR# (if enabled).

PAR is the calculated parity signal. PAR is even parity and is calculated on 36 bits–AD[31:0] signals plus C/BE[3:0]#. PAR is always calculated on 36 bits, regardless of the valid byte enables. PAR is valid one PCI clock after the corresponding address or data phase.

15.1.3 PCI Arbitration

The IFB arbitrates for the PCI Bus through the PHOLD# and PHLDA# signals.

15.2 Interrupt Controller

The IFB provides the functionality of two 82C59 interrupt controllers. The two controllers are cascaded so that 13 external and three internal interrupts are possible. The master interrupt controller provides IRQ [7:0] and the slave interrupt controller provides IRQ [15:8]. The three

internal interrupts are used for internal Functions only. IRQ2 is used to cascade the two controllers together and is not available to the user. IRQ0 is used as a system timer interrupt and is tied to Interval Timer 1, Counter 0. IRQ13 is connected internally to FERR#. The remaining 13 interrupt lines (IRQ1, IRQ3-IRQ12, IRQ14, and IRQ15) are available for external system interrupts. Edge or level sense selection is programmable on an individual channel by channel basis, except for IRQ0, IRQ2, IRQ8#, and IRQ13.

The Interrupt unit also supports interrupt steering. The IFB can be programmed to allow the four PCI active low interrupts (PIRQA#-PIRQD#) to be internally routed to one of 11 interrupts: 3 - 7, 9-12, 14 or 15.

The Interrupt Controller consists of two separate 82C59 cores. Interrupt Controller 1 (CNTRL-1) and Interrupt Controller 2 (CNTRL-2) are initialized separately and can be programmed to operate in different modes. The default settings are: IA-32 Compatibility Mode, Edge Sensitive (IRQ0-15) Detection, Normal EOI, Non-Buffered Mode, Special Fully Nested Mode disabled, and Cascade Mode. CNTRL-1 is connected as the Master Interrupt Controller and CNTRL-2 is connected as the Slave Interrupt Controller.

Note that IRQ13 is generated internally (as part of the coprocessor error support) by the IFB.

15.2.1 Programming the Interrupt Controller

The Interrupt Controller accepts two types of command words generated by the CPU or bus master:

15.2.1.1 Initialization Command Words (ICWs)

Before normal operation can begin, each Interrupt Controller in the system must be initialized. In the 82C59, this is a two to four byte sequence. However, for the IFB, each controller must be initialized with a four byte sequence. This four byte sequence is required to configure the interrupt controller correctly for the IFB implementation.

The four initialization command words are referred to by their acronyms: ICW1, ICW2, ICW3, and ICW4.

The base address for each interrupt controller is a fixed location in the I/O memory space, at 0020h for CNTRL-1 and at 00A0h for CNTRL-2.

An I/O write to the CNTRL-1 or CNTRL-2 base address with data bit 4 equal to 1 is interpreted as ICW1. For IFB-based systems, three I/O writes to “base address + 1" (021h for CNTRL-1 and 0A1h for CNTRL-2) must follow the ICW1. The first write to “base address + 1” (021h/0A1h) performs ICW2, the second write performs ICW3, and the third write performs ICW4.

ICW1 starts the initialization sequence.

ICW2 is programmed to provide bits [7:3] of the interrupt vector that will be released onto the data bus by the interrupt controller during an interrupt acknowledge. A different base [7:3] is selected for each interrupt controller.

ICW3 is programmed differently for CNTRL-1 and CNTRL-2, and has a different meaning for each controller.

For CNTRL-1, the master controller, ICW3 is used to indicate which IRQx input line is used to cascade CNTRL-2, the slave controller. Within the IFB interrupt unit, IRQ2 on CNTRL-1 is used to cascade the INTR output of CNTRL-2. Consequently, bit-2 of ICW3 on CNTRL-1 is set to a 1, and the other bits are set to 0's.

For CNTRL-2, ICW3 is the slave identification code used during an interrupt acknowledge cycle. CNTRL-1 broadcasts a code to CNTRL-2 over three internal cascade lines if an IRQ[x] line from CNTRL-2 won the priority arbitration on the master controller and was granted an interrupt acknowledge by the CPU. CNTRL-2 compares this identification code to the value stored in ICW3, and if the code is equal to bits [2:0] of ICW3, CNTRL-2 assumes responsibility for broadcasting the interrupt vector during the second interrupt acknowledge cycle pulse.

ICW4 must be programmed on both controllers. At the very least, bit 0 must be set to a 1 to indicate that the controllers are operating in an Intel Architecture-based system.

15.2.1.2 Operation Command Words (OCWs)

These are the command words which dynamically reprogram the Interrupt Controller to operate in various interrupt modes. Any interrupt lines can be masked by writing an OCW1. A 1 written in any bit of this command word will mask incoming interrupt requests on the corresponding IRQx line.

OCW2 is used to control the rotation of interrupt priorities when operating in the rotating priority mode and to control the End of Interrupt (EOI) Function of the controller.

OCW3 is used to set up reads of the ISR and IRR, to enable or disable the Special Mask Mode (SMM), and to set up the interrupt controller in polled interrupt mode. The OCWs can be written into the Interrupt Controller any time after initialization.

15.2.2 End of Interrupt Operation

15.2.2.1 End of Interrupt (EOI)

The In Service (IS) bit can be set to 0 automatically following the trailing edge of the second INTA# pulse (when AEOI bit in ICW1 is set to 1) or by a command word that must be issued to the Interrupt Controller before returning from a service routine (EOI command). An EOI command must be issued twice with this cascaded interrupt controller configuration, once for the master and once for the slave.

There are two forms of EOI commands: Specific and Non-specific. When the Interrupt Controller is operated in modes which preserve the fully nested structure, it can determine which IS bit to set to 0 on EOI. When a non-Specific EOI command is issued, the Interrupt Controller will automatically set to 0 the highest IS bit of those that are set to 1, since in the fully nested mode the highest IS level was necessarily the last level acknowledged and serviced. A non-specific EOI can be issued with OCW2 (EOI=1, SL=0, R=0).

When a mode is used which may disturb the fully nested structure, the Interrupt Controller may no longer be able to determine the last level acknowledged. In this case a Specific End of Interrupt must be issued which includes as part of the command the IS level to be reset. A specific EOI can be issued with OCW2 (EOI=1, SL=1, R=0, and LO-L2 is the binary level of the IS bit to be set to 0).

It should be noted that an IS bit that is masked by an IMR bit will not be cleared by a non-specific EOI if the Interrupt Controller is in the Special Mask Mode.

15.2.2.2 Automatic End of Interrupt (AEOI) Mode

If AEOI=1 in ICW4, then the Interrupt Controller will operate in AEOI mode continuously until reprogrammed by ICW4. Note that reprogramming ICW4 implies that ICW1, ICW2, and ICW3 must be reprogrammed first, in sequence. In this mode, the Interrupt Controller will automatically

perform a non-specific EOI operation at the trailing edge of the last interrupt acknowledge pulse. Note that from a system standpoint, this mode should be used only when a nested multi-level interrupt structure is not required within a single Interrupt Controller. The AEOI mode can only be used in a master Interrupt Controller and not a slave (on CNTRL-1 but not CNTRL-2).

15.2.3 Modes of Operation

15.2.3.1 Fully Nested Mode

This mode is entered after initialization unless another mode is programmed. The interrupt requests are ordered in priority from 0 through 7 (0 being the highest). When an interrupt is acknowledged, the highest priority request is determined and its vector placed on the bus. Additionally, a bit of the Interrupt Service Register (IS[0:7]) is set. This IS bit remains set until the processor issues an End of Interrupt (EOI) command immediately before returning from the service routine. If the AEOI (Automatic End of Interrupt) is set, the IS bit remains set until the trailing edge of the second INTA#. With the IS bit set, all further interrupts of same or lower priority are inhibited, while higher levels will generate an interrupt (which will be acknowledged if the processor internal interrupt enable flip-flop has been re-enabled through software).

After the initialization sequence, IRQ0 has the highest priority and IRQ7 the lowest. Priorities can be changed, as will be explained, in the rotating priority mode.

15.2.3.2 The Special Fully Nested Mode

This mode will be used in the case of a system where cascading is used, and the priority has to be conserved within each slave. In this case, the special fully nested mode will be programmed to the master (using ICW4). This mode is similar to the normal nested mode with the following exceptions:

When an interrupt request from a certain slave is in service, this slave is not locked out from the master's priority logic and further interrupt requests from higher priority IRQs within the slave will be recognized by the master and will initiate interrupts to the processor. (In the normal nested mode, a slave is masked out when its request is in service and no higher requests from the same slave can be serviced.)

When exiting the Interrupt Service routine, the software has to check whether the interrupt serviced was the only one from that slave. This is done by sending a non-specific End of Interrupt (EOI) command to the slave and then reading its In-service Register and checking for zero. If it is empty, a non-specific EOI can be sent to the master too. If not, no EOI should be sent.

15.2.3.3 Automatic Rotation (Equal Priority Devices)

In some applications, there are a number of interrupting devices of equal priority. Automatic rotation mode provides for a sequential 8-way rotation. In this mode, a device receives the lowest priority after being serviced. In the worst case, a device requesting an interrupt will have to wait until each of seven other devices are serviced at most once.

There are two ways to accomplish automatic rotation using OCW2; the Rotation on Non-Specific EOI Command (R=1, SL=0, EOI=1) and the Rotate in Automatic EOI Mode which is set by (R=1, SL=0, EOI=0) and cleared by (R=0, SL=0, EOI=0).

15.2.3.4 Specific Rotation (Specific Priority)

The programmer can change priorities by programming the bottom priority and thus fixing all other priorities. For example, if IRQ5 is programmed as the bottom priority device, then IRQ6 will be the highest priority device.

The Set Priority Command is issued in OCW2 where: R=1, SL=1; LO-L2 is the binary priority level code of the bottom priority device. See the register description for the bit definitions.

Note that, in this mode, internal status is updated by software control during OCW2. However, it is independent of the End of Interrupt (EOI) command (also executed by OCW2). Priority changes can be executed during an EOI command by using the Rotate on Specific EOI Command in OCW2 (R=1, SL=1, EOI=1 and LO-L2=IRQ level to receive bottom priority).

15.2.3.5 Poll Command

The Polled Mode can be used to conserve space in the interrupt vector table. Multiple interrupts that can be serviced by one interrupt service routine do not need separate vectors if the service routine uses the poll command.

The Polled Mode can also be used to expand the number of interrupts. The polling interrupt service routine can call the appropriate service routine, instead of providing the interrupt vectors in the vector table.

In this mode, the INTR output is not used and the microprocessor internal Interrupt Enable flip-flop is reset, disabling its interrupt input. Service to devices is achieved by software using a Poll Command.

The Poll command is issued by setting P=1 in OCW3. The Interrupt Controller treats the next I/O read pulse to the Interrupt Controller as an interrupt acknowledge, sets the appropriate IS bit if there is a request, and reads the priority level. Interrupts are frozen from the I/O write to the I/O read.

This mode is useful if there is a routine command common to several levels so that the INTA# sequence is not needed (saves ROM space).

15.2.4 Cascade Mode

The Interrupt Controllers in the IFB are interconnected in a cascade configuration with one master and one slave. This configuration can handle up to 15 separate priority levels.

The master controls the slaves through a three line internal cascade bus. When the master drives 010b on the cascade bus, this bus acts like a chip select to the slave controller.

In a cascade configuration, the slave interrupt outputs are connected to the master interrupt request inputs. When a slave request line is activated and afterwards acknowledged, the master will enable the corresponding slave to release the interrupt vector address during the second INTA# cycle of the interrupt acknowledge sequence.

Each Interrupt Controller in the cascaded system must follow a separate initialization sequence and can be programmed to work in a different mode. An EOI Command must be issued twice: once for the master and once for the slave.

15.2.5 Edge and Level Triggered Mode

This mode is programmed using bit 3 in ICW1. With IFB this bit is disabled and a new register for edge and level triggered mode selection, per interrupt input, is included. This is the Edge/Level control Registers ELCR1 and ELCR2. The default programming is equivalent to programming the LTIM bit (ICW1 bit 3) to a 0 (all interrupts selected for edge triggered mode). Note, that IRQ0, 1, 2, 8#, and 13 can not be programmed for level sensitive mode and can not be modified by software.

If an ELCR bit = “0”, an interrupt request will be recognized by a low to high transition on the corresponding IRQx input. The IRQ input can remain high without generating another interrupt.

If an ELCR bit = “1”, an interrupt request will be recognized by a high level on the corresponding IRQ input and there is no need for an edge detection. The interrupt request must be removed before the EOI command is issued to prevent a second interrupt from occurring.

In both the edge and level triggered modes, the IRQ inputs must remain active until after the falling edge of the first INTA#. If the IRQ input goes inactive before this time, a default IRQ7 will occur when the CPU acknowledges the interrupt. This can be a useful safeguard for detecting interrupts caused by spurious noise glitches on the IRQ inputs. To implement this feature, the IRQ7 routine is used for “clean up” simply executing a return instruction, thus ignoring the interrupt. If IRQ7 is needed for other purposes, a default IRQ7 can still be detected by reading the ISR. A normal IRQ7 interrupt will set the corresponding ISR bit; a default IRQ7 will not set this bit. If a default IRQ7 routine occurs during a normal IRQ7 routine, however, the ISR will remain set. In this case, it is necessary to keep track of whether or not the IRQ7 routine was previously entered. If another IRQ7 occurs, it is a default.

15.2.6 Interrupt Masks

15.2.6.1 Masking on an Individual Interrupt Request Basis

Each interrupt request input can be masked individually by the Interrupt Mask Register (IMR). This register is programmed through OCW1. Each bit in the IMR masks one interrupt channel, if it is set to a 1. Bit 0 masks IRQ0, Bit 1 masks IRQ1 and so forth. Masking an IRQ channel does not affect the other channel's operation, with one exception. Masking IRQ2 on CNTRL-1 will mask off all requests for service from CNTRL-2. The CNTRL-2 INTR output is physically connected to the CNTRL-1 IRQ2 input.

15.2.6.2 Special Mask Mode

Some applications may require an interrupt service routine to dynamically alter the system priority structure during its execution under software control. For example, the routine may wish to inhibit lower priority requests for a portion of its execution but enable some of them for another portion.

The difficulty is that if an Interrupt Request is acknowledged and an End of Interrupt command did not reset its IS bit (i.e. while executing a service routine), the Interrupt Controller would have inhibited all lower priority requests with no easy way for the routine to enable them.

The Special Mask Mode enables all interrupts not masked by a bit set in the Mask Register. Interrupt service routines that require dynamic alteration of interrupt priorities can take advantage of the Special Mask Mode. For example, a service routine can inhibit lower priority requests during a part of the interrupt service, then enable some of them during another part.

In the Special Mask Mode, when a mask bit is set to 1 in OCW1, it inhibits further interrupts at that level and enables interrupts from all other levels (lower as well as higher) that are not masked.

Thus, any interrupts may be selectively enabled by loading the Mask Register with the appropriate pattern.

Without Special Mask Mode, if an interrupt service routine acknowledges an interrupt without issuing an EOI to clear the IS bit, the interrupt controller inhibits all lower priority requests. The Special Mask Mode provides an easy way for the interrupt service routine to selectively enable only the interrupts needed by loading the Mask register.

The special Mask Mode is set by OCW3 where: SSMM=1, SMM=1, and cleared where SSMM=1, SMM=0.

15.2.7 Reading the Interrupt Controller Status

The input status of several internal registers can be read to update the user information on the system. The Interrupt Request Register (IRR) and In-Service Register (ISR) can be read via OCW3. The Interrupt Mask Register (IMR) is read via a read of OCW1. Here are brief descriptions of the ISR, the IRR, and the IMR.

Interrupt Request Register (IRR): 8-bit register which contains the status of each interrupt request line. Bits that are clear indicate interrupts that have not requested service. The Interrupt Controller clears the IRR's highest priority bit during an interrupt acknowledge cycle. (Not affected by IMR).

In-Service Register (ISR): 8-bit register indicating the priority levels currently receiving service. Bits that are set indicate interrupts that have been acknowledged and their interrupt service routine started. Bits that are cleared indicate interrupt requests that have not been acknowledged, or interrupt request lines that have not been asserted. Only the highest priority interrupt service routine executes at any time. The lower priority interrupt services are suspended while higher priority interrupts are serviced. The ISR is updated when an End of Interrupt Command is issued.

Interrupt Mask Register (IMR): 8-bit register indicating which interrupt request lines are masked.

The IRR can be read when, prior to the I/O read cycle, a Read Register Command is issued with OCW3 (RR=1, RIS=0).

The ISR can be read when, prior to the I/O read cycle, a Read Register Command is issued with OCW3 (RR=1, RIS=1).

The interrupt controller retains the ISR/IRR status read selection following each write to OCW3. Therefore, there is no need to write an OCW3 before every status read operation, as long as the current status read corresponds to the previously selected register. For example, if the ISR is selected for status read by an OCW3 write, the ISR can be read over and over again without writing to OCW3 again. However, to read the IRR, OCW3 will have to be reprogrammed for this status read prior to the OCW3 read to check the IRR. This is not true when poll mode is used. Polling Mode overrides status read when P=1, RR=1 in OCW3.

After initialization the Interrupt Controller is set to read the IRR.

As stated, OCW1 is used for reading the IMR. The output data bus will contain the IMR status whenever I/O read is active the address is 021h or 061h (OCW1).

15.2.8 Interrupt Steering

The IFB can be programmed to allow 4 PCI programmable interrupts (PIRQA#-PIRQD#) to be internally routed to one of 11 interrupts: 3 - 7, 9-12, 14 or 15. PCLK is used to synchronize the PIRQx# inputs. The PIRQx# lines are run through an internal multiplexer that assigns, or routes, an

individual PIRQx# line to any one of 11 IRQ inputs. The assignment is programmable through the PIRQx Route Control registers. One or more PIRQx# lines can be routed to the same IRQx input. If interrupt steering is not required, the Route Registers can be programmed to disable steering.

Bits 0-3 in each PIRQx Route Control register are used to route the associated PIRQx# line to an internal IRQ input. Bit 7 in each register is used to disable routing of the associated PIRQx#.

The PIRQx# lines are defined as active low, level sensitive to allow multiple interrupts on a PCI Board to share a single line across the connector. When a PIRQx# is routed to specified IRQ line, the software must change the IRQ's corresponding ELCR bit to level sensitive mode.

15.3 Serial Interrupts

The IFB supports a serial IRQ scheme. Because more than one device may need to share the single serial IRQ signal, an Open Collector signaling scheme is used. Timing is based on the PCI Clock. If the PCI clock is inactive when a device needs to signal an interrupt, the CLKRUN# signal must first be asserted by the device to restart the PCI clock.

The serial IRQ configuration is handled via the PCI configuration space. No other registers are associated with the scheme.

15.3.1 Protocol

Serial interrupt information is transferred using three types of frames: a Start Frame, one or more IRQ Data frames, and one Stop frame. There are also two modes of operation: Quiet Mode and Continuous Mode.

15.3.1.1 Quiet (Active) Mode

To indicate an interrupt, the peripheral brings the SERIRQ signal low for one clock, and then tri-states it. This brings all the state machines from IDLE to the ACTIVE states.

The IFB will then take control of the SERIRQ signal by driving it low on the next clock, and will continue driving it low for 3-7 clocks more (programmable). Thus the total number of clocks low will be 4-8. After those clocks, the IFB will drive SERIRQ high for one clock and then tri-state the signal.

15.3.1.2 Continuous (Idle) Mode

In this mode, the IFB initiates the START frame, rather than the peripherals. Typically this is done to update IRQ status (acknowledges). The IFB will drive SERIRQ low for 4, 6, 8 clocks depending on bits SERIRQC register (Function 0), bits [1:0].

This is the default mode after reset, and can be used to enter the Quiet mode.

15.3.1.3 Data Frame

Once the Start frame has been initiated, all of the serial interrupt peripherals must start counting frames based on the rising edge of SERIRQ. Each of the IRQ/DATA frames has exactly 3 phases of 1 clock each: a Sample phase, a Recovery Phase, and a Turn-around phase.

During the Sample phase, the device drives SERIRQ low if the corresponding interrupt signal is low. If the corresponding interrupt is high, then the devices will tri-state the SERIRQ signal. It will remain high due to pull-up resistors.

During the other two phases (turnaround and recovery), no device should drive the SERIRQ signal. The IRQ/DATA frames have a specific order and usage, as shown in Table 15-1.

If an SMI# is activated on frame 3, the IFB will drive its SMI# signal low. This will then generate an SMI# to the microprocessor if enabled.

Table 15-1. SERIRQ Frames

Data Frame Number	Usage	# Clocks Past Start
1	UNASSIGNED	2
2	IRQ1	5
3	SMI#	8
4	IRQ3	11
5	IRQ4	14
6	IRQ5	17
7	IRQ6	20
8	IRQ7	23
9	UNASSIGNED	26
10	IRQ9	29
11	IRQ10	32
12	IRQ11	35
13	IRQ12	38
14	UNASSIGNED	41
15	IRQ14	44
16	IRQ15	47
17	IOCHCK#	50
18	PCI INTA#	53
19	PCI INTB#	56
20	PCI INTC#	59
21	PCI INTD#	62
32:22	UNASSIGNED	96

15.3.1.4 Stop Frame

After all of the data frames, a Stop Frame will be done by the IFB. The IFB will drive SERIRQ low for 2-3 PCI clocks. The number of clocks determines the next mode:

- If SERIRQ is low for 2 clocks, then the next mode is the Quite Mode. Any device may initiate a Start Frame in the second clock (or more) after the rising edge of the Stop Frame.
- If SERIRQ is low for 3 clocks, then the next mode is the Continuous mode. Only the IFB may initiate a Start Frame in the second clock (or more) after the rising edge of the Stop Frame.

15.4 Timer/Counters

The IFB contains three counters that are equivalent to those found in the 82C54 programmable interval timer. The three counters are contained in one IFB timer unit, referred to as Timer-1. Each counter output provides a key system Function. Counter 0 is connected to interrupt controller IRQ0 and provides a system timer interrupt for a time-of-day, diskette time-out, or other system timing Functions. Counter 1 generates a refresh request signal and Counter 2 generates the tone for the speaker. The 14.31818 MHz counters normally use OSC as a clock source.

Counter 0, System Timer

This counter Functions as the system timer by controlling the state of IRQ0 and is typically programmed for Mode 3 operation. The counter produces a square wave with a period equal to the product of the counter period (838 ns) and the initial count value. The counter loads the initial count value one counter period after software writes the count value to the counter I/O address. The counter initially asserts IRQ0 and decrements the count value by two each counter period. The counter negates IRQ0 when the count value reaches 0. It then reloads the initial count value and again decrements the initial count value by two each counter period. The counter then asserts IRQ0 when the count value reaches 0, reloads the initial count value, and repeats the cycle, alternately asserting and negating IRQ0.

Counter 1, Refresh Request Signal

This counter provides the refresh request signal and is typically programmed for Mode 2 operation. The counter negates refresh request for one counter period (838 ns) during each count cycle. The initial count value is loaded one counter period after being written to the counter I/O address. The counter initially asserts refresh request, and negates it for 1 counter period when the count value reaches 1. The counter then asserts refresh request and continues counting from the initial count value.

Counter 2, Speaker Tone

This counter provides the speaker tone and is typically programmed for Mode 3 operation. The counter provides a speaker frequency equal to the counter clock frequency (1.193 MHz) divided by the initial count value. The speaker must be enabled by a write to port 061h (see NMI Status and Control ports).

15.4.1 Programming the Interval Timer

The counter/timers are programmed by I/O accesses and are addressed as though they are contained in one 82C54 interval timer. A single Control Word Register controls the operation of all three counters.

The interval timer is an I/O-mapped device. Several commands are available:

The Control Word Command specifies:

- Which counter to read or write.
- The operating mode.
- The count format (binary or BCD).

The Counter Latch Command latches the current count so that it can be read by the system. The countdown process continues.

The Read Back Command reads the count value, programmed mode, the current state of the OUT pins, and the state of the Null Count Flag of the selected counter.

The Read/Write Logic selects the Control Word Register during an I/O write when address lines A[1:0]=11. This condition occurs during an I/O write to port address 043h, the address for the Control Word Register on Timer 1. If the CPU writes to port 043h, the data is stored in the Control Word Register and is interpreted as the Control Word used to define the operation of the Counters.

The Control Word Register is write-only. Counter status information is available with the read back Command.

Because the timer counters wake up in an unknown state after power up, multiple refresh requests may be queued. To avoid possible multiple refresh cycles after power up, program the timer counter immediately after power up.

15.4.1.1 Write Operations

Programming the interval timer is a simple process:

1. Write a control word.
2. Write an initial count for each counter.
3. Load the least and/or most significant bytes (as required by Control Word bits 5, 4) of the 16-bit counter.

The programming procedure for the IFB timer is very flexible. Only two conventions need to be observed. First, for each counter, the control word must be written before the initial count is written. Second, the initial count must follow the count format specified in the control word (least significant byte only, most significant byte only, or least significant byte and then most significant byte).

Since the Control Word Register and the three counters have separate addresses (selected by the A1, A0 inputs), and each control word specifies the counter it applies to (SC0, SC1 bits), no special instruction sequence is required. Any programming sequence that follows the conventions above is acceptable.

A new initial count may be written to a counter at any time without affecting the counter's programmed mode. Counting will be affected as described in the mode definitions. The new count must follow the programmed count format.

If a counter is programmed to read/write two-byte counts, the following precaution applies: A program must not transfer control between writing the first and second byte to another routine which also writes into that same counter. Otherwise, the counter will be loaded with an incorrect count.

15.4.1.2 Interval Timer Control Word Format

The control word specifies the counter, the operating mode, the order and size of the count value, and whether it counts down in a 16-bit or binary-coded decimal (BCD) format. After writing the control word, a new count may be written at any time. The new value will take effect according to the programmed mode.

If a counter is programmed to read/write two-byte counts, the following precaution applies: A program must not transfer control between writing the first and second byte to another routine which also writes into that same counter. Otherwise, the counter will be loaded with an incorrect count. The count must always be completely loaded with both bytes.

15.4.1.3 Read Operations

It is often desirable to read the value of a counter without disturbing the count in progress. This is easily done in the IFB timer unit. There are three possible methods for reading the counters: a simple read operation, the Counter Latch Command, and the Read-Back Command.

15.4.1.4 Counter I/O Port Read

The first method is to perform a simple read operation. To read the counter, which is selected with the A1, A0 inputs (port 040h, 041h, or 042h), the CLK input of the selected counter must be inhibited by using either the GATE input or external logic. Otherwise, the count may be in the process of changing when it is read, giving an undefined result. When reading the count value directly, follow the format programmed in the control register: read LSB, read MSB, or read LSB then MSB. Within the IFB timer unit, the GATE input on Counter 0 and Counter 1 is tied high. Therefore, the direct register read should not be used on these two counters. The GATE input of Counter 2 is controlled through I/O port 061h. If the GATE is disabled through this register, direct I/O reads of port 042h will return the current count value.

15.4.1.5 Counter Latch Command

The Counter Latch Command latches the count at the time the command is received. This command is used to ensure that the count read from the counter is accurate (particularly when reading a two-byte count). The count value is then read from each counter's Count Register as was programmed by the Control Register.

The selected counter's output latch (OL) latches the count at the time the Counter Latch Command is received. This count is held in the latch until it is read by the CPU (or until the Counter is reprogrammed). The count is then unlatched automatically and the OL returns to "following" the counting element (CE). This allows reading the contents of the counters "on the fly" without affecting counting in progress. Multiple Counter Latch Commands may be used to latch more than one counter. Each latched counter's OL holds its count until it is read. Counter Latch Commands do not affect the programmed mode of the counter in any way. The Counter Latch Command can be used for each counter in the IFB timer unit.

If a Counter is latched and then, some time later, latched again before the count is read, the second Counter Latch Command is ignored. The count read will be the count at the time the first Counter Latch Command was issued.

With either method, the count must be read according to the programmed format; specifically, if the counter is programmed for two byte counts, two bytes must be read. The two bytes do not have to be read one right after the other. Read, write, or programming operations for other counters may be inserted between them.

Another feature of the IFB timer is that reads and writes of the same counter may be interleaved. For example, if the Counter is programmed for two byte counts, the following sequence is valid:

1. Read least significant byte.
2. Write new least significant byte.
3. Read most significant byte.
4. Write new most significant byte.

If a counter is programmed to read/write two-byte counts, a program must not transfer control between reading the first and second byte to another routine which also reads from that same counter. Otherwise, an incorrect count will be read.

15.4.1.6 Read Back Command

The third method uses the Read Back Command. The Read Back Command is used to determine the count value, programmed mode, and current states of the OUT pin and Null Count flag of the selected counter or counters. The Read Back Command is written to the Control Word Register, which causes the current states of the above mentioned variables to be latched. The value of the counter and its status may then be read by I/O access to the counter address.

The Read Back Command may be used to latch multiple counter output latches (OL) by setting the COUNT# bit D5=0 and selecting the desired counter(s). This single command is Functionally equivalent to several counter latch commands, one for each counter latched. Each counter's latched count is held until it is read (or the counter is reprogrammed). Once read, a counter is automatically unlatched. The other counters remain latched until they are read. If multiple count Read-Back Commands are issued to the same counter without reading the count, all but the first are ignored (i.e. the count which will be read is the count at the time the first Read-Back Command was issued).

The Read Back Command may also be used to latch status information of selected counter(s) by setting STATUS# bit D4=0. Status must be latched to be read. The status of a counter is accessed by a read from that counter's I/O port address.

If multiple counter status latch operations are performed without reading the status, all but the first are ignored. The status returned from the read is the counter status at the time the first status Read Back Command was issued.

Both count and status of the selected counter(s) may be latched simultaneously by setting both the COUNT# and STATUS# bits [5:4]=00. This is functionally the same as issuing two consecutive, separate Read Back Commands. The above discussions apply here also. Specifically, if multiple count and/or status Read Back Commands are issued to the same counter(s) without any intervening reads, all but the first are ignored.

If both count and status of a counter are latched, the first read operation from that counter will return the latched status, regardless of which was latched first. The next one or two reads (depending on whether the counter is programmed for one or two type counts) return the latched count. Subsequent reads return unlatched count.

15.5 Real Time Clock

The Real Time Clock (RTC) module provides a battery backed-up date and time keeping device with two banks of static RAM with 128 bytes each, although the first bank has 114 bytes for general purpose usage. Three interrupt features are available: time of day alarm with once a second to once a month range, periodic rates of 122 µs to 500 ms, and end of update cycle notification. Seconds, minutes, hours, days, day of week, month, and year are counted. Daylight savings compensation is optional. The hour is represented in twelve or twenty-four hour format, and data can be represented in BCD or binary format. The time keeping comes from a 32.768 kHz oscillating source, which is divided to achieve an update every second. The lower 14 bytes on the lower RAM block have very specific Functions. The first ten are for time and date information. The next four (0Ah to 0Dh) are registers, which configure and report RTC Functions.

The time and calendar data should match the data mode (BCD or binary) and hour mode (12 or 24 hour) as selected in register B. It is up to the programmer to make sure that data stored in these locations is within the reasonable values ranges and represents a possible date and time. The exception to these ranges is to store a value of C0 - FF in the Alarm bytes to indicate a don't care situation. All Alarm conditions must match to trigger an Alarm Flag, which could trigger an Alarm Interrupt if enabled. The UIP bit should be read as 0 before each access to these registers. The SET bit of register B should be one while programming these locations to avoid clashes with an update cycle. Access to time and date information is done through the RAM locations. If a RAM read from the ten time and date bytes is attempted during an update cycle, the value read will not necessarily represent the true contents of those locations. Any RAM writes under the same conditions will be ignored.

15.5.1 RTC Registers and RAM

The RTC internal registers and RAM are organized as two banks of 128 bytes each, called the standard and extended banks. The first 14 bytes of the standard bank contain the RTC time and date information along with four registers, A - D, that are used for configuration of the RTC. The extended bank contains a full 128 bytes of battery backed SRAM, and will be accessible even when the RTC module is disabled (via the RTC configuration register).

All data movement between the host CPU and the real-time clock is done through registers mapped to the I/O space at locations 70-73h.

I/O locations 70h and 71h are the standard RAM location for the real-time clock. I/O locations 72h and 73h are the extended RAM, and may be disabled. When disabled, 72h and 73h become aliases for 70h and 71h respectively. The addressing is done by a indexing scheme: 70h (72h) is written with the index, and 71h (73h) is written with data or read for data. This scheme is shown in Table 15-2.

Table 15-2. RTC (Standard) RAM Bank

Index Address	Name
00h	Seconds
01h	Seconds Alarm
02h	Minutes
03h	Minutes Alarm
04h	Hours
05h	Hours Alarm
06h	Day of Week
07h	Date of Month
08h	Month
09h	Year
0Ah	Register A
0Bh	Register B
0Ch	Register C
0Dh	Register D
0Eh - 7Fh	114 Bytes of User RAM

The extended RAM bank is also accessed using an indexed scheme. I/O address 72h is used as the address pointer and I/O address 73h is used as the data register. Index addresses above 127h are not valid.

15.5.1.1 Register A

Address Offset: 0Ah
Default Value: NA - This register is not affected by any system reset signal.
Attribute: Read/Write

This register is used for general configuration of the RTC Functions.

Bit(s)	Description
7	**UPDATE IN PROGRESS (UIP):** This bit may be monitored as a status flag. When asserted as a 1, the update is soon to occur or is in progress. If 0, the update cycle will not start for at least 244 µs. The time, calendar, and alarm information in RAM is always available when the UIP bit is 0.
6:4	**Division Chain Select (DVx):** These three bits control the divider chain for the oscillator. DV2 DV1 DV0 Function 0 1 0 Normal Operation 1 1 X Divider Reset 1 0 1 Bypass 15 stages (test mode only) 1 0 0 Bypass 10 stages (test mode only) 0 1 1 Bypass 5 stages (test mode only) 0 1 1 Invalid 0 0 0 Invalid
3:0	**Rate Select Bits (RSx):** Selects one of 13 taps of the 15 stage divider chain. The selected tap can generate a periodic interrupt if the PIE bit is set in register B. Otherwise this tap will set the PF flag of register C. If the periodic interrupt is not to be used, these bits should all be set to zero.

RS3	RS2	RS1	RS0	Periodic Rate
0	0	0	0	Interrupt never toggles
0	0	0	1	3.90625 ms
0	0	1	0	7.8125 ms
0	0	1	1	122.070 µs
0	1	0	0	244.141 µs
0	1	0	1	488.281 µs
0	1	1	0	976.5625µs
0	1	1	1	1.953125 ms
1	0	0	0	3.90625 ms
1	0	0	1	7.8125 ms
1	0	1	0	15.625 ms
1	0	1	1	31.25 ms
1	1	0	0	62.5 ms
1	1	0	1	125 ms
1	1	1	0	250 ms
1	1	1	1	500 ms

15.5.1.2 Register B

Address Offset: 0Bh
Default Value: X0000XXXb
Attribute: Read/Write

This register is used for general configuration of the RTC Functions.

Bit(s)	Description
7	**SET:** Enables the update cycles. When is zero, update cycle occurs normally once a second. If set to one, a current update cycle will abort and subsequent update cycles will not occur until SET is returned to zero. When set is one, the firmware may initialize time and calendar bytes safely. This bit is not affected by RSMRST#.
6	**Periodic Interrupt Enable (PIE):** If set to one, the Periodic Interrupt Enable (PIE) bit allows an interrupt to occur with a time base set with the RS bits of register A. This bit is cleared (set to zero) on active RSMRST#.
5	**Alarm Interrupt Enable (AIE):** If set to one, the Alarm Interrupt Enable (AIE) bit allows an interrupt to occur when the AF is one as set from an alarm match from the update cycle. An alarm can occur once a second, one an hour, once a day, or one a month. This bit is cleared on active RSMRST#.
4	**Update-ended Interrupt Enable (UIE):** If set to one, the Update-ended Interrupt Enable (UIE) bit allows an interrupt to occur when the update cycle ends. This bit is cleared on active RSMRST#.
3	**Square Wave Enable (SQWE):** The Square Wave Enable bit serves no Function in this device, yet is left in this register bank to provide compatibility with the Motorola 146818B. There is not SQW pin on this device. This bit is cleared on active RSMRST#.
2	**Data Mode (DM):** The Data Mode (DM) bit specifies either binary or BCD data representation. A one denotes binary, and zero denotes BCD. This bit is not affected by RSMRST#.
1	**Hour Format (HF):** This bit indicates the hour byte format. If one, twenty-four hour mode is selected. If zero, twelve-hour mode is selected. In twelve hour mode, the seventh bit represents AM as zero and PM as one. This bit is not affected by RSMRST#.
0	**Daylight Savings Enable (DSE):** The Daylight Savings Enable bit triggers two special hour updates per year when set to one. One is on the first Sunday in April, where time increments from 1:59:59 AM to 3:00:00 AM. The other is the last Sunday in October when the time first reaches 1:59:59 AM, it is changed to 1:00:00 AM. The time must increment normally for at least two update cycles (seconds) previous to these conditions for the time change to occur properly. These special update conditions do not occur when the DSE bit is set to zero. The days for the hour adjustment are those specified in United States federal law as of 1987, which is different than previous years. This bit is not affected by RSMRST#.

15.5.1.3 Register C

Address Offset: 0Ch
Default Value: 00h
Attribute: Read/Write

This register is used for various flags. All flag bits are cleared upon active RSMRST# or a read of Register C.

Bit(s)	Description
7	**Interrupt Request Flag (IRQF):** Interrupt Request Flag = PF * PIE + AF * AIE + UF *UFE. This also causes the CH_IRQ_B signal to be asserted.
6	**Periodic Interrupt Flag (PF):** Periodic interrupt Flag will be one when the tap as specified by the RS bits of register A is one. If no taps are specified, this flag bit will remain at zero.
5	**Alarm Flag (AF):** Alarm Flag will be high after all Alarm values match the current time.
4	**Update-ended Flag (UF):** Updated-ended flag will be high immediately following an update cycle for each second.
3:0	**Reserved.** Read as 0.

15.5.1.4 Register D

Address Offset: 0Dh
Default Value: NA - This register is not affected by any system reset signal.
Attribute: Read/Write

This register is used for various flags.

Bit(s)	Description
7	**Valid RAM and TIME Bit (VRT):** The Valid Ram and Time bit is set to one when the PWRGD (power good) signal provided is high. This feature is not typically used. This bit should always be set to 0 for write to this register.
6	Reserved. This bit always returns a 0 and should be set to 0 for write cycles.
5:0	**Date Alarm (DA):** These bits store the date of month alarm value. If set to 000000, then a don't care state is assumed. The host must configure the date alarm for these bits to do anything, yet they can be written at any time. If the date alarm is not enabled, these bits will return zeros to mimic the Functionality of the Motorola 146818B. These bits are not affected by RSMRST#.

15.5.2 RTC Update Cycle

An update cycle occurs once a second, if the SET bit of register B is not asserted and the divide chain is properly configured. During this procedure, the stored time and date will be incremented, overflow will be checked, a matching alarm condition will be checked, and the time and date will be rewritten to the RAM locations. The update cycle will start at least 244µs after the UIP bit of register A is asserted, and the entire cycle will not take more than 1984µs to complete. The time and date RAM locations (0-9) will be disconnected from the external bus during this time. To avoid update and data conditions, external RAM access to these locations can safely occur at two times. When a updated-ended interrupt is detected, almost 999ms are available to read and write valid time and date data. If the UIP bit of register A is detected to be low, there is at least 244µs before the update cycle begins. Because the overflow conditions for leap years and daylight savings adjustments are based on more than one date or time item, the time before one of these conditions should be set (when adjusting) at least two seconds before one of these conditions to ensure proper operation.

15.5.3 RTC Interrupts

The real-time clock interrupt is internally routed within the IFB both to the I/O APIC and the 8259. It is mapped to interrupt vector 8. This interrupt is not shared with any other interrupt. IRQ8# from the serial stream is ignored.

15.5.4 Lockable RAM Ranges

The real-time clock battery-backed RAM supports two 8-byte ranges that can be enabled via the configuration space. If the configuration bits are set, the corresponding range in the RAM will not be readable or writeable. A write cycle to those locations will have no effect. A read cycle to those locations will not return the actual location value.

Once enabled, this Function can only be disabled by a hard reset.

IFB Power Management 16

16.1 Overview

IFB is designed for desktop systems, and includes the following power management features for the desktop design:

1. Compliance with industry standard specifications:
   APM Rev 1.2
   ACPI Rev 1.0
   Energy Star (30W idle)
   PCI Power Management Rev 1.0
2. ACPI S1 Sleep State with STPCLK# and/or SLP# active.
3. ACPI S4/S5 Sleep States (Suspend-to-Disk and Soft-Off).
4. ACPI Power management timer.
5. APCI Compliant Power Button and Thermal Input signals.
6. Ability for firmware to cause SCI and ACPI software to cause SMI#: ACPI Requirement.
7. SMI# and/or SCI generation from various sources, including power management timer.
8. Alt Access Mode: Needed to allow for Suspend-to-Disk.

Table 16-1 shows the power states defined for platforms using the IFB.

Table 16-1. IFB Power States and Consumption

ACPI State/ Substate	Description
S0/C0	**ON:** CPU operating a full speed with no latencies.
S0/C1	**Auto Halt:** CPU has executed a Halt instruction. Returns to the S0/C0 state based on a break event.
S1	**Stop-Grant or Sleep:** IFB supports two versions of the S1 Sleep state. In the first the STPCLK# signal is active, much like the S0 state. In the 2nd, both STPCLK# and SLP# are active. This puts the CPU in an even lower power state, but it cannot maintain cache coherency. Entrance to the S1 state is performed by a write to the SLP_EN bit with the appropriate SLP_TYP field. Will return to S0 state based on Wake event.
S4	**Suspend to Disk (STD):** The context of the system is maintained on the disk. All power is then shut to the system except for the logic required to resume. Entrance to the S4 state is performed by a write to the SLP_EN bit with the appropriate SLP_TYP field. Will return to S0 state based on Wake event.
S5	**Soft Off (SOFF):** No system context saved.These two states are similar. For STD (ACPI S4 state), SOFF (ACPI S5 or G2 states) is the same, except the context is not saved. Entrance to the S5 state is performed either by a write to the SLP_EN bit with the appropriate SLP_TYP field, or by a power button override. Will return to S0 state based on Wake event.
G3	**Power Failure (PFAIL):** Power is lost to the system (typically because it has been unplugged). IFB still maintains the RTC and CMOS RAM with the external backup battery. Return after power resumption depends on AFTERG3 bit and state prior to the power failure.

16.2 IFB Power Planes

16.2.1 Power Plane Descriptions

The IFB contains three power planes:

RTC Plane	This plane includes the RTC, as well as some of the power management logic. It is intended to be backed up by a battery, even when all other power to the system is shut.
Resume Plane	This plane contains additional power management logic, as well as other circuits that can wake the systems from the S4-S5 states. The resume plane will typically be powered by the main power supply's trickle output.
Main (Core) Plane	This includes all other signals.

16.2.2 SMI# Generation

Table 16-2 shows which sources can cause the IFB to drive SMI# active. When operating with an ACPI-based Operating System, some of the causes of SMI# will instead be routed to cause an SCI.

Upon any SMI# event taking place, the IFB will assert SMI#. SMI# remains active until the EOS bit is set. When the EOS bit is set, SMI# will go inactive for a minimum of 1 PCICLK. If another SMI event occurs, SMI# will be driven active again.

Table 16-2. Causes of SMI#

SMI Event	Comment
ACPI SMI# bit (GBL_RLS)	ACPI sets bit to cause SMI#, SMM handler clears the bit. ACPI I/O offset 04h, Bit 2
GPIO Assertion	When a GPIO is programmed as an input and is set to a '1', an SMI# will be generated. The bit is cleared when the SMM handler clears the asserting device.
Overflow of ACPI Timer	Time-out every 2.34 seconds. If SCI_EN is set, the timer overflow will instead cause an SCI.
THRM# signal	The THRM# can cause an SMI# on either the rising or falling edge. If the SCI_EN is set, the THRM# signal will instead cause an SCI.
Master Aborts of the IFB DDMA Logic, USB Controller, or IDE Controller	Internal bus master state machine sets bit. SMM handler will typically clear the bit, then retry the cycle.
Legacy USB Support	Bit set based on address decode or incoming USB IRQ. SMM handler will clear bits.
1MIN Timer	Needed for legacy power management, this time will generate an SMI# every minute. The SMM handler can check the Wake/Break status register to see if there is any system activity. After n minutes of no system activity (where n is determine by the SMM handler), the SMM handler can decide to put the system into a lower power state.
SW SMI# Timer	Not to be confused with the above periodic SMI timer.

16.2.3 SCI Generation

In an ACPI environment, an SCI (system control interrupt) must be generated for any event that must be handled by ACPI software. If the SCI_EN bit is set, the IFB will generate an SCI based on the sources listed below in Table 16-3. Each source can be individually enabled/disabled.

Table 16-3. Causes of SCI#

SCI Event	Comment
Overflow of ACPI Timer	Time-out every 2.34 seconds. If SCI_EN is not set, the timer overflow will instead cause an SMI#.
THRM# Signal	The THRM# can cause an SCI# on either the rising or falling edge. If the SCI_EN is not set, the THRM# signal will instead cause an SMI#.
Setting of the BIOS_RLS Bit	This bit is set by the firmware to cause an SCI. The ACPI handler will clear the bit.
GPIO Assertion	When a GPIO bit is programmed as an input, and the register bit is set to a '1'. The bit will be cleared when the ACPI handler clears the asserting device.

SCI is a level mode interrupt. In non-APIC systems (default), the SCI IRQ is routed to IRQ9. The 8259 interrupt controller must be programmed to level mode for that interrupt. In APIC systems, the SCI IRQ can still be IRQ9, or can be routed to one of the APIC interrupts 20-23. In either case, the interrupt generated internally is active high level. The interrupt will remain high until all SCI sources are removed.

16.2.4 Sleep States

The IFB directly supports several sleep states (two of which will typically be mapped to the ACPI S1 state). From a IFB perspective, the two S1 states only differ on whether the SLP# signal is active. Additional Sleep states, such as S4 and S5 are also supported.

The entry to the Sleep states are based on several assumptions:

- After setting the SLP_EN bit, the IFB will assert the STPCLK# pin. The IFB will not continue to step through the sleep sequence unless the Stop Grant special cycle is received on the PCI bus.
- Entering the Sleep state without at least one wake event set is not recommended. This could lock up the system. However, the power button will always be a wake event.
- If using the S1 Sleep state with SLP# active, the PCI masters must be prevented from accessing memory, because the CPU cannot maintain cache coherence.
- In either of the IFB S1 Sleep states, the PCI clock will still be running, so the Serial IRQ stream will still be available. DMA, USB, and IDE will not be available, because the IFB bus masters will not be able to access main memory. The USB controllers, can still generate WAK event, however.
- In the S4 or S5 Sleep states, the PCI clock will NOT be running (and the PCI bus unpowered), so the Serial IRQ stream will NOT be available. If an external device is still powered during S4-S5, it should use some other mechanism to request that the IFB wakes the system.
- Upon exit from any Sleep states, the WAK_STS bit will be set.
- Upon exit from any Sleep state, the SLP_TYP bits will contain the originally programmed values.

16.2.5 ACPI Bits Not Implemented by IFB

Many ACPI registers and bits are optional, and do not have to be implemented for a standard desktop design. Table 16-4 shows which bits are not implemented by IFB.

Table 16-4. ACPI Bits Not Implemented in IFB

Offset	Register Name/Function		Comment
00-01h	PM1 Status		
	4	BM_STS	Not needed for standard desktop.
04-05h	PM1 Control		
	1	BM_RLD	Stopping CPU clock not supported.
0C-0Dh	General Purpose Status		
	9	GPI_STS	GPI not needed for desktop
	11	LID_STS	Lid not needed for desktop
OE-0Fh	General Purpose Enables		
	9	GPI_EN	GPI not needed for desktop
	11	LID_EN	Lid not needed for desktop
15h	Level 3 Register		Power state not needed for desktop

16.2.6 Entry/Exit for the S4 and S5 States

As part of the ACPI spec, as well as PC'97 specs, all new desktop systems must support the SOFT OFF (ACPI S5) state. The state will have the following characteristics:

- No system context preserved.
- Power shut to all subsystems except RTC and wake logic (typically just the power button).
- All system clocks shut except 32.768 kHz internal to the RTC logic.

There are two ways to enter the Soft-Off state:

1. The CPU will write a value of 100 to the SLP_TYP field and set the SLP_EN bit to 1.
2. The power button is pressed for 4 seconds. This is known as a power-button override event.

In either case, the entry to the Soft-Off state is done by asserting the SUSB and SUSC signals. This will cause the power to be shut and the PWROK signal is assumed to go low.

Note also that there is no need to enter the Soft-Off state gracefully. The STPCLK# and SUS_STAT# signals don't have to be asserted in any particular order, since the CPU and memory controller will be reset after the system is rebooted.

A Wake event will cause an exit from the Soft-Off state. The wake events that can wake from the S5 state are:

S5 Wake Event	Comment
RTC Alarm cause RTC_STS bit set	RTC_EN must be set for the wake event
Power button press causes PWRBTN_STS bit set	Power Button is unconditional wake event
Ring Indicate causes RI_STS bit set	RI_EN must be set for the wake event
GPIO Routed to SCI goes active	EXTSCI_EN must be set for the wake event
GPIO Routed to SMI goes active	EXTSMI_EN must be set for the wake event

16.3 Handling of Power Failures in IFB

A power failure is defined as any one of the following:

- PWROK goes low and the IFB did not yet cause SUSB and/or SUSC to go inactive.
- Power to the Resume plane (as detected by RSMRST# or VCCRESUME) goes inactive.

Upon detection of a power failure, the state machine will go to the G3 state and the IFB will set a new status bit called PWR_FAIL. That bit can be cleared only by writing a 1 to that bit position.

Add new config bit, AFTERG3, powered off RTC well to indicate what should be done after power failure:

- 1 = Return to S4/S5 state (see table below), 0 = Cold Boot.
- Software should set this bit prior to going to an S4 or S5 state if it desires the system to return there after the power failure. This bit is automatically set to 1 due to a Power Button Override event.

When RSMRST# (or the Resume Vcc) is inactive after a power failure, the state machine will transition based on the following table:

State Prior to Power Failure	AFTERG3	Action after Power Returns
S0, S1	0	Cold boot to return to S0 state. SUSB and SUSC go inactive.
S0, S1	1	No boot. SUSB and SUSC stay active (S5 state) until wake event.
S4	0	Cold boot to return to S0 state. SUSB and SUSC go inactive.
S4	1	No boot. SUSB and SUSC stay active (S4 state) until wake event.
S5	0	Cold boot to return to S0 state. SUSB and SUSC go inactive.
S5	1	No boot. SUSB and SUSC stay active (S5 state) until wake event.

If USB devices are attached, setting the AFTERG3 bit to 1 may not be a good idea, since the USB devices will lose their configuration during the power failure and may no longer be able to generate a wake events when the power is restored. In this case, the system may have to always boot after a power failure (except if placed into the S5 state due to Power Button Override).